Post-Synthesis Sleep Transistor Insertion for Leakage Power Optimization in Clock Tree Networks

Houman Homayoun[1], Shahin Golshan[1], Eli Bozorgzadeh[1], Alex Veidenbaum[1], Fadi J. Kurdahi[2]

[1]Center of Embedded Computer Systems
University of California, Irvine, CA
{hhomayou,ssgolsha,eli,alexv}@ics.uci.edu

[2]Department of Electrical and Computer Engineering
University of California, Irvine, CA
kurdahi@uci.edu

Abstract

Leakage power has grown significantly and is a major challenge in SoC design. Among SoC's components, clock distribution network power accounts for a large portion of chip power. In this paper, we propose to deploy sleep transistor insertion (STI) in the clock tree in order to reduce leakage power. We characterize the effect of sleep transistor sharing and sizing on clock tree wakeup time, leakage power, and propagation delay. We use these characteristics during leakage power optimization. We present post synthesis sleep transistor insertion (PSSTI), a heuristic clustering algorithm for sleep transistor insertion with the objective of total power minimization in a given clock tree. Sleep transistor sharing and sizing are deployed in order to meet the clock skew and wakeup delay constraints. We explored the potential benefits of STI using a standard industrial VLSI-CAD flow including sleep-transistor insertion and routing after clock synthesis and place-and-route of the benchmark circuits. Our results show that clock tree leakage power is reduced by 19%-32% depending on the topology of the synthesized clock tree.

Keywords

Sleep Transistor, Clock Tree

1. Introduction

Process scaling has enabled SoC's designs to offer much higher computational power. Technology scaling has, however, led to higher power dissipation, especially leakage power. In order to overcome such growing leakage problem, several approaches at technology level, circuit level, and architectural level have been proposed and have been applied to many on-chip blocks such as SRAMs, registers and arithmetic units. Clock distribution network power accounts for more than 40% of the overall power consumption of high performance VLSI chips due to frequent switching, driving large capacitances and large number of inverter buffers [1,2,3,4,17]. This paper focuses on leakage optimization in clock tree networks.

Recently a number of works has been proposed to reduce the power of clock trees. Clock gating has extensively been proposed and applied to reduce power by masking off clock signal where branches of clock tree are idle [1,13,14]. Clock buffer sizing [5,6,7], power-aware placement [10,12], exponentially tapering clock interconnect network [8] and using multiple-supply voltage [9] have been proposed for low power clock tree constructions. In addition, using high threshold voltage (V_{th}) gates in the clock tree networks has been studied in [11].

Most of these techniques focused on reducing dynamic power of the clock tree. Along with leakage power increase due to technology scaling, the significant reduction in dynamic power by applying aforementioned methods on clock trees leads to more visible (and hence non-negligible) contribution of leakage power in clock tree network. Hence, leakage power optimization of clock tree is equally, if not more, important to be addressed in the clock tree network.

For leakage power reduction, several techniques such as supply and threshold voltage optimization, sleep transistor insertion and power gating have been proposed and have extensively been studied in literature [3,15,16]. In this paper, we propose sleep transistor insertion in clock tree networks in order to reduce leakage power. STI is a well known technique for reducing leakage of an idle unit by isolating it from V_{dd} and Vss. STI has been extensively applied in many on-chip blocks such as SRAM memories [18,21]. However, to the best of our knowledge, this is the first attempt to deploy STI in the clock tree network to reduce the leakage power. In comparison with using high V_{th} gates in a clock tree [11], our proposed leakage optimization through sleep transistor insertion is an orthogonal approach and our results show significant leakage reduction in clock tree networks.

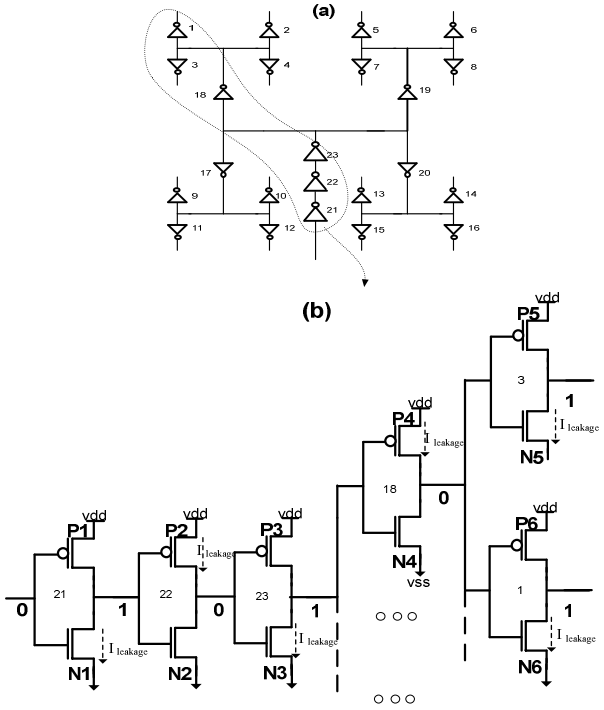

Figure 1. (a) H-tree clock network (b) source of leakage

Figure 1(a) shows an example of an H-tree clock network which uses inverter buffer chains to drive the clock signal from source to each flip-flop (sink). The source of subthreshold leakage in the H-tree clock network is illustrated in Figure 1(b). In order to reduce the leakage in the clock buffers both footer and header sleep transistors are inserted for all NMOS and PMOS transistors in the clock buffer. However, aside from the area overhead, it increases the propagation delay of the clock tree due to an increase in the rise/fall time of the drivers. In order to overcome such effects on the critical timing components we propose to deploy zigzag sleep transistor insertion technique which has been recently proposed to reduce peripheral circuit leakage power of SRAM memories [18]. We show that zigzag insertion can guarantee no effect on the driving flip-flop clock rise time. However, using one sleep transistor per inverter logic increases the area overhead of zigzag scheme. Moreover, zigzag insertion can affect the fall time and propagation delay of the clock signal. We propose to eliminate the impact of zigzag scheme on fall time and propagation delay by appropriately sizing the sleep transistors. To minimize area-overhead and further improve leakage power savings, we use sleep transistor sharing technique [18]. While sleep transistor sharing is effective in reducing leakage, it has a drawback of impacting circuit wakeup latency which occurs when the circuit is transitioning from sleep mode to active mode.

Sleep transistor insertion and sharing techniques lead to power saving when the underlying circuit is idle. Therefore, it is crucial to model the idle time patterns of different clock tree buffers during the course of execution.

