Research Article

Automated High-level Modeling of Power, Temperature and Timing Variation for Microprocessor

Zheng Wang*
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Shazia Kanwal
Thai-German Graduate School of Engineering, King Mongkut's University of Technology North Bangkok, Bangkok, Thailand

Lai Wang
Analog Devices, Beijing, China

Anupam Chattopadhyay
School of Computer Science and Engineering, Nanyang Technological University, Singapore

* Corresponding author. E-mail: wangz@ntu.edu.sg

DOI: 10.14416/j.ijast.2017.08.002
Received: 10 October 2016; Accepted: 18 April 2017; Published online: 10 August 2017

Abstract

The continued scaling of semiconductor technologies leads to diverse challenges such as power and temperature, which also forces reliability as another design metric of prime concern. There exists strong need to link reliability with physical metrics in a high-level architecture design environment, where estimation of reliability impacts can be performed in the early design stage. In this paper, we propose a joint modeling and simulation framework for power, thermal and timing variation, which is integrated into a commercial high-level processor design environment. A custom timing variation model is provided for estimation of dynamic timing variation, which is demonstrated using one nanoscale thermal effect known as Inverted Temperature Dependence. The complete modeling flow is automated for customized processor model with arbitrary architectural hierarchy, which assists designer to perform architectural and application-level design space exploration with power, thermal and reliability impacts.

Keywords: Power modeling, Thermal estimation, Timing variation, Simulation

1 Introduction

As reliability becomes an essential factor in the design of nanoscale digital system, it is important to integrate reliability as a design constraint in the traditional processor design flow, where instruction-set simulator plays an important role in architecture validation and performance estimation. Reliability effects, especially aging and soft errors, have direct relationship with other design parameters such as runtime, power and temperature. There is strong need to link reliability with other physical metrics in a high-level processor design environment, where realistic estimation of reliability effects can be simulated together with power and thermal footprints.

Processor power estimation techniques have been

Please cite this article as: Z. Wang, S. Kanwal, L. Wang, and A. Chattopadhyay, "Automated high-level modeling of power, temperature and timing variation for microprocessor," *KMUTNB Int J Appl Sci Technol*, vol. 10, no. 3, pp. 163–175, Jul.–Sep. 2017.

continuously a hot topic in both research and industry. Instruction level power model is proposed by Tiwari *et al*. [1], [2], where each instruction is provided with an individual power model. The run-time power can be determined through the profiling of executed instructions. Wattch [3] introduces architecture-level power model which decompose main processor units into categories based on their structures, and separates each of the units into stages and forms RC circuits for each stage. McPAT [4] models all dynamic, static and short-circuit power while provides joint modeling capability of area and timing. To increase modeling accuracy, a hybrid FLPA(functional level power analysis) and ILPA(instruction level power analysis) model [5] is elaborated which advantageously combines the lower modeling and computational efforts of an FLPA model and the higher accuracy of an ILPA model. The trade-off is further explained in [6] with a 3-D LUT and a tripartite hyper-graph.

The heat dissipation from power consumption leads to increased and un-evenly distributed temperature which causes potential reliability problems [7], [8], where the research committee demands highly for architecture-level thermal management techniques. Consequently, accurate architecture level thermal modeling has received huge interests. In this domain, HotSpot [9] is the de facto standard, where the thermal effects for individual architecture blocks can be fast estimated by creating a thermal RC network. HotSpot is easy to integrate with any source level power simulator, which spreads its appliance into huge research bodies [10], [11]. SUNRED [12] and 3D-ICE [13] are other prevalent thermal simulators, which can also simulate microfluidic cooling by modeling convective heat transfer.

Recently, there is an emerging research trend for logic-thermal co-simulation in the field of digital system. The logic-thermal simulation principle is first proposed in [14]. For gate-level, [15], and [16] introduce methodology for temperature dependent timing simulation in standard cell designs. To bridge the gap between simulation accuracy and complexity of modern digital system, several work performs logic-thermal simulation at higher design abstractions. Cacti [17] estimates power, area and timing specifically for memory system. McPAT [4] jointly models power, area and timing for individual system-level blocks including cores and memories. [18] applies a joint performance, power and thermal simulation framework for the design of network-on-chip. [19] extends the work with the ability to simulate optimization techniques such as Dynamic Voltage Frequency Scaling (DVFS) and Power Gating. [20] develops a logic-thermal simulation engine for circuit descriptions of multiple abstraction levels.

However, the previous work simulates the physical behaviors using off the shelf libraries on a higher abstraction level for individual blocks, which did not address to the complexity of processor architecture itself. On the other hand, accurate power modeling, which is the centralized module of the joint simulation, requires significant efforts due to manual characterization. An Application-Specific Integrated Processor (ASIP) can have arbitrary logic blocks which need detailed block level modeling. Previous work also lacks the ability to accurately estimate power/temperature with application specific switching activities. The reason is that modeling and simulation are treated as separate issues, where the modeling part is more tent to be provided from IP vendors as technology dependent databases. Furthermore, to the best knowledge, no work has been tried to integrate reliability issue directly into the high-level simulation framework. Such issues are still open to be addressed.

Contribution In this work, a joint modeling framework which integrating power, thermal and logic delay simulation is introduced in a industrial processor design environment, where both accurately modeling through low-level characterization and cross-domain simulation using cycle accurate instruction-set simulator are fast generated. The main contributions include:

- A processor power modeling technique which characterizes and estimates power for customized processor model with arbitrary architectural hierarchy.
- A processor thermal simulator which is efficiently achieved by integrating power simulator with HotSpot.
- A timing simulator for logic paths extended from the processor fault injector, where faults are modeled as delay variation on logic paths.
- Estimation of processor timing variation under one of the nanoscale thermal induced reliability issues.

