Alma Mater Studiorum - Università di Bologna

Dottorato di Ricerca in Ingegneria Elettronica,
Informatica e delle Telecomunicazioni

Ciclo XXI

Settore Scientifico Disciplinare di afferenza: ING/INF01 Elettronica

Tecniche di progettazione tollerante alle variazioni per circuiti digitali in tecnologie nanometriche

Tesi di Dottorato di:

Giacomo Paci

Relatore :

Chiar. mo Prof. Ing. Luca Benini

Coordinatore Dottorato:

Chiar. ma Prof.ssa Ing. Paola Mello

Esame finale anno 2009

VARIATION RESILIENT DESIGN TECHNIQUES OF NANOMETER-SCALE DIGITAL INTEGRATED CIRCUITS

A dissertation submitted to the
DEPARTEMENT OF ELECTRONICS, COMPUTER SCIENCE AND SYSTEMS
OF UNIVERSITY OF BOLOGNA

for the degree of Doctor of Philosophy

presented by
GIACOMO PACI
born June 27, 1979

March 2009

Keywords

Process Variation

Ultra-Low-Power

MultiProcessor System-on-Chip (MPSoC)

Hot Spot

Thermal Management

Self-timed Logic

Network-on-Chip (NoC)

Variation Compensation

Contents

List of Figures

List of Tables

Chapter 1

Abstract

The digital electronic market development is founded on the continuous reduction of the transistors size, to reduce area, power, cost and increase the computational performance of integrated circuits. This trend, known as technology scaling, is approaching the nanometer size.

The lithographic process in the manufacturing stage is increasing its uncertainty with the scaling down of the transistors size, resulting in a larger parameter variation in future technology generations. Furthermore, the exponential relationship between the leakage current and the threshold voltage, is limiting the threshold and supply voltages scaling, increasing the power density and creating local thermal issues, such as hot spots, thermal runaway and thermal cycles. In addiction, the introduction of new materials and the smaller devices dimension are reducing transistors robustness, that combined with high temperature and frequently thermal cycles, are speeding up wear out processes. Those effects are no longer addressable only at the process level.

Consequently the deep sub-micron devices will require solutions which will imply several design levels, as system and logic, and new approaches called Design For Manufacturability (DFM) and Design For Reliability. The purpose of the above approaches is to bring in the early design stages the awareness of the device reliability and manufacturability, in order to introduce logic and system able to cope with the yield and reliability loss.

The ITRS roadmap [1] suggests the following research steps to integrate the design for manufacturability and reliability in the standard CAD automated design flow: i) The implementation of new analysis algorithms able to predict the system thermal behavior with the impact to the power and speed performances. ii) High level wear out models able to predict the mean time to failure of the system (MTTF). iii) Statistical performance analysis able to predict the impact of the process variation, both random and systematic.

The new analysis tools have to be developed beside new logic and system strategies to cope with the future challenges, as for instance: i) Thermal management strategy that increase the reliability and life time of the devices acting to some tunable parameter,such as *supply voltage* or *body bias*. ii) Error detection logic able to interact with compensation techniques as *Adaptive Supply Voltage* ASV, *Adaptive Body Bias* ABB and error recovering, in order to increase yield and reliability. iii) architectures that are fundamentally resistant to variability, including locally asynchronous designs, redundancy, and error correcting signal encodings (ECC). The literature already features works addressing the prediction of the MTTF [2], papers focusing on thermal management in the general purpose chip as [3], and publications on statistical performance analysis as [4].

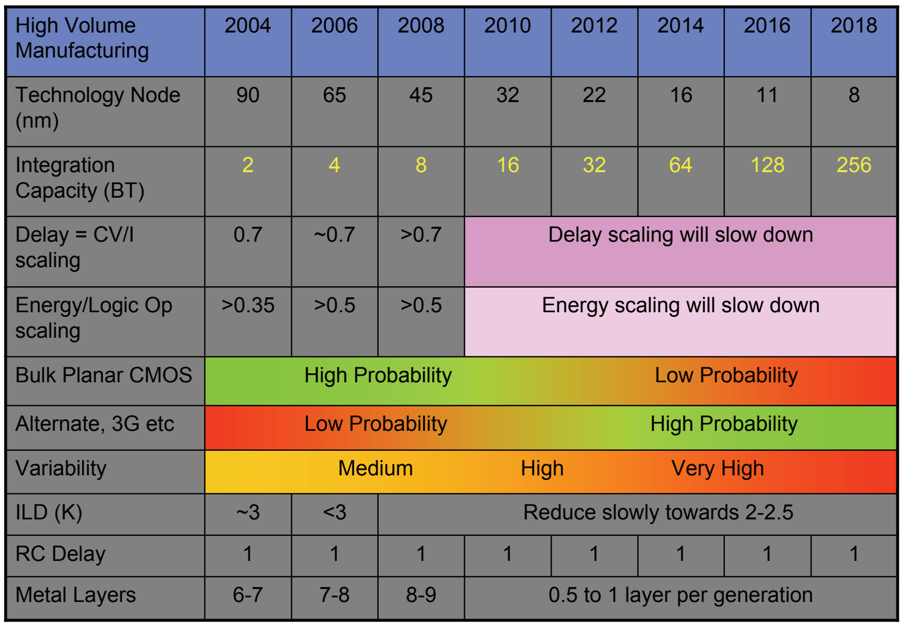
In my Phd research activity, I investigated the need for thermal management in future embedded low-power Network On Chip (NoC) devices.I developed a thermal analysis library, that has been integrated in a NoC cycle accurate simulator [5] and in a FPGA based NoC simulator [6]. The results have shown that an accurate layout distribution can avoid the onset of hot-spot in a NoC chip. Furthermore the application of thermal management can reduce temperature and number of thermal cycles, increasing the system reliability. Therefore the thesis advocates the need to integrate a thermal analysis in the first design stages for embedded NoC design.

Later on, I focused my research in the development of statistical process variation analysis tool that is able to address both random and systematic variations. The tool was used to analyze the impact of self-timed asynchronous logic stages in an embedded microprocessor. As results we confirmed the capability of self-timed logic to increase the manufacturability and reliability. Furthermore we used the tool to investigate the suitability of low-swing techniques in the NoC system communication under process variations. In this case We discovered the superior robustness to systematic process variation of low-swing links, which shows a good response to compensation technique as ASV and ABB. Hence low-swing is a good alternative to the standard CMOS communication for power, speed, reliability and manufacturability. In summary my work proves the advantage of integrating a statistical process variation analysis tool in the first stages of the design flow.

The thesis is structured as follows: ***Chapter I*** provides an exhaustive introduction to the future reliability and manufacturability challenges and solutions. ***Chapter II*** explores the temperature-aware design in low-power MPSoCs whereas ***Chapter III*** describes a HW-SW emulation framework for temperature aware design in MPSoCs. ***Chapter IV*** explains how to exploit the performance benefits of self-timed logic in synchronous design affected by process variation. ***Chapter V*** advocates the effectiveness of adaptive supply voltage and body bias as post-silicon variability compensation techniques for full-swing and low-swing On-Chip communication channels.

Chapter 2

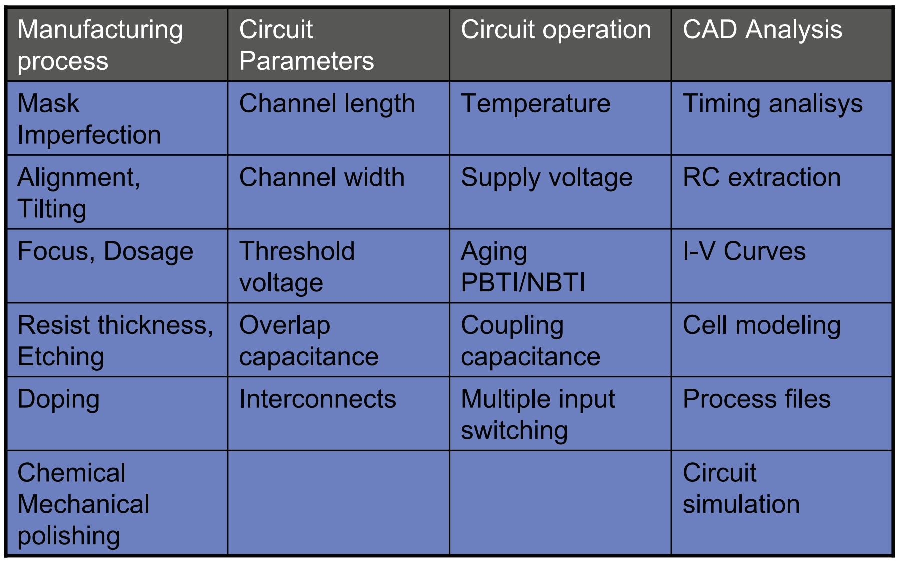
Reliability and Manufacturability Challenges



High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Technology Node (nm)	90	65	45	32	22	16	11	8
Integration Capacity (BT)	2	4	8	16	32	64	128	256
Delay = CV/I scaling	0.7	~0.7	>0.7	Delay scaling will slow down				
Energy/Logic Op scaling	>0.35	>0.5	>0.5	Energy scaling will slow down				
Bulk Planar CMOS		High Probability				Low Probability		
Alternate, 3G etc		Low Probability				High Probability		
Variability		Medium			High	Very High		
ILD (K)	~3	<3	Reduce slowly towards 2-2.5					
RC Delay	1	1	1	1	1	1	1	1
Metal Layers	6-7	7-8	8-9	0.5 to 1 layer per generation				

Figure 2.1: ITRS roadmap.

The digital electronic market is characterized by the continuous shrinking of the devices. Focusing on the growing handheld segment, the small size and high battery operation time are the must characteristics, whereas the applications are requiring more and more computational performance as are happening for the mobilephone, smartphone, camcorder, gps system. To solve the above requirements the research and industry are developing high integrated



Manufacturing process	Circuit Parameters	Circuit operation	CAD Analysis
Mask Imperfection	Channel length	Temperature	Timing analisys
Alignment, Tilting	Channel width	Supply voltage	RC extraction
Focus, Dosage	Threshold voltage	Aging PBTI/NBTI	I-V Curves
Resist thickness, Etching	Overlap capacitance	Coupling capacitance	Cell modeling
Doping	Interconnects	Multiple input switching	Process files
Chemical Mechanical polishing			Circuit simulation

Figure 2.2: Source of variability.

system with microcontroller, general purpose processors, digital signal processor (DPS), dedicated hardware and interfaces in only one chip. This solution will provide the performance required by the applications, with small power consumption and small dimensions. The above system are classified as System On Chip (SoC) or Network On Chip (NoC) depending to their complexity and communication strategy among the various components. The SoC and NoC are therefore composed by an higher transistors number and require a big chip area, rising to them problems already present in the microprocessor segment. In fact the big area increase the probability of manufacturing errors, decreasing the chip yield, and provides the space necessary to the rise of high temperature point called hot-spot, able to reduce the reliability of the systems. Those effect will be exacerbated in the future transistors technology. The ITRS roadmap (Fig. 2.1) [7] foresees the continuous doubling of the integration capacity every 2 years with a smaller energy per operation reduction. Those factors will produce an higher power density, which will increase the local temperature and the probability of hot spot. Furthermore, the future transistors will have an higher parameter variability, that will decrease the production yield and device reliability. The source of variability (Fig. 2.2) [8] are inherent to the manufacturing process, operation condition and CAD analysis:

Manufacturing process uses a lithographic system that realizes an integrate circuit in a silicon die by photo impression, chemical etching and dopant diffusion. The lithographic precision is related to the wavelength and to the minimum size of the transistors. Smaller the minimum size is , smaller the wavelength should be to keep constant the process resolution. In the future technology will not be possible to reduce the lithographic wavelength, hence the resolution will decrease, augmenting the transistor dimension variation , as the channel length.

Operation condition is critical for the transistor performance and life. In fact the commutation time is related to the temperature, supply voltage and aging. Moreover the leakage current increases exponentially with the increase of the temperature, and several wearing out processes are correlated to the thermal cycling. In the future technology there will be an higher power density, which may increase the local temperature (hot spot) and leakage currents, conducing to a possibly thermal run away. The power density may even exacerbate the thermal cycles causing a consequently faster wear out.

CAD analisys has to simulate the future behavior of the device. The simulation uses models which have their approximate precision, that will affect the results, with a consequent imprecision to the analyzed device. Furthermore the standard analysis takes in account several effect, as operation condition and process variations, as worst case, that strongly constraint the power and speed performance of the device. Such limitation will not be sustainable for the future technology where the process variation will be really high. In fact the higher variation will pull the worst case so far away that the consequently constraint may compromise the advantage related at the use of the new technology.

In the following sections will be further analyzed the impact of the cited variations to the manufacturability and reliability, with the possibly design resolution.

2.1 Design For Manufacturability

The technology scaling is approaching the nanometer size, rising new design challenges, especially for the increase of the manufacturing variation. The lithography process is not able to reduce the wavelength (see fig. 2.3) for the future transistors technology. Therefore the diffraction of the beam will become worst and worst, reducing the process precision more in deep the technology goes. Such effect will afflict the static variability of the transistor parameter, which is composed by random and systematic variations. The random variation is due to the uncertainty of the mask and the process in general, whereas the systematic variation is due to the different position of the transistor onto the silicon wafer. At the change of the beam hitting angle changes the way that

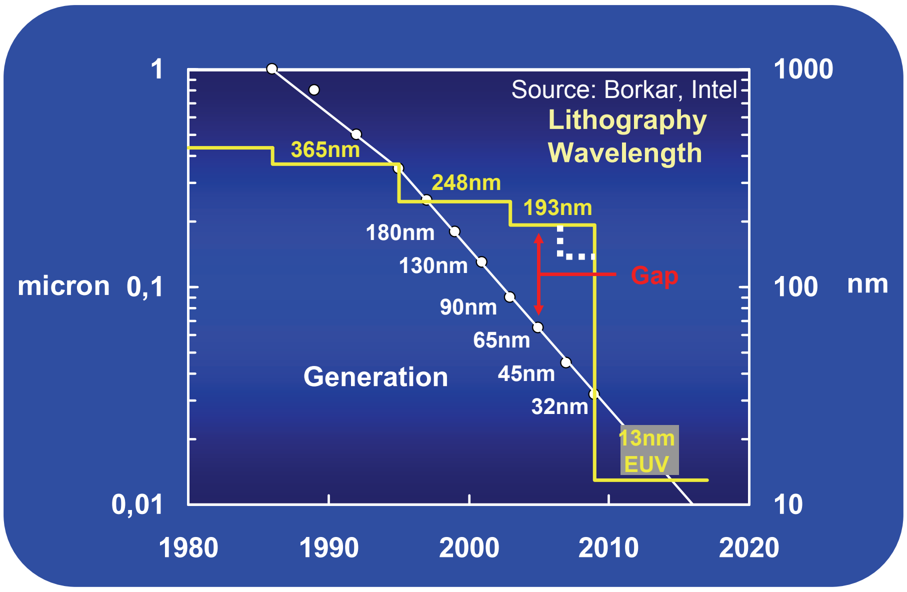


Figure 2.3: Sub-wavelength Lithography.

the mask is impressed, causing transistor size variation correlated to the position covered by the transistor itself. Therefore the systematic variation is predictable and it is composed in two components; *Within Die* (WID), that afflict the transistors inside a chip; and *Die To Die* (DTD), which sets different average systematic variation to different dies inside the wafer. The increase of the diffraction will reduce the process precision, that will enlarge the random and systematic variations. At the design level the random and WID variation has always been addressed with the worst case approach. But their enlargement is pushing the worst case so far that the future technology may not be able to improve the power and speed performance. Therefore is becoming mandatory to address those static variation at the design level, rising a new better than worst case design. Obviously the new design has to develop the capability to keep an higher yield, providing a good power and speed performance. This kind of design is called Design For Manufacturability (DFM) and aims to provide the devices with circuitry and system able to cope with variations, acting to some circuit parameters or providing a resilient characteristic to the critical stages. The robustness to process variation can be achieved with compensation technique, error detection technique and completion detection technique as illustrated below.

2.1.1 Compensation Technique

The static variations change the transistor channel length and threshold voltage, varying the performance of the devices. A way to compensate the variation therefore is to pull back the performance of the transistors changing their supply voltage or body bias voltage.

Adaptive Supply Voltage

The *Adaptive Supply Voltage* (ASV) is a compensation technique that changes the circuit supply voltage to calibrate and recover the performance lost due to the variation action. In fact the commutation speed of the transistor is linearly related to the supply voltage, providing a good knob to calibrate the performance. However the increase of the supply voltage will increase the power consumption, where the voltage is quadratically related to the power, limiting the application of ASV to the only critical area. To address this problem, the researcher are realizing the *voltage isles*, which provide different supply voltage to different circuit area. The voltage isles set an hierarchy to the supply distribution, in which there are *local supply* and *global supply* lines. The local supplies the power to the transistors inside the isle, while the global provide the power to the isles and then to the local lines. Usually the connection with the global to the local is done by a multiplexer or charge pump, depending if the global lines have one or multiple voltages. The charge pump is an high efficient circuit able to vary the voltage, whereas the multiplexer provides the capability of the local line to pick one of the few voltages carried by the global lines. The use of ASV is not without of troubles. In fact the communication between different voltage isles has to use circuit able to switch the swing of the signals from a voltage domain to another (*level shifter*). Especially if a lower voltage isle has to send data to an higher voltage domain, where the swing of the signal has to be managed by a level shifter to avoid static power consumption. Furthermore the clock distribution has to go to different voltage domains passing through several level shifters, exacerbate the global clock synchronization.

Adaptive Body Bias

The *Adaptive Body Bias* (ABB) is a compensation technique that changes the transistor *threshold voltage* to calibrate and recover the performance lost due to the variation action. In fact the change of the body bias changes the threshold voltage that changes the transistor commutation speed, providing a knob to calibrate the performance. However the variation of body bias to increase the performance will increase the leakage currents, that are exponentially correlated to the threshold voltage. Therefore is suggested to apply the forward

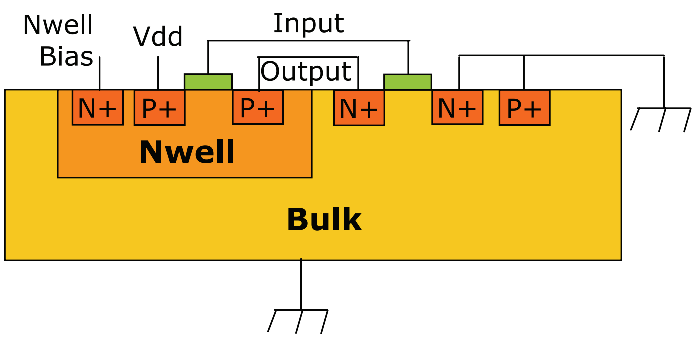


Figure 2.4: Single Well P-MOS Body Bias.

body bias only to the critical area. Since the P-MOSs are placed in a N-WELL (Fig. 2.4), it is possible to easily separate their body bias designs several N-WELLs. Whereas to guaranty different N-MOS biasing we have to use a *triple well* technology (Fig. 2.5), which its realization is more expensive then a *single well* technology. The use of the ABB to the only P-MOS will reduce the capa-

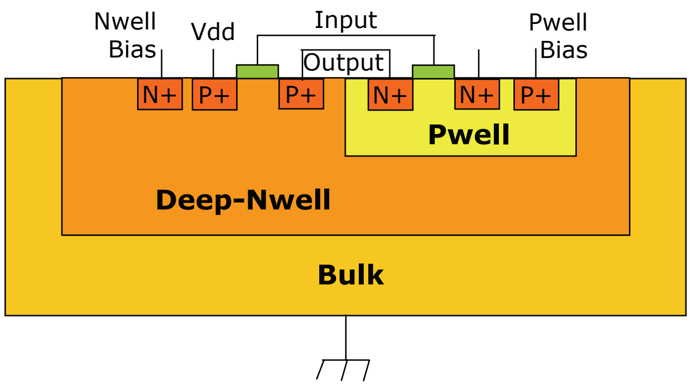


Figure 2.5: Triple Well CMOS Body Bias.

bility to tune the device performance, while the utilization of the both P and N-MOS in the ABB process will increase the manufacturing cost. Therefore will be mandatory to analyze during the design which ABB has to be applied to compensate the static variation.

ASV and ABB Issues

The ABB has all the isles with the same supply voltage, avoiding the use of level shifter and the consequent problem to the clock synchronization as in the ASV. Furthermore the ABB require lower extra power to compensate the same variation than ASV, but it has weaker capability to tune the device performance. Therefore the application of the ASV and ABB has to be addressed during the design phase with a support of a static variation analysis tool. In that way is possible to understand which compensation technique will be more suitable in terms of power and yield. Moreover the setting of the above compensations has to be done by another system, that could act at the test or operation time. The setting system has to be decided during the design and it could be an error detection system or a performance estimator allocated inside the device or in the test machine.

2.1.2 Error Detection Technique

The *Error Detection Technique* communicates the onset of some critical timing events or computation errors to the system, that activates a procedure to restore the correct data or operation environment. The detection is done using *error detection* circuitry as *razor* and *crystal ball* flip flop.

Razor

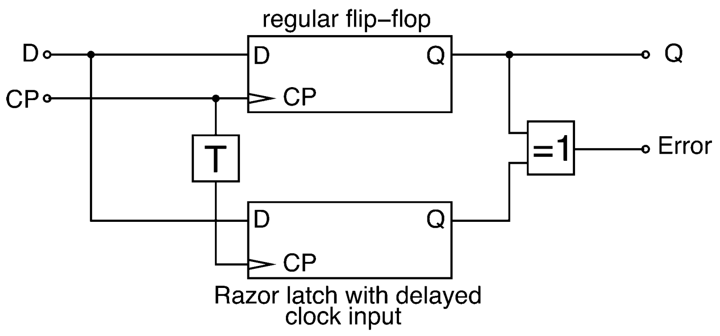


Figure 2.6: Razor Flip-Flop.

The razor detects when the input signal of a pipe stage register violets the flip-flop setup time. In this case the register latches a wrong data, causing a pipeline failure. The detection is based to the comparison of the outputs of two flip flops, which one of them has delayed clock. In fact the violation of the setup time will cause the wrong register latching while the clock delayed flip flop will register the correct data, and the comparison of the both flip flop outputs will notice the error(Fig. 2.6). Furthermore the data stored in the beside flip flop could be used to restore the correct value as follow [9]. When the system receives the error signal, the pipeline will be stalled, providing the time necessary to insert the correct value to the faulty pipe register. Moreover the system can act to the ABB or ASV to increase the performance, reducing the error probability. In [9] the system set the supply voltage to have the optimal error probability for a minimum power consumption and maximum throughput. Fig. 2.7 denotes as the razor has a lower power consumption then the worst case approach, power that could be spent to increase the manufacturing yield. However the use of the razor requires special attention to the logic

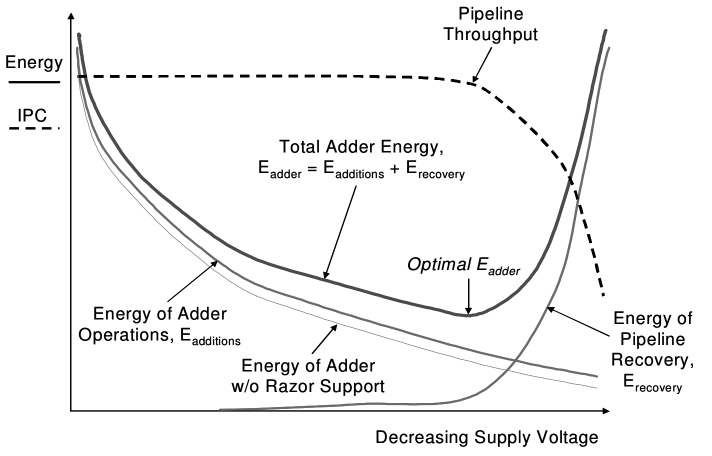


Figure 2.7: Razor:The Qualitative Relationship Between Supply Voltage, Energy and Pipeline Throughput (for a fixed frequency).

operation time. In fact the logic before the razor register has to have the minimum computation time bigger then the delay applied to the clock, to avoid to sign as an error a correct operation. Moreover the logic can not operate with a clock period smaller then the worst case minus the clock delay, in order to keep detectable all possible errors. Hence the razor require an extensive timing analysis in the design phase, where will be evaluated its applicability to the specific device.

Crystal Ball Flip Flop

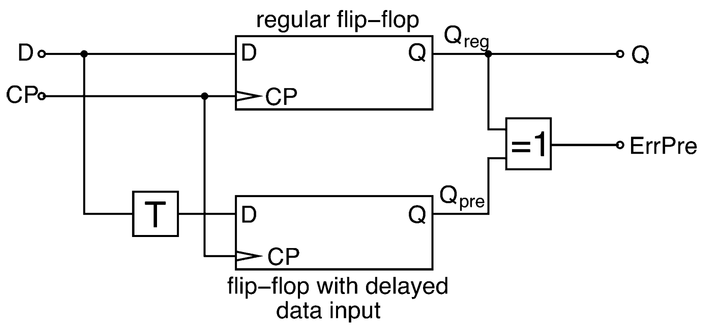


Figure 2.8: Crystal Ball Flip-Flop.

The crystal ball recognizes the insurgency of a critical condition that will violate pipeline register setup time [10]. The Fig. 2.8 shows two similar flip flop which one of them has a delayed input. In this way the delayed input flip flop will miss to latch the data every time the input signal will be near to the register setup time violation. Then the comparison of the two flip flop outputs will set the error signal, that will inform the system of its possible failure. Usually the device react to the crystal ball alert tunning some compensation system, as ASV or ABB. The crystal ball does not provide the support for a possibly error recovering as in the razor, but only an alert when the circuit is too near to the critical timing condition. The crystal ball does not need the evaluation of the minimum operation time of the logic as razor, but requires the same analysis for the clock period, that here can not be inferior to the worst case.

2.1.3 Completion Detection Technique

The *Completion Detection Technique* senses the complete operation execution and can be used by the system either to synchronize the various pipe stages (*Handshake*), or to stall the pipe to provide the right time to complete the operation (*Clock Gating*).

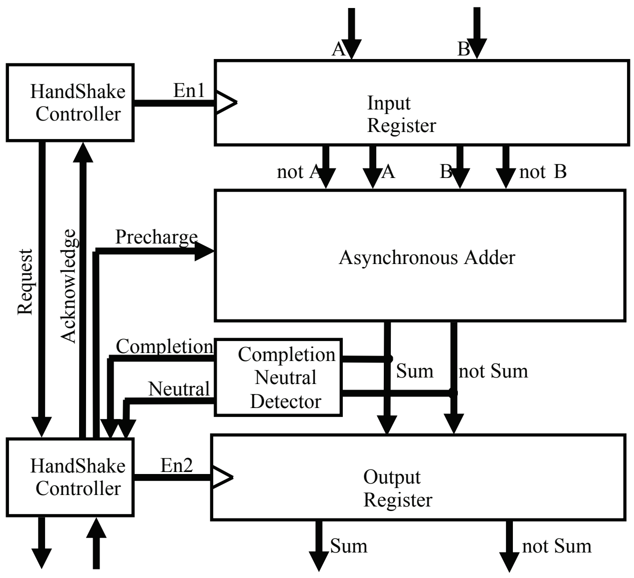


Figure 2.9: Handshake Pipe Stage.

The handshake regulates the pipeline flow, posing a controller every pipe register (Fig. 2.9). Each controller checks the arising of both logic completion and follow stage data request. When all of them occur, the controller activates the pipe register to latch the data, sends a new data request to the previous stage and sends the acknowledge to the follow controller. Hence the pipeline works asynchronously. This imply the use of the completion detection logic to the whole circuit, with an higher power cost. However the absence of the clock tree remove all the clock design concerns, as the clock skew and the high clock electromagnetic interferences. Moreover the asynchronous pipeline performance is related to the average computation speed, that is faster and less affected by process variation then the worst case. Therefore the handshake pipeline has a better resilience to the variation, and its faster performance can be spent to decrease the completion power cost.

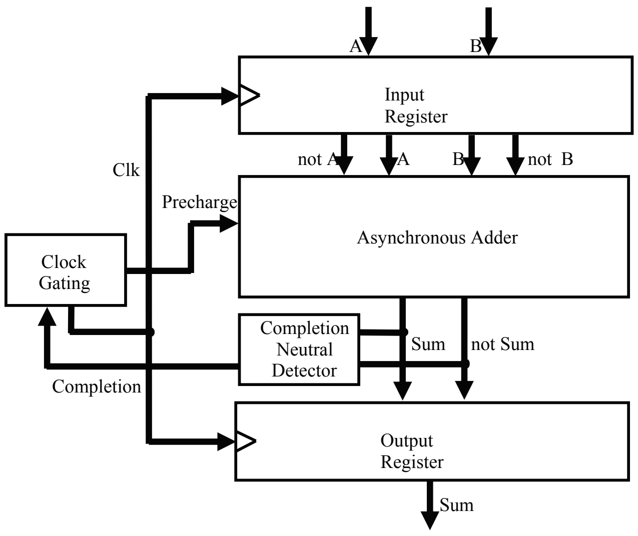


Figure 2.10: Clock Gated Pipe Stage.

The clock gating controller (Fig. 2.10) checks the operation completion and stall the clock when the operation needs more than one period to be executed. The use of a completion detection logic gives to the device a resiliency to the variation. In fact the clock gating can work with process variation, that will change only the average pipeline throughput. Furthermore the clock gating performance is related to the average throughput, that is less affected by process variation then the worst case design, increasing the manufacturing yield. Moreover the clock gating permit to integrate a completion detection pipe stage in a traditional pipeline. In this way is possible to substitute the critical stage with a completion detection one, improving the manufacturability and reliability, and avoiding the detection logic power expense to the whole pipeline.

The detection of the operation completion can be done by *current sensing* or *dual rail logic* as illustrated below.

Current Sensing

The CMOS devices have a static current near to zero. The variation of one input drastically increases the current flows through the device, which will return to the static value only when the entire logic has complied the operation. There-

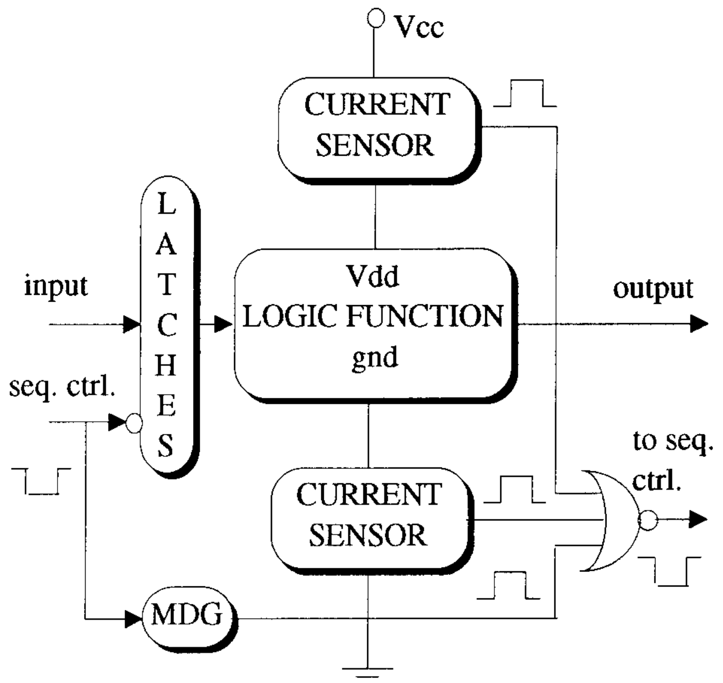


Figure 2.11: Current Sensing Completion Detection.

fore the *current sensing completion detection* profiles the current flows through the logic to detect the operation completion [11]. In fig. 2.11 the completion has been done by two current sensors. In fact the CMOS commutation involves either capacitance charging or discharging, causing respectively current to the P-MOS or N-MOS, which are detected from the both current sensors. Therefore the current sensors provide to the standard CMOS the completion detection signal, which can be used to realize asynchronous or clock gated circuit. However the impact of the current sensor to the CMOS design is not null. The voltage drop due to the current sensor will reduce the commutation performance of the device. Furthermore the current profiling requires biased transistors, which dissipate static power. Moreover some device commutations generate a smaller current variation that can not be easily detected and a few publication have resolved this problem inserting a timing out counter [11]. Then the clock gating solution is more suitable for the current sensing technique than asynchronous realization.

