A Look-Ahead Clock Gating Based on Auto-Gated Flip-Flops

Shmuel Wimer, *Member, IEEE*, and Arye Albahari

***Abstract*—Clock gating is very useful for reducing the power consumed by digital systems. Three gating methods are known. The most popular is synthesis-based, deriving clock enabling signals based on the logic of the underlying system. It unfortunately leaves the majority of the clock pulses driving the flip-flops (FFs) redundant. A data-driven method stops most of those and yields higher power savings, but its implementation is complex and application dependent. A third method called auto-gated FFs (AGFF) is simple but yields relatively small power savings. This paper presents a novel method called *Look-Ahead Clock Gating* (LACG), which combines all the three. LACG computes the clock enabling signals of each FF one cycle ahead of time, based on the present cycle data of those FFs on which it depends. It avoids the tight timing constraints of AGFF and data-driven by allotting a full clock cycle for the computation of the enabling signals and their propagation. A closed-form model characterizing the power saving per FF is presented. It is based on data-to-clock toggling probabilities, capacitance parameters and FFs' fan-in. The model implies a breakeven curve, dividing the FFs space into two regions of positive and negative gating return on investment. While the majority of the FFs fall in the positive region and hence should be gated, those falling in the negative region should not. Experimentation on industry-scale data showed 22.6% reduction of the clock power, translated to 12.5% power reduction of the entire system.**

***Index Terms*—Clock gating, clock networks, dynamic power reduction.**

I. Introduction

ONE of the major dynamic power consumers in computing and consumer electronics products is the system's clock signal, typically responsible for 30% to 70% of the total dynamic (switching) power consumption [1]. Several techniques to reduce the dynamic power have been developed, of which clock gating is predominant. Ordinarily, when a logic unit is clocked, its underlying sequential elements receive the clock signal regardless of whether or not their data will toggle in the next cycle. With clock gating, the clock signals are ANDed with explicitly predefined enabling signals. Clock gating is employed at all levels: system architecture, block design, logic design and gates [2], [3]. Several methods to take advantage of this technique are described in [4]–[6], with all of them relying on various heuristics in an attempt to increase clock gating opportunities. We call the above methods *synthesis-based*.

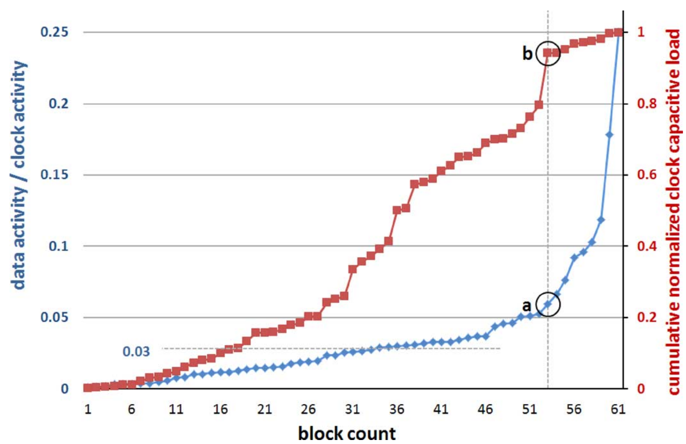

Fig. 1. Average data-to-clock toggling ratio and clock capacitive load of 61 blocks comprising 200 k FFs (blue curve). The red curve is the corresponding cumulative clock capacitive load.

Synthesis-based clock gating is the most widely used method by EDA tools [7]. The utilization of the clock pulses, measured by data-to-clock toggling ratio, left after the employment of synthesis-based gating may still be very low. Fig. 1 depicts the average data-to-clock toggling ratio, obtained by extensive power simulations of 61 blocks comprising 200 k FFs, taken from a 32 nm high-end 64-bit microprocessor. Those are mostly control blocks of the data-path, register-file and memory management units of the processor. The technology parameters used throughout the papers are of 22 nm low-leakage process technology.

Their clock enabling signals were derived by a mix of logic synthesis and manual definitions. The clock capacitive load is 70% of their total load. The blocks are increasingly ordered by their data-to-clock activity ratio. It is clearly shown that the data toggles in a very low rate compared to the gated clocks. Point (a) shows that in 87% of the blocks (53/61) the data toggles less than 6% compared to the gated clock, where the average shown by the horizontal dashed line is 3%. Fig. 1 also plots the corresponding cumulative clock capacitive load. Point (b) shows that the above 87% blocks are responsible for 95% of the total clock load. Consequently, the switching of a significant portion of the system's clock load is redundant, but consumes most of its power. This calls for other than synthesis-based methods to

Manuscript received May 02, 2013; revised August 19, 2013; accepted September 24, 2013. Date of publication January 02, 2014; date of current version April 24, 2014. This paper was recommended by Associate Editor M. Alioto.

S. Wimer is with the Electrical Engineering Faculty, Technion—Israel Institute of Technology, Haifa 32000, Israel and also with Bar-Ilan University, Ramat-Gan 5900, Israel (e-mail: wimer@ee.technion.ac.il).

A. Albahari is with Intel Corporation, Israel Design Center, Haifa 31095, Israel (e-mail: arye.albahari@intel.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2013.2289404

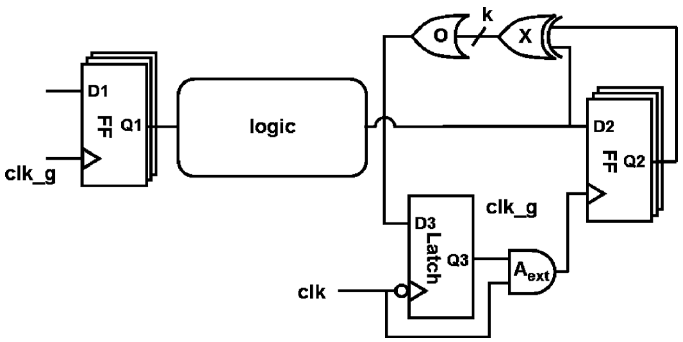

Fig. 2. Circuit implementation of data-driven clock gating.

