Power Efficient High-Level Synthesis by Centralized and Fine-Grained Clock Gating

Mohsen Riahi Alam, Mostafa Ersali Salehi Nasab, and Sied Mehdi Fakhraie

***Abstract*—Nowadays, power is a primary concern in digital circuits and clock distribution networks are particularly a significant power consumer. Therefore, clock gating is an effective technique in saving dynamic power by reducing the switching activities. In this paper, we propose a centralized and fine-grained microarchitecture-level clock gating for low power hardware accelerators which are automatically designed by high-level synthesis (HLS) tool. The basic principium of our idea is not to use any extra computation for generating clock enabled signals and exploit exiting signals of finite state machine for controlling the datapath clock network. After determining the current state in finite state machine, clock sub-tree of current state is enabled and the other sub-trees are disabled with a slight increase in circuit area. Our approach is implemented within an HLS design flow for automatic low power hardware accelerator generation in application specific integrated circuit design. Experimental results are obtained on a set of representative benchmark programs. Depending on the circuit size and number of registers, it is shown that 47%–86% reduction in power dissipation is observed.**

***Index Terms*—Clock gating, finite state machine, high-level synthesis (HLS), low power design.**

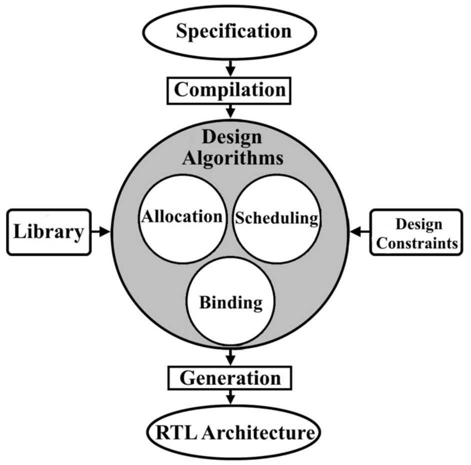

Fig. 1. HLS design flow.

I. Introduction

THE BROAD usage of portable computing devices has shifted the major concern of very large-scale integration (VLSI) design from high speed to low power. As we know, a specialized hardware accelerator will improve performance and energy efficiency compared to general purpose implementations. Heterogeneous and specialized architecture design is an appropriate technique to optimize power consumption in VLSI design.

Manual hardware design is difficult because it is very time demanding and requires complex register transfer level (RTL) coding, which is error prone and difficult to debug. On the other hand, the broad application domains and emerging application specific designs make us to incorporate fast and automatic high-level synthesis (HLS) tools. These tools help us to generate an RTL design realizing the specified behavior and also satisfying the design constraints in an easier and automatic manner.

HLS starts with the algorithmic description of an application (standard C/C++ program or SystemC description), RTL component library (including component characteristics such as area, delay, power, etc.), and design constraints (cost, performance, power consumption, resources, pin-count, testability criteria, etc.) and automatically generates RTL design of datapath and control logic. Two primary issues that all hardware designers must overcome are design cost and time-to-market. HLS reduces the cost and time-to-market significantly, while manual hardware design is more expensive and time consuming. HLS also expedite the design space exploration for efficient hardware/software co-design. Traditionally, HLS design objectives were developing small and fast designs. However, today, power consumption has gained more concerns as a more important design constraint.

New energy efficient architectures in age of dark silicon [1]–[3] are based on HLS tools which are used to automatically synthesize specialized processors for a wide range of applications. Therefore, power efficient designs exploiting HLS techniques are inevitable. Three main stages of HLS process, compilation, design algorithms, and RTL generation are shown in Fig. 1. Low power techniques in HLS can be applied to all of these stages. Compiler optimizations as the first stage of HLS can significantly improve hardware quality and power consumption of generated RTL designs [4], [5].

Manuscript received October 25, 2014; revised March 23, 2015; accepted May 20, 2015. Date of publication June 15, 2015; date of current version November 18, 2015. This paper was recommended by Associate Editor A. Srivastava.

M. Riahi Alam is with the School of Electrical and Computer Engineering, Faculty of Engineering, University of Tehran, Tehran 14395-515, Iran (e-mail: riahialam@ut.ac.ir).

M. E. Salehi Nasab is with the School of Electrical and Computer Engineering, Faculty of Engineering, University of Tehran, Tehran 14395-515, Iran, and also with the School of Computer Science, Institute for Research in Fundamental Science, Tehran 19395-5746, Iran (e-mail: mersali@ut.ac.ir).

S. M. Fakhraie, deceased, was with the School of Electrical and Computer Engineering, Faculty of Engineering, University of Tehran, Tehran 14395-515, Iran (e-mail: fakhraie@ut.ac.ir).

Digital Object Identifier 10.1109/TCAD.2015.2445734

Recently, some HLS design algorithms are proposed to alleviate the power consumption. Yeh and Wang [6] presented a clock-skew scheduling scheme in HLS flow and evaluated it with multithreshold CMOS technology for power and speed optimizations. Sengupta and Sedaghat [7] had introduced a novel multistructure design space exploration system based on genetic algorithm that solves the problem of integrated scheduling, allocation, and binding in HLS. Lhairech-Lebreton *et al.* [8] presented a flow for generating hardware accelerators for digital signal processing applications by bit-width aware HLS flows.

There is a tight correlation between the RTL design and power consumption in HLS. By applying RTL power optimization techniques to RTL generation process of HLS, low power HLS design could be achieved. These techniques can be roughly classified to: interconnect optimization, switching activity reduction, power gating, and clock gating. In this paper, we propose a new clock gating technique for HLS tools targeting design of low power hardware accelerators in application specific integrated circuit (ASIC) design flows.

Power consumption consists of dynamic and leakage power. In the contemporary CMOS technologies, due to scaled devices, active leakage power is a big challenge. Power gating is one of the effective techniques to reduce active leakage power. Several different methods of power gating have been developed so far. In regard to the case, we investigated in this paper, we will review two FSM-based power gating techniques in Section II.

Dynamic power is dissipated by a circuit during its operational mode. In synchronous circuits, the clock signal switches at each clock cycle and drives a large capacitance. As a result, the clock signal is a major source of dynamic power dissipation. In synchronous circuits, the clock of some sequential elements can be disabled without affecting the logic functionality. This technique is known as clock gating. Clock gating is the most effective and widely used technique for power saving. According to the context and design purpose, clock gating has been performed at different levels of abstraction and granularities.

1) Architecture-Level Clock Gating: In system-on-chip design, the clock signal of the whole processor core or a specific module is disabled until it receives a request from operating system or hardware power management unit. This is typically used while processor or the target module is in sleep or wait for interrupt state.

2) Microarchitecture-Level Clock Gating: Large architecture systems (e.g., superscalar processor) are composed of various logic blocks. These blocks may be mutually exclusive and are not exploited at the same time. Designer can locate the idle blocks and create corresponding clock gating conditions at RTL design phase to enable/disable clock of these blocks. Proposed techniques in [9]–[11] are classified in this level.

