On the Power Management of Simultaneous Multithreading Processors

Ahmed Youssef, *Student Member, IEEE*, Mohamed Zahran, *Senior Member, IEEE*, Mohab Anis, *Member, IEEE,* and Mohamed Elmasry, *Fellow, IEEE*

***Abstract*— SMT processors are widely used in high performance computing tasks. However, with the improved performance of the SMT architecture, the utilization of their functional units is significantly increased, straining the power budget of the processor. This increases not only the dynamic power consumption, but also the leakage power consumption due to the increased temperature. In this paper, a comparison of the static and dynamic sleep signal generation techniques for SMT processors is presented. This is conducted under various workloads to assess their effectiveness in leakage power management. Results show that the dynamic approach exhibits a three-fold increase in leakage savings, compared with that of the static approach for certain functional units.**

I. Introduction

Due to the tremendous increase in the number of on-chip transistors, multicore Simultaneous MultiThreading (SMT) chips have become affordable and mainstream [1]. SMT cores are the building blocks of multicore chips, such as the SUN Ultra-SPARC T2 processor, and the Intel i7. In SMT architecture the processor executes several threads simultaneously in the same pipeline. The hardware multithreading is combined with superscalar technology increasing the throughput and making better use of the pipeline resources [1]. The increased execution throughput results in an increase in the overall power consumption.

In SMT processors, the functional units are one of the major power consumers due to their high utilization which is translated into an increase in their dynamic power consumption. Additionally, the exponential increase in the leakage power, associated with the scaling of transistors' dimensions, means that the active leakage power in these functional units becomes a bottleneck in processor power budgeting. Although dynamic power consumption can be justified as a cost of operation, the active leakage is a constant cost that has no benefit. Power gating techniques are employed to reduce the leakage power consumption during the periods of functional units inactivity. Power gating usually employs a sleep transistor between the functional unit and the supply rails. When the functional units are inactive, the sleep transistor is shutdown through the assertion of a sleep signal. During the period when the sleep signal is asserted the leakage through the functional units is substantially decreased. Research in [2] shows that power gating of the functional units can save up to 18% of the total power. Since the floating point units in the UltraSPARC presented in [3] consume up to 22% of the core's dynamic power, power gating these functional units can save approximately 4% of the core's power.

Predictive leakage power reduction [4] [5] has been introduced as one of the most efficient techniques for reducing leakage of functional units in traditional processors. The success of predictive techniques depends on the predictability of the program running. However, with SMT processors, several programs are running simultaneously which affects predictability; and the whole technique of predictive leakage power reduction needs to be re-assessed.

There has been a large body of work in power-management and temperature aware computing [6] for traditional superscalar processor [7]. With the advent of SMT and mutlticore processors, more challenges emerge and also more opportunities. The challenges are that now there are more than one thread executing simultaneously, causing more power dissipation and thermal hot-spots. The opportunities is that we can use the capabilities of SMT that it can schedule and adjust the execution of each thread running on the system [8]. However, most of the work done in that area concentrates on scheduling of threads to increase performance and thread migration to reduce hot-spots. This work allows designers to understand how efficient are the static and dynamic schemes for generating the sleep signals in the context of SMT processors. The paper also presents how different workloads in SMT processors would impact the predictability of the static and dynamic schemes. Using that information, architects can develop SMT processors that employ both static and dynamic schemes and select which would be used based on the workload that is running. Also if an SMT processor has already a sleep-signal generator implemented, the work in this paper helps the designer implementing better thread scheduling based on the type and number of threads. Moreover, based on the complexity of the SMT (i.e. number of threads it can support), this work helps explain that sleep-signal schemes can be implemented and connected dynamically to functional units depending on the workload behavior.

