Reliable and Variation-tolerant Interconnection Network for Low Power MPSoCs

Mohammad Reza Kakoee

DEIS

Universita' di Bologna

A thesis submitted for the degree of

Philosophi æDoctor (PhD) in Electronic Engineering

2012 February

When we consider a new project,
we really study it... not just the surface idea,
but everything about it.

-Thomas, F and Johnston, O, 1981: The Illusion of Life.

Alma Mater Studiorum - Università degli Studi di Bologna

DOTTORATO DI RICERCA IN
Elettronica, Informatica e delle Telecomunicazioni

CICLO XXIV

Settore concorsuale di afferenza: 09/E3
Elettronica
Settore scientifico disciplinare: ING-INF/01
Elettronica

Reliable and Variation-tolerant Interconnection Network for Low Power MPSoCs

Presentata da: Mohammad Reza Kakoee

Coordinatore Dottorato:	**Relatore:**
Chiar. mo Prof. Ing. Luca Benini	Chiar. mo Prof. Ing. Luca Benini

---

Esame Finale Anno 2012

Abstract

The sustained demand for faster, more powerful chips has been met by the availability of chip manufacturing processes allowing for the integration of increasing numbers of computation units onto a single die. The resulting outcome, especially in the embedded domain, has often been called SYSTEM-ON-CHIP (SOC) or MULTI-PROCESSOR SYSTEM-ON-CHIP (MPSOC).

MPSoC design brings to the foreground a large number of challenges, one of the most prominent of which is the design of the chip interconnection. With a number of on-chip blocks presently ranging in the tens, and quickly approaching the hundreds, the novel issue of how to best provide on-chip communication resources is clearly felt.

Scaling down of process technologies has increased process and dynamic variations as well as transistor wearout. Because of this, delay variations increase and impact the performance of the MPSoCs. The interconnect architecture in MPSoCs becomes a single point of failure as it connects all other components of the system together. A faulty processing element may be shut down entirely, but the interconnect architecture must be able to tolerate partial failure and variations and operate with performance, power or latency overhead.

This dissertation focuses on techniques at different levels of abstraction to face with the reliability and variability issues in on-chip interconnection networks. By showing the test results of a GALS NoC testchip this dissertation motivates the need for techniques to detect and work around manufacturing faults and process variations in MPSoCs' interconnection infrastructure. As a physical design technique, we propose the bundle routing framework as an effective way to route the Network on Chips' global links. For architecture-level design, two cases are addressed: (i) Intra-cluster communication where we propose a low-latency interconnect with variability robustness (ii) Inter-cluster communication where an online functional testing with a reliable NoC configuration are proposed. We also propose dualVdd as an orthogonal way of compensating variability

at the post-fabrication stage. This is an alternative strategy with respect to the design techniques, since it enforces the compensation at post silicon stage.

Keywords

System-on-Chip (SoC)
Multi-Processor System-on-Chip (MPSoC)
Network-on-Chip (NoC)
Dynamic and static Variation
Physical Design
Test and Reliability in NoCs
Fault Tolerance in NoCs
Tightly coupled shared-L1 clusters
Ultra Low Latency On-Chip Interconnect Fabric
Variation Tolerant On-Chip Interconnect Fabric
Row-based Fine-grained Variability Compensation

Acknowledgments

First of all, I would like to thank my advisor over these years - Prof. Luca Benini - for his guidance and support. He has allowed me to get in contact with exciting research opportunities, with many brilliant people, and with world-class institutions. His technical and editorial advice was essential to the completion of this dissertation and has taught me innumerable lessons and insights on the workings of academic research in general. Thanks to him, I have also had the chance to travel to many places in the world, from North America to France.

My thanks also go to my co-advisors, Prof. Valeria Bertacco and Prof. Davide Bertozzi for helping me during this research and for reading previous drafts of this dissertation and providing many valuable comments that improved the presentation and contents of this dissertation. I would also like to thank the members of my major committee, ...

I would like to thank my wife Parisa for coping with my absences and, even worse, with my presences. I guess - in fact, I am pretty sure - it may have been a bumpy ride for her. Maybe it got a bit better when she began to realize that the chances of me coming back home in time for dinner were just... well, low.

My parents, Mohammad Hossein and Masoumeh, receive my deepest gratitude and love for their dedication and the many years of support during my undergraduate studies that provided the foundation for this work.

Of course this dissertation would never have been written without the contribution of many people. A special thanks to all the astounding people in Bologna that I have had the chance to meet. Probably, in not so many other jobs I would have had the same chances of making so many good friends on the workplace. To let the reader know who to blame, I just want to name a few. The friendship of Igor Loi is much appreciated and has led to many interesting and good-spirited discussions relating to this research. And then follow Andrea M., Andrea B., Francesco, Paolo, Giacomo, Federico, Antonio, Alessandro, who have all been great to work

with. Last, but not least, I would like to thank my friends in University of Michigan where I was a visiting scholar for 6 months. I would like to thank Alireza, Mehrzad, Armin, Andrea, Amirhossein, Mojtaba, Parisa, Elnaz, for their help from the day I arrived till the last day of my stay.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

1.1 SoC Road-map

Exponential increase in the circuit performance due to device scaling has nourished the recent electronics revolution for decades. Silicon technology with a minimum feature size of 32nm is already in volume production, and 28nm is rapidly ramping up. Billions of transistors can be integrated on a single chip; for instance 32nm CMOS has an impressive density of 1.5Mgate/mm2. Although integrating of an entire digital system onto a single silicon die is indeed possible, the non-recurring engineering cost for developing a new chip is in the tens of millions dollar range. Complex and flexible systems-on-chips (SoCs) are exploited by integrated device manufacturers (IDMs) to recover this huge non-recurring cost.

In addition, predictions that Moore's Law has reached it limits have been heard for years and have proven to be premature. We are now nearing the basic physical limits to CMOS scaling and the continuation of the price elastic growth of the industry cannot continue based on Moore's law scaling alone. This will require "More than Moore" through the tighter integration of system level components at the chip level. In the past scaling geometries enabled improved performance, less power, smaller size and lower cost. Today scaling alone does not ensure improvement of all four items. Figure 1.1 on the following page from ITRS 2010 shows this in more details.

System on Chip (SoC) technology provide a path for continued improvement in performance, power, cost and size at the system level without relying upon conventional CMOS scaling alone. A System-on-Chip

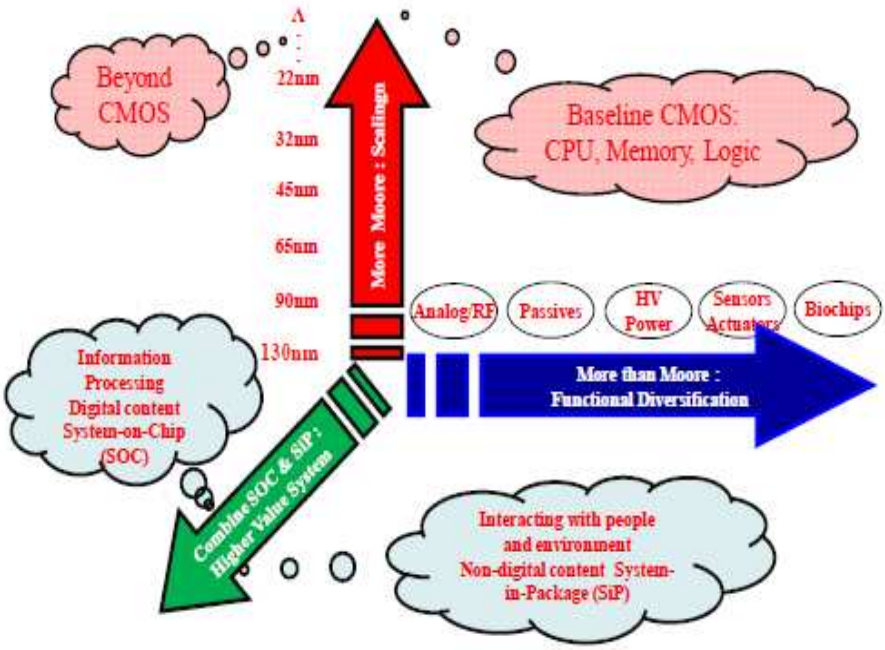


Figure 1.1: ITRS 2010, beyond CMOS scaling

(SoC) is a complex single-die device providing all the required electronic circuitry and hardware for a complete system in a specific application domain. An SoC may include one or more processor units, on-chip memory, I/Os, peripheral interfaces, hardware accelerators, data converters, and other modules which form a complete computing system.

Although, similarly to ASIC, an SoC is designed for a specific type of application, unlike traditional ASICs, which emphasized specialized-function hardware design, SOCs try to minimize the newly created part of the chip by reusing existing blocks or "cores" as much as possible. These reused blocks also called intellectual-property (IP) cores include a wide spectrum of the components such as analog and high-volume custom cores, as well as software technology blocks.

A key challenge is to design, develop, and maintain IP cores so that they can be reused for a range of applications. Normally, custom functions are hardly created as IPs because of economic reasons, as reducing design cost and risk is the principal goal. Moreover, reusable IP cores might need description properties such as "field of use" or "assumed design context" which are not usually specified when designing a stand-alone module.

Developing of an IC design component as an IP-core is considerably more challenging and difficult than creation for one-time use.

Moving to the scaled technology is a low-cost way to achieve a smaller area, lower power and higher performance. Therefore, SoCs take advantage of the aggressive technology evolution to improve the efficiency with a low design effort. From the design viewpoint, SoC design methods and goals are relatively conservative and have traditionally lower clock frequency and layout density compared to those of custom-designed processors (MPUs). However, the quality difference between ASIC/SOC and full-custom has continuously been reduced. From the 2001 ITRS roadmap edition, ASIC and MPU logic densities were modeled as being equal; and in addition "custom quality on an ASIC schedule" has been getting closer and closer thanks to the improved EDA techniques and tuning-based standard-cell methodologies. Recently, MPUs have entered into SOC domain following two directions of evolution. First, MPUs are being designed as IP-cores to be fitted in SoCs. Second, MPUs are themselves designed as multi-core SoCs to improve reuse and design productivity and achieve a desired power and performance. Moreover, SoCs for a number of high-end market domains, such as networking and video-gaming, are characterized by increasingly demanding performance specifications with required performance metrics (e.g., per-die floating point operations per second, or per-die external I/O bandwidth) beyond those of conventional general-purpose processors. In light of these needs, SoC designs are now becoming the driver for the growth of key parameters, such as number of cores per die, maximum frequency per core, and per-pin I/O bandwidth.

Early SoCs in the nineties contained typically one processor serving as central processing unit (CPU), one special-function processor, a number of fixed-function sub-systems targeted to specific applications, internal memory and plenty of IO interfaces. Figure 1.2 on the next page shows an instance of such a late nineties' SoC. This figure depicts the main architecture of Infineon's E-GOLD+ SoC for second-generation digital telephony, implemented in 0,25um technology.

The architecture is a heterogeneous dual-core, featuring a general purpose CPU (the C166CBC processor) for running the phone OS, the GUI and for overall system coordination, and a DSP for digital baseband decoding. A significant amount of on-chip data memory (both volatile and non-volatile) and program memory is integrated on the chip. Even though the corresponding silicon area is not highlighted in the chip microphotograph, several dedicated hardware blocks are also integrated, for modulation, digital filtering and equalization. In addition a number of standard IOs are supported (IrDA, keypad, etc.). The chip also integrates

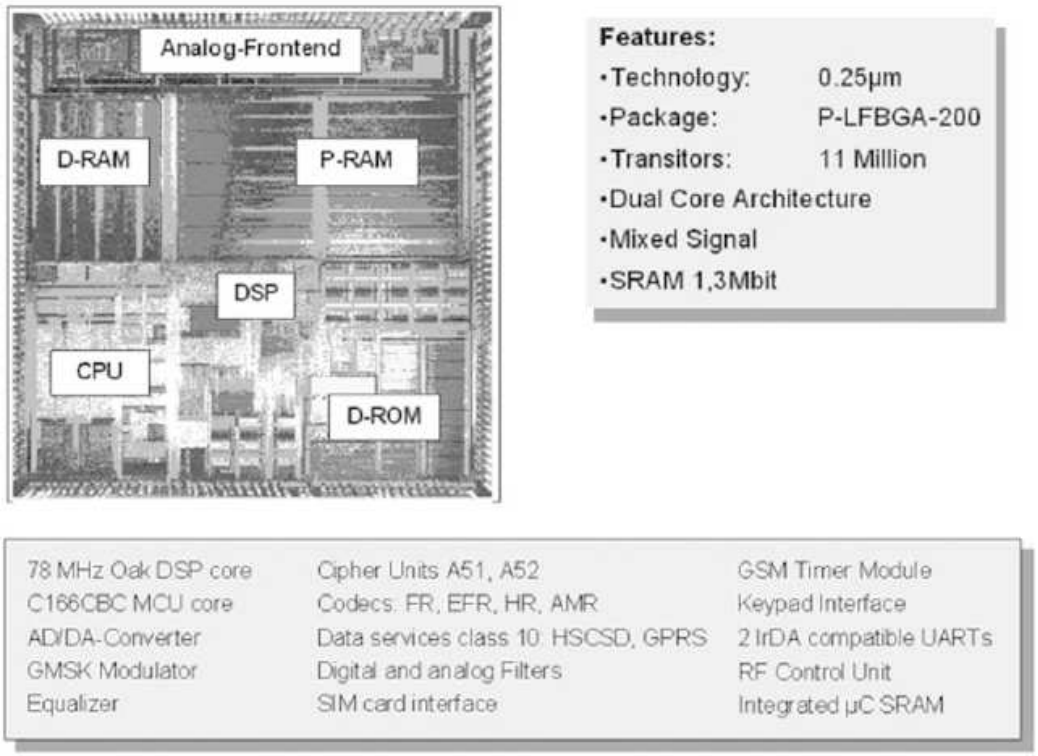


Figure 1.2: Infineon GSM transceiver E-GOLD+ (1999)

the analog front-end. From the architectural viewpoint, this SoC is only moderately parallel in terms of IP-cores, and highly heterogeneous. The trends toward multiple programmable cores, architectural heterogeneity and complex on-chip memory hierarchy are already evident in this early SoC. Note that the E-GOLD+ design pre-dates by almost one decade multicore processors for general-purpose computing.

The Lucent Daytona chip , introduced in year 2000, was one of the first examples of market-ready scalable Multiprocessor System-on-Chip (MP-SoC) architectures, which allow the integration of multiple arrays of homogeneous processors. The architecture of this SoC is shown in Figure 1.3 on the facing page. Daytona was designed for wireless base stations where identical signal processing is performed on multiple data channels. This is a perfect use case for a parallel architecture, as the target application is embarrassing parallel. Therefore, Daytona was developed as a symmetric architecture with four processors attached to a high-speed bus. The CPU architecture is based on an enhanced SPARC V8: 16x32 multiplication, division step, touch instruction, and vector co-processor. Each CPU has an

8-KB 16-bank cache. Each bank can be configured as instruction cache, data cache, or scratch pad. The processors are connected with the memory system and peripherals through an advanced on-chip bus capable of multiple outstanding transactions and out-of-order completion. The processors share a common address space in memory; the caches snoop to ensure cache coherency. The chip was 200 mm^2 and ran at 100 MHz at 3.3V in a $0.25\mu m$ CMOS process.

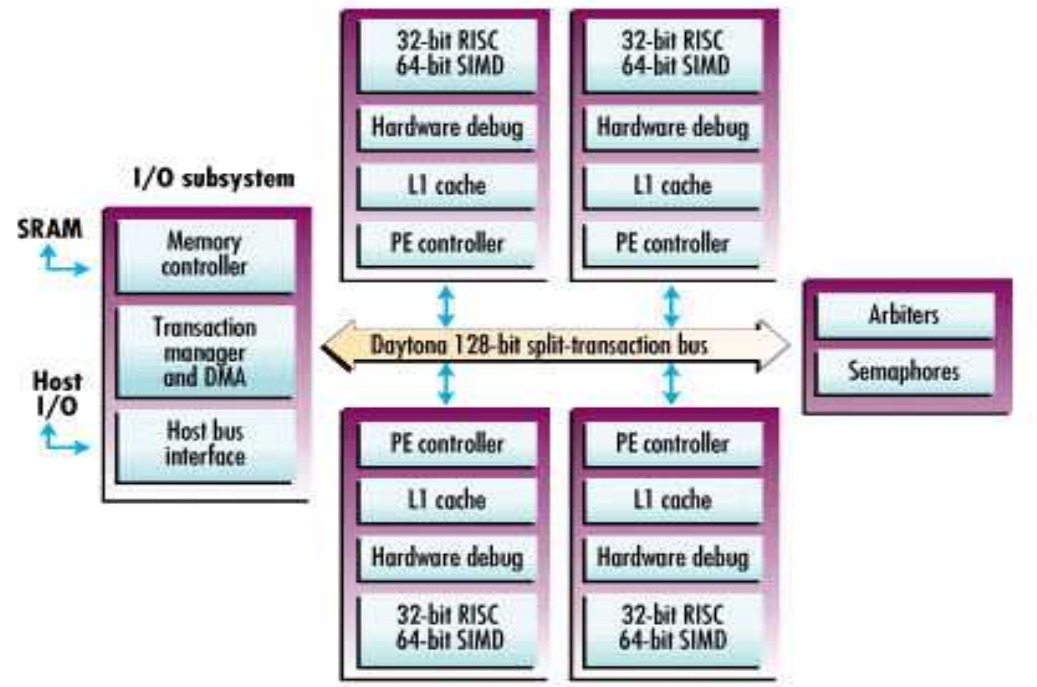


Figure 1.3: The Lucent Daytona Chip (2000)

Since 2000, several other scalable MPSoCs with a wide range of architectural variations were announced which support a spectrum of applications including video and multimedia, gaming, mobile telephony, and etc.

The main target of SoCs is high-volume consumer electronics applications. Because of short product life cycles and rapidly growing demands for functionality and performance in consumer products, the primary requirements for consumer SoCs are to achieve high performance and functionality with a short (six to nine months) time-to-market. Consumer SoCs can be classified, following the ITRS roadmap , into two categories: Consumer Portable and Consumer Stationary, with typical applications being mobile telephony and high-end gaming and video, respectively. The two

different categories are distinguished mainly by power consumption requirement: the Consumer Portable SoC must minimize power consumption to maintain the product battery life, while the Consumer Stationary SoC should have a high performance as the primary feature while the power consumption is the secondary. The SoC Consumer Portable (SoC-CP) Driver is probably the most representative of today's SoC design, as it is deployed in the fastest-growth products: smart phones and tablets.

The market demands for increased performance, functionality, smaller size, lower power and lower cost leads to having more and more processing engines on a single chip. Figure 1.4 shows quantified chip complexity trends for the SoC Consumer Portable. The exponential growth in the number of processing elements (PEs) is shown clearly in this plot, rapidly moving from hundreds toward thousands. Another interesting observation is that memory area is slowly but steadily dominating over logic area, even though main memory is projected to grow with PEs at a constant ratio. This is due primarily to the fact that most on-chip memory will be used as a cache to store partial copies of main memory. Unfortunately caches need to grow super-linearly with the number of cores to maintain external memory bandwidth under control. The ITRS projection is actually quite optimistic in terms of on-chip requirements, implicitly assuming that target applications will feature very good spatial or temporal locality.

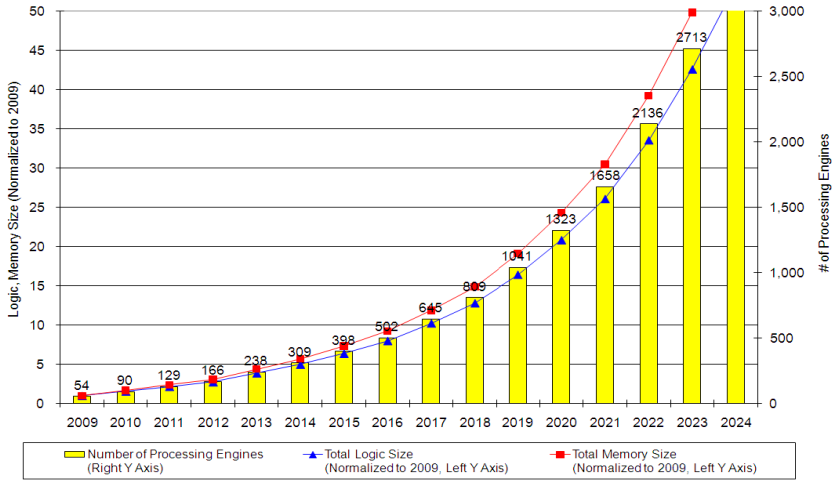


Figure 1.4: ITRS 2010 SoC complexity trends for SoC Consumer Portable

Figure 1.5 on the facing page shows the ITRS 2010's architectural template for the SoC Consumer Portable. It is a massively parallel architecture containing a small number of main processors, a much larger number of PEs (Processing Engines) and a corresponding amount of memory

(slightly decreasing PE/memory ratio). It also includes peripherals and related IO interfaces.

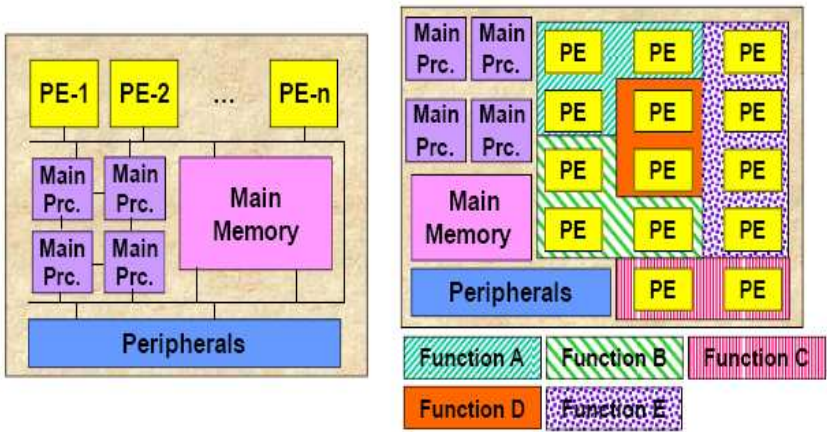


Figure 1.5: ITRS 2010 SoC architectural template for SoC Consumer Portable

In this figure, a PE is a hardware accelerator customized for a specific function. A function with a large-scale, highly complicated structure will be implemented as a set of PEs operating in parallel. Note that accelerators may take a wide range of architectural embodiments, from application-specific processors, to hardwired configurable units. The template does not mandate for specific processor array architectures or symmetric processors; the essential feature is the large number of PEs embedded within the SoC to implement a set of required functions. This template enables both high processing performance and low power consumption by virtue of parallel processing and hardware realization of specific functions. Figure 1.6 on the next page depicts a generic high-end mobile platform SoC, featuring four application subsystems for graphics, image, video and communication. These domain-specific computing fabrics have traditionally reached an order-of-magnitude higher power ($GOPS/W$) and area ($GOPS/mm^2$) efficiency than general-purpose multi-cores. This is achieved by carefully exploiting a mix of parallelism and specialization, often at the expense of flexibility and programmability.

In addition to SoC consumer portable and stationary products, main microprocessor manufacturers are migrating to chip multiprocessors (CMPs) for their latest products. In CMPs many cores are put together in the same chip and, as Moore's law continues to apply in the CMP era, we can expect to see a geometrically increasing number of processors and memories [111]. Today, even MPSoCs used in mid-range mobile phones

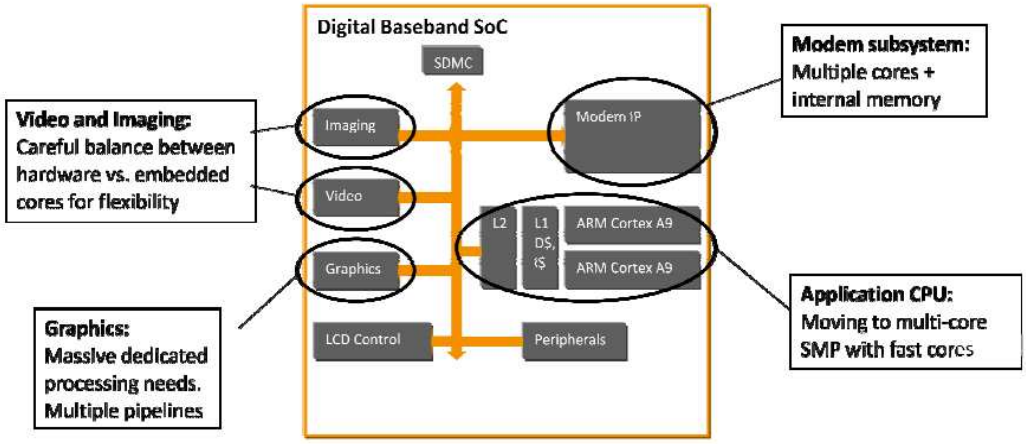


Figure 1.6: Digital Baseband SoC architecture and its heterogeneous computing resources

can easily contain tens of IP cores (Figure 1.7 on the facing page), and new chips with more than a hundred such internal units are appearing for various applications. Intel developed a research chip called *Single-chip Cloud Computer* (SCC) with 48 cores, under the Tera-scale computing research program [112] and a chip prototype that included 80 cores (known as TeraFlops research chip) [113]. More recently, Intel has announced the *Many Integrated core* (MIC) architecture [114] as the latest breakthrough in multi-core processors. The MIC project is the fruit of three research streams: the 80-core Tera-scale research chip program, the single-chip cloud computer initiative, and the Larrabee many-core visual computing project [115].

One important architectural challenge which is not explicitly highlighted in ITRS SoC architecture template is namely the SoC communication infrastructure. The ITRS template implicitly assumes that the on-chip interconnection infrastructure will act as an ideal transport layer for all communication among SoC sub-systems, but this is not an easy objective to achieve. The trend expressing the number of IP cores that can be integrated on a chip is exponential. How to effectively provide communication resources among such a number of building blocks is clearly a challenge. In fact, it is likely a key factor in determining the success or failure of upcoming MPSoCs will be the ability to efficiently provide the communication backbone into which to seamlessly plug a variety of IP cores. In the next section we give an overview of SoC communication infrastructure and its current trend.

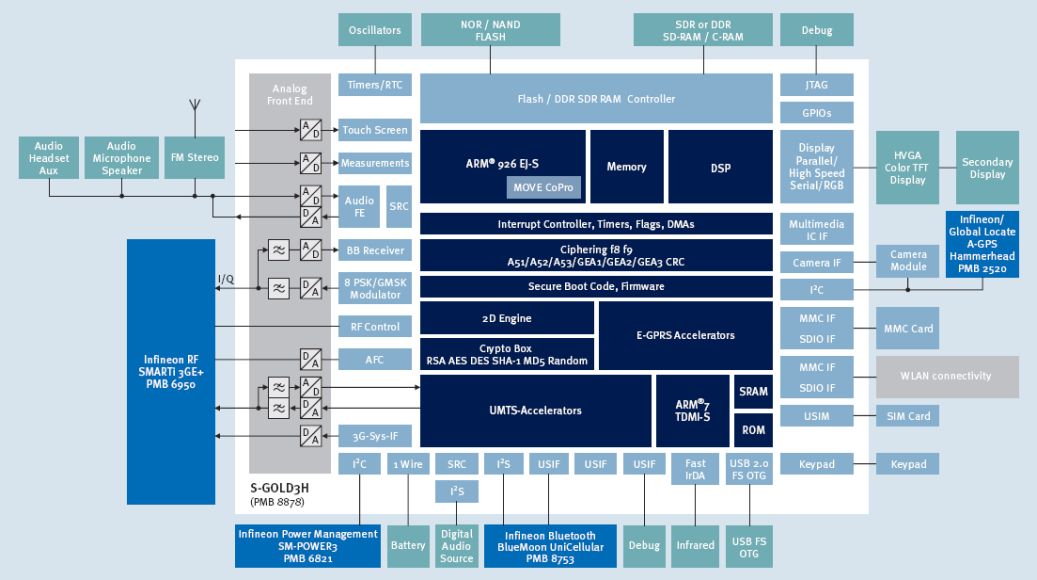


Figure 1.7: Block diagram of the Infineon S-GOLD3H chip for multimedia HSDPA mobile phones.

1.2

SoC communication infrastructure

SoCs need a kind of communication subsystems to interconnect processors, memories, application-specific integrated circuits ASICs, and other cores on a single-chip. The huge communication demands of complex applications running on an SoC and the abundant computation power available on chip put tremendous pressure on the communication architecture. Consequently, scalable communication architectures are needed for efficient implementation of future systems. This communication subsystem should feature low latency and low power as well as high performance.

1.2.1 Point-to-point (P2P) communication

Traditionally, two types of on-chip communication schemes have been considered, namely, point-to-point (P2P) and bus-based communication architectures. P2P communication architectures can provide the highest communication performance at the expense of dedicated links among all the communicating IP pairs. However, these architectures cannot be scaled in terms of high complexity, design effort and cost. Figure 1.8 shows

the communication diagram of a very simple SoC design (MPEG-2) and its related P2P graph. In this design due to the small number of point-to-point connections the link-to-node ratio in the MPEG-2 implementation shown in 1.8-b is only 1.5, while the same ratio is found to be 3.0 for a complete graph of the same size, even when all the links are unidirectional. It should be noted that for many other applications where a subset of cores communicate with all remaining nodes, the overhead incurred by dedicated channels of the P2P architecture is significant.

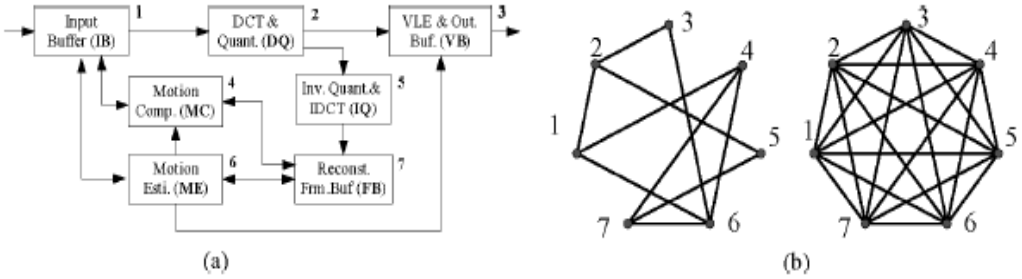


Figure 1.8: (a) MPEG-2 encoder implementation and its; (b) graph representation using a P2P communication architecture and a complete graph of the same size. NEED TO BE REVISED

1.2.2 Shared Buses

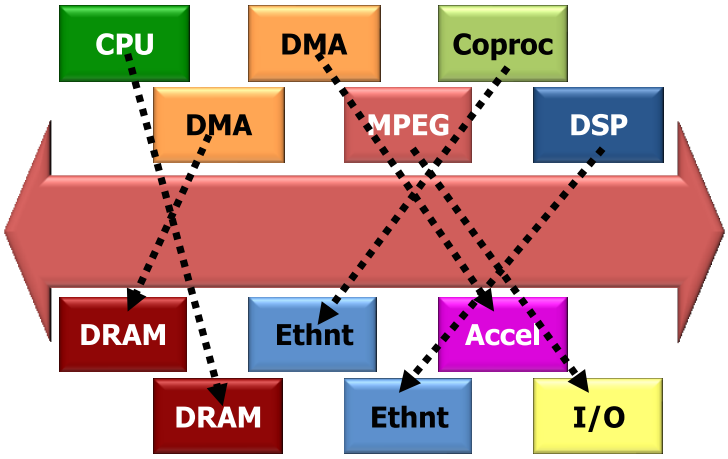
On the other hand, bus-based architectures can connect a few tens of IP cores in a cost-efficient manner by eliminating the dedicated channels required by P2P communication architectures resulting in a reduced design complexity.

Shared buses have as a main advantage their extreme simplicity, both in conceptual terms and circuit design terms. However, they are unsuitable for next-generation MPSoCs, due to the following fundamental limitations (Figure 1.9 on the next page):

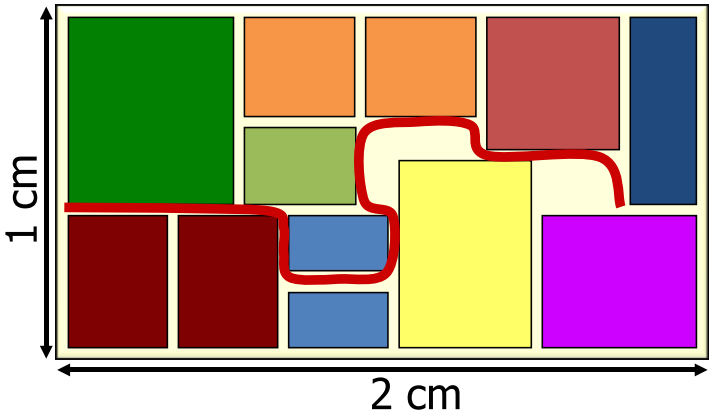
- Their maximum available bandwidth is limited by their shared nature, and this limit can easily be trespassed when the number of attached cores becomes more than a few.

- Their electrical performance degrades dramatically with new lithographic nodes. Since a shared bus is necessarily a structure composed of global wires, its propagation delay actually increases with

miniaturization. Therefore, with each new chip generation, a shared bus becomes slower in operating frequency, and even slower when compared to the progress in speed achieved by logic blocks. Long wires are also more vulnerable to crosstalk, variability and electrical noise, all of which represent increasingly serious problems in current technologies.



(a) Transactions cannot occur in parallel, even among independent pairs of devices.



(b) Global wiring spanning across the chip has poor electrical performance.

Figure 1.9: Shared bus limitations.

In response to these issues, buses have undergone evolutions [3, 4, 5] in two respects: protocols and topologies (Figure 1.10 on the following page).

Protocol evolution allows for more sophisticated handshakes occurring on the bus, such as multiple outstanding transactions, out-of-order

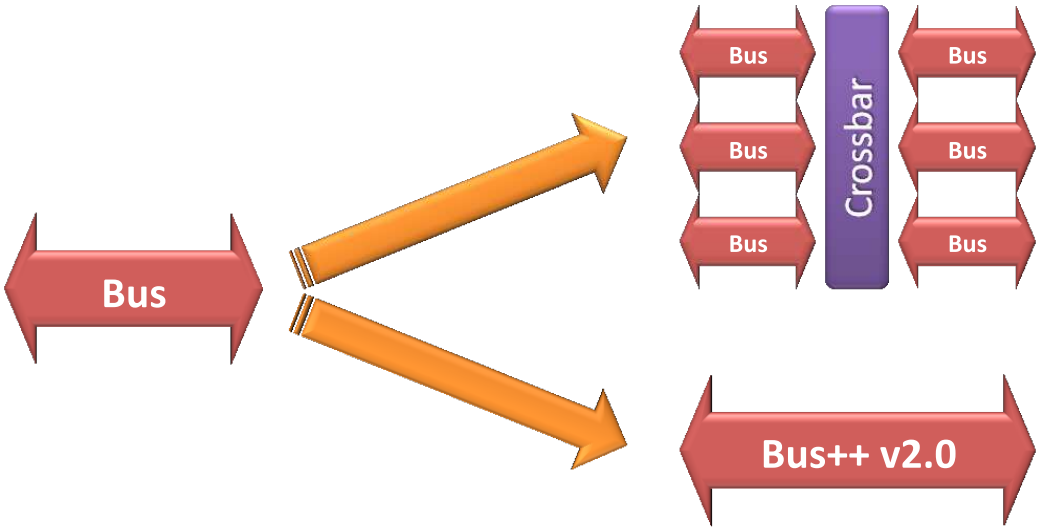


Figure 1.10: Evolution of shared buses towards hierarchical buses and more advanced protocols.

retirement of responses, burst requests, smarter arbitration, *etc.*. These evolutions help in making the best possible use of the limited available bandwidth. While useful in temporarily reducing the extent of bandwidth issues, they still do not provide a long-term solution to the fundamental limitations of buses.

Topology evolutions are a more radical departure from the original shared bus paradigm. The main principle is to deploy multiple buses, attached to each other by *bridges* or elements called *crossbars* - *i.e.*, devices providing full simultaneous connectivity among all their inputs and all their outputs. The outcome is often called *hierarchical bus* or *multilayer bus*. Hierarchical buses are a much better response to MPSoC design concerns, and in fact most MPSoCs today leverage hierarchical buses.

Even despite these improvements, buses are still a sub-optimal solution for next-generation MPSoCs, due to several factors:

- Hierarchical buses are mostly a manual workaround, by means of which designers try to fix the issues they are presently facing. The development and verification steps have to be performed mostly manually, and it is hard to guarantee that the design will scale upon the addition of more IP cores in the next revision of the design. Issues such as deadlock prevention, address mapping, and compliance with performance objectives are among the challenges left to designers.

- From the physical design point of view, hierarchical buses are not much better than shared buses. While allowing for some wire segmentation (wires only have to span regions of the whole system), wires are still normally laid with wide parallelism (typically more than 100 wires for a 32-bit bus and possibly close to 200 for a 64-bit bus), and they still connect multiple entities (a large *fanout*). Together, these issues mean that buses are still electrically inefficient, and difficult to route during physical design.

- There are conflicting trade-offs between compatibility requirements, driven by IP blocks reuse strategies, and the introduction of the new bus evolutions driven by technology changes. In many cases, introducing new features has required many changes in the bus implementation, but more importantly in the bus interfaces (for example, the evolution from AMBA ASB to AHB2.0, then AMBA AHB-Lite, then AMBA AXI), with major impacts on IP reusability and new IP design. As a consequence, bus architectures can not closely follow process evolution, nor system architecture evolution. The bus architects must always make compromises between the various driving forces, and resist change as much as possible.

1.2.3 Network on Chips (NoCs)

In contrast to these methods, the network-on-chip (NoC) approach emerged as a promising solution to on-chip communication problems [46, 54, 56, 82]. (Figure 1.11 on the next page).

NoCs are packet-switching networks, brought to the on-chip level. The rationale is that, since the complexity of on-chip communication is rapidly approaching that of large area systems in terms of actors, it makes sense to reuse some of the solutions devised in the latter space. Therefore, NoCs are based upon topologies of *switches* (also called *routers*) distributing packets around, over point-to-point *links*. Since existing IP cores do not normally communicate by means of packets, NETWORK INTERFACES (NIS) (also called *network adapters*) are in charge of protocol conversion; they convert commands appearing on the pinout of IP cores into packets, and vice versa at the receiving end of a transaction.

NoCs have the potential to bring a large number of advantages to on-chip communication, such as:

- Virtually unlimited architectural scalability. As known from wide area networks, it is easy to comply with higher bandwidth re-

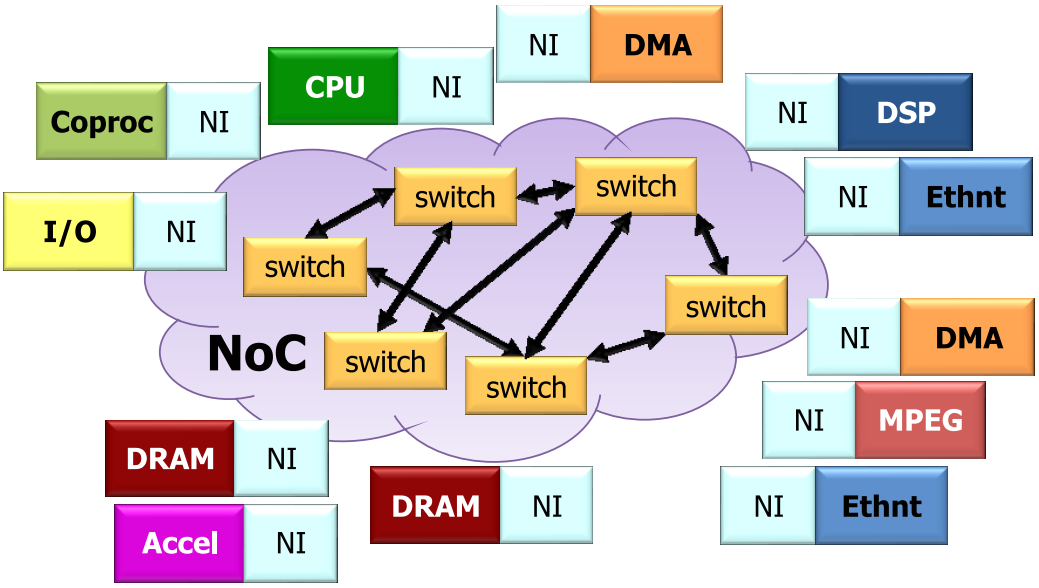


Figure 1.11: Conceptual view of a Network-on-Chip.

quirements by larger numbers of cores simply by deploying more switches and links.

- Much better electrical performance. All connections are point-to-point. The length of inter-switch links is a design parameter that can be adjusted. The wire parallelism in links can be controlled at will, since packet transmission can be serialized. All these factors imply faster propagation times and total control over crosstalk issues.

- Also due to the possibility of having narrower links than in buses (*e.g.* 20 bits instead of 100), routing concerns are greatly alleviated, and wiring overhead is dramatically reduced. This leads to higher wire utilization and efficiency.

- Faster and easier design closure achievement. Physical design improvements make NoCs, in general, more predictable than buses. Therefore, it is more unlikely that costly respins will be required upon physical design and performance qualification.

- Better performance under load. Since the operating frequency can be higher than in buses, the data width is a parameter, and communication flows can be handled in parallel with suitable NoC topology design, virtually any bandwidth load can be tackled.

- More modular, plug&play-oriented approach to system assembly. IP cores are attached in point-to-point fashion to dedicated NIs; NIs can be specialized for any interface that may be needed, either industry standards such as AMBA AHB or any custom protocol. Potentially any core may be seamlessly attached to a NoC given the proper NI. Computation and communication concerns are clearly decoupled at the NI level.

- Potential for the development of streamlined design flows. While hierarchical buses are often assembled by hand and therefore must be tuned and validated with manual intervention, a network can be designed, optimized and verified by automated means, leading to large savings in design times, and getting a solution closer to optimality.

- A much larger design space. NoCs can be tuned in a variety of parameters (topology, buffering, data widths, arbitrations, routing choices, *etc.*), leading to higher chances of optimally matching design requirements. Being distributed, modular structures, NoCs can also accommodate differently tuned regions. For example, some portions of a NoC could be tuned statically for lower resource usage and lower performance (*e.g.* by reducing the data width), or could dynamically adjust their mode of operation (*e.g.* frequency, voltage scaling).

At the same time, NoCs are facing a completely different set of constraints compared to wide area networks. While in the latter environment a switch is implemented with at least one dedicated chip, in a NoC the switch must occupy a tiny fraction of the chip real estate. This means that some of the principles acquired in wide area networking have to be revisited. Some of the challenges lying ahead of NoCs include:

- The trade-offs among network features, area and power budgets have to be studied from scratch. Policies which are widely accepted in general networking (*e.g.* dynamic packet routing) must be reassessed to evaluate their impact on silicon area.

- Performance requirements are very different in the on-chip domain, also due to the completely different properties of on-chip wiring. Bandwidth milestones are much easier to achieve, since information transfer across on-chip wires is much faster than across long cables. Conversely, latency bounds are much stricter; while milliseconds or even hundreds of milliseconds are acceptable for wide area

networks, IP cores on a chip normally require response times of few nanoseconds.

- Contrary to wide area networks, where nodes may often be dynamically connected to and disconnected from the network, in NoCs the set of attached IP cores is obviously fixed. In many applications, it is also relatively easy to statically characterize the traffic profiles of such IP cores. This opens up the possibility of thoroughly customizing NoCs for specific workloads. How to achieve this goal is, however, less clear.
- Design tools for NoCs can be developed, but, as above, how exactly is an open question. The customizability of NoCs, while an asset, is also an issue when it comes to devising tools capable of pruning the design space in search of the optimal solutions. The problem is compounded by the need to take into account both architectural and physical properties; by the need to guarantee design closure; and by the need to validate that the outcome is fully functional, *e.g.* deadlock-free and compliant with performance objectives.
- NoCs are a recent technology, and as such, they are in need of the development of thorough infrastructure. In addition to design tools, this includes simulators, emulation platforms (such as FIELD PROGRAMMABLE GATE ARRAY (FPGA) boards), and back-end flows for the implementation on both FPGAs and APPLICATION-SPECIFIC INTEGRATED CIRCUITS (ASICS).

The NoC approach features *Bandwidth scalability*, *Design reuse* and *Predictability*. However, NoCs have a relatively high latency and, therefore, are not adequate for low latency communications such as shared-L1 processor-to-memory where accessing to L1 memory should be completed in a few cycles. For these types of communication highly optimized special-purpose interconnects are required.

1.3 Cluster-based MPSoCs

Recently, several many-core architectures have been proposed that leverage tightly-coupled clusters as a building block. Examples include the HyperCore Architecture Line (HAL) processors from Plurality [172], ST Microelectronics Platform 2012 [173], or even GPGPUs like NVIDIA Fermi

[151]. In a shared memory paradigm, these designs try to overcome the scalability limitations encountered when increasing the number of processing elements (PEs) that share a unique interconnection and memory system [174] by creating a hierarchical design where PEs are clustered into small-medium sized subsystems. The small number of PEs makes it possible to design a low-latency interconnect between processors and L1 (in-cluster) memories, while scaling to larger system sizes is enabled by replicating clusters and interconnecting them with a scalable medium like a Network-on-Chip (NoC). Figure 1.12 depicts the overall architecture of ST Microelectronic Platform 2012 [173]. It is based on a modular infrastructure, as depicted in the figure. Fabric-level communication is based on an asynchronous NoC organized in a 2D mesh structure. The routers of this NoC are implemented in a Quasi-Delay-Insensitive (QDI) asynchronous (clock-less) logic. They provide a natural Globally Asynchronous Locally Synchronous (GALS) scheme by isolating the clusters logically and electrically. Asynchronous-to-Synchronous interfaces based on FIFOs connect the NoC to the different clusters. Following the GALS interface, a Network Interface (NI) is used as a logical link between a cluster and the NoC. It is used for encapsulating the address-based protocol of the cluster into a packet-based NoC-compatible protocol. It also gives access to the main clock, variability and power (CVP) controller that is used to control a power management harness.

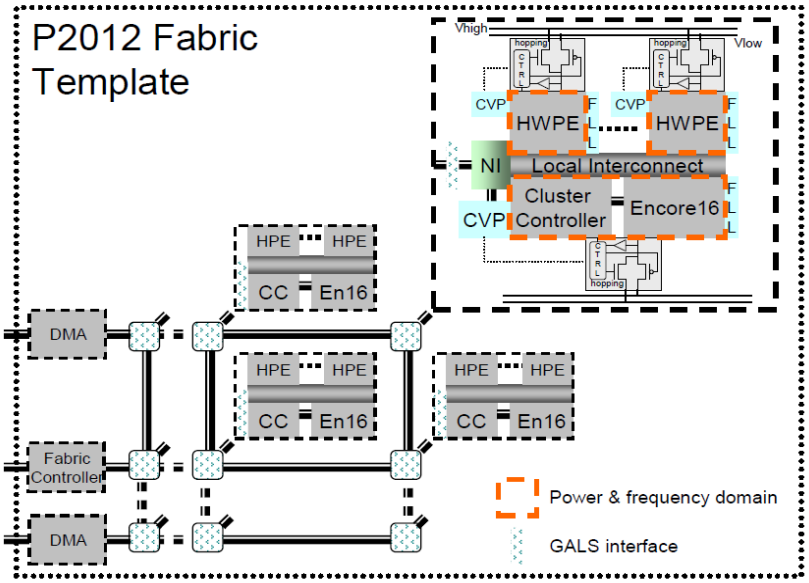


Figure 1.12: The ST Microelectronic P2012 fabric [173].

1.3.1 Inter-cluster communication

As mentioned, in the cluster-based SoCs each cluster has a few number of PEs. To enable scaling the system to a larger size, clusters are connected through a scalable communication infrastructure. Network on chip as described in Section 1.2.3 on page 13 is scalable and can be used for this purpose. Since clusters are isolated logically and electrically, the NoC used for inter-clusters communications is based on GALS. We will describe GALS NoCs in more details in Chapter 2 on page 25.

1.3.2 Intra-cluster communication

Each cluster contains a few PEs which usually share a multi-banked L1 memory. As mentioned, the cluster architecture is fixed and scaling is enabled by replicating clusters. Since the number of PEs inside each cluster is relatively small, we can use a high performance and low latency interconnection infrastructure for communication between cores and the shared L1 memory. This interconnection cannot be neither BUS nor NoCs because and a highly optimized special-purpose interconnect is required. We will describe this interconnect in more details in Chapter 6 on page 119.

1.4 MPSoC Design Flow: a quick look

Designing of an MPSoC requires several sequential and concurrent steps [2]. Figure 1.13 on the facing page shows an idealized MPSoC design flow.

Most designs start from an abstract level of the application described by customer or marketing requirement specifications. At the first step, the algorithm is selected, optimized and implemented by a high level programming language like C++. During or after this step, a complete verification is performed to evaluate the correctness of the algorithm and the implementation. In the next step, designers perform hardware-software partitioning to map some parts of the functionality on hardware and remaining on software. A careful architecture exploration is performed to choose the best hardware or software among the candidates. The output of this step is the *architectural model* of the system where the computation blocks of the whole design are specified communicating with each other in a very abstract manner. In the next step, the communication architecture is defined and synthesized for the architecture model created at the pre-

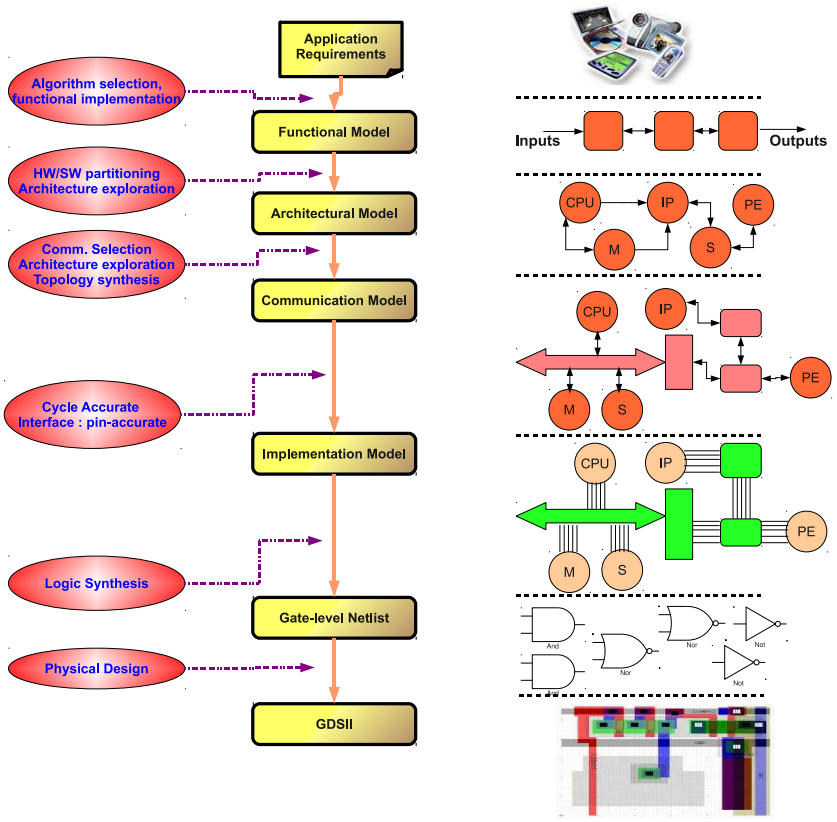


Figure 1.13: High level model of the MPSoC design flow

vious step. Here again a design space exploration is performed to choose the best communication architecture among different candidates (e.g. Bus, P2P, NoC, and etc) which meets the application constraints. Note that, different communication mechanism can be selected for different parts of the system. The result of this phase is *communication model* of the system. The communication model is an enhanced architectural model which defines not only the computational entities, but also the communication schemes. This model is usually simulated using high level HDLs like SystemC to verify the functionality of the whole system in terms of both communication and computation. The enhanced architecture model is then implemented at Register Transfer Level (RTL). To do so, the behaviors inside the computation blocks are implemented in a cycle-accurate manner, and the interfaces between computation and communication entities are refined to a pin-accurate level where all the signals and their timing are explicitly defined. The output of this step is the "implementation model" of the system which is essentially an RTL model. A high effort of verification

is performed to evaluate the correct functionality of this model. The RTL model is then imported into the logic synthesis tool to create a gate-level netlist of the design on a specific and pre-selected technology. Post synthesis verification as well as equivalence checking is performed to evaluate the correct mapping. After synthesis the back-end flow which includes Placement, Clock synthesis and Routing is performed. The result of this phase is a GDSII file which is delivered to the foundry to fabricate the MPSoC design. Note that, this flow is a high level model of the actual MPSoC design flow and does not show several iteration and verification steps as well as IP reuse steps which is usual in MPSoC design.

1.5 Contributions of the Dissertation

This dissertation attempts to cover a vertical line in the design flow of MPSoCs and gives insight on why designing an MPSoC is becoming more challenging in deep-submicron during all stages of the design flow. It aims to develop new mechanisms at different levels of abstractions including architecture, design, netlist, and layout to have robust, variation-tolerant and energy efficient MPSoCs. The main contributions of this dissertation are:

- Design, implementation and fabrication of a GALS-NoC test-chip in a 40nm process technology. Seven different synchronization techniques were developed in different sub-systems and implemented on the test-chip to evaluate the effectiveness of GALS-NoCs with respect to the fully synchronous NoCs. The yield results of this test-chip motivate the need for techniques to detect and work around manufacturing faults and process variations in interconnection networks. At the physical level, a new wire routing approach is developed to robustly route the wires of global links in the NoC in such a way that they get routed on the same trajectory leading to less delay and length variation. Having less delay variations between wires of each link makes the source synchronization more robust.

- Developing an efficient and reliable NoC switch architecture called *ReliNoC* which distributes the traffic to two different physical channels based on the class of the traffic, which can be either QoS or normal. *ReliNoC* takes advantage of the replicated components by using

them as replacements in presence of faults when needed, tolerating a wide range of combinations of network component failures.

- Designing a distributed functional test mechanism for NoCs which scales to large-scale networks with general topologies and routing algorithms. Each router and its links are tested using neighbors in different phases. The router under test is in test mode while all other parts of the NoC are operational. We use TMR (Triple Module Redundancy) for the robustness of all testing components which are added into the switch.

- A fully combinational Mesh-of-Tree (MoT) interconnection network suitable for shared-L1 processor clusters is implemented featuring single-cycle transfer from processor to memory and vice versa. Moreover, an advanced synthesis and physical optimization strategy which leverages standard design implementation tools is implemented to achieve high-quality results in terms of delay, power and area (DPA) even for large network instantiations. Furthermore, a wide range of network configurations are explored to analyze scalability and DPA trade-offs.