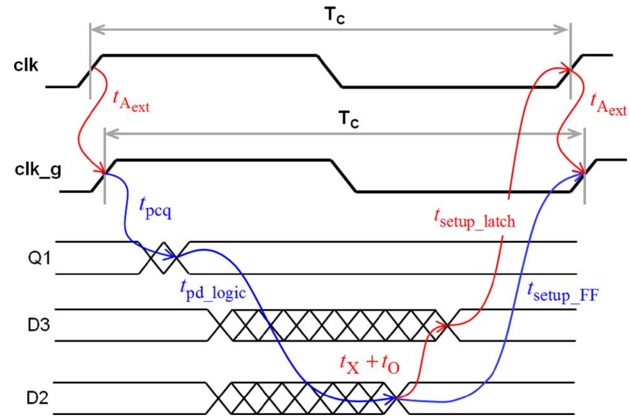

Fig. 3. Sequencing of gating logic in data-driven clock gating.

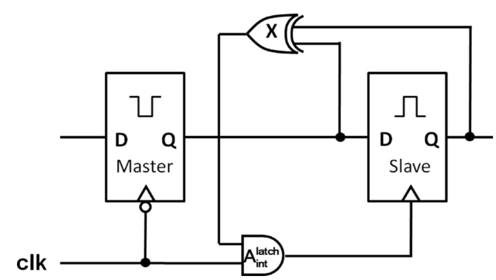

Fig. 4. An auto-gated flip-flop.

stop the 97% redundant clock pulses. A very low data-to-clock toggling ratio was also reported in [8], where extensive power simulations of a wide variety of industrial designs showed average toggling ratios of 0.02 to 0.05.

To address the above redundancy, a method called *data-driven clock gating* was proposed for flip-flops (FFs). There, the clock signal driving a FF, is disabled (gated) when the FF's state is not subject to change in the next clock cycle [9]. In an attempt to reduce the overhead of the gating logic, several FFs are driven by the same clock signal, generated by ORing the enabling signals of the individual FFs [8]. Based on the data-to-clock toggling probability, a model to derive the group size maximizing the power savings was developed. A comparison between the synthesis-based and data-driven gating methods showed that the latter outperforms for control and arithmetic circuits, while the former outperforms for register-file based circuits [10].

Data-driven gating is illustrated in Fig. 2. A FF finds out that its clock can be disabled in the next cycle by XORing its output with the present input data that will appear at its output in the next cycle. The outputs of k XOR gates are ORed to generate a joint gating signal for k FFs, which is then latched to avoid glitches. The combination of a latch with AND gate is used by commercial tools and is called Integrated Clock Gate (ICG) [11]. It is beneficial to group FFs whose switching activities are highly correlated. The work in [10] addressed the questions of which FFs should be placed in a group to maximize the power reduction, and how to find those groups.

Data-driven gating suffers from a very short time-window where the gating circuitry can properly work. This is illustrated in Fig. 3. The cumulative delay of the XOR, OR, latch and the AND gater must not exceed the setup time of the FF. Such constraints may exclude 5%-10% of the FFs from being gated due to their presence on timing critical paths [10]. The exclusion percentage increases with the increase of critical paths, a situation occurring by downsizing or turning transistors of non-critical path to high threshold voltage (HVT) for further power savings.

Another difficulty of data-driven gating is its design methodology. To maximize the power savings, the FFs should be grouped such that their toggling is highly correlated. This requires running extensive simulations characterizing the typical applications expected by the end-user. Those applications are in many cases unknown and the amount of redundant clock pulses may significantly increase for specific applications. Furthermore, IP providers who are delivering RTL code need to cast the gating circuitry per customer, which requires maintaining different versions of the same IP.

This paper proposes *Look-Ahead Clock Gating* (LACG). It computes the clock enabling signals of each FF one cycle ahead of time, based on the present cycle data of those FFs on which it depends. Similarly to data-driven gating, it is capable of stopping the majority of redundant clock pulses. It has however a big advantage of avoiding the tight timing constraints of AGFF and data-driven, by allotting a full clock cycle for the enabling signals to be computed and propagate to their gaters. Furthermore, unlike data-driven gating whose optimization requires the knowledge of FFs' data toggling vectors, LACG is independent of those. The embedding of LACG logic in the RTL functional code is uniquely defined and easily derived from the underlying logic, independently of the target application. This simplification is advantageous as it significantly simplifies the gating implementation.

The rest of the paper discusses the modeling, analysis, circuits, optimization and implementation of LACG. Section II presents the LACG circuits. Section III develops its power savings model. Section IV minimizes the logic overhead required to generate the clock enabling signals. Experimental results are shown in Section V. We conclude in Section VI.

II. Auto-Gated Flip-Flops

The basic circuit used for LACG is *Auto-Gated Flip-Flip* (AGFF) illustrated in Fig. 4 [12].

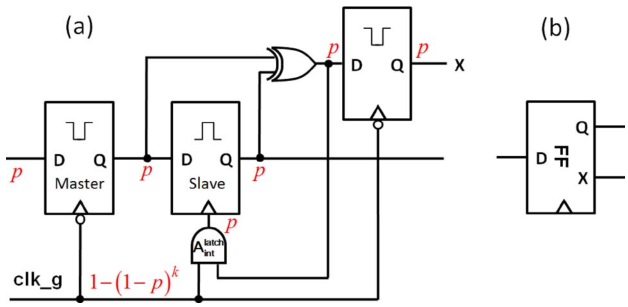

Fig. 5. Enhanced AGFF with XOR output used for LACG.

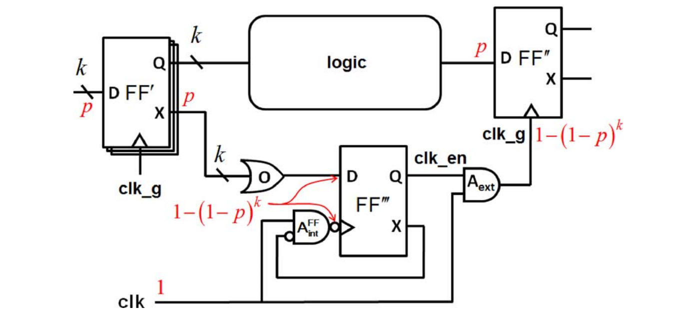

Fig. 6. LACG of general logic.

