Discrete Optimization 14 (2014) 104–110

Contents lists available at ScienceDirect

Discrete Optimization

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

Note

Easy and difficult exact covering problems arising in VLSI power reduction by clock gating

Shmuel Wimer *

Engineering Faculty, Bar-Ilan University, Ramat-Gan 52900, Israel
EE Faculty, Technion - Israel Institute of Technology, Haifa 32000, Israel

ARTICLE INFO

Article history:
Received 16 December 2013
Received in revised form 2 August 2014
Accepted 8 August 2014

Keywords:
Exact covering
Perfect matching
VLSI power minimization
Clock-gating

ABSTRACT

Several graph matching and exact covering problems arising in VLSI low-power design optimization by clock gating are presented. To maximize the power savings, clock gating requires optimal grouping of Flip-Flops (FFs), which depends on FFs' data toggling correlations and probabilities. These naturally lead to optimal matching and exact covering problems. We present three problems arising by different clock-gating techniques. In a method called *data-driven* clock-gating, the corresponding covering problem is intractable but can practically be solved by appropriate heuristics. In another method called *multi-bit* flip-flops, the covering problem is easily solvable in a closed-form, required only sorting. We finally present the covering problem arising in a new method called *look-ahead* clock-gating, for which the question of whether the exact covering problem is easy or difficult is left open.

1. VLSI clock-gating and covering problems

The clock network together with its underlying *Flip-Flops* (FFs), is typically responsible for 30%–70% of the total power consumed by modern *Very Large Scale Integration* (VLSI) digital systems, and is thus a primary candidate for power reduction [1,2]. Clock network power is consumed due to the toggling (switching) of the clock signal (pulse). A variety of techniques exist to reduce the clock power, of which *clock-gating* is a predominant. It disables the clock signal when the underlying driven circuits are not subject to change (toggle) their state, and hence do not need the clock.

Data-Driven Clock-Gating (DDCG) techniques have been shown to be very effective and saving up to 20% of the total chip power [3]. DDCG disables the clock pulse driving the system's FFs [4,5] if those will not change their state (data) in the next clock cycle. It requires clustering the system's n FFs in groups of k FFs each, sharing a common clock signal. The amount of power being saved by DDCG depends on the toggling probabilities and correlation of the FFs comprising a group. Obviously, the interest is that the joint clock signal driving a FFs group will be disabled as much as possible. In typical VLSI designs the size of n may vary from a few thousands to a few hundreds of thousands, while k may vary from two to a few dozens.

DDCG implies a corresponding *Min-Cost Exact Covering Problem* (MCECP) where an n-size set should exactly be covered by n/k k-size subsets G_i, $1 \le i \le \binom{n}{k}$, with cost w_i reflecting the subset's power consumption. The cost w_i in [5] was based on FFs toggling correlation (elaborated in Section 2), whereas finding the minimum power grouping has been shown in [6] to be NP-hard, and appropriate heuristics have been proposed for practical solution. The FFs grouping described in [7] was based on FFs toggling probabilities, rather than correlation. The implied MCECP has been shown to be well-solvable,

* Correspondence to: Engineering Faculty, Bar-Ilan University, Ramat-Gan 52900, Israel. Tel.: +972 35317208; fax: +972 37384051.
E-mail addresses: wimers@biu.ac.il, wimer@ee.technion.ac.il.

http://dx.doi.org/10.1016/j.disopt.2014.08.004
1572-5286/

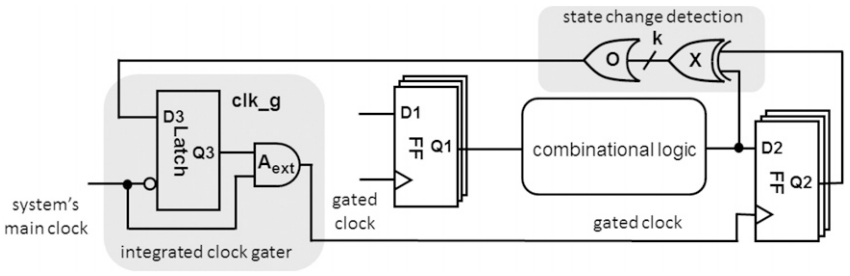

Fig. 1. Data-driven clock circuit. Overhead hardware is shaded in gray.

having a closed-form solution which requires only sorting (elaborated in Section 3). Another clock-gating method called *Look-Ahead Clock-Gating* (LACG) has been proposed in [8]. There, the FFs groups are uniquely determined by the system's logic. MCECP arises, since in LACG the grouping of FFs group is of interest (elaborated in Section 4). The clock gating methods discussed in this paper with their underlying matching and covering algorithms have lately been implemented in industrial environments and are currently used by companies such as Intel [8], Ceva and Mellanox [5].

