Our reference: SUSCOM 5 P-authorquery-v8

AUTHOR QUERY FORM

ELSEVIER	**Journal:** SUSCOM **Article Number:** 5	**Please e-mail or fax your responses and any corrections to:** **E-mail:** corrections.esch@elsevier.thomsondigital.com **Fax:** +353 6170 9272

Dear Author,

Please check your proof carefully and mark all corrections at the appropriate place in the proof (e.g., by using on-screen annotation in the PDF file) or compile them in a separate list.

For correction or revision of any artwork, please consult http://www.elsevier.com/artworkinstructions.

Any queries or remarks that have arisen during the processing of your manuscript are listed below and highlighted by flags in the proof. Click on the 'Q' link to go to the location in the proof.

Location in article	**Query / Remark: click on the Q link to go** **Please insert your reply or correction at the corresponding line in the proof**
	Reference(s) given here were noted in the reference list but are missing from the text – please position each reference in the text or delete it from the list.
Q1	Please check the placements of Figs. 3, 12 and Table 4, and correct if necessary.
Q2	Uncited references: This section comprises references that occur in the reference list but not in the body of the text. Please position each reference in the text or, alternatively, delete it. Any reference not dealt with will be retained in this section. Thank you.
Q3	Please provide the year of publication for Ref. [9,17].
Q4	Please check the year of publication for Ref. [36].

Thank you for your assistance.

Sustainable Computing: Informatics and Systems xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

On leakage power optimization in clock tree networks for ASICs and general-purpose processors

Houman Homayoun[a], Shahin Golshan[b], Eli Bozorgzadeh[b], Alexander Veidenbaum[b,*], Fadi J. Kurdahi[c]

[a] Dept. of Computer Science and Engineering, University of California, San Diego, CA, United States
[b] Dept. of Computer Science, University of California, 3056 Bren Hall, Irvine, CA, United States
[c] Dept. of Electrical Engineering and Computer Science, University of California, Irvine, CA, United States

ARTICLE INFO

Article history:
Received 19 October 2010
Received in revised form 29 October 2010
Accepted 29 October 2010

Keywords:
Sleep transistor
Clock tree network
Leakage power reduction

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. This paper proposes to deploy sleep transistor insertion (STI) in the clock tree of datapaths in ASICs or in general-purpose processors in order to reduce leakage power. It characterizes the effect of sleep transistor sharing and sizing on clock tree wakeup time, leakage power, and propagation delay. It then uses these characteristics during leakage power optimization. It describes a 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. The potential benefits of STI in ASIC design are evaluated using a standard industrial VLSI-CAD flow including sleep-transistor insertion and routing after the clock synthesis and place-and-route of the benchmark circuits. The results show that the clock tree leakage power is reduced by 19–32% depending on the topology of the synthesized clock tree. We also apply PSSTI in the clock tree of the datapaths in general-purpose processors using architectural control of the sleep mode. This achieves a reduction in the leakage power and the dynamic power of the clock tree within different datapath components (adder, multiplier, etc.) of as much as 80%. This approach is also applicable to other on-chip structure where inverters are large in size and commonly used, e.g. SRAMs or networks-on-a-chip, which combined with the clock tree savings will significantly reduce processor or ASIC overall power consumption.

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. Several approaches at technology, circuit, and architectural levels have been proposed to overcome the growing leakage problem 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 a large number of inverter buffers [1–4,17,27]. Clock tree network is also responsible for a large portion of power dissipated in general purpose processors. The Intel Xscale processor clock tree dissipates 17% of the total chip power [27] and the high-performance Alpha processor clock tree dissipated twice as much power. The leakage power reduction in clock tree networks is thus a very important problem in digital systems and is the focus of this paper.

Recently a number of approaches were proposed to reduce the power of clock trees. Clock gating has been extensively studied and applied to reduce power by masking off clock signal where branches of clock tree are idle [1,13,14]. Clock buffer sizing [5–7], power-aware placement [10,12], exponentially tapering clock interconnect network [8] and use of multiple-supply voltages [9] have been proposed for low power clock tree construction. 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. The significant reduction in dynamic power achieved by applying them in clock trees has lead to leakage power becoming a large fraction of total clock tree network power. Hence, leakage power optimization of clock tree is equally, if not more important to address in a clock tree network.

For leakage power reduction, several techniques such as supply and threshold voltage optimization, sleep transistor insertion

[*] Corresponding author. Tel.: +1 949 824 6188; fax: +1 949 824 4056.
E-mail address: alexv@ics.uci.edu (A. Veidenbaum).

2210-5379/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.suscom.2010.10.005

