Towards Automated Power Gating of Registers using CoDeL

Nainesh Agarwal and Nikitas Dimopoulos
Dept. of Elec. and Comp. Engineering
University of Victoria
Victoria, BC, Canada
{nagarwal,nikitas}@ece.uvic.ca

Abstract—In this paper, we use the CoDeL platform to develop test circuits and analyze the potential and performance impact of power gating individual registers. For each register, we examine the percentage of clock cycles for which they can be powered off, and the loss of performance incurred as a result of waiting for the power to be restored. Using a time-based technique to determine when the registers can be turned off results in 15% bit cycles saved at a performance loss of 2%. We then propose a method, which uses the information available to the CoDeL compiler to predict when the components can be powered off. Results show that our CoDeL assisted gating scheme allows up to 58% more power gated bit cycles than the time-based technique, with similar performance loss.

I. INTRODUCTION

To keep up with the requirements of miniaturization and long battery life for portable devices, it is essential to reduce power consumption in the VLSI circuit components of such devices. To reach this objective, the most effective method is to lower the supply voltage. As the supply voltage is reduced, by scaling the CMOS technology towards sub 100nm, an exponential growth in subthreshold leakage current is expected [1]. In addition, the reduction in logic gate sizes is causing a rapid rise in leakage currents [2]. As this trend continues, the leakage current will become the dominant source of total power dissipation in CMOS circuits.

To reduce leakage, power gating has been shown to be an effective technique [3]. Power gating relies on the detection of idle periods in the circuit. During these idle periods, the supply voltage can be switched off to the appropriate circuit component to conserve leakage power. At the end of the idle period, the supply voltage is restored to resume normal operation. Power gating approaches rely on trying to predict idle periods for either storage structures (SRAMs, registers) [4], [5] or functional units [6], [7].

We also use power gating as a mechanism to reduce leakage power in registers and functional units. To allow us to efficiently detect and utilize idle periods we use the CoDeL design platform. CoDeL (Controller Description Language) [8], [9] is a rapid hardware design platform that allows circuit description at the algorithmic level. Since CoDeL implements a design as a state machine it has sufficient information on the usage of registers and functional units to predict idle times and allow efficient power gating. The ultimate goal is to allow the CoDeL platform to automatically implement power gating, as appropriate, without any user intervention.

Here, we explore the gating potential of registers using two approaches. We first study the power savings expected by using a time-based technique, whereby power gating is performed after observing a pre-determined number of idle cycles, and the power is restored once an operation is detected. Second, we use CoDeL to predict when the idle periods occur and appropriately power gate the registers.

To test the power gating potential we have implemented and used the DSP kernel benchmarks from the DSPstone suite [10]. These kernel benchmarks consist of 14 code fragments which are commonly used in DSP algorithms, such as FIR/IIR filters.

II. POWER GATING

Power gating of a circuit block is performed by using an appropriate header or footer transistor. To begin power gating a “sleep” signal is applied to the gate of this transistor to turn off the supply voltage to the circuit block. To revive the block for use, the “sleep” signal is de-asserted and power is restored.

In the case of memory elements such as registers, multi-threshold CMOS (MTCMOS) [11] retention registers can be used. During normal operation, there is no loss in performance and during power-down mode the register state is saved to a “balloon” latch, which has a high voltage threshold resulting in minimal leakage. A MTCMOS register implementation is shown in figure 1. Using a MTCMOS register, all reads can be performed from the balloon latch. It is only when a write is necessary that we need to power up the high-performance low-threshold flip-flop.
In figure 2 we present the supply voltage and the various phases of a circuit component as it is power gated\(^1\). From time \(T_0\) to \(T_1\) the circuit component is busy and thus can not be gated. This period is \(T_{busy}\). At time \(T_1\), the component becomes idle. It takes the control logic from \(T_1\) to \(T_2\) to make the decision to engage gating. This period is called \(T_{idledetect}\). From \(T_2\) the supply voltage begins to drop. At point \(T_3\) the aggregate leakage power savings equals the overhead of switching the header transistor on and off. The period, \(T_{breakeven}\), from \(T_2\) to \(T_3\), is the minimum power gating duration to achieve net leakage power savings. During the period \(T_{sleep}\), from \(T_3\) to \(T_4\) the device is asleep and we accumulate net power savings. At \(T_4\) the control logic needs to reactivate the component. From \(T_3\) to \(T_5\) the supply voltage rises. During this period, \(T_{wakeup}\), a performance penalty may be incurred if the pending operation needs to wait for the power to be restored. Finally, at \(T_5\) the power is fully restored and the circuit can resume normal operation.