2. Covering problems implied by FFs toggling correlation

DDCG disables the clock signal driving a FF when the FF's state is not subject to change in the next clock cycle. A logic system comprising DDCG is illustrated in Fig. 1, where its hardware overhead to generate the clock disabling signal is shaded in gray. A XOR gate checks whether a FF's state is subject to change, thus finding out whether its clock can be disabled in the next cycle. k XOR gates are ORed and latched to generate a joint gating signal for the k FFs. There is a tradeoff between the number of saved (disabled) clock pulses and the hardware overhead, and the optimal k minimizing the power consumption was derived in [4].

The problem of which FFs should be placed in a group so as to minimize the power, and how to derive those groups, was studied in [6]. Let n FFs be clocked during $m+1$ cycles, and $\mathbf{a} = (a_1, \ldots, a_m)$ be the activity of a FF. An entry $a_t = 0$, $1 \le t \le m$, if the FF stays unchanged (no toggling) from time $t-1$ to time t, and $a_t = 1$ otherwise. The term $\|\mathbf{a}\| = \sum_{t=1}^{m} a_t$ is proportional to the power consumed by the FF's switching. All the $n(n-1)/2$ pairs $(\mathbf{a}_i, \mathbf{a}_j)$, $1 \le i < j \le n$, are bit-wise XORed to yield the number $\|\mathbf{a}_i \oplus \mathbf{a}_j\|$ of *redundant clock pulses* occurring if FF_i and FF_j are grouped and share a common gater. The smaller it is, the more desirable it is to jointly clock FF_i and FF_j.

To model the switching power consumed when driving FFs pairs ($k = 2$) with a common clock gater, an n-vertex complete weighted graph $G(V, E, w)$ is defined. Assume w.l.o.g that n is even (we could otherwise add a never toggling artificial FF and set to zero the weight of its entire incident edges). A vertex $v_i \in V$ is associated with FF_i's activity $\mathbf{a}_i$. An edge $e_{ij} = (v_i, v_j) \in E$ is associated with a joint activity vector $\mathbf{a}_i \mid \mathbf{a}_j$, where the OR is a bit-wise operation. An edge e_{ij} is assigned a weight $w(e_{ij}) = \|\mathbf{a}_i \oplus \mathbf{a}_j\|$, counting the number of redundant clock pulses incurred by clocking FF_i and FF_j with a common gater. Let $E' \subset E$, $|E'| = |V|/2$, be a vertex matching of $G(V, E, w)$. The total power consumed by the gated clock signal is proportional to the number P of pulses driving the underlying FFs, given by

$$P = 2\sum_{e_{ij}\in E'} \|\mathbf{a}_i \mid \mathbf{a}_j\| = \sum_{v_i\in V} \|\mathbf{a}_i\| + \sum_{e_{ij}\in E'} \|\mathbf{a}_i \oplus \mathbf{a}_j\| = \underbrace{\sum_{v_i\in V} \|\mathbf{a}_i\|}_{\text{(a)}} + \underbrace{\sum_{e_{ij}\in E'} w(e_{ij})}_{\text{(b)}}. \tag{1}$$

The term (a) of (1) is an essential component charged to the toggling of the individual FFs, independently of the pairing. Therefore, to consume minimum switching power it is necessary to minimize term (b), which turns into the well-known *Minimal Cost Perfect graph Matching* (MCPM) problem [9], for which polynomial complexity algorithms are known [10].

The extension for $k > 2$ is straightforward. Assume w.l.o.g that n is divisible by k. (We could otherwise add never toggling artificial FFs.) A complete k-uniform weighted hypergraph $H(V, E, w)$ is defined, where for a subset $\mathbf{v} \subset V$ and $|\mathbf{v}| = k$, $e_{\mathbf{v}} = \{v_u\}_{u\in\mathbf{v}} \in E$ defines a hyper edge. It follows that $|E| = \binom{n}{k}$. A hyper edge $e_{\mathbf{v}}$ is associated with a joint activity vector $\bigcup_{u\in\mathbf{v}} \mathbf{a}_u$, defined by the bit-wise ORing of the k toggling vectors. A hyper edge $e_{\mathbf{v}}$ is assigned the weight (2), which is the total number of redundant clock pulses incurred by clocking the k FFs corresponding to $e_{\mathbf{v}}$ with a common gater.

$$w(e_{\mathbf{v}}) = \sum_{v\in\mathbf{v}} \left\| \mathbf{a}_v \oplus \bigcup_{u\in\mathbf{v}} \mathbf{a}_u \right\|. \tag{2}$$

Let $E' \subset E$ be an exact cover of the vertices of $H(V, E, w)$ by n/k hyper edges (a vertex belongs to one and only one hyper edge). The total power consumed by the clock signal is proportional to the total number P of pulses driving the FFs, given in (3). The term (a) is an essential power component charged to the toggling of the individual FFs, and is independent of the

