Low Power Design Using Dual Threshold Voltage

Yen-Te Ho and Ting-Ting Hwang
Department of Computer Science
National Tsing Hua University HsinChu, Taiwan 300

Abstract

In this paper, we will study the reduction of static power consumption by dual threshold voltage assignment. Our goal is, under given timing constraint, to select a maximum number of gates working at high-Vth such that the total power gain is maximized. We propose an maximum independent set based slack assignment algorithm to select gates for high-Vth. The results show that our assignment algorithm can achieve about 68% improvement as compared to results without using dual Vth.

1 Introduction

Several techniques to reduce subthreshold leakage current by multi-threshold voltage were reviewed in [1]. It includes placing a high-Vth sleep transistor between Gnd (VDD) and fast low-Vth CMOS logic [2], variable threshold CMOS that relies on a triple well process [1], and multiple threshold voltage fabrication process option [3, 4, 5].

In dual Vth fabrication process, designers are provided with transistors that are either high threshold voltage (slow but low leakage) and low threshold voltage (fast but high leakage) by only an extra mask layer. A straightforward way to take advantage of process option is to use low-Vth transistors for gates on the critical-path and high-Vth transistors on the non-critical-path. This strategy is used in [3, 4, 5] and has shown to gain significant saving of leakage power and does not alleviate the performance of circuits. In [3], a simple backward Breadth-First-Search (BFS) strategy is employed to identify a subset of gates that can be assigned to high-Vth. In [4], they formulate this problem to a Specific Delay Fictitious Buffers (SDF-Displacement) problem and use Integer Linear Programming (ILP) technique to solve this problem. As compared to [3], it can save more power consumption. However, ILP can not solve large size problem, and thus this technique is not practical for large size circuit. The technique proposed by [5] is similar to [3], but it assigned high-Vth to all gates of the circuit first in order to get minimum leakage power dissipation. After that, it selects gates that are on paths violating timing constraint and assigns low threshold voltage to them. The selection method is similar to Depth-First-Search (DFS). Compared to [3], it can select more gates assigned to high-Vth due to its selection sequence.

It is clear that the above gate selection algorithms are either straightforward or time consuming. In order to select a maximum number of gates working at high-Vth such that the total power gain is maximized, we propose a maximum independent set based dual-Vth assignment algorithm. The problem is formulated as under a given timing constraint, select low-Vth transistors for gates on the critical-path and high-Vth transistors on the non-critical-path such that the leakage power is minimized.

2 Design Flow for Dual Threshold Voltage

At the beginning of the flow, an input design is synthesized using only low-Vth library by Design Compiler. Then this synthesized gate-level verilog code is placed and routed using Apollo. After finishing placement and routing, we output the timing information file, called SDF. In this file, we can get the accurate wire delay and cell delay. With this timing information, we calculate the arrival/required time of the circuit and analyze the timing slack. The node with zero slack is marked as critical-path. Then, we select nodes that are off the critical-path to work at high-Vth under timing constraint. We propose an algorithm, *Dual Vth Assignment*, to select cells for replacement. The detailed algorithm will be described in Section 3. After selecting the suitable cells to work at high-Vth, we get a modified gate-level verilog code with both high-Vth and low-Vth technology. Then we instruct Apollo to perform ECO placement with this code. The ECO placement is an incremental placement tool that will hold the unchanged cells and replace the altered cell, and then update the timing information. After the ECO placement step, the final design will be output.

3 Dual Threshold Voltage Assignment Algorithm

In this section, we will describe our dual-Vth assignment algorithm. Our algorithm is inspired by the maximum independent set based slack assignment (MISA) proposed in [6]. Under a given timing constraint, MISA algorithm will allocate allowable additional delay to nodes such that the total delay assinged to nodes is maximum. First, MISA algorithm will be reviewed. Then, our MISA-based dual-Vth assignment algorithm will be presented.

