Volume 4, Issue 4, April 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Dual Optimized System for Flip-Flop Grouping Using Data Driven Clock Gating Approach

K. Madhanmohan[1], R. Murugasami[2]
PG Student (AE), Dept of ECE, Nandha Engineering College, Erode, Tamilnadu, India[1]
Assistant professor, Dept. of ECE, Nandha Engineering College, Erode, Tamilnadu, India [2]

Abstract: ***Data driven clock gating is a popular technique used in many synchronous circuits for reducing dynamic power dissipation. Power optimization plays the major role in the recent years. So far the power is reduced by reducing the circuit size, or by reducing the capacitor, frequency and voltage value of the circuit. The existing system shows that the area is reduced by grouping the flip flops. The aim of the proposed work is that the dynamic power is reduced by multibit flip-flop (MBFF). Commonly the clock gate is used for power saving method. But the gate still leaves the large amount of redundant clock pulses. To reduce the hardware overhead involved, flip-flops are grouped to share the common enabling signal. By combining the data driven gating with multibit flip-flop (MBFF). The power consumption will be reduced. Here the Xilinx software tool will be used for implementing this proposal system.***

Keywords: Clock gating, clock networks, dynamic power Reduction, Multiple bit flip flop

I. INTRODUCTION

One of the major dynamic power consumers in computing and consumer electronics products is the system's clock signal, where it takes 30%–70% of the total dynamic power consumption [1]. There are many techniques used to reduce the dynamic power are developed, in which clock gating is predominant. Ordinarily, when a logic unit is clocked, it is based on the sequential elements receiving the clock signal, sequentially they will toggle in the next cycle whether it is required or not. 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 depending on various heuristics in an attempt to increase clock gating opportunities. With the rapid increase in design complexity, computer aided design tools supporting system-level hardware description have become commonly used. Although substantially increasing design productivity, such tools require the employment of a long chain of automatic synthesis algorithms, from register transfer level (RTL) down to gate level and net list. Unfortunately, such automation leads to a large number of unnecessary clock toggling, thus increasing the number of wasted clock pulses at flip-flops (FFs) as shown in this paper through several industrial examples.

In a recent paper, a model for data-driven gating is developed based on the toggling activity of the constituent FFs [9]. The optimal fanout of a clock gater yielding maximal power savings is derived based on the average toggling statistics of the individual FFs, process technology, and cell library in use. In general, the state transitions of FFs in digital systems depend on the data they process. Assessing the effectiveness of data-driven clock gating requires, therefore, extensive simulations and statistical analysis of the FFs' activity. Another grouping of FFs for clock switching power reduction, called multibit FF (MBFF). MBFF attempts to physically merge FFs into a single cell such that the inverters driving the clock pulse into its master and slave latches are shared among all FFs in a group. MBFF grouping is mainly driven by the physical position proximity of individual FFs, while grouping for data driven clock gating should combine toggling similarity with physical position considerations. While answered the question of what is the group size that maximizes power savings, this paper studies the questions of: 1) which FFs should be placed in a group to maximize the power reduction and 2) how to algorithmically derive those groups. We also describe a backend design flow implementation. In the next section, we briefly overview data-driven clock gating, which motivates this paper. Section III presents the problem of optimal FF grouping and its inherent difficulty. Section III introduces layout considerations into FF grouping and describes a near-optimal grouping algorithm. Section IV discusses the implementation of a practical design flow.

II. DATA-DRIVEN CLOCK GATING

Clock enabling signals are very well understood at the system level and thus can effectively be defined and capture the periods where functional blocks and modules do not need to be clocked. Those are later being automatically synthesized into clock enabling signals at the gate level. In many cases, clock enabling signals are manually added for every FF as a part of a design methodology. Still, when modules at a high and gate level are clocked, the state transitions of their underlying FFs depend on the data being processed. It is important to note that the entire dynamic power consumed by a system stems from the periods where modules' clock signals are enabled.

