Minimizing power consumption in digital circuits and systems: an overview*

N. Wehn and M. Münch

University of Kaiserslautern, Germany

Camera-ready Copy for

Kleinheubacher Berichte

Manuscript-No.

Offset requests to:
N. Wehn
Universität Kaiserslautern
Fachbereich Elektrotechnik und Informationstechnik
AG Mikroelektronische Systeme
Erwin-Schrödinger-Straße
D-67663 Kaiserslautern
Germany
`wehn@e-technik.uni-kl.de`

*Part of this work has been supported by the *Deutsche Forschungsgesellschaft (DFG)* under grant We 2442/1-1 within the *Schwerpunktprogramm „Grundlagen und Verfahren verlustarmer Informationsverarbeitung (VIVA)“.*

Minimizing power consumption in digital circuits and systems: an overview*

N. Wehn and M. Münch
University of Kaiserslautern, Germany

*Part of this work has been supported by the *Deutsche Forschungsgesellschaft (DFG)* under grant We 2442/1-1 within the *Schwerpunktprogramm „Grundlagen und Verfahren verlustarmer Informationsverarbeitung (VIVA)“*.

Abstract. In designing digital circuits and systems, minimizing power consumption has gained considerably in significance compared to the traditional cost metrics of silicon area, performance and testability. The increasing importance of low power consumption is due to the ever decreasing feature sizes of microelectronic circuits, higher clock frequencies and larger die sizes, as well as the growing number of mobile, battery-operated systems. To contain power optimization, modern design methodologies therefore allow optimizing power on all levels during the design flow, from system level down to technology level. These optimizations are either applied manually by the design engineer or automatically using CAD tools.

This paper gives an overview of the state-of-the-art in minimizing power consumption on all levels of the design flow of digital circuits and systems.

Zusammenfassung. Die Minimierung des Leistungsverbrauchs digitaler Schaltungen und Systeme hat in den vergangenen Jahren neben den Metriken Fläche, Durchsatz und Testbarkeit eine immer größere Bedeutung gewonnen. Diese Entwicklung ist wesentlich auf die stetig sinkenden Strukturgrößen mikroelektronischer Schaltungen bei gleichzeitig steigenden Taktfrequenzen und größerer Chipfläche sowie die wachsende Anzahl mobiler, batteriebetriebener Systeme zurückzuführen. Zur Beherrschung des Leistungsverbrauchs werden daher in modernen Entwurfsmethodiken Optimierungen auf allen Ebenen des Entwurfsablaufs durchgeführt, angefangen auf der Systemebene bis hin zur Technologieebene. Diese werden entweder manuell durch den Entwurfsingenieur oder automatisiert mit Hilfe von CAD–Werkzeugen durchgeführt.

Der vorliegende Artikel bietet einen Überblick über den Stand der Technik bei der Minimierung des Leistungsverbrauchs im gesamten Ablauf des Entwurfs digitaler Schaltungen und Systeme.

1 Introduction

During the past three decades, the semiconductor industry has sustained a remarkably short innovation cycle. As early as 1995, Gordon Moore predicted that the complexity of microprocessors would double every twelve months (*Moore's Law*) (Moore, 1965). Although this time-line had to be adjusted to 24 months, the data published in the 1997 Edition of the *National Technology Roadmap for Semiconductors* (SIA) predicts that this trend will continue far into the next century. For instance, current microprocessors have feature sizes (measured in gate lengths) of 180nm—this number is expected to be reduced to 25nm by the year 2014.

However, smaller feature sizes not only allow integration of a larger number of components on a chip, but also reduce the signal propagation delays which in turn permit higher clock frequencies. Together with the larger number of devices on a die, this results in an overall increase in power dissipation. According to the Roadmap, high-performance microprocessors will therefore consume up to 160 Watts on a single die by the year 2005, posing special challenges on heat dissipation, i.e. cooling and packaging technology. Table 1 lists the power consumption for some selected high-performance and embedded processors. It has in fact been noted (Chandrakasan and Broderson, 1998) that for microprocessors there is the following analytical relationship between power consumption, area and clock frequency which potentially limits integration density:

$$P_{\mu\mathrm{P}} = 0.063 \cdot \frac{\mathrm{W}}{\mathrm{cm}^2 \cdot \mathrm{MHz}} \cdot A \cdot f_{\mathrm{clk}},$$

where $P_{\mu \mathrm{P}}$ is the processor's power consumption, A its die size area in cm^2 and f_{clk} its clock frequency in MHz.

Processor	Process [nm]	Die Size [mm^2]	Transistor Count	Clock Frequency [Mhz]	Supply Voltage [V]	Power Consumption [W]
21064	750	233.52	1.68M	200	3.3	30
21164	500	298.65	9.3M	300	3.3	50
21264	350	313.96	15.2M	600	2.2	72
Pentium II	280	n/a	7.5M	300	3.3	43
UltraSPARC II	250	n/a	5.4M	400	3.3	21
PPC 4xx	500	36.60	600,000	25	3.3	0.2
StrongArm 110	350	49.92	2.5M	215	2	< 0.5

Table 1. Power consumption of selected high-performance microprocessors and embedded microprocessors (Gronowski et al., 1998; Gwennap, 1997; Sun Microsystems; Correale, 1995; Dobberpuhl, 1996).

The advent of portable, wireless computation and communication devices imposes very tight power budgets on the design to maximize battery life, minimize battery weight and allow usage of cost-efficient packaging technology. At the same time, real-time processing of audio and video data demand high computational power to meet the throughput requirements of sophisticated digital signal processing algorithms. Such algorithms typically imply a high power consumption due to the number and complexity of basic operations executed.

