Efficient Adaptive Voltage Scaling System Through On-Chip Critical Path Emulation

Mohamed Elgebaly and Manoj Sachdev
Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
{mgebaly, msachdev}@vlsi.uwaterloo.ca

ABSTRACT

Conventional voltage scaling techniques rely on the characterization and monitoring of a unique critical path. However, the uniqueness of the critical path is a difficult requirement to establish in modern VLSI technologies due to the growing impact of process variations and interconnect parasitics on delay. This paper presents an on-chip critical path emulator architecture which tracks the changing critical path. The ability to emulate the actual critical path recovers most of the large margin added by conventional systems to guarantee a robust operation at all conditions. Due to the reduced margin, the proposed architecture is up to 43% and 23% more energy efficient compared to conventional open-loop and closed-loop voltage scaling systems respectively.

Categories and Subject Descriptors

B.7 [**Hardware**]: Integrated Circuits

General Terms

Design, Performance

Keywords

Low-power, adaptive voltage scaling, CMOS

1. INTRODUCTION

Voltage scaling systems are very effective in saving power and energy due to the quadratic dependence on voltage. Whenever maximum performance is not required, supply voltage can be scaled so that the critical path can still meet timing while saving power. The actual performance of the core running under scaled voltage has to be characterized to guarantee a fail-safe operation while maintaining the required performance.

Characterizing system performance depends on the underlying voltage scaling methodology. The conventional approach to perform voltage scaling uses a target operating voltage for each required operating frequency. To guarantee a robust operation, the frequency-voltage relationship is determined via chip pre-characterization at worst case conditions. This technique is utilized in open-loop dynamic voltage scaling (DVS) systems where the frequency-voltage relationship is stored in a look-up table (LUT). Since such LUT is pre-loaded with voltage-frequency points, DVS systems are not able to adapt to process variations or environmental conditions.

Alternatively, the critical path of the system can be duplicated to form a ring oscillator which adaptively responds to environmental and process variations. Also, the critical path replica can be replaced by fan-out of 4 (FO4) ring oscillator [1] or a delay line [2]. In both cases, a closed-loop mechanism based on adaptive voltage scaling (AVS) is formed by monitoring the actual silicon speed. Therefore, worst case pre-characterization is no longer required. Since there is a direct relationship between the actual performance of the core and the speed of the ring oscillator (or the delay of the delay line), AVS systems adaptively adjust supply voltage to nearly the minimum level required to meet performance targets. A safety margin is added to account for any mismatch between the ring oscillator (or the delay line) and the actual critical path.

The methodologies described above work efficiently as long as the critical path is unique. However, this requirement is difficult to establish in modern VLSI technologies. In fact, the critical path can change with the changing supply voltage. Furthermore, at a fixed supply voltage, the critical path can change based on process and temperature conditions. In order to eliminate such safety margin, Ernst *et.al* [3] proposed the Razor approach based on a speculative timing pipeline. At each pipeline stage, an extra latch is introduced which works at a slow clock speed to detect whether the pipeline is failing as a result of voltage scaling. When an error is detected, the whole pipeline is flushed and voltage is increased by a certain step. The additional latches are introduced where sub-critical paths become critical at worst case voltage operation. If the number of sub-critical paths is limited, the overhead of the razor approach can be ignored. However, in order to guarantee a robust operation, system characterization at all conditions is required. This may require an increased number of razor flip-flops. Therefore, the overhead of the error detection circuitry may increase and the error probability may also increase resulting in a reduced efficiency.

Identifying the critical and sub-critical paths in a digital system is growing in complexity as a direct consequence of

ISLPED'04, August 9–11, 2004, Newport Beach, California, USA.

the current trend towards aggressive device scaling. Process variability and interconnect parasitics will have an increasing impact on performance in future technologies. The ITRS technology roadmap predicts that delay due interconnect wires in the 65 nm technology node will be 8× that of the 180 nm technology. Meanwhile, logic delay at 65 nm feature size is predicted to approach 20% compared to current technologies [4]. Moreover, the wide-spread use of dual- and multi-threshold technologies to suppress leakage power adds further complications in the determination of a unique critical path for a system.

