A Timing-Driven Algorithm for Leakage Reduction in MTCMOS FPGAs

Hassan Hassan Mohab Anis Mohamed Elmasry

Electrical and Computer Engineering Department
University of Waterloo
Waterloo, ON N2L 3G1
e-mail: h3hassan, manis, elmasry@vlsi.uwaterloo.ca

Abstract— A timing-driven MTCMOS (T-MTCMOS) CAD methodology is proposed for subthreshold leakage power reduction in nanometer FPGAs. The methodology uses the circuit timing information to tune the performance penalty due to sleep transistors according to the path delays, achieving an average leakage reduction of 44.36% when applied to FPGA benchmarks using a CMOS 0.13μm process. Moreover, the methodology is applied to several FPGA architectures and CMOS technologies.

I. Introduction

The exponential increase in leakage power dissipation in FPGA designs motivated several research projects targeting leakage power reduction [1, 2, 3, 4, 5, 6]. In [2, 5, 6], multi-threshold CMOS (MTCMOS) techniques, which are widely used in ASIC designs [7], are used for leakage reduction by cutting off the path to ground when the circuit is idle using a sleep transistor (ST). The authors of [2] proposed to use STs to turn off the unused parts of the FPGA while dynamically turning *on* and *off* the used parts depending on their activities. However, they did not propose a methodology to identify those logic blocks that can be turned *off* collectively at the same time. However, in [5], activity profiles are used to identify the logic blocks that exhibit similar idleness periods. These logic blocks are packed together and controlled by one ST to turn them off during their common idleness periods. However, the algorithm suffered from exponential complexity with the number of circuit inputs. The strategy used in both [2, 5] to size the ST adds a minimum of 5% speed penalty along all the circuit paths due to the added resistance of the ST.

This work proposes a timing-driven MTCMOS (T-MTCMOS) CAD methodology for subthreshold leakage power reduction in FPGAs using a modified version of the activity profile generation algorithm proposed in [5]. The main contribution of this work are the use of the circuit timing information to develop a new low-leakage packing algorithm, which is integrated into the VPR CAD flow [8], that can provide a significant reduction in subthreshold leakage power dissipation, when compared to those achieved by [5], while having the least impact on the critical path delay. Moreover, a less complex version of the activity generation algorithm than that in [5] is proposed. Unlike previous MTCMOS algorithms which used constant speed penalty for the whole circuit, the timing information is used to modulate the delay penalty along the circuit paths depending on the path criticality. To the best of the authors' knowledge, this is the first work that employs timing information to tune the speed penalty of STs in MTCMOS techniques.

In the targeted FPGA architecture, the logic blocks that exhibit similar idle periods are packed together and connected to the ground using one high-V_{th} ST, as shown in Fig 1. The latches inside each logic block are used to retain their outputs values when they enter the sleep mode. The logic blocks connected to one ST are called the *sleep region*. In order to minimize the area overhead, the sleep signals are generated dynamically using the dynamic reconfiguration parts available in modern FPGAs [2]. Hence, the main area overhead is due to the STs and the sleep signals interconnects, and is around 5-8% [6, 9]. Several methodologies have been proposed to predict the sequence of inputs to the circuit and hence develop a methodology for generating the sleep signals [10].

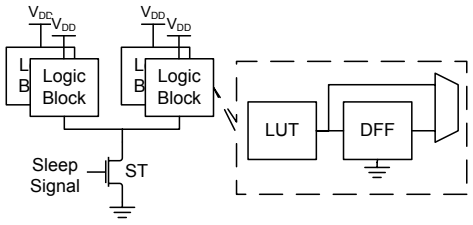

Fig. 1. FPGA fabric architecture.

II. Performance Loss and Sleep Transistor Size

The size of the ST is an important design factor to avoid incurring large performance penalties while maximizing the leakage power savings. Increasing the width of the ST, decreases its resistance, resulting in a reduction in the delay penalty incurred by the ST. However, a large ST results in a larger leakage current, a large area penalty, and a considerable dynamic power consumption during switching between the *on* and *off* states. Hence, a compromise is needed between the performance penalty, power savings, and area requirements.