3.1 Review of MISA Algorithm

A circuit network can be represented as a directed graph $G = (V, E)$. A node $v \in V$ corresponds to a gate in the

network. A directed edge $(u,v)\in E$ represents that node u is an immediate fanin of node v, and node v is an immediate fanout of node u. The set of all immediate fanins (fanouts) of node v is denoted by $FI(v)$ ($FO(v)$). If there is a directed path from node u to node v in G, u (v) is said to be a transitive fanin (fanout) of v (u). The set of all transitive fanins (fanouts) of node v is denoted by $TFI(v)(TFO(v))$. Each node $v\in V$ is associated with a cell delay $D(v)$ and each edge from u to v is also associated with an interconnection delay $d(u,v)$. The arrival time $AT(v)$ and the required time $RT(v)$ of node v are recursively computed by Equation 1:

$$\begin{cases} AT(v)=max_{u\in FI(v)}(AT(u)+d(u,v)+D(v)) \\ RT(v)=min_{w\in FO(v)}(RT(w)-d(v,w)-D(w)) \end{cases} \quad (1)$$

The arrival time of primary inputs are set zero, and the required time of primary outputs are set to the timing constraints. The slack (SL) between arrival time and required time of node v is defined to be

$$SL(v) = RT(v) - AT(v) \quad (2)$$

Before we describe the details of the algorithm of MISA, we review some definition and Lemma in [6].

Definition 3.1 *Given a graph $G = (V, E)$, an edge $(u,v)\in E$ is called sensitive if either $AT(u) + d(u,v) + D(v) = AT(v)$ or $RT(u) + d(u,v) + D(v) = RT(v)$. A sensitive edge $(u,v)\in E$ implies that the slack of u and v is sensitive to each others' delay change. A directed path is called sensitive if the path consists of only sensitive edges. Two nodes $u, v\in V$ are called slack sensitive if there exists a sensitive path from u to v or from v to u in G. Otherwise, they are called slack insensitive.*

This definition shows the relationship of any two nodes in the circuit. The relationship gives how the timing information is changed when a small positive amount delay is added to a node in the circuit. Suppose the delay of node v is increased by a small positive amount $\varepsilon > 0$, then its slack is reduced by exactly ε. However, the slack of any node $x\notin TFO(v)\cup TFI(v)$ does not change. Instead, the slack for any node $u\in TFO(v)\cup TFI(v)$ may or may not change, depending on its slack $SL(u)$ and the slack sensitivity of u and v. There are three cases in Lemma 3.1.

Lemma 3.1 *Let u be a transitive fanin/fanout node of any node $v\in V$, and ε be a small positive constant added to v. We have the following cases.*
Case(a) If $SL(u)\geq SL(v)$, then $SL(u)$ will decrease exactly ε as long as u and v are slack sensitive.
Case(b) If $SL(v) - \varepsilon < SL(u) < SL(v)$, then $SL(u)$ will decrease by $SL(u) - SL(v) + \varepsilon$ as long as u and v are slack sensitive.
Case(c) If either $SL(u)\leq SL(v) - \varepsilon$, or u, v are slack insensitive, then the $SL(u)$ remain unchanged.

To make full use of timing slack, the goal is to keep the minimal number of nodes whose slack will be reduced by the increased delay of the selected node. In this sense, Case (c) is the best case.

Hence, the MISA algorithm proceeds as follows. First, an initial circuit G is divided into several sensitive transitive closure graph $G_s = (V, E_s)$, where there is an edge $(u,v) \in E_s$ if there is a sensitive path from u to v. Next, the algorithm selects set of nodes V_{max} with maximum slack in G and construct an induced subgraph, $G_{max} = (V_{max}, E_{max})$, of G_s, such that there is an edge $(u,v)\in E_{max}$ if (u,v) is in G_s. Then, Maximal Independent Set (MIS) algorithm to find nodes in G_{max} is called. For node v in MIS, the additional delay, $\triangle d(v)$ is increased by ε. Note that the small constant ε added in each step should be smaller or equal to $SL_{max} - SL_{max-1}$. In this case, the minimal number of nodes in the circuit are affected. Finally, the slack of node in G is updated. This process repeats until $\varepsilon = 0$ (i.e., $G_{max} = \emptyset$).

3.2 MISA-based Dual-Vth Assignment Algorithm

First, some terms are defined. A node v is called a $feasible$ node if $SL(v)\geq\triangle d(v)$, where $\triangle d(v)$ is the additional delay allocated to node v.

