Static Power Reduction Techniques for Asynchronous Circuits

Carlos Ortega, Jonathan Tse, and Rajit Manohar
Computer Systems Laboratory
School of Electrical and Computer Engineering, Cornell University
Ithaca, NY 14853, U.S.A.
{cto3, jon, rajit}@csl.cornell.edu

***Abstract*—Power gating techniques are effective in mitigating leakage losses, which represent a significant portion of power consumption in nanoscale circuits. We examine variants of two representative techniques, Cut-Off and Zig-Zag Cut-Off [1], and find that they offer an average of 80% and 20% in power savings, respectively, for asynchronous circuit families. We also present a new zero-delay (ZDRTO) wakeup technique for power gated asynchronous pipelines, which leverages the robustness of asynchronous circuits to delays and supply voltage variations. Our ZDRTO technique offers a tradeoff between wakeup time and static power reduction, making it suitable for power gating pipelines with low-duty cycle, bursty usage patterns.**

***Index Terms*—asynchronous logic circuits; leakage currents; very-large-scale integration; pipeline processing; power gating**

I. Introduction

Reducing power consumption has become very important in recent years due to increases in transistor density and clock frequency as well as consumer trends in high-performance, portable, and embedded applications. Dynamic power losses are significant, but can be mitigated by techniques such as clock gating, which reduces the power consumption of idle sections of synchronous circuits [2]. Asynchronous designs offer this advantage inherently, as they are data driven and are only active while performing useful work. In other words, asynchronous circuits implement the equivalent of a fine-grained clock gating network. However, while dynamic power losses have been dominant in the past, static power loss has become a major contributor to power consumption in nanoscale technologies [3,4] due to leakage currents:

- *Source-to-Drain* (I_{sd}) leakage, also known as subthreshold leakage, has increased due to recent reductions in threshold voltages [5].
- *Gate-to-Channel* (I_g) leakage manifests as bidirectional electron tunneling between the substrate and gate through the gate oxide [5,6], which has increased due to shrinking gate oxide thickness.
- *Source/Drain-to-Substrate* (I_{inv}) leakage currents are another name for the reverse-bias currents between a transistor's active regions and bulk [5].

There are a wide array of techniques designed to reduce leakage currents [7–9]. The most effective techniques involve power gating circuits, essentially cutting the pull-up network (PUN) and pull-down network (PDN) off from one or both power rails during idle or "sleep" periods. During active periods, the circuit is reconnected to the power rails in a process known as "wake up" or power up. While power gating has been adapted for use in asynchronous circuits [1,10,11], most of these efforts involve direct application of synchronous techniques to asynchronous systems. As such, the unique capabilities of asynchronous circuits have not been fully leveraged in the context of power gating.

Many asynchronous circuit families are robust to a wide range of supply voltages, ambient temperatures, and process variations. We exploit this robustness in the context of power gating to enable a zero-delay wakeup scheme for pipelined computation: the first token traveling through a pipeline turns on downstream pipeline stages, hiding the latency cost of wake up in the computation time of upstream pipeline stages.

Synchronous circuits cannot take full advantage of such aggressive power gating control schemes, as local supply voltages must reach nominal values to prevent the synchronous circuit from violating its timing requirements, e.g. setup/hold constraints on state-holding elements. Therefore, inputs can only be applied to a pipeline stage once the supply voltage has reached an acceptable threshold. By leveraging the supply voltage operating range of asynchronous circuits, we can avoid this requirement and begin useful computation before the supply voltage has stabilized, reducing the forward latency seen by the first input token.

Section II presents a general overview of the two main classes of power gating techniques: (i) *Non-state preserving*, and (ii) *State-preserving*. Asynchronous circuits contain many pseudo-static gates, and robust circuit families like quasi-delay insensitive (QDI) asynchronous logic contain a significantly higher number of pseudo-static gates than an equivalent synchronous computation. To this end, we discuss the implementation details of power gating asynchronous circuits in section III, which focuses on applying non-state preserving and state preserving techniques to pseudo-static elements. Our evaluation of these techniques is given in section VI. In section IV, we formalize the aforementioned zero-delay turn-on power gating control methodology, which we call *Zero-Delay Ripple Turn On* (ZDRTO), and discuss our method of empty pipeline detection, a key component in power gating. Finally, in section VII, we present the results of our evaluation of ZDRTO, as well as a discussion of appropriate use cases.

II. Related Work

Power gating techniques essentially increase the effective resistance of leakage paths by adding sleep transistors between transistor stacks and power supply rails. Oftentimes, these power gating or sleep transistors are shared amongst multiple logic stacks to reduce the number of leakage paths as well as area overheads. Sharing the transistors effectively creates two new power nets: Gated-Vdd (*gvddv*) and Gated-Ground (*gvssv*), which replace V_{DD} and *GND* for power-gated logic stacks. *gvddv* is connected to V_{DD} using a head sleep transistor and *gvssv* is connected to *GND* using a foot sleep transistor.

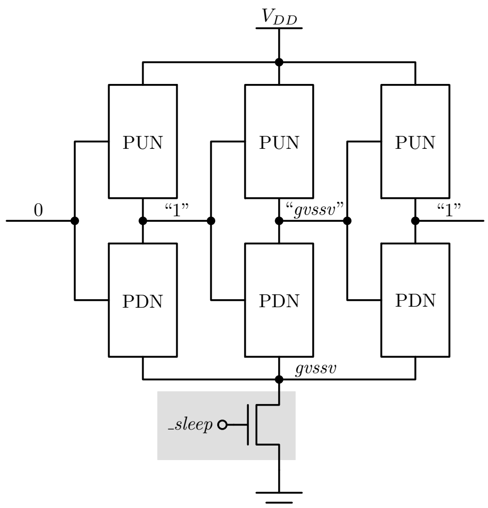

Fig. 1. Cut-Off (CO) power gating using a foot sleep transistor, which is shared by several logic blocks. The output nodes tend to drift to *gvssv*, which itself drifts towards V_{DD}.

Regardless of which rail is gated, the power gating or sleep transistor(s) should be made very large to meet the current draw of the circuit in active mode [7]. Typically, only one rail is gated due to area constraints. An nMOS foot transistor, as seen in Fig. 1, is preferred due to its greater drive strength—hence decreased area—compared to a pMOS transistor. To reduce the leakage even further, high-Vt thick gate-oxide devices are commonly used as power gating transistors.