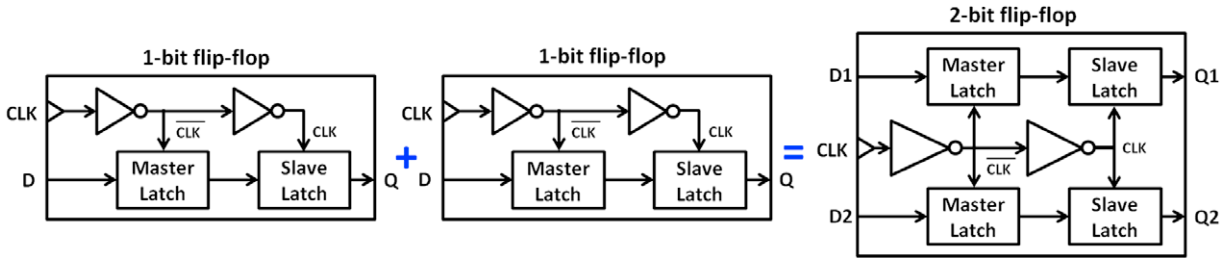

Fig. 2. Single-bit FF and 2-bit MBFF.

grouping.

$$P = \sum_{e_{\mathbf{v}} \in E'} k \left\| \bigcup_{u \in \mathbf{v}} \mathbf{a}_u \right\| = \sum_{v_i \in V} \|\mathbf{a}_i\| + \sum_{e_{\mathbf{v}} \in E'} \sum_{v \in \mathbf{v}} \left\| \mathbf{a}_v \oplus \bigcup_{u \in \mathbf{v}} \mathbf{a}_u \right\| = \underbrace{\sum_{v_i \in V} \|\mathbf{a}_i\|}_{\text{(a)}} + \underbrace{\sum_{e_{\mathbf{v}} \in E'} w\left(e_{\mathbf{v}}\right)}_{\text{(b)}} . \tag{3}$$

To consume minimum switching power it is necessary to minimize term (b), a problem shown in [6] to be NP-hard. With the above formulation, a solution of the problem can be obtained by solving the well-known NP-hard weighted *Set Partitioning Problem* (SPP) [11]. There, hyper edges are the variables covering the vertex constraints. The practical heuristic solution of the SPP is using iterations of MCPM and considerations of the physical proximity constraints of related FFs. Cases where n is not divisible by k and other implementation details can be found in [5,6]. Iterative application of MCPM for block-level clock gating was proposed in [12].

3. Covering problems implied by FFs toggling probabilities

While the above grouping relied on toggling correlations measured by XORing, we subsequently describe a grouping based on toggling probabilities. Shown in Fig. 2, an ordinary 1-bit FF is composed of two cascaded master and slave latches, driven by a clock CLK. Most of the energy within a FF is consumed by its internal clock driver. Data is usually stored in individual FFs, each having its own internal clock driver. In an attempt to reduce the clock power, several FFs are grouped into a module such that a common clock driver is used for all. A technique called *Multi-Bit Flip-Flop* (MBFF) is lately being adopted by the VLSI industry [13,14]. A grouping of two individual FFs into 2-bit MBFF is shown in Fig. 2. Notice that the clock driver in the MBFF is common to both FFs.

Groups of k FFs are called k-bit MBFF. The power savings of a MBFF depend on the amount of the average (expected) data toggling probability p of its individual FFs, called *data toggling probability*, *switching probability*, or shortly probability. We use those terms interchangeably. By definition, there is $0 \leq p \leq 1$, where $p = 0$ when the data is never toggling and $p = 1$ when the data is toggling at every clock cycle. In typical VLSI systems $0 < p \leq 0.2$. Simulations of 2-bit MBFF ($k = 2$) in [7] showed energy reduction of 35% for $p = 0.05$, and 15% for $p = 0.95$. Simulations for 4-bit MBFF ($k = 4$) showed higher savings of 55% for $p = 0.05$ and 23% for $p = 0.95$ [7].

Application of DDCG to MBFF was proposed in [14]. It took advantage of the FFs that are anyway grouped in MBFF, to derive a joint clock enabling signal with a relatively small hardware overhead. To simplify the FF grouping that minimizes the energy, it was proposed in [7] to consider FF toggling probabilities rather than toggling correlation, as in [5,6]. The implied optimization problem is a kind of k-MCECP, where for $k = 2$ it turns into a MCPM problem. It has been proven in [7] that for power cost model based on toggling probabilities rather than on toggling correlations, the implied MCECP is well-solvable, requires only sorting.