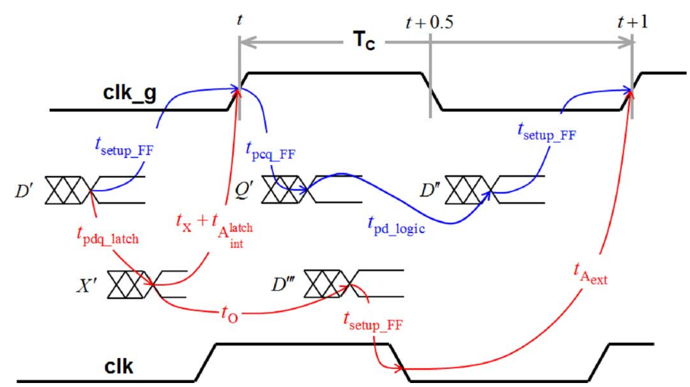

Fig. 7. Timing sequencing of LACG clock gating.

The FF's master latch becomes transparent on the falling edge of the clock, where its output must stabilize no later than a setup time prior to the arrival of the clock's rising edge, when the master latch becomes opaque and the XOR gate indicates whether or not the slave latch should change its state. If it does not, its clock pulse is stopped and otherwise it is passed. In [12] a significant power reduction was reported for register-based small circuits, such as counters, where the input of each FF depends on the output of its predecessor in the register. AGFF can also be used for general logic, but with two major drawbacks. Firstly, only the slave latches are gated, leaving half of the clock load not gated. Secondly, serious timing constraints are imposed on those FFs residing on critical paths, which avoid their gating.

LACG takes AGFF a leap forward, addressing three goals; stopping the clock pulse also in the master latch, making it applicable for large and general designs and avoiding the tight timing constraints. LACG is based on using the XOR output in Fig. 4 to generate clock enabling signals of other FFs in the system, whose data depend on that FF. There is a problem though. The XOR output is valid only during a narrow window of $[-t_{\text{setup}}, t_{\text{ccq}}]$ around the clock rising edge, where t_{setup} and t_{ccq} are the FF's setup time and clock to output contamination delay, respectively. After a t_{ccq} delay the XOR output is corrupted and turns eventually to zero. To be valid during the entire positive half cycle it must be latched as shown in Fig. 5(a). Fig. 5(b) is the symbol of the enhanced AGFF with the XOR output. The power consumed by the new latch can be reduced by gating its clock input clk_g. Such gating has been proposed in [16] and it involves another XOR and OR gates, useful for high clock switching probability. It is subsequently shown that clk_g probability is very low and it is therefore not further being gated.

Fig. 6 illustrates how LACG works. We call FF$''$ *target* and FF$'$ *source*. A target FF depends on $k \geq 1$ source FFs. It is required that the logic driving a target FF does not have an input externally of the block. Let $\mathbf{X}(D'')$ denote the set of the XOR outputs of the source FFs, and denote by $\mathbf{Q}(D'')$ the set of their corresponding outputs. The source FFs can be found by a traversal of the logic paths from D'' back to $\mathbf{Q}(D'')$, which can be performed either in the RTL or the net-list descriptions of the underlying system. The logic tree with root D'' and leaves $\mathbf{Q}(D'')$ is sometimes called the logic cone of D'' [13].

Let t and $t+1$ be two successive clock cycles shown in Fig. 7, where the time tics refer to the rising edge of the clock pulses. We use the notation $t-0.5$ and $t+0.5$ to denote the clock's preceding and succeeding falling edges, respectively. Clearly, $\sum_{\mathbf{X}(D'')} X(t) = 0$ is a sufficient condition for FF$''$ not to change state at $t+1$, where the summation means logical OR operation. FF$''$'s clock pulse could therefore be disabled at $t+1$ to save the switching power.

To generate the enabling signal obtained from data at t and ensure its validity at $t+1$, an oppositely clocked FF is introduced as shown in Fig. 6. Upon the clock's falling edge at $t+0.5$ there exists $D'''(t+0.5) = \sum_{\mathbf{X}(D'')} X(t)$. Since FF$'''$ is oppositely clocked, there exists $Q'''(t+0.5) = D'''(t+0.5) = \sum_{\mathbf{X}(D'')} X(t)$. The signal Q''' is stable during the time period $[t+0.5, t+1]$, obtaining $Q'''(t+1) = \sum_{\mathbf{X}(D'')} X(t)$. The gater A_{ext} can then appropriately gate the clock's rising edge at $t+1$ which drives FF$''$.

Using a FF for gating is a considerable overhead that will consume power of its own. This can significantly be reduced by gating FF''' as shown in Fig. 6. Notice that since FF''' is oppositely clocked and its data is sampled at the clock's falling edge, its clock enabling signal X''' must be negated. Also, FF''' is an ordinary FF where the internal XOR gate is connected between D''' and Q'''.

The signal sequencing of LACG is illustrated in Fig. 7. The delays of the ordinary logic are colored in blue, while those of the gating logic overhead are colored in red. LACG is advantageous over data-driven. While the latter must complete gating evaluation within t_{setup} delay, LACG has a full clock cycle to evaluate clk_en from $\mathbf{X}(D'')$. Real implementation may require long wires to generate the clock enabling signals, so the grace of a full cycle is a big relief.

The timing constraints imposed on LACG are derived from Fig. 7. A first constraint is

$$t_{\mathrm{pdq\_latch}} + t_{\mathrm{X}} + t_{\mathrm{A}_{\mathrm{int}}^{\mathrm{latch}}} \leq t_{\mathrm{setup\_FF}}, \tag{1}$$

where $t_{\mathrm{pdq\_latch}}$, t_{X} and $t_{\mathrm{A}_{\mathrm{int}}^{\mathrm{latch}}}$ are respectively the data to output propagation delay of the FF's internal latch, the delay of the FF's internal XOR gate and the FF's internal AND gater shown in Fig. 5. $t_{\mathrm{setup\_FF}}$ is the FF's setup time. A second constraint is

$$t_{\mathrm{O}} + t_{\mathrm{setup\_FF}} + t_{\mathrm{pcq\_FF}} + t_{\mathrm{A}_{\mathrm{ext}}} < Tc + t_{\mathrm{X}} + t_{\mathrm{A}_{\mathrm{int}}^{\mathrm{latch}}}, \tag{2}$$

where t_{O} is the delay of the OR logic required to capture the kX FFs' outputs on which the gating of a FF depends, and $t_{\mathrm{A}_{\mathrm{ext}}}$ is the delay of the clock driving network generating the gated clock signal. The constraint in (2) is independent of timing critical paths and thus can easily be satisfied. This is a big relaxation over the constraints in Fig. 3, where the clock enabling evaluation and propagation is constrained to a $t_{\mathrm{setup\_FF}}$ time window.

