Adaptable Functional Unit Design Using STT-NV Technology

Adarsh Reddy, Hamid Mahmoodi, Avesta Sasan, Houman Homayoun
George Mason University Technical Report, September 2013

Abstract

Unavailability of functional units and their unequal activity makes performance bottlenecks and thermal hot spot units in general-purpose processors. We propose to use Adaptable 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 have a 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. We developed several 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 benchmarks, when replacing CMOS integer and floating point multiplier and divider with Adaptable STT-NV LUT counterpart. In addition, reconfiguration reduces the 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 unit total power dissipation across SPECint benchmarks and almost 18% reduction in their power across SPECfp benchmarks. The Adaptable multiplier and divider design comes with no impact on area.

1. Introduction

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 points in time. The way to provide reconfigurability in designs today is primarily through the use of FPGA or Coarse Grain Reconfigurable Arrays (CGRA). For FPGAs, However, not only are there challenges in the 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.

In this paper, we present an alternative way to enable reconfigurability in an embedded processor architecture. Our solution is to enable reconfigurability 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 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 rule of thumb, 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 results in a good amount of power savings [42]. In this paper, we take the next step – We attempt to aggressively exploit STT-NV technology, by using it to design the Adaptable logic needed to support dynamic reconfiguration of functional units. We will explore ways to use reconfiguration to maximally improve power, performance and robustness of processor architecture.

As a first step, in this paper we investigate the design of a Adaptable functional unit in embedded processors. General-purpose embedded processor such as ARM, Atom, XScale, MIPS or Tensilica 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 high, and therefore is a thermal hotspot. Using STT-NV logic we are attempting to make dynamic Adaptable units to address these power, performance, and thermal challenges.

In this paper we present our analysis, demonstrating the benefits of a Adaptable STT-NV logic when deployed in the functional unit of the general-purpose processor in an MPSoC architecture. The novel contributions of this work are as follows:

- Utilizing STT-NV technology for dynamic reconfiguration of functional units resulting 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. Motivation

In this section we motivate this work by providing insight on why functional units in general-purpose processors are performance, power and temperature bottleneck units.

2.1 Performance

Unavailability of a 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 the processor pipeline, as the complexity of additional functional units will be significant [16, 17, 18, 39]. 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 will also 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

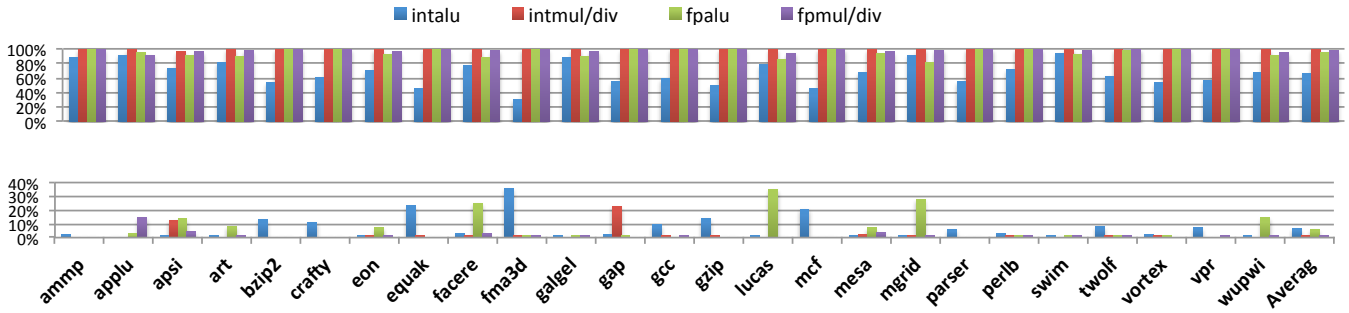

Figure 1 (a) Percentage of execution time (Y axis) that functional unit is idle. (b) Percentage of times functional unit requested but was not available (functional unit conflict).

the total number of functional units (which are equivalent to the maximum issue width) from 4 to 6, increase 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 benchmarks. 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 time, 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. In this research we use STT-NV fabric to realize this (more on this in section 3).

