Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2012, Article ID 561580, 12 pages
doi:10.1155/2012/561580

Research Article

An Optimization Mechanism Intended for Static Power Reduction Using Dual-V_{th} Technique

Rodolfo P. Santos, Gabriela S. Clemente, Abel Silva-Filho, Cristiano Araújo, Adriano Sarmento, Manoel Lima, and Edna Barros

Informatics Center, Federal University of Pernambuco, Aveinda Jornalista Aníbal Fernandes, Cidade Universitária, 50670-901 Recife, PE, Brazil

Correspondence should be addressed to Abel Silva-Filho, agsf@cin.ufpe.br

Received 16 July 2011; Accepted 14 September 2011

Academic Editor: Dhireesha Kudithipudi

Power consumption reduction is a challenge nowadays. Techniques for dynamic and static power minimization have been proposed, but most of them are very time consuming. This work proposes an algorithm for reducing static power, which can be perfectly inserted in the conventional design flow for integrated systems considering an open source environment (*open access* infrastructure). The proposed approach, based on a Dual-Threshold technique, replaces part of the cells of the circuit by cells with a higher threshold voltage without resulting in timing violations in the circuit. The decision to replace a cell is based on timing estimates of the circuit modeling with the cell replacement, before it is actually replaced. The fact that only some cells are replaced every iteration results in a reduction of the runtime of the algorithm. Additionally, results showed a reduction in static power up to 39.28%, when applying the proposed approach in the ISCAS85 benchmark circuits.

1. Introduction

Nowadays, power consumption is one of the biggest challenges faced by the semiconductor industry [1]. With the advancement of technology, the increasing market of systems that are portable and have long battery lifetime has created an area of great importance in academic research and industry [2]. The research for technologies that consume less power is a constant in our days. In the past, much focus on VLSI design consisted of improving performance and reducing area. With the growing demand for equipment using wireless network and portable devices, power consumption has become a critical problem and it is becoming one of the main concerns during the design of VLSI circuits. In this scenario, it is extremely important to enlarge the batteries lifetime as much as possible. In this sense, many efforts have been made in order to develop optimization mechanisms to reduce energy consumption of electronic systems at different design abstraction levels such as transistors [3], gate level [4], architecture level [5], and system level [6], as well as methodologies aiming low-power designs such as the work developed by [7] and (Keating et al., 2008).

Another aspect that has been taken into account is the impact of previously neglected power consumption sources. Until very recently, only dynamic power was considered as a significant source of power consumption, supported by "Moore's Law" (Gordon E. Moore, 1965). Moore predicted that the number of transistors on a chip would double every two years. However, with the reduction of the device sizes below 0.1 um, the static power is imposing new challenges for low-power circuits. For circuits implemented using 90 nm and 65 nm technologies (Keating et al., 2008), leakage power is nearly the same as dynamic power, and in many cases surpassing it, when a suitable optimization strategy is not adopted.

Recently, several works have targeted leakage power minimization. A method for reducing static power in standby mode on transistor level has been proposed in [9]. This approach proposes a heuristic which divides the optimization of static power consumption into two distinct stages: the scaling of transistor sizes and the assignment of the threshold voltage of each gate transistor, chosen between two possible values. This approach results in up to 21% of power savings for some ISCAS85 benchmark circuits [10].

However, the algorithm implementing this approach is very time consuming, since it uses as design space the libraries available, and these libraries are usually large in size. A similar disadvantage is also found in the approach proposed by Lee and Blaauw [11]. Another technique for power optimization at transistor level called MVT (Mixed-V_{th}) that is applied to CMOS circuits has been proposed by [12, 13]. This work reports that up to 20% more power savings can be achieved when compared to techniques applied at gate level. Using this approach, the final number of high V_{th} voltage (HVT) transistors is greater than the number of HVT transistors obtained after the application of gate level algorithms. However, this approach deals with very large standard cells libraries, which in general might have a high computational cost.

A dual-threshold algorithm that introduces a key priority factor for cells replacements has been proposed in [14]. In this work, the amount of subthreshold current reduction related to the increase of the cell delays determines the priority of a particular cell to be replaced in the circuit. Using this factor, a cell that causes the greatest reduction in power consumption and less degradation in the circuit timing has a higher replacement priority over the others. This technique reports an average static power reduction of 25% over the initial static power consumption. However, in this work when the replacements are made, the timing of the circuit is not updated. This can cause serious problems, since after each replacement, the critical paths of the circuit may change. So the next cells to be treated will have their timing constraints underestimated because a previous replacement might result in more severe constraints to these cells, and they will not be considered. Moreover, the input circuits used by this methodology are circuits with a single threshold voltage. This fact does not match with the more commonly used design flows, when more than one threshold voltage is available in libraries.

In the approach proposed by Elakkumanam et al. [15], the minimization of power regarding the leakage dissipation and glitches is treated. A dual-V_{th} technique and buffers insertion (for balanced paths) are applied. The problem is solved using an ILP-based (integer linear programming) technique. However, the application of an ILP-based approach may not be suitable for large designs, because it is a very time-consuming optimization approach. A technique for reduction of leakage power of 90 nm and 65 nm integrated circuits with multimillions gates by exchanging low V_{th} voltage (LVT) cells belonging to less critical paths of the circuit by HVT cells has been proposed by Gupta et al. [16]. This work targets the problem of long runtime and static power consumption using a heuristic method. Additionally, this algorithm aims at optimizing the *netlist* of an initial multi-V_{th} circuit. Unlike path-based approaches (in general, by optimizing the noncritical paths of the gate circuits), in [16] it does not update the timing at each cell replacement. Even with this feature, for circuits with multimillions gates, the algorithm runtime becomes a dominant factor during design flow. However, this approach always considers the worst cases when dealing with the delays estimation. Thus, further power savings could be obtained if more realistic values were estimated. Moreover, several heuristic parameters must be defined by the designer.

This work proposes a novel approach for reducing static power consumption that is suitable to be inserted in the conventional design flow of integrated circuits. This approach uses a dual-threshold technique, where the cells of the circuit are replaced by cells with a higher threshold voltage without causing timing violations in the circuit. More realistic delay values and slew estimations are used by the proposed technique in order to determine the replacement of each cell in the circuit, before it is actually replaced. Additionally, this work aims to implement a tool that has a runtime compatible with design constraints and file formats, which fits designs with tight time-to-market.

