Implementation of System Probabilistic Model using Simulink

[1] Padmini G Kaushik, [2] Dr.S.M.Gulhane, [3] P.D.Pawar

[1] Assistant Professor, [2] Professor , [3] Assistant Professor

[1]Electronics & Telecommunications & Tech

[1]Jawaharlal Darda Institute of Engineering Technology, Yavatmal, Maharashtra, India.

Abstract:

The increasing demand for low power mobile computing and consumer electronics products has refocused VLSI design in the last two decades on lowering power and increasing energy efficiency. Power reduction is treated at all design levels of VLSI chips, from architecture through block and logic levels, down to gate-level, circuit, and physical implementation. One of the major dynamic power consumers is the system's clock signal, typically responsible for up to 50% of the total dynamic power consumption [1]. Clock network design is a delicate procedure and is therefore done in a very conservative manner under worst-case assumptions. It incorporates many diverse aspects such as a selection of sequential elements, controlling the clock skew, and decisions on the topology and physical implementation of the clock distribution network [2].

Index Terms **– Clock Network, Power reduction, clock Gating.**

I. INTRODUCTION

Ordinarily, when a logic unit is clocked, its underlying sequential elements receive the clock signal regardless of whether or not they will toggle in the next cycle. With clock gating, the clock signals are ANDed with explicitly defined enabling signals. Clock gating is employed at all levels: system architecture, block design, logic design, and gates [3]. Clock enabling signals are usually introduced by designers during the system and block design phases, where the interdependencies of the various functions are well understood. In contrast, it is very difficult to define such signals at the gate level, especially in control logic, since the interdependencies among the states of various flip-flops (FFs) depend on automatically synthesized logic. We claim that a big gap exists between clock disabling that is derived from the HDL definitions and what can be achieved through detailed knowledge regarding the FFs' activities and how they are correlated with each other.

We present an approach to maximize clock disabling at the gate level, where the clock signal driving a FF is disabled (gated) when the FF state is not subject to a change in the next clock cycle. Few attempts to take advantage of this principle have been made before [5]–[7] for design levels higher than individual FFs; all of them rely on various heuristics in an attempt to increase clock gating opportunities. An activity-driven clock tree was presented in [6] for data flow blocks. This is a good example where the designer is aware of the interrelations between the various modules and therefore can introduce appropriate enable signals.

Clock gating does not come for free. Extra logic and interconnects are required to generate the clock enabling signals and the resulting area and power overheads must be considered. In the extreme case, each clock input of a FF can be disabled individually, yielding maximum clock suppression. This, however, results in a high overhead; thus suggesting the grouping of several FFs to share a common clock disabling circuit in an attempt to reduce the overhead. On the other hand, such grouping may lower the disabling effectiveness since the clock will be disabled only during periods when the inputs to all the FFs in a group do not change. In the worst case, when the FFs' inputs are statistically independent, the clock disabling probability equals the product of the individual probabilities, which rapidly approaches zero when the number of involved FFs increases. It is therefore beneficial to group FFs whose switching activities are highly correlated and derive a joint enabling signal. The state transitions of FFs in digital systems like microprocessors and controllers depend on the data they process. Assessing the effectiveness of clock gating requires, therefore, extensive simulations and statistical analysis of FFs activity.

The proposed gating will dynamically prune large portions of the clock tree if it becomes clear that none of the driven FFs is subject to a change in the next cycle.

II. Clock gating Techniques

2.1 Latch based clock gating technique

The period in which the change in enable signal can be sensed is called the active period and the period in which the change cannot be sensed is called passive or sleep period. But as shown in Figure 1. The negative half cycle by default becomes the active period and a positive period becomes the sleep period. The anomaly occurs when enabling signal changes during the sleep period leading to an incorrect design.

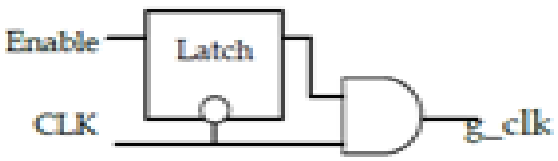

Fig.1 Latch based Clock gating Cell

2.2. Flip-Flop Based Clock Gating Technique

In this type of technique, a flip-flop is used as the control element. Enable is given the input signal. The change in enable signal is only tracked and shown as the output only when the negative edge of the clock arrives. If the output of the flip-flop is high, the following sequential circuit gets the clock. The same anomaly which existed in latch based clock gating exists here too. It is rather persistent here and the probability of missing the change on enable pin is high as the sleep period is longer in a flip-flop as compared to a latch. Figure 2. Shows a flip-flop based clock gating cell.

