Power Minimization Techniques at the RT-Level and Below

Afshin Abdollahi and Massoud Pedram

Dept. of Electrical Engineering

University of Southern California

Los Angeles, CA 90089 U.S.A.

Abstract – *Power consumption and power-related issues have become a first-order concern for most designs and loom as fundamental barriers for many others. And, while the primary method used to date for reducing power has been supply voltage reduction, this technique begins to lose its effectiveness as voltages drop to sub-one volt range and further reductions in the supply voltage begin to create more problems than are solved. Under these circumstances, the process of design and the automation tools required to support that process become the critical success factors. In the last decade, huge effort has been invested to come up with a wide range of design solutions that help solve the power dissipation problem for different types of electronic devices, components and systems. These techniques range from multiple voltage assignment and dynamic voltage scaling, to RTL power management and power-aware sequential logic synthesis, to leakage power reduction techniques. This tutorial paper explains a number of representative low power design techniques from this large set. More precisely, we will describe basic techniques, applicable at RT-level and below, that have proven to hold good potential for power optimization in practical design environments.*

1 Introduction

A dichotomy exists in the design of modern microelectronic systems: they must be low power and high performance, simultaneously. This dichotomy largely arises from the use of these systems in battery-operated portable (wearable) platforms. Accordingly, the goal of low-power design for battery-powered electronics is to extend the battery service life while meeting performance requirements. Unless optimizations are applied at different levels, the capabilities of future portable systems will be severely limited by the weight of the batteries required for an acceptable duration of service. In fixed, power-rich platforms, the packaging cost and power density/reliability issues associated with high power and high performance systems also force designers to look for ways to reduce power consumption. Thus, reducing power dissipation is a design goal even for non-portable devices since excessive power dissipation results in increased packaging and cooling costs as well as potential reliability problems. Meanwhile, following Moore's Law, integrated circuit densities and operating speeds have continued to go up in unabated fashion. The result is that chips are becoming larger, faster, and more complex and because of this, consuming increasing amounts of power.

These increases in power pose new and difficult challenges for integrated circuit designers. While the initial response to increasing levels of power consumption was to reduce the supply voltage, it quickly became apparent that this approach was insufficient. Designers subsequently began to focus on advanced design tools and methodologies to address the myriad of power issues. Complicating designers' attempts to deal with these issues are the complexities – logical, physical, and electrical – of contemporary IC designs and the design flows required to build them.

The established front-end approach to designing for lower power is to estimate and analyze power consumption at the register transfer level (RTL), and to modify the design accordingly. In the best case, only the RTL within given functional blocks is modified, and the blocks re-synthesized. The process is re-iterated until the desired results are achieved. Sometimes, though, the desired power consumption reductions may be achieved only by modifying the overall design architecture. Modifications at this level affect not only power consumption, but also other performance metrics, and may indeed greatly affect the cost of the chip. Thus, such modifications require re-evaluation and re-verification of the entire design. The architectural optimization techniques, however, fall outside the coverage of the present article.

This article reviews a number of representative RTL design automation techniques that focus on low power design. It should be of interest to designers of power efficient devices, IC design engineering managers, and EDA managers and engineers. More precisely, it covers techniques for, sequential logic synthesis, RTL power management, multiple voltage design, and leakage power minimization and control techniques. Interested readers can find wide-ranging information on various aspects of low power design in [1]-[3].

2 Multiple-Voltage Design

Using different voltages in different parts of a chip may reduce the global energy consumption of a design at a rather small cost in terms of algorithmic and/or architectural modifications. The key observation is that the minimum energy consumption in a circuit is achieved if all circuits paths are timing-critical (there is no positive slack in the circuit.) A common voltage scaling technique is thus to operate all the gates on non-critical timing paths of the circuit at a reduced supply voltage. Gates/modules that are part of the critical paths are powered at the maximum allowed voltage, thus, avoiding any delay increase; the power consumed by the modules that are not on the critical paths, on the other hand, is minimized because of the reduced supply voltage. Using different power supply voltages on the same chip of circuitry requires the use of level shifters at the boundaries of the various modules (a level converter is needed between the output of a gate powered by a low V_{DD} and the input of a gate powered by a high $V_{DD,}$ i.e., for a step-up change.) Figure 1 depicts a typical level converter design. Notice that if a gate that is supplied with $V_{DD,L}$ drives a fanout gate at $V_{DD,H}$, transistors N1 and N2 receive inputs at reduced supply and the cross-coupled PMOS transistors do the level conversion. Level converters are obviously not needed for a step-down change in voltage. Overhead of level converters can be mitigated by doing conversions at register boundaries and embedding the level conversion inside the flip flops (see [4] for details.)

A polynomial time algorithm for multiple-voltage scheduling of performance-constrained non-pipelined designs is presented by Raje and Sarrafzadeh in [5]. The idea is to establish a supply voltage level for each of the operations in a data flow graph, thereby, fixing the latency of that operation. The goal is then to minimize the total power dissipation while satisfying the system timing constraints. Power minimization is in turn accomplished by ensuring that each operation will be executed using the minimum possible supply voltage. The proposed algorithm is composed of a loop where, in each iteration, slacks of nodes in the acyclic data flow graph are calculated. Then, nodes with the maximum slack are assigned to lower voltages in such a way that timing constraints are not violated. The algorithm stops when no positive slack exists in the data flow graph. Notice that this algorithm assumes that the Pareto-optimal voltage versus delay curve is identical for all computational elements in the data flow graph. Without this assumption, there is no guarantee that this algorithm will produce an optimal design.

In [6], the problem is addressed for combinational circuits, where only two supply voltages are allowed. A depth-first search is used to determine those computational elements, which can be operated at low supply voltage without violating the circuit timing constraints. A computational element is allowed to operate at $V_{DD,L}$ only is all its successors are operating at $V_{DD,L}$. For example, Figure 2(a) demonstrates a clustered voltage scaling (CVS) solution in which each circuit path starts with $V_{DD,H}$ and switches to $V_{DD,L}$ when delay slack is available. The timing-critical path is shown with thick line segments. Here gray-colored cells are running at $V_{DD,L}$. Level conversion (if necessary) is done in the flip flops at the end of the circuit paths. An extension to this approach is proposed in [7], which is based on the observation that by optimizing the insertion points of level converters, one can increase the number of gates using $V_{DD,L}$ without increasing the number of level converters. This leads to higher power savings. For example, in the CVS solution depicted in Figure 2(a), assume that the path delay from flip-flop FF3 to gate G2 is much longer than that of the path from FF1 to G2. In addition, assume that if we apply $V_{DD,L}$ to G2, then the path from FF3 to FF5 through G2 will miss its target combinational delay i.e., G2 must be assigned a supply level of $V_{DD,H}$. With the CVS approach, it immediately follows that G3 must be assigned $V_{DD,H}$ although a potentially large positive slack remains in the path from FF1 to G2. The situation is the same for G4 and G5. Consequently, the CVS approach can miss opportunities for applying $V_{DD,L}$ to some gates in the circuit. If the insertion point of the level converter LC1 is allowed to move up to the interface between G3 and G2, the gates G3 through G5 can be assigned a supply of $V_{DD,L}$, as depicted in Figure 2(b). The structure shown there is one that can be obtained by the extended CVS (ECVS) algorithm. Both CVS and ECVS assign the appropriate power supply to the gates by traversing the circuit from the primary outputs to the primary inputs in a topological order. ECVS allows a $V_{DD,L}$-driven gate to feed a $V_{DD,H}$ driven gate along with the insertion of a dedicated level converter.

In [8], the authors propose an approach for voltage assignment in combinational logic circuits. First, a lower bound on dynamic power consumption is determined by exploiting the available slacks and the value of the dual-supply voltages that may be used in solving the problem of minimizing dynamic power consumption of the circuit. Next, a heuristic algorithm is proposed for solving the voltage-assignment problem, where the values of the low and the high supply voltages are either specified by the user or fixed to the estimated ones.

In [9], Manzak and Chakrabarti present resource and latency constrained scheduling algorithms to minimize power/energy consumption when the resources operate at multiple voltages. The proposed algorithms are based on efficient distribution of slack among the nodes in the data-flow graph. The distribution procedure tries to implement the minimum energy relation derived using the Lagrange multiplier method in an iterative fashion.

An important phase in the design flow of multiple-voltage systems is that of assigning the most convenient supply voltage, selected from a fixed number of values, to each operation in the control-date flow graph (CDFG). The problem is to assign the supply voltages and to schedule the tasks so as to minimize the power dissipation under throughput/resource constraints. An effective solution has been proposed by Chang and Pedram in [10]. The technique is based on dynamic programming and requires the availability of accurate timing and power models for the macro-modules in a RTL library. A preliminary characterization procedure must then be run to determine an energy-delay curve for each module in the library and for all possible supply-voltage assignments. The points on the curve represent various voltage assignment solutions with different tradeoffs between the performance and the energy consumption of the cell. Each set of curves is stored in the RTL library, ready to be invoked by the cost function that guides the multiple supply-voltage scheduling algorithm. We provide a brief description of the method for the simple case of control and data flow graphs (CDFG's) with a tree structure. The algorithm consists of two phases: first, a set of possible power-delay tradeoffs at the root of the tree is calculated; then, a specific macro-module is selected for each node in such a way that the scheduled CDFG meets the required timing constraints. To compute the set of possible solutions, a power-delay curve at each node of the tree (proceeding from the inputs to the output of the CDFG) is computed; such a curve represents the power-delay tradeoffs that can be obtained by selecting different instances of the macro-modules, and the necessary level shifters, within the subtree rooted at each specific node. The computation of the power-delay curves is carried out recursively, until the root of the CDFG is reached. Given the power-delay curve at the root node, that is, the set of tradeoffs the user can choose from, a recursive preorder traversal of the tree is performed, starting from the root node, with the purpose of selecting which module alternative should be used at each node of the CDFG. Upon completion, all the operations are fully scheduled; therefore, the CDFG is ready for the resource-allocation step.

More recently, a level-converter free approach is proposed in [11] where the authors try to eliminate the overhead imposed by level converters by suggesting a voltage scaling technique without utilizing level converters. The basic initiative is to impose some constraints on the voltage differences between adjacent gates with different supply voltages based on the observation that there will be no static current if the supply voltage of a driver gate is higher than the subtraction of the threshold voltage of a PMOS from the supply voltage of a driven gate. In [12], Murugavel and Ranganathan propose behavioral-level power optimization algorithms that use voltage and frequency scaling. In this work, the operators in a data flow graph are scheduled in the modules of the given architecture, by applying voltage and frequency scaling techniques to the modules of the architecture such that the power consumed by the modules is minimized. The global optimal selection of voltages and frequencies for the modules is determined through the use of an auction-theoretic model and a game theoretic solution. The authors present a resource constrained scheduling algorithm, which is based on applying the Nash equilibrium function to the game theoretic formulation.

