Energy-Delay Complexity of Asynchronous Circuits

Thesis by

Paul Ivan Pénzes

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2002

(Submitted May 24, 2002)

Édesapának, Édesanyának, Péternek, Mártinak és Nagyapának

Acknowledgements

I wish to thank Alain J. Martin for giving me the opportunity to work in his research group, for providing me with intriguing research topics and for being a patient and dedicated advisor. For six years he diligently guided my research and kept me enthusiastic about all aspects of computer science. It has been a truly unique experience to work with Alain.

I also wish to thank the members of the Asynchronous VLSI Group at Caltech for many stimulating discussions: Andrew Lines, Rajit Manohar, Mika Nyström, Robert Southworth, Catherine Wong, Karl Papadantonakis, and José Tierno from IBM, TJ Watson Research Center.

My Caltech years would have not been as pleasant without Rocio, Attila, Calin, and my fellow graduate students in computer science. I thank you all!

Abstract

In this thesis, a *circuit-level theory of energy-delay complexity* is developed for asynchronous circuits. The energy-delay efficiency of a circuit is characterized using the metric Et^n, where E is the energy consumed by the computation, t is the delay of the computation, and n is a positive number that reflects a chosen trade-off between energy and delay. Based on theoretical and experimental evidence, it is argued that for a circuit optimized for minimal Et^n, the consumed energy is independent, in first approximation, of the types of gates (NAND, NOR, etc.) used by the circuit and is solely dependent on n and the total amount of wiring capacitance switched during computation. Conversely, the circuit speed is independent, in first approximation, of the wiring capacitance and depends only on n and the types of gates used.

The complexity model allows us to compare the energy-delay efficiency of two circuits implementing the same computation. On the other hand, the complexity model itself does not say much about the actual transistor sizes that achieve the optimum. For this reason, the problem of transistor sizing of circuits optimized for Et^n is investigated, as well. A set of analytical formulas that closely approximate the optimal transistor sizes are explored. An efficient iteration procedure that can further improve the original analytical solution is then studied. Based on these results, a novel transistor-sizing algorithm for energy-delay efficiency is introduced.

It is shown that the Et^n metric for the energy-delay efficiency index $n \geq 0$ characterizes *any* optimal trade-off between the energy and the delay of a computation. For example, any problem of minimizing the energy of a system for a given target delay can be restated as minimizing Et^n for a certain n. The notion of *minimum-energy function* is developed and applied to the parallel and sequential composition of cir-

cuits in general and, in particular, to circuits optimized through transistor sizing and voltage scaling. Bounds on the energy and delay of the optimized circuits are computed, and necessary and sufficient conditions are given under which these bounds are reached. Necessary and sufficient conditions are also given under which components of a design can be optimized independently so as to yield a global optimum when composed. Through these applications, the utility of the minimum-energy function is demonstrated. The use of this minimum-energy function yields practical insight into ways of improving the overall energy-delay efficiency of circuits.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation and Goals

One of the main motivations for originally introducing CMOS technology was its low power consumption. CMOS circuits were the first digital components that did not consume static power. Power was only consumed when the circuits were operating. By using CMOS, it was believed that the power-consumption problem was solved. Since then, both the speed and the complexity of digital circuits have increased exponentially. As a consequence, power is often the limit to circuit size or speed.

Power consumption has become a major concern in VLSI design for several reasons. First, it is hard to supply integrated circuits with substantial amounts of current. Providing enough current to the chip requires many dedicated pins that increase both the size and the cost of the design. Second, it is hard to extract the heat resulting from the power dissipation of the integrated circuit. Hot chips require not only costly cooling devices but possibly also more expensive packages. Third, today's widespread portable digital devices have available limited amount of energy. While power consumption can be traded for operating frequency, the total available energy, i.e., the power-delay product, remains limited. Improvements in battery technology are easily offset by the increasing complexity and performance of these devices. As a consequence, without proper attention to power management, the historical performance trend in CMOS VLSI design is not sustainable.