This paper describes a technique to mitigate the negative impact of wire delay on performance characterization for voltage scaling systems. The proposed technique uses an emulated critical path that has nearly the same voltage scaling behavior of the actual critical path at all conditions. Before further discussion of this technique, a few terms used throughout this paper are defined below.

- *Reference path*: a path that has the largest delay at worst case and nominal supply voltage.
- *Potential critical path*: a path which becomes critical at a certain voltage or at a certain process/interconnect corner.
- *Logic speed*: the actual on-chip logic speed. Logic speed is used to indicate how fast the actual process is compared to worst case.
- *Interconnect speed*: the actual on-chip interconnects speed. Interconnect speed is used to indicate the condition of the actual interconnect parasitics compared to worst case.
- *Interconnect delay ratio*: ratio of the delay caused by interconnect wires in a certain path to the total delay of that path.

2. ADAPTIVE VOLTAGE SCALING FOR DEEP SUB-MICRON TECHNOLOGIES

The impact of interconnect parasitics on identifying a unique critical path is shown in Fig. 1. For a scaled supply voltage, delays of different paths implemented in the CMOS 0.13μm technology with different interconnect delay ratios are shown. The top set of delay plots represents delays for the slow logic process corner whereas the bottom set shows delays of the same paths at the fast process corner. For the slow process, the critical path, shown as a solid curve, is the reference path with an interconnect delay ratio of 50% at nominal supply voltage (V_{dd} = 1.3 V in the CMOS 0.13μm technology). The dashed curves represent a number of potential critical paths with delays close to the reference path delay. Since, logic delay scales faster with voltage than interconnect delay, delay scaling is different from one path to the other according to the contribution of logic and interconnects to the total delay of each path. When supply voltage is scaled based on performance needs, some potential critical paths become critical and their delays exceed that of the reference path. Once this happens, conventional systems which rely on characterizing or the monitoring the reference path alone tend to fail since supply voltage is not able to deliver the required performance. In order to accommodate for the changing critical path, a delay margin has to be added to the reference path delay to guarantee that it remains the most critical at all supply voltages and for all interconnect parasitic variations.

Another factor that adds more complexity when designing a voltage scaling system is process variations and the impact of environmental conditions on performance. For example, at a certain voltage, a critical path at one process corner would not necessarily remain critical for another process corner or at a different temperature. Fig. 1 shows this trend. The reference path at slow corner is no longer critical at the fast process (solid curve is moved down). As a result, conventional systems require enough safety margin to reliably scale supply voltage at any condition without causing a system failure.

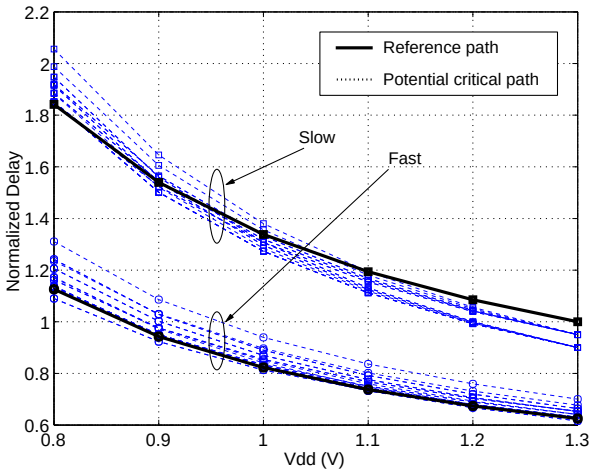

Figure 1: Reference path for the slow process changes due to the impact of interconnect delay and process variations.

Considering the case where the reference path at slow process corner is due to 50% interconnect and a sub-critical path is due to majority logic, the delay margin is increased as voltage is scaled down as shown in Fig. 1. Therefore, it is not sufficient to characterize and design the system based on worst case. One solution could be to use the logic path as the reference and add a small margin at the full scale voltage supply. This might not be sufficient if the logic process happens to be fast while interconnects remain at worst case as shown by the bottom set of delay plots in Fig. 1.