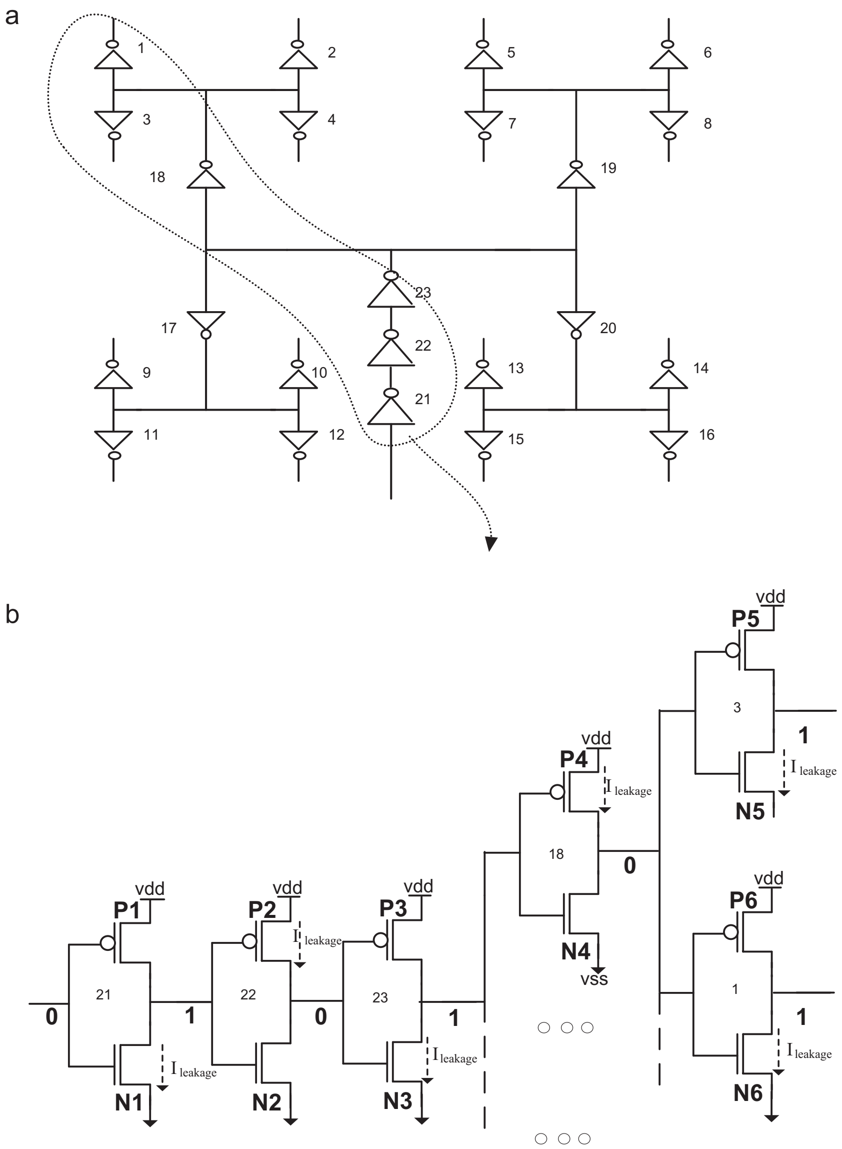

Fig. 1. (a) H-tree clock network and (b) source of leakage.

and power gating have been proposed and extensively studied in literature [3,15,16]. The approach proposed in this paper is based on sleep transistor insertion (STI) in clock tree networks to reduce leakage power. STI is a well known technique for reducing leakage of an idle unit by isolating it from V_{dd} and V_{ss}. STI has been extensively applied in many on-chip blocks such as SRAM memories [18,21]. However, to the best of our knowledge, [31] was the first attempt to deploy STI in a clock tree network to reduce its leakage power. The proposed leakage optimization through sleep transistor insertion is an orthogonal approach to using high V_{th} gates in a clock tree [11] and our results show that it achieves a significant leakage reduction in clock tree networks.

Fig. 1(a) shows an example of an H-tree clock network, which uses inverter buffer chains to drive the clock signal from the source to each flip-flop (sink). The source of sub-threshold leakage in the H-tree clock network is illustrated in Fig. 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, STI introduces an area overhead and increases the propagation delay of the clock tree due to an increase in the rise/fall time of the drivers.

In order to overcome these negative effects on the critical timing components, this work proposes deployment of zigzag sleep transistor insertion technique based on a recently proposed circuit design 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 identify the idle time patterns of different clock tree buffers during the course of execution.

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005

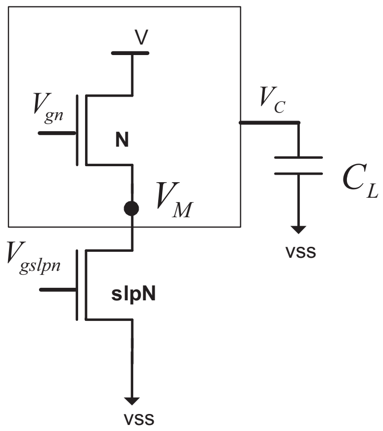

Fig. 2. Inserting sleep transistor to reduce leakage.

This paper evaluates the benefits of sleep transistor insertion and sharing in clock tree networks. It makes the following contributions:

- It characterizes the effects of sleep transistor sharing and sizing on clock tree wakeup time, leakage power, propagation delay and skew.
- It presents 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.
- It explores the potential benefits of PSSTI for standard industrial VLSI CAD flow in ASIC data-path design and in datapath components of general-purpose processors.
- It describes an approach to activate/deactivate the sleep transistors in the datapath for cases when idle/active times of operations do not follow a fixed schedule (e.g. in general-purpose processors).

The experimental results show a significant reduction in total power and leakage power, by 16% and 32%, respectively and on average, in the clock tree of the data-path designs in ASIC. The technique lowers the leakage power dissipation of the clock trees by as much as 80%, on average when applied on the datapath components of the general-purpose processor.

2. Sleep transistor insertion

Sleep transistor insertion was proposed to reduce the subthreshold (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 Fig. 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].

3. Reducing leakage in an inverter chain

We investigate how sleep transistor stacking can be applied to reduce subthreshold leakage in a chain of inverters where used extensively in the clock tree logic. First, we analyze the source of subthreshold leakage in the chain of inverters. An analysis of subthreshold leakage sources in a chain of inverters implemented as a four-stage inverter (buffer) chain (shown in xxx) is performed and a solution to reduce its leakage is proposed. The number of inverter stages is chosen to meet timing requirements. Assume that the inverter chain has to drive a logic value 0. Thus transistors *N*1, *N*3 and *P*2, *P*4 are in the off state and thus they are leaking. To reduce the leakage in these transistors we apply sleep transistor stacking in the inverters chain. Due to induced negative source to gate biasing, subthreshold leakage current in the inverter chain reduces significantly. However, rise and fall times and propagation delay of the circuit are affected. Next, we discuss several approaches to minimize this effect while maximizing leakage reduction (Fig. 3). **Q1**

3.1. A redundant circuit approach (“fine grained” sleep transistor insertion)

In this technique, PMOS and NMOS sleep transistors are stacked at the finest granularity level in the inverter chain to reduce the chain’s leakage during the off-state (Fig. 4(a)). We refer to this approach as the redundant circuit scheme. As discussed above, inserting the sleep transistor introduces metastability in the circuit, since high voltage nodes are discharged slightly during the sleep mode. To eliminate metastability, when the circuit is in sleep mode, we propose to insert a minimum size NMOS transistor in the last stage of the inverter chain [34], as shown in Fig. 4(a) (we refer to this transistor as a state preserving-transistor). During the sleep mode the state-preserving transistor is turned on to maintain the state of the inverter chain output by pulling it down to the ground.

The drawback of the redundant circuit technique is its impact on inverter chain output rise time, fall time, and hence, the propagation delay.

