Behavioral Level Dual-$V_{th}$ Design for Reduced Leakage Power with Thermal Awareness

Junbo Yu, Qiang Zhou, Gang Qu*, and Jinian Bian

Tsinghua National Laboratory for Information Science and Technology, Dept. of Computer Science
Tsinghua University
Beijing, 10084, China
yujb05@mails.thu.edu.cn, zhouqiang@tsinghua.edu.cn

*Dept. of Electrical and Computer Engineering and
Institute for Advanced Computer Studies
University of Maryland
College Park, USA
gangqu@umd.edu

Abstract—Dual-$V_{th}$ design is an effective leakage power reduction technique at behavioral synthesis level. It allows designers to replace modules on non-critical path with the high-$V_{th}$ implementation. However, the existing constructive algorithms fail to find the optimal solution due to the complexity of the problem and do not consider the on-chip temperature variation. In this paper, we propose a two-stage thermal-dependent leakage power minimization algorithm by using dual-$V_{th}$ library during behavioral synthesis. In the first stage, we quantitatively evaluate the timing impact on other modules caused by replacing certain modules with high $V_{th}$. Based on this analysis and the characteristics of the dual-$V_{th}$ module library, we generate a small set of candidate solutions for the module replacement. Then in the second stage, we obtain the on-chip thermal information from thermal-aware floorplanning and thermal analysis to select the final solution from the candidate set. Experimental results show an average of 17.8% saving in leakage power consumption and a slightly shorter runtime compared to the best known work. In most cases, our algorithm can actually find the optimal solutions obtained from a complete solution space exploration.

Leakage Power, Behavioral Synthesis, Dual-$V_{th}$, Thermal-aware

I. INTRODUCTION

With the widely use of mobile device and the emergence of energy crisis, power consumption plays an increasingly important role in modern integrated circuits (ICs) design. Power consumption consists of dynamic power and leakage power. In nanometer technology generation, supply voltage levels are much lower than before, which results in much lower threshold voltage levels so as to maintain low delay. Leakage power increases exponentially with decreasing threshold voltage levels. Therefore at 90nm and below, leakage power accounts for a significant portion of the total power [1]. We focus on the reduction of sub-threshold leakage as the other main source of leak, gate leakage, can be restrained by high-k material [2].

One of the most effective design techniques for reducing leakage is dual-$V_{th}$ design [3], where performance-critical transistors are made of low-$V_{th}$ to provide the required performance and high-$V_{th}$ transistors are used everywhere else to reduce leakage. Dual-$V_{th}$ assignment or allocation can be applied to all phases of the design flow. Although transistor level dual-$V_{th}$ allocation is the most effective for leakage reduction, it is also the most challenging due to the complexity of dealing with the billions of transistors in modern ICs. Thus it has been proposed [4][5] to allocate $V_{th}$ at behavioral level, where the solution space is much smaller than that at the transistor level. At behavioral level, dual-$V_{th}$ allocation can be converted to the module selection problem. The modules on non-critical path are selected to be replaced with high-$V_{th}$ implementation.

Due to factors like module sharing in behavioral level, it is difficult to model the timing relationship of modules precisely. And consequently, the optimal module selection will be hard to obtain. A complete search might still be prohibited due to time complexity.

In addition to threshold voltage, leakage power also depends on the IC’s dynamically changing operating environment, particularly the working temperature where raising temperature can cause exponential increase of leakage [6]. Given the increasing on-chip temperature variation, it becomes important to consider the temperature factor when estimating and minimizing leakage power. However, it often requires detailed floorplanning information and the time consuming thermal analysis tools to obtain on-chip temperature. Finally, the interdependency between temperature and power makes it even more challenging to take temperature into consideration for leakage minimization.

In this paper we propose a two-stage approach to the module selection problem. In the first stage, we quantitatively evaluate the timing impact of replacing modules by their high $V_{th}$ implementation. Based on this timing impact and the dual-$V_{th}$ module library, a small set of candidate solutions will be generated. In the second stage, we select the best solution from this candidate set based on the thermal information we obtain from thermal-aware floorplanning and thermal analysis.