Dual Rail Logic

The *Dual Rail Logic* embeds the information of the operation compilation inside the logical signal, that is encoded in two wires as shown in the table 2.1. One wire carries the true logic value while the other one encode the false value. The logical information is provided when one of the two wires has a voltage

True Wire	**False Wire**	**State**	**Logic Value**
gnd	*gnd*	Neutral	-
gnd	*vdd*	Valid	False
vdd	*gnd*	Valid	True
vdd	*vdd*	Not Allowed	-

Table 2.1: Dual Rail Encoding.

equal to vdd, while the *valid* state informs to the advent of the operation completion. Usually a dual rail circuit start e new operation when it has all the logic components to the *neutral* state. The execution flows through the circuit changing the state of every cell from neutral to valid, and the completion of the global operation is detected when all the outputs are in a valid state. Afterward the circuit will be brought back to the neutral state, to become ready for another operation. The implementation of the dual rail logic require the presence of both true and false evaluation stages. They can be done using a traditional CMOS circuit with a special attention to avoid the glitches occurrence [12], that can cause a false completion detection. However a logic operator realized in the above way, will take more then a double space of the standard CMOS. Another dual rail implementation uses the *Dynamic Cascode*

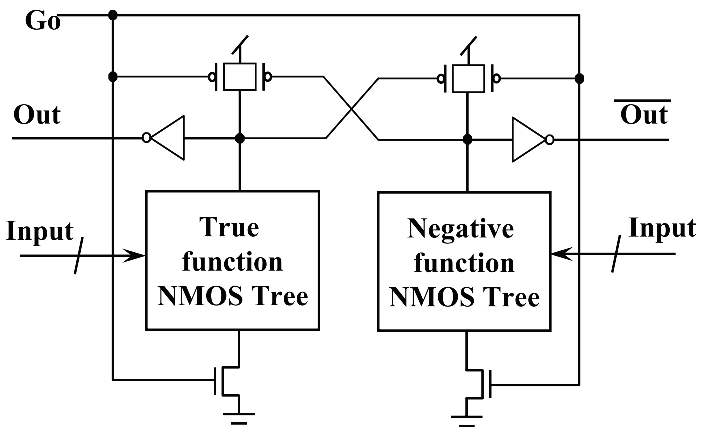


Figure 2.12: Dynamic Cascode Voltage Switch Logic.

Voltage Switch Logic (DCVSL) (Fig. 2.12. In this case the true and false stages are realized with the only n-mos branches, that are connected each other with a p-mos transistor. Such topology permits a smaller realization and the introduction of the precharge and footer transistors gives the way to bring the cell in the neutral state. The dual rail logic is therefore high suitable for handshake and clock gating pipeline, providing an higher robustness to process variation due to the self timed characteristic. However this kind of logic requires larger area for the duplication of the signal wires, the introduction of the completion detector and the presence of the precharge wire. Furthermore the constant switching from neutral to valid states and vice versa causes an higher power consumption, which can be balanced by the faster computation performance, that is related to the average computational speed instead to the worst one as happens in the CMOS logic. Moreover the dual rail is not supported by the commercial design tools and test approaches, obstructing its utilization in the future devices.

2.2 Design For Reliability

The continuous shrinking of the technology dimension is increasing the power density and transistor weakness, rising reliability challenges. The density and

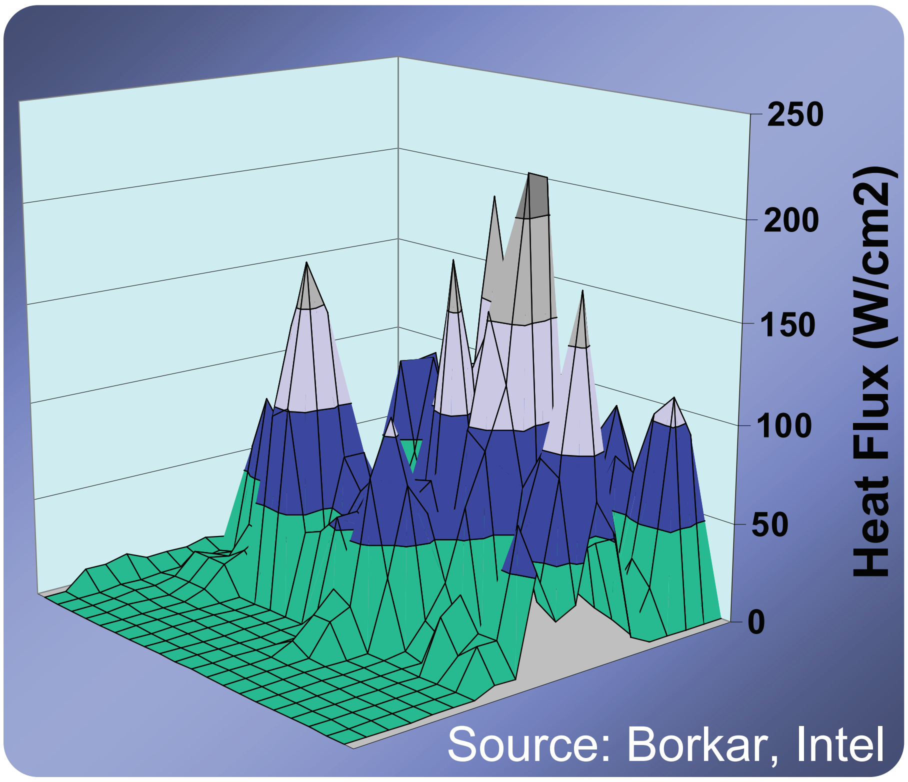


Figure 2.13: Heat flux in a Intel Microprocessor.

not well distribution of the power(Fig. 2.13) generate high temperature point called hot spot (Fig. 2.14). Since the transistor commutation speed is inversely proportional to the temperature, the insurgency of hot spot may unbalance the propagation time of the signals increasing the failure rate. Furthermore many wear out processes are proportionals to the temperature as *Electromigration*,

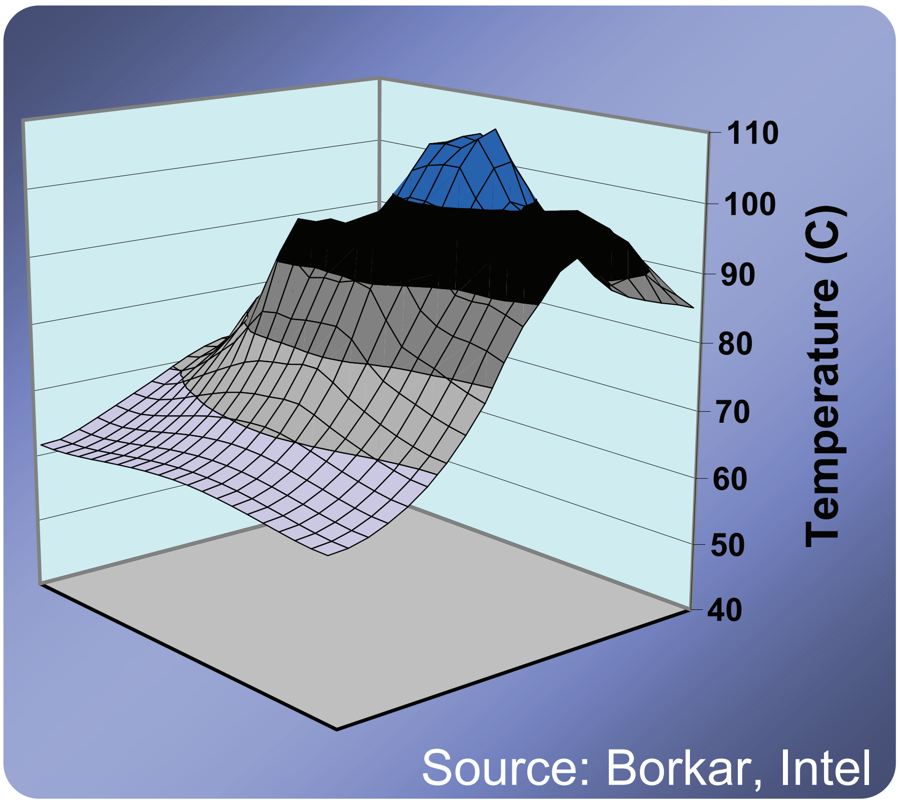


Figure 2.14: Temperature variation in a Intel Microprocessor.

Stress migration, *Time dependent dielectric breakdown* and *Thermal cycling*, causing a relation between mean time to failure (MTTF) and temperature as explained below [13]:

Electromigration The momentum transfer in a wire transports metal atoms from one end to the other, eventually leading to increased resistance and shorts. The temperature relates the electromigration's MTTF as in the equation 2.1.

$$MTTF_{EM} \propto (J)^{-n} e^{\frac{Ea_{EM}}{kT}} \tag{2.1}$$

Stress migration. In this phenomenon, mechanical stress causes metal atoms in the interconnect to migrate, much as they do in electromigration. Materials differ in their thermal expansion rate, and this difference causes thermomechanical stress that is related to the temperature as in the equation 2.2

$$MTTF_{SM} \propto |T_0 - T|^{-n} e^{\frac{Ea_{SM}}{kT}} \tag{2.2}$$

Time dependent dielectric breakdown. Also known as gate-oxide breakdown, time dependent dielectric breakdown (TDDB) is the result of the gate dielectrics gradual wear out, which leads to transistor failure and is related to the temperature as in the equation 2.3.

$$MTTF_{TDDB} \propto \left(\frac{1}{V}\right)^{a-bT} e^{\frac{[X+(Y/T)+ZT]}{kT}} \tag{2.3}$$

Thermal cycling. The accumulation of permanent damage experienced by the circuit for temperature cycles may lead to failure. Moreover the presence of hot spot increase the temperature difference (T_d), exacerbating the action of this wear out process as shown in the equation 2.4.

$$MTTF_{TC} \propto \left(\frac{1}{T_d}\right)^{q} \tag{2.4}$$

Therefore the future systems will have to manage the temperature to keep acceptable reliability and failure rate levels, using *Thermal Management* and *Temperature Aware Design* of the system.

2.2.1 Thermal Management

The thermal management acts to some tunable parameter as voltage and frequency, processors states and workload distribution to keep the temperature under a fixed threshold, or to limit the extension of the thermal cycle. The temperature is sensed in few strategic points inside the die surface. The usually action of the thermal management is to cool down the area that exceeds the temperature limit acting either to the circuit activity (a), supply voltage (Vdd) or operative frequency (f) to reduce the power consumption (P) as shown in the follows equation.

$$P \propto a \cdot f \cdot Vdd^2 \tag{2.5}$$

The voltage and frequency scaling is an efficient way to reduce the power consumption. In fact the power is quadratically related to the supply voltage and linearly related to the frequency as shown in the equation 2.5. However the reduction of the supply voltage implies a slower transistor switching speed, with a consequent slower operative frequency. Furthermore the change of the frequency needs a settlement time to stabilize the clock generator, affecting even more the computational performance. Therefore the voltage and frequency scaling has to be used carefully, avoiding an excessive computational performance penalty. The reduction of the power consumption can be also done freezing the processor activity, setting the idle state. But due to the excessive performance penalty, this procedure is suitable in the rarely presence of really high temperature that may damage the circuit. However in the multiprocessor system, the thermal management can distribute the workload to the various processors in the way to keep constant or limited the temperature. In order to do such kind of thermal management, the system has to know the temperature of each processor and adopt a distribution job logic, that causes the minor thermal impact. Furthermore, the realization of the thermal management at the operative system level, could include the estimation of the application performance requirement, providing the possibility to set the desired operative frequency and supply voltage to the processor unit most suitable for the right thermal behavior. Hence *the thermal aware design* is required to develop system able to support complex thermal management as illustrated above.

2.2.2 Thermal Aware Design

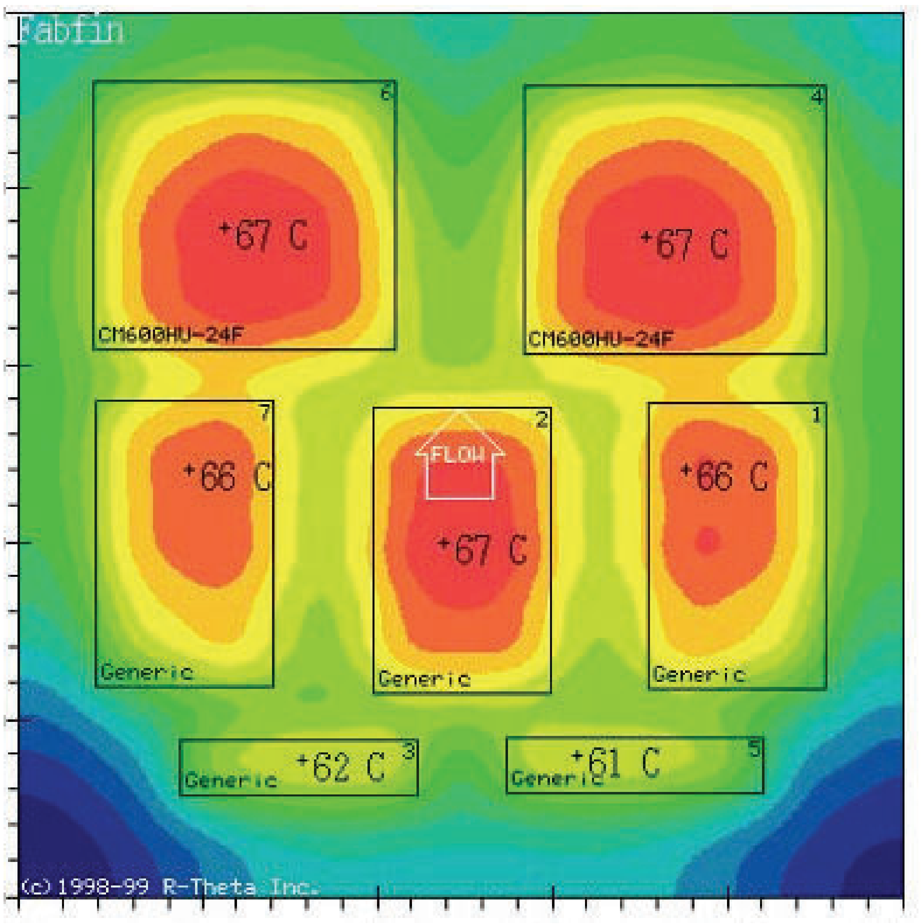


Figure 2.15: Temperature Distribution Inside a Multiprocessor System.

The increasing power density in the future transistors technology is rising thermal problem that are no longer addressable to the only packaging point of view. In fact the insurgency of hot spot will reduce the reliability of the system and increase the failure rate. Therefore the temperature has to be analyzed in the design phases, to evaluate different layout solutions, system organization and typology. The position of the different components strongly influence the thermal behavior of the chip. Then a better placement with an awareness of the future thermal behavior could avoid the insurgency of the hot spot increasing the reliability of the devices. Furthermore the analysis in the design stages permits to find the best position for the temperature sensors. Moreover the system simulation with temperature aware provides the support for thermal management implementation, and in conjunction with a wear out estimator, the support for reliability management development.

2.3 Thesis contribution

My Phd research activity has focused on design and analysis techniques for tackling the variations in the embedded low-power digital circuits. I developed a thermal and static variation analysis tools, to investigate the need and impact of manufacturability and reliability design techniques. The evolution of the embedded devices in *System on Chip* (SoC) and *Network on Chip* (NoC) is enlarging the area of the die, that in conjunction with the higher power density and parameter variability of the new transistor technologies, may increase the hot spot occurrence and will decrease the manufacturing yield. Such condition are pushing the designer to address both temperature and process variations at the first design stages, as system and logical, to keep good reliability and manufacturability levels. Therefore I developed a thermal analysis library able to predict the thermal behavior of a silicon die with traditional packages, suitable to work with system simulators. Then the library has been integrated in a NoC cycle accurate simulator [5] and in a FPGA based NoC simulator [6]. Thereafter both simulators were utilized to analyze hot spot insurgence in several NoC designs, evaluating the impact of the components placement, processor typologies and thermal managements. All the analyses have confirmed the need of a thermal analysis at the firsts design stages to provide a good reliability level in the future embedded system.

Later on I developed a static variation analysis tool, that is integrable into commercial design CAD flows, able to address both systematic and random variations. The tool was used to evaluate the benefit of a self-timed logic insertion in the critical stage of an embedded processor pipeline, where the commercial design flow was adapted to support the synthesis of DCVSL circuits. The results proved the benefit of the self-timed insertion, providing new paradigm to exploit the self timed benefit in a synchronous pipeline. Moreover the static variation analysis tool was used to extensively analyze the suitability of low swing technique for NoC communication networks. Low swing communication showed a good resilience to systematic and random variation, with low power consumption and good performance. As last, the static variation analysis tool was adapted to analyze the efficiency of the adaptive supply voltage and adaptive body bias in NoC communication. The results showed as ASV is the best solution to recover static variation in low swing links with a small power cost, whereas the ABB is the best choice for the traditional full swing communication channel in terms of power and efficiency. Moreover the use of the static analysis tool has proved its utility in the first stages of a low-power NoC design.

Chapter 3

Exploring temperature-aware design in low-power MPSoCs

3.1 Overview

The power density in high performance systems continues to rise with every process technology generation, thereby increasing the operating temperature and creating "hot spots" on the die. As a result, the performance, reliability and power consumption of the system degrade. To avoid these "hot spots", "temperature-aware" design has become a must. For low-power embedded systems though, it is not clear whether similar thermal problems occur. These systems have very different characteristics from the high performance ones: E.g., they consume hundred times less power. Therefore, I provide in this chapter guidelines to delimit the conditions for which temperature-aware design is needed.

3.2 Introduction

In recent years, the power densities in high performance microprocessors have doubled every three years [16]. Moreover, most power is consumed in a few localized spots which heat up much faster than the rest of the chip (e.g., the CELL processor [29]). These hot spots potentially increase leakage currents, cause timing errors and/or even result in physical damage. Heating has also become a big issue because expensive cooling solutions are not acceptable for consumer products. Several authors have therefore advocated the need for "temperature-aware" design techniques (see [33] for an overview). Some of them have already found their way in industrial designs (E.g., in the Intel Itanium processors [30]).

It is less clear whether such techniques are required for low-power multi-processors systems-on-a-chip (LP-MPSoC). These systems dissipate two orders of magnitude less power (max $3W$ instead of $100W$). Since portable systems (such as mobile phones) are the main target for LP-MPSoCs, they are built in a low standby power technology instead of a high performance one. The main difference is that subthreshold leakage is engineered to remain low, even at the expense of a higher power supply and thus more dynamic power (see table 3.1). Due to the smaller contribution of leakage power, the impact of temperature on the total power dissipation is limited [22]. However, packaging solutions for consumer electronics are much cheaper and rely on natural convection for removing the heat. As a result, the die is thermally more isolated and may still heat up.

$65nm$ technologies	$V_{DD}(V)$	$V_T(V)$	$Ioff(\mu A/\mu m)$
high performance	0.9	0.18	$7e-2$
low power	0.8	0.26	$5e-3$
low standby power	1.1	0.5	$2.5e-5$

Table 3.1: Technology bifurcation in the 2003 ITRS roadmap. Leakage is significantly lower in low standby power than high performance technologies.

The existence of hot spots on the die is even harder to predict (as shown in [18]). LP-MPSoCs for instance are built with a different computer architecture than high performance systems. E.g., multiple power-efficient processors instead of few complex super-scalar processors (compare a Intel P4 with a Philips Nexperia). Secondly, they operate at a lower power density: the processors run at a lower speed (typically around $500Mhz$ instead of $3GHz$) and contain a large amount of embedded memory. Finally, the die of a LP-MPSoC is typically smaller than that of a high performance computer. Therefore, the generated heat can more easily reach the corners of the chip, resulting in less temperature variations across the die. Several factors thus seem to indicate that hot spots are less likely on LP-MPSoCs than on high performance systems. However, this has to be verified.

The contribution of this chapter is to delimit the conditions for which hot spots become a critical problem in LP-MPSoCs. To investigate this problem, we have built an accurate thermal/power model of a multi-processor system-on-a-chip. Our thermal model is different from those of high performance systems because we investigate the thermal behaviour of multiple cores and embedded memories on a single die and we look at package solutions for LP-MPSoCs which have a much higher thermal resistance.

For modelling the heat flow, we rely on an equivalent electrical RC model (similar to HotSpots [33]) which we have calibrated against a 3D-finite element analysis. To obtain realistic traces of the activities on the die, we have integrated our thermal model in a multi-processor system-on-a-chip simulator. Experimental results for a typical LP-MPSoC show that the temperature differences on chip are limited and that the temperature changes only slowly with time.

3.3 Related work

Three problems arise due to an elevated operating temperature. First, the higher the temperature becomes, the more leakage currents occur and thus the more power is consumed. This has been investigated for instance by [34] [36] [24]. Secondly, a higher temperature also reduces the mean-time-to-failure of the system. For instance, the physical processes that trigger electromigration and stress migration become more active if temperature rises (see [28] [27]). Thirdly, a higher temperature impacts the performance of the transistors. On one hand it reduces the mobility of the charge carriers (electron and holes), but on the other hand it decreases the V_T [22]. Usually, higher temperatures decreases the performance of circuits. Timing violations then become more likely.

To investigate the above issues in detail, the authors of [33] have developed a thermal/power model for super-scalar architectures. It not only predicts the temperature variations between the different components of a processor, but also accounts for the increased leakage power and reduced performance. Their results clearly prove the importance of hot spots in high performance systems.

Based on this and/or similar models, many architectural extensions have been proposed to reduce the impact of hot spots and/or to prevent the die from breaching a critical temperature. The power density in super-scalar processors can be reduced with fetch toggling, decode throttling, frequency and/or voltage scaling (e.g., [15] [20] [32] [17] [30]). Except for frequency/voltage scaling, the above techniques are only applicable on super-scalar processors. Another approach for reducing the impact of hot spots is adding redundancy to the architecture. [33] advocates that a spare register-file and migrating computation between register files is the best in response to heating. Similarly, [19] examines the benefits of moving computation between multiple replicated units of a super-scalar processor.

Besides architectural solutions, the temperature can also be reduced at the system level. E.g., [31] stops scheduling hot tasks when the temperature reaches a critical level. In this way the system is idling and the CPU spends more time in low-power states, such that the temperature either locally or globally is decreased.

The related work discussed above is targeting high performance systems, where the power density hampers scaling. However, the context is different in low-power systems, which run at a lower speed and are subjected to less leakage power. [34] have investigated the impact of temperature and voltage variations across the die of an embedded core. Their results show that the temperature varies around 13.6 degrees across the die. Since they use a $130nm$ CMOS Silicon-on-Insulator technology, it is hard to extrapolate their results to a bulk CMOS technology. [21] explains an allocation and scheduling algorithm for eliminating hot-spots in a system-on-chip, but they target a much higher power budget (up to $15W$) than is present in systems-on-a-chip for portable applications (up to $3W$ see description of the system drivers in the ITRS roadmap).

Based on our literature study, we believe that the case for temperature-aware design has not yet been made for low-power multi-processor systems. Particularly, the existence of hot spots in these systems has not been validated yet.

3.4 A typical LP-MPSoC

In Figure 3.1, we show the floorplan of a typical LP-MPSoC. It consists of 16 ARM7 cores and 16 32kB shared memories. The shared memory is used for storing large data structures and communicating data between the processors. Each ARM7 core is attached to a local 8kB 2-way associative data cache and a 8kB direct mapped instruction cache. The memories and processors are connected using a XPipes Network-on-Chip [14] of which a 6X6 switch and network interface modules are shown on the floorplan. We have obtained the dimensions of the NoC circuits by synthesizing and building a layout. The dimensions of the memories and processors are based on numbers provided by an industrial partner. In the remainder of the chapter, we will use this floorplan to research the temperature effects inside an LP-MPSoC.

3.5 Power/thermal analysis

To estimate the power consumption and temperature of all architectural blocks inside an LP-MPSoC, we have built a simulation environment as depicted in Figure 3.2.

We use a cycle accurate simulation platform for measuring the activities in each of the memories and processing elements in the system[1]. We measure

[1]based on MPARM [26].

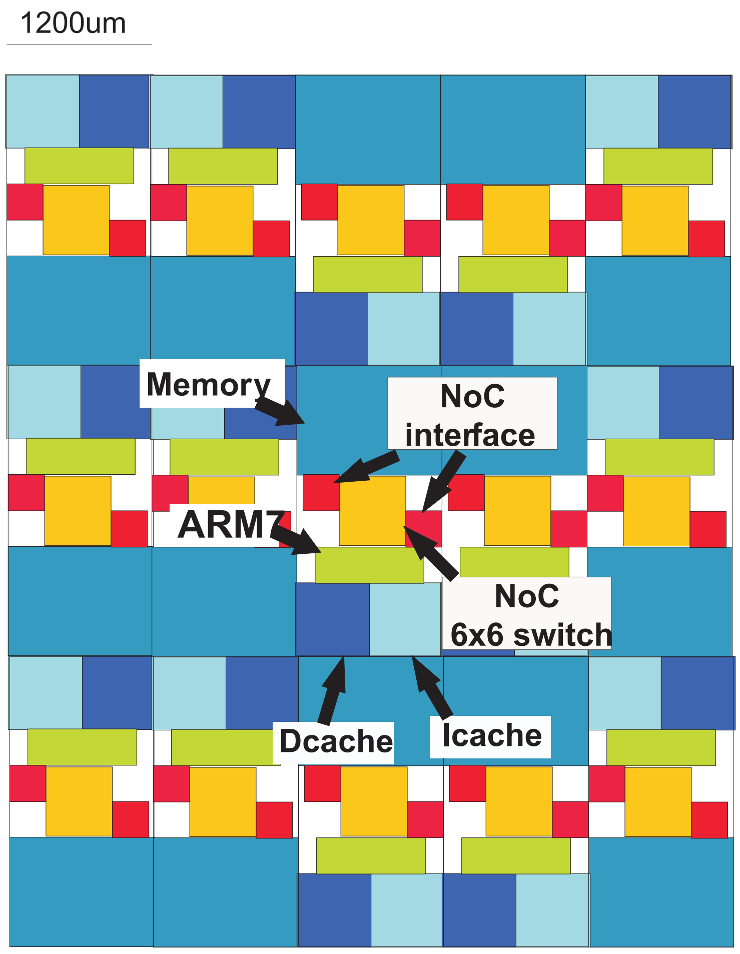


Figure 3.1: Floorplan of a multi-processor system on a chip in a $130nm$ technology: 16 ARM7 processors each connected to a 2-way associative 8kB data cache and a direct mapped 8kB instruction cache; 16 memory tiles of 32kB; a NoC connecting processors and memories. Thermal cells are $150\mu m \times 150\mu m$.

the time that each processor spends in active/stalled/idle mode. We trace the number and type of accesses to each of the memories (instruction cache/data cache and large shared memories). Based on these activities, we estimate every $10\mu s$ the energy consumption in each of the thermal cells of the layout. Thereafter, we feed this data into our thermal simulator. The latter one computes the temperature evolution of the die during the last $10\mu s$. The temperature map of the die is then logged in a file. Since the joint performance/power/thermal simulation is rather time-consuming, we also provide the option to dump the components' activities in a trace-file. Using this trace-file, we can for instance more quickly explore different packaging solutions.[2]

[2]In this case though, we cannot feedback the temperature effects to the performance simulator

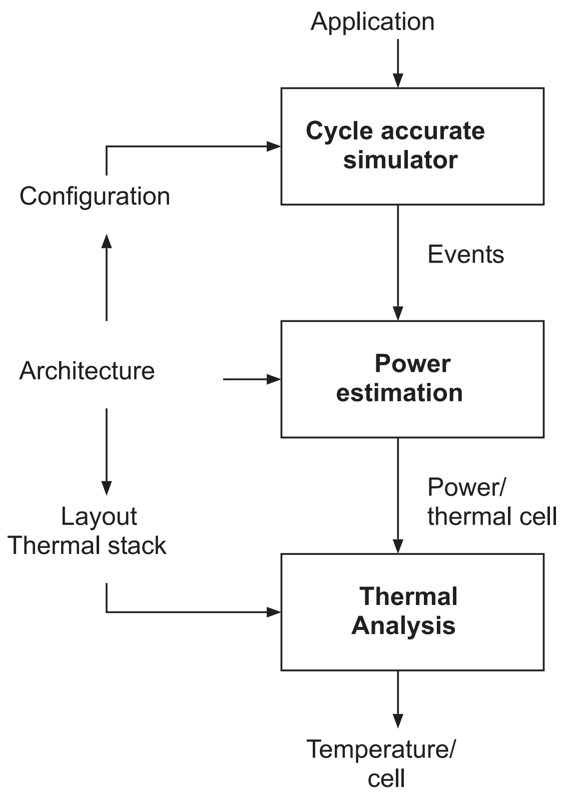


Figure 3.2: Power/thermal model integrated in a cycle-accurate LP-MPSoC simulator [25].

3.5.1 Power estimation

		Max. Power@$100Mhz$	Max. Power density
ARM7		5.5mW	$0.03W/mm^2$
DCache	8kB/2way	43mW	$0.012W/mm^2$
ICache	8kB/DM	11mW	$0.03W/mm^2$
Memory	32kB	15mW	$0.02W/mm^2$

Table 3.2: Power for the most important components of a LP-MPSoC in a $130nm$ bulk CMOS technology.

In Table 3.2, we outline the power consumption and power densities of the most important components of our system-on-a-chip. The table contains the maximum power numbers, but the effective power is normally lower, depending on the workload (activities of processors and memories). We ignore leakage energy. Leakage in mobile systems has to be limited for guaranteeing sufficient battery-life time. Typically, leakage is therefore eliminated at the device level by developing high V_T transistors. As a result, the leakage can be as low as $25pA/\mu m$ (see ITRS roadmap). High V_T transistors come at the expense of a higher V_{DD} and thus more dynamic energy (since the $V_{DD} - V_T$ has to be kept constant for retaining the same performance). A better option is to use leakage reduction techniques such as back-biasing. E.g., [23] jointly optimize V_{DD}/V_T by using dual-gate devices or back-biasing. The authors lower the V_{DD} for reducing the dynamic power. However, to retain the same performance, they have to reduce V_T as well, which unfortunately exponentially increases the subthreshold leakage. In the optimal V_{DD}/V_T operating point (where the V_{DD}/V_T are more aggressively scaled than on the ITRS roadmap[3]), leakage energy contributes only 10% of the total power. Hence, we believe that the impact of leakage on temperature for low-standby power systems is limited.

3.5.2 Modelling the heat flow

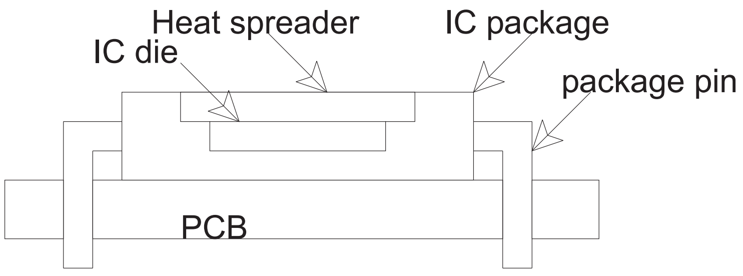


Figure 3.3: Chip packaging solution.

A LP-MPSoC is usually packaged in a cheap plastic ball grid array package ([35] and see Figure 3.3). The main purpose of this package is to electrically connect the die to the other circuits on the printed circuit board and to protect it against the environment. Besides, the package has also to remove the dissipated heat from the die. Typically, a heat spreader made of copper, aluminium or another highly conductive material is therefore attached to the reverse side of the die. Its goal is to increase the thermal conductivity of the package. In the context of this chapter, we assume that all surfaces but the one of the heat spreader are adiabatic. The spreader disposes the generated heat by natural convection with the ambient.

Equivalent RC thermal model

Similar to [33] [34] [19], we exploit the well known analogy between electrical circuits and thermal models. We decompose the silicon die and heat spreader

[3]Hence, the system operating with the optimal V_{DD}/V_T is more leaky than with ITRS conditions.

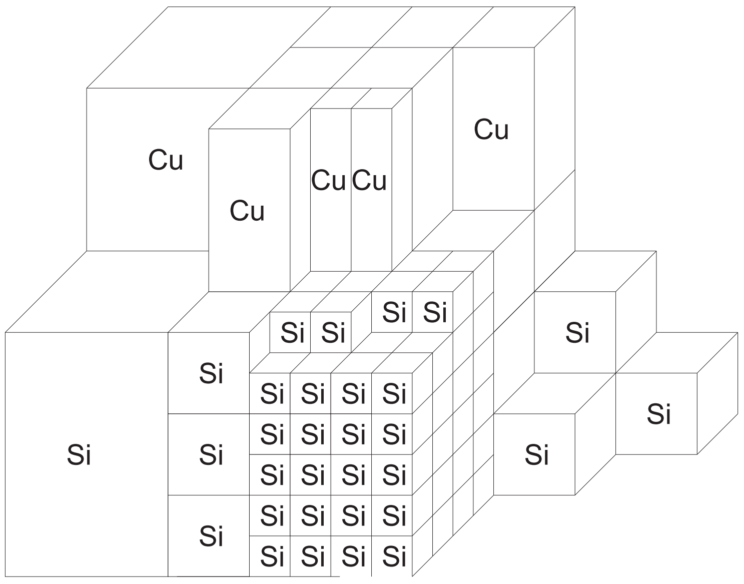


Figure 3.4: Dividing the chip into a finite number of cells.

in elementary cells which have a cubic shape (Figure 3.4) and use an equivalent RC model for computing the temperature of each cell.

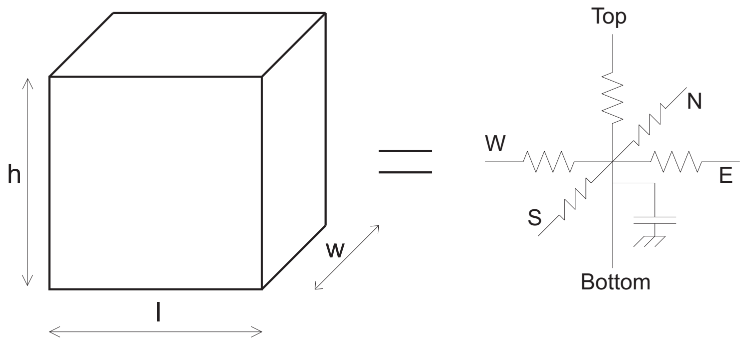


Figure 3.5: Equivalent RC circuit of a cell.

