LAP: A Logic Activity Packing Methodology for Leakage Power-Tolerant FPGAs

Hassan Hassan Mohab Anis Mohamed Elmasry
Electrical and Computer Engineering Department
University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
h3hassan, manis, elmasry@vlsi.uwaterloo.ca

ABSTRACT

As FPGAs enter the nanometer regime, several modifications are needed to reduce the increasing leakage power dissipation. Hence, this work presents some modifications to the FPGAs CAD flow to mitigate leakage power dissipation through the use of multi-threshold CMOS technologies to pack and place logic blocks that exhibit similar idleness close to each other so they can be turned *off* during their idle time. The modifications are integrated into the VPR flow and tested on several FPGA benchmarks using a CMOS 0.13μm dual-V_{th} technology, resulting in an average leakage power savings of at least 20%.

Categories and Subject Descriptors

B.7 [**Integrated Circuits**]: Design Aids

General Terms

Algorithms, Design

Keywords

FPGA, Basic Logic Elements (BLE), Configurable Logic Blocks (CLB), leakage power, activity profile, packing, sleep transistor (ST).

1. INTRODUCTION AND RELATED WORK

In order to maintain the performance improvement witnessed by the semiconductor industry in the past decades, future CMOS technologies will continue to be aggressively scaled down. However, this scaling created several challenges to circuit designers, of which the most significant is subthreshold leakage. Leakage current is estimated to increase by three to five folds on average per technology generation and the total leakage power dissipation is expected to reach 50% of the total design power in the 65nm CMOS technology [1]. Traditionally, power management in FPGAs was constrained to dynamic power minimization. However, with modern FPGAs built using 90nm CMOS technologies, leakage power is forming a road block to the widespread use of FPGAs in some applications like wireless personal communication systems, which have conservative leakage power dissipation requirements.

FPGAs suffer from leakage power dissipation more seriously than ASIC designs for several reasons. Firstly, for a certain design, not all of the FPGA logic and switching resources are utilized. In fact the average percentage utilization for FPGAs is around 60% [2]. Hence, almost 40% of the FPGA is consuming leakage power without delivering useful output. Secondly, in most applications, all of the utilized part is not used for all of the time. As a matter of fact, several parts of the utilized portion of the FPGA can be producing useless output for a long time of the operation time of the FPGA. Lastly, the whole FPGA design can be inactive for a long time period, for example in wireless communication systems, the idle periods can be longer than 50% of the operational time of the design. Hence, FPGAs need to be forced into a low-power (standby) mode during their idle periods.

Although leakage power reduction is a familiar topic in the ASIC industry, however, only recently FPGA researchers started to tackle this problem [2, 3, 4, 5]. Leakage power dissipation in FPGAs can be reduced by introducing new techniques at the circuit level, architecture level, and/or at the computer-aided design (CAD) level. In this work an architectural modification for FPGAs is adopted and the changes needed at the CAD level are devised to make full advantage of the architecture in maximizing the leakage savings. The architectural modification adopted is the use of multi-threshold CMOS (MTCMOS) technology in FPGAs [2].

MTCMOS has proven its success in reducing both the active and standby leakage in the ASIC domain by employing a high-V_{th} (HVT) sleep transistor (ST) to connect the pull-down network of a circuit to the ground, as shown in Figure 1. However, STs introduce a speed penalty due to the added resistance to the ground [6, 7]. In FPGAs, STs can reduce leakage by: (i) powering down the unutilized part of the chip, (ii) dynamically turning *on/off* the utilized parts of the chip depending on their activity, (iii) powering down all of the FPGA during its idle time, and (iv) reducing leakage current due to the stacking effect. In [8], a test FPGA tile based on MTCMOS architecture was fabricated in a CMOS 0.13μm technology to prove the applicability of the use of STs in FPGAs.

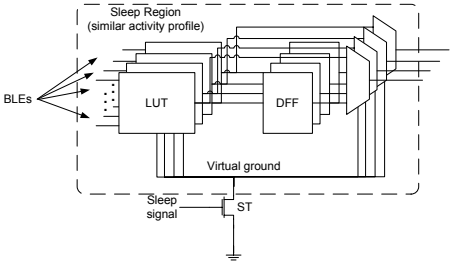

Figure 1: FPGA architecture using STs.

In [2], the authors did not provide a method to automatically identify these logic blocks that need to placed in the standby mode. They resorted to a manual step, which in turn is inappropriate to be included in a CAD tool. Moreover, in [2], the blocks that are turned *on/off* together were placed close to each other, irrespective of their connections. This approach can adversely affect the delays along the critical paths of the design.

Hence, this work proposes several modifications that need to be done to the CAD flow of FPGAs to safely and effectively employ MTCMOS techniques and are integrated into the VPR flow [9]. A flowchart of a typical CAD flow is shown in Figure 2(a) and a flowchart of the proposed modifications is shown in Figure 2(b). An activity generation phase, called Logic Activity Packing (LAP), is added to identify the BLEs that exhibit similar activity (thus will be forced into a standby mode together) and is explained in Section 4. A processing algorithm for the discharge current estimation through the STs is explained in Section 5. A modified T-VPack [9, 10] algorithm, called Activity TV-Pack (AT-VPack), is proposed in Section 6. Some modifications to the flexible power model [11] are introduced to account for the changes in the FPGA architecture are presented in Section 7. Finally, experimental results and discussions are presented in Section 8.