A. Non-State Preserving Power Gating

Non-state preserving techniques destroy state by allowing internal nodes to uniformly drift towards one of the power rails. This general class of power-gating techniques has various implementation methodologies:

- *Cut-Off* (CO): Both the logic and sleep transistors are implemented using regular-Vt devices.
- *Multi-Threshold* (MTCMOS): The logic is implemented using low-Vt transistors and the sleep transistors are implemented using high-Vt devices. This configuration allows the logic to be fast during active mode and the sleep transistors to properly cutoff source-to-drain subthreshold leakage currents during idle mode [12].
- *Boosted-Gate* (BGMOS): As in MTCMOS, BGMOS uses low-Vt logic, but very high-Vt thick-oxide sleep transistors, which hurt active mode performance. To mitigate this, the gate of the sleep transistor is driven above V_{DD} during active mode to improve current drive capability [13].
- *Super Cut-Off* (SCCMOS): The gate of the sleep transistor is driven past the supply voltages—above V_{DD} or below ground—during idle periods by using a bias voltage [14]. However, wake up time is increased with respect to schemes which do not over-drive the gate.

With the exception of Cut-Off power gating, all of these techniques require the foundry to provide devices with different thresholds and oxide thicknesses. Most modern CMOS processes have transistors with multiple threshold voltages available. BGMOS and SCCMOS require a bias voltage generator, e.g. a switched capacitor circuit, which increases the strain on the gate of the sleep transistor, and may introduce some undesirable parasitic effects such as latchup. To mitigate the increased strain on the gate of the sleep transistor, it is desirable to have thick-oxide devices [14]. However, the power consumed by the bias generation circuitry could offset the power savings from power gating, especially in ultra-low power systems or systems where the number of power-gated transistors is small. We examine the power consumption of simple bias generators in section VI.

The primary disadvantage of these techniques is that the state of internal nodes is lost. For example, in Fig. 1, the inputs to the first stage while idle are logic 0, and the output of the first stage is logic 1. However, if we assume that the gate (I_g) and the source-to-drain (I_{sd}) leakage currents are greater than the reverse-bias source/drain-to-substrate (I_{inv}) leakage current, i.e. $I_g + I_{sd} > I_{inv}$, the output of the second logic stage drifts to *gvssv*. In fact, over a long time period all CO power gated output nodes will drift to *gvssv*, as discussed in section VI.

B. State Preserving Power Gating

State preserving power gating techniques reduce leakage while retaining state. The tradeoff between these techniques and non-state preserving techniques is that they are not as effective at reducing leakage currents.

One technique, *Variable Threshold* (VTCMOS), varies transistor threshold voltages by biasing the substrate. By enforcing lower threshold voltages in active mode versus idle mode, this method retains performance while active and reduces leakage while idle. However, as with SCCMOS, the VTCMOS scheme requires a bias voltage generator, as well as the use of triple well processes [15]. VTCMOS does have the advantage of not requiring additional transistors aside from those used for control and bias generation.

If the idle state of a circuit is known at design time, and the area overhead of adding sleep transistors is acceptable, we can employ the *Zig-Zag Cut-Off* (ZZCO) power gating technique [16]. As in non-state preserving techniques, ZZCO introduces two power nets: Gated-Vdd (*gvddv*) and Gated-Ground (*gvssv*). Rather than gating every logic stage in the same fashion, the selection of head or foot transistor is governed by the desired logic level of the output node.

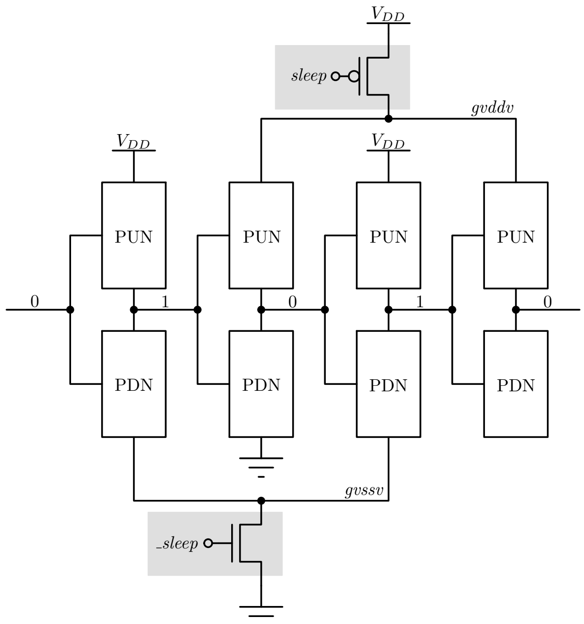

Fig. 2. Zig-Zag Cut-Off (ZZCO) using a pair of sleep transistors, which are shared between several logic blocks. The configuration of sleep transistors restores the output nodes to the appropriate idle state values.

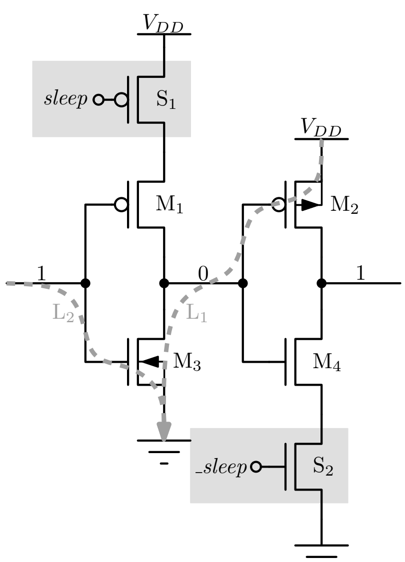

Fig. 3. Sneaky gate leakage paths in Zig-Zag Cut-off (ZZCO). The sleep transistors are shared between several logic blocks. For clarity, the substrate connections are shown for M_2 and M_3.

As shown in Fig. 2, *gvddv* and *GND* are used as power rails for logic blocks with a logic 0 output when idle and V_{DD} and *gvssv* for blocks with a logic 1 output when idle. In other words, if the desired idle output is 0, cut off the stack from V_{DD}, and vice versa for an idle output of 1. The ZZCO scheme can be combined with other techniques used in non-state holding power gating schemes as well, such as biased control signals as in ZSCCMOS [17] and BGMOS, or devices with different thresholds as in MTCMOS.

