Chapter 1

LOGIC SYNTHESIS FOR LOW POWER

Luca Benini
DEIS - Universitá di Bologna
lbenini@deis.unibo.it

Giovanni De Micheli
CSL - Stanford University
nanni@galileo.stanford.edu

Abstract Energy-efficient design of integrated circuits requires specialized tools and technologies. This chapter surveys some of the most important contributions in logic synthesis for achieving low-power consumption, by means of gate-level and register-transfer level restructuring. It presents also specialized techniques that leverage specific low-power silicon technologies.

Keywords: power optimization, dynamic power, static power, leakage power, voltage scaling, clock gating

1. Introduction

The design of integrated circuits (ICs) is highly constrained by their power dissipation for several reasons. First, many ICs are employed in mobile battery-powered systems, where the lifetime of the battery decreases as the power consumption of ICs and peripherals grows. Second, low-power design is required to either satisfy technical feasibility from a thermal profile standpoint, or to reduce the cost of the package and cooling means. Third, consuming electrical power costs and depletes non-renewable resources. Thus, economic, ecological and ethical reasons mandate the development of energy-efficient integrated circuits [1, 2, 3, 4].

Power minimization may target *average power* and/or *maximum instantaneous power* (called also *peak power*). Battery lifetime [13] and thermal energy dissipation [14] are impacted primarily by average power. Peak power is critical for power grid and power supply circuits design. Even though peak power is a serious issue, we will focus on average power optimization. In many cases, power can be reduced at the price of some performance degradation. For this reason, several metrics that account for both power and performance have been introduced in the past [15]. In many designs, the *power-delay product* (i.e., energy) is an acceptable metric to minimize. Alternatively, low-power design can be seen as a *constrained optimization* problem, where performance degradation is acceptable up to a given limit.

It has been observed many times that the most significant power savings can be achieved at high levels of abstraction, during early phases of the design process. However, an effective design flow should be power-conscious in its entirety, and energy efficiency should be pursued at every level of the design, from conception of functional models down to physical design. As the level of abstraction lowers, the number of design elements grows, and design automation is required to manage complexity. Hence, power optimization at the logic level cannot rely on human ingenuity alone, but it critically depends on *computer-aided design* (CAD) tools.

The main purpose of this chapter is to survey a few key concepts and techniques that have been developed for reducing power consumption at the logic level. We do not intend to cover in depth the vast literature on logic synthesis for low power (the interested reader is referred to previously published surveys [5, 7] and monographs [10]). Our objective is to identify basic cornerstones and major accomplishment in past work, and to identify areas of future challenges and opportunities for low-power logic synthesis in nanometer technologies (180nm and below).

1.1. Why is it so hard?

In CMOS technology, overall power consumption can be partitioned in three main components:

$$P = P_{dyn} + P_{sc} + P_{lk}$$

P_{dyn} is the *dynamic* or *switching power*. It is due to charging and discharging load capacitances. P_{sc}, called *short-circuit power*, is caused by the currents flowing from supply to ground when pairs of PMOS/NMOS transistors are conducting simultaneously. Finally, P_{lk}, called *leakage power*, is static in nature and it originates mainly from subthreshold MOS conduction. In most current CMOS IC technologies, P_{dyn} is predominant, but in deep submicron processes P_{lk} is significant. Moreover

its contribution is deemed to grow in the years to come. P_{sc} is generally dominated by the other two components, as long as the input and output transitions of the logic gates have similar duration. Design for low-power implies the ability of keeping all three components under control.

Switching power for CMOS circuits in a synchronous environment is modeled by the well-known equation:

$$P_{dyn} = \frac{1}{2} C_L A_{sw} V_{dd}^2 f_{clk}$$

where C_L is the load of a circuit node, V_{dd}^2 is the supply voltage, f_{clk} is the clock frequency and A_{sw} is the *switching activity* of the node, defined as the expected number of logic transitions during one clock cycle. P_{dyn} reduction targets the minimization of one or more factors in the equation above.

Supply voltage scaling has been the most widely adopted approach to power optimization, because of the quadratic dependence of P_{dyn} on V_{dd}. This approach, known as *power-driven voltage scaling* [16], assumes that voltage supply is a design variable subject to optimization. In this chapter we consider logic synthesis techniques under the assumption that supply voltage and frequency of operation are fixed, because they are determined by external compatibility constraints. This allows us to focus on logic restructuring techniques, that are orthogonal to voltage and frequency scaling. The assumption will be relaxed in Section 4.

Logic-level power optimization in this constrained (and common) setting targets the reduction of the *switched capacitance* $A_{sw} \cdot C_L$. Minimizing the switched capacitance is a difficult task primarily because of *estimation uncertainty*, as pointed out by Brand and Visweswariah [18]. Uncertainty on $A_{sw} \cdot C_L$ comes from imperfect knowledge of both A_{sw} and C_L during logic synthesis. Switching activity in internal nodes depends on input patterns, as well as on the internal structure of the logic networks. Both are generally not completely known during logic synthesis. Similarly, switched capacitance is generally uncertain before physical design, or, more so, when the netlist is not fully mapped to a technology library. Hence, the impact of logic synthesis on power can be compromised by estimation noise.

Another, often overlooked, factor limiting the impact of logic synthesis for low power is its scope of applicability. Logic synthesis is typically applied to the components of ICs implementing application specific logic functions (e.g., control and interface units). On the contrary, logic synthesis is seldom applied to data paths. Moreover, many ICs (e.g., processors [17]) contain large blocks that are custom-designed as well as memory arrays, where synthesis is not applicable. In these cases, logic

synthesis has little impact on overall power dissipation. Notice that an analogous conclusion does not hold for worst-case metrics, such as speed and signal integrity. Logic synthesis targeting worst-case metrics is critical even when only a small fraction of the system is synthesizable.

1.2. Two basic ideas

In the following sections we will examine several techniques for synthesizing energy-efficient circuits. We shall see that most approaches focus primarily on one of the following objectives:

- Minimization of *switching activity*. The circuit is transformed by adding logic that localizes computation in such a way that switching is substantially reduced. Area (i.e., capacitance) may increase because of the added logic. These transformations generally target coarse granularity blocks, to ensure that the cost of the added logic is amortized by significant switching activity reduction on many circuit nodes at the same time.

- Minimization of *switched capacitance*. In this case, transformations directly optimize logic-level approximations of dynamic power consumption. These techniques are generally local in scope, and overall power reduction is the compound effect of a large number of local transformations.

The boundary between the two objectives is not sharp, and several approaches achieve both, to some measure. However, switching reduction techniques have generally a non-negligible area overhead, as opposed to functional power minimization techniques that tend to reduce area.

This chapter is organized as follows. We consider first synthesis techniques at the gate level (Section 2). Most of such approaches target the reduction of switched capacitance at a fine granularity level. Next we address circuit optimization at the *register-transfer level* (RTL). (Section 3). In contrast, most of these techniques focus on switching-activity reduction. Last, we briefly mention the essential features of new technologies designed for low-power consumption, and we describe the specific synthesis and optimization methods that apply to them. (Section 4.)

2. Gate-level techniques

Gate-level power optimization techniques follow the same flow as traditional logic synthesis, ad depicted in Figure 1.1. Optimization is carried out in three steps, namely *technology independent* transformations, *library binding* and *re-mapping* optimizations. The synthesis flow must

be supported by a tightly coupled estimation flow, that drives the optimization process.

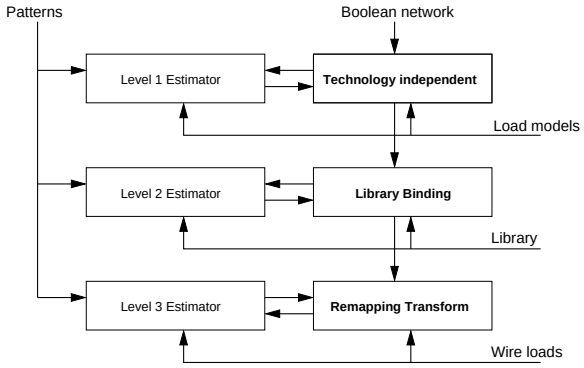


Figure 1.1. Gate-level Power Optimization Flow.