3) Circuit-Level Clock Gating: Most of the previous clock gating techniques are classified in this level; we will review some of them in Section II. For clock gating schemes at the circuit-level, some gating conditions have been extracted from RTL or gate-level based on data dependencies. These gating conditions have been exploited to enable/disable some branches of clock tree without affecting the architectural behavior.

Considering the limitations and overheads of previous circuit-level methods, we look at the subject from a different point of view and introduce a novel microarchitecture-level clock gating method for finite state machine with datapath (FSMD) circuits. In summary, the key contributions of our method which reduce the dynamic power of the datapath clock tree in FSMD architecture are as follows.

1) Our method exploits the FSM signals to manage clock gating conditions at RTL to avoid undesirable timing, area, and power overhead of clock control (gating function) logic circuit.
2) Our proposed centralized method gates the clock signals as close as to the root of clock tree to avoid energy wasting in the clock tree wires and repeaters.
3) Our state oriented fine-grained clock gating method routes clock signals only to the FSMDs current state registers which are a little portion of circuit's registers.
4) Our clock gating methodology has been integrated into an HLS design flow and performs RTL clock gating automatically.

This paper is organized as follows. In Section II, we briefly introduce some of the previous work on clock gating. The basics behind our idea, the motivation, and our main model will be discussed in Sections III and IV. Section V presents the architectural implementation and considerations of our proposed idea. The experimental platform and the way that we modify it for our clock gating purpose will be asserted in Section VI. In Section VII, our experimental results will be presented. Finally, this paper is concluded in Section VIII.

II. Related Work

Almost all the previous clock gating researches focus on clock gating conditions. These approaches first detect clock gating conditions and then create the suitable clock control logic for implementing clock gating [12]–[14]. Clock control logic usually adds undesirable timing, area, and power overhead to the main circuit. Previous researches have attempted to decrease these overheads. Gating conditions are classified in two types observable don't care (ODC) and stability conditions (STCs). When next state values of registers cannot be observed in the next clock cycles, the clock of register can be gated in those clock cycles by using ODC conditions. STCs are used for clock gating when register values do not change for two or more continuous clock cycles [12].

Traditional XOR-based clock gating circuits compare flip-flops (FFs) output with their present input and disables the next clock cycle when values are equal. This clock gating method is categorized in STC and has area and power overhead. An optimized XOR-based clock gating scheme is proposed in [15]. This scheme chooses only a subset of FFs to be gated selectively, and the complexity of choosing candidate FFs is reduced from exponential to linear. The XOR gates producing enable signals incur some delay and area overhead. Furthermore, since the clock signal is gated near the destination FFs, it causes some power dissipation in routing the clock signal.

Wimer and Albahari [16] proposed a look-ahead clock gating scheme which is categorized in XOR-based clock gating techniques. It generates the clock enabling signal of the target FF one cycle ahead of time. This technique enables the target FF according to the data of the FFs that the target FF is dependent to. Likewise in our clock gating technique, clock gating cells in FSM provides the enabling signals during previous clock cycle. However, our clock gating implementation has a negligible area overhead.

Babighian *et al.* [13] proposed an RTL clock gating method based on ODC conditions which is suitable for large-scale designs. Their proposed ODC conditions are extracted from existing steering logics such as multiplexers, tri-state buffers, and enable states. Therefore, for ODCs detection no redundant computation is needed. Fraer *et al.* [12] presented a method that unifies ODC and STC clock gating approaches into one framework to create gating conditions. In their method, finding STCs is robust and scalable. Due to the complexity of exact ODC condition computing, they computed a weaker ODC condition that is still safe for clock gating. This approach requires a lot of enable signals and like previous methods it also needs extra logic for generating these activation signals.

The most recent gate-level clock gating technique which simplifies clock control logic and reduces area overhead is proposed in [14]. The idea is to generate gating functions (clock control logic) by using the existing logic as far as possible. For this purpose, they use matching factored forms of gating functions with those of existing logic nodes. Some microarchitecture-level clock gating methods use FSM information for disabling clock signals. Benini and De Micheli [17] described a technique based on STC for the automatic synthesis of gated clocks for FSMs. These STCs are based on identifying self-loops in FSMs. When FSM arrives at a self-loop state, the clock is turned off. In this idle situation, the inputs of the combinational logic block do not change. If the input values cause to have an FSM state transition, again the clock signal is enabled and the circuit resumes operation. To identify self-loop conditions, some additional computations are needed. These computations are executed by extra logic circuits. Sometimes, there are several idle conditions in FSM design, so detecting all of them is time and power consuming. To save time, they have restricted the number of idle conditions and finally select a subset of these idle conditions that lead to the minimum power dissipation. Two major limitations of this technique are unsupported large state transition graph representations and also required combinational logic blocks for activation function.

Our proposed scheme as well as the one presented in [17] control clock gating with FSM signals, however, they have different optimization goals. Benini and De Micheli [17] focused on FSM power optimization and verify their idea using FSM benchmarks. However, our benchmarks depict that contribution of FSM power consumption (with one-hot assignment) is in average less than %5 of the whole FSMD design. Therefore, we focus on the datapath power optimization by enabling suitable clock sub-tree in each cycle.

Exploiting some architectural properties of FSMs such as their capability to be decomposed into independent parts can be used in clock and power gating methods [18]–[20]. For instance, the FSM decomposition feature for designing low power systems has been proposed in [20]. The main idea is to decompose the FSM into two sub-FSMs that jointly produce the functional behavior of the original FSM but with lower power consumption. When a state transition occurs at one of the sub-FSMs, the clock of the other sub-FSM will be disabled without affecting the FSMs main functionality. Similar to our proposed scheme, this paper is applied in architectural-level. However, the level of our clock gating scheme's granularity is in the order of a single state instead of a subset of states. Consequently, our proposed clock gating would be done in a more fine-grained manner.

Usami and Yoshioka [18] proposed an RTL run time power gating method for FSM circuits. This leakage reduction technique exploits the dynamically detected nonoccurring state transition condition signals for controlling sleep transistors. In [19], FSMD power gating method is proposed. In this method, the main FSMD is split into sub-FSMDs. At any given moment only one sub-FSMD circuit is turned on and the others are power gated. At run time, FSM behavior signals are predicted to determine which sub-FSMD circuit should be turned on next. Since the wake-up time of the power gated circuits have a delay overhead and hence reduce the performance, the two aforementioned methods divide the FSMD into a few state subsets and apply the power gating at the granularity of state subsets. Therefore, to avoid performance degradations, they cannot do power gating in a highly fine-grained manner. Since clock gating has not such an overhead, we proposed fine-grained clock gating to enable/disable clock sub-trees in each state.

III. Motivation

As mentioned in Section II, clock gating is a common technique in low power design and significantly decreases dynamic power. To illustrate this circumstance and further motivate this paper, we employed RTL simulations with the register toggle trace and simple FF energy model. We used an HLS framework with six CHStone [21] benchmarks (as described in Section VI) and also we use the FF energy model introduced in [22].