By automating the complete modeling and simulation flow, the processor designer can easily perform architectural and application-level design space exploration with power, tempera-ture and reliability factors.

The work is organized in following manner. Section 2 briefly discusses the approach of high-level power modeling and estimation for LISA based processor design framework. Section 3 illustrates the thermal modeling and integration with HotSpot. After that, Section 4 presents the extension to the high-level fault injection with timing fault simulation. Taking advantage of fault injector, Section 5 introduces the approach of high-level logic delay simulation. Section 6 focuses on the automation flow and analyses its runtime overhead. Finally, the paper is concluded in Section 7.

2 High-level Processor Power Modeling

This section briefly introduces the proposed high-level power modeling methodology. First, an overview on the LISA language is present. Second, an overall introduction on the power estimation flow is illustrated. The detailed experiments of proposed power modeling technique on embedded processors are referred in [21].

2.1 *Brief on LISA language*

LISA (Language for Instruction Set Architectures) is the state-of-the-art Architecture Description Language (ADL) for describing the micro-architecture of customized processors [22]. In LISA, the micro-architecture and Instruction Set Architecture (ISA) are described in the OPERATION section. The declarations for processor resources such as pipeline stages, registers, memories and ports are located in the RESOURCE section and they can be accessed from any LISA operation. Structural information can be added by assigning the operations to different pipeline stages. C-compiler, assembler, linker and cycle-accurate instruction set simulator are automatically generated by LISA compiler, while synthesizable RTL codes with different options of optimizations are produced by the HDL generator. For further information on LISA language please refer to [23].

2.2 *Power estimation flow*

High-level power estimation flow characterizes power models for LISA units (operations and resources) from low level power simulation. Such power models are applied later in instruction-set simulator to produce run-time power for targeted applications. The simulation is independent of gate-level power simulation so that significant efforts are saved. The simulation accuracy depends on the efforts in the characterization of power model.

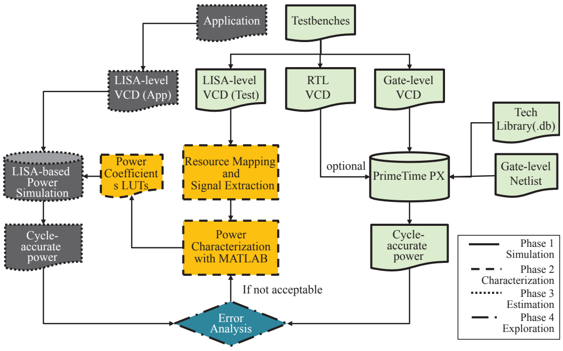

Figure 1: LISA-based power modeling and simulation flow.

Figure 1 explains the proposed power estimation flow which consists of simulation, characterization, estimation and exploration phases.

a) Simulation: High level power models are usually characterized by the data from low level simulation. In this phase, cycle accurate power data for special testbenches are gathered according to PrimeTime-based power simulation which can be performed at either RTL, gate-level or layout. Currently gate level is chosen for trade-off between simulation accuracy and modeling efforts. Each simulation testbench consists of single type of instructions such as ALU, Load/Store and Branch types. The testbenches are composed in such a way that operands and immediate values are randomly distributed. Special instruction features such as operand bypassing are also covered according to the target architecture. Larger coverage of instruction modes and operand values leads to enhanced accuracy of power modeling.

b) Characterization: Besides cycle accurate power data, the inputs of characterization phase also take into account the switching activities from instruction-set simulation. Both data are used to characterize the coefficients for the unit level power models which are detailed in the next section. Multivariate curve fitting technique is applied for extraction of coefficient values. Exploration between accuracy and extraction effort of coefficients can be explored through linear and polynomial modes in curve fitting.

c) Estimation: The power models are applied on target applications to estimate power consumption. The run-time switching activities from cycle accurate Instruction Set Simulator (ISS) are gathered and provided to the power simulator. Due to the nature of unit-based power modeling, instantaneous power consumption for each hardware unit is calculated and recorded in the PrimeTime recognized power format.

d) Exploration: The ISS-level run-time power is compared with low level simulation to determine the estimation accuracy. Based on the user requirements the unit-based power models can be further improved taking advantage of the techniques in the simulation and characterization phases.

3 High-level Processor Thermal Modeling

This section illustrates the integration of HotSpot thermal simulator with proposed power modeling technique to achieve a fast and dynamic thermal simulator for processors with generic architectures.

3.1 *Thermal modeling using HotSpot*

HotSpot is an opensource package for temperature estimation of architecture-level units. It has been applied in both academia and industry for architecture-level thermal modeling and management. HotSpot is easily integrated into any performance/power simulator by providing the floorplan and instantaneous power information. By transforming the floorplan into an equivalent thermal RC circuits which is called compact models, HotSpot calculates instantaneous temperature by solving the thermal differential equation using a fourth-order Runge-Kutta method. The temperature for each block is updated by each call to the RC solver. For details of applying HotSpot for thermal modeling please refer to [10].

3.2 *Integration of power simulator with HotSpot*

The integration of LISA power simulator with HotSpot generally follows the guideline in [24]. Two phases are required, the initialization and runtime phases, which are briefly explained in the following.