3 Dynamic Voltage Scaling and Razor Logic

The dependence of both performance and power dissipation on supply voltage results in a tradeoff in circuit design. High supply voltage results in high performance while low supply voltage makes an energy efficient design. *Dynamic voltage scaling* (DVS) [13] is a powerful technique to reduce circuit energy dissipation in which, the application or operating system identifies periods of low processor utilization that can tolerate reduced frequency which allows reduction in the supply voltage. Since dynamic power scales quadratically with supply voltage, DVS significantly reduces energy consumption with a limited impact on system performance [14].

Several factors determine the voltage required to reliably operate a circuit in a given frequency. The supply voltage must be sufficiently high to fully evaluate the critical path in a single clock cycle (i.e., critical voltage). To ensure that the circuit operates correctly even in the worst-case operating environment some voltage margins are added to the critical voltage (e.g., process margin due to manufacturing variations, ambient margins to compensate high temperatures and noise margins due to uncertainty in supply and signal voltage levels.)

To ensure correct operation under all possible variations, a conservative supply voltage is typically selected using corner analysis. Hence, margins are added to the critical voltage to account for uncertainty in the circuit models and to account for the worst-case combination of variations. However, such a worst-case combination of variations may be highly improbable; hence this approach overly conservative.

In some approaches the delay of an embedded inverter chain is used as a prediction of the critical path delay of the circuit and the supply voltage is tuned during processor operation to meet a predetermined delay through the inverter-chain [15]. This approach to DVS allows dynamic adjustment of the operating voltage to account for global variations in supply voltage drop, temperature fluctuation, and process variations. However, it cannot account for local variations, such as local supply voltage drops, intra-die process variations, and cross-coupled noise, and therefore requires the addition of some margins to the critical voltage. Also, the delay of an inverter chain does not scale with voltage and temperature in the same way as the delays of the critical paths of the actual design, which can contain complex gates and pass-transistor logic, which again requires extra voltage margins.

In [16] the authors propose a different approach to DVS, referred to as Razor logic, which is based on dynamic detection and correction of speed path failures in digital designs. The basic idea is to tune the supply voltage by monitoring the error rate during operation, which eliminates the need for voltage margins that are necessary for "always-correct" circuit operation in conventional DVS. In Razor logic, the operation at sub-critical supply voltages does not constitute a *failure*, but instead represents a trade-off between the power dissipation penalties incurred from error correction versus the additional power savings obtained from operating at a lower supply voltage.

The Razor logic based DVS utilizes a combination of circuit and architectural techniques for low cost error detection and correction of delay failures. Each flip-flop in the critical path is augmented with a *shadow* latch which is controlled using a delayed clock. The operating voltage is constrained such that the worst-case delay meets the shadow latch setup time, even though the main flip-flop could fail. By comparing the values latched by the flip-flop and the shadow latch, a timing error in the main flip-flop can be detected. The value in the shadow latch, which is guaranteed to be correct, is subsequently utilized to correct the delay failure.

This concept is illustrated in Figure 3(a) for a pipeline stage. The operation of a Razor flip-flop is shown in Figure 3(b). In clock cycle 1, the combinational logic L1 meets the setup time by the rising edge of the clock and both the main flip-flop and the shadow latch will latch the correct data. In this case, the error signal at the output of the XOR gate remains low and the operation of the pipeline is unaltered. In cycle 2, the combinational logic delay exceeds the intended delay due to sub-critical voltage scaling. In this case, the correct data is not latched by the main flip-flop. However, because the shadow-latch operates from a delayed clock, it successfully latches the correct data some time in cycle 3. By comparing the valid data of the shadow latch with the data in the main flip-flop, an error signal is generated in cycle 3. Later, in cycle 4, the valid data in the shadow latch is restored into the main flip-flop and becomes available to the next pipeline stage L2.

If an error occurs in pipeline stage L1 in a particular clock cycle, the data in L2 in the following clock cycle is incorrect and must be flushed from the pipeline. However, since the shadow latch contains the correct output data of pipeline stage L1, the instruction does not need to be re-executed through this failing stage. In addition to invalidating the data in the following pipeline stage, an error stalls the preceding pipeline stages (incurring one cycle penalty) while the shadow latch data is restored into the main flip-flops. Then data is re-executed through the following pipeline stage. A number of different methods, such as clock gating or flushing the instruction in the preceding stages, were presented in [16].

4 RTL Power Management

Digital circuits usually contain portions that are not performing useful computations at each clock cycle. Power reductions can then be achieved by shutting down the circuitry when it is idle.

4.1 Precomputation Logic

Precomputation logic, presented in [17], relies on the idea of duplicating part of the logic with the purpose of precomputing the circuit output values one clock cycle before they are required, and then uses these values to reduce the total amount of switching in the circuit during the next clock cycle. In fact, knowing the output values one clock cycle in advance allows the original logic to be turned off during the next time frame, thus eliminating any charging and discharging of the internal capacitances. Obviously, the size of the logic that pre-calculates the output values must be kept under control since its contribution to the total power balance may offset the savings achieved by blocking the switching inside the original circuit. Several variants to the basic architecture can then be devised to address this issue. In particular, sometimes it may be convenient to resort to partial, rather than global, shutdown, i.e., to select for power management only a (possibly small) subset of the circuit inputs.

The synthesis algorithm presented in [17] suffers from the limitation that if a logic function is dependent on the values of several inputs for a large fraction of the applied input combinations, then no reduction in switching activity can be obtained. In [18], the authors focus on a particular sequential precomputation architecture where the precomputation logic is a function of all of the input variables. The authors call this architecture the "complete input-disabling architecture." It is shown that the complete input disabling architecture can reduce power dissipation for a larger class of sequential circuits compared to the subset input-disabling architecture. The authors present an algorithm to synthesize precomputation logic for the complete input-disabling architecture.

4.2 Clock Gating

Another approach to RT and gate-level dynamic power management, known as gated clocks [19]–[21], provides a way to selectively stop the clock, and thus, force the original circuit to make no transition, whenever the computation that is to be carried out at the next clock cycle is redundant. In other words, the clock signal is disabled according to the idle conditions of the logic network. For reactive circuits, the number of clock cycles in which the design is idle in some wait states is usually large. Therefore, avoiding the power waste corresponding to such states may be significant.

The logic for the clock management is automatically synthesized from the Boolean function that represents the idle conditions of the circuit (cf. Figure 4.) It may well be the case that considering all such conditions results in additional circuitry that is too large and too power consuming. It may then be necessary to synthesize a simplified function, which dissipates the minimum possible power and stops the clock with maximum efficiency. The use of gated clocks has the drawback that the logic implementing the clock-gating mechanism is functionally redundant, and this may create major difficulties in testing and verification. The design of highly testable-gated clock circuits is discussed in [22].

Another difficulty with clock gating is that one must stop hazards/glitches on EN signal from corrupting the clock signal to the register sets. This can be accomplished by introducing a transparent negative latch between EN and the AND gate as shown in Figure 5.

4.3 Computational Kernels

Sequential circuits may have an extremely large number of reachable states, but during normal operation, these circuits tend to visit only a relatively small subset of the reachable states. A similar situation occurs at the primary outputs; while the circuit walks through the most probable states, only a few distinct patterns are generated at the combinational outputs of the circuit. Many researchers have proposed approaches for synthesizing a circuit that is fast and power-efficient under typical input stimuli, but continues to operate correctly even when uncommon input stimuli are applied to the circuit.

Reference [23] presents a power optimization technique by exploiting the concept of computational kernel of a sequential circuit, which is a highly simplified logic block that imitates the steady-state behavior of the original specification. This block is smaller, faster, and less power consuming than the circuit from which it is extracted and can replace the original network for a large fraction of the operation time.

The *p*-order computational kernel of an FSM is defined with respect to a given probability threshold p and includes the subset of the states, S_P, of the original FSM whose steady-state occupation probabilities are larger than *p*. The combinational kernel also includes the subset of states, R_P, where for each state in R_p there is an edge from a state in S_p to that state. As an example, consider the simple FSM shown in Figure 6(a) in which the input and output values are omitted for the sake of simplicity and the states are annotated with the steady-state occupation probabilities calculated through Markovian analysis of the corresponding state transition graph (STG.) If we specify a probability threshold of p=0.25, then the computational kernel of the FSM is depicted in Figure 6(b). States in black represent set S_p, while states in grey represent R_p. The kernel probability is $\text{Prob}(S_p) = 0.29 + 0.25 + 0.32 = 0.86$.

Given a sequential circuit with the standard topology depicted in Figure 7(a), the paradigm for improving its quality with respect to a given cost function (e.g., power dissipation, latency) is based on the architecture shown in Figure 7(b).

The basic elements of the architecture are: the combinational portion of the original circuit (block CL), the computational kernel (block K), the selector function (block S), the double state flip-flops (DSFF), and the output multiplexers (MUX.)

The computational kernel can be seen as a “dense" implementation of the circuit from which it has been extracted. In other terms, K implements the core functions of the original circuit, and because of its reduced complexity, it usually implements such functions in a faster and more efficient way. The purpose of selector function S is that of deciding what logic block, between CL and K, will provide the output value and the next-state in the following clock cycle. To take a decision, S examines the values of the next-state outputs at clock cycle n. If the output and next-state values in cycle n+1 can be computed by the kernel K, then S takes on the value 1. Otherwise, it takes on the value 0. The value of S is fed to a flip-flop, whose output is connected to the MUXes that select which block produces the output and the next-state. The optimized implementation is functionally equivalent to the original one. Computational kernels are a generalization of the precomputation architecture from combinational and pipelined sequential circuits to finite state machines. The authors in [23] proposed an algorithm for generating the computational kernel of a FSM by iterative simplification of the original network by redundancy removal.

In [24], the authors raise the level of abstraction at which the kernel-based optimization strategy can be exploited and show how RTL components for which only a functional specification is available can be optimized using the computational kernels. They present a technique for computational kernel extraction directly from the functional specification of a RTL module. Given the state transition graph (STG) specification, the proposed algorithm calculates the kernel exactly through symbolic procedures similar to those employed for FSM reachability analysis. The authors also provide approximate methods to deal with large STG’s. More precisely, they propose two modifications to the basic procedure. The first one replaces the exact probabilistic analysis of the STG with an approximate analysis. In the second solution, symbolic state probability computation is bypassed and the set of states belonging to the kernel is determined directly from RTL simulation traces of a given (random or user-provided) stream.