It is assumed that initially each node $v\in V$ works at low-Vth and its slack $SL(v)\geq 0$. Under the given timing constraint, any $feasible$ node may be selected to work at high-Vth for leakage power reduction. In general, however, not all $feasible$ nodes in G can work at high-Vth. The reason is that, once a node v is selected to work at high-Vth, the slack of node $u, u\in TFI(v)\cup TFO(v)$ may be reduced. As a result, some $feasible$ nodes may no longer be feasible. So, for any given circuit, our goal is, under given timing constraint, to select a maximum number of gates working at high-Vth such that the total power gain is maximized. Since MISA algorithm will allocate additional delays to nodes such that the total delay assigned to nodes is maximum, we will select nodes for high-Vth by MISA-based algorithm.

The pseudo-code of our single execution of dual-Vth assignment algorithm is shown in Figure 1. Given the arrival time (AT), required time (RT), and slack (SL) for each node, first, we construct a $SList(G)$ by $SL(v), v\in V$ in decreasing order. That is, $SList(G)$ is a list of nodes that sorted in decreasing order by their slack. Note that for each node we compute the delay difference of the node working at low-Vth and high-Vth, $DiffD(v)$. If the slack is less than the difference, we will not put it into the candidate list, $SList$, for selection. The reason is that, in this situation, it is impossible to swap this node from low-Vth to high-Vth even if all slack is allocated to the node. Before we proceed the algorithm, we set the current allocated delay, $AllD(v)$ to be zero, for all nodes in $SList$.

According to Lemma 3.1 Case (c), we first select the nodes which have the maximum slack and construct a graph G_{max}. Then, we will find the maximum independent set in G_{max}. Since the maximum independent set problem is NP-complete on general graph. We propose a heuristic to solve it.

First, we assign a weight to each node in the graph and then find the maximum weight independent set in the graph. The weight of a node v, $W(v)$, is defined to reflect the gain of selecting a node to work at high-Vth:

$$W(v) = \alpha * PW(v) + \beta * UW(v) + \gamma * LW(v, dist) \quad (3)$$

The first term, PW, is defined to represent the effective power saving when $\varepsilon\varepsilon$ slack is used. It is:

$$PW(v) = \frac{\triangle P(v)}{\varepsilon\varepsilon \times (1 + Adj(v))} \quad (4)$$

```
1 Algorithm SE-DVA ()
2 Input : Graph G = (V, E)
3 Output : Set of high-Vth gates, HList
4
5   For each node v, v∈V
6     If SL(v)≥DiffD(v) {
7       Add node v to SList by its SL(v);
8       AllD(v) = 0;
9     }
10  Construct G_max of G_s;
11  ε = SL_max − SL_max−1
12  While (ε≠0 and SList≠∅) {
13    /* Find MWIS in G_max */
14    Compute W(v) for each node v in G_max
15    While G_max≠∅ {
16      Select node v with maximum W(v) to MWIS;
17      Remove v and its neighbors in G_max
18    }
19    For all node v in MWIS do {
20      εε = min(ε, DiffD(v) − AllD(v)) ;
21      AllD(v) = AllD(v) + εε ;
22      If (AllD(v) = DiffD(v)) then
23        Move node v from SList to HList;
24    }
25    Update AT/RT/SL/ε for the graph G;
26  } }
```

Figure 1: Single Execution of Dual-Vth Assignment

where $\triangle P(v)$ is the power saving of node v when node v works at high-Vth, $\varepsilon\varepsilon$ is $min(\varepsilon, (DiffD(v) - AllD(v))$, and $Adj(v)$ denotes the number adjacent nodes to v in G_{max}. Note that $\varepsilon = SL_{max} - SL_{max-1}$.

The second term, $UW(v)$, in (3) is used to reflect how urgent of node v to be selected.

$$UW(v) = \frac{AllD(v)}{DiffD(v)}. \quad (5)$$

In this equation, $AllD(v), DiffD(v)$ are the additional delay allocated to node v and the delay difference of v when working at high-Vth and low-Vth, respectively. The closer the delay for v to work at high Vth, the urgent the node will be selected.

The third term, $LW(v, dist)$, in (3) is used to reflect the location information of the circuit. Since our proposed algorithm is called after the circuit has been placed. We can take cell location in the floorplan into account. When we select nodes to work at high-Vth, we can assign the nodes which have the most neighbors working at high-Vth a higher weight. This will bring nodes with high-Vth or low-Vth clustered in a region so as to have higher yield during fabrication process. $LW(v, dist)$ is defined as follows.

$$LW(v, dist) = hnode \quad (6)$$

where *hnode* is the number of nodes working at high-Vth that are within distance of *dist* from v. *dist* is a user defined parameter. α, β, γ in (3) are the parameter to control the importance of the three terms.