Both the conventional DVS and AVS systems tend to be power inefficient as interconnect delay contribution increases with technology scaling. Using either system requires a large delay and voltage margin. such margin reduces the power saved via supply scaling. Alternative to the conventional approach, a closer examination of the actual system behavior under different supply voltages and different operating conditions is necessary.

3. CRITICAL PATH EMULATOR ARCHITECTURE

The objective of the proposed architecture is to emulate the critical path of a system at all conditions and at all supply voltages. Emulating the real critical path can be performed if the actual logic and interconnect speeds are measured on-chip. Consequently, the effect of process and

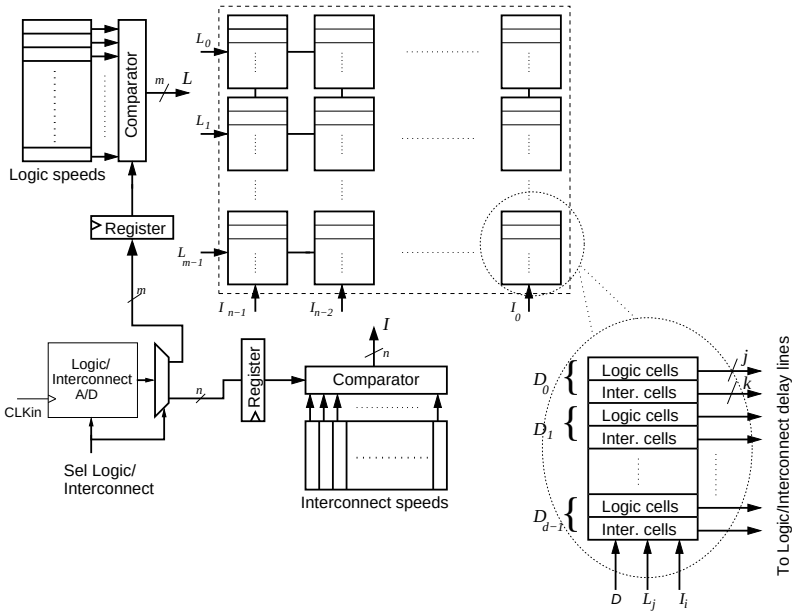

Figure 2: Critical Path Emulator Architecture.

interconnect variations on changing the critical path can be extracted. Based on the measured speeds, a critical path emulator is built using two delay lines. One delay line is composed of multiple stages of logic cells. This logic delay line is configured to have approximately the same delay as the logic delay portion of the actual critical path. Similarly, the other delay line is constructed using buffered interconnect wire segments with an overall delay approximately equal to the delay of interconnects in the real critical path. The critical path emulator is monitored to form a closed-loop feedback system. By measuring the speed of the critical path emulator, which represents the actual speed of the system, supply voltage is also adapted to the actual environmental conditions.

3.1 Proposed Architecture

The proposed architecture is shown in Fig. 2. A logic and interconnect variations estimator is used to measure the effect of on-chip process and interconnect variations on logic and interconnect speeds relative to the worst case. This is represented by the logic/interconnect A/D described below. Logic and interconnect speed are represented by m and n-bits respectively. Based on the values of both vectors, a single LUT from the LUT matrix is selected. The data stored in the selected LUT is used to construct two delay lines for each target delay, one for logic and one for interconnects. The target delay, D, is determined by the system's software and is set by the d-bit vector. For each of the d-bit values, the number of logic delay cells represented by the vector j is used to construct the logic delay line, whereas the number of interconnect delay cells, k, is used to construct the interconnect delay line. The overall delay of the two delay lines (critical path emulator delay) is approximately equal to that of the actual critical path. Furthermore, voltage scaling characteristics of the actual critical path and its emulator are nearly the same since their logic and interconnect delay compositions are approximately equivalent.