ISLPED'05, August 8–10, 2005, San Diego, California, USA

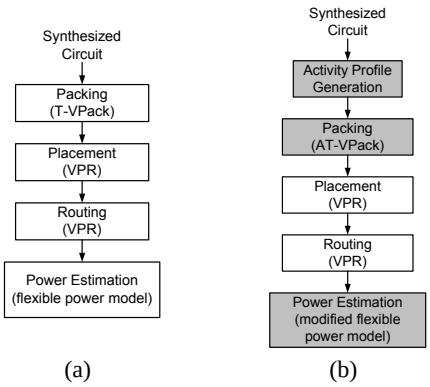

Figure 2: FPGA CAD flowchart. (a) Conventional flowchart. (b) Proposed flowchart.

2. TARGETED FPGA ARCHITECTURE

The targeted FPGA architecture is shown in Figure 1. Each BLE consists of a 4-input LUT, a flip-flop and a 2:1 multiplexer. Several BLEs are grouped together to form a CLB. Every n CLBs are connected to the ground through one ST to reduce leakage current and force the n CLBs into low-power modes during their inactive periods, as shown in Figure 1. The ST is controlled using a SLEEP signal to its gate, which is generated dynamically during the device operation using the partial reconfiguration property of modern FPGAs [12, 13] to avoid adding extra hardware. Moreover, in each CLB, the latches are used to retain the value of the BLEs outputs when they enter the sleep mode. The logic blocks served by one sleep transistor are called the sleep region. The size of the sleep region is controlled by; the maximum allowable size for the ST, hence, the peak current this transistor will hold, the maximum performance loss allowed due to the ST, and the maximum permitted ground bounce in the virtual ground $VGND$ lines. A diagram of the proposed FPGA fabric is shown in Figure 3, where it can be seen that the STs are prefabricated and hardwired to their corresponding SLEEP signals and virtual ground $VGND$ rails. Hardwiring the SLEEP signals and $VGND$ rails reduces the complexity of the routing stage of the CAD design flow.

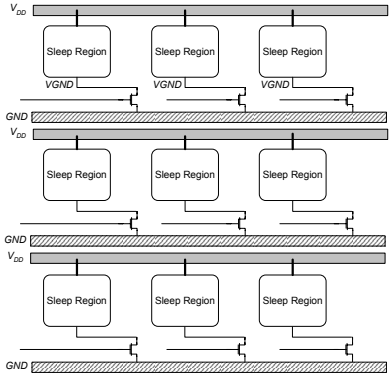

Figure 3: MTCMOS-based FPGA fabric.

In this work, a local ST architecture is adopted because of its several advantages. Firstly, the $VGND$ rails are treated as local connections, hence, there is no need to fabricate them using wide metal lines like V_{DD} and GND rails. Secondly, the routing complexity of the $VGND$ rails is reduced significantly in local ST architectures than global architecture. Lastly, local STs provide less routing overhead in terms of the criticality of the sleep signals, better noise margins, and higher turn *off* flexibility, thus higher power savings [8].

3. SLEEP TRANSISTORS SIZING

The proper selection of the ST size used in the FPGA is crucial to the success of the proposed low-leakage algorithm without incurring large penalties to the performance. To reduce the delay penalty, the size of the ST should be as large as possible. However, a large ST dictates a large area penalty and larger leakage current and dynamic power dissipation during the switching of the ST. In order to minimize the performance loss due to the use of STs while retaining sufficient leakage savings, 5% is set as the maximum allowable performance loss in this work. Hence, the size of the ST is expressed as [7]

$$\left(\frac{W}{L}\right)_{\text{sleep}} = \frac{I_{\text{sleep}}}{0.05\mu_n C_{ox}(V_{DD} - V_{th_L})(V_{DD} - V_{th_H})} , \quad (1)$$

where $\left(\frac{W}{L}\right)_{\text{sleep}}$ is the aspect ratio of the ST, I_{sleep} is the maximum discharge current the ST can hold, μ_n is the electrons mobility, C_{ox} is the MOS oxide capacitance, and V_{th_L} and V_{th_H} are the threshold voltage of the LVT and HVT devices, respectively.

4. ACTIVITY GENERATION

An *activity profile* is a representation of the periods that a BLE is active (switching). If a group of BLEs are expected to switch in the same time periods, then it is said that they have similar activity profiles. In order to maximize the power savings from the use of STs, BLEs with similar activity profiles are packed together to be served by one ST. The main goal of the activity generation is to identify the BLEs that have a similar activity profile so that the packing algorithm can cluster them together. By the end of this phase, all the BLEs in the design are given labels to divide them into several *activity regions* according to their activity profiles. The algorithm proposed for activity generation is called Logic Activity Packing (LAP). The LAP algorithm depends on the representation of the activities of each BLE in the design as a binary sequence. In order to properly explain this algorithm, several definitions and notations will be first explained.

4.1 Activity Vector

Definition: Activity Vector

Given a net x in a circuit netlist, the *activity vector* A_x of x is defined as:

$$A_x = [\ a_1 \quad a_2 \quad a_3 \quad .... \quad a_{2^n-1} \quad a_{2^n}]^T \quad , \quad (2)$$

where n is the total number of inputs to the circuit, a_i is a binary variable that is '1' if any of the outputs of the circuit depend on net x for evaluation when the inputs to the circuit are given by the i^{th} input vector, and T represents the transpose of the vector.

In FPGAs, each BLE has only one output; thus, the activity vector of each net resolves to be the activity vector of the BLE driving that net.