We associate with each cell a thermal capacitance and five thermal resistances (Figure 3.5). Four resistances are used for modeling the horizontal thermal spreading whereas the fifth is used for the vertical thermal behavior. The thermal conductivities and capacitance of the cell are computed as follows (where $k_{th}^{si/cu}$ is the thermal conductivity and $c_{th}^{si/cu}$ is the thermal capacitance per unit volume):

$$G_{th}^{NESW} = k_{th}^{si/cu} \cdot \frac{h \cdot w}{l} \tag{3.1}$$

$$G_{th}^{top} = k_{th}^{si/cu} \cdot \frac{l \cdot w}{h} \tag{3.2}$$

$$C_{th} = c_{th} \cdot l \cdot h \cdot w \tag{3.3}$$

We model the generated heat by adding an equivalent current source to the cells on the bottom surface. The heat injected by the current source into the cell corresponds to the power density of the architectural component covering the cell (e.g., a memory decoder or processor) multiplied with the surface area of the cell. Note that no heat is transferred down into the package from these bottom cells.

The heat from the cells on the top surface is removed through convection. We model this by connecting an extra resistance in series with their $R_{th}^{top} = 1/G_{th}^{top}$ resistance. The value of this resistance is equal to the package-to-air resistance weighted with the relative area of the cell to the area of the spreader.

silicon thermal conductivity[4]	$150 \cdot \left(\frac{300}{T}\right)^{4/3} W/mK$
silicon specific heat	$1.628e-12 J/\mu m^3 K$
silicon thickness	$350 \mu m$
copper thermal conductivity	$400 W/mK$
copper specific heat	$3.55e-12 J/\mu m^3 K$
copper thickness	$1000 \mu m$
package-to-air conductivity	$20 K/W$ in low power

Table 3.3: Thermal properties.

Thermal properties

In table 3.3, we enumerate the thermal properties of the package used during our experiments. The amount of heat that can be removed by natural convection strongly depends on the environment (such as the placement of the chip on the PCB, the case of the embedded system, etc.). A good average value is 20K/W (see [35]), even though this is much higher than the ones published by package vendors.

[4]The silicon thermal conductivity depends on the temperature.

Thermal model equations

The thermal model consists of a set of differential equations which are resolved in an iterative way. There is an equation associated with every cell, describing how it interacts with its neighbors:

$$\frac{\mathbf{C_{cell}}}{\mathbf{\Delta t}}[\mathbf{T_{cell}(i+1)} - \mathbf{T_{cell}(i)}] =$$

$$= G_{n1}[T_{n1}(i) - T_{cell}(i)] + \ldots + G_{nx}[T_{nx}(i) - T_{cell}(i)] + \ldots$$

$$\ldots + G_{nm}[T_{nm}(i) - T_{cell}(i)] + S_{cell}(i) \quad (3.4)$$

G_{nx}	is the conductance between the n cell and neighbour x
n	is the number of cell
x	$1 \leq x \leq m$ is the position number of the neighbour cell
m	is the total number of neighbours cells
$T_{cell}(i)$	is the cell temperature at $i-th$ time step
$T_{nx}(i)$	is temperature of the neighbour x of the cell n at $i-th$ time step
C_{cell}	is the cell capacitance
$S_{cell}(i)$	is the power burned at $i-th$ time step
Δt	is the time between two time step

The silicon thermal conductivity is not linear and it changes with the temperature. We have calculated its first order Taylor series expansion and we have used it inside the cell equation. The differences between the silicon thermal conductivity and its Taylor series are smalls as depicted in the graph (Figure 3.6). Therefore our silicon conductivity becomes.

$$k_{th}^{si} = k_0 + k_1 \cdot T_{cell} \quad (3.5)$$

The complexity of this model is thus proportional to the number of equations and thus cells. More cells increase the accuracy (resolution), but this comes at the cost of a higher simulation time of the model. To understand the impact of the cell granularity (the number of cells) on the accuracy, we have simulated the same die and heat source, but with a varying number of cells in the model (resp. 200,1200 and 8000 cells; see Figure 3.7).

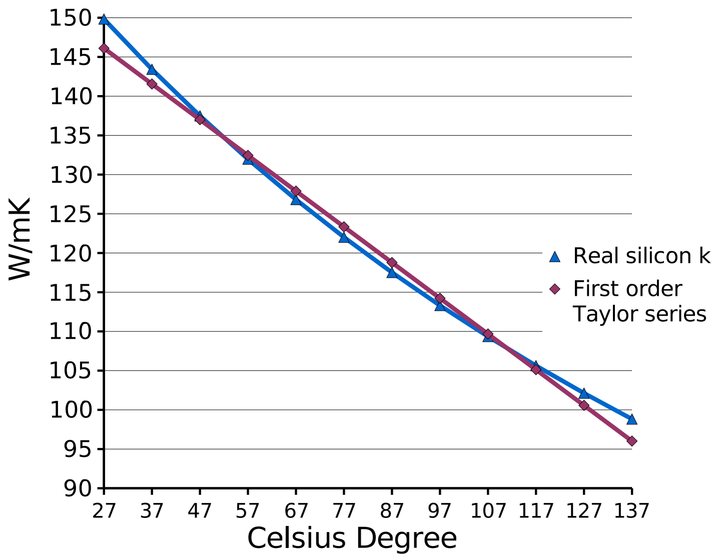


Figure 3.6: Silicon thermal conductivity can be approximate its Taylor series.

Cells number	Max temperature (K)	simulation time (Sec)
200	336.521 (99.62%)	18 (1×)
1200	337.345 (99.86%)	104 (5.77×)
8000	337.821 (100%)	1439 (79.94×)

Table 3.4: Maximum temperature and simulation time for three different resolutions in the $3.2 \times 3.2mm^2$ silicon surface with $0.64 \times 0.64mm^2$ heat source at $819mW$.

More cells result in only a minor change of the steady-state temperature (less than 0.4%), whereas the simulation speed is improved by a factor of 80. In addition, we have explored how the number of cells influences the temporal behavior. In Figure 3.8, we can observe that the loss in precision is always under the one observed steady-state. Moreover this error decreases if we reduce the power dissipated.

As a good trade-off between accuracy and simulation time, we have used a cell size of $150\mu m \times 150\mu m$ throughout our experiments (see Figure 3.1). We assume that the power is uniformly burned in this region (which is 1/8th of the size of an ARM processor in $130nm$). For technologies which have a worse thermal conductance (such as fully depleted SOI), we plan to use smaller thermal cells (down to the level of standard cells).

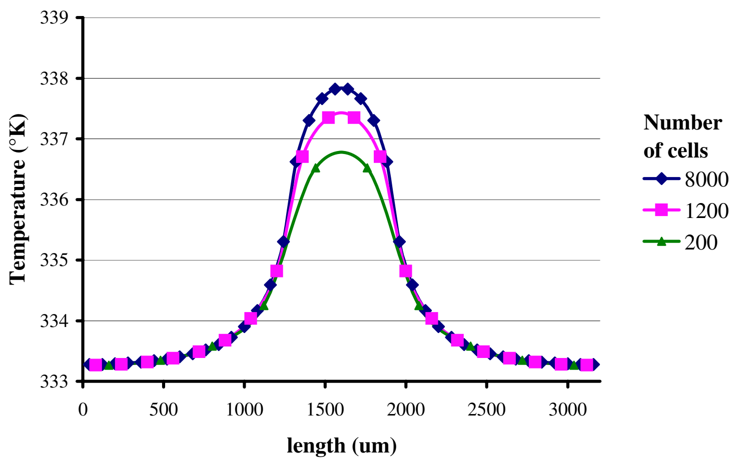


Figure 3.7: Steady-state temperature distribution for three different resolutions in the $3.2 \times 3.2mm^2$ silicon surface with $0.64 \times 0.64mm^2$ heat source at $819mW$.

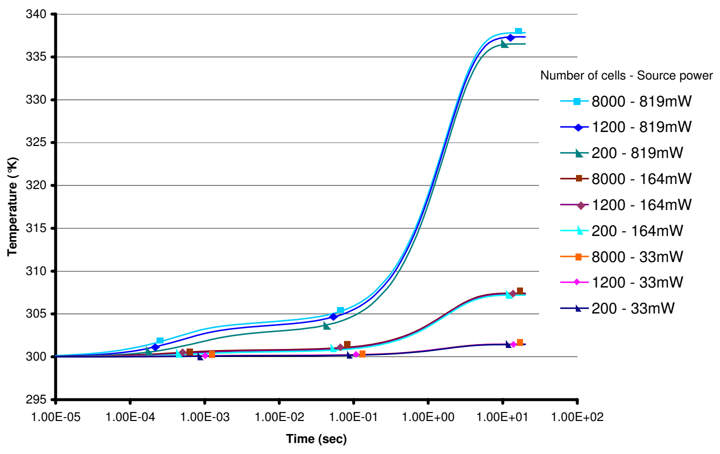


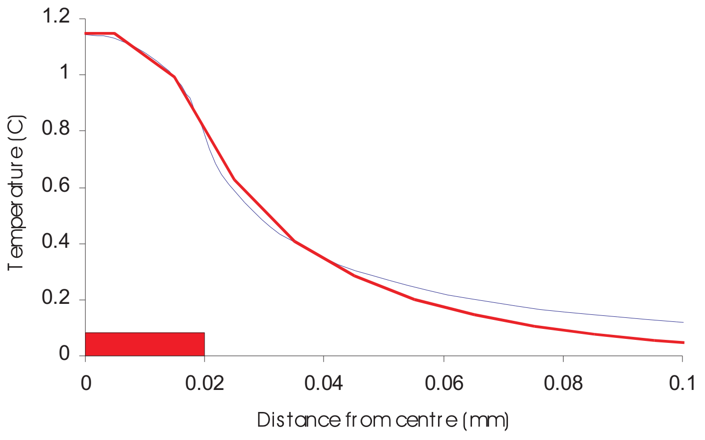
Figure 3.8: Temporal temperature distribution for three different resolutions and powers in the $3.2 \times 3.2mm^2$ silicon surface with $0.64 \times 0.64mm^2$ heat source.

Hot spot dimension(μm^2)	Temp 3D	Temp RC model
60×70	$1.64°C$	$1.58°C$
40×50	$1.14°C$	$1.12°C$
30×30	$0.77°C$	$0.83°C$

Table 3.5: Comparison of the maximum temperature reached with a 3D model and our equivalent RC model.

Model calibration

We have compared and calibrated our thermal model with a 3D-finite element package. For this purpose, we have modelled a single heat source located in the centre of the chip's bottom surface. The temperature of this source as predicted by the 3D model and our RC model is shown in table 3.5.



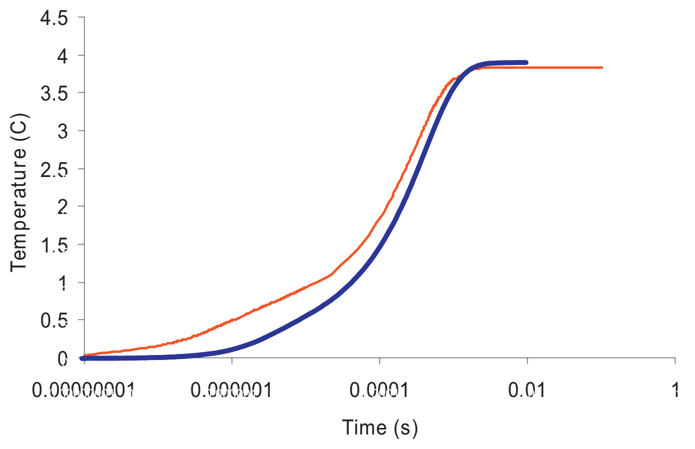


Figure 3.9: Comparison of the spatial (up) and temporal temperature (down) distribution evaluated with 3D finite element package (light line) vs. equivalent RC model (bold line).

Besides predicting the steady state temperature within the hot spot, we also validate our model's prediction of the spatial distribution of the temperature. In Figure 3.9-up, we illustrate how the temperature decreases in function of the radial distance from the centre of the heat source. The size of source is indicated by the grey box. The predictions of both models are again similar.[5] Finally, we test the accurateness of our thermal model for predicting the temporal behaviour of the die. We apply a sudden power load to the centre of the chip and illustrate the temperature response of the die in Figure 3.9-down.

[5]At large distances from the centre of the heat source, our RC model underestimates the temperature, because away from the source, we have used larger cell sizes for reducing its run time.

3.6 Experimental results

3.6.1 Thermal properties of the die

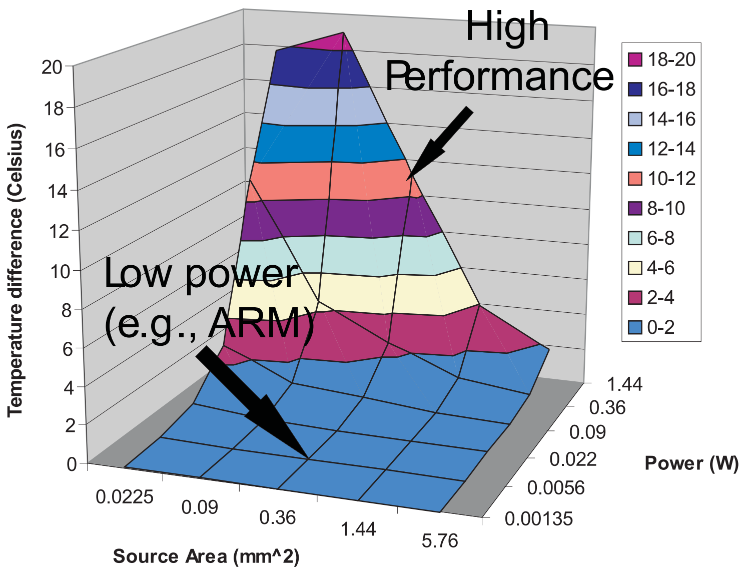


Figure 3.10: Temperature differences on chip ($6mm \times 7.2mm$). It contains in the centre a single source of which the area and power is varied.

To delimit the conditions when larger temperature differences on the die of the MPSoC will occur, an experiment with a single heat source was conducted of which the size and power is varied. The resulting maximum temperature differences on the chip are shown in Figure 3.10.

For a given area of the power source (e.g., $0.36mm^2$), the temperature difference increases proportional to the power (with $7.7K/W$). For a given power budget (e.g., $0.36W$), the temperature is proportional to the inverse of the square root of the area (with $1.8K \cdot mm$). The smaller the power source becomes, the smaller the surface becomes through which the heat can be removed. As a result, the thermal resistance increases with a decreasing diameter of the power source and the largest temperature drop occurs near the power source. The thermal resistance is thus mainly determined by the area of the source rather than the distance from the source to the coldest point on the die. Hence, even if a larger die size is used, the temperature differences will not increase significantly.

This observation is confirmed in Figure 3.11. In this experiment, we investigated the impact of the die size on the steady state-distribution. Burning the same heat source on two different die sizes ($3.2mm \times 3.2mm$ vs $12mm \times 12mm$), we note that the highest temperature difference occurs near the source; in the rest of the chip, there is almost no temperature difference.

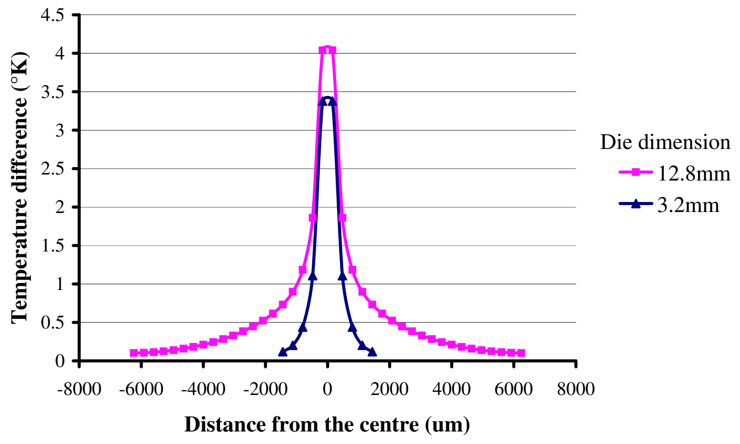


Figure 3.11: Temperature differences on chip for two different dimension of the die ($3.2mm \times 3.2mm$ vs $12mm \times 12mm$).

With Figure 3.10, designers can easily predict the temperature differences that will occur on their die and thus delimit the conditions for which thermal design is required. E.g., in our low-power embedded systems the area of the processor is around $0.3mm^2$ and consumes $5mW$. As can be seen in this graph (and further illustrated in the next section), the resulting hot spot will be very low. High performance systems operate in a different field: they consume much more power for the same area. This is mainly because they operate at a higher clock frequency (e.g., 30 times faster = 30 times more power). For achieving these high clock frequencies, they use a high performance technology in which the leakage contribution cannot be neglected (e.g., leakage is as important as dynamic energy = 2 times more power). Moreover, they use a different circuit style (such as dynamic logic vs. static logic), that is more power hungry (e.g., 2 times more power) and rely on more complex IP blocks (such as complex multiport register-files). As a result, they consume 100 times more power on the same area, reaching power densities larger than $1W/mm^2$. This results important hot spots on the die (12 degrees for our die).

To further validate our results for low power MPSoCs, we look in the thermal issues with more precise thermal/power simulations.

3.6.2 Steady-state thermal analysis

In Figure 3.12-top, we show the temperature differences estimated on the die when running a pipelined matrix multiplication on four processors. Matrix multiplications forms the core of most multi-media algorithms (DCT,Wavelets, etc.). It is a very compute and data intensive application (and thus power hungry). The hottest parts of the chip are the processor cores and their instruction memories, since they are most active and thus dissipate the most power. They are followed by the data cache. Even though the data cache consumes more energy per access than the instruction cache, it is less actively used[6] and therefore does not become a hot spot. When running at $100Mhz$, the temperature differences on the chip are limited (max. 0.128 Celsius). The on-chip temperature difference increases only slightly when six additional processors are started (see Figure 3.12-middle: max. 0.142 Celsius). This is because the processors with a relatively high power density are intermingled with memories which a much lower power density.

By scaling the frequency of the processors from $100Mhz$ to $1GHz$, the power increases with a factor 10. As a result and in agreement with the results of Figure 3.10, the temperature differences on the chip increase with a factor 10. We have measured a temperature difference of 1.53 Celsius.

In a next experiment, we test the impact of technology scaling. We scale all the dimensions of our layout with a factor 2 (reflecting a technology change from $130nm$ to $65nm$). We also assume that the power supply does not scale in future processing generations (which is a worst-case assumption) and use a clock of $1Ghz$ (which is high for low power systems). According to Figure 3.10, we find that scaling the area of the power source by four, results in 1.8 times larger temperature differences on chip. This estimation is confirmed by our accurate simulation: temperature gradients increase from 1.5 Celsius ($130nm@1Ghz$) up to 2.8 Celsius ($65nm@1Ghz$) (see Figure 3.12-bottom).

From these results, we conclude that the on-chip temperature differences of a typical LP-MPSoC are limited. More important, Figure 3.10 allows to easily predict the on-chip temperature differences.

3.6.3 Transient thermal analysis

So far, we have only considered spatial temperature differences. However, temporal temperature differences or thermal cycles are equally important (e.g., they impact reliability). We therefore plot the average die temperature in function of the time (see Figure 3.13-top). As the thermal resistance of the die with the environment is rather high ($20K/W$), it takes around $8s$ before the steady state temperature is reached. The resulting temperature depends on the power burned on the chip: the more processors are running, the higher the final temperature becomes (see the differences between 4 and 10 cores in the Figure). In the Figure 3.13-middle, we plot the maximum temperature difference in function of the time. The steady state temperature difference is reached after only $250ms$. This is much faster than the temperature equilibrium of the die with the

[6]The register file acts as an extra level of cache and thus eliminates accesses to the data cache

environment. It can be explained by the fact that the silicon die and the copper heat spreader are good thermal conductors. Since the thermal time constant on the die is low, the on-chip temperature differences may be very sensitive to variations of the power consumptions, i.e. thermal cycling. To analyse thermal cycles due to a varying workload, we have generated an artificial benchmark application running on a single processor. It consists of a period of high activity followed by one of low activity. Its power and temperature profile are shown in Figure 3.13-bottom. Globally, the temperature increases as the steady state temperature of the die has not been reached yet.[7] A series of thermal cycles is superposed on this gradual increase of temperature. Their amplitude is very small as the thermal resistance of the die is small.

Large thermal cycles are mainly due to the high resistance of the package with the environment and only occur at a large time scale. Therefore, we conclude that thermal cycling (and the resulting reliability issues [28]) are less of a problem in LP-MPSoCs than in high performance systems.

3.7 Conclusions

In this chapter, we have investigated the need for temperature-aware design in LP-MPSoCs. We have built a thermal model which we have calibrated with a 3D finite-element analysis. Based on our experimental results for a typical LP-MPSoC, we observe that no hot spots occur across the die. In the context of LP-MPSoCs implemented on bulk CMOS and under the plausible assumption that LP-MPSoCs will not rush for super-fast clocks (such as defined in the ITRS roadmap for high performance logic), we therefore do not see the immediate need for techniques to analyse and reduce hot spots. However, if more advanced packaging solutions (such as 3D stacking) and new low-k dielectrics are introduced in silicon bulk technology, the thermal model of the chip may fundamentally change. The presence of hot spots in these novel technologies has to be investigated for LP-MPSoCs. Furthermore, as the steady-state temperature depends on the packaging solution and the applied workload, temperature-aware design remains necessary to assure that the maximal temperature of the system is not breached.

[7]The measurement is done after $5s$, but the steady state temperature is only reached after $8s$.

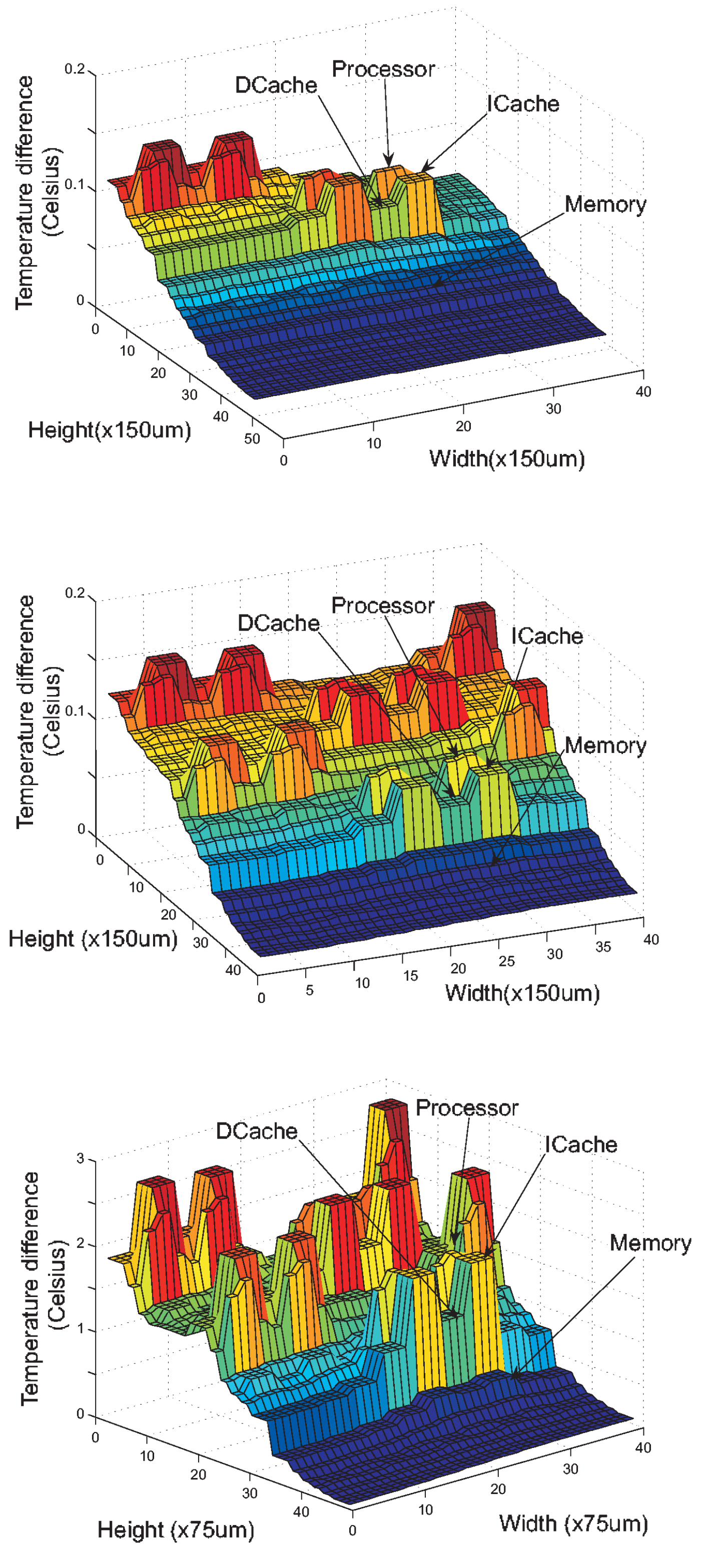


Figure 3.12: Temperature differences of a multi-processor system-on-a-chip. (top) 4 processors running at $100MHz$; (middle) 10 processors at $100Mhz$; (bottom) 10 processors at $1GHz$ on a scaled die size.

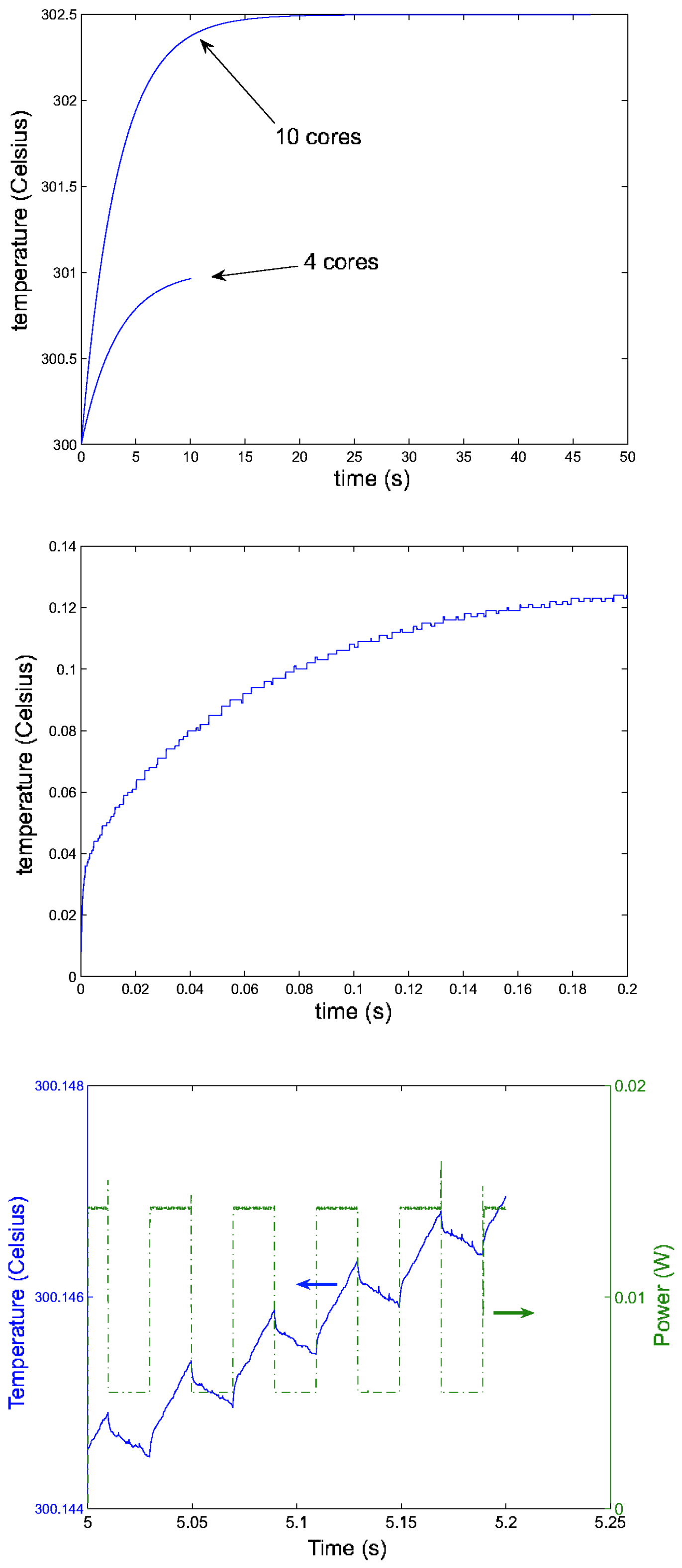


Figure 3.13: Temporal behaviour of the die: (top) average die temperature; (middle) temperature differences on chip; (bottom) thermal cycles on chip.

Chapter 4

HW-SW Emulation Framework for Temperature-Aware Design in MPSoCs

4.1 Overview

New tendencies envisage Multi-Processor Systems-On-Chip (MPSoCs) as a promising solution for the consumer electronics market. MPSoCs are complex to design, as they must execute multiple applications (games, video), while meeting additional design constraints (energy consumption, time-to-market). Moreover, the rise of temperature in the die for MPSoCs can seriously affect their final performance and reliability. In this chapter, we present a new hardware-software emulation framework that allows designers a complete exploration of the thermal behavior of final MPSoC designs early in the design flow. The proposed framework uses FPGA emulation as the key element to model the hardware components of the considered MPSoC platform at multi-megahertz speeds. It automatically extracts detailed system statistics that are used as input to our software thermal library running in a host computer. This library calculates at run-time the temperature of on-chip components, based on the collected statistics from the emulated system and the final floorplan of the MPSoC. This enables fast testing of various thermal management techniques. Our results show speed-ups of three orders of magnitude compared to cycle-accurate MPSoC simulators.

4.2 Introduction

An increasing number of multimedia services (e.g., multi-view video or multi-band wireless protocols) are being implemented on embedded consumer electronics thanks to the fast evolution of process technology. These new embedded systems demand complex multi-processor designs to meet their real-time processing requirements while respecting other critical embedded design constraints, such as low energy consumption or reduced implementation size. Moreover, the consumer market is reducing more and more the time-to-market and price [56], which does not permit anymore complete redesigns of such multi-core systems on a per-product basis. Thus, *Multi-Processor Systems-on-Chip (MPSoCs)* have been proposed as a promising solution for this context, since they are single-chip architectures consisting of complex integrated components communicating with each other at very high speeds [56]. Nevertheless, one of their main design challenges is the fast exploration of multiple *hardware (HW)* and *software (SW)* implementation alternatives with accurate estimations of performance, energy and power to tune the MPSoC architecture in an early stage of the design process. In addition, it has been recently outlined the problem of temperature rise in future technologies in the components of MPSoCs [68], which increases further their system integration complexity.

With the objective to explore the HW-SW interaction, several MPSoC simulators have been proposed, both at transaction and cycle-accurate levels using *Hardware Description Languages (HDL)* languages and SystemC [43, 44, 49]. Also, recent SW tools can be added to them to evaluate in detail thermal pressure in on-chip components based on run-time power consumption and floorplanning information of final MPSoCs [68]. Nevertheless, although these complex combined SW environments achieve accurate estimations of the system with thermal analysis, they are very limited in performance (circa 10-100 KHz) due to signal management overhead. Thus, such environments cannot be used to analyze MPSoC solutions with complex embedded applications and realistic inputs of the final working environment to cover the variations in data loads at run-time. Moreover, higher abstraction levels simulators attain faster simulation speeds, but at the cost of a significant loss of accuracy. Hence, they are not suitable for fine-grained architectural tuning or thermal modeling.

One solution for the speed problems of cycle-accurate simulators is HW emulation. Various MPSoC emulation frameworks have been proposed [46, 53, 38]. However, they are usually very expensive for embedded design (between $100K and $1M). Moreover, they are not flexible enough for MPSoC architecture exploration since they mainly aim at large MPSoCs prototyping or SW debugging. Typically, the baseline architectures (e.g., processing cores or interconnections) are proprietary, not permitting internal changes. Furthermore, to the best of our knowledge, no flexible interconnection interfaces between HW emulation and the existing thermal SW libraries exist today. Thus, thermal effects can only be verified in the last phases of the design process, when the final components have been already developed, which can produce large overheads in system integration due to cores and MPSoC architecture redesigns if any problem is discovered at that moment.

In this chapter we present a new HW-SW *Field-Programmable Gate Array (FPGA)*-based emulation framework that allows designers to explore a wide range of design alternatives of complete MPSoC systems at cycle-accurate level, while characterizing their thermal behavior at a very fast speed (i.e., 100 MHz) with respect to MPSoC architectural simulators. First, MPSoC HW components are mapped on an FPGA to extract a large range of critical statistics from three key architectural levels of MPSoC systems (i.e., processing cores, memory subsystem and interconnection mechanisms), while real-life applications are executed. Second, this run-time information is sent through a flexible interface (using a standard Ethernet connection) to a configurable thermal model SW tool running on a host PC, which evaluates at the same speed as the emulation executes the thermal behavior of the final MPSoC design, and returns this information to the FPGA emulating it. This final step enables testing run-time temperature management strategies in real-time. Our results illustrate that the proposed HW-SW framework achieves detailed cycle-accurate reports with speed-ups of three orders of magnitude compared to state-of-the-art cycle-accurate MPSoC simulators. Moreover, our experiments indicate the benefit of the proposed framework to study the importance of packaging floorplan features in MPSoC designs.