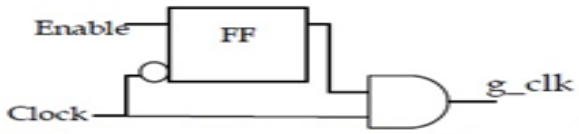

Fig. 2 Flip flop based clock gating Cell

2.3 Gate Based Clock Gating

This technique has proved to be efficient than the other two described above. It is one of the simplest methods. Also since gates are used instead of flip-flops the space required is reduced to a greater extent. Normally OR gates are used. Figure 3 shows this technique. Enable is the input signal. When input i.e. enable is '0', the output of the gate remains the same as the previous one i.e there is no change. The output is the same as the clock when enable becomes '1'.Thus the dynamic power dissipation is reduced to a larger extent. Also, the problem faced earlier of missed changes does not persist here. Thus the sleep period does not affect this circuit

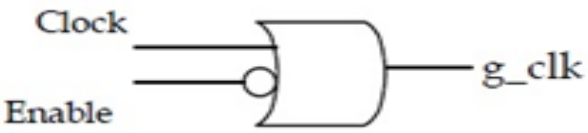

Fig.3 Gate based clock gating cell

III. Methodology of Clock Gating

3.1 Gated Clock Network Modeling

The construction of gated clock trees raises two questions. The first is: what should be the fan-out of a gater, i.e., how many FFs should a leaf gater drive, and similarly for higher levels of the tree, how many children gaters should be driven by one parent? The second question concerns what FFs should be grouped to share a common gater, and similarly for higher levels of the tree, which sibling gaters should be grouped for maximum power savings? To answer the first question we will use a power model that accounts for interconnects of a clock signal and the enabling (gating) signals overhead. While all works so far have assumed a binary clock tree model, we derive the optimal fan-out of the clock tree which maximizes the net switching power savings, accounting for the overhead incurred by the extra logic circuitry required to generate the gating signals. This, to the best of our knowledge, has not been addressed before. For the second question, the matching technique heuristically applied in is used here in a more formal way, but the problem was lately shown to be NP-complete.

A key aspect of the optimal solution of the above problems is the probabilistic behavior of FFs' toggling. However, unlike their register toggling and gating model that was developed based on random simulations, this paper uses a worst-case probabilistic model, yielding a result that provides a provably lower bound on the power savings. It is therefore uniformly applicable to any design and the actual power reduction obtained by the methodology proposed here can only be higher than that predicted by our worst-case model. It is important to note that the proposed methodology tests a large set of typical applications before clock tree construction in an attempt to find the probability and correlation of FF toggling and follow the best-case rather than the worst-case lower bound. FF toggling correlation is used for optimally grouping the FFs.

3.2 Adaptive Clock Gating Implementation

An XOR gate compares the FF's current Output with the present data input that will appear at the output in the next cycle. The XOR's clk_en output indicates whether a clock signal will be required in the next cycle. The clock driver in Fig. 1 is then replaced by a 2-way AND gate called clock gater. We will use the symbol in Fig. 2 to represent FFs that incorporate the generation of clk_en. In practice, the XOR is connected to the output of FF's internal master rather than D, as it is guaranteed to be stable when the FF's slave is transparent. We could drive several FFs with a common gater if we knew that they are toggling simultaneously most of the time, thus achieving almost the same power reduction, but with fewer gaters. The grouping may place up to several dozens of FFs in a single group and is usually done by synthesizers during the physical design phase [14]. Such tools are focusing on the skew, power, and area minimization, and are not aware of the toggling correlations of the underlying FFs.

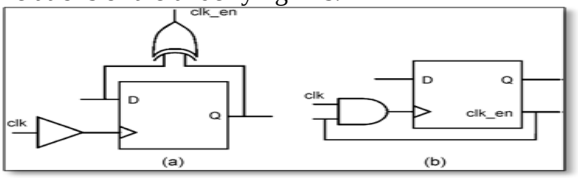

Fig. 4. Enabling of the Clock Signal.