2.1. Technology-independent synthesis

At the top level, several technology-independent techniques for power minimization have been proposed in the last decade, and they are surveyed in detail in [5, 10]. The basic algebraic techniques (see Chapter **??** of this book) have been revisited, by modifying the cost metric that drives the optimization engine. Algebraic transformations for low power include common sub-expression extraction (targeting common cubes and/or kernels), node factorization, substitution via algebraic division, selective collapse. In all cases, the search techniques are similar to those developed for area reduction, but the literal count metric $N_{lit} = \sum_i l_i$ is replaced by:

$$P = \sum_i l_i A_{sw_i}$$

where l_i is the number of occurrences of literal i and A_{sw_i} is the switching activity of the literal. This *switched literal* metric is correlated with switched capacitance, and, by consequence it also correlates with dynamic power.

Example 1 *Consider the logic function* $f = abc + ad + cd$*, and three factorizations:* $f^1 = a(bc+d)+cd$*,* $f^2 = c(ab+d)+ad$*,* $f^3 = d(a+c)+abc$*. Assume that* $A_{sw_a} = 0.5$*,* $A_{sw_b} = 0.2$*,* $A_{sw_c} = 0.1$*,* $A_{sw_d} = 0.1$*. Then,*

the power metric is minimal for f^1. *In fact, we have* $P^1 = A_{sw_a} + A_{sw_b} + 2A_{sw_c} + 2A_{sw_d} = 1.1$, $P^2 = 2A_{sw_a} + A_{sw_b} + A_{sw_c} + 2A_{sw_d} = 1.5$, $P^3 = 2A_{sw_a} + A_{sw_b} + 2A_{sw_c} + A_{sw_d} = 1.5$. *Notice that literal count is the same for all three factorizations.*

In contrast with algebraic approaches, Boolean techniques for technology-independent logic optimization exploit the flexibility provided by *don't care* conditions to manipulate a logic network. *Don't care*-based optimizations can change the local functionality of parts of the network, provided that the input-output behavior is unchanged (or compatible with specified external *don't care* conditions). In principle, Boolean optimizations are more general and powerful than algebraic transformations, but they are very computationally intensive in the computation and propagation of *don't cares*. When the set of *don't care* conditions have been computed for a node in the logic network, its simplification is a local problem, and it generally does not pose computational challenges. However, when targeting power optimization, the matter is more complex. In fact, optimizing a node's function f may change the switching activity at the node's output. This variation propagates to fanout nodes. Hence, a locally advantageous move can be globally harmful.

Example 2 *Consider the node function* $f = ab'c + a'b$ *with don't care* $f_{DC} = c'$. *A compatible implementation of the function is* $g = ab' + a'c$. *The switched-literal metric for* f *is* $P(f) = 2A_{sw_a} + 2A_{sw_b} + A_{sw_c}$. *For* g, *we have* $P(g) = 2A_{sw_a} + 2A_{sw_b}$. *We have* $P(f) > P(g)$, *hence* g *is locally a valid replacement for* f. *However, the switching activity on the node's output can increase. For instance, if* $Prob_{a=1} = 0.5$, $Prob_{b=1} = 0.5$, $Prob_{c=1} = 0.1$ *and all inputs are mutually independent and uncorrelated in time, we have* $A_{sw_f} = 0.377$, $A_{sw_g} = 0.5$. *If the node has a large fanout, its increased switching activity may cause an increase in the global switched-literal metric, despite the local decrease.*

To address this problem, two approaches have been followed. The first is to recompute switching in fanout cones of nodes that have been optimized, and stop recomputation when the perturbation on fanout switching "dies out". The second is to restrict the *don't care* set available for optimization to a safe subset called *power relevant don't cares* [10]. Using power relevant *don't cares* guarantees that switching does not increase in the fanout of the optimized node. Clearly, the second approach leads to more conservative optimizations.

The main task of the *Level 1* estimation engine (of Figure 1.1) coupled with technology-independent optimization is to compute switching activity for all literals in the network, and to update it when optimization

modifies its structure. At this level of abstraction, a *zero-delay* approximation of A_{sw} is used, where it is assumed that all nodes compute their output values instantaneously in response to input changes. Switching computation can either be based on *probabilistic* (or static) or on *statistical* (or dynamic) analysis [19]. Probabilistic techniques are generally faster, but less accurate than statistical techniques, that involve simulation of a pattern set. Hybrid estimators have also been proposed, where probabilistic analysis is employed only in the inner optimization loop.

Even though technology-independent optimizations have been extensively studied and prototype tools are publicly available [20], these techniques are not mainstream in the industrial practice. Probably, the main reason for this fact is that the improvements with respect to area-optimized circuits in switched literals are small in average (10% to 15%). Furthermore, the switched-literal metric is only a very rough approximation of actual dynamic power consumption, and small percentage improvements are well below the noise level of the approximation [18]. Thus, a reasonable and low-effort approach is to apply traditional area-oriented technology independent synthesis, and to introduce specialized power-oriented transformations in later synthesis steps.

2.2. Library binding

Library binding (also called *technology mapping*) moves from a technology-independent logic network and maps it onto gates taken from a target *technology library* (refer to Chapter **??** for a review). The key difference between technology independent power optimization and low-power technology mapping is that the latter can rely on more detailed and accurate power estimation. A this stage, overall power dissipation is partitioned in two contributions: *internal* or *cell* power and *external* or *node* power. Internal power is mainly due to short-circuit currents and charging and discharging of parasitic capacitances within a gate. External power is consumed for switching the capacitive load of a gate. Optimal mapping requires careful balancing of the two components, while satisfying side constraints (e.g., timing).

Level 2 estimation engines (of Figure 1.1) still rely on switching activity computation based on either static or dynamic propagation of input switching, but they require power models for all gates in the technology library. Internal power is modeled by lookup tables, indexed by switching activities and other data extracted from the gate's boundary (e.g., output load). External power estimation exploits the knowledge of gate input capacitances, which are known for all gates in the technology library.

Example 3 *The internal power model in Synopsys' Power Compiler [39] is based on a two-dimensional lookup table, indexed by output load capacitance (C_{out}) and weighted average input transition time (ω). The second index is obtained by computing the weighted average of each input transition time, weighted with the input's switching activity. In symbols: $P_{int} = LUT(C_{out}, \omega) \cdot A_{SW_{out}}$ with ω given by:*

$$\omega = \frac{\sum_{i \in Inputs} t_i A_{SW_i}}{\sum_{i \in Inputs} A_{SW_i}}$$

where, t_i is the transition time of input i, and A_{SW_i} its switching activity.

Even though *Level 2* estimators are much more accurate than *Level 1* estimators, it is important to stress that they are still affected by significant inaccuracies, caused by the lack of information on wiring-related effects (e.g., additional capacitive load, slope reduction, cross-coupling effects).

Technology mapping usually starts with a *decomposition* of the initial Boolean network into a small set of elementary logic functions. Decomposition is followed by *covering*. During covering, fragments of the decomposed network are *matched* with gates from the technology library. Matching tests for functional equivalence between portions of the decomposed network and cells in the library. Among all matching library elements, the mapping algorithm should select the one that minimizes the target cost metric. The critical issue in technology mapping is that a sequence of optimal choices provides an optimum solution only under stringent assumptions. Unfortunately, such assumptions do not hold in general when the objective function is power consumption. Thus heuristics have been used to guarantee good-quality (but not provably optimum) results.

All heuristic low-power mapping solutions involve iterative exploration over the large search space of alternative coverings and matchings. A successful approach is based on *dynamic programming*, which is quite effective in pruning the search space [10], by storing locally optimal solutions and re-using them as covering proceeds. Furthermore, dynamic programming can be adapted to deal with constrained optimization (e.g. power with delay constraints), by storing sets of solutions representing Pareto points of the speed vs. power design space.

