Automated Power Gating of Registers Using CoDeL and FSM Branch Prediction

Department of Electrical and Computer Engineering
University of Victoria
Victoria, B.C., Canada
{nagarwal,nikitas}@ece.uvic.ca

Abstract. In this paper, we use the CoDeL hardware design platform to 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. We propose a static gating method, with very low area overhead, which uses the information available to the CoDeL compiler to predict, at compile time, when the registers can be powered off and when they can be powered on. Static branch prediction is used in the compiler to more intelligently traverse the finite state machine description of the circuit to discover gating opportunities. We compare this static CoDeL based gating method to a dynamic, time-based technique. Using the DSPstone benchmark circuits for evaluation, we find that CoDeL with backward branch prediction gives the best overall combination of gating potential and performance, resulting in 22% bit cycles saved at a performance loss of 1.3%. Compared to the dynamic time-based technique, this method gives 52% more power gated bit cycles, without any additional performance loss.

1 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 voltage is reduced, by scaling the CMOS technology past sub 100nm, an exponential growth in subthreshold leakage current is seen [1]. 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 [2]. 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) [3, 4] or functional units [5, 6].

Here we examine how power gating techniques can also be used effectively for the reduction of leakage power in low level design. To allow us to efficiently detect and utilize idle periods we use the CoDeL design platform. CoDeL (Controller Description Language) [7, 8] 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.

In [9] we examine the potential of power gating registers using a time-based technique and show that a CoDeL assisted technique can significantly increase gating efficiency. However, a CoDeL assisted time-based technique can be expensive in terms of logic area. In the work reported here, we explore a set of purely static gating techniques, which require very little area overhead. These techniques use static branch prediction to increase gating efficiency by reducing mispredictions of future register usage. In addition, here we use CoDeL to predict when a "wakeup" is needed, reducing the performance penalty incurred while waiting for the supply voltage to be restored to a register.

2 Power Gating

Power gating of a circuit block is performed by using an appropriate header or footer transistor [6]. 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) [10] retention registers can be used (see figure 1). 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. 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 ($T_{idledetect}$) to make the decision to engage gating. From T_2 the supply voltage begins to drop. At 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_4 to T_5 the 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.

[1] Our model here follows the description presented in [6].

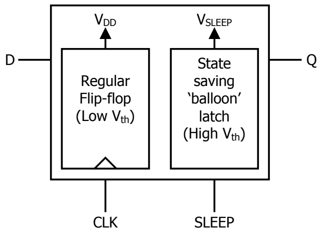

V_DD
V_SLEEP
D
Q
Regular Flip-flop (Low V_th)
State saving 'balloon' latch (High V_th)
CLK
SLEEP

Fig. 1. MTCMOS register

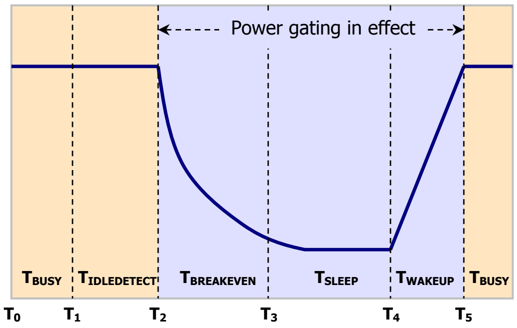


Fig. 2. Voltage during power gating phases

3 Gating Methods

3.1 Time-Based Power Gating

A simple technique to power gate circuit components is to dynamically observe their state and initiate power gating when a sufficient number of idle cycles are detected. Techniques such as this have been used for cache memories [3] and show significant leakage savings with minimal performance impact.

To implement this technique, each circuit component needs to have state machine logic similar to the one shown in figure 3. 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 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. Further, we only consider the savings while in the SLEEP state. There may be some additional power savings in the WAKEUP state, which we conservatively do not include.

According to this framework, we see that our results are dependent on three parameters: $T_{idledetect}$, $T_{breakeven}$, and T_{wakeup}. $T_{breakeven}$ is the time it takes to overcome the energy overhead of gating a unit. T_{wakeup} is the overhead of restoring the power to a unit. The parameters $T_{breakeven}$ and T_{wakeup} are a function of the VLSI technology and thus can not be controlled by circuit design.