After the weight of node in G_{max} is computed, the following heuristic is used to find the maximum weight independent set (MWIS). The heuristic begins with selecting the node v with maximum $W(v)$. Then node v is added to MWIS and v and all its neighbors are deleted from G_{max}. This process repeats until $G_{max} = \phi$. After finding the MWIS, $\varepsilon = SL_{max} - SL_{max-1}$ is computed, and a small delay of constant $\varepsilon\varepsilon = min(\varepsilon, (DiffD(v) - AllD(v)))$ is added to $AllD(v)$ for each node v in MWIS. If the delay added to node v is enough for it to work at high-Vth (i.e., $AllD(v) = DiffD(v)$), we will select this node to work at high-Vth and add this node to $HList$, where $HList$ is a list that store selected nodes that can work at high-Vth. If the current allocated delay of the selected node is not high enough to work at high-Vth, the node will remain in the list of $SList$. Finally, the timing information is updated for each sensitive nodes. This procedure repeats until $SList = \phi$ or $SL(v) = 0$, for all nodes in $SList$.

```
1 Algorithm Dual-Vth Assignment ()
2   Input : low-Vth, high-Vth Library, circuit
3   Output: set of nodes working at high-Vth
4
5   Read input file and create directed graph;
6   For each node, compute its AT, RT, SL ;
7   While not finish {
8     new_swap_number = SE-DVA algorithm();
9     If new_swap_number == 0 then
10      Finish
11    Else {
12      For each node v, v∈SList
13        AllD(v) = 0;
14      Update AT, RT, SL;
15    }
16  }
17  Output result;
```

Figure 2: Dual-Vth Assignment Algorithm

After finishing single execution of SE-DVA algorithm, we find that there exits some nodes whose allocated delay is higher than it needs to work at low-Vth but lower than the delay it needs to work at high-Vth. In order to fully utilize the slack assigned to these nodes to further reduce leakage power dissipation, we propose a loop to perform SE-DVA algorithm iteratively. After performing one iteration of algorithm, we reset all nodes v in $SList$ to the delay at low-Vth (i.e., $AllD(v) = 0$), recalculate the timing information, and repeat the iteration. We iterate the loop until no more node can be selected to work at high-Vth. The procedure is shown in Figure 2

4 Experimental Results

To demonstrate the effectiveness of our algorithm, 7 designs were selected. The descriptions of these 7 designs are shown in Table 1. The columns labeled **CN**, and **Characteristics** are the cell number of the design, and the characteristics of the design, respectively.

The first experiment is to understand the improvement of the iterative execution of the dual-Vth assignment algorithm as compared to single execution of the algorithm. The result is shown in Table 2. The column labeled **SE** is the number of cells working at high-Vth after single exe-

Table 1: Circuit Descriptions

Cir.	CN	Characteristics
TOP	455	An Alarm Clock
MAC	2601	Multiplier and Accumulator
AVG	8982	Average Number Calculator
GCC	9564	Gravity Center Calculator
CCU	16577	An ARM CPU
ACP	13573	Asymmetric Crypto-Processor
AES	15804	Advanced Encryption Core

Table 2: Swap Ratio of Single and Iterative Execution

Cir.	SE	IE	R
TOP	237	244	2.9%
MAC	1588	1721	8.3%
AVG	7706	7834	1.6%
GCC	5305	5971	12.5%
CCU	16084	16114	0.2%
ACP	8436	9524	12.8%
AES	15559	15577	0.1%
average			5.5%

cution of the dual-Vth assignment algorithm, the column labeled **IE** is the number of cells working at high-Vth after iterative execution of the algorithm, and the column labeled **R** is the ratio of improved percentage of cells working at high-Vth and is calculated as $\frac{IE}{SE}$. The results show that the iterative execution of algorithm can take full use of the slack as compared to the single execution one.

The second experiment is to compare the design flow proposed by us and a commercial flow. First, we want to create a dual-Vth library by ourselves because we are not able to access a library with dual threshold voltages. By examining dual-Vth library of .13 technology by TSMC, we found that for each gate, its high-Vth and low-Vth cell under .13 technology have the same layout information but different leakage power consumption and delay. We can use this characteristics to create our dual-Vth cells under .25 library. We set the original TSMC .25 library as low-Vth library. To create a high-Vth library, for each cell in the low-Vth library, we create a corresponding high-Vth cell. The area of the cell is set the same. The leakage power consumption is set 0.006 $\times$ (the leakage power consumption of its corresponding low-Vth cell), and the delay is set 1.6 $\times$ (the delay of its corresponding low-Vth cell).