At system startup, on-chip process and interconnect variations are estimated by measuring logic and interconnect delays relative to the worst case. A low-power high-resolution A/D converter is used to determine logic speed [5, 6] as shown in Fig. 3. FO4 inverters are used since their voltage scaling characteristics are nearly similar to most CMOS logic gates [7]. To eliminate the effect of temperature on the estimation process, supply voltage is adjusted such that performance is temperature independent [8]. At this voltage, temperature effect on delay is minimized leaving process and interconnect variations as the major factor affecting performance. As shown in Fig. 3, the output of the counter represents the high-order bits of the logic speed vector whereas lower bits are represented by the output from the decoder. Similarly, interconnect speed is also measured using buffered interconnect segments. In order to avoid device mismatching between logic and interconnect buffers, the arrangement shown in Fig. 3 is used. The two extra selectors are logic cells and should scale with voltage nearly the same way as the FO4 inverter [7].

The estimation process is performed in two steps. First, the selector is set to measure logic speed. Then, the interconnect A/D converter is constructed by connecting the inverters through the long interconnect wire segments. To exclude inverter delays in the interconnect delay line, logic speed measured earlier is used to separate interconnect delay from buffer delay and interconnect parasitic variation is determined.

The output of the logic speed A/D is compared to the pre-stored logic speeds as shown in Fig. 2. Based on this comparison, the appropriate selection line in the logic speed vector ($L = L_0 L_1 \ldots L_{n-1}$) is activated to enable a row in the LUT matrix. Similarly, measured interconnect speed is used to activate the appropriate bit in the interconnect speed vector ($I = I_0 I_1 \ldots I_{m-1}$) and the corresponding column is enabled. The architecture shown in Fig. 2 shows an m logic $\times$ n interconnect speed intervals and the cor-

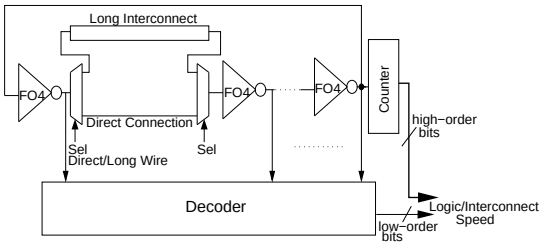

Figure 3: Logic and Interconnect low-power high-resolution A/D.

responding LUTs. Using the estimated process and interconnect variations, the proper LUT is selected. The details of the LUT are shown in Fig. 2. For each target delay, D, the corresponding number of logic cells, j, used to construct the logic delay line is selected. Similarly, the k-bit vector representing the number of interconnect delay cells is determined.

The delay line of the critical path emulator is constructed using the configuration shown in Fig. 4. The basic logic delay cell used in the logic delay line is the FO4 inverter. The interconnect delay cell is a long interconnect (e.g. minimum width and 1 mm long) with repeaters (FO4 inverters) at the driver and receiver ends of the wire. The logic delay line is programmed using the j-bit vector while the interconnect delay line uses the k-bit vector. The appropriate number of delay cells is selected using a multiplexor as shown in Fig. 4. The critical path emulator is configured by connecting the output of the logic delay line to the input of the interconnect delay line.

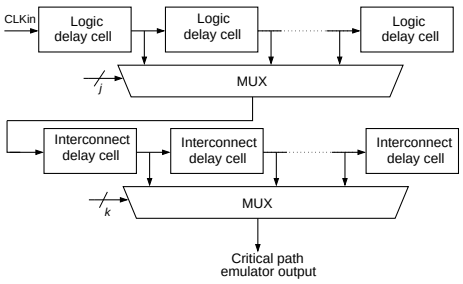

Figure 4: Implementation of logic and interconnect delay lines.

The number of logic and interconnect delay cells stored in the LUT shown in Fig. 2 can be determined through technology characterization. This process has to performed $m \times n$ times for the different process and interconnect splits. Instead of this lengthy and costly process, accurate modeling of both logic and interconnect delays is utilized. Using these models, the critical path delay at different conditions and different target speeds can be predicted and stored in the LUTs.

