A New Paradigm for Low-power, Variation-Tolerant Circuit Synthesis Using Critical Path Isolation

Swaroop Ghosh, Swarup Bhunia*, and, Kaushik Roy
School of Electrical and Computer Engineering, Purdue University, IN, USA
*Electrical Engineering and Computer Science, Case Western Reserve University, OH, USA

Abstract

Design considerations for robustness with respect to variations and low power operations typically impose contradictory design requirements. Low power design techniques such as voltage scaling, dual-V_{th} etc. can have a large negative impact on parametric yield. In this paper, we propose a novel paradigm for low-power variation-tolerant circuit design, which allows aggressive voltage scaling. The principal idea is to (a) isolate and predict the set of possible paths that may become critical under process variations, (b) ensure that they are activated rarely, and (c) avoid possible delay failures in the critical paths by dynamically switching to two-cycle operation (assuming all standard operations are single cycle), when they are activated. This allows us to operate the circuit at reduced supply voltage while achieving the required yield. Simulation results on a set of benchmark circuits at 70nm process technology show average power reduction of 60% with less than 10% performance overhead and 18% overhead in die-area compared to conventional synthesis. Application of the proposed methodology to pipelined design is also investigated.

1. INTRODUCTION

It is well-known that process parameter variations (both systematic and random) may cause parametric failures in logic circuits leading to yield loss. Conventional wisdom dictates a conservative design approach (e.g., scaling up the V_{DD} or upsizing logic gates) to avoid a large number of chip failures. However, such techniques come at the cost of power and/or die area. Process tolerance and low power, therefore, represent contradictory design requirements. Over the past few years, statistical design approach has been widely investigated as an effective method to ensure yield under process variations. Several gate-level sizing and/or V_{th} assignment techniques [1] have been proposed recently addressing the minimization of total power while maintaining the timing yield. On the other end of the spectrum, design techniques (e.g., adaptive body biasing [2]) have been proposed for post-silicon process compensation and process adaptation to deal with process-related timing failures.

Due to quadratic dependence of dynamic power of a circuit on its operating voltage, supply voltage scaling has been extremely effective in reducing the power dissipation. Researchers have investigated logic design approaches that are robust with respect to process variations and, at the same time, suitable for aggressive voltage scaling. One such technique [3] uses dynamic detection and correction of circuit timing errors to tune processor supply voltage. Design optimization techniques using gate sizing and dual-V_{th} assignment to improve power/area typically increase the number of critical paths in a circuit, giving rise to the so-called “wall effect” [4]. The uncertainty-aware design technique [4] describes an optimization process to reduce the wall effect. However, it does not address the problem of power dissipation.

In this paper, we present a novel design paradigm, which achieves robustness with respect to timing failure and provides the opportunity for aggressive voltage scaling by critical path isolation. The notion *critical path isolation* is used throughout this paper to indicate the confinement of critical paths of *synthesized design* to known logic block (or cofactor, as we will see later). Such isolation leads to *a design methodology for low power dissipation by making the critical paths predictable and rare under parametric variations. Any possible delay errors (that may occur under a single cycle operation) are predicted ahead of time and are avoided by two cycle operations (assuming all standard operations are single cycle). This lets us scale the supply voltage aggressively for low power dissipation.* In particular, the proposed technique:

- Isolates the critical paths and makes them predictable (by decoding few primary inputs) under parametric variations so that with reduced supply voltage, possible delay errors are deterministic and can be avoided by *two cycle* operation.
- Restricts the occurrences of the above *two-cycle* operations by reducing the activation probability of critical paths.
- Increases the delay margin between critical and non-critical paths by both logic synthesis and proper gate sizing for improved yield, reliability of operations and low power by voltage scaling.

We also present an application of the proposed methodology in pipeline based design for low power operation. The circuit is re-designed to operate at *fixed low supply voltage* with occasional *two-cycle* operations. The *two-cycle* operations are implemented by *stalling* the pipeline.

Some researchers have proposed techniques to correct variability-induced timing error during operation by voltage scaling. The technique in [3] referred as *RAZOR,* reduces or eliminates voltage margins by dynamic scaling of the supply voltage while monitoring the error rate. Razor allows the occurrence of errors at low voltage and then recovers. However, it does not modify the logic synthesis or gate sizing process and thus can perform poorly in presence of large number of critical paths. The technique proposed in this paper, on the other hand, synthesizes a circuit in specific way to facilitate voltage scaling for power reduction as well as to improve yield by making the delay failures deterministic.

2. PRELIMINARY ANALYSIS

In this section, first we present example of an adder to illustrate the proposed approach for low power robust circuit design. Next, we present the design flow followed by its analysis which allows us to apply similar approach to any random logic circuit.

2.1. Voltage scaling and two-cycle operations in a 4-bit adder

For the sake of simplicity, we choose a 4-bit ripple carry adder as shown in Fig. 1. Signals P_0-P_3 (G_0-G_3) are the propagate (generate) signals whereas $C_{i,0}$ ($C_{o,1}$-$C_{o,3}$) are carry-in (carry-out) signals [5]. As evident, the path from carry-in to carry-out is critical and determines the frequency of operation of the adder. However, note that the critical path is activated only when $C_{i,0} = 1$ and at the same time, $P_0P_1P_2P_3 = 1$. Since the probability of such occurrences is very low (as $p(P_0P_1P_2P_3C_{i,0}=1) = p(P_0)p(P_1)p(P_2)p(P_3)p(C_{i,0})$ is very low), one can reduce the supply voltage such that all operations with $P_0P_1P_2P_3 = 0$ and/or $C_{i,0} = 0$ can still be performed in one-cycle. However, when the critical path is activated, the correct results are obtained by evaluating the adder in two clock cycles (called *two-cycle* operation). The activation of critical path can be predicted by pre-computation of $P_0P_1P_2P_3$. In a nutshell, by making the critical path predictable and utilizing the available slack between critical and non-critical path, it is possible to operate the circuit at reduced supply voltage. Note that this approach incurs penalty of an extra clock cycle when the critical path is activated. However, by ensuring low activation probability of