4.4 State Machine Decomposition

Decomposition of finite state machines for low power has been proposed in [25]. The basic idea is to decompose the STG of a finite state machine (FSM) into two STGs that jointly produce the equivalent input-output behavior as the original machine. Power is saved because, except for transitions between the two sub-FSMs, only one of the sub-FSMs needs to be clocked. The technique follows a standard decomposition structure. The states are partitioned by searching for a small subset of states with high probability of transitions among these states and a low probability of transitions to and from other states. This subset of states will then constitute a small sub-FSM that is active most of the time. When the small sub-FSM is active, the other larger sub-FSM can be disabled. Consequently, power is saved because most of the time only the smaller, more power-efficient, sub-FSM is clocked.

In [26], the combinational logic block is partitioned (for example to CL1 and CL2) and the active part is decided based on the encoding of the present state. The states selected for one of the sub-FSMs (i.e., M1) are all encoded in such a way that the enable signal is always on for CL1 while it is off for CL2. Conversely, for all states in the other sub-FSM (i.e., M2), the enable signal is always off for CL1 while it is on for CL2. Consequently, for all transitions within M1, only CL1 will be active and vice-versa.

Consider as an example *dk27* FSM from the MCNC benchmark set, depicted in Figure 8. Assume that the input signal values, 0 and 1, occur with equal probabilities. The steady state probabilities which are shown next to the states in this figure have been computed accordingly. Suppose we partition the FSM into two sub-machines M1 and M2 along the dotted line. Then around 40% of the transitions occur in submachine M1, 40% of the transitions occur in submachine M2, and 20% of the transitions occur between sub-machines M1 and M2. Now suppose that the FSM is synthesized as two individual combinational circuits for sub-machines M1 and M2. Then we can turn off the combinational circuit for submachine M2 when transitions occur within submachine M1. Similarly, we can turn off the combinational circuit for submachine M1 when transitions occur within submachine M2. The states are partitioned such that the probability of transitions within any sub-FSM is maximized and the estimated overhead is minimized.

These methods for FSM decomposition can be considered as extensions of the gated-clock for FSM self-loops approach proposed in [27]. In FSM decomposition the cluster of states that are selected for one of the sub-FSMs can be considered as a "super-state" and then transitions between states in this cluster can be seen as self-loops on this "super-state".

4.5 Guarded Evaluation

Guarded evaluation [29] is the last RT and gate-level shutdown technique we review in this section. The distinctive feature of this solution is that, unlike precomputation and gated clocks, it does not require one to synthesize additional logic to implement the shutdown mechanism; instead, it exploits existing signals in the original circuit. The approach is based on placing some guard logic, consisting of transparent latches with an enable signal, at the inputs of each block of the circuit that needs to be power managed. When the block must execute some useful computation in a clock cycle, the enable signal makes the latches transparent. Otherwise, the latches retain their previous states, thus, blocking any transition within the logic block.

Guarded evaluation provides a systematic approach to identify where transparent latches must be placed within the circuit and by which signals they must be controlled. For Example, Let C be a combinational logic block (cf. Figure 9(a)), X be the set of primary inputs to C, and z be a signal in C. Furthermore, let F be the portion of logic that drives z and Y be the set of inputs to F. Finally, let $D_Z(X)$ be the observability don't-care set for z (that is, the set of primary input assignments for which the value of z does not influence the outputs of C). Now consider a signal s in C which logically implies $D_Z(X)$, that is, $s \Rightarrow D_Z(X)$. Then, if $s=1$, then the value of z is not required to compute the outputs of C. If we call $t_e(Y)$ the earliest time at which any input to F can switch when $s=1$, and $t_l(s)$ as the latest time at which *s* settles to one, then signal *s* can be used as the guard signal for *F* (cf. Figure 9(b)) if $t_l(s) < t_e(Y)$. This is because *z* is not required to compute the outputs of *C* when *s*=1, and therefore, block *F* can be shut down. Notice that the condition $t_l(s) < t_e(Y)$ guarantees that the transparent latches in the guard logic are shut down before any of the inputs to *F* makes a transition.

This technique, referred to as pure guarded evaluation, has the desirable property that when applied, no changes in the original combinational circuitry are needed. On the other hand, if some resynthesis and restructuring of the original logic is allowed, a larger number of logic shutdown opportunities may become available.

5 Sequential Logic Synthesis for Low Power

Power can be minimized by appropriate synthesis of logic. The goal in this case is to minimize the so-called switched capacitance of the circuit by low power driven logic minimization techniques.

5.1 State Assignment

State encoding/assignment, as a crucial step in the synthesis of the controller circuitry, has been extensively studied. Roy et al. was the first to address the problem of reducing switching activity of input state lines of the next state logic, during the state assignment, formulating it as a Minimum Weighted Hamming Distance problem [30]. Olson et al. used a linear combination of switching activity of the next state lines and the number of literals as the cost function [31]. Tsui et al. [32] used simulated annealing as a search strategy to find a low power state encoding that accounts for both the switching activity of the next state lines and switched capacitance of the next state and output logic.

For example, consider the state transition graph for a BCD to Excess-3 Converter depicted in Figure 10. Assume that the transition probabilities of the thicker edges in this figure are more than those of the thin edges. The key idea behind all of the low power state assignment techniques is to assign minimum Hamming distance codes to the states pairs that have large inter-state transition probabilities. For example the coding, S_0=000, S_1=001, S_2=011, S_3=010, S_4=100, S_5=101, S_6=111, S_7=110 fulfills this requirement.

In [33], Wu et al. proposed the idea of realizing a low power FSM by using T flip-flops. The authors showed that use of T flip flops results in a natural clock gating and may result in reduced next state logic complexity. However, that work was mostly focused on BCD counters which have cyclic behavior. The cyclic behavior of counters results in a significant reduction of combinational logic complexity and, hence, lowers power consumption. Reference [34] introduces a mathematical framework for cycle representation of Markov processes and based on that, proposes solutions to the low power state assignment problem. The authors first identify the most probable cycles in the FSM and encode the states on these cycles with Gray codes. The objective function is to minimize the Weighted Hamming Distance. This reference also teaches how a combination of T and D flip-flops as state registers can be used to achieve a low power realization of a FSM.

5.2 Retiming

Retiming is to reposition the registers in a design to improve the area and performance of the circuit without modifying its input-output behavior. The technique was initially proposed by Leiserson and Saxe [35]. This technique changes the location of registers in the design in order to achieve one of the following goals: 1) minimize the clock period; 2) minimizing the number of registers; or 3) minimize the number of registers for a target clock period.

Minimizing dynamic power for synchronous sequential digital designs is addressed in the literature. In [36], Monteiro et al. presented heuristics to minimize the switching activity in a pipelined sequential circuit. Their approach is based on the fact that registers have to be positioned on the output edges of the computational elements that have high switching activity. The reason for power savings is that in this case the output of a register switches only at the arrival of the clock signal as opposed to potentially switching many times in the clock period. Consider the simple example of a logic gate belonging to a synchronous circuit and a capacitive load driven by the output gate. In CMOS technology, the power dissipated by gate is proportional to the product of the switching activity of the output node of the gate and the output load. At the output of gate some spurious transitions (i.e., glitches) may occur, which can result in a significant power waste. Suppose a register is inserted between the output of the gate and the capacitive load. In the new circuit, the output of the register can make, at most, one transition per clock cycle. In fact, the gate output may have many redundant transitions but they are all filtered out by the register; hence, these logic hazards do not propagate to the output load.

The heuristic retiming technique of [36] applies to a synchronous network with pipeline structure. The basic idea is to select a set of candidate gates in the circuit such that if registers are placed at their outputs, the total switching activity of the network gets minimized. The selection of the gates is driven by two factors: the amount of glitching that occurs at the output of each gate and the probability that such glitching propagates to the gates located in the transitive fanout. Registers are initially placed at the primary inputs of the circuit, and backward retiming (which consists of moving one register from all gate inputs to the output) is applied until all the candidate gates have received a register on their outputs. Then, registers that belong to paths not containing any of the candidate gates are repositioned, with the objective of minimizing both the delay and the total number of registers in the circuit. This last retiming phase does not affect the registers that have been already placed at the outputs of the previously selected gates. In [37], fixed-phase retiming is proposed to reduce dynamic power consumption. The edge-triggered circuit is first transformed to a two-phase level-clocked circuit, by replacing each edge-triggered flip-flop by two latches. Using the resulting level-clocked circuit, the latches of one phase are kept fixed, while the latches belonging to the other phase are moved onto wires with high switching activity and loading capacitance.

Fixed-phase retiming is best illustrated by the example shown below. Figure 11(a) shows a section of a pipelined circuit with edge-triggered flip-flops. The numbers on the edges represent the potential reduction in power dissipation when an edge-triggered flip-flop is present on that edge, assuming that the rest of the circuit remains unchanged. Negative values of power reduction indicate an increase in power dissipation when a flip-flop is placed on an edge. This reduction in power dissipation can be achieved if the edge has a high glitching-capacitance product [3]. After replacing each edge-triggered flip-flop by two back-to-back level-clocked latches, the resulting circuit is fixed-phase retimed to obtain the circuit in Figure 11(b).

Assuming a non-overlapping two-phase clocking scheme $\pi = \langle \phi_0 = 4, \gamma_0 = 1, \phi_1 = 4, \gamma_1 = 1 \rangle$ such as the one shown in Figure 11(c), power dissipation can be reduced by 11.8 units. Specifically, the glitching on edges B→D, E→F and E→H is "masked" for 60% of the clock cycle which decreases power dissipation by $0.6 \times (12 + 13 - 2) = 13.8$ units of power. At the same time, the glitching on edges G→J and H→K is "exposed" for 40% of the clock cycle which increases power dissipation by $0.4 \times (10 - 5) = 2$ power units. In order to simplify the computation of changes in power dissipation for this example, it is assumed that glitching is uniformly distributed over the entire clock period and that the relocation of latches does not change glitching significantly.

In [38], Chabini and Wolf propose a hybrid retiming and supply voltage scaling. They observe that critical paths are related to the position of registers in a design so they try not only to scale down the supply voltage of computational elements that are off the critical paths, but also to move registers to maximize the number of computational elements that are off the critical paths, thereby further minimizing the circuit power consumption. Registers have to be moved from their positions by the standard retiming technique. Instead of unifying basic retiming and supply voltages scaling, the authors propose to apply "guided retiming" followed by the application of voltage scaling on the retimed design. Polynomial time algorithms based on dynamic programming to realize the guided retiming as well as the supply voltage scaling on the retimed design are proposed.