3.2 Delay Modeling of Logic and Interconnects

As previously mentioned, a simple, yet accurate, model for delay of logic and interconnect delay lines can replace characterization in the development of the critical path emulator. In this work, the delay model for both logic and interconnects is based on previously published models [9]. Additionally, accurate modeling of the rising/falling input signals is used since the input ramp to one stage of the delay line is the output from the previous stage.

Traditionally, rise/fall time is often categorized into a fast and a slow input ramp. For our delay lines, since the input ramp to one stage of the delay line reaches full scale supply voltage (V_{dd}) before the output reaches the $V_{dd}/2$ point, input ramp is considered fast. The output transition time, which is equal to the input rise/fall time to the next stage of the delay line, is defined in [9] and is given by

$$\begin{aligned} t_{TLH} = t_r &= \left(\frac{C_L V_{dd}}{0.7\ I_{Dp_{\max}}}\right) \frac{8 v_{D0_p}^2 (1+\lambda_p V_{dd})}{(4 v_{D0_p} - 1)(2+\lambda_p V_{dd})} \\ t_{THL} = t_f &= \left(\frac{C_L V_{dd}}{0.7\ I_{Dn_{\max}}}\right) \frac{8 v_{D0_n}^2 (1+\lambda_n V_{dd})}{(4 v_{D0_n} - 1)(2+\lambda_n V_{dd})} \end{aligned} \quad (1)$$

where C_L is the load capacitance, $I_{D_{\max}}$ is the maximum drain current at $V_{GS} = V_{DS} = V_{dd}$, v_{D0} is the drain saturation voltage at $V_{GS} = V_{dd}$ normalized by V_{dd}, and λ is the channel length modulation. The subscripts, p and n refer to the PMOS and NMOS parameters respectively.

Daga *et al.* [10] proposed an inverter delay model for fast input ramps based on the alpha-power model and the concept of inverter step response. The velocity saturation index is considered to be unity in [10]. However, PMOSs usually have a velocity saturation index which is greater than unity and greater than NMOSs for current CMOS technologies. In this paper, delay models provided in [10] are generalized to include the non-unity velocity saturation index, α. Using the rise/fall time given in (1), the delay time of a FO4 inverter for the fast input ramp case is given by

$$\begin{aligned} t_{HL} &= t_r \left[\frac{1}{2} - \left(\frac{1-v_{TN}}{\alpha_n + 1}\right)\right] + \frac{C_L V_{dd}}{I_{Dn_{\max}}} \left(1 + 2\frac{C_{GDP}}{C_L}\right) \\ t_{LH} &= t_f \left[\frac{1}{2} - \left(\frac{1-v_{TP}}{\alpha_p + 1}\right)\right] + \frac{C_L V_{dd}}{I_{Dp_{\max}}} \left(1 + 2\frac{C_{GDN}}{C_L}\right) \end{aligned} \quad (2)$$

where v_{TN}, v_{TP} are the zero-bias threshold voltages normalized by V_{dd} and C_{GDP} and C_{GDN} represent the input-to-output coupling capacitances for the PMOS and NMOS transistors respectively.

HSPICE simulations are compared to (2) for a FO4 delay line implemented in 0.13μm CMOS technology. The maximum error between the delay model and simulations is 4-5%. This small margin is considered when designing the emulator.

The FO4 inverter delay model described by (2) is used to model buffered interconnects. When buffers are inserted at optimal distances to minimize interconnect delay, overall delay of the buffered wire is found to be proportional to the square root of the buffer delay [11]. The interconnect delay is related to the buffer delay, $t_{d_{\text{buf}}}$, by the following relation

$$t_{d_{\text{int}}} \propto \sqrt{RC t_{d_{\text{buf}}}} \propto \sqrt{RC}\sqrt{t_{HL}} \quad (3)$$

where R and C are the resistance and capacitance per unit length of the wire. Using (2) and (3) to model voltage scaling behavior of both logic and interconnect delays takes into account process and interconnect variations. Therefore, the critical path at a certain process and certain parasitics corner can be predicted. Considering that worst case delay is the reference case, an algorithm is devised to determine such critical paths. This is described in detail below.