Therefore, regardless of how relatively small this period is, assessing the effectiveness of clock gating requires extensive simulations and statistical analysis of FFs toggling activity, as presented subsequently. Fig. 1 shows the FFs' toggling activity in an arithmetic block comprising 22K FFs, designed in 40-nm technology, taken from Ceva's X1643 DSP core for multimedia and wireless baseband applications.

As Fig. 1 shows, a FF toggled its state only 2.9% of the clock enabled time period, on the average, thus more than 97% of the clock pulses driving FFs are useless. Such a low toggling rate (of nonclock signals) is very common. Another example of a 40-nm control block comprising 37-K FFs (part of Mellanox ConnectX network processor has also been examined. There, the clock signal is enabled 20% of the time and within that window the average FF toggling is only 1.3%, and here too more than 98% of the clock pulses driving FFs are useless. It follows from the above examples that no matter what RTL and gate levels clock enabling signals are followed, there are still many opportunities to gate the clock signal at the FF level. The data-driven gating proposed in [9] is illustrated in Fig. 2.

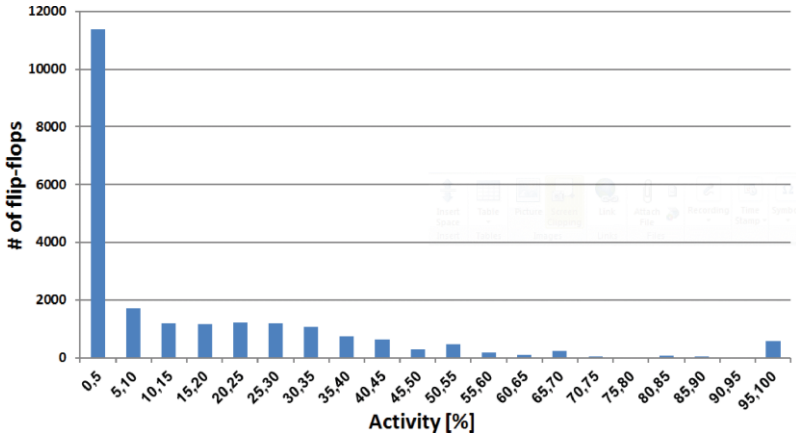

Fig 1: Toggling statistics of Ceva's X1643 DSP core over 240-K clock cycles.

A FF finds out that its clock can be disabled in the next cycle by XOR ing its output with the present data input that will appear at its output in the next cycle. The outputs of *k* XOR gates are OR ed to generate a joint gating signal for *k* FFs, which is the n latched to avoid glitches. The combination of a latch with AND gate is commonly used by commercial tools and is called integrated clock gate (ICG). Such data driven gating is used for a digital filter in an ultralow-power design. A single ICG is amortized over *k* FFs. There is a clear tradeoff between the number of saved (disabled) clock. pulses and the hardware overhead. With an increase in *k*, the hardware overhead decreases but so does the probability of disabling, obtained by OR ing the *k* enable signals. Let the average toggling probability of a FF (also called activity factor) be denoted by p ($0 < p < 1$).
The latch and gater (AND gate) overheads are amortized over *k* FFs.
It is shown in [9] that the number *k* of jointly gated FFs for which the power savings are maximized is the solution of

$$(1 - p)k \ln(1 - p)\,(cFF + cW) + clatch/k2 = 0 \qquad (1)$$

where *c*FF is the FFs clock input capacitance, *c*W is the unit-size wire capacitance, and *c*latch is the latch capacitance including the wire capacitance of its *clk* input. Table I shows how the optimal *k* depends on *p*. Such a gating scheme has considerable timing implications, which are discussed in [9]. Reference [24] reported 20% power savings. It took advantage of the very low dynamic range of the data in a digital filter. The gating logic is tailored to the structure of the filter, whereas the approach discussed in this paper is more general and applies to large scale and a wide range of designs.