For the circuit in Figure 4, blocks F and G must be *on* to generate the outputs of the circuit f and g, respectively. Consequently, the activity vectors A_f and A_g for blocks F and G, respectively, are given by

$$A_f = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]^T ,$$
$$A_g = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\ ]^T . \quad (3)$$

On the other hand, for computing the activity vector at the inputs of block

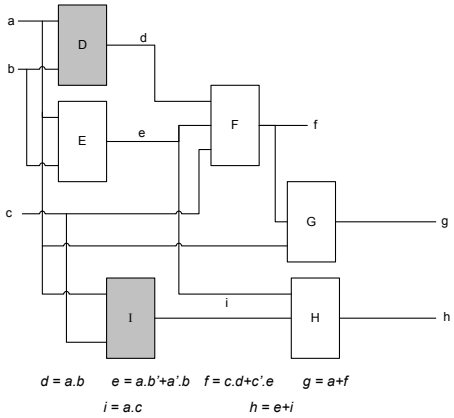

Figure 4: An example of a circuit.

F, it is noteworthy that block D will be only used to generate the output signal f if the input c is '1'. Similarly, block E is only used when c is '0'. Hence, the activity vectors for D and E, when f is evaluated, are represented by

$$A_d = [\ 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1\ ]^T ,$$
$$A_e|_f = [\ 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0\ ]^T . \quad (4)$$

However, to evaluate h, E will have the following activity vector:

$$A_e|_h = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0\ ]^T , \quad (5)$$

which differs from the one given in (4). Hence, the resulting A_e is given by

$$A_e = A_e|_1 + A_e|_2 = [\ 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0\ ]^T\ .$$

Finally, the activity vector for i is given by

$$A_i = [\ 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1\ ]^T\ . \tag{6}$$

From this discussion, it can be deduced that if F, G, and H are active for all the input combinations, packing them together will result in improved results. Moreover, E will be active for almost all of the input combinations except for only one, thus it can also be packed with F, G, and H in the same cluster. Therefore, the cluster containing E, F, G, and H will be always *on*. On the other hand, D and I have similar activity profiles for half of the input combinations, thus it will be a good strategy to group them together and turn *off* this cluster for half of the circuit operational time.

4.2 Hamming Distance and Weighted Hamming Distance

Definition 2: Hamming Distance
Given two binary sequences of length n; A_n and B_n, the *Hamming distance* $d_{(a,b)}$ between these two sequences is defined as

$$d_{(a,b)} = \sum_{k=0}^{n-1} |a_k - b_k|\ , \tag{7}$$

where a_k and b_k are the k^{th} elements of A_n and B_n, respectively.

From (7), the Hamming distances between the activity vectors given in (3) to (6) are written as

$$\begin{array}{llll} d_{(f,g)} = 0 & d_{(f,d)} = 4 & d_{(f,e)} = 1 & d_{(f,i)} = 4 \\ d_{(f,h)} = 0 & d_{(g,d)} = 4 & d_{(g,e)} = 1 & d_{(g,i)} = 4 \\ d_{(g,h)} = 0 & d_{(e,d)} = 5 & d_{(e,i)} = 5 & d_{(e,h)} = 1 \\ d_{(d,i)} = 4 & d_{(d,h)} = 4 & d_{(i,h)} = 4 & \end{array} \tag{8}$$

From (8), it is seen that the Hamming distance between the activity vectors of any two CLBs is a measure of the correlation between the activity profiles of the CLBs. A Hamming distance close to the absolute minimum of zero, indicates that the two blocks will exhibit the same activity profile, thus when positioned together in the same cluster will result in maximum power savings and vice versa. This is verified by examining the values in (8) and the results stated in the previous sub-section. However, the Hamming distance between the activity vectors of two CLBs does not take into consideration the probability of occurrence of the different input combinations. As a result, the quality of the results can be notably affected, especially for large circuits.

Definition 3: Weighted Hamming Distance
Given two binary sequences of length n; A_n and B_n, and a weighting vector W_n, the *weighted Hamming distance* $dw_{(a,b)}$ between these two sequences is defined as

$$dw_{(a,b)} = \sum_{k=0}^{n-1} w_k \times |a_k - b_k|\ , \tag{9}$$

where a_k, b_k, and w_k are the k^{th} elements of A_n, B_n, and W_n, respectively.

The weighting Hamming distance is an efficient way to incorporate the various probabilities of the input combinations into the algorithm. Hence, in this work, the weighted Hamming distance is used to group the different BLEs into activity regions.

4.3 The LAP Algorithm Operation

The LAP algorithm consists of two main phases: activity vector generation and activity labeling. The activity vector generation phase exhaustively simulates the circuit by iterating all the input vectors and finding the values of all the circuit nets resulting from that input vector. Afterwards, for each input vector iteration, each signal (or block) is tested to investigate whether or not it affects the evaluation of the circuit outputs. This is performed by complementing the value of the signal under consideration and then proceeding from that point towards the circuit outputs. If the output of BLEs that have this net as an input will change, then this change is taken to the next circuit level, otherwise, a '0' is placed in the corresponding row of the input vector. If a loop is found, then this net is given '1' in its activity vector for that input combination. It should be noted that the number of levels checked from the net under consideration increases the computational time significantly. In order to limit this computational complexity, the number of levels to be checked is limited to 3. After exhaustively generating all the activity vectors for all the circuit nets, the static probability of each net is calculated.