Denote by E_1 the energy consumed by an ordinary 1-bit FF. Shown by simulations, E_1 grows with the FF's toggling probability p as follows:

$$E_1(p) = \alpha_1 + \beta_1 p. \tag{4}$$

The parameter α_1 is the energy of the FF's internal clock driver, and the parameter β_1 is the energy of data toggling. For 2-bit MBFF there are three possible scenarios: none of the FFs toggle, a single FF toggles, and both FFs toggle. Assuming data toggling independence, the energy consumption E_2 is

$$E_2(p) = \alpha_2 (1-p)^2 + 2(\alpha_2 + \beta_2) p (1-p) + (\alpha_2 + 2\beta_2) p^2 \equiv \alpha_2 + 2\beta_2 p. \tag{5}$$

The parameter α_2 is the energy of the internal clock driver which drives the two FFs, and the parameter β_2 is the per-bit data toggling energy of the 2-bit MBFF.

For the general case of k-bit MBFF, let α_k be the energy of the MBFF's internal clock driver driving the k FFs, and let β_k be the per-bit data toggling energy in the k-bit MBFF. Considering all the combinations of toggling FFs, the energy

consumption E_k is

$$E_k(p) = \sum_{j=0}^{k} (\alpha_k + j\beta_k) \binom{k}{j} p^j (1-p)^{k-j}. \tag{6}$$

Rearrangement of (6) and showing by induction that $\sum_{j=0}^{k} \binom{k}{j} jp^j (1-p)^{k-j} = kp$ yields

$$E_k(p) = \alpha_k \sum_{j=0}^{k} \binom{k}{j} p^j (1-p)^{k-j} + \beta_k \sum_{j=0}^{k} \binom{k}{j} jp^j (1-p)^{k-j} = \alpha_k + k\beta_k p. \tag{7}$$

It was shown in [6] that the energy savings factor obtained from circuit simulations matches the ratio $kE_1(p)/E_k(p)$ for $k \geq 2$. Obviously, the lower the p, the higher the saving will be.

Let FF_i and FF_j be two FFs toggling independently of each other with probabilities p_i and p_j, respectively. We denote by $\mathrm{FF}_{(i,j)}$ their grouping (pairing) in the formation of a 2-bit MBFF. Similar to (5), the energy $E_{(i,j)}$ consumed by $\mathrm{FF}_{(i,j)}$ is $E_{(i,j)} = \alpha_2 + \beta_2 (p_i + p_j)$. Given four FFs FF_i, FF_j, FF_k and FF_l, paired in two 2-bit MBFFs, their energy consumption is $E_{(i,j)} + E_{(k,j)} = 2\alpha_2 + \beta_2 (p_i + p_j + p_k + p_l)$, which is independent of the pairing.

Once clock-gating is applied to MBFFs, pairing considerably matters for energy consumption. Since in MBFF the clock signal is common, when none of FF_i and FF_j changes its data, the clock pulse of $\mathrm{FF}_{(i,j)}$ is disabled and its internal clock driver does not waste any switching energy. When both FF_i and FF_j change their data, the clock pulse of $\mathrm{FF}_{(i,j)}$ is enabled and the switching energy of the internal clock driver is fully useful, hence is no energy waste. A waste happens when one FF is toggling, while its counterpart does not. There, the common clock pulse is enabled, driving both FFs, whereas only one needs it, thus causing a waste $W_{(i,j)}$ of half of the internal clock driver energy, given by

$$W_{(i,j)} = \frac{\alpha_2}{2} \left[p_j (1-p_i) + p_i (1-p_j) \right] = \frac{\alpha_2}{2} (p_i + p_j - 2p_i p_j). \tag{8}$$

We are interested in minimizing $W_{(i,j)}$. Let us pair four FFs in 2-bit clock-gated MBFFs. The following energy waste results in

$$W_{(i,j)} + W_{(k,l)} = \frac{\alpha_2}{2} \left[\underbrace{p_i + p_j + p_k + p_l}_{\text{(a)}} - 2 \underbrace{(p_i p_j + p_k p_l)}_{\text{(b)}} \right]. \tag{9}$$

While the term (a) of (9) is not dependent on the pairing, the term (b) is. $W_{(i,j)} + W_{(k,l)}$ is minimized when (b) is maximized. If $p_i \leq p_j \leq p_k \leq p_l$, the pairing $\{\mathrm{FF}_{(i,j)}, \mathrm{FF}_{(k,l)}\}$ is favored over $\{\mathrm{FF}_{(i,k)}, \mathrm{FF}_{(j,l)}\}$ optimal since $(W_{(i,j)} + W_{(k,l)}) - (W_{(i,k)} + W_{(j,l)}) = -\alpha_2 (p_i - p_l)(p_j - p_k)/2 \leq 0$.