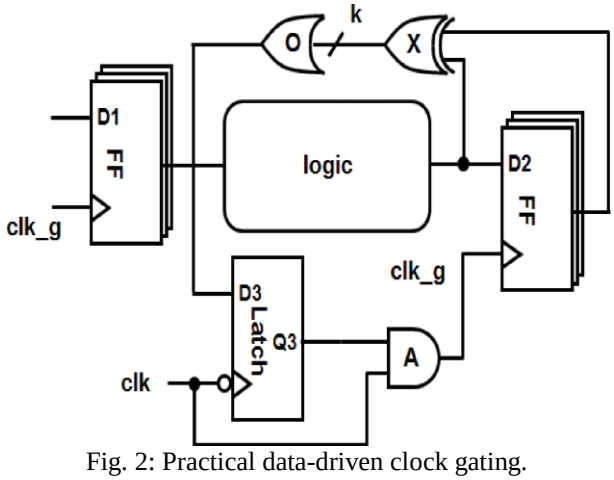

Fig. 2: Practical data-driven clock gating.

III. PHYSICAL LAYOUT CONSIDERATIONS IN FFS GROUPING

Finding sets of FFs that minimize the number of redundant clock pulses is not enough to maximize power savings. Grouping must account for the on-die locations of FFs and gates, which affect the power consumption due to the capacitive loads resulting from their connections. The physical locations of FFs affect also the delay and clock skew, and it is therefore desirable for FFs driven jointly by the same clock gates, to be placed in proximity of each other. A scheme for constructing clock trees when the positions of the FFs in leaves are known is described in [7]. A cost function weighting the sum of clock activities and clock pin distances is minimized. Such a cost function is problematic since the physical meaning of a weighted sum of activities and distances is not well defined and requires delicate tuning of the weights. An alternative of summing the products of activity by the diameter of smallest circles enclosing FF sets looks more appropriate and is used in this paper. This sum of products has the physical units of effective capacitance, thus explicitly measuring power consumption, and no weights are needed. To support activity-diameter products, the FF grouping activity defined before is modified in order to account for the FFs' layout proximity.

IV. IMPLEMENTATION AND INTEGRATION IN A DESIGN FLOW

In the following, we describe the implementation of data driven clock gating as a part of a standard backend design flow. It consists of the following steps.

1) Estimating the FFs toggling probabilities involves running an extensive test bench representing typical operation modes of the system to determine the size *k* of a gated FF group by solving (1). 2) Running the placement tool in hand to get preliminary preferred locations of FFs in the layout. 3) Employing a FFs grouping tool to implement the model and algorithms presented in Sections III and IV, using the toggling correlation data obtained in Step 1 and FF locations' data obtained in Step 2. The outcome of this step is *k*-size FF sets (with manual overrides if required), where the FFs in each set will be jointly clocked by a common gater. 4) Introducing the data-driven clock gating logic into thehardware description (we use Verilog HDL). This is done automatically by a software tool, adding appropriate Verilog code to implement the logic described in Fig. 2. The FFs are connected according to the groupingobtained in Step 3. A delicate practical question iswhether to introduce the gating logic into RTL or gatelevel description. This depends on design methodology in use and its discussion is beyond the scope of this paper. We have introduced the gating logic into the RTL description. 5) Re-running the test bench of Step 1 to verify the full identity of FFs' outputs before and after the introduction of gating logic. Although data-driven gating, by its very definition, should not change the logic of signals, and hence FFs toggling should stay identical, a robust design flow must implement this step. 6) Ordinary backend flow completion. From this point, the backend design flow proceeds by applying ordinary place and route tools. This is followed by running clocktree synthesis. Few timing-related comments are in order. The extra gating delay introduced by the feedback loop in Fig. 2 should not exceed the delay margins of paths from the clock input *clk_g* of FF1 to the data input D2 of FF2. In ordinary designs, notably in automatically synthesized blocks, most of the delay margins are large enough to absorb the introduction of the gating logic.