The next step is the calculation of the Hamming distance between each two BLEs in the design. This is performed recursively through all the design elements. Using the static probability of each net, the weighted Hamming distance dw between every two BLEs is then calculated. At this point, the activity labels can be assigned according to the weighted Hamming distance. However, this approach can result in performance deterioration as the connections between the different BLEs is not considered. Since those BLEs that will have a similar activity label are expected to be placed in the same sleep region, *i.e.*, will be placed close to each other. Hence, it seems that the wire length should be included in the activity labeling as well. Since at this stage the algorithm does not have any information about where each block will be placed, an approximation for the wire length is adopted. If any two BLEs share one net, then the distance between them l is considered as zero, *e.g.*, E and F in Figure 4. If there is one level of BLEs in between the two BLEs, then the distance is considered as 1, and so on.

Furthermore, the use of the weighted Hamming distance is not a sufficient measure for the difference in activity between the different BLEs. If for example, there are 2 BLEs with a weighted Hamming distance between them of 1. However, this net at which they differ is a very active one that keeps on toggling all of the time. This will mean that the ST will keep on turning on and off all of the time thus consuming most of the savings in leakage through dynamic power dissipation. In order to avoid such condition, the *transition density* [14] of the net is considered while calculating the Hamming distance. The transition density D is defined as the average number of transitions per unit time. Multiplying the transition density with the weighted Hamming distance results in the transition weighted Hamming distance ($\overline{dw} = dw \times D$).

In order to combine the transition weighted Hamming distance and the distance between BLEs, BLEs are assigned activity labels by minimizing the cost function given below

$$\min\{\overline{dw} + \delta \times l\}\ , \tag{10}$$

where δ is a normalization constant selected to be 0.5. To avoid having activity regions with a large number of BLEs, which will decrease the leakage savings, the size of the activity region is limited to 1.5 times the longest path from input to output in the circuit. This value was obtained by running the algorithm on several benchmarks. Assigning a constant value for activity region size, irrespective of the circuit size, results in impractical results. Increasing it than 1.5 times the longest path in the circuit results in having excessively large activity regions that are usually not fully filled up by the algorithm. On the other hand, decreasing the activity region size increases the number of activity regions in the final design.

Hence, the algorithm starts to greedily assign activity labels to the BLEs according to (10) until the maximum activity region size is reached. Afterwards, a new BLE is selected as a seed cell for a new activity region and the procedure is repeated. A pseudocode of the algorithm is listed in Figure 5.

```
for all the input combinations
   for all the nets in the circuit
     find the value of the net
   end for
   for each net in the circuit
     toggle the value of the net
     Activity[input_vector][net] = 0
     proceed with the new value of the net
     if the value of any output changes
       Activity[input_vector][net] = 1
     end if
   end for
end for
for each net in the circuit
   find the static probability
   find the transition density
   find the distance to each net in the circuit
end for
```

Figure 5: Pseudocode of LAP.

5. DISCHARGE CURRENT FEASIBILITY AND PROCESSING

Before deciding whether it is possible to add a certain BLE to any sleep region or not, the discharge current feasibility should be checked. In this work, the term *current feasibility* is used to denote whether by adding a certain BLE to a sleep region, the sum of discharge currents in the sleep region

at any instant will exceed I_{sleep} or not. The current feasibility check consists of a topological sorting of the BLEs in the sleep region so far as well as the one under consideration. Afterwards, the maximum of the resulting current vector I_{max} is checked against the value of I_{sleep}. If I_{max} is less than I_{sleep}, then the current feasibility test is valid, otherwise, it is infeasible to add this BLE to the current sleep region.

5.1 Pre-Processing of Discharge Currents

The value of the instantaneous discharge current of a certain BLE in a sleep region depends on the connections between this particular BLE and the other BLEs in the sleep region and the fanout of the BLE. When the output of a certain BLE A is connected to the input of another BLE B, as shown in Figure 6(a), if the output of both A goes low, then the discharge of B is not expected to start before that of A. As a matter of fact, B will start discharging after the discharge current of A reaches its maximum value. The discharge current pattern of each BLE can be linearly approximated using triangles, as shown in Figure 6(b), where the x-axis is the time at which a certain current value (the y-axis value) occurs. t_{max} is the value at which the peak of the discharge current is expected. Hence, the sum of discharge currents in any sleep region can be represented by the trapezoid approximation shown in Figure 6. Consequently, I_{sleep} can be designed to accommodate only the peak current of the trapezoid approximation rather than the sum of peak currents of all the BLEs in the sleep region.

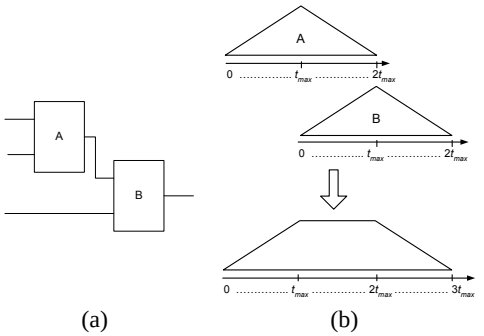

Figure 6: Linear vector approximation of discharge current and vector summation.

Moreover, the discharge current patterns, duration and maximum value, depends heavily on the load capacitance. However, the loading effect of BLEs is mainly due to the programmable routing resources, which to some extent is fixed. In order to account for the discharge pattern of the BLEs that are employed in the design, the BLEs are simulated in a 0.13μm CMOS process using HSpice and the shape of the discharge pattern, in terms of instantaneous values and duration, is recorded as a *current vector*, where each element is the discharge current value with a step of 5ps as in [7].