By limiting the performance loss to $x\%$, the aspect ratio of the ST $\frac{W}{L}\big|_{\text{sleep}}$ is approximated by [11]

$$\frac{W}{L}\bigg|_{\text{sleep}} = \frac{I_{\text{sleep}}}{x\mu_n C_{ox}(V_{DD} - V_{th_L})(V_{DD} - V_{th_H})}\,, \qquad (1)$$

where I_{sleep} is the maximum discharge current the ST can hold for a performance penalty of $x\%$, μ_n is the electrons mobility,

1-4244-0630-7/07/$20.00 ©2007 IEEE.

C_{ox} is the MOS oxide capacitance, and V_{th_H} and V_{th_L} are the threshold voltages of the high- and low-V_{th} devices, respectively. The worst-case value of I_{sleep} is the sum of the maximum discharge currents of all the logic blocks inside the sleep region.

A. Sleep Region Discharge Current

The discharge current pattern inside any sleep region depends, mainly, on the connections between the logic blocks inside it. Logic blocks that fan-out other logic blocks will start discharging first. The discharge current of each logic block can be represented by a vector, whose elements represent the discharge current value recorded at equal time intervals. In this work, the discharge current vectors are represented with a triangular approximation, as shown in Fig 2(e), where the x-axis represents the time and the y-axis value represents the current values recorded at the corresponding time instant [11].

B. Topological Sorting and Current Vector Addition

In order to find out the order at which the logic blocks inside each sleep region will discharge, the logic blocks should be ordered in a topological manner. Fig. 2 depicts an example of topological sorting applied to the simple circuit in Fig. 2(a). It should be noticed that logic blocks B and C were ordered on the same level simply because in step 2 (Fig. 2(b)), they both have only inputs coming from outside the sleep region.

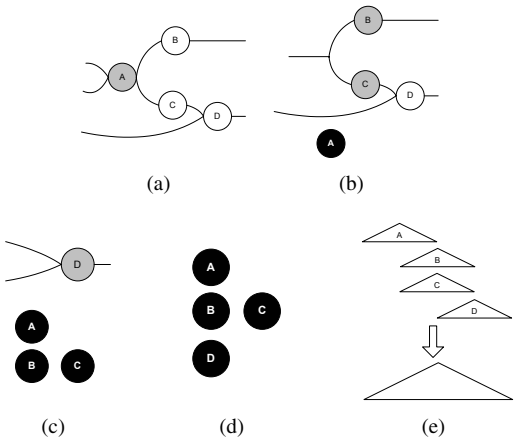

Fig. 2. Topology sorting steps for a combinational connected sleep region. (a) A is selected to be deleted, (b) A is ordered in the first position and B and C are selected for deletion, (c) B and C are ordered in the same position, (d) Final ordering, (e) Current vectors summation.

Utilizing the topological ordering, the discharge currents of the logic blocks can be arranged and summed in a vector manner as shown in Fig. 2(e). The resulting summation is the total discharge current inside the sleep region. Hence, the discharge current inside that sleep region will not exceed the maximum of the summation. Consequently, I_{sleep} is designed to accommodate only the peak current of the vector summation rather than the sum of the logic blocks peak currents.

C. Timing Information and I_{sleep}

From (1), it can be noticed that for the same $\frac{W}{L}\big|_{\text{sleep}}$ of the ST, the performance loss depends on I_{sleep}. A sleep region with a large I_{sleep} will have a larger performance loss than another one with smaller I_{sleep}, if they employ equal-sized STs.

This work makes use of this observation to avoid incurring a large performance penalty on the critical path. Hence, the maximum performance loss along the critical path can be limited to a value smaller than that along non-critical ones, *i.e.*, timing-driven MTCMOS (T-MTCMOS). The timing information of the logic blocks is used to vary I_{sleep} of each cluster according to its criticality using the proposed T-MTCMOS technique. In this work, the speed penalty is varied between 3% and 8%.