In this paper we evaluate the benefits of sleep transistor insertion and sharing in clock tree networks. In brief we make the following contributions:

- We characterize the effects of sleep transistor sharing and sizing on clock tree wakeup time, leakage power, propagation delay and skew.
- We present post synthesis sleep transistor insertion algorithm (PSSTI) on clock tree networks which clusters the clock buffers sharing the same sleep transistor with the objective of total power minimization subject to clock timing constraints.
- We explored the potential benefits of PSSTI using a standard industrial VLSI-CAD flow.
- We applied PSSTI on a subset of Express [24] benchmarks. The sleep transistors were inserted and routed for each clusters of clock buffers. The overall power reduction and the impact on clock tree timing components were obtained using standard industrial tools.

Results on a subset of Express [24] benchmarks show significant reductions in total power and leakage power by 16% and 32% on average, respectively.

2. SLEEP TRANSISTOR INSERTION

Inserting sleep transistors have been proposed to reduce sub-threshold (ID_{sub}) or weak inversion current [19]. ID_{sub} is an inverse exponential function of threshold voltage (V_{th}). Threshold voltage is a function of source-to-bulk Voltage (V_{SB}).

An effective way to reduce the leakage of a transistor is to increase its source voltage (for an NMOS increasing V_{SB}, the source to bulk voltage) [19, 20]. Inserting a sleep transistor (footer NMOS or header PMOS transistor) as shown in Figure 2 delivers this effect. In this figure, by coupling transistor N with slpN, source-to-body voltage (V_M) of transistor N increases. When both transistors are off, the increase in V_M increases the V_{th} of the transistor N and therefore reduces sub-threshold leakage current [19].

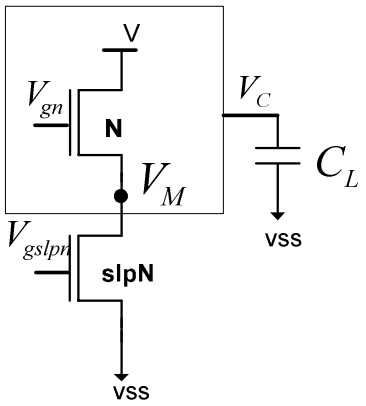

Figure 2. Inserting sleep transistor to reduce leakage

3. ZIGZAG SLEEP TRANSISTOR SHARING AND SIZING

To improve both leakage reduction and area-efficiency of the zigzag scheme, one set of sleep transistor is shared between multiple stages of inverters [18]. This is shown in Figure 3, where sleep transistor is shared across multiple levels of a buffer chain in a clock tree edge (buff21, buff22 and buff23) and across multiple edge of the clock tree (buff1, buff2, buff3 and buff4). Intuitively, by sharing sleep transistor, the virtual ground voltage (V_M in Figure 3) increases in comparison to when there is no sleep transistor sharing [18].

While sleep transistor sharing is effective in reducing leakage, it has a drawback of impacting circuit wakeup latency which occurs when the circuit is transitioning from sleep mode to active mode and requires the voltage of virtual ground to reach to the ground voltage [18]. Note that by increasing the number of clock buffers sharing one sleep transistor, the load on the sleep transistor increases. Assuming that n clock buffers are sharing one sleep transistor the capacitive load on the sleep transistor is as $\sum_i C_{wire}(i) + C_{diff}$, where the $C_{wire}(i)$ is the wire capacitance connecting the sleep transistor to each of clock buffer pull down transistor and C_{diff} is the inverter pull down transistor diffusion capacitance. As a result, increasing the number of clock buffers sharing one sleep transistor makes sleep transistor wakeup transition slower.

An effective way to minimize the impact of sharing on wakeup delay is to appropriately size the sleep transistors. The drawback of such resizing is on reducing leakage reduction and increasing sleep transistor dynamic power as explained next.

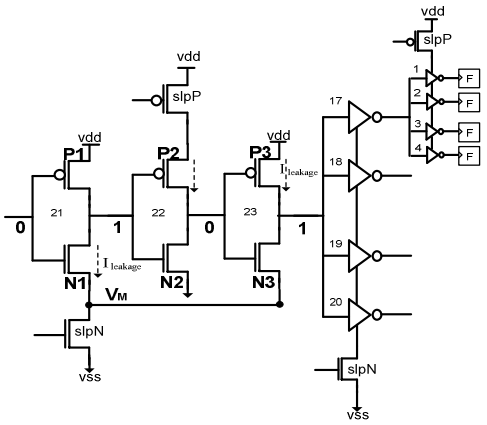

Figure 3. Zigzag sleep transistor sharing in clock tree network

In order to evaluate the benefit of sleep transistor sharing and sizing, we designed and analyzed a 16 sink H-tree clock network showing in Figure 1. We assume a symmetric distribution of flip-flops. We designed a 2-level H-tree clock network in a 0.1 x 0.1 mm chip in a 45 nm technology. We placed a driver at each source and sink. Half of the chip area is occupied by the flip-flops.

To model the interconnect we assume a coupling capacitance of 0.0960 fF/μm, and the capacitance to ground of 0.2450 fF/μm. The resistance is 0.1846 mΩ/μm. The interconnect capacitance is assumed to be as $C_{int} = (2\ C_{coupling} + \omega \times C_{coupling}) \times l$, where l is the wire length and ω is the wire width ratio to the nominal width. In clock tree design, one of the most effective techniques for adjusting clock skew is wire-sizing. In our experiments, a 2x wire is assigned to the first level of clock tree while a 1x wire is assumed for the second level.

To control the transition time of the clock network (which is important both for skew control and power dissipation [20]) we assume the constraint of $\eta = C_{out} / C_{in}$ to be 2.7, where C_{in} is the input capacitance of the driving clock buffer and C_{out} is the output load capacitance to drive. As indicated in [20], based on this assumption, the buffer sizing minimizes the delay of the chain of buffers. The proposed zigzag sharing approach is applied to the designed clock network. All simulations are done in a 45nm technology using Synopsys Hspice at typical corner (25º) and the supply voltage of 1V.

In Table 1 we report the impact of sleep transistor sharing on the wakeup delay. The sleep transistor is placed such that it has the same distance from all shared buffers. The results are for the worst case wakeup delay in which the distance between the sleep transistor and the shared buffers are assumed to be half of the maximum Manhattan distance between the farthest buffers in the designed clock tree network. This is shown in Figure 4 for buff1 and buff16.