To our best knowledge, this is the first study of power-management in SMT processors at the functional units level

II. Predictive Power Gating

For the SMT architecture the increased utilization of the processor by multiplexing the various threads leads to a large increase in the dynamic power dissipation. However, this increased utilization results in substantially higher operating temperatures. These elevated temperatures drive the leakage component of the power consumption higher due to the exponential relationship between the leakage and temperature. In addition, the leakage is expected to increase with transistor scaling. Accordingly, the functional units, due to their raised temperatures and their use of low threshold devices, contribute to an ever increasing leakage power.

The idea behind power gating techniques is to shut down parts of the circuit when they are not used by inserting a high threshold device, called a sleep transistor, between the functional unit and the supply rail [9]. The high threshold transistors exhibit ten times less leakage than low threshold transistors [9]. Turning off these transistors during standby reduces the leakage of the entire functional unit. However, shutting down the sleep transistors is associated with a dynamic power overhead because that arises from capacitance switching and the asserting the sleep signal from the power management unit. Therefore, the power gated circuit must remain idle long enough to break even. Accordingly, a sleep signal generator is needed to prevent the shutdown of the circuit for short sleep periods that will result in an increase in the total power. Also there is a trend to have several power-management techniques per chip. For instance in Intel latest processor code-named i7 [10], each of the 8 cores has its own power management. So there can be several sleep-signal generators per chip.

A. Youssef, M. Anis and M. Elmasry are with the University of Waterloo. M. Zahran is with the City University of New York.

To properly generate the sleep signal, several techniques have been suggested [4], [11]. The most simple and power-efficient approach to deal with sleep signal generation is the counter-based approach implemented in [12] and [11] . It relies on a Static Sleep Signal Generator (SSSG) based on a counter that triggers the sleep of the functional units of superscalar processors, if they are idle for a threshold of a prespecified number of cycles. It uses a state machine similar to the one in Fig. 1 to determine when to shutdown the functional unit. Their principal shortcoming is the inability to track the behavior of the processor. A low threshold results in erroneous shutdowns which will require wake-ups before the breakeven point is attained. On the other hand, a high threshold results in missing opportunities for more power savings.

In order to mitigate this shortcoming, the threshold is varied to match the operation of the processor based on a Dynamic Sleep Signal Generator (DSSG) [4]. DSSG dynamically changes the threshold according to the requirements of the running application. This approach limits the mutual exclusivity between the two alternatives. When the application is utilizing the functional unit in short repetitive bursts, i.e. the functional units idles frequently but for very short periods, the DSSG raises the threshold to limit the sleep signal generation in this application phase. On the other hand, when the application uses the functional unit infrequently, the DSSG lowers the threshold to cash-in on the ability to generate the sleep signal early on, maximizing the potential leakage savings. In order for the DSSG to predict the length of the standby periods, the DSSG uses a set of internal counters to keep track of how many times the standby period reached the breakeven point and vice versa. Using these counters, the DSSG bases its decision whether to raise the sleep signal, reduce or increase the threshold, on the history of utilization of the functional unit. The DSSG state machine in Fig. 2 demonstrates that the threshold is increased after repeated mistakes (shutdowns that do not remain until the breakeven point), and reduced after a series of correct decisions (shutdowns that remain beyond the breakeven point). This enables the DSSG to assert the sleep signal when it is most likely for the idle period to stay beyond the breakeven point.

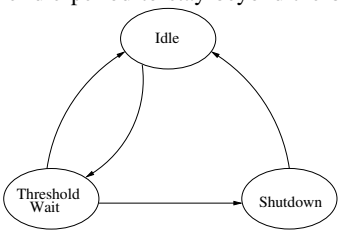

Fig. 1. SSSG state machine.

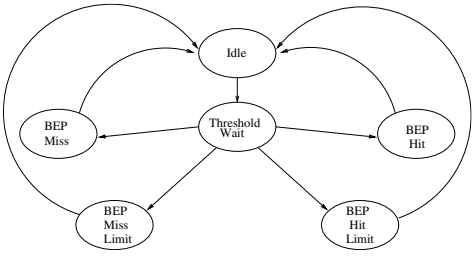

Fig. 2. DSSG state machine.