The above observation hints that optimal 2-bit MBFF grouping prefers pairs of FFs whose switching probabilities are close to each other. The generalization for pairing of n FFs is subsequently discussed. Let n be even and $\mathtt{P} : \{\mathrm{FF}_{(s_i,t_i)}\}_{i=1}^{n/2}$ be a pairing of $\mathrm{FF}_1, \mathrm{FF}_2, \ldots, \mathrm{FF}_n$ in $n/2$ 2-bit clock-gated MBFFs. The following energy waste $\mathtt{W}(\mathtt{P})$ results in

$$\mathtt{W}(\mathtt{P}) = \sum_{i=1}^{n/2} W_{(s_i,t_i)} = \frac{\alpha_2}{2} \left[\sum_{j=1}^{n} p_j - 2 \sum_{i=1}^{n/2} p_{s_i} p_{t_i} \right]. \tag{10}$$

$\mathtt{W}(\mathtt{P})$ is minimized when $\sum_{i=1}^{n/2} p_{s_i} p_{t_i}$ is maximized. The optimal pairing can be found in polynomial time by applying a MCPM [10] to the n-vertex complete weighted graph, the vertices of which are FF_i and its edge weights are $p_i p_j$, $1 \leq i < j \leq n$. A closed-form simpler solution of $O(n \log n)$ time complexity has been obtained in [7], as follows.

Theorem 1. *Let n be even and $\mathrm{FF}_1, \mathrm{FF}_2, \ldots, \mathrm{FF}_n$ satisfy $p_1 \leq p_2 \leq \cdots \leq p_n$. The pairing $\mathtt{P} : \{\mathrm{FF}_{(2i-1,2i)}\}_{i=1}^{n/2}$ is minimizing $\mathtt{W}(\mathtt{P})$ given in (10).*

In case of odd n it is possible to exclude either FF_1 or FF_n and obtain solutions for the corresponding problems of $n-1$ FFs, from which the better one is chosen.

The hardware overhead involved in clock-gating may sometimes make its application questionable for groups comprising two FFs. It has been shown in [4] that maximal power savings is obtained for three and more FFs, depending on their toggling probabilities. Let $\mathrm{FF}_{(i_1,\ldots,i_k)}$ denote a k-bit MBFF comprising $\mathrm{FF}_{i_1}, \ldots, \mathrm{FF}_{i_k}$. Consider the energy waste incurred by $\mathrm{FF}_{(i_1,\ldots,i_k)}$. When none of the k FFs are toggling, their gater disables the clock pulse, so energy is not wasted. When all are toggling, the clock pulse is required by all, hence no waste occurs. A waste occurs when $1 \leq q \leq k-1$ of the FFs are toggling, while $r = k - q$ are not. Since the clock pulse drives r non-toggling FFs, it results in energy waste of $\alpha_k r/k$, multiplied by the probability of that event. There are $\binom{k}{r}$ distinct events. For each $1 \leq m \leq \binom{k}{r}$ we split $\mathrm{FF}_{i_1}, \ldots, \mathrm{FF}_{i_k}$ into the sets of the toggling and non-toggling FFs, denoted by A_m and B_m, respectively, so $|A_m| = q$ and $|B_m| = r$. The corresponding energy

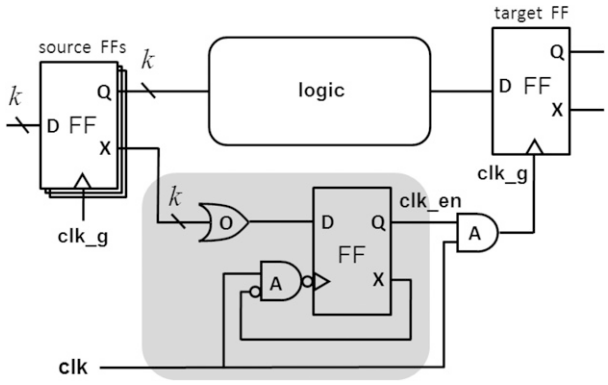

Fig. 3. Look ahead clock gating (LACG).

waste is

$$W_{(i_1,\ldots,i_k)} = \sum_{r=1}^{k-1} \alpha_k \frac{r}{k} \sum_{m=1}^{\binom{k}{r}} \prod_{s\in A_m} p_s \prod_{t\in B_m} (1-p_t)\,. \tag{11}$$

Assume that n is divisible by k (non-divisible n is discussed later). It has been shown in [7] that the MBFF k-bit grouping of the n FFs that minimizes the energy waste can be found in $O(n \log n)$ time complexity by sorting. Let $\mathtt{P} : \left\{\mathrm{FF}_{(s_{k(i-1)+1},\ldots,s_{ki})}\right\}_{i=1}^{n/k}$ be a grouping of $\mathrm{FF}_1, \mathrm{FF}_2, \ldots, \mathrm{FF}_n$ in n/k k-bit clock-gated MBFFs, and let $W_{(s_{k(i-1)+1},\ldots,s_{ki})}$ be the energy waste of $\mathrm{FF}_{(s_{k(i-1)+1},\ldots,s_{ki})}$, as given in (11). The following energy waste $\mathtt{W}(\mathtt{P})$ results in

$$\mathtt{W}(\mathtt{P}) = \sum_{i=1}^{n/k} W_{(s_{k(i-1)+1},\ldots,s_{ki})}. \tag{12}$$

The optimal grouping implies a k-MCECP that is NP-hard in general. In the setting of our problem though, where the costs of the groups are determined by sum of probabilities products, it is well-solvable [7].