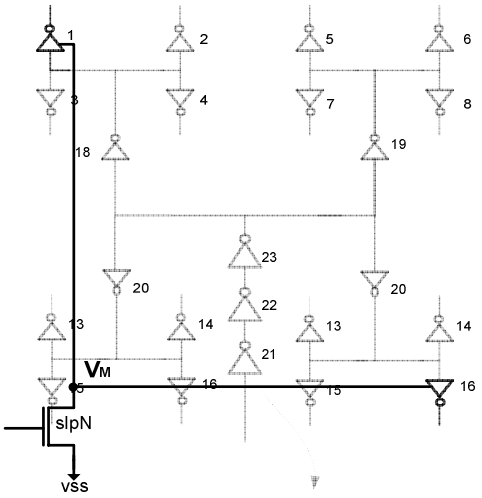

Figure 4. Sleep transistor insertion for worse case wakeup delay

The same table shows the impact of sleep transistor sizing on the wakeup delay. More sharing of sleep transistor results in larger wakeup delay. However, using a larger sleep transistor can reduce the wakeup delay.

Table 1. Impact of sleep transistor sharing and sizing on the wakeup delay

#shared buffers	W(1X) (ns)	W(2X) (ns)	W(3X) (ns)	W(4X) (ns)	W(5X) (ns)	W(6X) (ns)	W(7X) (ns)	W(8X) (ns)
1	0.256	0.137	0.093	0.064	0.045	0.037	0.032	0.029
2	0.620	0.367	0.273	0.205	0.155	0.136	0.124	0.115
3	1.190	0.732	0.583	0.464	0.381	0.345	0.321	0.309
4	1.655	1.072	0.877	0.736	0.637	0.596	0.564	0.556
5	2.130	1.438	1.214	1.065	0.952	0.905	0.884	0.882
6	2.595	1.817	1.609	1.453	1.336	1.298	1.277	1.275
7	3.050	2.196	1.983	1.830	1.739	1.708	1.699	1.696
8	3.525	2.609	2.432	2.291	2.203	2.178	2.171	2.170
9	4.010	3.036	2.887	2.767	2.695	2.675	2.667	2.663
10	4.450	3.471	3.338	3.235	3.182	3.168	3.163	3.160

The following equation expresses the switching delay of the sleep transistor as a factor of number of shared buffers: $\sum R_{eq} \times (C_{wire}(i) + C_{diff})$. By increasing the size of sleep transistor, its equivalent resistance (R_{eq}) becomes smaller which makes the wakeup delay smaller. As we increase the number of buffers sharing the sleep transistor, the equivalent output capacitive load of the sleep transistor increases, which increases the wakeup delay.

In Figure 5 we report the relative leakage power reduction as a function of sleep transistor size, and the number of sharing clock buffers. As the number of shared buffer increases, leakage power reduces further. Our simulation results show that the sleep transistor size does not have a significant impact on leakage power savings (shown in Figure 5).

In Figure 6, we report the impact of sleep transistor sizing on propagation delay. As the results show, increasing the size of sleep transistor reduces the propagation delay overhead. Our simulation results indicate that increasing the number of shared buffers has almost no impact on the propagation delay overhead.

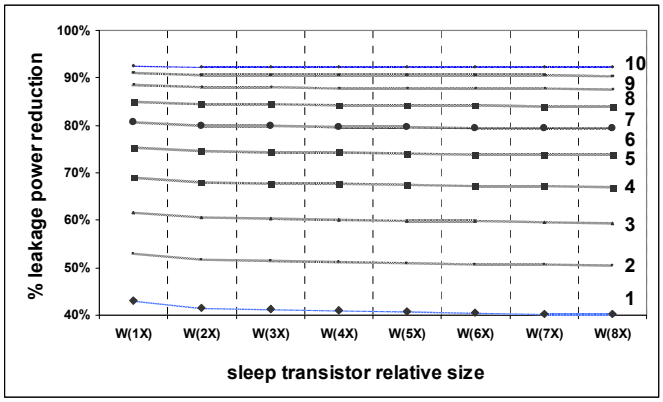

Figure 5. Relative leakage power reduction

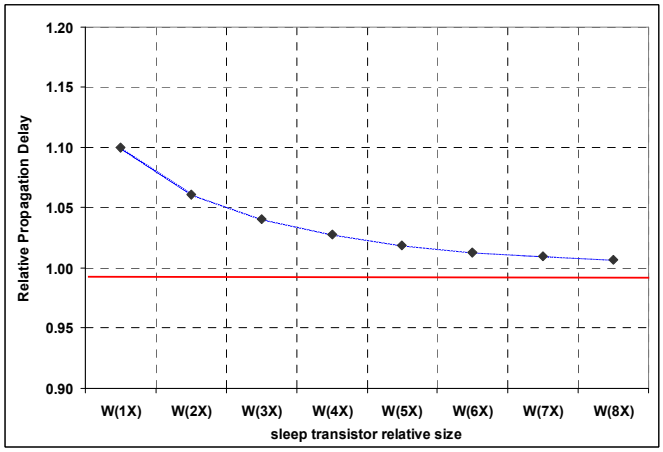

Figure 6. Impact of sleep transistor sizing on propagation delay

4. POST SYNTHESIS CLOCK TREE LEAKAGE POWER OPTIMIZATION

In the previous section, we have shown that STI inherently reduces the leakage power dissipation of a group of clock buffers by switching them to the sleep mode. In this section, we first analyze the idle cycles of the clock buffers and clusters of clock buffers in a clock tree. Then we present the formulations to model the dynamic and static power consumption of clock buffers when sleep transistors are inserted, considering the idle/active intervals of each clock buffer. Finally, we review the impact of STI on the timing integrity of the clock tree (i.e. skew variations). Throughout this section, we assume that the clock buffers have already been sized and placed.

4.1. Idle time patterns of clock buffers

As an abstraction of a clock tree (H-tree), we represent a clock tree with a rooted tree G(V, E), where the root is the source of the clock signal, and the leaves are synchronized with clock signals (e.g. flip-flop cells). Each edge represents the lumped buffer which drives the signal from the source point to sinks (Figure 7). As shown in Figure 7, each buffer can be associated with the node it drives.

For each circuit, we define the operation period as the total number of cycles it takes to perform the operation once all the primary inputs are available, for example, OP in Figure 7 is 4 cycles. We assume that the circuit resumes operation once the outputs of the circuit are computed.