From the above, it should become clear that there is a strong necessity for minimizing power consumption when designing complex microelectronic digital circuits and systems. Modern design methodologies offer designers tool support when taking a system specification down to a mask layout suitable for fabrication through a stepwise refinement process, across which the design is being described on different levels of abstraction. On each level through the refinement process, the designer can manually or automatically apply optimizing transformations to minimize the power consumption of the design. In the following, we will investigate power optimizing transformations on each level of abstraction and provide references to relevant literature. Due to their dominating position in microelectronic system implementations, we will restrict our discussion to digital CMOS circuits. It is important to note that not all of the techniques presented herein will be equally applicable to all types of designs. As Figure 1 shows, the components contributing to the overall power consumption vary considerably across different designs, making power optimization an inherently application-specific problem. When optimizing a design for low power consumption, it is therefore imperative to carefully analyze what the major contributors to the system's power consumption are and apply optimizations accordingly.

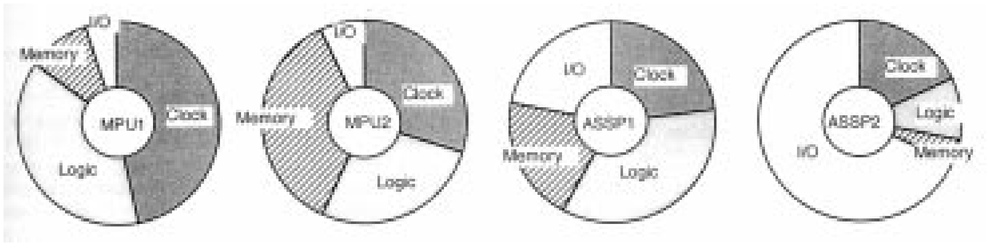

Figure 1. Power distribution in various heterogeneous designs (MPU1: low-end microprocessor for embedded use; MPU2: high-end CPU with large amount of cache; ASSP1: MPEG2 decoder; ASSP2: ATM switch).

The remainder of the paper is structured as follows: In Section 2 we will briefly discuss the sources of power consumption and the resulting objectives for designing low-power systems. In Section 3 we will review the different levels of abstraction on which digital circuits and systems can be described and the potential for power optimization given on each level. Section 4 will describe some of the most efficient techniques used to minimize power during the design process. Section 5 concludes the paper.

2 Sources of power consumption in digital CMOS circuits

There are four components which contribute to the average power consumption P_{avg} of a digital CMOS circuit:

$$P_{\mathrm{avg}} = P_{\mathrm{leakage}} + P_{\mathrm{sc}} + P_{\mathrm{switching}} + P_{\mathrm{static}},$$

where P_{leakage} is the leakage power, P_{sc} is the short-circuit power, $P_{\text{switching}}$ the switching component of power and P_{static} the static power consumption.

Leakage power is caused by substrate injection at p/n junctions and subthreshold effects which are primarily determined by fabrication technology. It contributes less than 1% to the average power consumption and can therefore be ignored for our purposes. It is, however, important in the context of *ultra low-power* systems with a very small V_{DD}, as used in microelectronic systems for medical applications.

Short-circuit power is caused by currents which temporarily occur when both the n- and p-parts of a gate are open while the gate is switching. Suppose the input x to the CMOS inverter in Figure 2 is making a transition from logic '0' to logic '1'. While the p-device is being turned off and the n-device is being turned on, there is a short period of time during which there is a direct current path from V_{DD} to ground. This current is called the *short-circuit current* I_{sc}. It typically accounts for 10% to 20% of the overall power consumption.

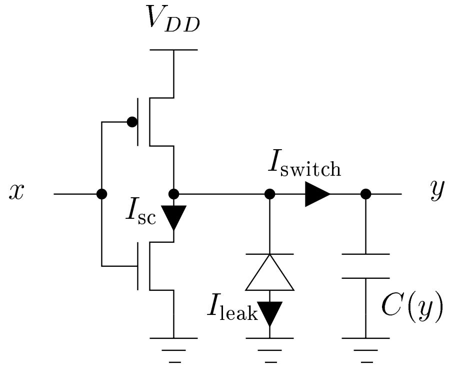

Figure 2. Capacitive power consumption in a CMOS inverter.

The *switching power*, also called *dynamic* or *capacitive* power, is by far the most significant component and accounts for approximately 80% of the overall power consumption. Switching power is dissipated when the capacitive load $C(y)$ of a CMOS gate is being charged by the current I_{switch} through the p-device to make a transition from ground to V_{DD}. The energy required for this transition is $C(y)V_{DD}^2$. Power-consuming transitions occur at a frequency $\frac{1}{2}\alpha(y)f_{\text{clk}}$ proportional to the clock frequency f_{clk}, where $\alpha(y)$ is the probability of signal y to make a '0' → '1' or '1' → '0' transition. The total switching power of a circuit is therefore

$$P_{\text{switching}} = \frac{1}{2} f_{\text{clk}} V_{DD}^2 \sum_{\text{signal } y} \alpha(y)C(y). \tag{1}$$

Eq. (1) shows that the switching power increases linearly with the clock frequency. To reduce the switching power even for increasing clock frequency, we therefore have the following opportunities:

– Since the supply voltage V_{DD} influences the $P_{\text{switching}}$ quadratically, it is most effective to reduce V_{DD}. However, a lower supply voltage also results in slower switching speed and therefore degrades performance of the circuit accordingly. The relationship between delay, i.e. clock frequency, can be expressed as follows (Taur, 1999):

$$f_{\text{clk}} \propto 0.7 - \frac{V_t}{V_{DD}},$$

where V_t is the threshold voltage of the given technology. In Section 4 we will see how to maintain the performance of a circuit when using slower—i.e. lower voltage—technologies through architectural transformations.

– To reduce the *switched capacitance* $\sum_{\text{signal } y} \alpha(y)C(y)$ of the circuit, we can decrease the *switching activity* of a signal y by appropriately restructuring and/or modifying the fanin logic feeding signal y.

– Alternatively, we can reduce the actual *physical capacitance* $C(y)$ to be switched by a signal y. $C(y)$ depends on the output capacitance of the corresponding gate, the wiring capacitance and the gate capacitances of the fanout gates. All of these are in turn dependent on the technology used. However, it is important to note that wiring length does not scale proportionally to the feature sizes of the underlying technology, but that its scaling behavior depends on the wire locality (Rabaey, 1996, Chapter 8). For global wires, capacitance typically *grows* with technology scaling. Therefore, global wiring capacitance is becoming a dominating parameter in $C(y)$ for deep-submicron technologies. To reduce power, we therefore have to pay special attention to global wiring capacitances.

