# Secure and Low Power Multicore Processor Design

by

Shayan Taheri

A term paper submitted in partial fulfillment of the requirements
for the CMOS IC design course

Instructor: Dr. Jiann-Shiun Yuan

University of Central Florida
Orlando, Florida

2015

# Contents

# Chapter 1

# Introduction

The emergence of parallel processors and machines dated back to the mid-1960s. However, they were left relatively untouched due to the difficulties of programming them. Since the past few years, major chip manufacturers have been interested in designing microprocessors based on the theory of parallel processing. In other words, multiple processors are placed inside a single chip. The main reason behind this trend is due to the requisiteness of following Moore's law, which is achieving growth in circuit integration and performance every two years. In fact, increasing clock rate and design complexities of uniprocessors provides performance improvement, but with the cost of extreme power dissipation and other issues.

Inclusion of more than one processing element in a single chip is called a multicore processor design that has the advantage of delivering significant performance growth by having more number of processing resources, but with much less power consumption. Nowadays, multicore processors are being accepted in all segments of the industry, especially for signal processing and embedded space applications. Figure 1.1 shows a sample high performance multicore architecture from ARM company.

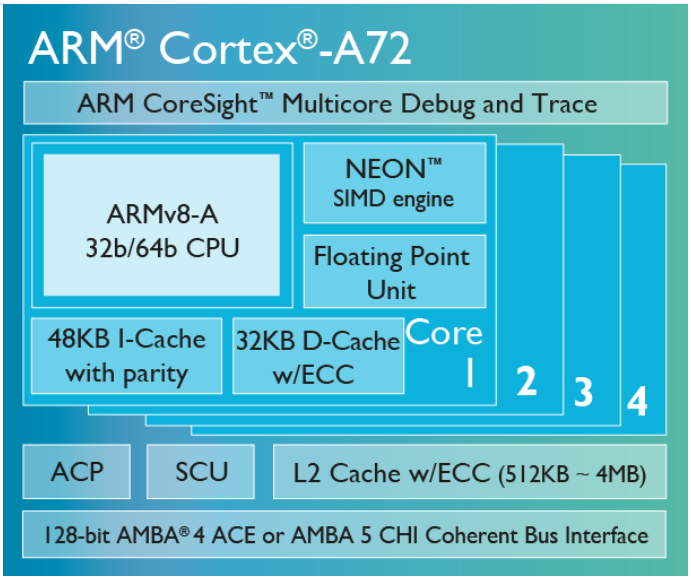



Figure 1.1: A high performance multicore architecture

Due to the rise of mobile and portable electronic devices and other forms of mobile computing where battery life, size, and weight are critical along with the notable demand in utilization of cloud-based data centers (a.k.a. warehouse-scale computers) that comes with the cost of remarkable energy usage, low power design is much more important than ever it was. Power consumption of a multicore processor or a single processing core can be reduced at different levels, including fabrication technology, circuit design, micro-architecture, instruction set architecture, run-time or operating system (O.S.), compiler, source code, algorithm, and application.

Low power design at architecture-level that is the interest of this study can be achieved by taking into account three roots of energy waste that are: (a) program waste – due to execution of instructions that are unnecessary for correct program execution; (b) speculation waste – due to speculative execution of instructions that do not commit their results; and (c) architecture waste – due to oversizing of processor structures. Also, a power management mechanism can be utilized inside a multicore architecture in order to cut unnecessary power consumption. An energy-efficient architecture from ARM company can be seen in Figure 1.2.

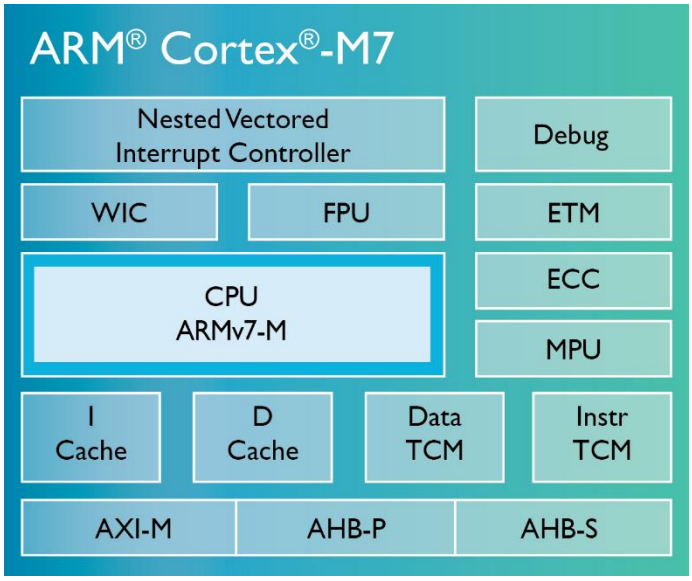



Figure 1.2: An energy-efficient processor architecture

With development of new cyber-attacks and escalation of either stored or transmitting digital data that might contain sensitive information, security has become one of the most important factors in processor design. Previously, a processor could be secured in confronting these attacks by protecting the integrity and confidentiality of the stored data and/or code in it, ensuring integrity of computations, preventing the execution of unauthorized code, and etc. However, the emergence of hardware attacks provides an impressive strength for the attackers to perform their malicious purposes due to gaining the ability of bypassing any employed software security mechanism. In this regard, new processor architectures need to be designed with security consideration (i.e. Security of Processor).

Recently, it has been investigated by the researchers and engineers that hardware implementation of defending and protection algorithms delivers higher performance than the software implementation of them. For example, a processor is designed and dedicated for execution of cryptographic algorithms. This topic of research can be referred to as Processor for Security. An example of secure processor architecture can be observed in Figure 1.3.

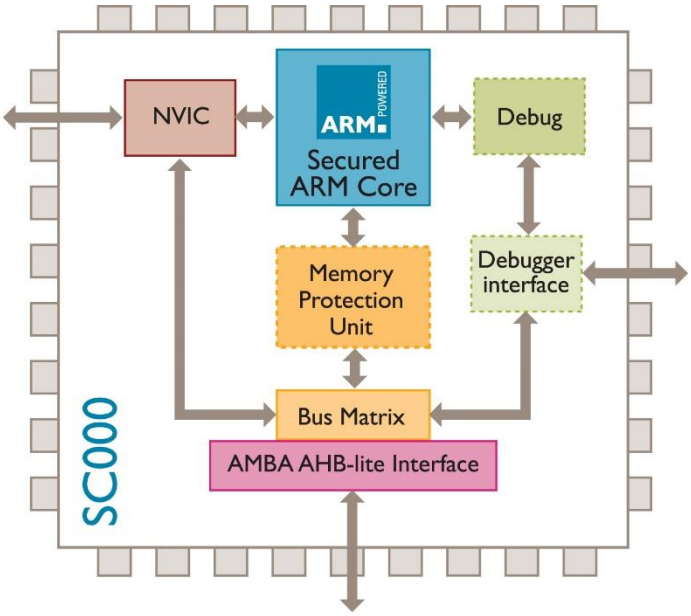



Figure 1.3: A secure processor architecture

The rest of this term paper is organized as follows. Section 2 discusses the security aspects of processor architecture design. The topic of low power processor design is investigated in Section 3. The experimental methodology for doing research in these areas is explained in Section 4. Finally, the paper is concluded in Section 5.