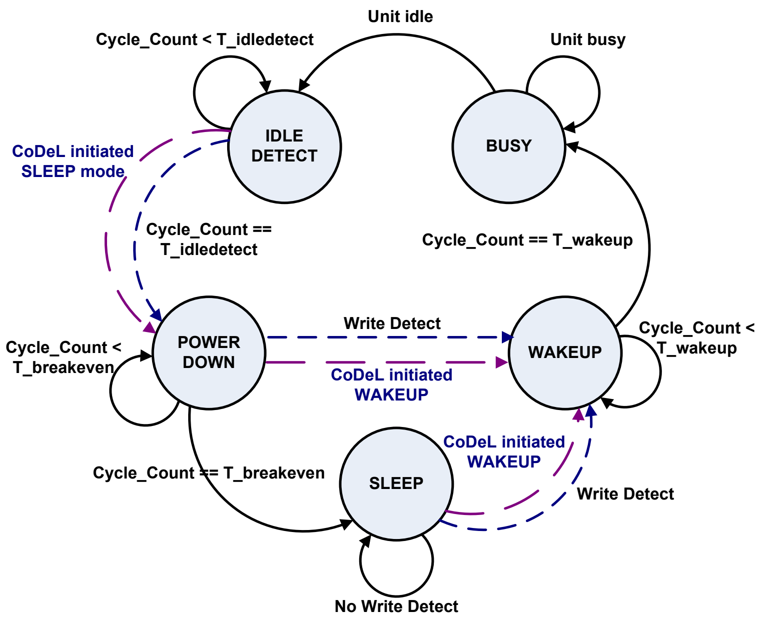


Fig. 3. Gating logic. The short dashed line is used for time-based gating. The long dashed transition line is used for CoDeL initiated power gating. Both dashed lines are used for CoDeL assisted time-based gating.

The $T_{idledetect}$ parameter, however, can be controlled to effect the aggressiveness of the power gating mechanism.

To implement this scheme each register would require a controller to count the idle cycles, and logic to detect a new value being written to the register. This logic is expensive in terms of area and power, and therefore motivates an alternative method of initiating power gating.

3.2 CoDeL Initiated Power Gating

The CoDeL platform [8] 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. For each register, at compile time, CoDeL iterates through each state of the state machine implementation of the circuit and looks ahead $T_{idledetect}$ states to determine if there are any potential writes to the register. If there is no write to the register in the next possible $T_{idledetect}$ states, a power off (SLEEP) suggestion is noted for the gating control logic. If during the next $T_{idledetect}$ possible states the register is written, a power off suggestion is not made. As with the time-based technique, the $T_{idledetect}$ parameter is chosen a priori, and is the same for all registers of the circuit under design.

To more efficiently wake up the registers, CoDeL performs a look ahead and prematurely powers up the register in anticipation of a write. This reduces the performance penalty normally incurred in waiting for a power up. For each register, at compile time, CoDeL examines each state of the state machine implementation of the circuit and looks ahead T_{wakeup} states to determine if there are any potential writes to the register. If there is a write to the register in the next possible T_{wakeup} states, a power on (WAKE) suggestion is noted. Otherwise, a power on suggestion is not made. For example, referring to figure 4(a), for $T_{idledetect} = 3$ and $T_{wakeup} = 1$, a sleep suggestion will be generated at state S2, while a WAKE suggestion will be generated at state S10.

CoDeL initiated gating corresponds to a static environment where only suggestions made by CoDeL can initiate power gating (long dashed line in figure 3). The wakeup mechanism is triggered by a CoDeL suggestion or a detected write.

To implement this static gating scheme only combinational logic is needed. The current state is used to generate the desired SLEEP and WAKE signals to power down and power up the register. Some sequential logic may be needed to generate the `AWAKE` signal, which indicates that the register is powered up and ready for use. This allows CoDeL to stall the register write until the AWAKE signal is asserted.

3.3 CoDeL Assisted Time-Based Power Gating

In CoDeL assisted time-based gating, the decision to initiate gating is still dependent on a streak of idle cycles as in the time-based technique (short dashed line in figure 3). In many cases, however, based on CoDeL's suggestion (long dashed line in figure 3), gating can be initiated prematurely without waiting for the full $T_{idledetect}$ cycles[2]. Also, based on CoDeL's suggestion, wakeups are initiated in anticipation of a register write to reduce the performance penalty.

The implementation of this gating scheme is the most complex as it requires the circuit features of the static and dynamic gating methods.