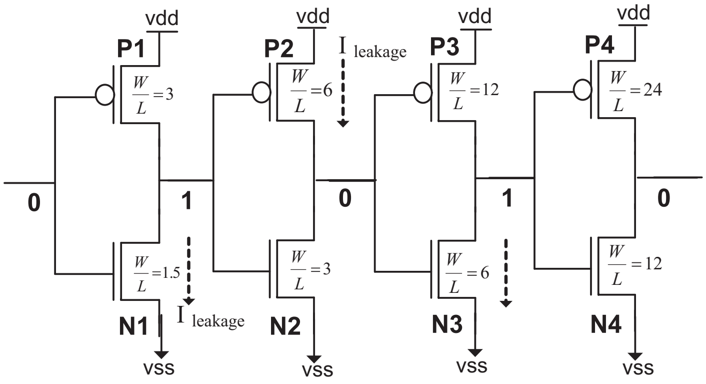

Fig. 3. Source of leakage in an inverter chain.

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005

The rise time and fall time of an inverter output is proportional to the Rpeq*CL and Rneq*CL, respectively, where Rpeq is the equivalent resistance of the PMOS transistor, Rneq is the equivalent resistance of the NMOS transistor, and CL is the equivalent output capacitive load. Inserting sleep transistors increases both Rneq and Rpeq, and thus the rise time and fall times and propagation delay of the inverter chain.

3.2. A zig-zag circuit

To alleviate the drawback of redundant scheme on an inverter chain rise time, fall time and propagation delay, the sleep transistors are inserted in a zig-zag fashion, as shown in Fig. 4(b). By inserting the sleep transistors in a zig-zag fashion the Rpeq for the first and third inverters and Rneq for the second and fourth inverters doesn't change. Therefore the fall time of the circuit does not change compared to the baseline circuit with no leakage control, unlike the redundant circuit.

Unlike the fall time, the rise time of the circuit is affected by the zig-zag scheme. This is due to the fact that the rise time of the circuit is determined by the fall time of the output of the first and third stage inverters and the rise time of the output of the second and fourth inverters, which both increase due to insertion of sleep transistors.

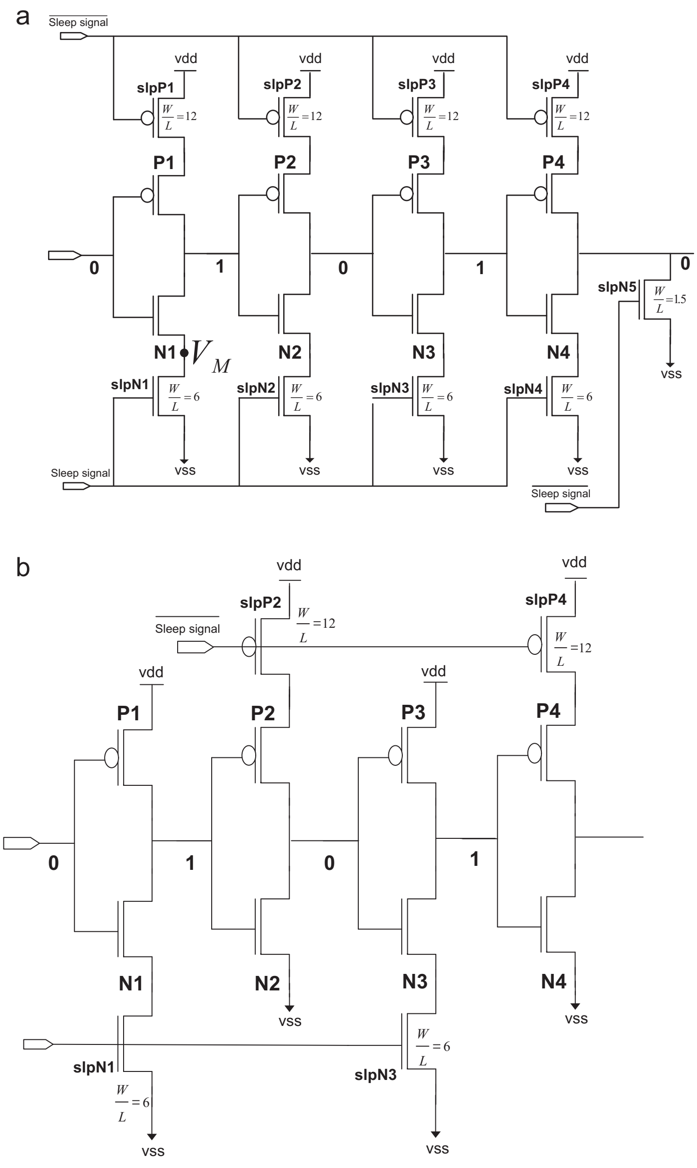

Fig. 4. (a) Redundant circuit and (b) zig-zag circuit.

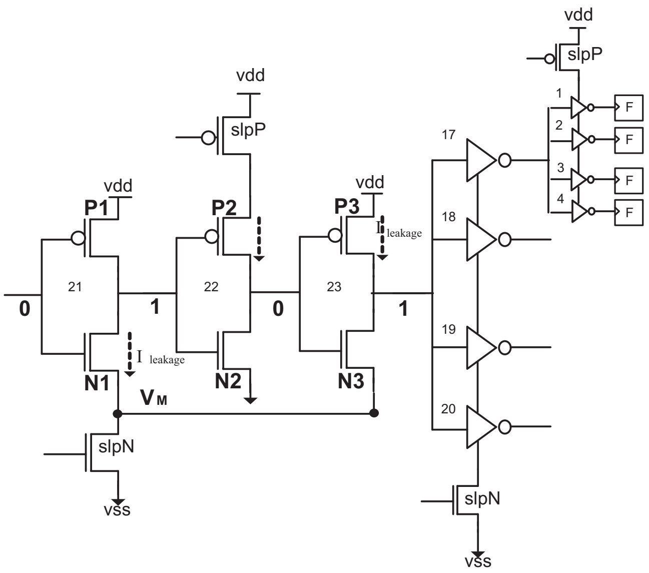

Fig. 5. Zigzag sleep transistor sharing in clock tree network.

To minimize the impact on the rise time, one can resize the sleep transistors. Increasing the size of sleep transistor would reduce its equivalent resistance and thus the inverter chain rise time, while it reduces the leakage savings. This is due to the fact that increasing the size of sleep transistor reduces the virtual ground node voltage (V_M in Fig. 4(a)), which in turn reduces the leakage savings. In addition, using one sleep transistor per inverter logic increases the area for the zig-zag scheme.