# Chapter 2

# Secure Processor Design

## 2.1 Overview

Nowadays, security of modern computing systems is more important than ever before, especially with emergence of new attacks and threats. These systems have security and protection requirements such as protecting the integrity and confidentiality of data and code stored in the processor, ensuring integrity of computations, and preventing the unauthorized access to the chip. Provision of security and applicability of defending mechanisms can become even harder due to the creation of hardware-based attacks, which provides a malicious person with a direct physical access to the system. In this way, any software-level security mechanism can be bypassed. In this regard, security is introduced as a new parameter to the architectural design of processors.

In general, a computing system can be classified into two regions: a trusted region and an untrusted region. The processor chip is considered trusted and it is assumed that code and data on-chip (in registers or caches) cannot be tampered with. All of the off-chip components, including the memory, buses, disk, BIOS, Ethernet card, and so forth are considered insecure and form the untrusted region of the system. However, the possibility of manipulating the processor design itself should not be overlooked. For example, a hardware Trojan can be designed and included inside a processor in order to perform data leakage and loss attack. Figure 2.1 shows a view of an inserted hardware Trojan inside a processor and the possible signals for its activation mechanism.

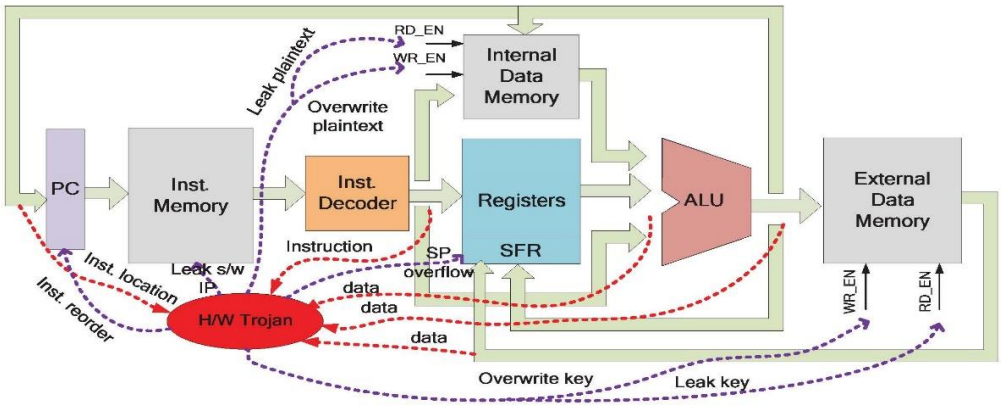



Figure 2.1: Inserted hardware Trojan inside a processor

Broadly, hardware-based attacks from data processing perspective can be classified into two main categories: passive attacks and active attacks. In passive attacks, a malicious person simply observes the data and/or code entering to or exiting from the processor chip in a non-intrusive manner. They can be implemented by placing a snooper or a logic analyzer on the memory bus and storing the information in a separate memory device. So, the sensitive information and data are leaked out without leaving behind any trace and interrupting the program execution.

Regarding the active attacks, an attacker manipulates the code and/or data of an application under execution with the purpose of changing the execution flow and eventually catching sensitive information and data. These attacks can be classified to: (a) spoofing attacks, where an attacker attempts to change the actual data at a memory location and pass it off as valid data; (b) splicing attacks, where the actual data are duplicated or replaced with the data of another location; and (c) replay attacks, where the previously recorded data are replayed at another time.

All of the discussed attacks and threats so far target the steady-state operation of computing systems, which means the processors have booted up securely. However, an attack might be conducted at boot time before the defending mechanisms can start protecting the system, which leaves the system completely vulnerable and weak in front of myriad number of attacks during steady-state operation. In order to provide a secure booting mechanism for a processor, three requirements must be satisfied. First, the root of trust should be located on-chip. Second, the added overhead by this mechanism needs to be reasonable. Third, the mechanism itself should not be comprisable. Meanwhile, all of the discussed topics belong to Security of Processor research area. In the rest of this chapter, the accomplished works related to defending and protection techniques are discussed.

## 2.2 Related Works to Architectural Countermeasures

Existing countermeasures for detecting attacks against processors generally fall into two categories that are physical analysis and design analysis. Physical analysis techniques are either destructive or non-destructive. In destructive methods, a processor chip must undergo backside thinning. Once the active layers are exposed, certain imaging equipment is used to map the processor's physical layout, which can then be compared against a reference specification. The weakness of these methods is their applicability only on a small number of chips. Meanwhile, the leveraged imaging technology in related equipment lags slightly behind state-of-the-art transistor technology feature size. Regarding non-destructive techniques, certain digital and analog test patterns are applied to the inputs of processor under test in order to observe and evaluate the outputs. The chip's functionality along with the electrical and timing characteristics of the outputs form a kind of "Fingerprint" for the processor that can be compared against the fingerprint of a reference processor chip for determination of its identity. These techniques can be used for a large number of chips. Another non-destructive technique is run-time verification, where the functionality and performance of a processor are examined during code execution and data processing.

The design analysis techniques are divided into two categories: (a) information flow tracking and controlling; and (b) design functionality and specification verification. The information flow tracking and controlling techniques are mostly used against software-based attacks in order to prevent data leakage and loss attack. Implementation of these techniques at hardware level (a.k.a. gate-level information flow tracking) helps to prevent suspicious flow of information between software components and hardware components. The belonging techniques to the second category are divided to: (a) formal verification, where the implemented algorithm underlying a design is verified and proved mathematically; and (b) model checking that is also known as VLSI testing and verification, where the functionality of a design is checked at different stages of IC design cycle.

Authors in [11] presented a non-destructive technique named, Security Checkers that was created based on the developed techniques for automatically generating synthesizable hardware entities that can verify temporal logic properties. A property in a temporal logic language like the Property Specification Language (PSL) can be verified at run-time using HDL structures that are automatically generated by the related tools. The main idea of this technique is adaptation of the tools for simplification of the creation of security-relevant run-time checkers that are capable of detecting the possible attacks. In this regard, a security assertion function (i.e. security-relevant run-time checker) is defined as an asserted PSL foundation language property that contains one or more security-relevant constants that map from the architectural specification to the processor design. Once the HDL entities of

security assertion functions are created, they are included in the design for functionality verification at different stages of design cycle along with after the chip fabrication.

XOM processor architecture [9] was designed with the purpose of code and data protection and securing the interaction between the processor and the main memory. According to this architecture, the main memory is in the execute-only form, which means the stored instructions/data inside it can only be executed/processed by the processor, but not manipulated. Any exiting code/data from the processor are encrypted and integrity protected using a message authentication code (MAC) before storing in the memory. When a ciphertext is brought back on-chip, it is decrypted before execution/processing by the processor. If integrity checking of a message fails, then an exception is triggered and execution of program and process of data are halted. One of the side effect of this architecture is impossibility of using a created cipher text by a computing process by another computing process, since each process uses its own key for encryption and decryption of messages. Figures 2.2 and 2.3 show the message format and the XOM architecture respectively. In Figure 2.2, the bottom rectangle shows the format of constructed message by the processor. The tag part is used to retrieve the required key for data decryption. The hash value is actually the MAC that is the outcome of entering the tag, data, and the interrupt counter to the hash function.

