IMPROVING PERFORMANCE AND MITIGATING TEMPERATURE RISE WITH RECONFIGURABLE STT-NV LOGIC BASED FUNCTIONAL UNIT

By

Adarsh Reddy Ashammagari
A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Master of Science
Electrical and Computer Engineering

Committee:

___________________________	Dr. Homayoun Houman, Thesis Director
___________________________	Dr. Brian Mark, Committee Member
___________________________	Dr. Hakan Aydin, Committee Member
___________________________	Dr. Andre Manitius, Department Chair
___________________________	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date:______________________	Summer Semester 2013 George Mason University Fairfax, VA

Improving Performance and Mitigating Temperature Rise with Reconfigurable STT-NV Logic Based Functional Unit

A Thesis submitted in partial fulfillment of the requirements for the degree Master of Science at George Mason University

By

Adarsh Reddy Ashammagari
Bachelor of Technology
Dr. M. G. R. University, 2011

Director: Dr. Houman Homayoun, Assistant Professor
Electrical and Computer Engineering

Summer Semester 2013
George Mason University
Fairfax, VA

DEDICATION

I dedicate this thesis to my advisor Dr. Houman Homayoun. I learned lot of techniques from him; he is responsible for improvement of my skills to maximum. Dr. Homayoun is a person who deals with all the issues very calmly and professionally and always with a smile on his face. There were many instances where I was stuck at a point and didn't find a way out, but Dr.Homayoun and his advising helped me in debugging those errors and proceeding further in my research. I learned how to perfect my work under his valuable guidance. I thank him for his hard work and his teaching skills in making me understand the concepts. I thank him wholeheartedly for his support and encouragement.
I would like to dedicate this thesis to my committee members Dr. Brian Mark and Dr. Hakan Aydin. I thank them for helping in betterment of my work and pointing out some very interesting views which can improve this research work.
I dedicate this thesis to my dear parents Madhava Reddy and Dr. Sridevi Reddy and my dear brother Aditya Reddy. Their support was always encouraging and energizing. I thank them for helping me with many difficult situations and motivating me constantly. My million thanks for having such a supportive and understanding family.
Lastly, I would like to dedicate my thesis and thank all my friends who helped me greatly and encouraged me. I enjoyed sharing different ideas with them and getting their feedback. Thanks again for each and every one.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

ABSTRACT

IMPROVING PERFORMANCE AND MITIGATING TEMPERATURE RISE WITH RECONFIGURABLE STT-NV LOGIC BASED FUNCTIONAL UNIT

Adarsh Reddy Ashammagari, M.S.

George Mason University, 2013

Thesis Director: Dr. Homayoun Houman

Unavailability of functional units and their unequal activity makes them performance bottleneck and thermal hot spots units in general-purpose processors. This thesis proposes to use reconfigurable functional units to overcome these challenges. A selected set of complex functional units that might be under-utilized (i.e. have low temperature), such as a multiplier, divider, etc. are realized in a time multiplexed fashion using a shared programmable Look Up Table (LUT) based fabric. This allows for run-time reconfiguration and migration of the activity from functional units that are responsible for thermal hot spots (i.e. high temperature functional units) to the units that are less active. LUT based implementation also allows under-utilized functional units to be dynamically reconfigured to the functional units that are performance bottleneck (i.e. heavily utilized functional units) and hence improving performance. The programmable LUTs are realized using Spin Transfer Torque (STT) Magnetic technology (also called STT-NV) due to its zero leakage and CMOS compatibility. This thesis presents, the several

developed power-thermal and performance-aware algorithms to most effectively reconfigure functional units at run-time. The results show significant performance improvement of 16% on average across standard benchmark. Also, reconfiguration reduces maximum temperature of functional units by up to 27^{o} C and almost eliminates the thermal variation across functional units. This comes with almost 16% increase in functional units total power dissipation across SPECint benchmarks and almost 18% reduction in their power across SPECfp benchmarks.

1 Introduction

With the current shrinking trend in CMOS technology, larger processing capabilities can be incorporated within the same die footprint. At the same time, the number of functions that are now computationally realizable has also increased in leaps and bounds. Therefore, an efficient allocation of functional resources becomes crucial to the overall performance of any processing unit. Under limited functional resources available to the processors, major performance bottlenecks arise from functional resource unavailability. With mobile devices being battery powered, energy efficiency of the processing units and the thermal stability of the design become major concerns. These concerns become serious with the growth rate in battery power falling short of the growth rate in consumer demands for higher data rates.

One promising way to address this energy-efficiency challenge is to exploit reconfiguration in designs, whereby the same hardware component can be "reconfigured" to execute different functionality at different point in time. The way to provide re-configurability in designs today is primarily through the use of FPGA or Coarse Grain Reconfigurable Arrays (CGRA). For FPGAs, However, not only there are challenges in their integration on the processor die, but they also exhibit quite poor power-efficiency.

Unlike FPGAs, CGRAs are extremely power-efficient and quite general-purpose accelerators (wherein most mobile applications can be accelerated). However CGRAs pose immense challenges to compiler technology [27-28], while at the same time their performance and power-efficiency is so critically dependent on the compiler optimization techniques.

This thesis presents an alternative way to enable re-configurability in embedded processor architecture. The solution is to enable re-configurability in the general-purpose core by using Spin Transfer Torque non-volatile (STT-NV) fabric. STT-NV is a new fabrication technology that is compatible with CMOS. It adds only a few metal layers between the layers 3 and 4 of the chip, and has been touted as one of the most promising "post-CMOS technology." The advantages of using STT-NV technology are its zero standby power, non-volatility, scalability, and thermally robust behavior. The most popular use of STT-NV is to implement low-power, high-density on-chip memories. As a thumb rule, it is possible to design 4 times denser memories, with almost the same read power and read times with STT-NV technology. Since caches (made up of RAM circuits) are the major contributors to the leakage power of the processor (which in turn is a significant chunk of the total processor power), using STT-NV based RAM result in a good amount of power savings [12]. This thesis takes the next step – attempt to aggressively exploit STT-NV technology, by using it to design the reconfigurable logic needed to support dynamic reconfiguration of functional units. This thesis explores ways to use reconfiguration to maximally improve power, performance and robustness of processor architecture.

1st first step, this thesis investigates the design of a reconfigurable functional unit in embedded processors. General-purpose embedded processor such as arm, atom, xscale, mips or tenscilica based cores typically have a certain number of functional units for each type of adder, divider, and multiplier, for instance. In these embedded cores a functional unit is a critical unit that is not only a performance bottleneck of the design, but also a temperature hotspot [7, 8]. Due to its high activity and small size, the functional unit's power density is large, and therefore is a thermal hotspot. Dynamic reconfigurable units to address these power, performance, and thermal challenges using STT-NV logic are being attempted in this thesis.

This thesis presents analysis, demonstrating the benefits of a reconfigurable STT-NV logic when deployed in the functional unit of the general-purpose processor in MPSoC architecture. The novel contributions of this work are as follows:

- Utilizing STT-NV technology for dynamic reconfiguration of functional units that results in lower power, higher performance, and more thermally balanced design
- Proposing performance aware reconfiguration algorithms to reconfigure functional units with the objective of performance enhancement.
- Proposing a thermal aware reconfiguration algorithm based on regional migration of computation from hot spots to cooler spots to achieve more thermal balancing.
- Comparative analysis of power, performance, and temperature of STT-NV design style versus custom CMOS that is augmented with state-of-the-art leakage reduction techniques such as power grating. This analysis is performed for various functional units to identify the best design style for each unit.

2 Related work

This section points out and discuss briefly the important papers / journals which, in past, have worked on improving the performance, reducing power and temperature of a general purpose processor.

2.1 Improving performance of functional unit

Previously, several dependencies have been targeted and were improved using micro-architecture techniques like pipelining, superscalar architecture, simultaneous multi-threading etc. Another take on performance gain is to improve the number of resources; multi core architectures have been implemented. Both homogeneous and heterogeneous multi-core architectures have been popular implementations.

In [4] they have implemented coupling of compile-time analysis routines and hardware synthesis tools, by configuring a given set of the hardware-programmable functional units (PFUs) i.e. augment the base instruction set architecture so that it better meets the instruction set needs of each application; Programmable Instruction Set Computers (PRISC), and defined a new primitive datapath operation.

In paper [30] architecture called as PipeRench is proposed, which uses a configured connections between programmable logic elements and registers using look up tables. They named this interconnected network of processing elements as a reconfigurable functional units (RFU), and the data set used to program interconnects and processing elements as configuration. Paper [31] used the reconfiguration technique mentioned in [30] and developed a prototype system; CHIMERA, which has the ability to support application specific operations; i.e. In [30,31] they have modified the control logic and are issuing the instructions to CMOS functional units depending on the behavior of the application. But, direct conversion of one functional unit to another has not been proposed yet.

2.2 Reducing power and temperature in Functional units

In order to reduce power consumption, leakage power has been targeted from several years. Other papers have been discussed which helped in reduction of functional units temperature.

In [3] they have proposed a time based approach, where the functional units are put to sleep mode when they are idle for longer times and are woken when an instruction requires a particular functional unit. A performance loss of 5% has been listed in this paper by implementing power-gating technique.

Paper [32] presents a hardware technique, where they substitute some of the power-hungry adders of a 64-bit superscalar processor, by others with lower power-consumption, and modify the slot protocol to issue many possible instructions to low power consumption units and states the loss in performance by implementing this technique.

Paper [33] concentrates on the sensitivity of leakage on temperature i.e. leakage increases with rise in temperature. This paper points to the variation of functional unit power consumption and thermal hotspot occurrence in the functional units. They introduce extremely small, accurate leakage sensors to reduce power consumption and variation. They proposed leakage-aware operation-to-FU binding mechanism (LAOFBM) and leakage-aware power gating (LA-PG) mechanisms to reduce the mean and standard deviation i.e. variation and in-turn reduce power consumption.