4 FSM Branch Prediction

CoDeL's gating and wakeup suggestions are dependent on a look-ahead search of the FSM description of the circuit to determine whether a register write is performed in the next $T_{idledetect}$ or T_{wakeup} possible states. In performing this search, branches in the state machine are handled in three different ways. The first method uses no branch prediction (figure 4(a)), and therefore searches all possible state paths. The second method uses static forward branch prediction and assumes that a branch to the furthest state forward is taken (figure 4(b)). The third method uses static backward branch prediction and assumes that a branch to the furthest state backward is taken (figure 4(c)).

[2] The value of $T_{idledetect}$ used for the CoDeL and time-based parts is the same.

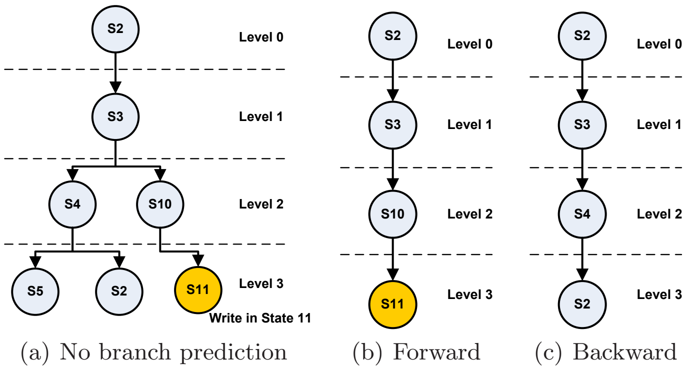


Fig. 4. States look-ahead to determine possible writes. $T_{idledetect} = 3$.

5 Evaluation Framework

To evaluate the power gating methods we use the the DSP kernel benchmarks from the DSPstone suite [11]. 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 (FXU) 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. 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.

6 Results

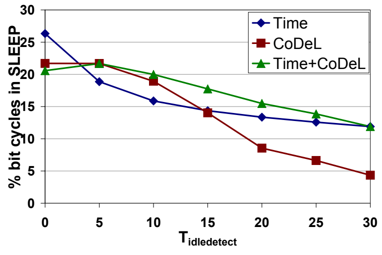
We examine the effects of $T_{idledetect}$, $T_{breakeven}$ and T_{wakeup} on the power gating ability and performance of the circuit. The power gating effectiveness is determined by the percentage of bit cycles in the SLEEP state. It is computed as

$$\frac{\sum_{i=0}^{N} \text{len}(R_i) \cdot (\text{cycles in SLEEP state})_i}{\text{total cycles executed} \cdot \sum_{i=0}^{N} \text{len}(R_i)} \cdot 100\%, \tag{1}$$

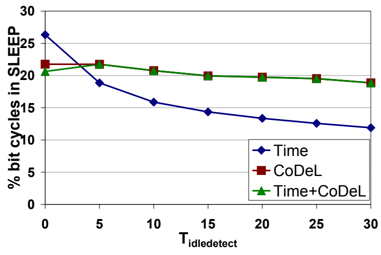
where N is the number of registers, R_i is the ith register and $\text{len}(R_i)$ is the bit width of the ith register. The performance impact of gating is computed as the number of additional clock cycles needed when power gating is introduced. This is computed as

$$\frac{\text{total cycles executed without gating}}{\text{total cycles executed with gating}} \cdot 100\%. \tag{2}$$

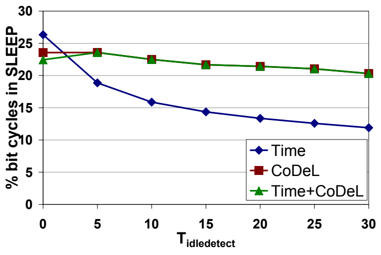
The results presented here are an arithmetic average of the results obtained for the 14 DSPstone kernels.



(a) No branch prediction.



(b) Forward prediction.



(c) Backward prediction.

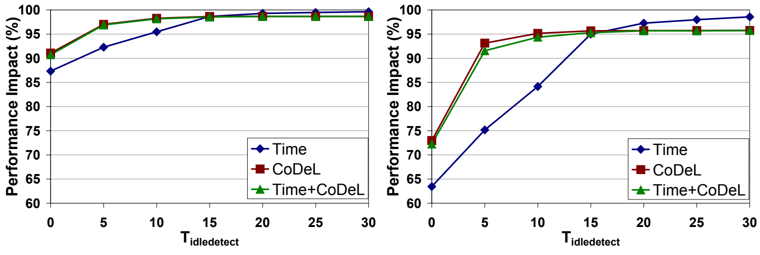
Fig. 5. Gating effectiveness with $T_{wakeup} = 2$. Results averaged over $T_{breakeven} = 5, 10, 15, 20$.