With respect to technology independent optimization, low-power library binding produces more reliable results, because it accurately accounts for gate input capacitances (and their effect on external power) and for internal power. Several prototype mappers have been implemented [10], and they have reported 10% to 15% power savings, on average, with respect to area-optimized circuits with the same speed. This

is further evidence that power and area reductions are usually positively correlated. Nevertheless, the corresponding optima may not coincide, as demonstrated by the following example.

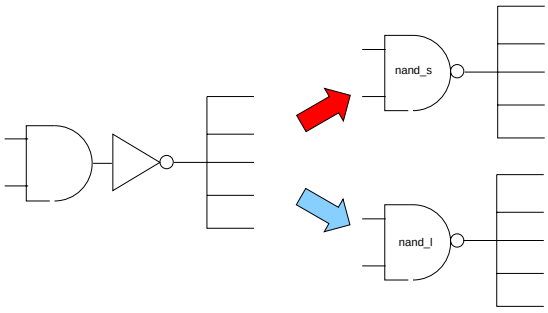


Figure 1.2. Alternative Mappings for area and power

Example 4 *Figure 1.2 shows a fragment of a decomposed netlist (AND-INV decomposition style), with a large-fanout node. Assume that speed constraints are loose in this part of the netlist. Mapping for area leads to the mapped circuit at the top right corner, where a minimum-size* `nand_s` *gate has been instantiated. In contrast, the minimum power implementation is shown at the bottom right corner. A larger-size* `nand_l` *has been instantiated. The mapping is different because of internal power in the cells driven by the nand gate. The minimum area solution causes slow input transitions on the fanout gates, thereby increasing their internal power (remember that internal power includes short-circuit power, which is adversely impacted by slow input transitions). The reduction of internal power in the fanout of* `nand_l` *also amortizes the increased external power due to its larger input capacitance.*

Low-power technology mapping has been implemented in commercial design tools [40]. A distinctive feature of the implementation described in [40] is the adoption of the *mapping graph* data structure, originally proposed by Lehman et.al. [41] to implicitly explore alternative logic decompositions during library binding. This technique helps in finding better matchings, at the price of increased computational effort for power estimation.

2.3. Re-mapping transformations

As the name suggests, re-mapping transformations are applied to gate-level netlists that are already mapped to a library, and they strive at improving the mapped netlist. The netlist is optimized through a large

number of local moves, that incrementally modify the original netlist. In principle, this approach should be less effective than technology independent optimizations and technology mapping because it is applied at a lower abstraction level, where the degrees of freedom for optimization are much reduced. In practice, this is not the case, and re-mapping transformations are currently the most successful gate-level power optimizations in commercial synthesis tools. There are three reasons for this fact. First, after mapping and after physical design, followed with back-annotation of wiring capacitances, power estimation reaches sign-off quality. Thus, power savings obtained during re-mapping are less affected by estimation noise. Second, re-mapping transformations can focus on hot spots (i.e., high-dissipation nodes and cells), which are precisely located by power analysis. Third, all top-down logic optimization steps are heuristic and advanced incremental synthesis [42] still finds significant room for improvement.

Basic re-mapping transformations include re-factoring, polarity assignment and pin swapping [39]. All these techniques are locally applied on the mapped netlist, and they focus on a single cell or a small group of cells.

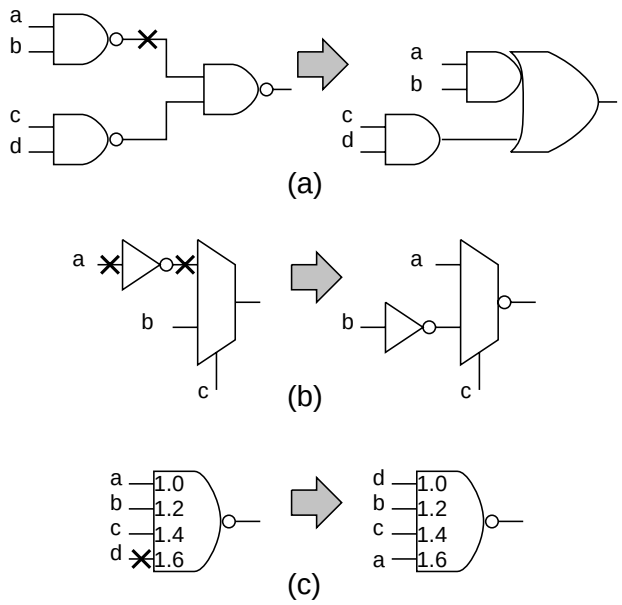


Figure 1.3. Local transformations: (a) re-factoring, (b) polarity assignment, (c) pin swapping.

Example 5 *Figure 1.3 shows three examples of local transformations. In Figure 1.3 (a) a re-factoring transformation is shown, where a high-activity node (marked with ×) is removed via a new mapping onto an* `and-or` *gate. In Figure 1.3 (b), polarity assignment is exploited to eliminate one of the two high-activity nets marked with ×. Pin swapping is applied in Figure 1.3 (c) to connect a high-activity net with the input pin of the 4-input* `nand` *with minimum input capacitance.*

More general and powerful re-mapping transformations are based on re-wiring [43, 44]. Re-wiring focuses on nets with high switching activity, and tries to eliminate them by finding alternative connections which make the high-switching net redundant. Rewiring also enables more aggressive cell re-mapping as a side effect. Most rewiring approaches exploit algorithms developed in the testing field for automatic test pattern generation [11]. Rewiring algorithms find degrees of freedom in a gate-level netlist in a very computationally efficient way, and they have been successfully applied to large netlists. A related approach, *generalized mapping*, targets the replacement of two or more cells at the same time[45].

Example 6 *The example of Figure 1.4, taken from [43], illustrates the nature of rewiring transformations. The input signal probabilities are* $Prob(a = 1) = Prob(b = 1) = Prob(c = 1) = 0.1$. *If we assume temporally uncorrelated and spatially independent inputs, switching activity for the nodes can be computed as* $2(Prob)(1 - Prob)$. *The circuit to the left is rewired by replacing the connection between input* `a` *and the* `XOR` *gate with a connection between* `e` *and the* `XOR` *gate. Obviously, re-wiring does not alter the input-output functionality, but it does decrease the switching activity at the output of the* `XOR` *gate, which goes from* 0.295 *to* 0.193.

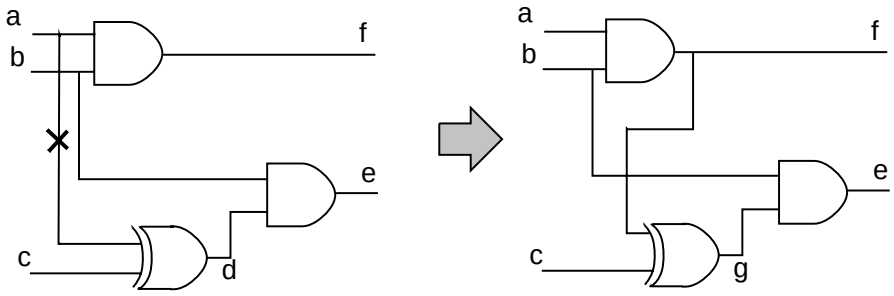


Figure 1.4. Rewiring transformation for low power.

Even though prototype implementations of rewiring and generalized matching have obtained promising results (15% to 20% in average, over power optimized netlists) we are not aware of any implementation of these techniques in commercial tools. Given the quality of results achieved, we expect that these techniques will be applied in the industrial practice in the near future, as more designs become tightly power constrained.

Another effective post-mapping technique is path equalization. Path equalization ensures that signal propagation from inputs to outputs of a logic network follows paths of similar length. When paths are equalized, most gates have aligned transitions at their inputs, thereby minimizing spurious switching activity (which is created by misaligned input transitions). This technique is very helpful in arithmetic circuits, such as counters (e.g., carry-save adders) of multipliers.

