Power Optimization using Dynamic Power Management

José C. Monteiro
IST / INESC
Rua Alves Redol, 9
1000 Lisboa, Portugal
jcm@inesc.pt

Abstract

Power dissipation has recently emerged as one the most critical design constraints. A wide range of techniques has already been proposed for the optimization of logic circuits for low power. Power management methods are among the most effective techniques for power reduction. These methods detect periods of time during which parts of the circuit are not doing useful work and shut them down by either turning off the power supply or the clock signal.

Several methods have been presented that perform shutdown on a clock-cycle base. Depending on the input conditions at the beginning of a clock-cycle, the clock driving some of the registers in the circuit can be inhibited, thus reducing the switching activity in the fanout of those registers. These techniques are referred to as data-dependent or dynamic power management techniques.

In this tutorial we will describe some of the most representative data-dependent power management techniques that have recently been proposed, namely: precomputation, guarded evaluation, gated-clock finite state machines (FSM)'s and FSM decomposition. Each of these techniques uses a different approach to identify the input conditions for which the circuit (or part of) can be disabled. These techniques are put into perspective and recent results are discussed.

1 Introduction

Power consumption has become a primary concern in the design of integrated circuits. Two independent factors have contributed for this. On one hand, low power consumption is essential to achieve longer autonomy for portable devices. On the other hand, increasingly higher circuit density and higher clock frequencies are creating heat dissipation problems, which in turn raise reliability concerns and lead to more expensive packaging.

During normal operation of well designed CMOS circuits, power consumption is determined by the switching activity in the circuit [5]. Under a generally accepted simplified model, the power dissipation at the output of a gate g in a logic circuit is given by:

$$P_g = \frac{1}{2} \cdot C_g \cdot V_{DD}^2 \cdot f \cdot N_g \qquad (1)$$

where P_g denotes power, V_{DD} the supply voltage, and f the clock frequency. C_g represents the capacitance gate g is driving and N_g is the switching activity at the output of gate g, *i.e.*, the average number of gate output transitions per clock cycle. The product $C_g \cdot N_g$ is called *switched capacitance*.

In the last few years, research on techniques for low power at various levels of design has intensified. Most power optimization techniques at different levels of abstraction target the minimization of the switched capacitance in the circuit [7].

Techniques based on disabling the input/state registers when some input conditions are met have been proposed and shown to be among the most effective in reducing the overall switching activity in sequential circuits. The disabling of the input/state registers is decided on a clock-cycle basis and can be done either by using a register load-enable signal or by gating the clock. A common feature in these methods is the addition of extra circuitry that is able to identify input conditions for which some or all of the input/state registers can be disabled. This class of techniques is sometimes referred to as *logic level* or *dynamic power management*.

In this paper, we review these logic level power management techniques. The different approaches to identify the input conditions for which the circuit (or part of) can be disabled are analyzed and compared.

2 Power Management

So called power management techniques shutdown blocks of hardware for periods of time in which they are not producing useful data. Shutdown can be accomplished by either turning off the power supply or by disabling the clock signal. A system-level approach is to identify idle periods for entire modules and turn off the clock lines for these modules for the duration of the idle periods ([5], Chapter 10).

However, there still is no real methodology for system level power management. It is up to the designer to devise a strategy for power management for a particular project. One first attempt at the register-transfer level is given in [8]. A scheduling algorithm that tries to maximize the shutdown period of execution units is proposed. Given throughput constraints and the number of execution units available, operations are scheduled such that those that generate controlling signals are computed first, thus indicating the flow of data through the circuit. Power is saved by only activating hardware modules involved in computing the final result and shutting down all other modules.

In contrast, a few techniques has been proposed at the gate level. These techniques are based on disabling the input/state registers when some input conditions are met, either by using a register load-enable signal or by gating the clock. In this situation there will be zero switching activity in the logic driven by input signals coming from the disabled registers. The main difference from system-level power management is that the shutdown of hardware is decided on every clock cycle, hence the name *dynamic power management*.

In order to decide whether to load or not new values into the registers, some extra logic has to be added to the original circuit. Naturally, this is redundant circuitry, increasing both area and power dissipation. In fact, logic-level power management techniques tradeoff some area for lower power consumption. The argument is that power is becoming the main constraint and that area is no longer critical.