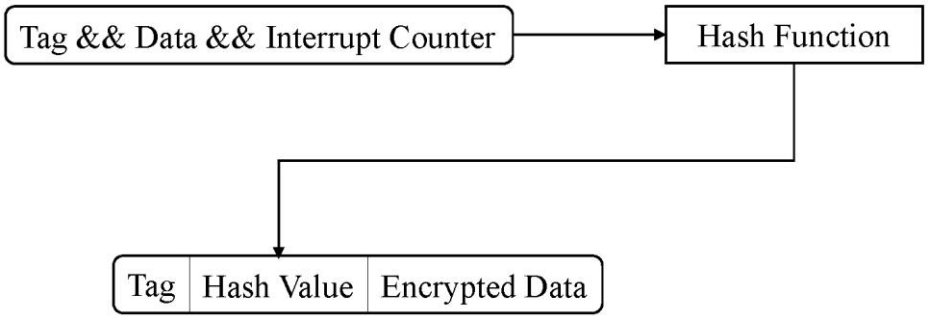



Figure 2.2: The message format

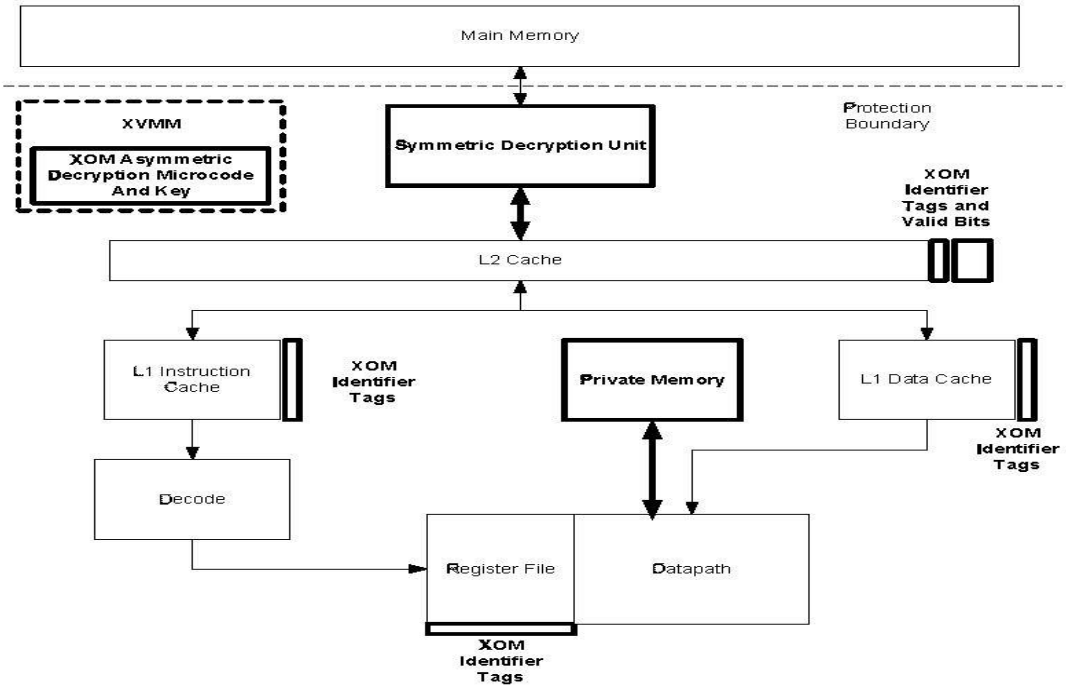



Figure 2.3: The XOM architecture

A VLSI testing and verification technique named, Functional Analysis for Nearly-unused Circuit Identification (FANCI) was presented in [8] that is used for finding possible infected sites inside a digital integrated circuit. The concept behind FANCI is discovering the effects of input wires on output wires. By saying "Wires", it means consideration of both exogenous and endogenous input or output wires (not merely primary inputs and primary outputs). In other words, the tool analyzes the output(s) of every gate and component of the design based on its Boolean function. The outcome of the tool is determination of "Nearly-Unused" wires or "Nearly Stuck-at Fault" wires that are called "Suspicious" wires. The process of finding suspicious wires is based on a newly defined parameter named "Control Value", which shows the effect's strength of each input wire on its related output wire. It means that based on the truth table of a gate or component, how the transition of each input wire can affect the output wire of the gate/component. The procedure of calculating control value for each wire has been shown in Figure 1. As it can be understood from the figure, each input wire of a gate or component has its own effect on related output wire, which causes creation of a vector of control values for the output wire.

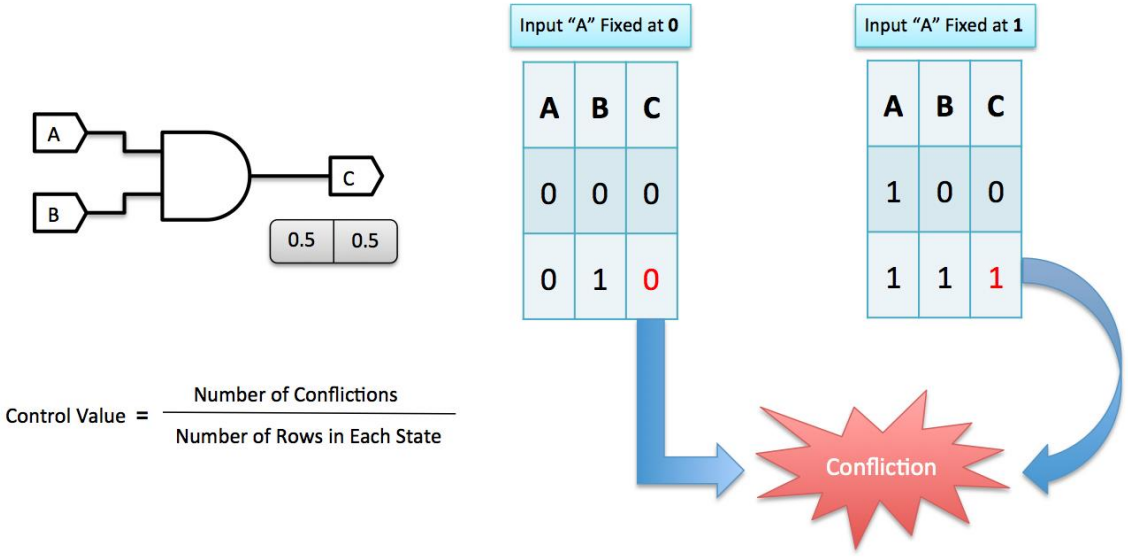



Figure 1: Description of FANCI

By realization of having a vector of data for each output wire, implementation of different kinds of heuristic algorithms for analysis of data vectors would be imperative. In initial FANCI, only three simple algorithms that are Median, Average, and Hybrid (consists of both Median and Average) algorithms, were implemented to calculate Final Control Value of each output wire based on the data of its vector. However, there is another algorithm named Triviality in the initial design, which calculates final control value of an output wire without consideration of its vector of control values. According to this algorithm, the final control value of an output wire is the number of rows that its value equals to logic 0 over total number of rows --- based on the truth table of its Boolean function. Finally, all of the output wires that have final control values lower than a cut-off threshold value are considered as suspicious wires.

