Journal of Circuits, Systems, and Computers
© World Scientific Publishing Company

LEAKAGE POWER REDUCTION OF ASYNCHRONOUS PIPELINES

BEHNAM GHAVAMI, HOSSEIN PEDRAM, AREZOO SALARPOUR

Computer Engineering Department, Amirkabir University of Technology
Tehran, Iran
ghavamib@aut.ac.ir, pedram@aut.ac.ir, arezoosalarpour@gmail.com

Received (30 5 2009)
Revised (19 6 2010)
Accepted (7 8 2010)

With CMOS technology scaling, leakage power is expected to become a significant portion of the total power. Dual-threshold CMOS circuit, which has both high and low threshold transistors in a single chip, can be used to deal with the leakage problem in high performance applications. This paper presents dual-threshold voltage technique for reducing leakage power dissipation of Quasi Delay Insensitive asynchronous pipelines while still maintaining high performance of these circuits. We exploited Dependency Graph model to produce a formal performance analysis. In order to reduce leakage power an efficient algorithm for selecting and assigning high threshold voltage to templates of a pipeline is proposed. Results obtained indicate that our proposed technique can achieve on average 40% savings for leakage power, while there is no performance *penalty*.

Keywords: Leakage Power; Asynchronous Pipeline; Threshold Voltage.

1. Introduction

Reduction in the size and the growth in number of the transistors in contemporary circuits signify the problem of global synchronization. Asynchronous circuits which operate without the aid of a global clock have been proposed as an alternative to synchronous circuits based on the potential for decreased dynamic power dissipation, increased performance, reductions in electromagnetic interference, and other characteristics Ref. 12.

As asynchronous circuits gain popularity due to their potential advantages, such as dynamic power saving and high performance, the complexity of design and optimization methods is highlighted. Among the numerous asynchronous design styles being developed, template-based fine-grained pipelines have decreased the complexity of design effort Ref. 3-4. Template based approaches also have the advantage of removing the need for generating, optimizing, and verifying specifications for complex distributed controllers, which is both difficult and error-prone Ref.5.

In recent years, power dissipation has emerged as the key issue for the design of both high performance as well as hand-held portable systems. In the sub-micron regime, supply and threshold voltages are reduced with the scaling of CMOS technology. The lowering of threshold voltage leads to an exponential increase in the subthreshold leakage current Ref.86-7-8. In asynchronous circuits as one class of VLSI circuits, leakage power increases with the scaling of CMOS manufacturing technology into deep sub-micron era Ref.12. Hence, designers require techniques that reduce leakage power while maintaining high performance of these circuits. One of the main component of the leakage power is due to the subthreshold leakage current and it is becoming an increasingly dominant component of overall power consumption in deep sub-micron technologies Ref.11. There is a lot of researches around employing subthreshold leakage power reduction techniques in synchronous circuits Ref.6-7-8-9-11. Based on our good knowledge, there has not been any report of leakage power reduction techniques in asynchronous circuits.

In this paper, we have introduced an efficient methodology for exploiting dual-threshold voltage techniques in Quasi Delay Insensitive (QDI) asynchronous pipelines. Dependency Graph model is used to produce a formal performance analysis. The remainder of this paper is organized as follows. We discuss about Quasi Delay Insensitive (QDI) asynchronous pipeline in Section 2. Section 3 is a review over background of dual-Vth technique and some discussions. Then Section 4 elaborates the proposed methodology for dual-Vth asynchronous linear pipeline. Section 5 is about our experimental results in detail by the use of some related benchmarks. Finally, some conclusions are drawn in Section 6.

2. Asynchronous Pipeline

Asynchronous circuits represent a class of circuits not controlled by a global clock but rely on exchanging local request and acknowledge signaling for the purpose of synchronization. An asynchronous circuit is called delay-insensitive (DI) if it preserves its functionality, independent of the delays of gates and wires Ref.13. It is shown that the range of the circuits that can be implemented completely DI is very limited Ref.18. Therefore some timing assumptions exist in different design styles that must hold to ensure the correctness of the circuit. Different techniques distinguish themselves in the choice of the compromises to the delay-insensitivity. Quasi delay insensitive (QDI) circuits are like DI circuits with a weak timing constraint Ref.13. QDI asynchronous circuits are composed of concurrent processes connected through handshake channels. These processes can be decomposed into fine grain processes that all fit in a fine grained template Ref.18. QDI implementations appear to be the most appropriate – class of asynchronous circuits that can be synthesized automatically from large high-level behavior specifications. This is because of the week timing constraint that can be easily managed in this design style. At present, most QDI pipelines are designed using Pre-Charged Half Buffer (PCHB) and Pre-Charged Full Buffer (PCFB) templates. This paper discusses the idea of leakage power reduction in the context of PCFB QDI circuits. However, proposed methodology can extend to other templates easily.

Template based asynchronous circuits are comprised of deterministic and non-deterministic pipelines. A deterministic pipeline is a pipeline system that does not contain choice pipeline stages that exhibit choice behavior such as arbitration, splits, and merges.