Even if area is of no concern, this extra logic, which is active all the time, increases power consumption. The power savings obtained by shutting down the registers must compensate for this overhead. The more complex this logic is, the larger the power overhead. Thus, on one hand, the more input conditions that are targeted for register shutdown, the larger the period of time during which the original circuit is being powered down. On the other hand, the larger the power penalty from the extra logic. In general, there is an optimum size for the extra logic and the goal of the different power management techniques at the logic level is to find this optimum.

The motto of logic-level power management is to have a small amount of logic that is active most of the time, but that is able to shutdown a much larger circuit during that time.

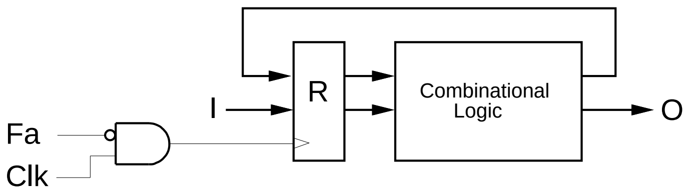

Figure 1. Example of gated-clock FSM.

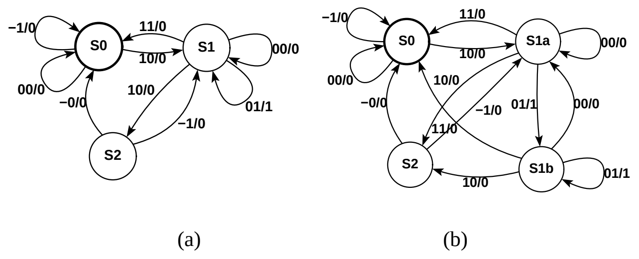

Figure 2. FSM transformation: (a) Mealy, (b) Locally Moore.

3 Dynamic Power Management Techniques

The most representative power management techniques that have been proposed are described in this section.

3.1 Exploiting Self-Loops in FSMs

The *gated-clock finite state machines* (FSMs) approach [4], depicted in Figure 1, is based on identifying self-loops in a Moore FSM. If the FSM enters a state with a self-loop, the signal F_a is asserted and the clock is turned off. In this situation, the inputs to the combinational logic block do not switch, and thus there is virtually zero power dissipation in that block. When the input values cause the FSM to make a state transition, the clock signal is again enabled and the circuit resumes normal operation.

In order to achieve significant power savings, the primary input lines have to be blocked. Thus, this procedure is only applicable to FSMs where the output lines do not depend directly on the primary input lines (*i.e.*, Moore FSMs). This is a severe limitation of the method.

To overcome this problem, techniques to locally transform a Mealy FSM into a Moore FSM are given in [4]. Consider the state transition graph (STG) of the Mealy FSM of Figure 2(a). The self-loops of state `s0` can be used directly since the output assumes the same value for all of them. However, the output depends on the input values for

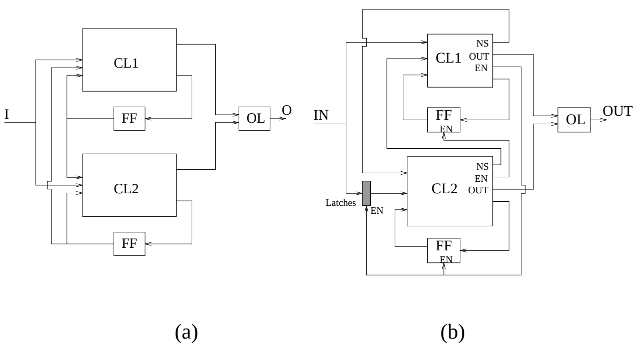

Figure 3. FSM general decomposition: (a) traditional, (b) with power management.

the self-loops of state `s1`, therefore no clock-gating is possible. The STG can be transformed into an equivalent one as shown in Figure 2(b), where state `s1` has been split into states `s1a` and `s1b`. Clock-gating is now possible for the self-loops in each of these states.