- Extending the single-cycle interconnection network to a resilient architecture which can tolerate delay variations due to aging or static variations with a small overhead on read/write latency. In case of delay variations, the network is reconfigured so that it can overcome the timing failure by inserting a pipeline stage in the path and increasing the latency of the read/write transaction.

- Finally, development of fine-grained row-based power management to have robust and variation-tolerant MPSoCs at near-threshold region. A dual-Vdd technique for Near-threshold operation is implemented that can be used to fine tune the performance of a circuit (in case of variation) by selectively powering up the timing critical gates in the design at a slightly higher supply voltage than the rest of the circuit while minimizing the total power of the circuit. We propose this technique as an orthogonal way of compensating variability at the post-fabrication stage. This is an alternative strategy with respect to the previous contributions which are design techniques, and it enforces post silicon detection and compensation.

This dissertation presents work jointly carried on by the author and by several co-authors. While efforts will be made to focus mainly on the areas

in which the author was primarily involved, it is actually impossible to completely decouple the contributions. Further, describing the complete framework in which the author's research was performed is helpful for a much better understanding of the goals and the achievements of this effort. A list of the papers on which this work is based, complete with the names of all the co-authors, is supplied in Appendix A on page 183.

This dissertation merely reflects the research choices done by the author and his co-authors based on the information and analyses available to them (which will be substantiated wherever possible across the present dissertation), and based on time constraints.

1.6 Organization of the Dissertation

To give a quick outline of this dissertation, Chapters 2 and 3 give an overview on designing GALS-based NoC and related issues. Chapter 4 and 5 cover Reliability and Testing in NoCs. Chapter 6 and 7 are related to robust low-latency communication suitable for shared-L1 clusters. Chapter 8 covers fine-grained performance compensation and power management techniques for energy efficient MPSoCs, and finally Chapter 9 concludes the dissertation.

In more details, Chapter 2 gives an overview on state-of-the-arts of GALS NoCs and presents a NoC architecture which is based on source synchronization. Later, an implementation of a chip in the advanced 40nm CMOS process aiming at the assessment of GALS technology for nanoscale designs is presented together with the experimental results obtained after chip fabrication.

Chapter 3 presents a new physical routing flow which is suitable for global links in a NoC especially GALS based NoCs. This algorithm is implemented in commercial back-end tools and considers all nets of a link together and efficiently routes NoCs links as an atomic entity (a bundle). As a result, the variability in link net attributes such as length, resistance, load, and delay is much reduced. Moreover, enforcing the same "trajectory" on all nets of a link facilitates wire spacing and shielding.

Chapter 4 introduces a NoC architecture which can withstand failures, while maintaining not only basic connectivity, but also quality-of-service support based on packet priorities. This network leverages a dual physical channel switch architecture which removes the control overhead of virtual channels (VCs) and utilizes the inherent redundancy within the 2-channel

switch to provide spares for faulty elements.

Chapter 5 presents a distributed functional test mechanism for NoCs which scales to large-scale networks with general topologies and routing algorithms. Each router and its links are tested using neighbors in different phases. The router under test is in test mode while all other parts of the NoC are operational.

Chapter 6 describes the design flow of a parametric, fully combinational Mesh-of-Trees (MoT) interconnection networkto support high-performance, single-cycle communication between processors and memories in L1-coupled processor clusters. The interconnect IP is described in synthesizable RTL and it is coupled with a design automation strategy mixing advanced synthesis and physical optimization to achieve optimal delay, power, area (DPA) under a wide range of design constraints.

In Chapter 7 a reliable and variation-tolerant architecture for shared-L1 processor clusters is presented. The architecture uses the single-cycle mesh of tree proposed in Chapter 6 as the interconnection network between processors and shared-L1 memory. The delay variation is mitigated by inserting an on-demand pipeline stage on the effected path.

Chapter 8 covers the fine-grained power management techniques for energy efficient and ultra-low-power MPSoCs. Row-based dual-Vdd is presented as a new technique for robust near-threshold design suitable for ultra-low-power MPSoCs. It can be used to fine tune the performance of a circuit in presence of variations by selectively powering up the timing critical gates in the design at a slightly higher supply voltage than the rest of the circuit while minimizing the total power of the circuit.

Finally, chapter 9 presents the overall contribution of this research and conclusions as well as the future work.

CHAPTER 2

Globally Asynchronous and Locally Syncronous NoCs

This chapter[1] evaluates GALS NoC as an effective and scalable interconnection network for MPSoCs. Asynchronous architecture is one of the main techniques to reduce power and increase performance in SoCs. GALS NoC can be used for inter-cluster communication in the cluster-based Multi-core systems. In this work seven different synchronization approaches are implemented on a fabricated test-chip at 40nm process technology to evaluate the effectiveness and issues of various NoC methodologies at deep submicron. The yield results of the testchip presented in this chapter motivate the need for working on variability and reliability issues in on-chip interconnection network at deep-submicron.

2.1 Motivation and Key Challenges

Many of today SoCs are Globally Asynchronous and Locally Synchronous. Global asynchronicity is helpful to reduce the (power, performance, area) cost of global clock distribution, which is becoming unaffordable when all sources of uncertainty are added up in the design of the clock tree.

There is today little doubt on the fact that a high-performance and cost-effective NoC can only be designed in 45nm and beyond under a relaxed synchronization assumption [38, 37]. One effective method to address

[1]Most of the credit for the work described in this chapter goes to Alessandro Strano, Prof. Davide Bertozzi, and Prof. Luca Benini.

this issue is through the use of globally asynchronous and locally synchronous (GALS) architectures, where the chip is partitioned into multiple independent voltage and frequency domains. Each domain is clocked synchronously while inter-domain communication is achieved through specific interconnect techniques and circuits [35]. The methodology of inter-domain communication is a crucial design point for GALS architectures. One approach currently experimented in GALS NoC prototypes consists of using purely asynchronous clockless handshaking for transferring data words across clock domains [39, 40]. A few chip demonstrators prove the viability of this solution [32, 33, 34], but they have not achieved widespread adoption of asynchronous NoCs in the industrial arena yet. In fact, asynchronous handshaking techniques are complex and use unconventional circuits (e.g., Muller C-elements) usually unavailable in industrial technology libraries. Moreover, asynchronous logic is not well supported by CAD tools. In this context, the most effective solution found so far for actual chip fabrication and industry-relevant designs consists of implementing routers and GALS interfaces as hard macros using ad-hoc design styles [34]. The hard macro methodology is however more a way of working around the lack of proper design and verification tools and thus guaranteeing performance during physical implementation rather than a way of optimizing area. In fact, this latter remains consistently larger than fully synchronous NoC counterparts (1,8x in [34]). What is currently missing in the open literature as well as in the industrial prototyping experience is a mature GALS NoC architecture making use of source synchronous communication techniques. This method achieves high efficiency by obtaining an ideal throughput of one data word per source clock cycle with a design style which is more easily compatible with common standard cell design flows. In spite of a few early works taking this approach [29, 28], this GALS design style has not been consistently brought to maturity over time, therefore reducing source synchronous communication to a nice concept with only limited relevance for real-life designs. For instance, the area and latency overhead associated with the use of synchronizers has never been tackled in a systematic way, and even advanced research prototypes from industry like the Intel Polaris chip [36] live with such an overhead. As a consequence, source synchronous communication has never truly evolved from a concept to a mature technology.

2.2

Related Work

Chip synchronization in the nanoscale regime is an urgent challenge orthogonal to all parallel computing platforms, even in the SoC domain [23]. Since clock tree design and tuning for variability robustness comes at an non-negligible cost [12, 13, 11, 9]. The work described in this chapter investigates how to live with uncertainties by means of proper architectural support in the on-chip interconnect. The GALS paradigm has been frequently experimented by using asynchronous logic [39, 40]. A chip dedicated to flexible baseband processing for 3G/4G wireless telecommunication applications and making use of an asynchronous NoC is described in [32, 33]. However, currently the intricacy of asynchronous design and its poor CAD tool support makes the design of hard macros with ad hoc design techniques [14] the only viable solution for industrial exploitation [34]. This penalizes flexibility and is not able to cut down on area overhead of timing-robust asynchronous realizations. The practical viability of synchronizer-based GALS networks has been proven in [29], where the hierarchical clock tree synthesis technique for such systems is detailed. Since there is no parametric exploration there, it is not possible to validate a common opinion (for instance, reported in [21, 27, 25]) that large on-chip power consumption can be reduced by replacing the balanced clock tree with a GALS clocking scheme which only guarantees minimal clock skew within the local processing elements. This common sense claim has remained unproved so far, without any precise quantification.

A circuit switched source synchronous GALS link is described in [28], making use of long distance interconnect paths. It is then experimented on a 65nm reconfigurable NoC for an heterogeneous GALS many-core platform. However, there is no comparison whatsoever with alternative clocking styles and/or implementations.

A mesochronous link is integrated within a Multi Processor tiled architecture based on a Network-on-Chip communication backbone on a CMOS 65 nm technology in [25]. The work builds on a full-duplex link architecture illustrated in [24] and on integrated flow control [22]. The baseline mesochronous synchronizer is instead proposed in [27]. However, the synchronizer is still an external module to the NoC. The same drawback is exposed by a different synchronization architecture, detailed in [36].

A new design style where synchronizers are merged with NoC building blocks is introduced in [19, 18]. The approach is applied to a mesochronous synchronizer and to a dual-clock FIFO [20]. Timing con-

straints as a function of varying architecture and physical design parameters are analyzed in [17].

2.3 GALS Test chip

Although significant potential was reported by the academia, there are no complete GALS-based NoC implementations of real applications in the industry reported to-date. However, the growing challenges, imposed by the unrelenting pace of technology scaling to the nanoscale regime, urge for an efficient and safe system-level integration methodology. Consequently, we targeted the implementation of a chip in the advanced 40 nm CMOS process, aiming at the assessment of GALS technology for nanoscale designs. The chip was named Moonrake. Part of the chip was dedicated to GALS-based NoCs. The intention was to evaluate GALS vs. standard synchronous technology on the same die, by implementing synchronous and GALS counterparts of the same baseline designs.

The need to validate a new synchronization technology and the poor experimentation of the newly developed 40nm technology at the time made the risk associated with testchip fabrication pretty high. In order to minimize such risk, the final decision for the NoC section of the testchip was to instantiate a number of small and replicated sub-systems each validating a different feature of the new synchronization technology. Replication of the sub-systems enables to amortize the risk of manufacturing concerns. To push this approach to the limit, the decision was to replicate sub-systems validating the same concepts even more in order to reflect different operating speeds. In practice, some sub-systems were selected for clocking from an external low-speed clock source through the JTAG interface, while other sub-systems with similar functionality were selected for higher-speed clocking from the PLL. The lower speed of clocking from an external source is associated not only with the inherent limitations of a possible test setup, but also with the limited driving capability of I/O pins. In both cases (internal vs external clocking), the sub-systems testing mesochronous interfaces were made capable of tolerating an increasing amount of clock skew between transmitter and receiver clock domains.

2.3.1 Subsystems of the testchip

Figure 2.1 shows the block diagram of the NoC section of the testchip. Each sub-system replicates the same baseline network-on-chip template, a 2-ary 1-mesh topology, where every switch of the network is connected to 2 cores (a memory core and a tester block). The cores are connected to JTAG interfaces in order to be programmable by external input pins. In particular, the testchip compares a fully-synchronous NoC subsystem with two mesochronous NoC sub-systems (a first one composed by loosely coupled and a second one by tightly coupled mesochronous synchronizers) and a GALS NoC sub-system integrating also dual-clock FIFO interfaces. As mentioned above, in order to test the reliability of the source synchronous communication and to compare sub-system latency, area and power under different frequency constraints, every sub-system is designed twice to work at low and high frequency. The sub-systems designed to work at low frequencies (i.e. frequency lower than 260MhZ) are fed by an external clock injected through the input pins of the JTAG interface. On the contrary, the sub-systems designed to work at high frequency receive the clock by a PLL (Phase Lock Loop) integrated into the testchip itself.

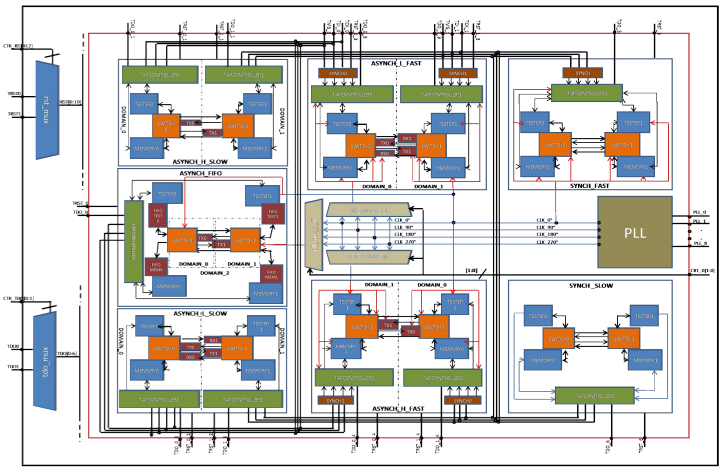


Figure 2.1: NoC section of the Moonrak testchip

The PLL is able to generate four different clock phases (0, 90, 180, 270) on distinct module output pins with a maximum speed of 400MhZ. The PLL frequency can be set through external pins. Moreover, each synchronizer-based sub-system receives two separated PLL clocks. In fact, the phase of the first clock is statically zero while the phase of the second clock can be selected by exploiting a 4x1 multiplexer and the external pins

for mux programming. A dedicated multiplexer is implemented for every synchronizer-based sub-system. As a result, the testing of the source synchronous interfaces can be performed also under different skew constraints. To notice that both the frequency and the phase of the PLL clock can be set during the power-on phase of the testchip. Also, the multiplexer delay ends up contributing to the clock phase offset between two mesochronous domains in the same sub-system, therefore such multiplexers have been synthesized for ultra-high speed of operation. The distinctive features of every sub-system can be described as follows:

The "Synch_Slow" sub-system

This design which is shown in Figure 2.2-right demonstrates a fully synchronous NoC communication at low frequency. The sub-system is composed by 1 JTAG controller, 2 switches, 2 memory cores and 2 tester blocks and is clocked by a unique external clock (see figure 2.b).

The "Synch_Fast" sub-system

As shown in Figure 2.2-left, this system implements a fully synchronous NoC communication at high frequency. The sub-system is composed by 1 JTAG controller, 2 switches, 2 memory cores, 2 tester blocks and 1 synch block. The synch block implements a brute force synchronizer and it is used to synchronize the JTAG inputs with the fast and real sub-system clock which comes from the PLL (see Figure 2.2-left).

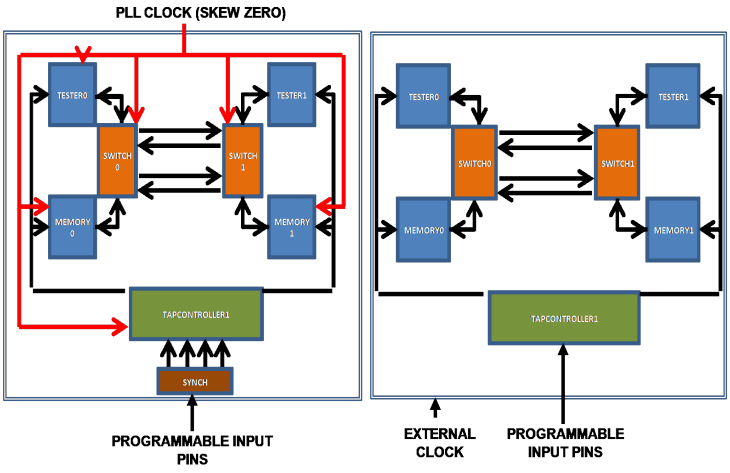


Figure 2.2: Synchronous sub-systems. left: The "Synch_fast" design; right: the "Synch_slow" design.

The "Asynch_Loose_Slow" sub-system

This sub-system is shown in Figure 2.3 on the following page-left. It has two mesochronous clock domains having the same low clock frequency but arbitrary phase offset. This sub-system uses loose mesochronous synchronizers (RX, for the datapath, and TX, for the control path) placed in the bidirectional link next to the switches. The RX module synchronizes the data and the TX module synchronizes the flow control signal absorbing the phase offset between the domains. The sub-system is composed by 2 JTAG controllers, 2 switches, 2 memory cores, 2 tester blocks, 2 loosely coupled RX datapath synchronizers and 2 loosely coupled TX control path synchronizers.

The "Asynch_Loose_Fast"

This design which is shown in Figure 2.3 on the next page-right has two mesochronous clock domains having the same high clock frequency but arbitrary phase offset. The domain-0 is driven by PLL clock with 0 skew while the domain1 is driven by the output of the dedicated multiplexer, thus selecting one of the pre-configured clock phase offsets. This sub-system uses loosely coupled mesochronous synchronizer (RX and TX) placed into the bidirectional link next to the switches. The sub-system is composed by 2 JTAG controllers, 2 switches, 2 memory cores, 2 tester blocks, 2 loosely coupled RX synchronizers (for the data path), 2 loosely coupled TX synchronizers (for the control path) and 2 synch blocks for synchronization with timing of the input pins.

The "Asynch_Hybrid_Slow"

As shown in Figure 2.4 on page 33-left, this design has two mesochronous clock domains having the same slow clock frequency but arbitrary phase offset. This sub-system has two tightly coupled mesochronous synchronizers integrated into the switch input stage (for the data path) and two loosely coupled TX synchronizers (for the control path) into the bidirectional link. The mix of the synchronizer implementation styles explains the name of this scheme ("hybrid coupling") and is motivated by the need to break a long timing path originating in the upstream output buffer, going through the mesochronous interface at the downstream switch and going back to the upstream switch. By implementing the control path synchronizer as loosely coupled, a minor area overhead is incurred (this is a 1-bit synchronizer), while breaking the timing path across the link and resulting in higher operating speeds for a given link length. The key

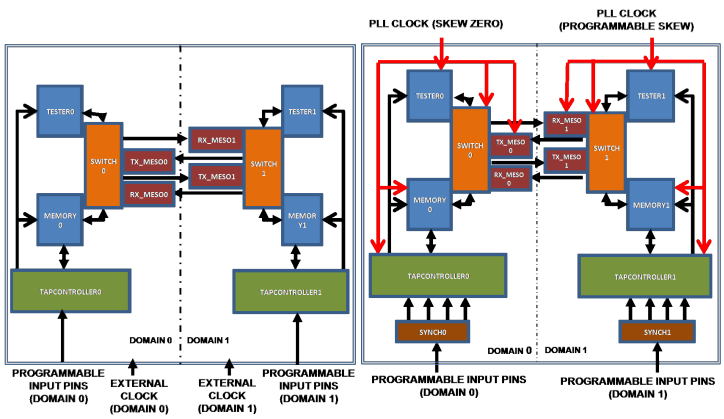


Figure 2.3: Loosely coupled sub-systems with mesochronous synchronizers. left: The "Asynch_Loose_Slow" design; right: the "Asynch_Loose_Fast" design.

take away is that an hybrid coupled design style closely tracks the latency/area/power benefits of a fully tightly coupled design style while at the same time proving more robust to design uncertainties and layout constraints. The sub-system is composed by 2 JTAG controllers, 2 switches, 2 memory cores, 2 tester blocks, 2 tightly coupled RX synchronizers and 2 loosely coupled TX synchronizers.

The "Asynch_Hybrid_Fast" sub-system

This design is shown in Figure 2.4 on the next page-right. It has two mesochronous clock domains having the same high clock frequency but arbitrary phase offset. This sub-system has two tightly coupled mesochronous synchronizers integrated into the switch input stage and two loosely coupled TX synchronizers into the bidirectional link. Again, this is a hybrid coupled synchronizer-based design. The domain-0 is driven by PLL clock with 0 skew while the domain1 is driven by the output of the dedicated multiplexer. The sub-system is composed by 2 JTAG controllers, 2 switches, 2 memory cores, 2 testers, 2 tightly coupled RX synchronizers, 2 loosely coupled TX synchronizers and 2 synch blocks for synchronization with the timing of the input pins.

The "Asynch_Fifo" sub-system

As showin in Figure 2.5 on page 34, this sub-system has three clock domains: domain-0 and domain-1 have the same high-speed clock with ar-

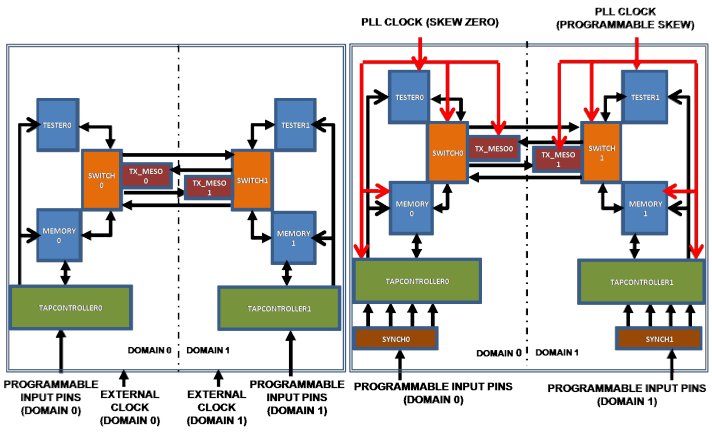


Figure 2.4: Hybrid coupled sub-systems. left: The "Asynch_Hybrid_Slow" design; right: the "Asynch_Hybrid_Fast" design.

bitrary phase offset while domain-2 has a distinct clock signal switching at a lower speed. The cores and the switches have true independent clocks with distinct frequency and offset. Since the sub-system has different clock frequencies, two tightly coupled dual-clock FIFOs are integrated into the switch input stage and two additional loosely coupled dual-clock FIFOs are instantiated next to the cores in order to synchronize the information coming from the cores and from the switches, respectively. Please note that the loosely coupled dual-clock FIFOs have not been merged with their associated network interfaces to reflect the case where custom network interfaces need to be rapidly reused in a GALS system, and therefore the merging process may turn out to be a lengthy or cost-ineffective process. In principle, nothing prevents from performing such a merging process with the network interface as well, even considering that the input stage of the network interface (and of its response path in particular) is composed by a FIFO.

Moreover, the sub-system has two tightly coupled mesochronous synchronizers integrated into the switch input stage and two loosely coupled TX synchronizers into the bidirectional link. The domain-0 is driven by PLL clock with 0 skew while the domain1 is driven by the output of the dedicated multiplexer, thus selecting a clock phase offset. Domain2 is clocked by an external clock through the JTAG interface. The sub-system is composed by 1 JTAG controllers, 2 switches, 2 memory cores, 2 tester blocks, 2 tightly coupled RX synchronizers, 2 loosely coupled TX synchronizers, 4 tightly coupled dual-clock FIFOs, 4 loosely coupled dual-clock FIFOs and 1 synch blocks (see figure 5). It is worth observing that this

sub-system reflects a typical operating condition: the IP cores operate at a slower speed than the on-chip network, where this latter can be inferred as the collection of mesochronous sub-domains.

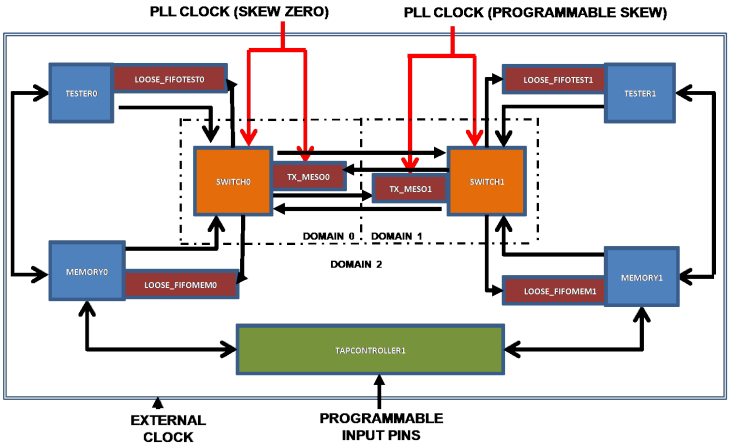


Figure 2.5: Dual-clock FIFO design.

2.3.2 Pin Requierments

As a final step, three further optimizations were required to meet the total pin budget. This latter was very tight, due to the need of making the baseband processor co-exist with the NoC test-structures on the same area-limited (due to cost constraints) silicon die. First of all, the programmable input pins and the external clocks ended up being shared between the 7 sub-systems. Consequently, in order to avoid an undesirable parallel setting of all the 7 sub-systems, the reset input signal crosses a de-multiplexer designed to enable a single sub-system at a time (i.e., the active reset drives only one of the seven sub-systems). As a result, it is possible to select the operative sub-system by driving the de-multiplexer with three dedicated input pins. Similarly, a multiplexer allows a single sub-system at a time to exploit part of the output pin resources (two shared output pins). In fact, each fast sub-system has both its output pins (one per JTAG) shared and each synchronizer-based slow sub-system has the output pin belonging to the domain0 (zero skew) dedicated but the output pin of the domain1 shared. This strategy allows slight output pin resource saving with a marginal impact on the testability of the design. Indeed, each slow sub-system (which can be viewed also as the safe and backup version with respect to the fast counterpart) has still a dedicated output pin ensuring

the sub-system testability, even when a failure in the output multiplexer occurs. Moreover, as mentioned above, the phase of the PLL clock is selected through a 4x1 multiplexer. As a result, each sub-system tightly belongs to its own clock phase multiplexer (the whole sub-system fails when a failure affects its multiplexer). Then, an instance of the clock phase multiplexer is replicated in front of each synchronizer-based sub-system to avoid multiples sub-system failures due to single multiplexer error. Anyway, to further reduce the number of global input pins, the three clock phase multiplexers are still driven by two shared input pins. To note that the final goal is to test a sub-system at a time and the parallel clock phase setting of the synchronizer-based sub-system does not reduce the testability of the design. As a conclusion, the input pins and the external clock sharing allows to save 33 input pins (they scaled from 44 to 11), the output pin sharing reduces from 11 to 8 the required output pins and, the clock phase multiplexers optimization saved 4 additional control pins. Finally, the global pin number scaled down from 70 to 30 (9 input/output pins are considered to drive the PLL). This approach enabled a massive optimization of the output/input pin requirements while marginally affecting the flexibility of the architectural test structures.

2.3.3 Floorplaning Constraints

The modularity of the testchip NoC architecture allowed an easier place-and-route. In fact, every sub-system was synthesized independently and treated as a soft-macro. Anyway, global and local constraints were imposed respectively in the testchip floorplan and in each sub-system. In particular, the final global testchip floorplan is represented in Figure 2.6 on the next page. The PLL is placed on the left side of the design and the seven NoC sub-systems are on the right side. To note that the PLL required a relevant percentage of the total testchip area. At a global floorplan level, the soft-macro of the fast sub-systems (in figure 6 denominated as Hybrid_Fast, Loose_Fast, Synch_Fast and Fifo_Fast sub-systems) were constrained to be placed as close as possible to the PLL. In fact, the distance between the PLL and the fast designs was minimized in order to reduce the length of the clock tree branches. This strategy mitigated the unpredictability of the skew between the clock leaves and, as a result, it increases the accuracy of the sub-system intra-domain skew set by the external pins.

Additionally, at a sub-system level, the source synchronous communication was constrained as well. In particular, Figure 2.7 on page 37 depicts the source synchronous communication in the hybrid coupled

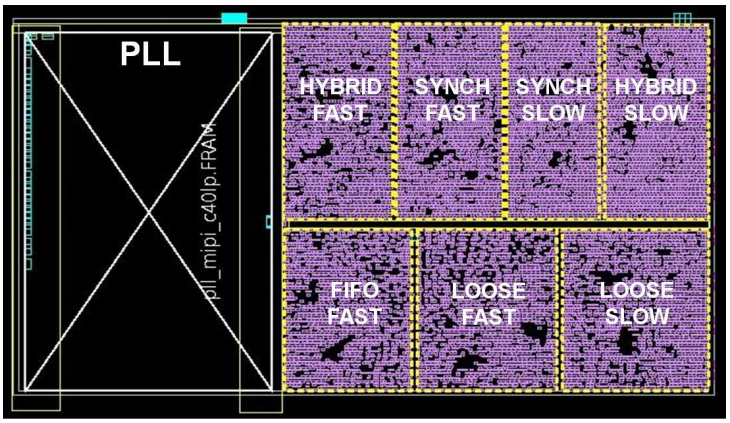


Figure 2.6: Global testchip floorplan.

sub-systems ("Asynch_Hybrid_Slow" and "Asynch_Hybrid_Fast"). In this case, the TX_MESO0 and TXMESO1 modules were placed respectively close to the SWITCH0 and SWITCH1 modules. That is required to ensure that an additional link delay does not nullify the ideal synchronization of the TX_MESO module output, i.e. the flow control signal. Moreover, the links were constrained in order to have the same delay regardless of their crossing direction (i.e. D1 = D2). As a result, the skew test experiments can be performed with an additional degree of freedom regardless of the data crossing direction. Finally, the data crossing the source synchronous links was routed together with the strobe signal to match their propagation delay (bundled routing) with industry-available physical synthesis techniques.

As regards the source synchronous communication in the loosely coupled sub-systems ("Asynch_Loose_Slow" and "Asynch_Loose_Fast"), also the RX_MESO0 and RX_MESO1 modules together with the TX_MESO0 and TX_MESO1 modules were placed respectively close to the SWITCH0 and SWITCH1 modules. As mentioned before, that is required to ensure the ideal synchronization of the data/flow control of the RXMESO/TXMESO output. Similarly to the hybrid coupled communication, the information on the link are bundled routed and the D1 crossing delay corresponds to the D2 crossing delay.

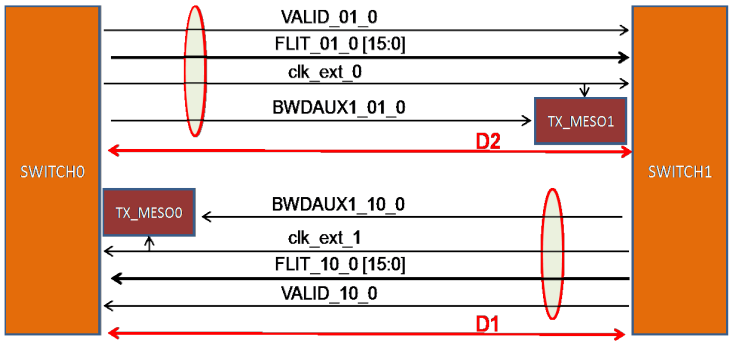


Figure 2.7: Source synchronous communication in the hybrid coupled sub-systems and PnR constraints.

2.3.4 GALS Testchip Results

Area

This section compares the seven sub-systems under an area point of view. Every sub-system has been synthesized independently by means of the 40nm standard cells Infineon Technology and treated as a soft-macro. In particular, the slow sub-system ("Asynch_Hybrid_Slow", "Asynch_Loose_Slow" and "Synch_Slow") were synthesized at 100Mhz while the fast sub-systems ("Asynch_Hybrid_Fast", "Asynch_Loose_Fast", "Asynch_FIFO" and "Synch_Fast") at 500Mhz. As mentioned before, the key idea of the testchip was to instantiate a number of small and replicated sub-systems each validating a different feature of the new synchronization technology. Basically, all the seven sub-systems integrate the same modules (2 memories, 2 testers, 2 switches and 1/2 JTAG) but the communication between these modules is enabled by different source synchronous interfaces. As a result, the source synchronous interfaces provide the main contribution in terms of area overhead with respect to the fully synchronous baseline sub-system ("Synch_Slow" and "Synch_Fast"). See Figure 2.8 on the next page for area results.

In particular, the fast solutions present a similar area footprint with respect to the slow counterpart. In fact, although the fast sub-systems were synthesized at a higher frequency than the slow sub-systems, both the solutions meet the target frequency constraint with a large slack and as a result the synthesis tool was able to provide a well optimized gate-level netlist in terms of area footprint in both the cases. Note that the additional "synch" modules integrated in the fast sub-systems provided a negligible area overhead. Interestingly, since the goal of the testchip was to evaluate

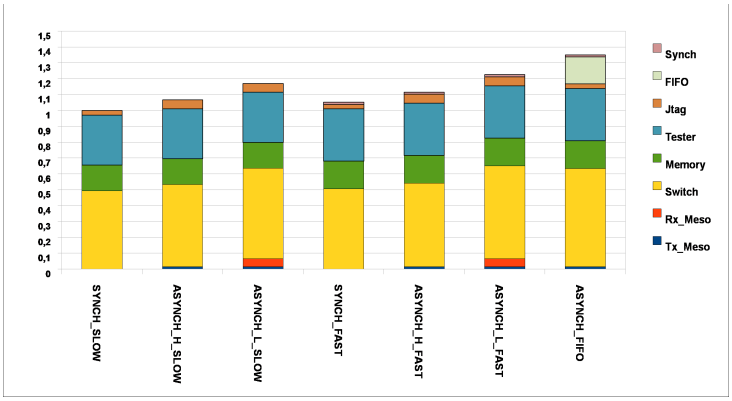


Figure 2.8: Area breakdown of the seven sub-systems.

the NoC technology, the instantiated memories and testers are elementary and as a result the switches required more than the 50% of the total area footprint of each sub-system. The area results were as follow:

- The "Asynch_Hybrid_Slow" sub-system required the 7% of additional area than the fully synchronous baseline sub-system ("Synch_Slow"). In particular, this area overhead is due to the additional JTAG module required to configure the second frequency domain (3%), the 2 Tx_Meso flow control synchronizers (1.5%) and the 2 switches integrating the data mesochronous synchronizer (2.5%).

- The "Asynch_Loose_Slow" sub-system required the 17% of additional area than the fully synchronous baseline sub-system ("Synch_Slow"). In particular, this area overhead is due to the additional Jtag module (3%), the 2 Tx_Meso flow control synchronizers (1.5%), the 2 Rx_Meso data synchronizers (5%) and the area overhead of the 2 switches (5.5%). Note that the input buffers of the two switches were extended by two slots since 2 clock cycles of latency are added in the round-trip from the transmitter and receiver synchronizers and vice versa on the link. Interestingly, the "Asynch_Loose_Slow" sub-system had a 10% of area overhead with respect to the "Asynch_Hybrid_Slow" sub-system.

- The "Synch_Fast" sub-system required 4% of additional area than the "Synch_Slow" sub-system mainly due to the higher synthesis frequency. The "Asynch_Hybrid_Fast" sub-system and the "Asynch_Loose_Fast" sub-system followed the same area trend. In fact, they feature respectively 7% and 17% of area overhead with respect to the "Synch_Fast" sub-system.

- The "Asynch_FIFO" sub-system required 30% of additional area than the fully synchronous fast sub-system ("Synch_Fast"). This area overhead is mainly due to the 4 loosely coupled dual-clock FIFOs (17%) and the 4 tightly coupled dual-clock FIFOs (9%). We should note that all the dual-clock FIFOs were with 5 buffer slots and, the tightly coupled dual-clock FIFOs have been merged with the switch input buffer. Then, since the xPipes switches are natively composed of 2 input buffer slots, the total amount of buffering resources is increased. In order to keep the original buffering resources and to reduce the final source-synchronous switch area overhead the output buffer size could be reduced from 6 to 3 slots, since additional buffer slots come with the synchronization-augmented input buffer.

As a final consideration, the global area footprint of the NoC testchip was composed by two additional main contributions: the PLL module, and the pin pads. In particular, almost half of the total required area was devoted to the PLL instantiation.

Yield

The "functional test" was executed as a second test to evaluate the yield and it was performed in two steps. In the first step, the slow sub-systems were analyzed ("Asynch_Hybrid_Slow", "Asynch_Loose_Slow" and "Synch_Slow"). Since the slow sub-systems are clocked by an external low-speed clock, the PLL was no longer required and it was switched-off. The goal of this test was to verify the functionality of all sub-systems at different frequencies and skew scenarios. The control input signal and the data input signal were injected in compliance with the target operating frequencies. The strobe setting at the output pins was performed similarly. The selected lower bound for the frequency sweep was 25 MhZ. Clearly, the test could also be performed below this frequency but we did not expect relevant results under this threshold. On the contrary, the upper bound of the frequency sweep was not specified a priori since the test was also meant to determine the maximum operating frequency that the slow sub-systems were able to achieve. This maximum speed is associated not only with the inherent limitations of the test setup, but also with the limited driving capability of the I/O pins. We should note that the slow sub-systems were synthesized at 100 MhZ although the synthesis tool met the frequency constraints with a large slack. Additionally, a skew sweep was performed for the "Asynch_Hybrid_Slow" and "Asynch_Loose_Slow" at every operating frequency under test. In this case, the domain1's

clock phase was gradually increased until 100% of the clock period was achieved. In order to perform a safe test, the skew was not only applied to the domain1 external clock but it was also applied to the control/data input signal and to the output pin strobe of the same domain.

The "Synch_Slow" sub-system passed every functional test in the range of 25Mhz and 200Mhz. Interestingly the sub-system was able to work at a frequency significantly higher than the frequency of synthesis. That can be due to two main reasons: (i) a significant slack was reported after the 100Mhz synthesis. (ii) designs were instantiated using the worst case standard cell libraries. The worst case library was probably assuming overly pessimistic conditions.

In the second part of the "functional test", the fast sub-systems were analyzed ("Asynch_Hybrid_Fast", "Asynch_Loose_Fast", "Asynch_FIFO" and "Synch_Fast"). The goal of this second testing part was still to verify the functionalities of the sub-systems (a sub-system at a time). Since in this case the sub-systems were clocked by the PLL, the frequency and the skew sweep should be performed according with the limitations dictated by the PLL specification. In particular, the PLL provides the following 4 clock phases: 0, 90, 180 and 270 (i.e. the highest provided skew corresponds with the 75% of the clock cycle). Unluckily, the frequency and the skew sweep in the fast sub-systems could not exploit the same timing set and vector data file since the frequency and the clock phase had to be set statically by driving the dedicated external pins. On the contrary, the clock phase and the clock frequency in the slow sub-systems were modified by means of the same EVCDs through straightforward steps. In fact, the clock phase in the slow sub-systems was simply set by delaying the injection of the clock and control/data input signals while the clock frequency was modified by scaling the timing of the vector data files. Then, a few test frequencies were selected to perform the functional test of the fast sub-systems. The selected PLL frequencies for test were 200Mhz and 400Mhz in the 4 available clock phase variants. It is useful to recall that the input pins were supposed to inject the data/control signals with a frequency of 25Mhz (going through the synchronizer) and the fast sub-systems were synthesized at 500Mhz.

The results of testing are as follows:

- The "Synch_Fast" functional test passed with the PLL frequency set at 200Mhz and 400Mhz with all the chips that previously were able to pass the continuity test. Note that only 1 chip out of 19 chips was discarded during the continuity test.

- The "Asynch_Loose_Fast" functional test passed with the PLL fre-

quency set at 200Mhz in each of the 4 skew scenarios for the 18 chips. The same sub-system operating at a frequency of 400Mhz passed the test with 17 chips. The sub-system was not able to work at 400Mhz in only 1 chip previously able to pass the continuity test.

- The "Asynch_Hybrid_Fast" functional test passed with the PLL frequency set at 200Mhz in each of the 4 skew scenarios for the 18 chips. The same sub-system operating at a frequency of 400Mhz passed the test with 12 chips. The sub-system was not able to work at 400Mhz in 6 chips previously able to pass the continuity test. Anyway, each of these 6 chips passed the functional test with a phase offset of 90, 180 and 270, on the contrary, they did not pass the test with 0 of skew. The 0 degree of skew represents the fully synchronous scenario and intuitively the source synchronous interfaces should provide the best timing margins in this condition.

- The "Asynch_FIFO" functional test passed with the PLL frequency set at 200Mhz in each of the 4 skew scenarios for the 18 chips. To note that this sub-system should not only absorb the skew into the source synchronous link but also synchronize data from 25Mhz to 200Mhz and vice versa. The same sub-system operating at a frequency of 400Mhz passed the test with only 11 chips out of 18. The output of the remaining 7 chips was not stable. In fact, although the sampled output was in most of the cases matching the expected one, the correct output was still not guaranteed when repeating the test several times. Moreover, the failing clock phase was usually only one at a time and, surprisingly, it was randomly changing although the test was performed on the same chip. Interestingly, the failing tests presented few unstable output bits always located in the same position of the output vector. On the contrary, when the "Asynch_Hybrid_Fast"' and the "Asynch_Loose_Fast" sub-systems were failing the output vectors presented the anomalous bits spread homogeneously all over the output vector. As a result, these information led us to suppose that specific patterns bring few bits of the "Asynch_FIFO" sub-system to a metastability condition that results into unpredictable results. At a better analysis, the "Asynch_FIFO" sub-system is the only one that integrates brute force synchronizers, composed of a sequence of 2 flip-flops, where the output bit can fall into a metastability condition. The probability of solving the metastability in these bits follows the MTTF law and tightly depends on the technology node and the operative frequencies. We

should note that the brute force synchronizers are stimulated only during the congestion conditions of the switches (i.e. when assertion of the STALL/GO flow control is required). The congestion of the switches is achieved in only two of the six transactions and it always affects the switches during the same clock cycles (in compliance with the EVCD patterns).

Summing up, Figure 2.9 illustrates on the y-axis the percentage of chips (out of the 18 ones which passed the continuity test) that proves functional behavior in each test case (identified by the operating frequency, skew and architecture variant parameters). The success percentage, for a first time implementation, is quite high. We start observing some issues at 400 MHz, as we approach the target synthesis frequency of 500 MHz. However, the failure pattern is such that improving upon such results should not be an issue in future work. In fact, the zero skew failure for the "Asynch_hybrid_fast" design does not seem to be related to the robustness of the interface. Also, failures at 400 MHz for "Asynch_loose_fast" design are independent from the applied skew, thus denoting a different problem than the robustness of the mesochronous synchronizers. Finally, the failures for the "Asynch_FIFO" design could be easily solved by cascading more flip flops in the brute-force synchronizers used inside dual-clock FIFOs.

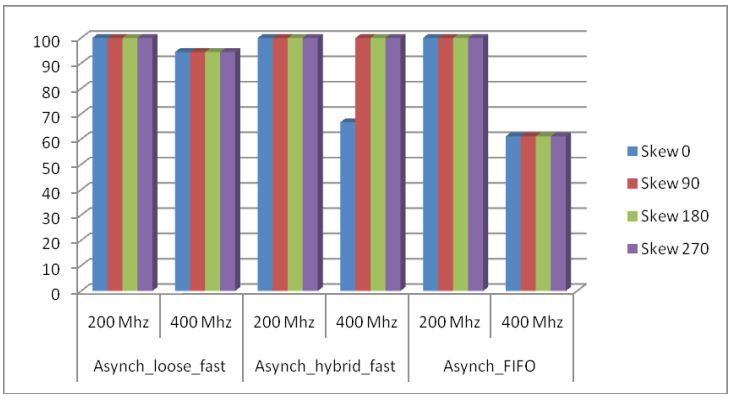


Figure 2.9: Percentage of working chips in each test case. (Yield)

Power

We then performed power measurements on the test structures. The IDDQ test outcome reflected the leakage power of the entire testchip, including baseband processor section. The calculated leakage was around 2

mW. Figure 2.10 on the next page shows the relative comparison between power of the designs under test clocked by the PLL. Taking a look at the plot, it is clear that it was not possible to inject significant traffic due to the constraints of the measurement setup. In fact, in order to achieve precise dynamic current measurements, it is essential that the pattern has a constant current profile and that it can be repeated in a loop to make sure the pattern is running as long as the current measurement is active (and there is no way of predicting how long the measurement will be. The tool uses auto-ranging, i.e., it executes several measurements to find the minimum measurement range where no overflow occurs.) Unfortunately, the NoC structures under test consist of a quite time consuming programming step of a much shorter execution phase, therefore most likely the programming step is the prevailing contribution to measured power figures. Also, the reader should recall that the power values indicated by the bar heights of the histogram include the leakage power of the baseband processor. Idle power is taken by selectively removing the reset but without giving any input traffic to the platform itself. Clearly, the power difference between the architectures under test is not significant at all: the synchronizer-based solutions tend to add 0.38 mW overhead to the power consumption of the synchronous sub-system (slightly more for the loosely coupled solution), which grows to 0.5 mW for the "Asynch_FIFO" design. Conservatively assuming that the entire IDDQ current is absorbed by the baseband processor, then the overhead in percentage terms would be 2.8% and 3.7% with respect to the synchronous sub-system for the mesochronous and for the dual-clock FIFO-based ones, respectively. Ultimately, we feel the key take away of these results is that synchronizer-based GALS technology comes at a marginal power overhead.

The above power results indicate just a trend and should be considered very carefully for the following reasons:

(i) The error margin of the test equipment is about 0.3mA. The current absorbed by the NoC structures differs by an amount which is close to the uncertainty of the measurement tool. In addition, a single measurement of the current was taken at regime. However, by sampling the current multiple times we were getting a further uncertainty of around 0.5mW. In spite of all this, we were able to observe a clear relative trend between the architectures under test which is orthogonal to the 18 working chips. Also measurements for slow designs confirm the same power trends.

(ii) Injecting heavy traffic into the test structures was virtually impossible, given the small size of the structures themselves and the large time needed to program the structures, prevailing with respect to the actual packet exchange time. For this reason, the most accurate power results

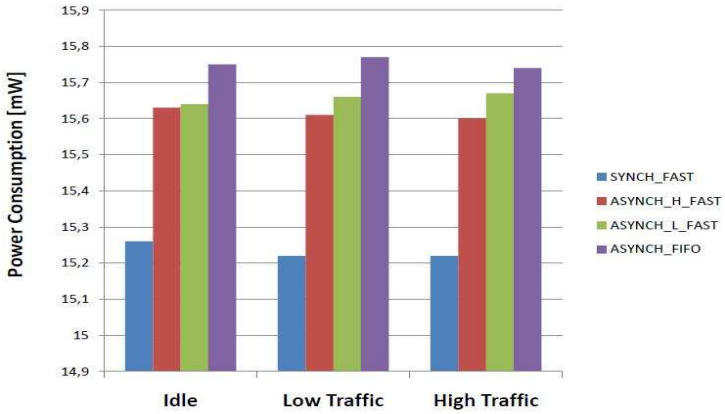


Figure 2.10: Relative power comparison.

certainly concern the idle power tests.

(iii) The "Asynch_FIFO" design has memories, tester and JTAG interfaces working at 25 MHz instead of 200 MHz like in the other fast platforms. However, this does not seem to favor this design much, from a power viewpoint.

2.4 Summary

GALS NoC as an effective and scalable interconnection network for MPSoCs was presented. This network can be used for inter-cluster communications in the cluster-based Multi-core systems. Moreover, we proposed a test-chip, named Moonrake, fabricated on 40nm process technology where seven different synchronization approaches are implemented to evaluate the effectiveness of various GALS methodologies.

The Moonrake chip is the first implementation of synchronizer-based GALS NoC technology in 40nm CMOS process. The objective was to see the issues at deep submicron including variations and to validate the synchronization technology, containing key innovations such as the tight coupling of the synchronizer with the switch input buffer, and, implicitly the design technology supporting it. The NoC section of the testchip was structured in a modular way, thus amortizing the manyfold risks associated with testchip fabrication: in essence, porting a new synchronization technology to a new manufacturing process, and then using new features of the test equipment to test such an advanced chip implemen-

tation. Overall, results are very promising. NoC test structures getting the clock from the external world provided an excellent result: frequencies from 25 to 256 MHz were swept, while at the same time varying the clock phase offset from 0 to 360 degrees. This means that the synchronization mechanisms, considered by themselves, can be ported to the 40nm technology and prove functional in such an environment. When it comes to the designs synthesized for and operating at a higher frequency, then a set of issues comes into play: not only higher stress of the technology platform, but also larger complexity of the test procedure. 91% of the performed tests completed successfully, and the failures seem more related to testing, variability and "systematic" issues in the technology platform rather than to the robustness of the synchronization interfaces. Finally, our power measurements indicate that the power overhead when moving from a fully synchronous test system to a mesochronous one or even to a multi-synchronous one is marginal.

The GALS testchip and the yield results presented in this chapter motivate the need for techniques to detect and work around manufacturing faults and process variations in interconnection networks. In the next chapters we are going to propose different approaches at physical and architecture levels to target these issues. For physical design, we propose the bundle routing framework as an effective way to route the NoCs global links. For architecture-level design, two cases are addressed: (i) Intra-cluster communication where we propose a low-latency interconnect with variability robustness (ii) Inter-cluster communication where a testing in tandem with a reliable NoC configuration are proposed. We also propose dualVdd as an orthogonal way of compensating variability at the post-fabrication stage. This is an alternative strategy with respect to the design techniques and enforces post silicon detection and compensation.

In the next chapter, we describe how to handle the physical routing of the global wires in a NoC to decrease noise and cross coupling effects and to have a reliable and variation-tolerant source synchronization in NoCs.

CHAPTER 3

Robust link physical routing in GALS NoCs

This chapter[1] introduces a link routing approach which considers all nets of a link together and efficiently routes NoCs links as an atomic entity (a bundle). This routing step is performed early in the design implementation flow, at the global routing step, and it integrates seamlessly with the design implementation tools to ensure that link quality will be fully preserved during detailed routing. As a result, the variability in link net attributes such as length, resistance, load, and delay is much reduced. Moreover, enforcing the same "trajectory" on all nets of a link facilitates wire spacing and shielding. Finally, bundled routing improves the routability of the NoCs link, as sometimes even a powerful place and route tool struggles to perform detailed routing of a large number of links. This can significantly reduce routing run time as well as the time for parasitics extraction.

3.1 Motivation and Key Challenges

Interconnects have moved to the forefront as the limiting factor in IC performance. Nowadays, there are cases where interconnect delay, especially that of global interconnects, accounts for more than 75 percent of the total path delay [55]. The most important global signals in a SoC are inter-

[1]The author would like to acknowledge contributions by Dr. Igor Loi, Antonio Pullini, Dr. Federico Angiolini, and Prof. Luca Benini.

core communication links, traditionally implemented as buses. However, buses are inadequate for medium to high complexity SoCs, and they have increasingly replaced by Networks-on-Chip [46, 60]. NoC links provide the global physical connectivity between switches, and they differ substantially from buses. First the number of buses in a design is relatively small, while there are many more links in an NoC. Moreover, most of the buses in a traditional SoC have a large set of fanouts while NoC links are point-to-point. Furthermore, advanced NoC designs links make increasingly use advanced synchronization and signaling schemes, to improve IP decoupling and energy efficiency [54, 58].

In this chapter we focus on physical routing of NoC links, which often span a significant on-die distance. At a first glance, link physical routing can be carried out with a standard physical implementation flow, which routes each net independently, while trying to match design constraints on delay, area and power. However, routing algorithms are extremely complex and influenced by a number of factors (such as wire congestion). Hence, in practice it is extremely difficult to ensure that wires which take widely different routes maintain closely-matched delay and power consumption. In other words, when the logical atomicity of a link is broken at the physical level, we experience significant length, resistance and load deviations among different wires of a link. Reducing delay and power variation between wires of a link is a key requirement for advanced NoC implementation styles. First and foremost, as mentioned in the previous chapter robust Mesochronous [18] and GALS NoCs [58, 59, 18, 64] rely on source-synchronous signaling, where the data and strobe signals have to arrive at the destination within a narrow timing window to achieve correct and high performance operation. The maximum phase mismatch between strobe and data signals for safe input data sampling is a small fraction of the clock cycle, (1/10 or less); otherwise, the receiver latch enable signal might be activated too late or too early, with disastrous consequences on probability of metastability and error rates.

In practice, precise delay matching between link wires can be achieved in regular (tile-based) NoC designs with manually planned bundled link routes. However, manual link routing is difficult or even impractical in SoCs with irregular tiles and topologies, where the NoC is automatically instantiated and then handled at the RTL to the design implementation flow. Unfortunately, even by using the most recent place&route (PnR) tools, there is always an amount of variation between the propagation delay of data, flow-control and timing signals which increase the phase mismatch between data and strobe signals. Figure 3.1 on the facing page shows a part of an NoC layout routed in the most recent version of Synop-

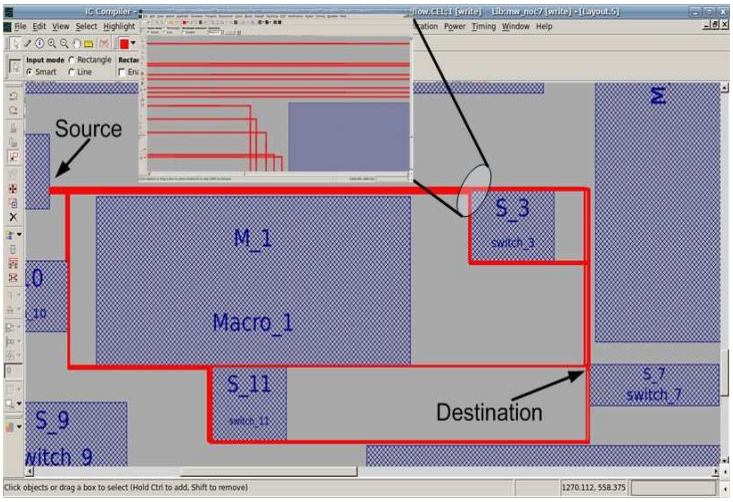


Figure 3.1: Link route obtained with state-of-the-art PnR.

sys IC Compiler [47]. Link wires have different routing trajectories, which leads to different lengths and delays. Although in some cases the nets have the same lengths, their delays are different due to vias. Note that this layout has been obtained with the automatic bus routing feature of the tool enabled. Even if it may theoretically be possible to achieve perfect delay matching for irregularly routed links, this would imply numerous design convergence iterations (which take days for an average size SoC).

Moreover, high-performance NoC implementation often relies on low-swing signaling to boost energy efficiency [62, 63, 66]. Signal-to-noise ratio is a serious concern in both low-swing and full-swing communication [65, 66]. Shielding and wire pacing techniques are commonly used to control the crosstalk effects in global interconnects [67]. Managing the routing path of the wires makes it easier and more efficient to apply these techniques on the global interconnects. These goals can be achieved by having tight control on the link trajectory and keeping all wires of a link together. This is helpful not only to have matched delays, but also to manage and reduce, through link shielding and spacing, crosstalk noise from bus wires as well as for other interconnects. We will describe this in more details in Section 3.4 on page 57 of this chapter.