As mentioned before $\sum_{\mathbf{X}(D'')} X(t) = 0$ is a sufficient clock disabling condition of FF$''$ at $t+1$. This unfortunately is not a necessary condition, since it is possible that due to the specific logic by which D'' is evaluated, there is $D''(t+1) = D''(t)$ while $\sum_{\mathbf{X}(D'')} X(t) = 1$. The clock pulse driving FF$''$ at $t+1$ will thus be redundant. A key question is therefore how large $\Pr\left[(D''(t+1) = D''(t)) \wedge \left(\sum_{\mathbf{X}(D'')} X(t) = 1\right)\right]$ is. It should ideally be zero, but practically it is not. Assuming the worst-case situation where the FFs toggle independently of each other, it is subsequently shown that the probability of clock redundancy to occur by LACG is small. LACG power savings is analyzed under a worst-case toggling independence model, so reality may yield higher power savings than the following analysis does.

III. Modeling the Power Savings

Let X be a random variable of the FF's data-to-clock toggling (hereby data toggling) and let $p = \Pr[X = 1]$ be its probability. Assuming that FFs are toggling their data independently of each other, there exists

$$\Pr\left[\sum_{\mathbf{X}(D'')} X(t) = 0\right] = (1-p)^k. \tag{3}$$

Notice that independency is a worst case assumption. In reality toggling correlation exists, which may increase the actual power savings obtained by the subsequent analysis [10]. It follows from X independency that the probability of enabling the clock while it could be disabled is

$$\Pr\left[\sum_{\mathbf{X}(D'')} X(t) = 1 \wedge X''(t+1) = 0\right] = \left[1-(1-p)^k\right](1-p). \tag{4}$$

We subsequently formulate the power savings in terms of capacitance and data toggling probability, since frequency and voltage do not matter for relative savings calculation. The product of capacitance and data toggling probability is called in VLSI jargon as *dynamic capacitance* or *cdyn* for short. We will use the terms power and cdyn interchangeably.

In Figs. 5 and 6 the toggling probabilities of the various nodes in the AGFF and LACG are shown in red color. Let c_{FF} be the clock input capacitance of a FF, and let $c_{\mathrm{FF+CLK}}$ include also the clock driver and its interconnecting wire capacitance. We charge 1/3 of $c_{\mathrm{FF+CLK}}$ to each of the three latches comprising the AGFF in Fig. 5. The saved cdyn stems from the low clocking rate of the master and slave latches shown in Fig. 5, which due to the LACG has $1-(1-p)^k$ probability, while otherwise those would have been clocked in one probability. It follows from (3) that the FF's cdyn savings in the master latch is $(1-p)^k c_{\mathrm{FF+CLK}}/3$. Unlike the master latch whose gating is determined by the k related source FFs, the auto gating of the slave latch is determined by the toggling of the target FF itself, independently of k. Nevertheless, being somewhat conservative we consider the internal AND gate $\mathrm{A}_{\mathrm{int}}^{\mathrm{latch}}$ as a part of the slave latch and charge a saving of $(1-p)^k c_{\mathrm{FF+CLK}}/3$ rather than $(1-p)\, c_{\mathrm{FF+CLK}}/3$.

The above savings does not come for free. To calculate the LACG overhead, consider the toggling probabilities shown in Figs. 5 and 6. Let c_{X} be the capacitance of the XOR gate, introducing pc_{X} cdyn overhead. We also added in Fig. 5 a latch, which introduces $\left[1-(1-p)^k\right] c_{\mathrm{FF+CLK}}/3$ cdyn overhead. The savings occurring within the target FF is therefore

$$\left\{2(1-p)^k - \left[1-(1-p)^k\right]\right\} \frac{c_{\mathrm{FF+CLK}}}{3} - pc_{\mathrm{X}} = \left[3(1-p)^k - 1\right] \frac{c_{\mathrm{FF+CLK}}}{3} - pc_{\mathrm{X}}. \tag{5}$$

Another cdyn overhead is introduced by FF$'''$ in forming one cycle delay. While FF$'''$ is gated and hence c_{FF} is multiplied by its clock enabling probability, $\mathrm{A}_{\mathrm{int}}$ is connected to the ungated clock and hence its toggling probability is one. The resulting overhead is therefore $\left[1-(1-p)^k\right] c_{\mathrm{FF}} + c_{\mathrm{A}_{\mathrm{int}}}$.

Another cdyn overhead occurs by the OR logic. Let c_{O} be the capacitance per input of the k-way OR gate shown in Fig. 5, including the wire connected to the X output of a source FF. Due to fan-in limits, the OR gate is usually implemented as a tree. The toggling probability of an OR gate input is p, introducing a kpc_{O} cdyn overhead. This is somewhat pessimistic since OR sub-trees can be shared among different FFs, which EDA logic synthesis tools are capable of optimizing. This is discussed in Section IV. The toggling probability of the OR gate output is $1-(1-p)^k$.