3.2 FSM Decomposition

Decomposition of finite state machines targeted for low power has been recently proposed [9, 6]. The basic idea is to decompose the STG of original finite state machine into two coupled STGs that together have the same functionality as the original machine. Except for transitions that involve going from one state in one sub-FSM to a state in the other, only one of the sub-FSMs needs to be clocked.

The techniques described in [9] and [6] differ both in the way the partitioning of the states is performed and in the structure of the final circuit.

The technique of [9] follows the standard general decomposition structure, as shown in Figure 3(a). The selection of the states is such that only a small number is selected for one of the sub-FSMs. This selection consists in searching for a small cluster of states such that summation of the probability of transitions between states in the cluster is high and with a very low probability of transition to and from states outside of the cluster. The aim is to have a small sub-FSM that is active most of the time, disabling the larger sub-FSM. The reason for requiring a small number of transitions to/from the other sub-FSM is that this corresponds to the worst situation when both sub-FSMs are active.

The power optimized structure is shown is Figure 3(b). Each sub-FSM has an extra output that disables the state registers of the other sub-FSM. This extra output is also used to stop transitions at the inputs of the large sub-FSM. To avoid the area/power overhead incurred by adding latches, and since when this technique is effective the small sub-FSM is in operation most of the time, the inputs to the small sub-FSM are not filtered.

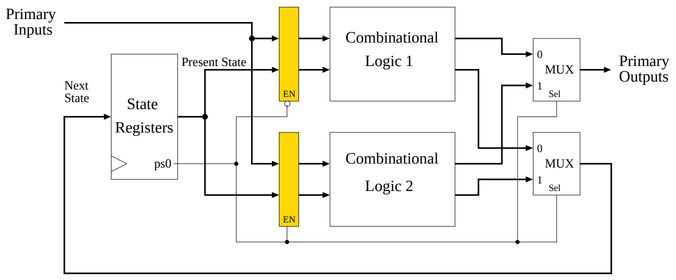

Figure 4. Partitioning the encoding of states.

The target structure of the decomposition strategy of [6] is not based on two different sub-FSMs. There is only one set of state registers, but the combinational logic block they drive is decided from the encoding of the present state. The structure for a 2-way decomposition is depicted in Figure 4.

The states selected for sub-FSM 1 are all encoded such that the present state line 0 (`ps0`) is always 0. Conversely, all states in sub-FSM 2 have `ps0`=1. Then, for all transitions within sub-FSM 1, only the combinational logic block 1 will be active and vice-versa.

In [6], the cost function used in the partitioning of the states tries to maximize the probability of transitions within one sub-FSM (with no constraint on the number of states) and to minimize an estimate of the overhead involved.

These methods for FSM decomposition can be seen as an extension of the gated-clock for FSM self-loops approach. In FSM decomposition the cluster of states that are selected for the one sub-FSM can be considered as a “super-state” and then transitions between states in this cluster are no more than self-loops in this “super-state”. The decision as to what states make up the “super-block” basically gives the opportunity to maximize the number of self-loops. Still, the overheads of these techniques can be quite different.

3.3 Precomputation

The *precomputation* method was first proposed in [1, 10], and has been extended to be applicable to combinational modules in [12]. In this method, a simple combinational circuit (the precomputation logic) is added to the original circuit. Under certain input conditions, the precomputation logic disables the loading of all or a subset of the input registers. Under these input conditions, no power is dissipated in the portions of the original circuit with only disabled registers as inputs.

The basic architecture of this method is shown in Figure 5. A is the original combinational logic. g_1 and g_2 constitute the precomputation logic and are designed such that they are a function of a subset of the inputs to A. Power

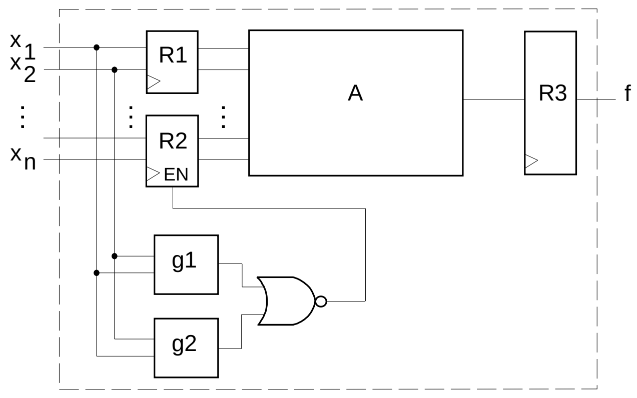

Figure 5. Precomputation architecture.