The value of I_{sleep} controls the maximum number of BLEs that can be placed in a single sleep region. Increasing the value of I_{sleep}, gives the packing algorithm more freedom to fill out all of the used sleep regions, but at the same time necessitates the use of larger STs, as noticed from (1), which in turn leads to larger leakage current and dynamic power dissipation during the switching of STs, as well as, an increase in the ground bounce on the virtual ground lines. On the other hand, a small value for I_{sleep} limits the margin given to the packing algorithm in packing BLEs in each sleep region, resulting in a large number of partially filled sleep regions. Consequently, a larger number of sleep signals will be employed in the design, leading to an increase in both power dissipation and complexity of the sleep signal generation procedure, and large leakage power dissipation in the unused CLBs located inside the used CLBs. For a sleep region of size 4 BLEs, this optimum value of I_{sleep} is found to be twice the maximum discharge current of one BLE. The same experiment is repeated for other sizes of the sleep region and the corresponding values of I_{sleep} are evaluated in terms of the maximum discharge current of one BLE.

5.2 Topological Sorting

In this phase the topology of the BLEs already in the sleep region is extracted to be used in properly adjusting and adding the current vectors of each BLE. There are three main cases that can be encountered while performing topology sorting: a combinational sleep region where each BLE shares at least one net with any other BLE in the sleep region (Figure 7(a)), a combinational sleep region with at least one BLE does not share any net with any other BLE in the sleep region (Figure 7(c)), or a sequential sleep region that contain one or more loops (Figure 7(c)).

For a combinational connected sleep region, the algorithm starts by converting the BLEs inside the sleep region into an undirected graph, the graph

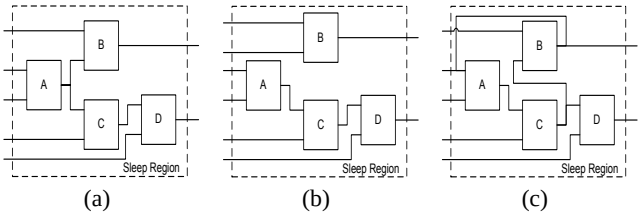

Figure 7: Different types of sleep regions.

in Figure 8(a) is equivalent to the sleep region in Figure 7(a). Afterwards, a topological sort for the resulting graph is used to find the relations between all the BLEs in the sleep region by converting the graph to a hierarchal data structure. An example of the topological sorting procedure is shown in Figure 8, where A is found as the parent node, B and C are ordered in the same level, and D in the last level. The triangular approximation for the discharge current for the BLEs in the sleep region is shown in Figure 8(e) as well as the resulting sum of the discharge current vectors. It can be noticed from Figure 8(e) that since nodes B and C are ordered in the same level, they are both expected to start discharging as soon as the discharge of A reaches its maximum, hence, their discharge currents start simultaneously.

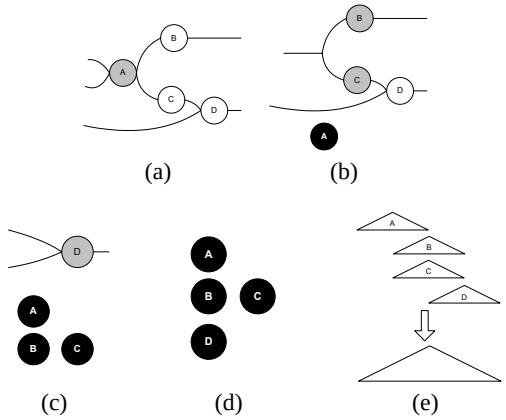

Figure 8: Steps of the current feasibility check for a combinational connected sleep region. (a) A is selected to be deleted, (b) A is ordered in the first position and B and C are selected for deletion, (c) B and C are ordered in the same position, (d) Final ordering, (e) Current vectors summation

If the graph contains unrelated nodes, as shown in Figure 9, instead of using the triangular approximation as discussed before, the discharge current is assumed to be constant and equal to the peak value for the unconnected BLE because it is difficult to predict when the unrelated node is expected to discharge. The unrelated or unconnected node is identified only during the first iteration of the algorithm. Figure 9(a) represents the graphical representation of the sleep region in Figure 7(b), nodes B is identified as an unrelated node. The algorithm then continues as the previous case to sort the rest of the graph. Thereafter, the current vector of the unrelated node B is represented as a rectangle with width equal to the sum of widths of the other vectors and added to the rest of the currents, as shown in Figure 9(e).

The last case is when the graph contains one or more loops. Having a loop in the graph makes the topological ordering infeasible, hence, a loop has to be detected before starting the topological sorting algorithm. Thus, before the topological sorting phase, loop detection is employed on the sleep region graph, if a loop is found, then a loop resolving algorithm is used. The presence of loops does not change the value of the peak current of the sleep region, it only affects the shape of the discharge current pattern by breaking the loop at any point while keeping a virtual edge to represent the broken edge.