# Chapter 3

# Low Power Processor Design

## 3.1 Overview

Due to the recent advances in microelectronic technology, more functionality and performance can be provided on an integrated circuit (IC), which creates the possibility of designing smaller devices. This progress caused a shift in portable applications from low performance products such as wristwatches and calculators to high throughput and speed, and computation intensive products such as notebook computers and cellular phones. The dark side of this trend is the urgent demand of these applications for low power consumption and reasonable battery life, size, and weight. Otherwise, the usability and applicability of these applications are questionable.

Therefore, this trend makes power consumption as important as performance and area. Figure 3.1 shows the reason for this requirement graphically. Another driving force for diving into low power design is its important role in silicon integration (i.e. more number of transistors on a single chip or on a multi-chip module), which otherwise cooling, packaging, and reliability can be serious issues for electronic products. Even, it is a priority for large and/or parallel computing machines. Nowadays, low power design has postulate in different technologies, such as telecommunications, computers, consumer electronics, and biomedical, which shows the importance of this research area.

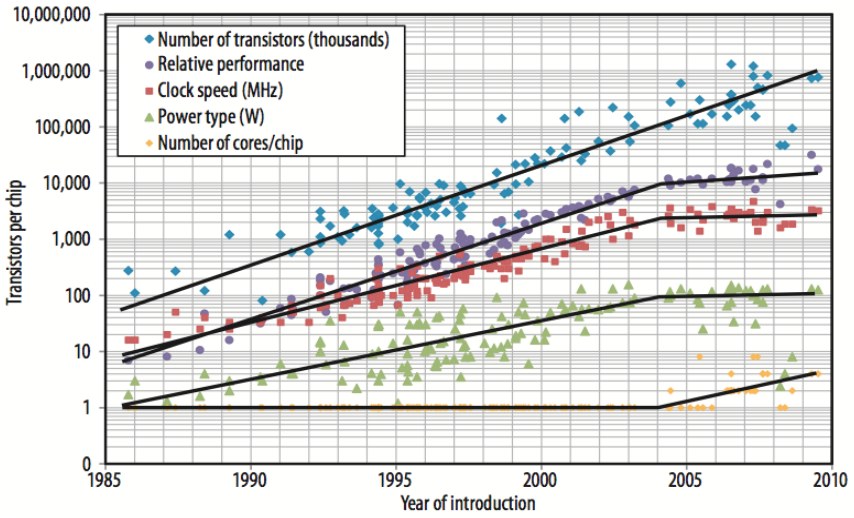



Figure 3.1: Requirement to low power design

Due to the ever improvement of architectural mechanisms and inclusion of more complex computational units in modern high-performance processors, power consumption and heat dissipation have become significant challenges in their design. Besides that, low power design is highly advantageous for general-purpose processors due to having idle time in their user-interactive mode. Power consumption of processors can be reduced at different levels, including fabrication technology, circuit design, micro-architecture, instruction set, run-time or operating system (O.S.), compiler, source code, algorithm, and application.

## 3.2 Origins of Power Dissipation and Solutions

CMOS circuits have both static and dynamic power dissipation. Static power is originated from to two types of leakage current: (a) sub-threshold leakage current that is generated when transistors don't turn off completely; and (b) P-N junction (diode) leakage current that is generated from existing diodes in transistor structure. While static power has its own importance, but it is insignificant in compare to dynamic power that is the dominant component of power dissipation in digital systems. This power is due to two noticeable currents: (a) switching current, which comes from the charging and discharging mechanism of load capacitance; and (b) short-circuit current, which is produced when both n-type and p-type transistors turn on during a transition. Load capacitance of a logical stage is constructed by gate output diffusion capacitance of current stage, gate input capacitance of next stage, and interconnecting metal layer capacitance.

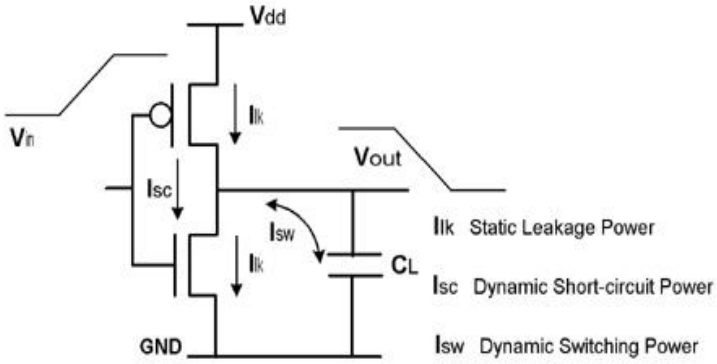



Figure 3.2: Static and dynamic power

Figure 3.2 is a graphical example for the aforementioned currents in digital systems. In this figure $I_{lk}$ represents the leakage currents, $I_{sw}$ is switching current, and $I_{sc}$ is short-circuit current. Equations 3.1 and 3.2 can be used in order to obtain static and dynamic power for a single transistor. In these equations, $V_{DD}$ is supply voltage, $K_{Design}$ is design parameter, $I_{Leakage}$ is total leakage current, $C$ is load capacitance, $A$ is activity factor, and $f$ is frequency (clock rate).

$$P_{Static} = V_{DD} \,.\, K_{Design} \,.\, I_{Leakage} \tag{3.1}$$

$$P_{Dynamic} = \frac{1}{2} \,.\, C \,.\, A \,.\, f \,.\, V_{DD}^2 \tag{3.2}$$

Shortage of attention to attenuate these currents properly leads to more current draw and consequently more power consumption, which causes having less effective energy capacity for battery, heating processor and affecting speed and reliability, and ground bounce (i.e. resistive and inductive supply voltage drop). Meanwhile, it should be mention that reducing power consumption may be provided at the cost of performance degradation. So, it is required to achieve a trade-off between delay and power. Equation 3.3 can be used in order to calculate the delay of a device. In this equation, $C$ is load capacitance, $I_{out\ (ave)}$ is the average of flowed currents into load capacitance, $V_{DD}$ is supply voltage, $K_v$ technology constant, $W$ is width, $V_{th}$ is threshold voltage, and $V_{DSAT}$ is drain voltage at saturation.

$$Delay = \frac{C}{I_{out\ (ave)}} \,.\, \frac{V_{DD}}{2} = \frac{C \,.\, V_{DD}}{K_v \,.\, W \,.\, (V_{DD} - V_{th} - V_{DSAT})} \tag{3.3}$$

As it was mentioned before, there are different design techniques in processor abstraction layers that can be utilized in order to reduce and manage power consumption. In this survey, the concentration is on power reduction and management techniques at architecture-level. Development of these techniques is mostly based upon utilization and exploitation of application characteristics and access/activation of computing or storing resources only when it is necessary. Figure 3.3 shows distribution of power consumption in different processor elements.