The rest of this paper is organized as follows. In Section 2, we briefly survey the related work. Section 3 describes preliminaries and problem formulation. Section 4 presents a motivating example. In Section 5, we propose the two-stage approach for module selection. Section 6 reports the experimental results and we conclude in Section 7.

II. RELATED WORK

There are many leakage power reduction techniques at various design levels. At the gate level, Srivastava et al. considered simultaneous $V_{nn}/V_{bb}$ assignment to optimize total power [8]. Simultaneous input vector control (IVC) and gate replacement were proposed to reduce leakage power in reference [9]. Mogal and Bazargan [10] proposed to replicate a hot module during floorplanning so as to actively migrate its computation, reducing the on-chip temperature and thereby the thermal-dependent leakage.

In behavioral level, researchers have also developed many leakage power reduction techniques. Ni and Memik [11] identified the number of resources and reduced the on-chip temperature in order to reduce thermal-induce leakage power. But their design flow did not include floorplanning and could not obtain accurate thermal profile. Khouri and Jha [4] performed high-level synthesis with a dual-$V_{th}$ library for leakage power reduction using a simple heuristic as a guide. In each iteration, the algorithm identifies a single candidate module based on the fixed priority for reducing leakage power. If the timing constraint is satisfied after replacement, the
selected module will be replaced. It is a greedy approach. Also with dual-Vth library, Tang et al. [5] presented a constructive algorithm for leakage power optimization based on maximum weight independent set (MWIS) problem. Primitively, the model MWIS is introduced for dynamic power reduction with multiple implementations in gate level [12]. Module sharing at gate level does not exist, however, it does exist at behavioral level. When using MWIS model to select replaced modules in behavioral level, it will amount to magnify the timing relationship among modules. This reduces the solution space considerably and sometimes misses the solution with low leakage power. Moreover this work doesn’t consider the temperature of module while selecting the replaced modules and analyzing leakage power.

In our approach, we quantitatively evaluate the timing relationship among modules. Based on the exact timing relationship, we explore a small part of the solution space and generate a set of candidate solutions. With thermal-aware floorplanning and thermal analysis, we obtain thermal information. Then we select the best solution from the candidate set for low thermal-dependent leakage power based on thermal information.

III. PRELIMINARIES AND PROBLEM FORMULATION

A. Leakage Power Model

In this subsection, we introduce some basic concepts for leakage power consumption. The leakage power model is based on the Berkeley short-cannel IGFET model BSIM [13]. When $V_{gs}=0$, the leakage current of an n-type MOSFET transistor is:

$$I_{leak} = \mu_0 C_{ox} \frac{W}{L} V_{T} e^{\mu_0 V_{gs} + \frac{V_{gs} - V_{th}}{V_{T}}} \left( 1 - e^{-\frac{V_{th}}{V_{T}}} \right)$$

(1)

where $V_{th}$ is the threshold voltage and $V_T$ is the thermal voltage [13]. From Equation (1), we can see that the leakage current decreases exponentially with the increase of the threshold voltage. Thus low-Vth module on the non-critical path can be replaced with high-Vth module in order to reduce leakage power consumption as long as the timing constraint is satisfied.

Leakage current also has a strong dependency on temperature and can be modeled as [20]

$$I_{leak}(T) = I_0 (c_2 (T - T_0)^2 + c_1 (T - T_0) + 1)$$

(2)

where $I_0$ is the leakage current at the nominal temperature $T_0$, $c_1$ and $c_2$ are technology dependent constants that are normally obtained empirically.

On the other hand, leakage current and leakage power will impact the temperature (see a model in [21]). Therefore when we compute a module’s leakage changes, we should consider both temperature’s impact on leakage, which might be large, and leakage’s impact on temperature, which is normally small. The leakage is obtained after several iterations when temperature and leakage converge. In general, replacing a module at high temperature region by its high-Vth implementation gives more leakage saving than replacing the same module at low temperature area. This is one of the main motivations of our work.