Example 7 *The multiply-accumulate (MAC) unit of the StrongARM processor [51] is based on a Wallace-tree multiplier coupled with a carry-lookahead adder. The Wallace-tree architecture was chosen because it is very fast, but also because it has low dynamic power consumption in the carry-save adder tree. The improvement comes from a sizable reduction in spurious switching, due to path delay balancing in the Wallace-tree. A 23% power reduction (as well as a 25% speedup) is achieved by the Wallace-tree architecture with respect to the array multiplier.*

Synthesized logic has much more irregular structure than arithmetic units, and its gate-level implementation is characterized by a wide distribution of path delays. These circuits can be optimized for power by automated *resizing*. Resizing focuses on fast combinational paths. Gates on fast paths are down-sized, thereby decreasing their input capacitance, while at the same time slowing down signal propagation. By slowing down fast paths, propagation delays are equalized, and power is reduced by joint spurious switching and capacitance reduction. Resizing does not always imply down-sizing. Power can be reduced also by enlarging (or buffering) heavily loaded gates, to increase their output slew rate. Fast transitions minimize short-circuit power of the gates in the fanout of the gate which has been sized up, but its input capacitance is increased.

Resizing is a complex optimization problem involving a tradeoff between output switching power and internal short-circuit power on several gates at the same time. Several resizing algorithms have been proposed in the literature (the reader is referred to [46, 47, 50] for more details), but it is unclear if they would produce large power savings when applied to aggressively power optimized netlists, where mapping (and re-

mapping) may have already found optimal gate sizes for large sections of the circuit.

The *Level 3* estimators (of Figure 1.1) supporting post-mapping optimization can in principle rely on detailed timing models and load capacitance information. However, timing-accurate, full delay simulation is generally too slow to be used within an iterative optimization loop. Zero-delay simulation is therefore employed during optimization, with unavoidable accuracy loss. Full-delay simulation is run only to validate results, at the end of the optimization process.

2.4. Library design

Cell library design has a strong influence on power dissipation [52]. From this viewpoint, probably the most critical cells in a digital library are sequential primitives, namely, latches and registers. First, flip-flops are extremely numerous in today's deeply pipelined circuits, second, they are connected with the most active network in the chip, the clock. Clock drivers are almost invariantly one of the largest contributors to the power budget of a chip, primarily because of the huge capacitive load of the clock distribution network. Flip-flop (and latch) design for low power focuses on minimizing clock load and reducing internal power dissipation when the clock signal is toggled. Significant power reductions have been achieved by carefully designing and sizing flip-flops [1].

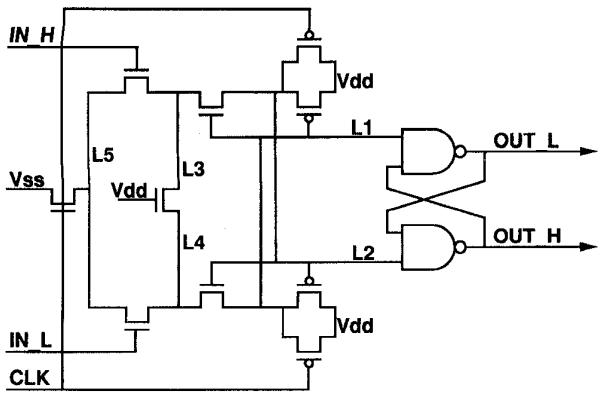


Figure 1.5. Low-Power Flip-Flop Used in the StrongARM Design.

Example 8 *The edge-triggered flip-flop shown in Figure 1.5 was used in the StrongARM processor [51] to reduce clock load. The flip-flop (FF)*

features a differential structure, similar to a sense amplifier. The internal structure of the FF is quite complex in comparison with a simple flow-through latch, such that used in the Alpha chip (the design from which the first StrongARM version was derived). However, the FF has small clock load (only three transistors). This key advantage gave a $1.3\times$ *overall power reduction over the latch-based design.*

Transistor sizing is also exploited to minimize power consumption in combinational logic cells. Rich libraries with many available transistor sizes are very useful in low-power design, because they help synthesis tools in achieving optimum sizing for a wide range of gate loads. Power savings can be obtained by adopting non-conventional logic implementation styles such as pass-transistor logic [23], which can reduce the number of transistors (and, consequently the capacitive load), for implementing logic functions which are common in arithmetic circuits (e.g., exclusive-or, multiplexers).

2.5. Summary of gate-level techniques

Gate-level power minimization is relatively well studied and understood. Unfortunately, due to the local nature of most optimizations, a large number of transformations has to be applied to achieve sizable power savings. This is a time consuming and uncertain process, where uncertainty is caused by the limited accuracy of power estimation. In many cases the saving produced by a local move are below the "noise floor" of the power estimation engine. As a consequence, logic-level optimization does not result in massive power reductions. Savings are in the 10% to 20% range, on average. Based on published results, we speculate that an additional 10% savings could be obtained though aggressive post-mapping optimizations which have not been implemented yet in commercial tools.

3. Register-transfer level techniques

Most digital circuits today are designed starting from specifications written in high-level hardware description languages (HDLs). Current synthesis tools impose restrictions on the generality of language constructs that can be used in a HDL specification, by defining a *synthesizable subset* of the language. We use the term *register transfer level* (RTL) to refer to synthesizable design specifications. Most synthesis tools parse the HDL description into a structural netlist of technology-independent *logic primitives*, which are then optimized and mapped to library gates during logic synthesis. Register-transfer level transformations are applied on the technology-independent netlist before logic synthesis.

At this level of abstraction, the tool has a global view of the architecture and of its various computational units. In a complex circuit, not every unit performs useful computation at every clock cycle. Local idleness can be exploited to save power by disabling the clock of idle units. This approach is known by *clock gating* and it is widely used by digital designers when power is a concern. Clock gating saves power at a coarse granularity, by reducing useless switching activity in idle units at the cost of some additional hardware for detecting idleness and stopping the clock. Needless to say, clock gating has deep implications on design testability [28], and it may impact performance (the gating logic may be on the critical path, and clock gating may increase clock skew). These effects must be taken in account very carefully when implementing a clock gating strategy.

Example 9 *The TMS320C5x DSP processor [30] heavily exploits clock gating to save power. It implements a two-level power reduction stategy (global and local). The clock signal feeding latches at the inputs of functional units is enabled only when useful data are available at the units' inputs. The gating signals are automatically generated by local control logic using information coming from the instruction decoder. Global clock gating is also available. It is software-controlled through dedicated power-down instructions,* `IDLE1`*,* `IDLE2` *and* `IDLE3`*, which enable power management with increasing strength and aggressiveness. Instruction* `IDLE1` *only stops the CPU clock, while it leaves peripherals and system clock active. Instruction* `IDLE2` *also deactivates all the on-chip peripherals. Finally, instruction* `IDLE3` *powers down the whole processor.*

Automated clock-gating insertion is a RTL transformation, because it leverages information on design architecture (e.g., identification of unit boundaries) that is generally lost when synthesis is carried out to completion. Gated-clock synthesis has been studied extensively in the literature. Consider the block diagram of a sequential circuit, shown on the left of Figure 1.6. The circuit consists of a combinational logic block and a set of flip-flops (a register) which store the inputs and some of the outputs of the combinational block (the *next-state* outputs). The corresponding gated-clock architecture is shown on the right.

For the sake of illustration, we assume a single-clock strategy with edge-triggered flip-flops. The combinational block F_a is controlled by the primary inputs and the present-state inputs and it implements the idleness-detection logic, also called *activation function* (in negative logic: when it has value 1 the clock is stopped). Its purpose is to stop the clock when the computation performed by the unit is either unneeded or redundant. Block L is a latch, transparent when the global clock

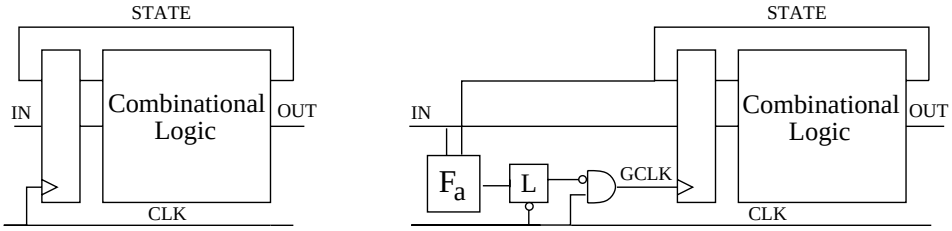


Figure 1.6. Example of Gated Clock Architecture.

signal *CLK* is inactive. Its presence is essential for a correct operation of the system, since it filters glitches that may occur at the output of block F_a.

