Characterization of Logic Circuit Techniques for High Leakage CMOS Technologies

Phillip Chin, Charles A. Zukowski
Columbia Integrated Systems Laboratory
Columbia University, New York, NY
pchin@cisl.columbia.edu, caz@columbia.edu

George D. Gristede, Stephen Kosonocky
IBM T.J. Watson Research Center
Yorktown Heights, NY
gristede@us.ibm.com, stevekos@us.ibm.com

ABSTRACT

Channel subthreshold and gate leakage currents are predicted by many to become much more significant in advanced CMOS technologies and are expected to have a substantial impact on logic circuit design strategies. To reduce static power, techniques such as the use of monotonic logic and management of various evaluation and idle modes within logic stages may become important options in circuit optimization. In this paper, we present a general, multilevel model for logic blocks consisting of logic gates that include a wide range of options for static power reduction, in both the domains of topology and timing. Existing circuit techniques are classified within this framework and experiments are presented showing how aspects of performance might vary across this range in a hypothetical technology. The framework also allows exploration of optimal mixing of techniques.

Categories and Subject Descriptors:
B.7.1 [Integrated Circuits]: Types and Design Styles - *VLSI*

General Terms:
Design, Experimentation, Performance.

Keywords:
Monotonic logic, leakage current, low power.

1. INTRODUCTION

Power dissipation has become an extremely important constraint in modern microprocessor design as CMOS technologies continues to advance. This power issue is driven by concerns about circuit reliability, packaging costs, and the proliferation of mobile devices dependent on battery life.

Historically, power dissipation in CMOS circuits has primarily been the result of the charging and discharging of load capacitances, referred to as dynamic power dissipation. However, as we begin to enter the realm of sub-100 nm technology, static power consumption is expected to become much more important. The maximum number of transistors on chips will increase dramatically. Supply voltages will continue to scale to reduce dynamic power, and threshold voltages will decrease to maintain transistor switching speeds. This will result in an increase in subthreshold current conduction. Also, decreasing gate oxide thicknesses will reach the 1.2-1.5 nm regime, where direct tunneling current will become significant. This combination of subthreshold and gate leakage will have a substantial impact on idle-state leakage currents, and greatly increase the standby leakage power of highly integrated circuits [1]. The International Technology Roadmap for Semiconductors predicts an exponential increase in leakage current over time as scaling continues [2]. These scaling issues will require consideration of both static and dynamic power in future circuit designs [3–6].

The potential impact of subthreshold and gate leakage currents must be well understood, either to deal with this growing problem or to evaluate the technology tradeoffs involved in avoiding it. A number of specific circuit techniques have been proposed that might help deal with this issue. Previous work in leakage resistant circuit topologies can be placed into two categories, critical path and non-critical path techniques. Non-critical path techniques that reduce leakage current at the expense of increased circuit delay include transistor sizing, transistor stacking [7–9], higher threshold voltages [10], lower supply voltage, and thicker oxides: collectively, these techniques have been referred to as statically-selected slow transistors (SSSTs). In critical paths, idle portions of circuits with fast, leaky devices are deactiveated with techniques such as body biasing [11], input vector control [12], and sleep transistors [10]: these techniques have been collectively referred to as dynamically-deactivated fast transistors (DDFTs) [13]. However, the entire spectrum of circuit options has not been fully explored yet, and the potential for mixing techniques has received little attention. Furthermore, studies have often been limited to technologies that are not too different than current ones, and have focused primarily on subthreshold conduction alone.

In this paper, we present a general model in Section 2 that includes likely topology and timing approaches for restoring logic circuit design. A framework is introduced to allow optimal mixing of circuit techniques. A key contribution is the inclusion of monotonic logic in its basic form, based on observations that it has some key advantages for high leakage technologies. Section 3 presents examples of how some specific circuit techniques can be viewed as a special instance of the model. Preliminary experiments are presented in Section 4, showing how our design framework can be used to produce a partially monotonic circuit that optimizes circuit characteristics.