Section 4 will show that on each level of abstraction, there are a number of circuit transformation techniques to reduce the switching activities $\alpha(y)$ of internal signals y as well as their physical capacitances $C(y)$.

For a more comprehensive treatment of the different aspects related to the power consumption of a digital circuit, the reader is referred to (Liu and Svensson, 1994; Chandrakasan and Brodersen, 1995; Rabaey and Pedram, 1996; Rabaey, 1996; Chandrakasan and Broderson, 1998).

3 Optimization potential at various levels of abstraction

As mentioned above, current design methodologies rely on a stepwise refinement flow to map a circuit specification onto a mask layout for fabrication. Within this flow, the circuit description is transformed and manipulated on different levels of abstraction. We generally distinguish between the following levels of abstraction (Gajski and Kuhn, 1983):

1. On *system level* the system is described in terms of a set of hardware, software and memory components and interacting algorithms they perform to provide a certain functionality.
2. On *behavioral* or *architectural level* the individual components (ASICs, datapaths, software processes) are described in terms of their algorithmic behavior, typically in a hardware description language, such as behavioral VHDL, or a high-level programming language such as C. Typically, there is no timing, but only causal information available on this level and the interface protocols controlling the communication between the interacting processes has already been fixed.
3. On *register-transfer level* hardware is described in terms of arithmetic modules, registers and multiplexors and interconnect to steer data flow. RT level representations describe the basic building blocks of a system and their interconnect on clock-cycle level; it specifies the "micro-architecture" of the system under consideration.
4. On *gate-level* a circuit is described in terms of a netlist of gates or a set of Boolean equations reflecting its functionality.
5. On *transistor level* a circuit is described in terms of its transistor network structure.
6. Finally, on *physical level* a circuit is described in terms of its mask layout as it will be used for fabrication. The basic building blocks on this level are polygons reflecting the different materials used for fabrication, such as metal, polysilicon, oxide *etc.*

There is a general consensus that design decisions on higher levels of abstraction have the most significant impact on the overall structure and quality of the resulting design. For instance, choosing different partitionings of functionality on design components on the system level can result in vastly different requirements on the characteristics of the individual components in terms of area, performance or power. Accordingly, optimizing transformations on higher levels of abstraction have the largest impact on the power consumption of the design as a whole. It is therefore desirable to take power into account as a cost function *as early as possible* in the design flow, i.e. on system and behavioral level, in the design flow, since in the later stages optimizations will typically only be of a very local scope, and hence of limited effectiveness. Figure 3 (Sproch, 1997) reflects the estimated potential power savings that can be obtained by optimization on the respective level of abstraction.

4 Optimization techniques

4.1 System level

The first decision to be made on system level is that of *algorithm selection*, i.e. selecting, from a set of given algorithms, one that provides the required functionality of the system under design while best meeting design

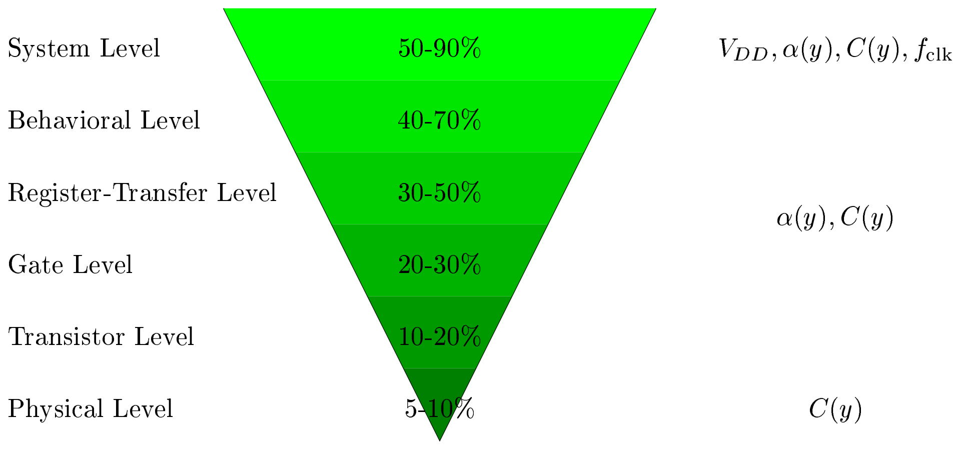

Figure 3. Potential power savings at different levels of abstraction (Sproch, 1997).

constraints. The power consumption induced by an algorithm depends on its characteristics in terms of overall complexity, complexity of the basic operations, communication requirements *etc.* Although the selected algorithm itself can later be optimized, subsequent optimizations are typically of a local scope and will not be able to compensate the differences in power consumption across different algorithms, which can be up to several orders of magnitude.

One of the most important factors to be taken into consideration during algorithm selection is that of memory access and management. Relative to a 16-bit multiplication, for instance, the energy required for a 16-bit memory transfer is one order of magnitude higher in both hardware and software (Catthoor et al., 1998; Wuytack, 1998). Figure 4 compares the relative energy required for typical computational and storage operations (Wuytack, 1998). It is therefore desirable to choose an algorithm which minimizes the number of memory transfers or, alternatively, an algorithm which is susceptible to further optimizations to reduce the number of memory transfers. As Catthoor et al. (1998) have shown, such optimizations can often be performed systematically on a given algorithm.