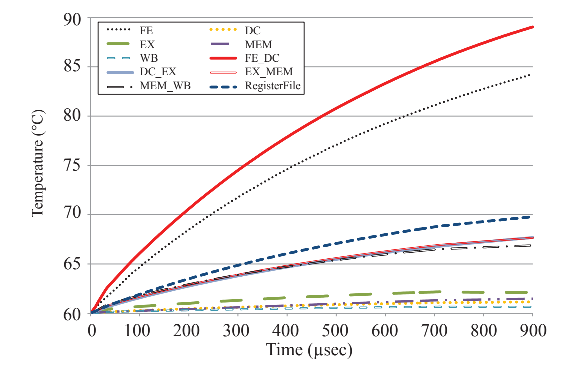

Figure 2: Instantaneous temperature generated by HotSpot.

- The initialization phase, where the RC equivalent circuits are first initialized based on user provided floorplan and thermal configurations, such as parameters for heat sink and heat spread. Afterwards, the initial temperature is set by the user. For instance, 60 degree is initialized for starting temperature while 45 degree is set as ambient temperature. The floorplan information of can be obtained from commercial physical synthesis tool such as Cadence SoC Encounter or derived according to the area report from logic synthesize.
- The runtime phase, where the simulator iteratively calls the temperature computing routine to update the block temperatures. Such routine does not need to be called during each clock cycle due to the nature of slow changing temperature. In practice, a sampling interval of 10 Kilocycles at 3 GHz is adapted, which corresponds a time of 3.33 microseconds. For different clock frequency, the same interval is maintained to make fair comparison. The power values which provide to HotSpot are the average values among the previous sampling interval.

3.3 *Temperature simulation and analysis*

Figure 2 shows an example of the runtime temperature simulation for BCH application under a synthesis frequency of 500 MHz. The unit of time is in nanosecond while the temperature is in Degree Celsius. Instead of the whole simulation time, a snapshot has been shown.

Table 1: Temperature and power of LT_RISC at different frequencies running BCH application

Frequencies	**25 MHz**		**100 MHz**		**500 MHz**	
Units	**Temp (°C)**	**Power (mW)**	**Temp (°C)**	**Power (mW)**	**Temp (°C)**	**Power (mW)**
FE	63.90	3.19e-3	69.39	9.08e-3	84.25	3.88e-2
DC	60.16	2.56e-2	60.43	7.18e-2	61.15	2.99e-1
EX	60.23	4.91e-2	60.60	1.40e-1	62.11	7.06e-1
MEM	60.17	3.88e-3	60.63	9.53e-3	61.48	3.72e-2
WB	60.05	2.69e-3	60.20	8.47e-3	60.66	3.86e-2
FE_DC	62.16	2.71e-2	68.44	1.05e-1	89.04	4.93e-1
DC_EX	60.74	5.72e-2	62.66	2.08e-1	67.68	1.08
EX_MEM	60.74	4.09e-2	62.40	1.50e-1	67.66	7.61e-1
MEM_WB	60.45	2.32e-2	61.74	8.73e-2	66.89	4.36e-1
RegisterFile	61.09	2.39e-1	63.25	7.54e-1	69.79	3.52

Table 2: Temperature of LT_RISC running BCH application using different floorplans

Units	**Power @500MHz (mW)**	**Floorplan 1**			**Floorplan 2**		
		Size (mm^2)	**Power Density (W/m^2)**	**Temp (°C)**	**Size (mm^2)**	**Power Density (W/m^2)**	**Temp (°C)**
FE	3.88e-2	0.01	4.95	84.25	1.00	0.04	60.19
DC	2.99e-1	1.24	0.24	61.15	1.24	0.24	61.15
EX	7.06e-1	1.50	0.47	62.11	1.50	0.47	62.11
MEM	3.72e-2	0.28	0.13	61.48	0.28	0.13	61.37
WB	3.86e-2	0.27	0.14	60.66	0.27	0.14	60.66
FE_DC	4.93e-1	0.08	6.03	89.04	1.00	0.49	62.40
DC_EX	1.08	0.69	1.57	67.68	0.76	1.43	67.02
EX_MEM	7.61e-1	0.48	1.59	67.66	0.49	1.55	67.42
MEM_WB	4.36e-1	0.26	1.65	66.89	0.30	1.45	66.17
RegisterFile	3.52	1.74	3.52	69.79	2.25	2.02	67.57
Total	-	6.55	-	-	9.08	-	-

Table 1 shows the temperature and power consumption for architectural units with different design frequencies, where BCH application runs on the processor. The same floorplan is applied for all frequencies. As the power increases dramatically with frequency, the temperature shows slightly increment for most of the units such as DC, MEM and WB. Rapid increment lies between 100 MHz and 500 MHz for units FE and FE DC, even though their power consumptions are relatively small compared with other units. On the contrary, units such as RegisterFile which incur higher power consumption shows only a slight increment in temperature. The reason behind this is the high power density on such units due to their small area provided by the floorplan.

Table 2 shows the temperature for BCH application using different floorplans. The first floorplan adopts the ratio of unit size from logic synthesis tools. However, the runtime tempera-ture shows strong differences among different architectural units, which has the potential to incur temperature related reliability issues. Floorplan 2 tries to increase the sizes of units with high power density (FE, FE DC and RegisterFile units) so that the power density will be significantly reduced. As seen from the thermal simulation, the temperature of hot units reduces dramatically so that the thermal footprints of pipeline registers and RegisterFile are finalizing at similar values.