GLSVLSI'04, April 26–28, 2004, Boston, Massachusetts, USA

2. GENERAL MODEL

To construct a very general logic circuit model, we must allow the use of either a static or dynamic approach for each logic gate, or even some combination of the two. Currently, entire CMOS logic blocks are often assumed to be either exclusively static or dynamic (e.g. domino), or have a well defined partition between static and dynamic portions. But in the future, the distinctions between static and domino logic may fade somewhat. As technologies scale, noise constraints may lead to the addition of static loads in domino circuits. In the other direction, power issues may lead to static logic with clocking to reduce power during idle phases of the clock cycle. Our model reflects the existence of a more continuous spectrum between the two as shown in Figure 1.

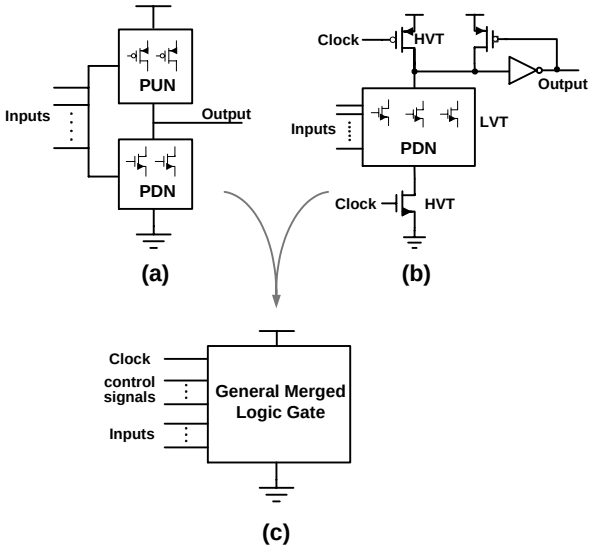

Figure 1: Example (a) static, (b) domino, and (c) general merged logic gate

The key attractive feature of domino logic for static power reduction is the asymmetry in signal transitions. Since only the pullup or pulldown is in the critical evaluation path in each stage, the fastest and leakiest switches can always be placed in series with slower and less leaky ones. This advantage arises, however, from the necessary monotonic property of the logic synthesis, and not from the dynamic nature of the circuit. Static logic can also avoid particularly leaky static paths if it has a similar monotonic property and a reset capability [14]. In addition, monotonicity can potentially reduce dynamic power by eliminating glitching. So in our model, we separate out the issue of monotonicity and the static vs dynamic choice. Even the reset option and technique can be a distinct question for each gate.

In general, monotonic logic circuits require monotonic input signals, and their gate count overhead increases with logic depth. As a result, it is natural to consider the use of monotonic logic in the initial stages of a logic block, and at some point (when the overhead becomes too large) transition to more general logic. If the transition is at the very beginning, the circuit would not be monotonic at all and would be completely general. In the other extreme, if the transition point were at the outputs, the logic would be completely monotonic. Our model allows circuits to cover this entire spectrum in search of an optimum. Within the monotonic portion of a logic block, static or domino techniques can be used, but outside, a purely domino approach cannot due to potential signal glitching.

2.1 Monotonicity

As a first step, monotonic logic must be defined more carefully as in [14] and [15].

In a random logic network, logic gates are not usually skewed to favor a certain switching direction since outputs can either switch high or low during evaluation. However, in a monotonic network, logic gates can be severely skewed to favor a certain switching direction by sizing up devices or using low V_t devices to speed up evaluation. High-skewed (HS) gates have monotonically rising outputs, and low-skewed (LS) gates have monotonically falling outputs.

In a logic network, a gate can be guaranteed to switch in only one direction during evaluation by enforcing the monotonicity rule: a monotonic gate must have all inputs coming from a monotonic gate of the opposite type. We can ensure that every other gate output in a path is either monotonically rising or falling.