III. Modified Activity Profile Generation

An *activity profile* is a representation of the periods that a logic block is active. Logic blocks with similar activity profiles are active during the same time periods, hence, should be packed in the same sleep region for maximum power savings. The activity generation algorithm identifies the blocks that have a similar activity profile. The activity of each block is represented by a binary sequence (*Activity Vector*) and the relation between the blocks activities is calculated based on the *Hamming distance* between their activity vectors [5].

The activity vector of logic block X A_x is a set of 2^n binary variables. In [5], n was used as the number of circuit inputs to get the global optimum activity matching, thus resulting in a complexity of $O(2^n)$. However, in this work, n is the number of inputs to the sleep region and the blocks are packed in a greedy manner in the same steps as T-VPACK [12]. This might cause a slight loss in matching the activity profiles, however, it reduces the complexity to $O(2^n)$, where n usually ranges between 7 and 10, depending on the architecture of the FPGA.

The i^{th} binary variable is a '1' if any of the outputs of the circuit depends on net x, the output of X, for evaluation when the sleep region inputs are given by the i^{th} input vector, and '0' otherwise [5]. As an example, consider the small section of a large circuit shown in Fig. 3, d will affect the value of f, only when $c = 1$ and $d = 1$. This case corresponds to only one input vector $abc = 111$. Using these guidelines, the activity vectors of logic blocks D, E, and F are given by

$$A_d = [\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ ]^T,$$
$$A_e = [\ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ ]^T,$$
$$A_f = [\ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ ]^T.$$

When two activity vectors overlap with a '0', this means that they can be both turned *off* during that input vector.

The Hamming distance between any two activity vectors gives the number of input combinations at which these two blocks will have different activities, *i.e.*, one is *on* and the other is *off*. Hence, the smaller the Hamming distance between the logic blocks, the closer their activities are. As an example, the Hamming distance between D and E in Fig. 3 is 3.

To account for the different probabilities of occurrence of the input combinations, the weighted Hamming distance ($dw_{(a,b)}$)

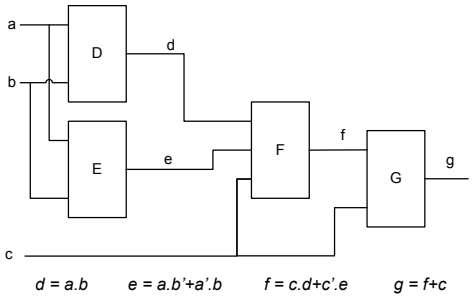

Fig. 3. A circuit example.

is used to represent the difference in activities

$$dw_{(a,b)} = \sum_{k=0}^{n-1} w_k \times |a_k - b_k| \ , \qquad (2)$$

where a_k, b_k, and w_k are the k^{th} elements of the activity vectors A and B and the inputs probability vector W_n, respectively.

IV. Packing Algorithm

In modern FPGAs, BLEs are packed together to form clusters (CLBs). In this work, the T-MTCMOS technique is incorporated into the T-VPack [12] algorithm to pack BLEs to minimize leakage power dissipation while considering timing information.

A. Conventional T-VPack Algorithm

T-VPack packs LUTs one at a time into clusters, while satisfying two hard constraints; (1) the number of BLEs in the cluster does not exceed the cluster size and (2) the number of input nets needed by the BLEs in the cluster and generated outside the cluster does not exceed the number of cluster inputs. T-VPack tries to fill the clusters to their full capacity while maximizing a cost function, $Attraction()$, that compromises between the delay across the critical path and total wire length. The $Attraction(B)$ between logic block B and cluster C is calculated as [8]

$$Attraction(B) = \lambda \times Criticality(B) + (1-\lambda) \times SharingGain(B,C) \ , \qquad (3)$$

where $Criticality(B)$ is a representation of the criticality of block B, $SharingGain(B,C)$ is the number of nets shared between B and C, and λ is a weighting constant [8]. The choice of λ depends on a compromise between timing improvement (setting λ close to 1) or routability and wirelength improvement (setting λ close to 0).