Theorem 2. *Let n be divisible by k and let $\mathrm{FF}_1, \mathrm{FF}_2, \ldots, \mathrm{FF}_n$ be ordered such that $p_1 \le p_2 \le \cdots \le p_n$. The grouping $\mathtt{P} : \left\{\mathrm{FF}_{(k(i-1)+1,\ldots,ki)}\right\}_{i=1}^{n/k}$ is minimizing $\mathtt{W}(\mathtt{P})$ given in (12).*

If n is not divisible by k, let $r = n \bmod k$, $0 \le u \le r$, $0 \le v \le r$ and $u + v = r$. We ignore the first u FFs with the smallest probabilities and the last v FFs with the largest probabilities. It results in $r+1$ instances of grouping $n-r$ FFs in $(n-r)/k$ k-bit MBFFs, plus another two MBFFs, one of u bits and another of v bits. From those $r+1$ instances, the one yielding the smallest energy is optimal. It is important to note that the above analysis assumed FFs toggling independency, which is a worst-case assumption, while real FFs toggling may be correlated. This however is a fortunate situation, where the real power being saved in the hardware can only be higher than anticipated by the worst-case modeling.

4. Covering problems implied by look-ahead clock-gating

Look-ahead clock-gating (LACG) has lately been proposed in [8]. It is addressing two goals: making gating applicable for large and general designs, and avoiding the tight timing constraints of DDCG. Similar to DDCG, LACG is based on XORing the FF's output and input as in Fig. 1. However, while in DDCG it is used to disable the clock pulse of that FF in the next clock cycle, LACG uses it to generate clock disabling signals one cycle ahead of time for other FFs in the system, whose data depend on that FF. This has considerable timing advantage over DDCG. Another fundamental difference of LACG from DDCG and MBFF is that in LACG the grouping of FFs to produce the clock enable signal is uniquely defined by the underlying logic, independently of FFs toggling probabilities and correlations.

Fig. 3 illustrates how LACG works, where its hardware overhead is shaded in gray, and the XOR output of a FF is denoted by X. A target FF depends on $k \ge 1$ source FFs. The k FFs' XOR outputs are ORed and generate the enabling signal. To ensure its validity one cycle ahead, an oppositely clocked FF is introduced. [8] studied the optimal tradeoff between the energy savings and gating overhead.

To minimize the overheads, target FFs share gating circuits. Fig. 4(a) shows two target FFs FF_i and FF_j, toggling independently of each other, with their corresponding OR trees, driven by k_i and k_j source FFs, respectively. The outputs of the OR trees are the clock enabling signals $\mathbf{e}_i(t)$ and $\mathbf{e}_j(t)$, where t denotes the clock ticks. FF_i and FF_j may share common

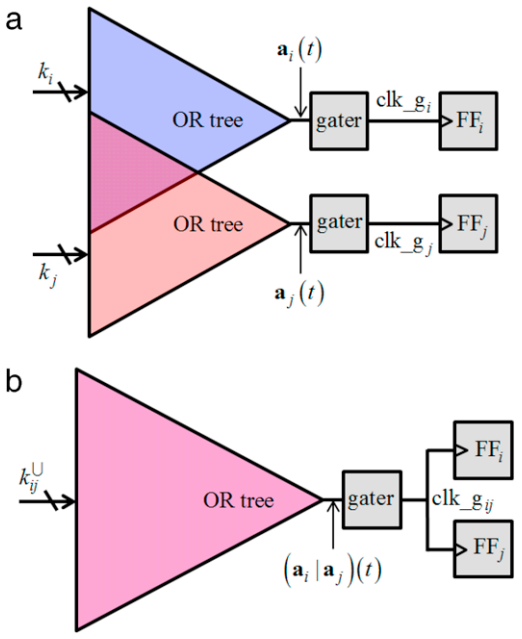

Fig. 4. Merging OR logic for joint gating.

source FFs, shown pictorially by their trees overlap. An alternative gating implementation is shown in Fig. 4(b) were the OR logic is merged and a single gater is used for the two FFs. The larger the overlap, and the more $\mathbf{e}_i$ and $\mathbf{e}_j$ are correlated, the more desirable the merge will be.