In this model, without considering the glitches, when clock pulse arrives at edge-triggered FFs, the values may toggle or remain unchanged. If circuit works for n clock cycles, the average consumed dynamic energy for each FF is obtained by

$$E = E_T N_T + E_R N_R \tag{1}$$

where E_T and E_R are consumed dynamic energies by FF when its value is toggled or remain unchanged at clock arrival time, respectively. The respective number of events during n cycle which the FF value is toggled or remain unchanged are N_T and N_R where $n = N_R + N_T$. In ordinary FSMD design, when the datapath circuit is clocked, all FFs receive the clock signal and consume energy regardless of whether or not there is data toggle. Although idle states' FFs do not change for sure, underlying current (active) state FFs values may change. Thus, in each clock arrival time, datapath FFs are classified in

TABLE I
COMPARING NUMBER OF ACTIVE STATE FFs AND IDLE STATES FFs IN CHSTONE BENCHMARKS

Benchmarks	Selection Function to make hardware accelerator	Number of clock cycle: N	Number of FFs of circuit: $N_{CS_i}+N_{IS_i}$	Sum of current state's FFs in n cycles $\sum_{i=1}^{n} N_{CS_i}$	Sum of idle states' FFs in n cycles $\sum_{i=1}^{n} N_{IS_i}$	Sum of all states' FFs in n cycle $\sum_{i=1}^{n}(N_{CS_i}+N_{IS_i})$	Current state's FF to all states' FFs (%) $\frac{\sum_{i=1}^{n} N_{CS_i}}{\sum_{i=1}^{n}(N_{CS_i}+N_{IS_i})}$	M_2 (%)
ADPCM	encode	39607	4716	1246942	185539670	186786612	0.6676	99.33
motion	Initialize_Bufffer	8500	964	339722	7854278	8194000	4.1459	95.85
blowfish	blowfish_main	337799	22665	13977125	7642237210	7656214335	0.1826	99.82
dfmul	float64_mul	287	3406	21889	955633	977522	2.2392	97.76
GSM	Gsm_LPC_Analys	6764	7803	301576	52477916	52779492	0.5714	99.43
dfadd	float64_add	958	5724	78864	5404728	5483592	1.4381	98.56

three groups: 1) current state toggled FFs; 2) current state unchanged FFs; and 3) idle states FFs. To determine the effect of clock gating on the amount of consumed energy in benchmark circuits, we use an energy model for FFs in our experiments by exploiting exact and nonprobability data-to-clock toggling ratio. From (1) and FFs portioning, the dynamic energy consumption for all FFs in datapath during one clock cycle can be modeled as

$$E_{\text{Clock\_Cycle}} = E_T N_{\text{CST}} + E_R N_{\text{CSR}} + E_R N_{\text{IS}}. \tag{2}$$

In each cycle, N_{CST} and N_{CSR} are, respectively, the number of toggled and unchanged FFs in current state. While N_{IS} is the number of FFs in idle states those are not certainly toggled. Energy consumption in all clock cycles is expressed by the following equation:

$$E_{\text{Total}} = \sum_{i=1}^{n} \left(E_T N_{\text{CST}_i} + E_R N_{\text{CSR}_i} + E_R N_{\text{IS}_i}\right) \tag{3}$$

where n is the number of clock cycles. E_T and E_R are architecture and technology dependent and are the same for all FFs. So the resulting equation is

$$E_{\text{Total}} = E_T \sum_{i=1}^{n} N_{\text{CST}_i} + E_R \sum_{i=1}^{n} N_{\text{CSR}_i} + E_R \sum_{i=1}^{n} N_{\text{IS}_i}. \tag{4}$$

The first term of (4) is an inevitable energy consumer as it is used to latch the logic outputs, while the second and third terms can be avoided to reduce energy wasting. Therefore, clock gating can prevent energy wasting in these two parts. The effectiveness of clock gating methods is measured by data-to-clock toggling ratio [18]. According to (4), the maximum efficiency is achieved by elimination of both second and third terms through ideal clock gating implementations. These two terms have different granularities and energy wasting. Thus, different clock gating strategies in different abstraction levels can be used together to eliminate each term.

We define two metrics to determine contribution of each term in energy wasting. The ratio of current state's not-toggled FFs to all FFs of circuit is called M_1 and the ratio of idle states' FFs to all FFs is introduced as M_2

$$M_1 = \frac{\sum_{i=1}^{n} N_{\text{CSR}_i}}{\sum_{i=1}^{n}\left(N_{CS_i} + N_{\text{IS}_i}\right)} = \frac{(1-P_T)\sum_{i=1}^{n} N_{CS_i}}{\sum_{i=1}^{n}\left(N_{CS_i} + N_{\text{IS}_i}\right)} \tag{5}$$

TABLE II
CURRENT STATE'S FFs TOGGLING RATIO

Hardware Accelerator's Circuit	$\sum_{i=1}^{n} N_{CST_i}$	$\sum_{i=1}^{n} N_{CS_i}$	$\frac{\sum_{i=1}^{n} N_{CST_i}}{\sum_{i=1}^{n} N_{CS_i}}$	M_1 (%)
encode	70416	1246942	0.0565	0.6298
Initialize_Buffer	12950	339722	0.0381	3.9879
blowfish_main	1474589	13977125	0.1055	0.1633
Gsm_LPC_Analysis	84386	301576	0.2798	0.4115
float64_mul	3703	21889	0.1692	1.8603
float64_add	9638	78864	0.1222	1.2623

$$M_2 = \frac{\sum_{i=1}^{n} N_{\text{IS}_i}}{\sum_{i=1}^{n}\left(N_{CS_i} + N_{\text{IS}_i}\right)} = 1 - \frac{\sum_{i=1}^{n} N_{CS_i}}{\sum_{i=1}^{n}\left(N_{CS_i} + N_{\text{IS}_i}\right)} \tag{6}$$

where P_T is the probability of toggling in current state's FFs. Analyzing information of register toggle traces through RTL simulations which are presented in Tables I and II helps us to calculate these sigma expressions and exact value of P_T.

Granularities in second and third terms of (4) are determined based on single FFs and state FFs, respectively. Circuit-level clock gating is suitable to avoid energy wasting in the two aforementioned granularities. However, circuit-level clock gating techniques have more hardware and timing overheads. On the other hand, in FSMD circuits, FSM module determines the active state in each cycle. Therefore, by using a higher level decision making method, clock gating techniques can use FSM information to enable/disable clock signals. Consequently, the third terms of (4) would be eliminated. Choosing between two clock gating methods depends on the amount of energy wasting in the second and third terms of (4) and hardware overheads of each method.