B. Activity Gain (*ActivityGain*)

The $ActivityGain$ is a representation of how close is the activity vector of block B to that of cluster C. The activity vector of a cluster is calculated by a wide OR operation of the activity vectors of all the blocks inside the cluster. Hence, the $ActivityGain(B,C)$ with respect to cluster C is calculated as

$$ActivityGain(B) = \frac{2^n - dw_{(b,c)}}{2^n} \ , \qquad (4)$$

where n is the number of cluster inputs.

The algorithm starts by calculating the $ActivityGain$ between all of the unclustered blocks and the seed block for the new cluster. Afterwards, whenever a new block is added to the cluster, the activity vector of the cluster is updated and the bits that experience a change in the cluster activity vector are recorded. If for example, the i^{th} bit in the cluster current vector changes to '1', then the algorithm updates the $ActivityGain$ of all of the unclustered blocks that have a '0' in the i^{th} position in their activity vector by incrementing 1 to their Hamming distance. This goes on until the cluster is full and a new cluster with a new seed block is started and the procedure is repeated.

C. Timing-Based Maximum Discharge Current

The maximum discharge current inside any cluster should not exceed the value used in (1) for a speed penalty of $x\%$. The value of the discharge current can vary from one cluster to the other depending on the criticality of each cluster, hence, the speed penalty imposed on the cluster. In order to account for the different criticalities along the signal paths, $I_{\text{sleep}}(C)$ is formulated as

$$I_{\text{sleep}}(C) = \left[1 + \delta\left(1 - \frac{Criticality(C)}{Max\_Criticality}\right)\right] \times \widehat{I}_{\text{sleep}} \ , \qquad (5)$$

where $\widehat{I}_{\text{sleep}}$ is the maximum discharge current calculated for the minimum performance penalty, *i.e.*, 3%, δ is a weighting constant, $Criticality(C)$ is the criticality of cluster C, and $Max\_Criticality$ is the criticality of the critical path(s) of the circuit. From (5), it can be noticed that if the criticality of the cluster is equal to the maximum criticality of the circuit, *i.e.*, the cluster lies on the critical path, the value of I_{sleep} will be equal to that for the minimum performance penalty, *i.e.*, 3%, otherwise, a larger value for I_{sleep} will be used, hence, a larger performance penalty.

The weighting factor δ is used to make sure that after adding block B to cluster C, the path does not become a critical path itself. If δ is set to a value close to 0, then all of the sleep regions will have an I_{sleep} very close to that for a 3% performance penalty. On the other hand, if δ is set to 1.6, the sleep regions will have a wide variety of I_{sleep} values, hence speed penalties, with a maximum penalty of 8%. However, a large value for δ increases the possibility that the added performance penalty might cause some uncritical paths to become critical. By conducting several experiments on the value of I_{sleep}, it was discovered that by adopting an adaptive update technique for δ, shown below, depending on the criticality ratio ($Criticality(C)/Max\_Criticality$), resulted in no new critical paths while having a wide variety for I_{sleep} values.

$$\begin{array}{cc} 0 < Criticality(C)/Max\_Criticality \leqslant 0.5 & \delta = 1.6 \\ 0.5 < Criticality(C)/Max\_Criticality \leqslant 0.8 & \delta = 0.8 \\ 0.8 < Criticality(C)/Max\_Criticality \leqslant 1 & \delta = 0 \end{array}$$

D. Timing-Driven MTCMOS Gain Function

The T-MTCMOS gain function used in this work is given by

$$\begin{aligned} Attraction(B) = \quad & (1-\alpha) \times \Big[\lambda \times Criticality(B) \\ & +(1-\lambda) \times SharingGain(B,C)\Big] \\ & +\alpha \times ActivityGain(B,C) \end{aligned} \quad (6)$$

where α is weighting constant ($0 \leqslant \alpha \leqslant 1$). Setting a large value for α will force the packing algorithm to pack the blocks that have the shortest Hamming distance in the same cluster, hence, same sleep region, without giving much weight to the timing information and wirelength. In the following experiments, α will be set to 0.5 and the impact of its value on both the leakage savings and speed penalty will be discussed later.