4. 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 Fig. 5, 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 Fig. 5) 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 $\sum_i C_{\text{wire}}(i) + C_{\text{diff}}$, where the $C_{\text{wire}}(i)$ 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.

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 Fig. 1. We assume a symmetric distribution of flip-flops.

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005

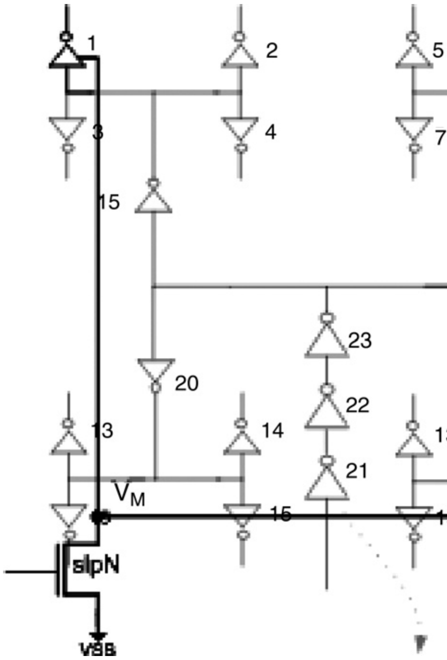

Fig. 6. Sleep transistor insertion for worst case wakeup delay.

We designed a 2-level H-tree clock network in a 0.1 × 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 $C_{\text{int}} = (2C_{\text{coupling}} + \omega \times C_{\text{coupling}}) \times l$, 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 2× wire is assigned to the first level of clock tree while a 1× 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 $\eta = C_{\text{out}}/C_{\text{in}}$ 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 45 nm technology using Synopsys Hspice at typical corner (25°) and the supply voltage of 1 V.

Table 1 reports 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 Fig. 6 for buff1 and buff16.

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.

The following equation expresses the switching delay of the sleep transistor as a factor of number of shared buffers: $\sum R_{\text{eq}} \times (C_{\text{wire}}(i) + C_{\text{diff}})$. 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.

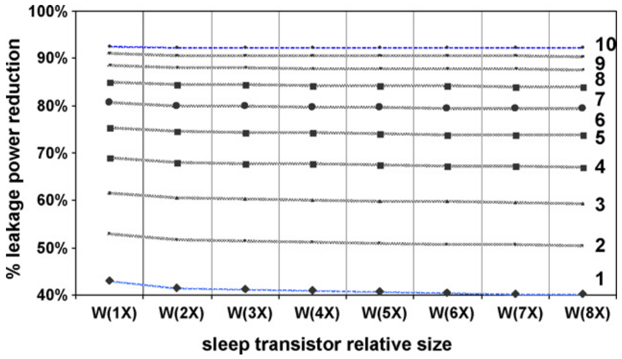

Fig. 7. Relative leakage power reduction.

In Fig. 7 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 Fig. 7).

In Fig. 8, 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.

5. 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.

5.1. 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

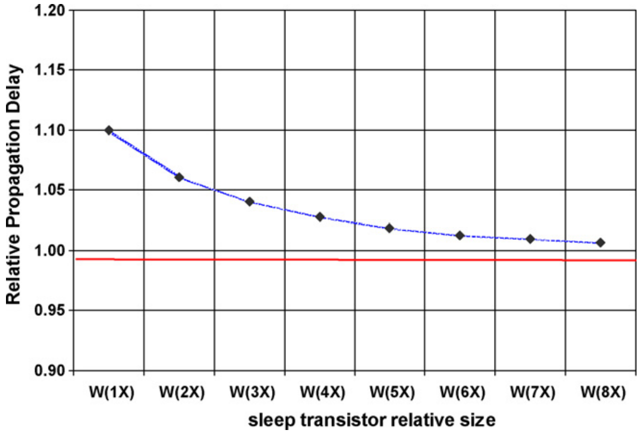

Fig. 8. Impact of sleep transistor sizing on propagation delay.

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005

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

# of 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

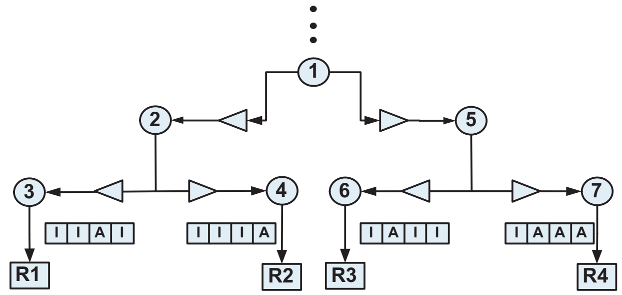

Fig. 9. Abstract model of clock tree and the ITP each node.

drives the signal from the source point to sinks (Fig. 9). As shown in Fig. 9, 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 Fig. 9 is 4 cycles. We assume that the circuit resumes operation once the outputs of the circuit are computed.

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 Fig. 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 Fig. 9, 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):

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

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. Fig. 8 illustrates the bottom–up calculation of ITP for a parent node based on its children nodes.

We define an idle pulse (IP) within an ITP as a range of time over which the clock buffer is idle. For example, in Fig. 8, the ITP of node 2 contains an idle pulse over the circularly contiguous range (*t*3, *t*1). 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:

As shown in Fig. 11, the *i*thelement 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* (Fig. 12).

Lemma. *ITP sequence generation produces a safe sequence.*

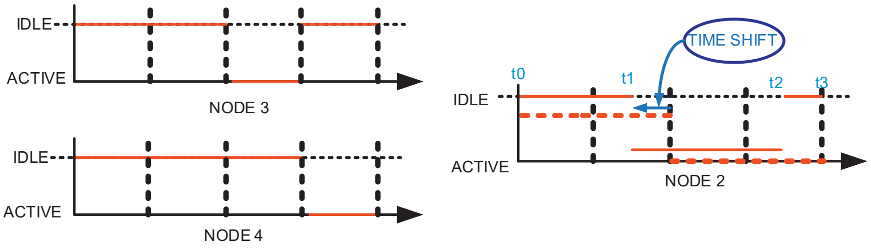

Fig. 10. Example of bottom–up ITP generation.

```
The algorithm for 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 $ITP_X((i+1 \text{ mod OP})\,.T) = 1$ then
      $ITP_X$[i] ← 1 ///idle
  else
      $ITP_X$[i] ← 0 ////active
```

Fig. 11. Algorithm for ITP sequence generation.