The primary disadvantage of ZZCO is the presence of sneaky-leakage paths; not all paths from the output nodes to the power rails are disabled. The primary leakage mechanism is through the gates of neighboring stacks. Consider, for example, two inverters using ZZCO power gating as shown in Fig. 3. Even assuming that sleep transistors S_1 and S_2 provide perfect cutoff from the power rails, there are two essentially equivalent paths: L_1, from V_{DD} to GND through the gate of M_2, and L_2, from the input to GND through the gate of M_3. Note that the gate-to-body voltage of the transistors ($|V_{gb}|$), specifically M_2 and M_3, is essentially $|V_{DD}|$. As the gate leakage is exponentially dependent on the electric field (voltage) across the gate, i.e. V_{gb}, ZZCO is not particularly effective at mitigating gate leakage currents.

III. Asynchronous Power Gating

A. Pseudo-Static Logic Overview

The production rules for an operator with a pullup network pun, pulldown network pdn, and output node z are shown below:

$$pun \;\mapsto\; z\uparrow \qquad\qquad pdn \;\mapsto\; z\downarrow$$

Such an operator is non-interfering and *combinational* if $pun \equiv \neg pdn$. The weaker constraint of $\neg pun \vee \neg pdn \equiv \textbf{true}$, denotes a non-interfering, *dynamic* operator. Adding a staticizer to the output node, z, of a dynamic operator ensures the output is always driven. Such an operator is known as a *pseudo-static* gate.

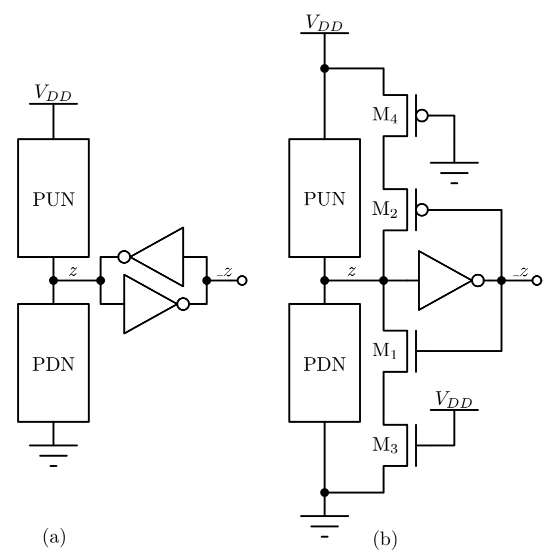

Fig. 4. (a) Pseudo-Static CMOS Gate, (b) Weak Feedback Inverter

An implementation of a generic pseudo-static operator is shown in Fig. 4a. The statizicer consists of two cross-coupled inverters attached to node z. Note that there is always opposition to any change in z due to the feedback inverter. To ensure correct operation, the transistors of the feedback inverter must be sized to be weaker than the logic stacks of the operator. Furthermore, the feedback transistors add parasitic capacitance to the output node. To mitigate this effect, each feedback transistor is split in two, as shown in Fig. 4b. The feedback stack now consists of a minimum sized transistor closer to the output, $\mathrm{M}_1(\mathrm{M}_2)$, and a long transistor closer to the power rails, $\mathrm{M}_3(\mathrm{M}_4)$. In order to reduce the load on node $\_z$, the gates of the long transistors, $\mathrm{M}_3(\mathrm{M}_4)$, are usually connected to $V_{DD}(GND)$ or to $Reset(\_Reset)$.

B. Non-State Preserving

Any of the previously discussed non-state preserving techniques can be applied to pseudo-static logic. However, waking up a circuit without resetting all its pseudo-static elements into known, safe states could result in incorrect circuit behavior, or even the potential for stable short-circuits between power rails.

This problem is not unique to power gating—in fact, it is a concern during the initial power up of asynchronous circuits, which use pseudo-static gates. Fortunately, the addition of reset transistors to initialize the appropriate circuit nodes is a viable solution. In the case of power up, the signals which drive the gates of these reset transistors are generated off-chip. However, initial power up is a global event. As the off-chip environment is unaware of the entire internal state of the chip, generating reset signals for each individual power gated circuit off-chip would prove to be practically impossible, even just considering package pins as a limitation.

To ensure correctness and safe operation, each power gated circuit requires its own self reset circuitry. In our asynchronous design methodology, we use transistors both in series and in parallel with pullup and pulldown stacks. To control the parallel and series reset transistors, we use *pReset* and *sReset* signals and their complements, respectively. While the order and delay between asserting *pReset* and *sReset* is flexible, *pReset* must be deasserted before *sReset* to prevent any short circuits between power rails. A typical reset sequence is as follows:

1) Assert $pReset$, $sReset$, and their complements and hold them until all the circuit output nodes have been charged to their appropriate safe states.
2) Deassert $pReset$ and its complement.
3) Deassert $sReset$ and its complement.

Note that in order for the self reset circuit to be QDI, it would have to instrument every output node in order to determine whether or not it has reached the appropriate safe state during step 1 above. This endeavor quickly becomes very costly in transistor count, area, complexity, and power. A similar argument applies for determining the appropriate delay between steps 2 and 3 above. As such, the self reset circuit we propose is not QDI, but instead relies on the timing assumption that a delay line, tailored to the circuit being reset, is sufficient to guarantee safe reset of all internal circuit nodes. Again, a similar argument involving a delay line between steps 2 and 3 applies.

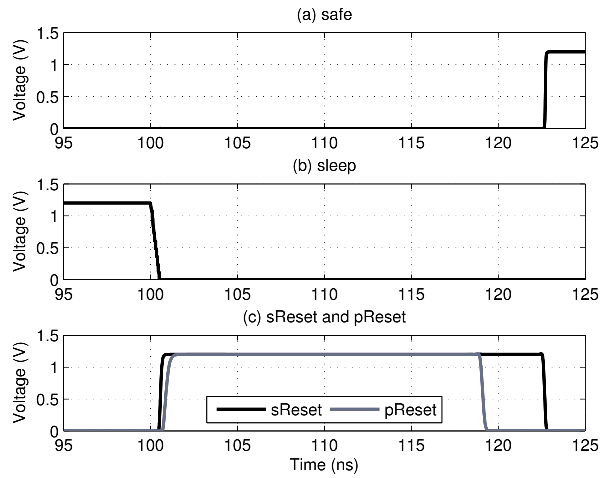

Fig. 5. Self reset circuit behavior immediately after *sleep* goes low.

Upon deasserting the *sleep* signal, i.e. waking up the circuit, the self reset circuitry will assert *sReset* and *pReset* in that order, then deassert them in reverse order as seen in Fig. 5. The timings between these transitions are controlled by delay lines. Note that *pReset* should be held long enough to account for the charge/discharge latency of the local supply rails—i.e. *gvssv*—and the worst case reset latency. Depending on process variations, it may be desirable to further increase the hold time of *pReset*. In fact, it is advisable to layout the delay line as close to the logic as possible in order to replicate localized systematic process variations. Once the self reset sequence is complete, a *safe* signal is raised, as seen in Fig. 5a.