ICCAD'06, November 5-9, 2006, San Jose, CA
Copyright 2006 ACM 1-59593-389-1/06/0011...$5.00

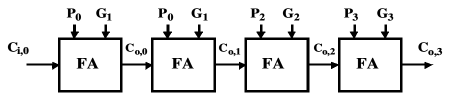


Fig. 1 Ripple carry adder [5]

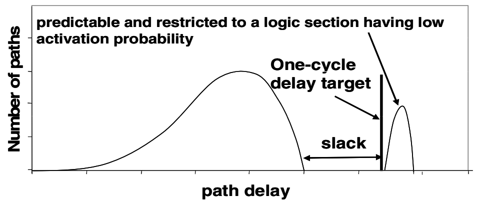


Fig. 2 Path delay distribution needed for the proposed methodology

critical paths, it may be possible to reduce the active and leakage power by rarely paying penalty of an extra clock cycle.

To evaluate the feasibility of this idea, we simulated a 4-bit ripple carry adder with 1V supply in Hspice. We used BPTM [6] 70nm devices for simulation. The critical path delay was found to be 260ps and average power consumption was 13.03uW. Assuming the clock period to be 260ps, we reduced the supply to 0.8V. Now the non-critical paths were within the *single-cycle* delay bound however, the critical path delay increased to 330ps and was evaluated with *two-cycles*. The new power consumption was 7.32uW, leading to 44% saving in total power.

2.2. Generalization to random logic

Earlier, we presented the idea of supply voltage scaling for an adder where the critical path was unique (assuming no process variation). However, a random logic can have many critical paths with corresponding input conditions for activation. Further, the critical paths may vary from chip-to-chip due to parametric variations. In such situations, the overhead associated with pre-decoding logic can overshadow the power savings. To exercise similar supply scaling technique on random logic circuits, we need to make sure that, (a) the critical paths are confined to a predictable logic section; and, (b) the non-critical paths remain non-critical under process variation by providing a safe timing slack. The timing slack between critical and non-critical paths will be the enabling factor for supply voltage scaling. An example of a possible path delay distribution (cartoon) is shown in Fig. 2.

To obtain the delay distribution shown in Fig. 2, the design needs to be partitioned and synthesized in such a way that the paths are divided into several logic blocks. The partitioning procedure should consider the fact that (a) these logic blocks can be active or remain idle based on the state of primary inputs; and, (b) the probabilities of activation of the logic blocks containing critical paths (called *critical block*) are very low. Therefore, it will be possible to predict the activation of a logic block (and the corresponding paths) just by decoding the states of inputs. Next, gate sizing can be performed on the partitioned logic blocks to maximize the slack between critical and non-critical blocks leading to further isolation of critical paths. Note that the suggested sizing approach will be opposite of the conventional sizing because in this case, *the critical paths should be made slower* while *non-critical paths should be made faster. By performing the partitioning and sizing, a path delay distribution similar to the one shown in Fig. 2 can be achieved.* Finally, supply voltage scaling can be done such that *non-critical* blocks meet the desired timing yield with respect to *one-cycle* delay target whereas *critical* block meet the yield with respect to *two-cycle* delay target. In other words, the *critical blocks* can operate in *two-cycles* while the *non-critical blocks* can operate in *single-cycle*. Since the probability of activation of the *critical block* is low, the new design operating at a scaled voltage will have minimum impact on performance. The overall design strategy is shown in Fig. 3. The partitioning and sizing is more clearly illustrated in Fig. 4 where a circuit is partitioned into four functional logic blocks f_1-f_4. The outputs are fed to an OR network to generate the final outputs. Suppose that by the virtue of proper partitioning, f_4 becomes the least activated functional block containing the critical paths. Then f_4 can be downsized further while the other functional blocks can be upsized to maximize the slack and further isolation of critical paths, as shown by arrows in Fig. 4. In Section 3, we will describe a Shannon based partitioning technique which helps in isolating the critical paths.

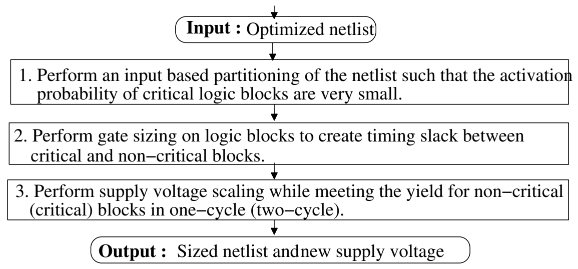

Input : Optimized netlist
1. Perform an input based partitioning of the netlist such that the activation probability of critical logic blocks are very small.
2. Perform gate sizing on logic blocks to create timing slack between critical and non−critical blocks.
3. Perform supply voltage scaling while meeting the yield for non−critical (critical) blocks in one−cycle (two−cycle).
Output : Sized netlist andnew supply voltage