To prevent large area overhead, a slight increment to the area of registers is introduced due to their initially large size. The area of FE, which is initially very small (0:01 mm^2), is increased 100x to achieve uniform temperature for all logic between pipeline stages. Overall a 38.6% area overhead is incurred to achieve

Table 3: Temperature of LT_RISC at 500 MHz for different applications

Units	Block-level Temperature (°C) for Different Applications									
	bch	**cordic**	**crc32**	**fft**	**idct**	**median**	**qsort**	**sieve**	**sobel**	**viterbi**
FE	84.25	60.38	61.10	60.57	61.70	73.67	72.62	61.27	60.73	72.65
DC	61.15	60.02	60.06	60.03	60.09	60.70	60.65	60.06	60.04	60.69
EX	62.11	60.05	60.15	60.06	60.20	61.50	61.51	60.07	60.10	61.69
MEM	61.48	60.03	60.07	60.03	60.09	60.94	60.95	60.02	60.04	60.80
WB	60.66	60.02	60.05	60.02	60.06	60.50	60.52	60.03	60.03	60.48
FE_DC	89.04	60.44	61.29	60.66	62.04	76.01	75.13	61.51	60.86	75.50
DC_EX	67.68	60.12	60.34	60.17	60.55	64.25	64.02	60.37	60.23	64.06
EX_MEM	67.66	60.12	60.36	60.18	60.54	64.32	64.14	60.38	60.25	64.22
MEM_WB	66.89	60.14	60.39	60.19	60.57	64.42	64.31	60.39	60.27	64.22
RegisterFile	69.79	60.15	60.44	60.22	60.70	65.33	65.17	60.48	60.30	65.10
Finish Time (μs)	900.2	6.7	20.0	10.0	33.3	350.1	333.4	23.3	13.3	320.1

the thermal footprints where all units show temperature under 68 degree. In other word, the improved floorplan reflects a maximal power density around 2.00 W=m^2 for arbitrary logic units. According to the relationship of temperature with power density, thermal optimization techniques could be investigated taking advantage of the fast estimation framework. Detailed optimization techniques will be investigated in future work.

Table 3 shows the temperature of processor units by end of the simulation time for 10 embedded benchmarks using the initial floorplan. The temperature differs among applications mainly due to the difference in execution time of the applications. For instance the BCH application which runs for 900 μs is significantly hotter on most of the units than other short applications. For applications with similar execution time such as CRC32 and Sieve, no huge differences in temperature among all units is detected. Note that change in temperature is a slow process compared with power consumption, where application dependent thermal effects will exhibit for long execution time. For instance, with 91.4% execution time of median application, viterbi achieves a slightly higher temperature in EX units, which is due to the nature of more ALU instructions. Assembly level profiling shows that viterbi incurs 59,739 ALU instructions (37.12% of all instructions) while median has the amount of 46,301 (26.39% of all instructions), which verifies viterbi's hotter temperature in EX pipeline unit than that for median.

4 High-level Processor Timing Fault Simulation

The LISA-based Fault Injection (FI) framework developed using Synopsys Processor Designer in [25] is utilized to simulate the behavior of cycle accurate processor models under faults. The framework supports standard fault models such as bit-flip and stuck-at faults. All the hardware resources such as registers, memories, global/local signals and interface pins are exposed to fault injection. The fault configuration file can be generated from a graphical user interface or directly written in XML format. Taking advantage of application programming interface associated with Processor Designer [22], a runtime scheduler injects the faults according to the injection clock cycles and duration provided by the fault configurations. In [25] it is reported that the high-level simulator achieves 10x simulation speed compared with Verilog-based fault injection [26] with similar accuracy.

However, such framework lacks the ability for physical timing simulation, which is usually acquired after simulation on post-layout netlist. To integrate low-level timing as a constraint for fault injection, LISA-based processor simulator is extended with timing annotation for logic paths, which are extracted from the timing analysis files. Such annotated path timing will be compared with runtime clock period, so that delay faults can be injected. The simulation kernel is extended by the modules in Figure 3, which are briefly discussed in the following.

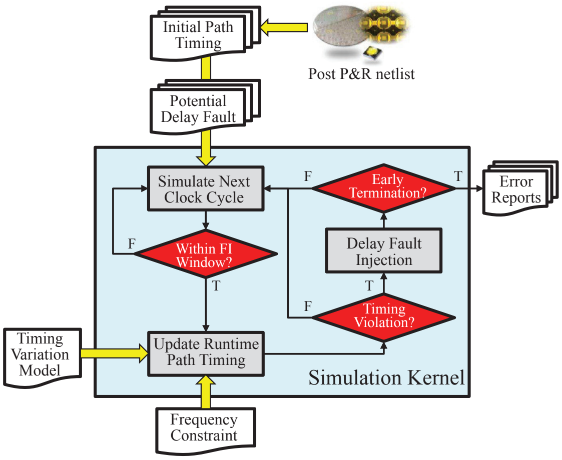

Figure 3: Simulator extension for timing faults.

- **Initial path timing**: It indicates the bit-level logic delay from initial flip-flop to the end flip-flop of a logic path, which are extracted during logic synthesis or placement and routing using timing analysis. Such delay information is back-annotated as extra information for the hardware resources in instruction-set simulator. For instance, Static Timing Analysis (STA) [27] calculates one timing value for each flop-to-flop logic path. Dynamic Timing Analysis (DTA) [28] will create a set of timing values for each path, according to instruction and operand types. Such values are kept in a Look-Up-Table (LUT) for runtime addressing.
- **Timing variation model**: which updates the runtime delay based on initial delay and specific timing variation function. For instance, temperature aware timing variation function is provided when the path timing is changed through temperature and time. In DTA, this could be the runtime state information from the executed instruction within each pipeline stage, used to select the corresponding entry in the path timing LUT.
- **Frequency constraint**: which is provided by the user to compare with the runtime delay, so that a fault could be injected. A runtime adjustable frequency can be applied to model Dynamic Frequency Scaling (DFS). Voltage variation can also be modeled by a fixed frequency with corresponded frequency swing.