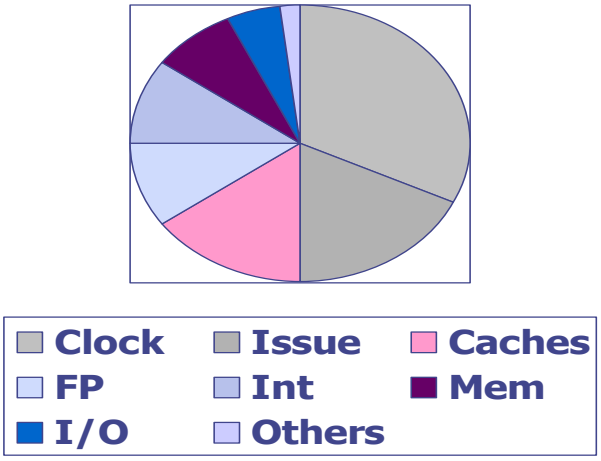



Figure 3.3: Processor power distribution

General architectural low power design techniques are classified into two categories: (a) processor-based, and (b) memory-based. The design techniques based on processor structure are: (1) voltage/frequency scaling (supply voltage and/or frequency reduction in idle time of processor); (2) clock gating (disabling clock to inactive dynamic components); (3) power gating (disabling supply voltage to inactive components), (4) pipeline gating (reducing mis-speculated instruction execution by fetch stalling when confidence is low); (5) pipeline balancing (adjusting effective pipeline ways for available instruction-per-cycle); and (6) efficient logical circuit for instruction-issue pipeline stage. The memory-based design techniques can be mentioned as: (1) banked or hierarchical register file; (2) sub-banked cache; (3) sequential access or way prediction caches; (4) dynamically adjusting cache size; (5) decay cache for reducing static power; and (6) low power DRAM cells with deep sleeping modes.

## 3.3 Low Power Design at Architecture-Level

There are three basic causes (or roots) for power dissipation at architecture-level that are: (1) Program waste of energy, which is due to execution of instructions that are unnecessary for correct program execution; (2) Speculation waste of energy, which is due to speculative execution of instructions that do not commit their results; and (3) Architectural waste of energy, which is due to oversizing of processor structures. Program waste occurs due to fetch and execution of unnecessary instructions. An instruction is labeled as "Unnecessary" when it produces either dead or redundant results. An instruction that generates dedicated data for an unnecessary instruction can also be considered as unnecessary. This type of instructions doesn't have any effect on processor state.

The unnecessary instructions can be classified in this way: (a) Dead instruction (`dead`), whereby dead results are produced. A result is labeled as "Dead" if it is written in register(s) or memory, but is not read before being overwritten by another instruction result; (b) Silent Store instruction (`stores`), whereby the old and new written data to a given address of memory are the same; (c) Silent Load instruction (`loads`), whereby the old and new read data from a given address of memory and written to a destination register are the same; (d) Silent Register instruction (`sir` and `sfp`), is defined as an integer- or floating point-based register operation that produces the same result as the existing one in the destination register; (e) Silent Conditional Move instruction (`cmov`), which is an unsatisfied and not performed conditional move operation (i.e. behaving like a null operation).

Figure 3.4 shows the amount of wasted energy by each class of unnecessary instructions relative to the total amount of processor energy usage (in percentage), for an out-of-order Alpha processor. The configuration of this processor is presented in Table 3.1. The processor was simulated for 300 million committed instructions using the conjunction of SMTSIM and Wattch simulators. Meanwhile, the employed benchmarks were chosen from SPEC2000 bench set. These results don't encompass the wasted energy by data generators (i.e. parent instructions) of these instructions. The all category includes all classes of unnecessary instructions. A number of solutions have been presented for this root of energy waste that work based on dynamically identifying, predicting, and eliminating unnecessary instructions.

| Parameter | Value |
|---|---|
| Fetch bandwidth | 8 instructions per cycle |
| Functional Units<br>Instruction Queues | 3 FP, 6 Int (4 load/store)<br>64-entry FP, 64-entry Int |
| Inst Cache | 64KB, 2-way, 64-byte lines |
| Data Cache | 64KB, 2-way, 64-byte lines |
| L2 Cache (on-chip) | 1 MB, 4-way, 64-byte lines |
| Latency (to CPU) | L2 18 cycles,<br>Memory 300 cycles |
| Pipeline depth | 8 stages |
| Min branch penalty | 6 cycles |
| Branch predictor | 21264 predictor |
| Instruction Latency | Based on Alpha 21164 |

Table 3.1: The examined processor configuration

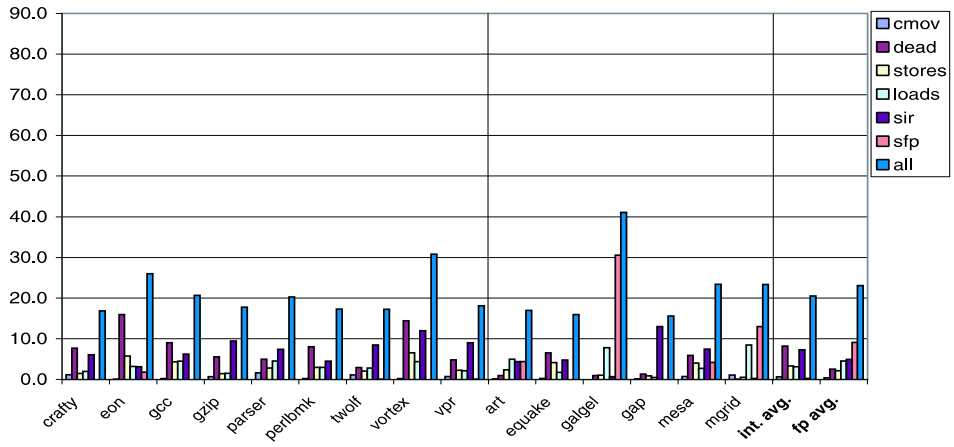



Figure 3.4: Wasted energy by unnecessary instructions

Speculation waste arises from wrong speculative fetch and execution of instructions due to mis-predicted branches. These instructions are entered into pipeline and consume resources, without being committed and changing processor state. Figure 3.5 shows the percentage of wasted energy by mis-speculated instructions. Possible solutions for this problem are: (a) improving the accuracy of branch predictors; (b) pipeline gating (i.e. keeping instructions from continuing down a processor pipeline once a low-confidence branch has been fetched); and (c) multithreading execution of instructions.