III. Experimental Results

To fully quantify the impact of the SSSG and the DSSG on an SMT processor, the accuracy of both generators should be investigated. However, adding a sleep signal generator to the functional units adds to the dynamic power overhead. Accordingly, the design and the power consumption of the DSSG and the SSSG is the second issue that needs to be analyzed. Thirdly, their performances under various workloads must be evaluated. In the following sections, these three parameters will be discussed, after the experimental setup is described.

A. Experimental Setup

To conduct the experiments to establish the idle and active periods of the functional units, a modified version of the cycle-accurate SMT-SIM simulator [13] is chosen. It faithfully implements a simultaneous multithreaded processor [1]. Table I lists the typical parameters of an SMT high performance processor. The simulator is modified to generate the required statistics.

For benchmarks, two groups of 2 and 4 thread multiprogrammed SPEC2000 workloads are selected. Each combination is allowed to run for 5 billion instructions, and all the benchmarks are compiled to the Alpha binary with the default compiler settings of the suite. To form these groups of heterogeneous workloads, similar combinations of memory-oriented and processor-oriented workloads, as in [14] are used. Table II summarizes the workload combinations. Each combination has a different proportion of memory-oriented workloads and processor-oriented workloads.

TABLE I
Test Processor Architecture.

Parameter	Value	
Fetch Queue	32 entries	
Branch Predictor	comb of bimodal and 2-level gshare; bimodal/gshar level 1/2 entries- 2048, 1024 (hist. 11), 4096 (global), resp.; combining pred. entries - 1024; RAS entries - 32 ; BTB - 512 sets, 4-way	
Branch misprediction latency	10 cycles	
Decode and Issue Width	8 instructions	
Reorder Buffer	128	
Load/Store Queue Entries	64	
Instruction TLB*	256 entry 4-way, 8K pages, 30 cycles	
Data TLB*	512 entry 4-way, 8k pages, 30 cycles	
Memory Latency	150 cycles	
L1 I-Cache*	64KB, 4-way, 64B line, 2 cycles	
L1 D-Cache*	64KB, 4-way, 64B line, 2 cycles	
L2 Unified	2 MB, 8-way, 128B line, 12 cycles	
Functional Units	Operational Latency	Issue Latency
3 Integer ALU	1 cycle	1 cycle
1 Integer Multiplier	3 cycle	1 cycle
2 Floating-Point ALU	2 cycle	1 cycle
1 Floating-Point Multiplier	4 cycle	1 cycle

TABLE II
Workloads Used

Mix (4 threads)	Mixture (2 threads)
apsi art equake mesa	applu ammp
ammp mesa swim vortex	crafty vortex
apsi crafty mcf mesa	applu vortex
ammp crafty vortex wupwise	mcf twolf
crafty vortex wupwise mesa	fma3d mesa
apsi art equake ammp	art crafty

B. Circuit Implementation and Power Consumption

Both the DSSG and the SSSG are designed with the same processor architecture and execution core. This ensures the fairness of the comparison between both techniques with respect to the associated power consumption overhead.

The following section describes the design of both the DSSG and the SSSG. Section III-B.2 presents the test circuits to emulate the functional units' circuits for the assessment of the breakeven point variation due to the introduction of the sleep signal generators. Finally, Section III-B.3 compares the DSSG and the SSSG with respect to their power consumption overhead.

1) DSSG and SSSG Circuits: In order to compare the SSSG and the DSSG, both sleep signal generators are custom-designed using a 90nm CMOS process. The designs are set to run at 2 and 4GHz to match the implementation of the functional units that will be discussed in Section III-B.2. The DSSG and the SSSG circuits are similar to the design in [4] with some transistor sizing modifications to help the migration to the new 90nm CMOS process and to achieve the higher frequencies of operation.