Summing up all the above components, the following cdyn overhead results in

$$\left[1-(1-p)^k\right] c_{\mathrm{FF}} + c_{\mathrm{A}_{\mathrm{int}}} + \left[1-(1-p)^k + kp\right] c_{\mathrm{O}}. \tag{6}$$

The net cdyn savings per target FF, denoted by $c_{\mathrm{dyn}}^{\mathrm{save}}$, is obtained by subtracting (6) from (5), which after rearrangement yields

$$c_{\mathrm{dyn}}^{\mathrm{save}} = (1-p)^k \left(c_{\mathrm{FF+CLK}} + c_{\mathrm{FF}} + c_{\mathrm{O}}\right) - p\left(c_{\mathrm{X}} + kc_{\mathrm{O}}\right) - \left(\frac{c_{\mathrm{FF+CLK}}}{3} + c_{\mathrm{A}_{\mathrm{int}}} + c_{\mathrm{FF}} + c_{\mathrm{O}}\right). \tag{7}$$

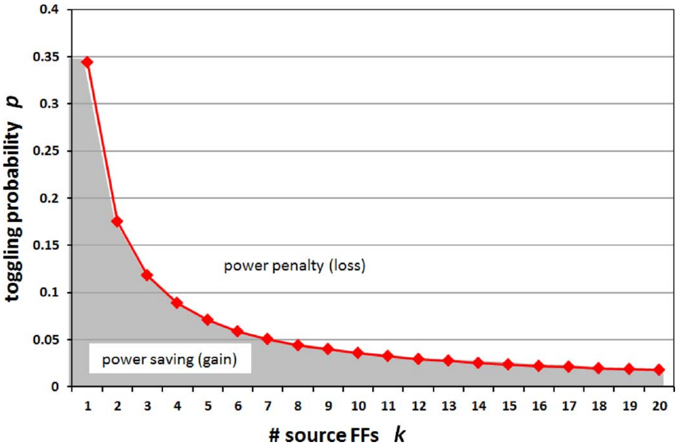

Fig. 8. Power saving breakeven curve.

TABLE I
TYPICAL CAPACITANCES IN 22 NM PROCESS TECHNOLOGY, VALUES ARE IN 10^{-15} F.

C_{FF}	C_{CLK}	C_{FF+CLK}	C_X	C_O	C_{Aint}
25.7	33.5	36.9	2.9	3.1	1.7

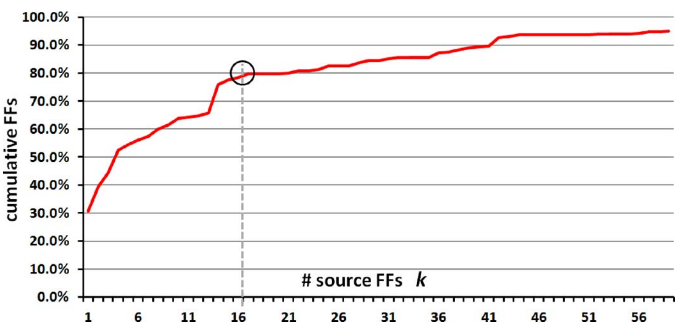

Fig. 9. The cumulative distribution of k in a block of 6 k FFs.

It is not difficult to verify from (7) that $c_{\text{dyn}}^{\text{save}}$ is decreasing with the increase of p and k. Clearly, large values of those may result in power loss rather than savings. We subsequently characterize the breakeven point. Substitute $c_{\text{dyne}}^{\text{save}} = 0$ in (7) implies a dependency between p and k, where the values of the various capacitances are known from the characterization of the cell library in use and by estimating the interconnecting wires. The dependency is shown by the Shmoo plot in Fig. 8 for the parameters in Table I, taken from a 22 nm process technology cell library. The capacitances are measured in 10^{-15}F.

Those FFs whose (k, p) point fall in the shaded area below the curve represent LACG that saves power, while for FFs whose (k, p) points fall above the curve LACG will lose power, and therefore they should not be gated. The smaller k is the higher power savings can potentially be achieved. Fig. 9 shows the distribution of k in a typical block comprising 6 k FFs, taken from a data cash control. With the reasonable assumption of average data toggling rate of 0.03 (see Fig. 1), Fig. 8 shows that LACG of FFs satisfying $k \leq 15$ will save power. According to Fig. 9 this applied for about 80% of the FFs.

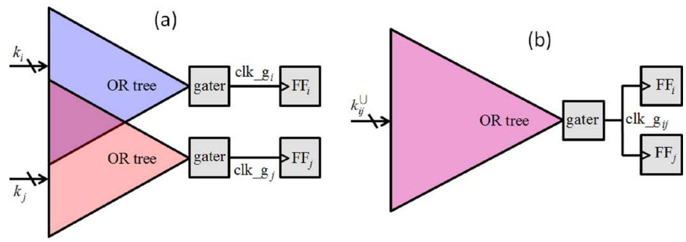

Fig. 10. Merging OR logic for joint gating.

The dynamic power overhead of LACG has been considered in the above breakeven analysis. There is also static power overhead. It should be noted that due to the full cycle allotted for the derivation of the enabling signals, the logic involved uses high threshold voltage and smallest devices. Moreover, as shown in the next section, the gating logic can be shared among several target FFs, which further reduces the overhead. We decided on LACG for a FF if it falls some safeguard margin apart the curve to compensate for leakage overhead. The detailed discussion of the design methodology is beyond the scope of this work.

IV. MINIMIZING THE GATING LOGIC

The savings expression in (7) assumed a separate gating logic for each target FF. This consumes a considerable power and area, and the gating logic should therefore be minimized. There are many cases where few target FFs depend on similar source FFs. In such cases there is no point in generating separate clock gating signals.

We subsequently develop logic sharing model to minimize the gating cost. The idea is illustrated in Fig. 10(a), showing two target FFs, FF_i and FF_j, with their corresponding OR trees, driven by k_i and k_j source FFs, respectively. FF_i and FF_j have common source FFs, shown pictorially by the overlap of the trees. A different implementation is shown in Fig. 10(b) were the OR logic is merged and a single gater is used for the two FFs. The larger the overlap is, the more desirable is the merge. In addition to logic reduction, the number of clock drivers and gaters will also be reduced.

