Probability-Driven Multi Bit Flip-Flop Design Optimization

*Doron Gluzer and Shmuel Wimer*

Feb. 9, 2015

**CE Tech Report # 003**

Available at http://www.eng.biu.ac.il/segalla/computer-engineering-tech-reports/

Based on Bar Ilan MSc. Thesis of D. Gluzer, supervised by S. Wimer

Faculty of Engineering

**BAR ILAN UNIVERSITY, Israel**
Abstract
Data-Driven Clock-Gating (DDCG) and Multi Bit Flip-Flops (MBFFs) in which several FFs are grouped and share a common clock driver are two effective low-power design techniques. Though commonly used by VLSI designers, those are usually separately treated. Past works focused on MBFF usage in RTL, gate-level and their layout. Though collectively covering the common design stages, the study of each aspect individually led to conflicts and contradiction with the others. MBFF internal circuit design, its multiplicity and its synergy to the FFs data toggling probabilities have not been studied so far. This work attempts to maximize the energy savings by proposing a DDCG and MBFF combined algorithm, based on Flip-Flops (FFs) data-to-clock toggling ratio. It is shown that to maximize the power savings, the FFs should be grouped in MBFFs in increasing order of their activities. A power savings model utilizing MBFF multiplicities and FF toggling probabilities is developed, which was then used by the algorithm in a practical design flow. We achieved 17% to 23% power savings compared to designs with ordinary FFs.

1. Introduction
A recently published paper has emphasized the usage of Multi-Bit Flip-Flops (MBFFs) as a design technique delivering considerable power reduction of digital systems [1]. The data of digital systems is usually stored in Flip-Flops (FFs), each having its own internal clock driver. Shown in Fig. 1, an edge-triggered 1-bit FF contains two cascaded master and slave latches, driven by opposite clocks CLK and \( \overline{CLK} \). It is shown in Section 2 that most of the FF’s energy is consumed by its internal clock drivers, which are significant contributors to the total power consumption.

In an attempt to reduce the clock power, several FFs can be grouped in a module such that common clock drivers are shared for all the FFs. Two 1-bit FFs grouped into 2-bit MBFF, called also dual-bit FF [1], is shown in Fig. 1. In a similar manner, grouping of FFs in 4-bit and 8-bit MBFFs are possible too. We subsequently denote a \( k \)-bit MBFF by \( k \)-MBFF. MBFF is not only reducing the gate capacitance driven by a clock tree. The wiring capacitive load is also reduced because only a single clock wire is required for multiple FFs. It also reduces the depth and the buffer sizes of the clock tree and also the number of sub-trees. Beyond clock power savings those features also reduce the silicon area.

![Fig. 1. 1-bit FF and 2-MBFF.](image-url)
MBFFs benefits do not come for free. By sharing common drivers, the slopes of the clock signals become slower, causing larger short-circuit current and clock-to-Q propagation delay ($t_{pCQ}$) degradation. Reported in [1], for a design implemented in a 90 nanometer, low-power, high voltage threshold (HVT) CMOS technology, the 4-MBFFs exhibit a per-bit 30% reduction of dynamic clock power, and a per-bit 10% area reduction. That came on the expense of a per-bit 20% data power increase and also 20% degradation of $t_{pCQ}$. However, due to the fact that the average data-to-clock toggling ratio of a FF is very small, varying from 0.01 to 0.1 in most designs [2], the clock power savings always outweighs the short-circuit power penalty on the data toggling. This work answers two questions; what should be the optimal bit multiplicity of MBFFs, and how to leverage from the knowledge of the average data-to-clock toggling ratio (called also activity and data toggling probability) of the FFs in the underlying design.

To remedy the short-circuit power penalty and $t_{pCQ}$ degradation due to the increase of the loads, the MBFF internal drivers can be somewhat strengthen. This is shown pictorially in Fig. 1 by the larger 2-MBFF drivers compared to 1-bit. The MBFF multiplicity $k$ depends on the data toggling probability $p$. Section 2 studies that dependency in an attempt to optimize the MBFF design flow and maximize the power savings. To our best knowledge, that has not been studied so far.