The activation function is synthesized from the Boolean function representing the idle conditions of the circuit. This is a challenging synthesis problem. In fact, stopping the clock for every idle condition may lead to an implementation of F_a that is too large, slow and power-consuming. Thus, we may be forced to synthesize only a simplified activation function, which dissipates minimal power, but stops the clock with maximum efficiency [27, 29].

Clock-gating well applies to the control portions of integrated circuits. Some specific techniques have also been devised for data-path switching reduction. Data paths have long and wide busses, with significant capacitive loading. Thus, reducing switching activity on data-path busses is a very useful power reduction strategy.

An effective approach for reducing switching activity is based on observability analysis [48]. Namely, bus switching is inhibited when the values they carry are not observed at the data-path boundary. Since data paths perform a wide range of operations, it is often the case that internal information is irrelevant during some cycles of the data-path operation. Switching reduction is achieved by controlling registers, multiplexers and tri-state drivers, which do not update the information on selected unobservable lines, thus avoiding the switching.

Synthesizing data paths with this low power feature entails constructing appropriate control circuitry for the corresponding registers, multiplexers and drivers. This, in turn, requires an observability analysis. Whereas a full observability analysis may be hard and lead to complex local controllers, a safe and simplified observability analysis can be achieved by assuming that arithmetic blocks (e.g., adders) are fully transparent. Thus signal observability can be derived by analyzing propagation through steering elements only [48].

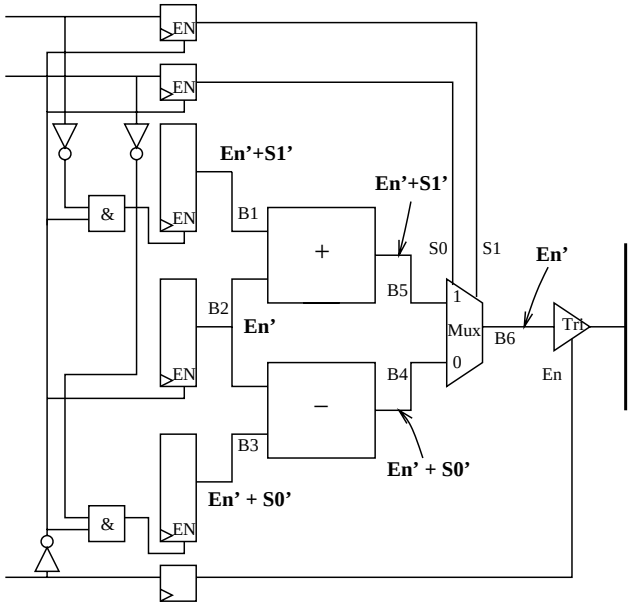


Figure 1.7. Example of data-path power management.

Example 10 *The data-path of Figure 1.7 has two steering levels. Three-state driver Tri has level 2. Multiplexer Mux has level 1. The input registers have level 0. We assume that the output bus is observable. The observability don't cares (ODCs) for all internal busses* $B1$, $B2$, ..., $B6$, *are shown in the figure (in boldface). For instance, consider bus* $B2$. *It has two fanouts to two computational units. The ODC of the input of a computational unit is the ODC of the output, because we consider them as transparent. Hence, the ODC of* $B2$ *is the intersection of the ODCs of the outputs of the computational units:* $ODC_{B2} = (En' + S_1')(En' + S_0') = En' + S_1'S_0' = En'$, *because* S_1 *and* S_0 *cannot be zero at the same time for correct multiplexing.*

In order to prevent useless switching activity on bus $B6$ *the flip-flops of the control inputs of the multiplexer are disabled by* En'. *Switching is reduced on* $B1$, $B2$, ..., $B5$ *by disabling the registers on level 0. From the schematic, it is evident that the insertion of power management circuitry has minimal area impact (only three inverters and two AND gates are added).*

Closely related to datapath gating, *operand isolation* inserts transparent latches on the input of (some) resources, and creates control circuitry so that such latches are opaque when the corresponding resource's result is not observed. This approach isolates the operand from the resource,

thus avoiding useless power dissipation. Operand isolation has often been used in industrial practice [17].

Clock gating and operand isolation are not the only options for RTL power reduction. Other approaches like *precomputation* [32, 33] and *decomposition* [34, 35, 49] aggressively modify the circuit to enhance the impact of clock gating. The basic rationale of these techniques is to create a small (and power-efficient) circuit that can bypass the computation of a much larger functional unit for a possibly small, but highly probable, subset of input patterns. Even though these techniques have shown promising results, they have not been thoroughly assessed in the industrial practice.

3.1. Summary of RTL techniques

Register-transfer level techniques aim at reducing switching activity by monitoring the idleness of its components and/or the observability of some components' outputs. Clock gating, and its enhancements, has been widely used in low-power digital design, and it has been very successful thanks to its conceptual simplicity and wide applicability, even though its implications on testability and circuit performance must be taken into account very carefully. Commercial tools for clock-gating insertion or just clock-gating support are available [36]. Moreover, clock-gating and data-path gating techniques are synergistic, and their combined application can provide significant energy savings.

4. The evolution of low-power synthesis

Integrated circuit scaling is probably one of the most astonishing achievements of modern technology. Elementary devices have been shrunk by factor $k = 0.7$ per technology generation for the last thirty years. Geometric scaling has profound implications on the electrical characteristics of active devices, and care must be take to preserve satisfactory transistor operation. *Constant-field scaling* [37], imposes the downscaling of silicon doping levels to maintain constant field across the gate oxide of the scaled MOS transistor. In this regime, power dissipation scales down with k^2. Power density (i.e., power dissipated per unit area) remains constant and speed scales up as $1/k$.

Figure 1.8 shows the average power consumption and power density evolution for mainstream Intel Microprocessors, versus technology generation [37, 38]. In contrast with constant field scaling, both average power and power density increase as minimum feature size shrinks. This trend is due to two factors. First, die size has been increasing over time; second, voltage supply has not been downscaled in accordance to

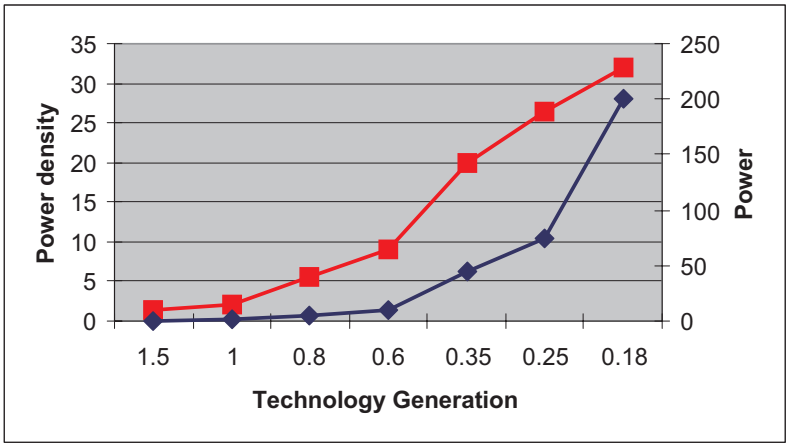


Figure 1.8. Power (diamonds) and Power Density (squares) vs. Technology Generation.

constant-field scaling. Supply scaling lags device scaling because supply voltage levels have been standardized (5 V, then 3.3 V), and also because transistors switch faster when we rise the electric field ("overdriving" the transistor).