Figure 5 presents the gating effectiveness using the three methods presented and the different branch prediction schemes. From figure 5(a) we see that when no branch prediction is used, the CoDeL based gating schemes perform poorly for larger values of $T_{idledetect}$. This is because since all possible branches are searched to find a write, many more writes are predicted than actually occurring resulting in missed gating opportunities. This is exacerbated in the case of only CoDeL initiated gating, since there is no help from time-based gating to reclaim some of the lost gating opportunities.

Examining the branch prediction schemes (figures 5(b) and 5(c)) we see that the CoDeL based and the CoDeL assisted time-based schemes significantly outperform the time-based technique. For $T_{idledetect} = 30$, the CoDeL schemes provide 59% more gated bit cycles than the time-based technique. It is interesting to note that for $T_{idledetect} \geq 5$, both CoDeL based schemes exhibit the same savings. This means that the dynamic decision criteria in the CoDeL assisted time-based technique presents no new gating opportunities in comparison to the purely static CoDeL scheme.

Comparing the forward and backward branch prediction schemes we find that the backward prediction results in more gating opportunities resulting in 8% more gated bit cycles. This means that more backward branches are taken in our designs than the forward branches.

Figure 6 shows the performance impact for different values of T_{wakeup}. Backward branch prediction is used here for the CoDeL schemes since it provides



(a) $T_{wakeup} = 2$ (b) $T_{wakeup} = 8$

Fig. 6. Performance impact (see equation 2). Backward branch prediction used for CoDeL schemes. Results averaged over $T_{breakeven} = 5, 10, 15, 20$.

the best gating potential as compared to forward and no branch prediction. In all cases, we see that the CoDeL schemes outperform the time-based technique for lower values of $T_{idledetect}$ (less than 15), while for larger $T_{idledetect}$, the time-based technique dominates. This is because the time-based technique gates registers less frequently for larger $T_{idledetect}$, and thus results in fewer wakeup procedures resulting in lower performance loss. Even for these larger $T_{idledetect}$ values, however, the difference in performance for the time-based and CoDeL schemes is very small (less than 3%). But the number of sleep cycles gained with the CoDeL method far exceeds those of the time-based method by more than 60%. Comparing the two CoDeL schemes we see they provide roughly the same performance. Expectedly, as the value of T_{wakeup} increases, performance decreases as more cycles are spent in waiting for the power to be restored.

In figure 7 we are able to more clearly see the entire design space consisting of the various techniques. We have also included the CoDeL based scheme used in [9] for comparison. The solid curves indicate results using a static gating method where the area overhead is extremely low. The dashed curves are for methods which employ a dynamic scheme resulting in significant overhead.

Common in all performance results, we see that lower values of $T_{idledetect}$ 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.

We see that the time-based technique provides very poor overall gating effectiveness and performance. Comparing the branch prediction schemes, we see that the performance loss of the CoDeL scheme with no branch prediction is the lowest, but it also results in the poor gating potential. The backward branch prediction provides better performance than the forward branch prediction since it is more accurately able to predict wakeups.

We find that CoDeL, with backward branch prediction, is able to provide an excellent compromise of high gating effectiveness and low performance loss. For $T_{idledetect} = 15$ we find that the CoDeL scheme with backward branch prediction provides 52% more bit cycles in SLEEP mode than the time-based technique for

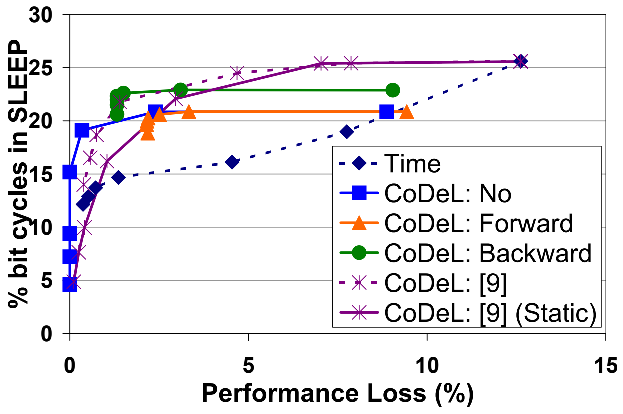


Fig. 7. Gating effectiveness vs performance loss for $T_{breakeven} = 10$ and $T_{wakeup} = 2$. $T_{idledetect}$ varies from 30 to 0 from left to right.

the same approximate performance loss of 1.3%. CoDeL with forward branch prediction provides a relatively poor combination of gating effectiveness and performance. This is due to the high rate of misprediction with this method. CoDeL with no branch prediction provides lower gating effectiveness but provides excellent performance for larger values of $T_{idledetect}$, and thus may be useful in cases where high performance is critical.