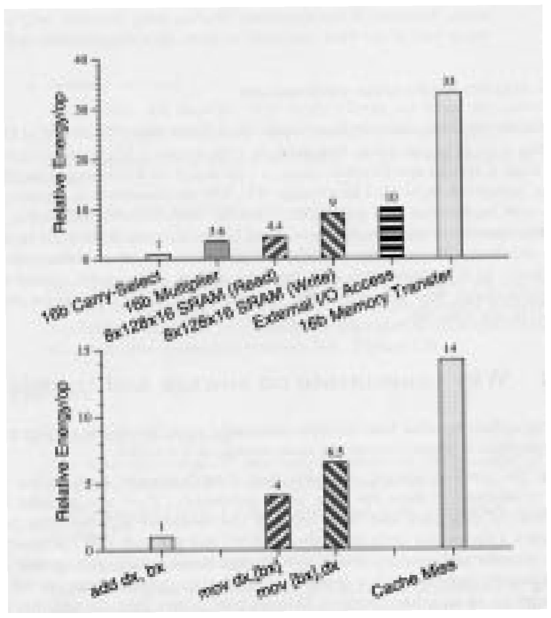

Figure 4. Comparing the relative energy required for computational and storage operations (Wuytack, 1998).

Upon algorithm selection, the different components of the algorithm—which typically consists of a set of communicating processes—will be mapped onto hardware and software components to execute the corresponding processes. This phase is called *hardware/software partitioning.* The decision of whether to execute a process on hardware or on software involves trading off flexibility vs. timing/performance constraints and power consumption. In general, a custom-specific hardware implementation is the most power-efficient, since it permits implementing an algorithm with the least redundancy. However, it has been shown that even the power consumption of software can be optimized within limits (Tiwari et al., 1994, 1996).

Hardware/software partitioning affects power consumption mainly through the communication required between different hardware components and processors. Partitioning should be performed to minimize the number of off-chip access operations, since these significantly contribute to the overall power consumption due to the large capacitances to be switched. Processes which communicate heavily should therefore be implemented on a single die—note that this does not preclude implementing the processes distinctly in hardware and software, since many processors used in system-on-chip designs are nowadays available as synthesizable cores.

When partitioning, one should also bear in mind what components of the system are executed under mutual exclusion. For instance, when two mutually exclusive processes are implemented in separate synchronous hardware components, we can turn off the clock of the currently inactive component, thereby saving power by preventing any switching activity to occur inside the inactive component. This technique is called *quiescent unit shutdown* and can in fact be applied on various levels of abstraction.

As we have seen in Eq. (1), power consumption depends on the square of the supply voltage. *Voltage scaling* is therefore an efficient means to achieve lower power consumption. To compensate for the performance degradation associated with a reduction in V_{DD}, we can apply architectural transformations which allow us to maintain the throughput of a circuit at the expense of an increase in area due to the higher degree of parallelism introduced. In a *parallel implementation*, a datapath is duplicated such that both instances of the datapath are clocked at half the speed of the original datapath. The input data is then fed to the datapaths in an alternating fashion; a multiplexor at the output selects the result from the corresponding datapath at the required throughput rate. In a *pipelined implementation*, pipeline registers are introduced into a datapath, thereby reducing the delay between sequential elements to allow for lower switching speeds. These techniques have been applied to various signal processing circuits; in (Chandrakasan and Brodersen, 1995, Chapter 4) case studies for a simple add/compare datapath are presented. With a parallel implementation, the power consumption could be reduced by 64% at a 3.4x increase in area; a pipelined implementation reduced power by 61% at a 1.3x increase in area. For a more detailed discussion of voltage scaling techniques, the reader is referred to (Chandrakasan and Brodersen, 1995, Chapter 4).

Voltage scaling can also be applied dynamically to sub-systems with variable throughput requirements. This technique is referred to as *voltage scheduling.* The idea is to dynamically adapt the supply voltage of a block such that the current throughput requirement as defined by the environment is met. Whenever the throughput is to be raised during operation, the supply voltage is increased accordingly. A lower throughput allows for a lower V_{DD}, respectively.

4.2 Behavioral level

Behavioral level power optimization is performed during *behavioral synthesis*, i.e. when mapping an algorithm onto a hardware register-transfer level description. We can distinguish between optimizations which operate on the behavioral description alone and those which optimize the mapping of operations onto hardware units.

Algorithmic transformations fall into the former class. Similar to the architectural transformations discussed in Section 4.1 in the context of voltage scaling, we can transform the behavioral description to allow voltage scaling while maintaining throughput. The objective of such transformations is to either replace slow operations by faster ones (e.g., replacing a multiplication with constants by a sequence of shift/add operations) or to prepare the behavioral description so as to later allow efficient power-optimizing transformations such as retiming or pipelining to offset the speed loss through voltage scaling. Common transformations employed are algebraic transformations (associativity, distributivity), loop transformations (unrolling, folding), constant propagation, *etc.*

During behavioral synthesis, a mapping of operations to clock cycles in which these operations will be executed, has to be performed. This process is called *scheduling.* A number of power-conscious scheduling algorithms have been proposed in literature, such as (Mehra and Rabaey, 1996; Martin and Knight, 1996; Monteiro et al., 1996; Lin et al., 1997). They typically have "proactive" character in the sense that they try to find a mapping of operations onto clock cycles which *allows* a power-efficient implementation of the specified behavior. Since scheduling determines the maximum delay within a clock cycle, for instance, it also impacts the applicability of voltage scaling techniques. It is therefore desirable to minimize the maximum delay within a clock cycle, to allow operation at the lower possible supply voltage. Alternatively, it is also possible to use different, but fixed supply voltages in a design. The *multiple supply-voltage scheduling problem* differs from a classical scheduling problem in that the delay of an operation varies depending on what voltage is used to power the module on which the operation is executed. The problem is therefore to optimally assign supply voltages to the operations in the behavioral description, such that timing constraints are satisfied at a minimum power consumption.

Moreover, a scheduling algorithm can support dynamic power management by evaluating conditionals in the behavioral descriptions as soon as possible, such that modules which depend on the outcome of the conditional can be disabled in subsequent clock cycles, thereby saving power. Other scheduling algorithms try to actively minimize power by taking into consideration spatial and temporal correlations within and across successive data words to be applied to functional modules. Operations whose input data are highly correlated in a positive fashion exhibit lower switching activity in the input stream and should therefore be implemented on (*bound to*) the same module to minimize the power consumption of the module. However, this is only possible when the corresponding operations are scheduled in different clock cycles.