Fig. 3 Design methodology

2.3. Analysis of the proposed design methodology

Let us consider two different designs for the same combinational circuit, *design-A* and *design-B* with timings as shown in Fig. 5. *Design-A* (*design-B*) is representative of conventional design (proposed design). In *design-A*, the slack of critical path is S_1 with respect to the clock period T_c whereas in *design-B*, the critical path (shown by hatched lines in Fig. 5) does not meet the timing constraint and has a negative slack of S_3. However, the non-critical paths (shown by dotted block in Fig. 5) in *design-B* maintain a maximum slack for S_2. We also assume that the activation condition of critical paths in *design-B* is known based on the states of few inputs (say, *N*). An extra decoder is needed in *design-B* for pre-determining the occurrences of critical path activation. Obviously, *design-B* can function properly with *two-cycle* operations for critical paths while a single cycle operation for non-critical paths. Let us now compare the power consumption of *design-A* and *design-B* where V_0 is the voltage at which *design-A* meets the slack requirement S_1, whereas, *design-B* meets slack S_2 for non-critical paths and S_3 for critical paths. Since voltage is proportional to (delay)$^{-1}$, the scaled voltage (V_A^{new}) for *design-A* can be determined as follows,

$$V_0 \propto \frac{1}{T_c - S_1} \text{ and, } V_A^{new} \propto \frac{1}{T_c} \qquad \Rightarrow V_A^{new} = V_0\left(1 - \frac{S_1}{T_c}\right) \tag{1}$$

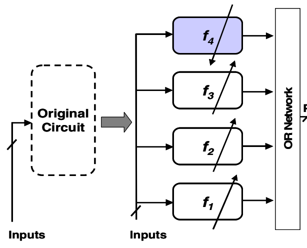


Fig. 4 Steps 1 and 2 of proposed design methodology

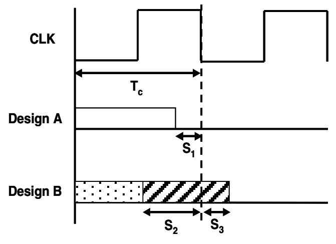


Fig. 5 Timing diagram of *design-A* and *design-B*

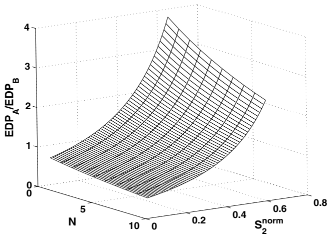


Fig. 6 Plot of EDP ratio of *design-A* and *design-B* for k = 0.2, $C_{d0}^{norm} = 0.001$ and n = 4

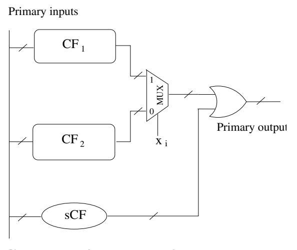


Fig. 7 Shannon's expansion based partitioning

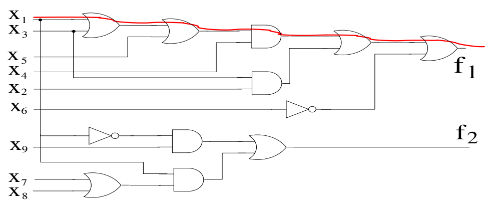
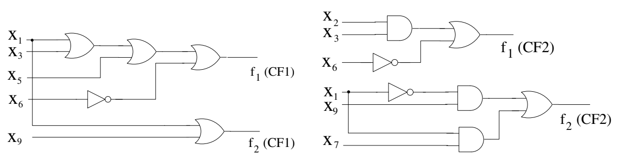
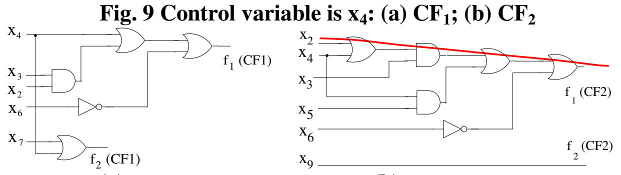


Fig. 8 Original circuit



(a) (b)

Fig. 9 Control variable is x_4: (a) CF_1; (b) CF_2



(a) (b)

Fig. 10 Control variable is x_1: (a) CF_1; (b) CF_2

Similarly,
$$V_B^{new} = V_0\left(1-\frac{S_2}{T_c}\right); \quad s.t.\ S_2 + S_3 \le T_c \tag{2}$$

If the performance penalty due to two-cycle operation in *design-B* is *p*, then the effective clock cycle delay of *design-B* is $T_c + pT_c$. The energy-delay product (*EDP*) of both designs are given by

$$EDP_A = C_A (V_A^{new})^2 (T_c); \quad EDP_B = (C_B + C_d)(V_B^{new})^2 (T_c + pT_c) \tag{3}$$

where C_A (C_B) is the average switched capacitance of *design-A* (*design-B*) and C_d is the average switched capacitance of the decoding logic (for determination of critical path activation).

The *EDP* ratio (after putting the values of V_A^{new} and V_B^{new}) is given by:

$$\frac{EDP_A}{EDP_B} = \left(\frac{1}{\frac{C_B}{C_A}+\frac{C_d}{C_A}}\right)\left(\frac{1-\frac{S_1}{T_c}}{1-\frac{S_2}{T_c}}\right)^2\left(\frac{1}{1+p}\right) = \left(\frac{1}{C_B^{norm}+C_d^{norm}}\right)\left(\frac{1-S_1^{norm}}{1-S_2^{norm}}\right)^2\left(\frac{1}{1+p}\right) \tag{4}$$

where, C_B^{norm} (C_d^{norm}) is the average switched capacitance of *design-B* (decoder logic) normalized with respect to C_A and S_1^{norm} (S_2^{norm}) is the slack of *design-A* (*design-B*) normalized with respect to T_c.