V. RESULTS

The design flow described in Section IV is experimented on a DSP core comprising 22 k FFs (Figs. 1, 3, and 4), another large vectored DSP core comprising 100 k FFs, a 3-D graphics accelerator [9], and a network processor control block [23]. The DSP cores include both arithmetic and control circuits. the theoretical optimal group size for various toggling probabilities. For FF toggling probabilities from $p = 0.01$ to $p = 0.05$, the group size maximizing the net power savings is The latter is due to the very low toggling rate of that processor. As shown in the tables, the net dynamic power savings are 15% for the DSP cores and 20% for the 3-D graphics accelerator. Experiments for the network processor control block also yield nearly 20% power savings where the optimal group size is $k = 8$. between $k = 8$ and $k = 4$. We measured the power savings compared with the nominal designs using the Spy Glass EDA power simulator [8]. The measurements accounted for the logic and latch overhead required by the gating (see Fig. 3),

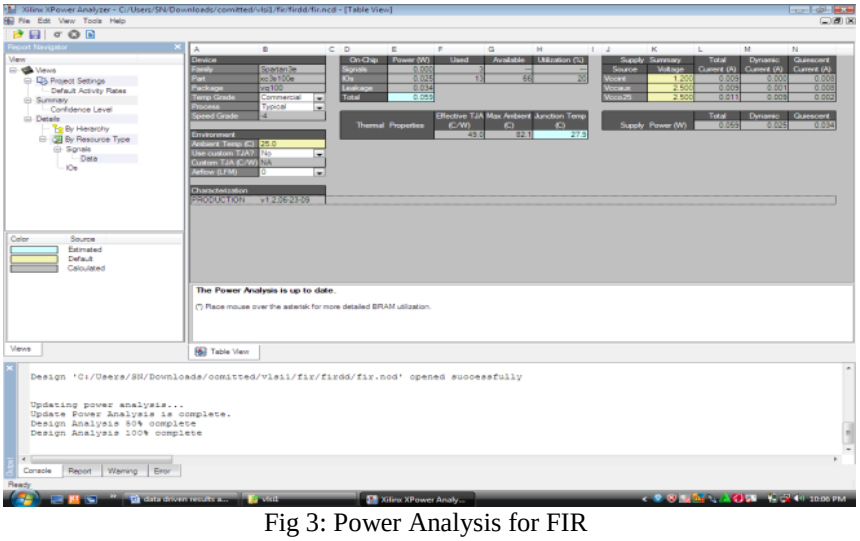

Fig 3: Power Analysis for FIR

thus they reflect the net power savings. the resulting power for a wide range of group sizes, where the maximum power savings are achieved for $k = 8$. Notice that the above results include not only the clock network and sequential power but also the power consumed in the combinational logic, which is about half of the total dynamic power. Hence, considering clocking power savings alone, 30%–40% is achieved.

Since the toggling probability is averaged across the entire FFs, it may happen that different sub-blocks will have different probabilities. It is therefore possible to further reduce the power by using various k values in different sub-blocks. By mixing $k = 8$ with $k = 16$, the power savings in reached 16.3%. Another interesting observation is the slight growth in the combinational logic power, due to the extra XOR connected at every FF and the other logic involved in data-driven gating (Fig. 4).