The contributions of this paper can be summarized as follows: (i) a novel approach for reducing static power consumption compatible with commercial design file formats and capable to be integrated with conventional design flow of integrated circuits; (ii) the calculation of the *maximum slew degradation* parameter used to evaluate the replacement of a given cell. Unlike other approaches available in the literature, this is made simulating all situations in which the substitute cells could replace the target cells without causing timing violations in the circuit; (iii) the use of open access infrastructure, an open source structure that was extended to support TSMC standard cells library and multi-V_{th} designs optimization.

The rest of this paper is structured as follows. The next section explains the dual-threshold technique which is used in this work for power optimization. Section 3 presents the proposed approach. In Section 4, experimental results are presented. These results were obtained by applying the proposed algorithm to several combinational circuits of the ISCAS85 benchmark. Finally, Section 5 gives conclusions and future directions.

2. DUAL-V_{th} Technique for Power Optimization

Basically, this technique explores the excessive slacks in noncritical paths targeting the reduction of static power consumption. The optimization process results in a circuit with reduced power consumption and two types of gates composed of HVT transistors and LVT transistors. Figure 1 shows a circuit optimized using this technique.

The use of two threshold voltage values is a technique for reducing power either when the circuit is in active or in standby mode. In order to obtain transistors with different threshold voltages, during the manufacturing process, several characteristics of the transistors may vary, such as the channel doping, the channel length, the polarization of the transistors body, and the thicknesses of gate oxide.

In situations where there are noncritical paths meaning that the signal propagation occurs more quickly than required, these paths might be delayed without affecting the circuit performance. As the transistors with a high V_{th} voltage (HVT) present low subthreshold currents, while the subthreshold currents of the transistors with low threshold voltage (LVT) are higher, this approach optimizes the

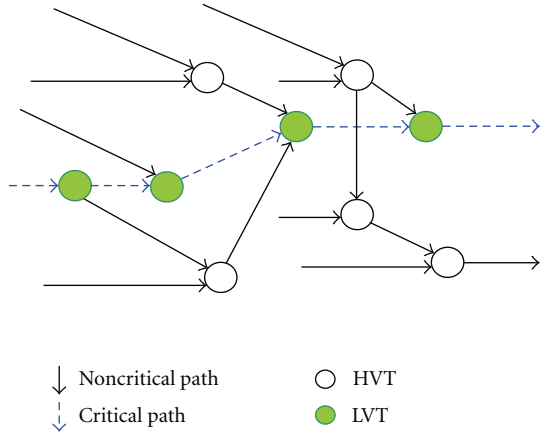

Figure 1: Circuit with two types of gates.

power consumption by replacing the LVT cells that are in noncritical paths by HVT cells. However, it must be guaranteed that the replacement of cells will have no impact on the overall performance of the circuit. It can be achieved through optimization strategies such as proposed in this work.

Figure 2 shows an example where there is a noncritical path composed of LVT cells. Considering that the critical path defines the required time of the circuit paths equal to 10 ns, there is a path that arrives 5 ns earlier than required which means that this path has a slack equal to 5 ns.

When the LVT cells are replaced by HVT cells, as shown in Figure 3, the path still presents a slack equal to 2 ns. This implies that the changes in the circuit caused by the replacement of some cells belonging to a noncritical path can delay the information by the value of the path slack without degrading the circuit performance. This illustrates how this approach can reduce the static power consumption, since HVT cells dissipate less static power. Currently, the application of a dual-V_{th} flow during synthesis has become relatively common [8]. This usually involves an initial synthesis using a primary standard cells library, followed by an optimization using one or more libraries with several thresholds voltage.

3. The Proposed Approach

There are several ways for optimizing static power consumption using threshold voltage (V_{th}) assignments including some approaches that try to find the optimal values of V_{th}. One problem is that these values are often not available in commercial standard cell libraries and the designers do not have such a variety of threshold voltages. In general, even a complex library has a finite and limited number of cells characterized by different V_{th} voltages; usually only two or three values are found.

Another important issue that should be pointed out about the other approaches is that many schemes do not take into account the fact that at each time a cell is replaced by another one with a different V_{th}, and the delays of the circuit must be updated, since the critical paths may have

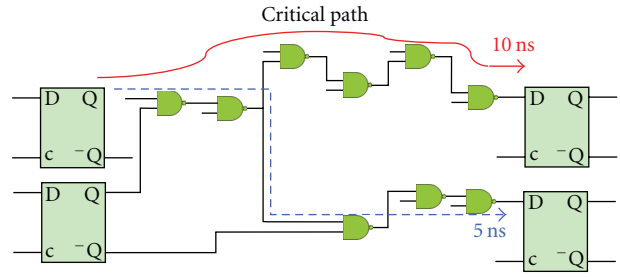

Figure 2: Critical and noncritical paths.

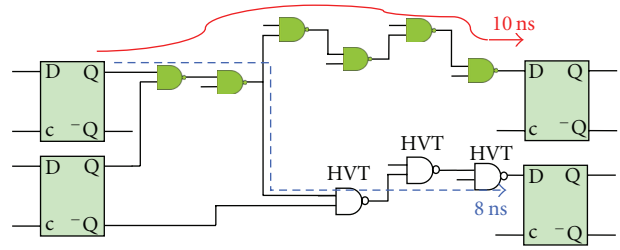

Figure 3: Replacement of LVT cells by HVT cells in noncritical path.

been changed. Furthermore, if the runtime of the power optimization algorithms is not considered, it may be impracticable to be applied in circuits composed of millions of cells. Thus, we must carefully choose the optimization approach, because many of them may be not useful. Depending on the applications, the optimization algorithm may have a very high computational cost, increasing the design time.

Since, in general, critical or noncritical paths are composed of a series of cells in cascade, the replacement of a single cell in a path can completely change the timing characteristics (slacks and slews) of the whole path. Path-based approaches incorporate the changes in the circuit due to cells replacement going through the paths level by level and updating the database after each change. A restriction of this method is that updating the database at every change leads to a huge increase in the algorithm runtime, especially for large circuits.

The problem of static power dissipation considered in this paper is being handled after the global routing stage, when the timing and area constraints have been resolved and information about the parasitic capacitances are available with good accuracy. However, this feature does not prevent it from being applied after the synthesis step.

The proposed approach has as main goal to reduce the leakage power consumption without increase of the computational cost. The main feature for maintaining the same computational cost is that no update of timing in the database is necessary every time a cell is analyzed, unlike the mentioned approaches. In contrast, this is done by simulating the potential library cells that can replace the circuit cell with a higher V_{th} voltage.