Proof. As shown in Fig. 11, 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 Fig. 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.

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:

$$ITP_C[k] = \prod\nolimits_{X_i \in C} ITP_{X_i}[k] : \forall k : 0 \le k < OP \tag{2}$$

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.

5.2. 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):

$$P(X, S) \;=\; LP(X, S) \;+\; DP(X, S) \tag{3}$$

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:

$$EIC(X) = \sum\nolimits_{IP_i \in ITP_X} (\text{length}(IP_i) - wc) \tag{4}$$

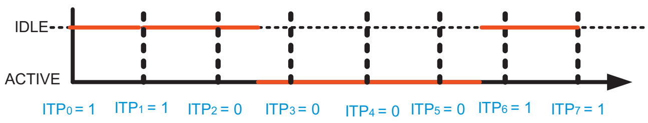

Fig. 12. An example of ITP sequence generation.

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

$$LP(X, S) = \frac{LP_S + LP_X \cdot \left\{(1-\alpha_X)\cdot EIC(X) + [OP - EIC(X)]\right\}}{OP} \tag{5}$$

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 $(1 - \alpha_X)\cdot LP_X$ 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 (Fig. 7). 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):

$$tgl(X) = \sum_{i=1}^{OP} \left| ITP(X, i + 1\text{mod}OP) - ITP(X, i) \right| \tag{6}$$

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:

$$DP(X, S) = \frac{[DP_S \cdot tgl(X) + DP_X \cdot (OP - EIC(X))]}{OP} \tag{7}$$

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). Eqs. (8) and (9) are extensions to leakage and dynamic power formulations expressed in Eqs. (5) and (7):

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

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

The parameters $tgl(C)$ and $EIC(C)$ are calculated using Eqs. (4) and (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.

So far in this section, the idle-time patterns for the clock buffers in the clock tree were analyzed under the assumption that the scheduling of operations is fixed (e.g. data-path design for ASIC circuits use a fixed controller to enable/disable data-path components). However, in a general-purpose processor, the operation of the data-path components does not follow a fixed pattern (i.e. it is dependent on the program being executed). Therefore, in such cases the average effective-idle-cycles EIC (Eq. (4)) and the average number of toggles tgl (Eq. (5)) over the whole execution cycles of all the profiled test benches on the processor is used. Similarly, the EIC of a cluster of buffers is calculated by averaging the EICs of the cluster over all the profiled test benches on the processor. This

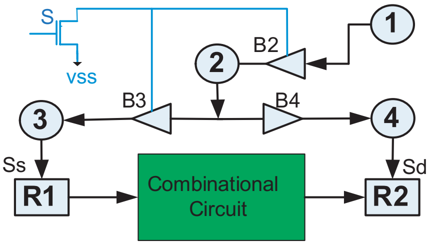

Fig. 13. Impact of propagation delay on clock integrity.

approach is used to perform PSSTI for the datapath components of a general-purpose processor.

5.3. 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:

As shown in Fig. 13, 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 Fig. 13, 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 Fig. 8 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]:

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

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

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.

6. 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 (Fig. 6), the reduction ratios (Fig. 7) and the propagation delays (Fig. 8) 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 Eqs. (9) and (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:

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

```
Algorithm PSSTI:
Inputs: Clock buffers $X = \{ X_1, ..., X_n \}$ and their ITP set, the lookup tables for wakeup delay overheads, propagation delay overheads, leakage power reduction ratio ($\alpha_X$), 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 $C = \{ C_1, ..., C_m \}$
merged ← true
For every $X_i$ in $X$
  add $C_i$ to $C$ where $C_i$ ← $\{ X_i \}$
  $C_i$ ← valid
While merged = true do {
   Best_reduction ← 0
   merged ← false
   For every valid $C_i$ in $C$
      For every valid $C_j$ in $C$ , $i \neq j$ {
         Calculate $MG(C_i , C_j)$
         If $MG(C_i , C_j) < Best\_reduction$ then {
      $C_k \leftarrow C_i \cup C_j$
      If no_skew_violation($C_k$) then {
   Best_reduction ← $MG(C_i , C_j)$
         merged ← false
         $C_{best} \leftarrow C_k$ }}}
  If (merged = true)
     Add $C_{best}$ to $C$
     $C_i$ ←      invalid
    $C_j$ ←      invalid }
```

Fig. 14. Outline of PSSTI.

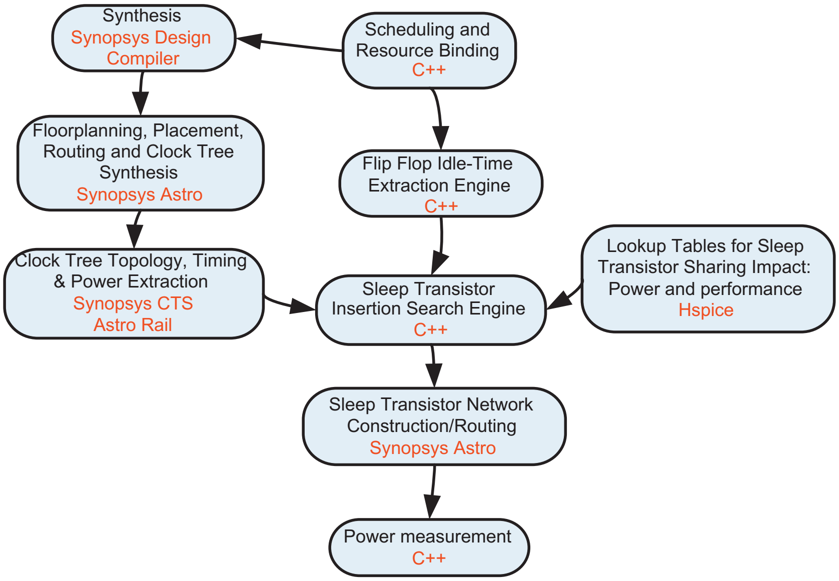
Synthesis
Synopsys Design Compiler
Scheduling and Resource Binding
C++
Floorplanning, Placement, Routing and Clock Tree Synthesis
Synopsys Astro
Flip Flop Idle-Time Extraction Engine
C++
Lookup Tables for Sleep Transistor Sharing Impact: Power and performance
Hspice
Clock Tree Topology, Timing & Power Extraction
Synopsys CTS
Astro Rail
Sleep Transistor Insertion Search Engine
C++
Sleep Transistor Network Construction/Routing
Synopsys Astro
Power measurement
C++

Fig. 15. Experimental flow.