In [12] they presented resistive computation, which aims at avoiding the power wall by migrating most of the functionality of a modern microprocessor from CMOS to spin-torque transfer magneto-resistive RAM (STT-MRAM), a CMOS-compatible, leakage-resistant, non-volatile resistive memory technology. They implemented much of on-chip storage and combinational logic using leakage-resistant, scalable RAM blocks and lookup tables.

Paper [8] points to a thermal aware floor-plan. They have taken adjacent block temperature effect and designed a new modified floor-plan which can prevent occurrence of thermal hotspots.

Paper [20] proposed an infrared measurement setup to capture run-time power consumption and thermal characteristics of modern chips. They used infrared cameras with high spatial resolution and high frame rate to capture thermal maps. To generate a detailed power breakdown (leakage and dynamic) for each processor floor-plan unit, and employed genetic algorithms; which finds a power equation for each floor-plan block that produces the measured temperature for a given thermal package.

3 Motivation

This section motivates the work by providing insight on why functional units in general-purpose processor are performance, power and temperature bottleneck unit.

3.1 Performance

Unavailability of functional unit is one of the major performance bottlenecks in general purpose embedded and high performance processors [1, 2, 3, 4, 5]. The functional unit conflicts occur when the processor pipeline has ready instructions, but there are no available functional units to execute them. Note that in spite of high functional unit conflicts, it is not design efficient to increase the number of functional units in processor pipeline, as the complexity of additional functional unit will be significant [16, 17, 18, 19]. As studied in several works, increasing the number of functional units in general purpose processors not only increases the power consumption of the processor but also it will significantly affect the complexity of several back-end pipeline stages including instruction queue, write-back buffers, bypass stage, register file design and could severely affect the processor performance, as the number of write-back ports increases significantly [16, 17]. Only increasing the total number of functional units (which is equivalent to the maximum issue width) from 4 to 6, increases the critical path delay and the total power of the processor by 21% and 18%, respectively[16, 17]. The major increase is due to the impact on the wake-up and bypass logic of the processor.

The utilization of each functional unit in a processor is significantly different. Figure 1 shows the utilization of each of the functional units for SPECK2K benchmarks. Figure 1 (a) shows the percentage of program execution time that each functional unit is idle. Across all benchmarks most functional units are significantly idle, except for IntAlu. Figure 1 (b) reports percentage of program execution time when a functional unit was requested but was not available. In most benchmarks a significant conflict is being observed in only one functional unit, which is not the same unit for all benchmark. Results from Figure 1 (a) and (b) suggest that if we could transform the idle units to the unit with high conflict we could reduce the conflict rate and potentially improve performance.

Note that for most of the benchmarks, the functional unit with high conflict was also idle for more than 80% of program execution time. This implies that, most of the times units are accessed in a burst and remain idle for most of the time. Note that there is no single unit which has high unavailability across all benchmarks. Therefore, there is a need for reconfiguration algorithms to manage the idle resources during a resource conflict (or unavailability) to reduce the conflict rate. This reconfiguration can be achieved by using a look up table (LUT) based functional units. This thesis uses STT-NV fabric to realize this (more on this in section 4).

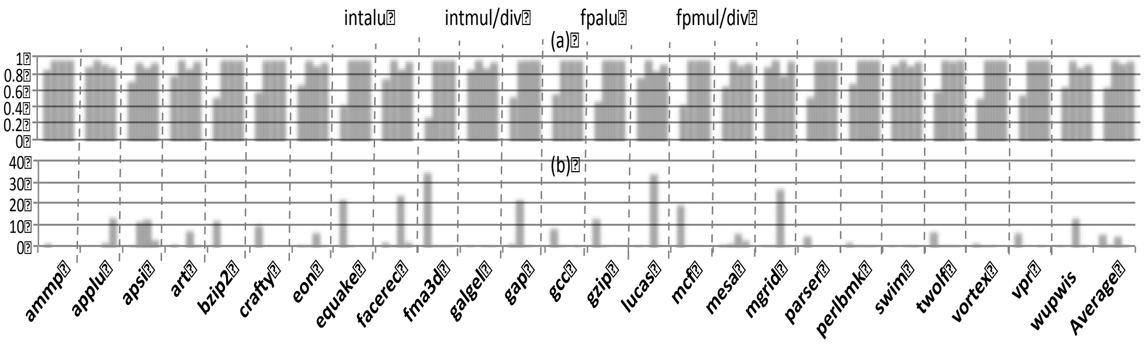


Figure 1 (a) Percentage of execution time that functional unit is idle. (b) Percentage of times functional unit requested but was not available (functional unit conflict).

3.1.1 Potential for improving performance by increasing number of functional units

In spite of large idle time for the functional units, increasing the number of functional unit improve performance significantly. Figure 2 reports the performance improvement in terms of IPC (average number of instruction committed per processor cycle) as the number of functional units increase to 2X, 3X and 4X times.

Figure 2(a) shows that increasing the number of int ADD improve performance significantly across many benchmarks. Interestingly, in spite of a very high idle time of integer mul/div, floating point add and floating point multiply and divide, increasing the number of these units, improve performance significantly for many benchmarks, as well. For instance in apsi and gap while intmul/div is idle for more than 96% of the time, doubling the number of this unit increase the performance by 13% and 23% respectively. To better understand this let's take a look at the conflict results in Figure 1(b).

Interestingly in these two benchmarks the intmul/div unit is the main source of conflict with 11% and 22%, respectively. In fact in these benchmarks the intmul/divide is requested in burst. While the average idle time is almost 95%, there are some intervals that the unit is being accessed very frequently and therefore additional intmul/divide unit during those intervals could reduce the conflict and potentially improve performance.

Also Figure 2(e) reports the speed up when increasing the number of all functional units at the same time. Doubling the number of functional units improve performance significantly by as much as 50%. The average speedup is 19%. While there are some benchmarks that tripling and quadrupling the number of functional units improve their performance substantially (applu, art, facerec, lucas, mesa, mgrid), the largest speed up is achieved when doubling the resources. Further gains are seen with increased number of functional unit, but the marginal gains drop off.

3.2 Power & Temperature

Power density of processors is increasing as technology is scaling down. High power density is known to create local hot spots, which results in excessive regional temperature and reduced reliability of the units and increases leakage current exponentially [8]. Increased cooling cost, higher probability of timing errors, physical damages, and lifetime reduction are just few of many consequences caused by higher power density. High active regions in a processor such as functional units and register file have shown to have more than 20-degree Celsius higher temperature compare to less active regions like on-chip caches [7]. In particular, functional units have shown to be a thermal hotspot component in many embedded and high performance processors [8, 20, 21].