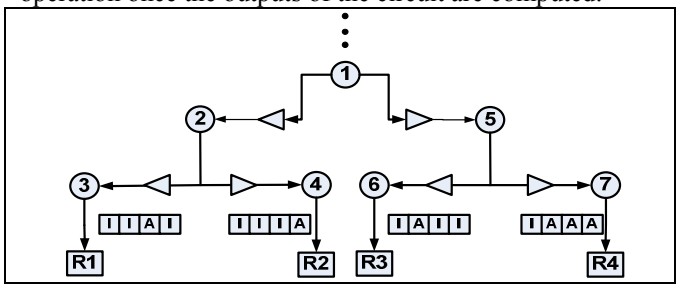

Figure 7. Abstract model of clock tree and the ITP each node

In order to explain the impact of sleep transistor insertion as well as the impact of the clock topology itself on leakage power saving, we need to know the idle time pattern (ITP) of a clock buffer, which is defined as the active/idle cycles of the clock buffer in one OP.

We first model the ITP as a continuous function of time which maps each time instant to the value "1" / "0" when idle/active. We then convert each ITP function to its discrete-time ITP sequence, in which the i^{th} element of this sequence corresponds to the value of the ITP function at the i^{th} clock cycle. When the index is written in brackets, the ITP implies the ITP sequence. For continuous ITP functions we use parentheses. The parameter X represents the clock buffer pertaining to the ITP. For simplicity, the ITP of a clock buffer may be referred to as the ITP of the clock node it drives. For example, the ITP of node 4 in Figure 7 is actually the ITP of the clock buffer driving it.

We assume that the activity patterns of the flip-flops have already been calculated in the previous stages of the design flow. As the flip-flops are assigned to resources and the active cycles are scheduled in scheduling and resource binding stages, the activity pattern of each flip-flop is determined. The leaves of a clock tree have the ITPs of the corresponding flip-flops. This information is used to obtain the ITP of each clock buffer in the clock tree. For example in Figure 7, if the operation period is 4, the ITP sequence for node 4 and node 5 are ITP4 = {1, 0, 0, 1} and ITP5 = {1, 1, 1, 0} respectively.

We can calculate ITP of each clock buffer X based on the ITPs of its children nodes in a recursive fashion using eq. 1:

$$ITP_X(t) = \prod_{Y \in children(X)} ITP_Y(t - D_{X \to Y} \bmod OP) \qquad (1)$$

Eq. 1 states that the internal node will be active at time instant t, if the child node is going to be active at time instant $t + D_{X \to Y}$, where $D_{X \to Y}$ is the total delay (buffer delay and wire delay) observed in the path from node X to node Y on the clock tree. It is crucial to include the time shifts as we move from the leaves of the clock tree towards the clock root, since the lags perceived in the ITPs of the nodes closer to the clock leaves relative to the nodes closer to the clock root might reduce the common idle times among ITPs of such nodes and as a result lighten the benefits of sleep transistor sharing. Figure 8 illustrates the bottom-up

calculation of ITP for a parent node based on its children nodes.

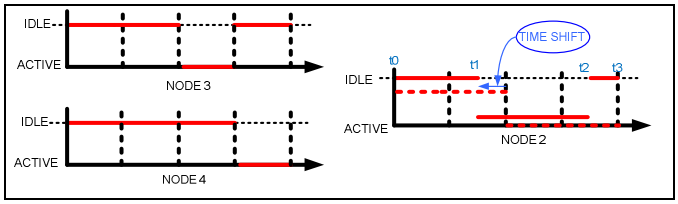

Figure 8. Example of bottom-up ITP generation

We define an idle pulse (IP) within an ITP as a range of time over which the clock buffer is idle. For example, in Figure 8, the ITP of node 2 contains an idle pulse over the circularly contiguous range (t3, t1). If the width of an idle pulse is less than one clock cycle, we are not able to exploit the idle time using a sleep transistor (since it is not possible to switch from active to idle and then from idle to active with a single clock cycle).

Once the continuous time ITP of a clock buffer is calculated, the discrete-time ITP sequence of a clock buffer can be obtained as outlined in the following algorithm:

Algorithm ITP sequence generation
Inputs: the continuous ITP of clock buffer X, the operation period (OP) in terms of no. of cycles, the clock period (*T*)
Output: the ITP sequence of clock buffer X
For every *i* from 0 to *OP*-1
If $ITP_X(i.T) = 1$ and $ITP_X((i+1 \bmod OP).T) = 1$ then
$ITP_X[i] \leftarrow 1$ ///idle
else

Figure 9. Algorithm for ITP sequence generation

As shown in Figure 9, the i^{th} element of this sequence is active if the continuous time ITP is active at the corresponding time instant. If the continuous time ITP is idle at the time instant, we have to see whether the ITP is idle in one cycle after the time instant. If it is idle, then we can safely set the current element of the ITP sequence to be idle. Otherwise we have to set it to be active. An ITP sequence is said to be safe if the continuous time ITP is not active over the entire clock cycle *i*.

Lemma. ITP sequence generation produces a safe sequence.

Proof. As shown in Figure 9, the only way we set an ITP element to be idle is when it is currently idle at the corresponding time instant and it will be idle one clock cycle after the time instant. Therefore, the ITP will be idle through this time cycle and it is safe to set to be idle □

An example of ITP sequence generation is illustrated in Figure 10. As depicted, $ITP_X[2]$ and $ITP_X[5]$ are set to be active, since in half of the corresponding clock cycles the clock buffer is active.

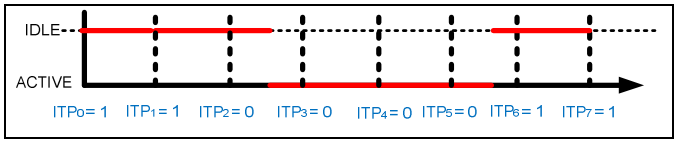

Figure 10. An example of ITP sequence generation

So far we formulated the ITP of each clock buffer individually. We now extend the concept of ITP to a cluster of clock buffers which are going to share the same sleep transistor. A cluster is denoted as $C = \{X_1, \ldots, X_n\}$ where X_i is a clocks buffer.

The following equation formulates the ITP of the cluster C:

$$ITP_C[k] = \prod_{X_i \in C} ITP_{X_i}[k] : \forall k : 0 \le k < OP \qquad (2)$$

Eq. 2 states that at clock cycle *k*, the cluster is considered idle if and only if all its member clock buffers are idle at cycle *k*.