The block diagram describing the proposed flow can be seen in Figure 4. This flow takes a circuit netlist as entry, and then a static analysis is performed in order to find the

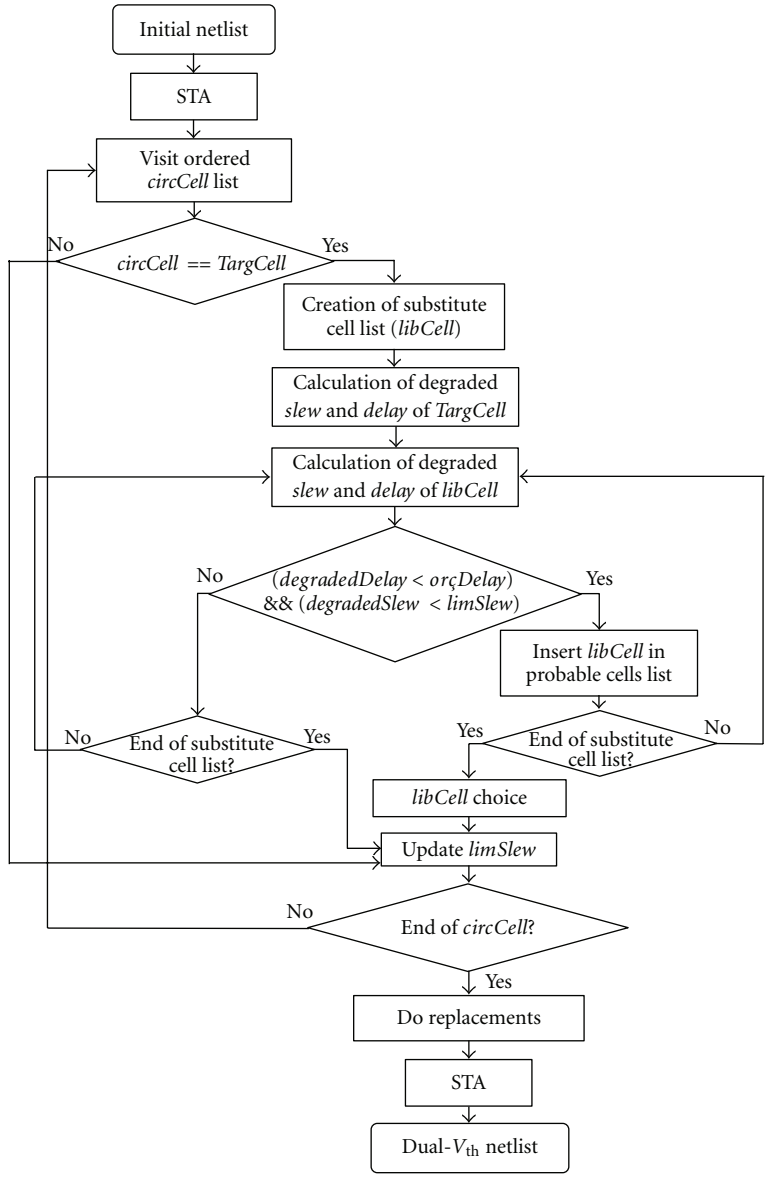

Figure 4: Detailed algorithm flow.

LVT cells with a gap of sufficient slack to be exchanged for HVT cells. Each cell of the circuit is analyzed in order to create lists of cells that can be replaced and establish an order for the treatment of the cells. Tests are done using cells from the library to enable the choice of the cell that will replace the cells of the circuit. These tests include checking the maximum fan-out of the cell from the library, checking the functionality of the cell to compare the degradation of the slew and delay of this cell with values heuristically defined. Only when all cells have been checked, the exchanges will be made, so that effectively only one static analysis is performed to ensure that no timing violation is caused by substitutions defined along the iterations of the algorithm. The following subsections explain the proposed flow and present some definitions that are necessary to understand the proposed approach.

3.1. Creating a List of Cells Candidates to Be Replaced (circCellList). Once the static timing analysis has been applied, the first step is to create a list (*circCellList*) that contains all cells (*targCells*) that can be replaced. In order to support a netlist containing cells with HVT (high V_{th}) and LVT (low V_{th}) cells, this list includes LVT cells and cells with a slack greater than a parameter defined as a safe slack (*folgSlack*). It should be noted that the ability to support an input netlist characterized with more than one V_{th} and the availability of multi-V_{th} libraries are essential features for optimization using multi-V_{th}-based techniques. The pseudocode for the creation of the *circCellList* is shown in Algorithm 1.

After the creation of the *circCellList*, the input slews and output loads of the cells are calculated. It should be emphasized that in this approach all the arcs of the cells are taken into account, so the worst case is always considered

```
CircCellList(netlist, folgSlack) {
        while (∗iterator getnext()) {
                if ((SlackCell > folgSlack) && (Cell = LVT)) { CircCellList.insert
                (iterator++, Cell)
                }
```

Algorithm 1: The pseudocode for the creation of the *circCellList*.

```
substituteList (circCell) {
    while (∗libCell = iterator.getnext()) {
        if ((cell = HVT) && (circCellFunction = libCellFunction)
        && (libCellFanout < circCellFanout) && (libCellFanout >
        circCellFanout/2)) {
                substituteList.insert(iterator++, libCell).
        }
```

Algorithm 2: Pseudocode for the creation of the list of substitute candidates.

when estimating the cells delays. This makes the timing analysis of the circuit more complex but more accurate. Thus, such precision leads to an increasing possibility of power reduction and better assurance that there are no timing violations in the circuit. Both delays arcs and slew arcs will be taken into account.

3.2. Substitute Candidates List (libCell). For each cell that can be replaced by the algorithm, a list with its respective substitute candidates will be determined. After that, this list is filtered in order to determine the substitute cell that will replace the target cells.

At each algorithm iteration, this list will be created (*circCell*) by taking into account some restrictions of the library. First, the library cells must have the same functionality as the cells in the list *circCell* because the design behavior cannot be changed. Moreover, as the purpose of the cell replacement is to reduce the static power consumption, all the cells in this list must be HVT cells.

A heuristic parameter is defined in order to limit the maximum fan-out of the possible cells in the list. All cells belonging to the substitute candidates list must have a maximum fan-out equal to or lower than the *circCell* maximum fan-out and higher than half of this value. These parameters were defined after exhaustive simulations in order to obtain the best optimization results, based on [16]. The pseudocode for the creation of the substitute candidates list is described in Algorithm 2.