B. Data Flow Graph and Composite Constraint Graph

The input of behavioral synthesis can be represented by a data flow graph (DFG), which is a directed acyclic graph $D=(V, E)$. Each node $v \in V$ represents an operation and each directed edge represents a data dependency. A simple DFG is shown in Figure 1(a), where there are 4 operations (3 addition and 1 multiplication) and 2 data dependencies (A and B need to completed before D and C, respectively).

In behavioral synthesis, each node in the DFG will be bound to a module such as adder, multiplier and so on. These modules are usually shared by many operations. For example, the module ADD is used to perform operations A, B, and D as shown in Figure 1(c). The Composite Constraint Graph (CCG) (V, Ec) is introduced in [5] to capture this module sharing constraint. In Figure 1(b), the two dashed edges have been added to the original DFG, indicating the order of execution of operations A, B, and D on the same module ADD.

We denote the delay at the node $v$ by $d(v)$, which is the execution time of $v$ on its assigned module $m$. For scheduling, it is normally helpful to compute the earliest and the latest possible starting time for a node $v$, denoted by $t_{ASAP}(v)$ and $t_{ALAP}(v)$. The slack of node $v$ is defined as $s(v) = t_{ALAP}(v) - t_{ASAP}(v)$. For each module $m$, its slack $S(m)$ is defined as

$$S(m) = \min_{v \in m} s(v)$$

(3)

where $v$ is a node in DFG that will be executed on module $m$. This value indicates the maximum delay increase we can have while meeting the latency constraint after replacing a low-Vth module by a high-Vth module implementation satisfies the latency constraint, we only need to ensure that the slack of nodes in the CCG are non-negative. In the synthesis result after replacement the schedule may be different from the initial result, but the bindings of operations to modules remain the same. With floorplanning and thermal analysis, the location and the temperature of each module can be obtained. It makes possible for our proposed two-stage temperature aware module selection for replacement.

Our dual-Vth module library contains low-Vth and high-Vth implementation of each module. The high-Vth ones have lower leakage power but longer delay. If we replace a module with its high-Vth implementation, the leakage power reduction can be calculated based on the following formula used in [5], where we have considered explicitly the temperature’s impact on leakage

$$LP_{low}(T) \times (t_{idle-1} + \alpha_i \times t_{act-1}) - LP_{high}(T) \times (t_{idle-h} + \alpha_h \times t_{act-h})$$

(4)

where $LP_{low}(T)$ and $LP_{high}(T)$ are the leakage power consumption in one unit time at temperature $T$ of the low-Vth module and the high-Vth module; $t_{idle}$ is the idle time of the module during the execution period; $t_{act}$ is the module’s active time; and $\alpha$ is the coefficient measuring the ratio of the module’s leakage at active mode and idle mode. Initial temperature is obtained after floorplanning and thermal analysis. Since there is bidirectional relationship between temperature and leakage power, leakage power can be obtained after several iteration of calculation based on Equation (2) and the model used in [21].
Based on the above description, this leakage power optimization problem is defined as follows:

Given a behavioral synthesis result, a floorplan and a dual-Vth library, replace some modules with high-Vth implementation, such that the total leakage power consumption is minimized under the given area and latency constraint.

The formulation of this problem is similar to that in [5] except that we have the floorplan information to take temperature into consideration. The solution to the problem will be a set of modules that can be replaced by their high-Vth implementations. The quality of a solution, which we will refer to as weight reduction, is the total amount of leakage saving by each module replaced computed based on Equation (4). We assign the same weight to the modules of the same type in order to help our approach find high quality solution. Weight reduction will be the objective for the first stage of our approach.

IV. MOTIVATING EXAMPLE

In this section we use an example to demonstrate that leakage power can be further reduction over the approach in [5].