4.2. Power formulation of sleep transistor clock buffer couples

Once the ITP of a cluster of clock buffers is determined, we determine the amount of power saving achieved by sleep transistor insertion. In order to model the total power consumption of clock buffer X coupled with sleep transistor S, we sum up the dynamic power consumption, *DP*(*X,S*) and leakage power consumption, *LP*(*X,S*) as stated in eq. 3:

$$P(X,S) = LP(X,S) + DP(X,S) \qquad (3)$$

In order to model the average leakage power of clock buffer consumed during the course of one operation period (OP), we need to consider the total number of idle cycles in its ITP, which translates into the amount of leakage power savings reached by turning off the clock buffer. This can be achieved by summing up the lengths of all the idle pulses (IP) of the ITP. However, we need to incorporate wakeup time of sleep transistors in the total number of cycles during which we can shut down the clock buffer. This parameter is referred to as effective-idle-cycles (EIC). In eq. 4, the wakeup delay (*wc)* is normalized based on the clock period:

$$EIC(X) = \sum_{IP_i \in ITP_X} (length(IP_i) - wc) \qquad (4)$$

Using the effective idle cycles, we model the leakage power consumption of clock buffer X coupled with a sleep transistor S as:

$$LP(X,S) = LP_S + LP_X \cdot \{(1-\alpha_X) \cdot EIC(X) + [OP - EIC(X)]\} / OP \qquad (5)$$

In eq. 5, LP_S and LP_X refer to the leakage power consumed by the sleep transistor S and the clock buffer X, respectively. For the idle and active cycles, the clock buffer consumes (1- α_X).LP_X and LP_X respectively. The parameter α_X is the total leakage power reduction achieved by turning off the clock buffer X through the sleep transistor S (Figure 5). Note that we average the leakage power over the operation period, since we are interested in reducing the effective leakage power consumption of the clock buffer and sleep transistor.

The dynamic power consumption of sleep transistor coupled with a clock buffer in an operation period is directly proportional to the number of times it toggles in the OP. Given the ITP of a clock buffer, the total number of toggles of the sleep transistor is calculated according to eq. 6:

$$tgl(X) = \sum_{i=1}^{OP} |ITP(X, i+1 \bmod OP) - ITP(X,i)| \qquad (6)$$

The function *tgl* indicates the number of times the sleep transistor needs to alternate to switch on/off the clock buffer.

The dynamic power dissipation of clock buffer paired with sleep transistor is modeled as:

$$DP(X,S) = \left[DP_S \cdot tgl(X) + DP_X \cdot (OP - EIC(X))\right] / OP \quad (7)$$

We assume that clock gating has been applied before implementing the proposed sleep transistor insertion. Therefore, the clock buffer only consumes dynamic power when it is active *(OP-EIC(X))*. The sleep transistor consumes dynamic power only when the clock buffer alternates between idle and active status (*tgl*(*X*)).

We extend the concepts of dynamic power and leakage power consumption to a cluster of clock buffers (C) which share the same sleep transistor (S). Eq. 8 and eq. 9 are extensions to leakage and dynamic power formulations expressed in eq. 5 and eq. 7:

$$LP(C) = LP_S + (\sum_{Xi \in C} LP_{Xi}) \cdot \{(1-\alpha_C) \cdot EIC(C) + [OP - EIC(C)]\} / OP \quad (8)$$

$$DP(C) = \left[\begin{array}{l} DP_S \cdot tgl(C) + \\ \sum_{Xi \in C} DP_{Xi} \cdot (OP - EIC(X_i)) \end{array} \right] / OP \quad (9)$$

The parameters *tgl*(*C*) and *EIC*(*C*) are calculated using eq. 4 and eq. 6. Provided that the set of standard buffers used in the design is limited, the reduction ratios (α_C) and the wakeup times (*wc*) for different clusters can be calculated and stored in a lookup table before finding the optimum clustering solution.

4.3. Impact of sleep transistor insertion on timing integrity

In this section we will review the impact of STI on the timing integrity of the sequential circuit. As STI modifies the propagation delays of the clock buffers to which it is coupled, there might be timing hazards as the clock signals reach the flip-flops at unacceptably different times. An example of such effect is depicted in:

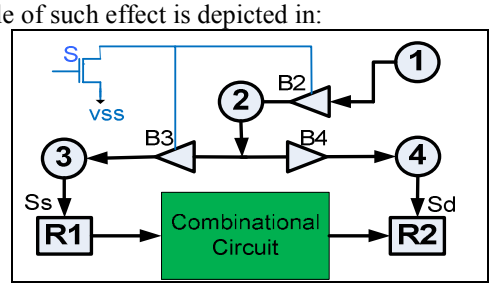

Figure 11. Impact of propagation delay on clock integrity

As shown in Figure 11, buffers B_2 and B_3 are sharing a sleep transistor. The output of flip-flop R_1 is fed to the combinational circuit and the output of the combination circuit is registered in flip-flop R_2. In this case, we refer to R_1 and R_2 as the source and destination flip-flops respectively. The signals S_s and S_d represent the clock branches triggering flip-flops R_1 and R_2 respectively.

The propagation delays from the clock source to the flip-flops change due to sleep transistor insertion and as a result the data integrity of the flip-flops might be violated. For example, in Figure 11, the propagation delay overhead for the clock branch S_s is more than the propagation delay overhead for S_d, since in the branch leading to S_s there are two buffers B_2 and B_3 sharing a sleep transistor, whereas in the branch leading to S_d only one buffer B_2 is connected to the sleep transistor. Although the results in Figure 6 suggests that in this simple example the propagation delay overhead is very low, sleep transistor insertion and sharing must be performed with caution in order to meet the clock skew constraints [22]:

$$(S_d - S_s) \geq D_{max} + t_S - T \quad (10)$$

$$(S_d - S_s) \leq D_{min} - t_H \quad (11)$$

Where t_S, t_H and T are the setup time, hold time and the clock period of the circuit. D_{min} and D_{max} are the minimum and maximum combinational circuit delays from the source flip-flop to the destination flip-flop. The constraints mentioned above must be held for every pair of source flip-flop and destination flip-flop being connected by a combinational circuit. Enforcing the constraint in inequality 10 prevents setup violation, while enforcing the constraint in inequality 11 prevents hold violation.

5. POST SYNTHESIS SLEEP TRANSISTOR INSERTION ALGORITHM (PSSTI)

The problem of post-synthesis clock sleep transistor insertion can be stated as:

Given a clock tree containing $X_1,\ldots, X_n$ clock buffers, the idle time patterns of the clock buffers, the leakage and dynamic power of clock buffers and sleep transistors, the wakeup delays (Figure 4), the reduction ratios (Figure 5) and the propagation delays (Figure 6) in lookup tables,

- Find the optimum partitioning of X into $C = \{C_1,\ldots,C_m\}$ clusters where the total power of the clock tree buffers is minimized
- Subject to the timing constraints defined on the clock tree.

The problem of clustering in general is NP-hard. We propose a heuristic algorithm to insert sleep transistors in the synthesized clock tree in order to optimize the power dissipation in clock tree buffers with no compromise in the integrity of the clock tree functionality.