The DSSG circuit closely matches the operation of the state machines in Fig. 2. However, the new 90nm design requires new flip-flops and XOR gates to enhance the performance to meet the new design targets and exhibit more accurate results with respect to the current SMT processors.

To quantify the impact of the DSSG and the SSSG on the sleep signal generation process, the breakeven point of the functional units in Table I must be identified. This is achieved by adopting an emulation testbench that depends on FO4 inverter chains to emulate the functional units.

2) Functional Units' Emulation: The FO4 inverter delay is a well accepted delay metric to identify the various parameters of VLSI circuits and processor performance [15]. FO4 inverter chains are further used for power optimal pipelining [16]. The optimum clock period is eight FO4 inverter delays for an integer unit, and six FO4 inverter delays for a floating point unit [15]. Consequently, several pipelined FO4 inverter chains with power gating high threshold devices are used to to emulate the integer and floating point functional units. The size of the inverter chain corresponds to the pipeline depth in Table I. In these circuits, the inverter chains are implemented with low threshold devices and are sized for a 2 and a 4GHz operation. Additionally, the sleep transistors are sized to ensure that the maximum delay penalty, due to the introduction of the series resistance, is less than 3% [17]. This sizing also ensures that the maximum ground bounce is less than 30mV [17]. A detailed description is provided in [5].

By assuming that the functional units are designed for 32-bit operation, the FO4 inverter chains are replicated, in parallel, 64 times for each functional unit, representing the 64 inputs that will be needed for the operation. Finally, for each functional unit, the operational latency in Table I is used as an indicator for the pipeline depth. For example, the floating point multiplier has an operational latency of four cycles; this corresponds to four FO4 inverter stages separated with registers cascaded in series. The breakeven points of the integer and floating point units, without the overhead from the sleep signal generator, are highlighted in Table IV. This corresponds to the overhead of shutting down the sleep transistor network compared to the leakage savings that can be achieved.

3) Power Consumption Overhead Comparison: To compare the power consumption overhead of the DSSG and the SSSG, the baseline in Table III is compiled by summarizing the dynamic power consumption of both generators, extracted from SPICE simulations. A comparison of the dynamic power of both generators with a 64-bit ALU in the dual core UltraSPARC microprocessor, running at approximately 400 mW, highlights the minimal impact that both techniques have on the power budget of the processor core [3].

TABLE III
POWER CONSUMPTION OF THE SLEEP SIGNAL GENERATORS

Frequency	SSSG (μW)	DSSG (μW)	Global Counter (μW)
2GHz	104	266	277
4GHz	208	531	554

More important is the impact of the two sleep signal generators on the breakeven points in Table IV. Additionally, the table shows the increase in the breakeven point values for both generators and their respective global counters. The breakeven point is calculated by

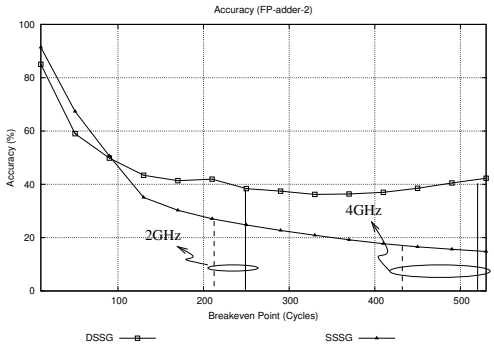

Fig. 3. DSSG vs. SSSG prediction accuracy for the floating point ALU.