For efficiency and clarity, it is useful to realize that the deterministic pipeline in this work can be partitioned into two classes. The first corresponds to linear pipeline where it performance constraint is to the interaction of neighboring pipeline stages in the circuit model. The second class, fork-join pipeline, where it dependencies corresponds to sequences of signal transitions that map to other alternate paths (linear pipelines) between pairs of circuit stages, i.e., fork-and-join paths. Figure 1 and Figure 2 show the PCFB linear and fork-join pipelines respectively.

A template at stage_n becomes active when it senses the presence of an incoming data from function block of stage_{n-1}. It then performs the computation and sends the result via output channels to stage_{n+1}. Communications through channels are controlled by handshake protocols Ref.13. Each one of these templates implements the same sequence of handshaking in the communicating channels. The encodings of the channels can be in a variety of ways. Return to zero handshaking protocol with dual-rail data encoding that switches the output from data to spacer and back is the most common QDI implementation form. One of the major protocols used in asynchronous circuits is four-phase protocol.

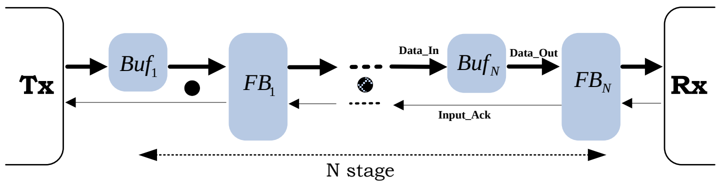


Fig. 1. PCFB linear-pipeline

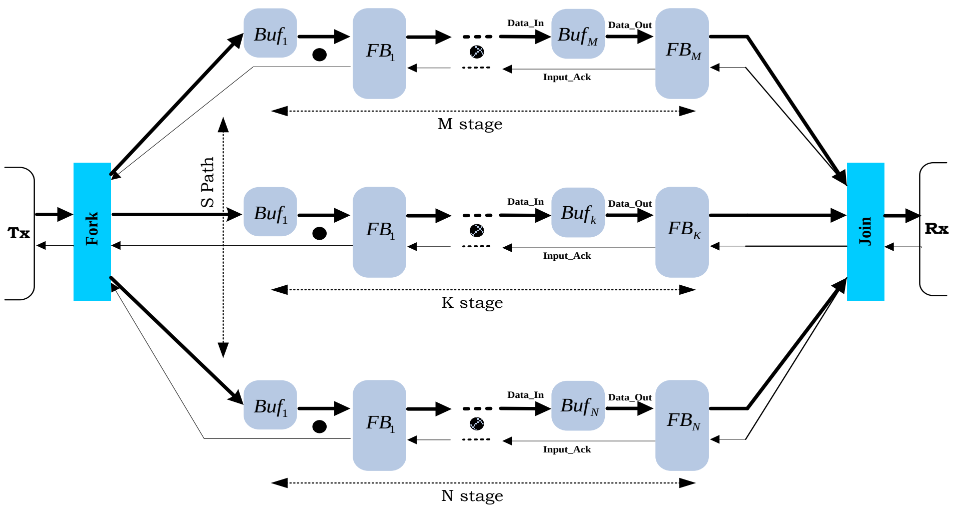


Fig. 2. PCFB fork-join pipeline

In four phase protocols, the data pattern consists of two tokens. The Evaluation Token which shows the data (01 or 10) and Reset Token which resets the logic to prepare it for the next evaluation. The token becomes a Bubble if the Acknowledgement signal, confirming that the following stage does not need the token any more, is received. Buffer Cycle Time, C_B , is the time needed by the buffer at $stage_n$ to process a complete data pattern.

Contributions to Journal of Circuits, Systems, and Computers will be reformatted from the electronic file provided by the authors. Contributions are to be in American English. Authors are encouraged to have their contributions checked for grammar. Abbreviations are allowed but should be spelt out in full when first used. Integers ten and below are to be spelt out. Italicize foreign language phrases (e.g., Latin, French). Upon acceptance, authors are required to submit their data source file including postscript files for figures.

3. Dual-Threshold voltage Design: Background and Discussion

Researchers have proposed different circuit techniques to reduce subthreshold leakage power of synchronous circuits Ref. 8. Multiple-threshold CMOS circuit, which has both high and low threshold transistors in a single chip, can be used to deal with the subthreshold leakage problem in low power and high performance applications. The high threshold transistors can suppress the subthreshold leakage current, while the low threshold transistors are used to achieve the high performance. Recently several multiple-threshold CMOS circuit design techniques have been provided such as multi-threshold voltage CMOS (MTCMOS), dual-threshold CMOS, variable-threshold CMOS (VTHCMOS) and dynamic-threshold CMOS (DTMOS), etc Ref.8.