Electronics Design Automation (EDA) tools, such as Cadence Liberate, support MBFF characterization. MBFF gate-level design is possible with the latest Cadence and Synopsys HDL compilers. Their logic-level internal considerations and algorithms of FF grouping into MBFFs have not been published. In spite of its importance, very little attention has been paid in the literature to MBFF multiplicity and grouping at the front-end design stage. MBFF grouping should be driven by logical, structural and FFs activity considerations.

Fig. 2. Power breakdown of MBFF compared to ordinary 1-bit FFs [1].
In a design reported by [1], 92% of the FFs have been grouped into MBFFs, the majority of which were 4-MBFFs, while the reset were 2-MBFFs. Fig. 2 shows the power breakdown of MBFF compared to 1-bit FF design. The power is normalized to the total power consumed by a 1-bit FFs core design (memories and IOs excluded). A 15% reduction of the total dynamic power is shown. Expectedly, power of the sequential logic and the power of the clock tree decreased, because the total number of clock drivers and the wire load connected to the MBFF internal drivers were reduced. The combinational logic power was increased, because some of the logic has been up-sized to recover of \( t_{PCQ} \) increase. To avoid the timing degradation occurred by \( t_{PCQ} \) increase we propose to introduce MBFF at the RTL design level. This will allow the backend and layout design stage to take \( t_{PCQ} \) into account and avoid timing problems upfront.

A work introducing the MBFF at the logic synthesis design stage was presented in [3], attempting to conclude on the pros and cons compared to a synthesis using ordinary FFs. The mapping of FFs to MBFF took place at the gate-level design produced by the RTL compiler. A 55nm 230MHz design of a System on a Chip (SoC) was experimented. The authors restricted the MBFF mapping to FFs belonging to the same bus, where both 2-MBFFs and 4-MBFF were used with 20% increase of their \( t_{PCQ} \). The usage of MBFFs reduced the number of clock sinks by 60%, leading to a simpler clock tree with 35% less clock buffers. That further reduced the clock skew by 30%. Table 1 summarizes the power savings. A dynamic power reduction of 13% is shown. Not surprisingly, power savings came on the expense of timing degradation, which has been remedy by introducing low voltage threshold (LVT) cells on critical paths, indicated by the increase of the leakage power.

<table>
<thead>
<tr>
<th>Number of FFs</th>
<th>Single bit</th>
<th>Multi bit</th>
<th>Difference [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-bit</td>
<td>29437</td>
<td>5267</td>
<td></td>
</tr>
<tr>
<td>2-bit</td>
<td>0</td>
<td>1860</td>
<td></td>
</tr>
<tr>
<td>4-bit</td>
<td>0</td>
<td>5216</td>
<td></td>
</tr>
<tr>
<td>Total FF area [( \mu^2 )]</td>
<td>394047</td>
<td>402955</td>
<td>+2.26</td>
</tr>
</tbody>
</table>

| FFs threshold type [%] | Standard | 11.5 | 49 | -33.4 |
| Skew [pSec]            | 249       | 176  | -29.4 |
| Seq. CKT power [mW]    | dynamic | 145  | 104 | -28.16 |
|                       | leakage  | 4.55 | 9.89 | +117 |
| Combinational power [mW] | dynamic | 124  | 117 | -12.4 |
|                       | leakage  | 28.9 | 28.8 | -0.35 |
| Clock Tree (sinks incl.) [mW] | 195 | 134 | -31.4 |
| Total dynamic power [mW] | 446 | 388 | -13.0 |
| Total chip power       | 682       | 629  | -7.77 |

Table 1. Power reduction obtained by MBFF design [3].

While the design flows supported by EDA tools handle MBFF at the RTL synthesis into gate-level implementation, they take very limited physical layout details into
account. Most importantly, the data toggling probability subsequently shown in Section 2 to significantly affect the MBFF grouping is completely ignored by those tools. The subsequently overviewed literature is mainly focused on MBFF physical implementation. Those works also ignore FFs’ activities, which this paper considers.