For each simulation clock cycle, the simulator first updates the clock cycle time using the initial delay and timing variation function for all annotated paths. In the next, the simulator checks timing violation for all annotated logic paths. In case there is a timing mismatch, the simulator overwrites the current value in the target resource by a random value which is either zero or one to model metastability or the value from previous clock cycle to model a failed latching of logic value. Otherwise, the simulator stores the current resource values which may be used thermal effects on threshold voltage dominates the delay change. as fault injection value for the following clock cycles.

The annotation of timing faults incur extra simulation over-heads. Such overhead is aggregated when more hardware sources are prone to timing faults. To address the speed issue, two special features are implemented which are explained in the following.

- **Fault time window**: In most case the timing faults are only injected within specific time windows. For instance, the boot-up code is usually skipped for FI when purely application-level error effects are focused on. FI window is also useful when vulnerability of different program segments need to be investigated. User can specify fault time windows by either clock cycles or debugging flags in the application.
- **Early termination**: Timing faults tend to disturb program execution flow, usually due to the error in branch address calculation. In most case the wrong branch will lead the processor into a deadlock state. We detect such deadlock situation by checking the value of instruction register of the processor. In case that among 100 continuous clock cycles the processor executes the same instruction, the simulation is forced to terminate earlier than the complete simulation time with a no response error report.

5 Thermal-aware Logic Delay Simulation

The effects of temperature on the logic delay of nanoscale CMOS technology have been heavily investigated such as Negative-Bias Temperature Instability (NBTI) [7] and Inverted Temperature Dependence (ITD) [8]. Most of previous work focus on device and gate-level. Such effects can be modeled using the architectural level thermal simulation framework

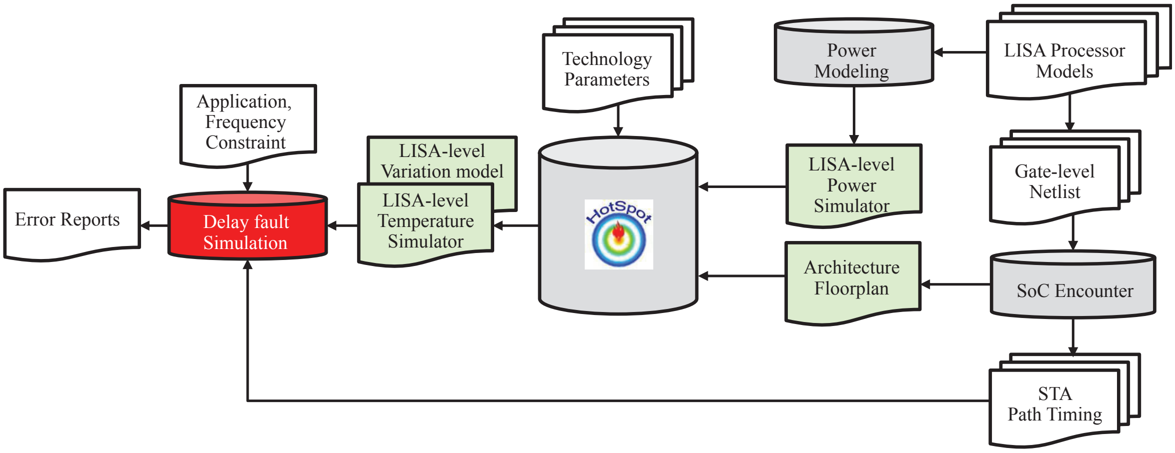

Figure 4: High-level thermal-aware fault injection.

proposed in this work, so that an thermal-delay simulator for generic processor architecture could be easily generated and explored.

Figure 4 shows the integration framework with power and thermal simulator to model the delay fault. As discussed in Section 3, the LISA-level temperature simulator is generated using power simulator, HotSpot package and architectural floorplan. Thermal directed delay fault is modeled combining the thermal simulator and the high-level timing fault injection discussed in Section 4, where the runtime delay of individual logic paths is updated using temperature and a user provided delay variation model. In this section, the effects of delay change with temperature are modeled according to a second order polynomial model for 65 nm technology. The effect of ITD for different applications running on a RISC processor is also presented.

5.1 *Inverted temperature dependence*

Propagation delay of CMOS transistor is widely modeled using the *Alpha-power* law [29] as:

$$Delay \propto \frac{C_{out}V_{dd}}{I_d} = \frac{C_{out}V_{dd}}{\mu(T)(V_{dd}-V_{th}(T))^{\alpha}}$$

where C_{out} is the load capacitance, α is a constant, $\mu(T)$ is the temperature dependent carrier mobility, $V_{th}(T)$ is the temperature dependent threshold voltage. The temperature affects the delay in two ways: at high voltage V_{dd}, delay is less sensitive to the term $V_{th}(T)$ but to the mobility, while at low temperature the thermal effects on threshold voltage dominates the delay change. As a consequence, for advanced technology which has small driving voltage, the increment in temperature could reduce the propagation delay rather than increase it for technologies with higher voltage. Such effect is named as Inverted Temperature Dependence (ITD) and the voltage which inverts the trend of thermal dependent, is the Zero-Temperature Coefficient (ZTC) voltage.

5.2 *Timing variation function for inverted temperature dependence*

The effects of ITD for 65 nm technology are modeled using the trend of delay change for clock tree network in [30]. Two assumptions are made to simplify the high-level modeling:

1) The delay of logic path follows the same ration of temperature/voltage dependency of individual logic buffer.

2) The temperature within one architecture block is uniform.

3) Other thermal effects on the change of threshold voltage such as NBTI is not modeled currently.