In dual-threshold technique, a higher threshold voltage can be assigned to some transistors in non-critical paths so as to reduce leakage current, while the performance is maintained due to the low threshold transistors in the critical path(s). Since dual-threshold design technique can reduce both active leakage power and standby leakage power without delay and area overheads, it is very attractive for low voltage and high performance circuit design. However, due to the complexity of a circuit, not all the transistors in non-critical paths can be assigned a high threshold voltage; otherwise, the critical path may change, thereby increasing the critical delay. Recently, researchers have proposed many design techniques, for selecting and assigning an optimal high threshold voltage to gates of synchronous circuits which reduce leakage power under performance constraints Ref. 9.

But importantly, in case of asynchronous circuits, applying these techniques has serious problems. This is due to facts that dual-Vth design influences the performance of circuits and analysis of the performance of asynchronous circuit remains a stumbling block (due to the dependencies between highly concurrent events). While synchronous performance estimation is based on a static critical path analysis affected only by the delay of components and interconnecting wires, it has been shown that the performance estimation of an asynchronous circuit is more complex Ref.14-15-16. In asynchronous circuits, the operation of a system proceeds at a rate determined by speed of its individual components, and the sequencing of the operation of components. The techniques required to analyze asynchronous systems resemble those used to determine the clock period of a synchronous system, which is, summing the delays along the longest path through the combinational logic connecting adjacent latches. In the clocked case, the critical path has a clear beginning and a clear end because all paths are broken by latches. But importantly, no clear separation is available in asynchronous circuits. Analysis procedures must deal directly with cyclic critical paths; thus, existing critical-path analysis tools cannot be easily applied to this problem. More details will be explained in the next section.

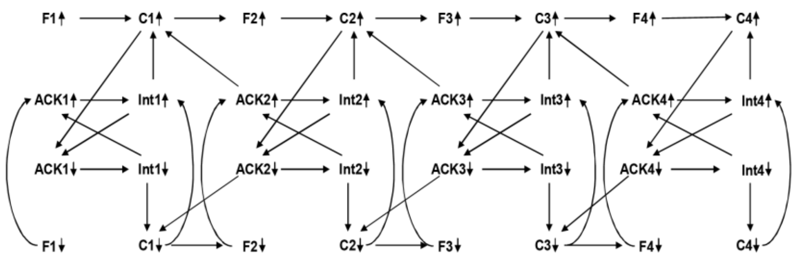
In brief, traditional dual-Vth techniques cannot be exploited directly to asynchronous circuits. In the next section an abstract performance model of asynchronous pipelines on which the dual-Vth problem can be applied is proposed.

4. Dual-Vth PCFB pipelines design methodology

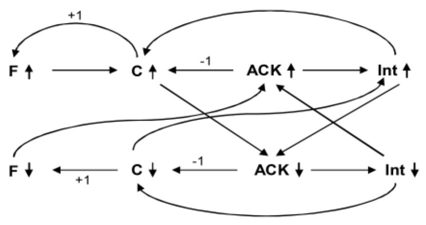
To employ dual-Vth technique, at first a suitable performance model for PCFB pipelines is presented. At last an efficient algorithm for assigning high and low Vth to components of PCFB circuits is proposed.

4.1. *Formal Performance-Analysis*

For analyzing of the pipeline performance, a formal and detailed equation is needed. We need exact equation to assign the suitable Vth for their circuits without affecting the performance.
In order to determine the cycle time of a pipeline, it is necessary to analyze the dependencies of the required sequence of transitions. These dependencies can be drawn in a marked directed graph where the nodes of the graph correspond to specific rising or falling transitions of circuit components, and the edges represent the dependencies of each transition on the outputs of other components. The delay of each transition is represented by a value attached to the corresponding node in the graph. These graphs will be called "Dependency Graphs" Ref.17. If all the stages have the same function blocks, the graph can be folded. Each edge in the Folded Graph is annotated with an integer weight giving the offset in stage indices to which that dependency refers. More details can be found in Ref.17. Figure 3 shows the Unfolded and Folded dependency-graphs of the PCFB Pipeline. In particular, each cycle in the graph has a cycle metric that is the sum of the delays of all associated transitions. The Total Cycle Time C_{Total} is "The total time the circuit takes to completely process a complete data pattern".



(A)



(B)

Fig. 3. PCFB (A): Unfolded / (B): Folded Dependency Graphs. (C: Buffer output, F: Function-Block output, Ack: Acknowledgment Signal, Int: C4 output, ↑/↓: UP/Down Transitions)

We use the methodology based on the Unfolded Dependency-Graph which introduce in Ref.17 for cycle time analyzing. This methodology can be summarized as follow.
(1). Make the Dependency-Graph of the circuit. (2). Define the main synchronization event-transition(an event-transition where all the circuit events are requisites). (3). From this transition, going up for example, trace the circuit until reaching the dual transition considering the longest of any parallel event sequences. This forms the first part of the equation. (4). Returning back to the main synchronization event-transition, considered in step 3, while considering the longest of any parallel event-sequences, forms the second part of the equation.
In the following, we apply this method to PCFB buffer type to extract a general stage equation $\mathrm{C}_{\mathrm{PCFB}_n}$. In this equation, the performance of any stage depends on the previous, current and next stages. Subsequently, the total cycle time ($\mathrm{C}_{\mathrm{Total}}$) will be the maximum of these stage equations and it can be obtained by the general equation of eq.1. Where n, is the stage index, N is the total number of stages and $\mathrm{C}_{\mathrm{PCFB}_n}$ is the general stage equation of PCFB buffer in the linear pipeline.