The degree of *resource sharing*, i.e. the execution of different operations on the same module in different clock cycles, is determined during *binding*, when operations and values in the behavioral specification are assigned to modules and registers in the RT structure, respectively, and its interconnect structure is being derived. Since resource sharing determines the statistics of the input data to modules, it has a significant impact on their power consumption. Binding hence has to consider the correlations of the input data streams between different operations to minimize switching activity at module inputs, as mentioned in the preceding paragraph. The same reasoning obviously applies to the assignment of values to registers. Several approaches have been presented in literature to incorporate switching activity information during binding (Raghunathan and Jha, 1994; Chang and Pedram, 1995).

Behavioral synthesis also impacts power consumption by the choice of functional units on which operations are implemented (*resource selection*). On behavioral level, the power consumption of different implementation alternatives can be estimated as a function of the input switching activity either using a parametric analytical model or a look-up table-based model (Landman and Rabaey, 1993, 1994; Landman, 1994). Such models can be used to guide resource selection in choosing modules that optimally trade off area vs. performance vs. power to implement a given operation.

Other efficient techniques for behavioral power optimization include the choice of *number representation* and *bus encoding*. These determine the transition probability in different regions of data words and therefore the switching activity at the input to functional units. In many digital signal processing applications, for instance, data words are typically temporally very correlated. This implies a low transition probability in the high-order (sign extension) region of a data word. In such a case, a sign-magnitude number representation is the preferred choice over the classical two's complement representation, since it yields extremely low switching activities in the high-order bit region.

Likewise, switching activity on busses can be reduced by an appropriate encoding depending on the characteristics of the data transmitted. For long sequences of successive numbers, for instance on address busses, a *Gray code*—in which two neighboring code words have a Hamming distance of one—reduces bus switching to a minimum. In a *bus-invert code*, data words are either transmitted as is or inverted, depending on what coding yields the lower Hamming distance to the previous transmitted data word. An extra bus line tells the data sink whether or not to invert the data to obtain the actually transmitted data.

The techniques discussed so far all address the data path portion of a system. A number of techniques exist to also reduce power consumption in the control part. For example, some of the switching power in the control part is caused by the application of state codes to the next state logic network and the state registers. By minimizing the Hamming distance between two successive state codes, we can therefore reduce the switching in the control logic network and state registers. The objective during state machine coding is therefore to assign minimum-distance code words to neighboring states in the state machine—typical choices are *one-hot* or *Gray code* encodings.

The idea of shutting down inactive units on system level can similarly be applied to state machines. A complex state machine can be decomposed into a set of smaller state machines, all of which operate under pairwise mutual exclusion. This permits shutting down all but the currently active state machine during circuit operation, thereby considerably reducing switching activity.

4.3 Register-transfer level

Optimizations on register-transfer level operate by structurally transforming RT level netlists. Automatic CAD tool support is available on RT level downward, unlike the techniques discussed in the previous sections, which either have to be applied manually or for which only prototype academic tools are available.

One of the most efficient techniques known on RT level is *clock gating*. The idea of clock gating is to turn off (*"gate"*) the clock to a register whenever the input to the register need not be stored, i.e. its contents can be retained. Consider, for instance, the scenario in Figure 5. In the original RT structure on the left hand side, the value of the output of the multiplier is stored in the register whenever the enable output `en` of the FSM is logic '1'. When `en` is logic '0' the current value in the register is fed back to its input and therefore retained for the following clock cycle. In the optimized circuit on the right hand side, the multiplexor is replaced by a clock gating cell which effectively ANDs the register clock with the enable signal from the state machine. Whenever `en` is logic '0' the clock signal is shut off from the register, therefore preventing any new value to be loaded into the register. The effect is two-fold: On the one hand, clock gating reduces the switching in the fanin of the register's data input, on the other hand it considerably reduces the capacitive load on the clock line, which can account for more than 50% of the overall power consumption. Clock gating has in fact obtained in practice power reductions of up to 60% for highly register-dominated designs. In the commercial domain, the Synopsys Power Compiler™ is a tool that permits automatic application of clock gating to RT level structures.

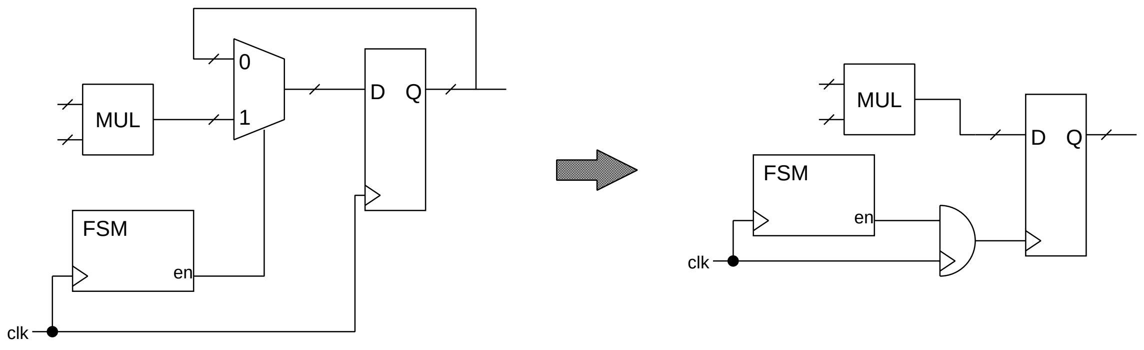

Figure 5. Gating register clocks.

The idea of *operand isolation* is similar to clock gating in the sense that it prevents switching at the input of combinational blocks whenever their result is not evaluated in the downstream circuit. Consider again the clock gating example discussed above in Figure 6. As we have seen, the enable signal `en` controls whether or not a new value is loaded into the register—and therefore whether the multiplier performs a *redundant* operation. Regardless of the value of `en`, however, the multiplier performs a multiplication whenever a data input signal toggles. Hence, it consumes power even when its output is *not* loaded into the register. With operand isolation, we can use the value of `en` to gate latches at the input to the multiplier, such that whenever `en` is logic '0' the current data is latched at the multiplier input, preventing the propagation switching activity into the multiplier. Accordingly, power is not consumed in the multiplier even when the inputs to the latch toggle.