However, we end up with a switching direction of a gate that is slow. Reset/precharge and evaluation phases are often required to take full advantage of monotonic logic. During reset, all monotonic gates are set to their initial state. During evaluation, each gate output either stays in its initial state or monotonically transitions. Figure 2 demonstrates LS and HS gates and their evaluation phases. The clock/reset control into the PDN and PUN may be necessary to help prevent short circuit currents during reset, depending on the status of the input signals at that time.

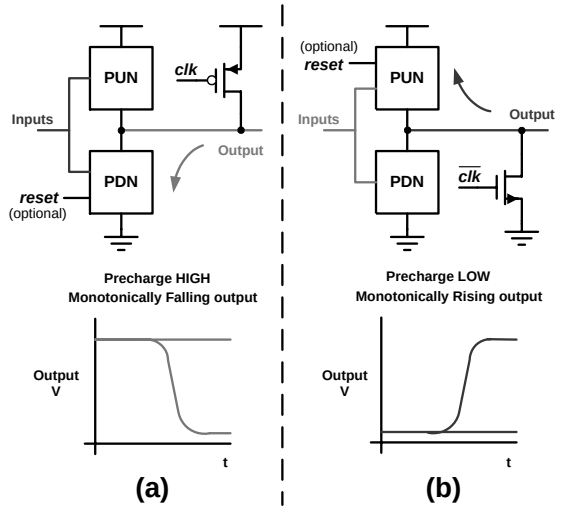

Figure 2: Example (a) low and (b) high skew gates

Resetting of monotonic logic increases the circuit overhead, but it can vastly improve the circuit performance. In general, it may be best to only directly reset the first level or strategically selected gates throughout, and let the logic propagate initial values [16, 17]. It is even possible that portions of a static, monotonic circuit would not warrant resetting at all.

To implement a monotonic circuit, the logic network must be unate, and all trapped inverters must be pushed out to

network boundaries. Binate to unate algorithms have been developed to remove trapped inversions and optimize networks [18]. Each monotonic gate is limited to a single type of input signal (i.e. either monotonically falling or monotonically rising). If both polarities of a signal are required (i.e. a trapped inversion), logic duplication is necessary to remove the inversion and provide both signal polarities with the same type. Logic duplication will at most double the amount of original logic while maintaining the same logic depth [14, 15].

2.2 Logic Block Level of Model

Our logic model begins with a generalized logic partitioning involving monotonic and non-monotonic logic stages. A general way to look at the impact of static leakage power is to see how it may affect the optimal location of this boundary as a function of technology characteristics as shown in Figure 3. Historically, the boundary has often been thought of as being between domino and static logic, but when leakage becomes significant, we expect that such an association will become less clear. The more important issue will be the monotonicity, where the monotonic logic might usually be standard static CMOS with dual threshold devices.

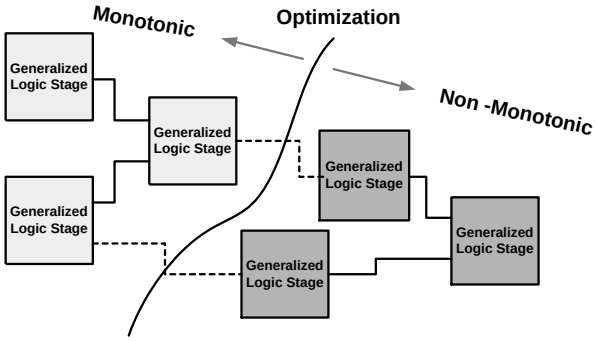

Figure 3: General Logic Partitioning

2.3 Logic Stage Level of Model

Shown in Figure 4 is a generalized logic stage that spans a wide range of circuit techniques, dynamic or static or mixed. Each box in this logic stage represents an optional variable conductance designed for specific tasks, whether for pull up or pull down, precharge high or precharge low, or for a complex keeper circuit to maintain noise margins of a dynamic node. The clocks in Figure 4 refer to general timing control signals which could range from a simple global clock to asynchronous handshake signals. It is a general model, where not all conductance boxes are required for a particular gate. Inclusion of conductance boxes depends on the context of the circuit. A static gate would have just P and N networks and may not use any clock inputs. A domino gate would have a keeper, but the P network may just be precharge.