3.2

Related Work

Physical routing has been researched in depth [49, 50, 51]. Most of the works proposed in the literature are focused on techniques to route single nets. However, bus routing has also been extensively investigated. Existing bus routing/planning methods are mostly used in floorplanning [42, 43, 44], where buses are limited to simple topologies like one-bend and T-shaped routes. However, all these algorithms leave the problem of organizing bits in the bus to the detailed routing and do not propose any methodology for that. Some works have investigated techniques for enhancing the regularity of bus routing: for instance, Persky and Tran [45] proposed a bus routing algorithm that identified the topological commonality of different nets of the same bus and tried to re-use a common routing topology. In this area, our approach is most closely related to the semi-detailed bus routing technique recently proposed [41] by F. Mo and R. K Brayton. Their approach is similar to ours for what concerns the algorithm used to ensure net length matching, however, their router is tuned on multi fan-out buses, but not on point-to-point links. Moreover, they do not consider pin placement and ordering and, they use a simplistic router to route the nets, while we take advantage of a state-of-the-art timing-driven router. Furthermore, they do not provide any data on signal integrity and do not propose any approach to control the crosstalk effect in bundled routing.

In global routing electrical buffers are extensively used to reduce delay [62]. However, in NoC design, latency-insensitive techniques [56, 57] are commonly used to meet timing constraints on long links. In latency-insensitive design, relay-stations perform functional pipelining and electrical re-buffering [52, 53] and are inserted at the RT level of abstraction. Therefore, at the physical level, NoC link wires are routed un-buffered between two switches (if a link is short) or between relay-stations. Our focus is solely on the routing of these un-buffered link segments: interaction between physical routing and relay-station planning is an interesting area for future research.

3.3 NoCs link routing flow

Physical routing is divided into two main steps: global routing and detailed routing. In the global routing phase the path of any net is deter-

mined as the Manhattan distance of that net without considering the routing blockages and constraints. In a traditional flow, detailed routing considers nets independently, and after this phase each net of a link in NoC has different trajectory leading to high deviation in resistance, capacitance, length and delay among bits of a link. Note that, a link is a group of logically related interconnects. As outlined before this chapter describes a methodology to manage the routing of NoC links in which we consider all bits of each link together. Figure 3.2 shows our routing methodology, consisting of four main steps: (i) pin placement and ordering (ii) prioritizing links (iii) creating virtual routing blockages and routing one net of each link as the link agent and (iv) applying the agent's trajectory to other nets of the current link.

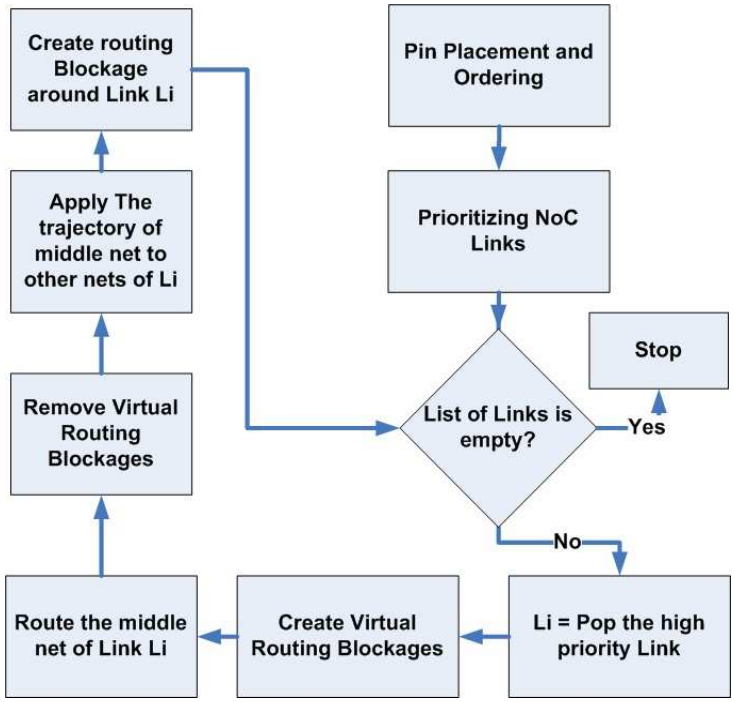


Figure 3.2: Link routing flow

3.3.1 Pin placement and ordering

Before routing, pin placement and ordering are performed. In this phase for each link we perform two steps: (i) determining the side of pins around the bocks, and (ii) specifying the pins order (named pin placement and pin ordering, respectively). We consider all components in the floorplan as

rectangular objects. Therefore, there are four possible sides (North-East-West-South) for the pin positions of each component, and, therefore, for each link we have 16 possible options for the pin placement (each link is connected to two blocks).

In the pin placement step, for each link we have four possible sides for both source and destination blocks. We empirically realized that to have the same length for all nets of a link it is better to place the input and output pins of a link on different sides. For example, if we place the output pins of a link on the East side of source block, we do not consider East side of destination block for the input pins; therefore, for each link we have 12 different solutions to place all input and output pins. Each solution is a pair of sides; for example Pair (West, East) shows that the output pins are placed on the West side of the source block and the input pins are place on the East side of the destination block. We start with those links whose source and destination blocks are farther from each other. From the 12 possible solutions for each link, we select a feasible case that leads to shorter Manhattan distance for each link. A solution is feasible if there are enough spaces on both sides of the pair to place all pins of the related link. However, if all pins of a link cannot be placed on a side because of already placed pins of other links or because of lacking space, the solution is not feasible and we have to select the next pair.

Table 3.1: Pin ordering for 6 pin placements

Pin placement	*(North,West)*	*(North,South)*	*(North,East)*
Pin order	*(LR,DU)*	*(LR,LR)*	*(LR,UD)*
Pin placement	*(West,North)*	*(West,South)*	*(West,East)*
Pin order	*(DU,LR)*	*(UD,LR)*	*(UD,UD)*

To have the same length for all nets of a link, we order pins after pin placement. In this step, we assign an order to pins of one side of each link and update the pins order of the other side based on that; order of pins can be from LSB to MSB or wise versa. To make the problem easier, for each pair of 12 possible placement options we define a pair of orders. Each item of the order pair shows the direction of LSB to MSB. For example the order Pair (LR, UD) means that the output pins at the source block should be placed in order of LSB to MSB from left (L) to right (R), and the input pins at the destination block should be placed in order of LSB to MSB from up (U) to down (D). Note that block position and routing trajectory of a link do not change the pin orders, and the only factor which affects the order of pins is the sides where pins are placed on source and destination blocks.

Therefore, we have a fixed table to determine the pin orders for each of 12 pin placement pairs. Table 3.1 on the preceding page shows pin orders of 6 different pin placement pairs.

3.3.2 Prioritizing links

After setting the pin positions and orderings for all links, we give a priority to each link to route the most critical links first. Priority is based on the Manhattan distance of each link, to increase the chance to meet timing constraints on more critical links which have to span a longer distance. In fact, the router will be faced with lower congestion when routing the high-priority long links. When routing resources become tight, routing short links that need less routing resources is much easier than routing long links.

3.3.3 Link routing

The basic idea is to first route one net of a link (called link agent), and then route the other nets along the same trajectory [41]. Our algorithm entails the following steps: (i) extending the routing blockages to force the router to find a path where the whole link can be routed (ii) routing the middle net of the link (iii) applying the trajectory of the middle net to the other nets (iv) creating routing blockages around the routed link. In the first preparation step, overlapping with routing blockages is prevented for the current link by extending existing routing blockages on all sides by $W/2$ where W is the width of the link. As a result, we create virtual routing blockages and force the router to route the link agent at a distance of $W/2$ from the existing actual routing blockages. These virtual routing blockages will be removed after routing the middle net leading to a narrow channel where the entire link can be routed.

After the preparation step, we consider the middle net of the current link as the link agent, and ask the router to route that net. To do so, differently from [41] we take advantage of a powerful timing-driven commercial router [47]. We set timing as the objective of the router. If the routing succeeded to route the middle net it creates a routing trajectory and a channel for the link where the entire link can be routed. However, if the router fails, the nets of the link can not be routed together with the same trajectory with the current constraints and floorplan which means the link is not routable. In this situation we add the current link to the unroutable links list, report a warning message and will move to route the next link

in the prioritized list. We will route all unroutable links using the normal flow of the tool as a fall-back option.

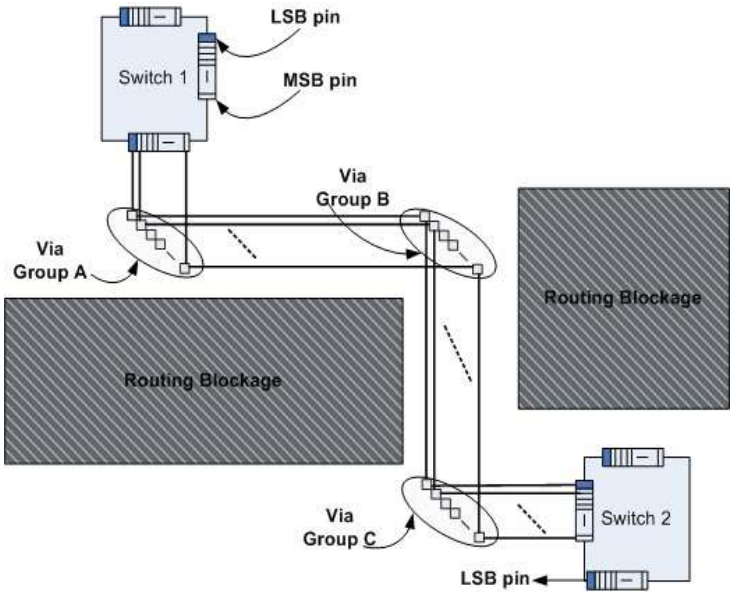


Figure 3.3: Link wires with the same trajectory and length.

After routing the middle net, we first remove the virtual routing blockages, and then apply the trajectory of the middle net to other nets of the link. To do this, we use a two-step algorithm. In the first step we replicate the vias of the middle net for all other nets at the appropriate position determined by the pin placement and ordering of the related link. In the second step, for each net we connect the source pin to the first via and then connect the vias together sequentially, and finally connect the last via to the destination pin. Note that if there is no via for the middle net, it means the middle net is fully horizontal or vertical, and in this case we simply replicate the middle net trajectory for other nets. For each net of the link, via positions are computed in such a way that the via-to-via length of all nets of a link is equal. Because of suitable pin ordering that we considered in the first step, the length difference between different nets of a link from source pin to the first via will be compensated by length difference from the last via to the destination pin. Figure 3.3 shows an example. As it can be seen in Figure 3.3, all net segments from via group A to via group B have the same lengths, this is also true for all segments from via Group B to via Group C. Note that, since the source pins are aligned, the segments from source pins to via Group A have different lengths, but this difference is compensated by length difference from via Group C to destination pins.

Therefore, all nets of the link have the same lengths.

To replicate the vias of the middle net for the other nets of the link, we have to calculate the positions of vias for each net. To do so, we have two functions F and G to calculate horizontal and vertical positions of each via, respectively. F and G are functions of middle-net's via position, wire to wire distance, and pin placement pair of the related link; they are defined for all 12 possible options of pin placement and ordering. To clarify the algorithm for routing the middle net and applying the trajectory of middle net to other nets, we show it step-by-step on an example shown in Figure 3.4.

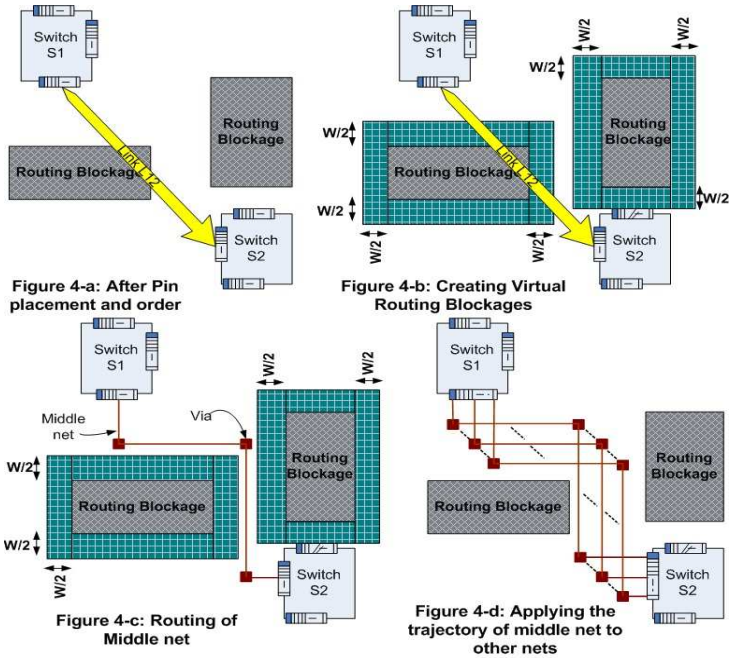


Figure 3.4: Link bundle routing steps.

Assume a part of an NoC layout shown in Figure 3.4-a in which there are some existing routing blockages. We have already performed the pin placement and ordering phase on this layout, and we are going to route link L12 from switch S1 to switch S2. First we extend the existing routing blockages as W/2 where W is the width of Link L12 which leads to the layout shown in Figure 3.4-b. Then we ask the router to route the middle net of the link, the middle net trajectory is shown in Figure 3.4-c. As it can be seen in Figure 3.4-c the middle net has 3 vias. Assume that the space between two wires for Link L12 is P, the wire width is H, and the link has M bits. To replicate the vias for other nets of Link L12 we use the formula:

$X_{ji} = F(X^i_{M/2}, j, P, H)$, where X_{ji} is the horizontal position of i-th via for the j-th bit of the related link. For Link L12 we have:

$F = X^i_{M/2} + (j - M/2) * (P + H) | j \in \{1, ..., M\}, i \in \{1, 2, 3\}$

$Y_{ji} = G(Y^i_{M/2}, j, P, H)$, where Y_{ji} is the vertical position of i-th via for the j-th bit of the related link. For Link L12 we have:

$G = Y^i_{M/2} + (M/2 - j) * (P + H) | j \in \{1, ..., M\}, i \in \{1, 2, 3\}$

After replicating all vias for all nets, we simply connect the vias of each net together and connect the source pin to the first via and the destination pin to the last via. Figure 3.4 on the previous page-d shows the routed link. We define a set of F and G functions for all 12 possible pin placements and orders. The distance between any two subsequent vias for all nets on a link is constant and does not depend on the bit number, thus the only factor that can change the wire length for different nets of a link is the distance from Source to the first Via (SV) and the distance from the last Via to the Destination (VD). By specifying the appropriate pin orders, and suitable F and G, the sum of SV and VD would be constant for all nets of a link and does not depend on the bit number, thus, leading to the same length and trajectory for all nets of that link.

Figure 3.5 shows the same layout of Figure 3.1 on page 49 which is routed by our algorithm. As can be seen in this figure all wires of each link are routed together with the same trajectory (outlined in red).

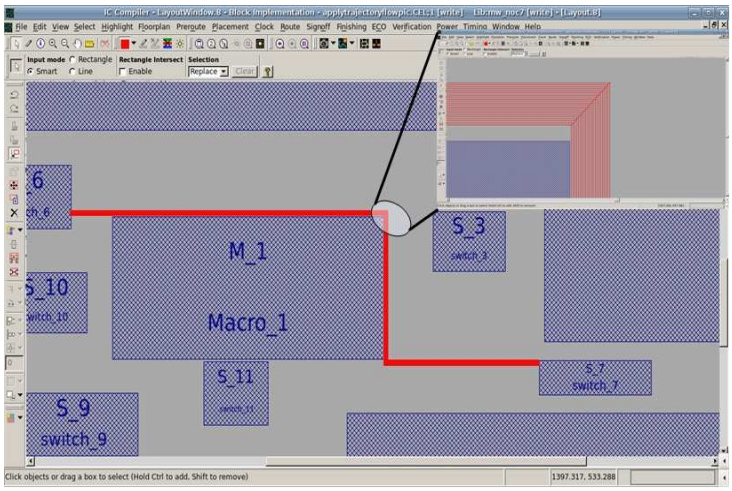


Figure 3.5: Bundled link routed layout example.

3.4

Crosstalk handling

There are two different types of crosstalk affecting bits of a link: the crosstalk between nets of a link and other nets of the design (inter-link crosstalk) and the crosstalk among different nets of a link, themselves (intra-link crosstalk). We have solutions to control both of them.

To handle the first case, we create a routing blockage around each link. As we know the trajectory of the entire link, we can easily create a routing blockage with any size around the routed link. Modifying this blockage's size avoids aggressor signals or other links being routed too close to the current link. For the intra-link crosstalk, we use two techniques to decrease the crosstalk between nets of a link: spacing and shielding. In the spacing method we tune the space between different nets of a link to reduce the crosstalk delay. We start with the minimum spacing and check the crosstalk delay. If it was satisfying we keep the spacing; otherwise we gradually increase the spacing to reduce the crosstalk effect. We modify the virtual routing blockages and possibly change the trajectory of the link to apply spacing.

In the shielding technique, we use ground or power nets as a shield for the nets of a link. This converts the coupling effect between two nets to coupling effects between nets and gnd/vdd and subsequently removes the crosstalk effect. As we route all nets of a link together, except the first and last nets we can use one shielding net for two nets of a link. Therefore, in our routing methodology, for a link with M number of bits we need M+1 shielding nets to remove the x-talk effect between different nets of that link. Also, one alternative solution is to mix shielding and spacing techniques to handle both inter and intra link crosstalk. This can be carried out by spacing wires in the link and decrease inter-link crosstalk with two shielding wires at the boundaries of the link to handle inter-link cross-coupling. Our tool flow supports all these options.

3.5 Experimental results

To evaluate our methodology, we applied it on a set of benchmarks which contain several macro blocks and different number of switches and links. Table 3.2 on the following page shows their characteristics. For each case, we performed bottom-up synthesis and placement and routing of the blocks. We used Synopsys IC Compiler [47] and Design Compiler for PnR

and logic synthesis, respectively. We mapped all designs on CMOS 45nm technology. For pin placement and ordering phase, we used IC compile commands to place pins around blocks. To implement our routing algorithm, we took advantage of routing guides and the proper script commands to create nets and vias in IC Compiler. Our algorithm and flow is implemented in TCL [48] using IC Compiler scripting facility.

Table 3.2: Pin ordering for 6 pin placements

Design	# of Macros	# of Switches	# of links
D1	*6*	*10*	*10*
D2	*7*	*12*	*8*
D3	*8*	*13*	*10*
D4	*21*	*18*	*15*
D5	*31*	*20*	*18*

We compared the original flow of the tool with our link routing technique. Note that the tool has a feature for manually drawing wires together; however we compared our algorithm with its alternative automatic flow. In the original automatic flow of the tool we used the bus routing feature of the tool which is a tools feature for automatic bus routing and compared the results of this flow with those of obtained by our algorithm. Results are compared in terms of wire length, resistance, load, and delay variations. We also compared the actual net delay of these two ways, to ensure that delay matching is not obtained at the price of a significant increase in average delay.

Figure 3.6 on the next page compares the wire length variation of different links in D3. As it can be seen in Figure 3.6 on the facing page, our routing methodology has lower length variations than the original flow. Almost in all cases the variability of wire length in a link for our method is zero. In some rare cases it is not zero, as our pin placement step could not find enough space around the related block and had to place pins on the same side of source and destination which leads to different length. However, this deviation is very small and comes from the length difference at the first and last vias and is at-most 2*W where W is the link width.

Figure 3.7 on the next page compares the number of vias per each link in D1. Generally, the number of vias in our routing is smaller than that of the normal flow. This is another factor which decreases variability and even improves delay.

Figure 3.8 on page 60 and Figure 3.9 on page 60 compare the wire resistance and delay variations respectively for D3. As seen, the delay variation

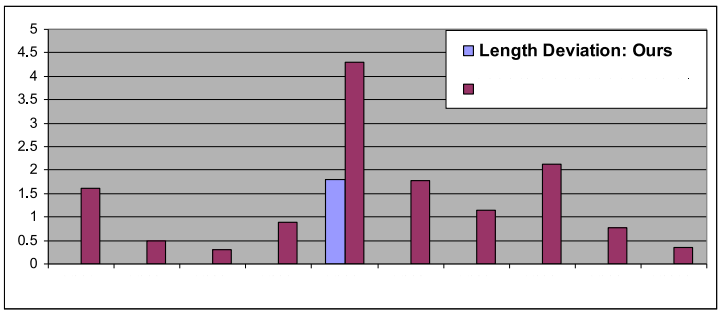


Figure 3.6: Wire length variation.

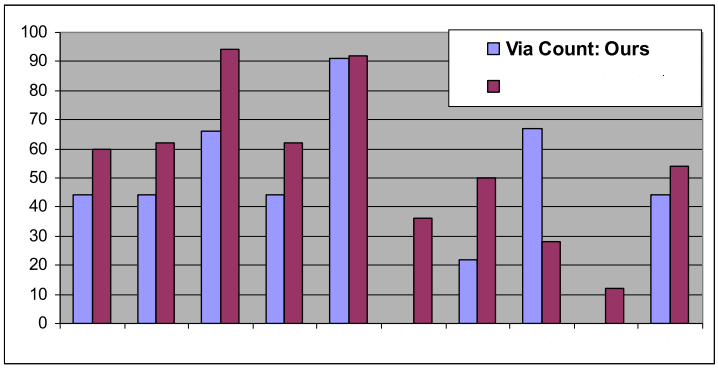


Figure 3.7: Number of vias.

between wires of links is decreased by an amount in the range of 30-70% in our routing methodology compared to that of the normal flow. Note that, the wire length deviation of, for example, Link1 in our flow is zero, but the delay and resistance deviation are non-zero. These deviations in delay and resistance are not related to the wire length, but are due to detailed deep-submicron manufacturing (DSM) effects that commercial tools consider to calculate the net characteristics. Synopsys IC compiler models the etching effects which will result in trapezoidal wire cross-sections for wires close to each other. This will effectively change the R and sidewall C, and therefore alter the wire delay.

Figure 3.10 on the following page,Figure 3.11 on page 61 and Figure 3.12 on page 61 compare the maximum and average of link delay deviation, link load deviation and the worst case delay for all designs, respectively. As it can be seen, our method does not impose timing overhead on the links and in some cases the delay of links is 5 to 10 percent better than that of the state-of-the-art timing driven flow.

As we explained before, from the signal integrity point of view our

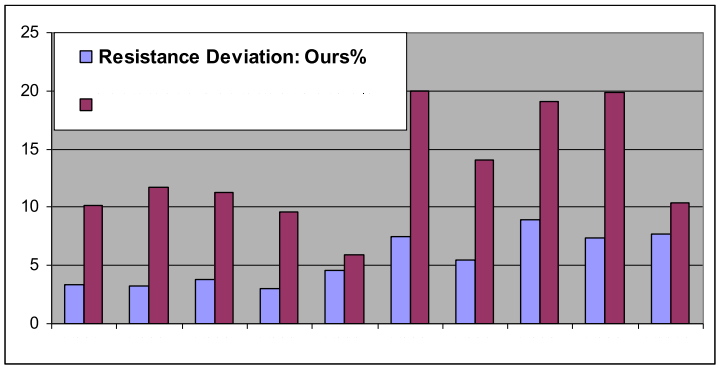


Figure 3.8: Resistance variation.

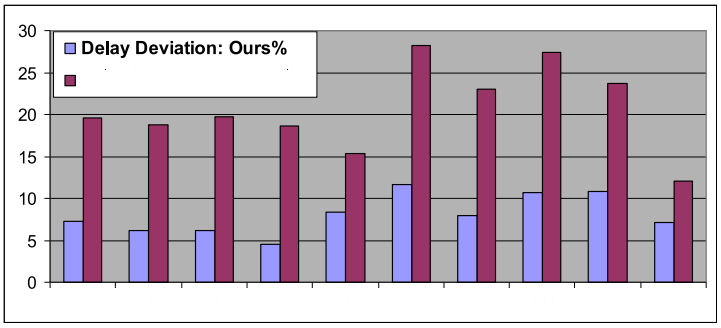


Figure 3.9: Delay variation.

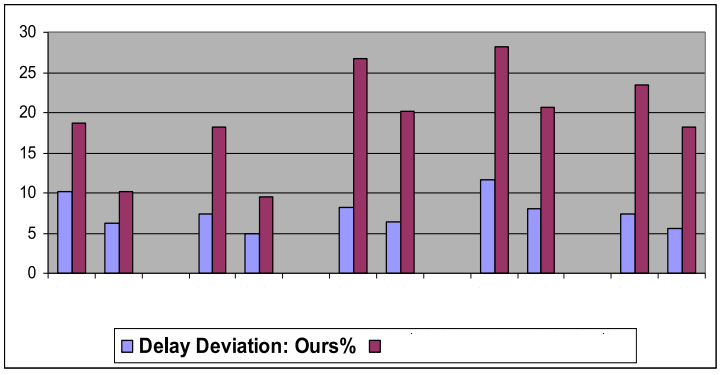


Figure 3.10: Average and Maximum of delay deviation in all 5 test cases.

methodology gives better results since we have control on the entire link route. We applied spacing and shielding to reduce the crosstalk effect on links. In spacing we increased the space between different wires of each link and in shielding we shielded all nets of each link with a grounded net.

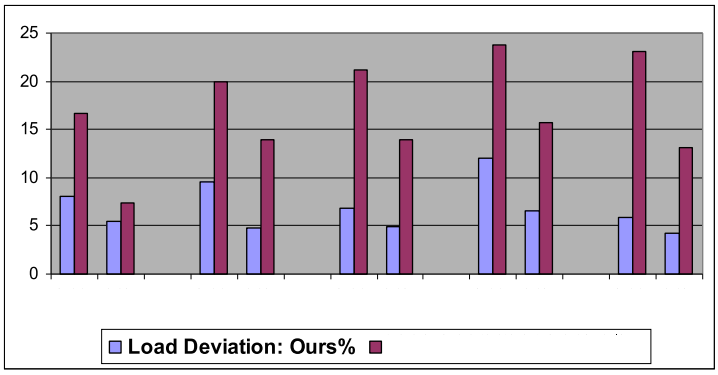


Figure 3.11: Average and Maximum of load deviation in all 5 test cases.

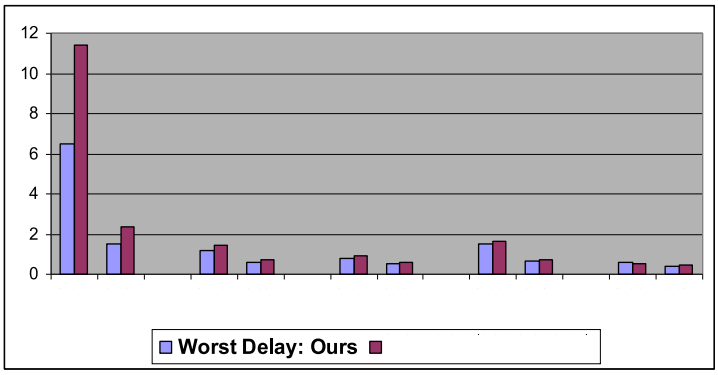


Figure 3.12: Average and Maximum of wors-case delay in all 5 test cases.

Figure 3.13 on the next page compares the average of crosstalk delta delay for each link in D2 for a normal flow with that of our approach with/without shielding. Crosstalk delta delay is a portion of the net delay which is related only to the crosstalk effect on that net. Figure 3.14 on the following page compares the average of delta delay for each link in D2 for the normal flow with that of our approach with three different spaces between nets of links. As can be seen, although the crosstalk delta delay of normal flow is sometimes smaller than that of our flow without shielding and spacing, we are able to reduce it in our routing flow by shielding and spacing. By shielding we decreased the crosstalk delay by 90 to almost 100 percent, and by increasing the space between wires of links by factors of 1.5x and 2x, we reduced the x-talk delay by 30-50 and 90-100 percent, respectively. Crosstalk can also be reduced in the standard flow. However, this is achieved by changing the spacing and trajectory of the nets, which leads to significant net attributes variation. Moreover, as in the normal flow the

trajectories of the nets of links are very diverse, controlling cross-talk is more difficult than in our flow.

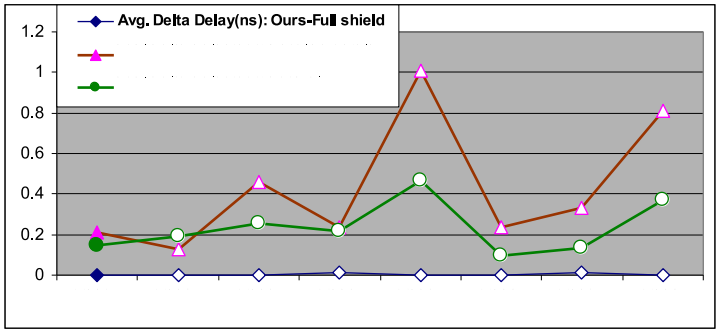


Figure 3.13: Crosstalk delay (ns) comparison in shielding.

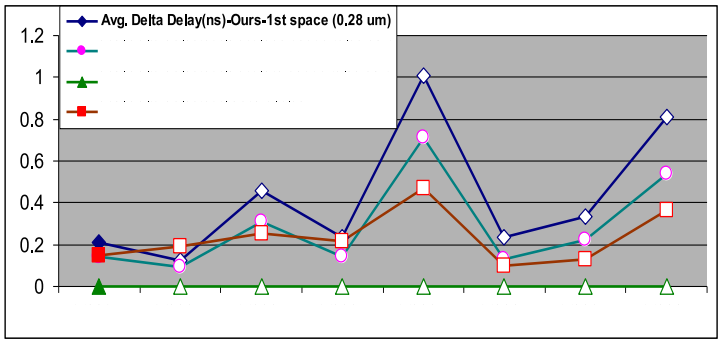


Figure 3.14: Crosstalk delay (ns) comparison in spacing.

To see the effect of the proposed routing methodology on link power consumption, we also compared the total net switching power of our method with that of the normal flow. Figure 3.15 on the next page compares the total net switching for all benchmarks. As it can be seen, our approach does not impose power overhead on the design. On the contrary, in most cases switching power is about 5 to 10 percent less than that of the normal flow, due to lower via count and less wire jogs. We also compared area. In all the benchmarks, our method does not impose area overhead, as we never had to change the size of floorplan. Finally, there are benefits on routing run-time as well. In our comparison experiments, the normal routing flow took from 3 to 5 times more run time than that of our bundled routing approach.

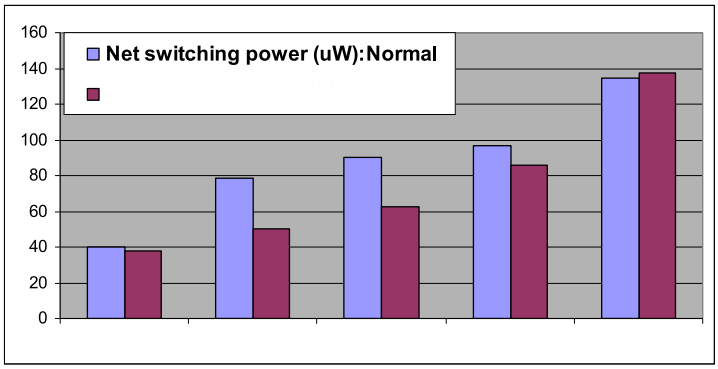


Figure 3.15: Net switching power (uW) comparison for all test cases.

3.6 Summary

In this chapter we proposed a new approach for bundled routing of NoC links. Our specialized routing flow produces highly regular link routes, thus leading to much reduced intra-link delay variations. Moreover, our flow supports spacing and shielding to ensure crosstalk immunity at the link level. Delay matching and low crosstalk noise are required features for advanced GALS synchronization and low-swing signaling, which are critical in NoC deployment for aggressively scaled CMOS. In the next chapter we focus on the reliability of NoCs at the architecture level and propose a reliable NoC design.

CHAPTER 4

Reliable Network on Chip Architecture

In the previous chapter we focused on reliability of NoCs at the layout level and described a robust physical routing mechanism suitable for global links of the NoCs. This chapter[1] directs attention to the reliability of NoCs at the architecture level and presents a robust design which is able to tolerate several logic faults inside the network. A dual physical channel NoC architecture is described which not only is reliable but also provides a soft Quality of Service (QoS). If there is no fault in the network, the two physical channels are used for QoS purposes; while if a fault occurs on one of the channels, the other one is used as a replacement.

4.1 Motivation and Key Challenges

Aggressive CMOS scaling accelerates transistor and interconnect wearout, resulting in shorter and less predictable lifespans for CMPs and MPSoCs [68]. It has been predicted that future designs will consist of hundreds of billions of transistors, with upwards of 10% of them being defective due to wearout and process variation [69]. Consequently, in order to support the current technology trends we must develop solutions to design reliable systems from unreliable components, managing both design complexity and process uncertainty [70, 71], with minimal nominal perfor-

[1]The author would like to acknowledge contributions by Prof. Valeria Bertacco and Prof. Luca Benini.

mance degradation.

The interconnect architecture in a chip multiprocessor becomes a single point of failure as it connects all other components of the system together. A faulty processing element may be shut down entirely, but the interconnect architecture must be able to tolerate partial failure and operate with performance or latency overhead [72]. Networks-on-chip [46] provide opportunities to address this issue, as redundant paths exist from point to point, potentially allowing for reconfiguration around failed components. The reliability of NoC designs is threatened by transistor wearout in aggressively scaled technology nodes. Wear-out mechanisms such as oxide breakdown and electromigration become more prominent in these nodes as oxides and wires are thinned to the physical limits. These breakdown mechanisms occur over time, so traditional post burn-in testing will not capture them. NoCs provide inherent structural redundancy and interesting opportunities for fault diagnosis and reconfiguration. Each router is comprised of input ports, output ports, decoders and FIFOs, which are replicated for each channel. By leveraging the redundancy that is naturally present in routers and the network, we are able to provide robustness to the communication subsystem.

On the other hand, each NoC imposes upper bound limits to the performance of the SoC in which it is implemented, heavily affecting the performance, latency, power and area of the chip. Therefore, NoC latency reduction is essential for performance as it is introduced to every communication stream within the SoC, becoming a critical design choice in presence of critical timing demands (real-time SoCs).

Priority-based NoC architectures are a well-known approach to provide quality of service (QoS) in networks and to reduce the latency of certain packet classes whose priorities is high [73, 74]. Virtual channels (VCs), first proposed by [73] as an effective workaround for head-of-line blocking, were proposed as an implementation of such priority-based QoS mechanisms by assigning statical priorities to each VC [73, 74, 75, 76, 77]: high priority VCs are used for QoS traffic classes and the low priority VCs are used for normal traffic classes.

The traditional VC switch architecture proposed for NoCs in the open literature [78], commonly used in VC-based QoS mechanism [74], is imported from the off-chip interconnection networks domain. Therefore, when adapted to the on-chip environment, the additional logic and components required for its implementation introduces new design challenges. On one hand, it adds new possible failure points, likely decreasing the switch reliability. On the other hand, it increments the complexity of the critical path, possibly reducing the maximum achievable frequency

when compared to a baseline VC-less switching fabric.

In this chapter we present an efficient and reliable switch architecture which distributes the traffic to two different physical channels based on the class of the traffic, which can be either QoS or normal. We name this architecture ReliNoC. ReliNoC is based on a novel design principle for VC switch architectures for NoCs. While traditional VC switch architectures replicate the buffering resources for each physical channel, our approach is to replicate the entire VC-less switch as many times as the intended number of VCs. Counterintuitively, the resulting design features a lower area than the traditional one. This result builds on a well known principle of logic synthesis: the area-performance trade-off for inferring the gate-level netlist of combinational logic. In ReliNoC, we push to the design principle of replicating networks components by replicating not only the switches, but also the switch-to-switch links. ReliNoC will take advantage of the replicated components by using them as replacements in presence of faults when needed, tolerating a wide range of combinations of network component failures. Additionally, in absence of faults, ReliNoC exploits the replicated resources by implementing a statical priority-based QoS mechanism that relies in the use of 2-channel switch architecture.

4.2 Previous work

4.2.1 VC-based and Multi-channel NoCs

A number of works has evolved the VC switch microarchitecture to optimize latency and to develop an infrastructure for QoS purposes in the CMP domain [73, 74, 75, 76, 77]. Virtual channel flow control was first proposed by [73] as an effective workaround for head-of-line blocking. VCs impose performance and power overhead due to virtual channel allocators [79].

Several works exists in the open literature aiming at reducing the delay and power consumption of VC-based switch architectures for NoCs. For example, the work in [80] presents a novel approach for improving the performance of virtual channels for the cases when the target application of the design is known, consisting in the customization of the virtual channel allocation based on the traffic characteristics of the target application. Summarizing, the basic techniques to improve switch energy and delay are: speculation [76, 78], bypassing [81, 77], lookahead [82, 83], mod-

ified virtual channels allocators [83, 87, 86, 84] and lookahead routing [88]. *Speculative* techniques try to reduce the delay by assuming that the virtual channels are always free to be allocated, when this is not the case some cycles are required in order to recover from the speculation failure.

Other works exists with different proposals. For example the work in [85] proposes the use of dynamic virtual channels that can be allocated by any input port when an additional virtual channels is needed, aiming at a global channel resource sharing.

Authors of [79] proposed a multi-plane virtual channel router which has multiple crossbar switches and a modified switch allocator. This is proposed as a way to increase the flit transfer rate between input and output queues. Unfortunately, their interesting finding ultimately implies increased switch complexity and a critical path degradation.

Gilabert *et al.* in [89] proposed another approach to VC implementation. Instead of replicating only buffers for VCs, they replicated the entire switch and proved that their solution is more area/power efficient while potentially operating at higher speeds. However, they do not target the reliability of the router.

Young *et al.* in [99] performed a comparative analysis in terms of network performance, area occupation and power consumption of using virtual-channels versus multiple physical planes. They found that when the total amount of storage in terms of sequential elements (flip-flops) that can be assigned to each router is small, it is convenient to build a multi-plane NoC instead of an equivalent VC NoC while the opposite is true if there is room for more storage.

4.2.2 Reliable NoCs

Reliable network design was first introduced by Dally *et al.* in [90]. This network used link-level monitoring and retransmission to accommodate the loss of a single link or router anywhere in the network, without interruption of service. Constantinides *et al.* presented the BulletProof router, which efficiently used a combination of spares and component-level NMR techniques for router level reliability [92]. However, NMR approaches are expensive, as they require at least N times the silicon area to implement. A recent architecture was proposed by Park *et al.* that focused on protecting the intra-router connections against soft errors [93]. They explored a hop-by-hop, flit retransmission scheme in case of a soft error with a three-cycle latency overhead. Though each component has localized protection, there is no guarantee that the network will be operational if certain hard faults occur.

Fick *et al.* proposed Vicis, a NoC design that can tolerate the loss of many network components due to wear-out induced hard faults [72]. Each router has a built-in-self-test (BIST) that diagnoses the locations of hard faults and runs a number of algorithms like ECC, port swapping, and a crossbar bypass bus to mitigate them. However, they use routing tables which have area and power overhead and need to be reconfigured after fault detection. This reconfiguration logic imposes area and power overhead. Moreover, they did not propose any technique to recover faults in routing tables, port swapper, and arbiters.

Qiaoyan *et al.* in [94] proposed a technique which exploits the inherent information redundancy in the control path of routers to manage transient errors, preventing packet loss and misrouting in a Mesh with XY routing. They focused on transient faults and did not mention any technique for permanent faults.

Routing algorithms for fault-tolerant networks have been extensively explored for network level reconfiguration [95, 96, 97, 98]. These algorithms direct traffic around failed network components in a way that avoids network deadlock. Although we must also accomplish this, we additionally reconfigures around faults using other methods as well. We adopt an implementation of the routing algorithm described in [97] for part of our network level fault recovery.

4.3 New VC design paradigm

This section introduces the new VC design paradigm presented in this chapter.

4.3.1 Baseline VC-less switch

We base our analysis on a packet-switching network model, called Xpipes, introduced in [15]. Each router in the Xpipes network is composed of N input and M output ports, and for our work we consider a typical router with N and M equal to five as shown in Figure 4.1 on the following page. Note that, in this figure we show only one input port (West) and one output port (East); other input and output ports have the same architectures.

The switch contains Control-path and Data-path as shown in the figure. Data path includes inter-switch links (except flow-control signals),

output multiplexers (crossbar), input/output buffers, and switch's data-path wires. Control path consists of routing computation (RCs), switch allocators, control logic of the buffers, control signals inside the switch, and inter-switch flow-control signals.

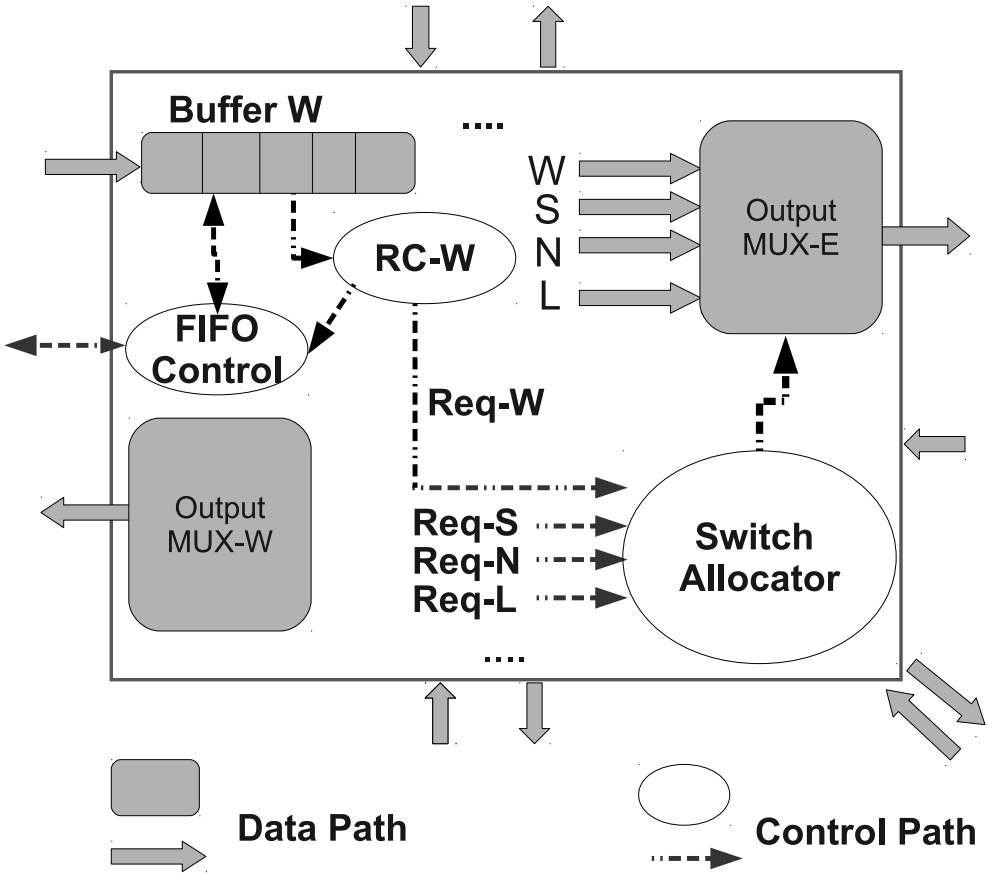


Figure 4.1: Xpipes switch used in the ReliNoC architecture. (RC: Routing Computation)

One pair of input/output ports is dedicated to the connection between the router and the core, while the remaining four pairs connect the router with the four adjacent routers. Switching is based on the wormhole approach, where a packet is partitioned into multiple flits (flow control units): a header flit, a set of payload flits followed by a tail flit. Flit is the smallest unit over which the flow control is performed and its size equals the channel width. For the case-study NoC topology, the communication channels between routers are defined to be 32-bit wide. Without loss of generality we consider a 2-D mesh as our topology with XY routing where a packet is first routed on the X direction and then on the Y direction before

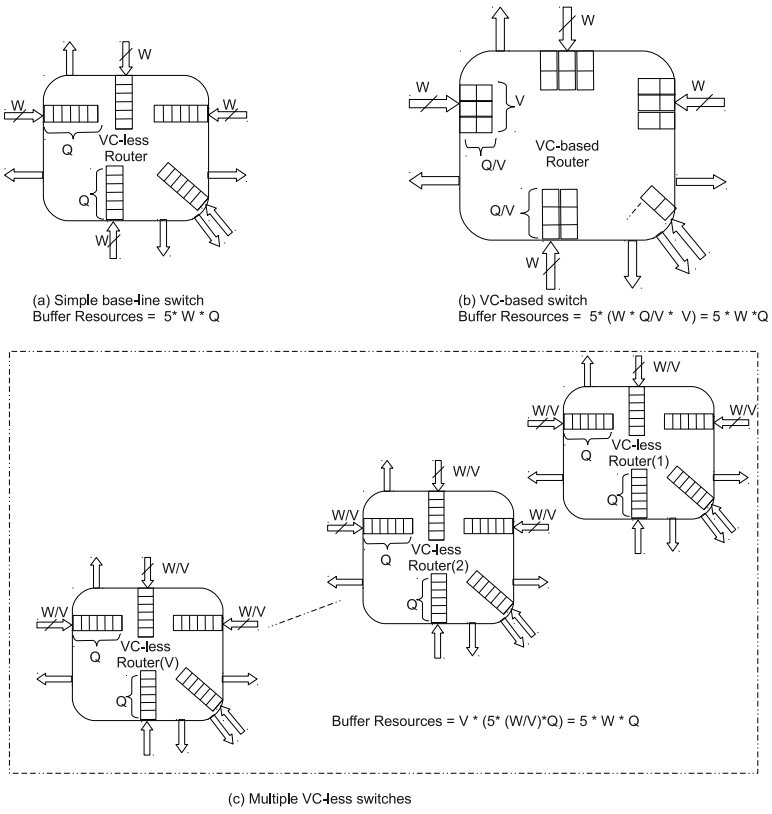


Figure 4.2: Different NoC architectures with the same buffer resources.

reaching its destination.

4.3.2 Multi-channel NoC

Typically, a single buffer is associated with each physical channel in NoC. Figure 4.2-a shows the simple VC-less switch with one physical channel at each direction. Virtual channels provide multiple buffers for each channel, so that when a certain virtual channel is congested, the packets in the other virtual channels can still progress through the same physical channel and the network throughput can be significantly improved. This is obtained through the partitioning of the storage resources to enable selectively buffering of the incoming worms so that they do not interfere during the forwarding process. Figure 4.2-b shows the VC-based router with the same number of physical channels as that of VC-less router. This VC-based router features V virtual channels. The depth of VCs are chosen in such a way that the amount of buffer resources be equal to that of the

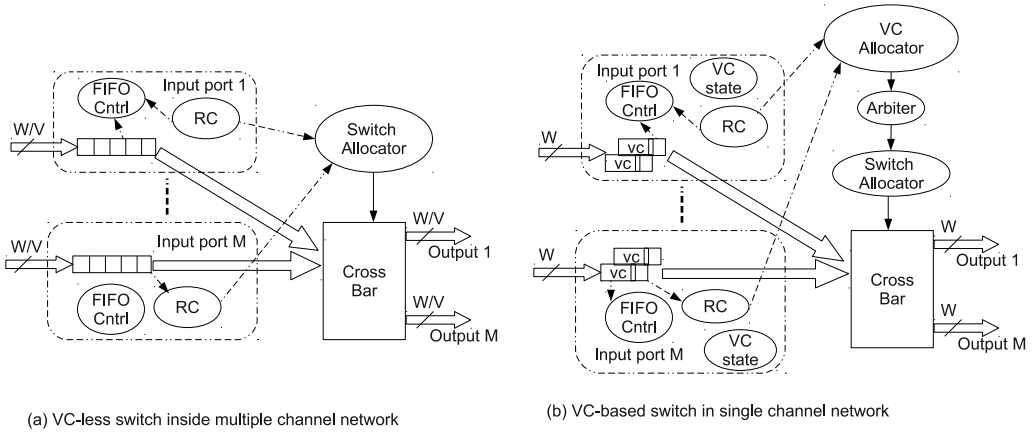


Figure 4.3: VC-less switch in multi-channel NoC vs. VC switch in single-channel VC-based NoC

baseline VC-less router.

As an alternative to VCs, the use of multiple physical networks on the same chip has been proposed to improve performance and keep traffic classes separate. For instance, in the RAW processor four separate and independent NoCs are used: two NoCs are statically routed and two are dynamically routed [91]. In this case, the entire baseline *VC-less router* is replicated as many times as the intended number of virtual channels. To keep the amount of buffering constant, the flit width of each router is divided by the number of virtual channels. The physical networks are completely separated and work in parallel. Network interfaces (NIs) choose the network to which the packets should be sent based on the class of the traffic. Figure 4.2 on the previous page-c shows the multi-channel architecture with the same amount of buffering as that of the base-line router.

The router architecture in the multi-channel NoC is simpler than that of the VC-based network. In the VC-based router we need another allocator and arbitration scheme for VCs as well as the related logic which stores the states of the VCs. Figure 4.3 shows the architecture of two routers: the VC-less router used in a multi-channel NoC and the VC-based router.

Since we replicate the entire baseline router for the multi-channel NoC, at the first glance it seems that the VC-based architecture is more power and area efficient than the multi-channel architecture. However, as explained in [89, 99] it depends on the number of intended VCs and buffer depths. When the number of intended VCs is small (less than 4) and the buffer depth is less than 8 flits, the multi-channel architecture has better power and area compare to the VC-based architecture provided that the

amounts of buffer resources in both architectures are the same. When comparing the baseline router and the VC-based router , this later has more functions on the critical path due to the VC allocators and output arbiters hence potentially resulting in a poor area/power-efficient gate-level netlist after logic synthesis. In fact, the multi-channel architecture with the simpler router certainly provides a higher maximum speed than the VC-based one. However, if we require the two architectures to be aligned to the speed of the slowest one (the VC-based switch), then combinational logic of the multi-channel design can be inferred with relaxed delay constraints and therefore optimized for area and power [89].

4.3.3 Proposed 2-channel NoC

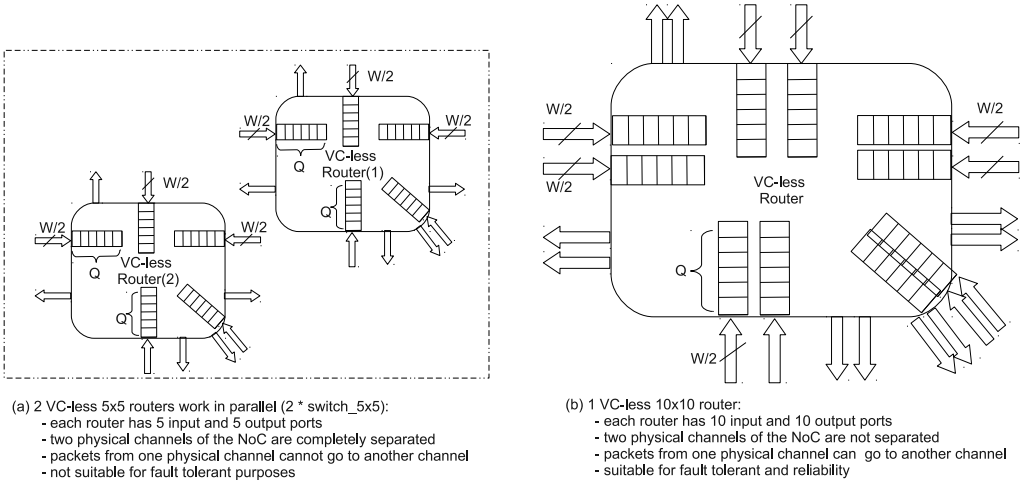
Based on the above discussion, the multi-channel network is better than the VC-based one when the number of channels is less than 4 and the buffer depth is small.

There could be two techniques to create a 2-channel NoC. The first method is to duplicate the baseline NxN router and make two parallel networks. The second approach is duplicating all the resources inside one switch including ports and links and building the network with $2N$x$2N$ routers. In this case, each router is connected to its neighbors using two different physical channels each of which has a width equals half of the width of channels in the NxN router. Figure 4.4 on the next page shows these two methods for a baseline 5x5 router. The amounts of buffering in both techniques are the same. The second approach is a little more complex due to bigger crossbars, but has better flexibility.

The main difference between these two approaches is the network adherence. In the first method two networks are completely separate and, therefore, one packet cannot move between channels. However, the NoC created by the second technique is more integrated and packets can traverse between two channels. This property is very suitable for making the NoC reliable and we use this architecture in our work and propose the related reliable NoC in the next section.

4.4 Reliable 2-channel NoC

In this section we describe how to make the proposed 2-channel router reliable. As mentioned earlier, the network has two physical channels.



(a) 2 VC-less 5x5 routers work in parallel (2 * switch_5x5):
- each router has 5 input and 5 output ports
- two physical channels of the NoC are completely separated
- packets from one physical channel cannot go to another channel
- not suitable for fault tolerant purposes

(b) 1 VC-less 10x10 router:
- each router has 10 input and 10 output ports
- two physical channels of the NoC are not separated
- packets from one physical channel can go to another channel
- suitable for fault tolerant and reliability

Figure 4.4: Different approaches for designing a 2-physical channel NoC

Channel one is dedicated to QoS traffic, while channel two is shared between QoS and normal traffic. We call these channels channel-1 and channel-2 respectively. The network and switch architecture are shown in Figure 4.5 on the facing page.

Utilizing the properties of the proposed 2-channel switch, a reliable switch architecture is introduced to ensure fault-tolerant operation of the router. We explore various possible failure modes within an NoC router, and propose detailed recovery schemes with minimum area and power cost. Our switch architecture possesses some inherent fault-tolerance due to its replicated design. This additional operational granularity may be utilized to allow replacement of a faulty component by another one, thus allowing operation of the router with latency overhead instead of a complete breakdown. Note that, our proposed scheme avoids the more traditional approach in fault-tolerance, which resorts to replication of resources without utilizing them in normal and non-faulty designs. Silicon real-estate and energy are at a premium in on-chip applications, thus necessitating the efficient re-use of existing resources.