$$C_{Total} = MAX_{(n=1..to..n=N)}[C_{PCFB_n}] \tag{1}$$

$$C_{PCFB_n} = \mathrm{Max}\,[(\mathrm{TF}(n+1)\uparrow + \mathrm{TC}(n+1)\uparrow + \mathrm{TAck}(n+1)\downarrow + \mathrm{TC}(n)\downarrow), (\mathrm{TAck}(n)\downarrow + \mathrm{TInt}(n)\downarrow + \mathrm{TC}(n)\downarrow)] + \mathrm{Max}\,[(\mathrm{TF}(n+1)\downarrow + \mathrm{TAck}(n+1)\uparrow + \mathrm{TC}(n)\uparrow), (\mathrm{TInt}(n)\uparrow + \mathrm{TC}(n)\uparrow)]. \tag{2}$$

Up to now only cycle time of linear pipelines is introduced. The second class, fork-join dependencies, corresponds to sequences of signal transitions that map to other undirected loops in the circuit model corresponding to alternate paths between pairs of circuit stages, i.e., fork-and-join paths. In synchronous pipelines, the number of pipeline stages along different paths between two stages must be the same to ensure correctness. In contrast, this is guaranteed for the correct operation of an asynchronous pipeline. However, widely unbalanced numbers of stages in fork-join paths may impact performance because one path may prevent data or bubble injection into the other path. These cycles define a fork-join cycle time that may also limit circuit performance. It is ambiguous for us to calculate the cycle time equations using the above loops and very hard to formulate an equation to express the cycle time of fork-join pipeline. Since, we introduce a new performance metric for this type of pipelines using cycle equations of linear pipeline to assign the suitable value of Vth for their circuits without affecting the performance.
Using RC-delay model, relation of cycle time (performance metric of asynchronous pipeline) by the value of Vth can be resolved.

4.2. *Leakage and delay models*

The leakage power of a CMOS circuit is determined by the leakage current through each transistor which has two main sources, reversed biased diode junction leakage current and subthreshold leakage current. Diode junction leakage is very small and can be ignored Ref.19. Leakage current of a CMOS circuit is the sum total of the leakage currents of all of the gates. According to the BSIM model Ref.20, leakage current of a MOS transistor can be approximated as follows:

$$I_{sub} = Ae^{q(Vgs-Vt)/nKT}(1-e^{\frac{-qV_{ds}}{kT}}) \tag{3-1}$$

where $A=\mu_0 C_{ox}\frac{W}{L}(\frac{KT}{q})^2 e^{1.8}$ in which, W and L are the effective device geometries and C_{ox} is the gate oxide capacitance per unit area. The term KT/q represent the thermal voltage. μ_0 is the zero bias mobility and Vth is the threshold voltage.

The relation between delay times and Vth is described in Ref.19. They reported that the high Vth cell has a larger delay than low Vthh cell. The gate delay of an inverter is expressed as

$$Delay\ \alpha\ \frac{C_{load}\cdot V_{DD}}{\mu.C_{ox}\cdot\left(W_{gate}/L_{gate}\right)\cdot(V_{DD}-V_t)^{\alpha}} \tag{3-2}$$

where C_{load} is load capacitance, V_{DD} is supply voltage, μ is the career mobility, W_{gate} is gate width, and L_{gate} is gate length.

According to equations *3-1* and *3-2*, dual threshold technique can reduce leakage power while maintaining performance of circuits by assigning a high threshold voltage to some transistors in non-critical loops, and using low-threshold transistors in critical loops. This leads to decrease the total power consumption. In the next section we present a heuristic algorithm to efficiently design of dual-Vth asynchronous pipelines.

5. A Heuristic Algorithm for Dual-Vth Assignment

In this section we propose a heuristic algorithm for the assignment of low-Vth and high-Vth to the elements of PCFB pipeline in such a way that the pipeline delay remains same as in the case of all low-Vth design, but reduce sub-threshold leakage current significantly. In this work, only deterministic pipelines are considered. As mentioned earlier in our model, deterministic pipeline is classified into two classes, linear pipeline and fork-join pipeline. The proposed method exploited these pipeline structures in tow consecutive phases.

5.1. *Phase A:*

The first step of the proposed method is to determine all possible linear pipelines of the circuit and subsequently constructing the Dependency Graph model of each pipeline. A dependency graph is a directed graph representing dependencies of several transitions towards each other. It is possible to derive an evaluation order or the absence of an evaluation order that respects the given dependencies from the dependency graph. Dependency graph represents the dependencies between signal transitions in the circuit. The nodes in such a dependency graph represent rising or falling signal transitions, and the edges represent dependencies between the signal transitions. Formally, a dependency is a marked graph.