Through general timing control inputs, a wide range of timing options are possible for either static or dynamic gates. In general, logic circuits only perform evaluation during small time windows when waves of activity pass by, as illustrated in Figure 5, where circuit mode is graphed as a function of time and logic depth. In between evaluations, a

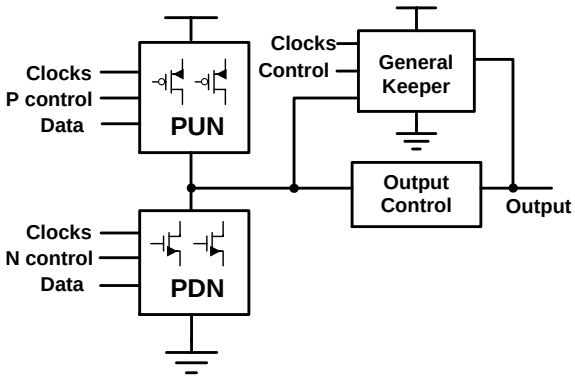

Figure 4: Generalized Logic Stage

logic stage could require a reset (e.g. a dynamic precharge), or might benefit from one (e.g. in monotonic static logic). Resets can occur all at once as shown in Figure 5 for simplicity, or pass through logic blocks in waves ahead of or behind evaluations (e.g. by using timing chains, self timed logic, or self resetting logic). In the period before and after evaluation, there is the potential for placing logic stages in some kind of lower power sleep state, particularly if no evaluation is required for long periods (e.g. many clock cycles in a synchronous system). Through the careful design of the time varying conductances pictured in Figure 4, each logic gate could potentially be kept in low power configurations as much as possible without significantly degrading other aspects of performance. Although the model contains general timing control inputs, there is of course overhead and uncertainty associated with producing these inputs that must be estimated or accounted for in some manner during optimizations.

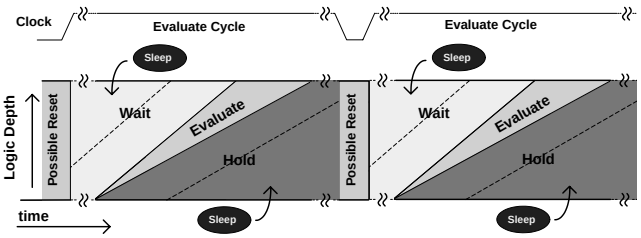

Figure 5: Example Spatial Cycle Time Window

3. APPLICATIONS

One of the applications of our general model for logic blocks constructed from leaky devices is to facilitate a systematic exploration of the design space. A first step is to classify existing techniques and measure their performance in a general environment. Some circuit techniques that have been proposed for universal logic families may not turn out to be very useful for that purpose, but might be advantageous for certain individual logic gates within a larger logic network. This section contains a few examples.

Our first example in Figure 6 illustrates a variable conductance in the pull down network of a particular gate. Shown is a standard domino gate with an alternative clocking scheme

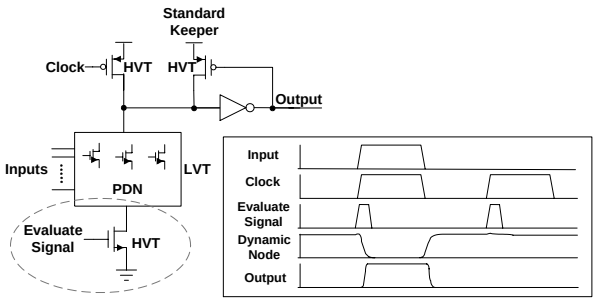

Figure 6: Example Study: Footer Device Control

for the evaluate footer device to create a time varying conductance. The footer is clocked with a short pulse that is long enough for the dynamic node to discharge, but short enough to reduce significant subthreshold leakage currents through the low V_t devices in the pull down network. For most of the cycle, the evaluate device is OFF in a lower leakage state. This technique is not limited to dynamic logic.