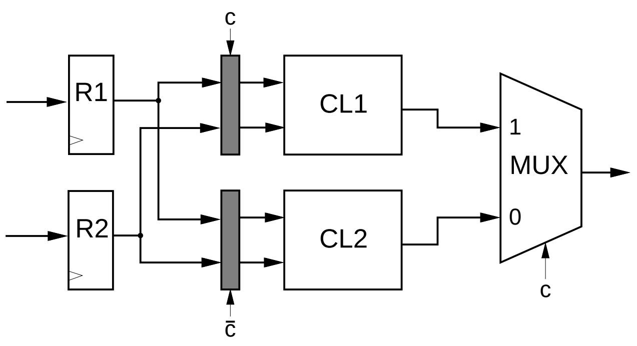

Figure 6. Example of guarded-evaluation.

dissipation in the original circuit A is reduced when the outputs of either g_1 or g_2 evaluate to 1.

The choice and the number of inputs to use for the g_1 and g_2 functions is critical. The more inputs used, the highest the probability the precomputation logic will be active, thus disabling logic in block A. However, the size of the precomputation logic increases, a circuitry overhead that is active all the time, thus offsetting the gains obtained by disabling A a larger fraction of the time.

Once the number of inputs to the precomputation logic is fixed, the input selection is based on the probability that the outputs can be computed without the knowledge of a specific input, *i.e.*, the size of the observability don't-care set (ODC):

$$\mathrm{ODC}_i = f_{x_i} \cdot f_{\overline{x_i}} + \overline{f}_{x_i} \cdot \overline{f}_{\overline{x_i}} \qquad (2)$$

Inputs with lowest $prob(\mathrm{ODC}_i)$ are selected to be in the precomputation logic.

Guarded evaluation [13] identifies cones internal to the circuit that can be shut down under certain input conditions. In the process, it creates new transition barriers (guards) in the form of additional latches or OR/AND gates. Instead of adding the precomputation logic to generate the clock disabling signal, guarded-evaluation uses signals already existing in the circuit.

This technique is depicted in Figure 6 where the transition guards are shown in grey. In this figure, only one of the combinational logic blocks, either CL_1 or CL_2, will have transitions at the inputs, defined by the the value of the select signal c. Therefore, there will be no power consumption in the other logic block.

Yet another similar technique, named *Computational Kernel Extraction* has been proposed recently [3]. This method is applicable to Finite State Machines (FSM) and is based on constructing an alternative logic block for a subset of the next state function. Power is reduced by using this simpler block and shutting down the original circuit whenever a transition in the selected subset is activated.

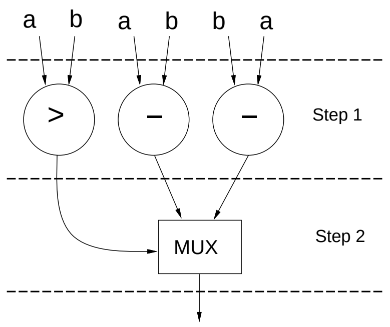

Figure 7. CDFG and schedule for $|a - b|$ using 2 control steps.

3.4 Scheduling to Enable Power Management

One first attempt at dynamic power management at the register-transfer level was proposed in [8]. A scheduling algorithm that tries to maximize the shutdown period of execution units is proposed. Given throughput constraints and the number of execution units available, operations are scheduled such that those that generate controlling signals are computed first, thus indicating the flow of data through the circuit. Power is saved by only activating hardware modules involved in computing the final result and shutting down all other modules.