3.3. Parameter limSlew. This parameter is the maximum slew degradation acceptable at the output pin after the cell replacement. Due to the cascading of the cells, this parameter influences and determines the input slews of all fan-out cells. In the proposed approach, the precise estimation of this parameter is one of the main issues. This parameter is calculated simulating all situations in which the substitute cells could replace the target cells without causing timing violations in the circuit. Instead of replacing the cell after the choice of its substitute and updating the circuit timing, the algorithm keeps the pair of cells to be exchanged with their substitutes, and the replacement is done only after all the cells have been analyzed. Thus, a single static timing analysis is performed at the end of the algorithm.

The *limSlew* is used during these simulations to estimate the degradation of the input slew of the cell being analyzed taking into account the possible slew degradation caused by any cell previously analyzed belonging to its fan-in cone. It is calculated based on the average of the degraded output slews of the cells in the substitute candidate's cell list as shown in the equation below, where *origSlew* is the original output slew related to the worst timing arc,

$$limSlew = [Average\_output\_slew\_with\_degraded\_input] - origSlew. \quad (1)$$

These degraded slews are determined considering the same output capacitance as the circuit cell output capacitance. So this parameter is dependent on the libraries used and the design characteristics, being very generic, and does not need to be configured by the user. Regarding the output capacitances used during the simulation, it is important to emphasize that we are dealing with technologies below 90 nm, so parasitic interconnections are dominant compared to the cell output load. Thus, the simulation with the actual wire capacitance represents the worst case for the simulation, and it guarantees that no timing violations will be inserted.

After the choice of the substitute cells using the slew degradation defined in (1), the *limSlew* parameter of all circuit cells is updated according to the real output slew degradation obtained at the end of the analysis of each cell. Concerning the *limSlew* parameter, it is classified according to the cell being analyzed. It may be (1) an HVT cell which

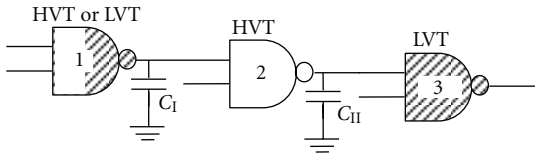

Figure 5: Situation of HVT cell to calculate the *limSlew*.

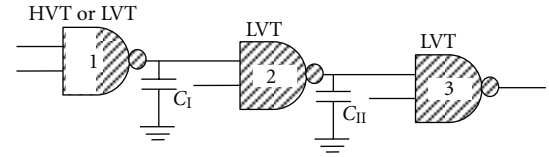

Figure 6: Situation of the LVT cell not changed for the calculation of *limSlew*.

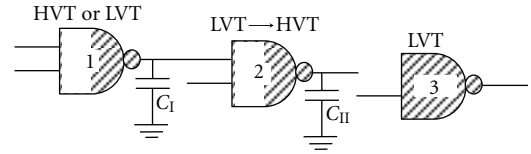

Figure 7: Situation of the LVT cell exchanged for the calculation of *limSlew*.

propagates its input slew to its output, (2) an LVT cell that will not be exchanged, (3) or an LVT cell that will be exchanged.

Situation 1 (HVT cells). Assuming a circuit with only three cells, cell I is the entry of the circuit, as shown in Figure 5.

In Figure 5, cell II is the one in which the *limSlew* is being calculated. The circuit is composed by three cells in cascade which means that the propagation of slew degradation must be taken into account. In this situation, cell II has at one of its inputs cell I that may or not have been replaced. In the case that cell I has been replaced, the output slew of cell II will be the same because its fan-in cone has no modification. This cell will not be replaced because it is an HVT cell. However, in the case where cell number 1 has been changed, cell number 2 must propagate the degraded slew due to the new output slew value corresponding to the substitute of cell number 1. As the situation has changed, the new degradation due to the slew propagation through cell number 2 reaching cell 3 must be calculated. Thus, cell 3 will have its input slew degradation determined by this parameter. Note that this requires traversing the circuit from its primary inputs to its primary outputs. The *limSlew* of cells that are not replaced is calculated as follows:

$$limSlew = circSlewDeg - origSlew. \quad (2)$$

The value *circSlewDeg* is given by the degraded slew at the output pin of the circuit cell applying a degraded signal at its input.

Situation 2 (LVT cells that will not be exchanged). Considering the same circuit shown before, in this situation, we assume that cell 2, which is considered to be LVT, has no substitute candidate or does not have a sufficient slack. This case is illustrated in Figure 6.

Since the algorithm runs from primary inputs to primary outputs, cell 3 has not been analyzed until cell 2 analysis starts being performed. As cell 2 will not be exchanged, there will be slew propagation through cell 2 that can be degraded, depending on what happened with cell 1. In order to consider the propagation, we must simulate the cell with the situation that may be new, depending on the exchange of cell 1. So, the equation to calculate the slew degradation in this case is the same as (2).

Situation 3 (LVT cells that will be exchanged). In this situation, the circuit cell has a substitute candidate that has passed the simulation tests. Figure 7 illustrates this situation.

In this case, the cell being analyzed (cell 2) will be replaced. Thus, replacement of this cell will result in a slew degradation that will be seen at the input pin of cell 3. Once the slew degradation of cell 1 due to its exchange or not is well defined and cell 2 was analyzed, following the above cases, we can determine the slew degradation caused by the exchange of cell 2 in accordance with the following equation:

$$limSlew = [Output\_slew\_of\_chosen\_cell] - origSlew. \quad (3)$$

All the situations discussed above are related to the calculation of the *limSlew* of a cell that will be needed at the next cell input as a parameter for the calculation of its input slew degradation.

3.4. Other Degradation-Delays-Related Parameters. The parameters related to delays due to degradation include *circDelayDeg, circSlewDeg, orçDelay, folgSlack,* and *fan-out limit, which* will be described in the following.

(i) *circDelayDeg*: this parameter corresponds to the delay degradation at the output of the circuit cell when a degraded input is applied. As the contribution of this work is that the proposed algorithm takes into account more realistic situations in the circuit, the degraded input signal will be based on the *limSlew* parameter related to all fan-in cells. This parameter will be used during the filtering process applied to the cells belonging to the substitute candidates list. The calculation of the delay degradation is performed to each timing arc of the cell.

(ii) *circSlewDeg*: the calculation of the *circSlewDeg* parameter that is related to the output slew degradation is similar to the calculation of the delay degradation of the cells in the circuit. So, it must be calculated arc by arc. It is also used during the analysis of the substitute candidates list in order to reduce the possible library cells that can replace a cell in the circuit.