From the expression shown in equation (4), it is possible to study the conditions under which it may be useful to opt for *design-B* rather than *design-A*. It is obvious that *design-B* can be better than *design-A* if $EDP_A/EDP_B > 1$. Since $(C_B^{norm} + C_d^{norm}) > 1$ (assuming $C_B \ge C_A$ due to design modifications) and $(1+p) > 1$, a necessary condition for *design-B* to be better than *design-A* is,

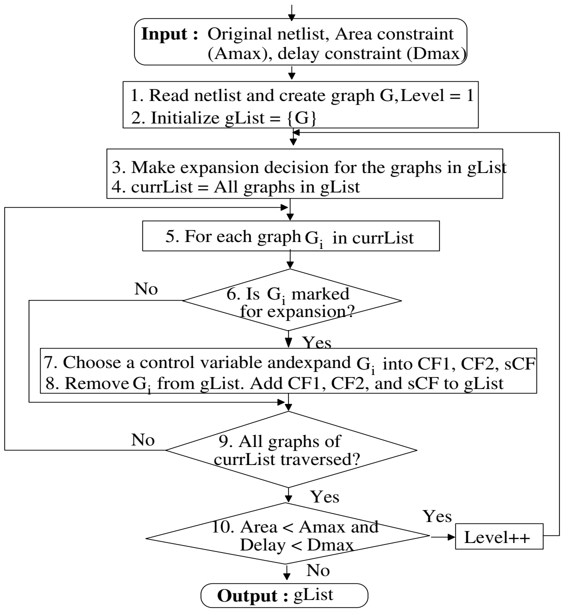
$$S_2^{norm} > S_1^{norm} \ i.e.,\ S_2 > S_1 \tag{5}$$

Therefore, a larger value of S_2 is better for power savings. However, the upper bound of S_2 is determined by constraint $S_2 + S_3 \le T_c$ (equation (2)). Hence, S_2 can be maximized by minimizing slack S_3. Let us explore the design space for which *design-B* can be beneficial. For the sake of simplicity, we model the normalized capacitances and performance penalty (*p*) as follows,

$$C_B^{norm} = 1 + \frac{k}{(1-S_2^{norm})}; \quad C_d^{norm} = NC_{d0}^{norm},\ S_1^{norm}\text{=0.05 and,} \quad p = \frac{N}{2^n}$$

where *k* is a constant, *N* is the number of input vectors that should be decoded to determine if critical paths are activated, C_{d0}^{norm} is the normalized average switched capacitance of decoding a single input vector and, *n* is the total number of primary inputs of the circuit. The EDP ratio plotted for different values of *N* and S_2^{norm} is shown in Fig. 6. From the *EDP* ratio profile shown in this figure, it is obvious that *design-B* is beneficial only if *N* is small (to minimize the switched capacitance of decoding logic). Also, the initial flat portion of the profile indicates that S_2^{norm} should be greater than S_1^{norm}. Although the EDP curve increases with S_2^{norm}, a large value of S_2^{norm} may increase the switched capacitance of the circuit (i.e. C_B^{norm} if gate sizing is used) and offset the saving in power.

In the analysis presented above, it can be concluded that the power saving in proposed method mainly comes from quadratic dependency of power on voltage. Power reduces *quadratically* while the delay increases only *linearly*, letting us reduce the EDP.



Input : Original netlist, Area constraint (Amax), delay constraint (Dmax)
1. Read netlist and create graph G, Level = 1
2. Initialize gList = {G}
3. Make expansion decision for the graphs in gList
4. currList = All graphs in gList
5. For each graph G_i in currList
6. Is G_i marked for expansion?
No
Yes
7. Choose a control variable andexpand G_i into CF1, CF2, sCF
8. Remove G_i from gList. Add CF1, CF2, and sCF to gList
9. All graphs of currList traversed?
No
Yes
10. Area < Amax and Delay < Dmax
Yes
Level++
No
Output : gList

Fig. 11 Automated synthesis flow

3. DESIGN METHODOLOGY

Based on the analysis and the guidelines derived above, we describe the details of each step of the design flow (Fig. 3). This is followed by simulation results on a set of benchmark circuits.

3.1 Circuit partitioning and synthesis for critical path isolation

Let us first consider performing an input based partition of the circuit such that *the critical paths are isolated and their activation probability is reduced*. To achieve this, we used Shannon expansion based partitioning [7] which partitions a Boolean expression *f* into disjoint sub-expressions as shown below:

$$\begin{aligned} f(x_1,...,x_i,...,x_n) &= x_i.f(x_1,...,x_i=1,...,x_n) + \bar{x}_i.f(x_1,...,x_i=0,...,x_n) \\ &= x_i.CF_1 + \bar{x}_i.CF_2 \end{aligned} \tag{6}$$

$$CF_1 = f(x_1,...,x_i=1,...,x_n); \quad CF_2 = f(x_1,...,x_i=0,...,x_n)$$

where ($x_1 \ldots x_n$) are input literals, x_i is control variable, and CF_1 and CF_2 are called cofactors. If *f* contains sub-expressions independent of control variable x_i, then we may also have a *Shared Cofactor* (sCF) (Fig. 7). In this work, we have used Shannon expansion based partitioning mainly due to its following inherent properties: (a) the circuit partitioning is done based on inputs; (b) the activation probability of partitioned logic blocks can be easily reduced by performing multi-level hierarchical expansion; and, (c) by properly choosing the control variables, it is possible to isolate the critical paths to a logic block having least activation probability. In the following paragraphs, first we explain multi-level expansion for reduction of

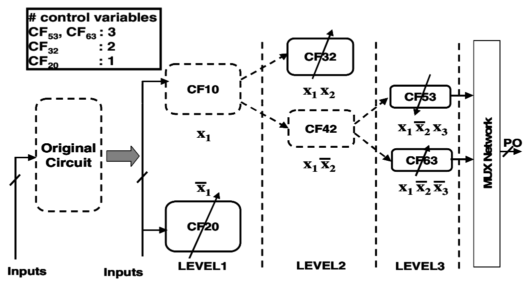


Fig. 12 Hierarchical expansion and sizing of cofactors

activation probability of cofactors, followed by the control variable selection strategy for critical path isolation during partitioning.