One of the earliest works on MBFF grouping at the physical layout stage was described in [4]. Each FF has been associated with time margins obtained from the layout comprising 1-bit FFs. The wires connected to the data input and output of a FF where anchored on their opposite side to the rest logic, while the position of the FF was allowed to move around, thus defining the region in layout where the FF can be displaced without violating timing. The merging of FFs pairs in 2-MBFFs was formulated as an optimization problem aiming at maximizing the count of merged FFs, such that the resulting MBFFs locations do not violate timing. There were also congestion constraints that were handled by dividing the silicon area into bins with limited occupancy of MBFFs. The problem has been solved with the aid of area proximity analysis. Following the above ad-hoc approach, a later work in [5] presented an algorithm with better computational efficiency to solve the same problem posed in [4], handling the same timing and area congestion constraints.

The FF clustering approach presented in [5] has later been used in [6] for replacement of 1-bit FF with Multi-Bit Pulsed-Latch (MBPL) in the physical layout. MBPL clock power was minimized by taking advantage of pulsed-latch timing behavior that is similar to a FF, and its time-borrowing capabilities similar to a latch. That offered more flexibility in meeting the timing constraints by expanding the allowable region where the original FFs could be displaced and merged in MBPLs. For further power saving the authors combined clock-gating (CG) into the MBPL structure. Few CG strategies were mentioned, but no details were provided, and the relation between the CG strategy, the FFs’ activities and their grouping in MBPLs was not conclusive. A recent work in [7] has addressed the combination of MBFF with CG. CG cloning was proposed in the decision of MBFF grouping, combined with layout proximity analysis of the 1-bit FFs, similar to [3-6]. Layout proximity considerations have also decided of whether 2-MBFF or 4-MBFF grouping was in order. The requirement of having a timing-converged layout as a starting point for the MBFF design flow is a burden, and practicality very restrictive. Timing constraints may be very tight, thus limiting the potential FFs merging. Moreover, having a timing-converged layout in hand reduces the incentive to change the design.

None of [3-7] considered FFs activity as a factor to drive the MBFF grouping. Our work proposes a systematic, toggling probability-driven MBFF grouping algorithm, provably maximizing the expected energy savings. In our mind MBFF should be introduced at the RTL and logic design level, based on architectural, structural and most importantly, on FFs activity considerations. The rest of the paper is organized as follows. Section 2 studies the effect of data toggling probabilities on the potential energy savings and Section 3 shows how to combine Data-Driven Clock-Gating (DDCG) with MBFF. Section 4 answers the question of what FFs should be grouped
in a DDCG MBFF, which answer is then used in Section 5 to present a grouping algorithm and its usage in a design flow. Section 6 presents experimental results and Section 7 concludes the discussion.

2. The effect of data toggling probabilities on energy savings

The dependency of the potential MBFF energy savings on its toggling probability is demonstrated in Fig. 3, obtained by SPICE simulations. It shows the energy consumed by 1-bit FF, 2-MBFF and 4-MBFF. Notice the “base” dynamic energy paid by the clock, regardless of the input activity. The base energy growth of 2-MBFF shown in Figs. 3 (b) and 3 (c) compared to 1-bit FF in Fig. 3(a) stems from its larger internal load. Expectedly, the energy consumption grows linearly with the data toggling probability, and it is twice larger when both inputs toggle simultaneously compared to single input toggling. Similar behavior is shown for 4-MBFF in Figs. 3(d) and 3(e).

![Fig. 3. The dependency of the MBFF energy savings on the toggling probability.](image)
Let $p$ be the data-to-clock toggling probability. Denote by $E_1$ the expected energy consumed by 1-bit FF. We conclude from Fig. 3(a) that

$$E_1(p) = \lambda_1 + \mu_1 p,$$

where $\lambda_1$ is the energy of the FF’s internal clock driver, and $\mu_1$ is the energy of data toggling. For 2-MBFF there are three possible scenarios: none of the FFs toggle, a single FF toggles, and both FFs toggle. Assuming data toggling independence, the expected energy consumption $E_2$ is

$$E_2(p) = \lambda_2 (1 - p)^2 + 2(\lambda_2 + \mu_2)p(1 - p) + (\lambda_2 + 2\mu_2)p^2 = \lambda_2 + 2\mu_2 p,$$

where $\lambda_2$ is the energy of the internal clock driver, and $\mu_2$ is the per-bit data toggling energy. For the general case of $k$-MBFF, let $\lambda_k$ be the energy of the MBFF’s internal clock driver and $\mu_k$ be the per-bit data toggling energy. Considering all the combinations of toggling FFs, the expected energy is

$$E_k(p) = \sum_{j=0}^{k} \binom{k}{j} \left( \lambda_k + j\mu_k \right) p^j (1 - p)^{k-j} = \lambda_k + k\mu_k p.$$  

The equality in (3) is obtained by applying some rearrangements [8].