The second step of algorithm is to initialize the circuit with a single low threshold voltage (The low-threshold is determined by the performance requirement). After initialization, all delay parameters ($R_N, T(x)\downarrow, T(x)\downarrow$) associated with each node of Dependency Graphs are computed. Using eq.2, the cycle time of each pipelines stage is determined and then eq.1 determines the global cycle time of every pipeline. In each pipeline, a cycle which has this cycle time is called as critical cycle of the pipeline.
The next step is to assign a high threshold to some elements on non-critical cycles under performance constraints. This is performed by assigning a slack for each cycle in a Dependency Graph model of pipeline. Slack of a cycle ($T_\delta(x)$) denotes the amount by which the cycle can be slowed down without affecting the pipeline performance. Greater slack means that more nodes in the cycle can be assigned to high-Vth.
It is more important to note that, the most known algorithms enumerating the elementary cycles of a directed graph (DG) are based on a backtracking strategy and so finding all cycles in a DG is a time consuming task Ref.21. However in asynchronous linear pipeline, only sequences of up to three stages contribute dependency constraints and no pipeline strategies in which the local pipeline constraint are delimited by more than three stages Ref.14 and moreover, the topology of Dependency Graphs for all of the PCFB pipelines is identical and so it is possible to find all the cycles in our model easily.
For the critical cycle of a pipeline which is determined by eq.1, slack is 0, and for any other cycle of the same pipeline it can be expressed by:

$$T_\delta(x) = C_{Total_n} - C_x \tag{4}$$

Where $T_\delta(x)$ is the slack of an arbitrary cycle 'x' and C_{Total_n} is the total cycle time of a pipeline which is determined with *equation 1* and C_x is the cycle time of the cycle 'x'.
At this step, slack of all non-critical cycles are resolved and all elements that do not participate in the pipeline's critical cycles have a potential to be assigned to high-Vth on the basis of cycle slack's value which contains these elements.
The procedure for choosing a high-threshold node in a pipeline works as follows. By definition (eq.4), for each cycle in a single threshold circuit, its slack is no less than 0. Increasing the threshold voltage of an element can result in a higher propagation delay and departure time of some nodes in Dependency Graph model. In this performance model by increasing the Vth of an element of PCFB templates, values of both $T(x)\downarrow$ and $T(x)\uparrow$ will be changed and therefore, some cycle stacks will decrease. An element should be assigned to a high threshold when the slack of the associated cycles in Dependency Graph remain positive after changing the threshold to high value.
Note that, some elements are located on more than one cycle and we must take into account the minimum value of slacks associated with this node. If we consider a non-minimum value, the associate element(s) in PCFB templates have the potential to be assigned to high Vth. If this occurs, it is possible that other cycles crossing this node violate the legal slack and decrease the performance.
For example, consider Figure 4. Node Ack2↓ is a shared node on cycles Cyc1 and Cyc2 (where Cyc1 is [F2↑, C2↑, Ack2↓, C2↓, F2↓, Ack2↑, C1↑] and Cyc2 is [C2↑, Ack2↓, Int2↓, C2↓, Int2↑]). We assume that the value of the total cycle time of this pipeline is 4.8 and the values of cycle time of Cyc1 and Cyc2 are 4.6 and 3.7 respectively. As a result, the slack values of Cyc1 and Cyc2 become 0.2 and 0.9 respectively. Also we assume that

assigning high-Vth to the element C3 in a PCFB template increases it propagation delay from 1 to 1.5, and therefore the delay value of node Ack2↓ in Dependency Graph increases to 1.5 in the case of high-Vth assignment (C3 element in the PCFB template is associated with Ack↓ and Ack↑ nodes in the Dependency Graph model). If we consider the slack of 1.1 for node Ack2↓, then we can assign high-Vth to element C3 of the associated PCFB template. Consequently, the cycle time of Cyc2 increases to 5.1, and the critical cycle will change. Thus, the pipeline's performance will decrease.

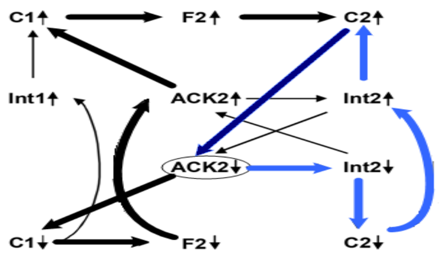


Fig. 4. Shared node in the Dependency Graph model

While a cycle has a positive slack, all of its nodes can be assigned to high-Vth one by one using a greedy algorithm. This process continues until there does not exist any node where its threshold is changed without forcing the slack value to be negative. Consequently, in cases where only one node of the cycle can be assigned to high-Vth, our algorithm discovers this node.

The sequence of selecting cycles in a pipeline and nodes in a cycle to assign high-Vth can affects the total number of high-Vth nodes. Since the number of possible sequences is $M!*N!$ (M is the number of cycles in a pipeline and N is the number of nodes in a cycle), finding optimal order for each pipeline is very time consuming.