4.3 Related Work

It is widely accepted that MPSoCs represent a promising solution for forthcoming complex embedded systems [56]. This has spurred research on modeling and prototyping MPSoC designs, using both HW and SW.

From the SW viewpoint solutions have been suggested at different abstraction levels, enabling trade-offs between simulation speed and accuracy. First, fast analytical models have been proposed to prune very distinct design options using high level languages (e.g., C or C++) [44]. Also, full-system simulators, like Symics [58] and others, have been developed for embedded SW debugging and can reach megahertz speeds, but they are not able to capture accurately performance and power effects (e.g., at the interconnection level) depending on the cycle-accurate behavior of the HW. Second, transaction-level modeling in SystemC, at the academic [63] and industrial side [49, 42], have enabled more accuracy in system-level simulation at the cost of sacrificing simulation speed (circa 100-200 KHz). Such speeds render unfeasible the testing of large systems due to the too long simulation times, conversely to the proposed emulation framework. Moreover, in most cases SW simulations are only limited to a number of proprietary interfaces (e.g., AMBA [40] or Lisatek [49]). Finally, important research has been done to obtain cycle-accurate frameworks in SystemC or HDL languages. Companies have developed cycle-accurate simulators using post-synthesis libraries from HW vendors [59, 71]. However, their simulation speeds (10 to 50 KHz) are unsuitable for very complex MPSoC exploration. In the academic context, the MPARM SystemC framework [43] is a complete simulator for system-exploration since it includes cycle-accurate cores, complex memory hierarchies (e.g., caches, scratch-pads) and interconnects, like AMBA or *Networks-on-Chip (NoC)*. It can extract reliable energy and performance figures, but its major shortcoming is again its simulation speed (120 KHz in a P-IV at 2.8 GHz).

An important alternative to MPSoC prototyping and validation is HW emulation. In industry, one of the most complete sets of statistics is provided by Palladium II [46], which can accommodate very complex systems (i.e., up to 256 Mgate). However, its main disadvantages are its operation frequency (circa 1.6 MHz) and cost (around $1 million). Then, ASIC Integrator [40] is much faster for MPSoC architectural exploration. Nevertheless, its major drawback is the limitation to up to five ARM-based cores and only AMBA interconnects. The same limitation of proprietary cores for exploration occurs with Heron SoC Emulation [53]. Other relevant industrial emulation approaches are System Explore [38] and Zebu-XL [50], both based on multi-FPGA emulation in the order of MHz. They can be used to validate intellectual property blocks, but are not flexible enough for fast MPSoC design exploration or detailed statistics extraction. In the academic world, a relatively complete emulation platform up-to-date for exploring MPSoC alternatives is TC4SOC [61]. It uses a proprietary 32-bit VLIW core and enables exploration of interconnects by using an FPGA to reconfigure the *Network Interfaces (NIs)*. However, it does not enable detailed extraction of statistics and performing thermal modeling at the other two architectural levels we propose, namely memory hierarchy and processing cores. Last, an interesting approach that uses FPGA prototyping to speed up co-verification of pure SW simulators is described in [60]. In this case the FPGA part is synchronized in a cycle-by-cycle basis with the C/C++ SW part by using an array of shared registers in the FPGA that can be accessed by both sides. This work shows a final speed for the combined framework of 1 MHz, outlining the potential benefits of combined HW-SW frameworks, which we fully exploit in this approach to reach an MPSoC emulation speed of 100 MHz.

Regarding thermal modeling, [68] presented a thermal/power model for super-scalar architectures. It can predict the temperature variations between the different components of a processor and show the subsequent increased leakage power and reduced performance. Additionally, [70] investigated the impact of temperature and voltage variations across the die of an embedded core. Its results show that the temperature can vary around 13.6 degrees across the die. Also, in [57] the temperature of FPGAs used as reconfigurable computers is measured using ring-oscillators, which can dynamically be inserted, moved or eliminated. This empirical measurement method is interesting, yet it is only applicable to FPGAs as target devices. Our method alternatively aims at estimating the temperature of integrated circuits implementing MPSoC designs. Nevertheless, all these works clearly prove the importance of hot spots in high-performance and reconfigurable systems, and the need for temperature-aware design and tools to support it.

Based on the previous and other similar thermal models, *Dynamic Thermal Management (DTM)* techniques have been suggested for processors using both architectural adaptation, *Dynamic Voltage Scaling (DVS)*, *Dynamic Frequency Scaling (DFS)* and profiling-based techniques. In [67], it is proposed to use formal feedback control theory as a way to implement adaptive techniques in the processor architecture. In [69] a predictive frame-based DTM algorithm, targeted at multimedia applications, is presented. This algorithm uses profiling to predict the theoretical highest performance within a thermally-safe HW configuration for the remaining frames of a certain type. Also, [45] performed extensive studies on empirical DTM techniques (i.e., DVS, DFS, fetch-toggling, throttling, and speculation control) when the power consumption of a processor crosses a predetermined threshold (i.e., 24W). Its results showed that DFS and DFS can be very inefficient if their invocation time is not set appropriately. Additionally, [65] suggested not to schedule hot tasks when the temperature reaches a critical level. In this way the CPU spends more time in low-power states, such that the temperature can be either locally or globally decreased. In this work we address the problem of empirical validation of such approaches with long thermal simulations using real-life workloads, which becomes feasible with our HW-SW thermal emulation tool. In fact, a simple DFS mechanism based on the previous works is presented in our experiments to illustrate the flexibility of the proposed HW-SW FPGA-based framework to interact with the SW part and to explore in real-time different temperature-management policies.

Finally, another interesting research line to ease the problem of temperature in future MPSoCs is temperature-aware placement [48, 47, 51]. In this case the temperature issues are addressed at design-time to ensure that circuit blocks are placed in such a way that they even out the thermal profile. All these techniques are complementary to our approach, since we assume that the final floorplan and core placement phases have been already performed. Hence, our tool is able to take the outcome of any of the previous approaches and validate their predicted results during the execution of realistic applications of the target working environments.

4.4 MPSoC Emulation Architecture

The proposed MPSoC framework uses FPGA emulation as the key element to model the HW components of MPSoCs at multi-megahertz speeds, and extract detailed system statistics used in our SW thermal library running in a host computer. An overview of the baseline HW architecture of the MPSoC emulation platform is depicted in Figure 4.1. It consists of three main elements:

1. Different MPSoC processing cores, such as, Power PC, Microblaze, ARM or VLIW cores.
2. The definition of configurable I- and D-cache, as well as main memories (i.e., private and shared memories between processors).
3. Various interconnection mechanisms between the L1 memory hierarchy and the main memory, namely, buses and NoCs.

These elements are designed in standard and parameterizable VHDL and mapped onto a Xilinx Virtex 2 Pro vp30 board (or V2VP30) with 3M gates, which costs $2000 approximately in the market, and that includes two embedded Power PCs, various types of memories (i.e., SRAM and DDR) and an Ethernet port. However, any other FPGA could be used instead. The only requirements are the availability of an Ethernet core to interact with the SW thermal tool, a compiler for the included cores, and a method to upload both the FPGA synthesis of our framework and the compiled code of the application under study. In our case, Xilinx provides all these basic tools in its *Embedded Development Kit (EDK)* framework for FPGAs.

In addition, note that the purpose of our emulator is not prototyping final HW components in MPSoC systems, but the construction of an emulation and fast exploration tool that can be used by designers to explore the desired characteristics and thermal effects of the eventual system. Therefore, our framework includes mechanisms to configure the exploration and hide the physical characteristics of the underlying HW that do not match the selected values (conversely to traditional prototyping).

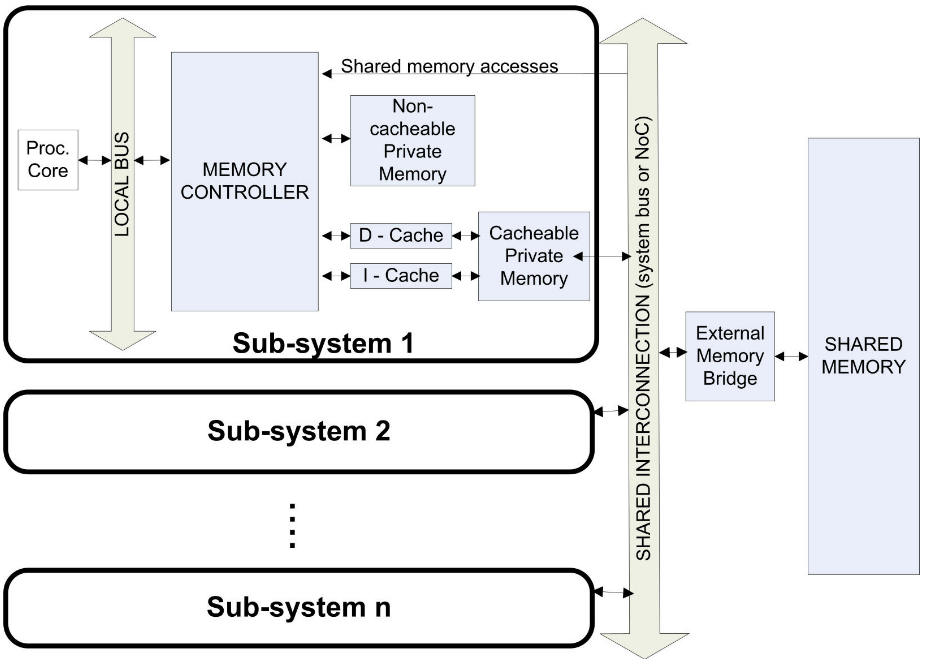


Figure 4.1: Overview HW architecture of emulated MPSoCs.

In the following subsections we describe in detail the architecture and advanced emulation mechanisms of the different elements included in our emulation platform. We also depict the synthesis figures for each component.

4.4.1 Processing Elements

In our framework, various types of processing cores can be included, both proprietary and public ones. The accepted input forms are netlists mapped onto the underlying FPGA and HDL languages (i.e., Verilog, VHDL or Synthesizable SystemC). This addition of cores is possible since the memory controller that receives the memory requests in our system includes an external pinout interface and protocol that can be easily modified to match the respective ones of the studied processor. Moreover, only the instruction-set emulation part of the core is required because its memory hierarchy (e.g., caches or scratch-pad) is replaced by our framework to explore different memory configurations.

In the current version of the system, we have ported a hard-core (PowerPC 405) and a RISC-32 soft-core (Microblaze) provided by Xilinx. None includes HDL sources, only netlist mapping, and the inclusion process for their pinout interfaces and protocols required one week. Regarding platform scalability for MPSoC designs, a complete Microblaze requires only 4% of the total resources of our V2VP30 FPGA (574 out of 13.696 slices).

4.4.2 Memory Hierarchy

As Figure 4.1 indicates, in the basic emulated architecture two memory levels presently exist: L1 cache memories and main memories. However, it requires few minutes to add additional cache memory levels or private memories to each processing element, either on a per processor or by processor-group basis. The main element in the memory hierarchy that enables this easy integration of new memory devices and protocols is the memory controller. One memory controller is connected to each processing core to capture all its memory requests. Then, the memory controller forwards them to the necessary element of the memory subsystem according to the demanded memory address. In the current implementation, it takes 2% of the total available resources of our V2VP30 FPGA (270 slices), and includes interfaces and protocols for four memory components and three different memory address ranges:

1. Private main memory, cacheable or non-cacheable, addressable in a configurable memory range of each processor: It is possible to configure its size and latency, as long as enough *Block Random Access Memory (BRAM)* resources exist. Its synthesis takes 1% of the V2VP30 (181 slices), apart from used BRAM that depends on the desired size.

2. Shared main memory, cacheable or non-cacheable according to user's configuration: It is possible to configure its total size and latency, and does not take any area in the FPGA since it uses real memories available on the board, namely, *Synchronous Random Access Memories (SRAM)* or *Double Data Rate Synchronous Dynamic RAM (DDR-SDRAM or DDR memories)*.

3. Private HW-controlled D- and I-cache: It is possible to define independently their total sizes, line sizes and latencies to explore different design alternatives. In our experiments, both caches are direct-mapped. However, their modular designs include in different concurrent processes the replacement policy and associativity features, making easy to change this configuration with additional algorithms to test. Its synthesis uses 1% of the V2VP30 (181 slices) and the amount of used BRAM varies according to the desired size.

Finally, each memory controller is able to observe and synchronize different clock domains, due to its multiple external interfaces (see Figure 4.1). It has internal counters for each type of connected memories to keep track of the elapsed time and compare it with the user-defined latencies. Then, the memory controller informs the *Virtual Platform Clock Manager (VPCM)* to stop the clock of the processor during the emulation each time one physical memory device cannot fulfill the defined latency. Hence, the stopped processor preserves its current internal state until it is resumed by the clock manager, when the memory controller informs that the information requested is available. This mechanism enables trade-offs between emulation performance and use of resources. Currently, our memory controller monitors two clock domains: one is used for the microprocessor and another one is used for the memories and the memory controller itself.

4.4.3 Interconnection Mechanisms

The third configurable element in our MPSoC emulation framework is the interconnection mechanism between the memory controller and main memory (i.e., in BRAM, SRAM, or DDR memories). At this level, we have included both buses and NoCs. To enable this variety of choices, apart from multiple types of interfaces of the memory controller, we have also included a configurable main memory bridge in the device side. It includes two different public pinout interfaces: one corresponds to the memory and the other one to the instantiated interconnection. Similarly as with the memory controller, this enables us to extend the current list of available interconnection mechanisms by modifying the required pinout and protocol.

In the current version, the two available buses on Xilinx FPGAs are included, i.e., *On-Chip Peripheral Bus (OPB)* for general-purpose devices and *Processor Local Bus (PLB)* for fast memories and processors. Also, we have created our own 32-bit data/address bus for exploration purposes. It is inspired by the basic functionality of the AMBA 2 AHB interconnect [39], where the bandwidth and arbitration policies can be configured. Thus, starting from the initial OPB scheme, we have removed the signals used for advanced arbitration schemes, transaction parking request, etc. Also, the arbitration protocol is specified at compile time; Hence, avoiding the need for a dedicated signal to set the arbitration mode. Currently, the allowed arbitration modes are: priority-based, and round robin. In addition to this, also the latency and bus-width can be configured. For our experiments the arbitration latency is one cycle, and is connected to all processing cores through an OPB interface and to an external SRAM memory through a custom SRAM controller. Its synthesis (including the SRAM controller) represents 1% of the V2P30 FPGA (210 slices).

In addition, we have included the possibility to explore custom-made NoC solutions. The synthesizable NoC code is generated using the Xpipes NoC Compiler [55]. It allows for studying topologies with any number of switches, links with bandwidth constraints and NIs to connect external cores to the NoC. We have modified the memory controller and the main memory bridges to generate *Open Core Protocol (OCP)* transactions as the Xpipes NIs require [55]. Regarding FPGA utilization, a complex NoC-based system with 6 switches of 4 input/output channels and 3 output buffers uses 70% of the V2P30 FPGA (9659 slices).

The inclusion of each of these buses and NoC interfaces in our framework required one week of work. Furthermore, as with processing cores, any other high-performance proprietary bus (e.g., AMBA, STBus, etc.) can be added to our emulation framework as a black-box, since the integration process only requires to know the used protocol and external bus pinout.

4.5 Statistics Extraction Subsystem

The main feature pursued in the design of the statistic extraction subsystem is its transparent inclusion in the basic MPSoC architecture to be evaluated, and with minimum performance penalty in the overall emulation process. For this purpose, as it is depicted in Figure 4.2, we have implemented HW sniffers that monitor certain signals of the memory controller and the external pinout of each device included in the emulated MPSoC. These sniffers calculate, among other statistics, the energy consumed in each cell of the floorplan of the emulated MPSoC, and stores the final values in a buffer created in the FPGA BRAM memory. Finally, the buffers are concurrently processed by our network dispatcher to generate *Medium Access Control (MAC)* packets in our own format, and send them by an Ethernet port to the SW thermal modeling library running in the connected computer. One key additional element in this extraction mechanism is the VPCM module, which enables stopping/resuming the statistics extraction mechanism in case of congestion of the Ethernet connection.

4.5.1 HW Sniffers

The HW sniffers transparently extract the statistics from each MPSoC component defined in the floorplan. From a design point of view, all sniffers in our platform share a common structure. They have a dedicated interface to capture internal signals from the module they are monitoring, and a connection to our custom statistics bus. To create a new sniffer the designer only needs to define what to monitor in the component and how to connect the sniffer to the bus. For temperature monitoring, HW sniffers measure the time that each processor spends in active/stalled/idle mode at run-time, and the number and type of accesses to the memories in the system (i.e., I- and D-cache, and large shared and private main memories). At the interconnection level (buses or NoC), the monitored values are the number of signals transitions. There is an skeleton

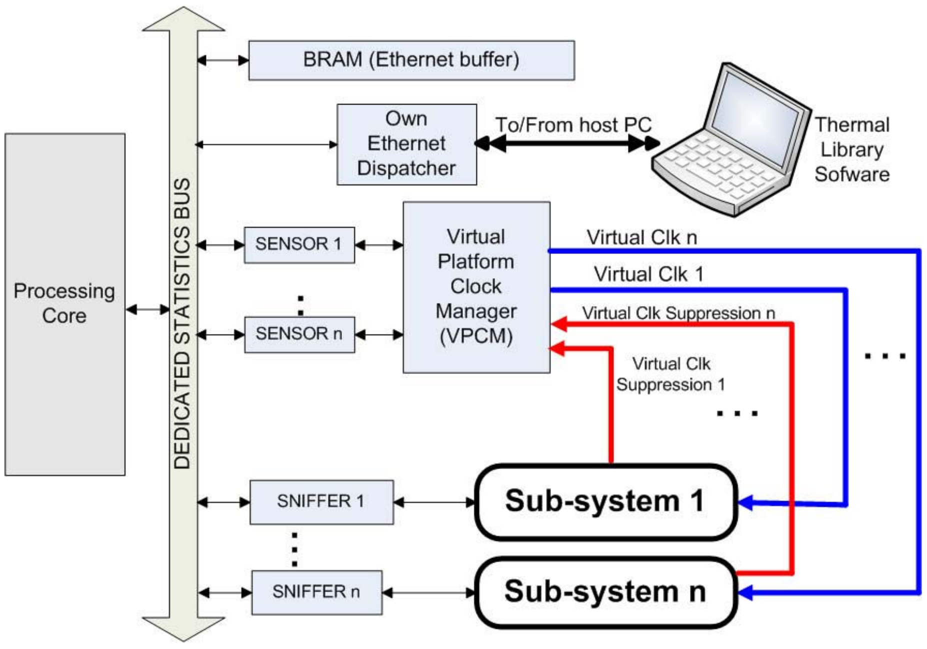


Figure 4.2: Overview of the statistics extraction subsystem.

available to ease the creation of new sniffers for this purpose. Currently, we provide two different types of sniffers. The first one, called event-logging, exhaustively logs all interesting events that occur in the platform. The second type, called count-logging, only counts events, such as cache misses, bus transactions, memory accesses, etc.; Thus, it generates more concise results, and what typically designers demand from cycle-accurate simulators to test their systems. Our experimental results with real-life MPSoC designs indicate that, practically an unlimited number of event-counting sniffers can be added to the design without deteriorating at all the emulation speed. This establishes one of the main differences with SW cycle-accurate simulation systems: the addition of additional cores or analysis sniffers to the MPSoC architecture does not slow down the emulation process, due to the implicit concurrent synchronization of signals between different HW modules working in parallel to be able to compose a complete MPSoC architecture. In fact, the HW sniffers merely act as an additional HW component that transparently monitor the switching activity of signal and internal states of the bare MPSoC HW architecture.

Finally, as an example to evaluate how much overhead in FPGA area the statistics extraction subsystem represents, the amount of resources used by one event-logging sniffer is 0.1% (14 slices), while for an event-counting sniffer is about 0.2% (31 slices).

4.5.2 Virtual Platform Clock Manager (VPCM)

The VPCM is the HW element used in our framework to provide multiple virtual clock domains. This module generates as output the clock signals used in the emulated MPSoC subsystems (VIRTUAL CLK signals in Figure 4.2). It receives three different types of input signals. First, the physical clock generated in the oscillator of the FPGA (not shown in Figure 4.2 for simplification purposes), which in the current implementation is set to 100 MHz. Second, one signal from each memory controller of the emulated MPSoC subsystems (VIRTUAL CLK SUPPRESSION 1..N in Figure 4.2) used to request a virtual clock inhibition period if any attached memory device of the emulated hierarchy is not able to return the requested value at this moment respecting its set user-defined latency. Third, signals coming from the different virtual temperature sensors (SENSOR 1..N in Figure 4.2) that monitor if any component has increased its temperature beyond/below a certain threshold. This mechanism enables the use of run-time thermal management policies. The virtual temperature sensors are regular registers that currently store the updated run-time temperature coming from the SW thermal library running in the host computer. However, in the final MPSoC they would be replaced by real sensors. Then, the use of virtual clock domains generated by the VPCM is two-fold:

- First, the emulation of MPSoCs can be done for different physical features than those of available HW components. Once the respective VIRTUAL CLK SUPPRESSION 1..N signal is high, the corresponding VIRTUAL CLK signal of that sub-system (or the set of sub-systems) is activated. Then, the stopped processor preserves its current internal state until it is resumed by the VPCM, when the memory controller informs that the information requested is available in the accessed memory. This mechanism allows us to implement the corresponding memory resources either in internal FPGA memory (optimal performance) or with external memories (bigger size), while balancing emulation performance and use of resources. For instance, if the desired latency of main memories are 10 cycles, but the available type of memory modules in the FPGA are slower (e.g., use of DDR instead of SRAMs), the VPCM can stop the clock of the processors involved at run-time. Thus, it can hide the additional clock cycles required by the memory. Our VPCM includes two clock domains: (1) microprocessor, memories and interconnections; (2) memory controllers.

- Second, the virtual clock of all or part of the components in the emulated MPSoC can be transparently stopped/resumed at run-time in case of saturation of the Ethernet connection during the download/upload of the extracted statistics/estimated temperatures.

The combination of these two mechanisms enables the execution and thermal modeling of HW configurations of the emulated MPSoC at a different speed than the allowed clocked speed of the available HW components. In fact, it is similar to the mechanism used in SW simulations, but at a much higher frequency. For instance, it is possible to explore the effects in thermal modeling of a final system clocked at 500 MHz, even if the present cores of the FPGA can only work at 100 MHz. To this end, instead of using a 10 ms statistics sampling frequency with a desired virtual clock emulation of 500 MHz, our framework uses a virtual clock of 100 MHz (maximum clock allowed in the FPGA emulation after synthesis). This clock is 5× slower than the desired emulated clock and collects the statistics every 50 ms, but the switching activity in each MPSoC component monitored at this interval is equivalent to the target system for 10 ms. Therefore, our framework samples every 50 ms of real execution, but is analyzed by the SW thermal library as representing 10 ms of the target MPSoC emulated execution. The major requirement in this case is the definition of the sampled/emulating frequency and the target MPSoC frequency to configure the SW thermal model accordingly. The SW thermal model is described next.

4.6 MPSoC SW Power/Thermal Modeling

Our SW thermal tool is a C++ library that enables thermal exploration of silicon bulk chip systems. It can evaluate the thermal behavior in devices modeled at different levels of abstraction (i.e., gate level, RTL level and architectural level). The switching activities of the wires and the components in the die for this thermal analysis are obtained from our FPGA-based MPSoC emulation. Then, the library can be configured in multiple ways to evaluate the thermal behavior of different alternatives for each final MPSoC chip. For instance, its space resolution for thermal accuracy is configurable (i.e., number of temperature cells in a fixed area) as well as many other packaging parameters (e.g., quality of heat sink, thermal capacitance of the different materials that compose the chip, etc.). In our experiments, this flexibility in the thermal library configuration is used to investigate the run-time thermal behavior of multiple cores and embedded memories on a single die in case of different package solutions and floorplan designs for MPSoCs.

In the next subsections, we first discuss the utilized power model. Second, we explain the thermal model in detail. Finally, we review the thermal calculation speed and accuracy of the current implementation of the library.

4.6.1 Power estimation

In Table 4.1, we outline the power consumption and power densities for the most important components of the evaluated MPSoCs as illustration of what our tool requires. We use, as Table 4.1 indicates, the maximum power numbers for each component as worst case, but the effective power can normally be lower, depending on the workload (activities of processors and memories), and can be given as an input by the designer for his particular design. These values have been derived from industrial power models for a 0.13 μm technology.

Regarding leakage power, we currently assign a fixed 10% weight to leakage energy. This figure actually corresponds to the indications of the *International Technology Roadmap for Semiconductors (ITRS)* [66] for low-standby power systems in 0.13 μ with supply voltage of 1.2-1.3V. ITRS outlines that in this case Vdd/Vt are very aggresively scaled to guarantee sufficient battery-life time; Thus, using an optimal Vdd/Vt operating point results in very limited leakage power variations for different working temperatures. However, in more recent technology nodes leakage variations will become more important; Thus, our SW thermal library and HW sniffers can be extended accordingly.

Table 4.1: Power for the most important components of an MPSoC design using a 0.13 μm bulk CMOS technology.

MPSoC Component	Max. Power (at 100 MHz)	Max. Power (at 500 MHz)	Max. Power density
RISC 32-ARM7	5.5mW	—	$0.03W/mm^2$
RISC 32-ARM11	0.3W	1.5W	$0.52W/mm^2$
DCache 8kB/DM (ARM7)	28mW	—	$0.028W/mm^2$
DCache 8kB/DM (ARM11)	142mW	710mW	$1.97W/mm^2$
ICache 8kB/DM (ARM7)	28mW	—	$0.028W/mm^2$
ICache 8kB/DM (ARM11)	142mW	710mW	$1.97W/mm^2$
Memory 32kB (ARM7)	11mW	—	$0.01W/mm^2$
Memory 32kB (ARM11)	55mW	275mW	$0.76W/mm^2$
NoC switch (6x6-32b)	56mW	257mW	$0.08W/mm^2$
NoC network interface	23mW	128mW	$0.02W/mm^2$

4.6.2 Thermal estimation

In our case we consider MPSoCs HW that are made of silicon die wrapped into a package placed on a *Printed Circuit Board (PCB)*, with a variable cost (from low-cost to high-cost packaging). In this case, as shown in Figure 3.3, the heat flow starts from the bottom surface of the die and goes up to the silicon, passes through the heat spreader and ends at the environment interface, where the heat is spread by natural convection [68]. Therefore, for modeling the heat flow, we rely on an equivalent electrical RC model (Figures 3.4, 3.5). Then, two different RC models are presently supported. On the one hand, we have developed our own thermal model of MPSoC considering non-linear resistances inside the silicon [62], in order to match the behavior of thermal conductivity. Then, we consider the heat spreader made of copper and use linear resistances to model it. Currently, we can analyze 2 seconds of simulation (in a 660-cell floorplan), in 1.65 seconds on a P-4 at 3GHz, which is fast enough to interact in real-time with our FPGA-based MPSoC emulation. On the other hand, we have cross-checked our results by including in our tool the possibility to use the Hotspot v3.0 thermal model [68]. It is an accurate model for high performance processors based on an equivalent circuit of linear thermal resistances and capacitances, which correspond to micro-architecture blocks and essential aspects of the thermal package. This model has been validated using finite element simulation. In the following subsections, we describe more in detail our own thermal model and refer to [68] for a more detailed explanation of this library.

Modeling the heat flow

A low-power MPSoC is usually packed within a plastic ball grid array package [72] (see Figure 3.3). In our library, we assume that all surfaces, but the one of the heat spreader are adiabatic. The spreader disposes the generated heat by natural convection with the ambient.

Then, similar to [68][70][52], we exploit the well-known analogy between electrical circuits and thermal models. We decompose the silicon die and heat spreader in elementary cells, which have a cubic shape (Figure 3.5) and use an equivalent RC model for computing the temperature of each cell. By varying the cell size and number of cells we can trade-off simulation speed of the thermal library with its accuracy. In our experiments we have used two basic floorplans: (a) 4 ARM7 cores at 100 MHz; (b) 4 ARM11 running at 100 or 500 MHz, both in 0.13 μm technology (Figure 4.3). The interconnect is clocked at the same frequency that the cores in each case. The cell sizes used in both cases are $150um * 150um$. We assume that the power is uniformly burned in this region, which represents 1/8th of the size of an ARM processor in 0.13 μm. For technologies with a worse thermal conductance, such as, fully depleted silicon-on-insulator [66], it is possible to use smaller thermal cells (down to the level of standard cells).

Equivalent RC thermal model

We associate with each cell a thermal capacitance and five thermal resistances (Figure 3.5). Four resistances are used for modeling the horizontal thermal spreading whereas the fifth one is used for the vertical thermal behavior. The

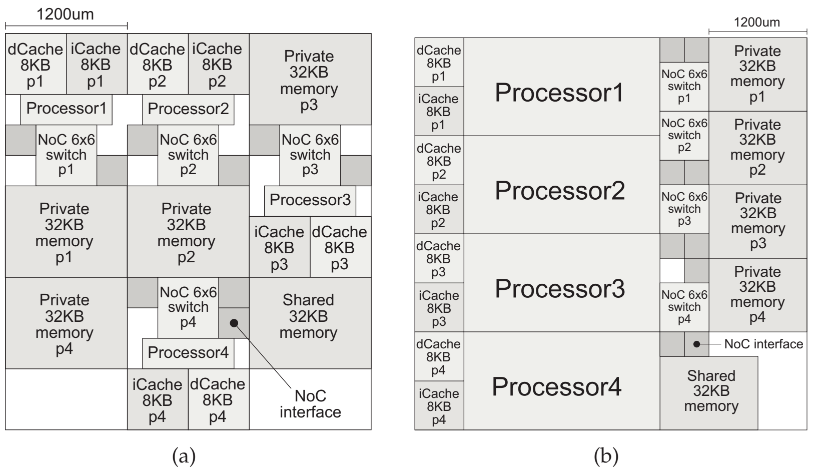


Figure 4.3: MPSoC floorplan with (a) 4 ARM7 cores and (b) 4 ARM11 cores.

thermal conductivity and capacitance of each cell is computed as equation 3.3, (where $k_{th}^{si/cu}$ is the thermal conductivity and $c_{th}^{si/cu}$ is the thermal capacitance per unit volume).

We model the generated heat by adding an equivalent current source to the cells on the bottom surface. The heat injected by the current source into the cell corresponds to the power density of the architectural component covering the cell (e.g., a memory decoder or processor) multiplied with the surface area of the cell. No heat is transferred down into the package from these bottom cells.

In contrast, the heat from the cells on the top surface is removed through convection. We model this by connecting an extra resistance in series where $R_{th}^{top} = 1/G_{th}^{top}$ resistance. The value of this resistance is equal to the package-to-air resistance weighted with the relative area of the cell to the area of the spreader.

Finally, the thermal model was calibrated against a 3D-finite element analysis given by an industrial partner.

Thermal properties

In Table 4.2 we enumerate the thermal properties of the different packaging options used during our experiments. The amount of heat that can be removed by natural convection strongly depends on the environment, such as the placement of the chip on the PCB, as in the case of embedded systems. Regarding package-to-air resistance, we consider the case of very low-cost packaging, where a good average value is 42W/K (see [72]), because of the uncertainty of final MPSoC working conditions. However, since this value is higher than the actual figures published by some package vendors, in our experiments we also study the effect of different packaging solutions for MPSoCs.

Table 4.2: Thermal properties.

silicon thermal conductivity	$150 \cdot \left(\frac{300}{T}\right)^{4/3} W/mK$
silicon specific heat	$1.628e-12J/um^3K$
silicon thickness	$350um$
copper thermal conductivity	$400W/mK$
copper specific heat	$3.55e-12J/um^3K$
copper thickness	$1000um$
package-to-air conductivity (low-cost)	$40K/W$

4.7 HW-SW MPSoC Emulation Flows

The key advantage of our framework for a realistic exploration of MPSoC designs with thermal management at high speed is its double integration of statistics extraction from HW emulation and SW thermal simulation of all MPSoC architectural blocks in one overall tool flow. The whole system flow used is depicted in Figure 4.4.

First of all, the HW and SW components of the system are defined. Regarding HW, the user specifies in this phase one concrete architecture and all the HW sniffers that need to be included in the system to extract statistics for each of the three main architectural levels that constitute the final MPSoC: processing cores, memory subsystem and interconnection to the main memories. This is done by instantiating, in a plug-and-play fashion, the predefined HDL modules available in our repository for each of the previous three levels and the respective sniffers. In our case we use Xilinx Integrated Software Environment. Related to the SW part, in this phase it is compiled the application/s to be tested in the emulated MPSoC. In this case, we use Xilinx EDK, which includes GNU C (`gcc`) and C++ (`g++`) compilers/linkers for the Power PC and Microblaze cores available in our repository. Also, EDK enables loading different binaries on each processor of the system. Thus, if the application to be tested is already written in any of these languages, no effort is required for the designer since the memory hierarchy and the utilization of the interconnection mechanism (e.g., generation of OCP transaction for the NIs of the NoC) are transparently generated by the underlying emulated HW architecture. For a complex MPSoC with 8 processors and 20 additional HW modules, this phase requires 10 to 12 hours overall, including the complete synthesis phase with standard tools. Moreover, modifications in the current configurations of the cores take less than 1 hour to be re-synthesized, while the compilation of additional SW part of a 4-processor emulation system only takes minutes.