The results of Tables I and II indicate that M_1 (the effect of the second term) is very smaller than M_2 (the effect of the third term) in all benchmarks. Therefore, the third term has lower amount of energy wasting compared with the second term. Moreover, using FSMs information for clock gating does not have any extra overheads. Therefore, applying microarchitecture-level clock gating method based on FSM to eliminate the third term of (4) is much more efficient than the circuit-level clock gating technique which also covers nontoggling FFs (the second term) in the current state. This is the reason that we choose the second method to apply clock gating targeting idle state FFs [the third term of (4)].

Therefore, we partition FFs into current state and idle states to determine how much wasted energy is reduced by applying clock gating method. In this way, (4) can be rewritten as

$$E_{\text{Total}} = [E_T P_T + E_R(1 - P_T)] \sum_{i=1}^{n} N_{CS_i} + E_R \sum_{i=1}^{n} N_{\text{IS}_i}. \quad (7)$$

Table I illustrates the superiority of the number of idle states' FFs to the number of current state's FFs in the selected benchmarks [sigma expressions in (7)]. Therefore, gating the clock of idle states' FFs decreases the wasted energy significantly.

IV. Basic Idea

In FSMD design, at any moment only one state is active and the others are idle. We assume idle states are categorized into temporal and spatial idle states. Depending on the FSMDs inputs, the useless states during FSMD operation are spatial idle states. Other states which are active in the specified time span and idle in other times are called temporal idle states. According to Fig. 2(b), our idea is based on splitting the clock tree to multiple sub-trees which are routing the clock for each state in datapath. In each clock cycle, sub-tree of active (current) states is enable and all idle states clock sub-trees are disable. There is no difference between the temporal and spatial idle states in our method. Therefore, regardless of the FSMDs inputs, the clock gated FSMD can be used in different conditions with runtime decision.

The power model proposed in the previous section shows the amount of power saving when only the underlying clock tree of current state is activated (instead of whole data path clock tree). In the other word, the number of charged and discharged input capacitances of FFs in leaf of clock tree is decreased and consequently dynamic power consumption is reduced. The other important power consumer parts in clock trees are wires and repeaters which route the clock from source to sequential elements. In this section, we present a geometric model to show how our approach reduces the length of active parts of the clock tree and hence reduce dynamic power consumption. Weste and Harris [23] introduced a model for energy per unit length to send a bit on a wire. This model considers both wire and repeater capacitances for repeaters that are optimally sized for minimum delay

$$\frac{E}{L} = 1.87 C_\omega V_{\text{DD}}^2. \quad (8)$$

Clock tree is made of some wires and repeaters, so switched capacitance in the clock tree consist of both capacitances of these wires and repeaters. Without considering the number of FFs that are fed by this clock signal, the power per unit length of clock tree depends on clock frequency (f), supply voltage (V_{DD}), and wire capacitances (C_ω), and is defined as

$$P_{\text{unit\_length}} = 1.87 C_\omega V_{\text{DD}}^2 f. \quad (9)$$

Total switching power of the clock tree is

$$P_{\text{clock\_tree\_switching}} = 1.87 C_\omega V_{\text{DD}}^2 f L \quad (10)$$

where L is the length of clock tree. As shown in Fig. 2(a), without clock gating, all sequential elements of FSMD are fed by one spread clock tree. As mentioned before, at any moment, only one state is active in FSMD design. However, the whole clock tree is toggling as depicted in Fig. 2(a). Preventing these wasteful switchings in the clock tree leads to more power saving. As shown in Fig. 2(b), to achieve this goal, we propose a microarchitecture-level clock gating approach by adding some clock gating cells to FSM for enabling separate clock subtrees in each state of the datapath. After adding clock gating cells, clock tree will be partitioned in two parts [as shown in Fig. 2(b)]. The first part is the clock tree of the underlying FSM logic including clock gating cells. Compared with the prior FSM, the length of this clock tree is a little longer. The reasons for the clock tree growth are the occupied area of clock gating cells (Table IV) and the new branches of clock tree which bring the clock signal to the clock gating cells. The second part is the set of sub-tree of underlying sequential elements of each state in datapath. At each cycle, current state's clock sub-tree is enabled. Hence, the average switching power of the clock tree during the ith active state (without considering underlying FSM clock tree) can be written as

$$P_{\text{clock\_switching\_during\_state}_i} = 1.87 C_\omega V_{\text{DD}}^2 \, f(L_{\text{extra\_branches}} + L_{\text{state\_}i\text{\_subtree}}) \quad (11)$$

where $L_{\text{extra\_branches}}$ is the length of all new branches in FSM clock tree which bring clock signal to clock gating cells and $L_{\text{state\_}i\text{\_subtree}}$ is the length of active state clock sub-tree. There are different $L_{\text{state\_}i\text{\_subtree}}$ in the circuit for each state, so depending on the current state of FSMD, the datapath have different switching power for the clock tree in each clock cycle. Based on this fact, depending on the time slice (t_i) in which each state is active, the switching power dissipation of the clock tree can be written as

$$P_{\text{switching}} = 1.87 C_\omega V_{\text{DD}}^2 \, f(L_{\text{extra\_branches}} + L_{\text{average}}) \quad (12)$$

where

$$L_{\text{average}} = \sum_{i=1}^{n} \left(\frac{t_i}{t}\right) L_{\text{state\_}i\text{\_subtree}} \quad (13)$$

where t is the total time of FSMD active operation. L_{average} is weighted average of active state clock sub-tree length, while t_i is the weight of each one.

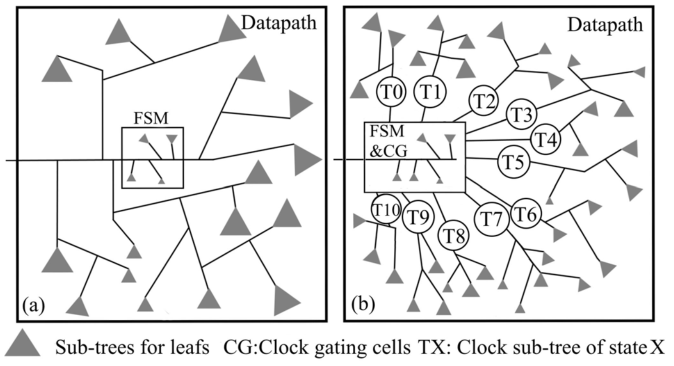
Datapath
FSM
(a)
Datapath
FSM &CG
T0
T1
T2
T3
T4
T5
T6
T7
T8
T9
T10
(b)
Sub-trees for leafs CG:Clock gating cells TX: Clock sub-tree of state X

Fig. 2. Clock tree scheme. (a) Conventional distributed clock tree for FSMD. (b) Our approach: separated clock sub-tree for each state in datapath which is generated with FSM and clock gate cells.