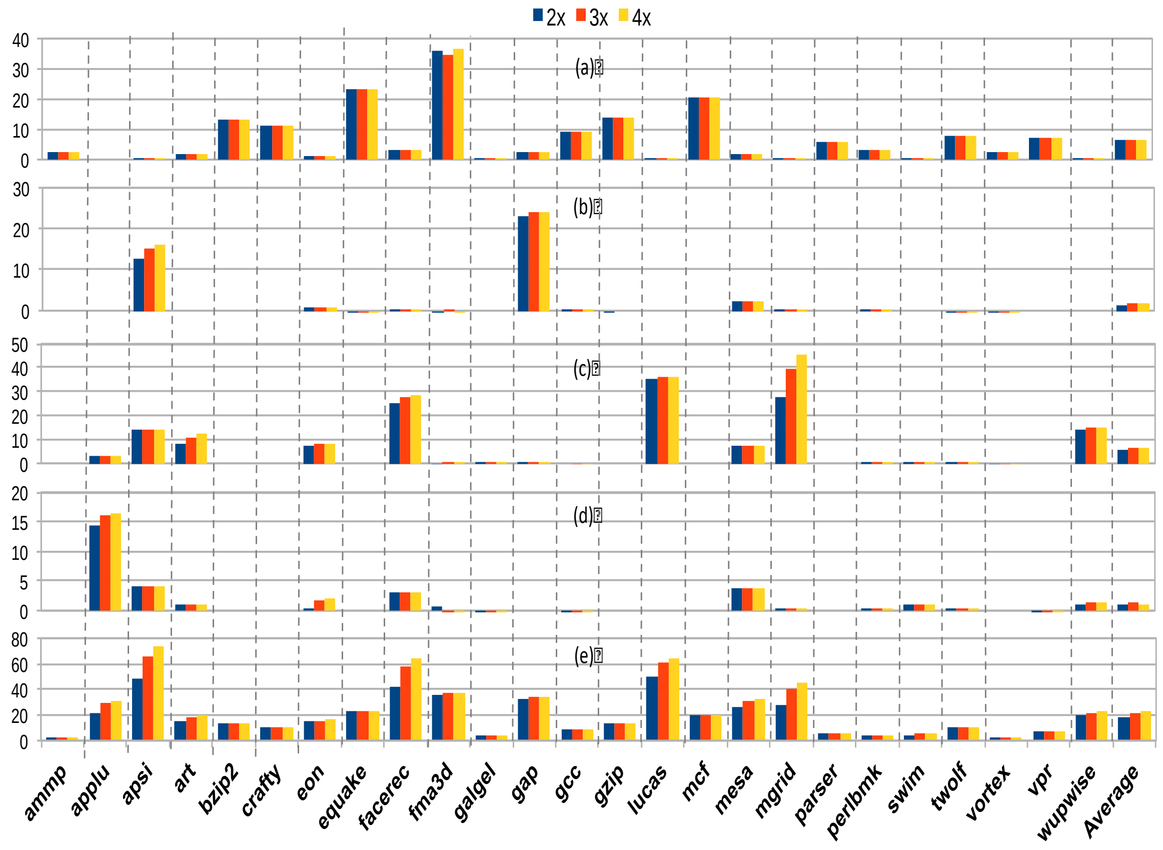
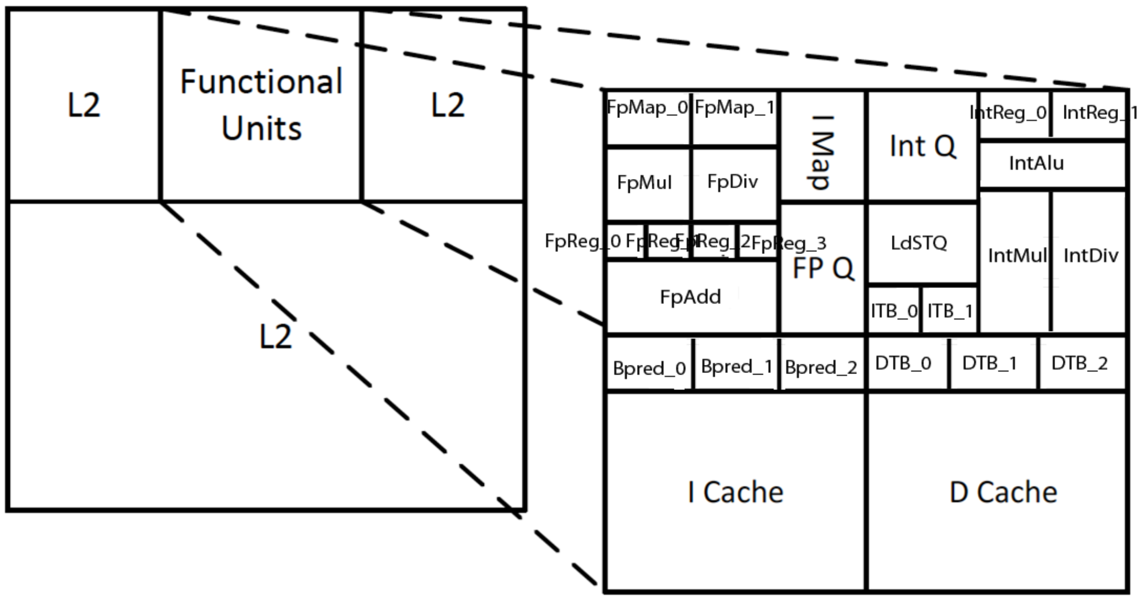
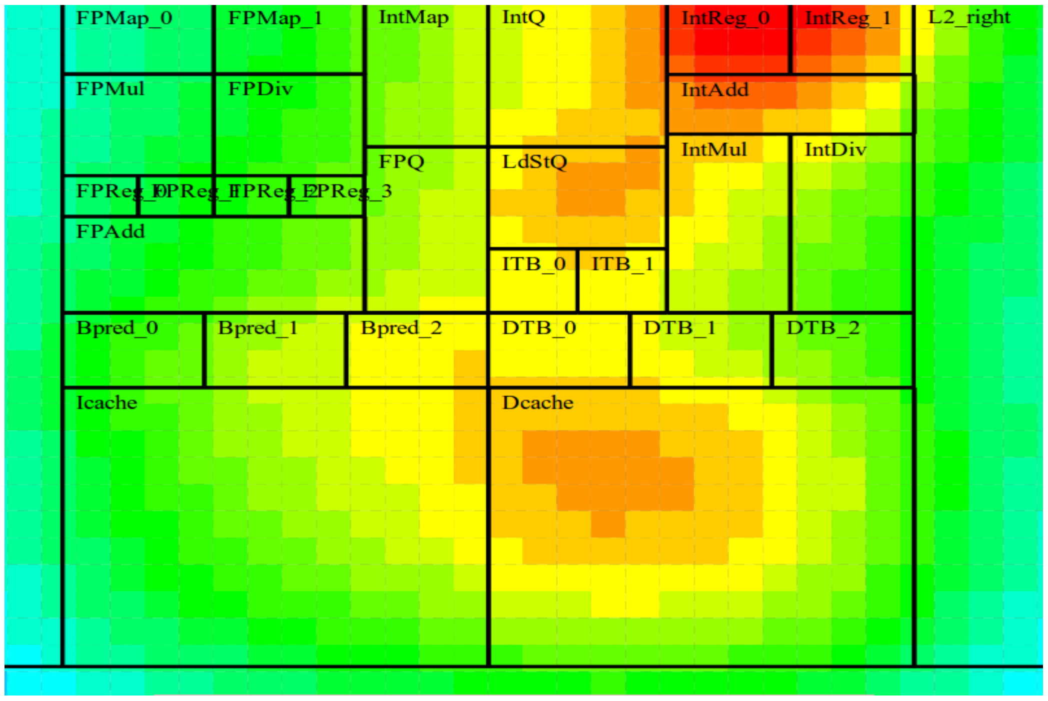


Figure 2 Relative performance improvement when the number of (a) IntADD (b) IntMUL/DIV (c) FPAdd (d) FPMUL/DIV, and (e) all units increase by 2X, 3X and 4X {vertical bar shows the % of performance improvement}



(a)



(b)

Figure 3 (a) Floor Plan (b) Processor Thermal Map

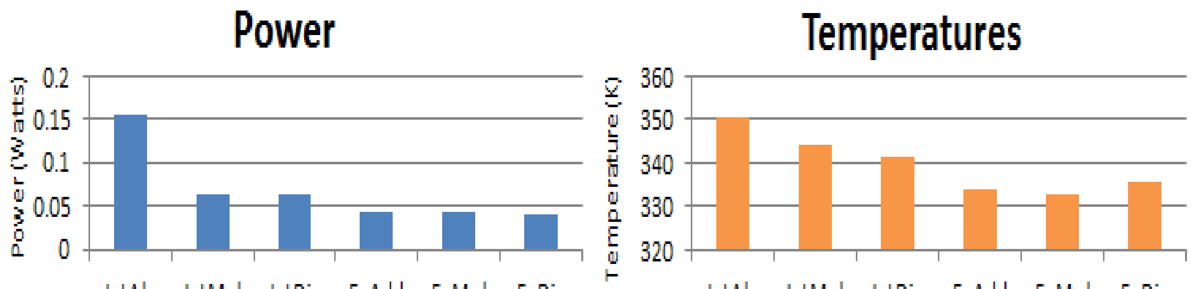


Figure 4 Power and Temperature variation in functional units

Figure 3 shows the thermal map of our studied architecture (more in section 5.1), verifying the functional units to be the thermal hotspot. Figure 4 shows the average power and average steady-state temperatures of various functional units in our studied architecture for SPEC2K benchmark. In general, temperature of a block rises because of its high activity i.e. high power density. IntAlu being highly active unit has the maximum temperature among other functional units, creating thermal hot spot. The temperature of a block not only depends on its power dissipation but also the adjacent block power 1dissipation. Due to adjacency to integer register file (shown in Figure 3) which is also a thermal hotspot, IntAlu unit temperature rises compare to other functional units.

This thesis describes the Reconfiguration and Migration (RC+M) technique to mitigate the activity of a hot functional unit to a cold functional unit and therefore reducing the maximum temperature of the functional units.

4 LUT based reconfigurable functional unit

4.1 Overview of STT-Based LUT Circuit

STT-NV technology utilizes Magnetic Tunnel Junctions (MTJ) to realize nonvolatile resistive storage. There have been several attempts to use MTJs for building logic circuits with the hope of exploiting the leakage benefit of MTJs in order to reduce the circuit power [12, 22-24]. However, due to the significant energy involved in changing the state of an MTJ, circuit styles that rely on changing the state of MTJs in response to input changes do not show any power and performance benefit [15]. An alternative to this approach has been to realize logic in memory by using LUTs that are built based on MTJs [12]. Resistive Computation [12] replaces conventional CMOS logic with Magnetic Tunnel Junction (MTJ) based Look-Up Tables (LUTs). It has been proposed for tackling the power wall. Figure 5 shows the schematic of a 3-input MTJ-based LUT that was used in [12]. An MTJ is selected by using the pull-down NMOS selection tree, and the current of the dynamic current source is divided between the selected MTJ and the reference resistor, resulting in a low swing differential voltage on nodes DEC and REF during the evaluation phase when clock (CLK) is high. This low swing voltage is then amplified using a sense amplifier stage to achieve full voltage swing outputs (Z and Z'). Figure 6 shows the plots of power, delay, and energy for LUT sizes ranging from 2 inputs to 8 inputs. This data is obtained for the cases where 50% of the MTJs are at the high state and remaining 50% at the low state. Simulations are performed in a 32nm predictive technology [11], where the expected RH and RL values are at 6.25K and 2.5K, respectively [12].