To assess the potential MBFF energy savings, Fig. 4 shows the energy ratio of two and four 1-bit FFs to that of 2-MBFF and 4-MBFFs. We divide the energy difference between $k$ individual FFs and $k$-MBFFs, by the energy of the $k$ individual FFs. For small $p$ it shows savings of $(1.6 - 1)/1.6 = 35\%$ for $k = 2$ and $(2.2 - 1)/2.2 = 55\%$ for $k = 4$. For large $p$ the savings is $(1.18 - 1)/1.18 = 15\%$ for $k = 2$ and $(1.3 - 1)/1.3 = 23\%$ for $k = 4$. In typical VLSI systems the average $p$ does not exceed 0.05, so high savings is realizable. Section 4 which considers the MBFF energy savings by introducing DDCG generalizes the energy consumption model in (3) to the case of
distinct data toggling probabilities. It is also important to note that the toggling independence is a worst-case assumption, where in reality the correlation of FFs toggling can be used to yield higher energy savings [9].

3. Introducing clock-gating into MBFF

The MBFFs discussed so far were driven by a free-running un-gated clock signal. Fig. 5 illustrates a DDCG integrated into a \( k \)-MBFF. All the shaded circuits reside within a library cell. It was shown in [2] that given an activity \( p \), the group size \( k \) which maximizes the energy savings solves the equation

\[
(1 - p)^k \ln(1 - p)C_{FF} + \frac{C_{latch}}{k^2} = 0, \tag{4}
\]

where \( C_{FF} \) and \( C_{latch} \) are the clock input loads of a FF and a latch, respectively. The solution of (4) for various activities is shown in Table 2 for typical \( C_{FF} \) and \( C_{latch} \).

![Fig. 5. DDCG integrated into \( k \)-MBFF.](image)

<table>
<thead>
<tr>
<th>( p )</th>
<th>0.01</th>
<th>0.02</th>
<th>0.05</th>
<th>0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k )</td>
<td>8</td>
<td>6</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 2. Dependency of the optimal MBFF multiplicity on toggling probability.

Unless otherwise stated the MBFFs discussed in the sequel are DDCG. To grasp the power savings achievable by DDCG of a \( k \)-MBFF, Fig. 5 has been simulated with SPICE for various activities \( p \) and multiplicities \( k = 2, 4, 8 \). Fig. 6 shows the power consumption of a 2-MBFF. Line (a) represents the power consumed by two 1-bit FFs driven independently of each other. The 3.8\( \mu \)W power consumed for zero activity is due to the toggling of the clock driver at each FF, and it is always being consumed regardless of the activity. Line (b) corresponds to the ideal case where the two FFs toggle simultaneously. In that case the clock driver shared by the two FFs either toggles for the sake of the two, or it is disabled by the internal gater shown in Fig. 5. Expectedly, the power consumed for zero activity is nearly half compared to two 1-bit FFs. As the activity increases, the power of (b) is growing faster than (a) since the gating circuit consumes power proportionally to the activity. There is no point in using a 2-MBFF beyond the 0.17 activity crossing point, a case where power starts being lost.
Fig. 6. Power consumption of 2 FFs vs. 2-MBFF.

Line (c) shows the case where the FFs are toggling disjoint. This is obviously the worst case since the clock driver works for the two FFs, while only one needs it. As for (b), in case of disjoint toggling there is no point in using 2-MBFF if the FFs activities are higher than 0.11. Given an activity, the power savings of 2-MBFF is the distance between line (b) or (c) to (a). Notice that for zero activity the per-bit power savings is \((3.8-1.8)/2 = 1.0 \mu W\).

Fig. 7. Power consumption of 4 FFs vs. 4-MBFF.