To help find methods that alleviate power consumption on the architectural/microarchitectural level, we first need a metric that allows us to compare two designs to see which is more efficient. The obvious choice for a low-power metric is the consumed power itself. However, this metric has a serious flaw. Given that energy is consumed mostly when circuit gates switch their outputs, one can always reduce the power by reducing the operating frequency, which is not a useful result.

An alternative metric could be the consumed energy, also called power-delay product, per operation. While this metric does not depend directly on the operating frequency, reducing the supply voltage reduces the required energy per operation. However, the lower supply voltage dramatically increases the delay of the operation, as well. As a consequence, the lowest-energy solution will also run very slowly.

To avoid these problems we need a metric that combines both the energy and the delay of the computation. Such a metric will help us optimize a VLSI system for energy-delay efficiency at several levels: architecture, circuit implementation, or physical realization. The goal of this work is to focus on the circuit implementation and to develop a *circuit-level theory of energy-delay complexity*. This theory will allow us to make accurate estimates of circuit performance and will facilitate a comparison of circuit-design choices at an abstract level, without going through the costly implementation steps of transistor sizing, layout, and electrical simulation.

Asynchronous circuits are particularly well fitted for energy-delay efficient design, since they are fast and use little energy. Asynchronous circuits do not use clocks. Communication and synchronization among circuit blocks are implemented by handshake protocols. The particular type of asynchronous circuits we are interested in are called quasi delay-insensitive, or QDI [30]. QDI circuits use no timing assumptions on the delays of the operators and wires, with the exception of some forks, called *isochronic forks*, in which the delays on the different branches of the fork are assumed to be similar. QDI circuits are the most conservative asynchronous circuits in terms of the use of delays. As a consequence, they are also the most robust to physical parameter variations. QDI circuits are interesting in the context of energy-delay efficient design because, first, they are inherently energy efficient owing to the absence of a global clock, locality of activity, automatic shutoff of inactive parts, absence of spurious transitions (glitches); and second, they exhibit an average-case delay of operation, as opposed to the worst-case delay experienced in synchronous circuits. The latter property eliminates timing margins and superfluous synchronizations and allows the circuit to run at its highest speed.

1.2 Contributions of this Thesis

This thesis makes the following contributions:

- It introduces and motivates a general Et^n metric for energy-delay efficiency.
- It develops an energy-delay complexity model for Et^n optimal circuits.
- It provides a set of transistor-sizing formulas for circuits optimized for energy-delay efficiency.
- It introduces the *minimum-energy function* as an abstract approach to energy-delay efficiency.
- It applies the concept of minimum-energy function to system-level design.

1.3 Organization of this Thesis

This thesis is organized as follows. Chapter 2 describes the electrical model used to characterize the energy consumption and delay of circuits. Chapter 3 introduces an energy-delay efficiency metric that combines the energy E and the delay t of a computation in the form Et^n. Chapter 4, defines the energy-delay complexity model for Et^n-optimal circuits. Theoretical and experimental evidence is presented in support of this complexity model. The same chapter explores a set of analytical formulas that closely approximate the optimal transistor sizes of an Et^n-optimal circuit. An efficient iteration procedure that can further improve the original analytical transistor sizing solution is studied. Based on these results, a novel transistor sizing algorithm for energy-delay efficiency is introduced. In Chapter 5, it is shown that the Et^n metric for the energy-delay efficiency index $n \geq 0$ characterizes *any* optimal trade-off between the energy and the delay of a computation. For example, any problem of minimizing the energy of a system for a given target delay can be restated as minimizing Et^n for a certain n. In the same chapter, the notion of *minimum-energy function* is developed and applied to the parallel and sequential composition of circuits in general and in particular to circuits optimized through transistor sizing and voltage scaling. Chapter 6 sums up the main results, makes some concluding remarks about the thesis, and suggests possible future extensions. Finally, the Appendix presents an energy estimation method for asynchronous circuits with application to an asynchronous microprocessor.