2.2 Power & Temperature

Power density of processors is increasing as technology is scaling down. High power density is known to create local hot spots, which result in excessive regional temperature and reduced reliability of the units and increased leakage current exponentially [8]. Increased cooling cost, higher probability of timing errors, physical damage, and lifetime reduction are just a 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 compared to less active regions like on-chip caches [7, 12]. In particular, functional units have shown to be a thermal hotspot component in many embedded and high performance processors [8, 20, 21].

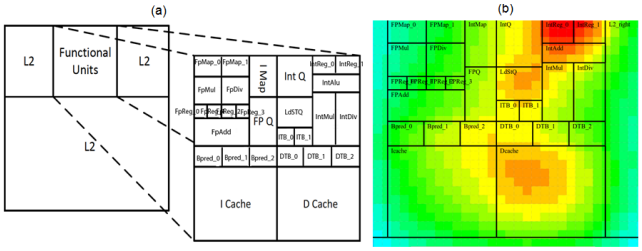

Figure 2 (a) Floor Plan (b) Processor Thermal Map

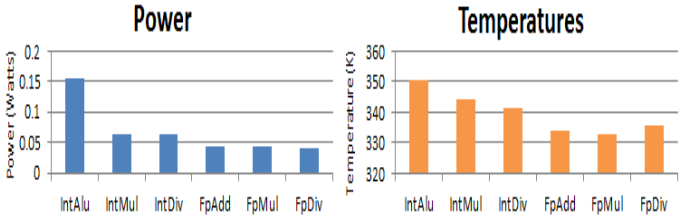

Figure 3 Power and Temperature variation in functional units

Figure 2 shows the thermal map of our studied architecture (more in section 5.1), verifying the functional units to be the thermal hotspot. Figure 3 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 a highly active unit has the maximum temperature among other functional units, creating a thermal hot spot. The temperature of a block not only depends on its power dissipation but also the adjacent block power dissipation. Due to adjacency to an integer register file (shown in Figure 2) which is also a thermal hotspot, IntAlu unit temperature rises compared to other functional units.

In this paper we describe 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.

3. LUT based Adaptable functional unit

3.1 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 [22, 23, 37, 42]. 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 benefits [15]. An alternative to this approach has been to realize logic in memory by using LUTs that are built based on MTJs [42]. Resistive Computation [42] replaces conventional CMOS logic with Magnetic Tunnel Junction (MTJ) based Look-Up Tables (LUTs); it has been proposed for tackling the power wall. Figure 4 shows the schematic of a 3-input MTJ-based LUT that was used in [42]. 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 5 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 the

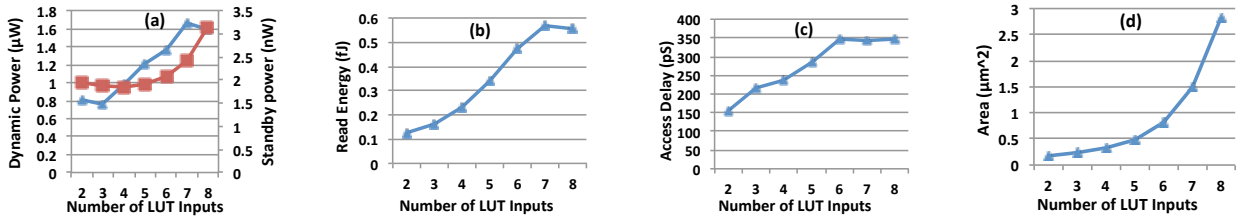

FIGURE 5 Power (dynamic: blue and leakage: red), performance, and area results of LUTs with high and low state MTJs.

remaining 50% at the low state. Simulations are performed in a 32nm predictive technology [38], where the expected RH and RL values are at 6.25K and 2.5K, respectively [42]