(iii) *orçDelay*: the delay budget of a cell is the maximum delay degradation value allowed due to the exchange

```
updateFan-inCell (circCell) {
        while (there is a fan-in cell of circCell that has not been visited) {
                if (the current fan-in cell is not ordered) {
                        updateFan-inCell (fan-inCell)}}
        set circCell ordered;
        insert circCell at the end of the ordered list;
}
```

Algorithm 3: Pseudocode of the cell sorting algorithm.

of the cell. Clearly, this value is proportional to the delay contribution of the cell to the worst path that it belongs to. This parameter is calculated for each arc of the cell. It is important for the algorithm, specifically in building the list of the probable cells list, since this parameter will be a key reference point for testing the cells in alternates list, so that the degradation in delay caused by the exchange of cell cannot exceed the value of *orçDelay*.

Its calculation is done by using the following equation, where the delay is the input-output arc delay of the cell:

$$OrçDelay = \left(\frac{Delay}{Slack}\right) \cdot (Slack - folgSlack). \quad (4)$$

The slack is the worst arc slack and is calculated with the static analysis. This parameter is based on [16].

(iv) *folgSlack*: this parameter corresponds to a compensation of errors that can occur due to any heuristics approximation and also compensates the propagation of degraded slew due the cascading of cells in the circuit.

This parameter is used by the designer to minimize the penalties of timing that can be caused by this heuristic. Thus, for each circuit, a value is determined to optimize the results and ensure that the resulting circuit will not have cells with negative slacks.

(v) *Fan-out limit*: in order to choose the cells that will be part of the substitute candidates list, a lower limit of 50% on the drive strength of the cell is imposed. This means that the library cells having drive strength less than 50% of the drive strength of the original cell in the circuit will not be part of the list. Moreover, the drive strength upper limit is defined as the drive strength of the original cell.

3.5. Cell Sorting. The procedure to determine the possible standard cells that may replace a gate in the circuit in order to reduce the static power consumption imposes an order in which the gates of the circuit should be analyzed. This order must be such that a gate may be analyzed only if all the gates belonging to the paths that arrive at the input of this gate have been analyzed. This condition is necessary because the criteria used during the simulations of the situations when a replacement might happen (e.g., the slew degradation) are dependent on the fan-in gates of the gate under consideration.

The method implemented for obtaining the desired order is based on the depth first search (DFS) algorithm. The search of the cells starts from the primary outputs of the circuit toward the primary inputs. The starting node of the ordering method is a cell connected to a primary output of the circuit. Then, all the cells connected to its input (cells belonging to the fan-in cone) are visited. For each fan-in cell visited, the previous step is again executed. This procedure is recursively executed until the search algorithm visits a cell whose input pins are connected to primary inputs. If this occurs, the cell is inserted at the end of the sort list.

During the search loop when the cell fan-ins are being visited if a fan-in cell has already been visited, the next iteration of the loop is executed. In Algorithm 3, there is a description of the pseudocode executed having as starting nodes each primary output of the circuit.

3.6. Calculating the Effect of Cell Replacement. Another two parameters that are key points in the algorithm are *libSlewDeg* and *libDelayDeg*. They consist, respectively, of the output slew and the delay of substitute cells when a degraded signal is applied to its inputs. This signal degradation is defined based on the parameters of the fan-in cells. The *libSlewDeg* is the degraded output slew of substitute cell when a slew degraded signal is applied at the input pins. The *libDelayDeg* is the delay of the substitute cell, as a degraded signal is applied to its input. It should be noticed that in both cases the values are determined considering the same output load of the *circCell* concerned.

Using the above-described parameters, it is possible to calculate the *degradDelay* and *degradSlew*. These parameters represent the degradation of the delay and slew, respectively, caused by the supposed replacement of the circuit cell by a substitute candidate cell that is being analyzed. Thus, the delay degradation is the subtraction between the circDelay-Deg and the original delay of the circuit cell, and the slew degradation is obtained by subtracting the circSlewDeg and the output slew of the original cell in the circuit. The pseudocode can be shown in Algorithm 4.

3.7. Calculating Probable Cells List and Replacing Cells. The determination of the probable cells list is done through a series of tests with the possible candidates list. These tests must ensure that once a new cell is placed in the circuit, such

```
loadCircCell = takeLoadCircCell (circCell);
inputSlewDeg = takeCircSlew(circSlew) + limSlew(takeInputCell(circCell);
libSlewDeg (circCell, loadCircCell, inputSlewDeg) {
while (∗arcLibCell = iterator.getnext()) {
        slew/delayArcVector = getTimingArcSlew/Delay(libCell);
        }
}
```

Algorithm 4: Pseudocode of *libSlewDeg* calculation.

```
probablesList(circCell, substituteList) {
    orçDelay = orçDelay(circCel);
    for (iterator = substituteList.begin(), iterator != substituteList.end(), iterador++) {
        while(∗arcLibCell = iterator.getnext()) {
            degradDelay = libDelayDeg – circDelayDeg;
            degradSlew = libSlewDeg – circSlewDeg;
            if((degradDelay > orçDelay) && (degradSlew > limSlew) {
                break;
            }
        }
        probablesList.insert();
    }
}
```

Algorithm 5: Pseudocode for determining the probable cells list.

cell will insert no timing violations in the circuit. The tests are based on heuristic parameters that were found by exhaustive simulations, in order to achieve the best power savings. The pseudocode can be shown in Algorithm 5.

They consist basically of two main criteria that allow the inclusion of the substitute candidates in this list. The first one ensures that the delay degradation of the substitute candidate will not exceed the delay budget of the cell in the circuit. The delay budget is the parameter that determines how much delay degradation is allowed due to the replacement of the cell in the circuit. The second criterion concerns the slew degradation. It says that a possible candidate is not allowed in the probable cells list if its slew degradation is larger than the *limSlew* of the cell in the circuit.

Once the list of probable cells has been created, a *libCell* must be selected; the criterion for choosing the cell among the probable cells to be replaced in the circuit is simple. The algorithm visits all the cells belonging to the list of probable cells and chooses the cell that has the lowest leakage power consumption. The information about the leakage power consumption of the probable cell is found in the standard cells library file.

All cell replacements are made after analyzing every cell of the circuit. Once the library cell that will replace each cell in the original circuit has been chosen, a list of pairs composed by a circuit cell and its corresponding substitute cell is created. When this list is completed, all cells replacements are performed at once. The static timing analysis (STA) is performed at the beginning of the process, and in a second moment, at the end of the algorithm. This is very important for reaching one of the goals of this work which is to implement a tool that has a runtime compatible with design constraints and which fits projects with tight time-to-market. This is due to the fact that the execution of STA is extremely costly during the algorithm execution.