6 Leakage Power Reduction Techniques

In many new high performance designs, the leakage component of power consumption is comparable to the switching component. Reports indicate that 40% or even higher percentage of the total power consumption is due to the leakage of transistors. This percentage will increase with technology scaling unless effective techniques are introduced to bring leakage under control. This section focuses mostly on RTL optimization and design automation techniques that accomplish this goal.

There are four main sources of leakage current in a CMOS transistor:

1. Reverse-biased junction leakage current (I_{REV})
2. Gate induced drain leakage (I_{GIDL})
3. Gate direct-tunneling leakage (I_G)
4. Subthreshold (weak inversion) leakage (I_{SUB})

Let I_{OFF} denote the leakage of an OFF transistor (V_{GS}=0V for an NMOS device which results in I_G=0.)

$$I_{OFF} = I_{REV} + I_{GIDL} + I_{SUB} \, .$$

Components, I_{REV} and I_{GIDL} are maximized when $V_{DB} = V_{DD}$. Similarly, for short-channel devices, I_{SUB} increases with V_{DB} because of the DIBL effect. Note the I_G is not a component of the OFF current, since the transistor gate must be at a high potential with respect to the source and substrate for this current to flow. An effective approach to overcome the gate leakage currents while maintaining excellent gate control is to replace the currently-used silicon dioxide gate insulator with high-K dielectric material such as TiO2 and Ta2O5. Use of the high-k dielectric will allow a less aggressive gate dielectric thickness reduction while maintaining the required gate overdrive at low supply voltages [39]. High-K gate dielectrics are expected to be introduced in 2006 [40]. Therefore, it is reasonable to ignore the I_G component of leakage. Among the three components of I_{OFF}, I_{SUB} is the dominant component. Hence, most leakage reduction techniques focus on I_{SUB}.

6.1 Power Gating and Multi-Threshold CMOS

The most obvious way of reducing the leakage power dissipation of a VLSI circuit in the STANDBY state is to turn off its supply voltage. This can be done by using one PMOS transistor and one NMOS transistor in series with the transistors of each logic block to create a virtual ground and a virtual power supply as depicted in Figure 12. In practice only one transistor is necessary. Because of the lower on-resistance, NMOS transistors are usually used.

In the ACTIVE state, the sleep transistor is on. Therefore, the circuit functions as usual. In the STANDBY state, the transistor is turned off, which disconnects the gate from the ground. To lower the leakage, the threshold voltage of the sleep transistor must be large. Otherwise, the sleep transistor will have a high leakage current, which will make the power gating less effective. Additional savings may be achieved if the width of the sleep transistor is smaller than the combined width of the transistors in the pull-down network. In practice, Dual V_T CMOS or Multi-Threshold CMOS (MTCMOS) is used for power gating [41][42]. In these technologies there are several types of transistors with different V_T values. Transistors with a low V_T are used to implement the logic, while high-V_T devices are used as sleep transistors.

To guarantee the proper functionality of the circuit, the sleep transistor has to be carefully sized to decrease its voltage drop while it is on. The voltage drop on the sleep transistor decreases the effective supply voltage of the logic gate. Also, it increases the threshold of the pull-down transistors due to the body effect. This increases the high-to-low transition delay of the circuit. This problem can be solved by using a large sleep transistor. On the other hand, using a large sleep transistor increases the area overhead and the dynamic power consumed for turning the transistor on and off. Note that because of this dynamic power consumption, it is not possible to save power for short idle periods. There is a minimum duration of the idle time below which power saving is impossible. Increasing the size of the sleep transistors increases this minimum duration.

Since using one transistor for each logic gate results in a large area and power overhead, one transistor may be used for each group of gates as depicted in Figure 13. Notice that the size of the sleep transistor in this figure ought to be larger than the one used in Figure 12. To find the optimum size of the sleep transistor, it is necessary to find the vector that causes the worst case delay in the circuit. This requires simulating the circuit under all possible input values, a task that is not possible for large circuits.

In [42], Kao and Chandrakasan describe a method to decrease the size of sleep transistors based on the mutual exclusion principle. In their method, the authors first size the sleep transistors to achieve delay degradation less than a given percentage for each gate. Notice that this guarantees that the total delay of the circuit will be degraded by less than the given percentage. In fact the actual degradation can be as much as 50% smaller. The reason for this is that NMOS sleep transistors degrade only the high-to-low transitions and at each cycle only half of the gates switch from high to low. If two gates switch at different times (i.e., their switching windows are non-overlapping), then their corresponding sleep transistors can be shared.

Although sleep transistors can be used to disconnect logic gates from ground, using them to disconnect Flip Flops from ground or supply voltage results in the loss of data. The authors of [43] solve this problem by using high threshold transistors for the inverters that hold data and low threshold transistors for other parts of Flip Flops. In the sleep mode, the low threshold transistors are disconnected from the ground, but the two inverters that hold data stay connected to the ground. Since high threshold transistors have been used in the inverters, their leakage is small. Other possibilities for saving data when MTCMOS is applied to a sequential circuit are to utilize leakage-feedback gates and flip flops [44] or balloon latches [45].

6.2 Multiple Threshold Cells

Multiple threshold voltages have been available on many CMOS processes for a number of years. Multiple-Threshold CMOS circuit, which has both high and low threshold transistors in a single chip, can be used to deal with the leakage problem. The high threshold transistors can suppress the subthreshold leakage current, while the low threshold transistors are used to achieve the high performance. Since the standby power is much larger for low V_T transistors compared to the high V_T ones, usage is limited to using low V_T transistors on timing-critical paths, with insertion rates on the order of 20% or less. Since T_{ox} and L_{gate} are the same for high and low V_T transistors, low V_T insertion does not adversely impact the active power component or the design size. Drawbacks are that variation due to doping is uncorrelated between the high and low threshold transistors and extra mask steps incur a process cost.

The technology used for fabricating circuits can restrict the manner in which transistors can be mixed. For example, it may not be possible to use different threshold voltages for transistors in a stack due to their proximity. Furthermore, to simplify the design process and Computer-Aided Design (CAD) algorithms, one may wish to restrict the way transistors are mixed. For example, when transistors of the same type are used in a logic cell, the size of multi-threshold cell library is only twice that of the original (single threshold) cell library. This reduces the library development time as well as the complexity and run time of CAD algorithms and tools that use the library.

In general, one expects that the leakage saving increases as the freedom to mix low and high V_T devices in a logic cell is increased. However, the percentage improvement is usually minor. Compared to the case of using logic cells with the same type of transistors (i.e., low threshold or high threshold) everywhere, reference [46] reports an average of only 5% additional leakage savings by using logic cells with the same type of transistors in a transistor stack.

Although using two threshold voltages instead of one significantly decreases the leakage current in a circuit, using more than two threshold voltages marginally improves the result [47]. This is true even when the threshold values are optimized to minimize the leakage for a given circuit. Thus, in many designs, only two threshold voltages are used.

6.3 Minimum Leakage Vector Method

The leakage current of a logic gate is a strong function of its input values. The reason is that the input values affect the number of OFF transistors in the NMOS and PMOS networks of a logic gate.

Table 1 shows the leakage current of a two-input NAND gate built in a 0.18μm CMOS technology with a 0.2V threshold voltage and a 1.5V supply voltage. Input A is the one closer to the output of the gate.

Table 1. **The leakage values of a NAND gate.**

Inputs		Output	Leakage Current (nA)
A	B	O	

0	0	1	23.06
0	1	0	51.42
1	0	0	47.15
1	1	0	82.94

The minimum leakage current of the gate corresponds to the case when both its inputs are zero. In this case, both NMOS transistors in the NMOS network are off, while both PMOS transistors are on. The effective resistance between the supply and the ground is the resistance of two OFF NMOS transistors in series. This is the maximum possible resistance. If one of the inputs is zero and the other is one, the effective resistance will be the same as the resistance of one OFF NMOS transistor. This is clearly smaller than the previous case. If both inputs are one, both NMOS transistors will be on. On the other hand, the PMOS transistors will be off. The effective resistance in this case is the resistance of two OFF PMOS transistors in parallel. Clearly, this resistance is smaller than the other cases.

In the NAND gate of *Table 1* the maximum leakage is about three times higher than the minimum leakage. Note that there is a small difference between the leakage current of the A=0, B=1 vector and the A=1, B=0 vector due to the body effect. The phenomenon whereby the leakage current through a stack of two or more OFF transistors is significantly smaller than a single device leakage is called the “stack effect”. Other logic gates exhibit a similar leakage current behavior with respect to the applied input pattern. As a result, the leakage current of a circuit is a strong function of its input values. It is possible to achieve a moderate reduction in leakage using this technique, but the reduction is not as high as the one achieved by the power gating method. On the other hand, the MLV method does not suffer from many of the shortcomings of the other methods. In particular,

1. No modification in the process technology is required.
2. No change in the internal logic gates of the circuit is necessary.
3. There is no reduction in voltage swing.

4. Technology scaling does not have a negative effect on its effectiveness or its overhead. In fact the stack effect becomes stronger with technology scaling as DIBL worsens.

The first three facts make it very easy to use this method in existing designs. This technique is also referred to as input vector control (IVC) [48]. The problem of finding MLV for an arbitrary circuit is NP-complete [49] for which a number of heuristics have been proposed including a random simulation based approach presented in [48]. In [49], the authors used a constraint graph to solve the problem for circuits with only a small number of inputs. An explicit branch and bound enumeration technique is described in [50]. For large circuits, bounds on the minimum and maximum leakage values were obtained by using heuristics. Abdollahi et al. [51] formulated the problem of determining the MLV using a series of Boolean Satisfiability problems and solved accordingly. The authors report between 10% to 55% reduction in the leakage by using the MLV technique. Note that the saving is defined as $(1-\frac{Leakage_{MLV}}{Leakage_{AVG}})\times 100$, where $Leakage_{MLV}$ is the leakage when the minimum leakage vector drives the circuit whereas $Leakage_{AVG}$ is the expected leakage current under an arbitrary input combination (this is used because the input value prior to entering the sleep mode is unknown.)