In equation (6), the activation probability of each cofactor is 50% (assuming 50% switching probability of inputs). *By performing multi-level expansion, the activation probability of the resulting cofactors can be reduced further.* For example, a 2nd level expansion of f (equation (7)) results in four cofactors, each with an activation probability of 25%.

$$f(x_1,...,x_i,..., x_n) = x_i x_j.CF_1 + x_i \bar{x}_j.CF_2 + \bar{x}_i x_j.CF_3 + \bar{x}_i \bar{x}_j.CF_4 \quad (7)$$

Control variable selection plays a very important role in achieving desired goals in Shannon's expansion based partitioning. In [8, 9], the most binate variable is chosen as control variable to minimize the area overhead. However, this heuristic may not lead to the confinement of critical paths of the circuit after expansion. For example, consider a multiple-output two-level Boolean logic function $f_1 = x_1x_4 + x_2x_3 + x_3x_4 + x_4x_5 + \bar{x}_6$ and $f_2 = x_1x_7 + x_1x_4 + \bar{x}_1x_9$. From the circuit realization shown in Fig. 8, it can be observed that f_1 is the *critical function* (or *critical output*). If $n(x_i)$ is the total literal count of x_i in f_1 and f_2 then, $n(x_1)=4$, $n(x_2)=1$, $n(x_3)=2$, $n(x_4)=4$, $n(x_5)=n(x_6)=n(x_7)=n(x_8)=n(x_9)=1$. Considering most binate variable as the preferable choice, either x_1 or x_4 can be picked as control variable. With x_4 as control variable, resulting cofactors are shown in Fig. 9. It can be noticed that the critical paths are distributed between the cofactors. However, if x_1 is chosen as control variable, the critical path has been confined to $f_1(CF_2)$ (Fig. 10). Clearly, a strategy is needed to isolate the critical paths and limit them to a particular cofactor. If a_i (b_i) is the literal count of variable x_i in true (complement) form in the *critical function* (*or output*), then following criterions should be fulfilled: (i) the control variable should be present in *critical function* (i.e. $min(a_i, b_i) > 0$); (ii) difference of a_i and b_i should be large to ensure that the paths are isolated to one cofactor and, (iii) the $max(a_i, b_i)$ should be small to minimize the probability of logic depth of isolated critical paths being reduced by logic optimization. Following metric can be used:

TABLE-1

```
Procedure performSizing()
Input  : target delay (T_c), yield (Y), list of cofactors (gList);
Output : sized netlist;
1.   maxLevel = maximum hierarchy of the cofactors in gList ;
2.   run SSTA on $G_i \in gList$;
3.   critCF=cofactor with critical paths at maxLevel hierarchy;
4    for each cofactors $G_i \in gList$
5.       calculate $G_i \rightarrow muxdelay$;
6.   end for
7.   $dTarget = \alpha T_c - critCF \rightarrow muxDelay$;
8.   downSize(critCF, dTarget, Y);
9.   $critDelay = critCF \rightarrow maxDelay + critCF \rightarrow muxDelay$;
10.  for each cofactors $G_i \in gList$
11.      if $G_i \neq critCF$
12.          $dTarget = critDelay - T_c - G_i \rightarrow muxDelay$ ;
13.          upSize($G_i$, dTarget, Y);
14.  end for
15.  Add mux's in $G_i \in gList$ to create a complete graph G;
16.  return G;
```

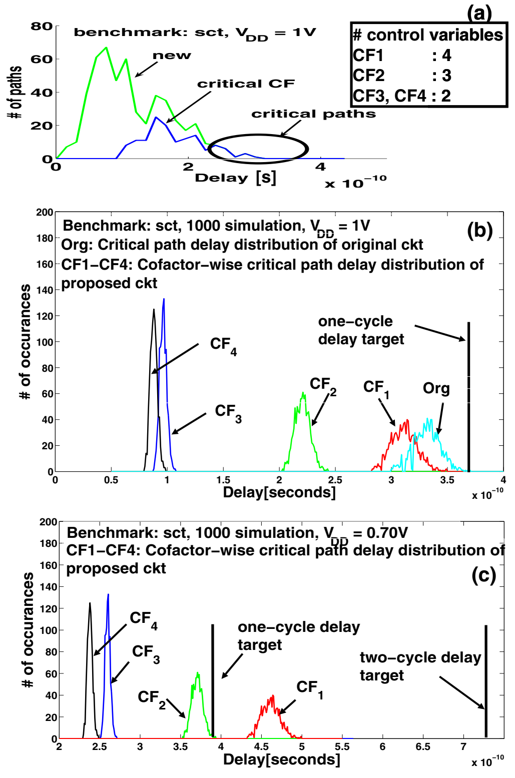


Fig. 13 Results for benchmark *sct*:(a) path delay distribution after partitioning and sizing;(b)cofactor-wise *critical path* delay distribution under V_t variation (V_{DD}=1V), (c) V_{DD}=0.7V

$$M_i = \frac{|a_i - b_i|}{\max(a_i, b_i)} \quad (8)$$

A literal with maximum value of M_i ensures that the critical path is isolated to a cofactor. Using this metric, we follow the steps described in [8] for choosing the control variable in our synthesis flow.