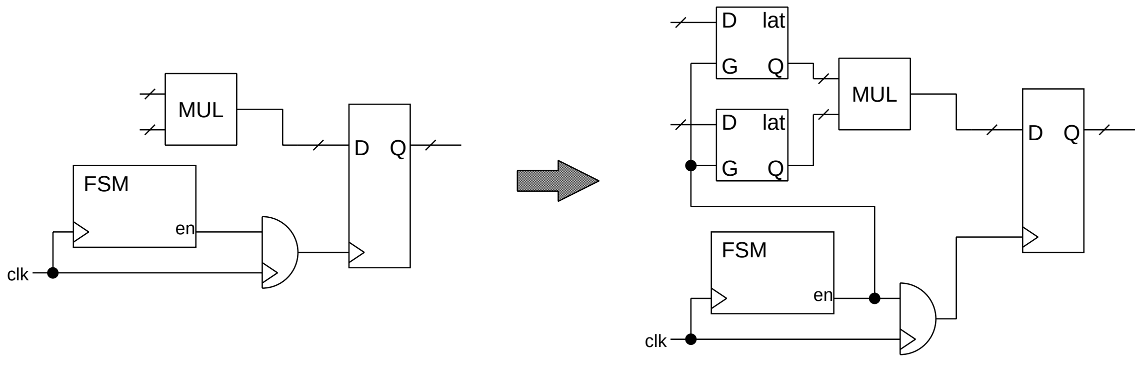

Figure 6. Preventing the propagation of switching activity by operand isolation.

Operand isolation was first presented in (Correale, 1995), where it was described as one of the techniques to reduce power consumption in the PowerPC 4xx family of embedded controllers. A similar technique called *guarded evaluation* (Tiwari et al., 1998) has been presented for optimizations on logic level. A strategy to automate the application of operand isolation on RT level has been presented in (Münch, 1999).

Pre-computation is used to reduce the power consumption in individual stages of pipelined designs. The techniques reduces the power consumption in a pipeline stage B whenever their result can be predicted from the results of a previous pipeline stage A. In such a case, a signal is generated by combinational prediction logic in stage A to gate the clock signal to the register banks feeding pipeline stage B. As a result, the input to B does not change, therefore preventing further switching activity in B until the gating signal becomes inactive again. Simultaneously, registers are added to hold the precomputed result of stage B at the output of stage A—the outputs of these registers are then used to feed the output of stage B in the subsequent clock cycle.

Figure 7 shows an example pre-computation architecture. In the example, the logic in stage B compares two numbers computed by combinational logic in stage A. For a two's complement representation, the most significant bit (MSB) indicates the sign such that whenever the two MSBs differ, no further comparison need be made. This fact is exploited as follows: Whenever the two MSBs are different (indicated by the exclusive OR evaluating '1'), the numbers are not stored in the output registers of A. Instead, one of the MSBs is stored to indicate the result of the comparison in the following pipeline stage—the result to be stored at the output of stage B is selected by the result of the exclusive OR operation.

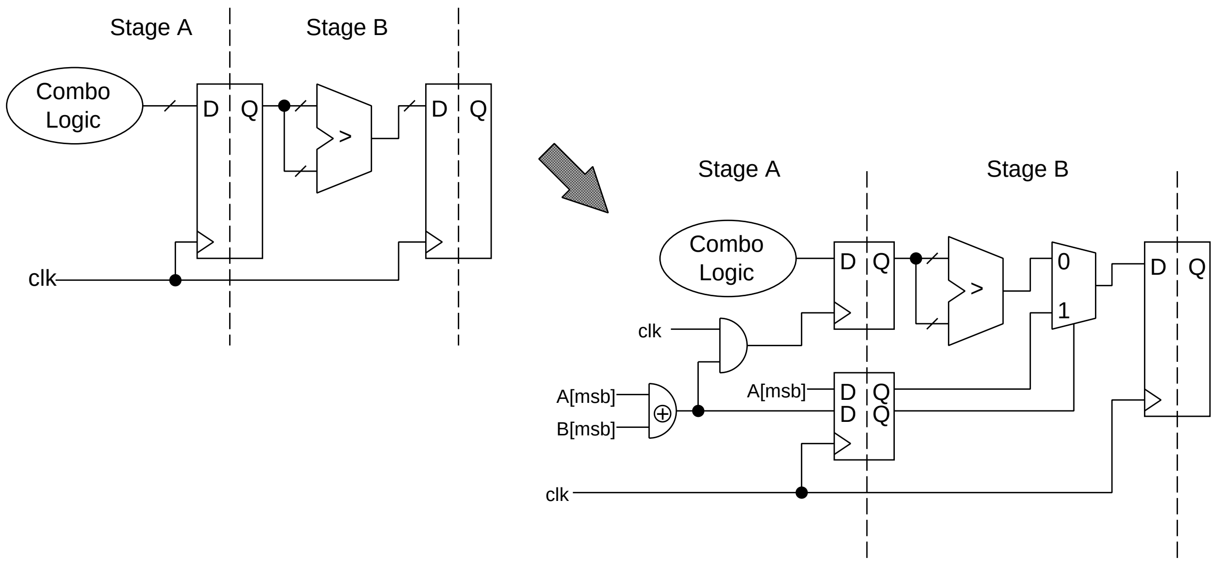

Figure 7. An architecture to pre-compute the result of a comparison of two's complement numbers.

To conclude, it should be noted that neither operand isolation nor pre-computation is yet available in commercial CAD tools.

4.4 Gate level

It is on gate level where most commercially available tools offer support for power optimization while mapping and optimizing an RT level description onto a cell library. This process typically proceeds in two steps: *Technology decomposition* followed by *technology mapping*, possibly along with other optimizations.