4.1. Low-Power technologies

In scaled transistors (below $0.8\mu n$) overdriving gives marginal speed gains and it is highly energy-inefficient, because electrons reach a limit speed when traveling across the channel (a phenomenon known as *velocity saturation*). For this reason, the most direct way to reduce power consumption, known as *power-driven voltage scaling* [3, 53], is to scale down the voltage supply. Depending on the relative importance of performance versus power, different voltage levels can be adopted. In more detail, since transistor speed does not depend on supply voltage V_{DD} alone, but on the *gate overdrive* $(V_{DD} - V_T)$, several researchers have studied the joint minimization V_{DD} and V_T for minimum energy, or minimum energy-delay product [3, 1].

The first-order quadratic model of CMOS ON-current $I_{DS} \propto (V_{GS} - V_T)^2$ leads to overly optimistic switching speed estimates for submicrometric transistors. In short-channel transistors, velocity saturation dictates a different current equation $I_{DS} \propto (V_{GS} - V_T)^m$, with $1 \leq m < 2$ (e.g., $m = 1.3$). Another important characteristic of CMOS transistors is sub-threshold conduction. When $V_{GS} < V_T$ the current is not zero, but it follows an exponential law $I_{DS} \propto e^{V_T/V_o}$, V_o being the technology-

dependent *subthreshold slope*. While velocity saturation pushes toward aggressive voltage scaling, sub-threshold conduction limits it, because of increased static current leaking through nominally OFF transistors. Intuitively, threshold and supply voltage optimization for minimum energy requires balancing ON-current and OFF-current, while at the same time maintaining acceptable performance.

The V_{DD} and V_T that minimize the energy-delay product are indeed very low: V_{DD} should be only slightly larger than $2V_T$, with V_T a few hundreds millivolts. Circuits operating in this regime are called *ultra-low-power* (ULP) CMOS [22]. Pushing toward ULP is extremely challenging, for three key reasons. First, in real-life processes, transistor thresholds cannot be perfectly controlled, and V_T can be slighly lower than nominal. As a result, many transistors may have sub-threshold currents that are orders of magnitude larger than expected (remember that sub-threshold current is exponential in $V_{GS}-V_T$). Second, sub-threshold current is exponentially dependent on temperature. Thus, ULP CMOS must be tightly thermally controlled, with adverse cost implications. Third, noise margins are greatly reduced in ULP CMOS because signal swings are extremely small. Many noise sources (such as cross-coupling, and both ionizing and non-ionizing radiation) can be seen as injecting spurious charges. Since capacitances and voltages scale down with technology, the relative frequency by which injected charges upset stored charges increases.

Example 11 *The StrongARM processor was initially designed in a three-metal, 0.35μm CMOS process, developed for high-performance processors (the DEC Alpha family). Supply voltage from 3.45 V to 1.5 V, with threshold voltage $V_{TN} = |V_{TP}| = 0.35$ V, obtaining a 5.3$\times$ power reduction. The performance loss caused by voltage scaling was acceptable, because StrongARM has looser performance requirements than Alpha.*

As a second example, the research prototype of the TMS320C5x DSP family adopted a supply voltage $V_{DD} = 1$ V in a 0.35μm technology. The aggressive V_{DD} value was chosen by optimizing the energy-delay product.

In recent technology generations (see Figure 1.9) supply voltage has started to decrease with shrinking device size even for high-performance transistors [38]. This trend is dictated by several reasons, such as gate oxide reliability, which are outside the scope this chapter [38]. From the point of view of power reduction, this trend has two important consequences. First, aggressively-scaled transistors with minimum channel length are becoming increasingly leaky in the OFF state and leakage power is rapidly gaining importance. Second, power-driven voltage scal-

ing cannot be a panacea any longer, because supply voltage is getting close to noise-induced limits.

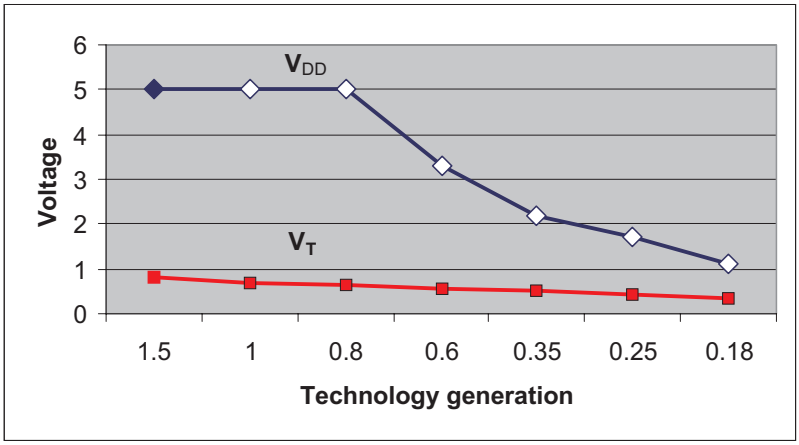


Figure 1.9. Threshold voltage (squares) and supply voltage (diamonds) vs. technology generation.

Leakage power containment is already a concern in current technologies, because it impacts battery lifetime when the circuit is quiescent. CMOS technology has traditionally been extremely power-efficient when transistors are not switching, and system designers expect low leakage from CMOS chips. To reduce leakage power in low-voltage technologies, *multiple threshold* and *variable threshold* circuits have been proposed [3]. Multiple-threshold CMOS require a fabrication process that creates transistors with two different thresholds. Low-threshold transistors are fast but leaky. These devices are employed on long, critical paths. High-threshold transistors are slower but they have minimal sub-threshold leakage. These devices can be employed in non-critical units/paths of the chip.

Power reduction based on multiple-threshold technologies is ineffective when most transistors are timing-critical. To overcome this limitation, it is possible to dynamically control threshold voltage through transistor substrate biasing. When a variable-threshold circuit is idle, the substrate of NMOS transistors is negatively biased, and their threshold increases because of the well known *body-bias effect* (PMOS transistors require positive body bias). On the negative side, variable-threshold approaches require additional body-bias control circuits.

Example 12 *The TMS320C5x DSP prototype [30] adopted a dual-threshold process to enhanced performance at the aggressively down-scaled* $V_{DD} =$

1 *V. The nominal threshold voltages were* 0.4 *V and* 0.2 *V for slow and fast transistors, respectively. Leakage current for the high-V_T transistors is below 1 nA/μm. For the low-V_T transistors, leakage current is below 1 μA/μm. The current of low-V_T transistors is typically twice that of the high-V_T devices.*

An MPEG4 Video Codec prototype [31] adopted the variable-threshold voltage scheme to improve energy efficiency. Substrate biasing is exploited to dynamically adjust the threshold: V_T is controlled to 0.2 *V in active mode and to* 0.55 *V when the chip is in standby.*

Supply voltage can be controlled as well to reduce power. *Multiple-voltage* and *variable voltage* techniques have proposed in the past [3]. In multiple-voltage circuits two or more power supply voltages are available on chip. Similarly to multiple-threshold circuits, timing-critical transistors are powered at high voltage, while most transistors are connected to the low voltage supply. Multiple voltages are also frequently used to provide standard voltage levels (e.g., 3.3 V) to input-output circuits, while powering internal logic at a reduced voltage to save power.

4.2. Synthesis for low-power technologies

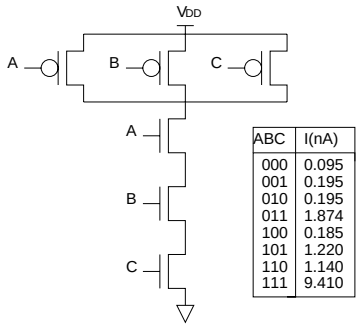
From the above overview, we can conclude that future low-power technologies will likely have multiple voltage supplies and device thresholds and they will be characterized by significant leakage. New logic synthesis algorithms and tools are needed for facing these challenges, and some recent research efforts have showed promising results.