Lee and Blaauw [52] used the combination of MLV and dual-V_T assignment for leakage power reduction. They observe that within the performance constraints, it is more effective to switch off a high-V_T transistor than a low-V_T one. Naidu et al. [53] proposed an integer linear programming (ILP) model for circuits composed of NAND or AOI gates, which obtains the MLV. Gao and Hayes [54] proposed an ILP model for finding MLV, called the virtual-gate or VG-ILP model. Virtual gates are cells that are added to the given circuit to facilitate model formulation, but have no impact on the functionality of the original circuit. The leakage current is viewed as a pseudo-Boolean function of the inputs, which is subsequently linearized. The authors resort to ILP to obtain the input MLV using linearized leakage current functions. They also propose a fast, heuristic technique for MLV calculation, which selectively relaxes variables of the ILP model, leading to a mixed-integer linear programming (MLP) model.

6.4 Increasing the Transistor Channel Lengths

Active leakage of CMOS gates can be reduced by increasing their transistor channel lengths [55]. This is because there is a V_T roll-off due to the Short Channel Effect (SCE). Therefore, different threshold voltages can be achieved by using different channel lengths. The longer transistor lengths used to achieve high threshold transistors tend to increase the gate capacitance, which has a negative impact on the performance and dynamic power dissipation. Compared with multiple threshold voltages, long channel insertion has similar or lower process cost, taken as the size increase rather than the mask cost. It results in lower process complexity. In addition, the different channel lengths track each other over process variation. This technique can be applied in a greedy manner to an existing design to limit the leakage currents [56]. A potential penalty is that the dynamic power dissipation of the up-sized gate is increased proportional to the effective channel length increase. In general, circuit power dissipation may not be saved unless the activity factor of the affected gates is low. Therefore, the activity factor must be taken into account when choosing gates whose transistor lengths are to be increased.

6.5 Transistor Sizing with Simultaneous Threshold and Supply Voltage Assignment

Increasing the threshold voltage of a transistor reduces the leakage current exponentially, but it has a marginal effect on the dynamic power dissipation. On the other hand, reducing the width of a transistor reduces both leakage and dynamic power, but at a linear rate only. Nguyen et al. in [57] report an average 60% and 75% reduction in the total power dissipation by using sizing alone and sizing combined with V_T assignment, respectively. The combination of the technique with dual Vdd assignment resulted in only a marginal improvement, probably because of the optimization algorithm used by the authors. Combining the three optimizations is currently an active area of research and will enable synthesizing lower power circuits in the near future.

7 Conclusion

Several key elements emerge as enablers for an effective low power design methodology. The first is the availability of accurate, comprehensive power models. The second is the existence of fast, easy to use high level estimation and design exploration tools for analysis and optimization during the design creation process, while the third is the existence of highly accurate, high capacity verification tools for tape-out power verification. As befitting a first-order concern, successfully managing the various power-related design issues will require that power be addressed at all phases and in all aspects of design, especially during the earliest design and planning activities. Advanced power tools will play central roles in these efforts.

An RTL design methodology supported by the appropriate design automation tools is one of the most effective methods of designing complex chips for lower power dissipation. Moreover, this methodology drastically reduces the risk of not meeting often harsh power constraints by the early identification of power hogs or hot spots, and enabling the analysis and selection of alternative solutions. Such methodologies have already been adopted by designers of complex chips and constitute the state-of-the-art in designing complex, high-performance, yet low power, designs.

This paper reviewed a number of RTL techniques for low power design of VLSI circuits targeting both dynamic and leakage components of power dissipation in CMOS VLSI circuits. A more detailed review of techniques for low power design of VLSI circuits and systems can be found in many references, including [58].

References

[1] M. Pedram and J. Rabaey (editors), *Power Aware Design Methodologies*, Kluwer Academic Publishers, Boston, 2002.

[2] E. Macii (editor), *Ultra Low-Power Electronics and Design*, Kluwer Academic Publishers, Boston, 2004.

[3] C. Piguet (editor), *Low Power Electronics Design*, The CRC Press, 2004.

[4] M. Hamada, M. Takahashi, H. Arakida, A. Chiba, T. Terazawa, T. Ishikawa, M. Kanazawa, M. Igarashi, K. Usami, and T. Kuroda,"A top-down low power design technique using clustered voltage scaling with variable supply-voltage scheme," in *Proc. IEEE Custom Integrated Circuits Conference* (CICC’98), May 1998, pp. 495-498.

[5] S . Raje and M. Sarrafzadeh, “Variable voltage scheduling,” in *Proc. Int’l. Workshop Low Power Design*, Aug. 1995, pp. 9–14.

[6] K. Usami and M. Horowitz, “Clustered voltage scaling technique for low-power design,” in *Proc. Int’l. Workshop Low Power Design*, 1995, pp. 3–8.

[7] Usami, K. Igarashi, M. Minami, F. Ishikawa, T. Kanzawa, M. Ichida, M. Nogami, K. “Automated low-power technique exploiting multiple supply voltages applied to a media processor," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 3, Mar. 1998, pp 463 – 472.

[8] C. Chen, A. Srivastava, and M. Sarrafzadeh, “On gate level power optimization using dual supply voltages,” *IEEE Trans. on VLSI Systems*, vol. 9, Oct. 2001, pp. 616–629.

[9] A. Manzak and C. Chakrabarti, “A Low Power Scheduling Scheme with Resources Operating at Multiple Voltages,” IEEE Trans. on VLSI Systems, vol. 10, no. 1, Feb. 2002, pp. 6-14.

[10] J. M. Chang and M. Pedram, “Energy minimization using multiple supply voltages,” *IEEE Trans. VLSI Systems*, vol. 5, no. 4, 1997, pp. 436–443.

[11] Y.-J. Yeh, S.-Y. Kuo, and J.-Y. Jou, “Converter-free multiple-voltage scaling techniques for low-power CMOS digital design,” *IEEE Trans. Computer-Aided Design*, vol. 20, Jan. 2001, pp. 172–176.

[12] A. K. Murugavel, N. Ranganathan, “Game Theoretic Modeling of Voltage and Frequency Scaling during Behavioral Synthesis,” in *Proc. of VLSI Design*, 2004, pp. 670-673.

[13] T. Mudge. “Power: A first class design constraint,” *Computer*, vol. 34, no. 4, April 2001, pp. 52-57.

[14] T. Pering, T. Burd, and R. Brodersen. “The Simulation and Evaluation of Dynamic Voltage Scaling Algorithms.” *Proceedings of Int’l Symp. on Low Power Electronics and Design 1998*, pp. 76-81, June 1998.

[15] R. Gonzalez, B. Gordon, and M. Horowitz, “Supply and Threshold Voltage Scaling for Low Power CMOS,” IEEE Journal of solid-State Circuits, 32 (8), August 1997.

[16] Dan Ernst, Nam Sung Kim, Shidhartha Das, Sanjay Pant, Toan Pham, Rajeev Rao, Conrad Ziesler, David Blaauw, Todd Austin, Trevor Mudge “Razor: A Low-Power

Pipeline Based on Circuit-Level Timing Speculation," *Proc. 36th Ann. Int'l Symp. Microarchitecture* (MICRO-36), IEEE CS Press, 2003, pp. 7-18.

[17] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. VLSI Systems,* vol. 2, no. 4, 1994, pp. 426–436.

[18] J. Monteiro, S. Devadas, A. Ghosh, "Sequential Logic Optimization For Low Power Using Input-disabling," *IEEE Trans. on Computer-Aided Design*, vol. 17, no. 3, 1998, pp. 279–284.

[19] L. Benini, P. Siegel, and G. De Micheli, "Automatic synthesis of gated clocks for power reduction in sequential circuits," *IEEE Design Test Computer Magazine*, vol. 11, no. 4, pp. 32–40, 1994.

[20] L. Benini and G. De Micheli, "Transformation and synthesis of FSM's for low power gated clock implementation," *IEEE Trans. on Computer-Aided Design*, vol. 15, no. 6, 1996, pp. 630–643.

[21] L. Benini, G. De Micheli, E. Macii, M. Poncino, and R. Scarsi, "Symbolic synthesis of clock-gating logic for power optimization of control-oriented synchronous networks," in *Proc. European Design and Test Conf.,* Paris, France, Mar. 1997, pp. 514–520.

[22] L. Benini, M. Favalli, and G. De Micheli, "Design for testability of gated-clock FSM's," in *Proc. European Design and Test Conf.,* Paris, France, Mar. 1996, pp. 589–596.

[23] L. Benini, G. De Micheli, A. Lioy, E. Macii, G. Odasso, and M. Poncino, "Synthesis of Power-Managed Sequential Components Based on Computational Kernel Extraction," *IEEE Trans. on Computer-Aided Design*, vol. 20, no. 9, September 2001, pp. 1118-1131.

[24] L Benini, G. De Micheli, E. Macii, G. Odasso, M. Poncino, "Kernel-Based Power Optimization of RTL Components: Exact and Approximate Extraction Algorithms," in *Proc. of Design Automation Conf.,* 1999, pp. 247-252.

[25] J. Monteiro and A. Oliveira. Finite State Machine Decomposition for Low Power. In *Proc. of Design Automation Conference*, June 1998, pages 758-763.

[26] S-H. Chow, Y-C. Ho, and T. Hwang. "Low Power Realization of Finite State Machines A Decomposition Approach," *ACM Trans. on Design Automation of Electronic Systems,* vol. 1 no. 3, July 1996, pp. 315-340.

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

[28] J. C. Monteiro, "Power optimization using dynamic power management," in *Proc. of the XII Symp. on Integrated Circuits and Systems Design*, Sep. 1999, pp. 134-139.

[29] V. Tiwari, S. Malik, and P. Ashar, "Guarded evaluation: Pushing power management to logic synthesis/design," in *Proc. ACM/IEEE Int'l. Symp. Low Power Design,* Dana Point, CA, Apr. 1995, pp. 221–226.

[30] K. Roy and S. Prasad, "Syclop: Synthesis of CMOS Logic for Low-Power Application," *Proc. of Int'l Conf. on Computer design*, pp. 464-467, Oct. 1992.

[31] E. Olson and S. M. Kang, "Low-Power State Assignment for Finite State Machines," in *Proc. of Int'l Workshop on Low Power Design,* pp. 63-68, April 1994.

[32] C. Y. Tsui, M. Pedram and A. M. Despain, "Low-Power State Assignment Targeting Two- and Multilevel Logic Implementation," *IEEE Trans. on Computer-Aided Design,* vol. 17, no. 12, Dec. 1998, pp. 1281-1291.

[33] X. Wu, J. Wei, Q. Wu, and M. Pedram, "Low-Power Design of Sequential Circuits Using a Quasi-Synchronous Derived Clock," *Int'l Journal of Electronics,* Taylor and Francis Publishing Group, vol. 88, no. 6, Jun. 2001, pp. 635-643.

[34] A. Iranli, P. Rezvani, and M. Pedram, "Low power synthesis of finite state machines with mixed D and T flip-flops," in *Proc. of Asia and South Pacific Design Automation Conference,* Jan. 2003, pp. 803-808.