V. Power Estimation

There are two standby modes for any circuit; full standby and partial standby. In the fully standby state, the whole circuit is in the idle state and all of the STs in the circuit should be turned off. During that period, the circuit only consumes standby leakage power. During partial standby, some parts of the circuit are in the active state and other parts are in the idle state. Hence, some of the STs are turned *on* and others are *off*. Thus, the circuit will consume a combination of dynamic power and active and standby leakage power.

The total power dissipation P_t is expressed as

$$P_t = t_{on} \times P_{on} + t_{off} \times P_{idle} \ , \quad (7)$$

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

$$P_{on} = [P_{dyn} + P_{sckt} + P_{leak}]_{utilized} + P_{leak}|_{unutilized} \ , \quad (8)$$

where P_{dyn}, P_{sckt}, and P_{leak} are the dynamic, short-circuit, and active leakage power dissipations, respectively, in the utilized CLBs, while $P_{leak}|_{unutilized}$ is the standby leakage in the unutilized CLBs.

The power estimator used in this work is a modified version of the flexible power model proposed in [13], which calculates the dynamic, short-circuit, and leakage power dissipation for the logic blocks and clock resources in the placed and routed design. Four different modifications were done to the original power model. (1) Leakage current is calculated only due to the ST rather than calculating the leakage through all the devices in the circuit because the STs act as a bottleneck for the subthreshold leakage current. (2) The leakage power dissipation in the unused logic blocks is calculated and added to the total power dissipation. (3) Short circuit power dissipation is approximated as 15% of the dynamic power dissipation rather than the 10% used in the original model to account for the increased rise/fall times of the logic blocks with STs. The 15% approximation was evaluated by simulating logic blocks with and without an ST using HSPICE. (4) The dynamic power consumed in the ST during switching between the *on* and *off* states is calculated and added to the total power dissipation.

VI. Results and Discussions

The T-MTCMOS algorithm is integrated into VPR and the modified power model is used to estimate the power savings in the final design in several FPGA benchmarks. The results for the savings in leakage power are summarized in Table I for a sleep region of size 4 logic blocks using a CMOS 0.13μm process. It should be noted that the maximum allowable performance degradation due to STs in all of the benchmarks is kept between 3% and 8% depending on the criticality of the logic blocks.

In this experiment it is assumed that the circuit has an *on* time of 100%. Moreover, the design was mapped to the minimum sized square array that can fit the resulting packed design. The percentage of unused clusters for each benchmark are listed in Table I. This assumption ensures maximum utilization among all of the benchmarks. For simplicity, this case is called *100% utilization*.

The decision whether to keep a utilized sleep region *on* all of the time or dynamically switching between *on* and *off* depending on its activity profile is based on a compromise between the leakage power savings and the dynamic power dissipated in the ST. Whenever the dynamic power dissipation in the ST exceeds the leakage savings from any cluster, the cluster is kept always *on*. Leakage power savings can be achieved when the cluster stays *off* for a certain period of time, $T_{\text{break even}}$. In this work, the transition density [14], the average number of transitions per cycle, is used as a measure of how long a signal stays in a certain state. Based on the transition density of each sleep signal, if the signal experiences a large number of transitions such that $T_{\text{break even}}$ is never or rarely reached, the sleep region is kept always *on*, otherwise, it is dynamically turned *on* and *off* depending on the activity profile.

TABLE I
Experimental results for FPGA benchmarks (maximum utilization and 100% on time).