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 Fig. 14. 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 $O(n+m)$.

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 Fig. 14, 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 $O(n^4)$.

7. Experimental results, case 1

The first case study of our proposed technique, applied the PSSTI algorithm on top of standard ASIC synthesis flow for data-path design. The test benches used in this section are a subset of the popular academic multimedia benchmarks [24], which comprise of 8-bits multipliers, adders and registers. The experimental flow used in this work is depicted in Fig. 15. 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 45 nm 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.

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 fan out	2, 4, 8, 16
Hierarchical clock gating	Enabled	Target skew	100 ps
Clock buffers	CLKBUFFX2, CLKBUFFX3, CLKBUFFX4, CLKBUFF8 CLKBUFF12, LKBUFF16, CLKBUFFX24, CLKBUFFX32		

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 objective of the search engine is to minimize the total power consumption of the clock tree without any clock skew violation. The lookup tables (Fig. 15) 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 Section 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]). 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%. 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.

8. Reducing embedded processor data path clock tree leakage, case 2

The previous section described the impact of the proposed technique in the standard ASIC synthesis flow for data-path design. This section applies our PSSTI technique to clock tree networks of processors' data-path components. It uses Intel Xscale like processor, an in-order issue embedded processor. Intel Xscale's clock tree dissipates 17% of total chip power as shown in [27].

We thus apply PSSTI to the clock tree network running across the functional units of Intel Xscale processor. This is done by inserting the sleep transistors in the clock network and using PSSTI for merging and sizing of the sleep transistors. Unlike ASIC data-path designs, where the scheduling of the operations is fixed and therefore can be inferred statically, for general-purpose processors the active times of the data-path components has to be studied over a wide range of applications running on the processors. Therefore, an important question to be answered is how to control the PSSTI, i.e. when to assert, de-assert the sleep signal to the sleep transistors' network. Let us first describe our experimental methodology used to analyze the problem in the data-path of general-purpose processors and then introduce the architectural approach to control PSSTI and reduce leakage power in the data-path components of the processor.

8.1. Experimental methodology

The percentage of execution cycles a functional unit clock tree network is being shut down and the leakage power reduction are used to evaluate the effect of PSSTI. A subset of MiBench benchmark suite [28] compiled for MIPS instruction set is used. Since the exact wakeup delay of the clock tree network is dependent on the topology of the clock tree, which is determined by the floorplan of the processor, functional units design specs and other parameters not available to us, a range of wakeup delays from 0 to 4 processor cycles is evaluated (intuitively based on the results in Table 1). An extensively modified version of SimpleScalar v3.0 toolset [29] is used to model an in-order embedded processor with an architecture similar to Intel's XScale core (Table 4).

One possible approach in implementing power gating of a functional unit is to monitor its state and turn it off after seeing a number of consecutive idle cycles. This is referred to as the time-based power gating technique [30]. It gates a functional unit when

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005

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

Table 4
Processor configuration.

Issue width	In-order:2	Inst/data TLB	32-Entry, fully-associative
Functional units	1 I-ALU, 1 F-ALU,1 I-MUL/DIV 1 F-MUL/DIV	L1 – Instruction/ Data Caches	32K, 32-way SA, 32-byte blocks, 1 cycle
BTB	128 entries	L2 Cache	None
Branch predictor	Bimodal, 128 entries	Register Update Unit	8 entries
Main memory	Infinite, 32 cycles	Load/Store queue	8 entries

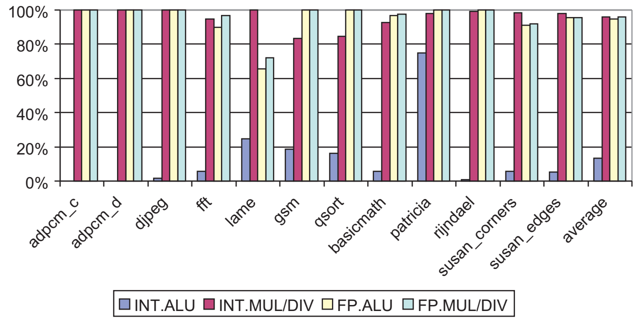

Fig. 16. Relative idle period larger than 10 cycles for various functional units in an Xscale-like processor.

the number of consecutive idle cycles seen exceeds a pre-decided threshold. This threshold is referred to as the idle detect threshold (IDT). Several new heuristics for implementing and improving the time-based power gating technique were introduced in [31] using different idle time detection thresholds and by using control dependency and decode information to wakeup gated functional units early to minimize performance degradation.

The technique presented in [31] is used in this paper for detecting idle time of functional units, to be able to shut down their associated clock tree network. Once an idle functional unit is detected, its clock tree network routing the clock signal can be shut down. Note that we assume that the PSSTI technique is already applied to individual functional unit clock tree network and the sleep transistors are inserted. The same sleep signal that shuts down an individual functional unit also controls the sleep transistor of the clock tree network and shuts it down too. In other words, the functional unit and its associated clock tree network are shut down at the same time using the time-based algorithm proposed in [31].

Fig. 16 shows how often each of the five functional units used in the Intel's XScale are idle and is an indication of potential leakage power savings. On average, three of the units, i.e., integer multiplier/divider, floating point ALU and floating point multi-

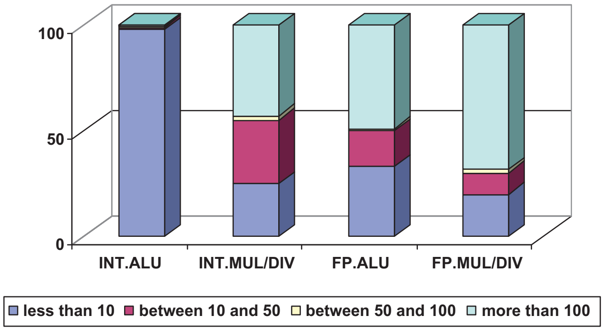

Fig. 17. Idle time distribution in different functional units.