Five major components of the router i.e. the routing computation unit (RC), the arbiters, the buffers, the output MUXes, and the links, are susceptible to permanent faults. We modified our 2-channel switch to overcome faults in all these components. The proposed reliable switch architecture is shown in Figure 4.6 on page 76. Four types of components are added to the 2-channel switch for fault tolerance purposes: 1) one MUX per each input channel 2) 4-bit input-fault-status register per each direction (2 bits for each channel) 3) one control logic for both channels at each direction 4)

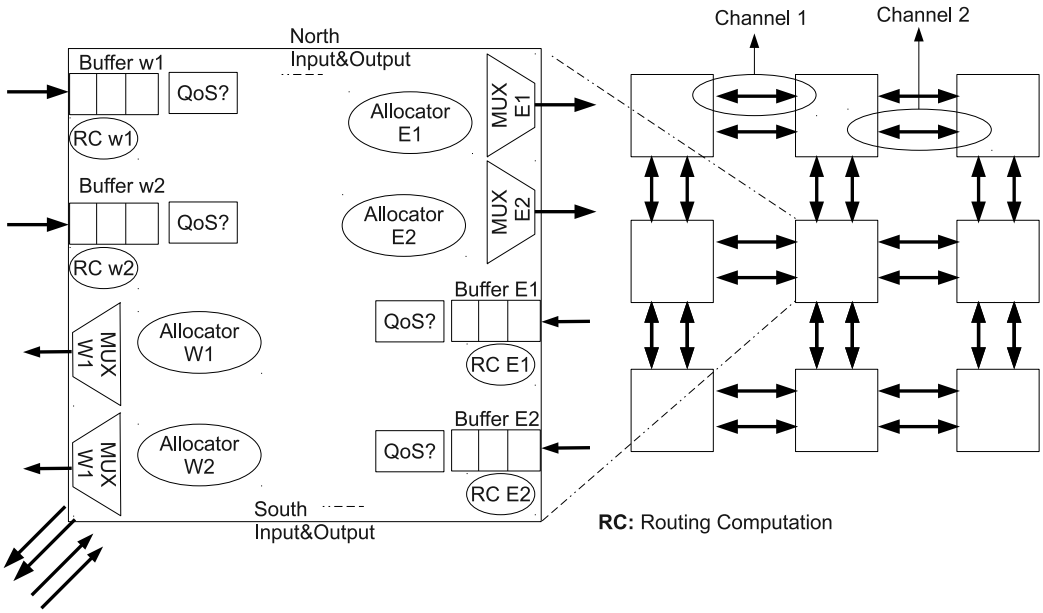


Figure 4.5: NoC with 2-channnel routers.

10-bit output-fault-status register for the entire router.

As shown in Figure 4.6 on the following page, one MUX is added to each channel at the entry of buffer (MUXBUFF). There is a 4-bit register for each direction storing the faultiness status of links and input ports of that direction. We distinguish faults in links and input ports and dedicate one status bit for each of them in the register. We call this register ISR (Input Status Register). One simple control logic reads the ISR and generates appropriate control signals to MUXBUFF based on the faultiness of each component. Finally, there is a 10-bit register for the entire switch tracing the faultiness of all output channels in that switch. We call this register OSR (Output Status Register) and there is only one OSR in each router. Note that, we consider one output channel as a faulty channel if there is a fault in at least one of the following components of that channel: allocator, out-multiplexer, link, and the input port of the corresponding next router. ISRs and OSR should be updated by a periodical testing approach. A NoC testing methodology will be described in the next chapter.

Every two connected routers in the network send the information of their corresponding ISR and OSR to each other by dedicated signals. Each router receives two signals from its neighbor per each channel. One shows the faultiness status of its neighbor's input channel and another one for the output channel. We call these signals input-channel-faulty (ICF) and

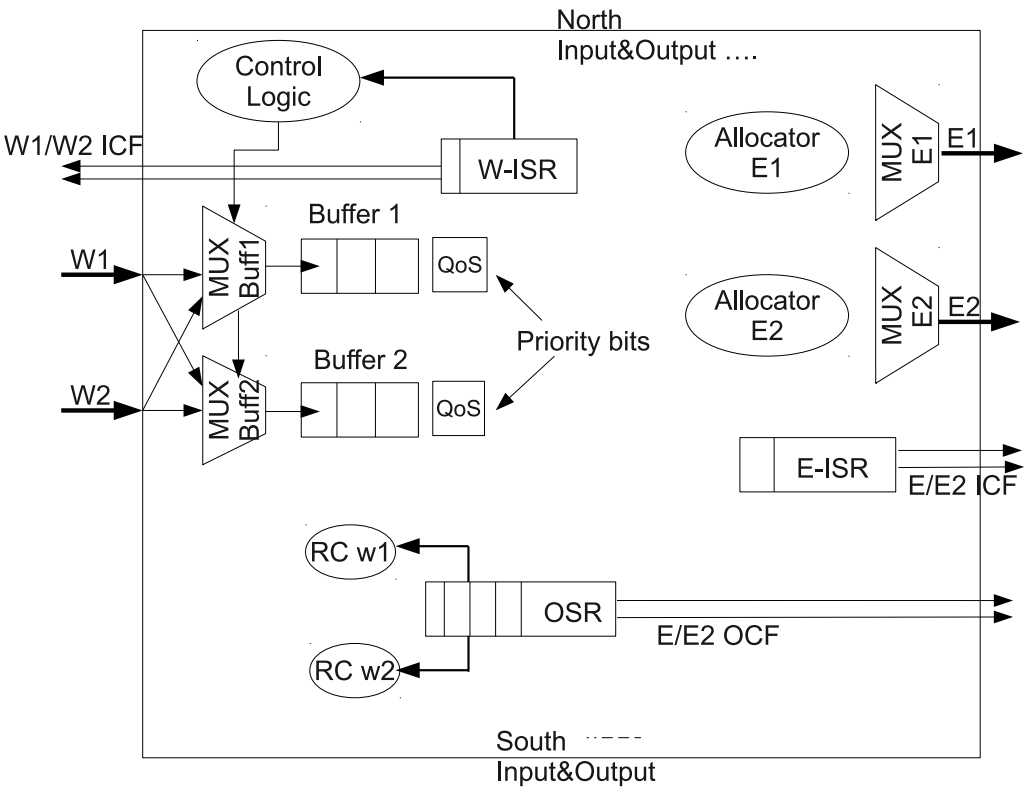


Figure 4.6: Reliable architecture based on the proposed replicated switch.

output-channel-faulty (OCF) respectively. Therefore, each router receives and sends two signals per each channel. Thus, the number of extra wires which need to be added to each channel is 4 bits which is straightforward due to availability of wires in today's SoC designs.

If router A receives an active OCF signal from router B for a specific channel, this means that router B never sends data to router A on that channel. This is equal to having faults on the input link of that channel in router A and, therefore, router A sets the "link" faultiness bit in the corresponding ISR to '1'. If router A receives an active ICF signal from router B on a specific channel, it means that router A should not send data on the corresponding output channel. This is equal to having faults on the output link of the related channel in router A and, therefore, router A sets the corresponding bit in its OSR to '1'. Therefore, the final "link" faultiness bit in the ISR is simply the OR of original "link" status bit and the corresponding OCF signal of previous router. Also, all the bits in OSR are simply ORed with the corresponding ICF signals coming from the neighbors.

Based on the values in ISR, control logics generates appropriate signals

for MUXBUFF of each channel. The information in OSR is used by the RC (Routing Computation) logic to specify the appropriate output channel for each packet. Depending on the status of ISR and OSR, a few different cases exist for each channel and each direction of the router. In the following paragraphs we outline these cases for the EAST direction and its two channels (E1 and E2) and describe our fault recovery schemes for each situation. The mechanisms for other directions are exactly the same as those of EAST.

All bits in ISRs of E1 and E2 are '0': this means all the input components at East channels (E1 and E2) of the current router and all the output components at West channels (W1 and W2) of its related neighbor are non-faulty. In this case the router works in its normal operation.

The bit related to input channel-1 in ISR is '1': this means one of the input port components at channel-1 is faulty. This includes buffer, RC and MUXBUFF. In this case we consider channel-1 as faulty. An ICF signal is sent to the previous router, and no data will arrive at channel-1. Here, channel-2 is utilized for both types of traffics. Thus, control logic does not change control signals for MUXes. Note that, even in this case the QoS traffic has priority over normal traffic because of the priority bit which is considered in allocators.

The bit of link-1 in ISR is '1': this means Link-1 or the output channel of the previous router is faulty. Therefore, no data will arrive at channel-1 because of the ICF signal which is already sent to the previous router. However, in this case if buffer-1 and MUXBUF-1 are non-faulty, they can be used by channel-2. Here, control logic generates appropriate signals so that channel-2 is connected to both buffer-1 and buffer-2. In this case buffer-1 is used for QoS traffic and buffer-2 for both QoS and normal traffic.

The East-channel-1 bit in OSR is '1' and East-channel-2 bit is '0': this means one of the components (arbiter, link, multiplexer) at the output channel E1 is faulty or the input channel of the next router which is connected to E1 is faulty (ICF is '1'). As all the RCs take the OSR bits into account, in this case no traffic will go through E1, and E2 is used by both QoS and normal traffic.

Both East-channel-1 and East-channel-2 bits in OSR are '1': this means both output channels at East direction are faulty. In this case, the packet cannot go to the East direction and should be detoured to another direction. To do so, we use Logic-Based Distributed Routing (LBDR) which is a fault tolerant routing algorithm proposed by Rodrigo *et al.* in [97]. This routing is a logic-based mechanism which can be implemented on top of any distributed routing like XY to detour the faulty link. This

logic eliminates the need of routing tables (either at routers or at end-nodes). Routing tables do not scale in terms of latency, power consumption, and area, thus being impractical for large NoCs [102]. As described in [97], the area overhead of the added fault tolerant logic on top of the traditional XY routing is around 4% which is much less than that of table based routings. This additional logic increases the timing path of RC by 20%. However, as the timing critical path of the router is not through the RC logic and this module (RC) is not the one setting the router frequency, this performance overhead on the RC does not have any effect on the router performance [97]. The added logic needs some status bits to find the detour port. Eight of them which represent the faultiness of each output port are already in OSR. Three other bits should be added to each direction to implement the fault-tolerant logic on top of XY. Interested reader can refer to [97] for more details on LBDR.

Since we do not recover faults in fault-status registers (ISR and OSR), the recovery techniques in our architecture rely on having robust design for these registers. Error correcting codes (ECCs) [103] are good robust mechanisms for registers. In an ECC, a few extra bits are added to the raw data to protect it against errors. Various ECC mechanisms have been integrated within VLSI chips with large internal memories and cache units [104] as well as NoCs [72].

The reliability mechanism presented in this chapter is based on knowing precisely the location of faults in the switch. Built-in-self-test (BIST), a well-known approach to diagnose faults, has been extensively addressed as a post-silicon technique for fault detection during the NoC lifetime [72, 105, 100, 101]. To utilize our recovery mechanisms, the network requires to go periodically into self-test in regular intervals and update ISRs and OSRs. Also, at the beginning of each power-on the network should test itself and update these registers.

4.5 Experimental Results

4.5.1 Hardware cost

To evaluate the hardware cost of the reliable router, we compared different architectures in terms of area and power. Synopsys design compiler is used for hardware synthesis and designs are mapped on CMOS 45nm technology from ST-Microelectronics.

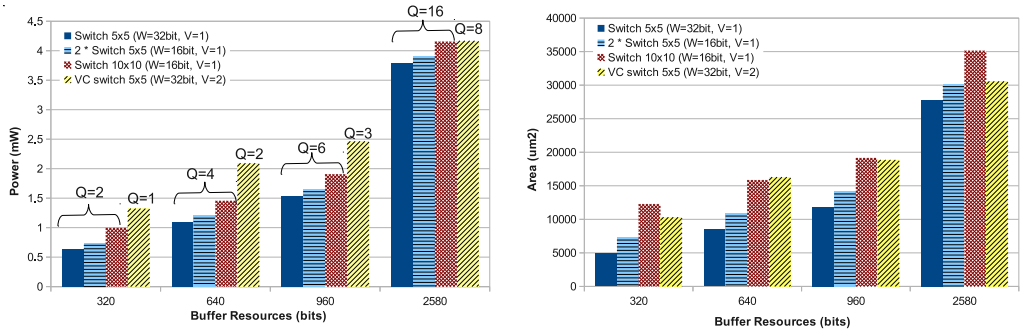


Figure 4.7: Synthesis results of different switch architectures

Four different routers are developed and synthesized to compare the area and power of various architectures. These four routers are: (i) simple VC-less 5x5 router with a flit-width of 32 bits (ii) duplicated VC-less 5x5 router with a flit-width of 16 bits (iii) simple VC-less 10x10 router with a flit-width of 16 bits (iv) VC-based 5x5 router with a flit-width of 32 bits and two VCs (V=2). Note that, for each comparison the buffer resources in all architectures are the same. Buffer resources depend on the depth of the FIFO in the baseline VC-less switch. For example if the depth of the FIFO is 2 flits, the total buffers needed in the whole router is $5 * 2 * 32 = 320bits$. We swept the depth of FIFOs from 2 to 16 flits and compared different architectures.

The synthesis results of power and area for different amounts of buffer resources are shown in Figure 4.7. As shown in this figure, when the depths of buffers are less than 6 flits (3 flits for VC-based router) the 10x10 router with a flit-width of 16 bits consumes less power than the VC-based 5x5 router with flit-width=32 bits. The results, also, show that the area of 10x10 router is almost the same as that of VC-based 5x5 router. This results again confirm that for a small number of buffers and VCs, physical channels are better than VCs and feature more area and power efficiency.

To see the hardware overhead of our reliable schemes on the 10x10 router, we added the new components including input MUXes, ISRs, OSR, fault tolerant RC and control block to the router and synthesized it. The power results are shown in Table 4.1 on the following page. As it can be seen, the power overhead is from 6.5% to 13% depending on the depth of the FIFOs. For a reasonable depth, e.g. 4 flits, the power overhead of all the components including ISR, OSR, control logics, and fault tolerant RC, is only 9%. The trend for area overhead is the same as that of the power.

Table 4.1: Power overhead in the reliable 10x10 router

# depth of FIFOs (flits)	*Power (mW) simple router*	*Power (mW) reliable router*	*Overhead*
2	1	1.13	13%
4	1.45	1.59	9%
6	1.90	2	6.5%

4.5.2 NoC latency evaluation

To evaluate our 2-channel and reliable switch architecture in terms of latency, we used Noxim [106] which is a cycle accurate NoC simulator implemented in SystemC. The switch model in this simulator has a 2-stage pipeline and therefore has 2 cycles minimum latency. It originally does not support virtual channels, and multiple physical channels. We extended it to support both virtual and physical channels. We also extended it for injecting faults on the NoC at high level of abstraction. Moreover, we modified the simulator for our reliability and recovery purposes and extended it to support the fault-tolerant routing algorithm proposed in [97].

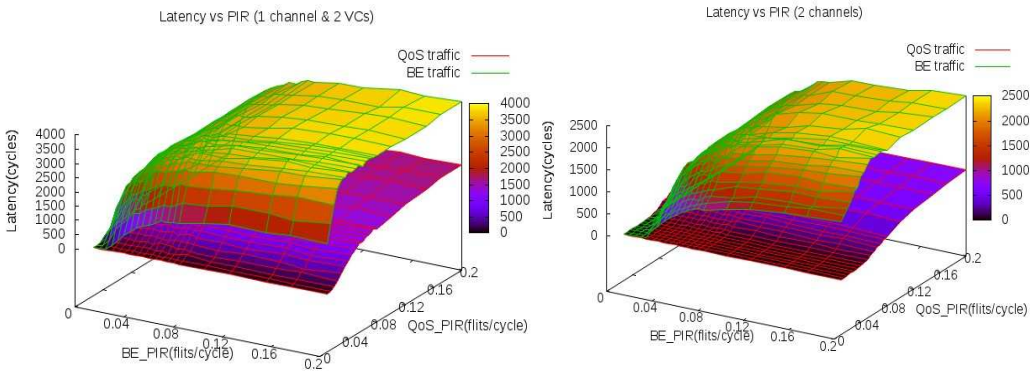


Figure 4.8: Latency vs. packet injection rate (PIR).

To see the effectiveness of our 2-channel router with respect to VC-based router, we measured the average latency on 8x8 Mesh networks of both switches. We applied different synthetic traffic patterns including Unified Random, Butterfly, Transpose, and Bit complement on both networks. We injected two classes of traffic to the network including QoS and normal, and measured the latency of both traffic types as a function of packet injection rate of both traffic classes. The results for unified random

traffic are shown in Figure 4.8 on the preceding page. In this 3-dimensional plot, the X and Y axes represent the packet injection rate (PIR) of normal and QoS traffic respectively. The Z axis shows the latency. The surfaces in Figure 4.8 on the facing page are the latency of QoS and normal traffic. As seen, in both routers the PIR of normal traffic does not have effect on latency of QoS traffic. This is because of the arbitration policy in the router which gives high priority to the QoS traffic. However as can be seen, the latency of normal traffic in network with 2-channel switch is much less than that of NoC with 2-VC switch. The 2-channel switch shows its efficiency over 2-VC switch while increasing PIR of QoS traffic. As shown in the plots, increasing PIR of QoS traffic has much less impact on latency of both traffic types in NoC with 2-channel switch compared to that of NoC with 2-VC switch. As seen, for any given point in the QoS PIR and normal PIR the latency of both QoS and normal traffic has been reduced by 30 to 50 percent.

4.5.3 Network connectivity

To see the effect of faults on the network, we injected faults on the entire 8x8 NoC. Our fault model is based on the area and covers six different components (RC, MUXBUFF, buffer, arbiter, out-MUX, and link) of each channel in each router. Therefore, each router has 60 components which can be subject to faults. Each component has a portion of the total switch's area. We calculated the area portion of each component by synthesizing the RTL version of the reliable switch. Later, we calculated the area portion of each component of every router in the entire network containing 64 routers. We injected faults on the network based on the area portion of each component in the network. Those components that have more portions of the total NoC's area have more probability to receive fault.

First, we compared the network connectivity of our reliable architecture with that of a network comprising 2-VC switches in presence of different number of faults. In this experiment, we injected different number of faults on the network on different random components by giving higher probability to larger components. Then, we checked statically if the network is fully connected or not. We consider a network as fully-connected if there is at least one physical path from each source to each destination regardless of routing algorithm or reliability mechanisms. For any number of faults, we injected that amount of faults on the network with 1000 different random seeds, and calculated the average number of fully-connected networks out of 1000 possible fault distributions. We did this experiment for two types of networks: i) 8x8 NoC with 2-channel switch and ii) 8x8

NoC with 2-VC switch. Figure 4.9 shows the plot of network connectivity of both cases in terms of number of faults. The Y axis in this chart is the probability of having a fully-connected network as a function of "number of faults" which is X axis.

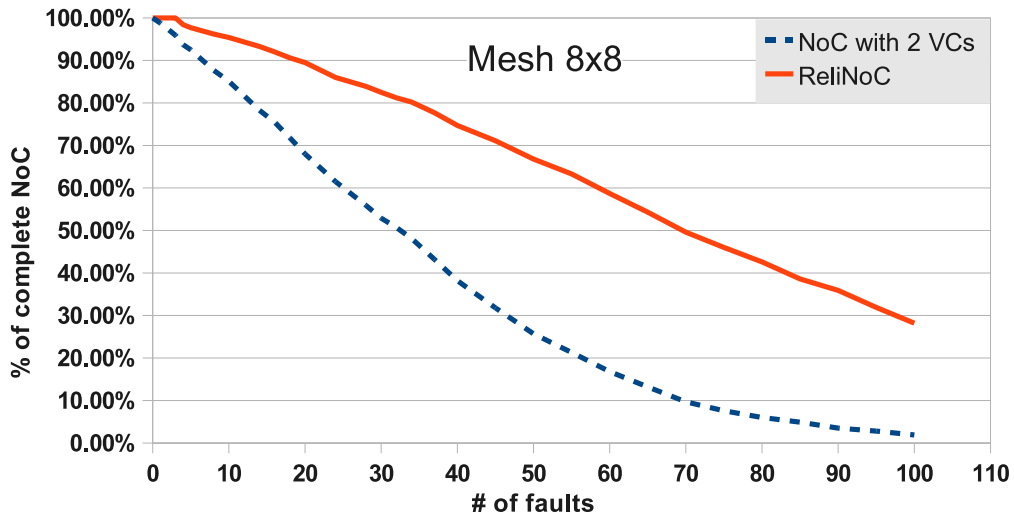


Figure 4.9: Network Connectivity vs Faults.

As can be seen in Figure 4.9, our 2-channel network has much better connectivity in presence of faults. For example, in presence of 20 faults, our 2-channel NoC has 90% probability of fully-connected network, while that of NoC with VC-based switches is 70%. For 40 faults, our reliable NoC shows 2 times better physical connectivity compare to the VC-based architecture.

4.5.4 Fault distribution on NoC components

To see how faults are distributed among different components in the NoC and to check which components are more susceptible to faults, we calculated the average number of times that each type of component receives fault in all experiments (different number of faults) and all different configurations (1000 random seeds). Figure 4.10 on the next page shows the related chart. As it can be seen, buffers are the most vulnerable components and receive around one third of the total faults. This is because buffers are the largest components inside our switch in terms of area. After buffers, output multiplexers are the largest components and, therefore, most sensi-

tive components to the faults. Although links receive only 5% of the total faults, their faultiness has high impact on the network connectivity.

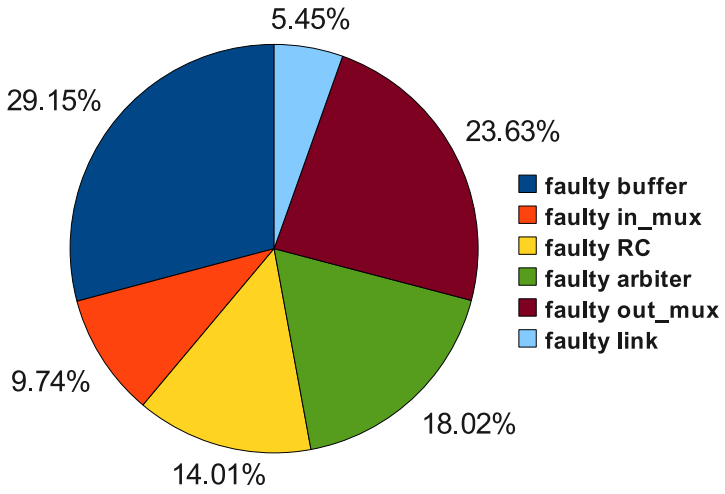


Figure 4.10: Percentage of faults on different components.

4.5.5 NoC recovery evaluation

To see the effect of faults on the latency and to verify our recovery mechanism, from previous experiments we selected those configurations that have fully-connected networks. Then we injected faults and ran the simulation for different synthetic traffics. Note that, since we selected only those configurations that do not break the network, all packets should finally arrive at the destination even in the presence of faults. For all fully connected networks in each case, we measured both average and worst-case latency among all configurations. Figure 4.11 on the following page shows the relation between average latency of both QoS and normal packets and number of faults for 2 synthetic traffics: random and butterfly. In this is obtained as can be seen, when the number of faults increases, the latency of both QoS and normal packets increases. However, in both benchmarks the QoS traffic pays less latency penalty compare to that of the normal traffic. Experiments showed that our recovery mechanism could handle all the fully-connected networks for up-to 50 faults, and all packets reached the destination with a reasonable latency overhead.

We also calculated the worst-case latency overhead in all fault configurations having fully-connected networks. Experimentally, we found the

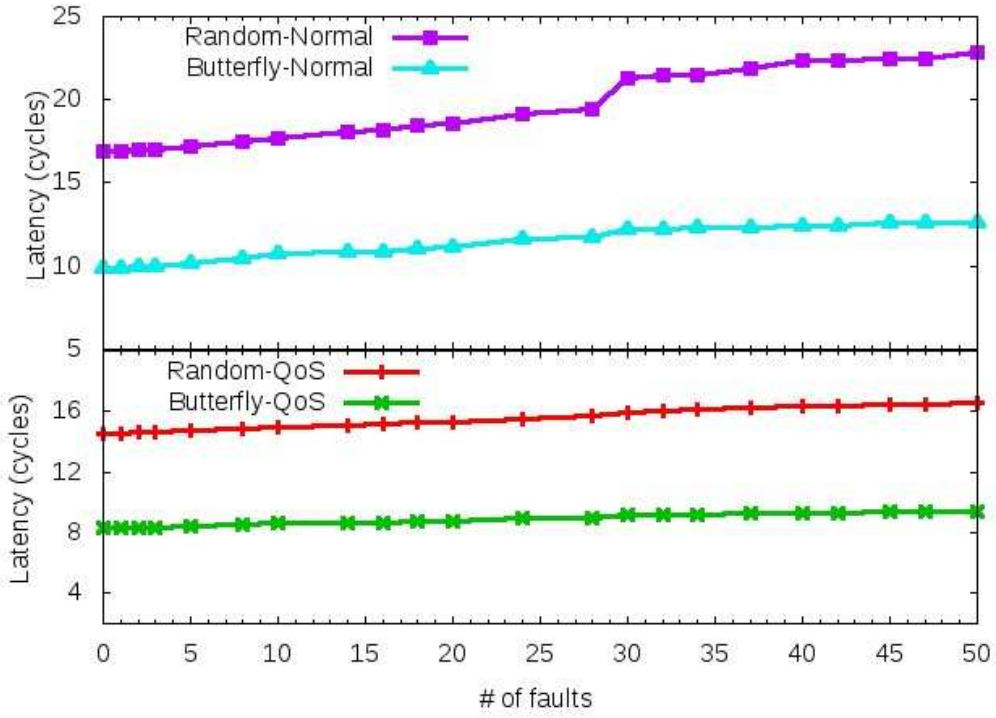


Figure 4.11: Average latency vs number_of_faults in 8x8 NoCs.

worst-case latency overhead in Bit-Complement traffic. Our Experiments showed that even in the worst-case, the latency overhead on QoS packets was 30% for up-to 25 faults. However, in the worst-case configuration, the normal packets payed much more latency overhead in presence of many faults compared to the QoS packets. The latency overhead in normal packets of Bit-Complement traffic for 15 faults in the worst-case configuration was 5X.

In addition to the synthetic traces, we also performed simulation to see the effect of faults on 5 real traffic traces from the PARSEC benchmarks, a suite of next-generation shared-memory programs for CMPs [107]. The traces used are for a 64-node shared memory CMP arranged as a 8x8 mesh. Each processor node has a private L1 cache of 32KB and 1MB L2 cache (64MB shared distributed L2 for the entire system). There are 4 memory controllers at the corners. To obtain the traces, we used Virtutech Simics [108] with the GEMS toolset [109], augmented with GARNET [110], simulating a 64-core NoC.

Like the previous experiments we chose those fault configurations that

give us fully-connected networks, injected faults and ran the simulation in Noxim for each benchmark trace. Each trace contains two types of packets: data and control. We considered control and data as QoS and normal packets respectively. Figure 4.12 shows the relation between latency and number of faults for each benchmark and for each packet type. As can be seen, in the real traffics our reliable network can recover up-to 50 faults with a very low penalty on latency in both traffic types. In all the benchmarks and for 50 injected faults, the maximum latency penalty on control and data packets is 10% and 30% respectively.

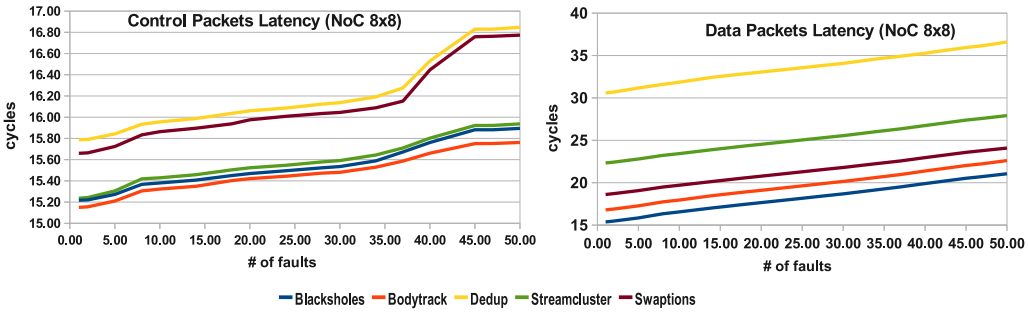


Figure 4.12: Latency vs number_of_faults in PARSEC benchmark.

4.6 Summary

In this chapter, a priority-based switch architecture which can provide reliability in presence of several faults inside the NoC has been proposed. First, a novel approach to design VC switch architectures for NoCs is introduced. With respect to a conventional virtual channel architecture, by simply replicating switches as many times as the intended number of VCs, a simple yet efficient implementation can be achieved. Then, we proposed the reliable architecture which utilizes the inherently redundant components inside the 2-channel switch as replacements for the faulty elements. In the next chapter we will focus on NoC online testing and propose a technique to find logic and delay faults inside a NoC during its lifetime.

CHAPTER 5

Distributed Functional NoC Online Testing

In the previous chapter we presented a robust NoC design which is able to tolerate several logic faults inside the network. However, we did not talk about the testing mechanism and how we can detect faults inside the network. This chapter[1] describes a scheme for on-line detection and diagnosis of faults in NoC's routers and links. We propose a functional test architecture, which can be implemented in routers and NIs with small hardware overhead. We also propose a hardware-efficient Fault Diagnosis Module (FDM) in each router, which can diagnose the location of the fault.

5.1 Motivation and Key Challenges

A Network on Chip consists of routers, links, network interfaces (NIs), and cores. Failures can occur in any of these components. We focus on failures in routers and links, and propose an on-line functional testing solution for the NoC infrastructure. A number of works has evolved NoC reconfiguration to counteract faults in both routers and links [118, 97, 95, 96, 98, 93, 119]. For instance, routing algorithms for fault-tolerant networks have been extensively explored for network-level reconfiguration. These algorithms direct traffic around failed network components to avoid network deadlock. However, in order to take countermea-

[1]The author would like to acknowledge contributions by Prof. Valeria Bertacco and Prof. Luca Benini.

sures against NoC faults, we must first detect and diagnose them. Most of the works proposed for fault-tolerant NoCs do not address the testing mechanism and how they find faulty components. Some of the them rely on off-line testing mechanisms, such as BIST and scan chains, which require external test sources, are not scalable and cannot be applied while the system is in operation. Others rely on traditional error detection and correction mechanisms, such as CRC and parity checkers, to detect data-path faults in both links and routers [120, 121, 122]. However, their fault coverage is very limited and, more importantly, they cannot be used for fault detection in control paths of the router.

Lubaszewski, *et al.* in [123, 124] proposed a new post burn-in testing for NoCs, which is based on the at-speed functional testing of several 2x2 meshes in an NxN NoC. They test each 2x2 mesh using a set of test sequences and patterns. However, as they showed in [124], the 2x2 mesh can give good fault coverage for the links but not for the routers, especially for the routing logics, FIFO's control paths and arbiters. To obtain higher fault coverage for the routers they added more than 12 other mesh configurations including 3x1, 1x3, 2x2, and etc. However, having those additional configurations is not efficient for on-line testing and keeps several switches busy in testing procedure, impacting the NoC's performance and packet latency. Moreover, their approach is suitable for manufacturing testing using ATE (Automatic Test Equipment), and implementing it in hardware leads to overhead especially with its enhanced version.

Similar to the work in [124] we use Test Pattern Generator (TPG) and Test Response Analyzer (TRA). However, to have minimum performance overhead on the NoC in on-line mode we take advantage of TPG and TRA in both router and NI. We also propose a hardware-efficient Fault Diagnosis Module (FDM) in each router, which can diagnose the location of the fault. Second, we propose a new test sequence to test each router. Moreover, we use TMR for the reliability of testing components added to the router. Unlike the work proposed in [124], we do not use 2x2 meshes to test the entire NoC, but we test each router separately and using its neighbors in on-line and at-speed functional mode. We use logic and delay fault models and the corresponding system-level failure modes to better tune the test pattern sequences. Without lack of generality, in this work we base our analysis on Xpipes described in the previous chapter. Each router in the Xpipes network is composed of N input and M output ports, and for our work we consider a typical router with N and M equal to five. We consider a 2-D mesh as our topology with XY routing. However, we will show that our testing methodology can work with any topology and routing algorithm.

5.2 Related Work

NoC infrastructure is composed of three main parts : routers, links, and NIs. Since the functionalities of these components are decoupled from each other, it is possible to test each of them separately. Testing each component separately in NoC, means we are testing NoC structurally. An alternative way is to functionally test the whole NoC. In this case testing different elements of NoC is not decoupled from each other and they will be tested all together. The latter method is cheaper, but does not identify where is the defected element unless we have a fault diagnosis mechanism. Since the focus of this work is on routers and links, we give an overview of previous works performed on testing these parts.

5.2.1 Link Testing

As density in chip increases, the distance between wires decreases and accompanied with higher functional frequency, links are more susceptible to crosstalk noises. Other faults that can happen in links are shorts between wires of links (so called bridging), opens, shorts to VDD (stuck-at-1), shorts to Ground (stuck-at-0), and delay fault which is caused by noises or process, voltage and temperature (PVT) variations.

Grecu et al. [120] propose a built-in self-test methodology for testing the channels of the communication platform. The proposed methodology targets crosstalk faults assuming the MAF fault model [125]. The authors also suggest that shorts between wires within a single channel are detected as well. The test strategy is based on two BIST blocks: test data generator (TDG) and the test error detector (TED). TDG generates the test vectors capable of detecting all possible crosstalk faults in a channel connecting two adjacent routers. The test vectors are launched on the channel under test from the transmitter side of the link and then are sampled and compared for logical consistency at the receiver side of the link by the TED circuit. Their technique is suitable only for testing inter-switch links in offline mode; moreover, they do not provide any data on the gate-level fault coverage of their technique.

In another approach, Pande et al. [126] propose to incorporate crosstalk avoidance coding and forward error correction schemes in the NoC data stream to enhance the system reliability.

The authors in [127], propose a functional test method for detecting

delay faults in asynchronous NoC links connecting between two switches, which are in two different clock domains. In this method they are sampling data which is passed in interconnect before and after the handshake signals. If there is a timing error in the line, the data will not be the same before and after the handshake signals. Therefore by comparing the two sampled data they can detect the delay fault. However, their method is offline and is suitable only for links.

Raik et al. in [128] proposed a technique for diagnosing faulty links by applying test patterns at the border I/Os of the network. While very high fault coverage was achieved, their method is offline and the time complexity of the test configurations is square with respect to the rank of the NoC matrix. Moreover, in order to apply test patterns from network boundaries at-speed, a large number of test pins are necessary.

Lubaszewski, *et al.* in [123] proposed a post burn-in testing for NoC interconnects, which is based on the at-speed functional testing of several 2x2 meshes in an NxN NoC. Their method is also capable of detecting faults between distinct inter-switch channels. However, their method is a post-burn testing which can not be applied in online mode and is suitable only for mesh-like networks.

5.2.2 Router Testing

One of the first works performed for router testing is proposed by Aktouf et al. in [129]. They use IEEE 1149 boundary scan standard (JTAG standard) in order to transfer test data in router ports and as wrapper for a group of routers-under-test. All routers are full scan. In this way, the full scan provides test access to inside of the routers. In [130], the authors propose to use partial scan for each router and a single modified IEEE 1500 standard as wrapper for the whole NoC. In this method, they have used a single wrapper that covers the whole NoC. Liu et al. in [131] and Cota et al. in [132] propose to reuse NoC as TAM for transferring test data to its own routers. For this reason they assume that the links between routers and network interfaces are tested before and they are fault free. Amory et al. [133] propose a scalable methodology for testing NoC routers by using scan chains and a specific router configuration during test so that the same set of vectors can be broadcast to all routers while responses can be compared within the chip. Grecu et al. [134] propose two schemes for testing the combinational blocks of the NoC routers based on unicast and multicast transmission of test data packets through the switching fabric.

The above structural testing approaches are not at-speed and cannot be applied in online mode and are not suitable for detecting delay faults.

Stewart and Tragoudas [135] proposed a functional fault model and a functional test strategy for routers and NIs of regular NoCs with a grid topology.However, their fault model is not at the gate level and they do not provide any data on low level fault coverage. More recently, Zheng et al. [136] proposed an accelerating technique for a functional test scheme. The test strategy is based on the technique proposed by Raik et al. [137] and assumes an external tester and the definition of several test path configurations to exercise all possible communications in the network. Lubaszewski, *et al.* in [124] proposed a post burn-in testing for both links and routers, which is based on the functional testing of several 2x2 meshes in an NxN NoC. However, both these techniques are offline and suitable for manufacturing testing using ATE (Automatic Test Equipment), and to get a reasonable fault coverage and implementing them in hardware leads to overhead.

Lin et al. in [138] propose a built-in self-test and self-diagnosis architecture to detect and locate faults in the FIFOs and in the MUXes of the switches. Unfortunately, they do not cover the control path of the router.

A. Strano et al. in [101] propose a built-in self-test/self-diagnosis procedure at start-up of the NoC by exploiting the inherent structural redundancy of the NoC architecture in a cooperative way. They detect faults in test pattern generators and achieve a high coverage. However, their approach is not online, and moreover, they do not discuss about the format of test patterns to obtain a good fault coverage. In addition, they do not consider the reliability of testing components.

5.3 NoC Fault Space Model

In this section we describe the fault models we used for this work. We classify faults into two groups: logic faults and delay faults. Both of these faults may lead to a system failure which shows itself in the NoC's operation.

5.3.1 Logic Fault Model

We define logic fault as a hardware failure in different locations of the NoC. This hardware failure may result in a completely wrong behavior in the NoC functionality. To demonstrate the effectiveness of our testing

mechanism, we consider realistic gate level fault models. We consider two types of gate level faults: stuck-at faults, and bridging faults.

Stuck-at Fault: Stuck at faults are the most common logic faults at the gate level. They are actually shorts to VDD (stuck-at-1) and shorts to Ground (stuck-at-0). Both routers and links are subject to these types of faults. Any wire in the links is susceptible to stuck-at faults. Stuck-at faults can occur in any component in the router including FIFOs, routing logics, MUXes, arbiters, and wires. In this work we cover stuck-at faults in both routers and links.

Bridging Fault: Bridging faults can occur mostly on the inter-router links as well as the internal wires of the router. They are shorts between wires of a link. These types of faults are different from stuck-at faults and need different test patterns to be detected. In this work, we also cover bridging faults in links.

5.3.2 Delay Fault Model

Delay fault is not a logic failure, but a timing failure where a combinational path between two stages of flip-flops has higher delay than expected. The delay fault shows itself only at speed while design is working with the target clock frequency. Delay faults are mostly due to aggressive place and route [16], circuit aging, PVT variations, and transistor wearout [139, 140]. Figure 5.1 shows a delay fault on an inter-switch wire (victim) caused by the cross-talk coupling effect of an aggressor signal (e.g. clock).

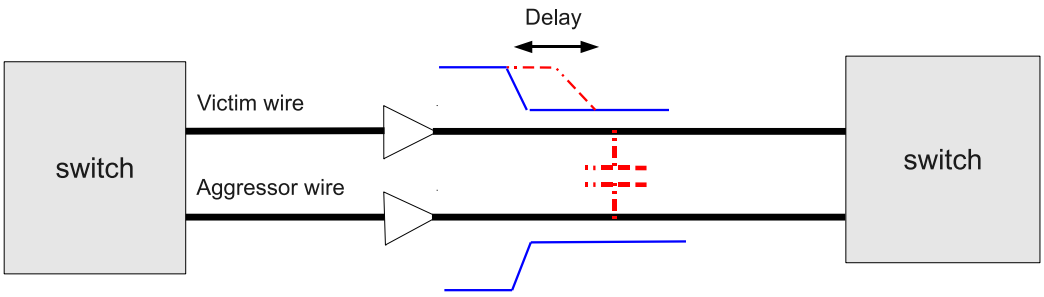


Figure 5.1: Delay fault due to the cross-talk coupling

Scaling down of process technologies has increased process variations and transistor wearout. Because of this, delay variations increase and impact the performance of the design. Delay faults in deep sub-micron are one of the primary reasons of system failure and need to be detected [141]. At-speed functional testing is one of the well-known methods to detect de-

lay faults [142] and we are going to use this technique in our NoC testing approach.

There are two classical fault models that have been developed in recent past to represent delay defects: transition delay fault, and path delay fault [143]. Transition delay fault assumes a large delay defect concentrated at one logical node such that any signal transition passing through that node will be delayed past the clock period. The transition delay fault model may miss the distributed and small delay defects in a combinational path which is more common in deep sub-micron than having a large delay on a single node. The fault list and coverage metrics of the transition delay faults are similar to those of stuck-at faults. Therefore, the stuck-at fault tools can support transition delay faults with minor modifications [144].

On the other hand, path delay assumes a distributed delay along the combinational path. A path is a sequence of connected gates between two stages of clocked elements or between IOs and flip-flops. A path delay fault is said to have occurred if the delay of a path (including the delay of wires and gates) is more than the specified clock period of the circuit. The number of paths in a circuit grows exponentially with circuit size, and hence, typical circuits contain millions and millions of paths making it impossible to test all the paths in the circuit. Table 5.1 shows the number of faults in various fault models inside different NoC switches (MxN switch has M input ports and N output ports). As can be seen, the number of path delay faults is much more than stuck-at and transition faults.

Table 5.1: Number of faults inside different switches

	# of faults		
Switch size	*Stuck-at*	*Transition delay*	*Path delay*
5x5	17080	14880	$8.6 * 10^5$
8x8	35546	32223	$1.5 * 10^6$
10x10	60608	55344	$> 2 * 10^6$
12x12	100257	92452	$> 2 * 10^6$
15x15	165335	155379	$>> 2 * 10^6$

Due to large number of timing paths, a practical approach to detect path delay faults is identifying a set of critical paths using a timing analyzer tool and applying functional test vectors to the design and measuring the output using operational clock frequency [144]. We use a similar technique in this work .

5.3.3 Fault Location

Considering a NoC as a network of routers and NIs, a logic or delay fault can occur in different places including: routers, links, and NIs. In this work we focus on the routers and links. We assume that NIs are already tested or can be tested together with cores. Faults in the links can be detected using a small number of test patterns. However, to fully test the routers several test patterns are needed. Faults in router can occur in two main parts: data-path and control-path. Data-path contains flip-flops in FIFOs, router wires, and output MUXes; control-path includes routing logics, FIFO's control part, and arbiters. In this work we cover faults in both control and data paths.

5.3.4 System-Level Failure Modes

As our approach is at-speed functional testing, we need to define some system level failure modes to better tune the test patterns which can give us higher gate-level fault coverage. Note that, these system level failure modes help us only to generate better test patterns, but the quality of testing approach is measured considering gate-level fault coverage.

Gate level faults may have different effects on the functional behavior of the system. They can put the system into a failure mode. In case of routers and links, as proposed in [145], we categorize the system level failures into four modes: dropped data, corrupted data, misrouting, and multiple copies. We will describe each failure mode in more details in the following.

Dropped Data: In this failure mode the switch receives the data, but never sends it to the intended output port. Dropped data failure can cause by any fault in the control-path of FIFOs, arbiters, or multiplexers of the switch. For instance, consider a case where the head counter of the FIFO is faulty and its current value is increased by 2 or more each time a buffered flit is being removed from the head of FIFO to be sent to its appropriate destination. In this situation, some of the flits in the FIFO will be lost and never exit the switch. Faults in arbiters may cause the select signals of the output multiplexers to not be activated and, therefore, the flit will be dropped. If the output multiplexer is faulty, e.g., one or more bits of its select signal are stuck at one or zero, one of its unintended inputs is selected and consequently the original flit will be dropped. Also, a delay fault on the output multiplexers or on the inter-switch links will lead to a dropped data failure. Detecting faults related to this failure mode is not easy and requires several test patterns probing all possible combinational

paths.

Corrupted Data: In this failure mode the switch receives the data, but it corrupts the data and sends it to the intended output port. This failure mode can also happen on the links. Any logic fault in the data-path of FIFOs (flipflops), multiplexers, links, and internal wires may lead to this failure. As this failure mode can only be created by logic faults in the data-path of the switch and links, faults related to this failure are easier to detect compare to those related to the dropped data.

Misrouting: Misrouting is the result of the faulty behavior of the switch router. Assume that the router has a faulty behavior and makes wrong decisions while routing its incoming data. As these faults happen in the control path of the switch, several test patterns are needed to cover all possible routing paths. Logic faults in the header bits of the FIFOs which are related to the destination address can also create this failure as they may change the intended output port. However, these faults are already covered using the patterns related to the corrupted data failure.

Multiple Copies: Multiple copies in time failure are originated from faults in FIFOs control-path. Consider a FIFO with a faulty "empty" signal. The false empty signal may cause the FIFO send its old data out of the switches port. This old data is usually an earlier packet's flits, and sending the same sequence of flits out of the switch's ports leads to repetition of a packet, i.e., multiple copies in time. The faults related to this mode can be detected easily using Test Response Analyzer (TRA) which checks the order of received flits in a test packet.

5.4 The Proposed NoC On-line Testing

We propose an at-speed functional testing technique which is capable of detecting logic and delay faults in routers and links of any network with minimum overhead on NoC performance. Figure 5.2 on the following page shows the overall architecture of our on-line NoC testing on a Mesh.

As can be seen, in our approach only the Router Under Test (RUT) and its links will put in the test mode, while other parts of the network are in the operational mode. All the packets that want to traverse through RUT will be held in the RUT's neighbors until the test is finished. This may have a little latency degradation for some packets. However, we use a token-based technique to make sure that only one router is under test at any given time. In other words, two or more routers cannot be put in test

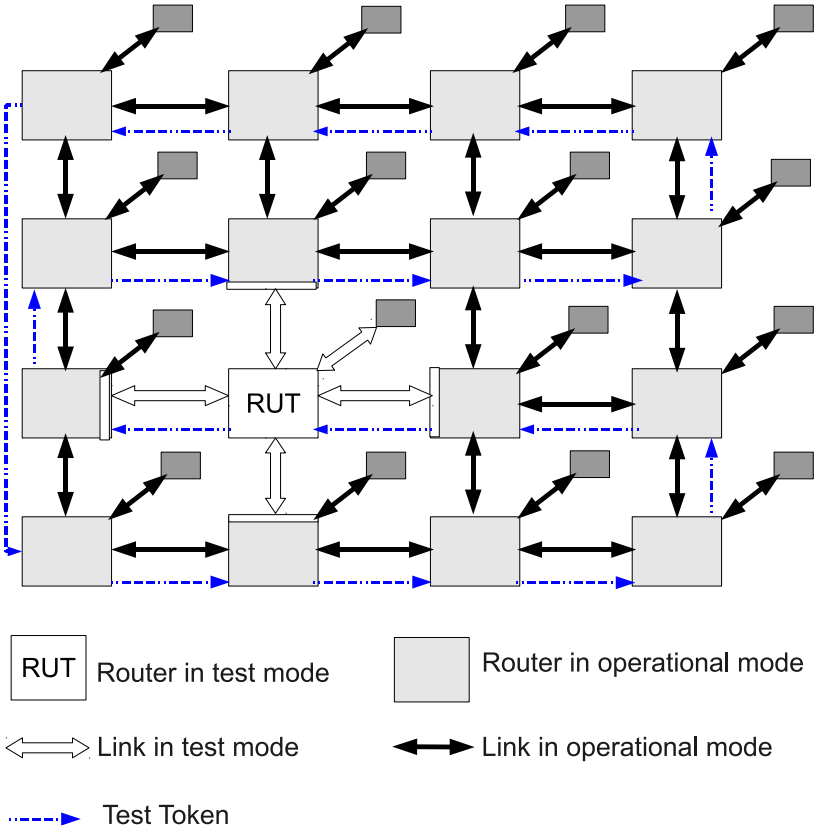


Figure 5.2: Overall architecture of our on-line NoC testing on Mesh. Each router which gets token can start testing itself with the help of its neighbors. Only one router and its links are in the test mode and others are in the operational mode.

mode simultaneously. Token is simply one bit traversing in the network. When a router receives the token, it decides whether it wants to test itself or not; this can be done using a counter in the switch or a request from the core connecting to it. After finishing the testing, the current switch passes the token to the next switch. As token is only one bit, we can use TMR for its reliability to make sure all the routers receive the token correctly.

When a router is in test mode, it should be tested by its neighbors. Each router contains three main test blocks: *Test Pattern Generator* (TPG), *Test Response Analyzer* (TRA), and *Fault Diagnosis Module* (FDM). TPG is responsible for generating test patterns and sending them to the neighbor router which is under test (RUT). TPG of the local port which is inside the router under test's NI sends test packets to the neighbors. Test response analyzer receives test patterns from the neighbor that is under test and

analyzes them to detect any fault in the corresponding router and links. TRA of the local port which is inside router under test's NI, receives test packets sent by neighbors and verifies them. Fault diagnosis module gets the status signals from the neighbors' TRAs as well as the local TRA and diagnoses the faulty channel. There is only one fault diagnosis module in each router. Since only one router can be tested at any given time, TPG and TRA can be shared between all the ports (except the local port). Therefore, for our testing mechanism, we need one pair of TPG and TRA inside each router and each NI. Also, we need one FDM for each router.

Table 5.2 shows the list of acronyms used during the rest of this chapter.

Table 5.2: Acronyms

Acronym	*Meaning*
TPG	Test Pattern Generator
TRA	Test Response Analyzer
FDM	Fault Diagnosis Module
RUT	Router Under Test
TR	Test Response

5.4.1 Test Architecture for Fault Detection

As mentioned before, in our testing approach each router is tested with the help of its neighbors. Thus, this mechanism is scalable to any size of the network with any topology. The architecture of testing one router with four neighbors and one NI is shown in Figure 5.3 on the following page.

When a router is under test, all its neighbors send the test packet generated by their TPG to it. Also, RUT sends test packet to the neighbors using the TPG inside its NI. This enables us to test the local channels connected to the NI as well. At all output ports (except the local) of the routers, we add a 2x1 multiplexer which selects data from either crossbar or TPG. This additional multiplexer is quite small and we use TMR for its reliability.

TPG generates a sequence of predefined packets based on the test phases. We will describe test phases in details in Section 5.4.3 on page 100. The packets generated in different phases are the same in terms of data flits. The header flit of the packet depends on the test phase. Therefore, the same data is sent to different destinations in different phases. The rationale behind this is that the data flits in each packet are responsible to cover the faults in data-paths and different phases of testing cover control

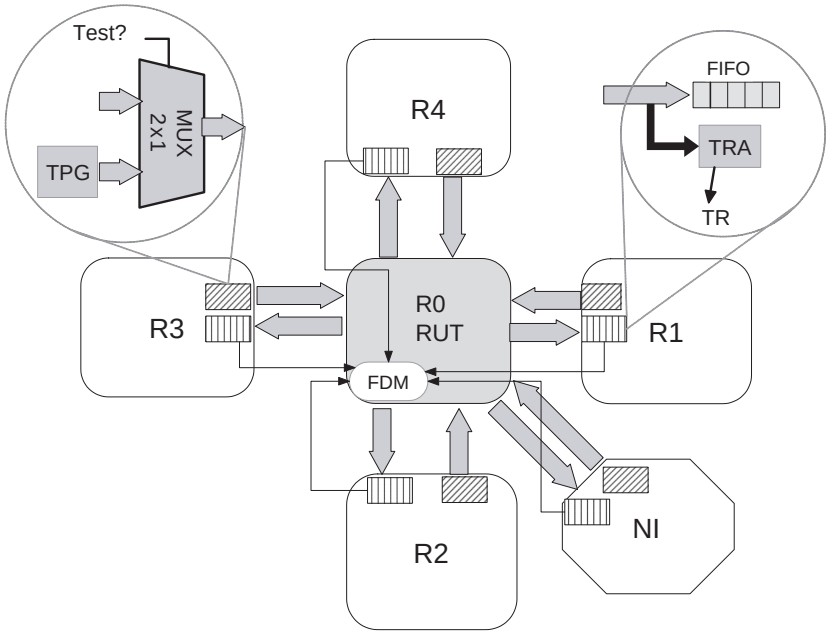


Figure 5.3: Each router and its links are tested using neighbors. (TPG:Test Pattern Generator, TRA: Test Response Analyzer, FDM: Fault Diagnosis Module)

paths' faults. We will describe the format of test packet in more details in the next subsection.

TRA verifies whether the incoming data corresponds to the predefined sequence generated by TPG or not. TRA sends a 2-bit signal (TR) to the RUT which shows the result of pattern checking. RUT receives TR signal from all neighbors and its NI, and diagnoses the faulty channel using Fault Diagnosis Module (FDM) which is described in Section 5.4.5 on page 103. In addition to the corrupted data checking, TRA is able to check the dropped packet failure. In each phase of the test, TRA waits for a specific pattern to arrive. If it receives the packet within a limited number of cycles, it checks the data and sends the results to the RUT. If TRA does not receive any packet, it generates a time-out error. For time-out checking we use a 5-bit counter inside TRA. If the counter reaches the limit, TRA generates a time-out signal (one bit of TR) and sends it to the RUT. Table 5.3 on the next page shows the meaning of different bits of TR signal generated by TRA. Note that, since TR is only 2 bits, parity checks can be used for its reliability with a small area overhead.

Table 5.3: 2-bit TR signal generated by Test Response Analyzer

TR (bits)	*Meaning*
00	No Error - Data is received correctly
01	Error - Data is received but is corrupted
10	Error - Data is not received (Dropped)
11	No Info - (reserved for unconnected ports)

5.4.2 Test Packet Format

As mentioned earlier, TPG generates the same packet targeting different destinations at different phases during test of the router. The format of the data flit in the packet is chosen so that it covers all stuck-at and bridging faults in the data-path which includes links, FIFO's flip-flops, switch internal wires, and output MUXes. This format is independent from the topology of the network and from the RUT's number of ports.

Two data flits of All-One and All-Zero are enough to cover all stuck-at faults in the data-path. However, to detect all bridging faults, we need a set of Walking-One data flits[2]. Therefore, the number of data flits required for bridging faults is equal to the flit-width i.e. 32 in our switch. As a result, to cover all stuck-at and bridging faults in the data-path we need at least 34 data flits. If we target only stuck-at faults, the number of data flits will decrease to 2. This can help to reduce the overhead of both TPG and TRA and to decrease the latency overhead as well. We will show this trade-off in more details in experimental results section. Figure 5.4 on the following page shows the packet format which is generated at each phase of testing in TPG. Note that, in TRA the same packet format is verified at each phase of the testing.

Since all packets are sent with headers and tails, faults may affect those flits, thus modifying the packet routing. When this occurs, the packet can be routed to any other node of the NoC or a tail flit may not be received. In both cases, TRA at the target node will not receive the packet or the tail within a predefined time interval, thus reporting a time-out error.

Since the network is in the normal mode of operation during the test, one might consider the possibility of other errors caused by the logic fault, such as a deadlock, in addition to packet losses and payload errors. We note that a fault cannot cause a deadlock within our test configuration because of the packet format and paths established during testing.

[2]Walking-One refers to a set of data patterns where, in each data word, a single bit is set to one and the rest to zero.