Figure 5 shows two critical paths for he RISC processor and their transverse architectural blocks, which are generated by the STA tools. The delay of the complete logic path equals to the sum of path delay of on the path. For instance, the critical operands from pipeline register and RegisterFile transverse in order

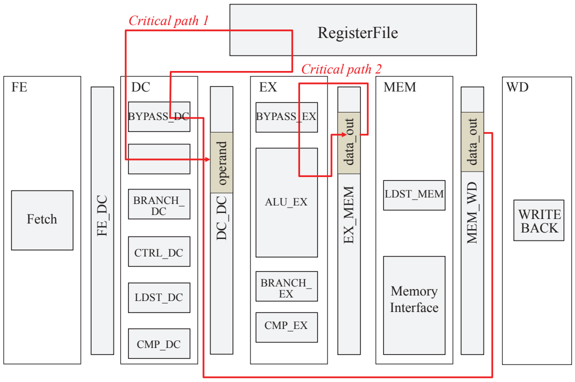

Figure 5: Critical paths and transverse blocks.

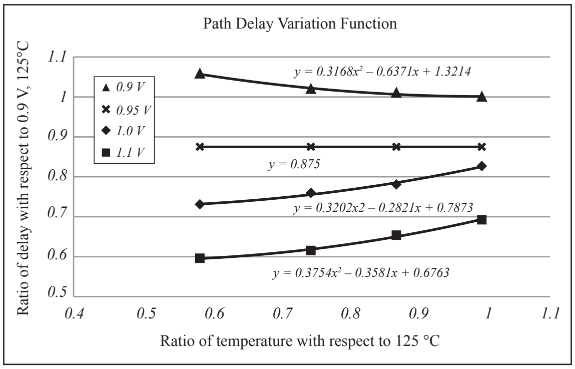

Figure 6: Delay variation function under several conditions.

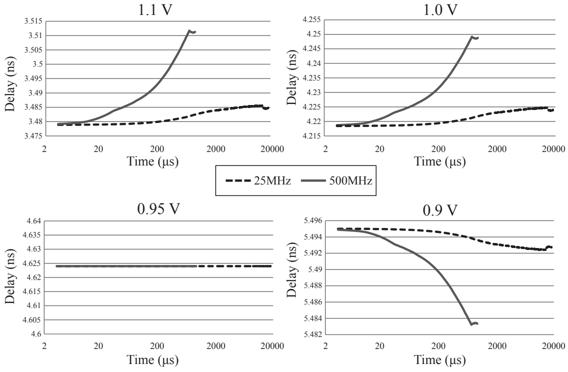

Figure 7: Runtime delay of critical path for BCH application.

the following block: *MEM_WB*, *DC*, *BYPASS_DC*, *DC*, *RegisterFile*, *DC*, *ALU_DC*, *DC*, *DC_EX*. The critical path 2 transverses *EX_MEM*, *EX*, *BYPASS_EX*, *EX*, *ALU_EX*, *EX*, *EX_MEM*. The delay within individual architectural units are updated using its own running temperature, which is generated from the thermal simulation. In extreme case, each cell uses its own running thermal footprints to update its delay, which can only be simulated using gate-level thermal analysis.

With the above assumption and the referred data for 65 nm technology in [30], the second order polynomials shown in Figure 6 are interpolated to represent the relationship between supply voltage, instantaneous temperature and propagation delay. Due to unavailability of ITD-induced timing behaviors under other process technologies, no estimation has been performed for other process corners.

It is observed that the trend of propagation delay with temperature differs with supply voltage. For 1.0 V and 1.1 V the delay increases with temperature while decreases at 0.9 V. In [30] the ZTC voltage is known to be 0.95 V for 65nm technology from STMicroelectronics, which proves the effect of ITD for advanced technology.

5.3 *Case study for ITD simulation*

The polynomials are used as the path timing variation models for the RISC processor and test the change of critical path running embedded applications. Figure 7 shows the runtime delay of the critical path for the RISC processor running BCH application. Curves are plotted for both frequency of 25 MHz and 500 MHz. The supply voltage is simulated using 0.9 V, 0.95 V, 1.0 V and 1.1 V. The initial delay of critical path extracted out of the timing analysis tool is for the worst case condition under 125°C, 0.9 V. It is observed that for high supply voltage such as 1.1 V and 1.0 V, the delay increases with temperature till a saturation point then slightly decrease according to the characteristics of the application. For low voltage of 0.9 V, the inverse trend is shown where the delay decreases with temperature till the saturation point and then slightly increases. Under the ZTC voltage which is 0.95 V, the delay is not effected by the temperature as expected. The effect of ITD shows potential of frequency overscaling under lower voltage, which is predicted for 65 nm and further technologies in [31]. With regard to different running frequencies, the processor running at 500 MHz consumes higher power which leads to higher temperature compared to the data at 25 MHz. Consequently, the speed of delay change shows more significant dependence on temperature for higher frequencies.

5.4 *Limitation on NBTI simulation*

NBTI is known as the most serious aging problem of nanoscale CMOS technology. To accurately model NBTI effect, not only instantaneous temperature but also switching activity for individual logic cells need to be carefully handled. However, in a high-level processor design environment, cycle-accurate switching information for individual cells are not present before technology mapping. Using the proposed framework, it can be demonstrated that it is incorrect to use one unique NBTI-induced timing variation function to estimate the aging of entire logic path. The reason is that NBTI affects timing of connected cells in alternate stress and release modes instead of stress/release of the entire logic path. An example is that one stressed inverter releases its connected inverter. The high-level estimation framework lacks the ability of cell-level instantaneous switching analysis, which gives an extremely pessimistic estimation of NBTI-induced aging effect. On the other hand, ITD-induced delay variation has less effect with cell activities, which can be approximated using high-level design environment.