Fig. 4 shows how to join k clk_en signals generated by distinct FFs into one gating signal. It saves the individual clock gaters at the expense of an OR gate and a negative edge-triggered latch that is required to avoid glitches of the enable signal. Due to the power consumed by the latch, such joining is justified only for K ≥3. The combination of a latch with an AND gate is commonly used by commercial tools and is called an integrated clock gate (ICG). The hardware savings increase, but the number of disabled clock pulses is decreasing. Thus, the scheme proposed in Fig. 5 to be beneficial, the clock enabling signals of the grouped FFs must be highly correlated. Such correlation and its implication on the clock savings are thoroughly studied below. Since the enabling signals ORed in

Fig. 5 are the outputs of XOR gates, which may have glitches, the question of the power penalty is in order. Fortunately, the probability of signal toggling is well known to be very low so the average amount of glitches is expected to stay small as well.

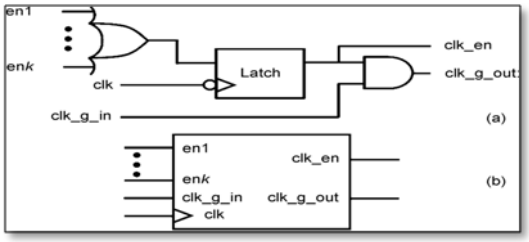

Fig. 5. Joining K Enabling Signals Generated By Distinct Flip-Flops Into One Gating Signal.

The proposed adaptive clock gating has considerable timing implications. Fig. 5 illustrates a FF to FF logic stage with its driving clock signals. In the physical implementation, the XOR gate is integrated into the FF, while the OR gate, AND gates and the latch are integrated into the clock gater. Fig. 6 depicts the timing sequence and its implied constraints. There are two distinct clock signals: clk_g is the ordinary gated signal driving the registers, while clk is driving the latches of the clock gaters. Both have the same period denoted by Tc.

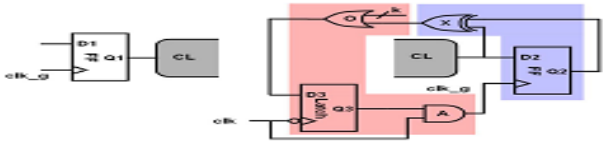

Fig. 6. Application of Clock Gating With Feedback Enabling Signals. The Components Highlighted In Red Centered Shaded Area Comprise The Clock Gater, While Those In Blue Reside In FF.

3.3 Joint Gating and Gater's Optimal Fan-Out

Assume that a circuit contains n=2N FFs whose clock signals are driven by the tree shown in Fig. 5. Its leaves are connected to the FFs and the gaters' fan-out is k=2N. We assume that N=αK, where is the number of levels of the clock tree. A leaf gater has a unit size (driving strength). The gater at the first level is connected to the leaf by a wire of unit length and unit width. We now introduce the following notations to quantify and analyze the power savings achieved by joint clock enabling: cFF —FF's clock input capacitance; clatch—latch capacitance; including the wire capacitance of its clk input; cw—unit wire capacitance; cgater —unit drive gater capacitance; cOR —OR gate capacitance; β—level to level gater's sizing factor; γ—level to level wire width sizing factor; δ—level to level wire length sizing factor.

In this notation the size of a gater in level j is βj-1 and the size of a wire connecting level j to j-1 is (γδ)j-1,1≤ j ≤α , as commonly happens in tree networks such as the H-tree [17]. The total capacitive load of the resulting clock tree is

$$C_{\text{tree}} = nc_{\text{FF}} + c_{\text{gater}} \sum_{j=1}^{\alpha} \left(\frac{n}{k^j}\right) \beta^{j-1} + c_{\text{w}} \sum_{j=1}^{\alpha} \left(\frac{n}{k^{j-1}}\right) (\gamma\delta)^{j-1}$$

$$= n \left[c_{\text{FF}} + \frac{c_{\text{gater}}}{\beta} \frac{1-(\beta/k)^{\alpha}}{1-\beta/k} + c_{\text{w}} \frac{k}{\gamma\delta} \frac{1-(\gamma\delta/k)^{\alpha}}{1-\gamma\delta/k} \right].$$

To assess the clock gating impact on the power we consider the toggling of FF as an independent random variable. A FF has the probability to change state and $q = (1-p)$ to stay unchanged. The probability of a group of FFs to stay unchanged (as a group) is therefore qk. The probability is sometimes called the activity factor. It is well known that the average activity factor of non-clock signals is very low, since a typical signal toggles very infrequently.

$$C^{1}_{net\_saving} \geq q^{k}(C_{FF}+C_W)-[C_{latch}/k+(1-q)(C_W+C_{OR})]$$

In logic-gate design-level, where clock gating usually takes place only at the first level of the tree. Such gating is what is currently supported by several CAD tools, leaving to the user the decision regarding the value of, usually by relying on experience. Equating to zero the derivative of (6) concerning yields the following implicit equation for the optimal k :

$$q^{k} lnq(c_{FF}+c_w)+c_{latch}/k^2=0$$

Notice that the gating overhead term (1-q) (cW – cOR) appearing in (6) does not affect the optimal since it is being paid by each of the FFs, regardless of the value of k. In an attempt to find the optimal value of k, Fig.7 is showing the normalized power savings per FF derived from (6). The savings is compared to the non-gated situation. Various values of $q=(1-p)$ have been examined to explore the behavior of the optimal. The relative capacitance of FFs, latches, OR gate and unit wires connecting the first level gater to the FFs depend on the specific technology and cell library in hand.