In the next phase, the floorplan to be evaluated according to the previous HW definition is defined. At this moment the different energy and frequency values for each HW component in the emulated MPSoC is set. Also, the con-

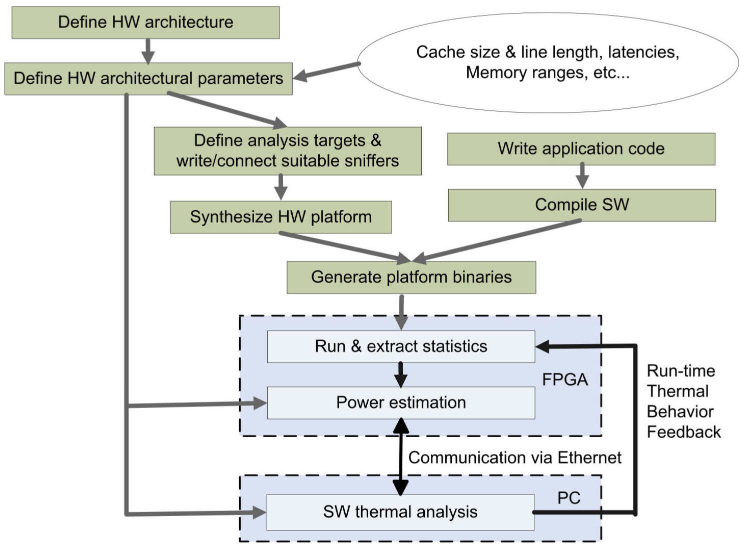

Define HW architecture
Define HW architectural parameters
Cache size & line length, latencies, Memory ranges, etc...
Define analysis targets & write/connect suitable sniffers
Write application code
Compile SW
Synthesize HW platform
Generate platform binaries
Run & extract statistics
FPGA
Power estimation
Run-time Thermal Behavior Feedback
Communication via Ethernet
PC
SW thermal analysis

Figure 4.4: Complete HW-SW flows included in the FPGA-based thermal emulation framework.

figurable granularity of the temperature updates and communication between the FPGA and the SW thermal library is configured at this moment. In particular, this value is fixed at $10ms$ in our experiments.

Next, the whole HW emulated MPSoC is uploaded onto the Xilinx FPGA-based platform using a JTAG device and the graphical interface of our SW thermal model is launched in the host computer. After this point our framework runs autonomously. While the emulated system is running, the statistics about the power values for each cell defined in the layout are concurrently extracted, and sent to the thermal simulator running onto the host computer via a standard ethernet connection. The thermal simulator calculates in real-time the new temperatures and feeds back the updated temperatures by sending MAC packets to the FPGA-based emulation framework. According to this new received information, the implemented temperature manager in our FPGA can be used to test different run-time thermal management policies on the emulated MPSoC.

4.8 Experimental Results

We have assessed the performance and flexibility of the proposed emulation framework in comparison with the MPARM framework [43] and its internal SW thermal library by running several examples of multimedia and intensive processing cores of MPSoC designs. Additionally, our experiments include the application of the presented framework to test a run-time DFS mechanism for one complex MPSoC case study based on ARM-11 cores, and with different thermal-aware floorplan floorplan solutions and various packaging techniques. In our experiments MPARM is executed on a P-IV at 3.0 GHz with 1 GByte SDRAM and running GNU/Linux 2.6.

4.8.1 MPSoC emulation vs simulation performance evaluation

In the first set of experiments we have assessed the performance of the bare MPSoC emulation framework (without thermal modeling) for system architecture exploration, in comparison to cycle-accurate simulators. To this end, we evaluated various configurations of interconnections and processors (1 to 8) using a complex L1 hierarchy for each core with 4 KB D-cache/I-cache, 16 KB of private memory, and a global 1-MB main shared memory. All processors used OPB and OCP buses. As an example, the MPSoC design with HW sniffers and 4 processors (1 hard-core PowerPC and 3 soft-core Microblazes) consumes 66% of the V2VP30 and runs at 100 MHz. Next, we have explored the use of NoCs [55] instead of buses. The tested NoC had 2 32-bit switches with 6 inputs/outputs and 3-package buffers. This NoC-based MPSoC required 80% of our FPGA.

As SW drivers, first, we used a kernel application (MATRIX in Table 4.3) that performs independent matrix multiplications at each processor private memory and combines the results in memory at the end. Second, we have used a dithering filter (DITHERING in Table 4.3) using the Floyd algorithm [64] in two 128x128 grey images, divided in 4 segments and stored in shared memories. This application is highly parallel and imposes almost the same workload in each processor. The obtained timing results are depicted in Table 4.3.

These results show that the HW-SW emulation framework scales significantly better than SW simulation. In fact, the exploration of MPSoC solutions with 8 cores for the Matrix driver took 1.2 seconds per run in our case, but more than 13 minutes in MPARM (at 125 KHz), resulting in a speed-up of 664×. Moreover, the exploration of NoCs with complex SW drivers (Dithering with 4 cores, 30 HW MPSoC components in total) shows larger speed-ups (1147×) due to signal management overhead in cycle-accurate simulators (Table 4.3). As a result, our HW-SW emulation framework achieved an overall speed-up of more than three orders of magnitude (1147×), illustrating its clear benefits for the exploration of the design space of complex MPSoC architectures compared to cycle-accurate simulators.

Table 4.3: Timing comparisons between our MPSoC emulation framework and MPARM.

	MPARM	HW Emulator
Matrix (one core)	106 sec	1.2 sec (88×)
Matrix (4 cores)	5 min 23 sec	1.2 sec (269×)
Matrix (8 cores)	13 min 17 sec	1.2 sec (664×)
Dithering (4 cores-bus)	2 min 35 sec	0.18 sec (861×)
Dithering (4 cores-NoC)	3 min 15 sec	0.17 sec (1147×)
Matrix-TM (4 cores-NoC)	2 days	5′ 02 sec (1612×)

4.8.2 MPSoC thermal modeling using cycle-accurate simulation vs HW-SW emulation

In the second set of experiments we have verified the capabilities of real-time interaction between the HW FPGA-based emulation and the SW thermal library components of our system, compared to pure cycle-accurate SW simulation. In this case we considered a low-cost package solution (see Table 4.2). From the HW viewpoint, we have defined a system with 4 RISC-32 processing cores. Each core was attached to a local 8KB direct-mapped instruction and data caches, using a write-through replacement policy. Also, each processor had a 32KB cacheable private memory and a 32KB shared memory was included in the system. The memories and processors were connected using a XPipes NoC of 4 6x6 switches and NI modules. The considered floorplan is shown in Figure 4.3 and included 128 thermal cells. We obtained the dimensions of the NoC circuits by synthesizing and building a layout. As SW driver for this MPSoC design, we defined a benchmark (Matrix-TM in Table 4.3) that keeps the workload of the processors close to 100% all the time, pushing the MPSoC to its processing power limits to observe effects in temperature. This benchmark implements a pipeline of 100K matrix multiplications kernels based on the Matrix benchmark (see Table 4.3). Each processor executes a matrix multiplication between an input matrix and a private operand matrix, then feeds its output to the logically following processor. The platform receives a continuous flow of input matrices and produces a continuous flow of output matrices. Every core follows a fixed execution pattern: (i) copy of an input matrix from the shared memory to its private memory; (ii) multiplication of the new matrix with a matrix already stored in the private memory; (iii) copy of the resulting matrix back to the shared memory. During the whole execution, interrupt and/or semaphore slaves are queried to keep synchronization, creating an important amount of traffic to the memories. The obtained timing results (Table 4.3) show that our HW-SW emulation framework takes 5 minutes approximately for the whole execution of the driver, including thermal

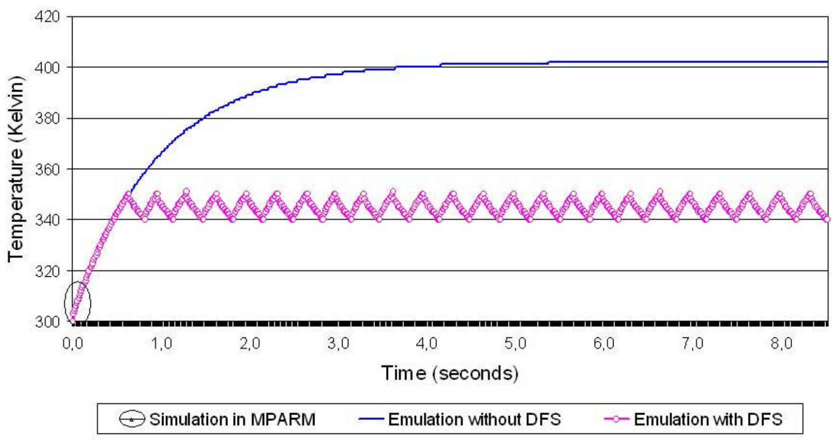


Figure 4.5: Average temperature evolution of Matrix-TM in a 4-core MPSoC at 500 MHz or using a two-choice DFS (500-100MHz).

monitoring, versus 2 days in MPARM for just 0.18 sec of real execution (left corner on Figure 4.5); Thus, our framework achieves more than three orders of magnitude of speed-ups (1612×) compared to SW-based thermal simulation, making feasible to study in a reasonable time long thermal effects.

4.8.3 Evaluation of dynamic thermal strategies in MPSoCs

In the third set of experiments we have performed a long thermal emulation in our framework to observe thermal effects on the MPSoC with real-life processing inputs of embedded applications. We ran the Matrix-TM workload for 100K iterations and the results for a 500-MHz emulation are shown in Figure 4.5. They indicate the need to perform long emulations to estimate thermal effects (note in Figure 4.5 that the previous simulation in MPARM only represents a very limited part of the overall MPSoC thermal behavior). Due to the high rise in temperature observed in the MPSoC design, we explored the possible benefits of DTM techniques within our HW-SW emulation framework. To this end, we implemented a simple threshold monitoring policy using the available HW temperature sensors in our framework. The policy consists in a simple dual-state machine that monitors at run-time if the temperature of each MPSoC component increases/decreases above/below two certain thresholds that we have defined (350 or 340 degrees Kelvin in this example). Then, the temperature sensors inform the VPCM, which performs DFS choosing between 500 or 100 MHz accordingly. The results are also shown in Figure 4.5 and indicate that this simple thermal management policy could be highly beneficial in MPSoC designs using low-cost packaging solutions (i.e., with values of package-to-air resistance of more than 40K/W). Furthermore, these results outline the potential benefits of our HW-SW emulation tool to explore the design space of complex thermal management policies in MPSoCs, compared to SW cycle-accurate simulators that suffer from important speed limits.

4.8.4 Floorplan selection exploration in MPSoCs

When an integrated system is built for a certain MPSoC, the definition of an appropriate floorplan is a very complex task for system integration designers. In fact, deciding a suitable placement of each block in the MPSoC architecture requires taking into account multiple constraints (e.g., power, energy, performance, etc) with values that are specific for each design. Recently, due to the increasing temperature in MPSoCs, thermal behavior has become another key factor to define the placement of each block of the design [48, 47]. In this set of experiments we have used our tool to evaluate two additional thermal-aware floorplans (Figure 4.6) for our initial case study with four processing cores and NoC-based interconnect working at 500 MHz (see Figure 4.3). The first alternative floorplan scatters the processing cores in the corners of the chip (Figure 4.6(a)), while in the second one all the cores are clustered together in the center of the chip (Figure 4.6(b)). We assumed the use of a low-cost packaging solution in all the cases (see Table 4.2).

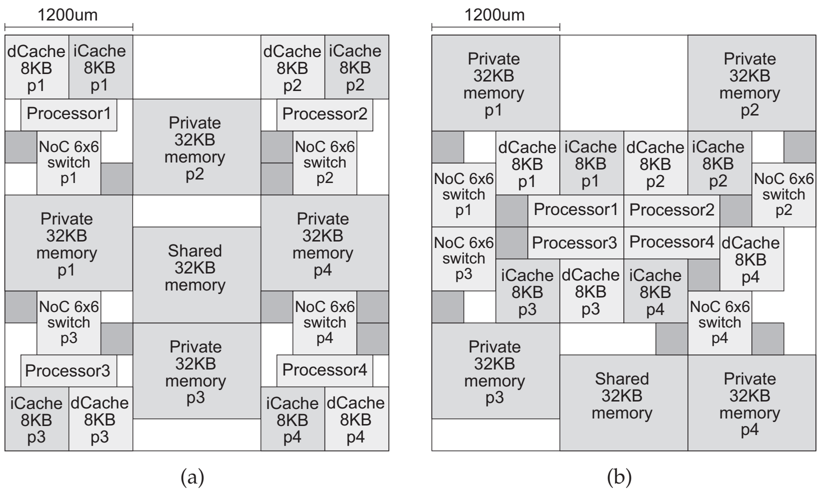


Figure 4.6: MPSoC floorplan with cores (a) scattered in the corners and (b) clustered together in the center of the chip.

The results are shown in Figure 4.7. In this case we can observe that the best floorplan to minimize temperature (15% less heating speed on average than the initial floorplan of Figure 4.3) was achieved with the placement technique that tries to assign the processing cores to the corners of the layout (labelled as `scattered` in Figure 4.7). Hence, this solution is the best out of the three thermal-aware placement options because it delays the most the need to apply

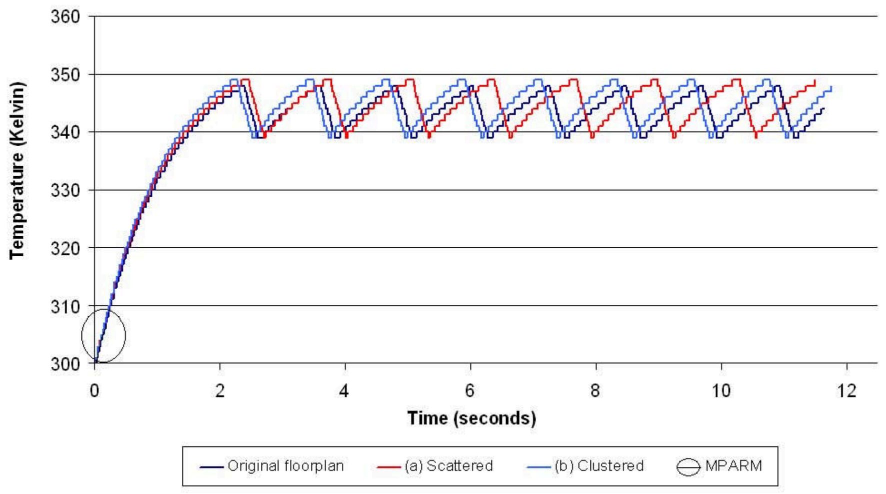


Figure 4.7: Average temperature evolution with different floorplans for Matrix-TM at 500 MHz with DFS on.

the available DFS mechanism in Figure 4.7, although its interconnects experience more heating effects due to the longer and more conflicting connection paths between components, which can originate more NoC congestion effects. Then, the solution that tries to place all the processing cores in the center of the chip (labelled as `clustered` in Figure 4.7) shows the worst thermal behavior, but just slightly worst in temperature (5% on average) than the original manual placement of cores used for this MPSoC design, while the delays in the interconnections between cores are minimal for the former due to their closest locations in the floorplan (see Figure 4.6(b)). The main conclusion from this study is that a more aggresive temperature-aware placement must be applied (e.g., placement of cores scattered in the corners of the chip) to justify the placement of cores apart, as tried in the original manual design, to compensate for the heating effects on the chip due to longer interconnects. Otherwise, the possible penalty for long interconnects may not be justified in the end since a uniform distribution of power sources does not need to lead to a uniform temperature in the die. Moreover, these results clearly outline the importance for designers of tools to explore the concrete thermal behavior of each design, and to select the most appropriate placement in an early stage of the integration flow.

4.8.5 Effect of different packaging technologies and SW thermal libraries

In this final set of experiments we have tested different packaging solutions and compared them with the thermal behavior of the low-cost value of 40K/W initially considered (Table 4.2) for our initial reference of MPSoC floorplan

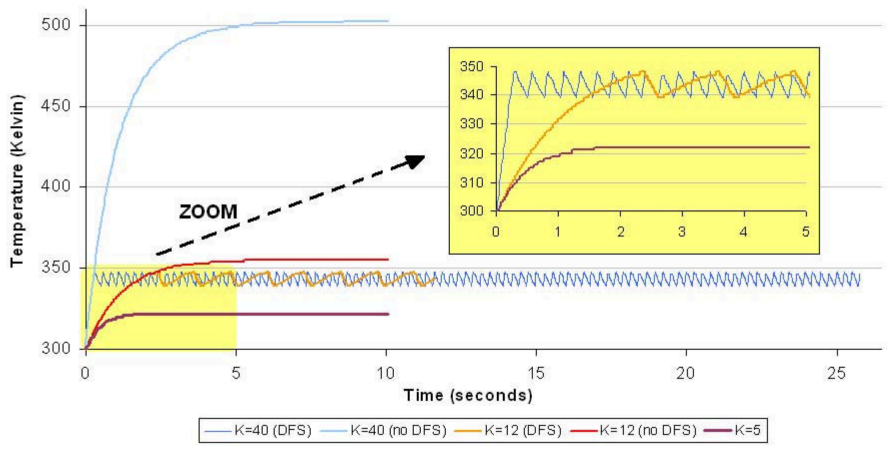


Figure 4.8: Thermal behavior for an MPSoC floorplan using low-cost, standard and high-cost packaging solutions.

with four RISC-32 processing cores working at 500 MHz and NoC interconnect (Figure 4.3). We simulated this floorplan with two additional values, namely, 12K/W in the case of standard packaging [41] and 5K/W in the case of high-cost and high-performance embedded processors [37]. The results obtained are shown in Figure 4.8.

As this figure shows, in the case of the standard packaging solution, the MPSoC design required more time to heat up and it reached a maximum value of 360 degrees Kelvin when the DFS mechanism was not applied, which is lower than the case of low-cost packaging (40K/W) that reached a temperature of more than 500 degrees Kelvin. However, the thermal behavior of the standard packaging system was similar to the low-cost solution (only its starting point was slightly shifted to the right due to the less steep temperature rise curve) when the presented threshold-based DTM strategy, fixed at 250 degrees Kelvin, was applied. Therefore, in this case, with this threshold value, no significant improvements were obtained with the standard package, and the low-cost solution would be preferably selected for this design using DTM. However, in the case of the high-cost packaging solution (for 5K/W), the system showed a completely different temperature behavior, where the chip never went beyond 325 degrees Kelvin. Therefore, this packaging solution creates a much lower thermal stress in the overall MPSoC implementation, and it does not require the application of DFS because the design never reaches a temperature above the 350-degree-Kelvin threshold. As a result, this solution could significantly increase the expected mean-time-to-failure of the component and be interesting in highly reliable versions of this MPSoC chip design. However, note that this type of package has the important drawback of the high cost for the manufacturer of the final embedded system, namely, typically 5 to 12× more than standard package solutions and more than 20× the low-cost package solution [54]; Thus, it can seriously increase the price of the final product and developers would like to avoid it if possible. Hence, this type of experiments and the presented framework can be a very powerful tool for designers to decide which type of packaging technique would be enough for a specific set of constraints in forthcoming generations of MPSoC designs.

Finally, we performed the same set of emulation experiments replacing our library with the well-known Hotspot v3.0 thermal library [68], configuring it with the same packaging options previously tested. The results of this additional set of experiments shown a very similar thermal behavior with this second RC thermal library in comparison to our own library in the case of high-cost packaging (less than 3 degrees Kelvin of difference), which is the original target of the Hotspot library. Then, in the case of low-cost and standard packaging, variations that range between 4-15 degrees Kelvin have been observed. The origin of these variations come from the non-linear dependency factor of silicon thermal conductivity with respect to the actual temperature in the die, which is included in our own library, but is not modeled in the Hotspot library. In fact, our results indicate that this non-linear part of the thermal equations is particularly important when the temperature rises beyond 360 degrees Kelvin in the case of low-cost packaging solutions, and needs to be considered at each moment of the emulation to get accurate thermal measurements for this type of MPSoC packaging technology.

4.9 Conclusions

MPSoC architectures have been proposed as a promising solution to tackle the complexity of forthcoming embedded systems. These future consumer devices will contain a really large amount of transistors thanks to nanoscale technologies, but will be very complex to design as they must execute multiple complex real-time applications (e.g., video processing or 3D games), while meeting several additional design constraints (e.g., energy consumption or short time-to-market). Moreover, the rise of temperature in the die for on-chip components can seriously affect performance and reliability of final MPSoC designs. In this chapter we have presented a new HW-SW emulation framework that provides designers with a powerful tool to study the thermal behavior of MPSoC designs at three different architectural levels, namely, processing cores, memory subsystem and interconnection mechanisms. The experimental results have shown that our proposed framework obtains detailed reports of the thermal features of final MPSoC floorplans, with speed-ups of three orders of magnitude compared to cycle-accurate MPSoC simulators. Also, the addition of more processing cores and more complex memory architectures in our emulation framework suitably scales. Thus, almost no loss in emulation speed occurs, conversely to cycle-accurate simulators, which enables long simulations of complex MPSoCs as thermal modeling requires. Then, the real-time interaction between HW emulation and SW thermal modeling through the Ethernet connection enables the application and testing of complex dynamic thermal management policies to the emulated MPSoC at run-time.

In addition, we have used our tool to evaluate different temperature-aware placement techniques that try to compensate the heating effects on MPSoCs. Our study indicates that significant overheads of power dissipated in long interconnects can clearly affect the overall thermal behavior of the final MPSoC, and that a uniform distribution of power sources in the die does not need to produce a uniform temperature in the final chip. Hence, MPSoCs designed in latest technology nodes require the use of tools to study their suitable placement in an early stage of system integration, according to the applications that will be executed in each final MPSoC. Also, we have illustrated the effectiveness of the presented thermal evaluation tool to rapidly study the effects of different packaging options for concrete MPSoC solutions. Our results indicate that the selection of final packaging solutions clearly depend on the thermal management techniques included in the target MPSoCs and more costly packagings may show from the same heating effects as low-cost ones; Thus, the need of expensive packaging solutions cannot be justified without prior extensive thermal exploration. Finally, we have shown the versatility of our tool to use various thermal libraries, and illustrated the need for different thermal models according to the implementation requirements of the target MPSoCs (e.g., high- or low-cost packaging).

Chapter 5

Exploit the performance benefits of self-timed logic in synchronous design

5.1 Overview

Ultra low power digital systems are key for any future wireless sensor nodes but also inside nomadic embedded systems (such as inside the digital front end of software defined radios). These systems require the highest possible energy efficiency of logic, which can only be achieved by operating in moderate inversion. Unfortunately, when operating near the threshold voltage, transistors become highly sensitive to process variations, thereby increasing leakage currents and complicating timing closure. Rather than pursuing a worst-case design approach for dealing with these uncertainties, we present a hybrid self-timed/synchronous approach. It will be demonstrated on the VEX VLIW core designed for ultra low-power operations. Experimental results of our approach demonstrate performance benefits up to 2x and significant energy savings at low throughput rates.

5.2 Introduction

Ultra low power (ULP) digital systems are a generic technology useful for wireless sensor nodes but also inside nomadic embedded systems (such as inside the digital front end of software defined radios). A key requirement for these systems is that they provide the highest achievable computational performance (1-10MOPS) consuming at most 1mW on average [75]. While targeting low to medium performance targets, the highest energy efficiency is achieved by operating in moderate inversion. Unfortunately, at these low operating voltages (near the threshold voltage), transistor switching speed and leakage power are extremely sensitive to process variations, thereby complicating timing closure of synchronous designs. Design margins at all levels of abstraction increase rapidly, causing a delay and power penalties. In this chapter we compare two alternative solutions for dealing with these extreme variations: the classic synchronous design based on corner point analysis and self-timed logic based on dual rail logic. Self-timed logic remains functionally correct under delay variations. Therefore, no additional design margins are needed to cope with process variations. In contrast to synchronous designs, also no need exists to over-design for input data variations. As a result, self-timed logic operates on average many times faster compared to synchronous logic. The introduction of dual rail logic is however far from trivial: (1) usually, synchronization between stages is performed using handshake protocol, but this is not compatible with commercial testing solutions; (2) the overhead of dual rail logic is high in terms of area and power and (3) dual rail logic requires dedicated logic libraries. In this chapter, we therefore propose an alternative technique to convert a synchronous design into a self-timed one. Rather than replacing the clock network with a handshake protocol, we retain the clock, but we apply clock-gating until logic has completed. The completion of logic is detected using dual rail logic. To limit circuit overhead this logic is inserted only in the most critical parts/stages of the design. In this way, we exploit the performance benefits of self-timed logic and remain compatible with existing design solutions for synchronous logic. This hybrid system can be clocked at high frequencies without generating functional errors. In this chapter, we delimit how the throughput increases with the clock frequency and in this way, how to optimize the clock frequency for the average case rather than worst case operating conditions. Experimental results on the VEX VLIW core [85] will be presented to quantify this hybrid approach. To this end, we have developed a complete flow to convert a synchronous standard-cell based combinational logic stage into a self-timed one. Our results indicate that significant performance benefits (up to 2x) can be realized by removing the pessimism of worst-case design. Both energy/throughput benefits strongly depend on the selected circuit topology.

5.3 Related Work

In recent years, both industry and academia have shown a large interest for ultra-low power systems, as enabling technology for autonomous systems [73][74]. Initial performance/energy requirements for autonomous systems for different possible application domains are described in [75]. We focus in this paper on high bandwidth systems, providing the highest achievable computational performance while consuming less than 1mW on average. A good approach for achieving this performance/energy target is reducing the power supply [75] and preferably in combination with scaled/domain specific process technologies [76]. The benefits of ultra low voltage operation have been validated down to silicon-level (e.g., the FFT-processor of [78]). However, as the voltage is reduced, the sensitivity to process variations increases [77]. As a result, the system may functionally fail and/or is subjected to large parametric variations (in delay/energy), which at the end result in yield loss. In practice, this limits the maximal achievable energy savings. Rather than restricting parametric variations at design-time with design margins, our objective is to build systems that measure delay variations and adapt their performance at run-time depending on actual system requirements (similar to the approach followed for memories in [81]). This requires logic that operates correctly under large variations and should not guarantee the performance at design-time. Hence, design margins for timing closure are no longer needed. Logic with Razor-latches partially exhibits this property and can operate in a better-than-worst-case fashion [9]. Unfortunately, it can only cope with limited delay variations (up to 50% of the clock). However, in moderate inversion larger variations may occur in practice. An alternative to Razor-latches is self-timed logic, which naturally deals delay-variations [82]. Under normal operating conditions (high voltage and limited variability), self-timed logic has an area, power, performance offset compared to a typical synchronous design. However, in the domain of ULP processing the balance may change in favor of self-timed logic as the amount of variations and the sensitivity to them is much higher. For instance, a better-than-worst-case DLX processor designed in asynchronous logic using matched delay-lines is presented [83]. Despite the fact that this approach is robust to design margins for systematic and environmental uncertainties, it cannot cope with random variations (which are said to be dominant below 90nm). In attempt to deal both with systematic and random variations in a single shot, we study the potential benefits of dual rail encoding logic in this paper. As this logic progresses based on the delay of the logic itself, it may eliminate these variations. In the next sections, we explain how we use dual rail logic for dealing with process variations and quantify its performance benefits.

5.4 Delay variation resilient

As indicated in the introduction, performance targets between 1-10MOPS can be achieved while operating in moderate inversion. While operating just above

True Wire	False Wire	State	Logic Value
gnd	*gnd*	Neutral	-
gnd	*vdd*	Valid	False
vdd	*gnd*	Valid	True
vdd	*vdd*	Not Allowed	-

Table 5.1: Dual Rail Encoding.

the threshold voltage, systems become very sensitive to process variations, thereby complicating timing closure. Rather than over-designing the system, we explore (partially) self-timed circuits that can naturally live with variations. These will operate on average faster compared to synchronous logic. Self-timed circuits consist of two parts: (1) a mechanism to detect the completion of the combinational logic; (2) a mechanism to synchronize sequential register stages of the logic. Both completion detection and synchronization circuits for self-timed logic are rather complex logic designs. Careful introduction of these components into the system is therefore mandatory. Consequently, to limit overhead, it should be feasible to build a system where *only the most critical components for variations have been made delay-variation resilient and where the performance of the entire system is determined by the completion time of these critical circuits*.

5.4.1 Completion detection using Dual Rail Logic

Several ways exist to build completion detection logic (see [82]). In the context of this paper, we have used dual rail logic, which detects the operation completion by coding the validity of the output signals. At the start of a new computation cycle, all signals are in the neutral state. During the computation the signals transit into either a valid 1/0 state. When all signals move into a valid state, this means that the computation has completed. To code the valid/invalid, dual rail logic uses two wires to represent each binary signal (see Table 5.1). We implemented dual rail logic with DCVSL (Dynamic Cascode Voltage Switch Logic). The main advantage of this logic style is its speed. A NAND2 gate implemented in DCVSL logic is shown in Figure 5.1. DCVSL is weakly indicating, i.e. it starts to compute valid outputs even when not all input signals have become valid. E.g., as soon as either A or B equal zero, the left pull down circuit will discharge the pre-charged left-node, and in this way drive the output of the NAND2 gate to a valid true signal. This makes the circuit faster compared to alternative strongly indicating approaches where all input data have to be valid before computing the output (e.g. [84][12]). Moreover, the pre-charge signal can reset the logic into the invalid state much faster compared to strongly indicating logic. In the latter, all paths have to become

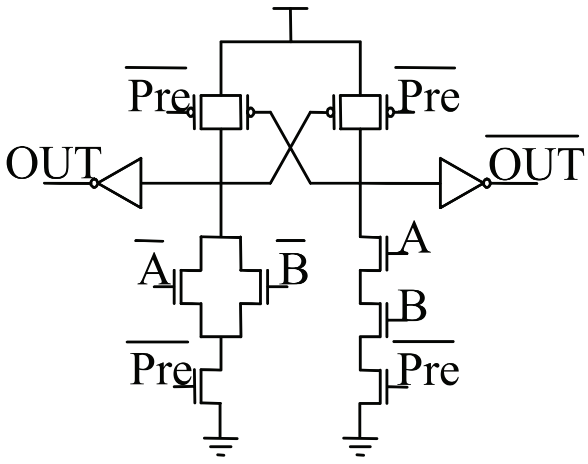


Figure 5.1: DCVSL NAND.

valid, before the logic can be invalidated again. The dual rail encoding is more complex compared to static CMOS (e.g., all wires are duplicated). Therefore, its power overhead has to be carefully analyzed.

5.5 4-way handshaking protocol vs. clock gating

We have explored two approaches to synchronize sequential logic. First, we have used a 4-way handshaking protocol [83], with a slight modification to work with the completion signal of the logic and for driving the pre-charge signal (see Figure 2.9). In our view, the introduction of handshake circuits to replace synchronous logic is involved. There are lots of additional timing issues at the physical design level. Furthermore, handshaking complicates testing significantly. Finally, with handshake circuits, it is difficult to partly introduce delay-variation resilient circuits in the most critical parts of the system. Therefore, as alternative, we examine an approach to replace the critical parts of the system with self-timed logic (see Figure 2.10). Basically, additional logic is inserted to gate the clock. The clock gating is controlled by the completion detection circuits: as long as these critical stages have not finished their computations, the clock is not distributed (i.e. it is locally stopped). This gated-clock network can be implemented similar to [9]. In contrast to [9], we gate the clock until the computations are done rather than stalling it until correct data has been restored. As a result, the clock does not need to be operated any longer at the worst-case delay. It may be operated faster without causing functional errors. As the logic usually completes faster than the worst-case, the throughput of the system can be significantly improved. The average throughput of the system can be computed with following formula:

$$t_{avg} = t_{clk} \cdot \sum_{\forall i>0} i \cdot P\left[t_{clk} \cdot (i-1) < t_{ops} + t_{cmp} < t_{clk} \cdot i\right] \tag{5.1}$$

Where t_{avg} is the average delay, t_{ops} is the combinatorial logic delay and t_{cmp} is the completion logic delay. As an illustration, we present the throughput of a

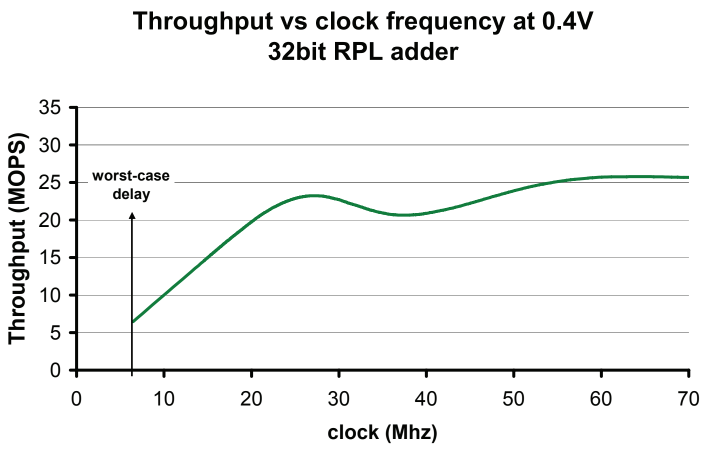


Figure 5.2: Throughput of self-timed logic (with clock gating) depends on clock speed.

ripple adder in function of the clock frequency in Figure 5.2. The arrow represents the synchronous reference adder. It can operate maximally at 6.5Mhz and thus achieve 6.5MOPS. The clock-gated design can be operated faster. Given the fact that in a ripple adder operations for most input patterns complete much faster than the worst-case one, the throughput can be increased almost linearly by speeding up the clock. The benefits saturate at 27Mhz, where a three and an half times higher throughput than for the synchronous one is achieved. Thereafter, the number of errors increases rapidly, explaining the throughput leveling off. It even slightly decreases, because the clock will trigger the registers only on next cycle after the logic completes. Finally, when increasing the frequency to extremely high values, the performance reaches the performance limit set by an ideal self-timed design. However, in practice, a maximal operating frequency exists in the proposed hybrid self-timed/synchronous approach: the clock frequency can only be increased until the critical path determined by the synchronous part of the design. Increasing clock frequency further would render these synchronous parts of the design functionally incorrect.