Fig. 7 shows the power consumed by 4-MBFF, where line (a), correspond to four 1-bit FFs driven independently of each other, line (b) represent the best case of simultaneous toggling of the 4-MBFF FFs, and line (c) represents the worst case of disjoint toggling. For zero activity the per-bit power savings is \((7.4-2.2)/4 = 1.3 \mu W\), larger than the 1.0\(\mu W\) obtained for 2-MBFF. Notice however that for the worst case of disjoint toggling, 4-MBFF stops saving at 0.08 activity, earlier than 0.09 in 2-MBFF. In the best case of simultaneous toggling however, 4-MBFF is always favored over 2-MBFF. Similar conclusions hold for 8-MBFF shown in Fig. 8. Its per-bit power savings for zero activity is \((15.3-2.5)/8 = 1.6 \mu W\). Savings of 8-MBFF stops
at 0.06 activity in the worst case of disjoint toggling, and at 0.40 in the best case of simultaneous toggling.

Fig. 8. Power consumption of 8 FFs vs. 8-MBFF.

4. What FFs should be grouped in a DDCG MBFF?

Section 2 quantified the $k$-MBFF expected energy savings $E_k(p)$ under the assumption of toggling independence and free-running un-gated clock. Section 3 showed how toggling correlation affects the breakeven probability where a MBFF stops saving energy. Clearly, the best grouping of FFs could be achieved for FFs whose toggling is almost completely correlated. The problem of FFs grouping yielding maximal toggling correlation, and hence maximal power savings, has been shown as NP-hard, and a practical solution yielding nearly maximum power savings was presented in [10]. Its drawback is the requirement of early knowledge of Value Change Dump (VCD) vectors, derived from many power simulations representing the typical operation and applications of the design in hand. Such data may not exist in the early design stage. More common information is the average toggling bulk probability of each FF in the design, which the following discussion takes advantage of in deriving an optimal toggling probability-driven FFs grouping.

The analysis so far assumed that all the FFs grouped in a MBFF have same data toggling probability $p$. FFs’ toggling probabilities are usually different of each other, and an important question is therefore how the probability varieties affect the FFs grouping. Past works considered either structural FFs grouping (e.g., successive bits in registers), or post-layout grouping driven by physical proximity. We subsequently show that data toggling probabilities matter and should be considered for maximizing energy savings.

Given $n$ FFs $\{\text{FF}_i\}_{i=1}^n$, consider their grouping in 2-MBFFs. Let a 2-MBFF, denoted $\text{FF}_{(i,j)}$, comprise $\text{FF}_i$ and $\text{FF}_j$, toggling independently with probabilities $p_i$ and $p_j$, respectively. When none is toggling, the clock of $\text{FF}_{(i,j)}$ is disabled and its internal
clock driver does not consume dynamic energy. When both FF\textsubscript{\(i\)} and FF\textsubscript{\(j\)} are toggling, the clock of FF\textsubscript{\((i,j)\)} is enabled and the clock driver energy is fully useful and there is no waste. A waste happens when one FF is toggling, while its counterpart does not. There, the clock pulse is enabled, driving both FFs, whereas only one needs it. A waste W\textsubscript{\((i,j)\)} of half of the internal clock driver energy \(\lambda_2\) thus occurs (see (2)), given by

\[
W_{(i,j)} = \frac{\lambda_2}{2} \left[ p_j (1 - p_i) + p_i (1 - p_j) \right] = \frac{\lambda_2}{2} \left( p_j + p_j - 2p_ip_j \right).
\]  

Given FF\textsubscript{\(i\)}, FF\textsubscript{\(j\)}, FF\textsubscript{\(k\)} and FF\textsubscript{\(l\)}, their pairing in two 2-MBFFs yields the energy waste

\[
W_{(i,j)} + W_{(k,l)} = \frac{\lambda_2}{2} \left[ p_i + p_j + p_k + p_l - 2 \left( \sum_{(a)} p_i p_j + p_k p_l \right) \right].
\]  