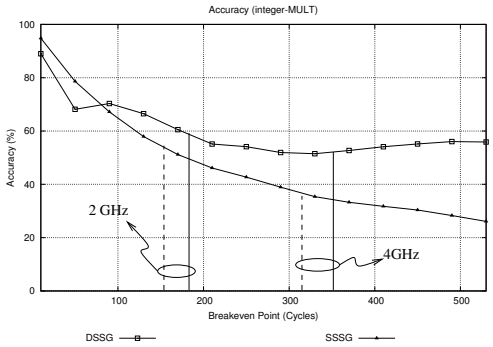

Fig. 4. DSSG vs. SSSG prediction accuracy for the integer multiplier unit.

assuming that there are four functional units to control[1], where the overhead of each sleep signal generator is calculated as follows:

$$P_{overhead} = P_{counter}/N + P_{SSG}, \quad (1)$$

where N is the number of functional units to be managed. and $P_{counter}$ is the power of the cycle counter. Table IV indicates that the breakeven points can increase from 12% to 34% by applying the DSSG instead of the SSSG. However, this does not allow for a complete comparison between both generators, since it ignores an important factor which is the accuracy of each generator in predicting the standby period length. This accuracy is a major factor, since a low overhead operation that is 40% accurate is worse than a slightly higher overhead operation that is 60% accurate. The following section analyzes the operation of both sleep signal generators under the benchmark workloads in Table II to identify their respective accuracies.

C. Accuracy of the Sleep Signal Generators

By adopting the traces extracted from the modified SMTSIM, the DSSG in [4] and the SSSG in [11] are implemented in C++. For both techniques, the breakeven point is changed from 10 to 500 cycles. These values correspond to the breakeven points, extracted earlier from the circuit simulations of both the DSSG and the SSSG in Section III-B. In addition, these values explore the design space for all the functional units in the test processor in Table I. To design the DSSG, the four parameters that control the change in the threshold: `step+`, `step-`, `miss-limit`, and `hit-limit` must be identified [4]. The `miss-limit` and `hit-limit` represent the number of misses (The functional units are shutdown without breaking even) and hits (the functional units are shutdown and they breakeven) before which the threshold is changed. The `step+` and `step-` represent the number of cycles the threshold is increased or decreased when a change is triggered by reaching either the

[1]The integer ALUs are heavily utilized, rarely remaining in standby beyond 50 cycles. As a result, no sleep signal generation circuit is attached to them.

TABLE IV
BREAKEVEN POINT OF TESTBENCH FUNCTIONAL UNITS WITH AND WITHOUT SLEEP SIGNAL GENERATORS' OVERHEAD.

Functional Unit	Frequency	Breakeven Point (cycles)	Breakeven Point (SSSG) (cycles)	Breakeven Point (DSSG) (cycles)	Percentage of Increase SSSG & DSSG (%)
Integer ALU	2GHz	136	210	278	32.71
Integer Multiplier		134	159	182	14.26
Floating-Point Adder		178	227	273	19.95
Floating-Point Multiplier		167	191	214	11.69
Integer ALU	4GHz	271	395	531	34.86
Integer Multiplier		269	309	355	14.66
Floating-Point Adder		357	438	529	20.74
Floating-Point Multiplier		334	374	419	11.97

`hit-limit` or the `miss-limit`. Accordingly, the simulations are conducted to explore the impact of the variation of these parameters between 1 and 20. This range results in the most accurate predictions by using the DSSG, as shown in Section III-C.2. On the other hand, for the SSSG, the threshold is varied between 10 and 80 cycles in steps of 10 cycles. Results in Section III-D show that the accuracy of the SSSG saturates around thresholds of 60∼80 cycles. Any increase in the threshold does not result in better performance.

Comprehensively comparing the results of the DSSG and the SSSG, regarding their capability to predict the length of the standby period, requires a common accuracy metric. In this paper, the accuracy is defined as the percentage of correct decisions to the total number of decisions, made by the sleep signal generator. The total number of decisions includes all the hits and misses. In the SSSG and the DSSG, a correct decision occurs when the sleep signal generator raises the sleep signal, and the functional unit, indeed, stays idle beyond the breakeven point. This is considered a hit. A Miss is encountered, when the sleep signal generator triggers the sleep signal but the functional unit does not stay idle beyond the breakeven point. The accuracy can be calculated as follows:

$$\text{Accuracy } (\%) = \frac{\text{Number of hits}}{\text{Number of hits + Number of misses}} \qquad (2)$$

To compare the SSSG and the DSSG techniques, the results for both techniques for the integer and floating point ALUs and multiplier units will be introduced in the following section.