3.3 Algorithm

The algorithm used to generate the information stored in the LUTs for different process and interconnect corners is shown in Algorithm 1. Logic speed, L, and interconnect speed, I, are used as indicators of process and interconnect variations, respectively. In order to take process variations into consideration, the entire logic speed range is divided into increments with each increment is equal to L_{inc}. Similarly, the interconnect speed increment is I_{inc}.

The initial state of the algorithm is determined at worst case logic and interconnect corners. All logic and interconnect speeds are normalized to this reference case. In addition to the reference path, a set of potential critical paths is determined. Delay models given by (2) and (3) are used to predict the voltage scaling behavior of each path in the set. The ratio of interconnect delay to logic delay, I_{ratio}, for each path is also recoded. Based on the logic and interconnect unit delays at worst case in addition to I_{ratio} of each potential critical path, the number of logic, l, and interconnect, i, unit delays required to emulate each path are computed.

Algorithm 1 Critical path emulator

```
START:
L = I = D = 1.0
Find a set of potential critical paths
For each path in the set:
  Compute (l, i)

for (L = 1.0 : L = Fast : L = L - L_inc) do
  for (I = 1.0 : I = Best : I = I - I_inc) do
    Find the reference path
    Find the subsequent potential critical paths
    while (D <> Minimum) do
      Find the critical path when (t_d = D):
      Record its (l, i)
      (j, k) ← (l, i)
      D = Next D
    end while
  end for
end for
```

The next step is to determine which l and i to use in emulating the actual critical path for each target delay, D, specified by the system's software and for each specific logic speed, L, and interconnect speed, I. The delay of each path in the set of potential critical paths is computed using (2) and (3). Then, the path which has a delay equal to the target delay is selected. In this case, delay of all other paths should be less than the target delay. Once the critical path is selected, its (i, j) pair is stored as (j, k) and used for emulation. The same procedure is repeated for the next delay target. Once the generation of the critical path emulator at all target delays is finished, the data required for one LUT in the matrix shown in Fig. 2 is determined. Each LUT is used to store the critical path emulator data for a specific logic and interconnect speed range. The information needed for the entire LUT matrix can be determined by repeating the above for all logic and interconnect speed ranges. The resulting delay of the critical path emulator closely tracks that of the real critical path. More importantly, voltage scaling behavior is nearly the same for both the real critical path and its emulator.

4. RESULTS AND COMPARISON

The above architecture is designed in the CMOS 0.13μm technology. A reference path at worst case with a certain I_{ratio} is selected. The effect of interconnect delay on the selection of a unique critical path is illustrated through a set of paths which have delays close to the reference and lower I_{ratio} (more logic delay). Since potential critical path delays scale faster with voltage, a margin is required which is proportional to I_{ratio} of the reference path. The algorithm described earlier is applied to these paths using the CMOS 0.13μm technology parameters. Logic and interconnect speeds are divided into 10 ranges each. The critical path emulator information for the 10 logic splits and the 10 interconnect parasitic corners is extracted. Therefore, $m = n = 10$ in Fig. 2, yielding a 100 different process and parasitic corners stored in 100 LUTs. In this design example, the number of bits used by the logic and interconnect delay multiplexers is equal to 5 (e.g. $j = k = 5$). Considering 4 target delays, approximately 4-Kbits of ROM are required to form all the LUTs.

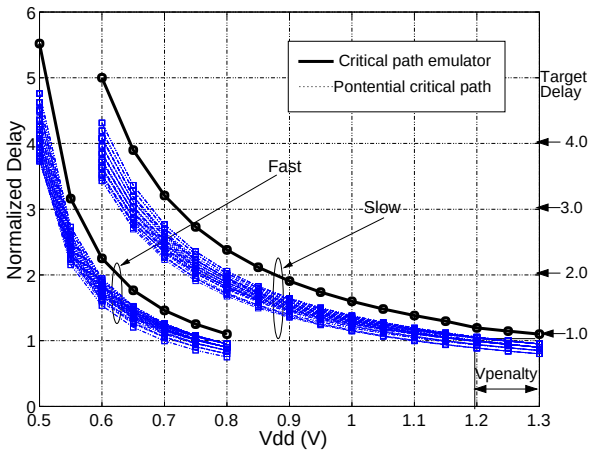

Figure 5: Delay of the critical path emulator exceeds delays of all other paths for the entire voltage range at both slow and fast process corners.