Some simple domino test structures designed with existing $0.13\mu m$ technologies were used to test this footer pulse technique by varying the duration of the evaluate signal. Preliminary simulations have shown potential average power savings up to 20% when the footer pulse was of the minimum duration to allow evaluation. As subthreshold leakage becomes more significant in future technologies, the potential savings may increase. Prior work involving similar techniques have been used in self resetting logic, where the evaluate signal is produced by some feedback to disable the footer device as soon as the gate has finished evaluating and the logic is no longer needed [19].

Figure 7 is a special case called a conditional keeper [20] that attempts to implement a time varying keeper conductance whose strength depends on whether or not the dynamic node discharges. There is a strong keeper and a weak keeper, with a total strength equivalent to a standard keeper. In this technique, the dynamic node is NANDed with a delayed clock to conditionally turn on the strong keeper only when the dynamic node remains HIGH. If the dynamic node discharges, the pulldown network only has to fight the weak keeper during evaluation. Only the weak keeper is on during the transition phase as seen in the figure, and the strong keeper turns on conditionally only if there is no transition. As shown in [20], this technique has the potential to reduce contention and maintain voltage levels specifically for circuits with very wide fanins. It may also be limited to logic where the evaluation wave is early in the clock cycle since the clock delay in the conditional keeper requires accurate timing. Thus, it may be natural to use this technique in conjunction with others to take advantage of their leakage control capabilities.

Our final example is a static and non-monotonic circuit technique called Output Prediction Logic (OPL) [21]. OPL potentially combines the speed of dynamic logic with the noise margins of static CMOS. Figure 8 shows a chain of static inverters combined with dynamic precharge and evaluate devices for performance improvements. The worst case behavior of a critical path is reduced by precharging all gates to a logic one, tristating them, and limiting evaluation through N trees only. However, this technique is highly dependent upon clock arrival because of the gates' inverting

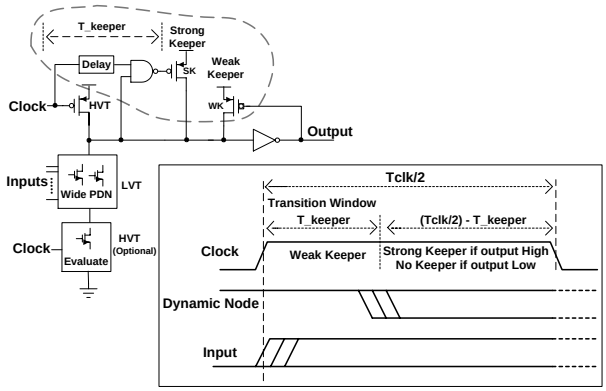

Figure 7: Special Case: Conditional Keeper

nature and the time required for inputs to stabilize. If the input is HIGH, CLK2 must not arrive until OUT1 is stable, or else significant glitching may occur. Previous work with this technique in [21] was mainly from a speed perspective, with simulations showing speedups of approximately 2.5X compared to conventional static CMOS. However, if glitching is minimized, OPL may have promising low power characteristics. Essentially, when a gate is not evaluating, we want a gate to maintain a low power configuration, and OPL disables the pulldown network before and after evaluation for a static gate. Hybrid techniques, combining static logic with precharging as found in dynamic techniques, may be helpful in order to reduce leakage but also maintain performance in the future.

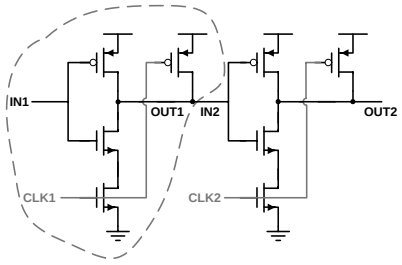

Figure 8: Example Non-monotonic technique: Chain of Output Prediction Logic Static Inverters

These three example techniques are all encompassed by our general abstract model. Many other techniques that have been proposed are as well. We propose characterizing these techniques within this general environment, assuming they could be combined in an optimal manner, with the optimum depending on technology characteristics and performance cost functions.