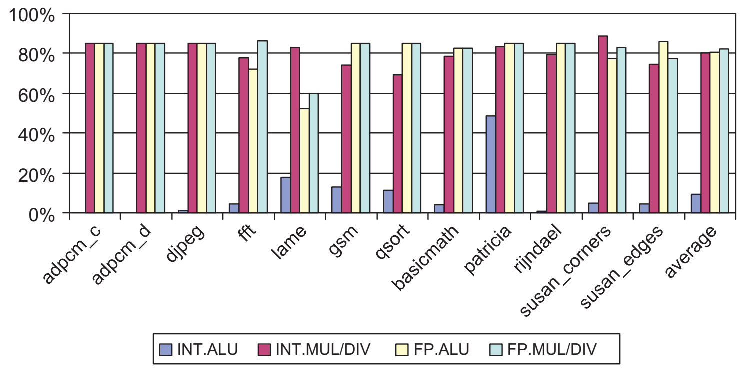

Fig. 18. Leakage power reduction in functional units' clock tree network using MIDT technique.

plier/divider are idle more than 95% of cycles. Average idle period is shortest for integer ALU (16%). Note that the numbers reported in Fig. 16 assume only idle periods of more than 10 consecutive cycles. A smaller idle period does not justify the power overhead associated with sleep transistor power gating (both the sleep transistors which is used to shut down the functional unit and the sleep transistors which is used to shut down the clock tree network). Also note that 100% idle times for some units in the figure for some benchmarks are due to the fact that the benchmarks do not use that unit at all. Accordingly, such units along with its clock tree network can be shut down for the entire runtime.

To provide a better insight, Fig. 17 shows the idle time distribution for each functional unit. The idle time distribution is quite different from one functional unit to another. As such, using a single IDT for all execution units is inefficient and can result in frequent power switching of the clock tree network and incur large power overhead.

To address this issue, a different IDT is used for each functional unit. We refer to this method as multiple IDT or MIDT. To pick the right IDT for every functional unit we took into account many factors, including how often the functional unit becomes idle and how long it stays idle. After testing many alternatives, the IDT values of 20, 80, 100 and 140 were picked for integer ALU, integer multiplier/divide, floating point ALU and floating point multiplier/divider, respectively. Note that multiple IDT could be implemented easily by using programmable registers.

The leakage power reduction for the clock tree network of various functional units is shown in Fig. 18. The clock tree network of floating point multiply/divide unit achieved the largest leakage power reduction, 82%. This is consistent with the result reported in Fig. 14 showing that the floating point multiply/divide unit spends more than 95% of the program execution time idle. The clock tree network of integer ALU exhibits the smallest leakage power savings, only 9%. This is not unexpected given that Fig. 16 shows this unit to be idle for only 16% of program execution time.

9. Conclusions

This paper described the problem of clock tree leakage power and described sleep transistor insertion in the clock tree to reduce the leakage power. We characterized the effect of sleep transistor sharing and sizing on clock tree wakeup time, leakage power, and propagation delay.

An algorithm is proposed utilizing 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, significant reductions in total power and leakage power were observed, by 16% and 32% on average respectively.

In addition, the PSSTI technique [31] was used to lower the leakage power of the clock trees within the data-path components of the general-purpose processors. A mechanism to control the activation/deactivation of the sleep transistors was proposed in the clock tree data-path components. The results indicate that the leakage power in the clock trees of the integer ALU, Integer multiply/divide, floating point ALU and floating point multiply/divide can be reduced by 9%, 80% 81% and 82% respectively.

Finally, it should be noted that a broader impact of PSSTI along with the zig-zag share circuit technique is not limited to the clock tree network. The techniques can also be applied to other on-chip logic structure where inverters are the dominant utilized standard cells. Examples are SRAM peripheral circuitry including input and output drivers [35,36], wordline drivers [35,36] and pre-decoder circuits. In addition, the proposed technique can also be deployed in on-chip buses and communication networks where large inverters are the major source of leakage [37].

Uncited references Q2

[32,33].

References

[1] M. Donno, E. Macci, L. Mazzoni, Power-aware clock tree planning, in: Proceedings on ACM/IEEE International Symposium Physical Design, 2004, pp. 138–147.

[2] N. Magen, et al., Interconnect-power dissipation in a microprocessor, in: Proceedings of Workshop on System Level Interconnect Prediction, 2004, pp. 7–13.

[3] N.S Kim, et al., Leakage current: moore's law meets static power, IEEE Computer Society (2003) 68–75.

[4] W.M. Elgharbawy, M.A. Bayoumi, Leakage sources and possible solutions in nanometer CMOS technologies, IEEE Circuits and Systems Magazine (2005) 6–17.

[5] V. Adler, E.G. Friedman, Repeater insertion to reduce delay and power in RC tree structures, IEEE Asilomar Conference on Signals Systems and Computers (1997) 749–752, November.

[6] J. Cong, C.-K. Koh, K.-S. Leung, Simultaneous buffer and wire sizing for performance and power optimization, ACM/IEEE International Symposium on Low-Power Electronics and Design (1996) 271–276, August.

[7] M. Vittal, M. Marek-Sadowska, Low-power buffered clock tree design, IEEE Transactions on CAD/ICAS 16 (9) (1997) 965–975, September.

[8] M.A. El-Moursy, E.G. Friedman, Exponentially tapered H-tree clock distribution networks, IEEE Transactions on VLSI Systems (2005).

[9] M. Igarashi, et al., A Low-Power Design Method using Multiple Supply Voltages, ISLPED-97. Q3

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

[11] W. Shen, et al., Leakage Power Optimization for Clock Network using Dual-V_{th} Technology, ISCAS, 2008.

[12] Y. Cheon, et al., Power-aware placement, in: DAC, 2005.