Figure 5.4: Format of test packet generated by TPG and verified by TRA at each phase of testing. This format is fixed and independent from network topology.

5.4.3 Test Phases

As described earlier with the specified packet format we are able to detect all stuck-at, delay and bridging faults in the data-path traversed by the packet. However, to provide coverage for faults in the control path we need different test rounds or phases. Each phase is responsible to cover some of the combinational paths inside the router while the data path is being tested multiple times. This phases depend on the network topology and the RUT's number of connected ports. In the following, we describe the suitable phases for a router with four neighbors and one NI. In Section 5.4.6 on page 107 we discuss how we can modify them for other topologies.

For a router with four neighbors and one NI, we consider 9 different phases to cover all the control paths including different routing, arbitration and FIFO's control. Figure 5.5 on the facing page shows these phases. Arrows in the figure show the source and destination of the test packet. At each phase, one test packet is sent from source to the destination through RUT (R0 in this figure) and its links. Those test packets that come from the neighbors are generated at the output port of the neighbors using their TPGs; while, packets whose sources is RUT are generated in the TPG of the RUT's NI. Therefore, packets originated from RUT also cover the local input channel. Similarly, those packets targeting the neighbors are verified immediately at the input port of the neighbor and before the FIFO. While packets whose destinations is RUT are verified in the RUT's NI. Thus, packets targeting RUT cover the local output channel as well.

As can be seen in Figure 5.5 on the next page, test phases are chosen in such a way that they cover all functional failures of the router related to the Misrouted Data, Dropped Data, and Multiple Copies failure modes which are the results of logic faults in the RC, FIFO's control path, or the arbiter.

Since for each input port there could be 4 different destinations, we need atleast 4 test phases to fully cover all the routing logics. Test phases 1 to 4 cover all different routing possibilities inside the switch without

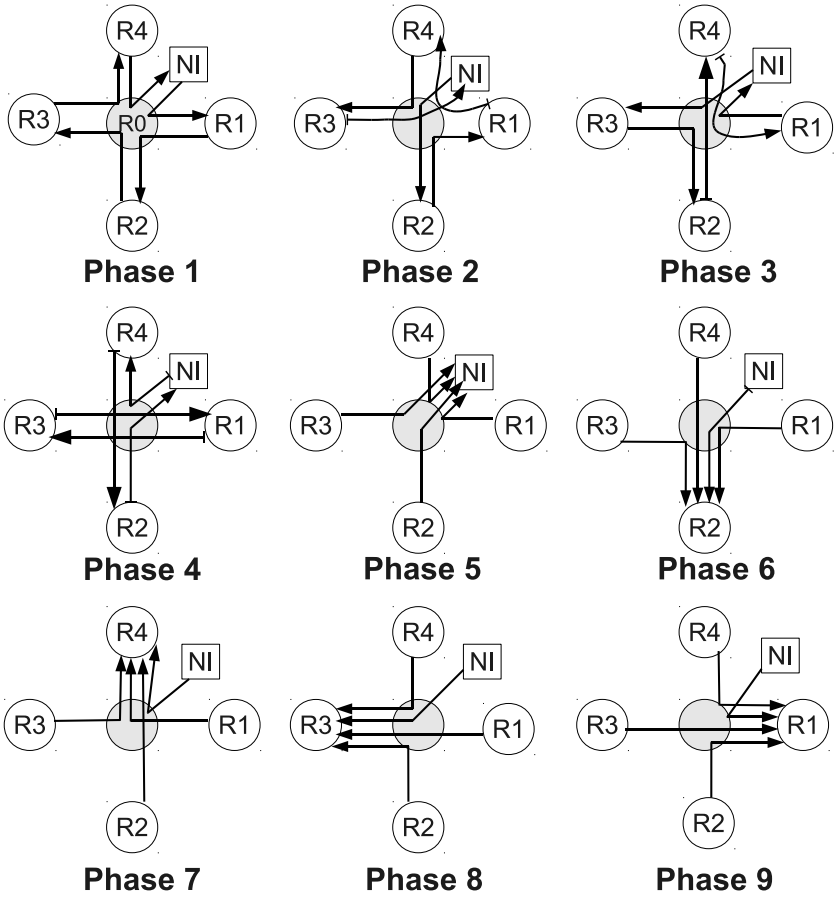


Figure 5.5: Nine test phases for a router with 4 neighbors and one NI. Phases 1 to 4 cover data-path and all routings without any arbitration. Phases 5 to 9 cover arbiters as well as FIFO's control logic.

any arbitration policy. These test phases (1 to 4) are chosen so that we can test all RC blocks inside the router with a minimum number of tests. However, in phases 1 to 4 the packet's destination (stored in header flit) is chosen so that for each output port of the RUT there is only one request. Thus, although some parts of the arbiter are being tested during these phases, these phases (1 to 4) are not sufficient to test the functionality of the arbiters. Also, they cannot fully test the control logic of the FIFO. The FIFO's control logic decides based on the arbiter grants. During these test phases the arbiter grant is always given to the requester immediately and, therefore, the output's busy signal which goes to the FIFO's control logic is always zero. Thus, in these phases FIFO's control logic always pop the data out of the buffer without waiting. We need more test vectors to

put the FIFO in the waiting state.

Both arbiter an FIFO control block can be covered by a set of test packets targeting the same destination. As we have five output and input ports, we need at-least 5 test phases to cover all arbiters and FIFOs. Test phases 5 to 9 are created to perform this task. At each of these phases there are four requests to the related arbiter. We should note that, there can be some other phases to completely test arbiters where there are two or three requests to each arbiter. However, we ignore since sending four (maximum) requests to each arbiter can test most of its functionality. In Section 5.4.6 on page 107 we show a general formula of the number of phases based on the number of ports.

As can be seen in Figure 5.5 on the previous page, at each phase of this set (phases 5 to 9) one arbiter together with 4 FIFOs are tested. At the end of these phase, each arbiter is tested once and each FIFO (both control and data paths) is tested four times. These test phases (5 to 9) are mainly responsible to cover the Dropped Data, Misrouting and Multiple Copies failure modes. In the TRA, at each phase we check whether all the packets are arrived or not. In these phases of the test, TRA should receive 4 consecutive packets from four different sources provided that there were no time-outs in the first 4 phases. If it does not receive the related packets it means that the arbiter has not given the grant to one of the requests and TRA generates time-out error. We note that, if there are time-outs in the first 4 phases, TRA does not check it again in phases 5 to 9 and only validates those packets that are supposed to arrive.

5.4.4 Delay faults

Since our testing technique is at-speed, it can detect the delay faults in the functional paths. Delay faults can happen in both control and data paths. In the following, we describe how our technique handles delay faults in different functional paths.

1) *Control paths*: Based on our timing analysis there are two critical control paths in our switch: (i) request to the arbiter for an output port (ii) grant from the arbiter.

Request paths to the arbiter: If there is a new packet, a request signal is sent to the related arbiter from routing computation block or from the FIFO controller (depending on the implementation). If a delay fault occurs on the path related to request signal, it does not effect the functionality of the router since the arbiter will get the request one clock cycle later. This is because the request signal remains high until the grant is given by the arbiter.

Arbiter's grant paths: If a grant is given by the arbiter, the related FIFO controller pops out the flit from the buffer and the arbiter will drive the appropriate signaling of the crossbar to send the flit out of the switch. Here, if a delay fault occurs on the path from arbiter to the crossbar, the related flit is not sent out of the switch and since it is already popped out of the buffer, we miss that flit. This delay fault will eventually lead to a dropped packet failure which can easily be detected in TRA.

Other control paths inside the switch are not critical enough to be considered for the path delay fault testing. However, if a huge delay fault occurs in these paths and leads to a functional failure, the packet may be either dropped or misrouted and the fault will be detected in TRA.

2) *Data path and links*:

If a delay fault occurs on the router's data-path or on the links, depending on the path where the delay fault occurred the receiver may or may not capture the right data. Figure 5.6 on the following page shows the timing diagram related to delay fault in data-path and links. As can be seen, if the same delay fault occurs on both control signal (*valid*) and *flit* bus the receiver is able to capture the right data one or more cycle after the expected cycle. In this case, since TRA waits for the expected packet to arrive, it can get the data eventually (before the timeout) and does not generate any error. However, if the delay occurs only on *valid* signal or only on *flit* bus, as shown in the figure, it leads to a dropped packet failure which can be detected in TRA. (Note that, we say a delay fault occurs on the *flit* bus if there is at-least one delay fault on the bus's bits.)

This timing behavior is valid for both input and output links of RUT. For input links, the receiver is the RUT buffer and fault may occur on input wires; for output links the receiver is the TRA of the related neighbor and fault may occur either on the data-path or on output wires of the RUT.

5.4.5 Fault Diagnosis

Using TPG we generate test packets in different test phases and in TRA we verify the incoming packet and detect if there is any error in it (either corrupted data or misrouted). TRA sends the results of checking to RUT using 2-bit TR signal. However, we still need a mechanism to diagnose the location of the fault. We perform this using Fault Diagnosis Module (FDM) inside RUT.

For data-path faults which are related to the Corrupted Data failure, FDM is able to diagnose fault location in the input or output channels of RUT. We define input channel as the data-path covering the input link and the FIFO, and the output channel as the path covering internal link, output

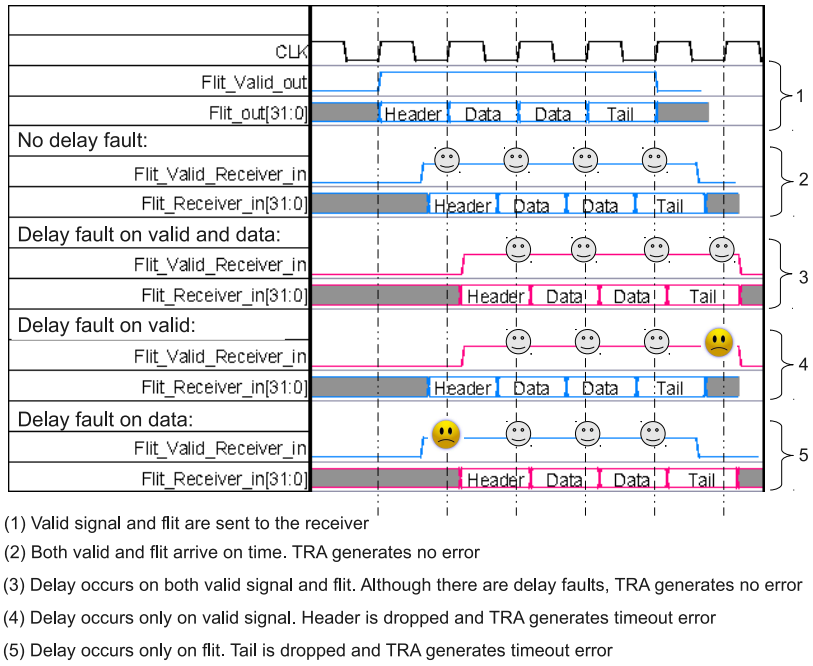


Figure 5.6: Detecting delay faults occurred in links and the router's data-path .

MUX, and the output link. Figure 5.7 shows input and output channels for West and East ports. Using 2-bit TR signals, FDM is able to find out which channel (either input or output) is not faulty.

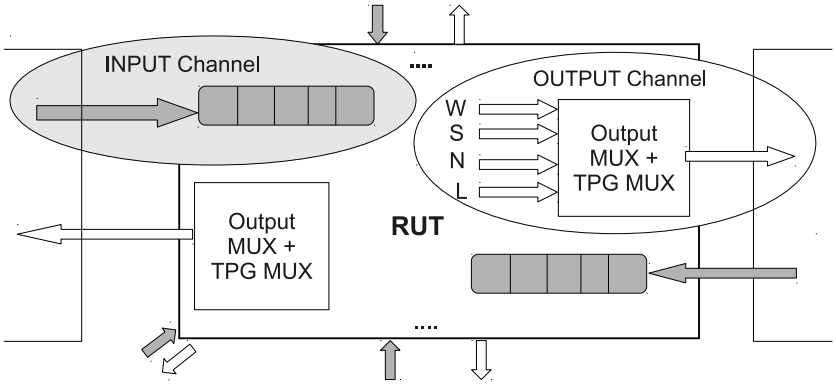


Figure 5.7: Channels in which Fault Diagnosis Module is able to diagnosis data-path faults.

For a 5x5 router, we have a 10-bit register which shows the status of each channel; we name this register CSR (Channel Status Register). Each

bit of CSR corresponds to one channel as shown in Figure 5.8. FDM receives TR signals and in each phase of the test updates CSR. FDM is based on finding non-faulty channels.

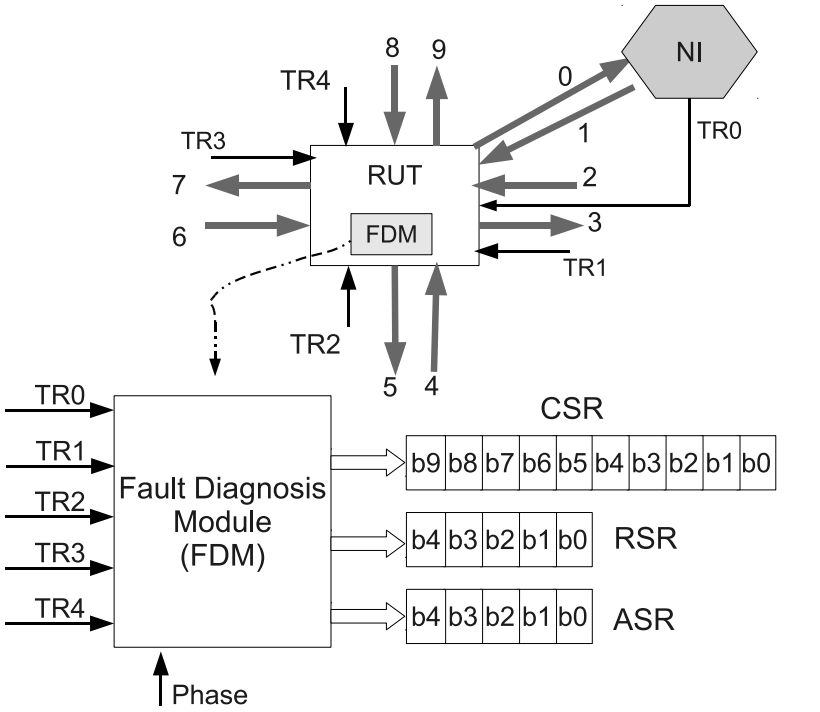


Figure 5.8: FDM gets TR signals and updates CSR, RSR, and ASR.

As seen in Figure 5.5 on page 101, each test packet covers 2 channels considering local channels. Thus, if a TR signal shows a corrupt error (01), FDM can not diagnose which of the two channels is faulty. However, if a TR signal shows no error (00), FDM can diagnose that both channels are non-faulty. At the beginning of the test, FDM sets all the bits in CSR to '0' meaning all channels are faulty. During the test, when FDM receives a TR signal (from either any neighbor or its NI) which shows no-error (00) it sets the corresponding bits of the CSR to '1' which are related to the channels in which the packet has traversed meaning they are non-faulty. If the bit is already set to '1', it does not change it. At the end of the testing, those bits of CSR which are '1' show the non-faulty channels. FDM is a simple combinational block which decides based on the TRs and the testing phase.

FDM is also able to diagnose which RC or arbiter is faulty. We have two 5-bit registers showing the status of RCs and arbiters in a 5x5 router; we call these registers RSR (Routing Status Register) and ASR (Arbiter Status Register) respectively. At the beginning of the test FDM sets all bits of both registers to '1' meaning all the RCs and arbiters are non-faulty. During the first 4 phases, if the FDM receives a TR signal of dropped error (10) it sets the corresponding bit in RSR to '0' which is decided based on the test phase. During phases 5 to 9, if FDM receives a TR of dropped error it sets the corresponding bit in ASR to '0'. As mentioned earlier, if TRA detects a time-out error in the first 4 phases, it does not check it again; therefore, if there is a fault in one RC, it does not have any effect in the results of phases 5 to 9. Thus, if TR shows a dropped error (10) in phases 5 to 9 it is related to the arbiter but not to the RC.

Distinguishing Delay faults

During the first four phases of the test, if FDM detects a time-out error, the failure is caused by either a delay fault in router/links or a logic fault in RC. We cannot differentiate these faults from each other using the current test pattern and phases. This is because both these faults (delay fault in router/links and logic fault in RC) leads to a dropped packet failure mode. Therefore, with the current test pattern we are only able to detect delay faults, but not to diagnose their locations.

To distinguish delay faults from logic fault in RC we can use a dedicated test packet and repeat all nine phases with this new packet. To decrease the testing time, we can repeat only those phases that did not pass the test. This requires a modification in TPG, TRA, and FDM to store the pass/fail information. The new packet is formatted so that it can bypass the delay faults, but is still vulnerable to the logic faults. Figure 5.9 shows the format of the new packet which can bypass the delay faults in data-path or links.

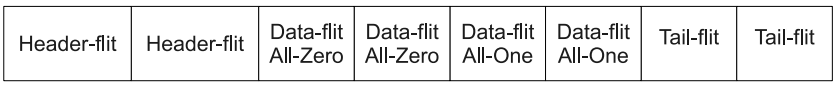


Figure 5.9: Packet for bypassing delay faults in data-path and links

As can be seen in the figure, in the new packet each flit is repeated one time. Note that, TPG and TRA should be a little modified to accept two similar consecutive flits during the second round of test phases. Figure 5.10 on the next page shows how this new packet can bypass the delay faults on the *valid* signal and on the *flit* bus.

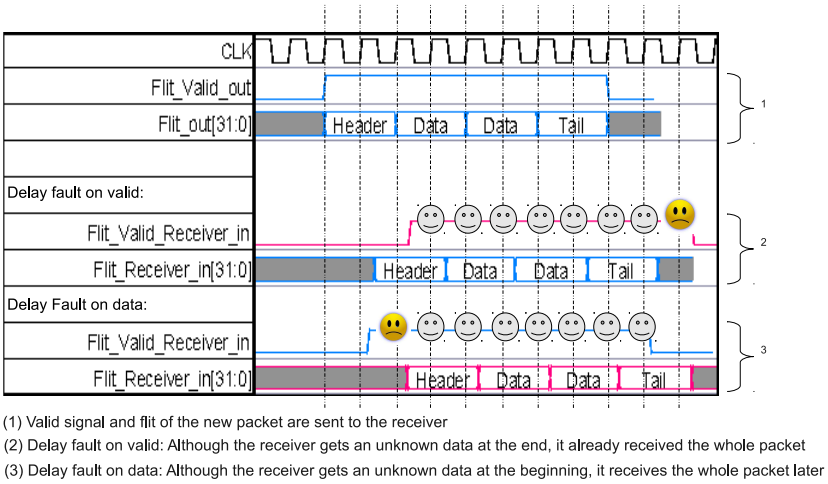


Figure 5.10: Delay faults on data-path and links are bypassed with the new packet

By repeating all nine phases (or just the failed phases) but with the new packet, we are able to distinguish delay faults in data-path/links from logic faults. For example, if in Phase 2 FDM receives a dropped packet signal but does not get it in the second round (Phase 2 with the new packet) it means the dropped packet failure was due to a delay fault in the data-path/links and not to the RC's logic fault. To store the information related to delay faults in each channel, we also need another 10-bit register which stores the delay faultiness status of the channels.

The main reason for distinguishing logic faults from delay faults, especially the delay faults on the inter-switch links, is the possibility to recover from them by post-silicon tuning techniques or by adjusting the clock frequency. An interested reader can refer to [146, 147] for more information about tuning techniques.

5.4.6 Testing of Custom Topologies

Our testing methodology is based on testing each router using its neighbors and the NI connected to it. Each router receives TRs from the neighbors as well as its own NI and updates the related status registers. Considering the switches at the boundary of the mesh, there are one or two ports which are not connected to any neighbor. We connect TRs of those ports to "11" meaning no information is available on those specific ports. FDM module does not update anything based on the TR of those unconnected ports. On the other hand, neighbors that are connected to the boundary

switches expect test packet from those ports of the RUT which are not connected to any other router. Therefore, they generate time-out error for those specific ports and RUT sets the related bits of RSR to '0' meaning those ports that are not connected to anywhere are faulty. This does not have any effect on the functionality of the router as those ports are not used. Based on this mechanism, our testing technique can be used for any topology and any routing.

TPG and TRA modules should be modified based on the routing algorithm. TPG should send test packet to all output ports of the RUT in different phases. If the routing algorithm and the maximum number of ports in the switch are known, TPG and TRA are the same for all routers and independent from the topology. However, they can be optimized based on the NoC topology to reduce their hardware overhead. For example, TPG does not need to send the test packet to those ports of the RUT which are not connected to any other router. Note that, sending test packet to those ports does not have any effect on the related bits of status registers because the TRs of those port are already connected to "11". Also, TRA does not need to check the time-out error for those input ports of the RUT which are not connected to any other routers. Moreover, CSR, RSR and ASR can be optimized based on the number of connected ports.

As mentioned earlier, test phases are chosen so that they can cover all the functional paths of the router. They depend on the number of ports in each router. Phases shown in Figure 5.5 on page 101 are suitable for maximum 5 input and 5 output ports. However, if a router inside the NoC has more ports, we need to define more test phases to cover those additional ports. Note that, for switches with less than 5 ports, the phases shown in Figure 5.5 on page 101 are more than enough and may be reduced to a smaller number for optimizing test time and hardware cost. For instance, Figure 5.11 on the facing page shows the optimized test phases for a 3x3 switch.

To give a general formula on the number of test phases based on the number of ports of the switch, we divide phases into two groups: (i) phases for RC testing (ii) phases for arbiter and FIFO control testing.

Let say N is the number of input/output ports inside the switch. Since each port can send data to all other ports, we need $N-1$ phases to test each RC component. RCs can be tested in parallel and therefore $N-1$ phases are enough for testing all RCs inside the switch.

For the arbiters, the situation is different. Each arbiter can get the request from all other ports; since we have N ports, each arbiter may receive a number of requests in the range of $[1..N-1]$. For 1 request, it is already tested in the previous phases. However, we need to test other situations.

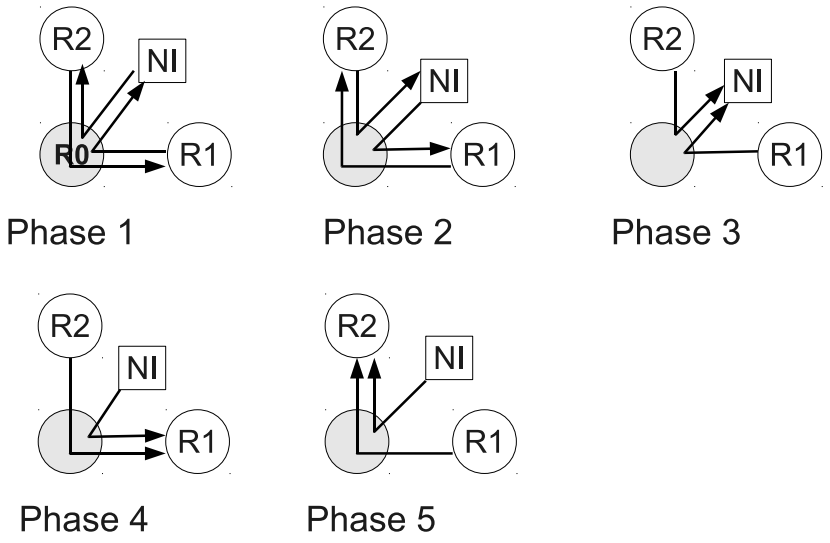


Figure 5.11: Test phases for a router with 2 neighbors and one NI. Phases 1 and 2 cover data-path and all routings without any arbitration. Phases 3 to 5 cover arbiters as well as FIFO's control logic.

For each number of requests $K \in [2..N-1]$, K of the input ports send requests to the arbiter. Therefore, we have a *binomial coefficient* $\binom{N}{K}$ for K requests to each arbiter. Thus, the number of test phases for each arbiter is C where C is the sum of all binomial coefficients as follow:

$$C = \sum_{K=2}^{N-1} \binom{N}{k}$$

Arbiters cannot be tested in parallel unless we have a complex control logic, and we consider testing them sequentially. Therefore, the number of test phases for all arbiters is $N * C$ and the total number of phases for the router is $M = (N - 1 + N * C)$. Note that, during all these phases links, the data-path, and control logics of FIFOs are tested multiple times.

5.5 Experimental Results

To evaluate our on-line testing methodology, we used Xpipes [56] which is a packet-switching synthesizable NoC IP. The switch is configured to have 5 input and 5 output ports with the flit width of 32 bits and the buffer

depth of 2. We used Synopsys design compiler for hardware synthesis and Synopsys Tetramax for logic fault simulation. All designs are mapped on CMOS 45nm technology from ST-Microelectronics.

5.5.1 Hardware Overhead

First, we implemented both TPG and TRA in Verilog to see their hardware overhead on a 5x5 router. As mentioned earlier, we can have one TPG and one TRA for the whole switch. We also need one pair of TPG and TRA for each NI. However, TPG and TRA of the NI are simpler than those of the switch because the sequence of test generated/verified by TPG/TRA inside the switch depends on the port while the sequence of those inside NI is fixed. As we mentioned earlier, TPG and TRA are the critical components and we need to use robust techniques to make them reliable. Since these module are quite small with respect to the switch and NI, we can use TMR for this purpose.

We implemented two kinds of TPG and TRA. One pair with all-one, all-zero, and walking-one sequence and another pair without walking-one sequence to cover only stuck-at and delay faults. The synthesis results are shown in Table 5.4 and Table 5.5 on the facing page.

As can be seen, if we want to cover stuck-at, delay and bridging faults in the 5x5 router the area overhead of all TPGs and TRAs with TMR is about 58% which is pretty high. This is due to the long walking-one sequence which needs a shift register. However, if we target only stuck-at and delay faults the area overhead with TMR is 16% which is acceptable considering TMR support for reliability.

Table 5.4: Synthesis results of TPG and TRA with walking-one sequence (Area of 5x5 Switch + NI = 8766 μm^2)

Module	*Area* (μm^2)	*Area* with TMR (μm^2)	*Area (with TMR) Overhead on 5x5 Switch + NI*
Switch TPG	413	1280	14%
NI TPG	363	1119	12%
Switch TRA	472	1463	16%
NI TRA	410	1270	14%
All TPG + TRA	1658	5132	58%

We also implemented the Fault Diagnosis Module (FDM) together with status registers in Verilog and synthesized them. Based on our synthesis results, the area overhead of the FDM module with TMR is 10% on the

Table 5.5: Synthesis results of TPG and TRA without walking-one sequence (Area of 5x5 Switch + NI = 8766 μm^2)

Module	*Area* (μm^2)	*Area* with TMR (μm^2)	*Area (with TMR) Overhead on 5x5 Switch + NI*
Switch TPG	121	375	4%
NI TPG	73	240	2.5%
Switch TRA	140	462	5%
NI TRA	117	366	4%
All TPG + TRA	451	1443	16%

Xpipes switch. Considering NI and switch together, the area overhead of FDM with TMR is only 7%. Therefore, the area overhead of all modules including TPGs, TRAs, and FDM with TMR support is 65% to cover stuck-at, delay and bridging faults and 23% to cover only stuck-at and delay faults.

Note that, the switch model we used is very small without any output buffer and with an input buffer depth of only 2 flits and its area is much less than that of the IP cores. For instance, compare to Plasma [148], a small synthesizable 32-bit RISC microprocessor, the switch area is 1/5 of the processor area. Therefore, the overhead on the whole NoC including IPs is very small. Considering switch, NI and a small sized IP core the area overhead of all modules with TMR is 21% to cover stuck-at, delay and bridging faults and 8% to cover only stuck-at and delay faults.

Test units with their TMR support may have effects on the maximum operational frequency of baseline router. The Xpipes switch is a soft IPs and the timing impact of our testability extensions depends on how it is instantiated. If we do not use any output buffer, the critical path of the router is from the input buffers through output ports and links. As already mentioned and shown in Figure 5.3 on page 98, TPG and TRA operate in parallel with the router and their critical paths are very small, hence they do not decrease the speed of the router. However, as shown in Figure 5.12 on the following page, in case of no output buffers MUXes that are added to the output ports with their related TMR are on the critical path of the router. To have a reliable and high-speed TMR, we used the architecture shown in Figure 5.12 on the next page for the voter of all TMRs which is proposed in [149]. We synthesized our testable router and extracted the data related to the delay of the MUX2x1 and the voter. The nominal delay of the path starting from MUX and ending at the output of the voter is

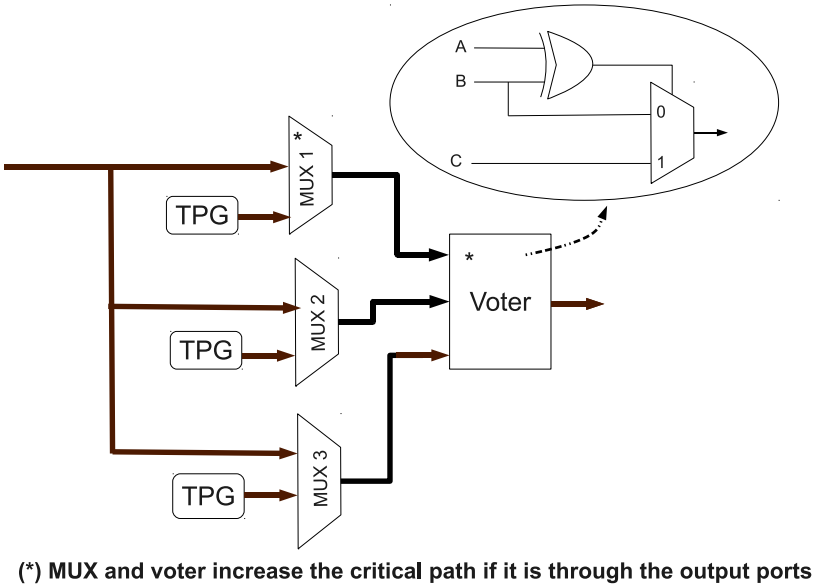


Figure 5.12: TMR for TPG and output port.

390ps. If we consider a baseline Xpipes switch working at 500mHz (critical path = 2ns) with no output buffer, we see that the working frequency of the testable switch will be 418mHz (critical path = 2.39ns) which is a 15% speed overhead.

The no-output-buffer configuration of Xpipes is only for low speed operation. For high-performance NoCs, output buffers are needed to avoid having critical paths through the output links. In this case, adding testability support has minimal overhead. The critical path is from input buffers to the output buffers and the testing components shown in Figure 5.12 are placed after the output buffers. Here the testing components may reduce the clock frequency only if $P_{clk} - D_t < D_l < P_{clk}$, where D_l is the delay of the output link, P_{clk} is the period of the clock without testing components, and D_t is the delay of the MUX2x1 and the voter which is 390ps. We should not that, the TMR support which is added to the output ports is not only for the testing purposes but is also useful during the normal operation of the router and makes the output ports fault tolerant.

Focusing on power consumption, it is clear that all test-mode components (except MUXes and the voter shown in Figure 5.12) are not used during the normal mode of the NoC. TPG and TRA should be turned on only if one of the neighbors is in the test mode, and FDM should be turned on only if the router itself is being tested. Therefore, we use clock-gating to disable switching activities inside these modules during off-state. This

helps us to decrease the power consumption of the router during the normal operation. Table 5.6 shows the power consumption of our testable router when all testing components are active, and Table 5.7 shows the results when all of them are turned off. As can be seen, during the test mode the difference between dynamic power consumption of the baseline router and that of our testable router (with all testing components and TMR) is only 510 μW; this difference for the leakage power consumption is 0.13 μW. As shown in Table 5.7, during the normal mode our testable router has a very small overhead of 90μW on dynamic power and 0.14μW on the leakage power.

Table 5.6: Power results of Testable router in test mode

Power	*Baseline router*	*Testable router in test mode*	*Difference test mode*
Dynamic (mW)	1.57	2.08	0.51
Leakage (μW)	2.15	2.28	0.13

Table 5.7: Power results of Testable router in normal mode (clock gating enabled)

Power	*Baseline router*	*Testable router normal mode*	*Difference normal mode*
Dynamic (mW)	1.57	1.66	0.09
Leakage (μW)	2.15	2.29	0.14

We should again note that our baseline switch is very small and if we consider a medium sized switch with NI and a small sized IP core the power overhead of all testing components is negligible even during the test mode.

5.5.2 Fault Coverage

We used Synopsys Tetramax to evaluate the fault coverage of our testing approach. We synthesized the Xpipes switch and simulated our 9 testing phases on it. Then we collected the corresponding VCD file to perform fault simulation. Table 5.8 on the next page shows the number of stuck-at faults related to each component in the Xpipes switch. As it was expected, the number of faults inside each component is proportional to the area of that component.

Table 5.8: Number of stuck-at faults of different components inside the 5x5 switch extracted from Tetramax

Component	*% of the Area*	*# of Faults*	*% of faults*
RCs	9%	1980	11.5%
Arbiters	20.5%	4342	25.4%
Buffers	51.5%	5820	34%
Output MUXes	19.5%	4722	27.6%
other gates	0.5%	80	0.1%
Entire Switch	100%	17080	100%

We applied the VCD vectors obtained from the simulation of 9 phases on the synthesized design in Tetramax to see the fault coverage of the vectors. Figure 5.13 shows the related results. These results are for stuck-at faults. Note that, for bridging faults, the test vectors give us 100% coverage on the links in the first 4 phases. As can be seen, the stuck-at fault coverage of the entire switch for the first four phases of the test is 68% which is not quite good. After applying the next 5 phases (total 9 phases) we could improve it to 85% which is an acceptable coverage for a functional test without any traditional test technique like scan.

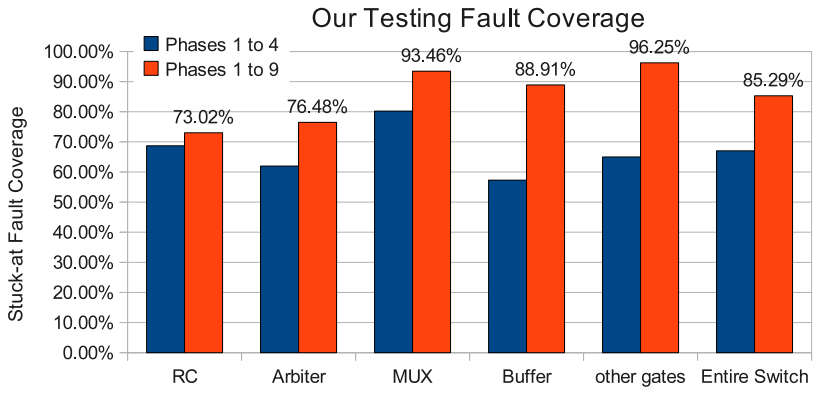


Figure 5.13: Stuck-at fault coverage of our on-line functional test approach.

Using the same VCD vectors, we achieved 67% coverage for the transitional delay faults. This is because not all the transitional delay faults lead to a functional failure, and the coverage for transition delay faults is always less than that of stuck-at faults.

For the path delay faults, we first imported the synthesized netlist of the switch into PrimeTime, the static timing analysis tool from Synopsys,

and extracted those paths whose delays were more than 80% of the clock period. Then, we analyzed the extracted paths and found two groups of paths. The first group was related to sending the flit from the input buffer to the output port when the grant is given by the arbiter. As we already discussed in Section 5.4.5 on page 106, delay faults on these paths lead to the dropped data failure mode and we detect them in TRAs. The second group was similar to the first one and was related to shifting the FIFO content when the grant is given by the arbiter and the flit located at the head of the FIFO is sent out from the switch. Delay faults on these paths lead to either a flit duplication or a dropped flit. Both of these failure modes can be detected in TRAs.

Note that, the achieved fault coverage can still be increased by adding to the test phases other network functional configurations (arbitration possibilities that were not applied, for example). However, insisting in a completely functional test approach will require an increasing number of test configurations and phases, for a decreasing number of detected faults. The trade-off seems to be not in favor of pushing further the functional testing approach. From the point of view of a functional test, about 10% of the remaining logic faults are undetectable because they are related to unreachable control states [123].

5.5.3 On-line Testing Evaluation in Simulation

To see the effect of our on-line testing mechanism on the latency of the packets, we performed system simulation. We used Noxim [106], a cycle accurate NoC simulator implemented in SystemC. We extended it for performing our on-line testing approach.

We periodically place a switch under test and hold all the packets traversing it. This is performed by arbiters in the neighbors. They do not give grant to the packets traversing the switch under test, until the test is finished. We compared the average latency of the packets in different synthetic and real traffics with and without on-line testing. We implemented a token-based mechanism in the simulator to avoid having multiple routers under test at the same time. We have a counter inside each router which counts the number of flits forwarded by the switch. When the counter reaches the threshold value if the router has the token it can start testing. We swept the threshold value from 1,000 to 20,000. The simulation has been performed on an 8x8 Mesh with XY routing algorithm.

Figure 5.14 on the following page shows the average packet latency of different synthetic traffics in the NoC while using our on-line testing mechanism. We performed simulation with and without walking-one se-

quence which is for covering bridging faults. As can be seen, when the threshold value of the test counter is low (less than 10,000 flits) the latency overhead is quite high, especially when the test packet includes walking-one sequence. This is due to the fact that with lower threshold values the switches go to the test mode more frequently. However, for threshold values greater than 10,000 flits the latency overhead of our on-line testing is almost negligible.

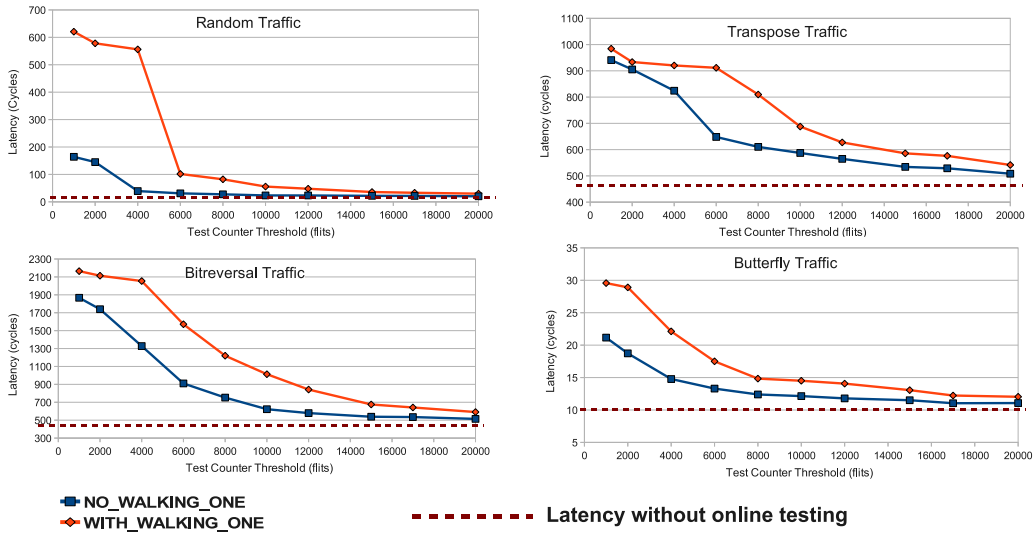


Figure 5.14: Simulation results for our on-line testing approach on different synthetic traffics with various test counter thresholds.

In addition to the synthetic traces, we also performed simulation to see the effect of on-line testing on real traffic traces from the PARSEC benchmarks [107], described in Chapter 4, Section 4.5.5. Like for the previous experiments we swept the test counter threshold from 1,000 to 20,000 flits. We performed the simulation for two types of test packets: with walking-one sequence and without that. The results are shown in Figure 5.15 on the next page. As can be seen, for all benchmarks with a threshold value of 10,000 the latency overhead is almost zero even with walking-one sequence. Based on our experiments, 10,000 flits is an appropriate value for the test counter's threshold for both synthetic and real traffic.

5.6

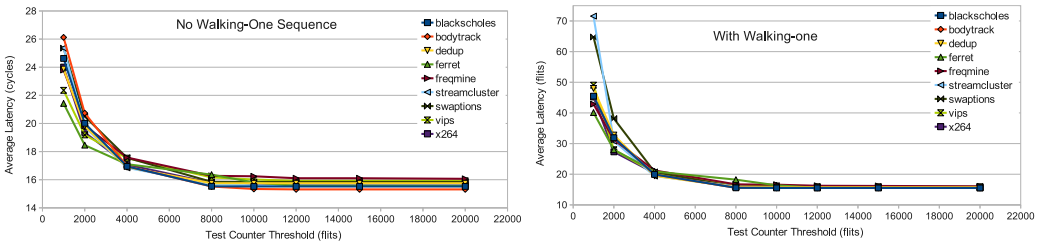


Figure 5.15: Simulation results on PARSEC benchmarks with various test counter thresholds.

Summary

In this chapter, we proposed a distributed functional test mechanism for NoCs which scales to large-scale networks with general topologies and routing algorithms. Each router and its links are tested using neighbors in different phases. The router under test is in test mode while all other parts of the NoC are operational. The proposed testing approach is able to detect both logic and delay faults inside data-path and control path of the NoC.

In the next chapter we direct attention on the intra-cluster communication for cluster based MPSoCs. We will propose an ultra-low latency interconnection network adequate for communications between processors and multi-banked shared-L1 memory.

CHAPTER 6

Low-latency Interconnection Network

So far, we have focused on NoC as an interconnection network suitable for inter-cluster communications among different tightly coupled processor clusters. This chapter[1] gives attention to the intra-cluster communication between the heavily shared multi-banked L1 memory and the cores of each tightly coupled processor cluster. In this chapter, first, a fully combinational Mesh-of-Tree(MoT) interconnection network suitable for shared-L1 processor clusters is presented featuring single-cycle transfer from processor to memory and vice versa. The network provides round-robin arbitration for fair access to memory banks, as well as fine-grained address interleaving to reduce memory bank conflicts. Second, we show an advanced synthesis and physical optimization strategy which leverages standard design implementation tools, and orchestrates them to achieve high-quality results in terms of delay, power and area (DPA) even for large network instantiations. Third, we explore a wide range of network configurations to analyze scalability and DPA tradeoffs.

6.1 Motivation and Key Challenges

Clearly, L1 processor-to-memory interconnects must provide a huge bandwidth, coupled with ultra-low latency. This level of performance is out of

[1]The author would like to acknowledge contributions by Abbas Rahimi, Dr. Igor Loi, and Prof. Luca Benini.

reach for traditional bus-based interconnects [153], even with advanced features like multiple outstanding transactions and out of order completion [4, 154]. Networks-on-Chip (NoC) [46], provide bandwidth scalability, but the latency of traditional NoCs is not adequate for L1 processor-to-memory communication [117], and highly optimized special-purpose interconnects are required.

The design of NxM (where N is the number of processors, and M is the number of memory banks) networks for high-performance architectures usually relies on custom circuit design techniques such as pass transistors, low-swing drivers etc. [155, 156, 157]. Unfortunately this approach is not suitable for architectures featuring soft cores and third-party IPs, which must be compatible with standard technology libraries provided by silicon foundries. An L1 processor-to-memory network provided as a parametric synthesizable IP is therefore highly desirable in this context. Such an IP should come with a carefully tailored logic and physical synthesis strategy, as interconnect delays must be accounted for and minimized to achieve acceptable quality of results.

6.2 Related Work

Parallel computing requires large memory bandwidth [158]: maximizing memory bandwidth is essential in many-core, where the large quantity of parallelism places a heavy request on the memory system [151, 152, 159]. For this reason, many research efforts have focused on developing ultra-high bandwidth interconnects for multi-banked on-chip memories.

A memory-centric NoC is implemented as an on-chip interconnection to support efficient data transactions for a multi-core processor with ten processing elements [160]. The memory-centric NoC consists of five custom-designed crossbar switches, and eight dual port SRAMs provide shared buffers for inter-PE data transactions. Although the star-connected on-chip network supports 11.2GB/s bandwidth [156], its crossbar switches fabric is partitioned by 4x4 tiles which are implemented by non-synthesizable NMOS pass-transistor logic [161]. As another alternative for custom design technique of crossbars, a 128x128 XRAM-a switched swizzle network- is fabricated in 65nm which achieves a bandwidth 1Tbit/s [117]. These crossbars are not compatible with standard technology libraries provided by silicon foundries.

In openSPARC T1 micro-architecture [162], a processor-cache crossbar

has been implemented to accept packets from each of eight SPARC CPU cores, and deliver the packet to one of the four L2-cache banks, the I/O bridge, or the floating-point unit. Although crossbar uses three stage pipelines, it takes more than one cycle to deliver the packet. Another low latency interconnection network based on MoT is implemented in [163, 164], to connect the processing clusters and the memory modules on-a-chip. It provides unique path between each processor and memory module using binary trees of switches. Processing clusters and memory modules are located at the root of trees, while in the traditional MoT [165, 166, 167, 168], they will be located at the leaves of trees which could degrade performance due to the interference. On the other hand, each packet spends one clock cycle in each switch: when the number of processors and memories increases this architecture becomes a latency bottleneck.

The HyperCore architecture [169] is an example of shared-L1 cluster with high performance-per-Watt. The architecture consists of a hardware synchronizer/scheduler, compact 32-bit RISC cores, shared on-chip memory which is accessed by a high-performance interconnect network. Every path from a RISC processor to the shared memory passes through a series of combinational switches where data and addresses move at hardware speed. Another shared memory system for a tightly-coupled multiprocessor is patented in [170]. Its Baseline interconnection network is implemented as a combinational circuit which contributes to making the interconnection network simple, lean and light-weight. No information is publicly available on the DPA of these networks, and their scalability properties have never been assessed in the open literature.

6.3 NETWORK ARCHITECTURE

This section provides an architectural description of the proposed interconnection network based on MoT [163]. It supports non-blocking communication between the processing clusters (PCs) and memories modules (MMs), within a single clock. As shown in Figure 6.1 on page 123, a combinational path is created through a network of primitive building blocks: routing primitives (circles blocks) and arbitration primitives (square blocks). The former are used to create independent routing paths (routing trees) from the PCs to the arbitration tree (and vice-versa). The latter are used to arbitrate concurrent requests (arbitration tree) and to

route them up to the MMs ports and vice-versa.

6.3.1 Routing Switches

The routing tree consists of simple routing switches which route each packet from processor side to memory side, and vice versa. The packets are routed individually based on packets address field. As shown in Figure 6.2 on page 124, the switch has two directions: forward (PC ports) which sends out the incoming packet form its input port at processor side to one of its output ports at memory side; backward which rolls packet back from memory side to processor side (MM ports). The forward packet contains address, data write, and control signal of memory while the backward packet contains the read data, and acknowledgment signal.

Each routing switch consists of a MUX, DEMUX, and a simple combinational control logic which provides a fine-grained address interleaving. By using the fully combinational routing switch, a packet with an active request does not need to spend any clock cycles for traveling from PC side up to end of routing tree. Similarly, the backward path used to get back the read data, is active with the request and hold for the entire clock cycle. No arbitration is performed in this block, and the selection (MUX selector) between the two MM_in ports is univocally based on the request address field (PC_in).

6.3.2 Arbitration Switches

The arbitration tree consists of simple arbitration switches which route packets through the routing trees to memory, and vice versa. As shown in Figure 6.3 on page 124, the switch has two parallel input ports at the processor side (PC_in ports) and uses a MUX to route data from PC_in to MM_out. On the other side, the response packets (MM_in) from the memory side will be routed to one of the two possible outputs (PC_out), based on pending grant status. The round-robin algorithm is used to provide a starvation-free arbitration; it means if a request from one processor loses the arbitration in the current clock cycle, it is quarantined to allocate the output in the next clock cycle. The clock signal is used in controller of arbitration switches in order to switch the round-robin flag in case of simultaneous requests.

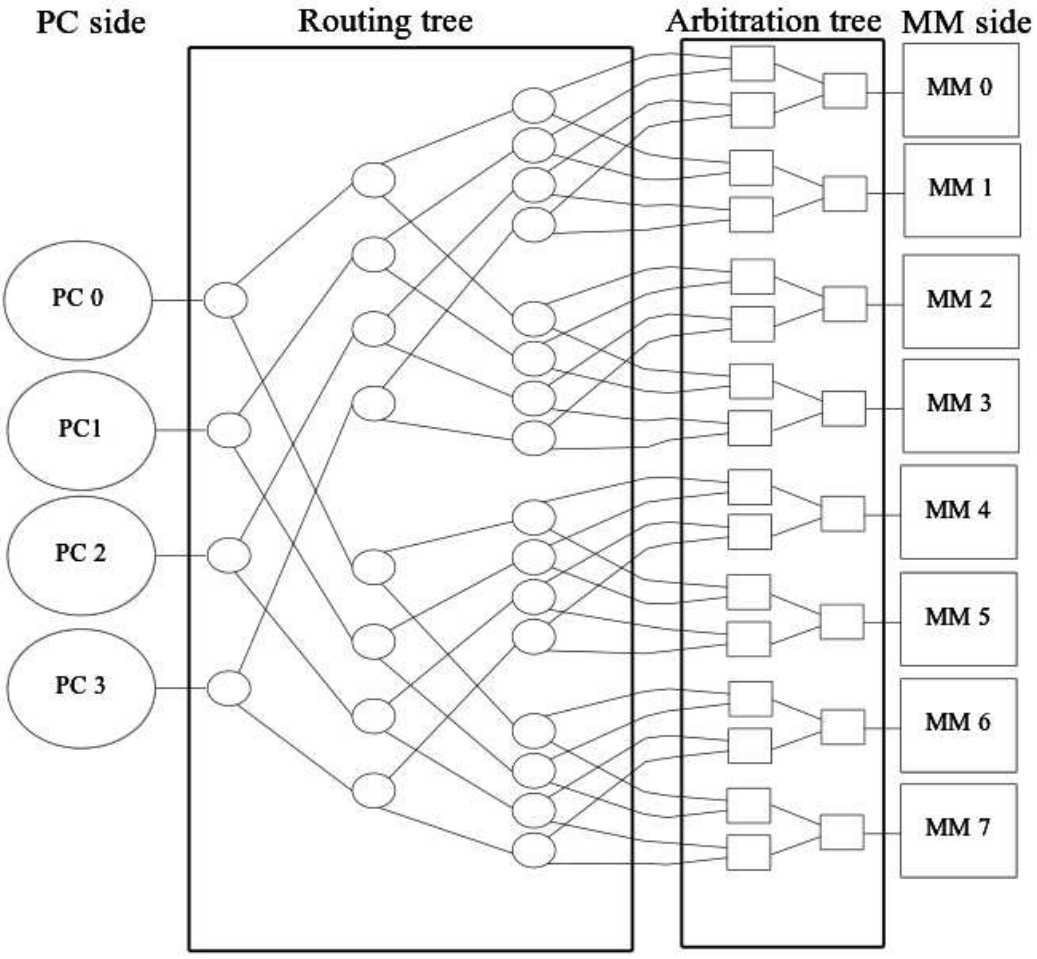


Figure 6.1: Mesh of trees 4x8: empty circles represent routing switches and empty squares represent arbitration switches.

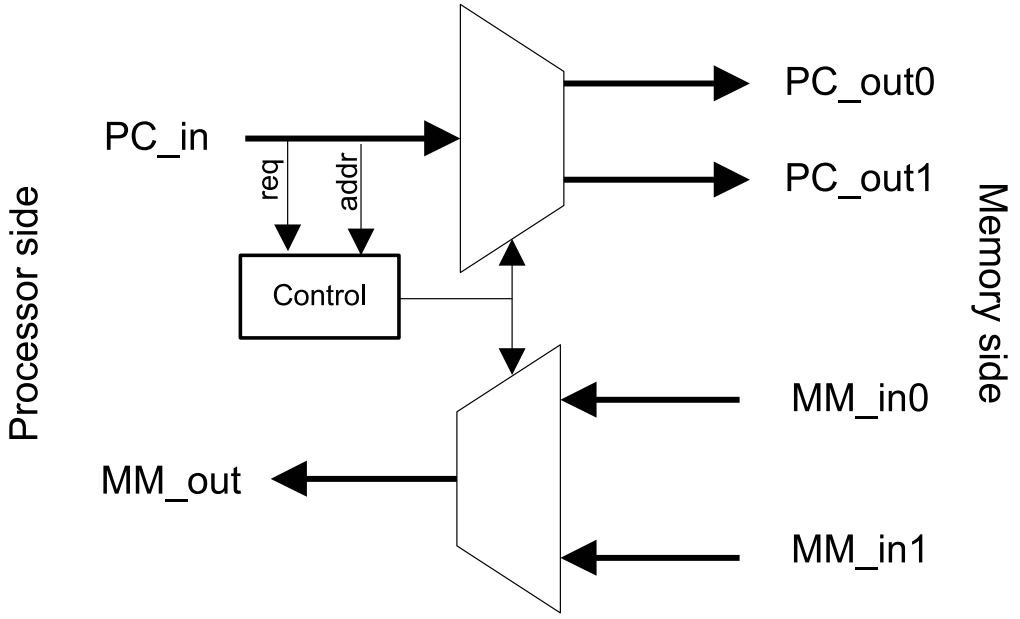


Figure 6.2: Routing switch.

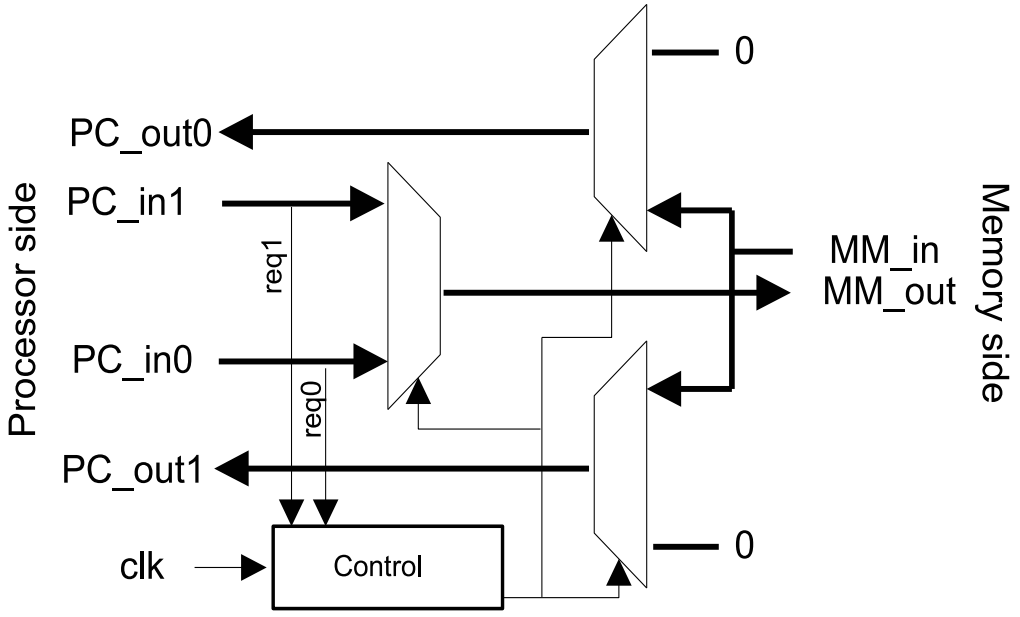


Figure 6.3: Arbiteration switch.