Chapter 2

Energy and Delay in CMOS Circuits

Our goal is to develop a circuit-level theory of energy-delay complexity. We want to introduce a theory that enables an abstract view on energy-delay-optimal circuits. As we will see in the next chapter, energy-delay optimality is defined in terms of the energy consumption and speed of the circuit. In this chapter, we describe the sources of energy consumption and delay in CMOS circuits. At the same time, we present the electrical model we will use to formalize the energy-delay optimization problem.

The most accurate way to compute the energy consumption E and delay t of a circuit is to solve the characteristic nonlinear equations of each transistor, e.g., with `hspice` [49]. However, this approach is computationally very expensive even for small circuits. Furthermore, building a simple abstraction of circuit level energy-delay optimality based on such a complicated model seems impossible.

The increased accuracy of transistor equations does not necessarily yield an increased prediction accuracy about the circuit. This is because more precise equations rely on a large set of process-dependent parameters that might not be known at the time of design or might have large variations at fabrication.

The results to be developed in this thesis are intended to be simple and easily usable by a circuit designer. This way, these results can be applied early on in the design process to reduce the space he has to search to find the optimal circuit solution. As a consequence, several simplifying assumptions will be made about the behavior of CMOS gates, resulting in a simple mathematical description of circuits.

2.1 Energy Dissipation in CMOS Circuits

CMOS circuits have three main sources of energy dissipation: dynamic currents (due to the charging and discharging of capacitors), short-circuit currents, and leakage currents. The total energy dissipated during operation can be written as follows:

$$E = E_{dynamic} + E_{short} + E_{leakage}.$$

In a CMOS logic gate, short-circuit current is present when a direct path from the power supply to ground exists. In a static CMOS inverter, there is no short-circuit current flow during the absence of transients on the input; however, during a transient on the input, there will be a time period during which both the nMOS and the pMOS transistors will conduct (if $V_{dd} > V_{T_n} + |V_{T_p}|$, where V_{dd} is the supply voltage, V_{T_n} and V_{T_p} are the corresponding nMOS and pMOS threshold voltages), causing a short-circuit current to flow from the power supply to the ground. This current flows as long as $V_{T_n} < V_{in} < V_{dd} - |V_{T_p}|$, where V_{in} is the voltage on the input signal.

Veendrick [23] has shown that, under some simplifying assumptions about the shape of the input signal, the power dissipation of a symmetric inverter (equal gain factors $\beta_n = \beta_p = \beta$ and equal threshold voltages $V_{T_n} = -V_{T_p} = V_T$) with no output capacitance is

$$P_{short} = \frac{\beta}{12}(V_{dd} - 2V_T)^3 \frac{\tau}{T},$$

where τ is the rise and fall time of the input signal and T is the time period on which the power is measured.

More recently, several researchers [24, 26, 27, 28] have extended Veendrick's work to account for carrier velocity saturation and gate-to-drain coupling. While Veendrick's work estimated the maximum short-circuit power dissipation to be up to 20% of the total power consumption, the more recent works place the average short-circuit contribution to about 1%. Based on these, for the purpose of this work, we will consider the energy contribution of short-circuit currents zero, i.e., $E_{short} = 0$.

There are two types of leakage currents: reverse-bias diode leakage on the transistor drains, and sub-threshold leakage through the channel of an "off" device. The magnitude of these currents is determined by the various fabrication-technology parameters. Diode leakage occurs when a transistor is turned off, and another active transistor charges up/down the drain with respect to the bulk potential of the first transistor. The leakage current is

$$i_o = i_s(e^{\frac{qV}{kT}} - 1),$$

where i_s is the reverse saturation current, V is the diode voltage, q is the electron charge, k is Boltzmann's constant and T is the temperature [57]. The magnitude of the sub-threshold current is a function of fabrication technology, device sizing, and is proportional to $e^{\frac{q(V_{gs}-V_T)}{kT}}$, where V_{gs} is the gate-to-source voltage [56].