NAND2	3.5
EXOR	2.2

Table 5.2: Area Penalty of DCVSL.

5.6 Evaluation Framework

5.6.1 Library Design

As baseline for all our experiments, we use both synchronous and self-timed cell-library containing 27 gates: NAND2, NAND3, NOR2, NOR3, EXOR, MULTIPLEXER, INVERTER, BUFFER and DFLIPFLOP. The INVERTER and BUFFER are designed in different versions: minimal size and with increased drive strength 1, 2, 4, 9. The cells are designed in IMEC's 130nm CMOS technology down to layout level. Particularly, the GDSII was generated with Synopsys Cadabra. Based on the layouts of the cells, we have generated the Synopsy target library (.lib) and physical library (.plib).

5.6.2 Synthesis

We create a synchronous gate-level netlist from a RTL description of the design using Synopsys Physical Compiler. The RTL design is mapped onto the gates of the synchronous library characterized with the nominal process conditions. Constraints were set to optimize power consumption. The self-timed design was built using a script that translates synchronous design into a self-timed one. This entails three steps (see Figure 5.3):

From synchronous to asynchronous gates: we replace every synchronous logic gate with its asynchronous counterpart having a similar input capacitance and drive strength. As the relative strengths/loads between the cells remain similar to the synchronous design, the technology mapping decisions of the synthesis tool can be preserved. Note that better ways exist to synthesize with dual rail gates, exploiting for instance the dual output during logic mapping. However, commercial synthesis tools do not support these optimizations, hence our results will be conservative with respect to an optimized self-timed synthesis flow, but realistic in the context of current commercial toolflows.

Adding signals: we interconnect all extra ports of the dual rail logic, i.e. the pre-charge signal and the signals' comple-ments. No extra flipflops are added to store signal complements.

Adding the synchronization logic: we add completion detection circuit and the synchronization logic (handshake circuit or the clock-gating logic).

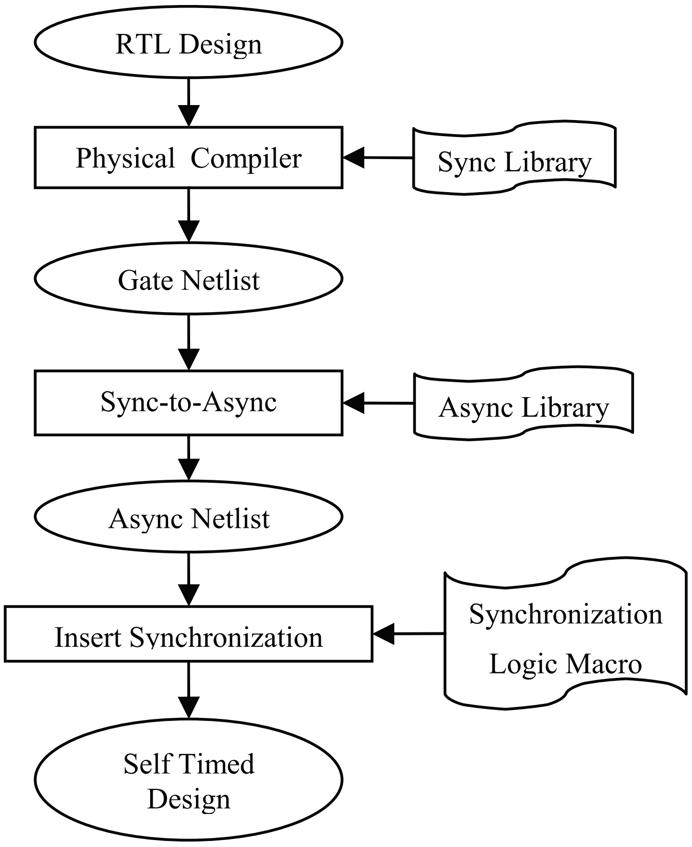


Figure 5.3: Synthesis of self-timed design.

The completion detection circuit consists of a balanced tree of NOR2 gates implemented in static CMOS logic.

5.6.3 Evaluation under process variations

We evaluate the both synchronous and asynchronous design under process variations. For this purpose, we characterize the standard cells after subjecting the cells' netlists to random β $(\sigma_\beta = 10\%)$ and V_t $(\sigma_{V_t} = 5\%)$ variations (see Figure 5.4). The results of this characterization are converted into a standard cell library. Besides storing the nominal delay/power of each cell, we also store the characterization results of each cell when subjected to random process variations. Then, at the gate-level, we analyze the impact of the process variations. For this purpose, we randomly replace the default cells with these variants. By generating several possible instances, statistics on the design's delay/power distribution are gathered.

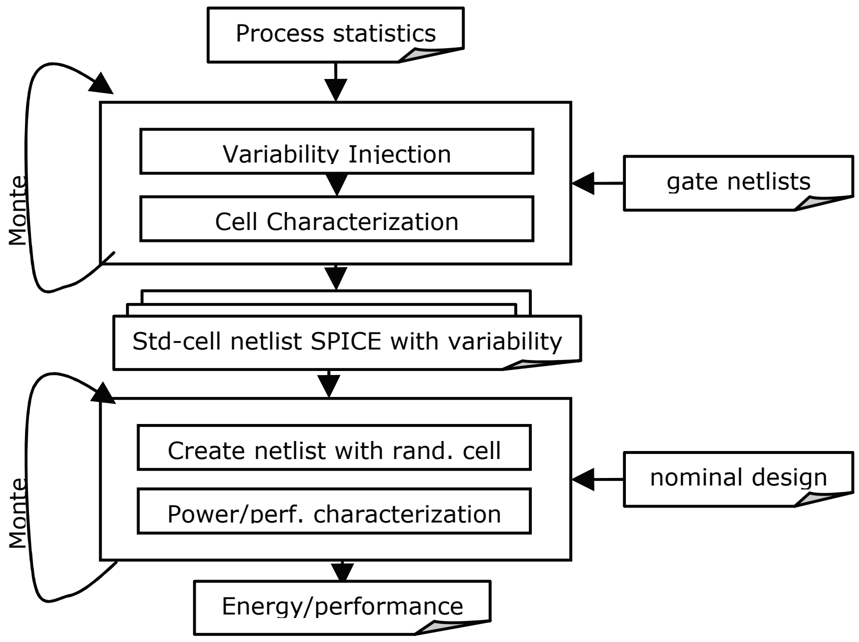


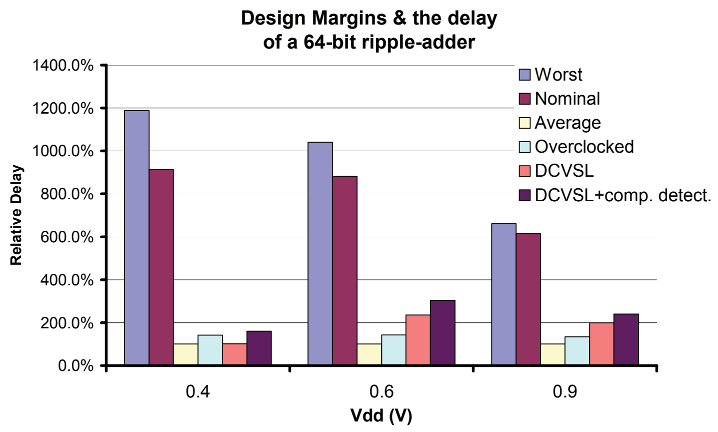
Figure 5.4: Performance characterization under process variation.

5.7 Experimental Results

In this section, we compare the performance/energy of synchronous vs. self-timed for both a ripple and a Brent-Kung adder based pipeline. In Figure 5.5, we indicate how the relative delay for of a 64-bit ripple (rpl) and Brent-kung (bk) adder. In a synchronous design, the system's throughput is defined by the clock frequency, set by the critical path for the worst-case corner. This results in large design margins, and thus performance loss compared to the average delay of the combinatorial logic, particularly when operating at low operating voltages. E.g., the delay difference between the average performance and the worst-case one for a ripple-adder is 12x at 0.4V. A self-timed logic style that removes both design margins for process and input variations, thus may significantly improve the performance.

5.7.1 Handshaking vs. clock-gating

As explained before, we compare synchronization with either handshaking circuits or using a solution based on clock-gating. Handshaking achieves the average performance of Figure 5.5. Due to the discrete nature of the clock signal, the throughput of a clock-gating based solution is up to 40% slower, but still outperforms synchronous logic by 8x (at 0.6V) for the ripple-adder and greatly simplifies testing. The operating frequency of the clock-gating based approach



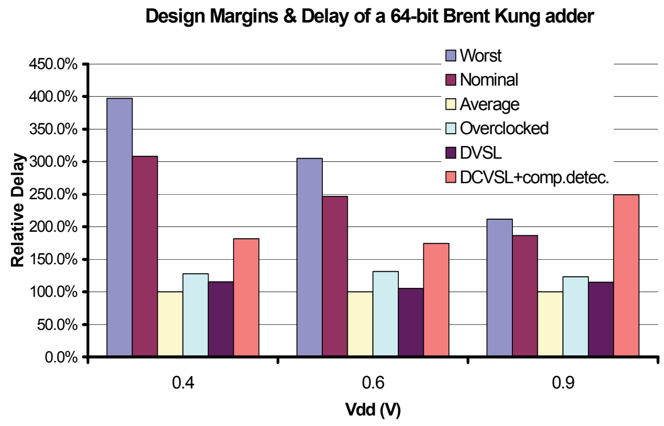


Figure 5.5: Exploiting variations - PUT.

has been optimized to achieve the maximum throughput.

5.7.2 From synchronous to dual rail logic

As can be seen in Figure 5.5 the DCVSL logic is in most cases considerably faster than the worst-case synchronous design. Lowering the voltage improves the relative performance of DCVSL (including completion detection logic) compared to the synchronous logic. At lower voltages, synchronous logic performs worse due to increasing short circuit currents and a poor Pmos/Nmos current ratio. The relative overhead of the Completion Detection Circuits (CDC) depends on the topology and operating voltages. As the logic depth of the adder becomes smaller (i.e. when going from a ripple adder to a Brent Kung one), the relative overhead of the CDCs significantly increases. This is why at 0.9V the synchronous BK adder, designed for the worst-case outperforms the dual rail one. From this experiment we thus conclude that significant performance gains can be achieved while using self-timed designs, particularly when operating at low supplies. Comparing energy consumption in Figure 7, we indicate

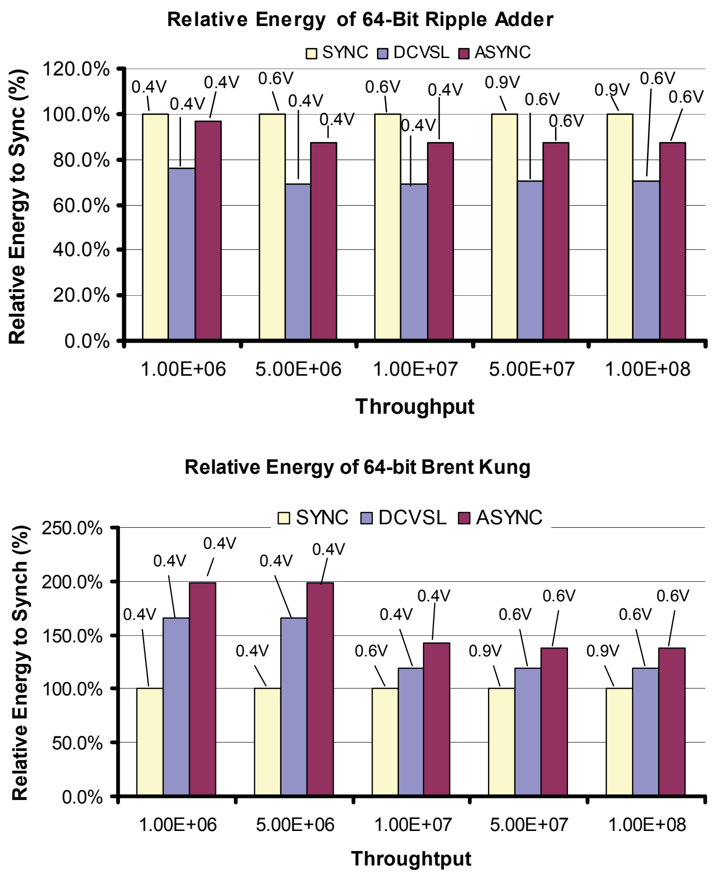


Figure 5.6: Relative Energy Using Asynchronous Logic Based on DCVSL (MOPS).

how this performance improvement may be converted into energy savings. We plot the energy consumption of both adders designed in static CMOS, DCVSL logic and DCVSL with completion detection circuits. The operating voltage has been optimized to achieve the desired performance target (see Figure 5.6). The benefits depend on the selected topology. In case of the ripple-adder, the DCVSL logic improves the energy efficiency between 24%-30% compared to static CMOS, despite its dynamic nature and high area overhead. At 10MOPS, its 30% energy efficiency is the result of a lower operating voltage (0.4V rather than 0.6 V) and less short circuit currents at these low power supplies. The total energy consumption of the asynchronous adder including CDC improves between 4% - 13%. The difference in energy consumption is caused by the CDC, of which the energy efficiency decreases at lower operating supplies In case of the Brent-Kung adder, self-timed logic is more energy-hungry than the synchronous ones due to complexity of dual rail logic. Therefore, while exploiting self-timed logic to improve the performance of an entire pipeline, designers should carefully analyze how/where to introduce self-timed logic. Replacing all synchronous logic with dual rail one, will result in the highest achievable performance but may come at a energy penalty. A selective introduction is therefore desirable to limit the overhead of the dual rail logic.

5.7.3 A hybrid self-timed/synchronous approach demonstrated on the VEX VLIW

In this section, we illustrate the energy vs. throughput benefits of a more selective introduction of dual rail logic on a VLIW core. The VEX VLIW [85] consists of 4 pipeline stages (fetch, decode, execute and write back). The VLIW has been synthesized for energy with a 1.2V library, and targets a frequency of 132 MHz. Initially, its critical path (7.53ns) runs through the ALU (EX-stage), containing a 64-bit ripple adder (ALU-delay=6ns, bypass=0.84ns, others=0.68ns).To improve the average throughput of the system, we replace the ALU with self-timed logic and over clock the system. Other parts of the system are kept synchronous. The maximum clock frequency of the resulting system is determined by the longest path in the synchronous parts of the system, i.e. any other stage than the ALU. In this case, this corresponds to the decode stage which has a delay of 3.75ns. Hence, in this case the system can be over-clocked by a factor two. While operating at the double frequency only very few operations of the ALU take more than one cycle. Rather than designing for extremely rate worst-case input patterns and process variations, an average-case design running at the double frequency increases throughput by a factor 1.992. Next, we investigate the energy benefits of the average-case design in more detail. In Figure 5.8, we plot the [throughput-energy consumption] of the VEX core implemented for the worst-case (VEX_WC_RPL) and average case (VEX_AVG_RPL) running at different operating supplies. Operating at the same voltage, the VEX core designed for the average case runs almost two times faster. This can be seen in Figure 5.8 where VEX_AVG_RPL design is shifted to the right compared to VEX_WC_RPL. At higher voltages, the asynchronous execute stage is 1.56 times more energy-hungry compared to synchronous one (see also previous paragraph). However, the energy penalty per operation for the entire system is only 21%. At lower voltages, asynchronous circuits become more efficient compared to synchronous ones. There replacing the execute stage makes the design both more energy efficient and faster. Finally, we have replaced the ripple-adder in the VEX ALU design with a faster carry-look ahead (CLA) one. As a result, the delay of the ALU reduces to 5.7ns, but remains the critical path in the design. However, the delay-ratio between this path and worst-case path in the synchronous parts of the design reduces to 1.5. Hence, the throughput difference between the worst-case and average case design is lower compared

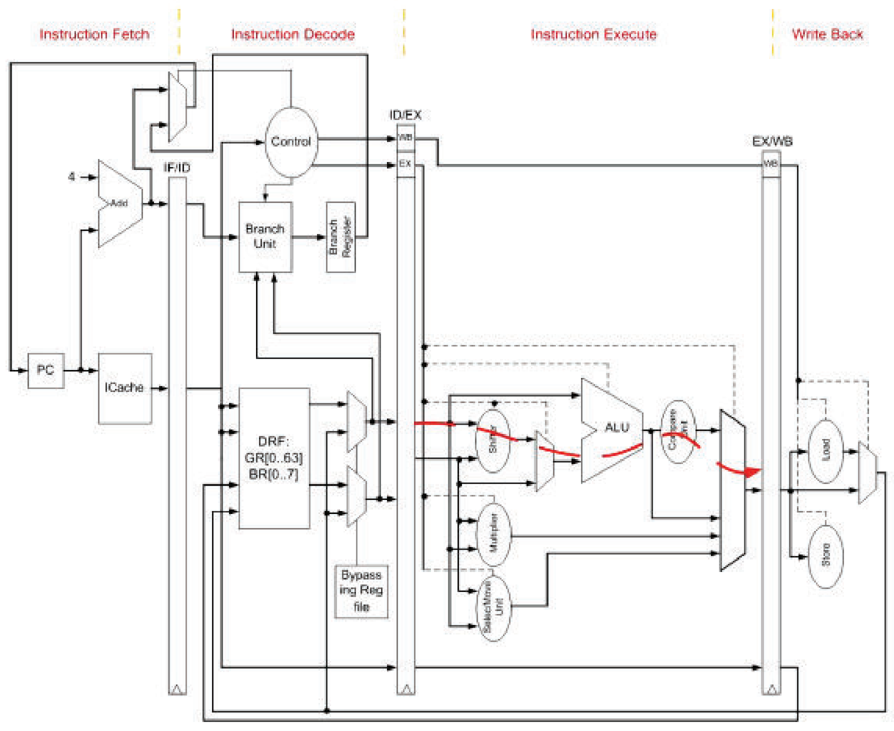


Figure 5.7: VEX VLIW: We have converted the execute stage into a self-timed one to increase the throughput by taking advantage of process and data-dependent input variations.

to the ripple-adder design. Energy trends are similar. From above, we conclude that a selective introduction of self-timed logic in a VLIW may improve energy efficiency for a targeted throughput by removing pessimism on data and process variations. The results are strongly design-dependent though.

5.8 Conclusions

The highest energy efficiency for ultra low power applications requiring a performance between 1-10MOPS is achieved while operating in moderate/weak inversion. In this operating region, design margins for timing closure due to process and as input variations considerably slow down the throughput. Self-timed logic, such as dual rail logic, can eliminate these design margins. However, self-timed logic implies a large area overhead and may slightly increase the power consumption. Therefore, it is important to only replace those critical parts of the system which are most sensitive to variations. To allow the partial introduction of self-timed logic in an existing synchronous system, we have proposed a synchronization solution based on clock-gating. This solution performs slightly worse than a handshake-based synchronization, but still out-

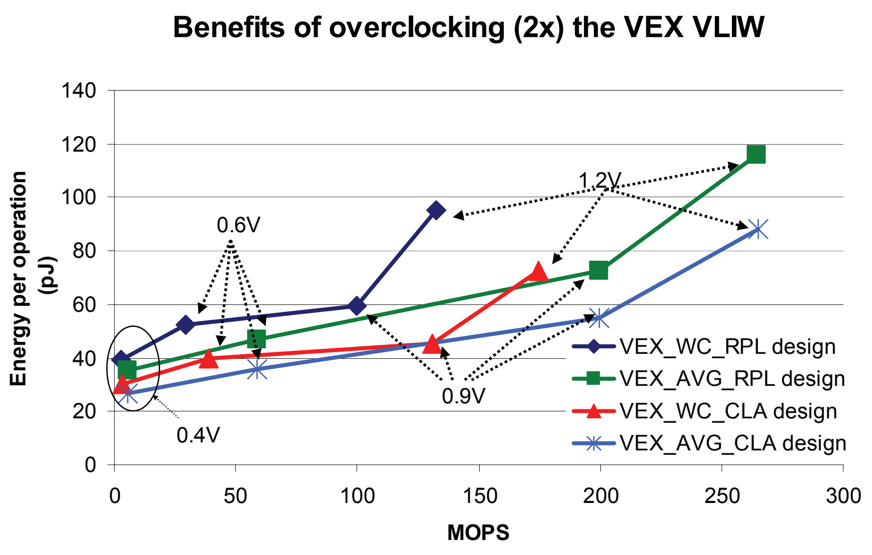


Figure 5.8: Better throughput/energy consumption for a partly asynchronous design of the VEX VLIW. VEX_WC_X design is a typical synchronous design instance. Its operating frequency is determined by worst-case conditions. VEX_AVG_X is an instance of the VEX designed for the average-case. To limit circuit overhead, only the ALU has been made self-timed logic rather than converting the entire design into an asynchronous one. The design was tested for two versions of the ALU: one where the ALU consists of a 64-bit ripple adder and one where it contains a 64-bit CLA.

performs synchronous logic. We have demonstrated this approach on a VLIW core. The results indicate that precise performance/energy benefits strongly depend on the specific logic architecture, but removing the pessimism inherent to worst-case design in all cases improves the throughput (up to 2x) and results in more energy-efficient designs at low operating supplies.

Chapter 6

Effectiveness of ASV and ABB for Full-Swing and Low-Swing Communication Channels

6.1 Overview

Adaptive body bias (ABB) and adaptive supply voltage (ASV) have been showed to be effective methods for post-silicon tuning of circuit properties to reduce variability. While their properties have been compared on generic combinational circuits or microprocessor circuit sub-blocks, the advent of multi-core systems is bringing a new application domain forefront. Global interconnects are evolving to complex communication channels with drivers and receivers, in an attempt to mitigate the effects of reverse scaling and reduce power. The characterization of the performance spread of these links and the exploration of effective and power-aware compensation techniques for them is becoming a key design issue. This work compares the variability compensation efficiency of ABB vs ASV when put at work in two representative link architectures of today's ICs: a traditional full-swing interconnect and a low-swing signaling scheme for low-power communication. We provide guidelines for the post-silicon variability compensation of these communication channels.

6.2 Introduction

As technology continues to shrink, process variations can have a significant negative impact on yield due to the wider spread of performance and power consumption. Post-silicon tuning allows the adjustment of device characteristics after a die has been manufactured to compensate for the specific deviations that occurred on that particular die [106, 107]. One of the methods utilizes the transistor body effect to change transistor threshold voltage by applying an adaptive body bias (ABB) to chip devices to modulate performance and power [115, 106]. The other method of performing post-silicon tuning is to adjust the supply voltage (ASV) to trade performance with power, thus achieving a similar effect to ABB in spite of the different physical mechanism, implementation overhead and trade-off curves. The effectiveness of ABB and ASV in reducing variability has been assessed and compared mainly on combinational logic circuits [109], key elements of microprocessor critical paths[107] and ring oscillators[105], sometimes achieving counterintuitive and even conflicting conclusions. According to [109], the difference in effectiveness is so small that choosing one method over the other should mainly be based on implementation overhead.[105] claims that although the frequency and power tuning range of ABB is more limited than that of ASV, its frequency tuning range proves effective for process-dependent performance compensation. In contrast, [107] concludes that using ASV together with ABB is much more effective than using any of them individually and is worth the cost. In essence, the effectiveness of ASV and ABB should not be assessed in general, but with reference to the variance of a specific manufacturing process and to the performance and power tuning requirements of the design at hand. With the advent of multi-core integrated systems, the assessment of post-silicon variability compensation techniques cannot be limited to the traditional testbenches of past research any more, such as combinational logic circuits or even microprocessor circuit sub-blocks. In fact, the new architecture trend requires long (global) interconnects for the connection of system-level blocks with each other. Unfortunately, physical properties of these on-chip interconnects are not scaling well with feature sizes, and they are becoming a key limiting factor for performance, reliability and timing closure of the whole system. A common practice is to overcome the effects of interconnect reverse scaling by means of circuit-level techniques, so that on-chip interconnects cannot be viewed as simple on-chip wires any more, but rather as communication channels including complex drivers and receivers [90, 104]. Analyzing the impact of process parameter variations on the performance and reliability of these communication channels and exploring effective means for their compensation is a key design issue. The relative effectiveness of ABB and ASV may greatly depend on the particular communication circuits they are applied to. A traditional design technique for long links consists of inserting equally spaced CMOS repeaters to deal with resistive loss along the wire. However, with the increase in number and density of the wires with each new technology, interconnect area and power are severely impacted [86]. The most effective technique for global interconnects to achieve significant power savings and energy-delay efficiency is to reduce the voltage swing of the signal on the wire [100] and, possibly, to avoid the use of repeater stages, like in [95]. On the other hand, low-swing signaling reduces noise immunity and poses non-trivial circuit design challenges. In many previous works (e.g., [112]) power, area and delay of communication links making use of full-swing vs low-swing signaling have been compared. This work investigates the robustness of the two signaling techniques to process variations, and assesses the effectiveness of ASV and ABB as variability compensation techniques for full-swing and low-swing interconnects. The ultimate objective is not to characterize the implementation cost of these techniques, but to find out which technique is worth the cost for a specific communication channel and under given process parameter variations. However, aware of the need for low cost compensation, this work also investigates the effectiveness of selective compensation of specific communication channel sub-blocks as opposed to channel-wide tuning. All our tests were conducted on an STMicroelectronics 65nm low-power technology. Given the emerging role of networks-on-chip (NoCs) as reference interconnect fabrics for MPSoC platforms [111], our study targets the on-chip wires used for switch-to-switch connectivity at the top level of the NoC architecture hierarchy.

6.3 Related Work

Most research on low-swing interconnects is focused on designing circuit structures with minimal impact on delay, area and power, so the inherent advantages of low-swing signaling are not swamped by transmitter and receiver overhead. An overview of drivers and receivers is illustrated in [100, 88]. [88] makes a comparison with traditional CMOS circuits and is one of the few papers dealing with repeater stages of low-swing interconnects. The use of repeaters is avoided in [102] by means of a swing limiter and an interconnect accelerator at the receiver. Carefully engineered voltage level converters are proposed in [87, 103], while an optimized level restoration scheme based on bootstrapping can be found in [93]. Sense amplifiers are commonly used to detect a small voltage swing in reduced-swing buses [100, 89]. The minimum interconnect swing should be set by the need to overcome noise at the receiver.

An adaptive sensing scheme is proposed in [99] to reduce the threshold voltage offset between a driver and a receiver and ensure low-swing reliable operation. An adaptive voltage swing is set at circuit initialization in [98] to drive interconnects based on their delay, thus coping with the increasing interconnect delay spread. To the limit, a self-calibrating interconnect can be designed [97, 91]. Differential current-mode signaling schemes have a distinctive advantage over the single-ended ones in terms of noise immunity and signal integrity [96]. Neighbor-to-neighbor crosstalk can be reduced with twists in the differential interconnect pairs [94]. Differential low-swing interconnects come at the cost of a significant area and power overhead, therefore are not considered in this chapter. Current variation models tend to ignore variations in wires [114], however the spread of technology parameters may jeopardize functionality of transmitting and receiving circuits, causing communication performance degradation or even failure. The traditional techniques for post-silicon compensation of variability are adaptive body biasing (ABB) [106, 110] and adaptive supply voltage (ASV)[108]. Comparative studies of ABB vs ASV when put at work for variability compensation in microprocessor sub-circuits or generic combinational logic circuits have not reached a unique conclusion, proving that the choice is tightly design- and technology-dependent. In [107, 109, 105] there is consensus on the fact that ASV has a larger tuning range of circuit properties and the combined use of ASV and ABB further extends this range. However, the measured yield improvements are different depending on the technology and the design at hand, so it is not unambiguous whether hybrid approaches are worth the cost or not. In many cases, ABB seems to suffice for the required range of post-silicon compensation. Only for core-to-core variations ASV seems the best compensation option [92]. [101] points out the dependence of ABB and ASV efficiency on the device type and operating temperature in 90nm technology, while [109] emphasizes the role of biasing resolution as well.

This work aims at extending the analyses performed so far to the link architectures for on-chip communication. First, the intrinsic robustness of full-swing vs low-swing signaling schemes to process variations will be explored. Second, ABB and ASV will be applied to find out which extent they can restore the nominal performance of sample communication channels affected by process variations and which is the incurred power cost. Third an entire 32bits link will be placed and routed in order to apply the above analysis in a real case. The layout will be drawn with 65nm ST technology and all parasitic capacitance and resistance will be extracted, providing fine lines model with crosscoupled capacitances. Our analysis can justify a later investment in the synthesis backend of nanoscale designs to support the most suitable variability compensation technique for a given communication channel and variation

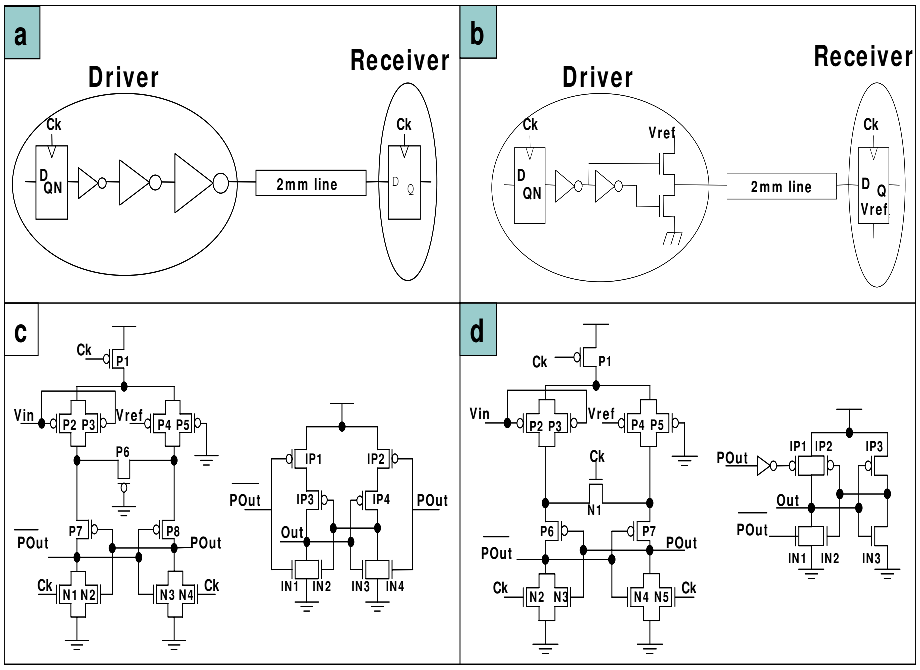


Figure 6.1: a) CMOS full-swing interconnect. b) Low-swing interconnect. c) PDIFF low-swing receiver from [100]. d) Optimized PDIFF low-swing receiver.

scenario.

6.4 Communication channel design

We at first present the design of the communication channels that will be assessed later on in terms of robustness to process variations and suitability for traditional post-silicon compensation techniques. Without lack of generality, we restrict our analysis to an intermediate layer wire with a length of 2mm, which is already the typical length of a switch-to-switch link in a regular network-on-chip architecture [111]. Inserting repeaters to reduce delay of a wire is effective only when the wire is at least twice as long as the critical length of the technology and of the specific routing layer. In our target 65nm technology, a 2mm wire falls below this threshold and the choice is therefore for an unrepeated interconnect. Even for longer links, solid network-on-chip implementation works like [95], that suggests the use of unrepeated wires for the point-to-point communication links between switches, unlike other scenarios where high-fanout nets are required. To the limit, link pipelining can be used to break long timing paths. Following these indications, this chapter assumes the use of unrepeated wires for network-on-chip communication. We at first model the on-chip wire by a Barkley $\pi 3$ distributed RC model. In that instance, we assume to tackle the cross-talk capacitance with physical-level techniques such as shielding or proper wire spacing. We leave the analysis of the interaction between cross-talk and variability compensation at the second instance, where We synthesize an entire 32 bit link and extract the parasitic resistance and capacitance with the RCc model. The reference link architecture uses a 1V full-swing signaling (Fig.6.1.a). The driver consists of a library flip-flop and a chain of buffers sized based on exponential horn methodology for minimum delay. The receiver is another library flip-flop. The alternative communication scheme is the low-swing pseudo-differential interconnect architecture reported in Fig.6.1.b. The basic circuit is taken from [100]. The driver is an NMOS-only push-pull driver which allows the use of very low power supplies and a quadratic energy reduction as a function of the voltage reference/swing V_{ref}. The receiver is still clocked but requires the voltage reference as an additional input. The original receiver circuit proposed in [100] is the clocked sense amplifier followed by a static latch illustrated in Fig.6.1.c. This pseudo-differential scheme uses single wire per bit while still retaining most advantages of differential amplifiers such as low input offset and good sensitivity. The major reliability degradation may come from the local device mismatch between the double input transistor pairs and from the variation between distant references of the driver and the receiver. In contrast, receiver operation is largely insensitive to V_{dd} supply noise, as opposed to other schemes. This was the basic motivation for picking up this scheme from [100]. However, we applied some improvements to this receiver, ending up with the circuit in Fig.6.1.d. First, PMOS transistor P6 in Fig.6.1.c has the task of equalizing the connected nodes, however it remains active even after the initialization, thus slowing down node transients. Moreover, it is not very conductive when the connected nodes reach an initialization value approaching its voltage threshold. In Fig.6.1.d it has been replaced by an NMOS transistor driven by the clock, thus achieving a better equalization and a faster node transition. Second, although the NOR static latch in Fig.6.1.c appears to be symmetric, it features uneven 0-to-1 and 1-to-0 switching times. Balancing rise and fall times makes the circuit actually asymmetric. The solution in Fig.6.1.d allows an easier balancing of these times while keeping the cross-coupled inverter pair fully symmetric: the outputs of the pseudo-differential receiver in fact directly drive the transistors (dis-)charging the flip-flop output capacitance, while the cross-coupled inverter pair keeps the sampled values. Output capacitance for the differential signal was tuned to be the same for POUT and POUTN signals. As a side effect, the flip-flop in Fig.6.1.d turns out to scale better from a performance viewpoint and enables higher operating frequencies for a comparable area than that of Fig.6.1.c. The voltage swing was chosen to be 200mV. Transistor sizing for the low-swing communication channel was done to keep the same (maximum) performance of the full-swing interconnect (1.68 GHz): driver sizing was used to achieve the same link propagation delay, while receiver and static latch sizing was used to enforce the same clock propagation time, so that the next logic stage fed by the communication channel is impacted in the same way. In particular, the library constraints for such propagation time were enforced.