The timing constraint of a circuit is set by the following procedure. First, we synthesize and map circuit to low-Vth library by Design Compiler. We set the timing constraint to zero to get the best timing performance. This best delay is the timing constraint of the design.

The experiment procedure of using the commercial design flow is as follows. First, we synthesize and map circuit to high-Vth library by Design Compiler and set the timing constraint. After synthesized, the Design Compiler will output a technology mapped gate-level circuit. Then we input this gate-level circuit to Design Compiler and add low-Vth library to cell library. After the second iteration of executing Design Compiler, Design Compiler will output a new technology mapped gate-level circuit with dual-Vth cells. Using this gate-level circuit, we instruct Apollo to do placement and routing. If there have any timing violation, Apollo will fix them and then output the final gate-level circuit. At last, Design Compiler is used to calculate the leakage power consumption. The experiment procedure of using our design flow follows the design flow described in Section 2.

Table 3: Results of Our Flow V.S. Commercial Flow

	Original		Our Flow			Commercial Flow		
Cir.	D	P	P	R	T	P	R	T
TOP	3	35	18	51%	1s	21	61%	10m
MAC	5	274	161	58%	7s	209	76%	1h
AVG	50	655	152	23%	75s	235	35%	5h
GCC	50	851	369	42%	300s	403	47%	12h
CCU	20	1421	62	4%	208s	336	23%	8h
ACP	5	1915	889	46%	680s	1532	79%	5h
AES	5	1771	25	1%	533s	988	55%	3h
avg.				32%			54%	

The comparison results are shown in Table 3. The column labeled **Original** is the results of the circuit with low-Vth cells mapped by Design Compiler. The column labeled **Our Flow** is the results of the circuit with dual-Vth cells mapped using our design flow. The column labeled **Commercial Flow** is the results of the circuit with dual-Vth cells mapped using the commercial design flow. The column labeled **D** is the timing constraint of the circuit in nano-second (ns). The column labeled **P** is the leakage power consumption in nano-watt (nW) of the circuit calculated by Design Compiler. The column labeled **R** is the ratio of the leakage power reduction by using dual-Vth to the original circuit and it is calculated as $\frac{dual-Vth\ power}{original\ power}$. The column labeled **T** is the CPU execution time in second (s), minute (m) or hour (h) of assigning cells to work at high-Vth. The results show that in average, our algorithm can finish the dual-Vth assignment procedure in a few minutes. But, the Design Compiler takes several hours to do the same work. The average power saving of our dual-Vth assignment algorithm can save 68% as compared to the original design, while the Design Compiler can only save 46% as compared to the original design.

References

[1] J. Kao, S. Narendra and A. Chandrakasan "Subthreshold Leakage Modeling and Reduction Techniques," *Proceddings of ICCAD,* pp. 141-148, 2002.

[2] S. Mutoh, et al., "1V Power Supply High-Speed Digital Circuits Technology with Multithreshold-voltage CMOS", *IEEE Journal of Solid State Circuits,* vol. 30, pp. 847-854, 1995.

[3] Liqiong Wei, Zhanping Chen, Mark Johnson and Kaushik Roy, "Design and Optimization of Low Voltage High Performance Dual Threshold CMOS Circuits", *Proceedings of DAC,* pp. 489-494, 1998.

[4] Vijay Sundararajan and Keshab K. Parhi, "Low Power Synthesis of Dual Threshold Voltage CMOS VLSI Circuits", *Proceedings 1999 International Symposium on Low Power Electronics and Design,* pp. 139-144, 1999.

[5] Nikhil Tripathi, Amit Bhosle, Debasis Samanta and Ajit Pal, "Optimal Assignment of High Threshold Voltage for Synthesizing Dual Threshold CMOS Circuits", *The 14th International Conference on VLSI Design,* pp. 227-232, 2001.

[6] Chuhong Chen, X. Yang and Majid Sarrafzadeh, "Potential Slack:An Effective Metric of Combinational Circuit Performance," *Proceedings of ICCAD,* pp.198-201, 2000.