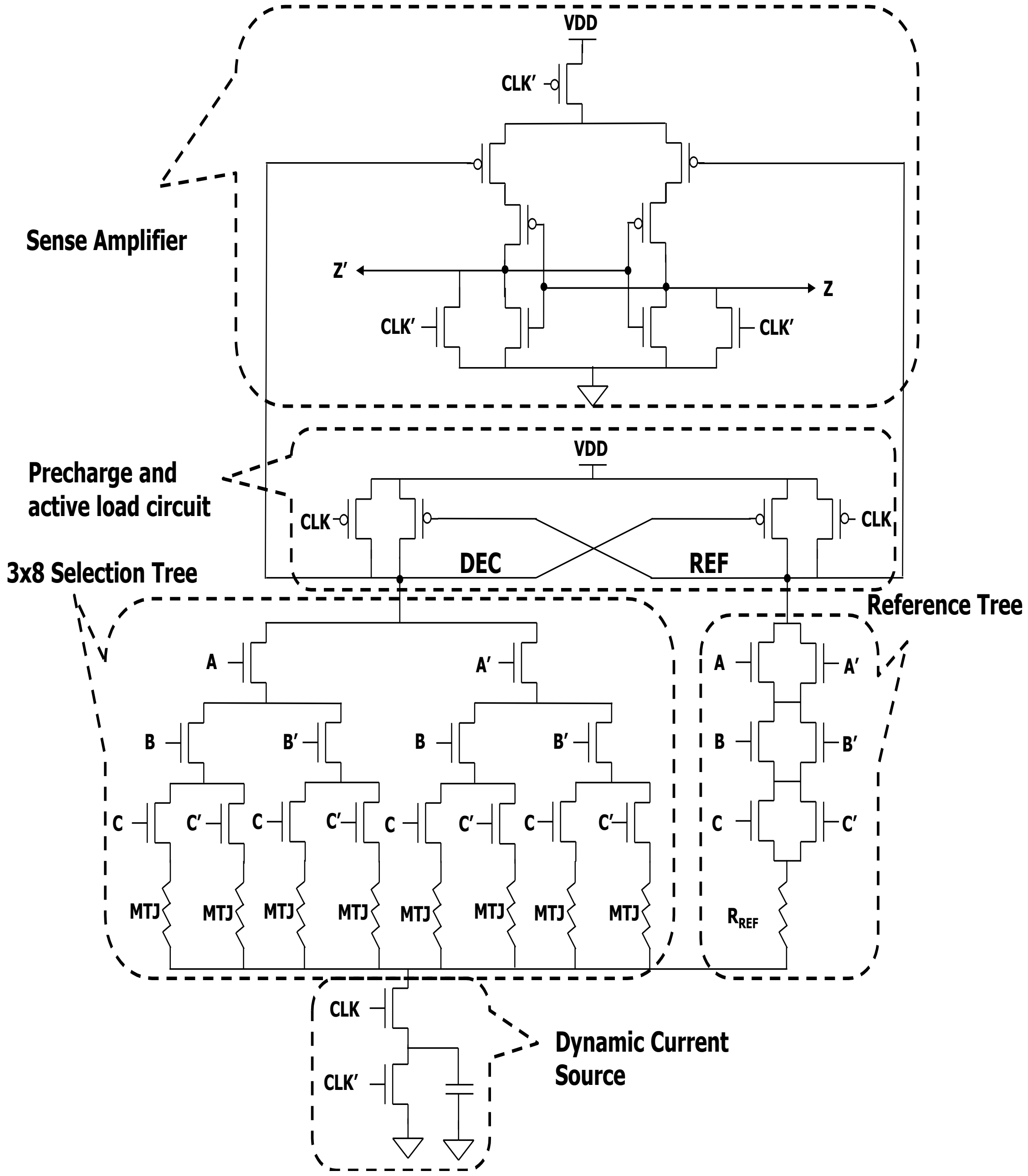


Figure 5 3-input MTJ-based look-up table [5] [9].

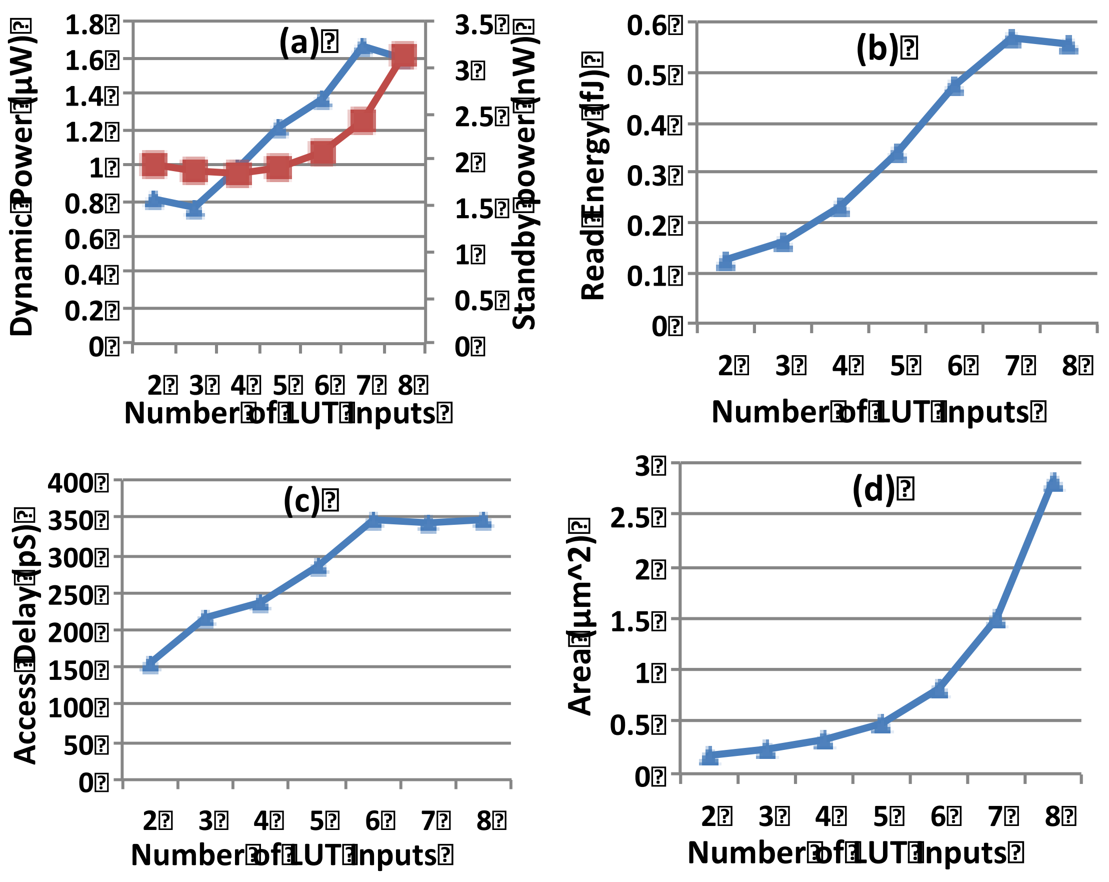


Figure 6 Power, leakage, performance, and area results of LUTs with high and low state MTJs (RH, RL).

4.2 Estimate of reconfiguration energy and performance

This estimation is performed for configuring a 64X64 multiplier unit to a 64-bit adder unit. Reconfiguring a LUT-based multiplier to an adder unit involves programming the LUTs. We have taken the HDL of multiplier and adder units and synthesized them using a commercial FPGA synthesis tool in order to get a count of LUTs needed for each design. The multiplier unit can be realized using 437 4-input LUTs and the adder using

65 such LUTs. Hence, we assume reconfiguring the multiplier unit to the adder or vice versa involve writing to at most 65 LUTs. Therefore, the total number of STT-Non-Volatile (STT-NV) bits to be written is 65 * 16 = 1040 bits or roughly 1 Kbits. The write access time to a single bit STT-NV is estimated to be 25ns [10], which is 25 cycles for 1GHz system clock. If LUTS are written in parallel using a 128-bit wide data bus, the reconfiguration is estimated to take about 8 write operations (i.e. 200 cycles).

Table 1 Comparison of adder and multiplier results in alternative styles

Metric	Unit	STT-NV LUT style	Static CMOS style
Delay	adder	2.89	1
	multiplier	2	1
Active mode power	adder	6.46	1
	multiplier	0.74	1
Standby mode (leakage) power	adder	0.17	1
	multiplier	0.23	1
Area	adder	3.89	1
	multiplier	0.90	1

The configuration bits for the LUTs that are different between the adder and multiplier configuration need to be stored in a ROM. A controller will read the configuration bits from ROM and write to the STT-NV LUTs. For configuration energy estimate, we have ignored the energy of reading the configuration bits from the ROM, since the configuration energy is expected to be dominated by the energy of writing to the STT-NV cells. Using the NVSIM tool, the write energy per bit cell is estimated to be 7.9 pJ [10]. Hence, the total energy estimated for the reconfiguration is 1040 * 7.9 pJ = 8.2 nJ. The above estimates are conservative because we assume all the bits of those 65 LUTS need to be re-written; whereas, in reality some of the bits could be same between the two configurations.

4.3 Estimate of Area, Power, and Performance

A case study is performed to obtain an estimate of area, power, and performance of an LUT based adder as compared to a static CMOS counterpart, on a 3-bit ripple carry adder implemented in both the static CMOS and the STT-NV LUT based styles in a 32nm predictive technology node [11]. Table 1 shows the results of the 3-bit adder implemented in both styles. The results indicate that except for the leakage power, the STT-NV based LUT has overhead in other metrics (especially for the adder).

This data is used to extrapolate the results for the 64-bit reconfigurable adder/multiplier unit. That means the performance of the reconfigurable adder will be 2.89X lower than that of the static CMOS adder counterpart. Its standby mode power is 0.17X lower, but its active mode power is 6.46X higher. For comparing the area, power, and performance, of an LUT based multiplier with a static CMOS counterpart, we use the results reported in [12]. Due to larger delay of reconfigurable STT-NV multiplier compare to the baseline CMOS style, the STT-NV multiplier implementation is pipelined two times deeper than the original CMOS based implementation. However this has shown to impact performance minimally [12]. Also in spite of advantage of a CMOS based multiplier over the STT-NV based design in terms of dynamic power, it still makes a lot of sense to replace it with the STT-NV design due to significant leakage advantage of the STT-NV design. Due to low utilization and high operating temperature of multiplier, the standby power becomes the major component of the total power.