6.3.3 Network Datapath

As highlighted in Figure 6.1 on page 123, the MoT network connects N=2n PCs and M=2m MMs. It contains Log_2^M levels of routing primitives and Log_2^N levels of arbitration primitives. Each memory request issued by PCs must pass through Log_2^M levels of routing primitives to reach at one of MxN leaf nodes in the arbitration switches; and arbitrates among Log_2^N levels of arbitration primitives to reach at MM side. In reverse order, memory responses propagate through arbitration and routing levels to reach at PC side. Although there is a unique combinational path between each processor and each memory, packets from different processors direct to the same memory module are arbitrated while passing through arbitration primitives. The equations 1 and 2 show the total number of routing and arbitration switches needed to connect M PCs to N MMs:

Total number of arbiteration switches = $\sum_{i=1}^{Log_N^i}(N * M)/(2^i)$

Thanks to the modularity features, the network can be easily customized for different cardinalities and different architectural features.

6.3.4 Network Operation

During a read/write operation, data and control signals are asserted by the PCs in the form of packet, as introduced at the beginning of this section. This packet is routed through routing switches (Figure 6.1 on page 123), until it reaches one of NxM ports of routing tree. In order to reach the memory module the packet must be arbitrated among the other simultaneous packets for the same memory module. After passing through all levels of arbitration switches, the packet reaches the memory module, and the read/write operation can be performed.

Packet routing and arbitration are performed in a combinational way, by using a request and acknowledgment signals for arbitration, and address for routing across the switches. Once the request reaches the last level of the arbitration tree and gets the grant, a valid acknowledgement is asserted and propagated back to the related PC through the routing switches (backward). By receiving the acknowledgment signal, PC is able to issue the next read/write operation at the next clock cycle, otherwise it waits until it is received.

In order to provide a single-cycle latency system, each read/write request must be concluded within the clock period. To achieve this goal, we assume that PCs and network (only for round robin priority switching) are clocked with the main clock CLK, whereas MMs are clocked with a skewed_CLK (same frequency and typically 180 phase shift) as depicted

in Figure 6.4 on the next page. These two signals are available at the input ports of the network, and are generated through an external block (PLL). Thanks to this strategy, requests asserted on the rising edge of the CLK (1) are propagated in the timing window (1) to (2); after (2) data must be stable, in order to be sampled by MMs during the rising edge of the Skewed_CLK (3). At this point, in case of write, the transaction is done. In case of read, elapsed the access time, the memory presents at its interface a stable data in (4), which is propagated back to PC side through the already allocated backward path (no arbitration is performed here). Data must reach the PC side before (5) and be stable up to (7) in order to avoid setup/hold violations and data corruptions. At the successive rising edge of the CLK in (6), the PC checks the acknowledgment signal: in case of active acknowledgment (active high), PC samples the read data (if any) and injects the next transaction; if zero it waits the next clock cycle keeping stable the packet on the PC ports.

To avoid timing violations, three conditions must be satisfied:

(i) The clock period must be equal or greater than the sum of the worst case delays from PCs to MMs and from MMs to PCs, plus the access time of memory module and setup time of the MM and PC. There exists another path from PCs to PCs, when the request signal is not granted in case of simultaneous request for a memory module. This latency is bounded by the entire clock period minus the setup time of the PC.

(ii) Latency from PCs to MMs must be equal or smaller than the clock skew minus the setup time of MM .

(iii) Latency from MMs to PCs must be equal or smaller than the clock period minus the skew, the memory access time and the setup time of PC.

These conditions are graphically depicted in Figure 6.4 on the facing page. As introduced in section 3.3, the entity of the network latencies increases with the number or switch levels. Is it clear that, for a given network configuration (NxM) and technology, the forward and backward latencies are lower bounded for maximum performances. By playing with the clock frequency, phase shift and memory access time, it is easy to meet the conditions, thus avoiding timing violations.

6.4 EXPERIMENTAL RESULTS

In this section, we discuss the experimental results for the single-cycle network in terms of delay, power, and area (DPA). Several configurations

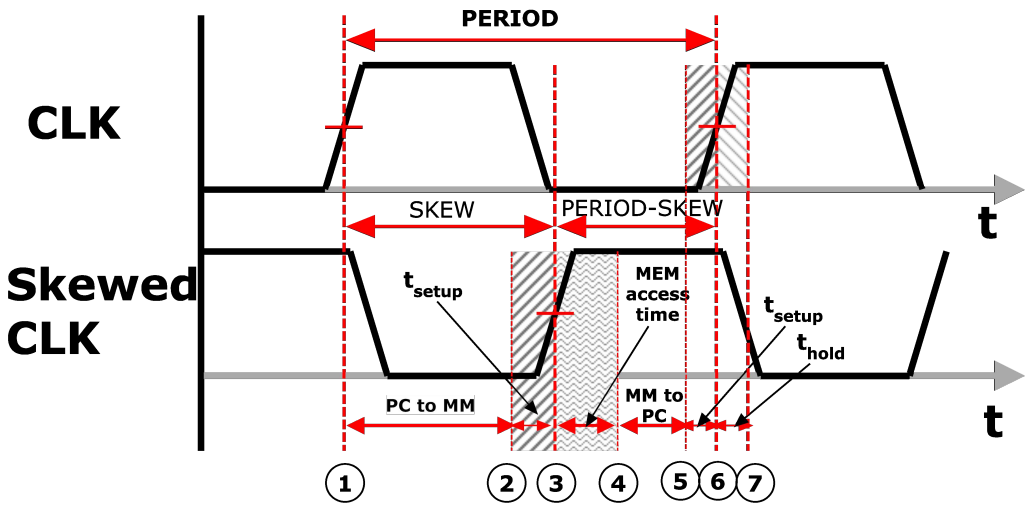


Figure 6.4: Clock and Skewed Clock representation, and related timing requirements. In (1) the PC injects the request and related datapath signals; stable request is received at MM side in (2), and sampled in (3). In case of write the operation is concluded, while in read case, after the memory access time (4) the read data is stable and is propagated back to PC side; in (5) stable read data reaches the PC ports and it is sampled in (6).

have been analyzed. We quantified the cost of the entire network for several PC and MM cardinalities. To get these results, we synthesized the network with the TSMC 65nm technology library (general purpose process). The front-end flow (Multi VTH) has been performed with Synopsys Design Compiler in topographical mode, while the back-end with Cadence SoC Encounter. The sign off has been made with PrimeTime, while functional verification is performed with Mentor Graphics ModelSim.

6.4.1 Design Flow

Figure 6.5 on the next page depicts an overview of the specialized design flow used to maximize speed and explore DPA trade-offs. The network is generated through our high-level generator which takes as input the network template (number of PCs and MMs) and user constraints (BW, datawidth etc.). The output of this stage is a Verilog synthesizable RTL description of the network, and the timing constraints at each port of the network. The synthesis stage is performed in two passes: the first is a pure logic synthesis, and the network is synthesized without physical constraints. This preliminary output netlist is used in the PnR tool to perform the power planning and floorplanning (with pin budgeting), and at the end of this step, the physical information are exported in a def file and back-annotated in the synthesis tool, with topographical features enabled. The second synthesis run performs both remapping and coarse placement plus physical optimization taking into account physical geometries based on the back-annotated floorplan . Using this methodology, good convergence between post-synthesis and post-layout results is achieved early in the flow with only one iteration cycle (correlations in the order or 90-95% is achieved).

The PnR flow is based on the top-down approach. To save runtime, we used dummy hard macros to mimic the timing behavior and physical obstructions of PCs and MMs. The network is flatten and placed in a single tile in the center of the die area as depicted in Figure 6.9 on page 133.

Finally, the extracted netlist, back-annotated parasitic and delays are used to perform area, delay and power analysis with PrimeTime. To ensure the correctness and quality of the achieved results, in parallel with the implementation flow we run the verification flow. A pass/fail test has been adopted for this purpose.

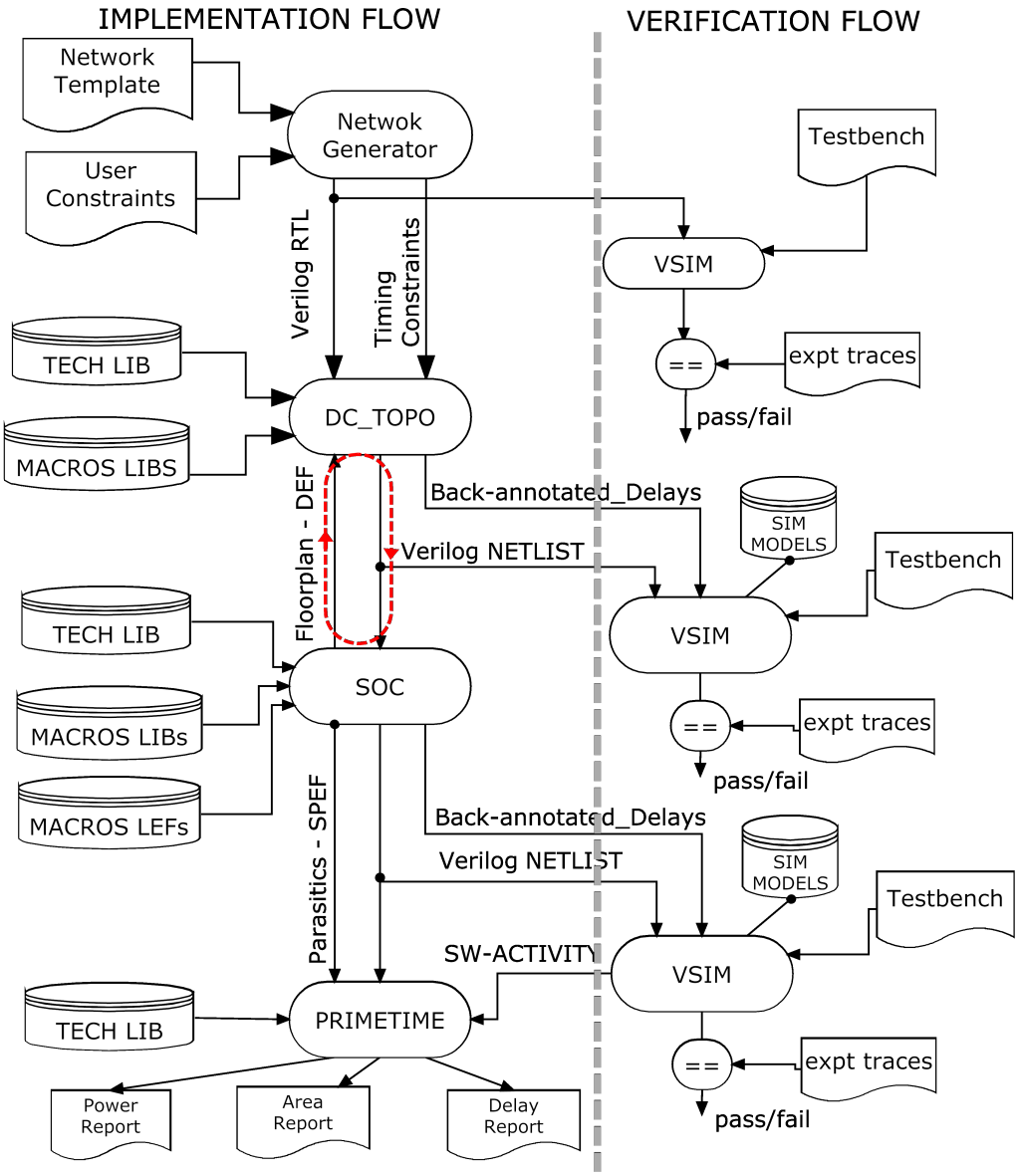


Figure 6.5: Overview of the design flow.

6.4.2 Combinational Network

In this sub-section we present the DPA results for the entire interconnection network for the following configurations (32bits data and address):
(i) 8PCs and 8, 16 and 32MMs

(ii) 16PCs and 16, 32 and 64MMs

(iii) 32PCs and 32 and 64 MMs

We choose to analyze the configurations where the numbers of MMs are equal or greater than the number of PCs, because essentially each PCs leads to one or more MMs. Moreover, the address space can be interleaved in different ranges, and assigned to different MMs, to reduce the memory contentions while increasing overall system bandwidth.

Figure 6.6 on the next page shows the post-layout delay results for the explored network cardinalities (32bit channels width). As described in section 3.4, equations 3, network performance is determined by two paths: forward path from PCs to the MMs, backward path in the reverse direction. The first involves both path arbitration and switch routing and network traversal, whereas in the backward, the path is already established and the delay contribution is entirely due the switches traversal. As shown in Figure 6.6 on the facing page, the delay of the network for roundtrip traversal ranges from 32FO4 (8x8) to 84FO4 (32x64). The delay for 8x16 configuration is 38FO4. When the number of both processors and memories rises by a factor of 4, the delay closely increases logarithmically by 2.2 because the levels of routing and arbitration trees are a logarithmic function from the number of processors and memories.

Figure 6.7 on the next page depicts the total cell area of the network for several cardinalities. As described in section 3.3, the area cost is directly related with the amount of routing and arbitration switches with O(NM). For the 8x16 configuration the area cost is limited to 32K equivalent gates, while for 32x64 it strictly increases less than O(NM), 380K equivalent gates . It shows the great ability of network for saving area while supporting large number of PCs and MMs.

Finally, Figure 6.8 on page 132 shows the power consumption results for a 32bit based networks in terms of cell internal, switching and leakage power. Starting from the 16x32 setup, the contribution of net switching power dominates the overall consumption, mainly because as the number of macros increases (length of wires is dominated by the size of the die), the wire-length and related power rises as well.

The layout view of the 32x64 configuration is shown in Figure 6.9 on page 133, where the Network is surrounded by PCs and MM clusters. Since the network is centralized, in order to minimize the wire-length,

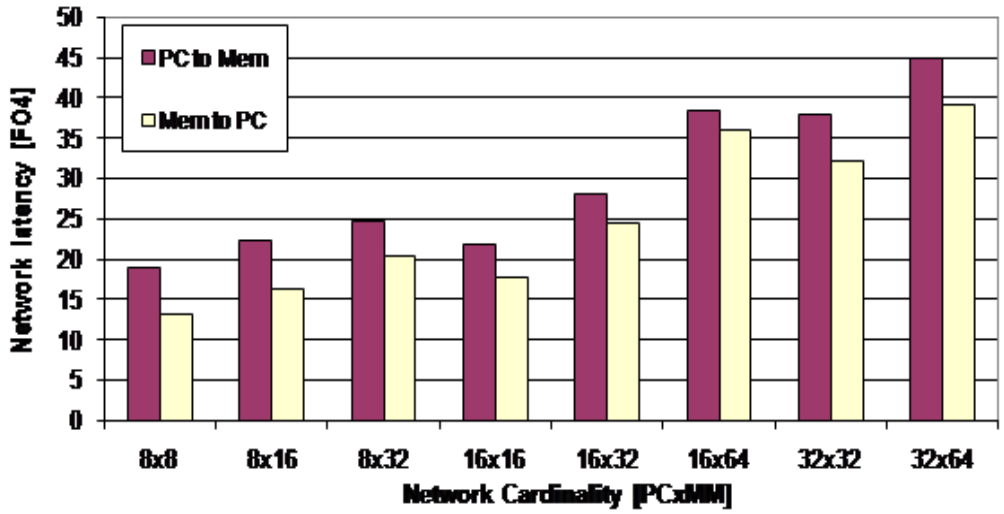


Figure 6.6: Forward and Backward network delay for different cardinalities (32bit).

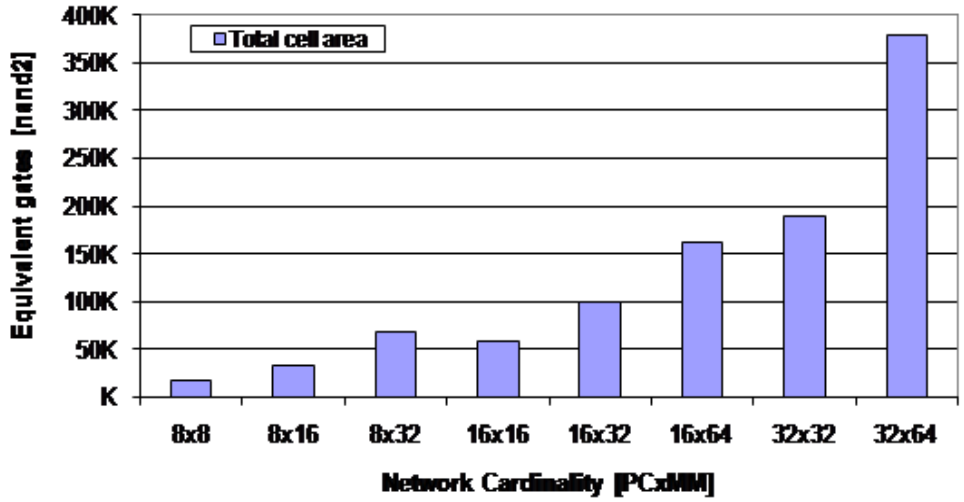


Figure 6.7: Network area cost for different cardinalities.

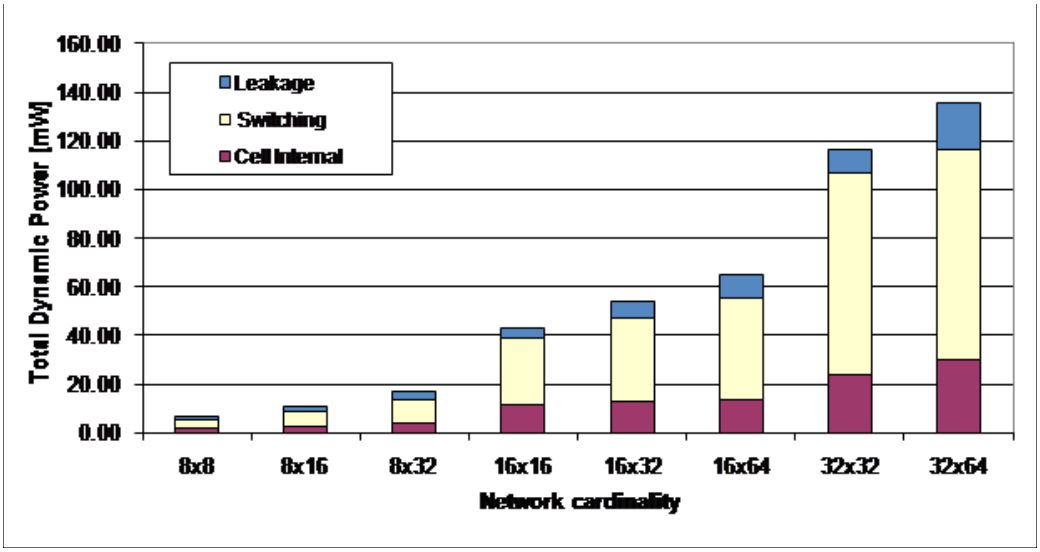


Figure 6.8: Network power consumption for different cardinalities.

the Network tile is placed in the center of the die, while no constraints are applied to MMs and PCs (any path from MM from/to PC is routed through the network). DPA results show the potential of the proposed to network to handle such a big system as a high-performance interconnect with acceptable area and power cost.

6.4.3 Delay-Power-Area Trade-offs

We explore DPA trade-offs for different design constraints for the 16x32 configuration. The trade-offs show the potential of our synthesizable interconnection network to work with different target frequencies and achieving area and power saving. So the interconnection network can be easily adapted to the demands of the new architectures (featuring soft cores and third-party IPs), thanks to the fully-synthesizable and fully automated flow. Figure 6.10 on page 134 shows the trade-off between total area cell of the network and its target frequency. Relaxing the target frequency, the tools are able to infer small (driving strength) and less power hungry cells (regular or high threshold voltage), thus saving both area and power. In comparison to the max-performance 16x32 configuration (310 MHz), we can save area up to 12%, at the expense of 30% performance degradation.

The trade-off between power and performance is illustrated in Figure 6.11 on page 135. By changing the target frequency form 310Mhz down to 210Mhz, 50% total power saving is achieved for network. The

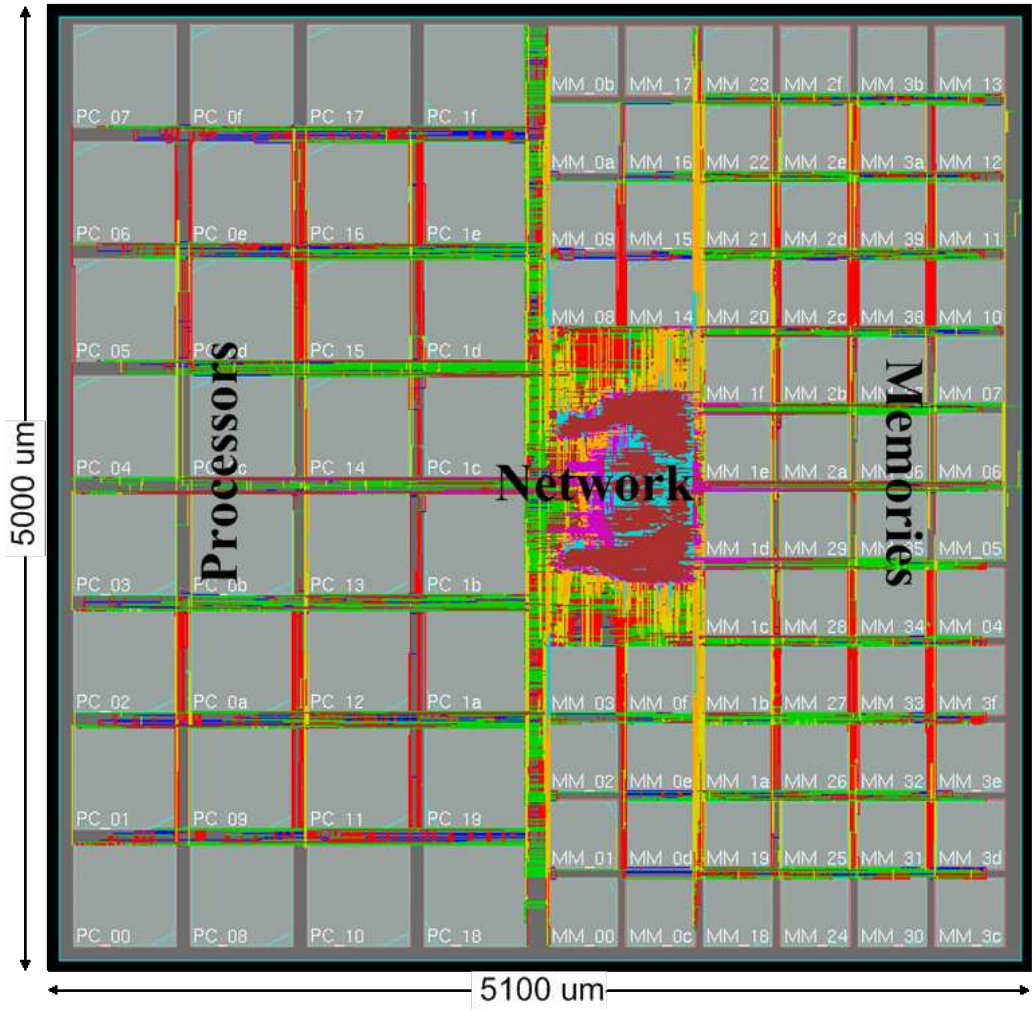


Figure 6.9: Layout of 32x64 configuration.

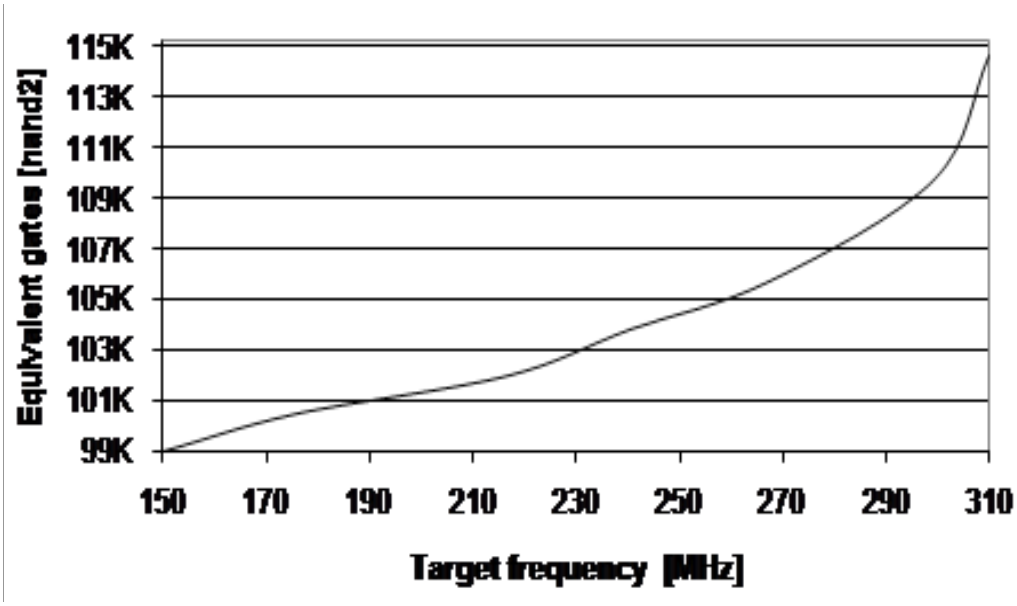


Figure 6.10: Area-Performance trade-off for 16x32 network.

plot shows three regions: starting from the relaxed timing constraint, the tool maps the architecture on high threshold (HVT) cells which yield lower power. As the target frequency increases, the design is mapped on regular voltage threshold (RVT) cells, in order to achieve simultaneous timing requirements and area minimization. This trend is sustained until 290MHz; after this point the timing constraints are very tight and design is dominated by low voltage threshold (LVT) cells, which are fast and power hungry (leaky cells).

6.5 Summary

In this chapter, we proposed a parametric, fully combinational Mesh-of-Trees (MoT) interconnection networkto support high-performance, single-cycle communication between processors and memories in L1-coupled processor clusters. Our interconnect IP is described in synthesizable RTL and it is coupled with a design automation strategy mixing advanced synthesis and physical optimization to achieve optimal delay, power, area (DPA) under a wide range of design constraints. We explored DPA for a large set of network configurationsin 65nm technology. In the next chap-

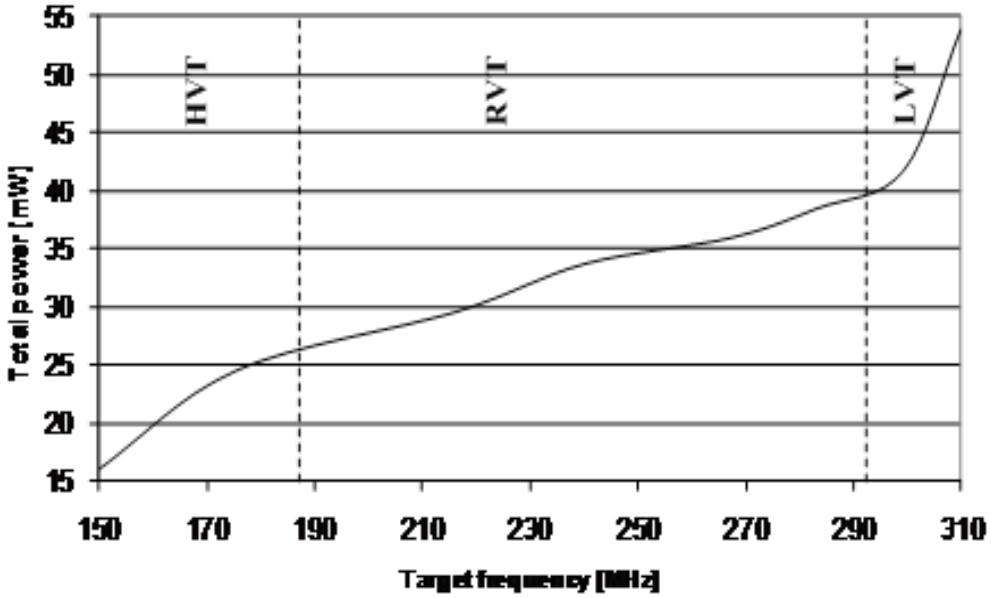


Figure 6.11: Power-Performance trade-off for 16x32 network.

ter we will extend this network to be variaiton-tolerant against the static delay variations.

CHAPTER 7

Variation-tolerant Low-latency Interconnection Network

In the previous chapter we presented a low-latency interconnection network suitable for intra-cluster communication between the heavily shared multi-banked L1 memory and the cores of each tightly coupled processor cluster. In this chapter[1] we enhance the proposed interconnection network and make it reliable and variation-tolerant. We propose an offline (i.e. boot time) technique which can detect timing failures in the system. Our detection approach is based on Test Pattern Generation and Diagnosis performed in off-line mode and during start-up of the system. Then, we propose a reconfiguration mechanism to be triggered after the detection phase. In this phase, we reconfigure the design so that it can overcome the timing failure by injecting pipeline stage in the path and increasing the latency of the read/write transaction. If there is no variation and therefore no timing failure the network operates with single cycle latency as before.

7.1 Motivation and Key Challenges

Scaling down of process technologies has increased process variations and transistor wearout. Because of this, delay variations increase and impact the performance of the design. Conventional inflexible designs often handle delay variations through conservative guard-bands in the operating frequency and voltage to ensure error-free operation across a wide range

[1]The author would like to acknowledge contributions by Prof. Luca Benini.

of dynamic and static variations over circuit lifetime [175]. Consequently, these inflexible designs cannot exploit opportunities for higher performance by increasing frequency or lower energy by lowering V_{cc} under favorable operating conditions. Since most systems usually operate at nominal conditions where worst-case scenarios rarely occur, static worst-case design severely limits performance and energy efficiency.

For the single-cycle interconnection network where achieving a satisfying working frequency is the key for the performance, conservative guard-banding may ensure safe write/read operation on shared-memory modules but with a high performance and power cost (i.e. over-sized gates and buffers, fast but leaky cells, etc.). Thus, over-design leads to a very inefficient system. For this reason, a design which can detect the variation and tune itself is desired to reduce guard-banding while guaranteeing error-free operation.

Some approaches in the literature are based on the error detection sequential (EDS) [176, 177, 178]. In these techniques a combination of a flipflop and a latch can detect the errors in the critical path timing and asserts an error signal. These techniques are suitable for the pipeline stages, and the recovery mechanism is performed by either instruction reply at a lower frequency or multiple-issue instruction reply at the original frequency. Since our timing path is fully combinational, we cannot use these techniques. Moreover, if there is no variation they impose area, power and delay overhead on the design.

Post-silicon tuning at the circuit level such as adaptive body biasing (ABB) and dual-Vdd are other types of methods to face with the delay variation [146, 147]. However, they impose leakage and dynamic power overhead and special care should be taken during the design to reduce these overheads. Moreover, since these techniques are at the circuit level, it is very difficult to apply them only on the critical paths.

7.2 Mega-Leon Architecture

A complete multi-core system named Mega-Leon is designed and implemented as a case study cluster for applying our approach. The Mega-Leon architecture is an evolution of the Multi-core Leon developed by Gaisler [179]. It contains sixteen SPARC-V8 processor cores which are connected through the high-bandwidth logarithmic network described in the previous section to a fast multi-banked, multi-ported Tightly-Coupled

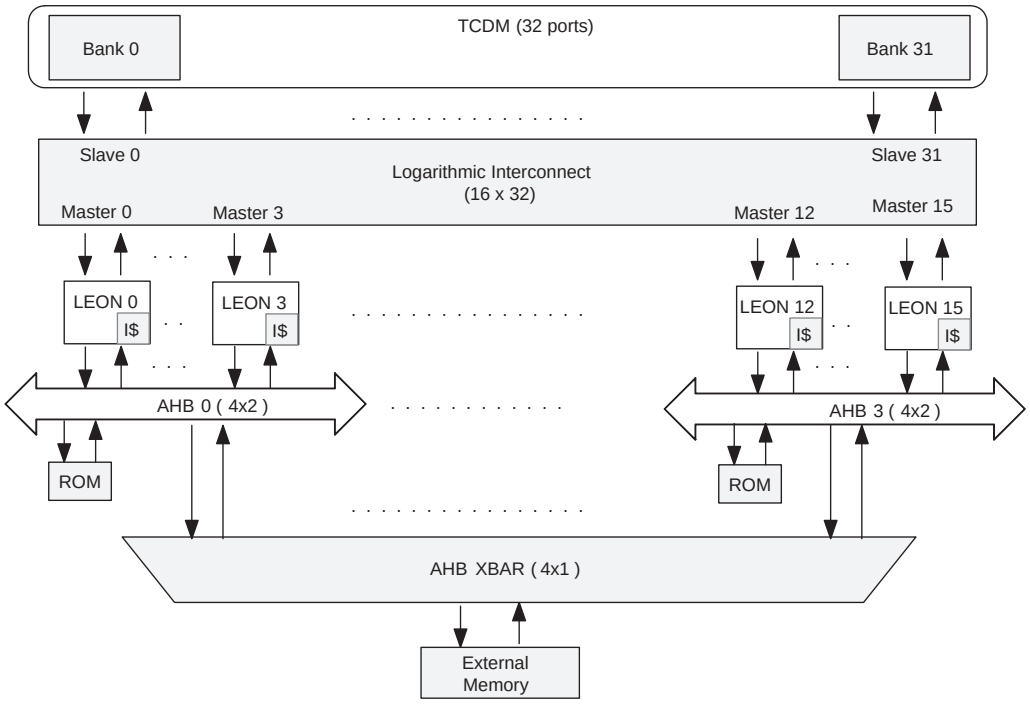


Figure 7.1: Mega-leon architecture.

Data Memory (TCDM). The number of memory ports in the TCDM is equal to the number of banks to allow concurrent accesses to different banks. Processors can synchronize by means of standard read/write operations at logarithmic network providing test-and-set semantics (hardware semaphores). The Cores are also connected to an AHB shared bus system to access the external modules (external memory, ROM, debug support unit and etc.). Figure 7.1 shows the diagram of this architecture.

The SPARC-V8 used in this platform has been customized. Its memory controller has been modified to make the interface with TCDM. The embedded data cache has been removed and replaced with the TCDM which is shared between 16 cores.

The request and response paths from processor to TCDM and vice-versa are single-cycle and fully combinational. These paths are shown in Figure 7.2.

It is shown in Figure 7.2 that the request and response paths include not only the interconnection network but also some logics in the processor's pipeline as well as the memory controller which decodes the address and communicates with the network.

To better understand the behavior of the request and response paths,

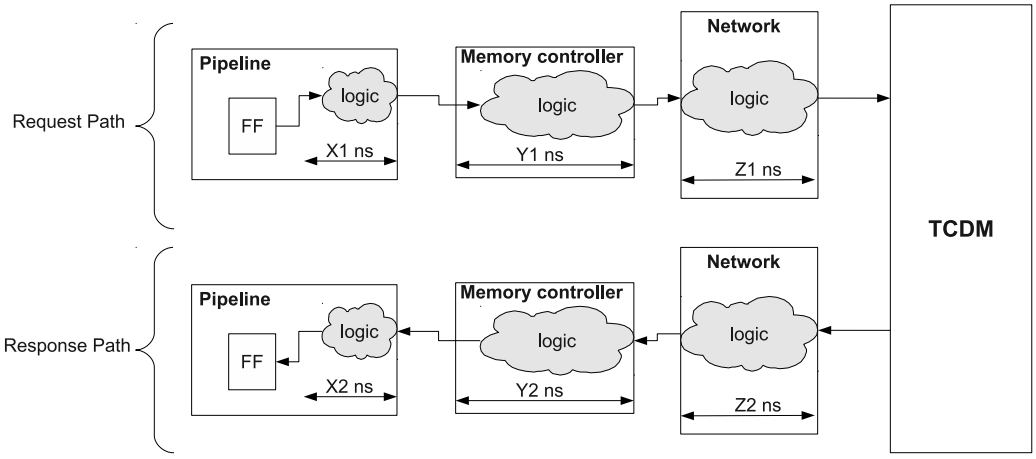


Figure 7.2: Request and response paths from processor to TCDM.

the timing diagram of a read access is shown in Figure 7.3.

It is discussed in the previous chapter that for a given network configuration (NxM) and technology, the forward and backward latencies are lower bounded for maximum performances. By tuning the clock frequency, phase shift and memory access time, it is possible to meet the target frequency, thus avoiding timing violations. However, in order to have a design working after fabrication we need to consider static variations and transistor wearout in the circuit which cause delay variations. Delay variations which happen in the combinational paths from PC to MM (and vice-versa) as well as the access time of the memory may lead to the complete failure of the PC-MM/MM-PC communication. In the next section we describe how we modify the architecture so that it can detect the timing violation and reconfigures itself to avoid communication failures.

7.3 Reliable Architecture

The main property of the proposed interconnection network is having single-cycle latency for the read/write transaction. Thus, the paths from processors to TCDM and vice-versa are fully combinational and have very tight timing constraints. Due to this fact, a little variation on the delay can lead to the complete failure of the processor-memory communication. In the following we discuss on some conventional approaches to face with this problem.

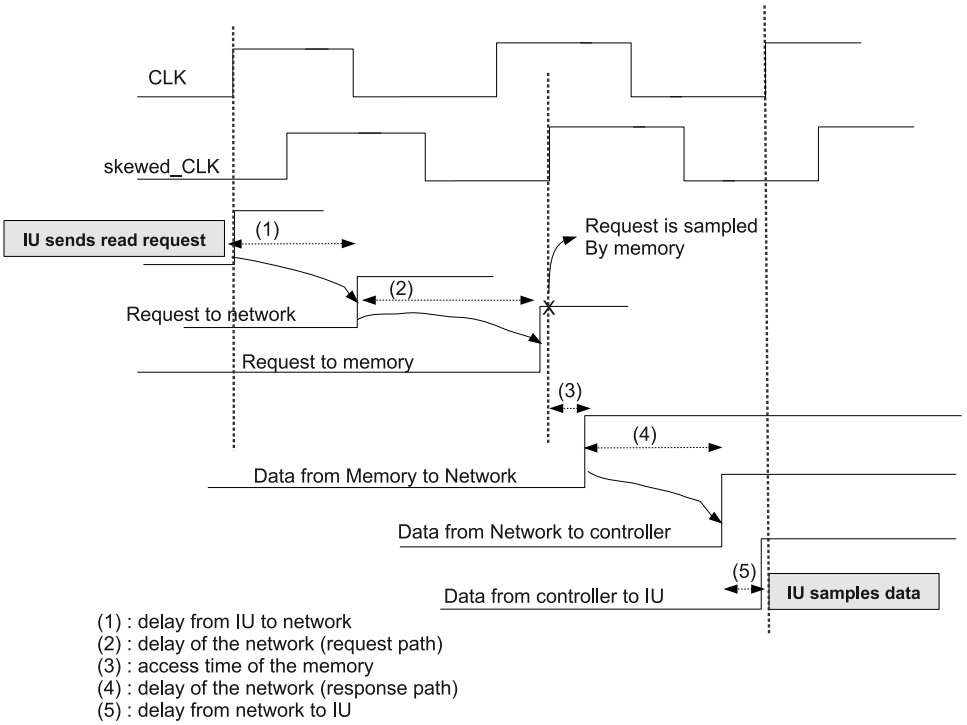


Figure 7.3: Detail timing of the read access from processor to TCDM.

(i) The first solution is adding conservative and sufficient margin for the delay variation. This can be achieved by designing at the worst-case corner. However, this technique limits the performance and energy efficiency of the system. Synthesizing the design at the worst case corner increases both area and power of the design due to using large and low-Vth cells and still has lower MAX-frequency with respect to the nominal corner.

(ii) Another solution is using methods which are based on double-sampling with time-borrowing (DSTB) error-detection sequential (EDS) [176, 177, 178]. These techniques require at-least one stage of the pipeline on the path; while our path is fully combinational and if we add a pipeline for that, it increases the latency of the processor-memory communication which limits the architecture performance in the favorable mode when there is no delay variation.

(iii) Another technique is simply breaking the path and putting one stage of the pipeline on it. As mentioned before, this approach violates the main property of the architecture which is having single-cycle latency and limits the performance when there is no delay variation.

All the above techniques have performance penalty even if there is no variation; but, we need a solution that imposes speed overhead only if a failure happens in the timing of the processor-memory communication. We propose a new approach which is based on the reconfigurable pipeline.

We add a reconfigurable module on the path so that it can compensate the effect of the delay variation by inserting one cycle of the latency on the request or response transaction relaxing the timing constraints. This module does not increase the latency in the normal mode (without delay variation) and the read/write operation is completed in one cycle which is the main property of the architecture. Our reliable architecture for one processor is shown in Figure 7.4.

The reconfigurable pipeline is inserted in the middle of the combinational path i.e. between memory controller and the network. Every Processor has one delay fault tester which can detect failures in the read/write operation. Detection is performed off-line as described later.

If the tester does not find any timing error, it sends a no-error signal to the reconfigurable pipeline. This module reconfigures itself in such away that the processor-memory path becomes fully combinational by selecting the second input of the Multiplexer (Figure 7.4); in this mode the Flip-flops are out of the path.

If variation happens and the tester finds an error, it asserts the related signal and the controllable pipeline switches to use Flipflop in the path by appropriate signaling with both memory controller and the network.

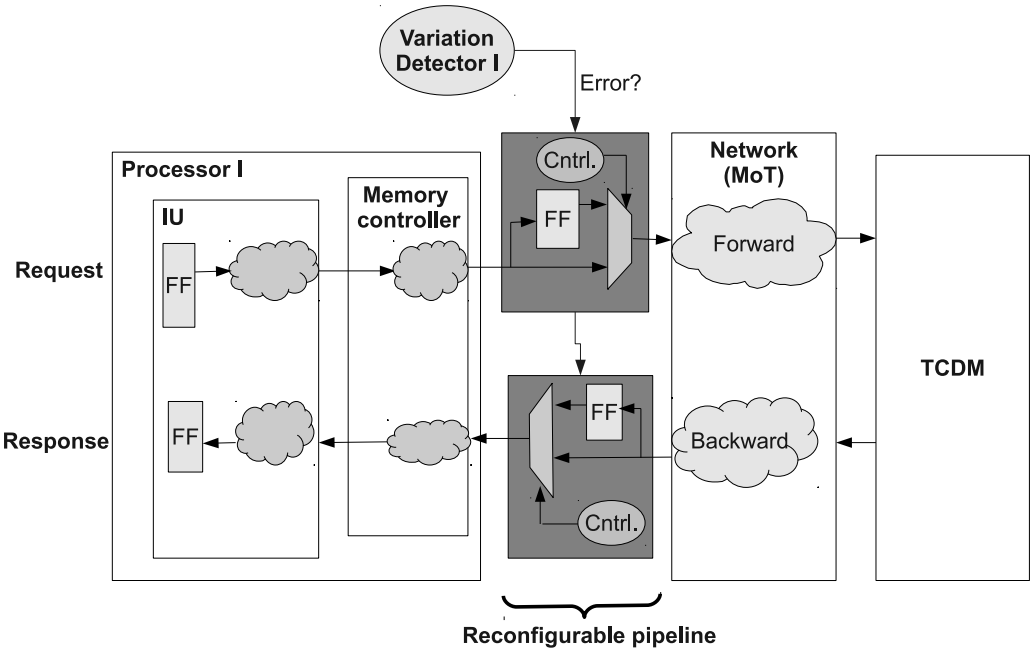


Figure 7.4: Request and response paths with variability-compensation modules.

Therefore, an extra cycle of latency is added to the read/write transaction. Figure 7.5 shows the timing diagram of the request path in the presence of delay variations (Flip-flops are in the path).

We should note two important facts:

(i) since the detection mechanism is offline and it is initiated at the start-up of the system, the control signals going from the testers to reconfigurable pipelines change only at the start-up. Therefore, the reconfiguration phase is also offline and we do not have any concern about switching between two modes (with or without pipeline) when the design is operational.

(ii) Because both detection and reconfiguration steps are offline, our approach is not suitable for dynamic delay variations due to the temperature changes or V_{cc} droops where the delay of the circuit changes dynamically when the system is working. Techniques which are based on Error Detection Sequential are very useful to mitigate these variations. Our approach is best suited for static delay variations due to aging, random variations such as doping related variations, variation in transistor threshold voltage caused by density variations of impurities in the transistor material, and systematic variations such as exposure pattern variation in lithography process or silicon surface flatness variations in the CMP (Chemical Mechanical Planarization).

7.3.1 Detection Mechanism

To detect the timing failure in the read/write operation we propose an offline technique which is based on functional test pattern generation and diagnosis. Similar techniques have been proposed in the literature to detect manufacture failures in the chip after fabrication [100, 123].

We use a tester module which is responsible to send the test patterns and to verify them. This module implements two similar state machines which generate data and address for the write and read transactions and verify the incoming data from memories. We have only one tester which is replicated for all processors. It gets the ID of the processor as an input.

The complete test is performed during M phases where M is the number of memory banks. During each phase all testers write the test pattern to the same bank but at different locations and then read and verify them. Testers get the phase number as an input and based on that generate the write/read address. The related phases for 16 processors and 32 memory banks are shown in Figure 7.6.

The test pattern (data and address) is chosen so that it can detect the timing failure in the read/write operation. The test patterns contain two

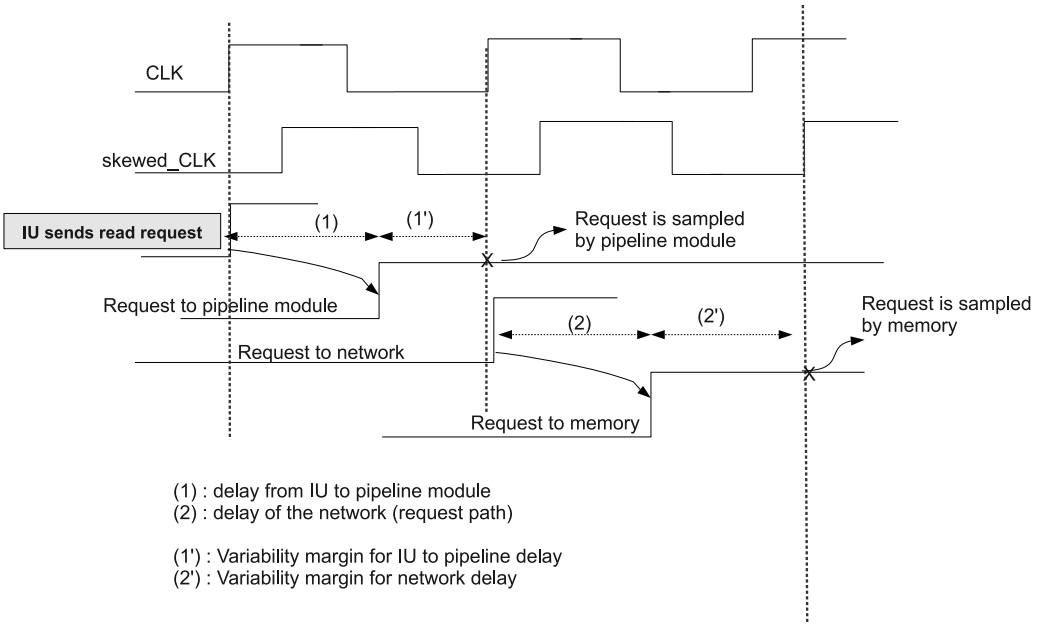


Figure 7.5: Detail timing of request path from processor to TCDM when there is variation (2 cycles latency).

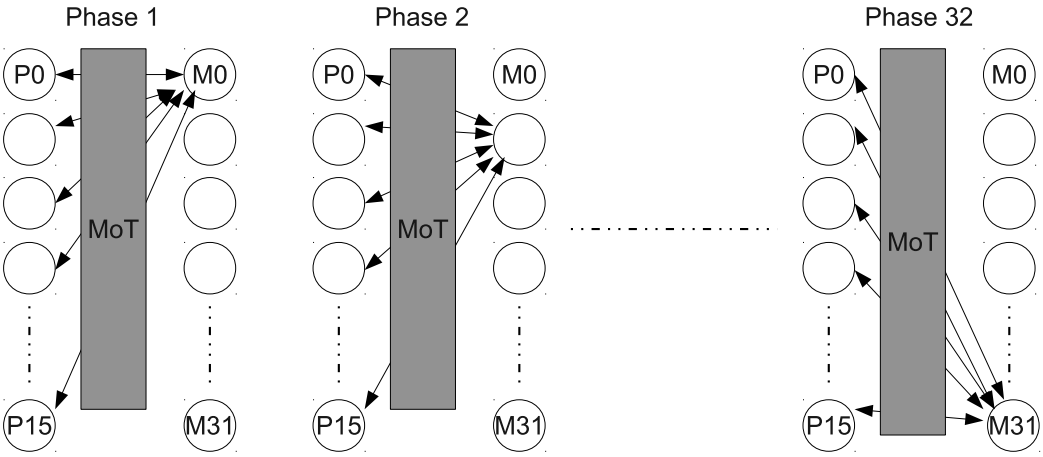


Figure 7.6: Test phases for 16 processors and 32 memories.

consecutive writes and two consecutive reads. The first data that is written is All-zero and the second data is 01010....01. The data are written at addresses ID and $ID+1$ in the Memory bank I where ID is the identifier of the processor and I is the number of the testing phase.

Since at each phase of the test all processors write/read to the same bank, the maximum contention occurs in the interconnection network and therefore all the arbitration paths as well as routing paths from all processors to Bank I (in Phase I) are tested.

Tester writes data to the specified locations of each bank and then reads the same locations. If it reads the wrong data, it means one of the write or read operations or both do not work correctly and tester generates an error signal and reconfigurable module is switched to the pipeline mode. After that we perform one extra test to see if after using the pipeline in the path the test is passed or not. If the test is not passed for the second time, we consider the related processor as faulty.

7.4 Experimental Results

In this section, we discuss the experimental results for the resilient Mega-Leon architecture in terms of delay, power, area and the latency. We quantify the cost of adding variation-tolerant modules into the original design. To get these results, we synthesized the whole architecture including 16 processors, and the interconnection network on a general purpose 65nm commercial technology library [180]. 32 memory banks of 8 KBytes (256 KB in total) are also obtained from the same technology library. Using RTL simulation and the compiler tool-chain of the SPARC-V8, we ran different application benchmarks (C program) on our multi-core architecture to analyze the effect of the compensation approach on the performance of the applications.

7.4.1 Design Flow of Resilient Architecture

To obtain the maximum frequency at which our Multi-core architecture can operate we use the advanced design and synthesis flow desribed in the previous section. Using this flow we could find the maximum frequency at which the design operates. That frequency for our architecture which includes 16 processors, logarithmic network and 32 memory banks (each 8KB) was 250 MhZ (critical path=$4ns$).

To compare our approach with the guard-banding technique where the design is synthesized with a very conservative margin, we performed the synthesis on the same library but at the worst-case corner ($1.1V$ and $125°C$) and with the same timing constraints. Table 7.1 shows the related results.

Table 7.1: Synthesis results of nominal and worst-case corners

	Multi-Vth	*Critical Path* ns	*Area* mm^2	*Leakage power* mW
Nominal	✓	4	3.993	2.8565
Worst-case	✓	4.75	4.078	3.4434
Worst Case overhead	**-**	**19%**	**3%**	**21%**

As seen in Table 7.1, by synthesizing at the worst-case corner, which contains slower cells, the critical path increases by 19%. This critical path was the best timing that we could get using the worst-case corner. The leakage power increases by 21% because the synthesis tool uses more low-Vth cells (which are more leaky) to meet the timing. The area increases by 3% because of using larger cells to meet the timing constraints.

After carefully reviewing the timing paths of the design, as it was expected, we found that the most critical paths are between processors and the TCDM. They start form the Integer Unit (IU) of the processors, traversing the memory controllers and interconnection network and ending at the memory banks. We also found that the delay of the paths whose sources and targets are inside the processors is almost $2/3$ of that of the critical path (PC to TCDM). Therefore, if any variation occurs, those paths that are inside the processors have reasonable margins to tolerate it. Thus, we have to take care of the paths between processors and memories.

To do so and to apply our variation-tolerant approach, we had to find a suitable location on the critical path to insert the reconfigurable pipeline. By carefully investigating all critical paths (processors to memories) of the whole architecture, we figured out that the best position is exactly before the interconnection network and after the memory controller. That location was about $2.1ns$ away from the beginning of the request path and $1.5ns$ from the end of it. It was $2.6ns$ away from the beginning of the response path and $1.1ns$ from the end of it. We put our reconfigurable pipeline exactly before the network. Putting the pipeline at this location creates a large margin for the delay variation and enables us to compensate a degradation of 90% ($(4 - 2.1)/2.1$) on the request path and 55% ($(4 - 2.6)/2.6$) on the response path. In other words, if the variation causes

that the delays of the request and response paths increase by 90% and 55%, respectively, we are still able to compensate it by switching to the pipeline mode.

To realize the range of the variation that can occur on the design, we performed worst-case timing analysis on the design synthesized at nominal corner. We calculated the delay of the request and response paths in both corners. Table 7.2 shows the results. Note that, for these results the design is synthesized at the nominal corner but the timing analysis is performed at both nominal and worst-case corners. It is shown that the maximum variations on the request and response paths are 66% and 30% respectively. Therefore, our approach can compensate the delay variations of both request and response paths if the design is in the worst-case mode.

Based on the results of Tables 7.1 and 7.2, if we synthesize our design at the nominal corner we would get a reasonable speed without a huge synthesis effort and power and area overhead. Then at the run-time we can re-adjust the design by adding the pipeline stage if we are in the worst case condition. Since the worst case is very rare, this will impact only very few chips. Moreover, although using the pipeline mode in the presence of the variation increases the latency of read and write transactions, it does not change the working frequency of the whole architecture and processors are still running at the full speed.