To achieve the dual objectives of isolating the critical paths to a cofactor while reducing its activation probability during partitioning and synthesis, we adhere to following approach: (a) we partition the circuit and determine the cofactor where the critical paths have been isolated (called *critical* cofactor); (b) we mark this cofactor (i.e. *critical* cofactor) for further expansion to reduce the activation probability of the critical paths. The above mentioned steps are repeated under a given area and delay constraint. Note that Synopsys Design Compiler [10] has been used for synthesizing the new cofactors. The overall synthesis flow is shown in Fig. 11. A complete example of hierarchical partitioning and synthesis is also illustrated in Fig. 12 where the original circuit is partitioned into four cofactors, CF_{20}, CF_{32}, CF_{53} and CF_{63}. The critical paths have been isolated to CF_{53} (which is activated by 3 inputs i.e. $x_1x_2'x_3$). Note that, in this example we do not have the shared cofactor (*sCF*). Shared cofactors are important in avoiding the logic duplication during partitioning. However, they are independent of control variable. Therefore our synthesis flow (Fig. 11) automatically chooses it for further expansion (if critical paths are isolated to it).

3.2 Gate Sizing for further isolation

In the previous subsection, we presented a circuit partitioning method to isolate the critical paths to a cofactor with small activation probability. The next step is to size the resulting cofactors individually to (a) further isolate the critical paths and, (b) create timing slack between *critical* and *non-critical* cofactors to allow lowering of supply voltage. To achieve this goal, all gates of the *critical* cofactor are downsized to make the corresponding paths further *critical*. The gates belonging to the remaining cofactors are *selectively* upsized to make them more *non-critical* and increase the slack (S_2, as discussed in Section 2.3). An example of the proposed sizing approach after

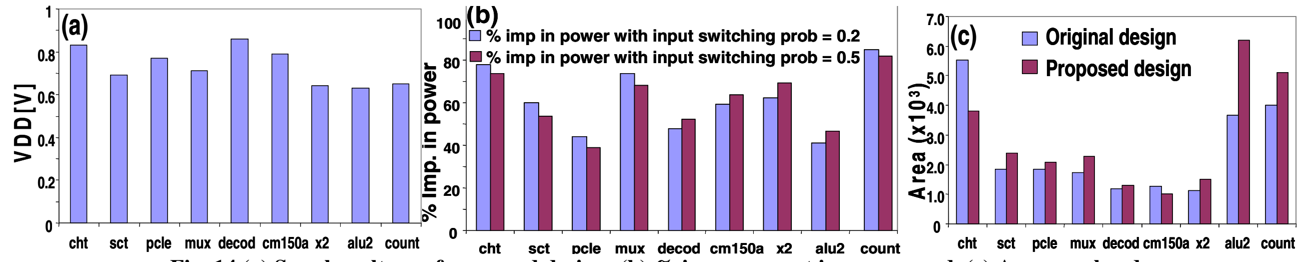


Fig. 14 (a) Supply voltage of proposed design; (b) % improvement in power; and, (c) Area overhead

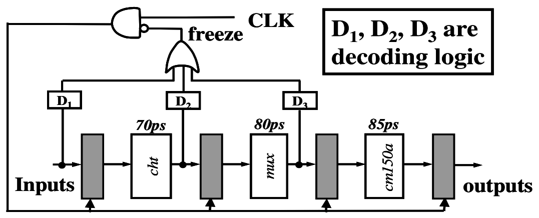


Fig. 15 Example of a pipeline design using proposed method

TABLE-2

```
Procedure pipelineDesign()
Input  : yield (Y), list of circuits(dList), $V_{DDL}$; /* $V_{DDL}$< 1V */
Output : list of re-designed circuits (dList);
1.  target delay ($T_c$) = max(stage delays);
2.  for each design $D_i \in dList$
3.      gList = performPartitioning($D_i$, $V_{DDL}$); /*Fig. 11*/
4       $D_i$ = performSizing(gList, $T_c$, Y, $V_{DDL}$); /*Table 1*/
5.  end for
6.  return dList;
```

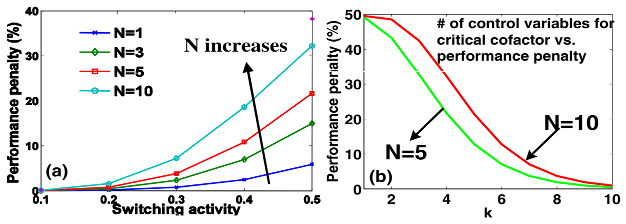


Fig. 16 Performance penalty for (a) *critical* cofactor at *k=4*, (b) different values of *k*

partitioning is shown in Fig. 12. The cofactors with dashed (solid) lines indicate expanded (non-expanded) circuits and *levels* indicate the hierarchy. As shown in the figure, cofactor CF_{53} is downsized to make it further *critical* while other cofactors are upsized to make them more *non-critical*. Note that the proposed sizing approach is very different from the conventional sizing because in this case, *the critical paths are made slower* while *non-critical paths are made faster*.

We follow the above mentioned sizing strategy in a Lagrangian Relaxation (LR) based gate sizing [12] as shown in Table 1. Given a delay target (T_c), it tries to meet the yield requirement with *minimum* area. The procedure takes *gList* (i.e., list of cofactors) and determines the cofactor at highest level of hierarchy, *maxLevel* for downsizing it. The target delay (*dTarget*) for sizing the *critical* cofactor candidate (i.e. *critCF*) is computed in Step 7 (with α=1.2, determined empirically to allow minimization of S_3 as discussed in Section 2.3). The delay targets of *non-critical* cofactors are obtained by subtracting T_c and multiplexer delays from overall critical path delay (Step 12). The non-critical cofactor candidates are now upsized while meeting the yield target (Step 13). The description of Table 1 is omitted for brevity.