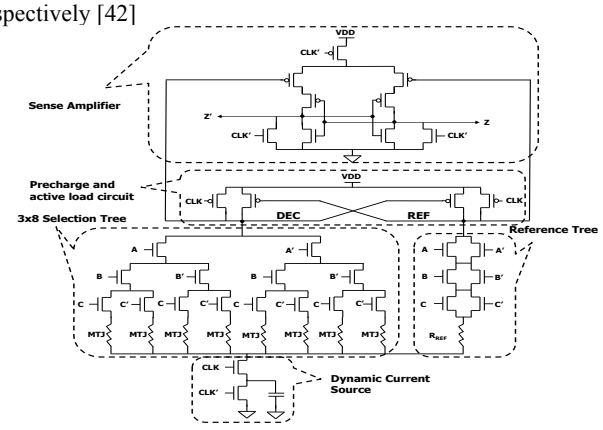

FIGURE 4 3-input MTJ-based look-up table [5] [9].

3.2 Estimate of Area, Power, and Performance

To obtain an estimate of area, power, and performance of an LUT based adder as compared to a static CMOS (ASIC) counterpart, we have performed a case study on a 64-bit ripple carry adder and a multiplier implemented in static CMOS, CMOS LUT based, and the STT-NV LUT based styles in a 32nm predictive technology node [38]. We used a commercial FPGA synthesis tool in order to get a count of LUTs and switch boxes (for routing) needed for each design. For static CMOS design we used design compiler to synthesis functional units (from the DesignWare) in a commercial 45nm technology and scaled the results to 32nm. Table 1 shows the results of the 64-bit adder and multiplier 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).

Table 1 Comparison of adder and multiplier results in alternative styles

Metric	Unit	STT-NV LUT style	CMOS LUT style	Static CMOS style
Delay	adder	2.89	3.24	1
	multiplier	2	3.73	1
Active mode power	adder	6.46	6.70	1
	multiplier	0.74	1.26	1
Standby mode (leakage) power	adder	0.17	3.87	1
	multiplier	0.23	1.42	1
Area	adder	3.89	4.61	1
	multiplier	0.90	1.83	1

That means the performance of the Adaptable adder in STT-NV style 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. Due to a larger delay of Adaptable STT-NV multiplier compared to the baseline CMOS style, the STT-NV multiplier implementation needs to be pipelined two times deeper than the original CMOS based implementation. However this has shown to impact performance minimally [42]. Also in spite of the 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 the multiplier, the standby power becomes the major component of the total power.

Also as results in the table 1 suggests, the CMOS LUT based style has no obvious advantage over the static CMOS style. While both STT-NV LUT and CMOS LUT are Adaptable, STT-NV LUT has advantage over CMOS LUT in several metrics, noticeably leakage power. The leakage power of a STT-NV style is at least 6X lower than the CMOS LUT counterpart. Based on the results presented in table 1 we select IntALU to be a non-Adaptable static CMOS as the power and area increase for a Adaptable IntALU is significant. Other functional units including multiplier and divider (Int and FP) are implemented with STT-NV LUT Adaptable style where they do not incur area overhead (the area of STT-NV LUT style is even smaller than the CMOS counterpart).

3.3 Estimate of reconfiguration energy and performance

The reconfiguration energy and performance estimation is performed for configuring a 64X64 multiplier unit to a 64-bit adder unit. This represents the worst-case scenario as reconfiguration between any other pair of functional units takes less energy and delay. Reconfiguring a LUT-based multiplier to an adder unit involves programming the LUTs. We have taken the HDL of the multiplier and adder units and synthesized them using a commercial FPGA (with 6 input LUT) synthesis tool in order to get a count of LUTs needed for each design. We have also taken into account the routing overhead including the switch boxes. 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 involves 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 are 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). 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 the 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 of LUTs 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. In addition to programming LUT we also need to program the router and switchboxes. The routing power overhead is not trivial. We used the results of FPGA synthesis to estimate the routing energy as 3.7nJ.

4. Reconfiguration Techniques

In this section, we describe our proposed techniques for reconfiguring the functional units. We compare our proposed architecture with CMOS based functional units (baseline architecture). We assume that in our baseline architecture, power leakage is suppressed using power-gating techniques reported in [6, 11, 19, 24]. For the purpose of performance, power, and thermal comparison we study four following 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+ST and RC+SAT: A design with a STT-NV based functional unit and Adaptable capability. Note that for all these designs, we assumed a CMOS implementation for IntAlu as described in section 3.