According to eq. 9 and eq. 10, we are able to determine whether merging two clusters results in reduction in total power dissipation or not. We define merging gain (MG) of clusters C_1 and C_2 as:

$$MG(C_1, C_2) = P(C_1) + P(C_2) - P(C_1 \cup C_2) \quad (12)$$

In eq. 12, $P(C_1)$, $P(C_2)$ and $P(C_1 \cup C_2)$ are the total power consumptions of clusters C_1, C_2 and $C_1 \cup C_2$ (the product of merging) respectively.

Merging two clusters is performed by merging the set of clock buffers of the two clusters and updating the ITP of the new cluster according to eq. 2. In order to find the proper size for the sleep transistor, we pick the smallest available sleep transistor that meets the timing constraint so that the total power overhead of the sleep transistor is minimal. If $MG(C_1,C_2)$ is negative, it indicates that merging C_1 and C_2 results in power reduction. Otherwise, we will not consider merging C_1 and C_2 into a single cluster.

The highlight of our clustering algorithm is shown in Figure 12. In our algorithm, we initially assume that each individual clock buffer is clustered by its own and is coupled with a distinctive sleep transistor. In each iteration of our

algorithm, we calculate $MG(C_i, C_j)$ for every pair of clusters as stated in eq. 12 and compare the value with the best reduction value obtained so far. At the beginning of every iteration, the best reduction value is set to 0. If this merging leads to further reduction in the total power compared to previously obtained best reduction value, then the merging is further examined to see whether it leads to any skew violations in the clock tree or not. If the merging is skew-violation-free, then such merging is accepted and the best reduction value and the best merged cluster are saved.

We add the best cluster in terms of power reduction to the set of valid clusters and invalidate the two clusters constituting the best cluster. The termination condition of this algorithm is reached when there is no further merging either because there is no power reduction in merging clusters (merging inherently leads to less number of idle cycles and eventually more power consumption) or all the merging lead to clock skew constraint violation.

If Elmore delay model is employed, all the delays from clock source to the flip-flops can be calculated by traversing the tree network two times in $O(n)$ [23], where n is the number of elements used in the clock tree (buffers and wires). Once the source to flip-flop delays are calculated, the clock skew constraints can be examined in $O(m)$ to see whether there is any clock skew violation in the clock tree or not. The parameter m here refers to the number of clock skew constraints between source-destination pairs of flip-flops. Altogether, given a set of clock buffer clusters and the clock tree, the complexity of finding clock skew violations is $O(n + m)$.

```
Algorithm PSSTI:
        Inputs: Clock buffers X = { X1, ..., Xn } and their ITP set, the
        lookup tables for wakeup delay overheads, propagation delay
        overheads, leakage power reduction ratio (αX), leakage and
        dynamic power of the sleep transistors and clock buffers and
        clock skew constraints between every pair of source-
        destination flip-flops.
        Outputs: set of clusters C = { C1, ..., Cm }
merged ← true
For every Xi in X
  add Ci to C where Ci ← { Xi }
  Ci ← valid
While merged = true do {
  Best_reduction ← 0
  merged ← false
  For every valid Ci in C
    For every valid Cj in C , i ≠ j {
      Calculate MG(Ci , Cj)
      If MG(Ci , Cj) < Best_reduction then {
        Ck ← Ci ∪ Cj
        If no_skew_violation(Ck) then {
          Best_reduction ← MG(Ci , Cj)
          merged ← false
          Cbest ← Ck }}}
  If (merged = true)
    Add Cbest to C
    Ci ← invalid
    Cj ← invalid }
```

Figure 12. Outline of PSSTI

Calculating MG in the worst case takes $O(OP.\ n)$, where OP is the operation period and n is the number of clock buffers in the design, provided that all the information pertaining dynamic and leakage power consumptions is available in lookup tables, accessible in $O(1)$.

In order to analyze the complexity of PSSTI algorithm in Figure 12, we start with the two inner FOR loops, through which every pair of two clusters is examined (there are $O(n^2)$ of such pairs). In case the pair turns out to be skew-violation-free and also leads to the most power reduction, the pair is merged into a single cluster and is added to the list of valid clusters. In other words, in the worst case, we perform one merge for every pair with complexity of $O(OP.\ n)$ and we check the skew violations in $O(n + m)$. The total complexity of execution of the two inner FOR loops is $O(n^3)$. Since each execution of the two inner FOR loops results in a merging of a cluster pair, and there are initially n clusters, the number of times the inner FOR loops are executed is $O(n)$. Therefore we conclude that total complexity of PSSTI algorithm is $O(n^4)$.

6. EXPERIMENTAL RESULTS

In order to evaluate the performance of our proposed technique, we applied PSSTI algorithm on a subset of the popular academic DSP and multimedia benchmarks [24], which comprise of 8-bits multipliers, adders and registers. The experimental flow used in this work is depicted in Figure 13.

We first perform scheduling and resource allocation for each benchmark under minimal latency constraints so that we obtain the RTL level description netlist. Once scheduling and resource binding is performed, the idle time pattern of each flip-flop is extracted.

We explored the potential benefits of PSSTI using a standard VLSI-CAD design flow, by leveraging Synopsys Design Compiler [25] for synthesis and Synopsys Astro [26] for floorplanning, power planning, placement, routing and clock tree synthesis.

We have used standard cell from TSMC 45nm low power (LP) library. For clock tree construction, we have used high threshold voltage clock buffers as well. By doing so, we take into account all benefits of using approaches like [11] in which multi-threshold transistors for clock tree network is used. We also enabled hierarchical clock gating to reduce the dynamic power consumption.