By comparing (10) and (12), after applying clock gating, switching power dissipation (without considering underlying FSM clock tree) is decreased and clock tree power is saved by this factor

$$\text{Saving\_Factor} = 1 - \frac{L_{\text{extra\_branches}} + L_{\text{average}}}{L_{\text{datapath}}}. \quad (14)$$

Area and portion of the sequential logic are good parameters for estimating the length of clock tree in digital circuits. According to Table IV, an FSM with clock gating cells is very smaller than the datapath's circuit. The number of FFs in all datapaths is shown in Table I. These FFs are more than 96% of FSMD FFs in average. Therefore, the length of the clock tree distributed on the FSM including clock gating cells is very shorter than the length of datapath clock tree (L_{datapath}). Furthermore, according to the number of active state FFs and their local distributions, the average length of active state clock sub-tree ($L_{\text{state\_}i\text{\_subtree}}$) is very shorter than the whole datapath clock tree (L). Hence, the numerator of (14) is very smaller than the denominator.

V. Architecture Overview

A. One-Hot Assignment for FSM

There are several assignment schemes for FSM implementation. The first discussion of one-hot state machines was given by Huffman [24]. In one-hot assignment, only a single bit is one at a time for each state. Our experimental results show that one-hot state assignments reduce the circuit size of the next-state logic which may lead to lower delay. However, it needs more registers which may increase dynamic power consumption. Experimental results which are shown in Table III show that FSMs with one-hot assignment are individually 2.5× to 4.5× more power-hungry than FSMs with regular binary assignments. However, considering the whole FSMD, the one-hot state assignment dynamic power overhead is at most 4.5%.

Although one-hot state machines dissipate an extra power, they are typically faster and their speed is independent of the number of states and only a single bit is one at any time [25]. Due to these attributes we used one-hot assignments for FSMs and its outputs are used as the activation signals for clock gating circuit.

B. Gated-Clock Design

Traditional clock gating techniques are based on identifying idle parts of modules and shutting down these parts of the clock tree by using AND gates at proper nodes of the clock tree. In these techniques, the clock control logic is added to the main circuit. Clock control logic consists of combinational elements and generates gating control signals indicating whether a clock pulse should be supplied to registers or not. Clock control logic and routing of the control signals to each gated buffer usually have overhead on timing, area, and power.

Different clock control logics are designed based on the clock gating methods. Typically for circuit-level methods, clock control logics are local and near target FFs. Whereas, clock control logics are central and applied close to the root of clock tree for higher level decision making in microarchitecture-level methods. According to the number of FFs in the datapath of each benchmark (Table I), applying circuit-level techniques have more overhead compared with the microarchitecture-level techniques.

TABLE III
FSM State Assignment Dynamic Power Comparison

Hardware Accelerator's Circuit	FSM			FSM & datapath		
	Regular Binary	One-hot	$\frac{One-hot}{RegularBinary}$	Regular Binary	One-hot	overhead (%)
encode	0.085	0.413	4.86	11.95	12.28	2.7
Initialize_Buffer	0.071	0.175	2.47	2.34	2.44	4.4
blowfish_main	0.345	1.465	4.25	37.96	39.08	2.9
Gsm_LPC_Analysis	0.188	0.828	4.41	20.78	21.42	3.1
float64_mul	0.033	0.281	8.42	5.60	5.87	4.4
float64_add	0.208	0.568	2.73	13.86	14.22	2.6

Dynamic power dissipation at 200 MHz with 90 nm CMOS technology and power numbers are reported in mW.

To avoid the clock control logic overhead in conventional circuit-level techniques and the FSMD structural features proposed in Sections III and IV, we propose a novel microarchitecture-level approach for dynamic power saving in a large datapath to reduce unnecessary switching activities on the clock tree. Our technique exploits separate clock sub-tree for each state by coupling the clock gating cells to FFs of each state in the one-hot FSM. Isolating clock sub-trees of different states prevents wasteful switching on the clock tree of datapath. Maximum delay constraint on a path from one FF to the next, assuming no clock skew, is introduced in [23]. By adding clock skew parameter to this formula, minimum clock period for the circuit can be written as

$$\min(T_c) = t_{\text{pcq}} + t_{\text{pd}} + t_{\text{su}} + t_{\text{skew}} \quad (15)$$

where t_{pcq} is FFs propagation delay, t_{pd} is the combinational circuit delay restricted by HLS, t_{su} is FFs setup time, and t_{skew} is the maximum clock skew. Gated clock trees usually have two basic timing issues. The first one is the increased skew for gated clock. In our design, all registers in datapath are only fed by gated clocks in contrast with conventional designs using both normal and gated clocks. Furthermore, gated clocks in our technique have only one extra AND gate in each clock path compared with hierarchical AND gates in conventional designs [26]. The inserted AND gates that are placed on the root of each clock sub-tree have equal delays, so the skew do not change and physical design tools easily handle these clock paths.

The second problem of gated clock trees is timing constraints violation for clock control logic. Our clock gating technique does not incorporate any auxiliary clock control logic rather we use one-hot FSM outputs as activation signals for clock gating circuits. Therefore, timing constraint violation is not a concern in our approach.

Our FSM along with clock gating cells is shown in Fig. 3. Each register in one-hot FSM design have a corresponding latch in the clock gating cell. The latch block is transparent

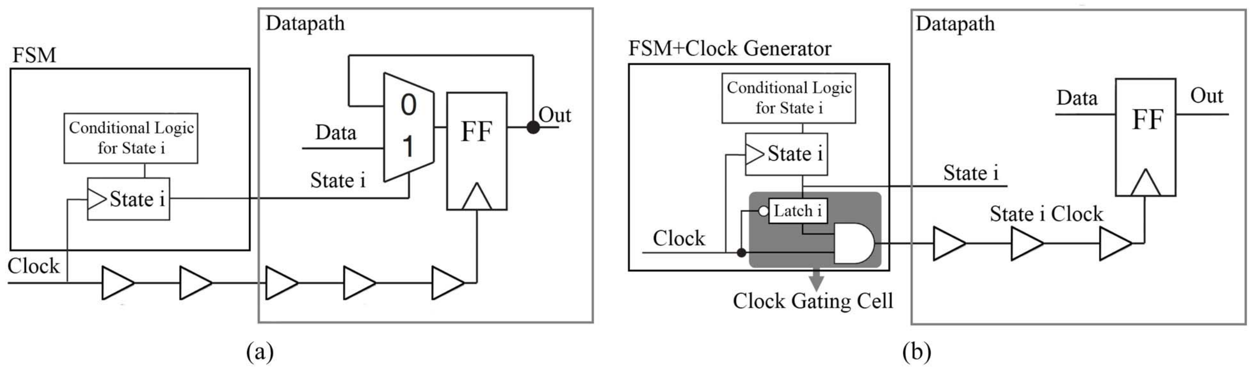

Fig. 3. Sample state circuit of FSMD with one-hot FSM assignment. (a) Conventional FSMD without clock-gating. (b) Our FSMD with clock gating.