The resulting circuit after the complete execution of the algorithm consists of a netlist compatible with commercial tools such as Cadence tool. As this is an approach applied on gate level in order to reduce the leakage power consumption, by replacing LVT cells by HVT cells, the algorithm generates a new netlist similar to the original, but with some replaced cells. Moreover, the changes in the circuit do not affect the placement and routing of the gates, so the information listed in the constraint files generated during the design flow will remain unchanged.

In order to illustrate in a detailed manner the proposed method, a step-by-step explanation will be described using as example the dual-V_{th} combinational circuit illustrated in Figure 8 and the appendix. Although it is a hypothetical example, the proposed method is able to receive an already optimized netlist as input. The following figure considers a circuit containing HVT and LVT cells as input, and the proposed method aims at improvements over this original circuit.

Once all the steps mentioned in the appendix are executed, the new circuit would have the cells 1 and 2 replaced by their corresponding substitutes. So the reduction in power could be supported even for a circuit synthesized with two threshold voltages.

The circuit generated by the proposed algorithm is shown in Figure 9.

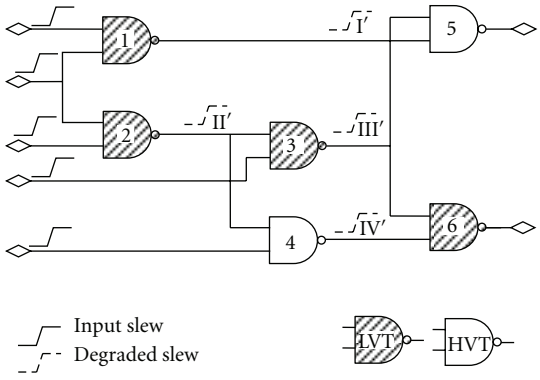

Figure 8: Hypothetical circuit with HVT and LVT cells for exemplification.

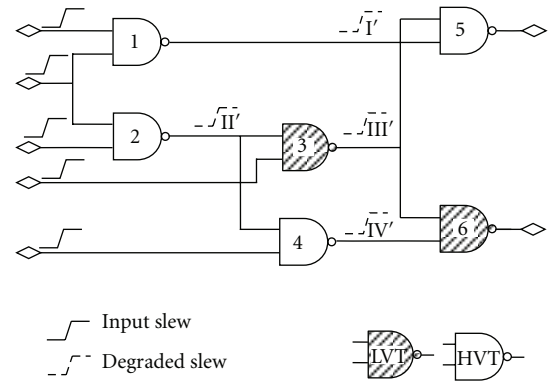

Figure 9: Circuit resulting after the algorithm's steps.

4. Experimental Results

In order to optimize the static power consumption, the proposed algorithm has been applied to several combinational circuits belonging to the ISCAS85 benchmark. The proposed algorithm was implemented in C/C++, using the open access infrastructure [17]. The synthesized circuits of this benchmark have varied size from 4 cells to 2327 cells, using a 90 nm TSMC standard cells library. The multi-V_{th} designs were created by the use of the *tcbn90lphphvttc* and *tcbn90lphphvttc* libraries, which characterize the HVT and LVT cells, respectively.

In order to execute the algorithm with all the benchmark circuits and extract the results as consistent as possible with a real situation within an integrated circuit design flow, each circuit has a value related to time constraints imposed. The timing restriction of each circuit was determined according to the cycle time of the most critical path, so the input circuit has no cell with negative slack. With an accuracy of 0.1 ns, the most conservative timing constraints were determined.

The results were satisfactory with an average static power reduction of 23.52% of the total static power of the original circuit. The maximum static power saved reached 39.28% as indicated in the last row of the Table 1. Except for circuit c17, which was discarded due to its very small size, all circuits presented a reduction in their power consumption. Another important point is the fact that the goal of developing a tool that perfectly fits the conventional design flow was achieved.

Since the main objective is the reduction in static power, the heuristics parameters were set up, so that it would provide the maximum reduction in static power. These settings have been chosen in a way that no timing penalties are inserted in the circuit. For example, the *folgSlack* parameter, which guarantees that the slews and delays propagation will not result in timing violations, must be chosen as small as possible. However, the optimization process must not cause any negative slack cells. Table 1 shows the results for the ISCAS85 benchmark circuits.

The results presented in Table 1 show that, as expected, the circuits with larger number of gates have a higher cycle time due to the increased number of parasitic interconnections in the circuit. Thus, their performance decreases with increasing size. With respect to the *folgSlack* parameter, we may notice a tendency to its growth with the growth of the circuits. As outlined before, this parameter works in order to compensate possible timing violations during the signals propagation because of the degradations caused when a cell is replaced. As the circuits grow, the value of this parameter also tends to grow because it must prevent the circuit from having timing violations.

Table 1: Experimental results.

Circuit	Total cells	*folgSlack* (ps)	Cycle time (ps)	Leakage reduction (%)
C499	146	0	1.9	5.00%
C1355	146	9	1.9	11.81%
C880	167	28	1.3	39.28%
C2670	287	33	2.0	24.50%
C3540	522	84	3.1	31.38%
C5315	616	17	2.8	29.10%
C7552	849	46	4.9	31.87%
C6288	2357	155	5.6	15.23%
Average				23.52%

Figure 10 shows the total number of cells replaced by the algorithm in each circuit. The amount of cells replaced is proportional to the amount of cells in the circuit and reaches an average percentage of approximately 28.48% for ISCAS85 applications indicated in Figure 10.

A host with dual-core 2.2 GHz and 1 GB of RAM memory was used to evaluate and synthesize all applications from ISCAS85. We can observe that the overhead for cell replacement in the largest circuit (C6288) from ISCAS85 is around 12 minutes. It is a large overhead yet; however, this time of processing may be reduced when executed in a more powerful machine.

The difficulty of finding available tools intended for dual-vt optimizations in the literature, as well as the standard cells library characterized for dual-V_{th}, meant that it was not possible to be compared with other static power techniques.