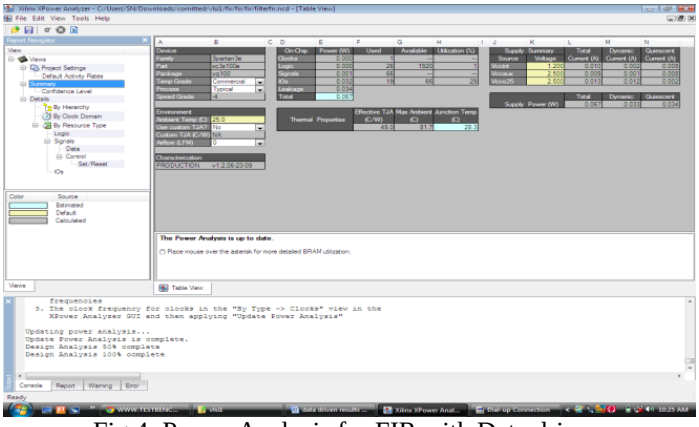

Fig 4: Power Analysis for FIR with Data driven

The growth is almost independent of k. Since Spy Glass is charging the latch overhead to the sequential power, this can be explained by the fact that most of the increase is due to the XOR gates, which is independent of k. It is interesting to compare the relative power savings achieved by applying synthesis-based gating only or data-driven gating only or both.To this end, we ran RTL power simulation of the processors in four gating modes: 1) no gating; 2) synthesis-based available from an EDA vendor ; 3) data-driven; and 4) both combined. summarizes the total power consumption of each case and compares it with the native design without any gating [shown in (1)]. The synthesis-based gating alone 2) reduced the total power to 84%, 34%, and 59%, respectively. Applying data-driven gating on top of the synthesis-based one 4) further reduced the power to 68%, 29%, and 50%, respectively. However, for all three designs, the application of datadriven gating alone yielded higher power savings than the combination of data-driven and synthesis-based, reduction the power to 65%, 27%, and 42%, respectively. This is due to the fact that data-driven gating stops any unnecessary clock pulse, and the inclusion of the synthesis-based gating only adds logic circuits that becomes redundant once the data-based gating is applied. Although synthesis-based clock gating is a well-established design methodology, the above experiments encourage its replacement by data-driven gating.

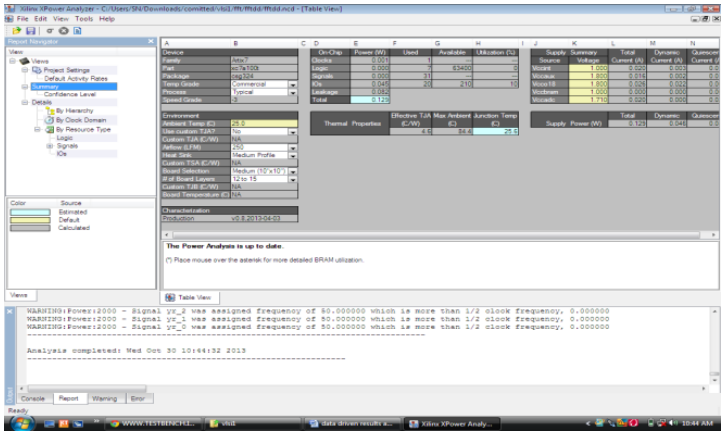

Fig 5: Power Analysis for FFT

To further characterize the benefits of the two gating modes, we analyzed several different circuit types, for example, control units, arithmetic units, and register files. We selected such units from the large designs that we analyzed above, where the type of the circuit is specified. The resulting power consumptions (in percentages), are relative to the no clock gating case, and the lowest power is indicated in red (circled). It can be seen that for control circuits, data-driven gating 3) is outperforming synthesis-based gating 2). This is explained by their very low toggling rate, where data-driven is most useful. Similar behavior is observed for arithmetic circuits (their IO registers are included). Expectedly, synthesis-based gating is still favored for register files. This follows since only one register is changing its data at a time, a condition that can easily be caught and defined in the RTL code.