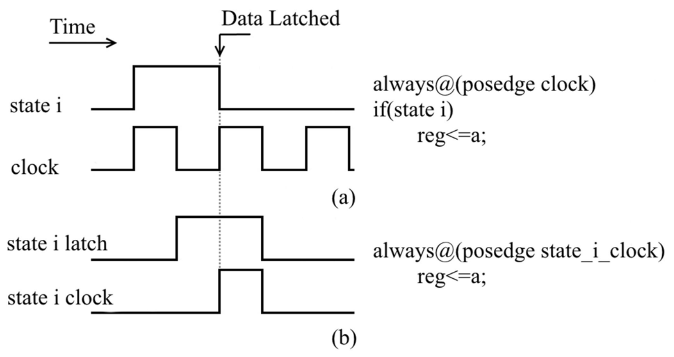

Fig. 4. Timing diagram of FF triggering in Verilog behavioral modeling. (a) Conventional approach: FF is triggered at clock rising edge and updates its value when current state is *i*. (b) Our approach: FF is triggered by special generated clock for state *i*.

when clock signal level is low and is fed with corresponding state registers. The latch is needed for correct behavior, because FSM has a transition on positive edge of clock and may have glitches that must not be propagated to AND gate when the clock is high. According to Fig. 4, in Verilog behavioral modeling of regular circuits, the FFs are triggered at rising edge of clock and current state signals (from FSM) act as the enable signals. After clock gating, in our behavioral modeling, the FFs are triggered by rising edge of the generated clock signals at each state.

VI. Experimental Framework

There are several HLS frameworks available on different domains. Our clock gating circuit should be incorporated into FSMD, so we need to perform some modifications in the RTL generation parts of state of the art HLS tools. There are some open source HLS tools suitable for this purpose. After investigating new HLS tools and their properties, we choose LegUp [27] which is a high-level synthesis tool developed at the University of Toronto. LegUp is developed to provide the performance and energy benefits of hardware and gets a standard C program and generates field-programmable gate array (FPGA)-based software/hardware system-on-chip.

In LegUp, MIPS processor and Altera Avalon interface are used as soft processor and processor/accelerator communication media, respectively. LegUp exploits open-source low level virtual machine (LLVM) [28] for compiler optimizations and high-level language parsing as well. LegUp developers create new backend compiler passes within LLVM for hardware synthesis. We apply modified LegUp hybrid flow to generate a system by choosing some program segments to be synthesized into RTL as hardware accelerators while the remaining program segments execute on an MIPS processor. In LegUp hybrid flow (hardware/software partitioning), MIPS processor begins execution of C programs until reaches the function which is implemented as hardware accelerator. Then, the processor sends arguments to the accelerator and asserts its start signal. The processor goes to stall state until a finish signal is asserted by the accelerator and then the processor resumes execution.

A. Benchmarks and Hardware Accelerators Selection

To evaluate our technique practically, we need to have some benchmark programs from various application domains. We choose six benchmarks from CHStone [21] consisting of a great set of computationally intensive programs in various categories that are compatible with LegUp. CHStone benchmarks' test vectors are self-contained and do not need any external library for compiling. The benchmarks are from arithmetic category (dfmul and dfadd), encryption domain (blowfish), and communication and multimedia fields such as Global System for Mobile Communications (GSM), Adaptive Differential Pulse Code Modulation (ADPCM), and motion. The LegUp framework includes a real-time hardware profiling capability. Using hardware profiling results at function-level and considering execution cycles, cache miss stalls, and energy consumption of each function, the best candidates for hardware accelerators have been chosen in the six CHStone benchmarks and are shown in Table II.

B. Modification

This section describes how the proposed clock gating technique is applied to the modified LegUp high-level synthesis framework (see Fig. 5). LegUp is an FPGA-based HLS tool and uses relevant library characterization for FPGA family to assign corresponding hardware operations to each instruction. In this paper, we want to achieve low power ASIC design flow for generating hardware accelerator through LegUp HLS tools.

TABLE IV
DYNAMIC POWER COMPARISON BEFORE AND AFTER CLOCK GATING FOR CHSTONE BENCHMARKS

	Before Clock Gating			After Clock Gating			
Hardware Accelerator's Circuit	FSM	Data-Path	FSMD	FSM+CGC	Data-Path	FSMD+CGC	Dynamic Power Reduction(%)
encode	0.4130	11.86	12.28	0.5057	2.09	2.59	78.89
Initialize_Buffer	0.1746	2.27	2.44	0.2215	1.06	1.28	47.57
blowfish_main	1.4646	37.61	39.08	1.8387	3.60	5.44	86.08
Gsm_LPC_Analysis	0.8278	20.60	21.42	1.0163	8.34	9.36	56.30
float64_mul	0.2814	5.58	5.87	0.3440	2.10	2.44	58.35
float64_add	0.5678	13.64	14.22	0.7163	6.71	7.43	47.78

Dynamic power dissipation at 200 MHz with 90nm CMOS technology and power numbers are reported in mW.
CGC: Clock Gating Cell.

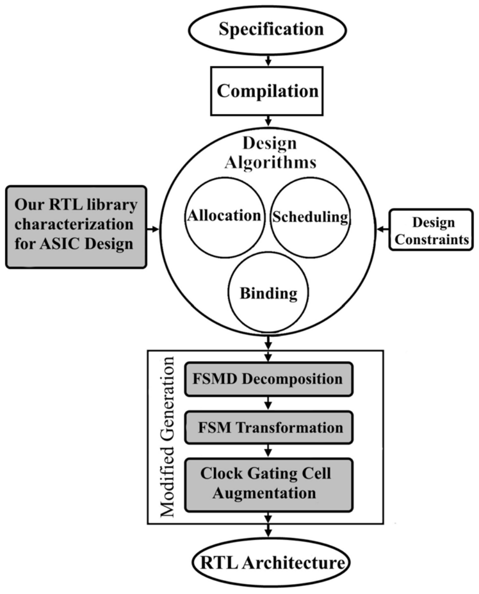

Fig. 5. Modified LegUp routines for clock gated ASIC design.

Therefore, an ASIC design library characterization similar to FPGA-based flow is required. We create proper library characterization by a commercial digital synthesis tool and CMOS 90 nm standard cell library.

In hybrid flow, LegUp gets the list of C functions of the program which should be implemented as hardware accelerators. Then generates Verilog modules for main functions and its sub-functions and finally attaches interfaces to them for bus communication. These Verilog modules include both FSM and datapath Verilog codes. To apply an RTL clock gating method to FSMD, we should modify LegUp to decompose FSMD Verilog module into separated FSM and datapath modules.

To exploit one-hot features for applying clock gating, the third modification in HLS is transforming FSM assignment from regular binary into one-hot. The final LegUp modification is adding clock gating cells to FSMD design. As shown in Fig. 3, for each FSM state, we need to add an extra latch with AND gate to one-hot FSM for behavioral simulation. For the gate-level simulation and power estimation, these gates are replaced with proper clock gating cells from the standard cell library. These extra logic gates have a little overhead on the one-hot FSM design. All of these LegUp modification are shown with gray boxes in Fig. 5.