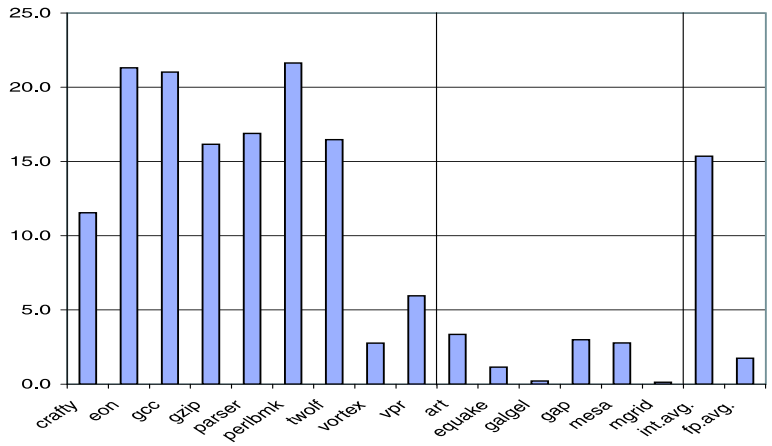
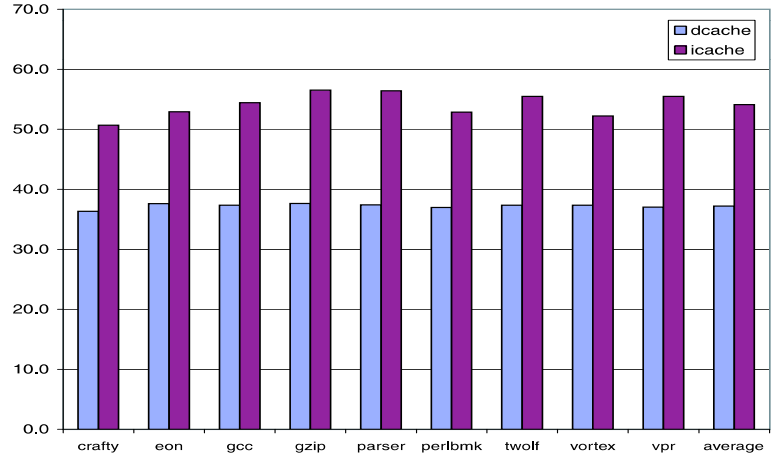
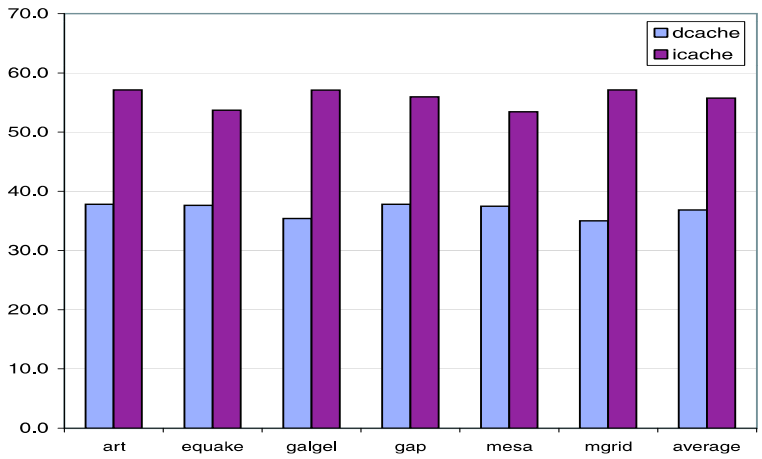



Figure 3.5: Wasted energy by mis-speculated instructions

Architectural waste is originated from oversizing of architectural structures, such as caches, instruction queues, branch predictors, and etc. Regularly, architectural structures are designed with large capacity for performance improvement. However, their capacities might be much larger than the enough amounts for exploitation of a given program. This causes extreme power dissipation due to retention of large amount of data, which might not be used in the next phases of program. Possible solutions for this problem can be mentioned as: (a) reconfigurable cache structures; (b) power downing a cache parts that are unused; (c) dynamically reconfiguration of the window size of issue queue; and (d) hierarchical design of cache, issue queue, and register file. Figure 3.6 shows the percentage of saved energy by an adaptive reconfigurable (AR) cache in compare to a fixed cache.




(a) Simulation by integer-based benchmarks




(b) Simulation by floating point-based benchmarks

Figure 3.6: Saved energy by AR cache

## 3.3 Other Architectural Techniques

Power management mechanism for multicore processors is highly important due to allowing power/energy savings of a number of processing cores that are not used effectively. It refers to selective shut-off or slow-down of logical components that are idle or underutilized and has been proven to be a particularly effective technique for power saving. The goal of a power management mechanism is to reduce the power consumption of a processor by putting its components into different states, each representing certain performance and power consumption level. The type and timing of these transitions are determined based on the behavioral history, workload and performance constraints. This mechanism can be implemented at software-level and controlled by operating system or at hardware-level and controlled by power management unit (PPU). According to ACPI (Advanced Power and Configurations Interface) specifications, O.S. controlled power management mechanism is the common approach that industry has adopted for low power design.

Due to the general purpose design of processors, they can meet most demanding application throughput requirements under worst case operating conditions. However, it causes an additional margin or power waste since it mostly functions under typical operating conditions. Due to this reason, an optimized version of dynamic voltage scaling named, adaptive voltage scaling can be used. It is a closed-loop dynamic power minimization technique that reduces power based on the actual operating conditions of the chip, which means the power consumption is continuously adjusted during the processor run time. Razor is an adaptive voltage scaling technique that operates based on dynamic detection and correction of circuit timing errors. Its key idea is tuning the supply voltage by monitoring the error rate during circuit operation in order to eliminate the requirement for voltage margins and exploitation of the data dependence of circuit delay.

# Chapter 4

# Experimental Methodology

In this chapter, the basic steps of common experimental approach for doing research in the area of computer architecture and electronic design automation is discussed. The first step is performance examination of a designed processor architecture that is achieved using Gem5 architectural simulator. In this regard, an architecture type is chosen from Gem5 suite and its region of interest is modified according to the designed plot. Next, a number of benchmarks are run on the processor in order to test its performance parameters. Figure 4.1 shows the performance analysis of standard out-of-order X86 processor architecture using MiBench suite.

| Benchmark Name | Cache Occupancy for Level-1 Data Cache (%) | Cache Occupancy for Level-1 Instruction Cache (%) | Miss Rate for Level-1 Data Cache | Miss Rate for Level-1 Instruction Cache |
|---|---|---|---|---|
| basicmath | 0.069317 | 0.484173 | 0.000003 | 0.000018 |
| bitcount | 0.080477 | 0.528225 | 0.000110 | 0.000429 |
| CRC32 | 0.175465 | 0.058465 | 0.000035 | 0.000003 |
| dijkstra | 0.860209 | 0.862271 | 0.001995 | 0.000349 |
| FFT | 0.635835 | 0.652761 | 0.000091 | 0.000031 |
| patricia | 0.987609 | 0.909262 | 0.002135 | 0.115211 |
| qsort | 0.976217 | 0.689045 | 0.000478 | 0.000038 |
| sha | 0.306238 | 0.652004 | 0.000038 | 0.000058 |
| susan | 0.559061 | 0.024594 | 0.000052 | 0.000002 |

| Benchmark Name | Miss Rate for Level-2 Cache | Branch Target Buffer Hit Rate (%) | Instruction Per Cycle (IPC) |
|---|---|---|---|
| basicmath | 0.981818 | 98.673403 | 1.791908 |
| bitcount | 0.957286 | 51.826504 | 1.600047 |
| CRC32 | 0.119490 | 39.992700 | 2.086998 |
| dijkstra | 0.062726 | 99.891064 | 1.504050 |
| FFT | 0.782401 | 72.320259 | 1.334647 |
| patricia | 0.012782 | 89.593522 | 0.975233 |
| qsort | 0.983102 | 99.972365 | 1.977259 |
| sha | 0.983287 | 99.701301 | 2.027326 |
| susan | 0.999070 | 99.229096 | 2.071750 |