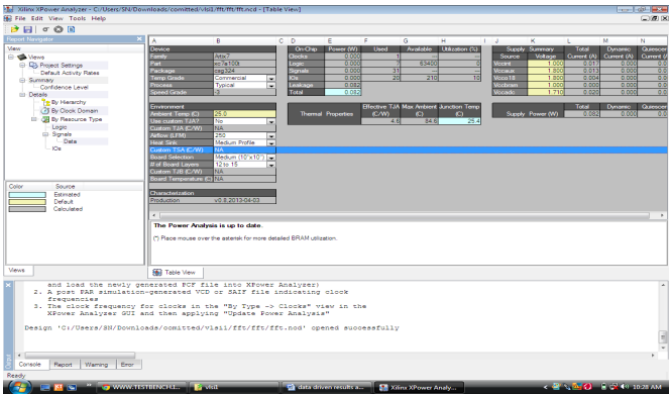

Fig 6: Power Analysis for FFT with Data driven

Therefore, applying datadriven gating on top of synthesis-based mostly adds circuit overhead. The results of the combined synthesis-based and data-based gating scheme are worse than the data-driven only gating for all seven circuits. Thus, unless register files can undergo only synthesis-based gating and data-based gating will not be applied to them, synthesis-based gating should be completely replaced by data-based gating. As mentioned earlier, the gating scheme may have considerable timing implications. Potential timing violations are avoided using Step 3 of the flow proposed in Section IV, which groups FFs based on the preliminary preferred locations of FFs in the layout found by Step 2. Such restriction increases the amount of redundant clock pulses in (4), but this is compensated by shortening the wires connecting the clock gaters to their corresponding FFs.

Table 1: Power Comparison Table

Circuit Type	Gating Modes		
	Synthesis (*ii*)	Data-driven (*iii*)	Combined (*iv*)
Control I (3D graphics pipeline)	81%	65%	71%
Control II (DSP video processing)	83%	56%	64%
Adder I (24 bit Carry-select)	92%	59%	61%
Adder II (32 bit Carry-skip)	89%	58%	62%
Multiplier (24 bit Booth Multiplier)	91%	71%	74%
Reg-file I (3D 32 Regs of 24-bit)	42%	57%	61%
Reg-file II (DSP 64 Regs of 32-bit)	47%	60%	69%

Table I compares grouping with and without FF location constraints, for a design comprising 4.9 k FFs and a test of 105 clock cycles, showing a 5%–10% increase in the redundant pulses. This has only a little impact on the saving measured in simulation (less than 1%). To quantify the timing impact of data-driven clock gating, we ran static timing analysis on the native 3-D graphics design without gating and then compared it with the design with the gating. Fig. 6 illustrates the margin (slack) distribution for 200-MHz clock cycle. It can be seen that the margin distribution has slightly worsened as more paths now have a negative slack. The violations need to be taken care of for timing closure and a variety of actions and techniques are possible, but their discussion is beyond the scope of this paper.

VI. CONCLUSION

This paper studied the problem of grouping FFs for joint clocking by a common gater to yield maximal dynamic power savings. Although the problem was NP-hard, we discussed several practical algorithms to solve it and found several of them to be useful in a real design automation implementation. The solution was integrated in a practical design flow. Experimental results of DSP cores, a network processor control block, and a 3-D graphics accelerator were presented, achieving 15%–20% total power reduction. The FF grouping problem also arises in MBFF, where distinct FFs were combined in one physical cell to share their internal clock drivers. It is interesting to consider the combination of data-driven gating with MBFF in an attempt to yield further power savings.

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. Very Large Scale Integr. (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 Intll. Conf. Integr. Circuit Design Technol.*, Jun. 2008, pp. 143–146.

[4] C. Chunhong, K. Changjun, and S. Majid, "Activity-sensitive clock tree construction for low power," in *Proc. Int. Symp. Low Power Electron. Design*, 2002, pp. 279–282.

[5] A. Farrahi, C. Chen, A. Srivastava, G. Tellez, and M. Sarrafzadeh, "Activity-driven clock design," *IEEE Trans. Comput.-Aided Design 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. Int. Symp. Phys. Design*, 2008, pp. 182–189.

[7] M. Donno, E. Macii, and L. Mazzoni, "Power-aware clock tree planning," in *Proc. Int. Symp. Phys. Design*, 2004, pp. 138–147.

[8] *SpyGlass Power* [Online]. Available: http://www.atrenta.com/solutions/ spyglass-family/spyglass-power.htm

[9] S. Wimer and I. Koren, "The Optimal fan-out of clock network for power minimization by adaptive gating," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 10, pp. 1772–1780, Oct. 2012.

[10] Y.-T. Chang, C.-C. Hsu, M. P.-H. Lin, Y.-W. Tsai, and S.-F. Chen, "Post-placement power optimization with multi-bit flip-flops," in *Proc. IEEE/ACM Int. Conf. Comput., Aided Design*, Nov. 2010, pp. 218–223.