We assumed all to be equal in Fig.7. As expected, the lower the toggling probability of FF is, the higher the optimal is. The optimal values obtained in the plots agree with the common practice of EDA tools. It is shown that significant savings can be achieved. Recall however that there is delay and area overhead and though high fan-out values result in fewer gaters, the OR fan-in is increasing accordingly, which will further increase area and delay overheads.

We observed similar amounts of savings based on gate-level Verilog simulations of designs, where adaptive gating was added to the first level of clock gater. This translates to 5% of the total dynamic power savings of the entire chip. The net savings were obtained on top of savings obtained by clock enabling signals which have already been introduced by the designer at the RTL Verilog.

Additional savings can be obtained by gating at higher levels of the tree. The normalized net power savings per FF for gating at three levels is illustrated in Fig. 10 as a percentage of the non-gated situation. There, gater's drive, wire width and wire length sizing factors of $\beta=\sqrt{2}$, $\gamma=\sqrt{2}$, $\delta=2$ and, respectively, have been used. As can be seen, higher power savings per FF are achieved by gating at the second and third levels. For low toggling probabilities, more power savings are obtained. Though the percentage is a bit lower than in Fig. 4, the total is higher since it is from a larger capacitance. On the other hand, once FFs toggling probabilities increase, the savings go rapidly down, and for $p>0.2$ there is only power loss. The area implications of the proposed scheme for acceptable values of the fan-out need to be further investigated by incorporating it into a backend layout flow. This, however, is beyond the scope of this study.

3.4 Grouping of Flip-flops

As described in the introduction the basic idea of the transformation is to switch the clock of flip-flops that take their own data. With this algorithm we will try to give to the flip-flop a "c1ock" only when it needs too, to be precise only when the input data is changing. We have named this special clock: Gated Clock

Clock gating is a transformation that is performed on a synchronous digital circuit. For each flip flop in the circuit, the hold condition is determined, Let the condition under which the value that is clocked into the flip flop is identical to its current value. Under this condition, a transition on the clock input of the flip flop can be suppressed without changing the circuit's behavior. Such a modified clock is called a gated clock. Since gating a clock involves a latch, and thus area overhead and extra power dissipation, flip-flops with similar hold conditions are grouped to be clocked by the same gated clock. Power reduction is achieved if a gated clock governs enough flip flops with a combined hold condition that is often true.

The logic required for finding the activity of Flip-flops is inserted in each flip flop and simulation is carried out.

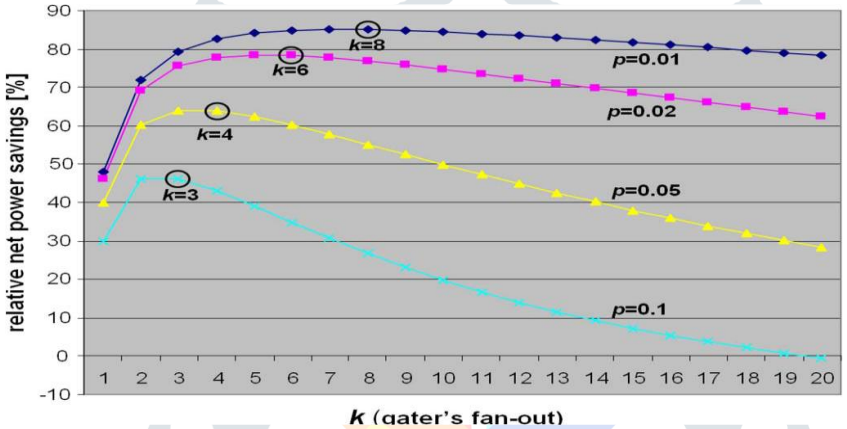

Fig. 7. Normalized power net savings per FF obtained by adaptive gating at1st level of clock tree in (6). The savings are compared to the non-gated situation. The optimal fan-out is marked for each toggling probability.