In the past, energy dissipation due to leakage currents represented a small fraction of the total energy consumption of a CMOS circuit. In today's submicron technologies, the relative importance of leakage currents has increased, particularly in the case of technologies with lowered threshold voltages targeted for high-speed devices. On the other hand, in low-power fabrication technologies that keep a relatively high threshold voltage, leakage currents can still be considered negligible. For the purpose of this work, we assume a low-power fabrication technology and ignore the energy contribution of leakage currents, i.e., $E_{leakage} = 0$.

The largest contribution to the energy consumption of a CMOS circuit is due to the dynamic currents charging and discharging the capacitive nodes of the circuit. This energy dissipation has the form

$$E_{dynamic} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{m} C_i n_i \tag{2.1}$$

where C_i is the total capacitance of node i, m is the number of energy consuming

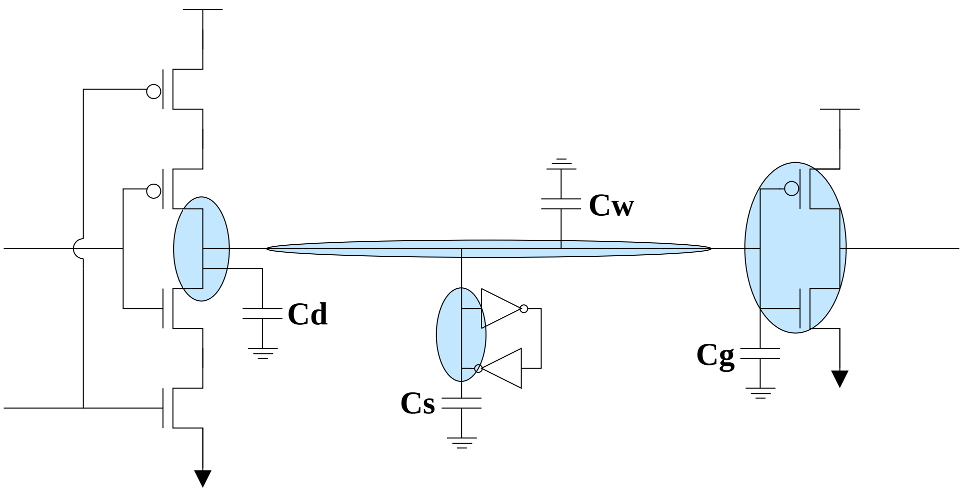

Figure 2.1: Capacitance of a CMOS circuit node.

nodes, and n_i is the *transition count* of node i during the measured time period.

I have shown that the dynamic energy is responsible for more than 90% of the total energy consumption of a wide class of CMOS circuits [50], including a two-million-transistor asynchronous MIPS microprocessor [33]. Other researchers have arrived at similar conclusions [17, 18, 19, 20]. Therefore, for the purpose of this work, we focus our attention on the dynamic energy consumption of CMOS circuits.

Assume, without loss of generality, that during the operation of an asynchronous circuit each of its nodes makes exactly two transitions, i.e., $n_i = 2$ (this is equivalent to "renaming" or "unrolling" nodes that see more than two transitions and ignoring the ones that never transition). Based on this assumption, the energy consumption given by Equation 2.1 can be written as

$$E = V_{dd}^2 \sum_{i=1}^{m} C_i. \tag{2.2}$$

As shown in Figure 2.1, the capacitance C_i of node i can be written as $C_i = C_{gi} + C_{wi} + C_{di} + C_{si}$, where C_{gi} is the gate capacitance of the transistors that are driven by node i, C_{wi} is the wiring capacitance on node i, C_{di} is the source/drain diffusion capacitance of the logic gates driving node i, and C_{si} is the capacitance due to the staticizer in case node i is state holding. Given the relatively small contribution of C_{di} and C_{si} to C_i [50], for the purpose of this work, we assume for all circuit nodes i that $C_{di} = 0$ and $C_{si} = 0$.