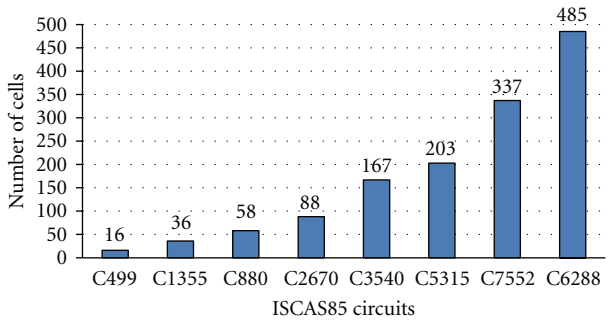

Figure 10: Total number of cells replaced.

5. Conclusions and Future Works

A Dual-V_{th} algorithm for reducing static power consumption with a viable runtime for commercial project of integrated circuits was presented in this paper. The average static power savings reached about 23.52% of the total static power consumption of the original algorithm, considering the ISCAS85 benchmark circuits.

Among the optimization possibilities of the proposed algorithm found over its implementation, we could identify some variations that were indicated as future work. Considering the slack of the path is not interesting that the replacement of a given cell consumes the complete slack of the path; thus, the *orçDelay* function can be rethought of in order to make better use of time in excess (in our case calculated by *folgSlack* function) due to replacement of the cell.

Another aspect is the extension of this work to contemplate support to sequential circuits. Currently, the proposed approach is limited to combinational circuits. The impact for this extension is not so great in the project; however, some functions must be implemented and added to component library following the *open access* infrastructure standard.

An extension of this work has been developed in parallel in order to provide a heuristic to reduce dynamic power consumption by using the same infrastructure. However, the main idea follows another strategy and is currently still in development.

Appendix

(1) *Sorting the cells of the circuit from primary input to primary output. In the example, the sequence is 1, 2, 3, 4, 5, and 6.*

(2) *Creating the circCellList list. Assuming that only the cells 1, 2, and 3 have slack greater than the parameter fogSlack = 3,*

Cell	*1*	*2*	*3*	*4*	*5*	*6*
Slack	*4*	*4*	*4*	*1*	*0*	*2*
Slew	*0.1*	*0.1*	*0.1*	*0.2*	*0.2*	*0.1*

(3) *Iteration through the cells of the circuit (all cells).*

(4) *Creating the substitute cell list of cell 1 (cell 1 is in circCellList).*

(5) *Assuming there are cells in the library that may be part of this list, execute the method for creating the initial limSlew. Take the average of the output slews of substitute cells with the degraded input (in this case, the input slew is the same because the cell is at an input of the circuit) and the same output load of cell 1.*

limSlew1 initial = 0.1.

(6) *Calculating circDelayDeg and circSlewDeg of cell I. With a degraded input signal (it will be the same as cell 1 is at the input of the circuit), the delay and the output slew for cell 1 are calculated,*

circDelayDeg (inputSlew, outputLoad) = 0.1,

circSlewDeg (inputSlew, outputLoad) = 0.1.

(7) *For each cell in the substitute cell list, the libDelayDeg and libSlewDeg are calculated,*

assuming cell 1 and cell 2 can be exchanged,

libDelayDeg I (inputSlew, outputLoad) = 0.3,
libSlewDeg I (inputSlew, outputLoad) = 0.3,
libDelayDeg II (inputSlew, outputLoad) = 0.2,
libSlewDeg II (inputSlew, outputLoad) = 0.2.

(8) *Considering the values obtained in step (7), the degradSlew and degradDelay of cell 1 are calculated,*

degradSlew I = 0.3 − 0.1 = 0.2,

degradDelay I = 0.3 − 0.1 = 0.2,

degradSlew II = 0.2 − 0.1 = 0.1,

degradDelay II = 0.2 − 0.1 = 0.1.

(9) *The orçDelay of cell 1 is calculated,*

orçDelay (cell I) = 0.1.

(10) *Test to create the probable cell list. If delay degradation is smaller than orçDelay and slew degradation is smaller than the limSlew, the substitute cell will be included in probable cell list,*

degradSlew1I > limSlew is not included in the probable list,

degradDelay 1 > orçDelay is not included in the probable list,

degradSlew 2 = limSlew—OK,

degradDelay 2 = orçDelay—OK.

(11) *If probable cell list is not empty, the cell that has the lower consumption of power will be chosen to replace cell 1;*

Cell 2 in the lib was chosen.

(12) *Assuming that there is a substitute for cell 1, the limSlew of cell 1 is updated. It will be taken into account during the analysis of cell 4. It will be the value of libSlewDeg minus the original slew of cell 1,*

limSlew 1 = libslewDeg *− initial slew (cell 1)= 0.2 − 0.1 = 0.1.*

(13) *Analyzing cell 2. Since cell 2 has the same conditions as cell 1, its treatment will be similar, so cell 2 will also have a substitute and its limSlew will be updated.*

After analysis limSlew 2 is updated = 0.2.

(14) *Analyzing cell 3. As one of its inputs is connected to the output of cell 2, the worst slew at the input should be taken in account.*

(15) *As cell 3 is present in circCellList, its substitute cell list is created.*

(16) *The initial limSlew is created similarly to the cells 1 and 2,*

with degradation of 0.2 *in the input,*

initial limSlew 3 = 0.1.

(17) *Calculating circDelayDeg and circSlewDeg of cell 3. With a degraded input (will be governed by limSlew generated and updated in cell 2), the degraded output delay and slew of cell 3 are calculated,*

circDelayDeg 3 (inputSlew, outputLoad) = 0.2,

circSlewDeg 3 (inputSlew, outputLoad) = 0.2.

(18) *For each cell in substitute cell list, the libDelayDeg and libSlewDeg are calculated,*

supposing two substitutes:

libDelayDeg 1 (inputSlew, outputLoad) = 0.5,

libSlewDeg 1 (inputSlew, outputLoad) = 0.5,

libDelayDeg 2 (inputSlew, outputLoad) = 0.4,

libSlewDeg 2 (inputSlew, outputLoad) = 0.4.

(19) *The degradSlew and degradDelay of cell 3 are calculated,*

degradSlew 1 = 0.5 − 0.2 = 0.3,

degradDelay 1 = 0.5 − 0.2 = 0.3,

degradSlew 2 = 0.4 − 0.2 = 0.2,

degradDelay 2 = 0.4 − 0.2 = 0.2.

(20) *The orçDelay of cell 3 is calculated,*

asuming that orçDelay (cell 3) = 0.1.

(21) *Test to create the probable cell list. If delay degradation is smaller than orçDelay and slew degradation is smaller than the limSlew, the substitute cell will be included in probable cell list,*

degradSlew 1 > limSlew is not included in the probable list,

degradDelay 1 > orçDelay is not included in the probable list,

degradSlew 2 > limSlew is not included in the probable list,

degradDelay 2 > orçDelay is not included in the probable list.