While the term (a) of (9) is independent of the pairing, the term (b) does depend. The expression \(W_{(i,j)} + W_{(k,l)}\) is minimized when (b) is maximized. If \(p_j \leq p_j \leq p_k \leq p_l\), the pairing \(\{\text{FF}_{(i,j)}, \text{FF}_{(k,l)}\}\) is favored over \(\{\text{FF}_{(i,k)}, \text{FF}_{(j,l)}\}\) since \(W_{(i,j)} + W_{(k,l)}\) - \(W_{(i,k)} + W_{(j,l)}\) = \(-\lambda_2 (p_i - p_j)(p_j - p_k)/2 \leq 0\). \(\{\text{FF}_{(i,j)}, \text{FF}_{(k,l)}\}\) is similarly favored over \(\{\text{FF}_{(i,k)}, \text{FF}_{(j,l)}\}\). The generalization for pairing of \(n\) FFs is straight forward. Let \(n\) be even and \(P: \{\text{FF}_{(i,i)}\}^{n/2}_{i=1}\) be a pairing of \(\text{FF}_1, \text{FF}_2, \ldots, \text{FF}_n\) in \(n/2\) 2-MBFFs. The following energy waste \(\mathbf{W}(P)\) results in

\[
\mathbf{W}(P) = \sum_{i=1}^{n/2} W_{(i,i)} = \frac{\lambda_2}{2} \left[ \sum_{j=1}^{n} p_j - 2 \sum_{i=1}^{n/2} p_{(i)} p_{(i)} \right].
\]  

Since \(\sum_{i=1}^{n} p_j\) is independent of the pairing, \(\mathbf{W}(P)\) is minimized when \(\sum_{i=1}^{n/2} p_{(i)} p_{(i)}\) is maximized. The optimal pairing minimizing \(\mathbf{W}(P)\) is defined by the following theorem [8].

**Theorem 1.** Let \(n\) be even and let \(\text{FF}_1, \text{FF}_2, \ldots, \text{FF}_n\) be ordered such that their toggling probabilities satisfy \(p_1 \leq p_2 \leq \cdots \leq p_n\). The pairing \(P: \{\text{FF}_{(2i-1,2i)}\}^{n/2}_{i=1}\) of successive FFs is minimizing \(\mathbf{W}(P)\) given in (10). The above result of grouping in 2-MBFFs is generalized for grouping in \(k\) -MBFFs as follows.
Theorem 2. Let \( n \) be divisible by \( k \), and let \( \text{FF}_1, \text{FF}_2, \ldots, \text{FF}_n \) be ordered such that their toggling probabilities satisfy \( p_1 \leq p_2 \leq \cdots \leq p_n \). The grouping 

\[
P: \left\{ \text{FF}_{\left( k(i-1)+1, \ldots, ki \right)} \right\}_{i=1}^{n/k}
\]

of successive FFs is minimizing the energy waste incurred by the \( n/k k \)-MBFFs.

The case where \( n \) is not divisible by \( k \) has also been addressed in [8].

5. Capturing everything together in a design flow

It was mentioned in Section 3 that the knowledge of the toggling vectors (VCDs) of every FF, derived from extensive simulations, may obtain the best FF grouping [9, 10]. Such data is not always available, and we therefore assume the model used in Section 4, assuming that the FFs toggle independently of each other. The relation between the power savings to FF’s activity \( p \) and MBFF multiplicity \( k \) has been discussed in Sections 2 and 3. Section 4 showed that grouping in monotonic order of \( p \) maximizes the power savings. The activity \( p \) and the multiplicity \( k \) must therefore be jointly considered in a design flow to maximize the power savings.

To this end we consider Figs. 6, 7 and 8, illustrating the power savings of 2-MBFF, 4-MBFF and 8-MBFF, respectively. The interim line (d) shown between the extreme cases of simultaneous and disjoint FFs toggling represents a more realistic operation, where FFs may toggle independently of each other. Knowing the activity of a FF, the decision in what MBFF size it should be grouped will follow the interim lines. Fig. 9 puts Figs. 6, 7 and 8 on a common scale of per-bit power consumption, for which they have been divided by their respective multiplicity.

![Maximal Saving Activity Regions](image)

Fig. 9. Division of the activity into ranges of maximal savings.

Fig. 9 illustrates how the range of FF activity is divided into regions to obtain maximal power savings. The black line follows the power consumed by a 1-bit un-gated FF. The triangular areas bounded between the black line and the green, blue and red per-bit power consumption lines, indicate the amount of power savings resulting
by grouping a FF in 2-MBFF, 4-MBFF and 8-MBFF, respectively. It shows that for very low activity it pays to group FFs in 8-MBFF. As the activity increases, there will be some exchange point where 4-MBFF pays more. At some higher activity 2-MBFF will better pay, up to an activity where the power savings stops. We take advantage of that behavior in the following MBFF grouping algorithm.