The dynamic power consumption, denoted as c_{dyn}, is obtained by the product of capacitance with signal switching probability. Two factors affect the clock-gating logic merging. The first is the similarity of the OR logic used in different target FFs. It indicates how much capacitance will be saved by the merging. The other is the correlation of the clock enabling signals, produced by the OR logic of each target FFs. c_{dyn} is obtained by their product. Let n target FFs be clocked during $m+1$ cycles and $\mathbf{a} = (a_1, \ldots, a_m)$ be an enabling signal produced by an OR tree. An element a_t, $1 \le t \le m$, corresponds to clock ticks, where $a_t = 1$ if the clock is enabled and $a_t = 0$ otherwise. The term $\|\mathbf{a}\| = \sum_{t=1}^{m} a_t$ counts the cycles of an enabled clock.

OR logic merging reduces hardware overhead, but does not come for free. The number of redundant clock pulses is increased since the clock gater is driven by a wider tree comprising more source FFs, thus increasing the clock enabling probability. To tradeoff the gain and loss, let $\mathbf{S}(\text{FF}_i)$ and $\mathbf{S}(\text{FF}_j)$ denote the source FFs of FF_i and FF_j, whose size is $k_i = |\mathbf{S}(\text{FF}_i)|$ and $k_j = |\mathbf{S}(\text{FF}_j)|$, respectively. A reasonable power estimation of the corresponding OR trees is $P_i = \|\mathbf{a}_i\| k_i$ and $P_j = \|\mathbf{a}_j\| k_j$, where $0 \le \|\mathbf{a}_i\|, \|\mathbf{a}_j\| \le m$ count the cycles of enabled clock. The power consumed by the gating is thus

$$P_i + P_j = \|\mathbf{a}_i\| k_i + \|\mathbf{a}_j\| k_j. \tag{13}$$

The number of inputs of the merged OR logic is $k_{ij}^{\cup} = |\mathbf{S}(\text{FF}_i) \cup \mathbf{S}(\text{FF}_j)|$. Shown in Fig. 4(b), the merging eliminates one gater. The amount $k_{ij}^{\cap} = |\mathbf{S}(\text{FF}_i) \cap \mathbf{S}(\text{FF}_j)|$ indicates how much of the OR logic is shared by the merge, and thus being saved. The power P_{ij} consumed by the merged OR trees in Fig. 4(b) is proportional to

$$P_{ij} = \|\mathbf{a}_i \mid \mathbf{a}_j\| k_{ij}^{\cup} = \left(\|\mathbf{a}_i\| + \|\mathbf{a}_j\| - \|\mathbf{a}_i \cap \mathbf{a}_j\|\right) \times \left(k_i + k_j - k_{ij}^{\cap}\right), \tag{14}$$

where the OR $\mathbf{a}_i \mid \mathbf{a}_j$ and the AND $\mathbf{a}_i \cap \mathbf{a}_j$ are bit-wise operations. Subtracting (14) from (13), the savings $c_{\text{dyn}\,ij}^{\text{save}}$ results in by merging the clock-gating logic of FF_i and FF_j is

$$c_{\text{dyn}\,ij}^{\text{save}} = \left[\left(\|\mathbf{a}_i\| + \|\mathbf{a}_j\|\right) k_{ij}^{\cap} + \|\mathbf{a}_i \cap \mathbf{a}_j\| \left(k_i + k_j\right)\right] - \left[\|\mathbf{a}_i\| k_j + \|\mathbf{a}_j\| k_i + \|\mathbf{a}_i \cap \mathbf{a}_j\| k_{ij}^{\cap}\right]. \tag{15}$$

Maximal savings occurs when $\mathbf{S}(\text{FF}_i) = \mathbf{S}(\text{FF}_j)$, a case where $k_i = k_j = k_{ij}^{\cap}$ and $\mathbf{a}_i = \mathbf{a}_j$, yielding by substitution in (15) power savings of a complete OR tree. Oppositely, if $\mathbf{S}(\text{FF}_i) \cap \mathbf{S}(\text{FF}_j) = \varnothing$, and thus $k_{ij}^{\cap} = 0$, substitution in (15) yields $c_{\text{dyn}\,ij}^{\text{save}} = -\left[\left(\|\mathbf{a}_i\| k_j + \|\mathbf{a}_j\| k_i\right) - \|\mathbf{a}_i \cap \mathbf{a}_j\| \left(k_i + k_j\right)\right]$. In the best case when $\mathbf{a}_i = \mathbf{a}_j$ there is $c_{\text{dyn}\,ij}^{\text{save}} = 0$. In the worst case when $\mathbf{a}_i \cap \mathbf{a}_j = \mathbf{0}$, maximal waste $c_{\text{dyn}\,ij}^{\text{save}} = -\left(\|\mathbf{a}_i\| k_j + \|\mathbf{a}_j\| k_i\right)$ occurs.

To maximize the total of $c_{\text{dyn}\,ij}^{\text{save}}$ in (15) over all the OR trees possible pairing, an n-vertex complete weighted graph $G(V, E, w)$ is defined. A vertex $v_i \in V$ corresponds to FF_i and an edge $e_{ij} = (v_i, v_j) \in E$ corresponds to a $(\text{FF}_i, \text{FF}_j)$ pair.