From the time the circuit has been power gated until the circuit completes its internal self reset, the outputs of the gated circuit are undefined. If the rest of the pipeline is operating, these undefined outputs should not corrupt the rest of the system, particularly pipeline stages which have been fully woken up. This impacts both the pipeline stage inputs—through acknowledge signals—and outputs—through data signals. *Isolation circuits* are introduced to make sure that all output signals from the power gated block remain in a well-defined state. Adding isolation circuits to the input of a stage prevents signals from interfering with the self reset of a stage, and isolation circuits on the output prevent any glitches from propagating to other pipeline stages during the self reset stage.

C. State Preserving

Our state preserving power-gating scheme is based on the Zig-Zag Cut Off (ZZCO) power gating scheme studied in [1], as it offers a good tradeoff between power savings and performance degradation for this class of power gating. In idle mode, we know there are no inputs and that all logic blocks have finished computation. Therefore, each individual logic block is waiting for data. By analyzing the handshaking expansions of each process, we can ascertain the value of most signals in the idle state. One exception involves the case of two-phase handshakes where the number of handshakes is not guaranteed to be even. Nevertheless, for most cases, we can use Zig-Zag power gating by connecting all the logic blocks whose output is logic 1 to $gvssv$ and all the nodes whose output is logic 0 to $gvddv$.

In order to efficiently power gate pseudo-static operators, we gate the forward inverter of the staticizer in addition to the logic stacks depending on the idle state output of the logic. Essentially, pseudo-static Zig-Zag Cut-Off (ZZCO) power gating adds sleep transistors to the logic stack and the feedback transistors of pseudo-static operator shown in Fig. 4b.

We can reduce the leakage through the feedback inverter by connecting the gates of M_3 and M_4 to *gvddv* and *gvssv*, as shown in Fig. 6a. Alternatively, their gates could be connected to the sleep signal directly, as in in Fig. 6b, but the area penalty would be high because the sleep signal would need to be routed individual staticizers, as opposed to just the shared sleep transistors. We refer to the technique of driving the gates of M_3 and M_4 with *gvddv* and *gvssv* as Zig-Zag Cut Off with Weakened Staticizers (ZZCO-WS).

Note that the only difference between ZZCO and ZZCO-WS is between which signals drive the gates of M_3 and M_4. Thus, the area overhead for implementation of ZZCO-WS versus ZZCO is negligible, as all the supply nets—i.e. *gvssv*, *gvddv*, *GND*, and V_{DD}—are locally accessible to each layout cell.

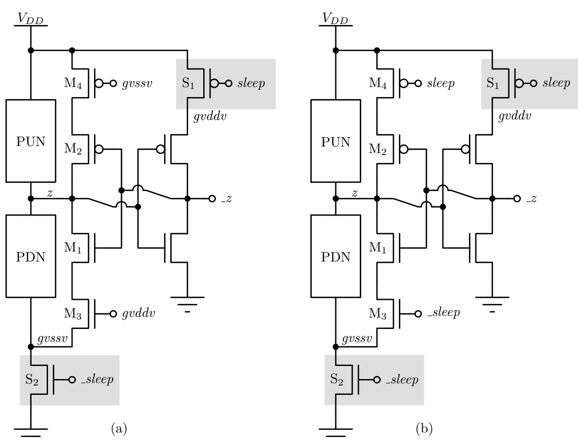

Fig. 6. Zig-Zag Power Gating with Weakened Staticizers (ZZCO-WS) using (a) Virtual Power Rails or (b) Sleep Signals

IV. Control Circuitry

In this section, we present our power gating control techniques for wake up and empty pipeline detection. These techniques are power gating scheme agnostic and can be used with any of the schemes outlined in sections II and III.

A. Zero-Delay Ripple Turn On

Our *Zero-Delay Ripple Turn On* (ZDRTO) power gating scheme allows the wake up latency of downstream pipeline stages to be hidden by the computation latencies of upstream stages, hence wakeup is "zero delay." This sequential or "ripple" turn on also minimizes the voltage fluctuations such as ground bounce that often occur during wake up of power gated circuits [18].

The CHP [19] process below describes an asynchronous N stage pipelined computation:

$$
\begin{aligned}
P \equiv\ & *[L_0?x_0; L_1!f_0(x_0)] \\
& \| \ldots \\
& \| *[L_n?x_n; L_{n+1}!f_n(x_n)]
\end{aligned}
$$

We group these pipeline stages into *clusters*, each with its own local $gvssv$ and $gvddv$ power nets and associated sleep transistors, allowing us to power gate each cluster individually, as shown in Fig. 7. The ripple turn on effect occurs upon arrival of an input token to program P. At this time, we wake up the first cluster, which wakes up the second cluster, and so on. This continues as the token travels through the pipeline with cluster i waking up cluster j, until the last cluster is active. Note that i and j do not have to be consecutive clusters—a token arriving at cluster i could potentially wake up the next few clusters.

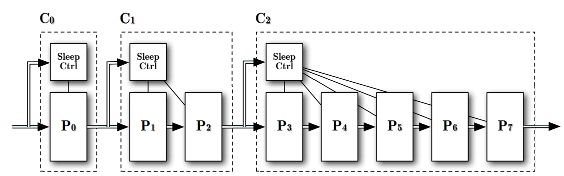

Fig. 7. Block diagram of our Zero-Delay Ripple Turn On (ZDRTO) power gating control scheme. A sample pipeline of 8-stages is divided into three unequal clusters: C_0, C_1, and C_2. Each cluster controls the power gating of the next inline cluster. With respect to Eq. 1, $j = i + 1$.