1. Sort the \( n \) FFs such that \( p_1 \leq p_2 \leq \cdots \leq p_n \).
2. Set \( i = 1 \).
3. Decide on optimal \( k \) by \( p_i \), based on Fig. 9.
4. Group \( FF_i, FF_{i+1}, \ldots, FF_{i+k-1} \) in a \( k \)-MBFF.
5. Set \( i \leftarrow i + k \).
6. If \( i > n \) stop. Else go to 3.

Few practical comments are in order. In addition to toggling probabilities, MBFF grouping should also consider logical relations and physical place and route constraints. An example is the pipeline registers of a microprocessor, which are natural candidates for MBFF implementation (see experimental results in Section 6). It makes no sense to mix bits of different pipeline stages. It is obvious and natural that the place and route tool will put bits belonging to same register close to each other, while FFs clusters of registers belonging to distinct pipeline stages will be placed apart of each other. FFs of different pipeline registers should therefore be not mixed in a MBFF, although from toggling probability standpoint their grouping may be preferred. Similar arguments hold for other system’s busses and registers such as those storing data, addresses, counters, and alike. Another example is the FFs of Finite State Machines (FSMs) in control units, whose MBFF grouping should not cross control logic borders.

Though the proposed algorithm is aimed at RTL or gate design levels, it can also be combined with the grouping methods proposed by [3-7]. There, an initial placement takes place as a “dry run” to obtain initial FFs’ layout proximity directives. The toggling probability-driven algorithm can then consider those to guide the MBFFs grouping. The later real place and route will use MBFF library cells, unlike [3-7] which rip up the old FFs and insert MBFFs replacements, a non-trivial and tedious layout task, which is saved by our design flow.

6. Experimental results

The proposed MBFF design flow has been used for a 32-bit pipelined MIPS processor, implemented in TSMC 65nm process technology. Workload of two programs has been used, shown in Table 3. For each test the average activity of a FF in the pipelined register is shown in blue color under the name of the pipeline stage. Notice the activity decrease with the progress of the pipeline stage from instruction fetch (IF) to write-back (WB).

Two MBFF grouping methods are examined. In the first, FFs have been grouped sequentially according to their bit number in their register. The second method
grouped FFs in increasing order of their activities, shown in Section 4 to be optimal when FFs are assumed to toggle independently of each other. Both grouping methods adhered to the constraints of not crossing clock domain boundaries and not mixing FFs of unrelated logic entities. Table 3 shows for each $k$-MBFF, $k = 2, 4, 8$ the average activity. In most cases grouping by monotonic activity is preferred (colored in green), though in few cases it worsened (colored in red). That can happen since the grouping is blind to toggling correlation.

<table>
<thead>
<tr>
<th>grouping method</th>
<th>IF / ID</th>
<th>ID / EXE</th>
<th>EXE / MEM</th>
<th>MEM / WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>by index</td>
<td>0.174</td>
<td>0.261</td>
<td>0.353</td>
<td>0.140</td>
</tr>
<tr>
<td>by activity</td>
<td>0.169</td>
<td>0.261</td>
<td>0.353</td>
<td>0.134</td>
</tr>
<tr>
<td>improve [%]</td>
<td>+2.9</td>
<td>0</td>
<td>0</td>
<td>+4.3</td>
</tr>
</tbody>
</table>

Table 3. Average FF activity of pipeline registers in 32-bit MIPS.

The pipeline registers were then implemented with MBFFs grouped by monotonic order of their activity. As shown in Fig. 9, the grouping starts with 8-MBFFs for the low activities, and then it is progressing to 4-MBFFs and 2-MBFFs with the FFs activities increase, up to the zero gain point where grouping stops and the rest FFs stay alone and un-gated. Those could of course be grouped in un-gated MBFFs, just to reduce the number of internal clock drivers. Table 4 shows the power savings achieved at each of the pipeline registers for the sort and matrix multiplication weighted workload. The results were measured with SpyGlass [11] simulation where the MIPS was operated in 1.1V and 200MHz. 34.6% savings was achieved. The pipelined registers consumed 65% of the entire MIPS power (memory not included), so the total power reduction of the entire power (CG HW overhead included) was 23%.