As an example, say we want to compute $|a - b|$ and use the approach shown in Figure 7. Assume that one control step is required for each of the three operations ($-$, $>$ and MUX). The only precedence constraint for this example is that the multiplexor operation can only be scheduled after all other three operations. If only two control steps are allowed to compute $|a - b|$, then necessarily the operations $a > b$, $a - b$ and $b - a$ (we need two subtractors) have to be executed in the first control step and the multiplexor in the second control step as indicated in Figure 7. If instead we are allowed three control steps, only one subtractor is

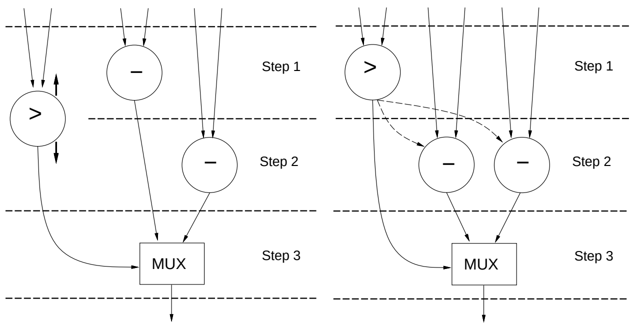

Figure 8. Schedule for $|a-b|$ using 3 control steps.

required if the operations $a-b$ and $b-a$ are scheduled in different control steps, one in the first control step and the other in the second. Operation $a>b$ can be scheduled in any of these two control steps and the multiplexor will be in the third control step, as shown in Figure 8(a). In both these cases, *both* $a-b$ and $b-a$ are computed although only the result of one of them is eventually used. This is obviously wasteful in terms of power consumption.

The scheduling algorithm proposed in [8] attempts to assign operations involved in determining the data flow (in this case $a>b$) as early as possible in the initial control steps, thus indicating which computational units are needed to obtain the final result. Only those units that eventually get used are activated. The algorithm chooses a schedule only if the required throughput and hardware constraints are met. In other words, the algorithm explores any available slack to obtain a power manageable architecture. In the previous example, assuming three control steps, the scheduling algorithm will assign $a>b$ to the first control step and $a-b$ and $b-a$ to the second. Depending on the result of $a>b$, only the inputs to one of $a-b$ and $b-a$ will be loaded, thus no switching activity will occur in the subtractor whose result is not going to be used. This situation is shown in Figure 8(b), where the dashed arrows indicate that the execution of the '$-$' operations depends on the result of the comparator.

4 Results and Discussion

Results for some of the techniques described in the previous section are presented in Table 1, taken from the respective references. The area overhead is given under column %A ($=(\frac{\mathrm{A_{opt}}}{\mathrm{A_{orig}}}-1)\times 100$) and the power savings under column %P ($=(1-\frac{\mathrm{P_{opt}}}{\mathrm{P_{orig}}})\times 100$). R is the ratio:$\mathrm{R}=\frac{\mathrm{A_{opt}}\times\mathrm{P_{opt}}}{\mathrm{A_{orig}}\times\mathrm{P_{orig}}}$ which can be used as a measure of the effectiveness of the optimization procedure. Entries with '–' indicate that either the technique is not effective for this example or the result is not available.

From this table, several observations can be made. Precomputation [1] seems to be able to reduce power dissipation for most circuits, yet the power reduction is not very significant for some. A reduction below 10% may be in the noise of the error of the estimation tool. It is the technique with the smallest area overhead.

The applicability of the self-loop technique [2] is more restrict, it works well mainly for reactive systems. Nevertheless, power reduction can be achieved for circuits for which the FSM decompositions approaches are ineffective, mostly because of the smaller circuitry overhead.

The FSM decomposition approaches [9, 6] allow for the largest power savings. At the same time, these techniques also present the largest area overhead. This overhead makes these techniques not applicable to small FSMs. The method using the traditional FSM decomposition [9] has some power gain over the partitioning of state encodings [6] because of the strategy to find a small sub-FSM that is the one active most of the time. The method of [6] cannot be tuned for this since it uses the same registers for both sub-FSMs. However, this sharing reduces the area overhead.

One advantage of the precomputation method is that it works at the circuit level. For the methods of [2, 6, 9], the STG of the FSM, if not given, has to be extracted from the circuit in order for the power optimization procedure to be performed. This severely limits the size of the sequential circuit to which these techniques can be applied. Research work to extend these techniques so that the STG can be handled implicitly is currently under way.

A final word on testing. Since all these methods add redundant logic to the original circuit, the testing of the power managed circuits is made more difficult because of all the redundant faults. Some initial work on solving this problem has been presented in [11].