An edge e_{ij} is assigned a weight $w\left(e_{ij}\right)$ as follows:

$$w\left(e_{ij}\right) = \max\left\{c^{\text{save}}_{\text{dyn}\,ij}, 0\right\}. \tag{16}$$

The reason for the zero in (16) is to avoid a merge yielding loss rather than gain. In this setup we look for a *Perfect Graph Matching* [9,15], $E' \subset E$, $\left|E'\right| = |V|/2$, of $G(V, E, w)$, maximizing $\sum_{e_{ij}\in E'} w\left(e_{ij}\right)$. This problem can be solved by the well-known *Maximal Cost Perfect Matching* algorithms [10]. If the solution contains edges $e_{ij} \in E'$ satisfying $w\left(e_{ij}\right) = 0$, the OR trees of FF_i and FF_j are not merged, leaving the clock of those be individually gated.

Merging more than two OR trees can be carried out by modeling the logic sharing with a complete hypergraph as for DDCG in Section 2, where for a subset $\mathbf{v} \subset V$, $e_{\mathbf{v}} = \{v_u\}_{u\in\mathbf{v}}$ defines a hyper edge, and $\mathbf{a}_{\mathbf{v}} = \bigcup_{u\in\mathbf{v}} \mathbf{a}_u$ is the joint toggling occurring by merging the OR trees implied by the FFs corresponding to $\mathbf{v}$. The number of inputs of a merged OR tree corresponding to a hyper edge is $k^{\cup}_{\mathbf{v}} = \left|\bigcup_{u\in\mathbf{v}} \mathbf{S}\left(\text{FF}_u\right)\right|$. The product $\mathbf{a}_{\mathbf{v}}k^{\cup}_{\mathbf{v}}$ is the weight associated with a hyper edge, reflecting the dynamic power c_{dyn} consumed by the merged OR tree. The question of whether the problem of optimally merging more than two OR trees for common clock-gating is intractable is not answered yet. It can meanwhile be solved heuristically as proposed in [6] for DDCG.

Acknowledgments

The author is grateful for the useful comments made by the anonymous referees.

References

[1] V.G. Oklobdzija, Digital System Clocking—High-Performance and Low-Power Aspects, John Wiley, 2003.
[2] L. Benini, A. Bogliolo, G. De Micheli, A survey on design techniques for system-level dynamic power management, IEEE Trans. VLSI Syst. 8 (3) (2000) 299–316.
[3] M. Donno, E. Macii, L. Mazzoni, Power-aware clock tree planning, in: Proc. ISPD, 2004, pp. 138–147.
[4] S. Wimer, I. Koren, The Optimal fan-out of clock network for power minimization by adaptive gating, IEEE Trans. VLSI Syst. 20 (10) (2012) 1772–1780.
[5] S. Wimer, I. Koren, Design flow for flip-flop grouping in data-driven clock gating, IEEE Trans. VLSI Syst. 22 (4) (2014) 771–778.
[6] S. Wimer, On optimal flip-flop grouping for VLSI power minimization, Oper. Res. Lett. 41 (2013) 486–489.
[7] S. Wimer, D. Gluzer, U. Wimer, Using well-solvable minimum cost exact covering problems for VLSI energy minimization, Oper. Res. Lett. 42 (2014) 332–336.
[8] S. Wimer, A. Albahari, A look-ahead clock gating based on auto-gated flip-flops, IEEE Trans. Circuits Syst. I 61 (5) (2014) 1465–1472.
[9] J.A. Bondy, U.S.R. Murty, Graph Theory, Srpinger, 2008.
[10] V. Kolmogorov, Blossom V: a new implementation of a minimum cost perfect matching algorithm, Math. Program. Comput. 1 (1) (2009) 43–67.
[11] E. Balas, M.W. Padberg, Set partitioning: a survey, SIAM Rev. (1976) 710–760.
[12] A. Farrahi, C. Chen, A. Srivastava, G. Tellez, M. Sarrafzadeh, Activity-driven clock design, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 20 (6) (2001) 705–714.
[13] I.H.-R. Jiang, C.-L. Chang, Y.-M. Yang, Tsai E.Y.-W., L.S.-F. Chen, INTEGRA: fast multi-bit flip-flop clustering for clock power saving based on interval graphs, in: ACM Proceedings of the 2011 International Symposium on Physical Design, ISPD 2011, pp. 115–122.
[14] C.-L. Chang, I.H-R. Jiang, Pulsed-latch replacement using concurrent time borrowing and clock gating, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 32 (2) (2013) 242–246.
[15] J. Edmonds, Paths, trees, and flowers, Canad. J. Math. 17 (3) (1965) 449–467.