Data Driven Clock Gating Scheme for Low Power Dissipation

Padmini G kaushik¹, Dr.S.M.Gulhane², J.Chaitra Hebbar³
Assistant Professor¹, Professor²
JDIET, Yavatmal, India

Abstract:
Clock signal are used in synchronous circuits only for synchronization and does not perform any computation. Clock signals are main source of power dissipation in synchronous circuits. Clock gating scheme is therefore employed to reduce switching activity of synchronous circuits. Clock gating scheme is employed at all level architecture, module design, logic design and gates to achieve maximum power reduction. In this paper, data driven clock gating scheme is presented for flip flops at gate level. The clock signal to flip flop is blocked whenever its state is not supposed to change in next clock cycle. In data driven clock gating scheme the output of the flip flop is XORed with present input. The operation is performed to determine whether its state is supposed to change in next clock cycle and possible disable clock signals. The combinational circuit is used to generate enable signal used to enable / disable clock signal to the flip flop. Latch based AND gate clock circuit has shown better performance as compared to only logic gate based clock gating. The latch is needed to avoid hazards to propagate through AND gate. Thus data driven clock gating scheme can avoid unnecessary switching and save power dissipation.

Key words: Dynamic power; Clock gating; Xilinx; Latch based; Flip-flop

I. INTRODUCTION

Several techniques to reduce the dynamic power have been developed, of which clock gating is predominant. Ordinarily, when a logic unit is clocked, its underlying sequential elements receive the clock signal regardless of whether or not 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 time 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.

In many cases clock gating is applied only to the first level of gaters directly driving FFs, since the majority of the load occurs at the leaves of the clock tree where the FFs are connected. Even if we could ideally stop the clock from driving all the FFs when it is not required, the rest of the network will continue pumping clock signals and wasting energy. We consider therefore gating higher levels of the clock tree (closer to root). These portions of the tree may also consume considerable power since they are using long and wide wires plus intermediate drivers to avail robust clock signals for far end FFs. 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

A. Latch based clock gating

As the name suggests, latch is the controlling element in this technique. Both the latch and the enable signals together decide whether the clock is to be provided for the particular cycle or not. Enable signal is given the input. During the negative clock cycle the latch reflects the change in enable signal whereas during the positive clock cycle the output remains the same as previous and no change of enable signal is reflected whatsoever. The clock reaches the sequential logic circuitry only when the output of latch is high. 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. negative half cycle by default becomes the active period and positive period becomes the sleep period. The anomaly occurs when enable signal changes during the sleep period leading to an incorrect design.

![Figure 1. Latch based Clock gating Cell](image1)

B. Flip-flop based clock gating

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 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.

![Figure 2. Flip flop based clock gating Cell](image2)

C. 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 also 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’, output of the gate remains the same as previous one i.e there is no change. The output is 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.

![Figure 3. Gate based clock gating cell](image3)

III. METHODOLOGY OF CLOCK GATING

A. 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 which accounts for interconnects of 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. 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.

![Figure 4. Clock tree distribution network](image4)

It is important to note that the proposed methodology tests a large set of typical applications prior to 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.

B. Adaptive clock gating implementation

Figure 5 shows how a FF can find out that its clock can be disabled in the next cycle. A 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. 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 skew, power, and area minimization, and are not aware of the toggling correlations of the underlying FFs.

Figure 6 shows how to join \( k \) clk\(_{\text{en}} \) 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 \( K \geq 3 \).

The combination of a latch with an AND gate is commonly used by commercial tools and is called integrated clock gate (ICG). Clearly, the hardware savings increases with, but the number of disabled clock pulses is decreasing. Thus, for the scheme proposed in Figure 5 to be beneficial, the clock enabling signals of the grouped FFs must be highly correlated. Since the enabling signals ORed in Figure 6 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.

**CONCLUSION AND FUTURE WORK**

We aim at reducing dynamic power consumption since it is responsible for about 30% to 70% of power dissipation. Achieved reduction in dynamic power using latch based technique for 3 bit Adder is 64%. This method can be applied to any generalized system (i.e synchronous sequential circuit) It Cost efficient method of reducing power dissipation despite
increase in circuitry. The FF grouping problem also aroused in multi bit flip-flop (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