1) Integer and Floating Point ALU: By referring to Table I, the test processor has three integer and two floating point ALUs. In order to test the integer ALUs, the modified SMTSIM simulator is instructed to keep track of all the instructions that utilize any of the three integer ALUs in the test processor core. This creates a standby trace that contains the information about all the periods of time during which the ALUs are employed. The evaluation of the accuracy metric in Equation (2) indicates that the DSSG and the SSSG have a very low accuracy in tracking the integer ALU units. This is expected since these units are heavily utilized and rarely stay idle long enough to allow for a shutdown.

For the floating point ALU, Fig. 3 reveals that the DSSG has a higher accuracy for predicting the standby length, especially at the higher breakeven points. The two vertical lines at 2 and 4GHz show the breakeven values from Table IV. Clearly, the small increase in the overhead, due to the application of the DSSG, is counter-balanced by the increased accuracy.

2) Integer and Floating Point Multipliers: By comparing the results for the integer multiplier in Fig. 4, it is clear that the DSSG outperforms the SSSG in accurately predicting the sleep length for the higher breakeven points. And similar to Fig. 3, the vertical lines are the actual breakeven points from Table IV. The floating point multiplier exhibits similar behavior to the integer multiplier.

Due to the very high utilization of the integer ALU units, the accuracy of both the DSSG and SSSG is low, and saturates for the breakeven points above 200. But still the accuracy of the DSSG is higher than that of the SSSG.

D. Workload Dependence

In this section, the effect of the workload characteristics on the accuracy of DSSG and SSSG is explained. This effect is an important parameter when choosing between the DSSG and the SSSG based on the expected system workload.

The workloads to test the DSSG and the SSSG are categorized into processor-bound workloads, memory-bound workloads, and neutral workloads. Since Processor-bound workloads are expected to have high utilization of execution units, the prediction accuracy is expected to be low. When a processor is executing a single thread, the prediction accuracy depends mainly on the thread. However, when several threads are executed simultaneously, as in the case of SMT, we see a combined behavior of several threads. Two predictable threads when combined can produce a less predictable behavior depending on their usage of processor resources and whether there will be contention. Alternatively, memory-bound workloads access the memory system more often, and hence the utilization of the execution units will be lower and the prediction accuracy higher. The neutral workloads fall somewhere in between as displayed in Figure 5. The other execution units exhibit similar behavior. The SSSG exhibits similar behavior but with a lower accuracy. It is important to note here that the behavior of the neutral workload differs with the execution unit and the number of threads. So sometimes the neutral has better accuracies than the memory and sometimes it is worse.

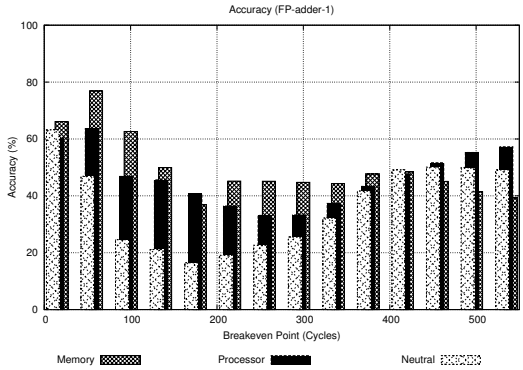

Fig. 5. DSSG prediction accuracy for floating point add/sub. unit running processor/memory/neutral-bound workloads

Another factor that affects the utilization of the execution units is the number of threads in each workload. Figure 6 portrays the DSSG accuracy for a floating point adder/subtracter with two-thread and four-thread workloads. As it is clear from the figure, the accuracy for the two-thread workload is higher due to the lower utilization compared to that of the four-thread workload.