### III. EVALUATION FRAMEWORK

To evaluate CoDel’s power efficient compilation we use the DSP kernel suite from the DSPstone benchmark [10]. All kernels from the suite are implemented using CoDel and compiled to generate synthesizable VHDL. To perform the required arithmetic operations, we have used a single cycle 16 bit fixed point unit. The fixed point unit (FXU) is written in VHDL using the fixed point package obtained from [12]. It is interfaced by the CoDel implemented kernels to perform the required arithmetic operations. For data storage, a single port memory is implemented in VHDL for simulation. The overall benchmark architecture is presented in figure 3.

We are interested in the power gating of the registers used in the implementation of the CoDel kernel module. Any registers in the FXU or the memory are not gated. All clock cycle results presented are based on trace data obtained from VHDL simulation of the kernel circuits.

### IV. TIME-BASED POWER GATING

One technique to power gate circuit components is to observe the state of the component and initiate gating when a sufficient number of idle cycles are detected. Techniques such as this have been used for cache memories [4] and show significant leakage savings with minimal performance impact.

To implement this technique, each circuit component needs state machine logic similar to the one shown in figure 4. Normally the component is in the IDLE_DETECT or BUSY state. As long as the component is being used, the state remains BUSY. Once the component becomes idle we enter the IDLE_DETECT state. When the consecutive idle cycle count increases beyond \(T_{idledetect}\), the component enters the POWER_DOWN state. Here it waits for period \(T_{breakeven}\) to allow for the voltage supply to reduce. If at any time the component is needed, a ready signal is generated causing the component to enter the WAKEUP state. Otherwise, after \(T_{breakeven}\) cycles, the SLEEP state is entered. When the circuit component is next needed, the WAKEUP state is entered where a waiting period of \(T_{wakeup}\) cycles is required to restore the supply voltage. Once the component is powered up, the BUSY state is entered. When the circuit prematurely goes from the POWER_DOWN state to the WAKEUP state, the component may not be fully powered down. Thus, for restoring the power it will not take the full \(T_{wakeup}\) cycles. However, we conservatively penalize the full \(T_{wakeup}\) cycles in this case.

To measure the leakage savings we only consider the savings while the circuit is in the SLEEP state. Although, there may be some additional power savings in the WAKEUP state, we conservatively do not include these power savings.

---

\(^1\)Our model here follows the description presented in [7].
According to this framework we see that our results are dependent on three parameters: \( T_{\text{idledetect}} \), \( T_{\text{breakeven}} \), and \( T_{\text{wakeup}} \). \( T_{\text{wakeup}} \) is the time it takes to overcome the energy overhead of gating a unit. \( T_{\text{breakeven}} \) is the overhead of restoring the power to a unit. The parameters \( T_{\text{breakeven}} \) and \( T_{\text{wakeup}} \) are a function of the circuit design and thus can not be controlled by the user. The \( T_{\text{idledetect}} \) parameter, however, can be controlled to effect the aggressiveness of the power gating mechanism. A lower value will result in the gating of shorter idle periods resulting in potentially more gated cycles with a greater performance loss.

V. CoDeL ASSISTED POWER GATING

The CoDeL platform [9], [13] targets the specification and design of hardware architectures at the behavioral level. CoDeL is a procedural language in which the order of the statements implicitly represents the sequence of activities. It extracts the data and control flow from the program automatically, assigns the necessary hardware blocks and exploits inherent parallelism. It is similar to the C programming language and is therefore easy to learn. The CoDeL compiler produces synthesizable VHDL code which can be targeted to any technology including FPGA or ASIC. Details of the language syntax can be found in [9].

CoDeL uses a sequential machine to determine the sequence of operations and data transfers in and out of registers. Because of this sequential machine, we know the exact time of the events, and we can anticipate them.