VII. EXPERIMENTAL RESULTS

A. Experimental Flow

We use six CHStone benchmarks in our experiments. For each benchmark, the best function for implementing hardware accelerators has been chosen. At first for clock gating justification, we determine the amount of wasted energy in input capacitance of the unchanged FFs through an exact and non-probability data-to-clock toggling ratio and simple FF energy model (Section III). For this determination, we instrument the RTL code by adding some instructions and perform a behavioral simulation.

To evaluate our clock gating technique and measure power saving, both regular and clock gated hardware accelerators were synthesized with a commercial digital logic synthesis tool using 90 nm CMOS standard cell library. The gate-level simulation generates the switching activity data for each benchmark. These switching activity data are analyzed by a commercial power analysis tool and power estimation for both regular and clock gated design are obtained.

B. Experimental Results

The execution information for the datapath of each benchmark and their register toggle trace results are shown in Tables I and II. By analyzing this information with the metrics defined by (5) and (6) from Section III and according to our discussion in Section IV, we assure that microarchitecture-level clock gating is useful for power reduction in datapath circuits. By applying the proposed clock gating method, clock signal of idle states' FFs are gated, therefore, only current state's clock sub-tree is toggled. Based on the relative percentage of the idle states' FFs in the experimental circuits [M_2 in (6)], significant dynamic power reduction would be achieved.

Table V compares dynamic power consumption before and after clock gating and reports the power gains measured after clock gating for the six representative benchmarks. Dynamic power results are reported for the FSM and datapath separately. Our clock gating technique only gates the datapath's clock signal by adding the clock gating cells to the FSM. Dynamic power comparison between the FSM and FSM with clock gating cells shows that we have only a little power overhead on the FSMD design. Table IV shows that our clock gating technique which uses one-hot assignment attribute needs negligible area for its implementation.

TABLE V
AREA OVERHEAD FOR CLOCK GATED CIRCUIT

Hardware Accelerator	FSM (μm^2)	FSM + CGC (μm^2)	FSM Area Over-head (%)	FSMD (μm^2)	FSMD Area Over-head (%)
Encode	2800	3196	14.14	332994	0.119
Initialize_Buffer	1036	1196	15.44	31759	0.504
blowfish_main	9000	10439	15.99	468136	0.307
Gsm_LPC_Analysis	5739	6520	13.60	578668	0.135
float64_mul	2124	2413	13.60	123955	0.233
float64_add	4415	5018	13.65	169254	0.356

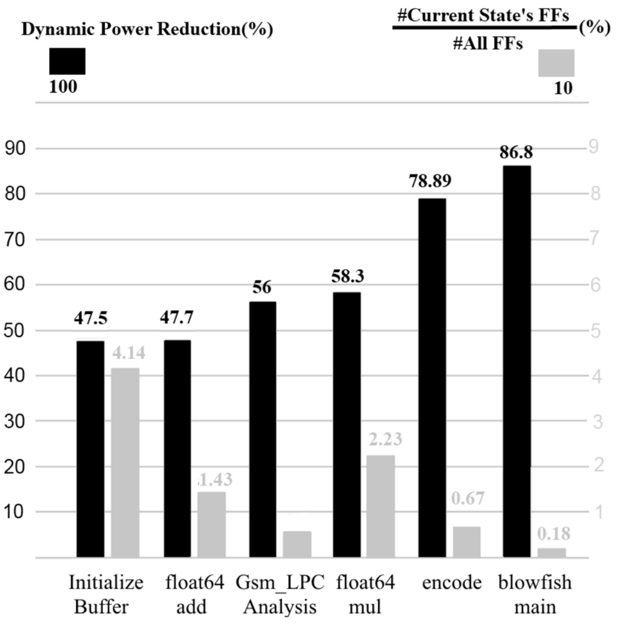

Fig. 6. Relation of dynamic power reduction and percentage of current state's FFs.

Fig. 6 shows the percentage of total active state's FFs in all execution clock cycles in relation to dynamic power reduction via the clock gating technique. In the hardware accelerator circuits of blowfish_main and encode, only 0.18% and 0.66% of FFs are active and clock gating reduces dynamic power more than 78% and 86%, respectively. While in Initialize_Buffer's circuit and float64_add circuit, 4.14% and 1.43% of FFs are active and the dynamic power reduction is almost 50%.

VIII. CONCLUSION

The clock distribution network currently represents a major dynamic power consumption bottleneck in ASIC design. In this paper, we have introduced a new fine-grained microarchitecture clock gating technique for hardware accelerators that reduces the clock tree power consumption and decrease energy wasting in low activity FFs. This technique is based on combining the clock gating cells (generate gated clocks for datapath) with a one-hot FSM. The clock gating methodology has been integrated into an HLS design flow and performs RTL clock gating automatically. We use LegUp HLS tool as the development platform and validation has been carried out on a set of practical C-based HLS benchmark programs. Our RTL clock gating technique in comparison to previous techniques does not need any extra computations for making clock gating conditions. Furthermore, area overheads of the clock gated circuits are less than 0.5%. The quality of our clock gating method does not depend on the FSMD size and FSM complexity, however, we obtain better power reduction for larger FSMDs. Based on the circuit size and active state's FFs, we have obtained up to 86% of dynamic power saving.

ACKNOWLEDGMENT

This work was supported by the School of Computer Science, Institute for Research in Fundamental Science under Grant CS1394-4-14.