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005

[13] O. Jaewon, P. Massoud, Gated clock routing for low-power microprocessor design, IEEE Transactions on CAD/ICAS 20 (6) (2001) 715–722, June.
[14] M. Donno, et al., Clock-tree power optimization based on RTL clockgating, in: Proc. DAC, 2003, pp. 622–627.
[15] R. Puri, et al., Pushing ASIC performance in a power envelope, in: Proc. DAC, 2003, pp. 788–793.
[16] M. Ekpanyapong, S.K. Lim, Integrated retiming and simultaneous V_{dd}/Vth scaling for total power minimization, in: Proc. ISPD, 2006, pp. 142–148.
[17] D. Chinnery, K. Keutzer, Closing the Power Gap Between ASIC & Custom, Springer, USA.
[18] H. Homayoun, 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.S Kao, et al., MTCMOS hierarchical sizing based on mutual exclusive discharge patterns, in: DAC, June, 1998.
[20] J.M. Rabaey, et al., Digital Integrated Circuits: A Design Perspective, second edition, Prentice Hall, 2003.
[21] K. Agarwal, et al., Power gating with multiple sleep modes, in: ISQED, 2006.
[22] J.P. Fishburn, Clock Skew Optimization, IEEE Transactions on Computers 39 (7) (1990) 945–951, July.
[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.
[27] G. Contreras, et al., XTREM: A Power Simulator for the Intel XScaler Core, LCTES, 2004.
[28] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, R. Brown, MiBench: a free, commercially representative embedded benchmark suite, in: IEEE 4th Annual Workshop on Workload Characterization, De, 2001.
[29] D. Burger, T.M. Austin, S. Bennett, Evaluating Future Microprocessors: The SimpleScalar Tool Set. CS-TR-96-1308, University of Wisconsin-Madison, 1996, July.
[30] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zuyuban, H. Jacobson, P. Bose, Microarchitectural techniques for power gating of execution units, in: Proc. ISLPED, 2004.
[31] H. Homayoun, S. Golshan, E. Bozorgzadeh, A.V. Veidenbaum, Kurdahi S F.J., Post-synthesis sleep transistor insertion for leakage power optimization in clock tree networks, IEEE-ISQED (2010) 499–507.
[32] H. Homayoun, A. Baniasadi, Reducing execution unit leakage power in embedded processors, IEEE SAMOS (2006).
[33] H. Homayoun, A. Baniasadi, Analysis of functional unit power gating in embedded processors, IEEE/IFIP International Conference on Very Large Scale Integration System on Chip (VLSI-SoC) (2005).
[34] K. Shi, D. Howard, Challenges in sleep transistor design and implementation in low-power designs, in: Proc. of the DAC, 2006, pp. 113–116.
[35] H. Homayoun, A. Sasan, A. Gupta, A. Veidenbaum, F. Kurdahi, N. Dutt, Multiple sleep modes leakage control in peripheral circuits of a all major SRAM-based processor units, in: ACM International Conference on Computing Frontiers, May, 2010.
Q4 [36] H. Homayoun, A. Sasan, A. Veidenbaum, MZZ-HVS: multi modes zig–zag horizontal and vertical sleep transistor sharing to reduce leakage power in on-chip SRAM peripheral circuits, IEEE Transactions on Very Large Scale Integration (VLSI) Systems (2011).
[37] H.S. Deogun, R. Senger, D. Sylvester, R. Brown, K. Nowka, A dual-VDD boosted pulsed bus technique for low power and low leakage operation, in: International Symposium on Low Power Design, ISLPED, 2006.

Houman Homayoun received the PhD degree from the Department of Computer Science at the University of California Irvine in 2010. He was named a 2010 National Science Foundation Computing Innovation Fellow by the Computing Research Association (CRA) and the Computing Community Consortium (CCC). He was a recipient of the 4-years UC-Irvine computer science department chair fellowship. His research is on power-temperature and reliability-aware memory and processor design optimizations and spans the areas of computer architecture, circuit design and VLSI-CAD, where he has published more than 30 technical papers on the subject. His research is among the first in the field to address the importance of cross-layer power and temperature optimization in memory peripheral circuits. He received his BS degree in electrical engineering in 2003 from Sharif University of technology, Tehran, Iran. He received his MS degree in computer engineering in 2005 from University of Victoria, Canada.

Shahin Golshan received the B.S. degree in computer engineering from Sharif University of Technology, Tehran, Iran in 2005, and the M.S. degree in systems from the University of California, Irvine (UCI) in 2009, where he is currently working toward the PhD. Degree. His research interests include VLSI/FPGA computer aided design, design automation for embedded systems (high level synthesis and physical design algorithms) with emphasis on low power and reliability, and reconfigurable computing. He is a member of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE).

Elaheh Bozorgzadeh (S'01ÐM'03) received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1998, the M.S. degree in computer engineering from Northwestern University, Evanston, IL, in 2000, and the Ph.D. degree in computer science from the University of California, Los Angeles, in 2003. She is currently an Associate Professor in the Department of Computer Science at the University of California, Irvine. Her research interests include design automation for embedded systems, and reconfigurable computing. Dr. Bozorgzadeh is recipient of NSF CAREER award and the Best Paper award in IEEE FPL 2006.

Alexander V. Veidenbaum received the PhD degree in Computer Science from the University of Illinois at Urbana-Champaign in 1985. He is currently a Professor of Computer Science at the University of California, Irvine. His research interests include computer architecture for parallel, high-performance, embedded and low-power systems and compilers. He is a member of the ACM, the IEEE and the IEEE Computer Society.

Fadi Kurdahi received his PhD from the University of Southern California in 1987. Since then, he has been a faculty at the Department of Electrical & Computer Engineering at UCI, where he conducts research in the areas of Computer Aided Design of VLSI circuits, high-level synthesis, and design methodology of large scale systems, and serves as the Associate Director for the Center for Embedded Computer Systems (CECS). He was Associate Editor for IEEE Transactions on Circuits and Systems II 1993–1995, Area Editor in IEEE Design and Test for reconfigurable computing, and served as program chair, general chair or on program committees of several workshops, symposia and conferences in the area of CAD, VLSI, and system design. He received the best paper award for the IEEE Transactions on VLSI in 2002, the best paper award in 2006 at ISQED, and four other distinguished paper awards at DAC, EuroDAC, ASP-DAC and ISQED. He also received the Distinguished Alumnus award from this Alma Mater, the American University of Beirut in 2008. He is a Fellow of the IEEE and the AAAS.

Please cite this article in press as: H. Homayoun, et al., On leakage power optimization in clock tree networks for ASICs and general-purpose processors, Sustain. Comput.: Inform. Syst. (2010), doi:10.1016/j.suscom.2010.10.005