[35] C. E. Leiserson and J. B. Saxe, "Optimizing synchronous systems," *Journal of VLSI Computer Systems,* vol. 1, no. 1, 1983, pp. 41–67.

[36] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *Proc. Int'l. Conf. Computer-Aided Design*, Santa Clara, CA, Nov. 1993, pp. 398–402.

[37] K. N. Lalgudi and M. Papaefthymiou, “Fixed-phase retiming for low power,” in *Proc. Int’l. Symp Low-Power Electronics and Design*, Aug. 1996, pp. 259–264.

[38] N. Chabini and W. Wolf, "Reducing Dynamic Power Consumption in Synchronous Sequential Digital Designs Using Retiming and Supply Voltage Scaling" *IEEE Trans. on VLSI Systems*, vol. 12, no. 6, Jun. 2004, pp.573-589.

[39] S. Borkar, “Design Challenges of Technology Scaling,” IEEE Micro, Vol. 19, Issue 4, Jul.-Aug. 1999.

[40] Semiconductor Industry Association, *Int’l Technology Roadmap for Semiconductors*, 2003 edition, http://public.itrs.net/.

[41] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, “ 1-V Power Supply High-Speed Digital Circuit Technology with Multithreshold CMOS,” *IEEE J. Solid-State Circuits* 30, No. 8, August 1995, pp. 847–854.

[42] J. T. Kao, A. P. Chandrakasan, ``Dual-threshold voltage techniques for low-power digital circuits," *IEEE Journal of Solid-State Circuits*, Vol. 35, July 2000, pp. 1009-1018.

[43] Hyo-Sig Won, et al., “An MTCMOS Design Methodology and Its Application to Mobile Computing,” *Proc. of the Int’l Symp. on Low power electronics and design*, August 2003.

[44] J. Kao and A. Chandrakasan, “MTCMOS sequential circuits,” *Proc. ESSCIRC*, 2001, pp. 332–339.

[45] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe, J. Yamada, “A 1-V high-speed MTCMOS circuit scheme for power-down application circuits,” *IEEE Journal of Solid State Circuits*, Vol. 32, June 1997, pp. 861-870.

[46] L. Wei, K. Roy, Y. Ye, and V. De, “Mixed-Vth (MVT) CMOS Circuit Design Methodology for Low Power Applications,” *Proc. Design Automation Conference*, Jun. 1999, pp. 430-435.

[47] A. Srivastava, “Simultaneous Vt Selection and Assignment for Leakage Optimization,” *Proc. Int’l Symp. on Low Power Electronics and Design*, Aug. 2003, pp. 146-151.

[48] Halter J., Najm, F., “A Gate-level Leakage Power ReductionMethod for Ultra Low Power CMOS Circuits,” *IEEE Custom Integrated Circuits Conference*, 1997, pp. 475-478.

[49] S. Bobba and I. N. Hajj, "Maximum leakage power estimation for CMOS circuits," Proceedings of *IEEE Alessandro Volta Memorial Int’l Workshop on Low Power Design*, Como, Italy, March 4-5, 1999. pp. 116-124.

[50] M. Johnson, D. Somasekhar, K. Roy, "Models and Algorithms for Bounds in CMOS Circuits," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, Vol. 18, No. 6, June 1999, pp. 714-725.

[51] A. Abdollahi, F. Fallah and M. Pedram, “Leakage current reduction in CMOS VLSI circuits by input vector control,” *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, Vol. 12 , Issue: 2 , Feb. 2004, pp. 140-154.

[52] D. Lee and D. Blaauw, “Static leakage reduction through simultaneous threshold voltage and state assignment,” *Proc. Design Automation Conference*, Jun. 2003, pp191-194.

[53] S. Naidu and E. Jacobs, “Minimizing Standby Leakage Power in Static CMOS Circuits,” *Proc. Design Automation and Test in Europe*, 2001, pp 370 - 376.

[54] F. Gao and J. P. Hayes, “Exact and Heuristic Approaches to Input Vector Control for Leakage Reduction,” *Proc. Intl. Conf. on Computer-Aided Design*, Nov. 2004, pp. 527-532.

[55] L. Wei, K. Roy, and V. De, “Low voltage low power CMOS design techniques for deep submicron ICs,” *Proc. Thirteenth Int’l Conference on VLSI Design*, 2000, pp. 24-29.

[56] L.T. Clark, R. Patel, and T.S. Beaty, “Managing standby and active mode leakage power in deep sub-micron design,” *Proc. Int’l Symp. on Low Power Electronics and Design*, Aug. 2004, pp. 274-279.

[57] D. Nguyen, A. Davare, M. Orshansky, D. Chinnery, B. Thompson, and K. Keutzer, “Minimization of Dynamic and Static Power Through Joint Assignment of Threshold Voltages and Sizing Optimization,” *Proc. Int’l Symp. on Low Power Electronics and Design*, Aug. 2003, pp. 158-163.

[58] M. Pedram and J. Rabaey (editors), *Power-Aware Design Methodologies*, Kluwer Academic Publishers, Boston, 2002.

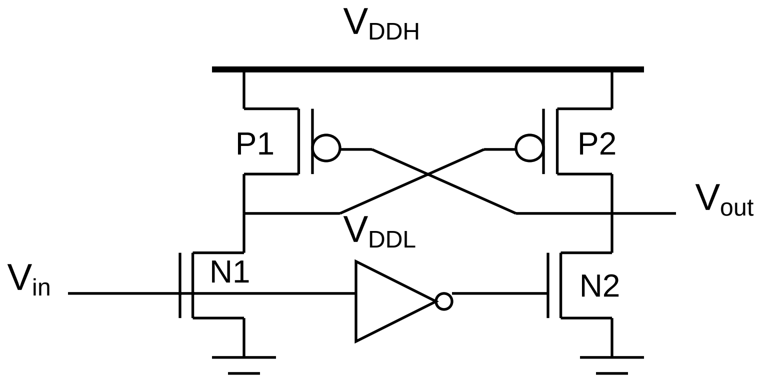
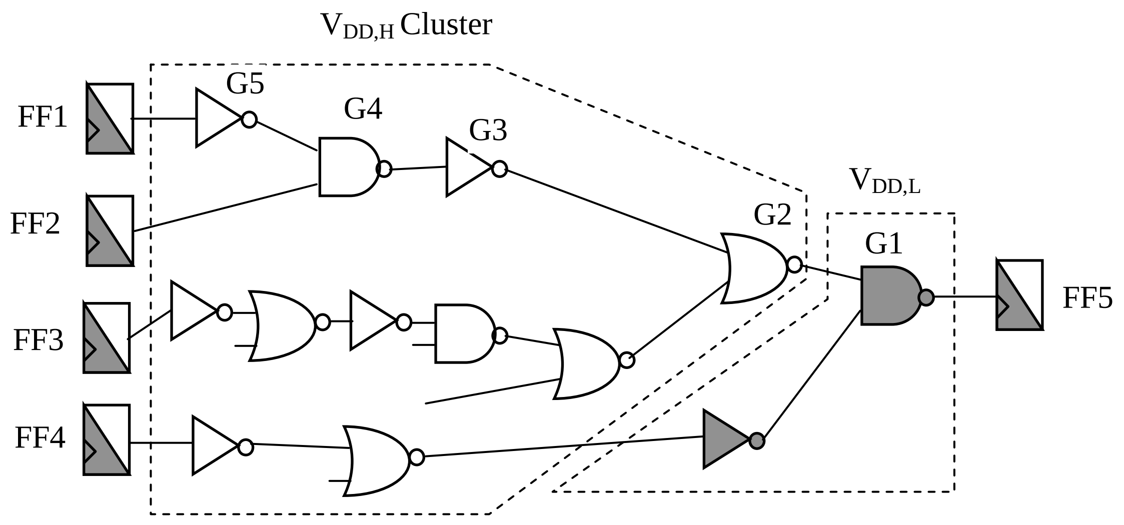
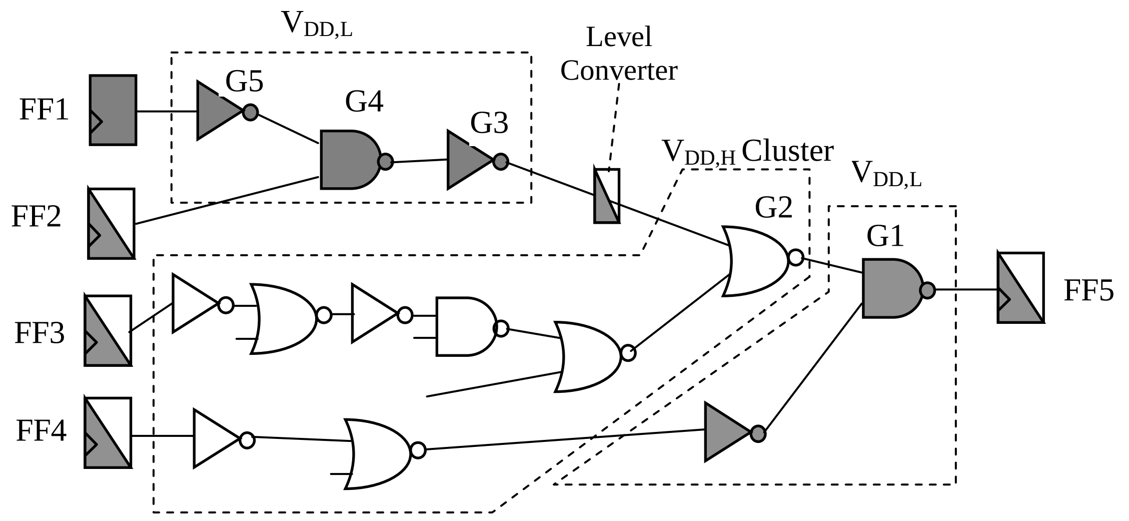


Figure 1: A typical level-converter design.