Circuit	# of BLEs	% of Unused Clusters	% Leakage Savings [5]	% Leakage Savings T-MTCMOS
alu4	1522	4.5	22.9	50.13
apex2	1878	2.48	20.7	46.87
apex4	1262	2.16	19.1	41.96
bigkey	1707	2.72	20.2	42.87
clma	8381	0.76	18.9	40.02
des	1591	0.25	16.8	39.67
diffeq	1494	6	22	51.34
dsip	1370	4.7	21.2	44.05
elliptic	3602	5.9	21.6	49.56
ex1010	4598	0.26	18.7	47.31
ex5p	1064	6.92	22.8	55.69
frisc	3539	1.56	18.2	41.88
misex3	1397	2.21	20.9	43.29
pdc	4575	0.78	17.7	33.43
s298	1930	8.32	28.3	64.57
s38417	4096	5.6	25.4	34.39
s38584.1	6281	1.56	18.9	39.96
seq	1750	0	14.2	23.64
spla	3690	3.6	18.3	39.43
tseng	1046	8.3	27	57.21

In each benchmark, the leakage power savings consist mainly 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 during operation. By taking a look at the results for the 'seq' benchmark in Table I,

this benchmark has no unused clusters while the leakage savings achieved is 23.64%, which is entirely from dynamically turning *on* and *off* the different used clusters in the design during operation. On the other hand, the 's298' benchmark has the maximum percentage of unused blocks among all of the benchmarks and resulted in the maximum leakage savings (64.57%) because the unused clusters are kept *off*. The average leakage savings across all of the benchmarks for the 100% *on* time case is found to be 44.36%.

By comparing the results in Table I to those achieved reported in [5], it is observed that the proposed T-MTCMOS algorithm achieves more leakage power savings than [5] across all of the benchmarks. The average improvement in the leakage power savings is almost 2X. The main reason behind the increase in leakage savings is that in [5], the discharge current constraint is a hard constraint across all of the benchmarks, thus the algorithm might fill a cluster with logic blocks that have different activity profiles and satisfy the current constraint, although there are other blocks that have closer activity profiles but violate the current constraint. On the other hand, T-MTCMOS allows the current constraint to be violated to a certain extent along non-critical paths, thus giving more freedom to the packing algorithm to pack logic blocks with close activity profiles to achieve more leakage power savings. Moreover, the resulting design from T-MTCMOS outperforms that of [5] in terms of timing properties. The algorithm in [5] incurs a minimum of 5% delay penalty across all the paths in the design. However, T-MTCMOS increases the delay across the critical path by a minimum of 3% while making sure no other critical paths get created.

T-MTCMOS might result in a different packing than that resulting from the conventional T-VPACK algorithm due to the inclusion of the $ActivityGain$ in the gain function and the maximum discharge current constraint. Hence the resulting placements might have different critical paths as well as critical path delays. By comparing the critical path delay resulting from T-MTCMOS to that resulting from the conventional T-VPACK, it was found that the maximum increase in the critical path delay across all of the benchmarks is around 9.5%. The value of α was then allowed to change from 0 to 1 and the maximum speed penalty and average leakage power savings were recorded plotted in Fig. 4. It can be deduced that increasing α increases both the delay penalty and the leakage savings. Hence, by changing α, the algorithm can be tuned to offer minimal speed penalty in high-performance applications and maximal leakage savings in slower applications.

In another experiment, the maximum performance penalty allowed is varied from 8% to 14%, while the minimum performance penalty is kept at 3% and the results for the 's298' benchmark are plotted in Figure 5. It was noticed that for a sleep region of size 4 logic blocks, the leakage savings increased with the maximum speed penalty until a speed penalty of 10%, after which the curve almost flattens. The increase in leakage savings can be justified by the fact that increasing the maximum speed penalty allows the packing algorithm to pack logic blocks that exhibit similar activity profiles in the same cluster without worrying about their discharge current. On the other hand, as the speed penalty is increased beyond a certain limit, the packing algorithm can not achieve more leakage savings because the sleep regions are now packed to their maximum (4 logic blocks). However, as the number of logic blocks per sleep regions is increased, more leakage savings can be experienced, as shown in Figure 5. It should be noted that increasing the size of the sleep region beyond 8 logic blocks, results in an increase in the leakage savings, however, the dynamic power dissipation in the STs, which are significantly upsized, increases to cancel out most of the leakage savings.