4.1 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 the 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 6. 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. We refer to this re-reconfiguration as the 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 needs to be adjusted back to a multiplier immediately. Note that the reconfiguration decision is made only once and after an initial learning period (N 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 is better suited 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, hence the power overhead would be small.

For the first N 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 N cycles:
Loop: for all idle units (i)
- Reconfigure idle units to active units:
  idle [i] → active_order [i%j]

Figure 6 RC+ST Pseudo code

4.2 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 M 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, the adjustment process is used to reconfigure back a reconfigured functional unit to its initial unit. The static adaptive algorithm is shown in Figure 7. This technique is well suited for the application for which the functional unit requirements change significantly at run-time.

Loop:
For every M 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 M cycles:
Loop: for all idle units (i)
- Reconfigure idle units to active units:
  idle [i] → active_order [i%j]

Figure 7 RC+SAT Pseudo code

4.3 Reconfiguration and Migration Technique (RC+M)

In this technique, we are mainly concerned 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, the 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 8. 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.

Loop: (for every M 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 M cycles:
Loop: for all units (j), j → [0,i]
- Reconfigure hottest unit to coldest unit:
  for (i ≠ j) : units [j] → units [i-j]

Figure 8 RC+M Pseudo code

Note that unlike RC+ST and RC+SAT in RC+M we do not dynamically change the number of any type of functional units, instead

we simply transfer the activity of one functional unit to another. Therefore this technique does not impact performance.

5. RESULTS

In this section we present our simulation methodology and the results demonstrating the performance, power and temperature benefits of the reconfiguration techniques, using Adaptable STT-NV design.

5.1 Methodology

For the performance estimation, we used a MASE simulator [26]. We modeled a dual issue processor which is similar in functionality to IBM PowerPC 750 FX architecture. Our baseline architecture parameter is shown in table 2. We used a SPEC2K benchmarks suite 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 the Hotspot configuration is shown. We developed a modified version of ev6 floor-plan for the 32nm technology node. Power values required for the Hotspot simulation were obtained from McPAT power simulation tool [25].

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)		0.00015
	Ambient temperature		318.15 K
	Convection capacitance		40 J/K
	Convection resistance		50 K/W
	Heat sink side		0.06 m
	Heat spreader side		0.03 m

5.2 Performance Results

In Figure 9 we report 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. It was experimentally determined that for RC+ST technique N=100 millions cycles and for RC+SAT technique M=1 million cycles maximize the performance and delivers a good power/performance trade-off. 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 requirements at run-time, and therefore a large

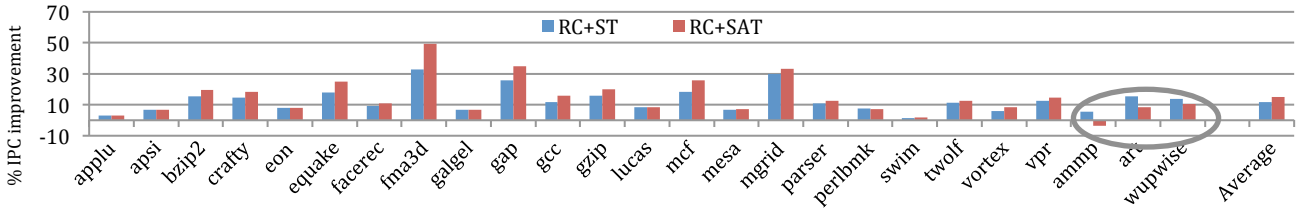

Figure 9 Performance improvement of Static (RC+ST) and Static Adaptive (RC+SAT) techniques