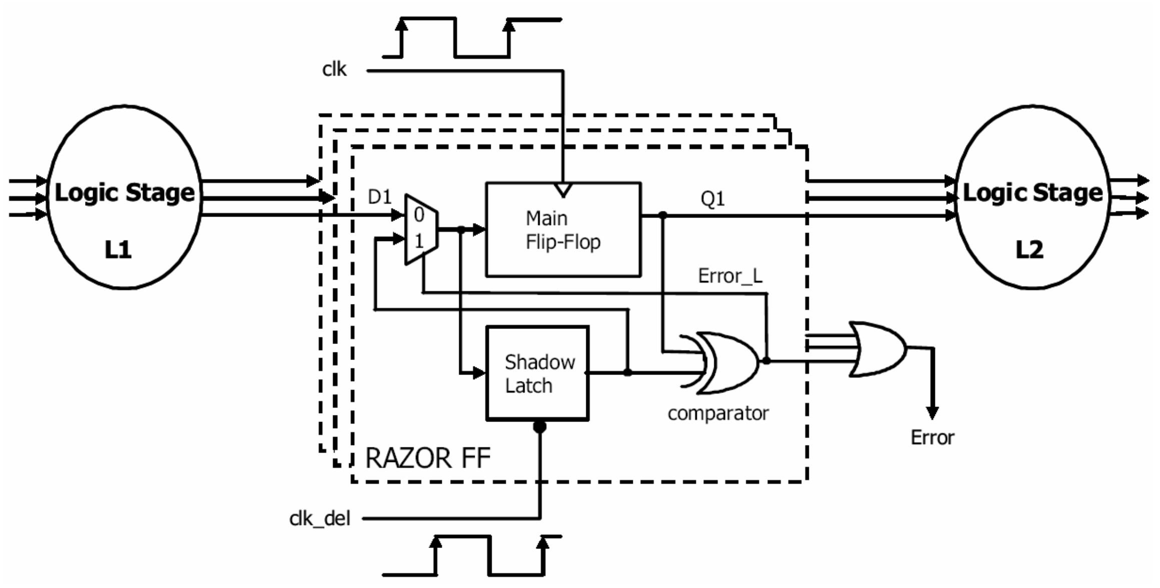


(a)

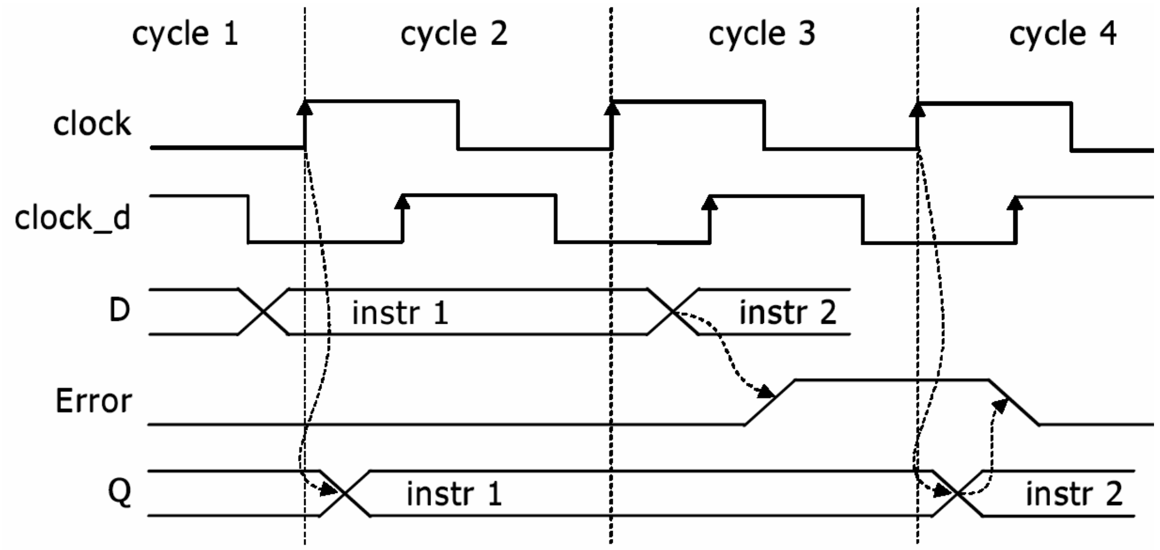


(b)

Figure 2: Examples of (a) CVS solution, (b) ECVS solution.



(a)



(b)

Figure 3. Illustration of Razor logic and DVS (a) Pipeline augmented with Razor latches. (b) Control lines for RAZOR flip-flops.

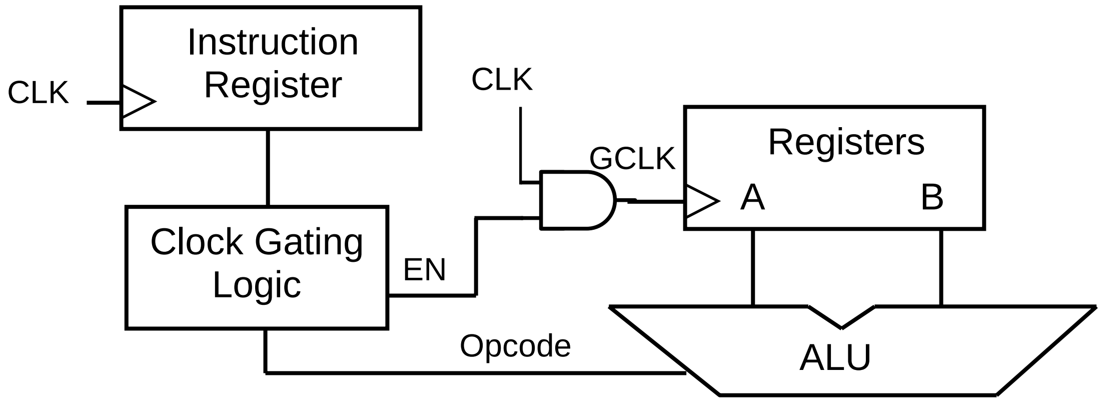


Figure 4: Clock gating logic for ALU in a typical processor microarchitecture with negative-edge triggered flip-flops.

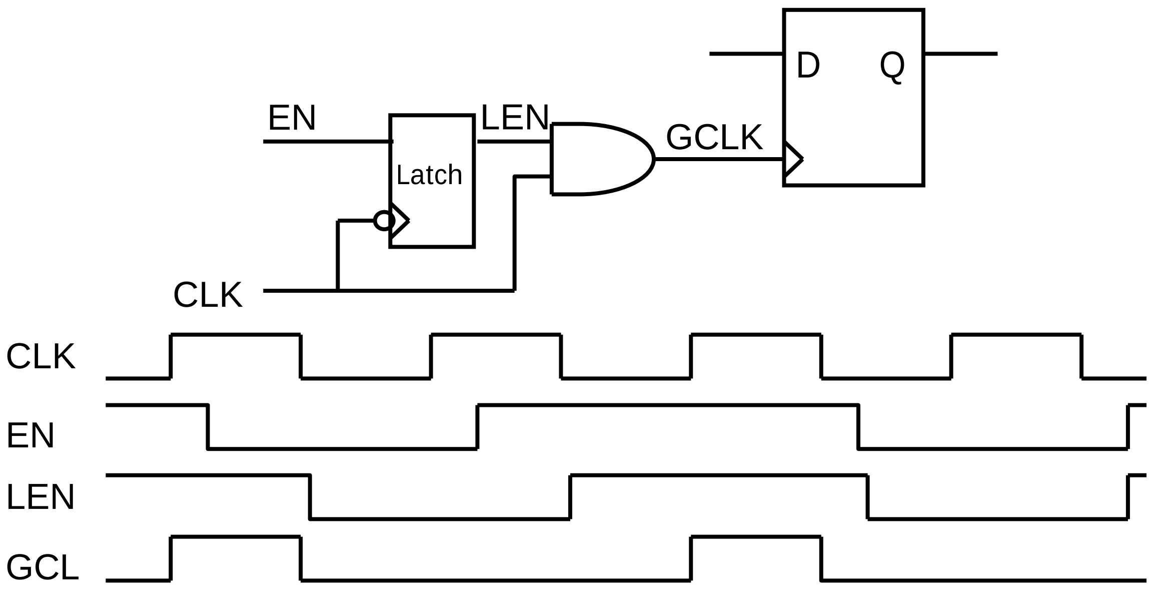


Figure 5: Clock is disabled when EN = 0; Furthermore, a hazard on EN will be stopped from reaching GCLK.

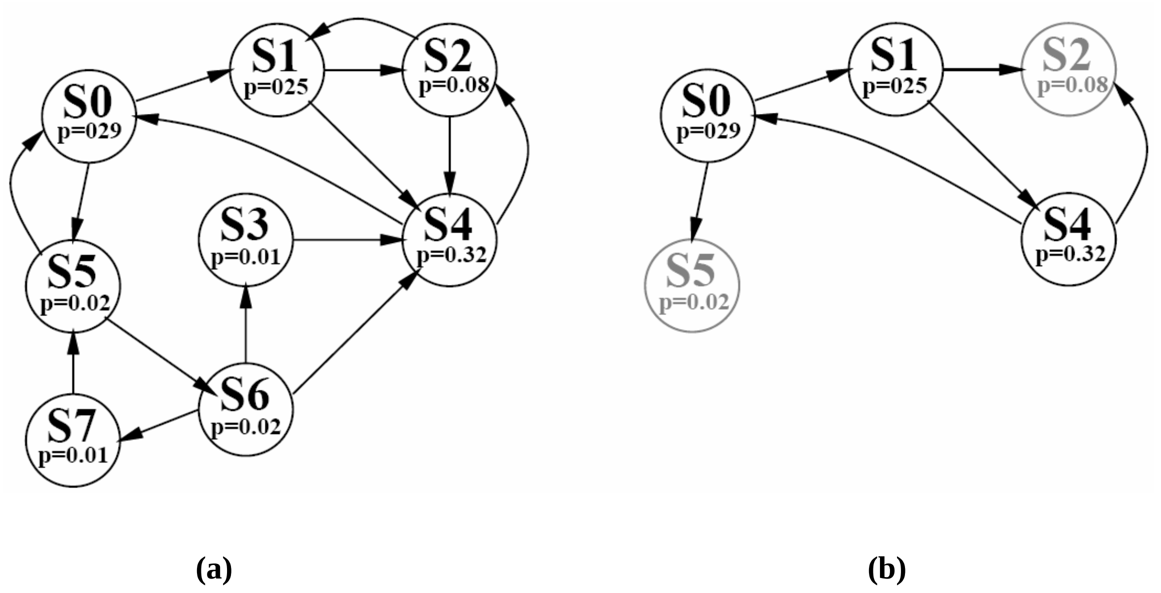
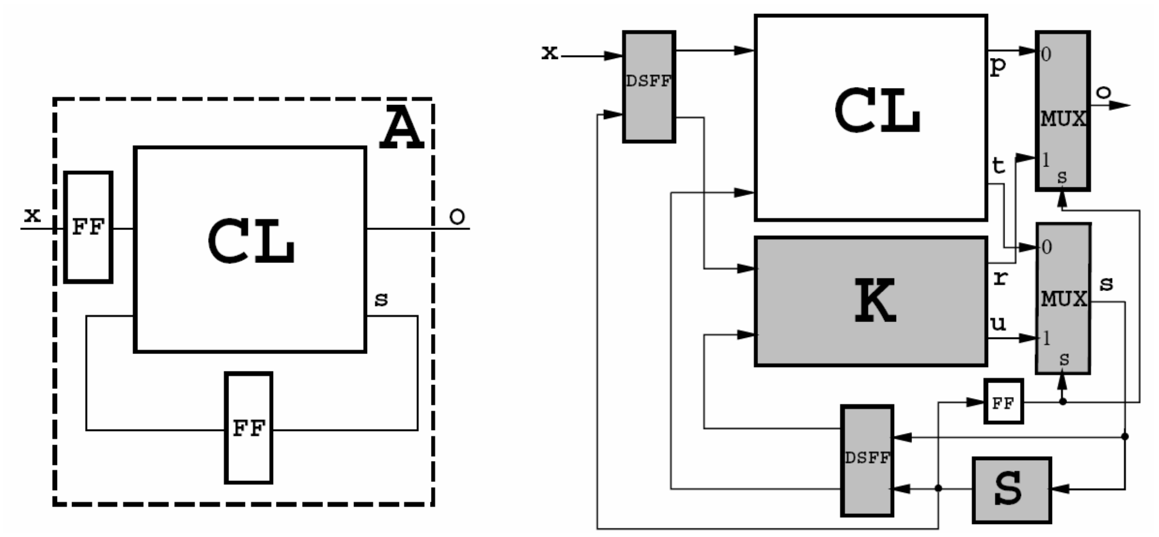