OR logic merging however does not come for free. It may increase the amount of redundant clock pulses since the clock gater is driven by a wider tree comprising more source FFs, which increases the clock enabling probability. To consider the gain and loss balance, let $\mathbf{S}(\text{FF}_i)$ and $\mathbf{S}(\text{FF}_j)$ denote the source FFs sets of FF_i and FF_j, and let their size be $k_i = |\mathbf{S}(\text{FF}_i)|$ and $k_j = |\mathbf{S}(\text{FF}_j)|$, respectively. The total number of inputs of the merged OR logic is $k_{ij}^{\cup} = |\mathbf{S}(\text{FF}_i) \cup \mathbf{S}(\text{FF}_j)|$. As shown in Fig. 10(b) the merging eliminates one gater (comprising FF$'''$ and A_{ext} in Fig. 6). The amount $k_{ij}^{\cap} = |\mathbf{S}(\text{FF}_i) \cap \mathbf{S}(\text{FF}_j)|$ indicates how much of the OR logic can be shared (and thus be saved) by the merge.

The efficiency of the OR logic merging depends on the $k_{ij}^{\cap}/k_{ij}^{\cup}$ ratio. It follows that $0 \leq k_{ij}^{\cap}/k_{ij}^{\cup} \leq 1$. In the best case $k_{ij}^{\cap} = k_{ij}^{\cup}$, while the worst case is $k_{ij}^{\cap} = 0$. The higher the ratio is, the less OR logic overhead is used to jointly gate the clocks of FF_i and FF_j. Fig. 11 shows the distribution of $k_{ij}^{\cap}/k_{ij}^{\cup}$ for all the FF pairs of the 6 k FFs block shown in Fig. 9. Obviously, FF pairs whose $k_{ij}^{\cap}/k_{ij}^{\cup}$ is nearly 1 are favored for merging,

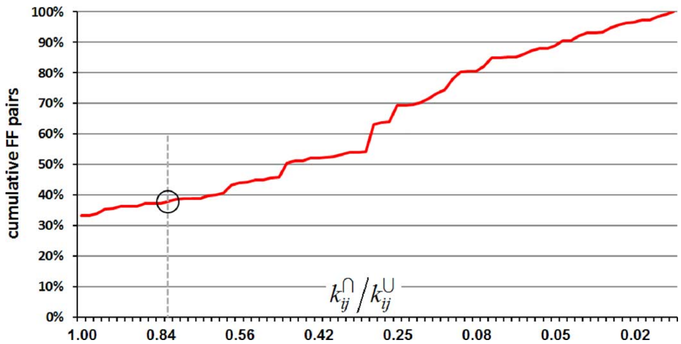

Fig. 11. All FF pairs $k_{ij}^{\cap}/k_{ij}^{\cup}$ cumulative distribution of a 6 k FFs block of Fig. 9.

yielding 50% reduction of the gating logic. Fig. 11 shows that for 38% of the pairs there is $0.8 \leq k_{ij}^{\cap}/k_{ij}^{\cup} \leq 1$, potentially reducing the gating logic of those pairs by 40% to 50%.

Having n FFs, there are $n(n-1)/2$ possible pairs, and the question of how among those to choose the $n/2$ best pairs, follows. An algorithm is subsequently presented. Consider the cdyn savings per FF in expression (7). For sufficiently small p and k it can be approximated by the following linearization

$$c_{\mathrm{dyn}}^{\mathrm{save}} \simeq \underbrace{\frac{2c_{\mathrm{FF+CLK}}}{3} - c_{\mathrm{A_{int}}} - pc_{\mathrm{X}}}_{(a)} - \underbrace{pk\left(c_{\mathrm{FF+CLK}} + c_{\mathrm{FF}} + 2c_{\mathrm{O}}\right)}_{(b)}. \quad (8)$$

Term (a) is independent of k and thus not affected by the FF merging, while term (b) does. The change $\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}}$ in savings resulting by merging the gating logic of FF_i and FF_j is therefore

$$\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}} = p\underbrace{(k_i + k_j)\left(c_{\mathrm{FF+CLK}} + c_{\mathrm{FF}} + 2c_{\mathrm{O}}\right)}_{(a)} - p\underbrace{k_{ij}^{\cup}\left(2c_{\mathrm{FF+CLK}} + c_{\mathrm{FF}} + 2c_{\mathrm{O}}\right)}_{(b)}. \quad (9)$$

Term (a) in (9) follows from the independent gating of FF_i and FF_j, while term (b) follows from their joint gating. Notice that $c_{\mathrm{FF+CLK}}$ in (b) is multiplied by two since as shown in Fig. 10(b), both FF_i and FF_j are clocked by clk_g_{ij}. Substitution of $k_{ij}^{\cup} = k_i + k_j - k_{ij}^{\cap}$ in (9) yields

$$\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}} = p\left[\underbrace{k_{ij}^{\cap}\left(c_{\mathrm{FF}} + 2c_{\mathrm{O}}\right)}_{(a)} - \underbrace{\left(k_i + k_j - 2k_{ij}^{\cap}\right)c_{\mathrm{FF+CLK}}}_{(b)}\right]. \quad (10)$$

Terms (a) and (b) in (10) are the increase and decrease, respectively, of the power savings due to the joint clock gating. To validate the correctness of (10), consider the case $\mathbf{S}(\mathrm{FF}_i) = \mathbf{S}(\mathrm{FF}_j)$, and hence $k_i = k_j = k_{ij}^{\cup} = k_{ij}^{\cap}$, where maximal $\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}}$ is expected. Indeed, term (b) in (10) vanishes. In the opposite case $\mathbf{S}(\mathrm{FF}_i) \cap \mathbf{S}(\mathrm{FF}_j) = \emptyset$ and $k_{ij}^{\cap} = 0$, where minimal $\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}}$ (maximal negative) is expected. Indeed, term (a) in (10) vanishes.

To characterize whether the merging the gating logic of FF_i and FF_j pays, we set $\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}} = 0$ in (10). Since there is always $2 \leq (k_i + k_j)/k_{ij}^{\cap}$, the following merging criteria follows

$$2 \leq \frac{k_i + k_j}{k_{ij}^{\cap}} < 2 + \frac{c_{\mathrm{FF}} + 2c_{\mathrm{O}}}{c_{\mathrm{FF+CLK}}}. \quad (11)$$

Substituting for instance the values of Table I in (11) yields the merging criteria $2 \leq (k_i + k_j)/k_{ij}^{\cap} < 2.86$ for the 22 nm technology and cell library used by this work. Only those pairs satisfying (11) should be considered for merging. Those pairs that do not will decrease the cdyn savings compared to gating FF_i and FF_j independently. One can use the data of Fig. 11 to find out how many of the FFs pairs satisfy (11). Substitution of $k_i + k_j = k_{ij}^{\cup} - k_{ij}^{\cap}$, the inequality $2 \leq (k_i + k_j)/k_{ij}^{\cap} < 2.86$ for the 22 nm technology is equivalent to $0.53 < k_{ij}^{\cap}/k_{ij}^{\cup} \leq 1$. Fig. 11 shows that it is satisfied by more than 40% of the pairs. For the 6 k FFs of the block, there are $0.40 \times \left(6 \times 10^3\right)^2/2 = 7.2 \times 10^6$ useful candidate pairs.