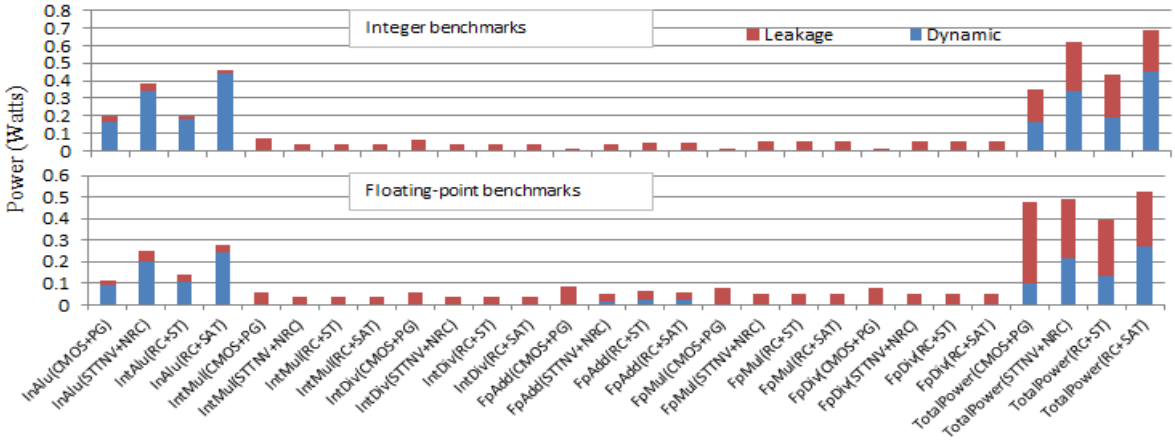

Figure 10 Power variations across functional units

performance improvement is observed. In all benchmarks the adaptive Adaptable technique, RC+SAT provides larger performance benefits compared to the static RC+ST technique. Exceptions are in wupwise, art, ammp benchmarks that are highlighted with a circle in the figure. In these benchmarks functional unit requirements change 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 diminishing 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.

5.3 Power Analysis

In Figure 10 we present the power dissipation of functional units in each studied design. To have a better understanding of the power dissipation among several benchmarks, we have separated Integer benchmarks (top) 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 the form of dynamic power. The remaining functional units have significant leakage power, as they are idle most of the execution time (Figure 1).

In Figure 10 we report the power breakdown of CMOS based and various STT-NV based designs we studied in this work. Note that for CMOS based design we assumed 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 power leakage is reduced in STT-NV based design compare to a CMOS based design. In integer benchmarks, for IntALU, the power leakage is lower in STT-NV designs compare to CMOS based design. Note that in CMOS based design, there is a small opportunity to suppress leakage using power-gating techniques, as the integer unit is busy most of the times. Overall in integer benchmarks, the total power leakage of all functional units increase slightly in STT-NV designs compare to CMOS based design. In floating point benchmarks, the total power leakage of all functional units is reduced substantially by up to 37%, compared 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 compared to CMOS design. Among all STT-NV designs, RC+ST in floating point benchmarks has lower total power dissipation compared to a CMOS+PG design, by 18%, on average. In integer benchmark all STT-NV designs has higher total power dissipation compared to CMOS+PG design. This is mainly due to a significant rise in dynamic power for STT-NV designs compared to CMOS+PG designs. Overall, an STT-NV design (RC+ST) is more power efficient compared to a CMOS+PG design when running floating-point benchmarks. For integer benchmarks a CMOS+PG design is always more power efficient.

5.4 Thermal Analysis

Figure 11 shows the thermal analysis of CMOS based and various STT-NV based designs we studied in this paper. We report the maximum temperature of each of functional units during program execution time. 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 migrates 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.

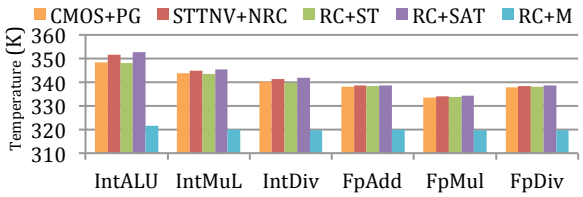

Figure 11 Thermal Analysis of functional units before and after RC+M

The largest reduction is in IntALU unit, by 27°C on average compared to CMOS+PG design. RC+M technique reduces the temperature across all 6 functional units by more than 12°C compared to CMOS+PG. Compare to a STT-NV design with no reconfiguration (STT-NV+NC), RC+M technique reduces temperature by up to 30°C (in IntALU). We also studied how temperature variation is affected for our 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 temperature variation is expected i.e. more thermal hot spots and vice versa.