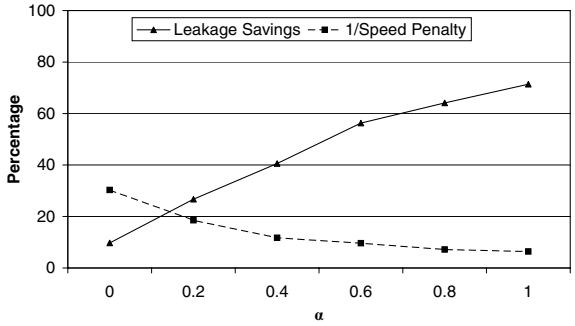

Fig. 4. Percentage leakage savings and performance versus α.

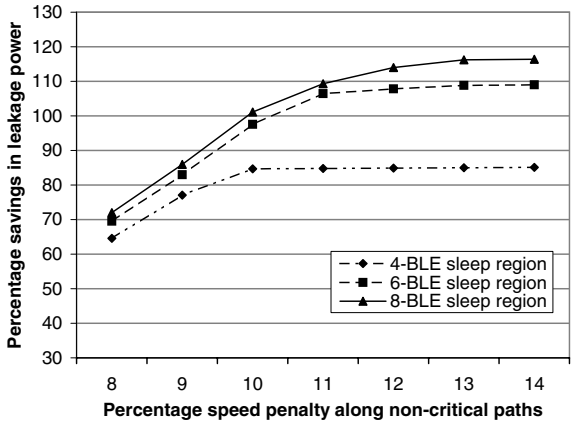

Fig. 5. 's298' leakage savings vs maximum speed penalty.

In another experiment, the maximum speed penalty is set to 12 for the 's298' benchmark while varying the minimum speed penalty (Fig. 6). As the minimum performance penalty increases (by up-sizing the ST), the leakage savings increases, until a certain limit after which the savings decrease because of the increase in dynamic power of the STs. It can be noticed that the breakpoint gets smaller as the size of the sleep region increases because larger sleep regions employ large STs.

In order to investigate the scalability of the T-MTCMOS methodology, the methodology is applied on all of the benchmarks built with some current CMOS technologies (130nm, 90nm) and some future CMOS processes (65nm, 45nm) [15], and the results of the average leakage savings in all of the benchmarks are plotted in Figure 7. From Figure 7, it can be noticed that the average leakage savings increases almost exponentially with the technology. Hence, utilizing the T-MTCMOS technique in FPGAs would become more effective in sub 100nm technologies, as subthreshold leakage increases exponentially with scaling.

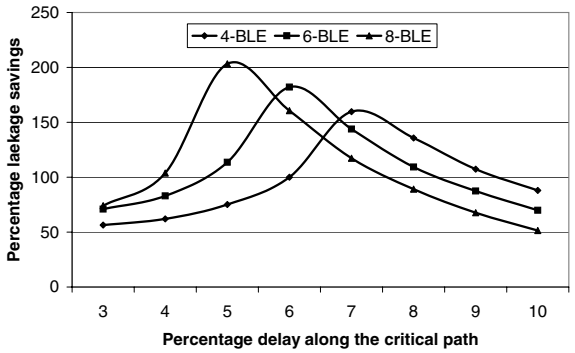

Fig. 6. 's298' leakage savings vs minimum speed penalty.

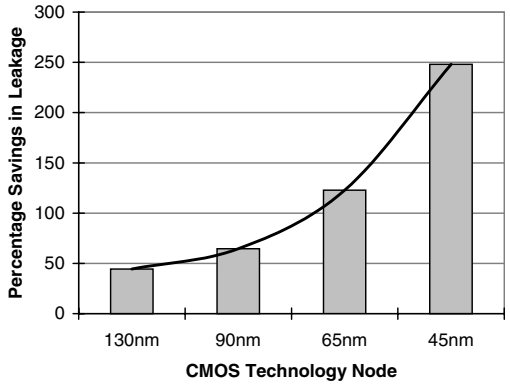

Fig. 7. Leakage savings for several CMOS technologies.