5 Reconfiguration Techniques

This section describes the proposed techniques for reconfiguring the functional units. This section compares the proposed architecture with CMOS based functional units (baseline architecture). In baseline architecture leakage power is assumed to be suppressed using power-gating techniques reported in [6]. For the purpose of performance, power and thermal comparison this section study the following four architectures:

-CMOS+PG (baseline): A design with a CMOS based functional units and power gating technique.

-STT-NV+NR: A design with a STT-NV based functional units and no reconfiguration capability.

-RC: A design with a STT-NV based functional unit and reconfigurable capability.

Note that for all the design described above, we assumed a CMOS implementation for IntAlu, since we cannot benefit much from making this a reconfigurable STT-NV unit, as it is the most utilized unit.

5.1 Static Technique (RC+ST)

In this algorithm, the application is being profiled for an initial phase (learning phase) and based on the profiling information, the reconfiguration decision is being made for the rest of program execution. During the learning period active and idle functional units are being identified. At the end of the learning period all idle units are reconfigured to active units in the order of their activity. The reconfiguration pseudo-code is shown in Figure 7**.**

```
For the first 100M cycles:
- Monitor the functional units
- Identify the idle units: idle [1, 2, 3, … i]
        (i is the total number of idle units)
- Identify the active units: active [1, 2, 3, … j]
        (j is the total number of active units)
- Order active units based on their activity: active_order [i]
At the end of 100M cycles:
Loop: for all idle units (i)
    -   Reconfigure idle units to active units:
        idle [i] → active_order [i%j]
```

Figure 7 RC+ST Pseudo code

In this technique when a unit that has been reconfigured is requested and therefore is not available it needs to be reconfigured back to its original function. This re-reconfiguration is referred as adjustment process. The adjustment process is asynchronous - For example if a multiplier is reconfigured to an adder and later in the program execution a multiply operation request a multiply unit, then the reconfigured adder need to be adjusted back to a multiplier, immediately.

Note that the reconfiguration decision is made only once and after an initial learning period (after the first 100M cycles). Since only one reconfiguration is allowed at the end of the learning phase, at most one adjustment process is performed during program execution time. This technique suits better for the application which does not change their behavior in terms of functional unit utilization very frequently at runtime. This is a low power overhead technique, since we reconfigure the units only once and hence the power overhead would be small.

5.2 Static Adaptive Technique (RC+SAT)

This algorithm is very similar to static technique except that the monitoring is done periodically. In this technique the functional units are monitored for every 100k cycles and hence have a better chance to predict and adapt to the behavior of the application. In this technique, similar to the static technique, adjustment process is used to reconfigure back a reconfigured functional unit to its initial unit. The static adaptive algorithm is shown in Figure 8.

```
Loop:
For every 100k cycles:
- Monitor the functional units
- Identify the idle units: idle [1, 2, 3, … i]
        (i is the total number of idle units)
- Identify the active units: active [1, 2, 3, … j]
        (j is the total number of active units)
- Order active units based on their activity: active_order [i]
At the end of 100M cycles:
Loop: for all idle units (i)
    -   Reconfigure idle units to active units:
        idle [i] → active_order [i%j]
```

Figure 8 RC+SAT Pseudo code

This technique suits well for the application for which the functional unit requirements change significantly at run-time.

5.3 Reconfiguration and Migration Technique (RC+M)

This technique main concern is with the temperature of the functional unit. The functional units are frequently monitored to get a temperature feedback. According to the temperature information obtained, the activity of the hottest unit is migrated to the coldest unit, 2nd hottest unit's activity to the 2nd coldest unit and so on. For activity migration, the hottest unit is reconfigured to the coldest unit and vice-versa. This technique is shown in Figure 9.

```
Loop: (for every 100k cycles)
- Monitor the functional units
- Get the temperature feedback
- Order the units from hottest to coldest based on the feedback:
        - units [1, 2, 3, ….i]
        (i is the total number of functional units)
At the end of 100k cycles:
Loop: for all units (j), j → [0,i]
    -   Reconfigure hottest unit to coldest unit:
        for ( i ≠ j) : units [j] → units [i-j]
```

Figure 9 RC+M Pseudo code

For better comparison of the result and to set motivation for this technique, an ideal case has been considered, which sets the upper bound i.e. the maximum gain this technique can achieve. The ideal case is when all the functional units have same power throughout the entire application run time. This model strives to equate the power between all the units using a perfect Activity Migration.

Note that in this technique we do not increase the number of any type of functional units, as in RC+ST and RC+SAT reconfiguration technique, instead just transferring the activity of one functional unit to another. By doing so, we migrate power from a hot unit to a cold one and hence try equating temperature between all functional units.

6 Results

This section presents our simulation methodology and the results demonstrating the performance, power and temperature benefits of the reconfiguration techniques, using reconfigurable STT-NV- logic deployed in the functional units of the processor.

6.1 Methodology

For the performance estimation, MASE simulator [26] is used. A dual issue processor is modeled, which is similar in functionality to IBM PowerPC 750 FX architecture. The baseline architecture parameter is shown in table 2. SPEC2K benchmarks suite is used for evaluation. All the benchmarks were simulated for 500M instructions after fast forwarding for 500M instructions. For thermal models Hotspot 5.02 is used [7]. In Table 2 Hotspot configuration is shown. This thesis uses a modified version of ev6 floor-plan for the 32nm technology node; floor-plan is shown in Figure 3(a). Power values required for Hotspot simulation were obtained from McPAT power simulation tool [25].

6.2 Performance Results

Figure 10 reports the performance improvement of the static (RC+ST) and Static Adaptive (RC+SAT) techniques normalized to the baseline architecture (CMOS design and without reconfiguration) for SPEC2K benchmarks. For RC+SAT the performance improvement is 16% over the baseline. For RC+ST the improvement is lower, an improvement of 13%, on average. In most benchmarks the RC+ST and RC+SAT technique are able to capture functional unit requirement at run-time and therefore a large performance improvement is observed.

Table 2 Baseline Processor Configuration and Hotspot configuration

Number of cores	4	**Register file**	64 entry
L1 I-cache	8KB, ,4 way, 2 cycles	**Memory**	50 cycles
L1 D-cache	8KB, 4 way, 2 cycles	**Instruction fetch queue**	8
L2-cache	256KB, 15 cycles	**Load/store queue**	16 entry
Pipeline	12 stages	**Complex unit**	2 INT
Processor speed and Voltage	1 GHz, 1.0 V	**Issue**	dual, out-of-order
Fetch, dispatch	2 wide	**Arithmetic units**	3 integer
Thermal Parameter	Chip thickness (m) Ambient temperature Convection capacitance Convection resistance Heat sink side Heat spreader side		0.00015 318.15 K 40 J/K 50 K/W 0.06 m 0.03 m

In all benchmarks the adaptive reconfigurable technique, RC+SAT provides larger performance benefit compare to the static RC+ST technique. Exceptions are in wupwise, art, ammp benchmarks that are highlighted with circle in the figure. In these benchmarks functional units requirement changes significantly at run-time. Therefore using RC+SAT algorithm a large number of functional unit reconfiguration is performed. The 200 cycles cost of reconfiguration overhead, thus diminishes the performance gain of RC+SAT technique in these benchmarks. Unlike RC+SAT, in RC+ST technique a very small number of reconfiguration is performed which makes the overhead very low.

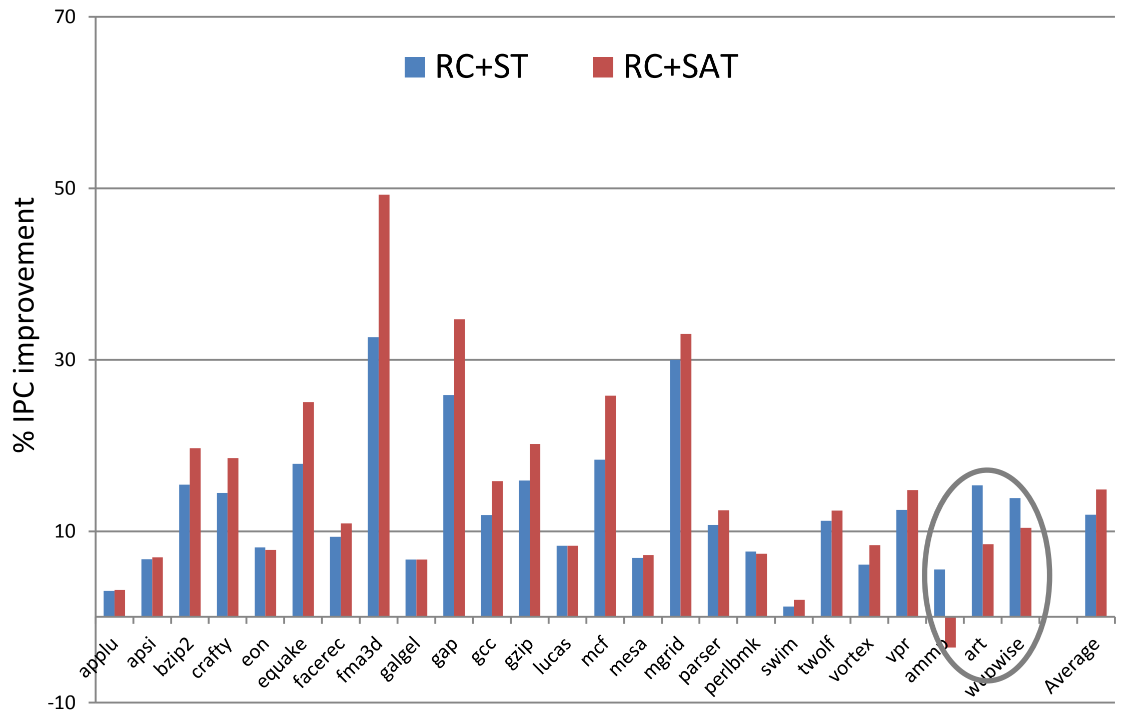


Figure 10 Performance improvement of Static (RC+ST) and Static Adaptive (RC+SAT) techniques

6.3 Power Analysis

Figure 11 presents the power dissipation of functional units in each studied design. To have a better understanding of the power dissipation among several benchmarks, Integer benchmarks (top) are separated from Floating point benchmarks (bottom). The results are averaged across SPEC2K benchmarks. For each functional unit, the power dissipation is shown for baseline design (CMOS+PG), STT-NV+NRC (without reconfiguration), RC+ST and RC+SAT. The overall power for floating point benchmarks is less than the integer benchmarks. Among all the units, IntAlu has the highest power dissipation, mainly in forms of dynamic power. The remaining functional units have significant leakage power, as they are idle most of the execution time (Figure 1).