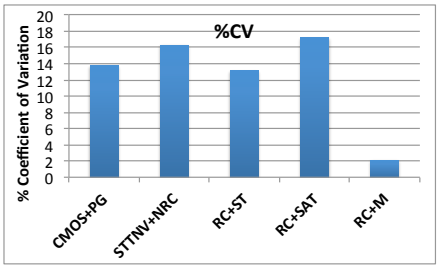

Figure 12 Coefficient of variation (C_v) among different techniques

In Figure 12 we show 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 a large 14% thermal variation across functional units. The thermal

variation is also significant for Adaptable 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).

7. Related work

The concept of Adaptable design is not new and has been explored extensively in various researches [29, 30, 31, 32, 36, 40, 41]. Several work explored the concept of augmenting GPP processor with a Adaptable array logic to accelerate performance [30].
A number of works also explored reconfiguration of functional units. Examples are PRISC[33], PipeRench [31], OneChip[35], XiRisc[39] and Chimaera[40] that proposed tightly coupled integration of a GPP with fine-grain programmable hardware. All of these designs require new programming models, new opcodes, new compilers, and source code modification. For instance Chimaera and PipeRench allowed the definition of single-cycle, Programmable Function Unit (PFU) operations using a TLB-like management and replacement scheme to virtualize the space of PFU instructions, exploiting dynamic reuse of PFU instructions. The size of the PFUOP is fixed by the architecture. PFUOPs are constrained by the sequential ISA that can only be executed sequentially. Therefore, the model does not directly, allow the architecture to scale and exploit additional parallel hardware. In addition while the main focus of these work was on accelerating performance, they are silent on power and temperature of functional unit. To the best of our knowledge our work is the first to introduce and explore the benefit of dynamic reconfigurability in the functional unit of a GPP processor using low leakage STT-NV technology where it address performance, power and temperature simultaneously. There has been also several works proposed in literature to reduce power and temperature of functional units using various combinational circuit and architectural techniques [3, 6, 8, 21]. These techniques mainly trade performance with power or temperature and can be also explored in the new Adaptable design we are proposing in this work.

6. 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 high, and therefore is a thermal hotspot. In addition, due to unequal activity of functional units on a chip, a large thermal variation exists among various types of functional units. This paper has proposed the novel concept of functional unit reconfiguration to address the performance, power, and thermal efficiency challenges. A selected set of complex functional units that might be under-utilized such as multiplier and divider are realized using a Adaptable STT-NV based look up table fabric in time multiplied fashion. This allows for run-time reconfiguration of such functional units to the functional units that might be creating performance, power or thermal bottlenecks, and hence improving computing efficiency. The results show significant performance improvement of 16% on average across standard benchmarks. Additionally, reconfiguration reduces maximum temperature of functional units by up to 27^{o}C and almost eliminates the thermal variation across functional units. The Adaptable design comes with no impact on area.