Practically, the utilization percentage is less than the maximum utilization assumption used in Table I [2]. Moreover, the 100% *on* time assumption made earlier is also impractically high with an average *on* time of around 50% to 20% [11]. Figure 8 plots the average power savings for different values of the utilization and *on* time for a sleep region of 4 logic blocks. Fig 8 that the average leakage savings increases by about 48% when the device idle time increases from 0 to 50%, while decreasing the utilization from 100% to 80%, increases the power savings by about 53%.

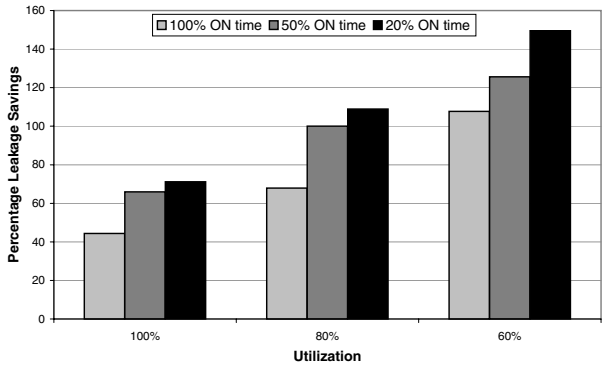

Fig. 8. Percentage savings in power for different utilizations and operational time.

Figure 9 plots the relative path delays distribution in the 'ex5p' benchmark with respect to the critical path delay. From Figure 9 it can be deduced that the number of critical paths did not increase. In addition, the maximum circuit delay changed by only 3%. The final shape of the delay distribution can be varied by changing δ.

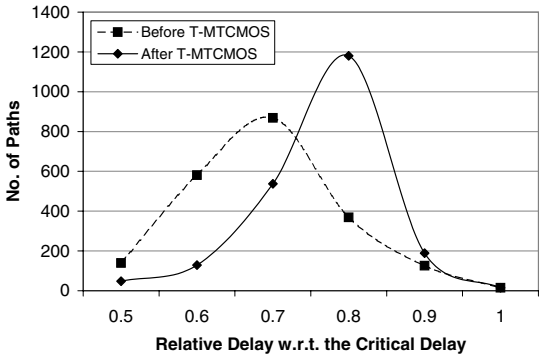

Fig. 9. Critical path distribution for timing-driven MTCMOS designs.

VII. Conclusion

This work proposed a new timing-driven MTCMOS methodology for subthreshold leakage power reduction in FPGAs. The worst case average leakage savings achieved is around 44.36%, while having a minimum impact on the critical path delay. Moreover, the T-MTCMOS methodology will provide more leakage savings for future CMOS technologies.

References

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

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

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

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

[5] H. Hassan *et al.*, "LAP: A Logic Activity Packing Methodology for Leakage Power-Tolerant FPGAs," in *Proc. of ISLPED*, 2005, pp. 257–262.

[6] T. Tuan *et al.*, "A 90nm Low-Power FPGA for Battery-Powered Appliations," in *Proc. of ISFPGA*, 2006, pp. 3–11.

[7] S. Mutoh *et al.*, "1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold-Voltage CMOS," *IEEE J. Solid-State Circuits*, vol. 30, no. 8, pp. 847–854, Aug 1995.

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

[9] B. Calhoun *et al.*, "A Leakage Reduction Methodology for Distributed MTCMOS," *IEEE J. Solid-State Circuits*, vol. 39, no. 5, pp. 818–826, May 2004.

[10] Z. Hu *et al.*, "Microarchitectural Techniques for Power Gating of Execution Units," in *Proc. of ISLPED*, 2004, pp. 32–37.

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

[12] 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.

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

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

[15] The Berkeley Predictive Technology Model website. [Online]. Available: http://www-device.eecs.berkeley.edu/∼ptm/