4. EXPERIMENTS

Our preliminary investigation explores the boundary between monotonic and non-monotonic logic blocks. By mapping portions of the logic network to monotonic circuits, we can determine if an optimal boundary exists and how dependent it is on technology characteristics.

Figure 9 shows a simple example test circuit that is mapped to a monotonic circuit. We first begin with a unate non-monotonic static circuit in (a) that is assumed to have been

		70 nm model		130 nm model	
Monotonic Levels	♯ transistors	Avg Power	Evaluation Delay	Avg Power	Evaluation Delay
none (non-monotonic)	102	199.0 μW	221.0 ps	437.4 μW	708.2 ps
2 Levels	125	192.5 μW	149.9 ps	415.8 μW	486.8 ps
3 Levels	129	194.1 μW	147.8 ps	423.3 μW	493.1 ps
4 Levels	157	209.8 μW	191.0 ps	439.8 μW	638.9 ps

Table 1: Preliminary Simulation Results

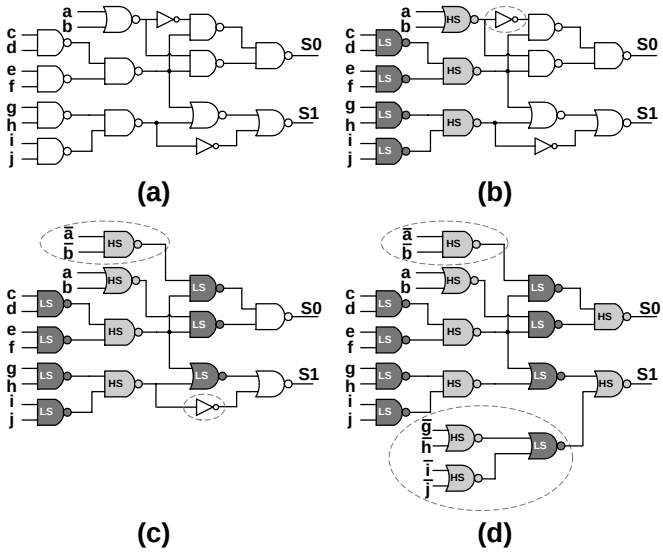

Figure 9: Mapping to Monotonic Circuits

optimized but with two trapped inverters. Example unate mapping algorithms can be found in [14,22]. Other assumptions are that all inputs (a-j) are freely available, including their complements, and ideal latches force inputs high or low to provide the correct signals during evaluation and reset.

Beginning with the first level of logic, the gates can be made either HS or LS, depending on the circuit. The requirement for monotonicity is that all LS gates must have inputs from HS gates and all HS gates must have inputs from LS gates. In (b), the first level of gates is made LS and the second level is HS. The rest of the logic is left non-monotonic. At this point, there is not much overhead. In the third logic level, there is a trapped inverter that causes a conflict. The path through the inverter creates a conflicting LS signal into the next NAND gate. To alleviate the problem, we must duplicate the logic that precedes the inverter using DeMorgan's Theorem and provide the correct signal polarity, which results in (c). Another trapped inverter exists when trying to map the fourth level of logic. We must duplicate three more gates to provide the correct signal, resulting in (d), which is a completely monotonic circuit with alternating low skew and high skew gates. As the example shows, the overhead of using monotonic circuits can significantly increase with depth.

A simple ISCAS benchmark circuit (rd53) was synthesized with the above described monotonic mapping. Synopsys Design Compiler was used to create the initial unate non-monotonic circuit using 2-input AND gates, 2-input OR gates, and inverters. Using a similar algorithm as in [14], the circuit was then mapped to static NAND, NOR, and inverter gates as shown in Figure 9(a). Following the above example, each level of logic was made successively monotonic. The input signals were randomly generated and identical for each monotonic iteration, except that precharge and evaluate cycles were required once part of the circuit was made monotonic.