It can be concluded that as the functional unit utilization increases, it becomes more difficult to make a prediction. However, the accuracy of the DSSG remains higher than that of the SSSG, especially at the high breakeven points. Another issue that should be further explored is the effect of the instruction scheduling [18] in SMT on power

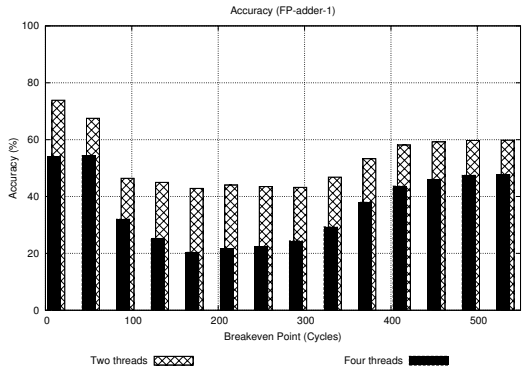

Fig. 6. DSSG prediction accuracy for floating point add/sub. unit running two and four thread workloads

dissipation. Instruction scheduling affects the performance of each thread, as well as the functional unit utilization. Therefore, instruction scheduling, in the context of SMT and power dissipation, static and dynamic, is part of the research agenda.

E. Leakage Savings Potential for Sleep Signal Generators

To fully characterize the performance of the DSSG and the SSSG, their leakage savings for the microprocessor's functional units in Table I needs to be identified. To achieve this, the impact of the accuracy performance, discussed in Section III-C, is described in terms of the number of cycles the functional units are idle after the breakeven point. The number of cycles after the breakeven is the real performance metric, since the accurate prediction of the breakeven point guarantees that only the processor is not losing power. It is the counting of the cycles beyond the breakeven that will determine the actual leakage savings. However, the number of cycles, where there is a pre-breakeven wake-up, must be deducted from the savings, since these cycles represent wasted power.

Table V lists the leakage savings per cycle, multiplied by the net savings achieved by each signal generator in cycles and the clock period to exemplify the achievable leakage energy savings. Table V also provides the difference in savings between the DSSG and the SSSG. The SMT traces are used to simulate all the 2 and 4 thread benchmarks running in series. However, due to the limited history dependence of both the DSSG and the SSSG, the trends in Table V are expected to remain independent of the order in which the benchmarks are run.

The results in Table V are calculated using the best performing threshold for the SSSG (ranging between 50 and 80 cycles) since the accuracy of the SSSG saturates beyond 80 cycles. For the DSSG the best performing combination of the parameters `step+`, `step-`, `miss-limit`, and `hit-limit` is used.

TABLE V
ENERGY SAVINGS FOR SSSG AND DSSG ON SMT PROCESSORS.

Functional Unit	#	Freq.	Leakage Savings (μW)	Total Energy Savings (μJ)		Diff. (%)
				SSSG	DSSG	
Integer Multiplier		2GHz	7.10	158.42	158.42	0
Floating-Point Adder	1		3.55	53.40	61.38	14.95
Floating-Point Adder	2		3.55	32.29	40.09	24.18
Floating-Point Multiplier			7.19	58.99	69.06	17.07
Integer Multiplier		4GHz	7.10	70.33	74.24	5.56
Floating-Point Adder	1		3.55	18.18	28.65	57.56
Floating-Point Adder	2		3.55	7.68	18.89	145.96
Floating-Point Multiplier			7.19	13.79	31.83	130.77

The DSSG and the SSSG are equivalent for the integer multiplier at 2GHz. In this case, the SSSG is better suited for the task due to its simplicity. This trend is broken for the other functional units at 2GHz, where the DSSG outperforms the SSSG by 15% to 24%. At 4GHz, the DSSG consistently outperforms the SSSG for all the functional units. This is attributed to the higher breakeven points, associated with the shorter clock period, and the relatively lower accuracy of the SSSG at these breakeven points. On the other hand, both the DSSG and the SSSG cannot be used to power gate integer ALUs, since they consume more power during the misses than the total power savings.