6. PACKING ALGORITHM

Modern island-style FPGAs have a hierarchal architecture, where several BLEs are packed together to form clusters (CLBs). The main aim of the available packing algorithms is to minimize the total area (by packing clusters to their full capacity), minimize the delay (by packing LUTs on a certain critical path together [15]), and/or maximize routability (by minimizing the number of inputs to each cluster). However, the goal of minimizing power dissipation, either dynamic or leakage power dissipation, has

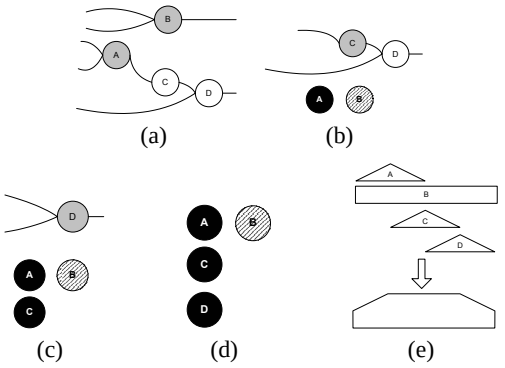

Figure 9: (a) Both A and B are selected for deletion, (b) A and B are ordered on the same position, with B an unrelated node and C is marked for deletion, (c) C is ordered in the next position, (d) Final ordering, (e) Current vectors summation.

been rarely addressed. In this work, the activity profiles obtained in Section 4 are incorporated into the T-VPack [15] algorithm to pack BLEs to minimize leakage power dissipation.

6.1 T-VPack

In T-VPack, LUTs are packed one at a time into clusters, while satisfying two main hard constraints; (i) the number of BLEs in the cluster must be less than the cluster size and (ii) the number of input nets needed by the BLEs in the cluster and generated outside the cluster must be less than or equal to the number of cluster inputs. The original T-VPack tries to fill the clusters to their full capacity while minimizing a cost function that includes the delay across the critical path and wire length. This is performed by calculating an attraction force, $Attraction()$ between each unclustered BLE and the cluster under investigation. $Attraction()$ is calculated based on a compromise between the criticality of the nets connected to the BLE and the number of connections that the BLE shares with the BLEs previously clustered in the cluster. The BLE with the maximum $Attraction()$ is added to the cluster. Afterwards, a hill climbing procedure is called to ensure that the cluster is full to its maximum capacity. This continues on until all the BLEs in the circuit are clustered.

6.2 AT-VPack

In this work, the T-VPack algorithm is modified to include activity profiles, thus the modified T-VPack is called Activity T-VPack (AT-VPack). In AT-VPack, a set of BLEs are selected as candidates to be added to the cluster under investigation. The selection criteria for these candidate BLEs are: (iii) the combined discharge current of the BLEs inside the cluster plus the BLE to be added does not exceed I_{sleep} and (iv) the activity label of the BLE to be added is the same as that of the BLEs inside the cluster. From the pool of candidate BLEs, the one that maximizes $Attraction$ and satisfies (i) and (ii) is selected to be added to the cluster. If AT-VPack fails to fill out all of the spaces in the cluster, the hill-climbing approach used in the original T-VPack is invoked to start filling the vacant places while satisfying both (iii) and (iv).

Unlike T-VPack, AT-VPack might still be unable to fill the cluster to its maximum capacity due to the additional two constraints (iii) and (iv). Hence, a second hill-climbing stage is used that employs simulated annealing to swap the BLEs in the cluster with other candidate BLEs that have not been clustered yet and then try to fill the cluster. If the set of BLEs currently in the cluster is given by A and the set of BLEs that had not been clustered is called B, the algorithm swaps block i from set A with block j from set B while satisfying constraint (iii) using the following cost function

$$\min \quad \alpha \ \kappa \ \ Attraction(A_i) - Attraction(B_j) + (1-\kappa)\frac{\text{number of vacant places}}{\text{total number of places}} \quad , \quad (11)$$

where α is a variable that represents the transition weighted Hamming distance between A and B_j ($\alpha = 1 + \overline{dw}$) and κ is a weighting constant ($0 \leqslant \kappa \leqslant 1$) that is used to give importance to either filling up the cluster with any blocks or to consider the attraction force. A small value of κ would result in a faster filling for the cluster, while a decrease in the $Attraction()$ can be tolerated, while a large value will keep the decrease in $Attraction()$ to a minimum and accepting partially filled clusters. By performing several experiments using AT-VPack for different FPGA benchmarks, it is found that the best value for κ is 0.5. The value chosen for α forces the algorithm to start looking at first for blocks with the same activity as the cluster before looking for blocks with other activities. Even when it does look for BLEs with different activities, it always searches for those with close activity profiles. This ensures maximum leakage savings (clusters with blocks that have different activity profiles will be *on* for a longer period).

Moreover, the cost function in (11) minimizes the loss in the quality of the solution, in terms of the attraction force, by minimizing the difference between $Attraction(A_i)$ and $Attraction(B_j)$. Similarly, the current constraint is kept as a hard constraint throughout this hill-climbing stage. By the end of this hill-climbing stage, the cluster is full to its maximum capacity. A pseudocode for AT-VPACK algorithm is listed in Figure 10.

```
Perform T-VPACK with 2 extra constraints:
   max I_cluster ⩽ I_sleep
   activity_blocks in cluster is constant
while there are empty spaces in the cluster
   for all unclustered blocks
      find blocks i and j with min cost (equation(11))
      add i to the cluster
      if max I_cluster > I_sleep
         remove i
      end if
   end for
end while
```

Figure 10: Pseudocode for the modified T-VPACK algorithm

7. POWER ESTIMATION

In order to evaluate the performance of the proposed algorithms, the power dissipation in the placed and routed design is compared to that of the same benchmark without STs. The flexible power model proposed in [11], which calculates dynamic, short-circuit, and leakage power, is used to estimate the power dissipation in the design without STs. In order to measure the power dissipation in the design with STs, several modifications are proposed to the power model. The original power model starts by calculating the switching activity along the nets in the design and then uses this information to calculate the dynamic power dissipation across the design. Afterwards, the short-circuit is approximated as 10% of the dynamic power dissipation. Finally, the leakage power is calculated in all of the design.