6 Automation Flow and Overhead Analysis

In this section the purposed automated estimation flow for Power/Thermal/Delay is briefly documented, which functions as an simulator wrapper to the Synopsys Processor Designer [22]. Furthermore, the overheads for both characterization and simulation are discussed.

6.1 *Flow summary*

Figure 8 illustrates the complete analysis framework, where the architecture description and application of interests are provided as inputs. The framework consists of characterization and simulation phase. The power characterization phase consists of 4 modules, which are briefly explained:

a) Testbench generation: is used to generate processor specific testbenches for power characterization. This module parses the syntax section of processor description to produce instructions with random operands. One testbench is generated for each type of instruction, which runs for a predefined simulation clock cycles.

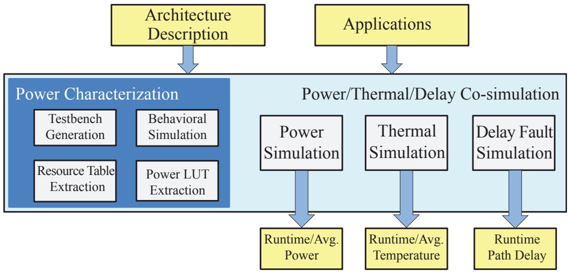

Figure 8: Automation flow of power/thermal/logic delay co-simulation.

b) Resource table extraction: gets the hierarchical information of the architecture and extracts input and output signals for each architecture unit. Read and write power models in the form of interpolated polynomial will be generated to each unit.

c) Behavioral simulation: dumps the runtime hamming distance of input/output signals per architecture unit, which is used for power coefficient extraction.

d) Power LUT extraction: interpolates power coefficients in the form of LUT using hamming distance and data from low-level power simulation. The interpolation itself is carried out using Matlab tool.

e) Power simulation: takes loops to simulate processor behavior and power consumption until end of the simulation cycles. In each control step the simulator calculates power consumption based on the architecture unit specific instruction type, runtime hamming distances of the pins and power coefficient of the architecture units. Instead of list based implementation of power LUT, hash container is applied to increase the speed of instruction-architecture specific LUT addressing. The hierarchical power data for the targeting architecture is dumped during simulation. More modeling architecture units lead to higher overhead of power estimation.

f) Thermal and delay simulations: are automatically generated once upon power simulator is ready, since no further characterization steps is required for thermal and delay simulation.

The proposed flow is demonstrated using Synopsys Processor Designer and is portable to any high-level architecture simulation environment and architectures. Further work includes the porting of the framework into other ADL such as SystemC.

6.2 *Overhead analysis*

Table 4 shows the timing and accuracy for power characterization phase under two groups of testbenches, where 10 architecture units are modeled. The first group consists of 14 types of instructions to cover the most generalized processor instructions. For instance, ALU instructions such as add, sub and and which operate on 2 register operands and 1 immediate are grouped together in one instruction type. The second group consists of 33 types of instructions where each instruction type consists of exact one operational mode. The characterization is performed on the machine with Intel Core i7 CPU at 2.8 GHz. Each instruction file is running for 2,000 clock cycle.

Table 4: Time and accuracy of power characterization for testbench groups

Number of Testbenches	14 Instructions	33 Instructions
Time (minutes)	3	8
Average Error (%)	21.3	8.6

As shown in the Table 4, group one achieves faster characterization time than group two. However, group two achieves higher estimation accuracy when benchmarked with gate-level power estimation. Generally, the power characterization time in the range of several minutes is acceptable for power modeling of embedded processors.

Table 5 represents the runtime overhead of different simulation mode including pure behavioral simulation, power estimation, thermal estimation and delay simulation, where 10 architecture units are modeled. It is observed that the runtime overhead significantly lies in the power estimation compared with behavioral simulation. The thermal simulator achieves only 1.2% of overhead compared with power simulator, which is due to the light weight implementation of HotSpot package and smooth integration with power simulator. The delay simulation achieves in average 6.3% of overhead compared with thermal simulator, which is mainly due to the parsing of delay information from timing analysis file which contains delay of the longest 1,000 paths.

Table 5: Runtime overhead for different simulation modes

Apps	Behavior (sec)	Power (sec)	+%	Thermal (sec)	+%	Delay (sec)	+%
BCH	2.04	124.94	61x	125.47	0.4	129.72	3.4
Viterbi	0.82	43.49	53x	44.37	2.0	47.86	7.9
Median	0.87	49.40	57x	49.45	0.1	53.00	7.2
Qsort	0.81	45.45	56x	46.65	2.6	48.53	4.0
IDCT	0.19	5.17	27x	5.22	1.0	5.69	9.0
Average	-	-	51x	-	1.2	-	6.3

7 Conclusions

In this work, a processor power/thermal/timing variation joint modeling framework is presented for LISA-based processor design environment. Detailed experiments are conducted which explore the usability of the framework with several design parameters such as applications, technologies and layouts. An automatic setup has be constructed which performs estimation and analysis according to such parameters. The proposed framework helps processor designer to explore the physical effects in early design stage.

Future work includes the modeling of process and aging induced timing variation. Application-level error analysis caused by physical effects will be explored. Future case studies involve complex architectures will also be presented.

References

[1] V. Tiwari, S. Malik, and A. Wolfe, "Power analysis of embedded software: A first step towards software power minimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437–445, 1994.

[2] V. Tiwari, S. Malik, A. Wolfe, and M. Tien-Chien Lee, "Instruction level power analysis and optimization of software," *The Journal of VLSI Signal Processing*, vol. 13, no. 2, pp. 223–238, 1996.