Figure 4.1: Performance analysis results

Second, the RTL codes for an analogous processor are generated by FabScalar tool. The codes are modified in accordance to the designed processor architecture. The design functionality is validated using Cadence NCSim and SimVision. Third step belongs to the IC design synthesis that is accomplished using Synopsys Design Compiler. A technology standard cell library is used for this element and the next one. Figure 4.2 shows the achieved results (i.e. Area-Delay-Power (ADP) report) for a benchmark circuit using SAED 32nm standard cell library.

| Area | Delay | Power |
|---|---|---|
| $39451.67\ \mu m^2$ | $2.29\ ns$ | $1.2215 \times 10^4\ \mu W$ |

Figure 4.2: Synthesis ADP report

In the last step, the design is placed and routed using Synopsys IC Compiler. Figures 4.3 and 4.4 show the achieved results from this tool that are the layout ADP report and the circuit layout respectively.

| Area | Delay | Power |
|---|---|---|
| $49420.3992\ \mu m^2$ | $2.25\ ns$ | $8.9988 \times 10^3\ \mu W$ |

Figure 4.3: Layout ADP report

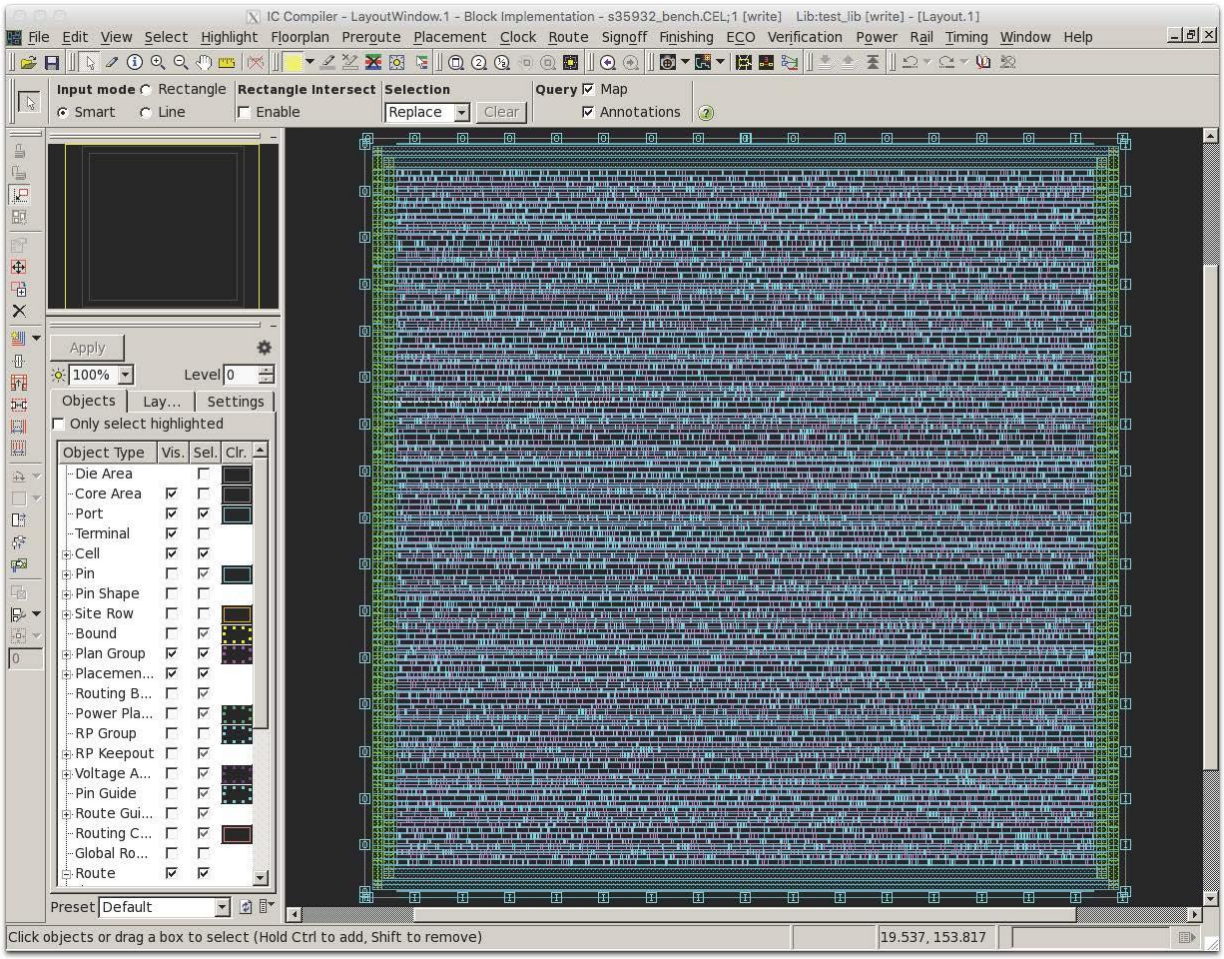



Figure 4.4: Circuit Layout

# Conclusions

The internet of things is one of the emerging technologies that is highly desirable in healthcare system, environmental monitoring, intelligent transportation, and so forth. In the IoT world, everything can be sensed easily by just attaching a battery powered sensor to it. The sensors are required to work for enough time without necessitation for maintenance and recharging. Handling these tasks along with any other processing task should be accomplished using an energy-efficient central processing unit or multicore processor. Also, the processor must be high performance enough to execute diverse applications, such as bridge monitoring and surveillance camera. Due to the ever increase in computational complexities of the processors and more demand for boosting performance (especially for the IoT applications), low power design can be a serious challenge. Therefore, effective techniques are required to be developed in order achieve a trade-off between these two design parameters. Low power design techniques at architecture-level can be greatly effective and much cheaper in compare to the techniques of other abstraction layers of supercomputing system design. However, they haven't been studied sufficiently and require further research and attention. Three roots of power dissipation at architecture-level are determined as: program waste, speculation waste, and architectural waste. These roots of energy waste are originated from unnecessary instructions, mis-speculated instructions, and oversized architectural structures respectively.