We have utilized Synopsys Astro to generate different clock topologies to evaluate and understand the impact of clock topology on the effectiveness of PSSTI algorithm: For each RTL netlist, we generated four clock tree with maximum fan out of 2 (similar to binary clock tree), 4, 8 and 16. The clock tree information (including tree topology, buffer size, buffer location and buffer power dissipation) is then extracted once clock tree synthesis is done. All designs are synthesized and placed and routed for 500 Mhz clock frequency. The summary of design constraints is shown in Table 2. Once the clock information is extracted, it is fed into the STI search engine, which is an implementation of PSSTI clustering algorithm. The search engine tries to cluster the clock buffer together so that all the clock buffers in a cluster share a sleep transistor. The main

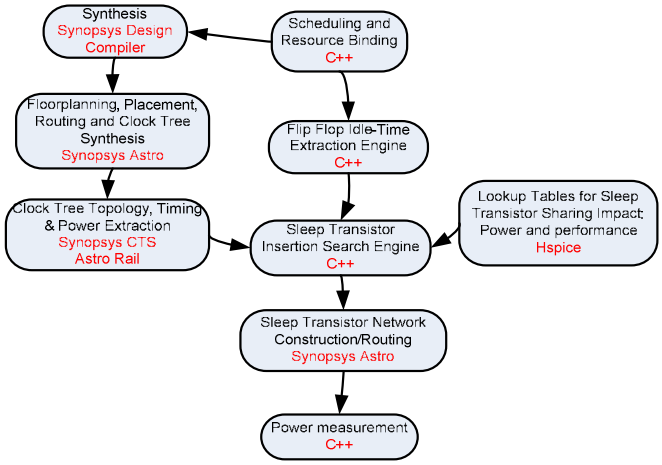

Figure 13. Experimental flow

objective of the search engine is to minimize the total power consumption of the clock tree without any clock skew violation. The lookup tables (Figure 13) used in the search engine contain information regarding the power benefits of sleep transistor sharing, as well as propagation delay overheads and wakeup time overheads on the clock buffers. In the lookup tables we store the power benefit of sleep transistor sharing for different cluster sizes, different clock buffer sizes and different sleep transistor sizes. The results obtained using HSPICE simulation and the same methodology explained in Sec. 3.

Once the clusters are identified, the sleep transistor network is constructed and routed. Note that this network is routed to drive the sleep transistors. We use a separate netlist and an additional place and routing flow to construct this network.

We calculated the power consumption of the clock tree in two cases. In the first case, we assume no sleep transistor is inserted in the clock tree. In the second case, we apply our PSSTI algorithm on the clock tree to cluster the clock buffers and insert sleep transistors for each cluster. For power calculation we have taken into account the sleep transistor leakage and switching power dissipation as well as the additional routing network to drive them. Since the clock tree is constructed using high threshold voltage clock buffers, all power benefits of using low power cells are already taken into account (similar to [11]).

Table 2. Design Synthesis Constraints

Technology	TSMC45-LP	**Synthesis Frequency**	550 MHz
Max Core Utilization	80%	**Place and route frequency**	500 MHz
Max Allowed Routing Congestion	2%	**Clock Tree Max FanOut**	2, 4, 8, 16
Hierarchical Clock Gating	Enabled	**Target Skew**	100 ps
Clock buffers	CLKBUFFX2, CLKBUFFX3, CLKBUFFX4, CLKBUFF8 CLKBUFF12, LKBUFF16, CLKBUFFX24, CLKBUFFX32		

The results of our experiments are shown in Table 3. For each benchmark and for each maximum fan out constraint (2, 4, 8 and 16), we have extracted the maximum clock skew and total power consumption of the clock trees. It also contains the number of clock buffers in the clock tree and the number of clock buffer clusters calculated using our STI search engine. As reported, there is a significant reduction in the leakage power of clock trees after sleep transistor insertion. In fact we reached up to 40%, 36%, 28% and 22% reduction in total leakage power consumption of the clock tree when sleep transistors are inserted in netlists with maximum fan outs of 2, 4, 8 and 16 respectively. On average the leakage power improvements are 32%, 29%, 24%, 19%.

Table 3. Experimental results

Benchmark	No. of clock buffers				No. of clusters				Baseline clock max. skew (ps)				Baseline total power (mW)				Leakage power reduction (%)				Total power reduction (%)			
	Clock tree fan out				Clock tree fan out				Clock tree fan out				Clock tree fan out				Clock tree fan out				Clock tree fan out			
	2	4	8	16	2	4	8	16	2	4	8	16	2	4	8	16	2	4	8	16	2	4	8	16
arf	47	24	8	4	15	8	3	2	24	27	16	8	0.23	0.22	0.20	0.20	24.4	22.9	14.3	13.7	7.7	7.7	2.8	0.8
horner	30	15	6	3	9	5	2	1	24	22	21	10	0.21	0.20	0.19	0.19	33.8	30.9	26.7	11.9	15.8	15.6	10.7	1.2
fir1	40	21	7	4	13	8	3	2	20	26	25	16	0.19	0.18	0.16	0.16	17.4	15.0	9.8	9.2	5.8	4.9	2.0	1.2
motion	80	39	17	7	27	13	6	3	26	27	17	16	0.28	0.26	0.24	0.23	30.1	27.4	24.9	25.0	12.9	12.3	10.1	8.3
ewf	62	33	13	6	17	10	5	2	30	18	16	27	0.33	0.31	0.30	0.29	37.0	32.5	24.0	14.9	20.3	18.9	9.8	4.3
fir2	40	21	7	4	14	7	3	2	12	32	26	8	0.15	0.13	0.12	0.12	22.5	20.8	14.9	11.8	9.1	9.2	5.0	2.5
feedback	95	47	19	8	22	13	7	3	50	26	25	50	0.46	0.43	0.41	0.40	35.8	30.8	22.4	16.4	19.2	17.6	10.3	5.2
cosine1	111	58	23	10	26	14	7	4	43	49	33	51	0.60	0.57	0.54	0.52	41.1	40.6	34.8	29.6	21.8	26.9	19.3	11.3
h2v2	40	21	7	4	9	7	3	2	13	26	20	21	0.37	0.36	0.34	0.34	35.9	30.4	25.0	18.1	19.2	17.3	11.2	5.3
cosine2	111	58	23	10	28	15	7	4	41	21	25	71	0.61	0.58	0.55	0.53	38.7	38.3	32.0	31.0	19.0	23.1	16.7	12.2
collapse	111	58	23	10	37	21	8	4	30	32	15	55	0.58	0.55	0.52	0.50	30.5	25.5	22.4	21.0	12.4	10.2	8.2	5.8
interpolate	201	101	40	20	59	31	14	7	44	27	47	12	0.84	0.78	0.73	0.72	28.1	25.9	23.2	17.4	12.9	12.5	10.2	5.5
matmul	119	61	24	10	41	21	8	4	40	51	29	56	0.93	0.90	0.87	0.85	23.0	21.0	20.0	17.3	9.5	9.0	8.7	5.9
jpeg_fdct	160	80	32	14	39	21	10	5	38	17	11	62	1.24	1.19	1.15	1.13	33.1	29.3	22.8	19.7	17.6	17.8	11.8	8.7
idctcol	191	97	38	16	39	25	11	5	41	27	28	48	1.54	1.48	1.44	1.41	41.3	35.3	31.8	22.6	24.4	24.0	19.7	10.0
smooth	254	129	51	26	51	30	14	8	44	33	38	13	1.62	1.54	1.48	1.47	42.2	39.8	36.6	28.2	24.2	28.0	23.1	11.6
Average	**105.8**	**53.9**	**21.1**	**9.8**	**27.9**	**15.6**	**6.9**	**3.6**	**32.5**	**28.8**	**24.5**	**32.8**	**0.63**	**0.60**	**0.57**	**0.56**	**32.2**	**29.2**	**24.1**	**19.2**	**15.7**	**15.9**	**11.2**	**6.2**