When considering multiple-threshold circuits, only two (or, at best, three) different threshold voltages are likely to be available on a single chip, because of technology constraints. In this setting, the design problem is to select either low-threshold or high-threshold transistors for implementing logic gates in a complex network. Hence, for each gate in a technology library, a *LowT* and a *HighT* version are available.

Several algorithms (based on back-tracing and iterative improvement) have been proposed to automate the choice of *LowT* and *HighT* gates [55, 56]. The rationale behind these techniques is to start from a slow, *all-HighT* implementation and to replace gates on critical paths with fast but leaky *LowT* gates. Alternatively, it is possible to start from a fast *all-LowT* implementation and to replace non-critical *LowT* gates with their *HighT* counterparts. If only a small fraction of *HighT* gates is needed to meet performance constraints, leakage power remains small. More advanced approaches [57, 58] perform simultaneous threshold selection and gate sizing. Dual-threshold optimization techniques achieve promising results on industrial designs: Sirichotiyakul et.al. [57] report

that leakage power is reduced by a factor between 3 and 6 with speed penalty below 20%.

A problem related with dual threshold design is that of finding the input pattern minimizing leakage power in a combinational logic network. The sensitivity of leakage power to input values is caused by the body effect in MOS transistors, which raises the threshold when the transistor's source-bulk voltage is non-null. For logic gates with transistor stacks longer than one (in the pull-up or in the pull-down network), leakage is much reduced when two or more stacked transistors are off. In a multi-level network, the dependency of total leakage power from primary input values is not easily predictable, and several algorithms have been developed for finding the input pattern that minimizes it [24, 25, 26]. These algorithms have reported contrasting results for different circuits. On average, a careful choice of input pattern reduces leakage power by 15% to 20%.



ABC	I(nA)
000	0.095
001	0.195
010	0.195
011	1.874
100	0.185
101	1.220
110	1.140
111	9.410

Figure 1.10. 3-Input NAND gate and Leakage Current vs. Input Values

Example 13 *Figure 1.10 shows a 3-input NAND gate and a table with input values vs. leakage current, obtained through SPICE simulation [57]. The difference between minimum leakage and maximum leakage is more than three orders of magnitude. Observe also that leakage current decreases with increasing number of stacked transistors in the OFF state.*

Multiple supply-voltage circuit design poses several distinctive challenges [54]. In these circuits, power savings come from the replacement of non-critical gates powered by a high voltage supply with slower gates powered at low voltage. However, it is unwise to directly connect low-V_{DD} gates to the inputs of high-V_{DD} gates, because the supply mismatch

would prevent complete switch-off of the pull-up transistors in the high-V_{DD} gates. Hence, zero-static power level converters are required at the interface. Additionally, physical design is complicated by the necessity of distributing two supply voltages. For these reasons, it is generally not possible to arbitrarily assign low-V_{DD} gates one by one, but high-supply and low-supply gates should be clustered. Algorithms for automating the design of clustered dual-voltage circuits are presented in [54]. This technique, that has been used for designing large industrial prototypes [31], reduces power by as much as 50% with no performance penalty.

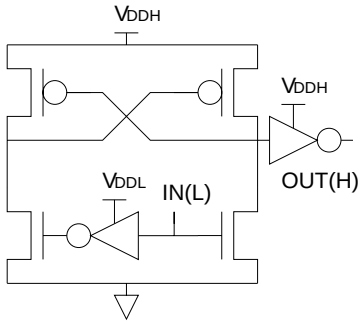


Figure 1.11. Low-to-high Level Shifter for Dual-Voltage Circuits

Example 14 *The level shifter of Figure 1.11 is proposed in [54] to interface low-V_{DD} gates with high-V_{DD}, with no static power consumption. Even though the circuit is relatively small, it does have a dynamic power and delay overhead, hence its usage should be minimized.*

4.3. Summary of recent approaches

New silicon technologies for low-power circuits support multiple supply and/or threshold voltages. Such voltages may be optimally chosen at design time, or varied during circuit operation. To leverage the advantages of these new degrees of freedom, a new set of algorithms, methodologies and cell libraries are being developed.

5. Conclusions

Even though it has been claimed that power issues are best addressed at high levels of abstraction, we believe logic synthesis for low-power dissipation is still crucial. In this chapter, we have reviewed the main accomplishments in logic synthesis for low power. First, we surveyed approaches targeting the reduction of switched capacitance in gate-level

netlists. We described technology independent optimization, library binding, and post-binding transformations. Many of these techniques have been implemented in commercial tools, and are routinely employed in the industrial practice. Second, we surveyed synthesis techniques targeting switching activity reduction, which automate the common design practice of gating the clock signal. Finally, we outlined the path of evolution for advanced low-power technologies and we examined a few novel challenges for future low-power synthesis tools. Low-power synthesis is still an important field for research and development. It will maintain its key role in current and future low-power design.

Acknowledgements

This work was supported in part by NSF, under grant CCR-9901190, and in part by the MARCO Gigascale Research Center.

References

[1] J. Rabaey, M. Pedram, *Low Power Design Methodologies*, Kluwer, 1996.

[2] J. Mermet, W. Nebel, *Low Power Design in Deep Submicron Electronics*, Kluwer, 1997.

[3] A. Chandrakasan, R. Brodersen, *Low-Power CMOS Design*, IEEE Press, 1998.

[4] L. Benini, G. De Micheli, *Dynamic Power Mangement: Design Techniques and CAD Tools*, Kluwer, 1998.

[5] M. Pedram, "Power Estimation and Optimization at the Logic Level," *Intl. Journal of High-Speed Electronics and Systems,* vol. 5, no. 2, pp. 179-202, 1994.

[6] D. Singh, J. Rabaey, M. Pedram, F. Catthoor, S. Rajgopal, N. Seghal, T. Mozdzen, "Power conscious CAD tools and methodologies: a perspective," *Proceedings of the IEEE*, vol. 83, no. 4, pp. 570-594, April 1995.

[7] M. Pedram, "Power minimization in IC design: principles and applications," *ACM Transactions on Design Automation of Electronic Systems*, vol. 1, no. 1, pp. 3-56, January 1995.

[8] E. Macii, M. Pedram, F. Somenzi, "High-Level Power Modeling, Estimation, and Optimization," *IEEE Transactions on Computer-Aided Design*, Vol. 17, No. 11, pp. 1061-1079, November 1998.

[9] L. Benini, G. De Micheli, "System-Level Power Optimization: Techniques and Tools," *ACM Transactions on Design Automation of Electronic Systems*, vol. 5, no. 2, pp. 115-192, April 2000.

[10] S. Iman, M. Pedram, *Logic Synthesis for Low Power VLSI Designs*, Kluwer 1998.

[11] W. Kunz, D. Stoffel, *Reasoning in Boolean Networks: Logic Synthesis and Verification using Testing Techniques*, Kluwer, 1997.

[12] G. De Micheli, *Synthesis and Optimization of Digital Circuits*, McGraw-Hill, 1994.

[13] T. Martin, D. Siewiorek, "The Impact of Battery Capacity and Memory Bandwidth on CPU Speed-Setting: A Case Study," *International Symposium on Low Power Electronics and Design*, pp. 200-205, 1999.

[14] R. Viswhanath, V. Wakharkar, A. Watwe, V. Lebonheur, "Thermal Performance Challenges from Silicon to Systems," *Intel Technology Journal*, Q3, 2000.

[15] T. Burd, R. Brodersen, "Processor Design for Portable Systems, *Journal of VLSI Signal Processing Systems*, vol. 13, no. 2-3, pp. 203–221, August 1996.

[16] A. P. Chandrakasan, S. Sheng, R. W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, Vol. 27, No. 4, pp. 473-484, April 1992.

[17] V. Tiwari, D. Singh, S. Rajgopal, G. Metha, R. Patel, F. Baez, "Reducing Power in High-Performance Microprocessors," *ACM/IEEE Design Automation Conference*, pp. 732–737, 1998.

[18] D. Brand, C. Visweswariah, "Inaccuracies in power estimation during logic synthesis," *ACM/IEEE International Conference on Computer-Aided Design*, pp. 388–394, 1996.