In figure 7 we provide two results based on the CoDeL scheme presented in [9]. The dynamic scheme ("CoDeL: [9]") is a CoDeL assisted time-based technique with no branch prediction and no "wakeup" prediction. Since we have not factored in overhead, we see that it performs quite well. Due to the dynamic nature of this method, however, it entails significant overhead. Assuming a three bit counter to count the elapsed idle cycles, our preliminary results suggest that the overhead reduces the SLEEP mode bit cycles by 18% (below what is indicated in figure 7). This overhead considerably reduces the apparent effectiveness of all dynamic techniques presented (dashed curves). The static version of this scheme ("CoDeL: [9] (Static)") is a modified version of the method presented in [9]. This method is the same as the "CoDeL: No" method without "wakeup" prediction. We see that the lack of "wakeup" prediction causes increased performance loss.

7 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 initiated power gating scheme with static backward branch prediction provides an overall superior combination of high gating effectiveness and low performance loss. For high performance applications, CoDeL with no branch prediction (full state space exploration) is the best choice. In both these methods, since the gating decisions are made at compile time, there is very little circuit area overhead. We are currently investigating other branch prediction schemes (including dynamic prediction) which may help in further reducing mispredictions, and thus improve performance without degrading gating effectiveness.

Here, 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. We are also working on more accurately defining the power gating overhead needed. This will allow more accurate break even analysis, and thus allow us to determine the minimum register size that should be power gated.

References

1. Kim, N.S., Austin, T., Blaauw, D., Mudge, T., Flautner, K., Hu, J.S., Irwin, M.J., Kandemir, M., Narayanan, V.: Leakage current: Moore's law meets static power. Computer 36(12), 68–75 (2003)
2. Powell, M., Yang, S.H., Falsafi, B., Roy, K., Vijaykumar, T.N.: Gated-vdd: a circuit technique to reduce leakage in deep-submicron cache memories. In: ISLPED 2000, pp. 90–95. ACM Press, New York (2000)
3. Flautner, K., Kim, N.S., Martin, S., Blaauw, D., Mudge, T.: Drowsy caches: simple techniques for reducing leakage power. In: ISCA 2002, pp. 148–157. IEEE Computer Society Press, Los Alamitos (2002)
4. Liao, W., Basile, J.M., He, L.: Microarchitecture-level leakage reduction with data retention. IEEE Transactions on VLSI Systems 13(11), 1324–1328 (2005)
5. Rele, S., Pande, S., Onder, S., Gupta, R.: Optimizing static power dissipation by functional units in superscalar processors. In: Horspool, R.N. (ed.) CC 2002 and ETAPS 2002. LNCS, vol. 2304, pp. 261–275. Springer, Heidelberg (2002)
6. Hu, Z., Buyuktosunoglu, A., Srinivasan, V., Zyuban, V., Jacobson, H., Bose, P.: Microarchitectural techniques for power gating of execution units. In: ISLPED '04, pp. 32–37. ACM Press, New York (2004)
7. Sivakumar, R., Dimakopoulos, V., Dimopoulos, N.: CoDeL: A rapid prototyping environment for the specification and automatic synthesis of controllers for multiprocessor interconnection networks. In: SAMOS III, pp. 58–63 (July 2003)
8. Agarwal, N., Dimopoulos, N.: Power efficient rapid hardware development using CoDeL and automated clock gating. In: ISCAS 2006 (May 2006)
9. Agarwal, N., Dimopoulos, N.: Towards automated power gating of registers using CoDeL. In: ISCAS 2007 (May 2007)
10. Mutoh, S., Douseki, T., Matsuya, Y., Aoki, T., Shigematsu, S., Yamada, J.: 1-v power supply high-speed digital circuit technology with multithreshold-voltage cmos. IEEE Journal of Solid-State Circuits 30(8), 847–854 (1995)
11. Zivojnovic, V., Martinez, J., Schläger, C., Meyr, H.: DSPstone: A DSP-oriented benchmarking methodology. In: ICSPAT 1994 (October 1994)
12. Bromley, J.: Synthesizable vhdl fixed point arithmetic package (2006), `http://www.doulos.com/knowhow/vhdl_designers_guide/models/fp_arith/`