In order to achieve the "zero-delay" effect, the cluster grouping should be chosen so that the forward propagation delay, $t_{fp}(i, j)$, from cluster i to j hides the latency, $t_w(j)$, of waking up cluster j, as seen in Eq. 1.

$$t_w(j) \leq t_{fp}(i, j) \quad \forall \{i, j | i < j\} \tag{1}$$

Achieving this requirement is not difficult in modern processes, especially for low duty cycle pipelines. Note that the value of t_w is variable, as asynchronous circuits have a wide operating voltage range. Furthermore, by selecting different power gating techniques the value of t_w is coarsely tunable. A conservative choice of t_w such that $gvssv$ and $gvddv$ are equal to GND and V_{DD}, respectively, for any particular cluster by the time the first token arrives—with the exception of the first cluster—ensures each cluster is ready to perform useful computation the moment data arrives. This is the origin of the "zero-delay" latency hiding effect. A more aggressive choice of t_w such that $gvssv > GND$ and $gvddv < V_{DD}$ results in additional power savings at the cost of a longer forward propagation delay of the first tokens for that cluster—and a longer pipeline latency overall. Correctness and stability are conserved, so long as $gvssv$ and $gvddv$ have reached safe values when t_w has elapsed.

B. Empty Pipeline Detection

Up to this point, we have discussed waking up power gated circuits, but not the power down sequence. It is of particular importance to determine whether a pipeline is empty before power gating it in order to prevent data loss and incorrect execution.

There are several methods for empty pipeline detection, which can be loosely classified into one of two categories: methods that instrument each pipeline stage, or those which monitor token flow within a pipeline. The former requires the addition of extra circuitry within each pipeline stage to detect empty status or computation completion [**?**]. The instrumentation overhead grows linearly with the number of stages, making this method effective only for small pipelines.

Linear-overhead token flow techniques also exist: assuming a FIFO pipeline, inject a flagged NOP token and block further token injection. The exit of the flagged token corresponds to empty pipeline state. However, as with the instrumentation technique, each stage in the datapath must be altered to accept a flagged token.

Another token-flow option is to count incoming and outgoing tokens. While this method does not require instrumentation of individual pipeline stages, it does incur a $\lg(n)$ overhead in area, where n is the number of stages, due to the number of bits needed to count tokens. It is essential that the token counting process have a minimal effect on token flow, as any additional latency in token entrance/exit will decrease the throughput of the entire system. Furthermore, the latency of counter operations should be independent of n, especially in the case of aggressively pipelined systems where n is large.

One solution is to use a pair of rotary counters, one at the start and end of the pipeline to count incoming and outgoing tokens respectively. If the counter values match, the pipeline is empty—i.e. the same number of tokens have entered and left. However, no assumptions can be made about arrival or departure times of tokens in an asynchronous pipeline. As a result, if a token arrives or departs during a counter value comparison, the result of the comparison will be unstable.

We propose a monolithic counter which is capable of servicing increments (token entrance), decrements (token exit), and zero-value (empty pipeline) checks in constant time, similar to the bounded response time counters proposed by [20,21]. Zero checks are performed after servicing an increment or decrement, resulting in a stable output. The simultaneous arrival of increment and decrement events effectively cancel one another, so the counter can afford to do nothing, saving power. The pathological case occurs when the arrival of one or another event overlaps with the servicing of a prior event, stalling the new event and token entrance/exit. However, a pipeline operating at full throughput issues consecutive token entrance/exit events. Thus, if an event has been stalled, the next time the counter is available it will see "simultaneous" events—i.e. it will see simultaneous increments and decrements in steady state. If throughput remains an issue and additional overhead is acceptable, interleaving a pair of counters may be appropriate. Adding an alternating split processes on the increment and decrement channels allows one counter to observe odd tokens and the other even tokens.

We implemented this interleaved counter system for empty pipeline detection in single-input, single-output pipelines. Each counter is constructed of an array of single-bit counters, each of which maintains its own value as well as an additional *sticky-zero* bit. The sticky-zero bit is true if all of the more significant counter bits are 0, and false if any of the more significant bits are 1. If a carry operation occurs during the update of a particular single-bit counter, it will send an increment or decrement command to the next higher-order counter and receive an update to its local sticky-zero bit from the higher-order counter. Thus, the zero-state of the entire counter array can be determined in constant time by examining only the value and sticky zero bit of the least significant single-bit counter. The evaluation of our design is presented in section VII.

V. Simulation Methodology

All simulations presented in this paper use the BSIM4 device model, which explicitly accounts for gate, substrate and reverse biased junction leakage [6]. We evaluate our techniques using 65 and 90nm commercial technologies running at $25°\mathrm{C}$. Both technologies feature regular-Vt and high-Vt transistors. T_{ox} in the 90nm technology is 2.1nm and 2.0nm in the 65nm technology. Based on the spice models, we included additional wire load in the SPICE netlist for every gate in the circuit. Based on prior experience on post-layout simulations, our load wires estimates are conservative and circuit performance is typically higher in post-layout simulations. Capacitances at the virtual power rails were calculated as a function of the drain capacitances and the number of devices attached to them. All simulations are at the typical-typical (TT) corner.

We applied our power gating techniques to a FIPS-compliant, 128-bit Advanced Encryption Standard (AES) encryption/decryption engine [22]. We chose to use the AES engine because of its complexity, wide datapath, and low duty cycle—encryption engines are usually inactive for long periods of time. We examine the AES round operation, which consists of four operations, as seen in Table I. Note that the *BS* operation is implemented with the *sbox* design presented in [23].

TABLE I
AES Round Operations

	Transistor Count
Add Round Key (*AK*)	8400
Byte Substitute (*BS*)	84144
Shift Rows (*SR*)	7567
Mix Column (*MC*)	30000
Control Circuitry	18000
Total	148111

Our architectural decisions and transistor sizings were chosen to minimize *energy* and *static power*. In particular, we based our sleep transistors sizing on the work presented in

[24]. A detailed discussion of optimal transistor sizing is beyond the scope of this paper.

VI. Power Gating Evaluation

We shall first examine the power savings of applying non-state-preserving and state-preserving power gating techniques to each individual AES operation block in isolation. We chose Cut-Off (CO) and Zig-Zag Cut-Off (ZZCO) as our non-state holding and state holding power gating techniques, respectively, as neither requires bias voltages or multiple-well capabilities. The complexity and tradeoffs of bias voltage generation made it unattractive to implement. For example, even though SCCMOS offers better leakage reduction versus CO, the current draw of the bias generation circuits make SCCMOS viable for only large circuits. In our 90nm technology, a switched capacitor bias generator, based on the baseline generator from [25], consumes an average of 116μW. As such, power gating schemes which require on-chip bias generation with conventional circuits are inappropriate for any ultra-low power applications with static power in the sub-microwatt regime.