3.3 Determination of supply voltage

After circuit partitioning and sizing, we obtain the path delay distribution similar to Fig. 2. Now we may assign a lower supply voltage to reduce the power dissipation while meeting robustness. To achieve this, we start from nominal supply and iteratively reduce it with two stopping criterions: (a) delay violation of any of the *non-critical* cofactors (*one-cycle* delay target) for the given yield constraint; and, (b) delay violation of the *critical* cofactor (*two-cycle* delay bound) for the target yield. Finally, another stopping criterion is the $3V_{th}$ limit for reliable super-threshold operations [5]. The new voltages for a set of MCNC benchmarks are shown in Section 3.4.

3.4 Simulation results

In previous sections, we presented a methodology to make the possible delay errors (that may occur under *single-cycle* operation) predictable and rare (using circuit partitioning and sizing). We also discussed the determination of new supply voltage. In this section, we present simulation result on a set of MCNC benchmarks to demonstrate the feasibility of this methodology. In particular, we show (a) isolation of critical paths to a cofactor (having low activation probability); (b) reduction of supply voltage for low power dissipation while maintaining robustness. In the following paragraphs we present simulation setup followed by the results and discussion.

For logic optimization in our synthesis flow, we have used Synopsys Design Compiler [10]. For a basis of comparison, the original benchmarks are also optimized for area in Synopsys. The mapping is done to a standard cell library. The circuit delays are computed by using SSTA for BPTM 70nm technology. The parametric variations (L, T_{ox}, doping etc) have been lumped into threshold voltage variation. The change in V_{th} due to inter-die (ΔVt_{inter}) and intra-die (ΔVt_{intra}) process variations are modeled as Gaussian variables with zero mean and standard deviations of 80mV and 40mV, respectively. The total change in transistor V_{th} is given by the summation of ΔVt_{inter} and ΔVt_{intra}. The delay target (T_c) for sizing procedure is chosen by plotting the area-delay curve of the circuit and selecting the delay at which the slope of the curve is -1. The area and delay constraints for Shannon based partitioning are kept at 40% and 20% more than original area and delay respectively. The yield targets of original circuit and the cofactors for gate sizing are set to 95%. The yield target of cofactors operating on *one-cycle* (*two-cycle*) after application of reduced supply is fixed to 95% (100%). For power estimation, the circuits are simulated in Hspice by applying a set of 200 random input patterns having input switching probabilities of 0.2 as well as 0.5. The runtime of the entire methodology is found to be small (6.03s for largest benchmark *cht* on SUN blade 1000 workstation).

To illustrate the isolation of critical paths to the *critical* cofactor, we have plotted the path delay distribution of an example MCNC benchmark circuit (i.e., *sct*) after partitioning and sizing (Fig. 13(a)). This figure clearly indicates that the critical paths of the re-synthesized design are limited to the *critical* cofactor. We also present it's cofactor-wise critical path delays distribution under process variation (V_{th} variation, Fig. 13(b)). From this figure, note that: (a) CF_1 remains *critical* even under parametric variation while the other cofactors remain *non-critical* and; (b) there is a delay slack present between CF_1 and other cofactors. Also, note that the *critical* cofactor CF_1 is at the 4th hierarchical level (i.e. 4 control variables) to minimize its activation probability. The delay distribution at reduced supply is shown in Fig. 13 (c). It shows that CF_1 operates in *two-cycles* while rest of the cofactors operates in *single-cycle*.

In Fig. 14, we show the area, power and new supply voltage for a set of MCNC benchmark circuits. It can be observed from Fig. 14 (a)

TABLE-3 SIMULATION RESULTS FOR 3-STAGE PIPELINE

V_{DDL} (V)	% Imp in power			Overall imp (%)	% Area overhead			Overall area penalty (%)	# of cycles reqd. [*cht, mux, cm150a*]	Perf. penalty (%)
	cht	*mux*	*cm150a*		*cht*	*mux*	*cm150a*			
0.75	40.06	59.70	-5.24	41.99	62.79	28.96	159.7	75.75	[1,2,2]	10.9
0.80	64.64	58.20	56.69	62.16	-10.69	15.17	11.87	0.50	[1,2,2]	10.9
0.85	**61.27**	**56.71**	**54.59**	**59.43**	**-17.51**	**5.86**	**0.76**	**-7.85**	**[1,1,2]**	**5.89**
0.90	49.49	50.74	39.63	49.05	-12.55	1.37	6.51	-5.01	[1,1,2]	5.89

that supply voltage required for the re-designed circuits are significantly less than nominal supply (1V). This results in 60% average saving in total power as shown in Fig. 14 (b). The partitioning is performed such that the *critical* cofactors of *all* the benchmarks are at the 4th level of hierarchy. Therefore, the activation probability of critical paths (thereby, the number of *two-cycle* operations) is very low. This is also discussed in next section. Fig. 14 (c) shows the area overhead (18% on average) associated with the proposed design methodology. This comes from two sources: (a) logic duplications during Shannon based partitioning and; (b) upsizing of certain cofactors. However, for two benchmarks namely, *cht* and *cm150a*, we observed area improvement. This is mainly due to better optimization after control variable isolation and multi-level Shannon expansion [8].

It is worth noting that even if the circuit path delays are skewed towards the target delay (as in many industrial circuits), it is possible to create a delay slack by proper partitioning and sizing.

4. APPLICATION IN PIPELINE BASED DESIGN

In this section, we investigate application of the design methodology described in Section 3 to pipeline-based design. We assume that *each pipeline stage is designed using this methodology*.