When a circuit is idle, it only consumes leakage power. Hence, to include the possibility that the whole circuit can be forced into the standby mode during its idle time, the total power dissipation P_t is expressed as

$$P_t = t_{on} \times P_{on} + t_{off} \times P_{idle} \ , \quad (12)$$

where t_{on} and t_{off} are the percentages of *on* and *off* times of the FPGA and P_{on} and P_{idle} are the power dissipation during the active and idle modes of operation of the FPGA. P_{on} is expressed as

$$P_{on} = [P_{dyn} + P_{sckt} + P_{leak}]_{utilized} + P_{leak}|_{unutilized} \quad (13)$$

where P_{dyn}, P_{sckt}, and P_{leak} are the dynamic, short-circuit, and active leakage power dissipations, respectively, in the utilized portion of the FPGA, while $P_{leak}|_{unutilized}$ is the standby leakage in the unutilized CLBs. On the other hand, P_{idle} is the standby leakage for the whole FPGA.

In order to estimate the power dissipation of the placed and routed design using the proposed low-leakage algorithm, several changes are made to the original power model. (a) The method used to calculate the leakage current is changed to reflect the presence of a sleep transistor by only calculating the leakage due to the sleep transistor. (b) The leakage in the unused CLBs is calculated. (c) The percentage of short-circuit power dissipation is increased to 15% to account for the increased rise/fall times of the BLEs with STs. The 15% approximation was evaluated by simulating a BLE with and without an ST using HSPICE. (d) The dynamic power in the switching of the ST is calculated and added to the total power dissipation.

8. RESULTS AND DISCUSSIONS

The activity generation algorithm and the AT-VPACK discussed in Section 6 are integrated into VPR and the modified power model is used to estimate the power savings in the final design. The proposed algorithms are tested on several FPGA benchmarks to assess their capability in minimizing both standby and active leakage power dissipation. The experiments are conducted on a 900MHz Ultra Sparc III machine with 5Gbytes RAM, and the results are summarized in Table 1 for a sleep region of size 4 BLEs. It

should also be noted that the maximum allowable performance loss due to STs in all of the benchmarks is kept less than 5% (Section 3). The third column in Table 1 lists the number of resulting clusters and the minimum FPGA array that can be used to map the circuit, *i.e.*, maximum utilization percentage. For simplicity, the case where the design is mapped onto the minimum FPGA array is called 100% utilization. The power dissipated by each design is calculated using the modified power model discussed in Section 7 and the percentages savings in leakage are listed in Table 1.

Table 1: Experimental results for FPGA benchmarks

Circuit	# of BLEs	# of Clusters	% of Unused CLBs	% Saving in Leakage (100% on time)
alu4	1522	382 (20×20)	4.5	22.9
apex2	1878	472 (22×22)	2.48	20.7
apex4	1262	317 (18×18)	2.16	19.1
bigkey	1707	429 (21×21)	2.72	20.2
clma	8381	2100 (46×46)	0.76	18.9
des	1591	399 (20×20)	0.25	16.8
diffeq	1494	376 (20×20)	6	22
dsip	1370	344 (19×19)	4.7	21.2
elliptic	3602	904 (31×31)	5.9	21.6
ex1010	4598	1153 (34×34)	0.26	18.7
ex5p	1064	269 (17×17)	6.92	22.8
frisc	3539	886 (30×30)	1.56	18.2
misex3	1397	353 (19×19)	2.21	20.9
pdc	4575	1147 (34×34)	0.78	17.7
s298	1930	485 (23×23)	8.32	28.3
s38417	4096	1028 (33×33)	5.6	25.4
s38584.1	6281	1575 (40×40)	1.56	18.9
seq	1750	441 (21×21)	0	14.2
spla	3690	926 (31×31)	3.6	18.3
tseng	1046	265 (17×17)	8.3	27

In each benchmark, the power savings consist of two parts; savings from permanently turning *off* all the unused clusters and savings from dynamically turning *on* and *off* the used clusters in the design depending on their activity profile. By taking a look at the results for the 'seq' benchmark in Table 1, this benchmark has no unused CLBs while the leakage savings achieved is 14.2%. This saving is entirely from dynamically turning *on* and *off* the different used clusters in the design depending on their activity profile and the stacking effect of the ST.

On the other hand, the 's298' benchmark has the maximum percentage of unused blocks among all of the benchmarks and it resulted in the maximum leakage savings (28.3%). The leakage savings from permanently turning off the unused clusters in this benchmarks is 14.5%. The average leakage savings due to dynamically turning *on* and *off* the used CLBs among all of the benchmarks tested is 14.3%, while that due to turning *off* the unused blocks is proportional to the number of unused blocks. The average leakage savings across all of the benchmarks for the 100% on time case is found to be 20.68%.

Practically, the utilization percentage is less than the maximum utilization assumption used in finding the results in Table 1. Typically, the utilization in FPGAs ranges from 80% to 60% [2]. Moreover, the 100% on time assumption made earlier is impractically high. The average on time of most applications is around 50% to 20% for some hand-held applications [6]. Hence, the same benchmarks are tested again using utilization percentages of 80% and 60% and on times of 100%, 50%, and 20%. The average leakage savings among all of the benchmarks in each of these cases is plotted in Figure 11. From Figure 11, it can be noticed that the average leakage power savings increases by about 46% when the device idle time increases from 0 to 50%. Moreover, Decreasing the utilization from 100% to 80%, increases the leakage power savings by about 39%.