Hence, we chose these nodes randomly. It should be noted that the number of high-Vth nodes depends on the values of high and low Vth. This fact influences our method to select optimal value for high-Vth, as explain in the following.

To assign more nodes to high-Vth, it is better to identify those nodes which are located on only one cycle, and then consider nodes that are shared in two or more cycles. This is due to the fact that a shared node reduces the slack of more than one cycle. We exploit this improvement with assigning a chosen-priority to each node.

Vth assignment algorithm for linear pipeline.

```
Assign-Vth () {

  1. Find all possible linear pipelines.
  2. Assign Low-Vth to all elements of circuit
  3. For each pipeline of circuit, DO
  {
    4. Calculate the propagation delay Tphl(x), Tplh(x) of each node x in
       Dependency Graph model
    5. Determine possible loops of pipeline
    6. For every node (x) of a cycle calculate
         $\text{Priority}_X = 1/N_{Share\,Cycles}$
       (where $N$ is the number of cycles which this node is located on
       them)
    7. Calculate $C_x$ of cycles for each pipeline.
    8. Calculate $C_{Total}$ of pipeline.
    9. Calculate $T_\delta(x)$ of cycles
    10. DO while (there is an unmarked cycle in selected cycle)
      {
        11. Select a random cycle from unmarked cycles and mark it and
            unmarked all of it nodes.
        12. DO while (there is an unmarked node in selected cycle)
          {
            13. Select a random node with highest priority and assign high-Vth
                to it
            14. Update the propagation delays of nodes T↓(x)/T↑(x) ) and
                Calculate new $C_x s$ of associated cycles
            15. Calculate new $T_\delta(x)s$ of associated cycles
            16. If (all new $T_\delta(x)s$ >0):
                  Accept high-Vth
                else:
                  Reject high Vth and Mark it
          }
      }
}
```

Following the assignment of high-Vth to a node, the network is updated, and by using eqs.1and 2, the cycle parameters will be recalculated. This procedure repeats again until no more nodes can be assigned to high-Vth. It is important to note that, the procedure of Vth-Assignment is performed for all linear-pipelines simultaneously and which makes our proposed method fast. The pseudo-code for the Vth-assignment procedure is shown in previous page.

Since the slack of each cycle is greater than or equal to 0, the total cycle time of the pipeline will not be changed. Hence, the performance of the pipeline is maintained. While the intention of our proposed algorithm is not to change the performance of any linear pipeline, it is guaranteed that the total performance of the circuit remains the same as in the case of an all low-Vth design. Results presented in the next section indicate that this method does not force any performance penalty.

5.2. *Phase B:*

Up to this point only linear pipelines are considered. However, some elements which can be assigned to high-Vth without affecting the total performance of the circuit may not be distinguished in this method. An example of such a case is an unbalanced fork-join structure shown in Figure 5. In this case, the overall delay is determined by the top branch, and the delay of other branches can be increased by the difference between their cycle time and the maximum cycle time. Whereas, using "phase A" algorithm causes the delay of all branches to remain unchanged (Elements of stages T7, T8 and T9 can be assigned to high-Vth). In the second class of deterministic pipelines (fork-join dependencies) widely unbalanced numbers of stages in fork-and-join paths may impact performance because one path may prevent data or bubble injection into the other path. We exploited this dependency to assign more nodes to high-Vth using a complementary algorithm.
The complementary algorithm receives the output of "Assign high-Vth" algorithm, and then determines all possible fork-join structures of the circuit. The next step of this algorithm is to calculate the slack of each path in a fork-join structure. Slack of a path ($\mathrm{T}_{\delta\mathrm{Path}}(\mathrm{x})$) denotes the amount by which a path can be slowed down without affecting the fork-join pipeline performance. To determine slack of a path we define delay-path as follow:

$$T_{Path_k} = \sum_{(n=1..to..n=N)} [C_{PCFB_n}] \tag{5}$$

where N is the number of stages in a path and $\mathrm{C}_{\mathrm{PCFB}_n}$ is determined by eq.2. The delay-path of Critical Path in a fork-join pipeline is determined by eq.6.

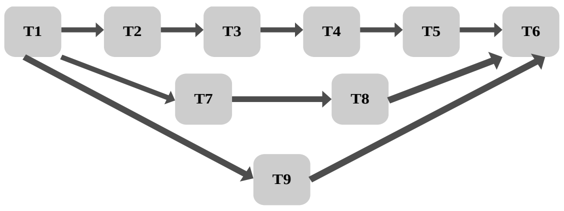


Fig. 5. Unbalanced fork–join structure: The cycle time of templates T7, T8, T9 can increase.