[19] F. Najm, "A Survey of Power Estimation Techniques in VLSI Circuits," *IEEE Transactions on VLSI Systems*, Vol. 2, No. 4, pp. 446-455, December 1994.

[20] S. Iman, M. Pedram, "POSE: power optimization and synthesis environment," *ACM/IEEE Design Automation Conference*, pp. 21–26, 1996.

[21] P. Landman, "High-Level Power Estimation," *ISLPED-96: ACM/IEEE International Symposium on Low Power Electronics and Design,* pp. 29-35, Monterey, CA, August 1996.

[22] Z. Chen, J. Shott, J. Plummer, "CMOS Technology Scaling for Low Voltage Low Power Applications," *International Symposium on Low Power Electronics*, pp. 56-57, Oct. 1994.

[23] K. Taki, "A survey for pass-transistor logic technologies," *Asia-Pacific Design Automation Conference*, pp. 223–225, Jan 1998.

[24] J. Halter, F. Najm, "A gate-level power reduction method for ultra-low-power CMOS circuits," *IEEE Custom Integrated Circuits Conference*, pp. 475–478, 1997.

[25] Y. Ye, S. Borkar, V. De, "A New Technique for Standby Leakage Reduction in High-Performance Circuits," *IEEE Symposium on VLSI Circuits*, pp. 40–41, 1998.

[26] M. Johnson, D. Somasekar, K. Roy, "Models and algorithms for bounds on leakage in CMOS circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 6, June 1999.

[27] L. Benini, G. De Micheli, "Automatic synthesis of low-power gated-clock Finite-State Machines," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 6, pp. 630–643, June 1996.

[28] L. Benini, M. Favalli, G. De Micheli, "Design for testability of gated-clock FSMs," *IEEE European Design and Test Conference*, pp. 589–596, March 1996.

[29] L. Benini, G. De Micheli, E. Macii, M. Poncino, R. Scarsi, "Symbolic Synthesis of Clock-Gating Logic for Power Optimization of Synchronous Controllers", *ACM Transactions on Design Automation of Electronic Systems*, Vol. 4, No. 4, pp. 351-375, October 1999.

[30] V. Lee et al., "A 1-V Programmable DSP for Wireless Communications," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 11, pp. 1766–1776, Nov. 1997.

[31] M. Takahashi et al., "A 60-mW MPEG4 Video Coded Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 11, pp. 1772–1780, Nov. 1998.

[32] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 2, No. 4, pp. 426-436, December 1994.

[33] J. Monteiro, S. Devadas, A. Ghosh, "Sequential Logic Optimization for Low Power Using Input-Disabling Precomputation Architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 17, No. 3, pp. 279-284, March 1998.

[34] L. Benini, G. De Micheli, A. Lioy, E. Macii, G. Odasso, M. Poncino, "Computational Kernels and their Application to Sequential Power Optimization," *ACM/IEEE 1998 Design Automation Conference*, pp. 764-769, San Francisco, California, June 1998.

[35] L. Benini, G. De Micheli, E. Macii, G. Odasso, M. Poncino, "Kernel-Based Power Optimization of RTL Components: Exact and Approximate Extraction Algorithms," *ACM/IEEE Design Automation Conference*, pp. 247-252, New Orleans, Louisiana, June 1999.

[36] M. Munch, B. Wurth, R. Mehra, J. Sproch, N. Wehn, "Automatic RT-level operand isolation to minimize power consumption on datapaths," *IEEE Design Automation and Test in Europe*, pp. 624–631, 2000.

[37] S. Borkar, "Design Challenges of Technology Scaling," *IEEE Micro*, vol. 19, no. 4, pp. 23–29, July-Aug. 1999.

[38] S. Thompson, P. Packan, M. Bohr, "MOS Scaling: Transistor Challenges for the 21st Century," *Intel Technology Journal*, Q3, 1998.

[39] B. Chen, I. Nedelchev, "Power Compiler: A Gate Level Power Optimization and Synthesis System," *IEEE International Conference on Computer Design*, pp. 74–79, 1997.

[40] O. Coudert. R. Haddad, "Integrated resynthesis for low power," *IEEE International Symposium on Low Power Electronics and Design*, pp. 169–174, 1996.

[41] E. Lehman, Y. Watanabe, J. Grodstein, H. Harkness, "Logic decomposition during technology mapping," *IEEE International Conference on Computer-Aided Design*, pp. 264-271, 1995.

[42] O. Coudert, J. Cong, S. Malik, M. Sarrafzadeh, "Incremental CAD," *IEEE/ACM International Conference on Computer-Aided Design*, pp. 236-242, 2000.

[43] B. Rohfleisch, A. Kolbl, B. Wurth, "Reducing power dissipation after technology mapping by structural transformations," *IEEE/ACM Design Automation Conference*, pp. 789–794, 1996.

[44] I. Bahar, D. Lampe, E. Macii, "Power Optimization of Technology Dependent Circuits Based in Symbolic Computation of Logic Implications," *ACM Transactions on Design Automation of Electronic Systems* vol. 5, no. 3, pp. 267–293, July 2000.

[45] L. Benini, P. Vuillod, G. De Micheli, "Iterative re-mapping for logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. vol. 17, no. 10, pp. 948–964, Oct. 1998.

[46] R. Bahar, H. Cho, G. Hachtel, G. Hachtel, E. Macii, F. Somenzi, "Symbolic timing analysis and resynthesis for low power of combinational circuits containing false paths," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 16, no. 10, pp. 1105–1115, Oct. 1997.

[47] O. Coudert, "Gate sizing for constrained Delay/Power/Area optimization," *IEEE Transactions on VLSI Systems*, vol. 5, no. 4, pp. 465–472, Dec. 1997.

[48] H. Kapadia, L. Benini and G. De Micheli, "Reducing Switching Activity on Datapath Buses with Control-Signal Gating," *IEEE Journal of Solid State Circuits*, vol. 34, no. 3, pp. 405-414, March 1999.

[49] G. Lakshminarayana, A. Raghunathan, K. S. Khouri, N. K. Jha, S. Dey, "Common-Case Computation: A High-Level Tecnique for Power and Performance Optimization," *ACM/IEEE Design Automation Conference*, pp. 56-61, New Orleans, LA, June 1999.

[50] H. R. Lim, T. T. Hwang, "On determining sensitization criterion in an iterative gate sizing process," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 2, pp. 231–238, Feb. 1999/

[51] J. Montanaro et al., "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1703–1714, Nov. 1996.

[52] C. Piguet, "Design of low-power libraries," *IEEE International Conference on Electronics, Circuits and Systems*, pp. 175–180, 1998.

[53] L. Wei, K. Roy, V. De, "Low Voltage Low Power CMOS Design Techniques for Deep Submicron ICs," *IEEE International Conference on VLSI*, pp. 101–107, 2000.

[54] K. Usami et al., "Automated low power technique exploiting multiple supply voltages applied to a media processor," *IEEE Journal of Solid-State Circuits*, vol. 33, no. 3, pp. 463–472, March 1998.

[55] L. Wei, Z. Chen, K. Roy, M. Johnson, Y. Ye, V. De, "Design and Optimization of Dual-Threshold Circuits for Low-Voltage Low-Power Applications," *IEEE Transactions on VLSI Systems*, vol. 7, no. 1, pp. 16-24, March 1999.

[56] V. Sundararajan, K. Parhi, "Low power synthesis of dual threshold voltage CMOS VLSI circuits," *IEEE International Symposium on Low-Power Electronics and Design*, pp. 139–144, 1999.

[57] S. Sirichotiyakul et al., "Stand-by power minimization through simultaneous threshold voltage selection and circuit sizing," *IEEE/ACM Design Automation Conference*, pp. 436–441, 1999.

[58] L. Wei, K. Roy, C. Koh, "Power minimization by simultaneous dual-V_{th} assignment and gate-sizing," *IEEE Custom Integrated Circuits Conference*, pp. 413–416, 2000.