Figure 6: (a) Moore-type FSM and (b) its 0.25-order computational kernel.



(a) (b)

Figure 7: Illustration of computational kernel utilization (a) Baseline architecture (b) Kernel-based optimized architecture.

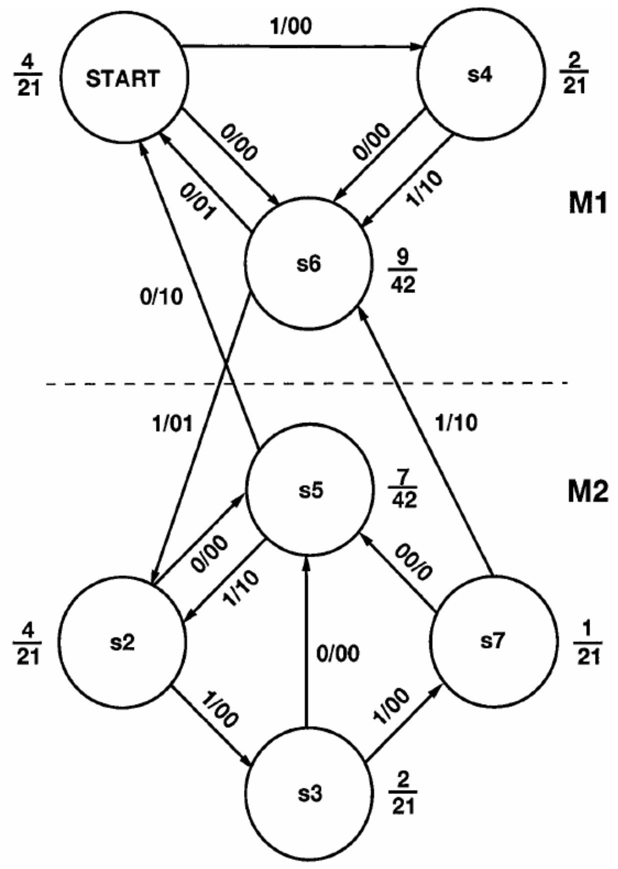
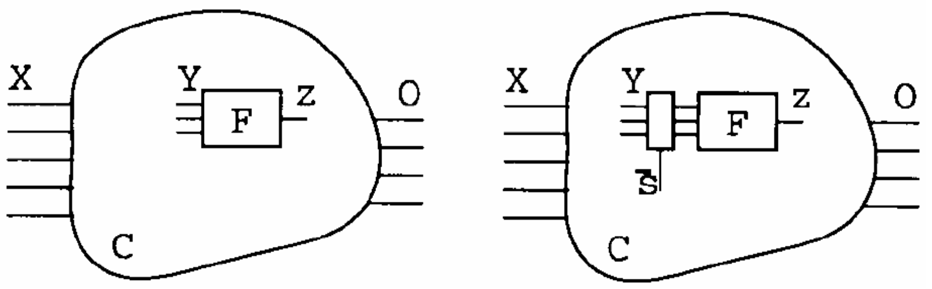


Figure 8: Example of an FSM (dk27) that may be decomposed into two sub-FSMs such that one sub-FSM can be shut off when the other is active and vice versa.



(a) (b)

Figure 9: Example of guard logic insertion.

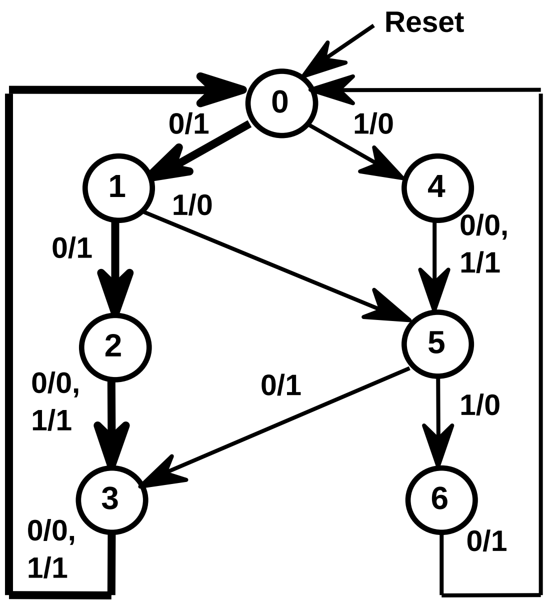
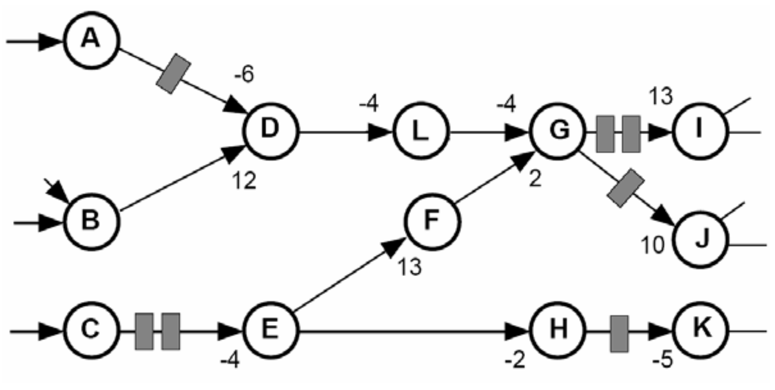
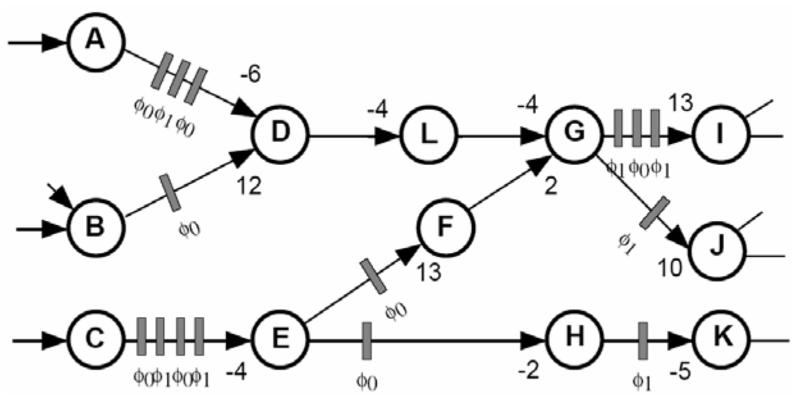


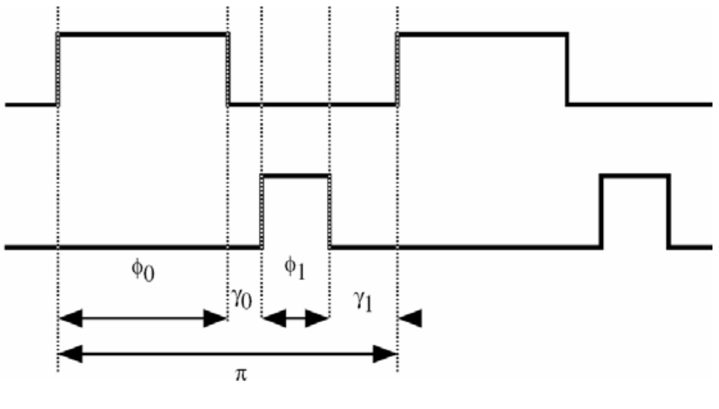
Figure 10: Excess-3 Converter state transition graph.



(a)



(b)



(c)

Figure 11: Illustration of fixed-phase retiming. (a) Initial edge-triggered circuit. (b) Fixed-phase retimed circuit. (c) A two-phase clocking scheme $\pi = \langle \phi_0 = 4, \gamma_0 = 1, \phi_1 = 4, \gamma_1 = 1 \rangle$.

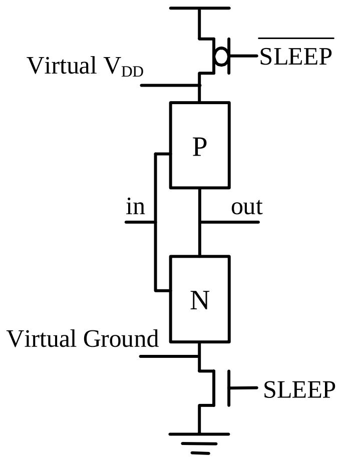


Figure 12: Power gating circuit.

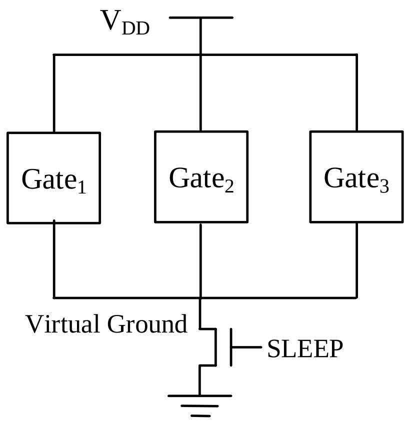


Figure 13: Using one sleep transistor for several gates.