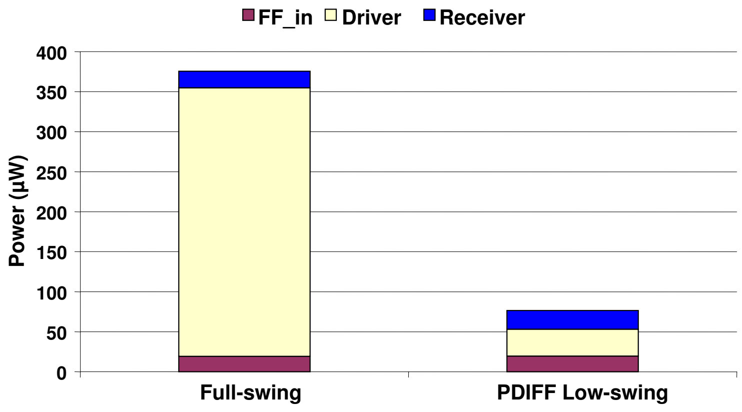


Figure 6.2: Power breakdown at 1.68 GHz, i.e. the maximum performance achievable by full-swing signaling.

6.4.1 Communication circuitry characterization

The first step of our link characterization imply power and area exploration of the full-swing vs PDIFF low-swing signaling schemes in order to provide the same performance applyed to the Barkley $\pi 3$ line model. Power results with 100% input switching activity are reported in Fig.6.2. Our low-swing channel consumes almost 5x less power than the full-swing one, confirming its power efficiency. Most of the power savings obviously come from the driver and from its reduced reference voltage. The input flip-flop is the same, and so is the power. Moreover, the PDIFF receiver almost equals the power of the library flip-flop in the full-swing scheme, which was chosen with the minimum driving strength. Low-swing signaling also achieves 28.5% lower leakage power. Most of the savings come again from the driver, but also the PDIFF receiver has a lower leakage than the library flip-flop, due to the power gating PMOS transistor in pre-charge mode and to the minimum area NMOS transistors that are switched off in evaluation mode. As regards area, the low-swing channel has a negligible 1% increase in area. The low-swing receiver has a slightly larger area than the library flip-flop, which is counterbalanced by the lower area footprint of the low-swing driver. Please observe that the PDIFF receiver consumes the same total power of the library flip-flop with more area, and this is due to the fact that some of its internal nodes switch with a lower swing. Finally, by modeling and simulating wire lengths larger than 2mm, we got almost the same quadratic delay increase for the full-swing and the low-swing interconnects as shown in Fig.6.3, since the time constant stays the same. Given a target frequency for a network-on-chip design, the NoC must ensure a maximum link length, eventually enforced by applying link pipelining techniques.

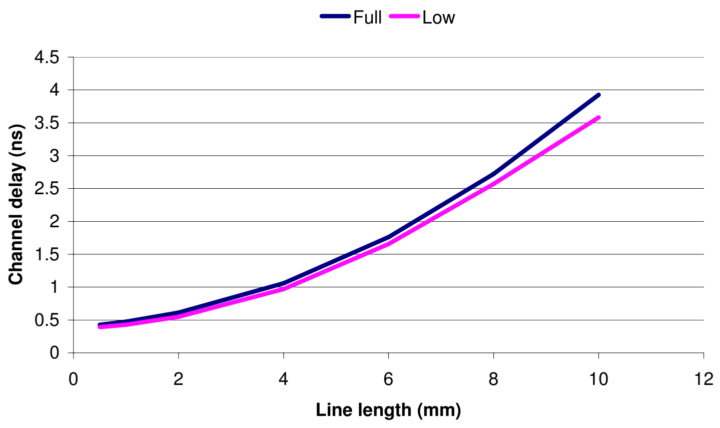


Figure 6.3: Channel Delay vs line length.

6.4.2 32bit synthesized channel characteristic

At the second step We have synthesized 32bit communication channel by Synopsys Physical compiler and placed and routed by Cadence SoCEncounter. This link is typical in NoC [111] and has transmitter and receiver 2mm far, 32 bit channel width, STALL and VALID control signals. The Synthesizer was instructed to realize unrepeated lines and to generate the clock tree. Afterward We extracted the parasitic resistance and capacitance generating two link SPICE netlists with fine line $\pi 3$ RC model and within/out crosstalk capacitance. Those communication channel netlists were connected with full-/low-swing transmitter and receiver spice netlist and run by HSPICE engine.

Power at 910Mhz (With cross coupled capacitance)					
	Busy (mW)(100% activity)			Idle	Leakage
	Max	Average	Min	(mW)	(μW)
Fullswing	5.26	4.49	3.71	0.26	3.8
Lowswing	1.36	1.33	1.30	0.42	3.1

Table 6.1: Statistic pattern dependent power analysis of the full- low-swing link with the 32 bit synthesized channel.

The Fig. 6.4 reports the routed channel by SoCEncounter. As possible to see the lines are counterintuitive routed not straight as flit30, this implies a

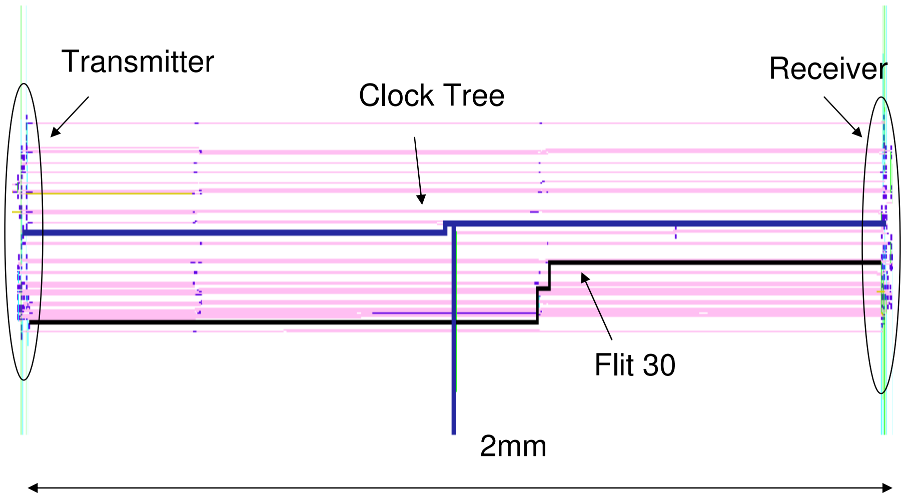


Figure 6.4: 32-bit communication channel layout.

complex crosstalk interaction among the lines resulting in a hard to find worst, average and best data patterns for power analysis. Hence we decided to execute a statistical power analysis feeding the transmitter with a batch of random data patterns. The results are shown in the table 6.1, where now the low-swing saves 3.4 times less power than full-swing in the average case. Considering the lines capacitance, we found a smaller value in the synthesized lines than the Barkley model (270fF vs 380fF). Such capacity reduction decreases the weight of the driver in the total power, reducing therefore the power saved by the use of the low-swing link. Table 6.1 presents also leakage power, that are similar for both links, and power dissipation for links in idle state (0% activity and clock on). Low-swing performs worse idle state due to the PDIFF dynamic differential nature, where in each clock period one of the two branches has to switch. Further analysis actuated by changing the input activity of the links, has been resulted with smaller power dissipation of the low-swing for activity bigger than 5%, justifing the use of the low-swing instead of the traditional CMOS in all the links where the enable signal gates the clock.

6.4.3 Crosstalk interference

In this section we discuss the consequence of the crosstalk in both full- and low-swing link when the worst performance scenario is applied. Every bit switch in the same direction $(0/1 \rightarrow 1/0)$ except one, which does the opposite transition $(1/0 \rightarrow 0/1)$ putting its driver line in the worst dis/charging cross coupled capacitance condition. We applied the worst scenario to each line. The firsts analysis have been result in a loss in both links performance

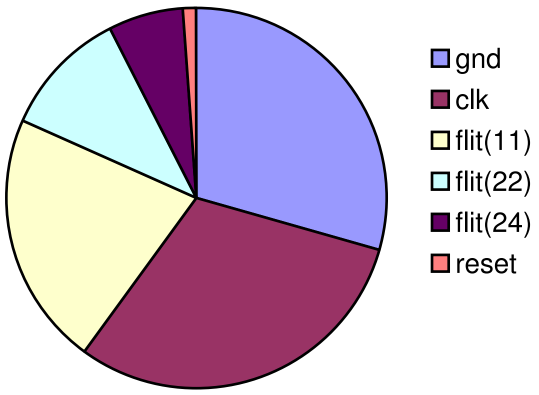


Figure 6.5: Composition of the flit28 capacitance.

from 1.28Ghz, reached without crosstalk, to 910Mhz. Furthermore the receiver clock propagation time (taken from the clock sampling edge to the 50% output voltage swing) has increased by 6% in the low-swing, denoting a little higher sensitivity to crosstalk than full-swing link. Those results came out using a clock tree routed with constraint to avoid corsstalk to the link lines (e.i. shielding, proper wire spacing). Whereas without constraint the clock is routed as in Fig.6.4, generating high crosstalk to the flit28, where the cross coupled capacitance with the clock is the 30.5% of the whole line capacitance Fig.6.5. In such condition the full-swing lose 16.5% performance while the low-swing receiver miss to sample the flit28. Therefore is mandatory to route the clock tree with physical constraint in order to reduce the crosstalk in low-swing link.

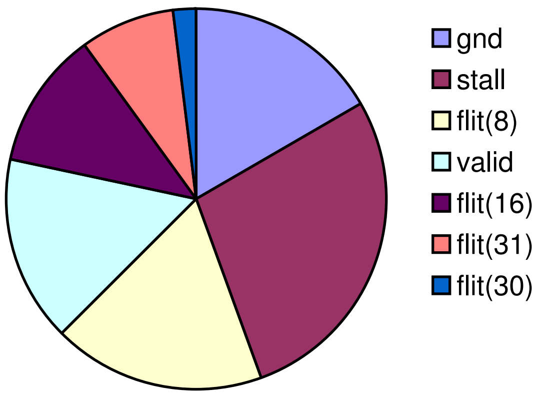


Figure 6.6: Composition of the flit7 capacitance.

Another physical constraint that could concern the route of a low-swing link is the interaction with full-swing lines. In order to analyze such situation we have designed a low-swing link with full-swing control signals and we applied the worst performance case scenario. The results shows a loss in performance of 31.7% respect a whole low-swing link, especially located in line7 Fig.6.6 where the cross coupled capacitance with control signal is the 43.5% of the whole line capacitance. Then we suggest to route a low-swing link with a physical constraint to avoid high interaction with other full-swing lines present in the design.

6.5 Robustness to process variations

The first objective of this chapter is to compare the inherent robustness of full-swing and PDIFF low-swing signaling schemes respect to process variations, while compensation techniques will be addressed later on. Our focus is on within-die variations, which happen at the length scale of a die, and that can be further divided into two contributors: systematic and random. Systematic variations can be predicted prior to fabrication and exhibit space locality. In contrast, random variations are due to the inherent unpredictability of the semiconductor technology itself. In our tests we inject effective gate length variations, which have implications on the threshold voltage as well, as computed by the SPICE device models of our target library. HSPICE is used as our simulation engine. We ignore variations in wires, in agreement with current variation models (e.g., [114, 108]), consequently all the conclusion are valid for both Barkley line and synthesized link.

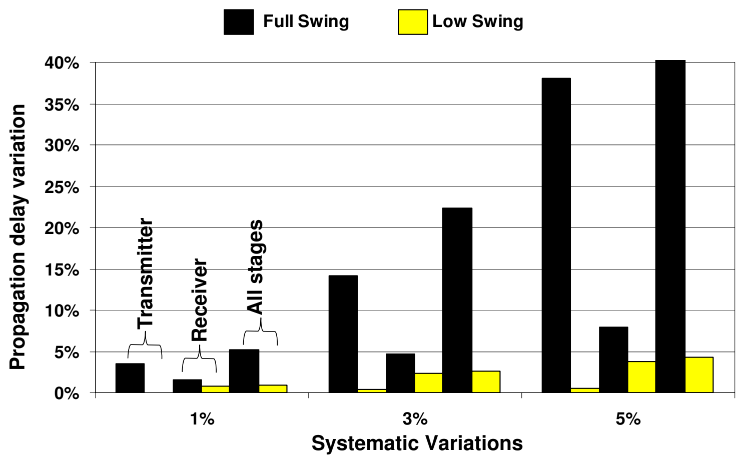


Figure 6.7: Sensitivity to systematic variations.

Fig.6.7 shows the sensitivity of the signaling schemes to *systematic variations*. The sensitivity is measured as the variation-induced deviation of the clock propagation time of the receiver from the nominal value. The propagation time goes from the clock sampling edge to the 50% voltage swing of the receiver output, and its nominal value is the same for both full- and low-swing channels, since they were designed to impact the next stage of the design in the same way. Systematic variations have been applied selectively to the transmitter, to the receiver and to the whole channel, so the bars in Fig.6.7 should be read pairwise. It can be clearly observed that low-swing signaling proves a far more robust scheme to systematic variations. By restricting the analysis to the full-swing channel, its transmitter turns out to be the weak point of this scheme. The reason lies in the high sensitivity of the library flip-flop (i.e., the receiver) to the settling time of its input signal. This latter significantly deviates from nominal conditions when systematic variations affect the transmitter, and this explains the large degradation of the whole full-swing channel performance. In contrast, the receiver seems much more robust, and variations affecting the whole channel introduce only an incremental degradation with respect to the one caused by the transmitter. The only exception occurs for channel-wide 5% systematic variations, where nominal delay is degraded by 90% (height of the last column for full-swing is truncated to preserve the scale). This is much more than one could expect by looking at the transmitter-degraded case, but this is due to the fact that we are working close to the point where full-swing channel operation fails: in this region, delay is highly sensitive to process parameter variations. The opposite holds for the low-swing channel. The PDIFF receiver does a good job in providing a noise margin to the perturbations of its input signal induced by systematic variations in the transmitter. However, when variations affect directly the receiver, the PDIFF scheme suffers from increased switching delay. Clock propagation delay variations are much smaller for low-swing channels with respect to the full-swing ones anyway, and might more easily induce the following stage in the design to fail, since it may be impossible to leave a 90% performance degradation margin for 5% systematic variations, as required by the full-swing channel. We detected a failure of the full-swing channel when the transmitter is affected by 6/7% variations (tolerating a maximum propagation delay degradation by 90%), while the low-swing channel can keep working also under 70% systematic variations affecting both transmitter and receiver, after that the channel fails. At that time, however, propagation delay is degraded by 40%.

The sensitivity of the channels under test to random variations ($3\sigma/\mu$=15%) is illustrated in Fig.6.8. Delay variability is similar in the two cases, with a slightly more tightened distribution for the low-swing channel. Again, we found the transmitter to be the most critical part of the full-swing channel, while the receiver is obviously the weak point of the low-swing channel. In fact, its pseudo-differential behaviour makes it very sensitive to random process variations, although we found only a negligible amount of malfunction-

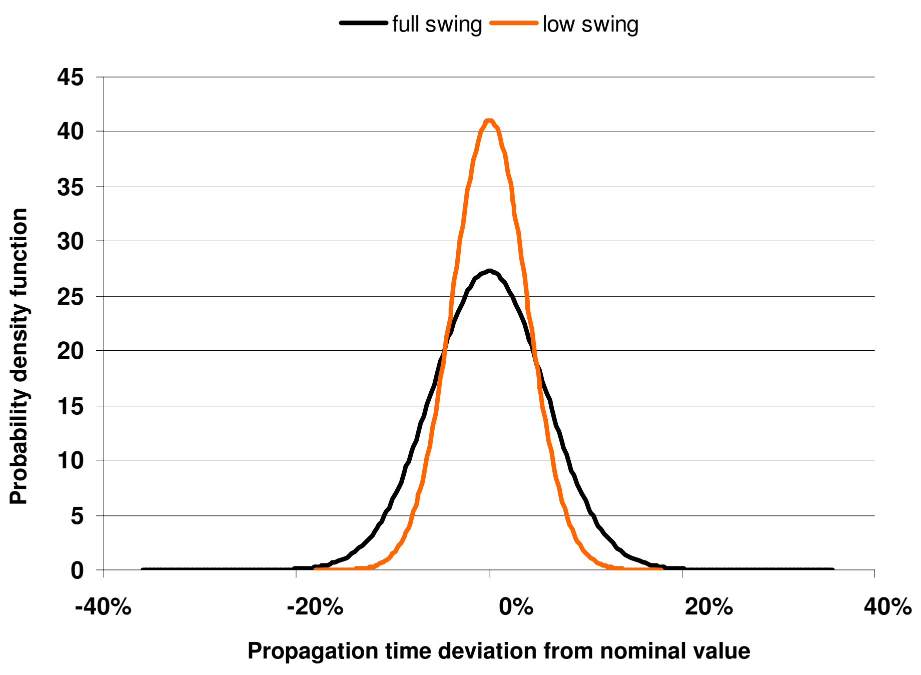


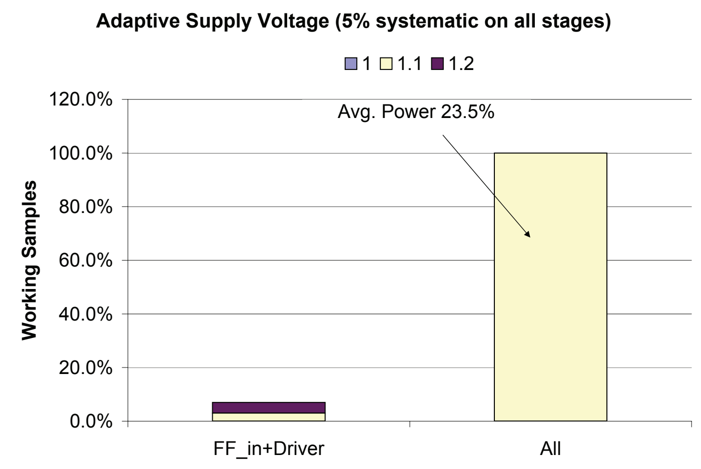
Figure 6.8: Sensitivity to random variations.

ing channels with $3\sigma/\mu$ lower than 20%. This indicates that under such variations, the unbalancing of the differential branches remains within the noise margin of the receiver and correct 1/0 sampling takes place in due time. Delay variations pointed out in Fig.6.7 and Fig.6.8 indicate that compensation is apparently more challenging in full-swing channels, though the effectiveness of compensation depends not only on the entity of delay variations, but also on the sensitivity of channel delay to the different channel sub-blocks and also to the interaction among them, as illustrated hereafter.

6.6 Post-silicon compensation

Next, we explore the effectiveness of ABB (and forward body bias, FBB, in particular) vs ASV in bringing channel instances slowed down by process variations back within nominal performance. Compensation is applied to both the driver and the receiver for channel-wide tuning, but also selectively to individual sub-circuits to capture sensitivity of channel performance to that of these sub-circuits and eventually come up with lower-cost compensation techniques.

The ideal performance tuning range of each technique is investigated, without regarding of implementation issues, to justify an investment on the most suitable technique for each kind of communication channel later on.



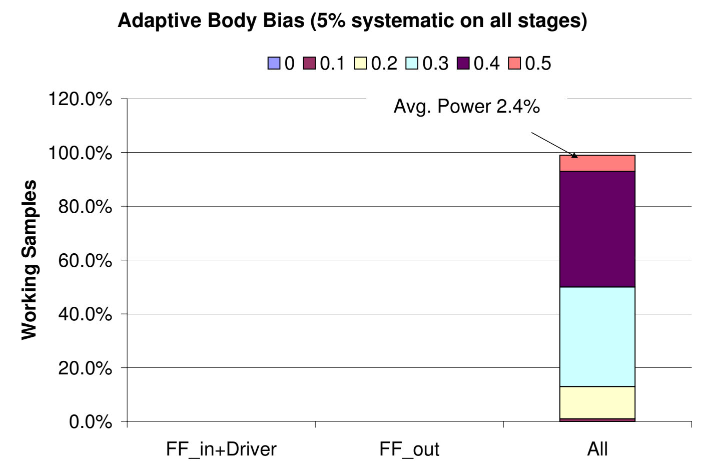


Figure 6.9: Working samples after compensation of *full-swing channels*. x-axis indicates the channel (sub-)circuits to which compensation has been applied.

6.6.1 Experimental framework

Our experiments encompass the compensation of a representative subset of variation scenarios. Similarly to [92, 108], worst-case systematic variations of +5% of parameter nominal value are assumed and superimposed to random variations. For these latter, the $3\sigma/\mu$ of channel length distribution is varied from 10, 15 to 20%, thus giving rise to three scenarios featuring the same amount of worst-case systematic variations and an increasing parameter spread associated with random variations.

Recently, advanced modeling frameworks have been proposed to propagate variation information from the transistor compact model up to the system level, offering a correlated view on yield, timing, dynamic and static energy [113]. They also improve the traditional Monte Carlo statistical static timing analysis techniques by accounting for rare events in variability distributions.

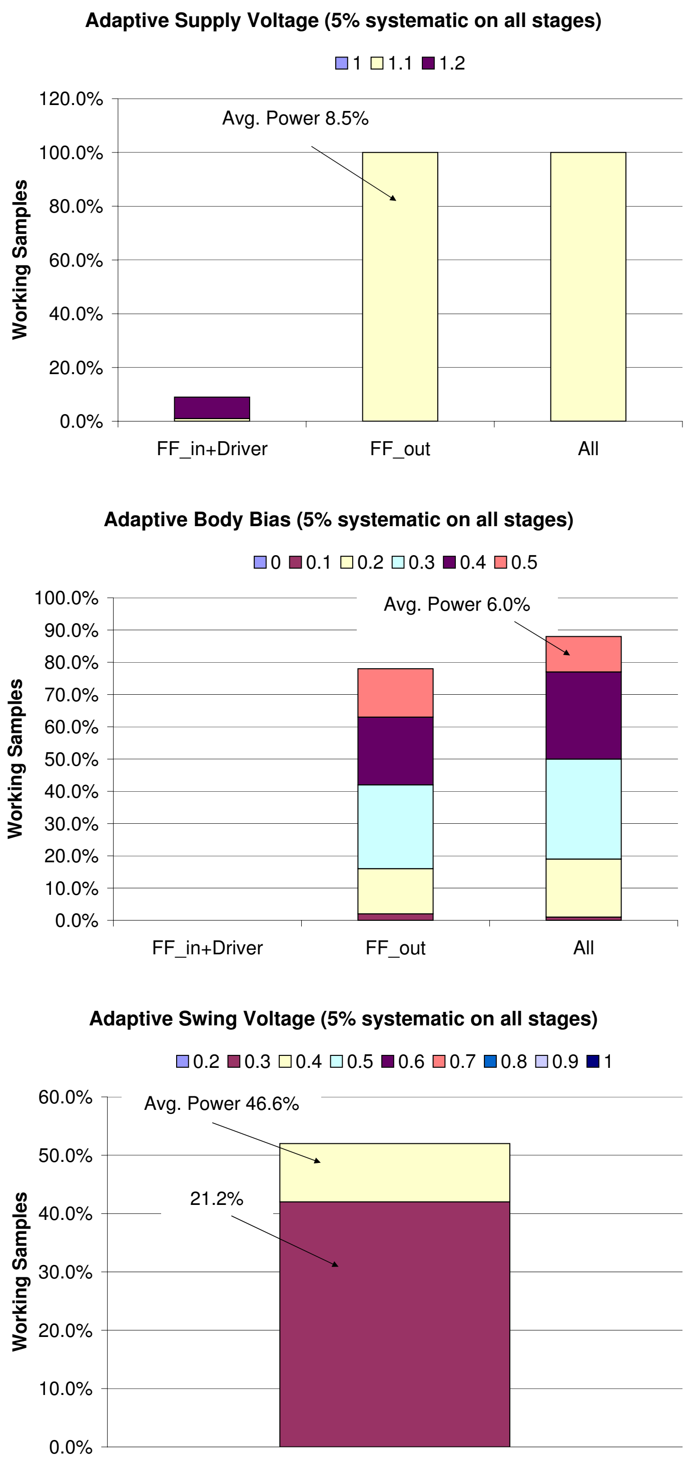


Figure 6.10: Working samples after compensation of *PDIFF low-swing channels*. x-axis indicates the channel (sub-)circuits to which compensation has been applied.

Since this chapter focuses on a relatively small yet critical amount of logic, we developed an ad-hoc and simplified methodology based on Monte Carlo analysis to study the impact of systematic and random variability and how effectively it can be compensated. For each signaling scheme, variation scenario and compensation technique, we perform Monte Carlo simulations with a statistically significant sample set. Each Monte Carlo run (i.e., a channel instance with different random variability injections) goes through the compensation methodology illustrated in Fig.6.11. At first, we check for nominal performance requirements. If met, a new instance is analyzed. If not, a compensation step is applied. In practice, if FBB is under test, an incremental reduction step of the body bias is applied so to improve performance. Similarly, the supply voltage is increased when ASV is assessed. Decrements/Increments are applied with steps of 100 mV both for ASV and FBB. This choice stems from the conclusion of previous works [106] and from considering realistic resolutions of low-cost voltage regulators. After the compensation step, performance is re-evaluated and eventually an additional compensation step is applied. The process completes when nominal performance is finally met OR when the voltage range limit is reached: 500 mV for forward body bias (to avoid turning on the source pn junction of transistors) and 200mV for ASV (for reliability and technology library constraints). Since our target 65nm manufacturing process does not provide a triple well, we apply forward body biasing only to PMOS transistors. Our analysis aims to capture whether this lower cost solution suffices for compensation purposes in on-chip communication channels. In addition, it is not possible to selectively apply ASV only to the receiver of a full-swing channel, since this would require a voltage level shifter which is not there. In contrast, such level shifter comes for free in a low-swing channel, which therefore allows PDIFF receiver selective compensation with ASV. FBB does not have any kind of constraints in any signaling scheme.

Effectiveness of a technique is expressed as the percentage of the sample set that can be brought back within nominal performance by the compensation technique under test. We denote those effectively compensated samples as *working samples*. Nominal performance means correct sampling at 1.68GHz for the Barkley model and 910Mhz for the synthesized link, with clock propagation time constraints met at the output of the receivers. Moreover, the average power overhead for compensating channel instances with the highest power supply value (lowest PMOS body bias value) is measured, denoting *power efficiency* of the compensation techniques. For low-swing signaling, we also explore adaptive voltage swing as an additional and built-in compensation technique by raising the voltage swing in increments of 100mV. Afterward, *systematic variations* were applied to the *whole channel but also selectively* to the receiver

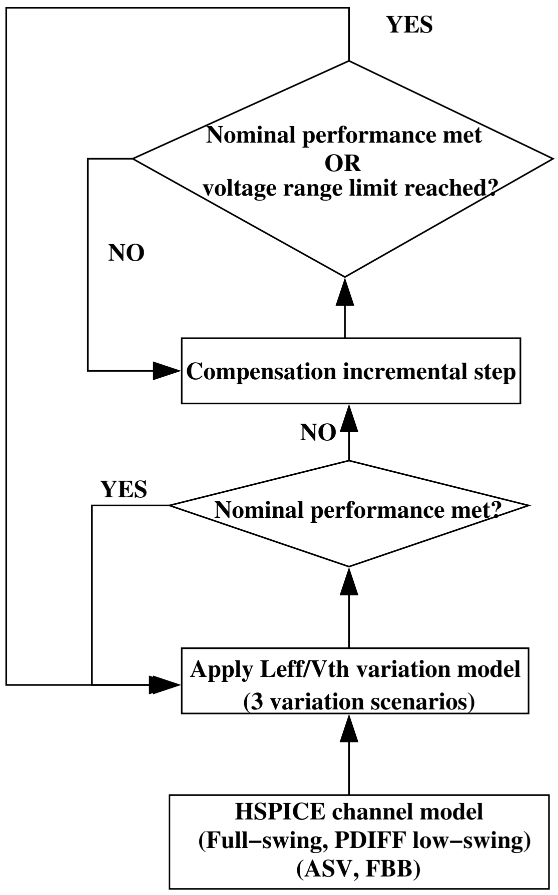

YES
Nominal performance met
OR
voltage range limit reached?
NO
Compensation incremental step
NO
YES
Nominal performance met?
Apply Leff/Vth variation model
(3 variation scenarios)
HSPICE channel model
(Full−swing, PDIFF low−swing)
(ASV, FBB)

Figure 6.11: Framework for assessing the effectiveness of variability compensation techniques.

and to the transmitter to account for place&route effects. In fact, transmitter and receiver might be far apart from each other, thus suffering from systematic variations to a different extent, or they might be placed close to each other. In this latter case, physical parameters of the whole communication channel would be skewed by the same amount. We hereafter report only this latter case and the differences (if any) with the other variation scenarios are discussed in the text. We also recall that *random variations were always applied* to the circuits of the *whole channel*, and in the first set of experiments $3\sigma/\mu$ is assumed to be 15%. See afterward for different values. When systematic variations were injected in the entire channel (like random ones), we found almost no channel instances in the sample set working without compensation, both for full-swing and low-swing channels. So, in the experiments that follow, the entire sample set needs to be compensated. Finally we analyzed the compensation techniques applied to the synthesized link with the previous variation scenarios, in order to evaluate their effectiveness under crosstalk regime.

6.6.2 Compensation efficiency in full-swing links

As can be observed from Fig.6.9, neither ASV nor FBB are able to restore functionality of all working samples by only acting upon the transmitter or (for FBB) the receiver. The compensation in this case would be totally ineffective. Variability can only be compensated by tuning all the circuits of the channel. In fact, under 5% systematic variations performance of full-swing channels is highly sensitive to the interaction between the signal provided by the transmitter and the requirements imposed by the receiver. Moreover, such variations (recall Fig.6.7) significantly impact both the transmitter and the receiver. As a consequence, an effective compensation can only be carried out by acting upon both modules at the same time. However, while ASV requires a single voltage step to reach 100% working samples, FBB needs its entire voltage range to achieve the same objective. Even the large variations taking place in full-swing channels can be offset by FBB in spite of its inherently weaker performance tuning capability by exploiting the sensitivity of channel performance to the circuits compensation is applied upon. The main difference lies however in the power efficiency of the techniques. When ASV raises the supply voltage to 1.1V, the communication channel instances on average exhibit a 23% power overhead with respect to the variation-free scenario. In contrast, a 500mV forward body bias incurs only an average power overhead of 2.4%, almost negligible. When we applied systematic variations only to the transmitter (flip-flop and driver), we observed that tuning only the transmitter circuits only partially solved the problem. ASV could restore about 80% of the samples, while FBB about 60% by remaining in the voltage range limits. This indicates the impact of random variations, which require a tuning of the receiver as well to restore 100% working samples. The situation is even worse when only the receiver is affected by systematic variations: while no selective tuning of the flip-flop is feasible with ASV due to a lack of a voltage level shifter, only 20% of working samples were achieved by selective FBB. Again, the only option was to tune the entire channel, finding again the same power efficiency gap between FBB and ASV.