There are other capacitive nodes present in the actual circuit. They are mainly internal nodes of transistor chains. We choose to ignore the energy dissipated in charging and discharging these internal nodes. We do this because, on the one hand, their capacitance is usually negligible compared to the other capacitive terms and, on the other hand, because these internal nodes do not always do full swings between the power rails.

If we assume node i is driving transistors of minimum length l_{min} that have total gate width $\sum w$, then $C_{gi} = C_{ox} l_{min} \sum w$, where the oxide capacitance $C_{ox} = \epsilon_{ox}/t_{ox}$ with ϵ_{ox} the permittivity of the gate insulator and t_{ox} the thickness of the gate insulator (See Figure 2.2). The minimum-length assumption is reasonable, since there is usually no reason to set the lengths of transistors in a digital circuit to anything other than the minimum allowed by the fabrication technology: increasing the length increases both the resistance and the capacitance and hence worsens both the energy and delay. Only for special circuits, such as staticizers or pad drivers, devices might not have minimal length. Let us define p_i as the wire capacitance of node i normalized in transistor widths units, i.e., $p_i = C_{wi}/(C_{ox} l_{min})$. With these assumptions, we can write the total capacitance on node i as

$$C_i = C_{gi} + C_{wi} = C_{ox} l_{min} \Big(\sum w + p_i \Big). \tag{2.3}$$

With $K_E = C_{ox} l_{min} V_{dd}^2$, Equation 2.2 becomes

$$E = K_E \sum_{i=1}^{m} \Big(\sum w + p_i \Big). \tag{2.4}$$

Equation 2.4 expresses, under our assumptions, the total energy consumption of a CMOS circuit as a function of the transistor widths w and wire parasitics p_i.

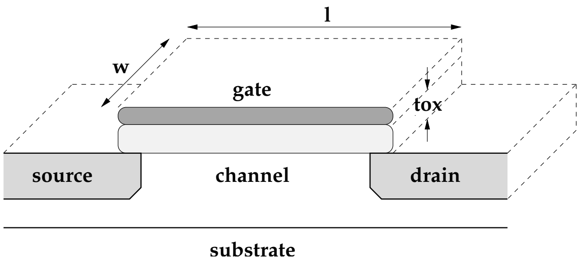

Figure 2.2: A MOFSET.

2.2 Delay in CMOS Circuits

Delay in a CMOS circuit results from two sources: first, the delay taken by logic gates to switch their outputs, i.e., to charge or discharge their output capacitance; and second, the delay taken by signals (waveforms) to propagate through the wires connecting the logic gates of the circuit. For the purpose of this work, we focus on the logic gate delays and ignore the propagation delay through wires, by assuming that wire resistance and time-of-flight are zero.

A CMOS transistor uses the charge on its gate to control the movement of the charge carriers (electrons or holes) between source and drain through the channel under the gate (See Figure 2.2). For small drain-to-source voltage V_{ds}, the time it takes for a carrier, traveling at an average velocity v, to get through the channel of length l is given by [56]

$$\tau = \frac{l}{v}. \tag{2.5}$$

The velocity v is proportional to the electric field E driving the carrier:

$$v = \mu E = \mu \frac{V_{ds}}{l}, \tag{2.6}$$

where the proportionality constant μ is the mobility of the charge carrier under the given electric field in the conducting material of the channel region. Combining Equations 2.5 and 2.6, we get

$$\tau = \frac{l}{v} = \frac{l^2}{\mu V_{ds}}. \tag{2.7}$$

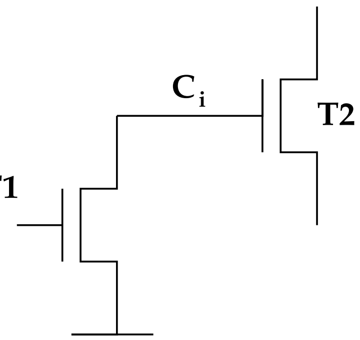

Figure 2.3: One transistor driving another transistor.