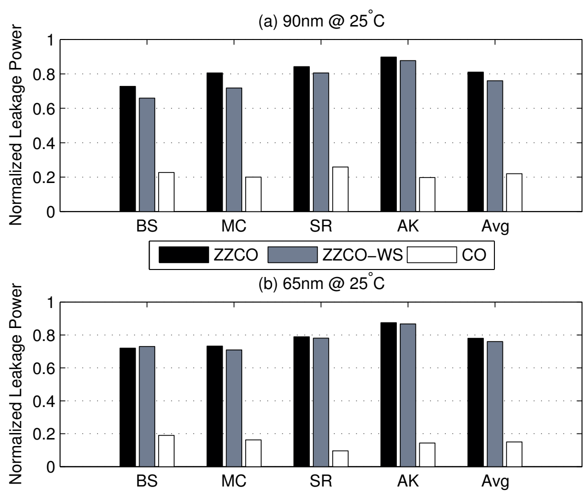

Fig. 8. Static power consumption of each AES round operation. Each operation is power gated in isolation, and results are normalized to a baseline implementation of no power gating.

As seen in Fig. 8, ZZCO reduces leakage power by an average of 20%. If we weaken the staticizers (ZZCO-WS) during idle time as discussed in section III-C, we save an additional 5%. However, the maximum savings in power come from using CO power gating, as it offers a 82% reduction in leakage power on average. The power reductions from ZZCO and ZZCO-WS are similar in both 65nm and 90nm technologies; however, CO power gating saves an additional 8% of static power in 65nm versus 90nm.

As for performance, ZZCO has the most pronounced effect on average operating frequency with a 29% degradation in 90nm and a 28% degradation in 65nm. ZZCO-WS is slightly better with degradations of 24% and 21% in 90nm and 65nm, respectively, and CO has the least impact of the three schemes, averaging a 23% degradation in 90nm and a 20% degradation in 65nm. Using *gvssv* and *gvddv* to drive the gates of the series transistors instead of *GND* and V_{DD} weakens the feedback stack, reducing leakage as well as the opposition to changing the output node *z*, which origin of the performance improvements.

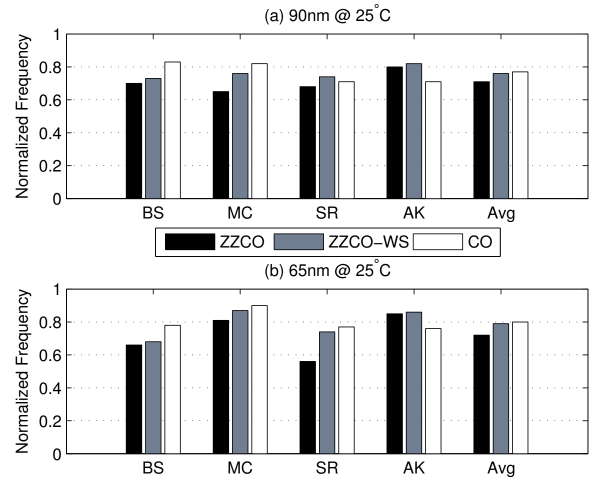

Fig. 9. Average operating frequency of each AES round operation. Each operation is power gated in isolation, and results are normalized to a baseline implementation of no power gating.

Our examination of the Cut-Off (CO) scheme revealed interesting transient behaviors, as seen in Fig. 10 for a *sbox* circuit from our AES engine in idle state, which we commanded to sleep at $t = 100$ns. Fig. 10b shows the trace of *gvssv*, virtual ground, and Fig. 10a plots supply current. Note that before *sleep* is asserted, the power consumption essentially matches that of an ungated *sbox* circuit. After *sleep* is asserted, the power consumption increases dramatically as *gvssv* floats towards V_{DD}.

We attribute this dramatic rise in power consumption to saturation-mode current in the nMOS stacks. Before *gvssv* settles, the nMOS transistors go from cut off to saturation. This transient behavior can last for longer than 200μs, which leads us to conclude that CO is not appropriate for circuits that spend relatively little time in sleep mode—less than 200μs, for example.

As discussed earlier, the CO power gating scheme destroys the state of all logic gates, and not just those which have idle outputs of 0. This is illustrated by the trace of an internal signal, *in.e*, in Fig. 10. Before *sleep* is asserted, all inputs to the driver of *in.e* are low, activating the PUN and driving *in.e* to V_{DD}. Once *sleep* is asserted, all nodes tied to *gvssv* drift towards V_{DD}. As soon as $gvssv > V_{DD} - V_{th}$, the PUN goes into cut-off and as a result, *in.e* is no longer driven high. Therefore, *in.e* discharges to the value of *gvssv*. Because of this effect, all nodes need to be restored to their nominal values before restarting operation.

From our results it is clear that ZZCO-WS is better than ZZCO both in static power savings and performance retention.

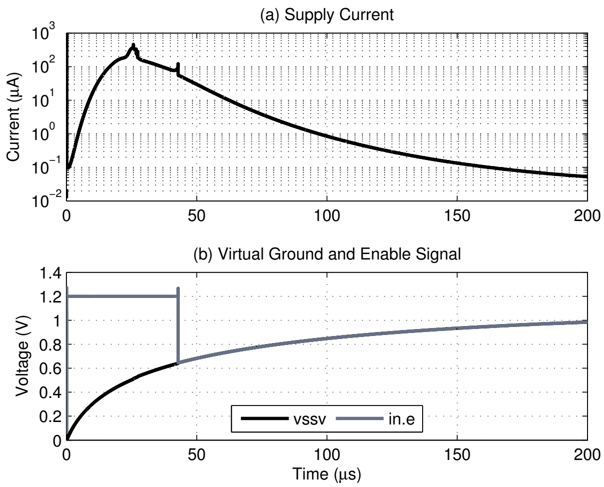

Fig. 10. Transient behavior of CO power gating. Note the peak in supply current immediately after *sleep* is asserted at $t = 100$ns.

Since the overheads of ZZCO and ZZCO-WS are the same, we believe that ZZCO-WS should be the preferred choice between the two schemes. The choice between ZZCO-WS and CO is not as clear, however. Performance degradation between the two is similar, as seen in Fig. 9, but CO offers dramatic improvements in static power reduction over ZZCO-WS. As discussed earlier, the transient behavior of the Cut-Off power gating scheme makes it unattractive for applications where the duration of a circuit's idle period is less than several hundred microseconds. In comparison, the transient behavior of ZZCO-WS is well-behaved, so it can be used to power gate circuits for periods in the several hundred nanosecond range. As a result, ZZCO-WS is suitable for circuits with short sleep periods, whereas the CO scheme is more appropriate for long-term sleep applications.