References

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou. Precomputation-Based Sequential Logic Optimization for Low Power. *IEEE Transactions on VLSI Systems*, 2(4):426–436, December 1994.

[2] L. Benini and G. De Micheli. Transformation and Synthesis of FSMs for Low Power Gated Clock Implementation. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 21–26, April 1995.

[3] L. Benini, G. De Micheli, A. Lioy, E. Macii, G. Odasso, and M. Poncino. Computational Kernels and their Application to Sequential Power Optimization.

Table 1. Power reduction and area overhead for the different techniques.

Circuit Name	Precomputation [1]			Self-Loops in FSM [2]			Partition of Encoding [6]			FSM Decomposition [9]		
	%A	%P	R	%A	%P	R	%A	%P	R	%A	%P	R
bbara	6	11	0.94	24	49	0.62	–			–		
bbsse	0	7	0.93	15	2	1.13	48	15	1.26	–		
cse	1	37	0.64	–			32	39	0.80	58	43	0.90
ex1	2	10	0.92	–			20	48	0.62	38	20	1.10
keyb	3	10	0.93	14	11	1.01	46	18	1.20	27	38	0.79
planet	–			–			14	44	0.64	32	51	0.65
s1488	2	34	0.68	–			18	59	0.48	-1	80	0.20
s1494	2	32	0.69	–			22	26	0.89	-3	77	0.22
s298	18	12	1.03	0	10	0.90	–			–		
s420	–			11	18	0.91	–			–		
s510	2	3	0.99	–			20	33	0.80	46	55	0.66
sand	1	20	0.81	–			–			43	10	1.29
scf	10	10	0.99	–			–			47	48	0.76
sse	0	7	0.94	–			48	15	1.26	40	50	0.70
styr	0	16	0.84	–			23	35	0.80	35	36	0.86
tbk	0	4	0.96	–			-11	24	0.68	-22	46	0.42
Average	2.9	15.2	0.88	12.8	18.0	0.91	25.5	32.4	0.86	28.3	46.2	0.71

In *Proceedings of the* 35^{th} *Design Automation Conference*, June 1998.

[4] L. Benini, P. Siegel, and G. De Micheli. Automatic Synthesis of Low-Power Gated-Clock Finite-State Machines. *IEEE Transactions on Computer-Aided Design*, 15(6):630–643, June 1996.

[5] A. Chandrakasan, T. Sheng, and R. Brodersen. Low-Power CMOS Digital Design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, April 1992.

[6] S-H. Chow, Y-C. Ho, and T. Hwang. Low Power Realization of Finite State Machines – A Decomposition Approach. *ACM Transactions on Design Automation of Electronic Systems*, 1(3):315–340, July 1996.

[7] S. Devadas and S. Malik. A Survey of Optimization Techniques Targeting Low Power VLSI Circuits. In *Proceedings of the* 32^{nd} *Design Automation Conference*, pages 242–247, June 1995.

[8] J. Monteiro, S. Devadas, P. Ashar, and A. Mauskar. Scheduling Techniques to Enable Power Management. In *Proceedings of the* 33^{rd} *Design Automation Conference*, pages 349–352, June 1996.

[9] J. Monteiro and A. Oliveira. Finite State Machine Decomposition for Low Power. In *Proceedings of the* 35^{th} *Design Automation Conference*, pages 758–763, June 1998.

[10] J. Monteiro, J. Rinderknecht, S. Devadas, and A. Ghosh. Optimization of Combinational and Sequential Logic Circuits for Low Power Using Precomputation. In *Proceedings of the 1995 Chapel Hill Conference on Advanced Research on VLSI*, pages 430–444, March 1995.

[11] J. Monteiro and J. Silva. Testability Analysis of Circuits using Data-Dependent Power Management. In *Proceedings of the IX IFIP International Conference on Very Large Scale Integration*, pages 353–364, August 1997.

[12] A. Mota, J. Monteiro, and A. Oliveira. Power Optimization of Combinational Modules Using Self-Timed Precomputation. In *Proceedings of the International Symposium on Circuits and Systems*, May 1998.

[13] V. Tiwari, P. Ashar, and S. Malik. Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 221–226, April 1995.