4.1. Pipeline design methodology and performance analysis

Our pipeline design methodology is based on a *given* reduced supply voltage constraint. The procedure is shown in Table 2. The maximum stage delay is chosen as target delay (T_c) for all the stages (step-1). Next, one design is picked at a time and circuit partitioning is performed as explained in Section 3 (step-3). Note that the delays are computed by using SSTA at *specified supply voltage* (V_{DDL}). The output of step-3 is a list of cofactors which is sized to meet the required delay target at supplied voltage (step-4). Steps-2 to 5 is repeated on each of the stages.

Next, let us evaluate the performance of the new pipeline design. Consider a 3-stage linear pipeline after re-design (Fig. 15) where decoders D_1, D_2 and D_3 predict the activation of critical cofactors of the individual stages. A *two-cycle* operation is needed whenever the *critical* cofactor of any of the pipeline stages is activated. Under such circumstances, the pipeline is *stalled* by gating the clock (using signal *freeze* in Fig 15). Let p_i be the activation probability of *critical* cofactor of i^{th} stage and p_{total} is probability of *two-cycle* operation in the pipeline. Further, we assume that *critical* cofactors of each of the stages have same number of control variables. Hence,

$$p_1 = p_2 = \ldots p_N = p = (\text{input switching activity})^k \quad (9)$$

where k is the hierarchy (or, number of control variables) of *critical* cofactor. Then p_{total} is given by

$$p_{total} = 1-(1-p)^N \quad (10)$$

If the ideal clock cycle-per instruction (CPI) of the pipeline is given by CPI_{ideal}, then the new CPI is given by

$$CPI_{new} = CPI_{ideal} + p_{total}.(\text{stall penalty}) \quad (11)$$

The performance penalty due to occasional two-cycle operation is given by

$$\text{Perf. penalty} = \frac{CPI_{new} - CPI_{ideal}}{CPI_{new}} = \frac{p_{total}.(\text{stall penalty})}{CPI_{ideal} + p_{total}.(\text{stall penalty})} = \frac{p_{total}}{1+p_{total}} \quad (12)$$

The performance penalty for different N and input switching activities is shown in Fig. 16(a). In this plot, we assume that the *critical* cofactor of each stage is activated by 4 inputs (i.e. $k = 4$). It can be observed from this plot that with the switching activities of the control variable between 0.1 and 0.3, performance penalty can be restricted within 10%. For deeper pipeline (i.e. large N), the penalty may increase. We suggest following techniques for reducing the performance penalty, (a) tune the control variable selection metric to pick low switching inputs as control variables; and, (b) reduce the activation probability of *critical* blocks further (i.e., by increasing k as shown in Fig. 16(b)).

4.2. Simulation results

We performed experiment on a 3-stage pipeline where each stage is an MCNC benchmark circuit. The pipeline with the stage delays (for 95% yield with BPTM 70nm devices) is shown in Fig. 15. After performing step-1 of the *pipelineDesign()*, delay target is chosen to be 85ps. We re-design the pipeline stages for V_{DDL}'s ranging from 0.75V to 0.90V. After re-design, the entire pipeline is simulated in Hspice using 200 random test patterns with uniform switching activity of 0.5. The results are shown (Table 3). It is interesting to note that the overall power saving increases initially but declines at lower supply voltages. This is due to increased switching capacitance to meet the delay target at low supply voltage. The negative value of area overhead for *cht* is due to better optimization (Section 3.4). It should also be noted that the *critical* cofactor of *cht* does not need *two-cycle* operations for the given range of V_{DDL}. This is due to the increased delay target. Circuit *mux* may need *two-cycle* operations only when V_{DDL} = 0.8V and 0.75V. However, circuit *cm150a* may need *two-cycle* operations for the entire voltage range. Therefore, the pipeline performance penalty varies between 6-11%. Table-3 clearly shows that it is beneficial to design the pipeline for V_{DDL} *= 0.85V* where the power saving is significant (~60%) with low performance penalty (~6%). Similar technique could also be extended for an N-stage pipeline.

5. CONCLUSION

We have proposed a new design paradigm based on critical path isolation, which achieves low power operation while being robust with respect to parametric delay failures. The proposed design technique makes the possible delay errors predictable and rare under parametric variations. The critical paths have been isolated to a *known* logic block by Shannon partitioning and gate sizing. This leads to a robust circuit design which allows us to reduce the supply voltage aggressively while using the predictability to prevent occurrence of any delay violations. We have also demonstrated that this technique can be effectively applied to low power pipeline design.

Acknowledgement: We thank GSRC MARCO for supporting this research. We also thank Dr. Saibal Mukhopadhyay for valuable suggestions and discussions.

REFERENCES

[1] A. Srivastava et al., Statistical optimization of leakage power considering process variations using dual-V_{th} and sizing, DAC 2004.
[2] S. Borkar et al., Design and reliability challenges in nanometer technologies, DAC, 2004.
[3] D. Ernst et al.., Razor: a low-power pipeline based on circuit-level timing speculation, MICRO, 2003.
[4] X. Bai et al., Uncertainty-aware circuit optimization, DAC, 2002.
[5] J. M. Rabaey, Digital integrated circuits, *Prentice Hall*, 1996.
[6] BPTM 70nm: Berkeley predictive technology model.
[7] L. Lavagno et al., Timed Shannon Circuits: A Power-Efficient Design Style and Synthesis Tool, DAC, 1995.
[8] S. Bhunia et al., A novel synthesis approach for active leakage power reduction using dynamic supply gating, DAC, 2005.
[9] S. Kundu et al., Design of robustly testable combinational logic circuits, TCAD, 1991.
[10] Synopsys Design Compiler, www.synopsys.com.
[11] K. Kang et al., Statistical timing analysis using levelized covariance propagation, DATE, 2005.
[12] B. C. Paul et al., Novel sizing algorithm for yield improvement under process variation in nanometer, DAC, 2004.