VII. ZDRTO Evaluation

As discussed in section IV-A, to implement our *Zero-Delay Ripple Turn On* (ZDRTO) power gating control scheme, we must organize our pipeline stages into clusters. Our clusters are simply the different operations of the AES round computation described earlier, each of which is a pipelined computation. *BS* and *SR* are transformations on individual bytes, by slicing the datapath in 8-bit chunks, we could swap their ordering with no effect on correctness. We swap them now because the *BS* operation has a higher transistor count, as seen in Table I, and thus takes a longer time to wake up. Furthermore, reordering the *BS* and *SR* stages also allows for hardware reuse between encryption and decryption. The final pipeline stage clustering is as follows: *AK*, *SR*, *BS*, *MC*.

TABLE II
Interleaved Counter Overhead

	Transistor Count	Static Power (nW)
Additional Bit	400	19
Constant Overhead	1900	95

To fully implement power gating in a pipeline, we need empty pipeline detection in the form of our interleaved empty pipeline detection counter described in section IV-B. The total depth of our AES round pipeline is 10 half-stages, so we use a 4-bit interleaved counter. The overheads added by the counter are summarized in Table II for our 90nm process, broken up by the overhead of adding additional bits and the constant overhead of the counter arbitration and control circuitry. The average operating frequency is relatively low—350MHz in 90nm. Given these characteristics, our interleaved counter is suitable for deep low energy pipelines.

TABLE III
Pipeline Configurations

	AES Round Cluster			
No ZDRTO	*AK*	*SR*	*BS*	*MC*
Baseline	N/A	N/A	N/A	N/A
CO	CO	CO	CO	CO
ZZ	ZZ	ZZ	ZZ	ZZ

	AES Round Cluster			
ZDRTO	*AK*	*SR*	*BS*	*MC*
ZZ-ZDRTO	ZZ	ZZ	ZZ	ZZ
Mixed-A	N/A	ZZ	ZZ	CO
Mixed-B	N/A	ZZ	CO	CO

Legend

N/A	No Power Gating
CO	Cut-Off Power Gating
ZZ	Zig-Zag Cut-Off with Weakened Staticizers

In order to evaluate our ZDRTO scheme, we compare several different classes of pipeline: a baseline pipeline without any power gating, power gated pipelines which are controlled as a monolithic unit, i.e. the entire pipeline is woken up simultaneously as in synchronous circuits, and power gated pipelines which are controlled by our ZDRTO scheme. All of our different combinations of control schemes and power gating techniques are detailed in Table III.

The first pipeline configuration, *Baseline*, is a completely unaltered AES round pipeline without any power gating, power gating control, or empty pipeline detection circuitry to which we compare all of our other configurations. We add our empty pipeline detection counter to all other pipeline configurations, as all of the other configurations are power gated. The *ZZ* and *CO* configuration consist of the same AES round pipeline, but with the addition of ZZ-WS and CO power gating respectively. No ZDRTO control is used for these configurations. Instead, the entire pipeline is woken up as a monolithic unit upon the arrival of the first input token, as would be the case in a synchronous pipeline. Note that the *CO* pipeline configuration has isolation circuitry at the start and end of the entire pipeline. The *ZZ-ZDTRO* configuration uses *ZZCO-WS*, with the addition of the ZDRTO scheme. Each ZZCO-WS power gated cluster wakes up the next one in sequence as the first token flows through the pipeline. We chose not to do a detailed investigation of a *CO-ZDRTO* configuration, where all the pipeline clusters are gated using the CO scheme and wake up is controlled by our ZDRTO control scheme. Early simulations indicated that the wake up latency of such a configuration was comparable to that of the non-ZDRTO-enabled *CO* configuration, thereby making the additional overhead of adding per-cluster self reset and isolation circuits unattractive.

We also investigated two additional ZDRTO-enabled pipeline configurations, *Mixed-A* and *Mixed-B*. These two pipeline configurations have been optimized in order to minimize the wake up latency. The first cluster, *AK*, is not power gated at all so that computation can be started immediately upon data arrival. In parallel with beginning computation in the *AK* cluster, we turn on the next cluster, *SR*, which is power gated using our ZZCO-WS scheme. *MC* is CO power gated, so waking it up requires the addition of isolation and self reset circuits between clusters. This is also true of the *BS* cluster in the *Mixed-B* configuration. Note that the only difference between the *Mixed-A* and *Mixed-B* schemes is in which power gating scheme is applied to the *BS* cluster, as seen in Table III. The purpose of this difference is to illustrate the tradeoffs between power gating with ZZCO-WS and CO deep into the pipeline. As ZZCO-WS power gated clusters have faster wake up times than CO power gated clusters, it is desirable to use ZZCO-WS power gating near the beginning of the pipeline to improve wake up time and CO power gating near the end to take advantage the superior power savings of CO.

TABLE IV
ZDRTO RESULTS (90NM)

No ZDRTO	Wake Up (ns)	Leakage (μW)	Freq. (MHz)
Baseline	0.00	7.10	285
CO	32.89	1.50	262
ZZ	5.9	6.34	180

ZDRTO	Wake Up (ns)	Leakage (μW)	Freq. (MHz)
ZZ-ZDRTO	5.6	6.46	182
Mixed-A	18.4	6.05	226
Mixed-B	26.2	1.62	260

However, to retain a competitive advantage in wake up latency, the wake up sequence of CO power gated clusters must be started in parallel with upstream pipeline stages. For example, in *Mixed-A*, *SR* wakes up both *BS* and *MC* in order to hide the longer latency of waking up *MC*, as it is CO power gated. A similar control scheme applies to *Mixed-B*, where *AK* wakes up *BS* and *SR* wakes up *MC*. The results of our study, done in a 90nm commercially-available process, are presented in Table IV. Wake up time is calculated by comparing the full pipeline propagation latency of the first arriving token in each pipeline configuration to the propagation latency of the first token arriving in the baseline configuration.

As expected, the *CO* pipeline configuration offers the best in terms of leakage power, but it has the longest wake up time compared to the other configurations. With the obvious exception of the baseline configuration, the ZDRTO-enabled pipeline configurations offer the best wake up times, and competitive leakage power reductions. ZZ is not as effective at reducing leakage power but it has shorter wake up times than CO.

The *Mixed-B* configuration hides most of the wake up latency of the CO power gated clusters while reducing leakage by almost the same amount as the *CO* configuration. On the other hand, the *Mixed-A* configuration does not offer the same benefits. Using CO power gating only on the *MC* cluster is a poor design choice since *MC* only accounts for roughly 20% of the transistors while having large overhead in isolation circuitry.