It is well known that security issues aren't new in the development of supercomputing systems. Since more than three decades ago when personal computers became ubiquitous rapidly, and the introduction of the World Wide Web, they are the victim of various forms of threats such as viruses, Trojans, worms, and so forth. These threats might target sensitive information, system functionality and/or performance, and other cases to perform malicious purposes. It is imaginable that in the IoT world that every possible thing is connected to a gigantic network scope, more sophisticated and complex hacking and attacking systems are created. In order to circumvent the problem of security in the IoT world, processors and their interfaces should be positioned in the first place in the IoT security (due to their greater importance and also more vulnerabilities in compare to the communication network and other devices). Intensive research efforts have been done against the security issues in embedded systems. However, security is often misconducted by processor designers as the addition of features, such as specific cryptographic algorithms and security protocols to a design. In fact, it is a new dimension that designers should consider throughout the design process along with other parameters (i.e. cost, performance, and power). Provision of security needs to cover all aspects of processor design from architecture to implementation and fabrication. In this term paper, different aspects and techniques for secure and low power processor design were discussed. Also, the experimental methodology for doing research in this area was explained. Hope this paper can be a valuable guideline for development of the next generation of either central processing units or multicore processors, especially for the IoT applications.

# References


[1] Ernst, Dan, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Rajeev Rao, Toan Pham, Conrad Ziesler et al. "Razor: A low-power pipeline based on circuit-level timing speculation." In Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on, pp. 7-18. IEEE, 2003.

[2] Ghose, Kanad, and Milind B. Kamble. "Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation." In Low Power Electronics and Design, 1999. Proceedings. 1999 International Symposium on, pp. 70-75. IEEE, 1999.

[3] Shiue, Wen-Tsong, and Chaitali Chakrabarti. "Memory exploration for low power, embedded systems." In Proceedings of the 36th annual ACM/IEEE Design Automation Conference, pp. 140-145. ACM, 1999.

[4] Burd, Thomas D., and Robert W. Brodersen. "Energy efficient CMOS microprocessor design." In System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on, vol. 1, pp. 288-297. IEEE, 1995.

[5] Su, Ching-Long, Chi-Ying Tsui, and Alvin M. Despain. "Low power architecture design and compilation techniques for high-performance processors." In Compcon Spring'94, Digest of Papers., pp. 489-498. IEEE, 1994.

[6] Pedram, Massoud. "Power minimization in IC design: principles and applications." ACM Transactions on Design Automation of Electronic Systems (TODAES) 1, no. 1 (1996): 3-56.

[7] Seng, John S., and Dean M. Tullsen. "Architecture-Level Power Optimization-What Are the Limits." Journal of Instruction-Level Parallelism 7, no. 7 (2005): 3.

[8] Waksman, Adam, Matthew Suozzo, and Simha Sethumadhavan. "FANCI: identification of stealthy malicious logic using boolean functional analysis." In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 697-708. ACM, 2013.

[9] Lie, David, Chandramohan Thekkath, Mark Mitchell, Patrick Lincoln, Dan Boneh, John Mitchell, and Mark Horowitz. "Architectural support for copy and tamper resistant software." *ACM SIGPLAN Notices* 35, no. 11 (2000): 168-177.

[10] Wang, Xinmu, Tatini Mal-Sarkar, Aswin Krishna, Seetharam Narasimhan, and Swarup Bhunia. "Software exploitable hardware Trojans in embedded processor." In *Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2012 IEEE International Symposium on*, pp. 55-58. IEEE, 2012.

[11] Bilzor, Michael, Ted Huffmire, Cynthia Irvine, and Tim Levin. "Security checkers: Detecting processor malicious inclusions at runtime." In *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, pp. 34-39. IEEE, 2011.

[12] Chhabra, Siddhartha, Yan Solihin, Reshma Lal, and Matthew Hoekstra. "An analysis of secure processor architectures." In *Transactions on computational science VII*, pp. 101-121. Springer Berlin Heidelberg, 2010.

[13] Schauer, Bryan. "Multicore processors–a necessity." *ProQuest discovery guides* (2008): 1-14.

[14] Keckler, Stephen W., and H. Peter Hofstee, eds. *Multicore processors and systems*. Berlin: Springer, 2009.

[15] Venu, Balaji. "Multi-core processors-An overview." *arXiv preprint arXiv:1110.3535* (2011).

[16] Suh, G. Edward, Charles W. O'Donnell, and Srinivas Devadas. "Aegis: A single-chip secure processor." *Design & Test of Computers, IEEE* 24, no. 6 (2007): 570-580.

[17] Fletcher, Christopher W., Marten van Dijk, and Srinivas Devadas. "A secure processor architecture for encrypted computation on untrusted programs." In *Proceedings of the seventh ACM workshop on Scalable trusted computing*, pp. 3-8. ACM, 2012.

[18] Gubbi, Jayavardhana, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. "Internet of Things (IoT): A vision, architectural elements, and future directions." *Future Generation Computer Systems* 29, no. 7 (2013): 1645-1660.

[19] Atzori, Luigi, Antonio Iera, and Giacomo Morabito. "The internet of things: A survey." *Computer networks* 54, no. 15 (2010): 2787-2805.

[20] Suo, Hui, Jiafu Wan, Caifeng Zou, and Jianqi Liu. "Security in the internet of things: a review." In *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, vol. 3, pp. 648-651. IEEE, 2012.

[21] Limin, He. "Embedded System for Internet of Things." *Microcontrollers & Embedded Systems* 10 (2010): 5-8.

[22] Ukil, Arijit, Jaydip Sen, and Sripad Koilakonda. "Embedded security for Internet of Things." In *Emerging Trends and Applications in Computer Science (NCETACS), 2011 2nd National Conference on*, pp. 1-6. IEEE, 2011.

[23] Kumar, Rakesh, Keith Farkas, Norman P. Jouppi, Parthasarathy Ranganathan, and Dean M. Tullsen. "Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction." In *Microarchitecture, 2003. MICRO-36. Proceedings. 36th Annual IEEE/ACM International Symposium on*, pp. 81-92. IEEE, 2003.

[24] Ekanayake, Virantha, Clinton Kelly IV, and Rajit Manohar. "An ultra low-power processor for sensor networks." *ACM SIGARCH Computer Architecture News* 32, no. 5 (2004): 27-36.

[25] Luo, Yan, Jia Yu, Jun Yang, and Laxmi Bhuyan. "Low power network processor design using clock gating." In *Proceedings of the 42nd annual Design Automation Conference*, pp. 712-715. ACM, 2005.

[26] Gonzalez, Ricardo E. "Xtensa: A configurable and extensible processor." *IEEE micro* 2 (2000): 60-70.

[27] Gautham, Parthsarathy, R. Parthasarathy, and Karthi Balasubramanian. "Low-power pipelined mips processor design." In *Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on*, pp. 462-465. IEEE, 2009.

[28] Brennan, J. Patrick, Alvar Dean, Stephan Kenyon, and Sebastian Ventrone. "Low power methodology and design techniques for processor design." In *Proceedings. 1998 International Symposium on Low Power Electronics and Design (IEEE Cat. No. 98TH8379)*. 1998.