Equation 2.7 provides, under the stated assumptions, the delay τ of a one-square transistor charging or discharging another one-square transistor. Assume now, as shown in Figure 2.3, that a transistor T1 charges or discharges not only a one-square transistor but an output load capacitance C_i. Let us define the rise time t_r as the time it takes for a node to rise from 10% to 90% of its steady-state value and the fall time t_f as the time it takes for a node to fall from 90% to 10% of its steady-state value. When both the linear and the saturation operating regions of a transistor are taken into consideration, it can be shown that [57]

$$t_f = k\frac{C_i}{\beta_n V_{dd}}, \quad \text{with} \quad \beta_n = \mu_n C_{ox}\frac{w_n}{l_n}, \tag{2.8}$$

$$t_r = k\frac{C_i}{\beta_p V_{dd}}, \quad \text{with} \quad \beta_p = \mu_p C_{ox}\frac{w_p}{l_p}, \tag{2.9}$$

where k is a constant that dependents on the threshold voltage and the supply voltage V_{dd}, μ_n, μ_p are the effective surface mobilities of electrons and holes in the channel. The length and width of transistor T1 are l_n, w_n in case T1 is an nMOS transistor (as shown in Figure 2.3) or l_p, w_p in case T1 is a pMOS transistor.

Based on Equation 2.3, the output load C_i has two components: the load due to the gate capacitances of output transistors such as T2, and the wire capacitance connecting the source/drain of transistor T1 to the gate of output transistors such as T2. If we assume that transistor T2, and all other transistors connected to the output of T1 have length $l = l_{min}$ and total widths $\sum w$, we can write $C_i = C_{ox} l_{min} \sum w + C_{ox} l_{min} p_i$, where p_i is the wire capacitance at node i normalized in transistor widths units. If we further assume that transistor T1 has length $l_n = l_p = l_{min}$, Equations 2.8 and 2.9 become

$$t_f = K_t \frac{\sum w + p_i}{w_n}, \tag{2.10}$$

$$t_r = K_t \frac{\sum w + p_i}{\frac{w_p}{\mu}}, \tag{2.11}$$

where $K_t = k l_{min}^2 / (\mu_n V_{dd})$ and $\mu = \mu_n / \mu_p$ is the ratio of electron mobility to hole mobility.

The delay of a simple logic gate may be approximated by constructing an "equivalent" inverter [57]. This is an inverter whose pull-down nMOS and pull-up pMOS transistors have sizes that reflect the effective strength of the real pull-up and pull-down of the logic gate. If we assume that the resistance of a transistor channel is inversely proportional to the channel widths, one way to construct the equivalent inverter is to consider its nMOS and pMOS channel resistance equal to the corresponding sum of channel resistances in the pull-down or pull-up of the original gate. For example, if a logic gate has a pull-down composed of k_n transistors of widths w_{ni}, and a pull-up composed of k_p transistors of widths w_{pi}, then the equivalent inverter has the nMOS transistor of widths

$$w'_n = \frac{1}{\sum_{i=1}^{k_n} \frac{1}{w_{ni}}}, \tag{2.12}$$

and the pMOS transistor of widths

$$w'_p = \frac{1}{\sum_{i=1}^{k_p} \frac{1}{w_{pi}}}. \tag{2.13}$$

If all transistors in the pull-down of a logic gate are assumed to be equal to w_n, and all transistors in the pull-up of a logic gate are assume to be equal to w_p, Equations 2.12 and 2.13 simplify to