Fig. 5 shows paths delays at the slow and fast corners and the resultant critical path emulator for each case. The reference path delay has an interconnect delay ratio of 50%. For both process corners, the critical path emulator, shown as a solid curve, has a safety margin above all the other paths at all target delays. Target delays, shown on the right of Fig. 5, are set externally by the system's software.

The proposed critical path emulator architecture closely tracks the actual critical path at any given target delay. Therefore, the large delay margin required to account for worst case conditions can be saved. This delay margin is translated to a voltage overhead resulting in an extra energy dissipation which is given by

$$\text{Energy Loss} = 1 - (V_{\text{fast}}/V_{\text{slow}})^2 \qquad (4)$$

where V_{slow} and V_{fast} are the supply voltages required to achieve the target delay with and without using a delay margin respectively.

When logic and interconnect speed intervals are taken to be equal to L_{inc} and I_{inc} respectively, the error range in determining the actual silicon condition becomes $\pm L_{\text{inc}}/2$ and

$\pm I_{\text{inc}}/2$ for logic and interconnect, respectively. Assuming that $L_{\text{inc}} = I_{\text{inc}} = 10\%$, the maximum absolute error becomes 10% which is directly translated to a delay margin. In addition, a delay margin of 5% is added to compensate for model mismatch. Hence, the maximum delay margin required by the proposed system is 15%. From Fig. 5, this delay margin corresponds to a voltage overhead of approximately 115 mV. Using (4), the maximum energy loss of the proposed system is approximately 17%. This energy loss can be reduced by increasing the granularity of process and interconnect speed sampling. However, increasing the granularity entitles more LUTs and additional selection overhead that reduces the energy efficiency.

Conventionally, the reference path is selected at the slow process corner and worst interconnect parasitics. Therefore, conventional open-loop systems require a delay margin to compensate for two factors, process variations in addition to the difference in voltage scaling characteristics between logic and interconnects. Energy savings obtained by adapting to process variations reach 27% when considering a sigma-distribution and the 10 process split information used by the proposed architecture [8].

When the critical and the potential critical path delays are mainly due to logic delay, a replica of the critical path is sufficient to emulate the actual delay. However, as the interconnect delay ratio, I_{ratio}, increases, the delay margin required to accommodate for any sub-critical path formed of pure logic delay also increases. This is due to the fact that logic delay scales faster than interconnect delay. When the reference path delay is assumed to have a certain I_{ratio} and the potential critical path delay is totally due to logic, this delay margin can be derived from

$$\frac{[\text{Margin} + (1 - I_{\text{ratio}})] * [t_{dl}]_{\text{V=V}_{min}} + I_{\text{ratio}} * [t_{di}]_{\text{V=V}_{min}}}{\left(1.0 * [t_{dl}]_{\text{V=V}_{min}} + 0\right)} = 1$$

where $[t_{dl}]_{\text{V=V}_{min}}$ and $[t_{di}]_{\text{V=V}_{min}}$ are logic and interconnect delays at the minimum supply voltage respectively. Consequently, the delay margin can be expressed in terms of I_{ratio} as

$$\text{Margin} = I_{\text{ratio}} \left(1 - \left[\frac{t_{di}}{t_{dl}}\right]_{\text{V=V}_{min}}\right). \quad (5)$$

Based on (5), (4) is used to compute the energy efficiency of the proposed architecture compared to both the conventional open-loop and closed-loop systems as shown in Fig. 6. Since open-loop DVS systems are designed at worst case process and parasitic conditions, the proposed system is up to 43% more energy efficient. Compared to conventional closed-loop systems, only the delay margin given by (5) is required since process variations can be factored out. Therefore, energy efficiency of the proposed system compared to conventional closed-loop systems approaches 23%.