These results indicate the our ZDRTO scheme is appropriate for use in low-duty cycle, bursty applications where wake up time is critical. For pipelines where wake up time is not critical and performance is critical, a choice such as the *CO* configuration will save in static power and provide high performance.

VIII. Future Work

We would like to implement the various techniques presented in this paper in the Sensor Network Asynchronous processor developed at Cornell University [26], especially in Silicon-on-Insulator (SOI) processes. One necessary step in our future work is to develop a CAD tool to aid designers in order to choose the power gating technique that best fits their application, and to cluster regions appropriately to minimize leakage power. These tools could also automatically validate sleep device drive strength based on an analysis of power supply nets.

IX. Conclusion

We present an evaluation of different power gating schemes in the context of asynchronous circuits. Zig-Zag Cut-Off (ZZCO) power gating for pseudo-static logic gates offers fast wake up time, but only reduces static power by 30% on average. Cut-Off (CO) power-gating offers an average of 80% power savings, at the cost of increased complexity and the need for careful timing analysis. We offer an example analysis and evaluation of power gating applied to an asynchronous AES encryption/decryption pipeline as well as a generic empty pipeline detection technique to be incorporated into power gating control circuitry with minimum area, power, and performance overheads. Finally, we introduce a novel *Zero-Delay Ripple Turn On* (ZDRTO) technique to reduce the penalty of waking up a pipeline from sleep mode. Furthermore, we present ZDRTO using hybrid power gating schemes in order to exhibit the trade offs between maximum static power savings and minimum wake up time.

Acknowledgements

The authors would like to thank Derek Lockhart of the Cornell Computer Systems Laboratory for his contributions in the design, implementation, and validation of the empty pipeline detection counter.

The research described in this paper was supported in part by NSF grants CNS-0834582, CCF-0428427, CCF-0424422, and in part by Blue Highway. Equipment support was provided by NSF infrastructure grant CNS-0708788, and the processors were donated by Intel.

REFERENCES

[1] M. Imai, *et al.* "Fine-grain leakage power reduction method for m-out-of-n encoded circuits using multi-threshold-voltage transistors." *IEEE ASYNC*, pp. 209 – 216, 2009.

[2] G. Tellez, *et al.* "Activity-driven clock design for low power circuits." *ICCAD*, pp. 62 – 65, 1995.

[3] A. Keshavarzi, *et al.* "Intrinsic leakage in low power deep submicron cmos ics." *IEEE ITC*, pp. 146 – 155, 1997.

[4] K. Roy, *et al.* "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits." *Proc. IEEE*, 91(2):305 – 327, 2003.

[5] K.-S. Yeo and K. Roy. *Low Voltage, Low Power VLSI Subsystems*. McGraw-Hill, Inc., 2005.

[6] K. Cao, *et al.* "Bsim4 gate leakage model including source-drain partition." "IEDM Technical Digest," pp. 815–818. 2000.

[7] K. Shi and D. Howard. "Sleep transistor design and implementation - simple concepts yet challenges to be optimum." *IEEE VLSI-DAT*, pp. 1 – 4, 2006.

[8] F. Hamzaoglu and M. Stan. "Circuit-level techniques to control gate leakage for sub-100nm cmos." *ACM ISLPED*, 2002.

[9] Y. Shin and H.-O. Kim;. "Cell-based semicustom design of zigzag power gating circuits." *IEEE ISQED*, pp. 527 – 532, 2007.

[10] Y. Thonnart, *et al.* "Automatic power regulation based on an asynchronous activity detection and its application to anoc node leakage reduction." *IEEE ASYNC*, pp. 48–57, 2008.

[11] T. Lin, *et al.* "Fine-grained power gating for leakage and short-circuit power reduction by using asynchronous-logic." *IEEE ISCAS*, pp. 3162 – 3165, 2009.

[12] S. Mutoh, *et al.* "1-v power supply high-speed digital circuit technology with multithreshold-voltage cmos." *IEEE SSC*, 30(8):847 – 854, 1995.

[13] T. Inukai, *et al.* "Boosted gate mos (bgmos): device/circuit cooperation scheme to achieve leakage-free giga-scale integration." *IEEE CICC*, pp. 409 – 412, 2000.

[14] H. Kawaguchi, *et al.* "A super cut-off cmos (sccmos) scheme for 0.5-v supply voltage with picoampere stand-by current." *IEEE SSC*, 35(10):1498 – 1501, 2000.

[15] T. Kuroda, *et al.* "A high-speed low-power 0.3 m cmos gate array with variable threshold voltage (vt) scheme." *IEEE CICC*, pp. 53 – 56, 1996.

[16] M. Horiguchi, *et al.* "Switched-source-impedance cmos circuit for low standby subthreshold current giga-scale lsi's." *IEEE VLSIC*, pp. 47 – 48, 1993.

[17] K.-S. Min and T. Sakurai. "Zigzag super cut–off cmos (zsccmos) scheme with self–saturated virtual power lines for subthreshold–leakage–suppressed sub–1–v–vddlsi's." *IEEE ISSCC*, pp. 679 – 682, 2002.

[18] S. Kim, *et al.* "Understanding and minimizing ground bounce during mode transition of power gating structures." *IEEE ISLPED*, pp. 22 – 25, 2003.

[19] A. J. Martin. "Compiling communicating processes into delay-insensitive VLSI circuits." *Distributed Computing*, 1(4):226–234, 1986.

[20] J. Kessels. *Calculational Derivation of a Counter with Bounded Response Time*, volume 683/1993. 1993.

[21] J. Ebergen and A. Peeters. "Design and analysis of delay-insensitive modulo-n counters." *Formal Methods in System Design*, 3(3):211–232, 1993.

[22] FIPS. *Advanced Encryption Standard (AES)*, 2001.

[23] J. Wolkerstorfer, *et al.* "An asic implementation of the aes sboxes." "CT-RSA: Cryptographer's Track at the RSA Conference on Topics in Cryptology," pp. 67–78. Springer-Verlag, 2002.

[24] J. Kao, *et al.* "Transistor sizing issues and tool for multi-threshold cmos technology." *DAC*, pp. 409 – 414, 1997.

[25] H.-J. Song. "A self-off-time detector for reducing standby current of dram." *IEEE SSC*, 32(10):1535–1542, 1997.

[26] C. Kelly, *et al.* "Snap: a sensor-network asynchronous processor." *IEEE ASYNC*, pp. 24– 33, 2003.