6.6.3 Compensation efficiency on low-swing links

Quite different considerations hold for variability compensation in low-swing channels. This time, ASV can be selectively applied to the receiver since the level shifter is built-in in the signaling scheme. Fig.6.10 clearly shows that a selective tuning of the receiver with both ASV and FBB reaches a high percentage of working samples. With just one voltage increment step applied to the output flip-flop, ASV can restore performance of all slow samples. More interestingly, the average power overhead is limited to 8.5%, much lower than in a full-swing channel. In low-swing channels, the transmitter is marginally impacted by systematic variations (recall Fig.6.7). At the same time, receiver performance is much less sensitive to the perturbations of the input signal than in full-swing channels. Therefore, acting upon the receiver proves an effective compensation method. Unfortunately, FBB cannot reach 100% working samples with a selective compensation at the receiver, and neither a channel-wide compensation can (90% is the best result achieved with a 500mV FBB). This is essentially due to weak performance knob represented by FBB, which is not boosted by any circuit level property in this case (for instance, no high sensitivity of channel performance to transmitter-receiver interaction). The worst-case average power overhead incurred by FBB is around 6%, comparable with that of ASV. Considering the cases where systematic process variations affect only the transmitter or the receiver, we found that FBB is not able again to reach 100% of working samples (best coverage is 90%). ASV instead works effectively. However, in all cases and for both ASV and FBB, selective compensation at the receiver turns out to be as effective as full channel compensation. Power overhead for ASV is around 7 and 8%, while for FBB is around 3%. There is a slightly higher power overhead of ASV which is the price to pay to achieve a higher compensation efficiency and, in the end, a higher yield. However the absolute extra power required to the ASV to recover all the sample is $6.51\mu W$ whereas the ABB in full-swing link requires $9.02\mu W$, confirming the better power efficiency of the low-swing, even for process variation compensation technique.

Fig.6.10 also shows the efficiency of an intuitive compensation technique which stems from the possibility to tune the voltage swing in the low-swing channel. Although intuitive, this technique proves highly ineffective to restore channel performance. By increasing the voltage swing from 200mV to 400mV, only 50% of the slow samples can be saved. Interestingly, by further increasing the swing proves useless, and no further improvements can be achieved, thus spending power uselessly. This is due to the fact that compensating process variations is not just an issue of speeding up signal propagation across the link, but to restore correct functionality at the transmitter and at the receiver. Only when the transmitter is impacted by systematic variations while the receiver is not, then speeding up the link with a swing of 400mV achieves 82% working samples. Compensating receiver variability proves more difficult (about 60% working samples). Another argument against reference voltage scaling is power. The measured average power overhead for the worst case compensations (those at 400mV) amounts to a significant 46%. This confirms the results of the work in [91], showing that using the voltage swing to speed up a low-swing link is highly power inefficient.

6.6.4 Role of random variations

When we repeated the experiments with a $3\sigma/\mu = 10\%$ and below, the minor role played by random variations translated into a better compensation efficiency of FBB in low-swing channels, since working samples were always close to 100%. The lower delay spread makes the worst-case compensation scenario affordable also for the tuning capability of FBB, so that this latter can be considered also for low-swing signaling as the impact of random variations decreases. Finally, $3\sigma/\mu$ was set to 20%. In this case, even for full-swing channels FBB could not bring all samples within nominal performance bounds, although still achieving around 95% working samples. Interestingly, in low-swing channels the effectiveness of FBB was as low as 70% working samples.

6.6.5 Compensation technique with crosstalk

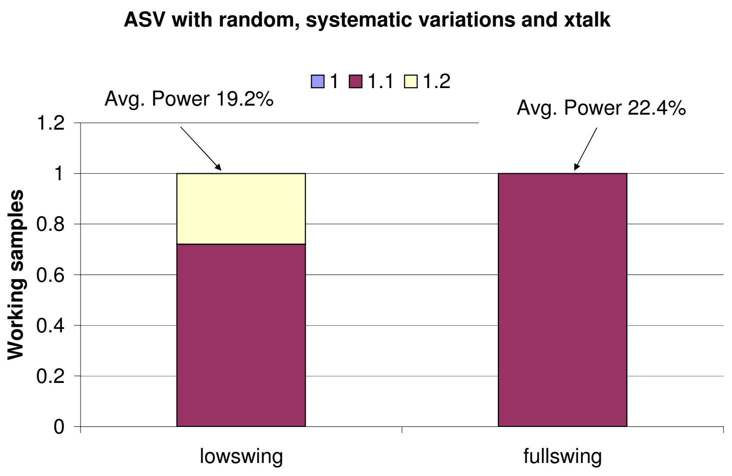


Figure 6.12: Working samples after ASV compensation with random, systematic variations and crosstalk.

As last experiment we applied the previous best found compensation techniques to the synthesized link described by RCc parasitic extracted SPICE netlist. We used ASV and ABB to the all channel for the full-swing link, while the low-swing had ASV to the receiver and ABB to the all channel, in according to the previous analysis. The introduction of crosstalk did not change the capability of ASV to recover 100% of the sample set with only one step for the full-swing link, see Fig.6.12. The average power overhead is 22.4% like the overhead required without crosstalk. Whereas the low-swing requires two ASV step to recover all the sample set, exhibits 19.2% power overhead. Such results are in agreement to the low-swing higher crosstalk sensitivity, which requires another ASV step to compensate the variation. However the expense exhibited by the low-swing link is still smaller than the full-swing one, denoting it suitability in the future links.

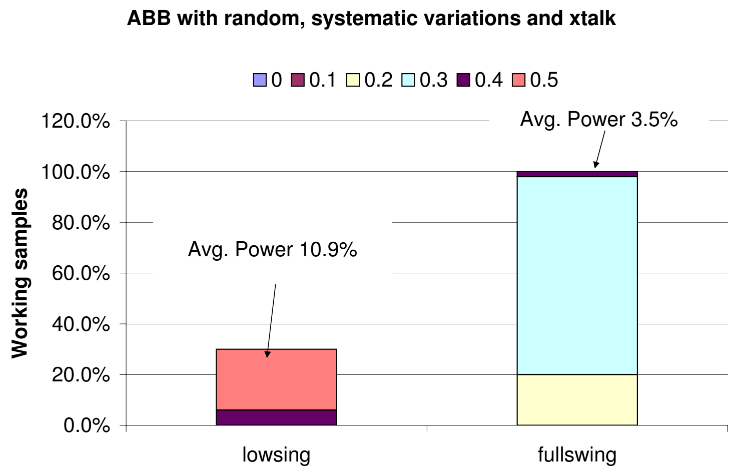


Figure 6.13: Working samples after p-mos n-mos ABB compensation with random, systematic variations and crosstalk.

Thereafter we ran the simulations with ABB correction technique. In this case the presence of crosstalk affected both full- and low-swing link, where only a few samples were brought back to the nominal performance. The application of the ABB to the only p-mos impose by the presence of a single well technology, drastically increase the influence of the crosstalk. In fact when we ran simulation with ABB applied to both p- and n-mos transistors Fig.6.13, the full-swing channel has all the sample set working at the nominal performance, while the low-swing exhibits its small ABB sensibility. To conclude, in crosstalk regime the full-swing channel has is best performance with ABB applied to the all channel and both p- and n-mos transistors, whereas the low-swing link is more suitable for ASV compensation applied at the only receiver.

6.7 Conclusions

This work explores the effectiveness of ASV and FBB as post-silicon variability compensation techniques for on-chip communication channels. Our work shows that FBB is effective for tuning performance of full-swing channels with minimum power overhead. In contrast, when applied to low-swing channels, FBB proves not capable of compensating all variation patterns, since its limited performance tuning capability is not amplified by any circuit property. On the other hand, ASV can exploit the built-in voltage level shifter in low-swing channels and achieve an effective and low power-overhead compensation.

The results of this chapter point out the superior robustness of low-swing channels to process variations. After considering a realistic range of systematic and random WID process variations and exploring all the possible countermeasures based on ABB and ASV, it is evident that low-swing channels (i) can better cope with systematic variations (lower delay deviations and functional correctness guaranteed over a wider range of variations), (ii) feature a lower delay spread under random variations, (iii) can be compensated with success against delay variability at a low power cost. These features add up to the reference characteristic of low-swing channels, which is their inherent low power consumption.

Conclusions

The shrinking transistor dimensions with increasing local temperature and process variation, are pushing the designers to face new reliability and manufacturability challenges. The standard approach based on the worst case design, in conjunction with manufacturing and packaging refinements, will not be efficient anymore. Therefore researchers and industries are developing new system and logic design techniques aware of circuit reliability and manufacturability.

The thesis has introduced readers into the new technology challenges, presenting novel design approaches to increase robustness and yield of the future integrated circuits. Furthermore the contribution of my Phd research activity in the reliability and manufacturability issues has been illustrated. At first I developed a thermal analysis tool, that has been integrated in a NoC cycle accurate simulator and in a FPGA based NoC simulator. Meanwhile the need for a thermal analysis tool in the first design stages has been investigated: i) Showing as an accurate layout distribution can avoid the onset of hot-spot. ii) Proving the temperature and thermal cycles reduction due thermal management application. Moreover I developed a statistical process variation analysis tool able to address both random and systematic variations. The need of a static analysis tool at the first design stages has been advocated: i) Proving the system manufacturability increment due self-timed stages insertion in an embedded microprocessor. ii) Analyzing the NoC link robustness, where the low swing technique has shown a better power, speed and yield behaviors. iii) Testing the efficiency of process compensation technique in NoC communication channel, noticing the ASV as best solution for low swing and ABB for full swing links.

Acknowledgements

Finally I am approaching the end of the philosophy doctorate, finally used only as temporal meaning. In fact those three years have deeply changed my life. I learned a lot of things, not only in my research field. I knew, met, worked and enjoyed the life with a lot of people from everywhere in the world. I appreciated different culture and maybe I understood better mine, that is evolving from Italian to European, or maybe I would define myself as world wide citizen, always maybe, remembering all the time I was and will be proud of my Italian, and Pesarese origin.

However, let me thanks my professor Luca Benini and senior Research Pol Marchal, which have given me the possibility to do the doctorate. Thanks to Davide Bertozzi, whom I have collaborated for the last year. Moreover I wanna thanks all the guys of Micrel Lab and μIdea. Special Thanks to all the guys I met in Leuven, IMEC, and the guys of Ferrara.

At the end I wanna thanks my parents and brother, who have always supported me, and Silvia, who has filled my empty heart since nine months ago till i don't know.

Bologna
March 11, 2007

Bibliography

[1] International Technology Roadmap For Semiconductors
2007, cap Design
www.itrs.net

[2] Srinivasan, J., Adve, S., Bose, P., Rivers, J., Kun Hu, C. (2003). ”RAMP: A Model for Reliability Aware MicroProcessor Design”. In *RC23048*.

[3] Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K., and Tarjan, D. (2003). ”Temperature-Aware Microarchitecture”. In *Proc. IEEE/ACM ISCA*, pages 2-13, San Diego, CA, USA.

[4] Papanicolaou, A. et al., ”At Tape-out: Can System Yield in Terms of Timing/Energy Specifications be Predicted?”, *IEEE Custom Integrated Circuits Conference*, pp.773-778, 2007.

[5] Benini, L., Bertozzi, D., Bogliolo, A., Menichelli, F., Olivieri, M. (2005). ”Mparm: Exploring the multi-processor SoC design space with SystemC”. *Journal of VLSI Signal Processing* 41, 2, pp.169-182.

[6] Atienza, D., Del Valle, P., Paci, G., Poletti, F., Benini, L., De Micheli, G., Mendias, J.M., Harmida, R. (2007). ”HW-SW emulation framework for temperature-aware design in MPSoCs.” *ACM TODAES,*Volume 12, Issue 3 Article No. 26

[7] International Technology Roadmap For Semiconductors
www.itrs.net

[8] Kundu, S. ”Managing Variations: from Devices to Systems”, *DATE 2008 Workshop: Impact of Process Variability on Design and Test* , Munich, March 14, 2008.

[9] Ernst, D. et al ”Razor: a low-power pipeline based on circuit-level timing speculation” *IEEE/ACM MICRO-36, 2003*, Proceedings, pp. 7-18, 3-5 Dec. 2003

[10] Eireiner, M.; Henzler, S.; Georgakos, G.; Berthold, J.; Schmitt-Landsiedel, D. ”In-Situ Delay Characterization and Local Supply Voltage Adjustment for Compensation of Local Parametric Variations” *IEEE Journal of Solid-State Circuits*, Volume 42, Issue 7, July 2007 Page(s):1583 - 1592

[11] Dean, M.E.; Dill, D.L.; Horowitz, M. ”Self-timed logic using current-sensing completion detection (CSCD)” *Proc. IEEE ICCD '91*, 14-16 Oct 1991

[12] Kondratyev, A.; Lwin, K. ”Design of asynchronous circuits using synchronous CAD tools” *IEEE Design and Test of Computers*, vol.19, no.4, pp.107-117, Jul/Aug 2002

[13] Srinivasan, J.; Adve, S.; Rivers, P. ”Lifetime reliability: toward an architectural solution” *IEEE Micro*, vol.25, Issue 3, pp.70-80, May/Jun 2005

[14] Bertozzi, D., Jalabert, A., Murali, S., Tamhankar, R., Stergiou, S., Benini, L., and DeMicheli, G. (2005). NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip. *IEEE T. Parallel and Distributed Systems*, 16(2):113–129.

[15] Brooks, D. and Martonosi, M. (2001). Dynamic thermal management for high-performance microprocessors. In *Proc. IEEE HPCA*, pages 171–182, Nuevo Leone, Mexico.

[16] D.Frank, Dennard, R., Nowak, E., Solomon, P., Taur, Y., and Wong, H. (2001). Device Scaling Limits of Si MOSFETs and Their Application Dependencies. *Proc. IEEE*, 89(3):259–288.

[17] Gunther, S., Binns, F., and and. J. Hall, D. C. (2001). Managing the impact of increasing microprocessor power consumption. *Intel Technology Journal Q1*.

[18] Hamann, H., Weger, A., Lacey, J., Cohen, E., and Atherton, C. (2006). Power Distribution Measurements of the Dual Core PowerPC 970MP Microprocessor. In *Proc. IEEE/ACM ISSCC*, pages 2172 – 2179, San Francisco CA, USA.

[19] Heo, S., Barr, K., and Asanovic, K. (2003). Reducing Power Density through Activity Migration. In *Proc. IEEE/ACM ISLPED*, pages 217–222, Seoul, Korea.

[20] Huang, M., Renau, J., Yoo, S., and Torrellas, J. (2000). A framework for dynamic energy efficiency and temperature management. In *Proc. ACM/IEEE Micro*, pages 202–213, Monterey, CA, USA.

[21] Hung, W., Xie, Y., Vijaykrishnan, N., Kandemir, M., and Irwin, M. (2005). Thermal-Aware Task Allocation and Scheduling for Embedded Systems. In *Proc. IEEE/ACM DATE*, pages 898–899, Munich, Germany.

[22] Kanda, K., Nose, K., Kawaguchi, H., Lee, S., and Sakurai, T. (2001). Design Impact of Positive Temperature Dependences on drain current in Sub 1V CMOS VLSIs. *IEEE J. Solid State Circuits*, 36(10):1559–1564.

[23] Kao, J., Miyazaki, M., and Chandrakasan, A. (2002). A 175mV Multiple-Accumulate Unit Using an Adaptive Supply Voltage and Body Bias Architecture. *IEEE J. Solid-State Circuits*, 37(11):1545–1555.

[24] Liao, W., Li, F., and He, L. (2003). Microarchitecture level power and thermal simulation considering temperature dependent leakage model. In *Proc. IEEE/ACM ISLPED*, pages 211–216, Seoul, Korea.

[25] Loghi, M., Angiolini, F., Bertozzi, D., Benini, L., and Zafalon, R. (2004). Analyzing Chip Communication in a MPSoC Environment. In *Proc. IEEE/ACM DATE*, pages 752–757, Paris, France.

[26] Loghi, M. and M. Poncino, L. B. (2004). Cycle-Accurate Power Analysis for Multiprocessor Systems-on-a-Chip. In *Proc. GLSVLSI*, pages 401–406, Boston, MA, USA.

[27] Lu, Z., Huang, W., Lach, J., Stan, M., and Skadron, K. (2004). Interconnect Lifetime Prediction under Dynamic Stress for Reliability-Aware Design. In *Proc. IEEE/ACM ICCAD*, pages 327–334, San Jose, CA, USA.

[28] McPherson, J. (2001). Scaling-Induced Reductions in CMOS Reliability Margins and the Escalating Need for Increased Design-In Reliability Efforts. In *Proc. ISQED*, pages 123–130, San Jose, CA, USA.

[29] Pham, D., Asano, S., Bolliger, M., Day, M. N., Hofstee, H. P., Johns, C., Kahle, J., Kameyama, A., Keaty, J., Masubuchi, Y., Riley, M., Shippy, D., Stasiak, D., Suzuoki, M., Wang, M., Warnock, J., Weitzel, S., Wendel, D., Yamazaki, T., and Yazawa, K. (2005). The design and implementation of a first-generation CELL processor. In *Proc. IEEE/ACM ISSCC*, pages 184–186, San Francesco, CA, USA.

[30] Poirier, C., McGowen, R., Bostak, C., and Naffziger, S. (2005). Power and temperature control on a 90nm Itanium-Family processor. In *Proc. IEEE/ACM ISSCC*, pages 304–305, San Francesco, CA, USA.

[31] Rohou, E. and Smith, M. (1999). Dynamically managing processor temperature and power. In *Proc. 2nd workshop on Feedback-directed optimization*, pages 1–8, Haifa, Israel.

[32] Sanchez, H., Kuttanna, B., Olson, T., Alexander, M., Gerosa, G., Philip, R., and Alvarez, J. (1997). Thermal management system for high performance PowerPC microprocessors. In *Proc. IEEE Compcon*, pages 325–330, San Jose, CA, USA.

[33] Skadron, K., Stan, M., Huang, W., Velusamy, S., Sankaranarayanan, K., and Tarjan, D. (2003). Temperature-Aware Microarchitecture. In *Proc. IEEE/ACM ISCA*, pages 2–13, San Diego, CA, USA.

[34] Su, H., Liu, F., Devgan, A., Acar, E., and Nassif, S. (2003). Full chip leakage estimation considering power supply and temperature variations. In *Proc. IEEE/ACM ISLPED*, pages 78–83, Seoul, Korea.

[35] Vandevelde, B., Driessens, E., Chandrasekhar, A., and Beyne, E. (2001). Characterisation of the polymer stud grid array (PSGA), A lead free CSP for high performance and high reliable packaging. In *Proc. SMTA*, Los Angeles, CA, USA.

[36] Zhang, Y., Parikh, D., Sankaranarayanan, K., Skadron, K., and Stan, M. (2003). HotLeakage: A Temperature-Aware Model of Subthreshold and Gate Leakage for Architects. Technical Report CS-2003-05, Univ. of Virginia Dept. of Computer Science.

[37] AMD. 2004. Thermal performance comparison for am486dx2 and dx4 in pdh-208 vs pde-208 package. http://www.amd.com.

[38] APTIX. 2003. System explore. http://www.aptix.com.

[39] ARM. 2003. Arm AMBA 2 AHB Specification. http://www.arm.com/products/solutions/AMBA_Spec.html.

[40] ARM. 2004a. Arm Integrator Application. http://www.arm.com.

[41] ARM. 2004b. ARM7TDMI-STR71xF TQFP144 and TQFP64 10x10 Packages - Product Datasheets. http://www.arm.com/products/CPUs/ARM7TDMI.html.

[42] ARM. 2002. PrimeXSys Platform Architecture and Methodologies, white paper. http://www.arm.com/pdfs/ARM11%20Core%20&%20Platform%20Whitepaper.pdf.

[43] BENINI, L., BERTOZZI, D., BOGLIOLO, A., MENICHELLI, F., AND OLIVIERI, M. 2005. Mparm: Exploring the multi-processor SoC design space with SystemC. *Journal of VLSI Signal Processing 41,* 2, 169–182.

[44] BRAUN, G., WIEFERINK, A., SCHLIEBUSCH, O., LEUPERS, R., MEYR, H., AND NOHL, A. 2003. Processor/memory co-exploration on multiple abstraction levels. In *Proceedings of DATE*.

[45] BROOKS, D. AND MARTONOSI, M. 2001. Dynamic thermal management for high-performance microprocessors. In *Proceedings of HPCA*.

[46] CADENCE. 2005. Cadence palladium II. http://www.cadence.com.

[47] CHEN, G. AND SAPATNEKAR, S. 2003. Partition-driven standard cell thermal placement. In *Proceedings of ISPD*.

[48] CHU, C. AND WONG, D. 1998. A matrix synthesis approach to thermal placement. *IEEE Transactions on Computer-Aided Designs (T-CAD) 17,* 11, 1166–1174.

[49] COWARE. 2004. Convergence and Lisatek product lines.

[50] EMULATION AND VERIFICATION ENGINEERING. 2005. Zebu XL and ZV models. http://www.eve-team.com.

[51] GOPLEN, B. AND SAPATNEKAR, S. 2005. Thermal via placement in 3D ICs. In *Proceedings of ISPD*.

[52] HEO, S., BARR, K., AND ASANOVIC, K. 2003. Reducing power density through activity migration. In *Proceedings of ISLPED*.

[53] HERON ENGINEERING 2004. Heron mpsoc Emulation. http://www.hunteng.co.uk.

[54] IBM. 2006. IBM Packaging Solutions. http://www-03.ibm.com/chips/asics/products/packaging.html.

[55] JALABERT, A., MURALI, S., BENINI, L., AND DE MICHELI, G. 2004. xpipescompiler: A tool for instantiating application specific networks-on-chip. In *Proceedings of DATE*.

[56] JERRAYA, A. AND WOLF, W. 2005. *Multiprocessor Systems-on-Chips*. Morgan Kaufmann, Elsevier.

[57] LÓPEZ-BUEDO, S., GARRIDO, J., AND BOEMO, E. I. 2000. Thermal testing on reconfigurable computers. *IEEE Design & Test of Computers 17,* 1, 84–91.

[58] MAGNUSSON, P. S., CHRISTENSSON, M., ESKILSON, J., FORSGREN, D., HALLBERG, G., HOGBERG, J., LARSSON, F., MOESTEDT, A., AND WERNER, B. 2002. Simics: A full system simulation platform. *IEEE Computer 35,* 2, 50–58.

[59] MENTOR GRAPHICS. 2003. Platform express and Primecell. http://www.mentor.com/products/embedded_software/platform_baseddesign/.

[60] NAKAMURA, Y., HOSOKAWA, K., KURODA, I., YOSHIKAWA, K., AND YOSHIMURA, T. 2004. A fast HW/SW co-verification method for SoC by using a c/c++ simulator and fpga emulator with shared register communication. In *Proceedings of DAC*.

[61] NAVA, M. D., BLOUET, P., TENINGE, P., COPPOLA, M., BEN-ISMAIL, T., PICCHIOTTINO, S., AND WILSON, P. 2005. An open platform for developing mpsocs. *IEEE Computer*, 60–67.

[62] PACI, G., MARCHAL, P., POLETTI, F., AND BENINI, L. 2006. Exploring temperature-aware design in low-power mpsocs. In *Proceedings of DATE*.

[63] PAULIN, P., PILKINGTON, C., AND BENSOUDANE, E. 2002. Stepnp: A system-level exploration platform for network processors. *IEEE Design & Test 19,* 6, 17–26.

[64] R. W. FLOYD, E. A. 1985. Adaptive algorithm for spatial gray scale. In *Proceedings of ISDT*.

[65] ROHOU, E. AND SMITH, M. 1999. Dynamically managing processor temperature and power. In *Proceedings of FDO*.

[66] SEMICONDUCTOR INDUSTRY ASSOCIATION (SIA). 2004. The International Technology Roadmap for Semiconductors. http://public.itrs.net/.

[67] SKADRON, K., STAN, M., HUANG, W., VELUSAMY, S., SANKARANARAYANAN, K., AND TARJAN, D. 2002. Thermal-RC modeling for accurate and localized dynamic TM. In *Proceedings of HPCA*.

[68] SKADRON, K., STAN, M. R., SANKARANARAYANAN, K., HUANG, W., VELUSAMY, S., AND TARJAN, D. 2004. Temperature-aware microarchitecture: Modeling and implementation. *Trans. on Architecture & Code Optimizations 1,* 1, 94–125.

[69] SRINIVASAN, J. AND ADVE, S. V. 2003. Predictive dynamic thermal management for multimedia applications. In *Proceedings of ICS*.

[70] SU, H., LIU, F., DEVGAN., A., ACAR, E., AND NASSIF, S. 2003. Full chip leakage estimation considering power supply and temperature variations. In *Proceedings of ISLPED*. 78–83.

[71] SYNOPSYS. 2003. Realview Maxsim ESL environment. http://www.synopsys.com.

[72] VANDEVELDE, B., DRIESSENS, E., CHANDRASEKHAR, A., AND BEYNE, E. 2001. Characterisation of the polymer stud grid array, A lead-free CSP for high performance and high reliable packaging. In *Proceedings of SMTA*.

[73] Butler, D. "Everything, Everywhere". *Nature*, Vol.440, no. 23, pp 402-205, 2006.

[74] Rabaey, J. "Traveling the Wild Frontiers of Ultra Low Power Design". *Keynote IEEE PATMOS*, Belgium, 2005.

[75] Nazhandali, L. et al "Energy Optimization of Subthreshold-Voltage Processors". *proc. ACM/IEEE ISCA*, June 2005.

[76] Raychowdhury, A. et al "Computing With Subthreshold Leakage: Device/Circuit/Architecture Co-Design for Ultralow-Power Subthreshold Operation". *IEEE TVLSI*, Vol. 13, no. 11, pp 1213-1224, Nov., 2005.

[77] Cao, Y. et al "Yield optimization with energy-delay constraints in low-power digital circuits". *proc. IEEE Electronic Devices and Solid-State Circuits*, Dec., 2003.

[78] Wang, A. et al "A 180-mV Subthreshold FFT Processor Using a Minimum Energy Design Methodology". *IEEE Journal of Solid-State Circuits*, Vol.40, no. 1, pp. 310-319, Jan., 2005.

[79] Sylvester, D., Blaauw, D., Karl, E. "ElastIC: An Adaptive Self-Healing Architecture for Unpredictable Silicon". *IEEE D&T*, Vol.23, no. 6, pp. 484-490.

[80] Mani, M., Orshansky, M. "A New Statistical Optimization Algorithm for Gate Sizing". *proc ICCD 2004*, pp. 272-277.

[81] Papanikolau, A. et al. "A system-level methodology for fully compensating process variability impact of memory organizations in periodic applications". *proc CODES+ISSS 2005*, pp. 117-122.

[82] Sparsø, J. et al. "Principles of asynchronous circuit design - A systems perspective". *Kluwer Academic Publishers*, 2001.

[83] Cortadella, J. et al. "Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications.". *IEEE Trans*, Vol. 25, Issue 10, pp 1904-1921, Oct 2006.

[84] Sokolov, D. et al. "Design and Analysis of Dual-Rail Circuits for Security Applications". *IEEE Trans*, Vol. 54, Issue 4 pp. 449-460, Apr. 2005.

[85] Fisher, J.A., Foraboschi, P., Young, C. ”Embedded Computing. A VLIW Approach To Architecture, Compilers and Tools”. *Morgan Kaufmann*, San Francisco, 2005.

[86] D. Sylvester and K. Keutzer, ”Getting to the bottom of deep sub-micron II: A global paradigm”, Proc. IEEE Int. Symp. Physical Design, pp.193-200, 1999.

[87] M.Karlsson, M.Vesterbacka, L.Wanhammar, ”Low-Swing Charge Recycle Bus Drivers”, ISCA ’98, pp.117-120, 1998.

[88] Rjoub, A.; Koufopavlou, O.; ”Efficient drivers, receivers and repeaters for low power CMOS bus architectures”, ICECS ’99, pp.789 - 794 vol.2, 1999.

[89] Byung-Do Yang; Lee-Sup Kim; ”High-speed and low-swing on-chip bus interface using threshold voltage swing driver and dual sense amplifier receiver”, ESSCIRC ’00, pp.105 - 108, 2000.

[90] R. Dobkin, A. Morgenshtein, A. Kolodny, R. Ginosar ”Parallel vs. serial on-chip communication”, SLIP 2008, pp.43-50.

[91] S.Medardoni, M.Lajolo, D.Bertozzi, ”Variation tolerant NoC design by means of self-calibrating links”, DATE’08, pp.1402-1407, 2008.

[92] E.Humenay, D.Tarjan, K.Skadron; ”Impact of process variations on multicore performance symmetry”, DATE ’07, pp.1653 - 1658, 2007.

[93] Garcia, J.C.; Montiel-Nelson, J.A.; Nooshabadi, S.; ”High performance bootstrapped CMOS low to high-swing level-converter for on-chip interconnects”, ECCTD 2007, pp.795 - 798, 2007.

[94] Mensink, E.; Schinkel, D.; Klumperink, E.A.M.; van Tuijl, E.; Nauta, B.; ”Optimal Positions of Twists in Global On-Chip Differential Interconnects”, IEEE Transactions on VLSI Systems, Volume 15, Issue 4, April 2007, Page(s):438 - 446.

[95] Kangmin Lee; Se-Joong Lee; Hoi-Jun Yoo; ”Low-power network-on-chip for high-performance SoC design”, IEEE Transactions on VLSI Systems, Volume 14, Issue 2, pp.148 - 160, 2006.

[96] A.Narasimhan, B.Srinivasaraghavan, R.Sridhar, ”A low-power asymmetric source driver level converter based current-mode signaling scheme for global interconnects”, Int.Conf.on VLSI Design, 4 pp., 2006.

[97] F.Worm, P.Ienne, P.Thiran, G.De Micheli, ”A robust self-calibrating transmission scheme for on-chip networks”, IEEE Trans. on VLSI Systems, pp.126 - 139, 2005.

[98] Jeong, W.; Paul, B.C.; Kaushik Roy; "Adaptive supply voltage technique for low swing interconnects", ASP-DAC 2004, pp.284 - 287, 2004.

[99] Chang-Ki Kwon; Kwang-Myoung Rho; Kwyro Lee; "High speed and low swing interface circuits using dynamic over-driving and adaptive sensing scheme", ICVC '99, pp.388 - 391, 1999.

[100] H.Zhang, V.George, J.M.Rabaey, "Low-swing on-chip signaling techniques: effectiveness and robustness", IEEE Trans. on VLSI Systems, pp.264-272, Vol.8, no.3, June 2000.

[101] von Arnim, K.; Borinski, E.; Seegebrecht, P.; Fiedler, H.; Brederlow, R.; Thewes, R.; Berthold, J.; Pacha, C.; "Efficiency of body biasing in 90-nm CMOS for low-power digital circuits", IEEE Journal of Solid-State Circuits, Volume 40, Issue 7, July 2005, Page(s): 1549 - 1556.

[102] Venkatraman, V.; Anders, M.; Kaul, H.; Burleson, W.; Krishnamurthy, R.; "A Low-swing Signaling Circuit Technique for 65nm On-chip Interconnects", International SOC Conference, pp.289 - 292, 2006.

[103] Garcia, J.C.; Montiel-Nelson, J.A.; Nooshabadi, S.; "High performance CMOS symmetric low swing to high swing converter for on-chip interconnects", IEEE Northeast Workshop on Circuits and Systems, 2007, pp.566 - 569, 2007.

[104] Joonsung Bae; Joo-Young Kim; Hoi-Jun Yoo; "0.6pJ/b 3Gb/s/ch transceiver in 0.18 um CMOS for 10mm on-chip interconnects", ISCAS 2008, pp.2861 - 2864, 2008.

[105] Meijer, M.; Pessolano, F.; Pineda De Gyvez, J.; "Technology exploration for adaptive power and frequency scaling in 90nm CMOS", ISLPED '04, pp.14 - 19, 2004.

[106] Tschanz, J.; Kao, J.; Narendra, S.; Nair, R.; Antoniadis, D.; Chandrakasan, A.; Vivek De; "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage", IEEE Journal of SSCs, pp.1396 - 1402, vol.37, no.11, 2002.

[107] Tschanz, J.W.; Narendra, S.; Nair, R.; De, V.; "Effectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors", IEEE Journal of SSCs, pp.826 - 829, vol.38, Issue 5, 2003.

[108] Bonesi S., Bertozzi D., Benini L., Macii E., "Process variation tolerant pipeline design through a placement-aware multiple voltage island design style", DATE 2008, pp.967 - 972, 2008.

[109] Chen, T.; Naffziger, S.; ”Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation”, IEEE Transactions on VLSI Systems, Volume 11, Issue 5, Oct. 2003, Page(s):888 - 899.

[110] Gregg, J.; Chen, T.W.; ”Post silicon power/performance optimization in the presence of process variations using individual well adaptive body biasing (IWABB)”, 5th International Symposium on Quality Electronic Design, Page(s):453 - 458, 2004.

[111] I.Hatirnaz, S.Badel, N.Pazos, Y.Leblebici, S.Murali, D.Atienza, G.De Micheli; ”Early wire characterization for predictable network-on-chip global interconnects”, SLIP’07, Page(s):57-64, 2007.

[112] D.Bertozzi, L.Benini, B.Ricc, ”Parametric timing and power macromodels for high level simulation of low-swing interconnects”, ISLPED 2002: pp.307-312.

[113] A.Papanicolaou et al., ”At Tape-out: Can System Yield in Terms of Timing/Energy Specifications be Predicted?”, IEEE Custom Integrated Circuits Conference, pp.773-778, 2007.

[114] E.Humenay, D.Tarjan, K.Skadron; ”Impact of Parameter Variations on Multi-Core Chips”, Int.Workshop on Architectural Support for Gigascale Integration, 2006.

[115] H. C.Wan et al., ”Channel doping engineering of MOSFET with adaptable threshold voltage using body effect for low voltage and low power applications”, Int. Symp. VLSI Technology, Systems, and Applications, 1995, pp.159-163