3.5 Algorithm for Flip-Flop Grouping:

1) Let a pre-enabled clock signal have n FFs and be switching during $m + 1$ cycles
2) Individual activity vectors are drawn for each flip-flop. The current output and current input of a particular flip-flop is XOR-ed to check to toggling probability $a=(a_1....,a_m)$ is the activity toggling of flip-flop

3)1- $\| a_i \oplus a_j \| / m$ measures the activity correlation between FF_i and FF_j.

4) $\| a_i \oplus a_j \|$ is the number of redundant clock pulses occurring jointly clocking FF_I and FF_j

5) The Flip-flop having approximately the same activity are grouped together

6)

IV. DESIGN FLOW

The design flow begins with a design in RTL. It is important, to begin with, a design that has been proven to work properly. The design must not include any IP's (intellectual property) or RTL sources that are not visible to the user, and therefore cannot be edited.

At this point, the design flow supports implementation for a single clock domain. Moreover, the sequential and combinational logic must be separated in the RTL for the scripts to run properly.

The Simulation environment steps: Add tracing code to the design. In our project, we have added this manually in our design.

- Analyzing the activity vectors of each flip flop
- Grouping of Flip-flop with same activity:

In this stage flip-flops having approximately the same activity are grouped together and are given the same clock.

- Clock gating implementation:
- In this stage, we are manually adding clock gating logic to the design.
- Design with clock gating:

This design has the same logic behavior as the initial design. However, it contains the gate components according to the implementation process.

- Implementing the system using Xilinx System Generator

In this stage, we are generating the model of the given system using the Xilinx System generator.

V. RESULTS:

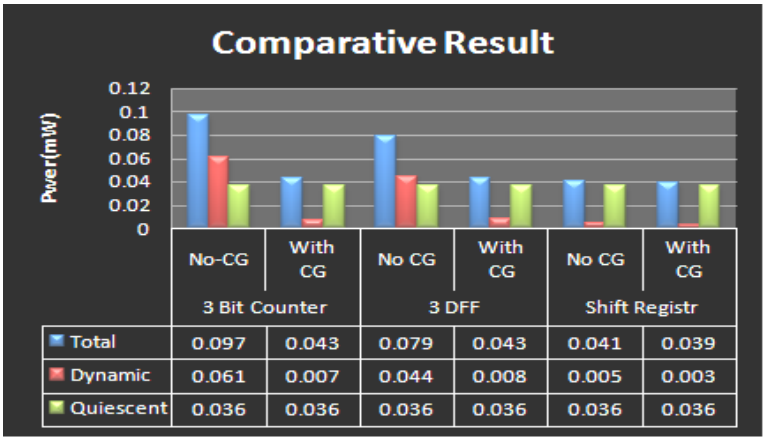

Fig. 9. Comparative results of power reduction With clock gating and without clock gating

Figure 9 shows the achieved dynamic power reduction for a 3 bit counter and shift register using the proposed technique.

VI. CONCLUSION AND FUTURE WORK

We aim at reducing dynamic power consumption since it is responsible for about 30% to 70% of power dissipation. Achieved a reduction in dynamic power for 3 Dff is 44%, for 3 bit Counter is 6% and for shift register is 4.7%. This method can be applied to any generalized system (i. e synchronous sequential circuit) It Cost-efficient method of reducing power dissipation despite the increase in circuitry.

The FF grouping problem also aroused 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] Jagrit Kathuria, M.Ayoubkhan, Arti Noor, "A review of Clock Gating Techniques".
[2] Shmuel Wimer, Member, IEEE, and Israel Koren, Fellow, IEEE," Design Flow for Flip-Flop Grouping in Data-Driven Clock Gating".
[3] Shmuel Wimer and Israel Koren, Fellow, IEEE, "The Optimal Fan-Out of Clock Network for Power Minimization by Adaptive Gating"
[4] Shmuel Wimer, "On optimal flip-flop grouping for VLSI power minimization"
[5] Dushyant Kumar Sharma, "Effects of Different Clock Gating Techniques on Design".
[6] T. Lang, E. Musoll, and J. Cortadella, "Individual flip-flops with gated clocks for low power datapaths," IEEE Trans. Circuits Syst. II, Exp.Briefs, vol. 44, no. 6, pp. 507–516, Jun. 1997.
[7] W. Aloisi and R. Mita, "Gated-clock design of linear-feedback shift registers," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 55, no. 5, pp. 546–550, Jun. 2008.
[8] 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.