Benchmark: cosine2
Operation Number: 82   Latency constraint: 20 control steps
Module usage: 16  (5 MULs and 3 EXPs included)

<table>
<thead>
<tr>
<th>Module Instance</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>102</td>
</tr>
<tr>
<td>10</td>
<td>80</td>
</tr>
</tbody>
</table>

Fig. 2. The benchmark cosine2 and its MISG[5].

From CCG we know that there are both data dependency and module sharing dependency among nodes. After the replacement of each module, the slack needs to be updated to ensure that the latency constraint will still be satisfied. Therefore the sequence of module replacement will impact the leakage power reduction. In this section, we propose a metric for determining the sequence of replacement, specifically which one of any two modules should be selected. We call the selected module as the better one. Our metric uses three levels of priority to achieve this.

In the first level, if two modules have different weight, we select the one with larger weight. When a module is replaced, we can compute its weight by Equation (4). Weights of different types are normally considerably different and we can sort them in the decreasing order to obtain a sorted list $Lt$. Large modules such as EXP (exponential operator), and MUL (multiplication) have much heavier weight than ADD (adder) and SUB (substracter). They will play a more important role and deserve more consideration in leakage optimization. We refer to them as impact considered (IMC) type. We also define the referred set $R$ as the set of modules whose slack reduction should be considered after a replacement.

When two modules have a tie according to the first level metric, we use the following second level metric: the module that has less impact on the other module’s slack will be selected. If the replacement of one module reduces less slacks on other modules of IMC type in $R$, more flexibility will be left for future module replacement. Compared to MISG, we give a quantitative evaluation of slack impact, particularly to modules of IMC type in referred set $R$. We propose a value $V_m(p,R)$ to represent the slack impact on all the modules of IMC type $p$ in referred set $R$ by the replacement of module $m$.

$$V_m(p,R) = \sum_{a \in R} (\text{Stack}_{\text{in}}(a) - \text{Stack}_{\text{out}}(a))$$

Based on the DAC05 approach, after nodes 16 and 14 are replaced by high Vth implementation and included in MWIS, there are still many modules of type CanR as shown in the first row of Table 1. However, DAC05 approach will then select the best available node 15 and forms the MWIS {14,15,16}. The delay penalty for changing nodes in MWIS to high-Vth makes it impossible to replace any other nodes by their high Vth implementation. That is, there will not be any module of CanR type.

On the other hand, our approach will not select the best module 15. Instead, we choose module 12 which still leaves a non-empty set of CanR modules and leads us to the global optimal solution for MWIS {16,14,12,13,9}. As shown in Table 2, this optimal solution consumes 32.6% less leakage power and results in 6.2°C lower peak temperature than the solution {14,15,16} reported in [5]. This is mainly because that the DAC05 approach ignores the slack impact to other CanR modules and does not consider temperature information when selecting modules for replacement.

Table 2. Results obtained by three approaches

<table>
<thead>
<tr>
<th>Modules to be replaced</th>
<th>Leakage Power</th>
<th>Peak Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Replacement (NR)</td>
<td>{ }</td>
<td>8.9</td>
</tr>
<tr>
<td>DAC05 Approach [5]</td>
<td>{16,15,14}</td>
<td>5.3</td>
</tr>
<tr>
<td>Our approach</td>
<td>{16,14,12,13,9}</td>
<td>3.6</td>
</tr>
<tr>
<td>Imp. vs NR</td>
<td></td>
<td>59.6%</td>
</tr>
<tr>
<td>Imp. vs DAC05</td>
<td></td>
<td>32.6%</td>
</tr>
</tbody>
</table>

V. A TWO-STAGE APPROACH FOR MODULE SELECTION

A. The Metric for Module Selection

From CCG we know that there are both data dependency and module sharing dependency among nodes. After the replacement of each module, the slack needs to be updated to ensure that the latency constraint will still be satisfied. Therefore the sequence of module replacement will impact the leakage power reduction. In this section, we propose a metric for determining the sequence of replacement, specifically which one of any two modules should be selected. We call the selected module as the better one. Our metric uses three levels of priority to achieve this.