Two Berkeley Predictive Technology Models (http://www-device.eecs.berkeley.edu/~ptm) were used to simulate the circuit: a 70 nm and 130 nm BPTM model with supply voltages of 0.9 V and 1.3 V respectively. These two models predict subthreshold leakage in future technology generations but they do not model gate leakage.

All gates (monotonic and non-monotonic) were sized with effective channel widths of $1.5\mu m$ for the PMOS pull up network and $0.5\mu m$ for the NMOS pull down network. Precharge transistors were added to force gates to monotonically transition during evaluation as shown in Figure 2 and sized accordingly as stated above.

The overhead of a monotonic circuit increases with depth as the number of devices rises. Also included in the overhead is the time required for precharge. Whether the precharge time is important completely depends on the context of the circuit and may or may not result in penalties in performance. For this example, we focus only on evaluation time.

Making a circuit partially monotonic can improve performance in some cases, even if resizing is not done to take advantage of the asymmetry. Table 1 shows our preliminary simulation results for the benchmark circuit. The circuit has a total of five logic levels, with monotonic mapping beginning with the first two. As the depth of monotonic gates increases, there is an optimal boundary between the second and third logic levels with respect to power and delay of the longest path. For these technologies, the 70 nm and 130 nm models, our results show improvements of approximately 3.3% and 5% for average power and 32% and 31% for evaluation delays, respectively, compared to the initial completely non-monotonic static case. As the circuit is monotonically mapped beyond the second level, the average power and evaluation delays begin to degrade due to the overhead. Logic duplication and clock loading increases power, and it also increases the loads on some gates, reducing their drive strength and increasing evaluation delays.

The preceding experiment illustrates the optimization of the monotonic/non-monotonic boundary, but does not show all of the potential advantages of monotonicity. The monotonic gates were not sized to evaluate faster in one direction, so there is capacity for even more improvement in delay. Also, the assumed technologies did not include any gate leakage currents and did not have multiple device choices (e.g. fast and leaky vs slower and less leaky). For technologies that have these, carefully designed sleep/reset states for monotonic stages could also lead to significant reduction in static power.

5. CONCLUSIONS

In this paper, we proposed a general model and framework to allow optimal mixing of circuit techniques as leakage currents continue to increase. A key issue is the use of monotonic logic, based on its asymmetric properties and the advantages it can hold in high leakage technologies. Several existing circuit techniques were presented that fit well into our logic model framework. Our preliminary experiments show that an optimal boundary between monotonic and non-monotonic logic blocks may exist that optimizes circuit characteristics such as power and evaluation delay. Our general logic model provides a framework for exploring a wide range of circuit techniques and finding optimal mixes for future high speed and low power circuit designs.

6. ACKNOWLEDGMENTS

This work was funded in part by IBM and the New York State Office of Science, Technology, and Academic Research (NYSTAR) , through the Microelectronics Design Center (MDC).

7. REFERENCES

[1] S. Borkar, "Design challenges of technology scaling," *IEEE MICRO*, vol. 19, no. 4, 1999.

[2] *International Technology Roadmap for Semiconductors (ITRS)*, 2002, (http://public.itrs.net/).

[3] J. A. Butts and G. S. Sohi, "A static power model for architects," in *Proc. of the 33rd Annual IEEE-MICRO*, pp. 223–234, 2000.

[4] D. Duarte, N. Vijaykrishnan, M. Irwin, H.-S. Kim, and G. McFarland, "Impact of scaling on the effectiveness of dynamic power reduction schemes," in *Proc. IEEE Int. Conference on Computer Design*, pp. 382–387, September 2002.

[5] R. S. Guindi and F. N. Najm, "Design techniques for gate-leakage reduction in CMOS circuits," in *IEEE International Symposium on Quality Electronic Design*, pp. 61–65, March 2003.

[6] D. Lee, W. Kwong, D. Blaauw, and D. Sylvester, "Analysis and minimization techniques for total leakage considering gate oxide leakage," in *Proc. of the Design Automation Conference*, pp. 175–180, June 2003.