Table 7.2: Timing analysis of different paths on design synthesized at nominal corner

Path	*Nominal corner delay (ns)*	*Worst-case corner delay (ns)*	***Variation*** %
PC-Network	2.10	3.5	**66%**
Network-TCDM	1.57	2.4	**52%**
TCDM-Network	2.6	3.38	**30%**
Network-PC	1.1	1.3	**18%**

7.4.2 Hardware and timing overhead

Our approach requires a few hardware modules including the tester and the reconfigurable pipeline to be added to the original design. These additional blocks are per processor. Therefore, we have 16 tester and pipeline modules for the whole system. We also need one global controller which controls the flow of the testing and synchronizes different test phases. Table 7.3 shows the hardware overhead of these modules.

Table 7.3: Area overhead of our resilient architecture

Module	*Area* um^2	*Overhead on one processor*	*Overhead on Mega-leon*
Tester	2250	2%	0.8%
Reconfigurable pipeline	1200	1%	0.5%
Global controller	957	-	0.01%
All	4407	-	1.3%

As can be seen in this table, the total area overhead of our approach on the whole Mega-Leon architecture is less than 2% which is very small for our resilient architecture.

The reconfigurable pipeline adds one Multiplexer on the critical path and it may increase the delay even if there is no variation. We calculated this additional delay and it was around $20ps$ (0.5% of the critical path ($4ns$)). This small additional delay had no effect on the target frequency since the synthesis tool was able to optimize it.

7.4.3 Testing time

As mentioned before, each test sequence contains two writes and two reads. All processors perform at-least one test sequence during each phase. If the first test sequence fails then the related reconfigurable pipeline is switched to use the extra stage and another test sequence is initiated. Therefore, each processor performs a maximum of two test sequences in each phase. During each phase, the first test sequence takes 4 clock cycles in the tester, and 4 cycles in the processor to memory communication (8 cycles in total). The second test sequence (if the first one fails) takes 4 more cycles (12 cycles) due to having pipeline in the path. Therefore, each processor needs a maximum of 20 clock cycles for each test phase. Since during each phase all processors communicate with the same bank of the memory, maximum congestion happens in the interconnection network. In the worst case processors can get grant sequentially. Therefore, each phase of the test takes 320 clock cycles ($16 * 20$). Since we have 32 different phases the testing procedure requires $32 * 320 = 10240$ clock cycles. We consider some other cycles for synchronization between different phases. At the worst-case we need 11000 clock cycles for the testing procedure to be finished. With a clock period of $4ns$, the complete detection process takes 44μs. We should again note that, the testing procedure is performed offline and during start-up of the system.

7.4.4 RTL simulation and performance analysis

As described, our speed adaptation technique increases the latency of read and write operations. This may slow-down the applications running on the processors. The amount of the slow-down depends on the type of the application. If the application is ALU-dominated and has a few transactions with the memory then the slow-down is very small. However, if the program is memory-dominated and has a lot of load and store instructions, the slow-down is higher.

To evaluate the effect of our approach on the application's run-time, we created some benchmark programs with the help of SPARC-V8 tool-chain. These programs are written so that they can cover a range of applications from very ALU-dominated to very memory-dominated. For accurate evaluation, we did not use any high level simulation, but we used the real RTL model with a commercial RTL simulator.

Figure 7.7 shows the results of our speed adaptation approach on four benchmarks. The curves in the figure show the completion time of each program. X axis is the number of processors for which our speed adaptation technique is applied.

As can be seen, for an application like Fibonacci that is ALU-dominated the overhead of our approach on the run-time is almost negligible. For memory dominated applications like Matrix Multiplication and Transpose, if variation occurs on up-to four processor-memory paths, the overhead is very small (less than 10%). If the variation occurs on more than 8 paths, the overhead is from 20% to 50% for memory dominated programs. Even this overhead is reasonable since the working frequency of the design does not change and we are able to run the whole system under the original frequency even in the presence of the variation.

7.5 Summary

In this chapter, a reliable and variation-tolerant architecture for shared-L1 processor clusters is proposed. The architecture uses a single-cycle mesh of tree as the interconnection network between processors and a unified Tightly Coupled Data Memory (TCDM). The proposed technique is able to compensate the effect of process variation on processor to memory paths. By adding one stage of controllable pipeline on the processor to memory paths we are able to switch between two modes: with and without

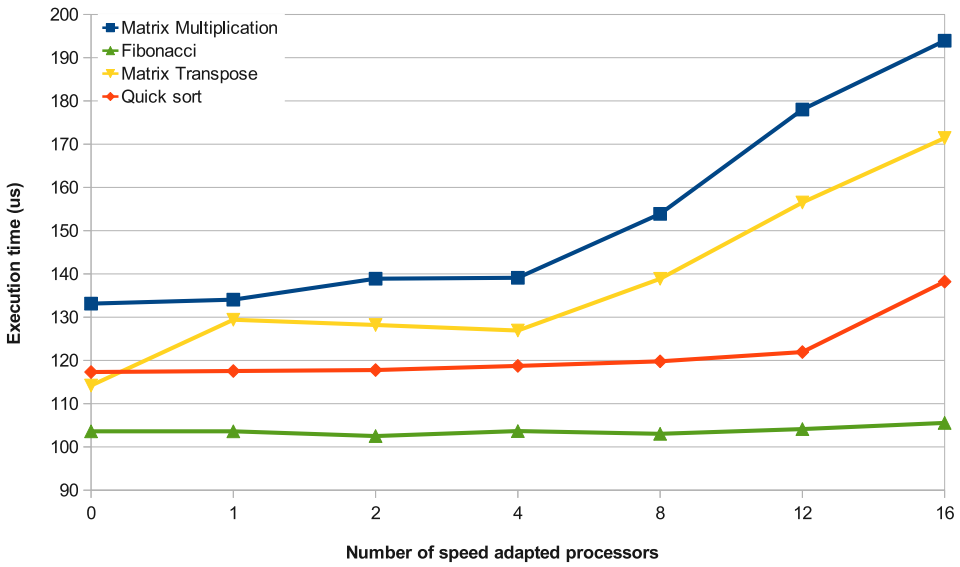


Figure 7.7: Performance overhead due to the speed adaptation on 4 benchmark programs.

pipeline. If there is no variation, the processor to memory path is fully combination and we have single-cycle read and write operations. If the variation occurs, the controllable pipeline is switched to pipeline mode and by increasing the latency of the read/write operation we mitigate the effect of the variations. We also propose a configuration-time approach to conditionally add the extra pipeline state based on detection of timing-critical paths.

In the next chapter we direct attention on the energy efficiency of MP-SoCs. We will propose a robust design methodology which is suitable for ultra-low power SoCs working at near-threshold region.

CHAPTER 8

Robust Ultra-low Power MPSoCs Using Near-Threshold Computing

This chapter[1] describes a dual-Vdd technique for NT operation which can be used to fine tune the performance of a circuit by selectively powering up the timing critical gates in the design at a slightly higher supply voltage than the rest of the circuit while minimizing the total power of the circuit. This technique is very efficient to reduce the power consumption of the interconnection network described in Chapter 6 on page 119 if we need to speed-up the circuit to compensate the effect of variation. Since the Logarithmic network is single-cycle and fully combinational it is very sensitive to variation and this technique can boost the performance with a low power overhead. This is an orthogonal way of compensating variability at the post-fabrication. It is an alternative strategy with respect to the design techniques proposed in the previous chapters.

8.1 Motivation and Key Challenges

Power has become the primary design constraint for a large set of SoCs ranging from mobile phones, PDAs and MP3 Players to sensor nodes. The need to design at ever-higher energy efficiency has triggered a number of research efforts to explore the minimum energy point (MEP) operation, which is reached in the sub-threshold region for CMOS circuits [181]

[1]The author would like to acknowledge contributions by Dr. Ashoka Visweswara Sathanur, Antonio Pullini, Prof. Jos Huisken, and Prof. Luca Benini.

[182]. A whole class of applications and designs that can make use of sub-threshold operation have been demonstrated in the literature [183] [184] [185] [186]. However, due to very low performance, low functional yield and very high performance/energy variability [187], sub-threshold logic applications in commercial products have been limited. Efforts are being made to find the supply voltage "sweet spot" suitable for a wide range of applications with significant performance and robustness requirements. Hence, the traditional quest for minimum energy point operation has paved way to minimum power/energy operation with a given performance [188] as the general methodology to design ultra low power circuits.

In this context, near-threshold operation, where $V_{dd} \geq V_{th}$, but only slightly so, is more robust to process variability and achieves higher functional yield as compared to the sub-threshold operation. Recent works [189] [190] and [188] have shown that Near-Threshold Computing (NTC) has much higher impact across a wider variety of applications with moderate performance requirements and hence looks very promising for future energy-efficient integrated circuit design. Near-threshold operation can also be used in highly complex systems. For instance, in [191] and [192] the authors propose a chip multiprocessor architecture consisting of several cores operating at near-threshold clustered together with a shared, faster L1 cache. Authors in [193] have exploited data parallelism and demonstrated the application of sub/near threshold design to achieve ultra low energy JPEG co-processor. Other works have demonstrated novel logic styles for NTC. Authors in [188] have analyzed MOS operation in near-threshold regime and have provided compact models for current, delay and energy. In [194], authors propose energy-performance tunable logic using pseudo-static logic style circuits and applying skewed supply voltages to the logic gates. They demonstrate 65% lower energy and 2$\times$ higher energy-performance tuning range as compared to conventional static circuits.

The rationale of NTC is that the energy curve is quite flat around the MEP [188] [190] and hence a substantial performance boost can be obtained by moving from the MEP to the near-threshold region without a major loss in energy. To put this statement in quantitative terms, the energy and performance curves for a 51-stage ring oscillator in a commercial 90nm technology are plotted as a function of the supply voltage in Figure 8.1 on the next page. We normalized the energy with respect to the energy at MEP while the delays were normalized with respect to V_{dd} = 1.2V. As we see from the figure, by sacrificing a maximum of $3\times$ in energy, one can achieve up-to $40\times$ improvement in performance over the MEP

point.

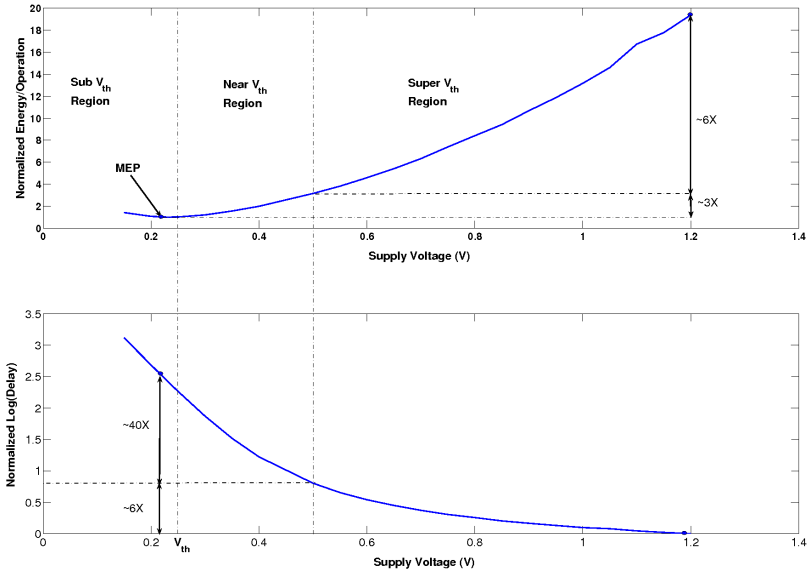


Figure 8.1: Energy-Delay trade off for a chain of 51 inverters in 90nm Technology.

NTCs come however with distinctive challenges, namely: limited performance, significant variation and increased functional failure [190]. As shown in Figure 8.1, normalized delay increases from 0.12 to 1.25 when operating at $400mV$ with respect to $1.1V$ leading to an operation performance loss of around $10\times$; in addition, total performance uncertainty is increased by $20\times$ due to the compound effect of increased threshold voltage and supply noise sensitivity. Functional failures due to storage elements (memory cells, latches and flip-flops) are also orders of magnitude more likely, due to weaker drive of cross-coupled inverters used as the basic bit-storage element. Finally, leakage power in idle state becomes much larger relatively to the greatly reduced dynamic power consumptions of NTCs. This problem can be tackled by post-silicon supply voltage calibration. We observed that the rate of performance change with supply voltage ($\delta t_d/\delta V$) is very high in near threshold regime. As shown in Figure 8.1, a 200mV change in supply voltage from 0.3V to 0.5V causes roughly $30\times$ change in performance. Hence, a desired performance level can in principle be recovered by precise calibration of supply voltage. Unfortunately, this approach suffers from two main shortcomings, which are addressed in the present work. First, if we raise the voltage supply for an entire design we pay a significant power price, as we speed-up not only the critical

paths, but also a very large number of gates on paths which are not critical and which may still be powered at a lower supply voltage without violating speed constraints. Secondly, off-chip (and even more on-chip) voltage regulators cannot achieve arbitrarily fine resolution if one needs very high efficiency, low noise and low cost [195] [196] [197]. In general, only a limited set of supply voltage settings are available, and we need to sacrifice energy efficiency by powering a circuit at the lowest *available* supply voltage which meets performance requirements. Note that in the NT region even a mismatch of 50mV leads to significant energy efficiency loss.

In this chapter, we propose a dual-Vdd technique for NT operation and show that our technique can be used to fine tune the performance of a circuit by selectively powering up the timing critical gates in the design at a slightly higher supply voltage than the rest of the circuit while minimizing the total power of the circuit. The key contribution of this chapter is to acknowledge that in NT a small voltage supply increase leads very significant speedup and hence the two V_{dd}s needed can be only 50 to 100 mV apart. For such a small voltage difference, the static power at the interface between low-voltage and high-voltage supply gates is not extremely large. If we properly select the sub-set of gates to be powered up at a higher voltage, we achieve a significant speedup at an affordable power cost, which is often significantly smaller than what would have to be paid to power up the entire circuit at a higher voltage. To limit the extra cost of dual voltage distribution, we apply our dual-Vdd assignment at the level of entire rows in the layout. Our results demonstrate that this row voltage assignment granularity leads to significant advantages even for small circuits (with balanced aspect ratio layouts), while the extra area cost for multiple V_{dd} distribution is negligible.

Note that, our approach is a post-fabrication speedup technique, i.e. we would like the design to meet the target at low power and working at a specified voltage (V_{ddL}), and then be able to get the timing even in case of degradation due to process variation after fabrication with a minimum power overhead. This will be achieved by powering up the critical rows. An alternative approach would be having a design that exceeds the target (i.e. an overdesign) and can be powered down to meet precisely the target. However, unfortunately to exceed the target, we would need to put larger drives, buffers, etc. leading to a power overhead which is hard to recover by powering down rows.

We should note that our technique is not an alternative to power gating or multi-Vt, but they can be used together. In other words, our fine-grained dual-Vdd is complementary to Multi-threshold CMOS (MTCMOS), as it can be applied as post-silicon speed-up technique even with

designs with multiple threshold voltages.

We propose a novel linear-time heuristic algorithm for dual-Vdd row assignment, and we fully integrate it in a commercial synthesis and back-end flow. Hence, our results are validated at the level of post-layout circuit simulation. Although performance is not critical for designs working in near-threshold region, having the circuit working at target frequency is essential. This might not be achieved after fabrication due to process variation. Here, performance boost is needed but with minimum power overhead especially for near-threshold designs. Therefore, we evaluate our approach using these two metrics: performance improvement and power overhead.

8.2 Related work

M. Wong et.al in [198, 199, 200] proposed an algorithm for voltage island grouping based on the physical proximity of the critical cells in a post-placement voltage assignment for Multi-Vdd design. Multi-Vdd is an effective and well-known method to reduce both dynamic and leakage power [201, 202]. It assigns high-Vdd to timing critical cells while low-Vdd is assigned to non-critical cells, so that power can be saved without degrading the overall circuit performance. In all the techniques proposed in the literature, level shifters need to be inserted at the boundaries between low-Vdd and high-Vdd, causing extra penalty in area, delay and power [203]. Moreover, these techniques perform voltage assignment at either gate-level or block-level. Dual-Vdd operation is extremely interesting in near-threshold, because a relatively small Vdd increase brings very significant speedup and for moderate Vdd differences the level shifters can be omitted. On the other hand, extreme care should be taken to minimize static power increase at the interface between low-Vdd and high-Vdd regions.

Recently, authors in [147] have presented a methodology to mitigate within-die process variation using centralized supply voltage and local body bias compensation approach. In their work, the authors propose to partition the chip into multiple regions with localized sensors which drive the body bias controller to set the appropriate bias voltages to compensate for frequency shift induced by within-die process variation. However the granularity level is still at the block level which leads to a higher leakage power due to compensation.

Multi-threshold CMOS (MTCMOS) utilizes transistors with multiple threshold voltages (Vth) to optimize delay or power [204, 205]. Lower threshold voltage devices are used on critical delay paths to achieve higher performance and higher threshold voltage devices are used on non-critical paths to reduce static leakage power without incurring a delay penalty. Unfortunately, we can not use MTCMOS for post-silicon tuning since the cells and their Vth are fixed at design time. Multi-Vth can be used as a low-power design technique together with our fine-grained dual-Vdd for post-silicon tuning in case of any delay variation and performance degradation. In other words, our fine-grained dual-Vdd is complementary to MTCMOS, as it can be applied as post-silicon speed-up technique even with designs with multiple threshold voltages. Moreover, there is the obvious benefit that if there is no timing variability, dual-Vdd does not have any overhead, but MTCMOS would still have its leakage penalty.

8.3 Row-based dual-Vdd Layout

In this section, we show how we have implemented dual-Vdd on a traditional standard cell based layout style.

Figure 8.2 shows a view of the post-silicon compensation using our dual-Vdd technique. As can be seen, when there is no need for compensation both V_{ddH} and V_{ddL} are connected to V_{dd1} and the design works normally. However, to increase the performance of the design in case of any degradation due to variability, V_{ddH} will be connected to V_{dd2} which is slightly higher than V_{dd1}.

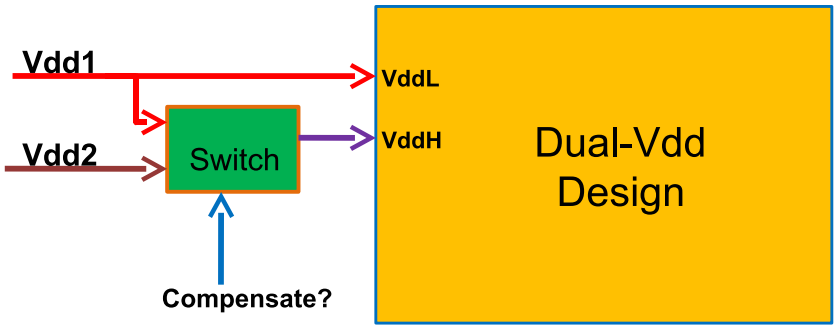


Figure 8.2: Post-silicon compensation using our dual-Vdd technique.

Applying different supply voltages at a single-cell level makes the

power planning at the physical level extremely challenging. Therefore, we used row-based style to implement our dual-Vdd approach, as it is much easier to provide different voltages to each row of the placed design than giving different voltages to each cell separately. Moreover, having 2 Vdd rails instead of one is a quite straightforward task, and we do not need to modify any rule in the power grid creation since there is enough space available between 2 adjacent power stripes. Thus, we implemented our dual-Vdd allocation algorithm on the standard cell based designs with row-level granularity. Dual-Vdd cab be applied at a higher-level of abstraction, for instance at the block level. However, the higher the abstraction level, the more power overhead. We selected row-based methodology as it is a middle-ground between cell-level and block-level approaches and it is easy to implement using standard design flow.

Figure 8.3 on the next page shows an abstract view of the layout with dual-Vdd. In this figure, cells in Row_1 are critical and therefore Row_1 is connected to V_{ddH}, and other rows are connected to V_{ddL}. As can be seen, we incur a very small overhead on the height of floorplan due to well separation required when adjacent rows (Row_1 and Row_2) in the design are assigned to different supply voltages. However, we do not need any well separation on each row, since all the adjacent gates on a row receive the same voltage. For all our experiments the overhead on floorplan due to row separation was less than 5%.

As commonly done in the design practice, we reserve metal layers to run the power networks. The extra utilization of these may now make it a little harder to get through-vias down to the cells for signal routing due to the extra voltage. In experimental results we show that the overhead of our dual-Vdd technique on the routing congestion and runtime is very low, especially for large designs.

8.4 Assessing the Interface Cost

Our technique is based on dual-Vdd approach where given a supply voltage V_{ddL} at which a circuit is operating, we find a sub-set of timing-critical gates for which we rise the supply voltage by δV. We use V_{ddL} as lower supply voltage and $V_{ddL} + \delta V = V_{ddH}$ as the higher supply voltage. Since these supply voltages are not more than 100mV apart, we do not use any voltage level shifters at the interface of V_{ddL}-V_{ddH} domains. This is a key point, as level shifters impose severe area, timing and power overhead if

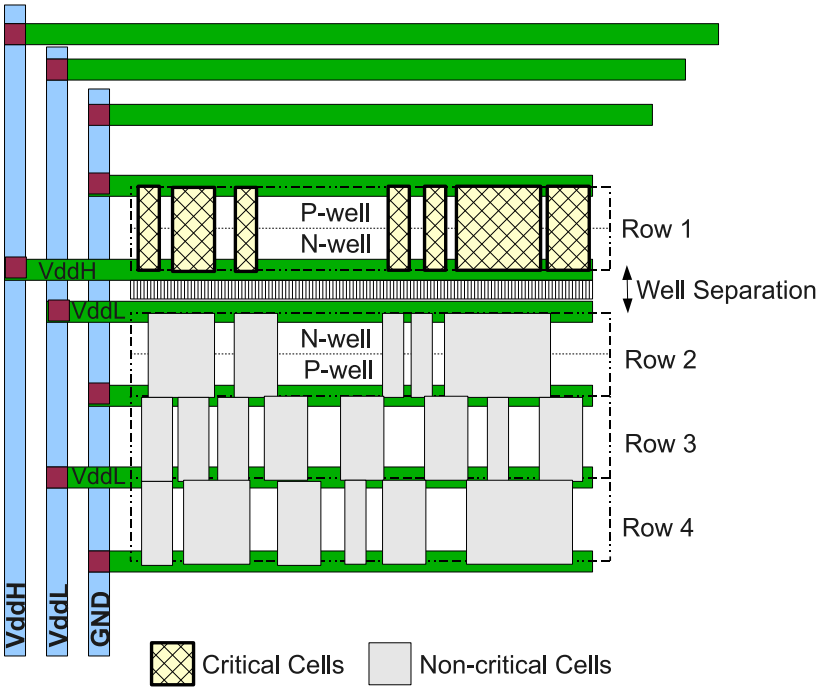


Figure 8.3: An abstract view of a portion of the standard cell layout with dual-Vdd.

applied at a fine granularity, and they should be used only for very coarse-grained multi-Vdd design [206].

We apply our technique on standard cell based designs. Each standard cell row of a design can be powered either by V_{ddH} or V_{ddL}. Initially the circuit is synthesized for a given performance at V_{ddL}. Depending on the percentage of post-fabrication performance boost potential desired, one can then apply the algorithm proposed in Section 8.6 on page 164 to select timing critical rows which can be powered up at V_{ddH}, while minimizing the total power consumption of the circuit, by accounting for the power cost at the V_{ddL} to V_{ddH} interfaces that are created when we select a subset of rows to be powered up. The performance degradation can be detected using detection sensors and monitoring circuits. Several works in the literature have been targeted this issue. Interested readers may refer to [210, 211, 212, 213] for more details.

Let us denote a cell powered at V_{ddH} and driven by V_{ddL} as $cell_{lh}$. In this case, the interface cell ($cell_{lh}$) is powered at V_{ddH} while its input is coming from V_{ddL} domain. Therefore, when the input is '1' i.e. (V_{ddL}) the NMOS is ON while the PMOS is not completely off since the V_{gs} of that PMOS is ($\delta V = V_{ddH} - V_{ddL}$) instead of zero and therefore leakage increases

exponentially. It also reduces the drive strength of the NMOS pull down transistors in $cell_{lh}$ increasing the fall delay of those cells. To minimize this leakage and delay effects, one has to minimize the number of $cell_{lh}$ cells in the design.

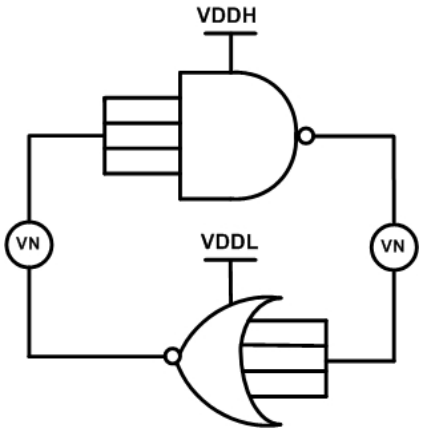


Figure 8.4: 4-input NAND and 4-input NOR back to back configuration.

In addition to the interface power and speed cost, another key challenge is *functional yield*. As discussed in [207], the shape of the Static Noise Margin (SNM) of a logic gate is important for signal regeneration and thus is a key indicator of the gate's functionality. In contrast to the above-threshold regime, I_{on}/I_{off} in near/sub-threshold regime is much lower and as a consequence gates will have lower SNM. Even though operating in near-threshold region improves the I_{on}/I_{off} considerably with respect to sub-threshold, when dual-Vdd technique we propose in this work is applied, the SNM of $cell_{lh}$ gets severely degraded and in some cases might lead to functional failure of the cell. To analyze this, we consider the worst-case possibility of this happening by considering NOR and NAND gates connected back-to-back, as described in [187]. As most libraries allow a maximum of four-transistor stacks, we consider four input NAND (NAND4) and NOR (NOR4) gates. Worst-case dual-Vdd connection occurs when NAND4 is power by V_{ddH} and NOR4 is powered by V_{ddL} as shown in the Figure 8.4. This results in a worst-case connection due to parallel leaking PMOS in NAND4 having much higher leakage and reduced ON current when the input is at V_{ddL} thus increasing the possibility of its failure. We also consider process variability to fully evaluate the functional correctness of this setup. We chose five process corners typically used for characterization, namely TT, FF, SS, FS and SF. where 'T' stands for typical, 'S' stands for slow and 'F' stands for fast. Then each pair

indicates the corner for PMOS and NMOS respectively. With this setup, we varied V_{ddL} from V_{th} to 0.5V and δV from 50mV to 200mV. We found that V_{ddL} = 0.3V and δV of 100mV are the minimum V_{ddL} and maximum δV required for correct functional operation.

8.5

Optimal dual-Vdd allocation algorithm

We first cast the dual-Vdd allocation problem into an ILP to find the optimal clusters with their respective voltages. Then, we propose a two-pass linear time heuristic to solve the problem. The dual-Vdd allocation problem can be defined as follows. *Given a placed design with a set of rows, partition the design into C ($C = 2$) clusters (sub-sets of rows), each with its own voltage such that the overall timing of the design is met while minimizing the leakage power.*

We cast this problem into an ILP where the objective function is to minimize the total leakage power one has to spend in order to speed up the design and hence obey the timing constraints. The design is partitioned into C ($C = 2$) clusters to achieve this goal. The set partitioning problem can be stated as the following ILP:

$$\textbf{Minimize} \sum_{i=1}^{N}\sum_{j=1}^{P} x_{i,j} * L_{i,j} \tag{8.1}$$

Subject to

$$\sum_{i=1}^{N}\sum_{j=1}^{P} a_{i,j,k} \cdot x_{i,j} \leq b_k, \quad \text{for } k \in \Pi \tag{8.2}$$

$$\sum_{j=1}^{P} x_{i,j} = 1, \quad \text{for } i \in \{1, \ldots, N\} \tag{8.3}$$

$$\left.\begin{array}{l} \sum_{i=1}^{N} x_{i,j} \leq F * y_j, \quad \text{for } j \in \{1, \ldots, P\} \\ \sum_{j=1}^{P} y_j \leq C \end{array}\right\} \tag{8.4}$$

$$x_{i,j}, y_j = \{0 \; or \; 1\}, \quad \text{for } i \in \{1, \ldots, N\}, j \in \{1, \ldots, P\} \tag{8.5}$$

The problem is expressed in terms of the binary variables $x_{i,j}$, where $x_{i,j} = 1$ indicates that row i is assigned to voltage j. Equation 8.1 is the objective function, which expresses the minimization of the total power in the design, where $L_{i,j}$ is the power of row i when it is assigned to voltage j, P is the number of voltages (for our case: $P = 2$) and N is the number of rows. The constraint of Equation 8.2 expresses the timing constraints relative to the critical path set Π. The value $a_{i,j,k}$ denotes the reduction in path delay of a path $k \in \Pi$, where Π is the set of critical paths, when $V_{ddH} = v_j$ is applied to the row i. This is calculated by first determining the gates in the row i that are on the path k and with a supply voltage of v_j and then summing up the reduction in delay of those gates. This can be done as follows. Let $Q_{i,k}$ be the number of cells on row i and on path $k \in \Pi$, and $d_l, l \in \{1, \ldots, Q_{i,k}\}$ the delays of these gates. When applying a higher supply voltage to row i with a V_{ddH} of v_j, the delay of these gates will reduce by a quantity $\delta_l, l \in \{1, \ldots, Q_{i,k}\}$. Then, $a_{i,j,k} = \sum_{l=1}^{Q_{i,k}} \delta_l$. Note that there are P different voltages leading to P such coefficients for each row and for each path.

The right hand side of the constraint b_k for $k \in \{1, \ldots, M\}$ indicates the speed-up required for each path k in the critical path set Π, where M is the size of Π. This value is computed as $b_k = D_{crit} - (p_k * (1 + \beta))$ for $k = \{1, \ldots, M\}$; where D_{crit} is the critical delay, p_k is the delay of path k and β is the speed-up factor. Note that there are as many constraints (M) as the number of paths in the critical path set Π. The constraint of Equation 8.3 says that a row can belong to only one cluster (one supply voltage), out of the possible P different clusters.

Equation 8.4 defines the constraints for which only C or less clusters or supply voltages are allowed out of P possible values. This is done by introducing auxiliary variables [208] in the formulation. We introduce P such variables y_js, each for one cluster. Note here that F is a very large number. The first inequality in this group (4) denotes that several rows could be assigned to a same cluster, and the second inequality shows that the number of possible clusters should be less than or equal to C which is defined by the designer. For example, if there are 4 different voltage values (P=4) but the designer would like to have only two different clusters, then C should be defined as 2 (2 out of 4). Finally, Equation 8.5 defines the bounds for $x_{i,j}$ and y_j variables, which are binary variables.

8.6

Heuristic dual-Vdd row assignment

In addition to the ILP formulation, we propose a linear-time heuristic which solves the allocation problem in a greedy way. Here, we define the dual-Vdd allocation problem like that of the previous section. The heuristic algorithm has two main phases which are shown in Algorithm 1 on the facing page and Algorithm 2 on page 167, respectively. In the first phase, we prioritize rows based on their time criticality. There are rows in the design which have gates in the non-critical paths and hence can tolerate a lower supply voltage resulting in reduction of the power. The first phase begins with calculating the timing criticality co-efficient for each row. We compute this factor as follows. For any given design M, and any speed-up factor of $X\%$ (we define speed-up as the reduction percentage in the critical path delay), if D is the delay of the critical path in M, the delay after speed-up must be less than $D_s = D * (1 - X\%)$. Therefore, all paths in design M whose delays are more than D_s have to be boosted. Let $CritPath$ be a set containing all these critical paths (steps 3-9), and $N_{i,k}$ be the number of cells on row i and on path k. Then, the timing criticality co-efficient (TC) of row i is calculated as follows (steps 10-21):

$$TC_i = \sum_{k \epsilon CritPath} Delay_k \cdot N_{i,k} \tag{8.6}$$

where $Delay_k$ is the delay of path k.

After calculating timing criticality coefficients for all the rows, we perform row ranking based on these coefficients in decreasing order (steps 22-23) and go to the second phase of the algorithm. At the beginning of Phase 2, all rows are connected to the same supply voltage which is the nominal and original voltage at which the design is already synthesized; we name this voltage V_{ddL}. The aim of the second phase is to find a subset of rows, R_{VddH}, and a corresponding supply voltage value, V_{ddH}, so that we can decrease the critical path delay of the design to D_s with minimum power overhead after powering up all the rows in R_{VddH} to V_{ddH}.

Two constraints are given to this phase of the algorithm. First, as described in Section 8.4 on page 159, to achieve a reasonable leakage power overhead at the interface of V_{ddL} and V_{ddH} and to have the correct functionality, the difference between V_{ddL} and V_{ddH} must be less than 100mV ($\delta V < 100mV$). The second constraint is the step value on which we sweep all the voltages from V_{ddL} to $V_{ddL} + \delta V$. We set this value to 50mV as it is a reasonable resolution voltage that can be produced efficiently using on-chip DC-DC converters [197]. Based on the first and second constraints, we have two possible values for V_{ddH} which are (line 1):

Algorithm 1 Phase 1 of the dual-Vdd algorithm.

```
1: { //R=Rows; X=Speed-up;} { P = Timing Paths; D=Critical delay}
2: PhaseOne(R,X,P,D)
3: Ds=D*(1-X);
4: CritPath={};
5: for all path in P do
6:    if delay(path) > Ds then
7:       add path to CritPath;
8:    end if
9: end for
10: for all row in R do
11:    TC(row)=0; {t}iming criticality co-efficient
12:    for all path in CritPath do
13:       Sum=0
14:       for all cell in row do
15:          if cell exist in path then
16:             Sum=Sum+1
17:          end if
18:       end for
19:       TC(row)= TC(row)+Sum*delay(path)
20:    end for
21: end for
22: SR=Sort(R,TC,decreasing);
23: return SR
```

$V_{ddHs} = \{V_{ddL} + 50mV, V_{ddL} + 100mV\}$. However, our algorithm is able to solve the dual-Vdd allocation problem for any δV and step value which are given as constraints.

We calculate the power and delay of the design for all voltage values in V_{ddHs}. The goal behind this is that we want to see what the maximum speed-up achievable by the dual-Vdd technique is. Actually, the maximum speed-up is obtained when we put all rows to the V_{ddH} instead of V_{ddL}. We keep the related power together with the critical path delay of each voltage in V_{ddHs}. To do that, for each V_{ddH} in V_{ddHs} we power up all rows to V_{ddH} and compute the power and critical path delay and save the values (MaxPowers, MinDelays: steps 3-12). Then, we put back all rows to the original voltage i.e. V_{ddL} (step 13).

In the last step of Phase 2, we find a set of rows which need to be powered up and the related V_{ddH}, as well. To carry out this, we iterate through the voltages in V_{ddHs}. For each V_{ddH} value in the V_{ddHs}, we first check if the

desired delay (D_s) is achievable using the current V_{ddH}. This can be easily preformed by comparing D_s with the related value in MinDelays list (step 19). If the speed-up is achievable by the current voltage of V_{ddHs}, we fix V_{ddH} (step 20) and go to find corresponding rows. Actually, V_{ddH} is the lowest V_{dd} from the V_{ddHs} list that satisfy the timing constraints considering the speed-up factor.

As our goal is to boost the design critical path with the minimum power overhead, the best heuristic way is to put the most critical rows at higher supply voltage, first. Therefore, we iterate through the rows in the sorted rows list (SR) from the most critical row to the least critical one (steps 21-29). At each iteration we put the current row at the higher supply voltage (V_{ddH}) and add it to the related list which stores the rows at high V_{dd} (step 23). Then, we check the timing. If we meet the speed-up, we have found the solution and return the V_{ddH} and the related row list, R_VddH (steps 25-28); otherwise we continue to the next row until we meet the desired speed-up and find a solution.

In the algorithm, we can define a Max-Power threshold and for each given speed-up, after assigning each row to V_{ddH} we check the power and if we violate the Max-Power constraint we assign the row back to V_{ddL} and generates warning that the current speed-up leads to Max-Power violation.

The computational cost of the heuristic algorithm is $Cost = Max(C_{phase-one}, C_{phase-two})$. The complexity of first phase of the algorithm is $O(CP * R * C)$, where CP is the number of timing critical paths (size of $Critpath$), R is the number of rows and C is the maximum number of cells in a row. The cost of second phase of the algorithm is less than the first phase and is equal to $O(2 * R) = O(R)$. Therefore the overall complexity of the algorithm is $O(CP * R * C)$ which is linear in the number of rows and in the number of timing critical paths which depends on the speed-up value.

8.7 Placement Optimization

Since our methodology is a design time optimization, it can be improved by re-placing certain cells to reduce costs at interface gates and to reduce the number of rows which are assigned to the higher supply voltage. To perform this, we put a few placement constraints on the critical cells. These constraints guide the placement tool to place critical cells near each other and on a few rows. To perform this, before the placement phase we

Algorithm 2 Phase 2 of the dual-Vdd algorithm.

```
1: {R=rows; SR=sorted rows from Phase 1}. {X=speed-up; P=timing
   paths} {VddL = the base voltage where design is synthesized}
2: Phasetwo(R,SR,X,D,P,VddL)
3: VddHs={VddL+0.050,VddL+0.100};
4: MaxPowers={};
5: MinDelays={};
6: i=0;
7: for all vddh in VddHs do
8:    power_up_all_rows(R,vddh)
9:    MinDelays[i]=delay(critical_path)
10:   MaxPowers[i]=total_power(design)
11:   i++;
12: end for
13: power_down_all_rows(R,VddL);
14: Ds=D*(1-X);
15: i=0;
16: R_VddH={};
17: for all vddh in VddHs do
18:    Solution_found=false;
19:    if Ds > MinDelays[i++] then
20:       VddH=vddh;
21:       for all row in SR do
22:          power_up_row(row, VddH);
23:          R_VddH=R_VddH + row;
24:          Dn=delay(critical_path);
25:          if Dn<Ds then
26:             Solution_found=true;
27:             return (VddH, R_VddH );
28:          end if
29:       end for
30:    end if
31: end for
32: return
```

extract the critical paths and the related cells based on the target speed-up that we want to achieve with dual-Vdd technique. As seen in Section 8.6 on page 164, the critical paths are those paths whose delays are more than D_s (see Section 8.6 on page 164). After extracting the critical cells, we put soft placement guides on those cells and ask the tool to place them near

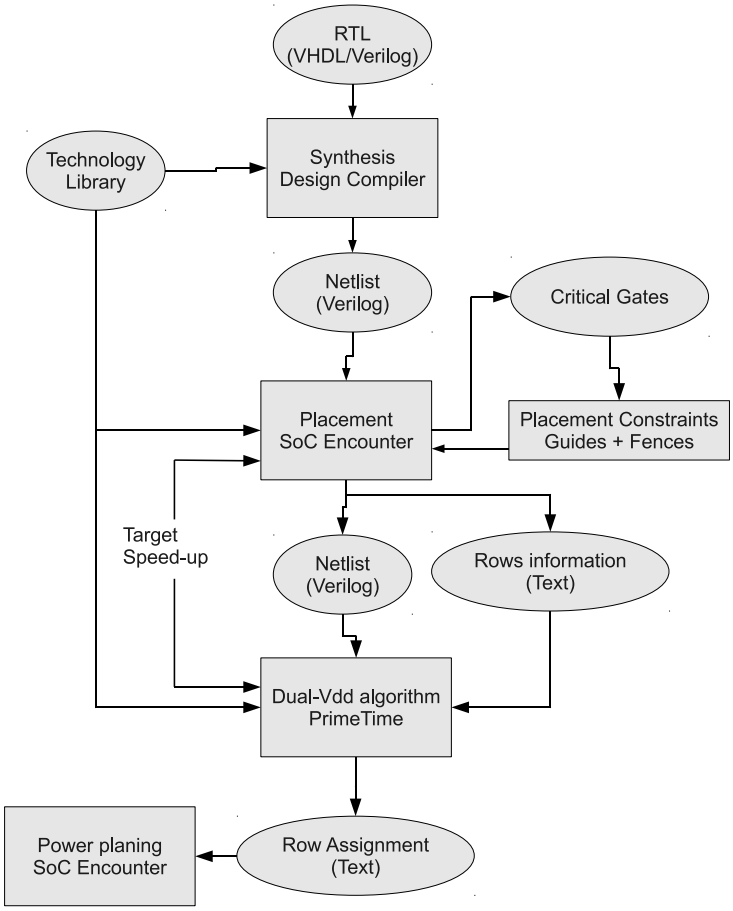


Figure 8.5: The flow of our dual-Vdd row assignment methodology.

each other and on a few rows. Note that, these placement constraints are soft and if the tool cannot meet the timing with them, it ignores them.

Adding placement constraints may have negative effect on the performance of the design due to changing the timing-driven placement. However, we show in the experimental results that this timing overhead is negligible and the power saving that can be achieved using this technique overweights it.

As mentioned earlier, our dual-Vdd technique is fully integrated into the commercial synthesis and back-end flow. Figure 8.5 shows the flow diagram of our dual-Vdd technique using tools from Synopsys and Cadence.

8.8

Experimental Results

We applied our dual-Vdd approach to a total of eight designs; 3 of them are public-domain benchmarks taken from the ISCAS85 suite, while the remaining 5 are circuit modules belonging to industrial SoCs. Each design was synthesized and placed using a 90nm general-purpose CMOS technology library from TSMC. For each of the gates in the library, we characterized its delay and power for all voltages in the range of [0.4V, 0.45V, 0.5V, 0.55V, 0.6V, 0.65V, 0.7V] using the library characterization tool from Cadence, ETS. The TSMC90nm CMOS models we used for library characterization were at standard voltage threshold (SVT) with a threshold voltage of 0.2V.

We synthesize the designs using Synopsys Design Compiler and placed them using Cadence SoC Encounter. After placement of the design we dump a set of files which are loaded into Synopsys PrimeTime. We load the design and the list of rows dumped by Cadence SoC Encounter into the PrimeTime. The dual-Vdd algorithm is implemented using PrimeTime scripting. Since our heuristic algorithm is very fast, run-times are small and are completely dominated by the times spent in synthesis and P&R in the original flow.

As mentioned before, the main concern in our dual-Vdd approach is the leakage power at the interface gates, where a gate which has low V_{dd} supply voltage drives a gate at higher V_{dd}. In this situation the leakage current of the gate at higher V_{dd} increases because of lower voltage level at the gate of transistors. Note that, power waste happens only if the input of the gate is at high logic level i.e. '1'. This leakage overhead is not modeled in the standard library format and therefore cannot be computed automatically by PrimeTime. In order to take it into account, we performed SPICE simulations and calculated the leakage at the interface in the worst case situation for each cell; then we back annotated the values into the PrimeTime. For each pair of (V_{ddL}, V_{ddH}) in the voltage ranges, and for each gate in the library we performed a SPICE simulation by putting the gate supply voltage to V_{ddH} and its inputs to V_{ddL} and computed the leakage. We subtracted the original leakage of the gate from the calculated leakage and stored it in a table. The values of the related table are later back-annotated into the PrimeTime scripting and added to the original power calculated by the tool. SPICE simulation for all voltages and for all the cells are performed and results are back-annotated into static timing analysis (STA) tool (PrimeTime). Since analysis is carried out in SPICE for each cell and each voltage and we exploited the standard and powerful STA provided

by PrimeTime, the values of delay and power during different iterations of the algorithm are accurate enough for row assignment algorithm.

8.8.1 Dual-Vdd evaluation

Table 8.1: Results of dual-Vdd approach on 8 different benchmarks

Design	Gates [#]	Rows [#]	Total Power at V_{ddL} [uW]	Speed-Up [%]	Rows at V_{ddH} in dual-Vdd [#]	Leakage at interface dual-Vdd [uW]	dual-Vdd power overhead [%]	single-Vdd power overhead [%]
Big ALU	3215	72	42.57	5 20	1 18	0.1 3.08	0.63 19.31	28.26
Simple ALU	78	11	0.96	5 20	1 3	0.01 0.17	4.22 27	29.3
Multiplier 16bit	9185	88	89.08	5 20	2 18	0.24 1.24	1.54 8.48	27.75
Multiplier 32bit	37595	182	393.0	5 20	2 22	0.15 4.22	0.66 5.22	27.74
switch 12x12	54074	273	265.90	5 20	3 54	1.39 25.55	0.64 13.65	28.69
s13207	3827	87	82.32	5 20	3 12	0.86 2.56	1.94 6.54	28.04
c6288	2199	55	17.16	5 20	6 23	0.64 0.91	7.58 19.42	28.32
s15850	5117	104	46.03	5 20	3 13	0.78 2.4	2.41 8.47	28.46

To evaluate the effectiveness of our dual-Vdd approach, we compared timing and power results with those of single V_{dd} while putting the entire design at the higher V_{dd} to obtain the desired speed-up. We performed experiments for different speed-ups targets in a range from 4% to 60%. Note that speed-up is defined is the percentage reduction in the critical path delay and is less than 100%. We synthesized all designs at $V_{ddL} = 0.4V$ and applied the dual-Vdd algorithm on them. Table 8.1 summarized results for all benchmarks at $V_{ddL} = 0.4V$, $V_{ddH} = 0.45V$ and for 2 different speed-ups (5% and 20%). Results for the complete range of speed-up values are shown in Figure 8.6 on the next page for four designs.

In Table 8.1, Columns 2 and 3 are the number of gates and rows of each design. Column 4 is the voltage at which the design is initially synthe-

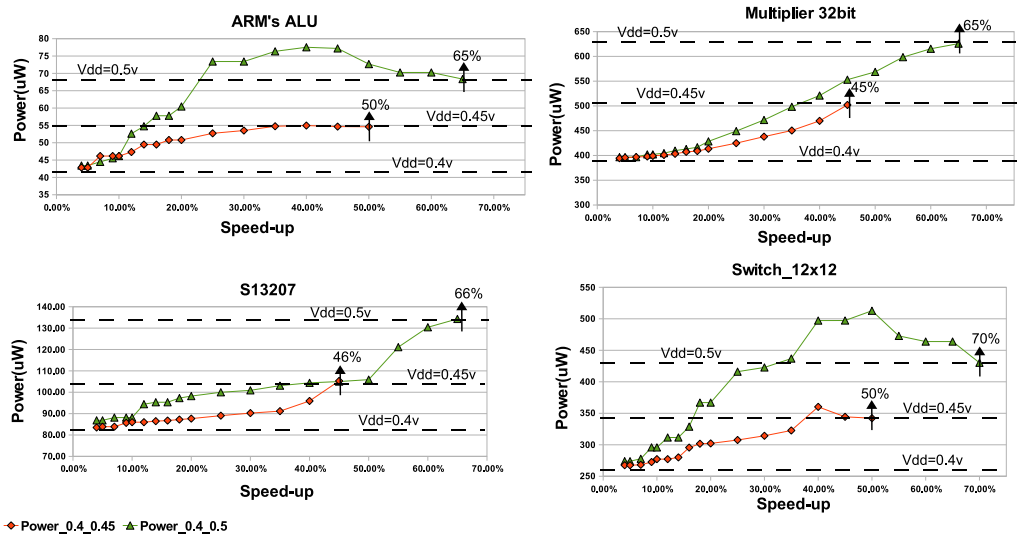


Figure 8.6: Results of dual-Vdd technique applied on four benchmark circuits.

sized. Column 5 denotes the V_{ddH} value that dual-Vdd algorithm finds to achieve the desired speed-up at minimum power cost. As it can be seen, for up to 20% speed-up the algorithm uses $V_{ddH} = 0.45V$. To achieve the desired speed-up one can use the $V_{ddH} = 0.45V$ as the single voltage for all cells. However, as shown in Columns 10 and 11, the power overhead of dual-Vdd approach is always less than that of single V_{ddH} in all three different speed-ups. Column 9 is the leakage power at the interface gates (V_{ddL}-to-V_{ddH}) in dual-Vdd approach. It can be seen that this leakage power is always less than 10% of total power for all the speed-up ranges and even less than 5% for 6 of the benchmarks.

The results related to the power for 4 of the benchmarks are analyzed in detail in Figure 8.6. For these experiments, we used two different δV for the algorithm: $0.050V$ and $0.100V$, corresponding to a high-accuracy voltage supply generator ($50mV$ resolution) and to a medium accuracy one ($100mV$ resolution). The three flat dashed lines in the graphs are powers related to the single V_{dd}. The bottom line ($V_{dd} = 0.4V$) is the power of the circuit when all the cells are powered at $V_{dd} = 0.4V$. The middle line ($V_{dd} = 0.45V$) is the power of the circuit at $V_{dd} = 0.45V$ and the top line ($V_{dd} = 0.5V$) is the circuit's power at $V_{dd} = 0.5V$. The arrows and related values on the middle and top dashed lines show the speed-up percentage (reduction in delay) of each V_{ddH} with respect to the base V_{dd} i.e $0.4V$. For example, in the plot of *S13207* the value of 46% on the top of the arrow

on the middle dashed line indicates that if we power-up all cells of the *S13207* to $V_{dd} = 0.45V$, the critical path delay of the circuit will decrease by 46% compare to the delay of of the circuit when all cells are powered at $V_{dd} = 0.4V$. As seen, the related delay reduction for this design with $V_{dd} = 0.5V$ is 64% compare to that of $V_{dd} = 0.4V$. The two solid lines in the graphs that are not flat (*Power_0.4_0.45 and Power_0.4_0.5*) are the power of the circuit when we used the dual-Vdd technique. *Power_0.4_0.45* is the circuit's power in dual-Vdd technique when $V_{ddL} = 0.4V$ and $V_{ddH} = 0.45V$ and *Power_0.4_0.5* is the power of dual-Vdd technique with $V_{ddL} = 0.4V$ and $V_{ddH} = 0.5V$. As shown in the graphs, the power overhead of dual-Vdd technique is proportional to the speed-up factor.

As it can be seen in Figure 8.6 on the preceding page, with a δV of 0.050V and for most of the speed-up factors our algorithm can find a dual-Vdd solution with $V_{ddH} = 0.45V$ and $V_{ddL} = 0.4V$. The power of that solution is less than that of the single high V_{dd} (middle dashed line). Note that, the powers shown in the graphs are the total power consisting of dynamic power, leakage power and the interface leakage power in dual-Vdd approach.

Figure 8.6 on the previous page shows that in almost all benchmarks the maximum speed-up achievable by dual-Vdd with $V_{ddL} = 0.4V$ and $V_{ddH} = 0.45V$ is around 45%, which is equal to the speed-up that obtained by powering up all cells to the V_{ddH} (arrows on the middle dashed lines). It can be seen that in almost all benchmarks and for all the speed-up factors the power of dual-Vdd technique when $V_{ddL} = 0.4V$ and $V_{ddH} = 0.45V$ is between the power at $V_{ddL}(V_{dd} = 0.4V)$ and the power at V_{ddH} ($V_{dd} = 0.45V$).

As shown in Figure 8.6 on the preceding page, the voltage of $0.45V$ is not enough to achieve more than 45% speed-up. In this case our algorithm selects $V_{ddH} = 0.5V$ and $V_{ddL} = 0.4V$. *Power_0.4_0.5* is the power related to dual-Vdd technique when $V_{ddL} = 0.4V$ and $V_{ddH} = 0.5V$. As it can be seen in the graphs the power overhead of dual-Vdd approach for more than 45% speed-up factors is sometimes higher than that of single V_{ddH} $(0.5V)$. This is because of leakage overhead at the interface. When the difference between V_{ddL} and V_{ddH}, and the speed-up and therefore the number of rows connected to V_{ddH} increase, the leakage overhead at the interface gates will dominate the power saving obtained by dual-Vdd technique. This will lead to more power overhead in comparison with the single V_{ddH} technique. However, our dual-Vdd approach is able to recover this by applying the same voltage ($0.5V$ for this case) to both V_{ddH} and V_{ddL} during compensation and, therefore, removing the interface leakage power.

8.8.2 Back-end of dual-Vdd

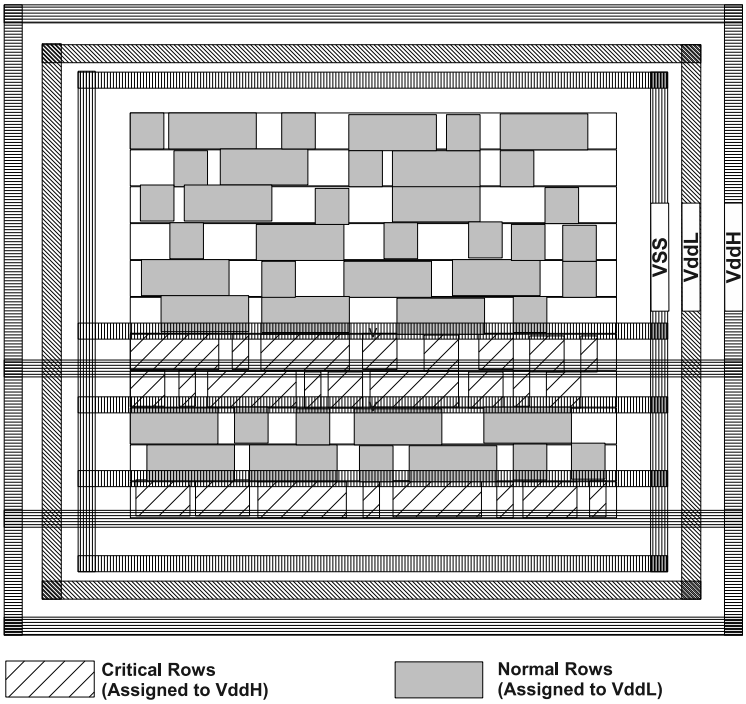


Figure 8.7: Placed and power-routed ALU design with dual-Vdd (Rows 1,4,5 from bottom are assigned to VddH).

To give a detailed view on the back-end implications of our technique, we analyzed a case study of a simple ALU. The complete placed and power-routed layout is shown in Figure 8.7. In this case we ran the algorithm for 20% speed-up. The number of rows that need to be powered up to V_{ddH} to achieve 20% speed-up as shown in Table 8.1 on page 170 was 3. In Figure 8.7 there are two different rails: V_{ddH} and V_{ddL}. There are 3 different power rings which are V_{ddH}, V_{ddL}, and V_{ss}. As this design was very small, we did not use any stripes for the powers. If the design is larger we have to use vertical stripes. However, this will not change the power planning and the connections to the rails are made at the stripes instead of rings.

Those highlighted rows and related cells in the layout are rows which are powered up at V_{ddH}. As shown in the layout, in this example we do not have two adjacent rows which receive different supply voltages i.e. V_{ddL} and V_{ddH}. However, in such a situation where two adjacent rows should receive different supply voltages but they are sharing one V_{dd} line, we need

to modify the layout and separate those two rows for applying different voltages to them. This row separation imposes a very small area overhead; however, we do not need any well separation on each row, since all the adjacent gates on a row receive the same voltage. This is another reason that we choose row-based style for our algorithm. In all our experiments the area overhead was below 5%.