In the first level, if two modules have different weight, we select the one with larger weight. When a module is replaced, we can compute its weight by Equation (4). Weights of different types are normally considerably different and we can sort them in the decreasing order to obtain a sorted list $Lt$. Large modules such as EXP (exponential operator), and MUL (multiplication) have much heavier weight than ADD (adder) and SUB (substracter). They will play a more important role and deserve more consideration in leakage optimization. We refer to them as impact considered (IMC) type. We also define the referred set $R$ as the set of modules whose slack reduction should be considered after a replacement.

When two modules have a tie according to the first level metric, we use the following second level metric: the module that has less impact on the other module’s slack will be selected. If the replacement of one module reduces less slacks on other modules of IMC type in $R$, more flexibility will be left for future module replacement. Compared to MISG, we give a quantitative evaluation of slack impact, particularly to modules of IMC type in referred set $R$. We propose a value $V_m(p,R)$ to represent the slack impact on all the modules of IMC type $p$ in referred set $R$ by the replacement of module $m$.

$$V_m(p,R) = \sum_{a \in R} (\text{Stack}_{\text{in}}(a) - \text{Stack}_{\text{out}}(a))$$
where the type of module $a$ is $P$, $\text{Slack}_\text{init}(a)$ and $\text{Slack}_\text{end}(a)$ is the slack of module $a$ before and after the replacement of $m$. We check $V_M(p, R)$ for every IMC type $p$ according to the order in $L_t$. If two modules have different impact on the current IMC type, the module with less impact will be selected as the better one. Otherwise we will check their impact on the next IMC type in $L_t$.

After the first and second levels are used, if we still cannot judge which module is better, the third level strategy will be used. As we have mentioned earlier, leakage goes up as temperature raises. Therefore, the replacement of a module with high temperature will achieve more leakage power reduction. In addition, replacing a module at a high temperature region will help this part of the chip to cool down which will in turn reduce leakage. So at this level the module with higher temperature will be better.

### B. Solution Space Exploration

The first stage of our approach for module selection explores the solution space and yields a small set of candidate solutions for replacement based on the metric described above.

The replacement of one module, $M_b$, may make other modules unreplaceable. We call such modules conflict with $M_b$ and denote them as a set $L_c$. In this case, we will cancel the replacement of $M_b$ and try to select the best module $M_c$ from $L_c$. We now describe an important property: the best module for replacement is either $M_b$ or $M_c$.

We assume that $M_f$ is the module replaced just before $M_b$, and $M_c$ is the best that conflict with $M_b$. Consider another module, $M_o$, for replacement after $M_f$. If $M_o$ does not have a conflict with $M_b$, it can still be considered for replacement in the future; if $M_o$ conflicts with $M_b$, then by definition $M_c$ is better than $M_o$. Therefore, our approach ensures that the number of choices for the next module to be replaced is never more than two. This will reduce the complexity of our approach. Next we describe our algorithm.

#### SelectModule($S_s, M_b, R$)

Input: The set of selected modules, $S_s$
- Referred set before selecting $M_b$, $R$
- Generate the list of $CanR$ modules, $L_a$
- Generate the list of modules, $L_c$, which conflict with $M_b$
- If ($L_c$.size()>0 and the number of $M_b$'s brother less than two)
  - Delete the previous solution $M_b$ from $S_s$
  - Select best module $M_c$ from $L_c$ with referred set $R$
  - Record the $M_c$ to $S_s$
  - SelectModule($S_s, R$)
  - End if
  - If ($L_a$.size()>0)
    - Select best module $M_a$ from $L_a$ with referred set $R$
    - Record the $M_a$ to $S_s$
    - SelectModule($S_s, L_a$)
  - Else
    - Record the solution to candidate set
  - End if

Output: candidate set of solutions

**Figure 3. Description of function SelectModule().**

We update the slack of nodes and generate the list of $CanR$ modules $L_a$ and the list of conflict modules $L_c$. If there are modules in $L_c$ and the number of $M_b$'s brother is less than two, we will try to replace the two complementary modules respectively. For the first choice we accept the replacement of $M_b$ and continue selecting another module from $L_a$. For the other choice we undo the replacement of $M_b$ and select the module $M_c$ that conflicts with $M_b$.