Complementary high-Vth assignment algorithm

```
Complementary Assignment High-Vth ()
{
    1.  Receive the output of Assign high-Vth
    2.  Find all Fork-join structures.
    3.  For each Fork-join structures, DO
    {
        4.  Calculate $T_{Path_k}$ of each path in fork-join pipeline.
        5.  Calculate $T_{Crictial}$ of fork-join pipeline.
        6.  Calculate $T_{\delta Path}$ of each path.
        7.  DO while (there is an unmarked low-Vth node in selected path)
        {
            8.  Select a random node with highest priority and assign high-Vth to it
            9.  Update the propagation delays of nodes (T↓(x)/T↑(x)) and Calculate
                new $C_x s$ of associated cycles
            10. Calculate new $T_{\delta Path}$ of selected path
            11. If ( new $T_{\delta Path}$ >0):
                    Accept high-Vth
                else:
                    Reject high Vth and Mark it
        }
    }
}
```

$$T_{Crictial} = Max\left[T_{Path_n}\right] \forall n \in fork - join\ structure \tag{6}$$

For the critical path of a fork-join pipeline which is determined by eq.7, slack is 0, and for any other cycle of the same pipeline it can be expressed by:

$$T_{\delta Path}(x) = T_{Critical} - T_{Path_n} \tag{7}$$

At this step, all elements of a path which were not assigned to high-Vth after performing "Assign high-Vth" algorithm are detected and can be assigned to a high-value using the same procedure which was specified for linear pipelines. The pseudo-code for the Complementary Assignment High-Vth procedure is shown in Figure 7. Note that, the procedure of Complementary Vth-Assignment is performed for all fork-join pipelines simultaneously.

6. Experimental Results

Table 1. Active and standby leakage power savings for dual-Vthh Asynchronous PCFB circuits.

Circuit	Leakage power (μW) in active mode			Leakage power (μW) in standby mod			Throughput (1/Cycle Time)	
	Single-VTH	Dual-VTH	% Reduction	Single-VTH	Dual-VTH	% Reductior	Single-VTH	Dual-VTH
c432	81.63	63.99	41.04	5.90	4.87	33.25	0.281	0.281
c499	103.12	83.11	36.86	7.45	6.21	31.54	0.203	0.203
c880	195.82	145.88	48.45	14.13	11.19	39.52	0.185	0.185
c1355	278.14	230.02	32.87	20.16	17.18	28.12	0.192	0.192
c7552	1791.3	1454.5	35.72	129.41	108.45	30.78	0.287	0.287
s382	92.34	70.64	44.65	6.77	5.38	39.14	0.297	0.297
s400	94.86	75.79	38.19	6.89	5.73	31.92	0.240	0.240
s420	119.78	92.35	43.51	8.64	7.04	35.15	0. 196	0. 196
c432	203.07	167.93	32.87	14.35	11.81	33.63	0.272	0.272
s641	243.76	178.18	51.11	17.68	13.53	44.65	0.270	0.270
s838	279.06	225.20	36.67	20.84	17.32	32.11	0.175	0.175
s1196	81.63	63.99	41.04	5.90	4.87	33.25	0.281	0.281

The proposed method to reduce leakage power of asynchronous circuits using dual-threshold voltage transistors has been implemented on SUN Ultra Sparc 10 workstation with 0.5 gigabyte of memory and using C programming.

We used the well-known sequential benchmark of synchronous circuits referred to as ISCAS'89, along with some other circuits, to better explore the efficiency of the proposed algorithms in practice. We developed a simple converter tool automatically translates the gate level synchronous circuit specification to its asynchronous dual-rail PCFB buffer equivalent.. An asynchronous synthesis toolset is employed to synthesis our ISCAS benchmarks and produce HSPICE Netlist. In order to simplify the analysis, technology-mapping was used to map the circuits to a library which contains NAND gates and NOR gates.

All the simulation results were obtained using HSPICE. Values of transistor parameters are extracted using BSIM3V3 model and for 0.18µ technology process as a particular instance. For the active mode and standby mode of the circuit, temperatures are assumed to be 110℃ and 25℃ respectively. The supply voltage is assumed to be 1.0V, and zero biased low threshold voltage and high threshold voltage used in our experiments are 0.2V and 0.38V respectively. Since threshold voltage increases about 0.8 mV for $1^{\circ}C$ decrease in temperature, the threshold voltages at standby mode is about 68 mV higher than the corresponding threshold voltages in the active mode.

Experimental results are presented in Table 1. Column 10 identifies the circuit throughput with single-Vth. Column 11 identifies throughput of the resulted dual-Vth circuits. As it is shown, both circuits demonstrate the same throughput. Column 3 identifies the number of gates in the circuit. Requirements of leakage energy in active mode of the operation of the circuits with single-Vth and dual-Vth realizations are shown in Column 4 and

Column 5 of Table 1 respectively. In Column 7 and Column 8 of Table 1, requirements of leakage energy in standby mode of the operation are given for the circuits with single-Vth and dual-Vth realizations. Percentage reduction in leakage energy in dual-Vth asynchronous circuits with respect to their single-Vth counterparts are shown in Column 6 and Column 9 of Table 1 in energy in dual-Vth asynchronous circuits with respect to their single-Vth counterparts are shown in Column 6 and Column 9 of Table 1 in active mode and standby mode of the operations. It is observed that, on the average, in dual-Vth asynchronous circuits by 21.3% and 17.8% the leakage power can be reduced in active and standby mode respectively.