(22) *Assuming that there are no substitute cells able to pass the tests.*

(23) *The limSlew will be updated. However, this parameter is governed by the circSlewDeg minus the original output slew of cell 3 (worst arc),*

limSlew 3 = circSlewDeg−original output slew of cell 3 *= 0.2 − 0.1 = 0.1.*

(24) *Analyzing cell 4. As this is an HVT cell, automatically its treatment will be different. So the algorithm only will be concerned with the propagation of possible delay degradations of its fan-in cone.*

(25) *Calculating circDelayDeg and circSlewDeg of cell 4. With a degraded input (it will be governed by limSlew generated and updated in cell 2), the degraded output delay and slew of cell 4 are calculated,*

circSlewDelay 4 (inputSlew, outputLoad) = 0.2.

(26) *The limSlew will be updated. However, this parameter is governed by the circSlewDeg minus the original output slew of cell 4 (worst arc),*

limSlew 4 = circSlewDeg − original output slew of cell 4 = 0.2 − 0.1 = 0.1.

(27) *Analyzing cell 5. At the input of the cell, there are the output degradation of cell I 1 (governed by its substitute) and cell 3 (governed by the propagation of its input, this cell will not be replaced).*

(28) *Choosing the worst degradation between the inputs of cell 5 for the calculation of its limSlew,*

worst degradation—*generated by cell 2 = 0.2.*

(29) *As this is an HVT cell, it will not be replaced and its treatment will be similar to the cell 4,*

circSlewDeg 5 (inpuSlew, outputLoad) = 0.3,

limSlew = circSlewDeg − original output slew of cell 5 = 0.3 − 0.2 = 0.1.

(30) *Analyzing cell 6. Even being an LVT cell, it is not part of cirCellList list. Its treatment will be different of the treatments done in the LVT circuit cells. A similar treatment to the one given to HVT cells will be done.*

(31) *With the worst limSlew between cells 3 and 4, the degradation in its input is calculated in order to calculate the circSlewDeg,*

worst degradation—generated by cell 2 or cell 4 = 0.1.

(32) *With circSlewDeg, its limSlew is created being governed by the subtraction between circSlewDeg and the original output slew of cell 4,*

circSlewDeg 4 (inputSlew, outputLoad) = 0.2,

limSlew 4 = circSlewDeg − original output slew of cell 4 = 0.2 − 0.1 = 0.1.

References

[1] N. S. Kim, T. Austin, D. Blaauw et al., "Leakage current: Moore's law meets static power," *IEEE Computer Society*, vol. 36, no. 12, pp. 68–75, 2003.

[2] International Technology Roadmap for Semiconductors (ITRS). 2008, http://public.itrs.net/.

[3] D. Lee, D. Blaauw, and D. Sylvester, "Gate oxide leakage current analysis and reduction for VLSI circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 2, pp. 155–166, 2004.

[4] J. C. Chi, H. H. Lee, S. H. Tsai, and M. C. Chi, "Gate level multiple supply voltage assignment algorithm for power optimization under timing constraint," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 6, pp. 637–648, 2007.

[5] A. G. Silva-Filho and S. M. L. Lima, "Energy consumption reduction mechanism by tuning cache configuration using nios II processor," in *Proceedings of the IEEE International SoC Conference (SOCC '08)*, pp. 291–294, Newport Beach, Calif, USA, 2008.

[6] A. G. Silva-Filho, F. R. Cordeiro, R. E. Sant'anna, and M. E. Lima, "Heuristic for two-level cache hierarchy exploration considering energy consumption and performance," in *Proceedings of the Integrated Circuit and System Design. Power and Timing Modeling, Optimization, and Simulation (PATMOS '06)*, pp. 75–83, Montpellier, France, 2006.

[7] M. Pedram and J. M. Rabaey, *Power Aware Design Methodology*, Kluwer Academic, Boston, Mass, USA, 2002.

[8] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi, *Low Power Methodology Manual for System-on-Chip Design*, Springer, London, UK, 2007.

[9] M. Ketkar and S. S. Sapatnekar, "Standby power optimization via transistor sizing and dual threshold voltage assignment," in *Proceedings of the IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 375–378, San Jose, Calif, USA, 2002.

[10] F. Brglez and H. Fujiwara, "A neural netlist of 10 combinational benchmark circuit and a target transistor in fortran," in *Proceedings of the International Symposium on Circuits and Systems*, pp. 663–398, Kyoto, Japan, 1985.

[11] D. Lee and D. Blaauw, "Static leakage reduction through simultaneous threshold voltage and state assignment," in *Proceedings of the 40th Design Automation Conference (DAC '03)*, pp. 191–194, Anaheim, Calif, USA, 2003.

[12] L. Wei, Z. Chen, K. Roy, M. C. Johnson, Y. Ye, and V. K. De, "Design and optimization of dual-threshold circuits for low-voltage low-power applications," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 7, no. 1, pp. 16–24, 1999.

[13] L. Wei, Z. Chen, K. Roy, Y. Ye, and V. De, "Mixed-Vth (MVT) CMOS circuit design methodology for low power applications," in *Proceedings of the 36th Annual ACM/IEEE Design Automation Conference*, pp. 430–435, New Orleans, La, USA, 1999.

[14] J. Jaffari and A. Afzali-Kusha, "New dual-threshold voltage assignment technique for low-power digital circuits," in *Proceedings of the 16th International Conference on Microelectronics (ICM '04)*, pp. 413–416, Tunis, Tunesia, 2004.

[15] P. Elakkumanan, K. Thyagarajan, K. Prasad, and R. Sridhar, "Optimal Vth assignment and buffer insertion for simultaneous leakage and glitch minimization though Integer Linear Programming (ILP)," in *Proceedings of the 48th Midwest Symposium on Circuit and Systems (MWSCAS '05)*, vol. 2, pp. 1880–1883, 2005.

[16] S. Gupta, J. Singh, and A. Roy, "A novel cell-based heuristic method for leakage reduction in multi-million gate VLSI designs," in *Proceedings of the 9th international Symposium on Quality Electronic Design (ISQED '08)*, pp. 526–530, San Jose, Calif, USA, 2008.

[17] M. Guiney and E. Leavitt, "An introduction to openAccess: an open source data model and API for IC design," in *Proceedings of the 2006 Asia and South Pacific Design Automation Conference*, pp. 434–436, Yokohama, Japan, 2006.