Figure 11 reports the power breakdown of CMOS based and various STT-NV based designs presented in this work. Note that, for CMOS based design a state-of-the-art power gating technique has been applied to suppress the leakage power by up to 90% in floating points units and up to 45% in integer units. [6].

In both integer and floating point benchmarks, for IntMUL, IntDIV units the leakage power reduces in STT-NV based design compare to a CMOS based design. In integer benchmarks, for IntALU, the leakage power is lower in STT-NV designs compare to CMOS based design. Note that in CMOS based design there is small opportunity to suppress leakage using power-gating techniques, as integer unit is busy most of the times. Overall in integer benchmarks the total leakage power of all functional units increase slightly in STT-NV designs compare to CMOS based design. In floating point benchmarks the total leakage power of all functional units reduces substantially by up to

32% compare to CMOS based design (in RC+ST design). The dynamic power increases in both integer and floating point benchmarks in STT-NV designs. This is somewhat expected as STT-NV designs attempts to put more functional units into work and therefore they have higher dynamic power dissipation compare to CMOS design. Among all STT-NV designs, RC+ST in floating point benchmarks has lower total power dissipation compare to a CMOS+PG design, by 2%, on average. In integer benchmark all STT-NV designs has higher total power dissipation compare to CMOS+PG design. This is mainly due to significant rise in dynamic power for STT-NV designs compare to CMOS+PG designs. Overall, an STT-NV design (RC+ST) is more power efficient compare to a CMOS+PG design when running floating-point benchmarks. For integer benchmarks a CMOS+PG design is always more power efficient.

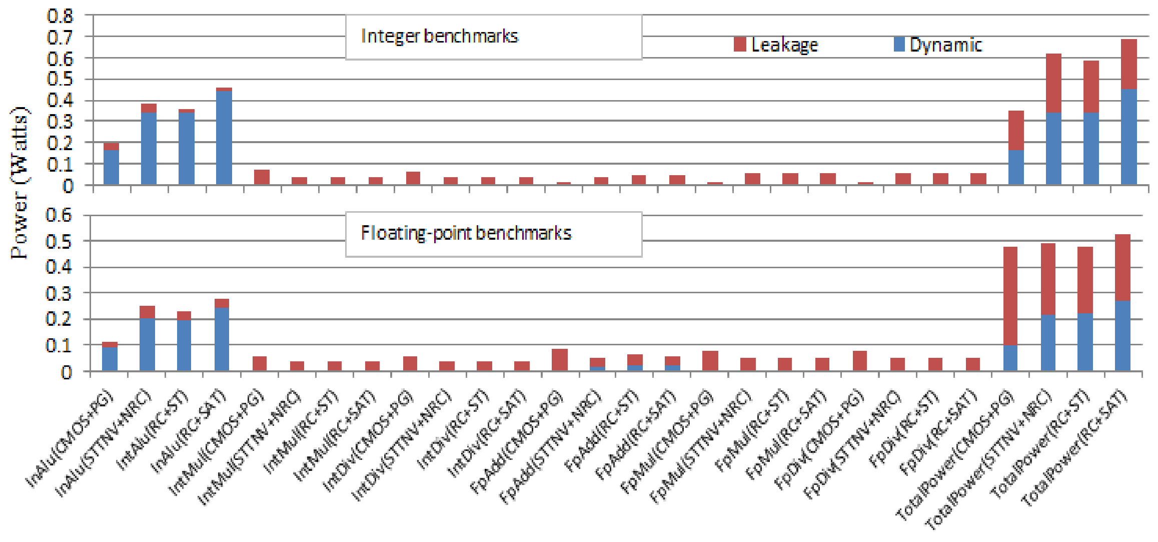


Figure 11 Power variations across functional units

6.4 Thermal Analysis

Figure 12 shows the thermal analysis of CMOS based and various STT-NV based designs studied in this thesis. The maximum temperature of each of functional units during program execution time is reported in this section. For STT-NV+NRC, RC+ST and RC+SAT the temperatures of all units increased compare to CMOS+PG. For RC+M unit a significant thermal reduction is observed. RC+M technique migrate the activity of a high temperature unit to a low temperature unit after each monitoring cycle. Hence the temperature across all units is reduced substantially.

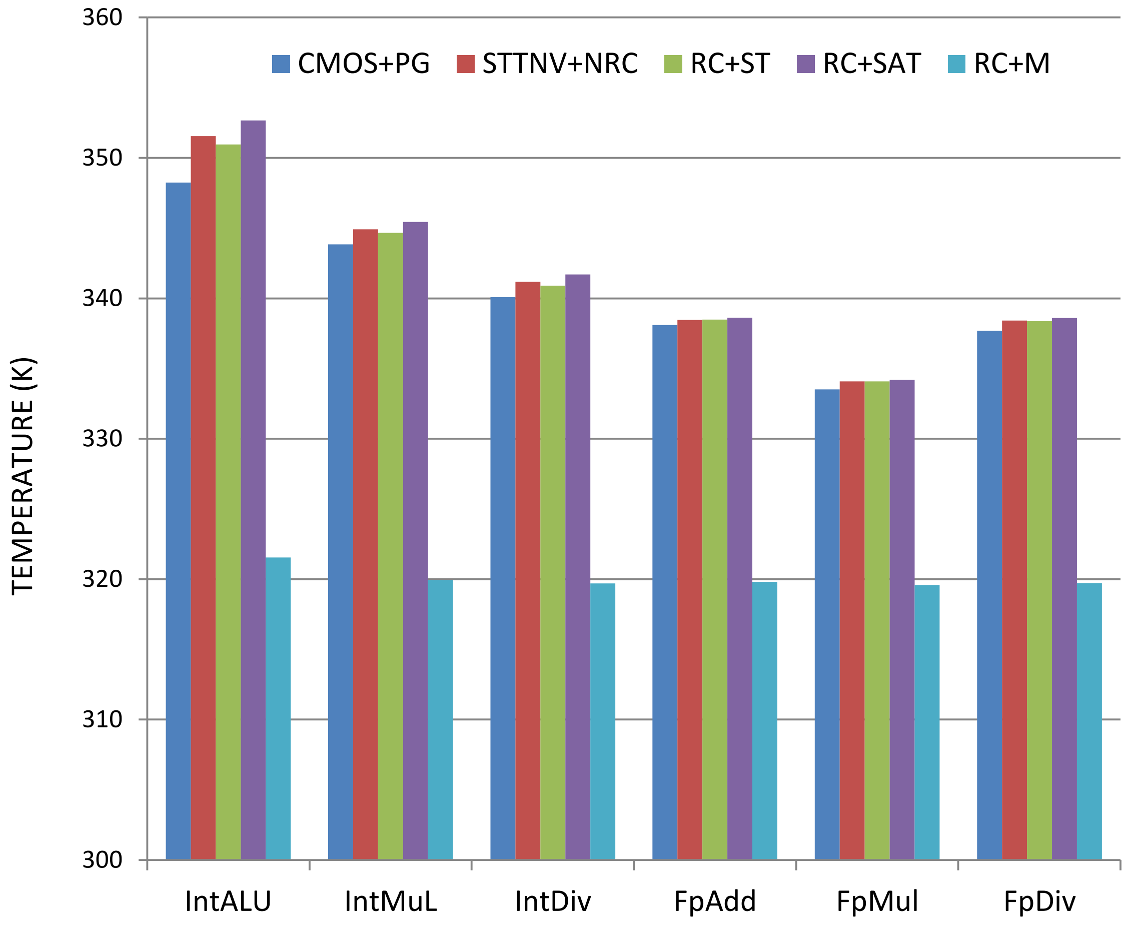


Figure 12 Thermal Analysis of functional units before and after RC+M

The largest reduction is in IntALU benchmark, by 27^{o} C on average compare to CMOS+PG design. RC+M technique reduces the temperature across all 6 functional units by more than 12^{o} C compare to CMOS+PG. Compare to a STT-NV design with no reconfiguration (STT-NV+NC), RC+ST technique reduces temperature by up to 30^{o} C (in IntALU).