7. Conclusion and Future works

In this paper, we introduced dual-threshold design technique for asynchronous PCFB pipelines. Formal performance analysis using the Dependency Graph method was followed by presenting an efficient algorithm for selecting and assigning a high threshold voltage in order to reduce the leakage power under performance constraints starting with a single low-Vth circuit. This study on asynchronous circuits design with dual threshold and its results demonstrate that both active and standby leakage power can be reduced by more than 50% for some of the circuits. Reduction of both active power and standby leakage power without area and delay penalty makes dual-Vth technique a good candidate for high performance asynchronous applications. Although in this paper, the Dependency Graph model is applied to PCFB templates, the approach can easily be applied to other template based asynchronous circuits.

References

1. ARM, Cambridge, U.K., “ARM offers first clockless processor core,” 2006.
2. Babak Rahbaran and Andreas Steininge,“Is Asynchronous Logic More Robust Than Synchronous Logic?,” IEEE Transactions on dependable and secure computing, vol. 6, no. 4, 2009.
3. Sangyun Kim, “Pipeline Optimization for Asynchronous circuits”, PHD Thesis, University of Southern California, August 2003.
4. M. Singh and S. M. Nowick, ”MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines,” IEEE Transactions on very large scale integration (VLSI) systems, vol. 15, no. 6, 2007.
5. Y. Monnent, M. Renaudi, and R. Leveugle, “Practical Assessment of Asynchronous Design,” Proc. 12th IEEE Int’l On-Line Testing Symp. (IOLTS), 2006.
6. Barry Pangrle, Shekhar Kapoor, “Managing leakage power at 90 nm and below,” EEdesign.com , 2004.
7. Yu Wang, Xukai Shen, Rong Luo, Huazhong Yang, “Leakage Power Reduction Through Dual Vth Assignment Considering Threshold Voltage Variation,” journal of circuits, systems, and computers (jcsc), volume: 18, issue: 7, pp. 1243-1261, 2009.
8. Amit Agarwal, Saibal Mukhopadhyay, Arijit Raychowdhury, Kaushik Roy, Chris H. Kim. “Leakage power analysis and reduction for nanoscale circuits”, IEEE Micro, vol. 19, no. 4, pp. 23-29, July-Aug. 2006.
9. J.T. Kao et al., “Dual-Threshold Voltage Techniques for Low-Power Digital Circuits”, IEEE J. Solid-State Circuits, vol. 35, no. 7, pp. 1,009-1,018, July 2000.

10. Yifang Liu , Jiang Hu, A new algorithm for simultaneous gate sizing and threshold voltage assignment, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v.29 n.2, p.223-234, February 2010 .
11. Yujia Feng , Shiyan Hu, The epsilon-approximation to discrete VT assignment for leakage power minimization, Proceedings of the 2009 International Conference on Computer-Aided Design, November 02-05, 2009, San Jose, California.
12. B. Ghavami, Hossein Pedram, "Dual-Threshold Voltage Technique for Asynchronous Pre-Charge Full Buffer Linear-Pipelines", 6th IEEE Dallas Circuits and Systems Workshop on System -on-Chip (SoC), 2007.
13. Peter A. Beerel, Recep O. Ozdag, "A Designer's Guide to Asynchronous VLSI," CAMBRIDGE Press, 2010.
14. Sangyun Kim and Peter a. Beerel. "Pipeline optimization for asynchronous circuits: complexity analysis and an efficient optimal algorithm. IEEE Trans. on computer-aided design of integrated circuits and systems, vol. 25, no. 3, march 2006.
15. Mohsen Raji, B. Ghavami, H. Pedram, H. Zarandi, Process Variation Aware Performance Analysis of Asynchronous Circuits, Microelectronics Journal, Elsevier 2010.
16. A. Kondratyev, M Kishinevsky, A. Taubin, J. Cortadella, L. Lavagno, "The use of Petri Nets for the Design and Verification of Asynchronous circuits and systems," journal of circuits, systems, and computers (jcsc), vol. 8, issue: 1, pp. 67-118, 1991
17. Eslam Yahya, M.Renaudin, "QDI Latches Characteristics and Asynchronous Linear-Pipeline Performance Analysis", Research Report, TIMA-RR--06/-01—FR, 2006.
18. C G. Wong and Alain J. Martin, "High-Level Synthesis of Asynchronous Systems by Data Driven Decomposition", Proc. Of 40th DAC, Anneheim, CA, USA, June 2003.
19. Jan M. Rabaey, "Digital Integrated Circuits", New Jersey, Prentice-Hall, 1996.
20. B.J. Sheu, D.L. Scharfetter, P.K. Ko, and M.C. Teng, "BSIM: Berkeley Short-Channel IGFET Model for MOS Transistors", IEEE J. Solid-State Circuits, SC-22, No.4, pp. 558-566, 1987.
21. Hongbo Liu, Jiaxin Wang, "A new way to enumerate cycles in graph", Proceedings of the Advanced International Conference on Telecommunications and International Conference on Internet and Web Applications and Services (AICT/ICIW), 2006.