7. References

[1] Arun Kejariwal et al., "Comparative Architectural Characterization of SPEC CPU2000 and CPU2006 Benchmarks on the Intel Core 2 Duo Processor," SAMOS VIII , 2008.
[2] Folegnani, Daniele, and Antonio González. "Energy-effective issue logic." ISCA 2001.
[3] Hu, Zhigang, et al. "Microarchitectural techniques for power gating of execution units." ISLPED 2004.
[4] Razdan, et al. "A high-performance microarchitecture with hardware-programmable functional units." MICRO 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." TVLSI 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." ICCD 2005.
[10] X. Dong, et al. "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory." TCAD 2012.
[11] Homayoun, Houman, and Amirali Baniasadi. "Reducing execution unit leakage power in embedded processors." Embedded Computer Systems: Architectures, Modeling, and Simulation. Springer Berlin Heidelberg, 2006. 299-308.
[12] Homayoun, Houman, et al. "RELOCATE: register file local access pattern redistribution mechanism for power and thermal management in out-of-order embedded processor." High Performance Embedded Architectures and Compilers. Springer Berlin Heidelberg, 2010. 216-231.
[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., Makhzan, M., & Veidenbaum, A. (2008, June). Dynamic register file resizing and frequency scaling to improve embedded processor performance and energy-delay efficiency. DAC 2008.
[18] Folegnani, Daniele, and Antonio González. "Energy-effective issue logic." ISCA 2001.
[19] Homayoun, Houman, Kin F. Li, and Setareh Rafatirad. "Functional units power gating in SMT processors." Communications, Computers and signal Processing, 2005. PACRIM. 2005 IEEE Pacific Rim Conference on. IEEE, 2005.
[20] F. J. Mesa-Martinez, J. Nayfach-Battilana, J. Renau, "Power model validation through thermal measurements", ISCA 2007.
[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., Low power magnetic quantum cellular automata realization using magnetic multi-layer structures. JETCAS, 1(3).
[24] Homayoun, Houman, and Amirali Baniasadi. "Analysis of Functional Unit Power Gating in Embedded Processors." IFIP International Conference on Very LargeScale Integration System on Chip. 2005.
[25] Sheng Li, ane et. Al., "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore 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 '03.

[28] H. Park, and et. Al., Edge-centric modulo scheduling for coarse-grained reconfigurable architectures. PACT 2008.
[29] Singh, Hartej, et al. MorphoSys: an integrated reconfigurable system for computation-intensive applications. *IEEE Transactions on* 49.5 (2000).
[30] Compton, Katherine, and Scott Hauck. "Reconfigurable computing: a survey of systems and software." *ACM Computing Surveys 2002*.
[31] S. C. Goldstein, and et. al. "PipeRench: A Coprocessor for Streaming Multimedia Acceleration", ISCA 1999.
[32] Homayoun, Houman, et al. "Improving performance and reducing energy-delay with adaptive resource resizing for out-of-order embedded processors." ACM Sigplan Notices. Vol. 43. No. 7. ACM, 2008.
[33] R. Razdan, and M. Smith, "A high-performance microarchitecture with hardware-programmable functional units," in Proc. MICRO 1994.
[35] J. E. Carrillo, and P. Chow, "The effect of reconfigurable units in superscalar processors," in Proc.of the 2001 ACM/SIGDA FPGA, 2002.
[36] Houman Homayoun, Vasileios Kontorinis, Ta-Wei Lin, Amirali Shayan and Dean M. Tullsen. "Dynamically heterogeneous cores through 3D resource pooling". International Symposium on High-Performance Computer Architecutre, HPCA, 2012.
[37] Karunaratne, D. K. and Bhanja, S., 2012: Study of single layer and multilayer nano- magnetic logic architectures. J. of Applied Physics, 111(7).
[38] Predictive technology models. http://ptm.asu.edu/
[39] A. Lodi, et al., "A VLIW Processor with Reconfigurable Instruction Set for Embedded Applications," IJSSC 2003.
[40] Zhi Alex and e. al. CHIMAERA: A High−Performance Architecture with a Tightly−Coupled Reconfigurable Functional Unit, ISCA 2000.
[41] Homayoun, Houman, et al. "Dynamic register file resizing and frequency scaling to improve embedded processor performance and energy-delay efficiency." Design Automation Conference, 2008. DAC 2008. 45th ACM/IEEE. IEEE, 2008.
[42] Guo, X., et al., 2010: Resistive computation: Avoiding the power wall with low-leakage, stt-mram based computing. Power, 371–382.
[43] A. Makhzan, H. Homayoun, A. Eltawil, F. J. Kurdahi. "Process Variation Aware Cache for Aggressive Voltage-Frequency Scaling". Design, Automation & Test in Europe, DATE 2009, Nice, France.
[44] Le-Nguyen Tran, Houman Homayoun, Fadi Kurdahi, Ahmed Eltawil "Heterogeneous Memory Management for 3D-DRAM and External DRAM with QoS", 18th Asia and South Pacific Design Automation Conference (ASP-DAC), 2013.