5. CONCLUSION

Conventional voltage scaling systems require a large delay and voltage margins to mitigate the increasing impact of process variations and interconnect delay on the determination of a unique critical path in modern VLSI systems. In order to recover this large margin required by conventional systems, an adaptive voltage scaling architecture with an on-chip critical path emulator was presented. The proposed system has the ability to adaptively track process and parasitic variations and environmental changes through a closed-loop feedback mechanism. Efficiency of the proposed architecture compared to conventional systems depends on the interconnect delay ratio of the reference path. The proposed architecture is up to 43% and 23% more energy efficient compared to open-loop and closed-loop systems, respectively.

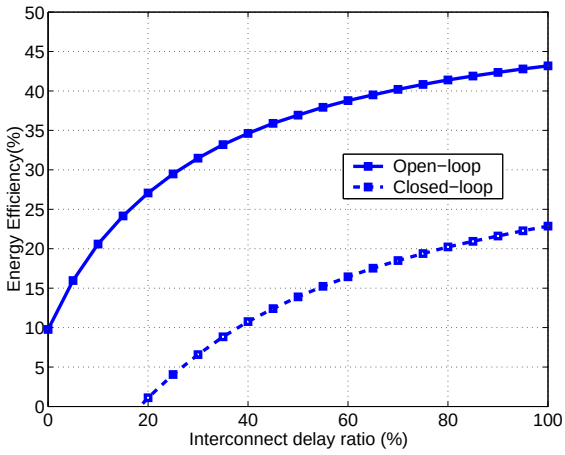

Figure 6: Energy efficiency of the proposed architecture compared to the conventional DVS and AVS systems as a function of interconnect delay ratio of the reference path.

Acknowledgment

The authors would like to thank Muhammad Nummer of the University of Waterloo, Canada, and Amr Fahim and Inyp Kang of Qualcomm Inc. for the enlightening discussions. This work was supported by a strategic grant from the Natural Sciences and Engineering Research Council (NSERC) of Canada.

6. REFERENCES

[1] T. Burd *et.al*, "A Dynamic Voltage Scaled Microprocessor System," *JSSC*, vol. 35, no. 11, pp. 1571–1580, Nov. 2000.

[2] J. Kim and M. Horowitz, "An Efficient Digital Sliding Controller for Adaptive Power-Supply Regulation," *JSSC*, vol. 37, no. 5, pp. 639–647, May 2002.

[3] D. Ernst *et.al*, "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation," in *Micro Conf.*, 2003.

[4] Semiconductor Industry Association, "Itrs, 2003 ed. [http://www.public.itrs.net]" .

[5] A. Chandrakasan et. al, "Data-Driven signal processing: An approach for energy-efficient computing," in *ISPLED*, 1996, pp. 347–352.

[6] G. Wei *et.al*, "A Variable-Frequency Parallel I/O Interface with Adaptive Power-Supply Regulation," *JSSC*, vol. 35, no. 11, pp. 1600–1610, Nov. 2000.

[7] R. Gonzalez and M. Horowitz, "Supply and Threshold Voltage Scaling for Low Power CMOS," *JSSC*, vol. 32, no. 9, pp. 1210–1216, Aug. 1997.

[8] M. Elgebaly *et.al*, "Robust and Efficient Dynamic Voltage Scaling Architecture," in *ASIC/SOC*, 2003, pp. 155–158.

[9] T. Sakurai and R. Newton, "Delay Analysis of Series-Connected MOSFET Circuits," *JSSC*, vol. 26, no. 2, pp. 122–131, Feb. 1991.

[10] J. Daga and D. Auvergne, "A Comprehensive Delay Macro Modeling for Submicrometer CMOS Logics," *JSSC*, vol. 34, no. 1, pp. 42–55, Jan. 1999.

[11] R. Ho *et.al*, "The Future of Wires," *IEEE Proc.*, vol. 89, no. 4, pp. 490–504, Apr. 2001.