REFERENCES

[1] G. Venkatesh *et al.*, "Conservation cores: Reducing the energy of mature computations," *ACM SIGARCH Comput. Architect. News*, vol. 38, no. 1, pp. 205–218, 2010.
[2] G. Venkatesh *et al.*, "Qscores: Trading dark silicon for scalable energy efficiency with quasi-specific cores," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit.*, Porto Alegre, Brazil, 2011, pp. 163–174.
[3] S. Gupta, S. Feng, A. Ansari, S. Mahlke, and D. August, "Bundled execution of recurring traces for energy-efficient general purpose processing," in *Proc. 44th Annu. IEEE/ACM Int. Symp. Microarchit.*, Porto Alegre, Brazil, 2011, pp. 12–23.
[4] J. Cong, P. Zhang, and Y. Zou, "Optimizing memory hierarchy allocation with loop transformations for high-level synthesis," in *Proc. 49th Annu. Design Autom. Conf.*, San Francisco, CA, USA, 2012, pp. 1233–1238.
[5] Q. Huang *et al.*, "The effect of compiler optimizations on high-level synthesis for FPGAs," in *Proc. IEEE 21st Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Seattle, WA, USA, Apr. 2013, pp. 89–96.
[6] T.-H. Yeh and S.-J. Wang, "Power-aware high-level synthesis with clock skew management," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 1, pp. 167–171, Jan. 2012.
[7] A. Sengupta and R. Sedaghat, "Integrated scheduling, allocation and binding in high level synthesis using multi structure genetic algorithm based design space exploration," in *Proc. IEEE 12th Int. Symp. Qual. Electron. Design (ISQED)*, Santa Clara, CA, USA, 2011, pp. 1–9.
[8] G. Lhairech-Lebreton, P. Coussy, D. Heller, and E. Martin, "Bitwidth-aware high-level synthesis for designing low-power DSP applications," in *Proc. 17th IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*, Athens, Greece, 2010, pp. 531–534.
[9] R. I. Bahar and S. Manne, "Power and energy reduction via pipeline balancing," in *Proc. 28th Annu. Int. Symp. Comput. Archit.*, Goteborg, Sweden, 2001, pp. 218–229.
[10] Y. Luo, J. Yu, J. Yang, and L. Bhuyan, "Low power network processor design using clock gating," in *Proc. 42nd Annu. Design Autom. Conf.*, Anaheim, CA, USA, 2005, pp. 712–715.
[11] H. Li, S. Bhunia, Y. Chen, K. Roy, and T. Vijaykumar, "DCG: Deterministic clock-gating for low-power microprocessor design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 3, pp. 245–254, Mar. 2004.
[12] R. Fraer, G. Kamhi, and M. K. Mhameed, "A new paradigm for synthesis and propagation of clock gating conditions," in *Proc. 45th ACM/IEEE Design Autom. Conf. (DAC)*, Anaheim, CA, USA, 2008, pp. 658–663.
[13] P. Babighian, L. Benini, and E. Macii, "A scalable algorithm for RTL insertion of gated clocks based on ODCs computation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 1, pp. 29–42, Jan. 2005.
[14] I. Han and Y. Shin, "Simplifying clock gating logic by matching factored forms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 6, pp. 1338–1349, Jun. 2014.
[15] L. Li, K. Choi, and H. Nan, "Activity-driven fine-grained clock gating and run time power gating integration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1540–1544, Aug. 2013.
[16] S. Wimer and A. Albahari, "A look-ahead clock gating based on auto-gated flip-flops," *IEEE Trans. Circuits Syst. I Reg. Papers*, vol. 61, no. 5, pp. 1465–1472, May 2014.
[17] L. Benini and G. De Micheli, "Automatic synthesis of low-power gated-clock finite-state machines," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 15, no. 6, pp. 630–643, Jun. 1996.

[18] K. Usami and H. Yoshioka, "A scheme to reduce active leakage power by detecting state transitions," in *Proc. 47th Midwest Symp. Circuits Syst. (MWSCAS)*, vol. 1. Hiroshima, Japan, 2004, pp. I-493–I-496.

[19] C.-H. Shin, M.-H. Oh, S. N. Kim, and S. W. Kim, "Fine-grained power gating of datapath using FSM," in *Proc. IEEE 2nd Int. Conf. Networked Embedded Syst. Enterprise Appl. (NESEA)*, Fremantle, WA, Australia, 2011, pp. 1–5.

[20] J. C. Monteiro and A. L. Oliveira, "Implicit FSM decomposition applied to low-power design," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 10, no. 5, pp. 560–565, Oct. 2002.

[21] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii, "CHStone: A benchmark program suite for practical C-based high-level synthesis," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Seattle, WA, USA, May 2008, pp. 1192–1195.

[22] T. Lang, E. Musoll, and J. Cortadella, "Individual flip-flops with gated clocks for low power datapaths," *IEEE Trans. Circuits Syst. II Analog Digit. Signal Process.*, vol. 44, no. 6, pp. 507–516, Jun. 1997.

[23] N. H. E. Weste and D. M. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective*. Boston, MA, USA: Addison-Wesley, 2011.

[24] D. A. Huffman, "The synthesis of sequential switching circuits," *J. Franklin Inst.*, vol. 257, no. 3, pp. 161–190, 1954.

[25] S. Golson, "State machine design techniques for Verilog and VHDL," *Synopsys J. High-Level Design*, vol. 9, pp. 1–48, Sep. 1994.

[26] J. Oh and M. Pedram, "Gated clock routing for low-power microprocessor design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 6, pp. 715–722, Jun. 2001.

[27] A. Canis *et al.*, "LegUp: An open-source high-level synthesis tool for FPGA-based processor/accelerator systems," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 2, 2013, Art ID 24.

[28] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proc. Int. Symp. Code Gener. Optim. (CGO)*, 2004, pp. 75–86.

Mohsen Riahi Alam was born in Shiraz, Iran, in 1987. He received the B.Sc. degree in computer engineering from Shiraz University, Shiraz, Iran, in 2011, and the M.Sc. degree in computer architecture from the University of Tehran, Tehran, Iran, in 2014.

He is currently a Research Assistant with the Silicon Intelligence and very large-scale integration (VLSI) Signal Processing Laboratory, University of Tehran. His current research interests include embedded system design, low power and energy-efficient design, power analysis for VLSI circuits, specialized architecture, and high-level synthesis.

Mostafa Ersali Salehi Nasab was born in Kerman, Iran, in 1978. He received the B.Sc. degree in computer engineering from the University of Tehran, Tehran, Iran, in 2001, the M.Sc. degree in computer architecture from the University of Amirkabir, Tehran, in 2003, and the Ph.D. degree in the School of Electrical and Computer Engineering, University of Tehran in 2010.

From 2004 to 2008, he was a Senior Digital Designer with SINA Microelectronics Inc., Tehran, and the Technology Park of University of Tehran, Tehran. He is currently an Assistant Professor with the University of Tehran. His current research interests include novel techniques for high-performance, low-power, and fault-tolerant embedded system design.

Sied Mehdi Fakhraie received the M.Sc. degree in electronics from the University of Tehran, Tehran, Iran, in 1989, and the Ph.D. degree in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 1995.

He was a Professor with the School of Electrical and Computer Engineering, University of Tehran. He was the Director of the Silicon Intelligence and VLSI Signal Processing Laboratory, the Electrical and Electronics Engineering, and the Computer Hardware Engineering with the School of Electrical and Computer Engineering, University of Tehran. He was a Visiting Professor with the University of Toronto from 1998 to 2000. He was with Valence Semiconductor Inc., Irvine, CA, USA, from 2000 to 2003. He was in Dubai, United Arab Emirates, and Markham, Canada Offices of Valence as the Director of Application Specified Integrated Circuit (ASIC) and System-on-a-Chip Design, and the Technical Leader of Integrated Broadband Gateway and Family Radio System Baseband Processors. He was involved in many industrial integrated circuit design projects, including design of network processors and home gateway access devices, digital subscriber line modems, pagers, and digital signal processors for personal and mobile communication devices. His last research interests include system design and ASIC implementation of integrated systems, novel techniques for high-speed digital circuit design, and system-integration and efficient VLSI implementation of intelligent systems. He has co-authored a book entitled *VLSI-Compatible Implementations for Artificial Neural Networks* (Kluwer: Boston, MA, USA, 1997). He has authored/co-authored over 230 reviewed conference and journal papers. He passed away on December 7, 2014.