In our approach, we consider both choices to increase the possibility of finding the optimal solution. If there are no modules in $L_c$ and $L_a$ is non-empty, we will only select the best module in $L_a$ for replacement. Then we continue selecting other modules for replacement in the next iteration. If there is no module in $L_a$, a complete solution is generated and we record it in the candidate set for sifting further in the second stage. We use a recursive function SelectModule() to implement the above algorithm. The pseudo code of this function is shown in Figure 3.

### C. Selection Approach in Stage Two

After the approach in stage one, we have obtained a set of candidate solution $S_c$. Ideally we could perform thermal analysis to all the solutions in $S_c$ and choose the one with the minimum thermal-dependent leakage power as the final solution. However thermal analysis is time cost and we cannot afford. In this subsection, we describe how to reduce the number of solutions from $S_c$ for thermal analysis and we will choose the best one from them as the final result.

The approach in this stage contains three steps. In the first step, based on the Equation (6) we select a subset of $S_c$, $S_f$. According to the first level priority described in subsection 5.A, we think the optimal solution must have enough weight reduction.

$$S_f = \{ s_i | 1 - \frac{W_i}{W_{\text{max}}} \leq \alpha \}$$

Where $s_i$ is one of solution in set $S_c$, $W_i$ is the weight reduction of $s_i$, $W_{\text{max}}$ is the maximum weight reduction of solutions in $S_f$, $\alpha$ is an empirical constant. $\alpha$ is set to 0 based on our experiment. That’s means that solutions in $S_f$ all have the maximum weight reduction.

In the second step we further delete some solutions from $S_f$ based on thermal information. Due to the deduction in section 3.A, if we select the solution with higher temperature module, more leakage power reduction will achieve. Therefore we delete the solution with lower temperature modules while other factors are the same between two solutions and this one is called the “worse” solution. We check whether every solution $s_i$ in $S_f$ should be deleted. If $s_i$ is found worse than any solution in $S_f$, $s_i$ is deleted from $S_f$.

In the last step, we proceed the thermal analysis for the left solutions in Set $S_f$. After thermal analysis, the solution with the minimum leakage power consumption will be picked out as the final result.

**Figure 4. The course of our approach in stage one.**

To illustrate the two stages of our algorithm, we take an example shown in Fig. 4 and Table 4. On the left of Fig.4, the course of module section is represented by a binary tree. The list of $CanR$ module correlative to the selected path is shown on the right. Based on our metric for module selection, module 16 is selected as the best one and module 7 is the best one among modules which conflict with module 16. The two kinds of module are respectively denoted by blue and orange. And all the conflict modules have underline in the list of $CanR$ module. Every path from root to leaf in the binary tree constructs a solution as shown in Table 4. Next we use approach in stage two to pick out the final results. The two blue solutions in Table 4 are the two maximum weight reduction solutions. Since the temperature of module 9 is higher than module 11, we select $\{16, 14,$
12, 13, 9} as the final result. It is the optimal solution validated by complete exploration.

Table 4. Solutions in candidate set and its weight reduction

<table>
<thead>
<tr>
<th>No.</th>
<th>Solution</th>
<th>Weight Re.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{ 7, 0 }</td>
<td>25.2</td>
</tr>
<tr>
<td>2</td>
<td>{ 7, 12 }</td>
<td>96.5</td>
</tr>
<tr>
<td>3</td>
<td>{ 16, 14, 12, 13, 9(98℃) }</td>
<td>444</td>
</tr>
<tr>
<td>4</td>
<td>{ 16, 14, 12, 13, 11(93℃) }</td>
<td>444</td>
</tr>
<tr>
<td>5</td>
<td>{ 16, 14, 12, 3 }</td>
<td>292.7</td>
</tr>
<tr>
<td>6</td>
<td>{ 16, 15, 14 }</td>
<td>306</td>
</tr>
<tr>
<td>7</td>
<td>{ 16, 10, 12, 9, 13 }</td>
<td>422</td>
</tr>
</tbody>
</table>

VI. EXPERIMENTAL RESULTS

In this section, we first describe the experimental setup and then report the experimental results.