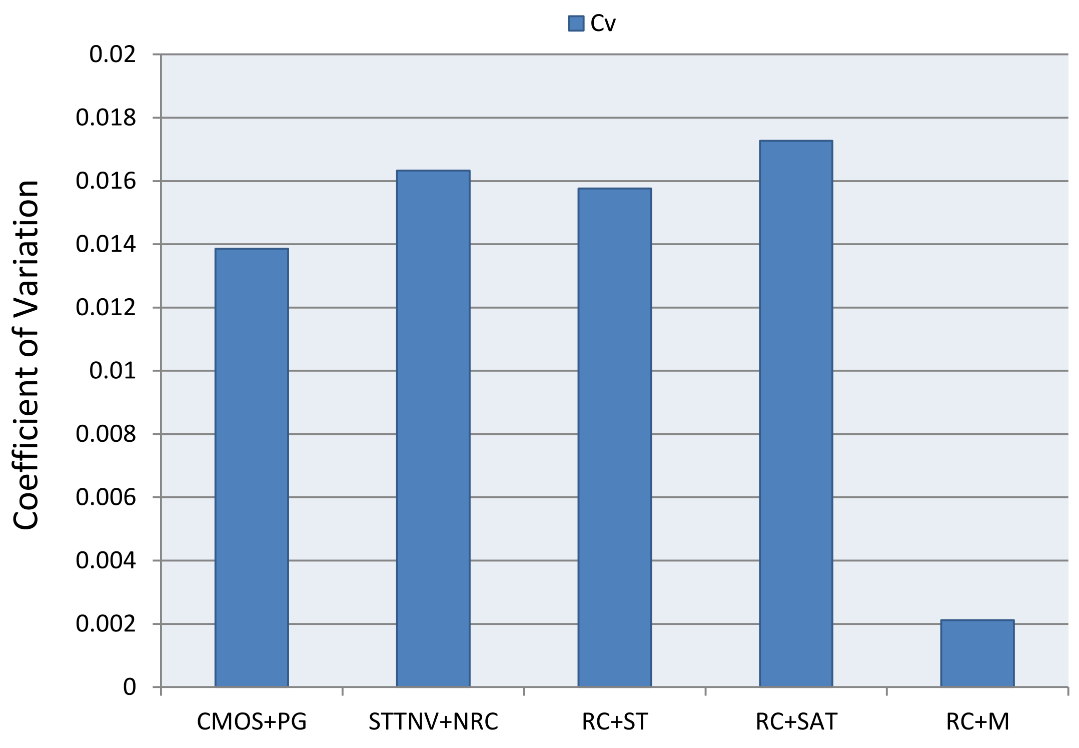


Figure 13 Coefficient of variation (Cv) among different techniques

This section also reports, how the temperature variation is affected for the proposed design compared to a CMOS based design. Temperature variation can be best measured using the coefficient of variation (Cv). The higher the Cv the larger the temperature variation is expected i.e. more thermal hot spots and vice versa. Figure 13 shows the Coefficient of variation across all 6 functional units for CMOS+PG, STT-NV+NRC, RC+ST, RC+SAT and RC+M. In our baseline CMOS+PG technique there is large 14% thermal variation across functional units. The thermal variation is also significant for reconfigurable design; for RC+SAT the thermal variation is even larger and it reaches to 16%. RC+M technique reduces the temperature variation across all functional units substantially to only 2% by distributing the concentrated power and hence reducing the coefficient of variation of temperature (i.e. reducing the power density of hot spot).

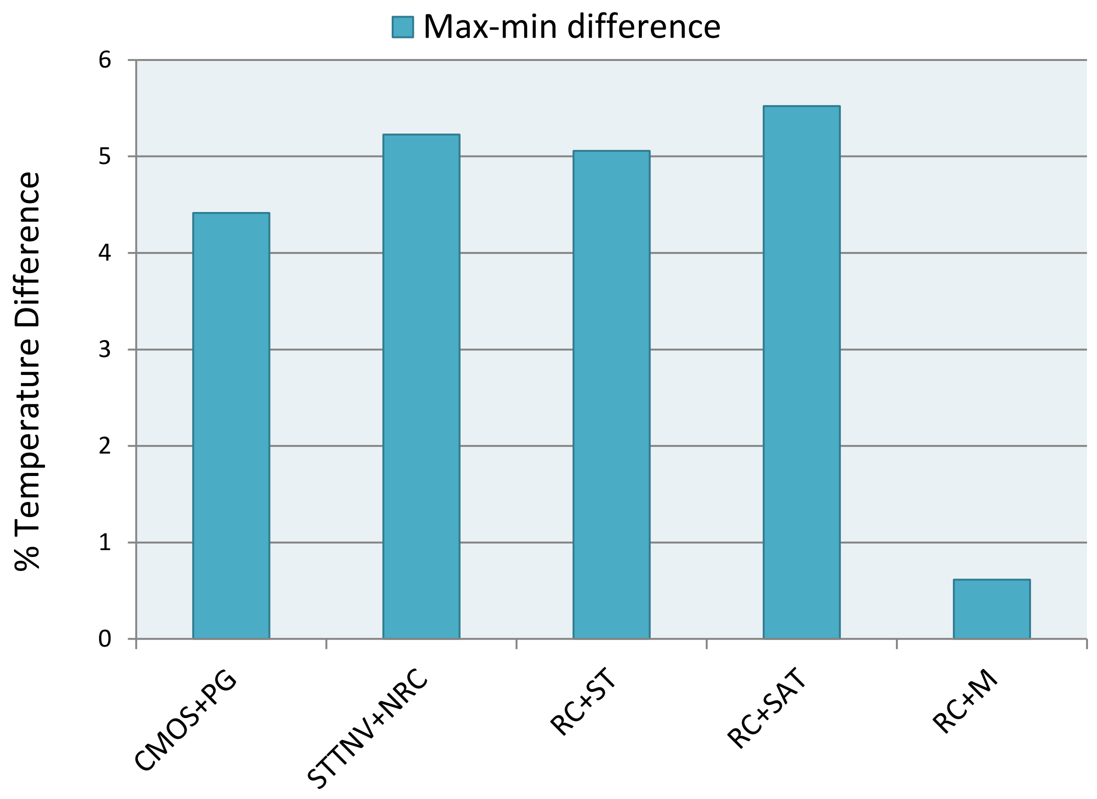
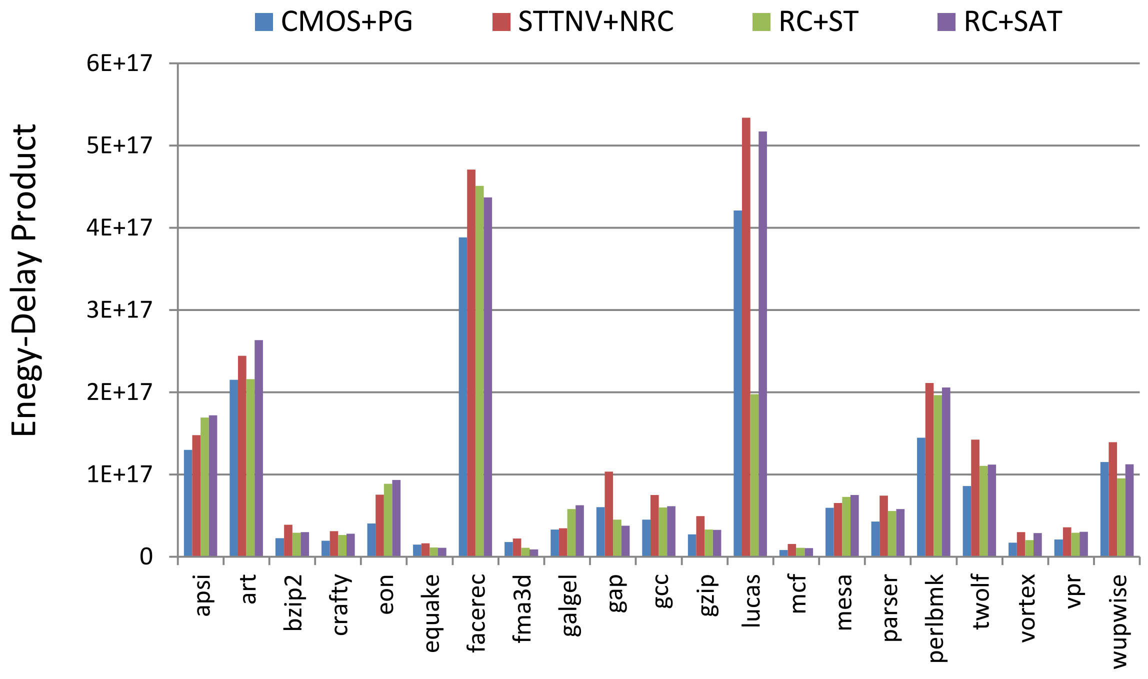


Figure 14 Difference of maximum and minimum temperatures among functional units.

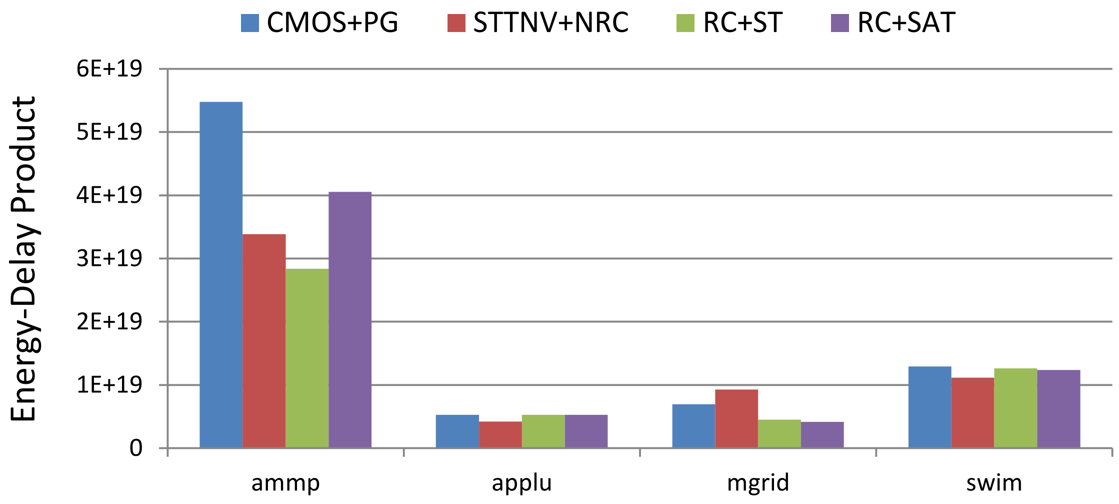
Apart from the Coefficient of variation (Cv), in-order to understand the temperature variation, a fairly simple method is to compare the differences of maximum and minimum temperatures of the functional units. Figure 14 shows the percentage difference between maximum and minimum temperatures of the functional units. RC+M techniques have the temperature difference close to zero. This technique pulls the architecture close to the ideal case i.e. architecture with no temperature variation. RC+ST and RC+SAT techniques have a greater temperature difference, but the difference is not more than 1% over the CMOS+PG baseline. Figure 14 looks similar to Figure 13, suggesting that the variation in temperature across all the functional units is mainly due to the variation between maximum and minimum temperatures and other intermediate variations impact less.