IV. CONCLUSION

Reducing the active leakage power of SMT processor through power gating is a viable approach, provided that the sleep signal generator is accurate. This paper presents the results of running static and dynamic sleep signal generators to manage the active leakage power. However, both techniques are inappropriate for the heavily utilized integer ALUs. The static approach is found to be more adequate for the integer multiplier units at lower frequencies, whereas the dynamic approach is more suitable for the remaining functional units. Finally, the workload-dependency of these signal generators is studied with an emphasis on the processor, neutral and memory-bound applications.

REFERENCES

[1] Tullsen et al., "Simultaneous Multithreading: Maximizing On-Chip Parallelism," in *Proc. ISCA*, 1995, pp. 392–403.
[2] Tschanz et al., "Dynamic Sleep Transistor and Body Bias for Active Leakage Power Control of Microprocessors," *JSSC*, vol. 38, no. 11, pp. 1838–1845, Nov. 2003.
[3] Takayanagi et al., "A Dual-Core 64-bit UltraSPARC Microprocessor for Dense Server Applications," *JSSC*, vol. 40, no. 1, pp. 7–18, Jan 2005.
[4] Youssef et al., "Dynamic Standby Prediction for Leakage Tolerant Microprocessor Functional Units," *Proc. Micro*, pp. 371–384, Dec. 2006.
[5] A. Y. et al., "A Comparative Study between Static and Dynamic Sleep Signal Generation Techniques for Leakage Tolerant Designs," *IEEE Transactions on VLSI*, no. 9, pp. 1114–1126, Sept. 2008.
[6] K. Skadron, M. Stan, W. Huang, S. Velusamy, K. Sankaranarayanan, and D. Tarjan, "Temperature-aware computer systems: Opportunities and challenges," *Micro, IEEE*, vol. 23, no. 6, pp. 52–61, Nov.-Dec. 2003.
[7] D. Brooks and M. Martonosi, "Dynamic thermal management for high-performance microprocessors," in *HPCA '01: Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, 2001.
[8] J. Donald and M. Martonosi, "Leveraging simultaneous multithreading for adaptive thermal control." in *Second Workshop on Temperature-Aware Computer Systems (TACS) in conjunction with ISCA-32*, 2005.
[9] Wei et al., "Low Voltage Low Power CMOS Design Techniques for Deep Submicron ICs," *Proc. VLSI Design*, pp. 24–29, Jan. 2000.
[10] "Intel Nehalem Processor (i7)." [Online]. Available: http://www.intel.com/products/processor/corei7/index.htm
[11] Hu et al., "Microarchitectural Techniques for Power Gating of Execution Units," *Proc. ISLPED*, pp. 32–37, 2004.
[12] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicrometer CMOS Circuits," *in Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, Feb. 2003.
[13] Tullsen et al., "Simulation and Modeling of a Simultaneous Multithreading Processor," in *Proc. Computer Measurement Group Conference*, 1996, pp. 819–828.
[14] Choi et al., "Learning-Based SMT Processor Resource Distribution via Hill-Climbing," in *Proc. ISCA*, 2006, pp. 239–251.
[15] Hrishikesh et al., "The Optimal Logic Depth Per Pipeline Stage is 6 to 8 FO4 Inverter Delays," *Proc. ISCA*, pp. 14–24, May 2002.
[16] Heo et al., "Power-Optimal Pipelining in Deep Submicron Technology," *Proc. ISLPED*, pp. 218–223, 2004.
[17] Anis et al., "Design and Optimization of Multi-Threshold CMOS (MTCMOS) Circuits," *Trans. CAD*, vol. 22, no. 10, pp. 1324–1242, Oct. 2003.
[18] Parekh et al., "Thread-Sensitive Scheduling for SMT Processors," *Technical report, Department of Computer Science and Engineering, University of Washington.*, 2000.