[7] T. Kawahara, M. Horiguchi, Y. Kawajiri, G. Kitsukawa, T. Kure, and M. Aoki, "Subthreshold current reduction for decoded-driver by self-reverse biasing," *IEEE Journal of Solid State Circuits*, vol. 28, pp. 1136–1144, November 1993.

[8] M. C. Johnson, D. Somasekhar, L.-Y. Chiou, and K. Roy, "Leakage control with efficient use of transistor stacks in single threshold CMOS," *IEEE Transactions on VLSI Systems*, vol. 10, no. 1, 2002.

[9] S. Mukhopadhyay and K. Roy, "Accurate modeling of transistor stacks to effectively reduce total standby leakage in nano-scale CMOS circuits," in *International Symposium on VLSI Circuits*, pp. 53–56, June 2003.

[10] J. T. Kao and A. P. Chandrakasan, "Dual-threshold voltage techniques for low-power digital circuits," *IEEE Journal of Solid-State Circuits*, vol. 35, no. 7, 2000.

[11] S. V. Kosonocky, M. Immediato, P. Cottrell, T. Hook, R. Mann, and J. Brown, "Enhanced multi-threshold (MTCMOS) circuits using variable well bias," in *Proc. of Int. Sym. on Low Power Electronics and Design*, pp. 165–169, August 2001.

[12] A. Abdollahi, F. Fallah, and M. Pedram, "Runtime mechanisms for leakage current reduction in CMOS VLSI circuits," in *Proc. of Int. Sym. on Low Power Electronics and Design*, pp. 213–218, August 2002.

[13] S. Heo, K. Barr, M. Hampton, and K. Asanovic, "Dynamic fine-grain leakage reduction using leakage-biased bitlines," in *Proc. of IEEE Int. Sym. on Computer Architecture*, pp. 137–147, May 2002.

[14] T. Thorp, G. Yee, and C. Sechen, "Design and synthesis of dynamic circuits," *IEEE Transactions on VLSI Systems*, vol. 11, pp. 141–149, February 2003.

[15] D. Harris, "Skew-tolerant circuit design," in *Ph.D. Thesis*, Stanford University, Stanford, CA 1999.

[16] N. Sirisantana and K. Roy, "Selectively clocked CMOS logic style for low-power noise-immune operations in scaled technologies," in *IEEE Proc. of the Design, Automation, and Test in Europe Conference and Exhibition*, pp. 11160–11161, March 2003.

[17] A. Solomatnikov, D. Somasekhar, and K. Roy, "Skewed CMOS: Noise-tolerant high-performance low-power static circuit family," *IEEE Transactions on VLSI Systems*, vol. 10, no. 4, 2002.

[18] R. Puri, A. Bjorksten, and T. E. Rosser, "Logic optimization by output phase assignment in dynamic logic synthesis," in *Proc. IEEE/ACM Int. Conference on Computer Aided Design*, pp. 2–8, November 1996.

[19] Y. W. Li, G. Patounakis, A. Jose, K. L. Shepard, and S. M. Nowick, "Asynchronous datapath with software-controlled on-chip adaptive voltage scaling for multirate signal processing applications," in *Proc. of Int. Sym. on Asynchronous Circuits and Systems*, pp. 216–225, May 2003.

[20] A. Alvandpour, R. Krishnamurthy, K. Soumyanath, and S. Borkar, "A conditional keeper technique for sub-0.13μm wide dynamic gates," in *International Symposium on VLSI Circuits*, pp. 29–30, 2001.

[21] L. McMurchie, S. Kio, G. Yee, T. Thorp, and C. Sechen, "Output prediction logic: a high performance CMOS design technique," in *Proc. IEEE Int. Conference on Computer Design*, pp. 247–256, September 2000.

[22] K.-W. Kim, C. Liu, and S.-M. Kang, "Implication graph based domino logic synthesis," in *Proc. IEEE/ACM Int. Conference on Computer Aided Design*, pp. 111–114, November 1999.