The maximum (average) total power reduction obtained through sleep transistor insertion for maximum fan outs of 2, 4, 8 and 16 are 25% (16%), 27% (16%), 23% (11%) and 13% (6%). As clock trees with maximum fan out of 2 use more clock buffers compared to clock trees with higher fan outs, it is more likely to find clock buffers with similar idle time patterns, especially for the clock buffers closer to the sinks, which leads to less dynamic power overhead for the sleep transistor. Since the idle time pattern of each clock buffer is in fact dependant on the idle time patterns of all the children clock buffers (as stated in eq. 1), for high fan out clock trees, the sleep transistors have to toggle between on and off more frequently which results in more sleep transistor dynamic power consumption. Therefore, for lower fan out clock trees the overall power reduction is more significant.

Our experiments show that the maximum allowable routing congestion was met after sleep transistor insertion. The core utilization also increased negligibly yet met the design constraints. This indicates the sleep transistor insertion and routing increases the area overhead insignificantly. The clock skew variation caused by STI is very negligible. No clock skew violation was reported after sleep transistor insertion. The maximum skew observed in the clock trees after applying STI was increased by 10%, which satisfies clock skew constraints of the circuit.

7. CONCLUSION

In this paper we have proposed sleep transistor insertion in a clock tree in order to reduce leakage power. We characterized the effect of sleep transistor sharing and sizing on clock tree wakeup time, leakage power, and propagation delay. We have developed an algorithm which utilizes these characteristics to reduce the leakage and the total power dissipation through clustering clock buffers and sharing sleep transistors. We have also combined our sleep transistor insertion engine to the standard design synthesis flow for ASIC designs (using industrial synthesis tools), including data flow graph scheduling, resource binding, design synthesis, placement and routing. Having explored the power consumption of clock trees with different fan outs, we noticed significant reductions in total power and leakage power, by 16% and 32% on average respectively.

8. References

[1] M. Donno, E. Macci, and L. Mazzoni, “Power-Aware Clock Tree Planning,” Proc. ACM/IEEE Int. Symp. Physical Design, 2004, pp. 138-147.

[2] N. Magen, A. Kolodny, U. Weiser, and N. Shamir, “Interconnect-power Dissipation in a Microprocessor,” Proc. Workshop on System Level Interconnect Prediction, 2004, pp. 7-13.

[3] Nam Sung Kim, Todd Austin, David Blaauw, et al, “Leakage current: Moore’s law meets static power”, IEEE Computer Society, pp. 68-75, 2003.

[4] Walid M. Elgharbawy, Magdy A. Bayoumi, “Leakage sources and possible solutions in nanometer CMOS technologies”, IEEE Circuits and Systems Magazine, pp. 6-17, 2005.

[5] V. Adler, E. G. Friedman, “Repeater Insertion toReduce Delay and Power in RC Tree Structures,” IEEE Asilomar Conference on Signals, Systems and Computers, pp. 749-752, Pacific Grove, CA, November 1997.

[6] J. Cong, C.-K. Koh; K.-S. Leung, “Simultaneous Buffer and Wire Sizing for Performance and Power Optimization,” ISLPED-96: ACM/IEEE International Symposium on Low-Power Electronics and Design, pp. 271-276, Monterey, CA, August 1996.

[7] Vittal, M. Marek-Sadowska, “Low-Power Buffered Clock Tree Design,” IEEE Transactions on CAD/ICAS, Vol. 16, No. 9, pp. 965-975, September 1997.

[8] M. A. El-Moursy, E. G. Friedman, “Exponentially tapered H-tree clock distribution networks”, IEEE Trans. VLSI Syst. 2005.

[9] M. Igarashi, et al.“A Low-Power Design Method using Multiple Supply Voltages,” ISLPED-97.

[10] Weixiang Shen, et al, “Activity-aware registers placement for low power gated clock tree construction”, in Proc ISVLSI, pp. 383-388, 2007.

[11] Weixiang Shen, et al., ”Leakage Power Optimization for Clock Network Using Dual-Vth Technology”, ISCAS 2008.

[12] Yongseok Cheon, et al., “Power-Aware Placement”, in DAC 2005.

[13] Jaewon Oh and Massoud Pedram, “Gated clock routing for low-power microprocessor design”, IEEE Transactions on CAD/ICAS, Vol. 20, No. 6, pp. 715-722, June, 2001.

[14] Monica Donno, Alessandro Ivadldi, Luca Benini, Enrico Macii, “Clock-tree power optimization based on RTL clockgating”, in Proc. DAC, pp. 622-627, 2003.

[15] Ruchir Puri, Leon Stok, John Cohn, et al, “Pushing ASIC performance in a power envelope”, in Proc. DAC, pp. 788-793, 2003.

[16] Mongkol Ekpanyapong, Sung Kyu Lim, “Integrated retiming and simultaneous Vdd/Vth scaling for total power minimization”, in Proc. ISPD, pp. 142-148, 2006.

[17] David Chinnery and Kurt Keutzer, “Closing the Power Gap Between ASIC & Custom”, Springer US.

[18] H. Homayoun and A. Veidenbaum, ZZ-HVS: Zigzag Horizontal and Vertical Sleep Transistor Sharing to Reduce Leakage Power in On-Chip SRAM Peripheral Circuits, ICCD 2008.

[19] J. Kao, S. Narendra, and A. Chandrakasan, “MTCMOS hierarchical sizing based on mutual exclusive discharge patterns,” DAC, June 1998.

[20] J. M. Rabaey et al., Digital integrated circuits: a design perspective, Prentice Hall, Second. Edition, 2003.

[21] K. Agarwal, H. Deogun, D. Sylvester, K. Nowka. Power gating with multiple sleep modes. In ISQED 2006.

[22] J. P. Fishburn, Clock Skew Optimization, IEEE transactions on computers, V. 39, N. 7, P. 945-951, July 1990

[23] R. Gupta, et al., The Elmore delay as a bound for RC trees with generalized input signals, in DAC 1995.

[24] http://www.ece.ucsb.edu/EXPRESS/benchmark/

[25] Design Compiler Ultra, Synopsys Incorporation

[26] Astro synthesis tool, Synopsys Incorporation