Technology decomposition is the problem of mapping a logic network onto a functionally identical logic network consisting of gates only from a set of generic basic gates, such as two-input NAND gates and inverters. For low power consumption, is it advisable to structure the decomposed network such that the sum of the switching activities at its internal nodes is minimized. Figure 8 shows two different alternatives in decomposing a four-input AND gate into a network of two-input AND gates. In this case, the balanced tree realization yields the lowest sum of switching activities of the signals connecting the gates. In general, *path balancing* is often a preferred implementation because it equalizes the delay on the different paths in a network, therefore reducing the power consumption due to glitching activity.

The process of implementing a generic logic network obtained through technology decomposition with one or more cells from the actual cell library is termed *technology mapping*. Power consumption can be reduced during technology mapping by hiding signals with high switching activities inside gates, where they drive a lower

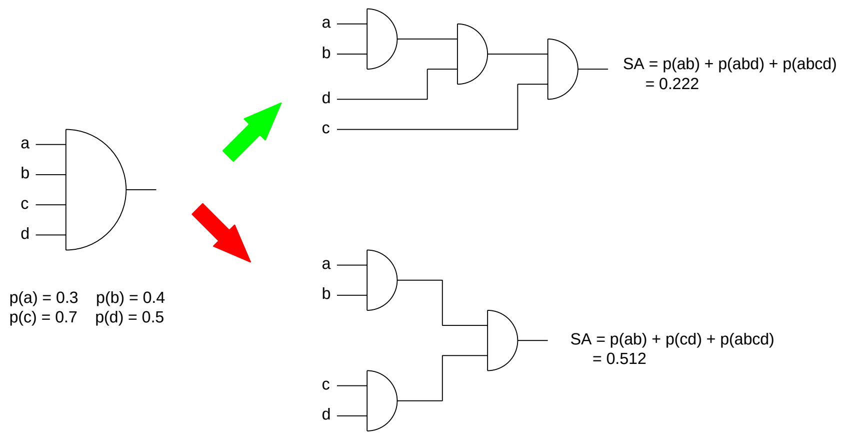

Figure 8. Technology decomposition to minimize the switching activity at internal nodes.

capacitance. When implementing signals with a high switching activity as inter-gate signals, the capacitance they have to switch depends on the actual routing of the corresponding net and hence cannot be predicted during technology mapping. It is therefore desirable to “cover” gates which are connected by strongly switching signals by as few library cells as possible. An example is shown in Figure 9. The bold dark lines represent highly active signals. The figure shows a cover of four library cells that allows hiding all active signals inside gates where they drive the least capacitance. Note that the functionality of the generic cell with a fanout of two is actually implemented twice in two different library cells.

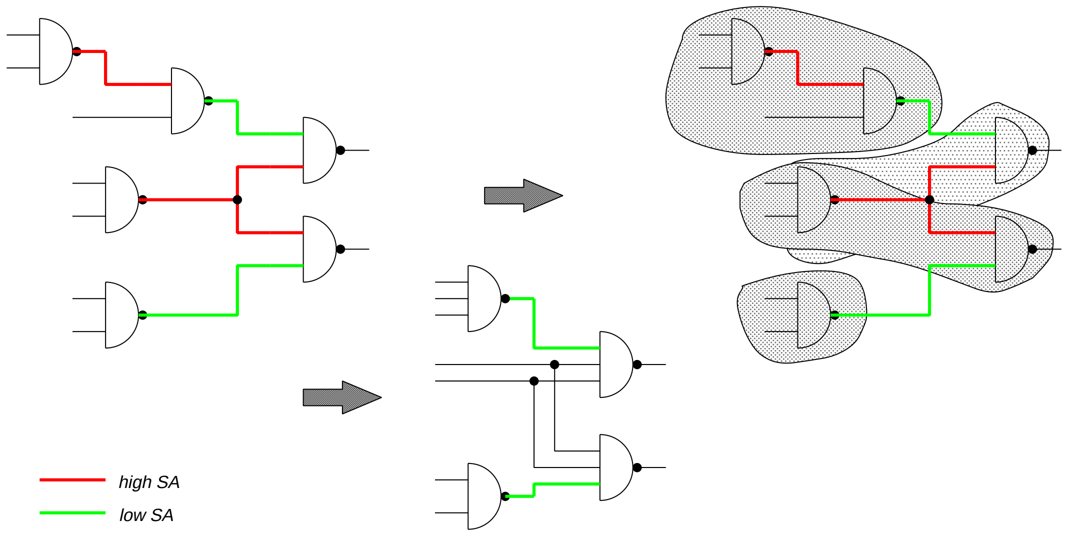

Figure 9. Technology mapping to hide highly active signals within gates.

Other gate level optimizations work by structurally modifying a netlist either prior to or after technology decomposition or mapping. *Pin swapping*, for instance, re-assigns nets to pins of gates such that the highest active nets are connected to the pins with the lowest input capacitance. *Buffer insertion* on signals with a high capacitive load reduces the time for ‘0’ $\leftrightarrow$ ‘1’ transitions and therefore the period of time during which a short circuit current contributes to the short circuit power consumption.

4.5 Transistor and physical level

On physical level, power consumption can be addressed through different circuit techniques. For both sequential and logic elements, a variety of different circuit techniques exist. Without going into further detail, it should be noted that all parameters affecting power consumption can be addressed on cicuit level, i.e. leakage power, switching power by minimizing the capacitance driven by highly active internal signals, short-circuit power by minimizing transition times, voltage swing, *etc.* In sequential elements, it is also desirable to minimize the number of clocked transistors to reduce the capacitance of the clock network, which itself is typically a major contributor to the overall power consumption. A good overview of low power circuit techniques can be found in (Rabaey and Pedram, 1996, Chapter 3) and (Chandrakasan and Broderson, 1998).