We have used CoDeL to help determine the idle states where power gating can be used, statically. For each register, at compile time, CoDeL iterates through each state of the state machine implementation of the circuit and looks ahead \( T_{\text{idledetect}} \) states to determine if there are any potential writes to the register. If there is no write to the register in the next possible \( T_{\text{idledetect}} \) states, a power off suggestion is noted for the gating control logic. If during the next \( T_{\text{idledetect}} \) possible states the register is written, a power off suggestion is not made. An example is shown in figure 5, where a \( T_{\text{idledetect}} \) of three is used. It should be noted that, when performing the search, CoDeL looks at all possible branches in the state machine. As with the time-based technique, the \( T_{\text{idledetect}} \) parameter is chosen a priori, and is the same for all registers of the circuit under design.

As we can see in figure 4, the primary decision to initiate gating is based on the occurrence of a streak of idle cycles. However, in many cases, CoDeL, through analysis of the state machine implementation, is able to initiate gating at a much earlier stage (dashed line in figure 4).

VI. RESULTS

Here we examine the effects of the three parameters \( (T_{\text{idledetect}}, T_{\text{breakeven}} \) and \( T_{\text{wakeup}} \)) on the power gating ability and the performance of the circuit. Specifically we look at the percentage of cycles that a register or execution unit spends in the SLEEP state and weight the saved clock cycles by the width of the register, we call this the percentage of bit cycles in the SLEEP state. It is computed as

\[
\frac{\sum_{i=0}^{N} \text{len}(R_i) \cdot (\text{cycles in SLEEP state})}{\text{total cycles executed} \cdot \sum_{i=0}^{N} \text{len}(R_i)} \cdot 100\%,
\]

where \( N \) equals the number of registers, \( R_i \) is the \( i \)th register and \( \text{len}(R_i) \) is the bit width of the \( i \)th register. The performance impact of gating is computed as the number of additional clock cycles needed when power gating is introduced. This is computed as

\[
\frac{\text{total cycles executed without gating}}{\text{total cycles executed with gating}} \cdot 100\%.
\]

The DSPstone benchmark consists of 14 different kernels [10]. The results presented here are an arithmetic average of the results obtained for each kernel implementation.

In figure 6 we see the effect of \( T_{\text{idledetect}} \) for various \( T_{\text{breakeven}} \) values (9, 14, 24 and 29) on the gating effectiveness of the circuits. The figure presents the percentage of bit cycles in SLEEP mode using the time-based and the CoDeL assisted techniques. We see that in all cases, the percentage of SLEEP
bit cycles decreases roughly linearly with $T_{\text{idledetect}}$. In all cases, we see that the CoDeL assisted method performs as well or better than the time based technique. This difference is more pronounced at lower values of $T$.

Figure 7 shows the performance impact for various values of $T_{\text{wakeup}}$ (3, 6 and 9). We see that there is very little variation in the performance impact between the time-based and CoDeL schemes. This suggests that the CoDeL scheme is not able to discover many new gating opportunities. However, it is able to begin gating earlier resulting in increased gating effectiveness. Further, once gating is initiated, the wakeup mechanism between the two schemes is the same, resulting in the same performance loss. Examining the results, we see that lower values of $T_{\text{idledetect}}$ cause significant performance loss. This means that although there are a large number of short idle periods which can benefit from gating, the performance degrades since this causes a large increase in the number of cases where the circuit needs to wait for a power up to occur.

Figure 8 more clearly shows the benefit of the CoDeL assisted technique. We find that for any given performance loss, CoDeL is able to provide better gating effectiveness. From the results above we can see that a $T_{\text{idledetect}}$ between 11 and 21 cycles gives a favorable balance of high gating efficiency and low performance impact. For $T_{\text{idledetect}} = 16$, we see that the time-based technique provides 15% bit cycles in SLEEP mode at a performance loss of 2%. In comparison, the CoDeL assisted technique gives 51% more SLEEP mode bit cycles at about the same performance loss. For $T_{\text{idledetect}} = 11$ we find the largest variation where the CoDeL assisted scheme provides 58% more bit cycles in SLEEP mode than the time-based technique.

VII. CONCLUSION

Test circuits, implemented using the CoDeL platform, were examined to determine the expected savings that can be achieved from power gating individual registers, and the associated performance impact. It was found that a CoDeL assisted power gating scheme provides up to 58% more bit cycles of SLEEP mode with about the same performance loss.

In this paper, we have introduced a methodology for implementing efficient power gating using the CoDeL platform. Using the ideas presented we hope to enhance the CoDeL design environment and fully automate the process of power gating in VLSI circuits.

REFERENCES