To maximize the total of $\delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}}$, we define an n-vertex complete weighted graph $G(V, E, w)$, called *gating logic merge graph*. A vertex $v_i \in V$ corresponds to FF_i and an edge $e_{ij} = (v_i, v_j) \in E$ corresponds to a $(\mathrm{FF}_i, \mathrm{FF}_j)$ pair. An edge e_{ij} is assigned a weight $w(e_{ij})$ as follows:

$$w(e_{ij}) = \begin{cases} \delta c_{\mathrm{dyn}\,ij}^{\mathrm{save}} & \text{if (11) is satisfied} \\ 0 & \text{otherwise} \end{cases}. \quad (12)$$

The reason for the zero weight in (12) is to avoid a merge that reduces cdyn savings. In this setup we wish to find a *Perfect Graph Matching*[14] $E' \subset E$, $|E'| = n/2$, of $G(V, E, w)$, maximizing the expression $\Delta c_{\mathrm{dyn}}^{\mathrm{save}}$ that counts the extra cdyn savings obtained by merging,

$$\Delta c_{\mathrm{dyn}}^{\mathrm{save}} = \sum_{e_{ij} \in E'} w(e_{ij}). \quad (13)$$

For the 6 k block example it means that we seek $6 \times 10^3/2 = 3 \times 10^3$ pairs maximizing the cdyn savings. Those can be found ad subsequently described among the 7.2×10^6 useful pairs shown in Fig. 11.

The above problem can be solved by the well-known *Maximal Cost Perfect Matching* (MCPM) algorithms [15]. If the solution of MCPM contains edges $e_{ij} \in E'$ satisfying $w(e_{ij}) = 0$, the gating logic of FF_i and FF_j are not merged and they will stay individually gated. Though the 61 block test bench comprises 200 k FFs, it should be noted that FFs are merged only within their blocks, involving few thousands FFs each.

MCPM was employed in [10] to merge FFs in data-driven clock gating, based on data toggling correlation. Merging FFs for joint gating in LACG is different. It depends on the similarity of the source FFs to target FF connection rather than on toggling vectors correlation as in [10]. MCPM allows solving the general case of merging the gating logic of more than two FFs. This has been studied in [17] for data-driven clock gating. A heuristic

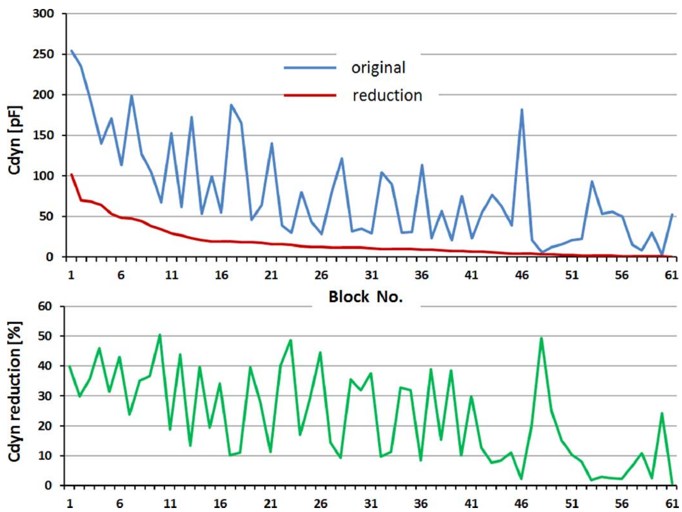

Fig. 12. cdyn reduction in 61 blocks comprising total of 200 k FFs.

solution was proposed, which can be adapted to LACG. It successfully merged FFs in blocks comprising few thousands FFs.

V. Experimental Results

LACG has been experimented on the 61 blocks used in Fig. 1. Those are mostly control blocks of the data-path, register-file and memory management units of the microprocessor. The capacitance parameters are of 22 nm process technology. For the specific test bench used in this work the dynamic to static power split was 80% and 20%, respectively [18]. The relatively low static power is due to a variety of design techniques and process features used by the design.

The gating scheme presented in Fig. 6 was first verified by a formal verification EDA tool and it was found equivalent to the original circuit before the gating logic was introduced. Though not surprising, it is a must in an industrial environment where the method was experimented. It is important to note that the introduction of LACG made most of the gate-level clock gating techniques employed by this design redundant. Those were therefore dropped, which somewhat compensated the LACG power and area overhead.

The cdyn (power) savings for each block is illustrated in Fig. 12. The blocks are decreasingly sorted according to the cdyn reduction achieved. The top graph shows that for block No. 1, whose total cdyn was originally 250 pF, LACG achieved 100 pF savings, which is 40% as shown in the bottom graph. For block No. 2 whose total cdyn was originally 230 pF, LACG achieved 70 pF savings, which is 30% as shown in the bottom graph, etc.. The cumulated savings is presented in Fig. 13. It is shown that the total clock cdyn is 4750 pF, of which 1065 pF has been reduced by the LACG, concluding that the total clock dynamic power reduction is 22.5%.