The physical design of the clock distribution network is of growing importance particularly as we are approaching gigahertz clock frequencies. Power consumption can be addressed by choosing *buffering schemes* which minimize the physical capacitance to be switched on the clock lines, or by relaxing *skew constraints* to allow for slightly unbalanced clock delays and therefore again lower capacitance. A comprehensive discussion of low power clock distribution can be found in (Rabaey and Pedram, 1996, Chapter 5).

5 Conclusion

Power consumption has become a critical design parameter for both high-performance and portable applications. Power therefore has to be addressed in all stages of the design flow, from system/algorithmic level down to the physical level. We have discussed the different parameters affecting the power consumption of digital CMOS designs, how they can be optimized on various levels of abstraction traversed during system design, and what improvements these optimizations can yield. It should be pointed out that while commercial tool support is readily available up to the gate level and partially on register-transfer level, many of the optimizations presented still require manual interaction by the designer. For more detailed discussions on various aspects of power consumption and how it can be optimized, the reader is referred to the references listed below.

References

Catthoor, F., Wuytack, S., Greef, E. D., Balasa, F., Nachtergaele, L., and Vandecappelle, A., *Custom Memory Management Methodology: Exploration of Memory Organisation for Embedded Multimedia System Design*, Kluwer Academic Publishers, 1998.

Chandrakasan, A. and Broderson, R., eds., *Low-Power CMOS Design*, IEEE Press, 1998.

Chandrakasan, A. P. and Brodersen, R. W., *Low Power Digital CMOS Design*, Kluwer Academic Publishers, 1995.

Chang, J.-M. and Pedram, M., Register Allocation and Binding for Low Power, in *Proceedings of the 32nd Design Automation Conference*, 1995.

Correale, A., Overview of the Power Minimization Techniques Employed in the IBM PowerPC 4xx Embedded Controllers, in *Proceedings of the 1995 ACM/IEEE International Symposium on Low Power Design*, pp. 75–80, 1995.

Dobberpuhl, D., The Design of a High Performance Low Power Microprocessor, in *Proceedings of the International Symposium on Low Power Electronics & Design '96*, pp. 27–34, 1996.

Gajski, D. and Kuhn, R., Guest Editors' Introduction: New VLSI Tools, *IEEE Computer*, *6*, 11–14, 1983.

Gronowski, P. E., Bowhill, W. J., Preston, R. P., Gowan, M. K., and Allmon, R. L., High-Performance Microprocessor Design, *IEEE Journal of Solid-State Circuits*, *33*, 676–686, 1998.

Gwennap, L., Pentium II Debuts at 300 MHz, *Microprocessor Report*, *11*, 1997.

Landman, P. E., *Low-Power Architectural Design Methodologies*, Ph.D. thesis, College of Engineering, University of California, Berkeley, 1994.

Landman, P. E. and Rabaey, J., Black-Box Capacitance Models for Architectural Power Analysis, in *Proceedings of the 1994 International Workshop on Low Power Design*, pp. 165–170, 1994.

Landman, P. E. and Rabaey, J. M., Power Estimation for High-Level Synthesis, in *Proceedings of the European Conference on Design Automation*, pp. 361–366, 1993.

Lin, Y.-R., Hwang, C.-T., and Wu, A. C.-H., Scheduling Techniques for Variable Voltage Low Power Designs, *ACM Transactions on Design Automation of Electronic Systems*, *2*, 1–22, 1997.

Liu, D. and Svensson, C., Power Consumption and Estimation in CMOS VLSI Circuits, *IEEE Journal of Solid-State Circuits*, *29*, 663–670, 1994.

Martin, R. S. and Knight, J. P., Optimizing Power in ASIC Behavioral Synthesis, *IEEE Design & Test of Computers*, pp. 58–70, 1996.

Mehra, R. and Rabaey, J., Exploiting Regularity for Low-Power Design, in *Digest of Technical Papers, ACM/IEEE International Conference on Computer-Aided Design*, pp. 166–172, 1996.

Monteiro, J., Devadas, S., Ashar, P., and Mauskar, A., Scheduling Techniques to Enable Power Management, in *Proceedings of the 33rd Design Automation Conference*, pp. 349–352, 1996.

Moore, G. E., Cramming More Components onto Integrated Circuits, *Electronics Magazine*, *38*, 114–117, 1965.

Münch, M., *Synthesis and Optimization of Algorithmic Hardware Descriptions*, Ph.D. thesis, University of Kaiserslautern, 1999.

Rabaey, J. M., *Digital Integrated Circuits: A Design Perspective*, Electronics and VLSI Series, Prentice Hall, 1996.

Rabaey, J. M. and Pedram, M., eds., *Low Power Design Methodologies*, Kluwer Academic Publishers, 1996.

Raghunathan, A. and Jha, N. K., Behavioral Synthesis for Low Power, in *Proceedings of the IEEE International Conference on Computer Design*, pp. 318–322, 1994.

SIA, *The National Technology Roadmap for Semiconductors*, SIA Semiconductor Industry Association, 1997.

Sproch, J., High Level Power Analysis and Optimization, *Tutorial, 1997 International Symposium on Low Power Electronics and Design*, 1997.

Sun Microsystems, UltraSPARC II Microprocessor Specifications, Sun Microsystems, available at `http://www.sun.com/microelectronics/UltraSPARC-II/`, 1998.

Taur, Y., The incredible shrinking transistor, *IEEE Spectrum*, pp. 25–29, 1999.

Tiwari, V., Malik, S., and Wolfe, A., Power Analysis of Embedded Software: A First Step Towards Software Power Minimization, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *2*, 437–445, 1994.

Tiwari, V., Malik, S., Wolfe, A., and Lee, M. T.-C., Instruction Level Power Analysis and Optimization of Software, *Journal of VLSI Signal Processing*, *13*, 1996.

Tiwari, V., Malik, S., and Ashar, P., Guarded Evaluation: Pushing Power Management to Logic Synthesis/Design, *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, *17*, 1051–1060, 1998.

Wuytack, S., *System-Level Power Optimization of Data Storage and Transfer*, Ph.D. thesis, Katholieke Universiteit Leuven, 1998.