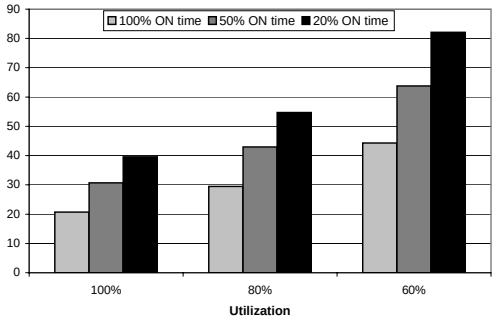

Figure 11: Percentage savings in leakage.

In another experiment, several sizes for the sleep region are tested for different CLB sizes. The size of the CLB is changed from 3 to 6 BLEs and the size of the sleep region is changed from 1 to 5 CLBs. The leakage savings in each of these experiments are recorded and plotted in Figure 12. From Figure 12, it can be noticed that for each CLB size, there is an optimum sleep region size. Moreover, leakage savings is always maximum for sleep regions of size around 8 BLEs. This proves the fact stated earlier that too large (will require a large ST, which results in large standby leakage and dynamic power dissipation in the ST) and too small (will result in partially unfilled clusters, which will increase the area and decrease the number of permanently *off* sleep regions, hence, increases the total leakage power) sleep regions will result in lower leakage savings.

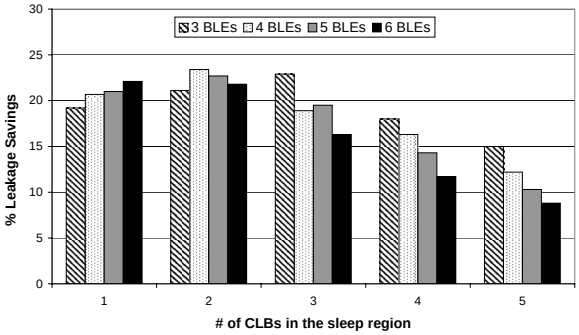

Figure 12: Impact of the sleep region size on the leakage savings.

In order to find the execution time penalty of the proposed packing algorithm compared to the original VPR algorithm, the same benchmarks are packed and placed using VPR on the same machine and the execution time is recorded. The average increase in execution time is found to be 37%, which can be reduced by employing heuristics to find the activity vectors.

9. CONCLUSIONS

This work presented a CAD technique that can be used to minimize leakage power dissipation in MTCMOS based FPGAs. The methodology is tested using a 0.13μm dual-V_{th} CMOS technology, resulting in an average power savings of 20.68% for a 100% ON time FPGA. Moreover, the optimum size for the sleep region is found to be 8 BLEs for maximum leakage savings.

10. REFERENCES

[1] The International Technology Roadmap for Semiconductors website. [Online]. Available: http://public.itrs.net

[2] A. Gayasen *et al.*, "Reducing Leakage Energy in FPGAs Using Region-Constrianed Placement," in *Proc. of ISFPGA*, 2004, pp. 51–58.

[3] A. Rahman *et al.*, "Evaluation of Low-Leakage Design Techniques for Field Programmable Gate Arrays," in *Proc. of ISFPGA*, 2004, pp. 23–30.

[4] J. Anderson *et al.*, "Active Leakage Power Optimization for FPGAs," in *Proc. of ISFPGA*, 2004, pp. 33–41.

[5] F. Li *et al.*, "Low-Power FPGA Using Pre-defined Dual-Vdd/Dual-Vt Fabrics," in *Proc. of ISFPGA*, 2004, pp. 42–50.

[6] J. Kao *et al.*, "Dual-Threshold Voltage Techniques for Low-Power Digital Circuits," *IEEE JSSC*, vol. 35, no. 7, pp. 1009–1018, July 2000.

[7] M. Anis *et al.*, "Design and Optimization of Multithreshold CMOS (MTCMOS) Circuits," *IEEE Trans. CAD*, vol. 22, no. 10, pp. 1324–1342, October 2003.

[8] B. Calhoun *et al.*, "A Leakage Reduction Methodology for Distributed MTCMOS," *IEEE JSSC*, vol. 39, no. 5, pp. 818–826, May 2004.

[9] A. Marquardt *et al.*, "Timing-driven placement for FPGAs," in *Proc. of ISFPGA*, 2000, pp. 203–213.

[10] V. Betz *et al.*, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer Academic Publishers, 1999.

[11] K. Poon *et al.*, "A Flexible Power Model for FPGAs," in *Proc. of Intl. Conf. on FPGAs*, 2002, pp. 312–321.

[12] Altera. Stratix Device Handbook, Volume 1. [Online]. Available: http://www.altera.com/literature/hb/stx/stratix_handbook.pdf

[13] Xilinx. Two Flows for Partial Reconfiguration: Module Based or Difference Based. [Online]. Available: http://direct.xilinx.com/bvdocs/publications/xapp290.pdf

[14] F. Najm, "Transition Density, a Stochastic Measure of Activity in Digital Circuits," in *Proc. of DAC*, 1991, pp. 644–649.

[15] A. Marquardt *et al.*, "Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density," in *Proc. of ISFPGA*, 1999, pp. 37–46.