To assess the impact of the power savings achieved by LACG on the total power dissipation, recall that there is also the dynamic power of the logic and the static (leakage) power that is independent of the switching activity. In the case of the 61 blocks, the clock cdyn is 70% of the total cdyn, and the

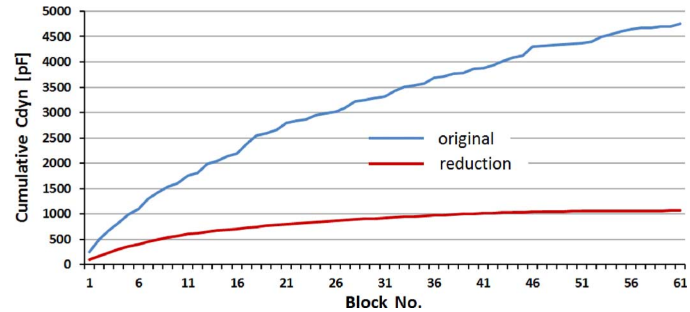

Fig. 13. Cumulative cdyn before gating and the amount of reduction achieved by LACG.

static power is 20% of the total power. Taking those factors into account, the 22.5% clock cdyn reduction was translated into 12.6% reduction of the total power.

VI. Conclusion

Look-ahead clock gating has been shown to be very useful in reducing the clock switching power. The computation of the clock enabling signals one cycle ahead of time avoids the tight timing constraints existing in other gating methods. A closed-form model characterizing the power saving was presented and used in the implementation of the gating logic. The gating logic can be further optimized by matching target FFs for joint gating which may significantly reduce the hardware overheads. While this paper discussed the case of merging two target FFs for joint gating, clustering target FFs in larger groups may yield higher power savings. This is a matter of a further research.

Acknowledgment

The authors would like to thank O. Vaserberger, M. Abozaed, and G. Tamir of Intel Corporation for supporting this work, and E. Aberbach and O. Cohen of Bar-Ilan University for helpful discussions. They are also grateful for the useful comments made by the anonymous reviewers.

References

[1] V. G. Oklobdzija, *Digital System Clocking – High-Performance and Low-Power Aspects*. New York, NY, USA: Wiley, 2003.
[2] L. Benini, A. Bogliolo, and G. De Micheli, "A survey on design techniques for system-level dynamic power management," *IEEE Trans. VLSI Syst.*, vol. 8, no. 3, pp. 299–316, Jun. 2000.
[3] M. S. Hosny and W. Yuejian, "Low power clocking strategies in deep submicron technologies," in *Proc. IEEE Int. Conf. Integr. Circuit Design Technol., ICICDT 2008*, pp. 143–146.
[4] C. Chunhong, K. Changjun, and S. Majid, "Activity-sensitive clock tree construction for low power," in *Proc. ISLPED*, 2002, pp. 279–282.
[5] A. Farrahi, C. Chen, A. Srivastava, G. Tellez, and M. Sarrafzadeh, "Activity-driven clock design," *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 20, no. 6, pp. 705–714, Jun. 2001.
[6] W. Shen, Y. Cai, X. Hong, and J. Hu, "Activity and register placement aware gated clock network design," in *Proc. ISPD*, 2008, pp. 182–189.
[7] Synopsys Design Compiler, Version E-2010.12-SP2.
[8] S. Wimer and I. Koren, "The Optimal fan-out of clock network for power minimization by adaptive gating," *IEEE Trans. VLSI Syst.*, vol. 20, no. 10, pp. 1772–1780, Oct. 2012.
[9] M. Donno, E. Macii, and L. Mazzoni, "Power-aware clock tree planning," in *Proc. ISPD*, 2004, pp. 138–147.
[10] S. Wimer and I. Koren, "Design flow for flip-flop grouping in data-driven clock gating," *IEEE Trans. VLSI Syst.*, to be published.

[11] M. Muller, S. Simon, H. Gryska, A. Wortmann, and S. Buch, "Low power synthesizable register files for processor and IP cores," *INTEGRATION, The VLSI J.*, vol. 39, pp. 131–155, 2006.
[12] A. G. M. Strollo and D. De Caro, "Low power flip-flop with clock gating on master and slave latches," *Electron. Lett.*, vol. 36, no. 4, pp. 294–295, Feb. 2000.
[13] C. E. Stroud, R. R. Munoz, and D. A. Pierce, "Behavioral model synthesis with Cones," *IEEE Design Test Comput.*, vol. 5, no. 3, pp. 22–30, Jun. 1988.
[14] J. A. Bondy and U. S. R. Murty, *Graph Theory*. : Srpinger, 2008.
[15] V. Kolmogorov, "Blossom V: A new implementation of a minimum cost perfect matching algorithm," *Math. Prog. Comp.*, pp. 43–67, 2009.
[16] J. Kathuria, M. Ayoub, M. Khan, and A. Noor, "A review of Clock Gating Techniques," *MIT Int. J. Electron. and Commun. Engin.*, vol. 1, no. 2, pp. 106–114, Aug. 2011.
[17] S. Wimer, "On optimal flip-flop grouping for VLSI power minimization," *Oper. Res. Lett.*, vol. 41, no. 5, pp. 486–489, Sep. 2013.
[18] "A Comparison of Intel's 32 nm and 22 nm Core i5 CPUs: Power, Voltage, Temperature, and Frequency," Oct. 2012 [Online]. Available: http://blog.stuffedcow.net/2012/10/intel32 nm-22 nm-core-i5-comparison/

Shmuel Wimer (M'10) received the B.Sc. and M.Sc. degrees in mathematics from Tel-Aviv University, Tel-Aviv, Israel, and the D.Sc. degree in electrical engineering from the Technion—Israel Institute of Technology, Haifa, Israel, in 1978, 1981, and 1988, respectively.

He worked for 32 years at industry in R&D, engineering and managerial positions, for Intel Corporation from 1999 to 2009, and prior to that for Sagantec, microCAD, IBM, National Semiconductor and Israeli Aircraft Industry. He is presently an Associate Professor with the Engineering Faculty, Bar-Ilan University, Ramat-Gan, Israel, and an Associate Visiting Professor with the Electrical Engineering Faculty, Technion. His current interests include VLSI circuits and systems design optimization and combinatorial optimization.

Albahari Arye received the B.Sc. degrees in electrical engineering from the Technion-Israel Institute of Technology in 2006.

He works for Intel Corporation since 2002 in microprocessors design, where he designed various custom blocks such as register files, data-path and small signals arrays. Arye participated in the foundation of Intel's circuit design team at Bangalore, India, where he served as a technical leader of an owner of mid-level cache design. He is currently leading Intel's cores low-power design and develops the low-power design methodologies of Intel's next generation processors.