<table>
<thead>
<tr>
<th>grouping method</th>
<th>IF / ID</th>
<th>ID / EXE</th>
<th>EXE / MEM</th>
<th>MEM / WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>by index</td>
<td>0.203</td>
<td>0.311</td>
<td>0.422</td>
<td>0.169</td>
</tr>
<tr>
<td>by activity</td>
<td>0.198</td>
<td>0.294</td>
<td>0.388</td>
<td>0.174</td>
</tr>
<tr>
<td>improve [%]</td>
<td>+2.5</td>
<td>+5.5</td>
<td>+8.1</td>
<td>-3.0</td>
</tr>
</tbody>
</table>

Table 4. Power savings in the pipeline registers of a 32-bit MIPS.

We finally show the power savings achieved by the grouping algorithm for a complete industrial network processor designed in 28nm TSMC process technology, operating in 800MHz. The processor is divided into seven units, named A to G, shown in Table 6. It consumes a total of 6.2 Watts, in which 45% is charged to the
clock network with its underlying FFs. The original design comprises un-gated MBFFs, so the power savings is purely due to the addition of the clock gating in Fig. 5, on top of the savings obtained by less drivers in the un-gated MBFFs that existed in the original design. Furthermore, the original design includes extensive clock enable logic signals, defined by both RTL compiler and manual insertions.

The activities of the FFs were profiled first and then sorted. Table 5 shows a total of 8% net power savings, where the power measurements include both dynamic and static components and all the CG HW overheads. The 8% power savings was obtained on top of 9% savings that had been achieved by changing from 1-bit FFs to un-gated MBFFs, yielding a 17% combined savings. Such savings is highly appreciated by the industrial VLSI design community. The area penalty due to the introduction of clock-gating circuitry was 2.3%.

<table>
<thead>
<tr>
<th>unit</th>
<th>FF CLK power [mW]</th>
<th>total CLK power [mW]</th>
<th>total unit power save [mW]</th>
<th>FF CLK power save (%)</th>
<th>FF CLK power save [%]</th>
<th>total CLK power save [%]</th>
<th>total unit power save [%]</th>
<th>area penalty [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>80</td>
<td>1,112</td>
<td>1,802</td>
<td>44</td>
<td>57.6</td>
<td>4.09</td>
<td>2.52</td>
<td>1.7</td>
</tr>
<tr>
<td>B</td>
<td>304</td>
<td>316</td>
<td>1,638</td>
<td>104</td>
<td>33.4</td>
<td>32.5</td>
<td>6.22</td>
<td>2.8</td>
</tr>
<tr>
<td>C</td>
<td>184</td>
<td>268</td>
<td>760</td>
<td>76</td>
<td>41.9</td>
<td>28.6</td>
<td>10.1</td>
<td>2.7</td>
</tr>
<tr>
<td>D</td>
<td>72</td>
<td>172</td>
<td>294</td>
<td>32</td>
<td>45.2</td>
<td>19.2</td>
<td>11.2</td>
<td>2.3</td>
</tr>
<tr>
<td>E</td>
<td>162</td>
<td>368</td>
<td>884</td>
<td>88</td>
<td>53.4</td>
<td>23.8</td>
<td>9.90</td>
<td>4.3</td>
</tr>
<tr>
<td>F</td>
<td>112</td>
<td>204</td>
<td>252</td>
<td>80</td>
<td>69.7</td>
<td>38.2</td>
<td>31.0</td>
<td>1.3</td>
</tr>
<tr>
<td>G</td>
<td>124</td>
<td>368</td>
<td>556</td>
<td>72</td>
<td>57.4</td>
<td>19.7</td>
<td>13.0</td>
<td>1.9</td>
</tr>
<tr>
<td>total</td>
<td>1,040</td>
<td>2,804</td>
<td>6,186</td>
<td>496</td>
<td>47.5</td>
<td>17.7</td>
<td>8.00</td>
<td>2.3</td>
</tr>
</tbody>
</table>

Table 5. Power savings in an 40nm network processor.

References