6.5 Energy-Delay Product (EDP)

Reducing the energy consumption without having a loss of performance is the best possible architecture. Energy-delay product (EDP) [29] is the product of Energy and Latency, comparing Energy-Delay Product across the discussed techniques gives the efficient way to execute an application. Low EDP indicates high performance with less energy comparison. This section presents the comparison of EDP across all SPEC2k benchmarks for the proposed techniques.



(a) Low EDP benchmarks



(b) High EDP benchmarks

Figure 15 Energy-Delay Product Comparison (a) Low EDP benchmarks (b) High EDP benchmarks

Figure 15 shows the comparison of EDP for CMOS+PG, STTNV+NRC, RC+ST and RC+SAT techniques individually for all the benchmarks. For the low EDP benchmarks as shown in Figure 15 (a), CMOS+PG have less EDP than RC+ST and RC+SAT in most of the cases. RC+ST and RC+SAT in most of the cases have EDP close to EDP of CMOS+PG. In Lucas, Art, Gap, Fma3d benchmarks RC+ST technique prove efficient.

For the high EDP benchmarks as shown in Figure 15 (b), both RC+ST and RC+SAT have a less EDP than the EDP of CMOS+PG, proving efficient.

7 Conclusion

In embedded general-purpose processors a functional unit is a critical unit that is not only a performance bottleneck for the design, but also a temperature hotspot. Due to its high activity and small size, the functional unit's power density is large, and therefore is a thermal hotspot. In addition, due to unequal activity of functional units on chip, power dissipation happen unevenly and hence a large thermal variation exists among various types of functional units. This thesis has proposed the novel concept of functional unit reconfiguration to address the performance, power, and thermal efficiency challenges. This thesis attends to these performance and thermal issues separately, by proposing techniques which improve performance or reduce on-chip temperature.

A selected set of complex functional units that might be under-utilized such as multiplier, divider etc., are realized using a shared programmable STT-NV based look up table fabric in time multiplexed fashion. This allows for run-time reconfiguration of such functional units to the functional units that might be creating performance bottleneck, and hence improving performance. The results show significant performance improvement of 16% on average across standard benchmark.

Functional units that are heavily utilized also dissipate huge power and under-utilized units dissipate less power, this cause the differences in temperature on-chip i.e. hotspots.

This thesis uses the said novel idea of functional unit reconfiguration to equate the power dissipation by reconfiguring the hottest functional unit to the coldest functional unit and vice-versa i.e. reducing the temperature differences and hence minimizing the occurrence of a hotspot. This reconfiguration reduces maximum temperature of functional units by up to 27^{o} C and almost eliminates the thermal variation across functional units.

REFERENCES

[1] Arun Kejariwal et al., “Comparative Architectural Characterization of SPEC CPU2000 and CPU2006 Benchmarks on the Intel Core 2 Duo Processor,” International Symposium on Systems, Architectures, Modeling and Simulation SAMOS VIII , 2008.

[2] Folegnani, Daniele, and Antonio González. "Energy-effective issue logic."Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on. IEEE, 2001.

[3] Hu, Zhigang, et al. "Microarchitectural techniques for power gating of execution units." Proceedings of the 2004 international symposium on Low power electronics and design. ACM, 2004.

[4] Razdan, et al. "A high-performance microarchitecture with hardware-programmable functional units." Proceedings of the 27th annual international symposium on Microarchitecture. ACM, 1994.

[5] Pleszkun, et al.. The performance potential of multiple functional unit processors. Vol. 16. No. 2. IEEE Computer Society Press, 1988.

[6] Anita Lungu, Pradip Bose, et al, “Dynamic Power Gating with Quality Guarantees” ISLPED, 2009.

[7] Huang, Wei, et al. "HotSpot: A compact thermal modeling methodology for early-stage VLSI design." Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 14.5 (2006).

[8] Y. Han, et al, “Temperature Aware Floorplanning”, Workshop on temperature Aware Computer Systems, June 2005.

[9] Ramirez, Marco A., et al. "A new pointer-based instruction queue design and its power-performance evaluation." Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005.
[10] X. Dong, et al. "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory." In IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Volume 31, Number 7, Jul, 2012.

[11] Predictive technology models. http://ptm.asu.edu/.

[12] Guo, X., et al., 2010: Resistive computation: Avoiding the power wall with low-leakage, stt-mram based computing. Power, 371–382.

[13] Das, J., Alam, S., and Bhanja, S., 2012b: Ultra-low power hybrid cmos-magnetic logic architecture. Circuits and Systems I: Regular Papers, IEEE Transactions on, 59(9).

[14] Das, J., Alam, S., and Bhanja, S., 2011: Low power magnetic quantum cellular automata realization using magnetic multi-layer structures. IEEE JETCAS, 1(3).

[15] F. Ren and D. Markovic. True energy-performance analysis of the mtj-based logic-in-memory architecture (1-bit full adder). Electron Devices, IEEE Transactions on, 57(5):1023 –1028, 2010.

[16] Palacharla, Subbarao, Norman P. Jouppi, and James E. Smith. Complexity-effective superscalar processors. No. 2. ACM, 1997.

[17] Homayoun, H., Pasricha, S., Makhzar, M., & Veidenbaum, A. (2008, June). Dynamic register file resizing and frequency scaling to improve embedded processor performance and energy-delay efficiency. In Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE (pp. 68-71). IEEE.

[18] Folegnani, Daniele, and Antonio González. "Energy-effective issue logic." Computer Architecture, 2001. Proceedings. 28th Annual International Symposium on. IEEE, 2001.

[19] Ramirez, Marco A., et al. "A new pointer-based instruction queue design and its power-performance evaluation." Computer Design: VLSI in Computers and Processors, 2005. ICCD 2005. Proceedings. 2005 IEEE International Conference on. IEEE, 2005

[20] F. J. Mesa-Martinez, J. Nayfach-Battilana, J. Renau, "Power model validation through thermal measurements", in International Symposium on Computer Architecture.

[21] K. Skadron, M. R. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, D. Tarjan, "Temperature-aware microarchitecture," in ISCA 2003.

[22] Das, J., et al "Ultra-low power hybrid cmos-magnetic logic architecture". Circuits and Systems I: Regular Papers, IEEE Transactions on, 59(9).

[23] Das, J., Alam, S., and Bhanja, S., 2011: Low power magnetic quantum cellular automata realization using magnetic multi-layer structures. IEEE JETCAS, 1(3).

[24] Karunaratne, D. K. and Bhanja, S., 2012: Study of single layer and multilayer nano-magnetic logic architectures. Journal of Applied Physics, 111(7).
[25] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, Norman P. Jouppi “McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures”, in Micro 2009.

[26] Nathan Binkert, et al. “The gem5 simulator”, ACM SIGARCH Computer Architecture News, 2011.

[27] J.-E. Lee, K. Choi, and N. D. Dutt. “Compilation approach for coarse-grained reconfigurable architectures”, IEEE Design and Test of Computers, 20(1):26-33, 2003.

[28] H. Park, K. Fan, S. A. Mahlke, T. Oh, H. Kim, and H.-s. Kim “Edge-centric modulo scheduling for coarse-grained reconfigurable architectures”, In Proc. PACT, pages 166-176, 2008.

[29] Daniel Back, Anders Loof, Martin Ronnback, Daniel Sandstrom, “Evaluation of techniques for reducing the energy-delay product in a JAVA processor” 1999.

[30] S. C. Goldstein, H. Schmit, M. Moe, M. Budiu, S. Cadambi, R. R. Taylor, and R. Laufer. "PipeRench: A Coprocessor for Streaming Multimedia Acceleration", Proceedings of the 26th Annual ACM/IEEE International Symposium on Computer Architecture, May 1999.

[31] Zhi Alex Ye, Andreas Moshovos, Scott Hauck and Prithviraj Banerjee “CHIMAERA: A High−Performance Architecture with a Tightly−Coupled Reconfigurable Functional Unit”, SEPT 2000.

[32] Guadalupe Miñana, Oscar Garnica, José Ignacio Hidalgo, Juan Lanchares, José Manuel Colmenar “Power Reduction of Superscalar Processor Functional Units by Resizing Adder-Width” Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation - pp 40-48, 2005.

[33] Shrivastava, A. Kannan, D. ; Bhardwaj, S. ; Vrudhula, S. “Reducing Functional Unit Power Consumption and its Variation Using Leakage Sensors” IEEE (Volume:18 , Issue: 6), NOV 2009.

CURRICULUM VITAE

Adarsh Reddy Ashammagari is a candidate for Master of Science in Electrical and Computer Engineering at George Mason University. He is currently a Graduate Research Assistant under Dr. Houman Homayoun. Adarsh’s research interests are High performance and Low power, Low temperature architectures for general purpose and embedded processors. Prior to this, he was a Graduate Teaching Assistant for Signals and Systems course, Bio-engineering Department. Prior to this, Adarsh was a CMS / ColdFusion Developer and helped in developing several George Mason University websites. He was awarded with “Caught in the Act” award by Administrative Services Department at George Mason University. Adarsh obtained his Bachelor of Technology form Dr. M.G.R. University, India in 2011.