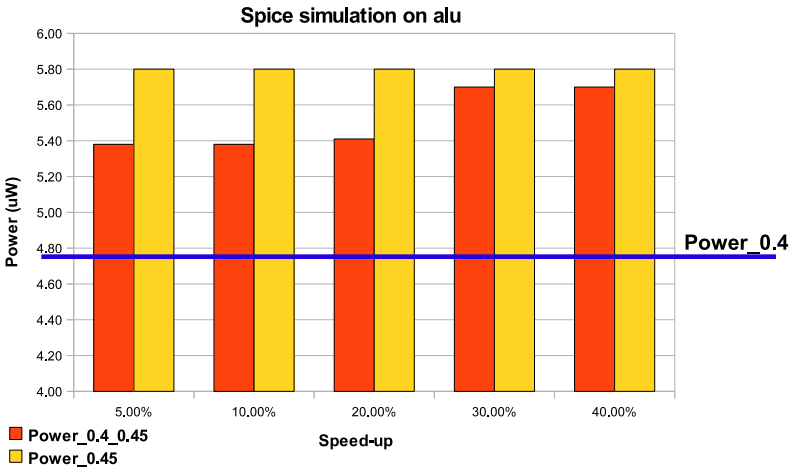


Figure 8.8: SPICE simulation results of dual-Vdd on simple ALU.

Full SPICE simulation on the simple ALU after dual-Vdd allocation has been performed for 5 different speed-ups: 5%, 10%, 20%, 30% and 40% with $V_{ddL} = 0.4V$ and $V_{ddH} = 0.45V$. We created the appropriate SPICE netlist by extracting it from the layout. Then, for each speed-up factor we powered up the cells to either V_{ddL} or V_{ddH} based on the results of dual-Vdd algorithm. The results of SPICE simulation are shown in Figure 8.8. The flat line shows the average power of the circuit at $V_{ddL} = 0.4V$. These results show that, as it was expected from the previous experiments on standard cell based flow, with $V_{ddL} = 0.4V$ and $V_{ddH} = 0.45V$ the average power of the circuit in dual-Vdd mode is less than that of the single V_{dd} mode where all cells are powered at $V_{ddH} = 0.45V$ for all the speed-up factors. Note that our technique gives sizable savings (7% power reduction for 20% speed-up) even for such a small circuit, but it is best suited for larger circuits with many rows and delay paths.

Table 8.2: Results of routing congestion and runtime overhead due to dual-Vdd on different benchmarks.

Design	H. Congestion single-Vdd Avg	H. Cong. dual-Vdd Avg	V. Cong. single-Vdd Avg	V. Cong. dual-Vdd Avg	Runtime single-Vdd Minute	Runtime dual-Vdd Minute	Routing Runtime overhead%
S13207	0.45	0.47	0.89	0.95	12	14	18
S15850	0.46	0.47	0.82	0.89	17	20	17
NoC Switch	0.96	1.2	1.1	1.15	34	41	20
Mac32	0.97	1.1	0.91	1.0	36	42	16
Big ALU	0.7	0.76	0.81	0.86	44	50	13

8.8.3 Routing overhead

As described in Section 8.3 on page 158 the extra utilization of metal layers dedicated to the power may now make it a little harder to get through-vias down to the cells for the signal routing due to extra voltage. We performed experiments to see the effect of power strips for the extra voltage on routing congestion. We compared the congestion and routing run-time in designs with and without dual-Vdd. The results are shown in Table 8.2. As the results show, our fine-grained dual-Vdd has quantitatively very limited effect on routing. It increases the routing run-time from 13% to 20%, the horizontal congestion from 2% to 24% and the vertical congestion from 6% to 12%[2]. As can be seen in the table, the routing runtime overhead becomes smaller when the size of design gets bigger. This shows that our technique has small runtime overhead on routing even for large designs.

As any other standard cell based design flow, in our approach the power nets are routed on top metal layers and are different from signal wires. For dual-Vdd design we need one more power ring and each row has only one strip like that of single-Vdd. For large designs, we need to distribute two grids for two Vdds. We can either use one dedicated metal layer for each grid or distribute both grids on one layer. In the first case, our approach is similar to single-Vdd design. However, in the second case, since we distribute two different grids on one layer, the pitch of each grid is lower than that a single grid would have, this implies larger resistance in the grids. However, the metal layers that we use can support standard

[2]The congestion rate is the occupy rate (capacity/demand) in Global Route Cells (GRCs)).

over threshold design, since we operate at near threshold, our current requirements are much lower. Therefore, although we cut in half the density of the power grid, the fact that we are in near threshold region decreases current requirement on the grid.

Since the maximum difference between two voltages is $100mV$, the noise between two domains is negligible if the driver of one domain and the receiver of the other domain are close to each other. However, when there is a long wire going from V_{ddH} to V_{ddL}, the cross-talk effect may increase. Here, we add a buffer (powered with V_{ddH}) on the long wire and close to the V_{ddL} domain. This buffer eliminates the cross-talk effect of V_{ddH} on V_{ddL}. We still have the cross-talk effect in V_{ddH} domain, but it is the same as that of single-Vdd design.

8.8.4 Cluster Optimization

To better optimize our fine-grained dual-Vdd and to reduce the leakage overhead even more, as described in Section 8.7 on page 166, we developed a cluster-optimized placement on top of the timing-driven placement in which we try to put all critical cells in a subset of rows. Then, we applied our dual-Vdd algorithm on the new design that is placed using cluster-based placement.

We compared the final design with the original timing-driven placed design in terms of performance and interface leakage. The results are shown in Figure 8.9 on the facing page. As it can be seen, the new placement strategy has a performance degradation of maximum 7% due to changing in timing-driven placement. However, for *Simple ALU* and *MAC32* the performance is improved with the new placement constraints. With this new placement, we could decrease the interface leakage overhead of our fine-grained dual-Vdd by 25% to 90% compared to that of dual-Vdd applied on designs placed with timing-driven placement with no cluster optimization during placement. This result points out that some performance penalty is to be expected if placement can be driven by dual-Vdd considerations, but that the extra power savings are interesting. This result indeed strengthens the claim that our technique is viable in practice, as it can extract extra benefits in association with new placement algorithms that cluster cells that can be powered-up. However, the development of a new placement algorithm fully oriented to this goal is out of the scope of our contribution. The interested reader can refer to [209] for an example of such an algorithm.

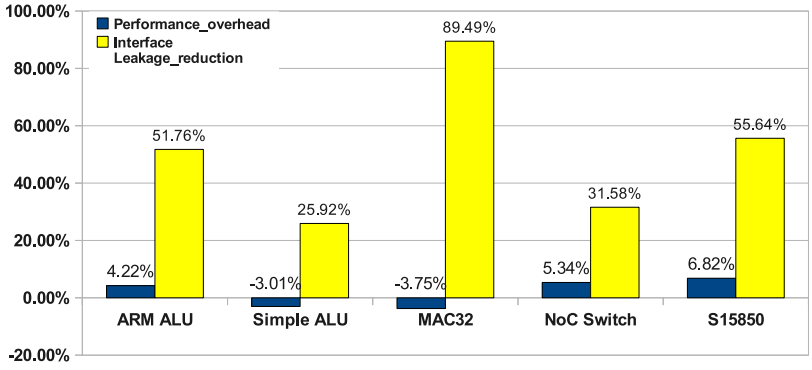


Figure 8.9: Fine-grained dual-Vdd using cluster-based placement on some benchmarks.

8.8.5 OCV Timing analysis

Finally, on-chip-variation (OCV) analysis is performed on the benchmarks using PrimeTime. We calculated the slack distribution of the critical paths in the designs with OCV. We also calculated the slack distribution after applying dual-Vdd. Figure 8.10 on the next page shows the related plots for S15850. As seen, 20% derate on the timing delay (0.8 for *early derate* and 1.2 for *late derate*) leads to negative slacks for most of the paths. As shown, after applying our dual-Vdd technique with 25% speed-up, the slack distribution of all the paths becomes positive with OCV. Also, it can be seen in the figure that 25% speed-up using dual-Vdd is able to compensate even 40% derate in the timing delay.

8.9 Summary

In this chapter, we proposed a novel dual-Vdd scheme in near threshold operation and show that our technique can be used to achieve fine-grained control of energy and performance of MPSoCs operating in this region. We have proposed a novel dual-Vdd algorithm and applied our technique on various benchmark circuits and validated using a commercial 90nm CMOS technology. Results show that our technique can achieve fine-grained control on performance of the circuits while keeping the overhead power under control.

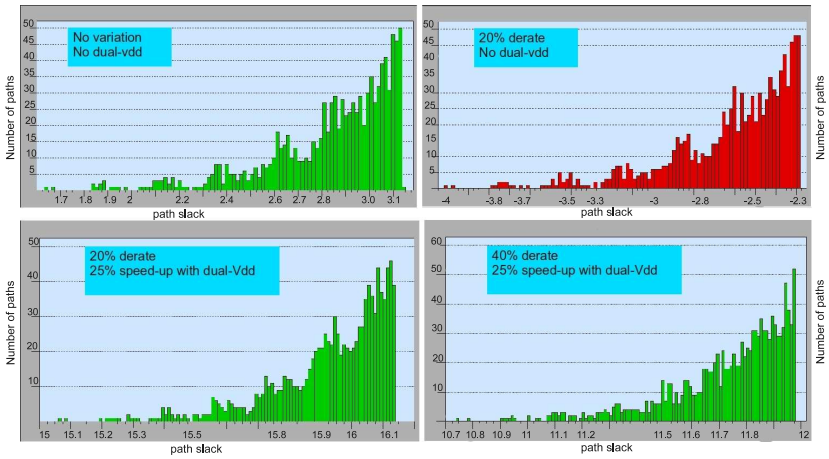


Figure 8.10: Slack distribution in S15850 using OCV analysis with and without dual-Vdd.

CHAPTER 9

Conclusions and Future Work

9.1 Conclusions

Scaling down of process technologies has increased process and dynamic variations as well as transistor wearout. Because of this, delay variations increase and impact the performance of the MPSoCs. The interconnect architecture in MPSoCs becomes a single point of failure as it connects all other components of the system together. A faulty processing element may be shut down entirely, but the interconnect architecture must be able to tolerate partial failure and variations and operate with performance, power or latency overhead. This dissertation focused on techniques at different levels of abstraction to face with the reliability and variability of interconnection network in MPSoCs.

9.1.1 Inter-cluster communication

In the cluster-based SoCs each cluster has a few number of PEs. To enable scaling the system to a larger size, clusters are connected through a scalable communication infrastructure. Network on chip is scalable and can be used for this purpose. Since clusters are isolated logically and electrically, the NoC used for inter-clusters communications is based on GALS. In this work, we developed and fabricated a test-chip on a 40 nm technology node to motivate the need for techniques facing with the reliability and variability issue in interconnection networks. Then, we proposed a reliable architecture for NoCs which takes advantage of their inherent redundancy to provide robustness against physical faults. We also pre-

sented an on-line functional testing which is able to detect logic faults in data paths and control paths of the NoC. Moreover, in this dissertation we proposed a new physical routing flow to mitigate the variation in global wire's delay and length at the physical level.

9.1.2 Intra-cluster communication

In a cluster-based multicore architecture, each cluster contains a few PEs which usually share a multi-banked L1 memory. The cluster architecture is fixed and scaling is enabled by replicating clusters. Since the number of PEs inside each cluster is relatively small, we can use a high performance and low latency interconnection infrastructure for communication between cores and the shared L1 memory. This interconnection cannot be neither BUS nor NoCs and a highly optimized special-purpose interconnect is required. In this dissertation, we implemented the design flow of a fully combinational Mesh-of-Trees (MoT) interconnection network to support high performance, single-cycle communication between processors and memories in L1-coupled processor clusters. We extended this network to be reliable and variation-tolerant. To do so, we reconfigure the design so that it can overcome the timing failure by injecting an on-demand pipeline stage in the path and increasing the latency of the read-/write transactions. If there is no variation and therefore no timing failure the network operates with single cycle latency.

9.1.3 Variation-tolerant low power MPSoCs using near-threshold computing

Power has become the primary design constraint for a large set of SoCs ranging from mobile phones, PDAs and MP3 Players to sensor nodes. The pace dictated by the Moores law has almost stopped and CMOS scaling which drove the semiconductor growth during the past several decades no longer delivers the energy gains. Indeed, beyond 65nm the supply voltage has remained essentially constant and dynamic energy efficiency improvements have stagnated, while leakage currents continue to increase. In this era, further energy gain can be achieved by moving to the near-threshold computing (NTC) domain. Operating at near threshold voltages exacerbates the effects of both global and local variations, which are already significant issues in todays advanced process technologies such as 32 nm, and we need appropriate techniques to overcome it. In this dissertation we proposed a fine-grained power management technique called row-based dualVdd which is able to compensate the effect of variations

on performance at the post-silicon stage with minimum power overhead featuring robust near-threshold design.

9.2 Future Work

My future research plans revolve around the development of high performance, reliable, variation-tolerant, and low power MPSoCs. Within this domain, in the near term I would like to work in the following areas:

9.2.1 Reliable and variation tolerant interconnection network

In my thesis I proposed reliable architectures for NoCs and low-latency interconnection network. I proposed two-channel NoC which is able to tolerate logic faults [2]. Moreover, I developed a distributed online functional testing to detect logic faults in NoC [3]. However, in deep sub-micron, delay faults due to static and dynamic variations as well as wear-out mechanisms are becoming more important than logic faults and need to be considered as one of the primary sources of system failure. In this research project I plan to target delay faults and variability in both NoC and low-latency mesh of tree network. Since dynamic variations and wear-out occur over time, post burn-in testing approaches cannot capture delay faults and we have to develop mechanisms to detect them during life-time of the chip. This detection can be performed in either online or offline mode. In addition to detection, I plan to work on reconfiguration techniques to overcome delay faults. In my thesis, I proposed a new approach [6] for tolerating delay faults due to static variations in low-latency interconnection network. It is based on on-demand pipeline insertion on critical paths. I plan to extend this work to consider dynamic variations due to temperature and Vdd drop; moreover, I would like to apply this approach to other communication infrastructures like NoCs.

9.2.2 Multi-Core Cluster with Shared-Memory HW Accelerators

Architectural heterogeneity is an effective solution to improve energy efficiency of SoC designs. General-purpose multi-core processors can be coupled with hardware functional units, whose specialized logic can achieve

10× to 100× better energy efficiency over key computation-intensive activities. In traditional acceleration techniques two distinct memory spaces for processors and accelerators are considered and this communication infrastructures implies frequent data movements from one memory space to another thus undermining acceleration benefits. Shared memory is a convenient abstraction to simplify both problems and allow for efficient processor-to-accelerator communication. In this project, we plan to work on a tightly-coupled multi-core cluster architecture where several homogeneous parallel cores are coupled to a number of dedicated hardware accelerators. To interface main program execution with HW accelerators we are going to design a dedicated architecture template where HW accelerators and processors communicate through shared memory using the variation-tolerant and low latency interconnection network proposed in Chapter 6 on page 119 and Chapter 7 on page 137.

APPENDIX A

Author's Relevant Publications

This appendix lists the works in which the author has been involved and which are relevant to the dissertation's subject.

Journal papers:

- Mohammad Reza Kakoee and Luca Benini, "Robust Near-Threshold Design with Fine-Grained Performance Tunability," IEEE Transactions on Circuits and Systems I (TCAS-I), to appear, 2012.

- Mohammad Reza Kakoee and Luca Benini, "Fine-Grained Power and Body-Bias Control for Near-Threshold Deep Sub-Micron CMOS Circuits," IEEE Transactions on Emerging and Selected Topics in Circuits and Systems, Issue 2, Volume 1, pp. 131-140, 2011.

Conference papers:

- Mohammad Reza Kakoee, Igor Loi, Luca Benini, "A Resilient Architecture for Low Latency Communication in shared-L1 processor clusters," in ACM/IEEE DATE, 2012.

- Mohammad Reza Kakoee, Valeria Bertacco, Luca Benini, "ReliNoC: A reliable network for priority-based on-chip communication", ACM/IEEE DATE, 2011, France.

- Mohammad Reza Kakoee, Valeria Bertacco, Luca Benini, "A distributed and topology-agnostic approach for on-line NoC testing", ACM/IEEE NOCS, 2011, USA.

- Mohammad Reza Kakoee, Igor Loi, Luca Benini, "A New Physical Routing Approach for Robust Bundled Signaling on NoC Links", GLSVLSI 2010, USA.
- Abbas Rahimi, Igor Loi, Mohammad Reza Kakoee, Luca Benini, "A fully-synthesizable single-cycle interconnection network for Shared-L1 processor clusters", ACM/IEEE DATE, 2011, France.
- Milos Krstic, Xin Fan, Eckhard Grass, Luca Benini, Mohammad Reza Kakoee, Christoph Heer, Birgit Sanders, Alessandro Strano, Davide Bertozzi, "Moonrake Chip - GALS Demonstrator in 40 nm CMOS Technology," IEEE International Symposium on System-on-Chip, 2011, Finland.

Bibliography

[1] P. T. Wolkotte, G. J. M. Smit, N. Kavaldjiev, J. E. Ecker, and Becker, "Energy model of networks-on-chip and bus," In Proceedings of the International Symposium on System-on-Chip, 2005.

[2] S. Pasricha, and N. Dutt, On-chip communication architectures: system on chip interconnect, ISBN 9780123738929, Elsevier / Morgan Kaufmann Publishers, 2008.

[3] "AMBA3AXI overview," ARMLtd., 2005, http://www.arm.com/products/solutions/AMBA3AXI.html.

[4] "STBus," STMicroelectronics, http://www.st.com/.

[5] D. Wingard, "Micronetwork-based integration for SOCs," in Proceedings of Design Automation Conference (DAC), 2001, pp. 673677.

[6] C. Hernandez, F. Silla, and J. Duato, A Methodology for the Characterization of Process Variation in NoC Links, Proceedings of the Design, Automation and Test in Europe (DATE), 2010.

[7] C. Hernandez, A. Roca, F. Silla, J. Flich, and J. Duato, Improving the Performance of GALS-based NoCs in the Presence of Process Variation, Proceedings of the International Symposium on Networks-on-Chip (NOCS), 2010.

[8] M. Mosin, R. Tamer, W. Xiang, A. Adnanm and M. Yehia, Provisioning On- Chip Networks under Buffered RC Interconnect Delay Variations, Proceedings of the International Symposium on Quality Electronic Design, 2007.

[9] M.R. Guthaus, D. Sylvester, R. B. Brown, Clock Tree Synthesis with Data- Path Sensitivity Matching, Proceedings of the 2008 Asia and South Pacific Design Automation Conference, 2008.

[10] ITRS roadmap, online at http://www.itrs.networks

[11] V. Khandelwal, A. Srivastava, Variability-driven formulation for simultaneous gate sizing and post-silicon tunability allocation, in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 27, Issue 4, April 2008.

[12] K. Nagaraj, S. Kundu, A Study on Placement of Post Silicon Clock Tuning Buffers for Mitigating Impact of Process Variation, Proceedings of the Design, Automation and Test in Europe (DATE), 2009.

[13] J. Tsai, L. Zhang, C. Chen, Statistical Timing Analysis Driven Post-Silicon- Tunable Clock-Tree Synthesis, Proceedings of ICCAD, 2005.

[14] A. J. Martin, M. Nystrom, Asynchronous Techniques for System-on-Chip Design, Proceedings of the IEEE, vol.94, no.6, pp.1089-1120, 2006.

[15] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, G. De Micheli, xpipes Lite: a Synthesis Oriented Design Library for Networks on Chips, Proceedings of the Design Automation and Test in Europe Conference, pp. 1188-1193, 2005.

[16] M. R. Kakoee, I. Loi, L. Benini, A New Physical Routing Approach for Robust Bundled Signaling on NoC Links, Proceedings of the 20th Great Lakes Symposium on VLSI, pp.3-8, 2010.

[17] D. Ludovici, A. Strano, G. N. Gaydadjiev, L. Benini, D. Bertozzi, Design Space Exploration of a Mesochronous Link for Cost-Effective and Flexible GALS NOCs, Proceedings of Design, Automation and Test in Europe (DATE), pp. 679-684, 2010.

[18] D. Ludovici, A. Strano, D. Bertozzi, Architecture design principles for the integration of synchronization interfaces into Network-on-Chip switches, Proceedings of the 2nd International Workshop on Network on Chip Architectures, pp. 31–36, 2009.

[19] D. Ludovici, A. Strano, D. Bertozzi, L. Benini, G. N. Gaydadjiev, Comparing tightly and loosely coupled mesochronous synchronizers in a NoC switch architecture, Proceedings of the 2009 3rd ACM/IEEE International Symposium on Networks-on-Chip, pp.244-249, 2009.

[20] A. Strano, D. Ludovici, D. Bertozzi, A Library of Dual-Clock FIFOs for Cost-Effective and Flexible MPSoCs Design, Proceedings of the

International Conference on Embedded Computer Systems: Architectures, MOdeling and Simulation (SAMOS), 2010.

[21] T. N. K. Jain, Asynchronous Bypass Channels: Improving Performance for Multi-Synchronous NoCs, NOCS 2010, pp. 51-58, 2010.

[22] S. Saponara, F. Vitullo, R. Locatelli, P. Teninge, M. Coppola, L. Fanucci, LIME: a Low-Latency and Low-Complexity On-Chip Mesochronous Link with Integrated Flow Control, Proceedings of the Euromicro Conference on Digital System Design (DSD), pp. 32-35, 2008.

[23] D. Mangano, A. Scandurra, C. Pistritto, Relieving Physical Issues in New NoC-based SoCs, Proceedings of the International Conference on Nano- Networks, 2007.

[24] D. Mangano, G. Falconeri, C. Pistritto, A. Scandurra, Effective Full-Duplex Mesochronous Link Architecture for Network-on-Chip Data-Link Layer, Proceedings of the Euromicro Conference on Digital System Design (DSD), pp. 519-526, 2007.

[25] F. Vitullo et al., Low-Complexity Link Microarchitecture for Mesochronous Communication in Networks-on-Chip, IEEE Trans. on Computers, Vol.57, issue 9, pp.1196-1201, 2008.

[26] Circuits Multi-Projects, Multi-Project Circuits, http://cmp.imag.fr

[27] D. Mangano, R. Locatelli, A. Scandurra, C. Pistritto, M. Coppola, L. Fanucci, F. Vitullo, D. Zandri, Skew Insensitive Physical Links for Network on Chip, 1st International Conference on Nano-Networks (NANO-NET 2006), Proceedings of the 1st International Conference on Nano-Networks (NANO-NET), 2006.

[28] A. T. Tran, D. N. Truong, B. Baas, A Reconfigurable Source-Synchronous On-Chip Network for GALS Many-Core Platforms, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol.29, no.6, 2010.

[29] I. M. Panades, F. Clermidy, P. Vivet, A. Greiner, Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture, Proceedings of the International Symposium on Networks-on-Chip (NOCS), pp. 139-148, 2008.

[30] D. Wentzlaff et al., On-Chip Interconnection Architecture of the Tile Processor, IEEE Micro, vol.27, no.5, pp. 15-31, 2007.

[31] D. A. IIitzky, J. D. Hoffman, A. Chun and B. P. Esparza, Architecture of the Scalable Communications Core's Network on Chip, IEEE Micro, vol.27, Issue 5, pp.62 - 74, 2007.

[32] F. Clermidy, R. Lemaire, X. Popon, D. Ktenas, Y. Thonnart, An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application, Proceedings of the Euromicro Conference on Digital System Design (DSD), pp. 62-74, 2009.

[33] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, N. Wehn, A 477mW NoC-based Digital Baseband for MIMO 4G SDR, Proceedings of the Internation Solid State Circuits (ISSC), pp. 278-279, 2010.

[34] Y. Thonnart, P. Vivet, F. Clermidy, A Fully-Asynchronous Low-Power Framework for GALS NoC Integration, Proceedings of the Design, Automation and Test in Europe (DATE), pp. 33-38, 2010.

[35] M. Krstic et al., Globally Asynchronous, Locally Synchronous Circuits: Overview and Outlook, IEEE Design and Test of Computers, vol. 24, no. 5, pp. 430-441, Sept. 2007.

[36] S. Vangal et al., An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS, IEEE Journal of Solid-State Circuits, Vol.43, Issue 1, pp.29-41,2008.

[37] E. Flamand, Strategic Directions Towards Multicore Application Specific Computing, Proceedings of the Design, Automation and Test in Europe (DATE), pp. 1266, 2009.

[38] S. Borkar, Design Perspectives on 22nm CMOS and Beyond, Proceedings of the Design Automation Conference (DAC), 2009.

[39] R. Dobkin, V. Vishnyakov, E. Friedman, R. Ginosar, An Asynchronous Router for Multiple Service Levels Networks on Chip, Proceedings of ASYNC, pp. 44-53, 2005.

[40] T. Bjerregaard, J. Sparso, A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip, Proceedings of the Design, Automation and Test in Europe (DATE), pp. 1226-1231, 2005.

[41] F. Mo and R.K. Brayton, "A Semi-Detailed Bus Routing Algorithm with Variation Reduction," ISPD, pp. 143-150, 2007.

[42] H.Xiang, X.Tang and M.D.F. Wong, "Bus-Driven Floorplanning," ICCAD, pp. 66-73, 2003.

[43] F. Rafiq, et.al, "Integrated Floorplanning with Buffer/Channel Insertion for Bus- Based Microprocessor Designs," ISPD, pp. 56-61, 2002.

[44] J.H.Y. Law and E.F.Y. Young, "Multi-Bend Bus Driven Floorplanning," ISPD, pp. 113-120, 2005.

[45] G. Persky and L.V. Tran, "Topological Routing of Multi-Bit Data Buses," DAC, pp. 679-682, 1984.

[46] L. Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm," IEEE Computers, Jan. 2002.

[47] http://www.synopsys.com

[48] http://www.tcl.tk

[49] Naveed Sherwani, Algorithms for VLSI Physical Design Automation, Kluwer Academic Publishers, 1999.

[50] J. Huang, et.al, "An Efficient Timing Driven Global Routing Algorithm," DAC, pp. 560-600, 1993.

[51] C. E. Leiserson, et.al, "Algorithms for routing and testing routability of planar vlsi layouts," STOC, pp. 69-78, 1985.

[52] M. Galles. "Scalable Pipelined Interconnect for Distributed Endpoint Routing," The SGI SPIDER Chip. Hot Interconnects Symposium IV, pp. 141-146, 1996.

[53] Kodi, A.K. Sarathy, A. Louri, A, "iDEAL: Inter-router Dual-Function Energy and Area-Efficient Links for Network-on-Chip (NoC) Architectures," ISCA, pp. 241-250, 2008.

[54] J. Dally and B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2004.

[55] http://www.eetimes.com/story/OEG20030623S0027

[56] D. Bertozzi and L. Benini, "Xpipes: A network-on-chip architecture for gigascale systems-on-chip," IEEE Circuits and Systems Magazine, vol. 4, Issue 2, pp. 18-31, 2004.

[57] R. Collins and L. P. Carloni, "Topology-Based Optimization of Maximal Sustainable Throughput in a Latency-Insensitive System," DAC, pp. 410-415, 2007.

[58] Simone Medardoni, Marcello Lajolo, Davide Bertozzi, "Variation tolerant NoC design by means of self-calibrating links," DATE, pp. 1402-1407, 2008.

[59] G. Campobello, et.al, "GALS networks on chip: a new solution for asynchronous delay-insensitive links," DATE, pp. 160-165, 2006.

[60] http://www.intel.com/research/platform/terascale/teraflops.htm

[61] Liang Deng, et.al, "Buffer insertion under process variations for delay minimization," ICCAD, pp. 317-321, 2005.

[62] William J. Dally, "Enabling Technology for On-Chip Interconnection Networks," NoCS Keynote, 2007.

[63] Kangmin Lee, et.al, "Low-power network-on-chip for high-performance SoC design," IEEE Transactions on VLSI Systems, Vol. 14, Issue 2, pp.148-160, 2006.

[64] Mesgarzadeh, B. Svensson, C. Alvandpour, A., "A new mesochronous clocking scheme for synchronization in SoC", ISCAS, pp 605-608, 2004.

[65] Giacomo Paci, et.al, "Effectiveness of adaptive supply voltage and body bias as post-silicon variability compensation techniques for full-swing and low-swing on-chip communication channels," DATE , pp. 1404-1409, 2009.

[66] Davide Bertozzi, et.al, "Error Control Schemes for On-Chip Communication Links: The Energy-Reliability Tradeoff," IEEE Trans. on CAD, Vol. 24, Issue 6, pp. 818-831, 2005.

[67] Ravishankar Arunachalam, et.la, "Optimal Shielding/Spacing Metrics for Low Power Design," ISVLSI, pp. 167-172, 2003.

[68] J. W. McPherson, "Reliability challenges for 45nm and beyond," Proc. ACM/IEEE *DAC*, pp. 176-181, 2006.

[69] S. Borkar,"Microarchitecture and design challenges for gigascale integration," Proc. ACM/IEEE *MICRO*, keynote address, pp. 3-3, 2004.

[70] S. Mitra, et al., "Robust System Design to Overcome CMOS Reliability Challenges," Emerging and Selected Topics in Circuits and Systems, IEEE Journal on , vol.1, no.1, pp. 30-41, March 2011

[71] S. Borkar,"Designing reliable systems from unreliable components: the challenges of transistor variability and degradation," IEEE Micro, Vol. 25, No. 6, pp. 10-16, 2005.

[72] David Fick et al., "Vicis: A Reliable Network for Unreliable Silicon," Proc. ACM/IEEE *DAC*, pp. 812-817, 2009.

[73] W.J.Dally "Virtual-Channel Flow Control," Proc. ACM/IEEE *ISCA*, pp. 60-68, 1990.

[74] N.Banerjee et al., "A Power and Performance Model for Network-on-Chip Architectures," Proc. ACM/IEEE *DATE*, pp. 21250, 2004.

[75] A.Mello et al., "Virtual Channels in Networks-on-Chip: Implementation and Evaluation on Hermes NoC," Proc. ACM/IEEE *SBCCI*, pp. 178-183, 2005.

[76] R.Mullins et al., "Low-Latency Virtual-Channel Routers for On-Chip Networks," Proc. ACM/IEEE *ISCA*, pp. 188, 2004.

[77] A.Kumar et al., "Express Virtual Channels: Towards the ideal Interconnection Fabric," ACM Computer Architecture News, Vol. 35, No. 2, 2007.

[78] L.S. Peh et al., "A Delay Model and Speculative Architecture for Pipelined Routers," Int. Symp. on High-Performance Computer Architecture, pp.255-266, 2001.

[79] S.Noh et al. "Multiplane Virtual Channel Router for Network-on-Chip Design," Proc. IEEE *ICCE*, pp. 348-351, 2006.

[80] T. Huang et al., "Virtual Channels Planning for Networks-on-Chip," Proc. the 8th International Symposium on Quality Electronic Design, pp. 879-884, 2007.

[81] S.S. Mukherjee et al., "The Alpha 21364 network architecture," Hot Interconnects 9, pp 113-117, 2001.

[82] P. Gratz, "Implementation and Evaluation of On-Chip Network Architectures," Computer Design, pp. 477-484, 2006.

[83] A. Kumar et al., "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," Computer Design, pp. 63-70, 2007.

[84] D. Becker et al., "Allocator implementations for network-on-chip routers," Proc. Conf. on High Performance Computing Networking, Storage and Analysis, pp. 52:1-52:12, 2009.

[85] S. Wung et al., "Dynamic Channel Flow Control of Networks-on-Chip Systems for High Buffer Efficiency," Workshop on Signal Processing Systems, pp. 493-498, 2007.

[86] Min Zhang et al., "Low-Cost VC Allocator Design for Virtual Channel Wormhole Routers in Networks-on-Chip," Networks-on-Chip, pp, 207-208, 2008.

[87] N. Kavaldjiev et al., "A virtual channel router for on-chip networks," Int. SOC Conference, pp. 289-290, 2004.

[88] M. Galles, "Spider: a high-speed network interconnect," IEE Micro, 17(1):34-39, 1997.

[89] F. Gilabert et al., "Improved Utilization of NoC Channel Bandwidth by Switch Replication for Cost-Effective Multi-Processor Systems-on-Chip," Proc. ACM/IEEE *NoCS*, pp. 165-172, 2010.

[90] W. J. Dally et al. "The reliable router: A reliable and high-performance communication substrate for parallel computers," Proc. *PCRCW*, pp. 241-255, 1994.

[91] M. B. Taylor et al., "The Raw microprocessor: A computational fabric for software circuits and general purpose programs," IEEE Micro, Vol. 22, No. 2, pp. 25-35, 2002.

[92] K. Constantinides et al. "BulletProof: a defect-tolerant CMP switch architecture," Proc. IEEE *HPCA*, pp. 5-16, 2006.

[93] D. Park et al., "Exploring fault-tolerant network-on-chip architectures," Proc. ACM/IEEE *DSN*, pp. 93-104, 2006.

[94] Qiaoyan Yu; Meilin Zhang; Ampadu, P.; , "Exploiting inherent information redundancy to manage transient errors in NoC routing arbitration," Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on, pp.105-112, 1-4 May 2011.

[95] D. Fick et al.,"A highly resilient routing algorithm for fault-tolerant NoCs," Proc. ACM/IEEE *DATE*, pp. 21-26, 2009.

[96] M. E. Gomez et al., "An efficient fault-tolerant routing methodology for meshes and tori," IEEE Computer Architecture Letters, vol. 3, No. 1, pp. 3-3, 2004.

[97] S. Rodrigo et al., "Addressing Manufacturing Challenges with Cost-Efficient Fault Tolerant Routing,", Proc. ACM/IEEE *NoCS*, pp. 25-32, 2010.

[98] C.-T. Ho et al., "A new approach to fault-tolerant wormhole routing for mesh-connected parallel computers," IEEE Trans. on Computers, Vol. 53, No. 4, pp. 427-439, 2004.

[99] Young Jin Yoon et al."Virtual channels vs. multiple physical networks: a comparative analysis," Proc. ACM/IEEE *DAC*, PP. 162-165, 2010.

[100] M.R. Kakoee, V. Bertacco, L. Benini, "A distributed and topology-agnostic approach for on-line NoC testing," Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on , pp. 113-120, 1-4 May 2011.

[101] A. Strano, et al., "Exploiting Network-on-Chip structural redundancy for a cooperative and scalable built-in self-test architecture," ACM/IEEE DATE, pp.1-6, 14-18 March 2011.

[102] J. Flich et al., "An efficient Implementation of Distributed Routing Algorithms for NoCs," Proc. ACM/IEEE *NoCS*, pp. 87-96 , 2008.

[103] W. Peterson et al., Error-Correcting Codes, 2nd ed. Cambridge, MA: MIT Press, 1972.

[104] A. Agarwal et al., "Process variation in embedded memories: Failure analysis and variation aware architecture," IEEE J. Solid-State Circuits, Vol. 40, No. 9, pp. 1804-1814, 2005.

[105] R. Ubar et al., Testing Strategies for Network on Chip, Kluwer Academic Publisher, pp. 131-152, 2003.

[106] F. Fazzino et al, Noxim: Network-on-chip simulator [Online]. http://noxim.sourceforge.net

[107] Christian Bienia et al., "The PARSEC benchmark suite: characterization and architectural implications," Proc. ACM *PACT*, pp. 72-81, 2008.

[108] P. S. Magnusson et al., "Simics: A full system simulation platform," IEEE Computer, Vol. 35, No. 2, pp. 50-58, 2002.

[109] M. M. K. Martin et al., "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," SIGARCH Computer Architecture News, Vol. 33, No. 4, pp. 92-99, 2005.

[110] L.-S. Peh et al., "GARNET: A detailed on-chip network model inside a full-system simulator," Proc. IEEE *ISPASS*, pp. 33-42, 2009.

[111] ITRS Home-2010, 2010 International Technology Roadmap for Semiconductors. [Online]. Available: http://www.itrs.net/home.html

[112] http://techresearch.intel.com/articles/Tera-Scale/1421.htm.

[113] http://techresearch.intel.com/articles/Tera-Scale/1449.htm.

[114] Intel "Intel News Release: Intel Unveils New Product Plans for High Performance Computing," http://www.intel.com/pressroom/archive/releases/20100531comp.htm, May 2010.

[115] L. Seiler, et al., "Larrabee: A Many-Core x86 Architecture for Visual Computing," IEEE Micro, vol.29, no.1, pp.10-21, Jan.-Feb. 2009.

[116] http://www.tilera.com/products/processors.php.

[117] T. Bjerregaard, Shankar Mahadevan, "A survey of research and practices of network-on-chip," ACM Computer Survey, Vol. 38, No. 1, 2006.

[118] M.R. Kakoee, et al., "ReliNoC: A Reliable Network for Priority-Based On-Chip Communication," Proc. ACM/IEEE *DATE*, pp. 667-672, 2011.

[119] Wei Song Edwards, D. Nunez-Yanez, J.L. Dasgupta, S., "Adaptive stochastic routing in fault-tolerant on-chip networks," Proc. ACM/IEEE *NoCS*, pp. 32-37 , 2009.

[120] C. Grecu, et al., "On-line fault detection and location for NoC interconnects," Proc. IEEE *IOLTS*, pp. 6-11, 2006.

[121] A. Alaghi, N. Karimi, M. Sedghi, Z. Navabi, "Online NoC Switch Fault Detection and Diagnosis Using a High Level Fault Model," Proc. IEEE *DFT*, pp. 21-29, 2007.

[122] Young Hoon Kang, Taek-Jun Kwon, Jeffrey Draper, "Fault-Tolerant Flow Control in On-chip Networks," Proc. ACM/IEEE *NoCS* , pp. 79-86, 2010.

[123] E. Cota, F.L. Kastensmidt, M. Cassel, M. Herve, P. Almeida, P. Meirelles, A. Amory, M. Lubaszewski, "A High-Fault-Coverage Approach for the Test of Data, Control and Handshake Interconnects in Mesh Networks-on-Chip," IEEE Trans. on Computers, Vol. 57, No. 9, pp. 1202-1215, 2008.

[124] M. Herve, P. Almeida, F.L Kastensmidt, E. Cota, M. Lubaszewski, "Concurrent test of Network-on-Chip interconnects and routers," Proc. IEEE *LATW*, pp. 1-6, 2010.

[125] M. Cuviello, S. Dey, X. Bai, and Y. Zhao, "Fault Modeling and Simulation for Crosstalk in System-on-Chip Interconnects," Proc. IEEE/ACM Int'l Conf. Computer-Aided Design, pp. 297-303, 1999.

[126] P.P. Pande, et al., "Design of Low Power and Reliable Networks on Chip through Joint Crosstalk Avoidance and Forward Error Correction Coding," Proc. 21st IEEE Intl Symp. Defect and Fault Tolerance in VLSI Systems, pp. 466-476, 2006.

[127] T. Bengtsson, et al., "Offline testing of delay faults in NoC interconnects", In Proceeding of the 9th EUROMICRO Conference on Digital System Design, pp. 677-680, IEEE 2006.

[128] J.Raik, V.Govind, R.Ubar, "Test Configurations for Diagnosing Faulty Links in NoC Switches", Proc. ETS, 2007.

[129] Aktouf, C. "A Complete Strategy for Testing an on-chip Multiprocessor Architecture". In IEEE Design Test of Computers, vol. 19, no. 1, pp. 18-28, 2002.

[130] Wu, Y. and MacDonald, P. "Testing ASICs with Multiple Identical Cores", In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 22, no. 3, pp. 327-336, 2003.

[131] C. Liu, Z. Link and D.K. Pradhan. "Reuse-based test access and integrated test scheduling for Network-on-Chip", In Proc. Design, Automation and Test in Europe, vol. 1, pp. 303-308, March 2006.

[132] E. Cota, L. Carro, and M. Lubaszewski, "Reusing an On-Chip Network for the Test of Core-Based Systems," ACM Trans. Design Automation of Electronic Systems, vol. 9, no. 4, pp. 471-499, 2004.

[133] A.M. Amory, E. Briao, E. Cota, M. Lubaszewski, and F.G. Moraes, "A Scalable Test Strategy for Network-on-Chip Routers," Proc. IEEE Int'l Test Conf., p. 9, 2005.

[134] C. Grecu, et al., "Methodologies and Algorithms for Testing Switch-Based NoC Interconnects," Proc. 20th IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems, pp. 238-246, 2005.

[135] K. Stewart and S. Tragoudas, "Interconnect Testing for Networks on Chips," Proc. 24th IEEE VLSI Test Symp., p. 6, 2006.

[136] Zheng Y, et al., "Accelerating strategy for functional test of NoC communication fabric," IEEE Asian Test Symposium. pp.224227, 2010.

[137] Raik J, Govind V Ubar R, "An external test approach for network-on-a-chip switches," 15th Asian Test Symposium, pp. 437442, 2006.

[138] S.Y.Lin, C.C.Hsu, A.Y.Wu, "A Scalable Built-In Self-Test/Self-Diagnosis Architecture for 2D-mesh Based Chip Multiprocessor Systems", IEEE Int. Symp. on Circuits and Systems, pp.2317-2320, 2009

[139] E. Mintarno, et al., "Self-Tuning for Maximized Lifetime Energy-Efficiency in the Presence of Circuit Aging," IEEE Transactions on CAD, vol.30, no.5, pp.760-773, May 2011.

[140] J. Henkel, et al., "Design and architectures for dependable embedded systems," ACM CODES+ISSS , USA, 2011.

[141] Pei Songwei, Li Huawei, Li Xiaowei, "A unified test architecture for on-line and off-line delay fault detections," IEEE VLSI Test Symposium (VTS), pp.272-277, 1-5 May 2011.

[142] S. Natarajan, et al., "Path coverage based functional test generation for processor marginality validation," IEEE International Test Conference (ITC), pp.1-9, 2-4 Nov. 2010.

[143] A.Krstic and K.T.Cheng, "Delay Fault Testing for VLSI Circuits," Boston, MA:Kluwer,1998.

[144] M. Sharma, J.H. Patel, "Testing of critical paths for delay faults," IEEE International Test Conference, pp.634-641, 2001.

[145] T. Bengtsson, S. Kumar, Z. Peng, "Application Area Specific System Level Fault Models: A Case Study with a Simple NoC Switch," Proc. *IDT*, 2006.

[146] M. R. Kakoee, L. Benini, "Fine-Grained Power and Body-Bias Control for Near-Threshold Deep Sub-Micron CMOS Circuits," IEEE Transactions on Emerging and Selected Topics in Circuits and Systems, vol.1, no.2, pp.131-140, June 2011.

[147] A. Ghosh et al., "A centralized supply voltage and local body bias-based compensation approach to mitigate within-die process variation," in ACM/IEEE ISLPED, 2009, pp. 45-50.

[148] Plasma microprocessor description (http://www.opencores.org)

[149] T. Ban, L.A. de Barros Naviner, "A simple fault-tolerant digital voter circuit in TMR nanoarchitectures," NEWCAS Conference (NEWCAS), 2010 8th IEEE International , pp.269-272, 20-23 June 2010.

[150] S. Akram, R. Kumar, D. Chen, "Workload adaptive shared memory multicore processors with reconfigurable interconnects," in Proc. of the 7th IEEE Symposium on Application-Specific Processors, SASP, July 2009, pp. 7-14.

[151] NVDIA, The next generation CUDA architecture, code named Fermi,

$$online$$

. Available: www.nvidia.comobjectfermi_architecture.html

[152] J. Nickolls and W. J. Dally, "The GPU computing era," IEEE Micro, vol. 30, no. 2, pp 56 - 69, April 2010.

[153] F. Angiolini, P. Meloni, S. Carta, L. Raffo, and L. Benini, "A layout-aware analysis of networks-on-chip and traditional interconnects for mpsocs, " IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems," vol. 26, no. 3, pp. 421-434, March 2007.

[154] ARM Ltd., The Advanced Microcontroller Bus Architecture (AMBA) Homepage. [Online] . Available: www.arm.com/products/solutions/AMBAHomePage.html

[155] S. Satpathy, Z. Foo, B. Giridhar, D. Sylvester, T. Mudge, D. Blaauw, "A 1.07 Tbit/s 128x128 swizzle network for SIMD processors," IEEE Symposium on VLSI Circuits (VLSI-Symp), pp. 81-82, 2010.

[156] K. Lee et al., "A 51 mW 1.6 GHz on-chip network for low-power heterogeneous SoC platform," in IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers, 2004, pp. 152-153.

[157] Se-Joong Lee, et. al., "An 800MHz star-connected on-chip network for application to systems on a chip," IEEE Digest of International Solid State Circuits Conference, vol. 1, 2003, pp. 468-489.

[158] M. Horowitz, and W. Dally, "How scaling will change processor architecture," In IEEE International Solid-State Circuits Conference (ISSCC), 2004, pp. 132-133.

[159] George L. Yuan , Ali Bakhoda , Tor M. Aamodt, "Complexity effective memory access scheduling for many-core accelerator architectures," in Proc. of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009, pp. 34-44.

[160] D. Kim et al., "Memory-centric network-on-chip for power efficient execution of task-level pipeline on a multi-core processor," IET Computers & Digital Techniques, vol. 3(5), pp. 513-524, September 2009.

[161] Kim et al., "Solutions for real chip implementation issues of NoC and their application to memory-centric NoC," Int.Symp. on Networks-on-Chip, 2007, pp.30-39.

[162] OpenSPARC T1. [online] .Available: www.opensparc.net/opensparc-t1/index.html

[163] A. O. Balkan, G. Qu, and U. Vishkin, "A mesh-of-trees interconnection network for single-chip parallel processing," In Proc. of the App.-Specific Systems, Architectures and Processors (ASAP), 2006, pp. 73 80.

[164] Aydin O. Balkan , Gang Qu , Uzi Vishkin, "An area-efficient high-throughput hybrid interconnection network for single-chip parallel processing," in Proc. of the 45th annual conference on Design automation, 2008, pp. 435-440.

[165] R. I. Greenberg and L. Guan, "On the area of hypercube layouts," Information Processing Letters, 84:4146,2002.

[166] A. DeHon, "Compact, multilayer layout for butterfly fat-tree," In Proc. of Symposium on Parallel Algorithms and Architectures (SPAA), 2000, pp. 206215.

[167] C.-H. Yeh, "Optimal layout for butterfly networks in multilayer VLSI," In Proc International Conference on Parallel Processing (ICPP), 2003, pp. 379 388.

[168] C. E. Leiserson, "Fat trees: Universal networks for hardware-efficient supercomputing," IEEE Trans. Computer, 34(10):892901, Oct. 1985.

[169] Plurality, Ltd. The hyperCore architecture, white paper, January 2010.

[170] N. Bayer, A. Peleg, "Shared memory system for a tightly-coupled multiprocessor," Pub. no. WO/2009/060459, 2009.

[171] A. Rahimi, I. Loi, M.R. Kakoee, L. Benini "A Fully-Synthesizable Single-Cycle Interconnection Network for Shared-L1 Processor Clusters ," in Proc. of the ACM/IEEE DATE, 2011.

[172] Plurality Ltd. The HyperCore Processor. www.plurality.com/hypercore.html

[173] ST Microelectronics and CEA. Platform 2012: A Many-core programmable accelerator for ultra efficient embedded computing in nanometer technology. 2010.

fermi_white_papers/NVIDIA_

Fermi_Compute_Architecture_Whitepaper.pdf, 2010.

[174] Tilera Corp. Product Brief. Tilepro64 processor. 2008.

[175] K. Bowman, et al., "Circuit techniques for dynamic variation tolerance," in ACM/IEEE DAC, pp. 4-7, 2009.

[176] K.A. Bowman, J.W. Tschanz et.al, "A 45 nm Resilient Microprocessor Core for Dynamic Variation Tolerance," Solid-State Circuits, IEEE Journal of , vol.46, no.1, pp.194-208, Jan. 2011

[177] D. Bull, S. Das et al. "A Power-Efficient 32 bit ARM Processor Using Timing-Error Detection and Correction for Transient-Error Tolerance and Adaptation to PVT Variation," Solid-State Circuits, IEEE Journal of , vol.46, no.1, pp.18-31, Jan. 2011.

[178] D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," in Micro-36, pp. 7-18, 2003.

[179] http://www.gaisler.com

[180] http://www.st.com

[181] B. Calhoun, A. Wang and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1778-1786, September 2005.

[182] R. Gonzalez, B. M. Gordon, M. A. Horowitz, "Supply and threshold voltage scaling for low power CMOS," *IEEE Journal of Solid-State Circuits*, Vol. 32, no. 8, pp. 1210-1216, August 1997.

[183] A. Tajalli, Y. Leblebici, "Design Trade-offs in Ultra-Low-Power Digital Nanoscale CMOS," *IEEE Transactions on Circuits and Systems I*, Vol. PP, No. 99, pp. 1-12, March 2011.

[184] A. Wang, A. Chandrakasan, "A 180mV FFT processor using subthreshold circuit techniques," *IEEE ISSCC* , pp. 292-295, USA, 2004.

[185] C. Kim, H. Soeleman, K. Roy, "Ultra-Low-Power DLMS Adaptive Filter for Hearing Aid Applications," *IEEE Transactions on VLSI Systems*, Vol. 11, no. 6, pp. 1058-1067, December 2003.

[186] B. Zhai, S. Pant, et.al, "Energy efficient subthreshold processor design," *IEEE Transactions on VLSI Systems*, Vol. 17, no. 8, pp. 1127-1137, August 2009.

[187] David Bol, et.al, "Technology flavor selection and adaptive techniques for timing-constrained 45nm subthreshold circuits," *ACM ISLPED* , pp. 21-26, USA, 2009.

[188] Markovic, D. et.al, "Ultralow-Power Design in Near-Threshold Region," *Proceedings of the IEEE*, Vol. 98, no. 2, pp. 237-252, February 2010.

[189] Bo Zhai et.al, "Theoretical and Practical Limits of Dynamic Voltage Scaling," *ACM/IEEE DAC*, pp. 868-873, USA, 2004.

[190] R. G. Dreslinski et.al, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proceedings of the IEEE*, Vol. 98, no. 2, pp. 253-266, February 2010,

[191] Bo Zhai, et.al, "Energy efficient near-threshold chip multi-processing," *ACM ISLPED* , pp. 32-37, USA, 2007.

[192] R. Dreslinski, et.al, "An energy efficient parallel architecture using near threshold operation," *IEEE PACT*, pp. 175-185, Romania, 2007.

[193] Yu Pu, et.al, "An ultra-low-energy/frame multi-standard JPEG co-processor in 65nm CMOS with sub/near-threshold power supply," *IEEE ISSCC* , pp. 146-147, USA, 2009.

[194] B. Nezamfar et.al, "Energy-Performance Tunable Logic," *IEEE CICC* , pp. 183-186, USA, 2009.

[195] S. Sakiyama et.al, "An On-Chip High-Efficiency and Low-Noise DC/DC Converter Using Divided Switches with Current Control Technique," *IEEE ISSCC* , pp. 156-158, USA, 1999.

[196] Siyuan Zhou, and Gabriel, "A High Efficiency, Soft Switching DC-DC Converter With Adaptive Current-Ripple Control for Portable Applications," *IEEE Transactions on Circuits and Systems*, Vol. 53, no. 4, pp. 319-323, April 2006.

[197] T Hirose, et.al, "Power-supply circuits for ultralow-power sub-threshold MOS-LSIs," *IEICE Electronics Express*, Vol. 3, no. 22, pp. 464-468, November 2006.

[198] H. Wu, et.al, "Post-Placement Voltage Island Generation under Performance Requirement," *IEEE/ACM ICCAD*, pp. 309-316, 2005.

[199] H. Wu, et.al, "Timing-Constrained and Voltage-Island-Aware Voltage Assignment," *ACM/IEEE DAC*, pp. 429-432, 2006.

[200] H. Wu, M. Wong, "Improving Voltage Assignment by Outlier Detection and Incremental Placement," *ACM/IEEE DAC*, pp. 459-464, 2007.

[201] D. E. Lackey, et.al, "Managing power and performance for system-on-chip designs using voltage islands," *ACM/IEEE ICCAD*, pp. 195-202, 2002.

[202] J. Hu, et.al, "Architecting voltage islands in core-based system-on-a-chip designs," *ACM/IEEE ISLPED*, pp. 180-185, 2004.

[203] D.Bertozzi, S.Bonesi, L.Benini, E.Macii, "Process Variation Tolerant Pipeline Design Through a Placement-Aware Multiple Voltage Island Design Style", *ACM/IEEE DATE*, pp. 967-972, 2008.

[204] M. Anis and M. Elmasry, Multi-Threshold CMOS Digital Circuits, Managing Leakage Power. Norwell, MA: Kluwer, 2003.

[205] B.H. Calhoun, et.al, "A leakage reduction methodology for distributed MTCMOS," *IEEE Journal of Solid-State Circuits*, Vol. 39, No. 5, pp. 818-826, May 2004.

[206] Michael Keating, et.al, Low Power Methodology Manual for System on Chip Design, Springer Publications, NewYork, 2007.

[207] N. Verma, et.al, "Nanometer MOSFET variation in minimum energy subthreshold circuits," *IEEE Transactions on Electron Devices*, Vol.55, no. 1 , pp. 163-174, January 2008.

[208] Frederick S. Hillier, Gerald J. Liberman,"Introduction to Operations Research," McGraw-Hill, *Eight Edition*.

[209] S. Kulkarni, D. Sylvester, D. Blaauw, "Design-Time Optimization of Post-Silicon Tuned Circuits using Adaptive Body Bias", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 27, No. 3, pp. 481-494, March 2008.

[210] Basab Datta, Wayne Burleson," A 45.62 13.4w 7.1v/v resolution subthreshold based digital process-sensing circuit in 45nm CMOS," *ACM Great Lakes Symposium on VLSI, GLSVLSI*, pp. 133-138, May 2011.

[211] S. Das, C. Tokunaga, S. Pant, W.-H. Ma, S. Kalaiselvan, K. Lai, D.M. Bull, D. Blaauw, "RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance," *IEEE Journal of Solid-State Circuits,* Vol. 44, No. 1, pp. 32-48, Jan. 2009.

[212] R.Rao, K.Jenkins and J.Kim, "A completely digital on-chip circuit for local random variability measurement," *IEEE International Solid States Circuits Conference*, pp. 412-623, Feb. 2008.

[213] M.Meterelliyoz, P.Song, F.Stellari, J.Kulkarni and K.Roy, "A high sensitivity process variation sensor utilizing subthreshold operation", *In Proceedings of IEEE Custom Integrated Circuits Conference*, pp. 125-128, Sept. 2008.