[3] D. Brooks, V. Tiwari, and M. Martonosi, "Watch: A framework for architectural-level power analysis and optimizations," in *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, 2000, pp. 83–94.

[4] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M.

Tullsen, and N. P. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proceedings 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2009, pp. 469–480.

[5] H. Blume, D. Becker, M. Botteck, J. Brakensiek, and T. Noll, "Hybrid functional and instruction level power modeling for embedded processors," in *Proceedings Embedded Computer Systems: Architectures, Modeling, and Simulation*, 2006, pp. 216–226.

[6] Y. Park, S. Pasricha, F. Kurdahi, and N. Dutt, "A multi-granularity power modeling methodology for embedded processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 19, no. 4, pp. 668–681, 2011.

[7] M. A. Alam and S. Mahapatra, "A comprehensive model of PMOS NBTI degradation," *Microelectronics Reliability*, vol. 45, no. 1, pp. 71–81, 2005.

[8] K. Kanda, K. Nose, H. Kawaguchi, and T. Sakurai, "Design impact of positive temperature dependence on drain current in sub-1-v cmos vlsis," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 10, pp. 1559–1564, 2001.

[9] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware microarchitecture," *ACM SIGARCH Computer Architecture News*, vol. 31, no. 2, pp. 2–13, 2003.

[10] W. Huang, S. Ghosh, K. Sankaranarayanan, K. Skadron, and M. R. Stan, "Hotspot: Thermal modeling for CMOS VLSI systems," *IEEE Transactions on Componenst Packaging and Manufacturing Technology*, pp. 200–205, 2005.

[11] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *Proceedings 33rd International Symposium on Computer Architecture (ISCA'06)*, vol. 34, 2006, pp. 78–88.

[12] L. Pohl, "Multithreading and strassens algorithms in sunred field solver," in *Proceedings 14th International Workshop on Thermal Inveatigation of ICs and Systems*, 2008, pp. 137–141.

[13] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschwiler, "3d-ice: A compact thermal model for early-stage design of liquid-cooled ics," *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2576–2589, 2014.

[14] V. Szekely, A. Poppe, A. Pahi, A. Csendes, G. Hajas, and M. Rencz, "Electro-thermal and logi-thermal simulation of vlsi designs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 5, no. 3, pp. 258–269, 1997.

[15] A. Timár and M. Rencz, "High resolution temperature dependent timing model in digital standard cell designs," *Journal of Low Power Electronics*, vol. 9, no. 4, pp. 414–420, 2013.

[16] A. Timár and M. Rencz, "Temperature dependent timing in standard cell designs," *Microelectronics Journal*, vol. 45, no. 5, pp. 521–529, 2014.

[17] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "Cacti 6.0: A tool to model large caches," *HP Laboratories*, pp. 22–31, 2009.

[18] M.-y. Hsieh, A. Rodrigues, R. Riesen, K. Thompson, and W. Song, "A framework for architecture-level power, area, and thermal simulation and its application to network-on-chip design exploration," *ACM SIGMETRICS Performance Evaluation Review*, vol. 38, no. 4, pp. 63–68, 2011.

[19] F. Terraneo, D. Zoni, and W. Fornaciari, "An accurate simulation frame-work for thermal explorations and optimizations," in *Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, 2015, pp. 5:1–5:6.

[20] G. Nagy and A. Poppe, "A novel simulation environment enabling multilevel power estimation of digital systems," in *Proceedings 17th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*, 2011, pp. 1–4.

[21] Z. Wang, L. Wang, H. Xie, and A. Chattopadhyay, "Power modeling and estimation during adl-driven embedded processor design," in *Proceedings 2013 4th Annual International Conference on Energy Aware Computing Systems and Applications (ICEAC)*, 2013, pp. 97–102.

[22] Synopsys. (2017). Processor Designer. Synopsys Inc., CA. [Online]. Available: http://www.synopsys.com/Systems/BlockDesign/processorDev

[23] A. Chattopadhyay, H. Meyr, and R. Leupers, *LISA: A Uniform ADL for Embedded Processor Modelling, Implementation and Software Toolsuite Generation*, Morgan Kaufmann, 2008, pp. 95–130.

[24] HotSpot. (2015, Jun.). HotSpot 6.0.[Online]. Available: http://lava.cs.virginia.edu/HotSpot/documentation.htm

[25] Z. Wang, C. Chen, and A. Chattopadhyay, "Fast reliability exploration for embedded processors via high-level fault injection," in *Proceedings International Symposium on Quality Electronic Design (ISQED)*, 2013, pp. 265–272.

[26] D. Kammler, J. Guan, G. Ascheid, R. Leupers, and H. Meyr, "A fast and flexible Platform for fault injection and evaluation in verilog-based simulations," in *Proceedings 3rd IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI)*, 2009, pp. 309–314.

[27] J. Bhasker and R. Chadha, *Static timing analysis for nanometer designs: A practical approach*. Springer Science & Business Media, 2009.

[28] A. Krstic,Y.-M. Jiang, and K.-T. Cheng, "Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 3, pp. 416–425, 2001.

[29] T. Sakurai and A. R. Newton, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 2, pp. 584–594, 1990.

[30] A. Sassone, A. Calimera, A. Macii, E. Macii, M. Poncino, R. Goldman, V. Melikyan, E. Babayan, and S. Rinaudo, "Investigating the effects of inverted temperature dependence (ITD) on clock distribution networks," in *Proceedings 2012 Design, Automation & Test in Europe Conference &Exhibition (DATE)*, 2012, pp. 165–166.

[31] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *IEEE Transactions on Electron Devices*, vol. 53, no. 11, pp. 2816–2823, 2006.