A. Experimental Setup

Figure 5 depicts our overall experimental flow. We have used benchmarks from two sources: applications from the MediaBench suite and some popular DSP applications that are widely used in behavioral synthesis. All the benchmarks are represented by DFG and come from [15]. The second column of Table 6 reports the numbers of operations in each DFG. The input DFGs have been simulated to generate switching probabilities for operations using a trace of 5,000 input values. The operations of DFGs are allocated onto our dual-Vth module library which consists of five different types of ALU. These modules can perform five arithmetic operations and six logic operations. The area and dynamic power of each module refer to the data in [16].

The first set of results present the leakage power reduction of TSMS and CMPL compared with DAC05 for each benchmark, depicted in column two and three of Table 5. TSMS reduces the leakage power consumption by as high as 32.6%, and 17.8% on average compared with DAC05. The leakage power of circuit fir11 is the same between TSMS and DAC05. The reason is that the module number of fir11 is small and the slack relationship among modules is relative simple. So DAC05 can achieve the optimal solution. Except circuit matrixmul, solutions generated by TSMS are the same as the solutions generated by CMPL. In other words TSMS achieve the optimal solution for almost all the benchmarks. To circuit matrixmul, the leakage power of TSMS is only 1.8% worst than the optimal solution. However our set of candidate solution generated by stage one includes the optimal solution. If we cost two more seconds to explore more solutions of the set in stage two, we will achieve the optimal solution.

The last three columns in Table 5 show the runtime of the three approaches. Implemented on our server with Intel Pentium D 3.0GHz CPU and 6G memory, TSMS is 0.01 second faster than DAC05 on average and 74 times faster than CMPL. It demonstrates that TSMS can achieve optimal solution without runtime overhead. In Table 6, the column three reports the total number of modules. And the last two columns show the usage of high-Vth modules obtained by DAC05 and TSMS. Due to our slack impact considering, TSMS can replace more modules than DAC05 except for the first two circuits. It is one part of reason for lower leakage power achieved by TSMS.
and the on-chip peak temperature by as high as 4.7℃. When a timing conflict was met, we explored complementation solutions to the second stage. Experimental results show an average of 17.8% saving in leakage power and 4.7℃ decrease of the temperature, the lower temperature gives the contribution to the leakage power reduction.

The other part of reason is that TSMS consider thermal factor when optimizing leakage power. The last four columns in Table 7 show the average and peak temperature obtained by DAC05 and TSMS. TSMS reduces the average temperature by 2.3℃ on average and the on-chip peak temperature by as high as 4.7℃ on average. The temperature reduction is due to the replacement of more modules than DAC05 and modules with higher temperature than DAC05. Since the leakage power decreases exponentially with the decrease of the temperature, the lower temperature gives the contribution to the leakage power reduction.

### VII. CONCLUSIONS

In this paper, we proposed a two-stage algorithm to select some modules for replacement with high-Vth implementation in order to optimize thermal-dependent leakage power consumption. In the first stage, we selected modules to replace in the order which based on weight reduction, slack impact and thermal information. When timing conflict was met, we explored complementation solutions to insure optimal solution including. Then we accorded to thermal information to select the final solution from the candidate set in the second stage. Experimental results show an average of 17.8% saving of leakage power and 4.7℃ reduction of peak temperature compared with the best previous work. And runtime of our algorithm is also a little shorter. In most cases, our algorithm can achieve the global optimal solution. The results indicate that our approach is effective in the reduction of thermal-dependent leakage power without overhead.

### ACKNOWLEDGEMENT

This work was supported by National Natural Science Foundation of China under award 60876026 and 60833004. And Gang Qu’s work is partially supported by grant CNS0615222 from the National Science Foundation, USA.

### REFERENCES