$$w'_n = \frac{w_n}{k_n}, \tag{2.14}$$

$$w_p' = \frac{w_p}{k_p}. \tag{2.15}$$

Using Equations 2.14 and 2.15, Equations 2.10 and 2.11 generalize to simple logic-gates as

$$t_f = K_t k_n \frac{\sum w + p_i}{w_n}, \tag{2.16}$$

$$t_r = K_t k_p \frac{\sum w + p_i}{\frac{w_p}{\mu}}. \tag{2.17}$$

where the logic gate driving the output load is assumed to have a pull-down composed of a chain of k_n transistors of identical widths w_n and a pull-up composed of a chain of k_p transistors of identical widths w_p. We will also refer to k_n and k_p as logic-gate topologies, since they determine the connectivity of the given logic gate to the rest of the gate network. Further theoretical and experimental evidence in support of Equations 2.16 and 2.17 can be found in [25, 54].

Another way to determine the k_n and k_p of a logic gate is through electrical simulation. One can setup an `hspice` simulation in which the delay of different length pull-down and pull-ups is compared with the delay of the pull-down and pull-up of a single gate for various transistor widths and output loads.

The operation of an asynchronous circuit, i.e., of an asynchronous logic-gate network, can be represented as a directed graph in which each rising and falling transition of a logic gate has a corresponding vertex and each input of a logic gate has a corresponding edge [7, 8]. The vertex corresponding to the rising transition of a logic-gate output represents the pull-up of the logic gate, while the vertex corresponding to the falling transition represents the pull-down of the logic gate. In such a graph, a *path* corresponds to the sequence of switched logic-gate outputs in a given execution and a closed path forms a *cycle*. What the total delay of a CMOS circuit is, depends on how delay is measured. If we are interested in a repetitive system, the measure of delay is the *cycle time* of the circuit [7], i.e., the time between two consecutive rising or falling transitions of a given node on the critical cycle. On the other hand, if we are interested in a non-repetitive system, the measure of delay is *latency*, i.e., the time taken by the circuit from input arrival to output generation. In either case, the delay of the logic-gate network is given by the maximum sum of gate delays over all relevant cycles or paths of the circuit, depending on the chosen measure of delay: cycle time or latency.

In order to avoid ambiguity between the measure of delay (cycle time or latency), we will state our results in the context of repetitive systems in which the relevant measure of delay is cycle time. However, our arguments extend, unless stated otherwise, to nonrepetitive systems where the measure of delay is latency. The difference is that instead of analyzing the cycles of a circuit, one has to consider the relevant paths of the circuit.

In writing the energy consumption and delay equations, we have made several simplifying assumptions. In particular, we have assumed that all transistors in a given pull-up/pull-down have the same widths. The great benefit of this assumption is that it significantly reduces the parameter space. Furthermore, the energy and delay equations now become *posynomial* functions of the transistor widths. A posynomial in variables w_i is a function of the form $f(w_0, w_1, ..., w_{m-1}) = \sum_{0<i\leq q} \alpha_i w_0^{\beta_i^0} w_1^{\beta_i^1} ... w_{m-1}^{\beta_i^{m-1}}$ where $\alpha_i \geq 0$. A *posynomial problem* is the minimization of one posynomial while simultaneously satisfying a set of upper-bound constraints on other posynomials. With the substitution $w_i = e^{x_i}$, a posynomial can be transformed into a convex function. A convex function has the property that a local minimum must necessarily be a global minimum. This property greatly simplifies the task of finding energy-delay-optimal circuits.

Figure 2.4 illustrates a circuit fragment in which a Muller C-element is driving an inverter. Based on Equation 2.4, the energy consumed in charging and discharging node i is

$$E = K_E(w_{pout} + w_{nout} + p);$$

based on Equation 2.16, the time it takes to discharge node i is

$$t_f = K_t 2 \frac{w_{pout} + w_{nout} + p}{w_n};$$

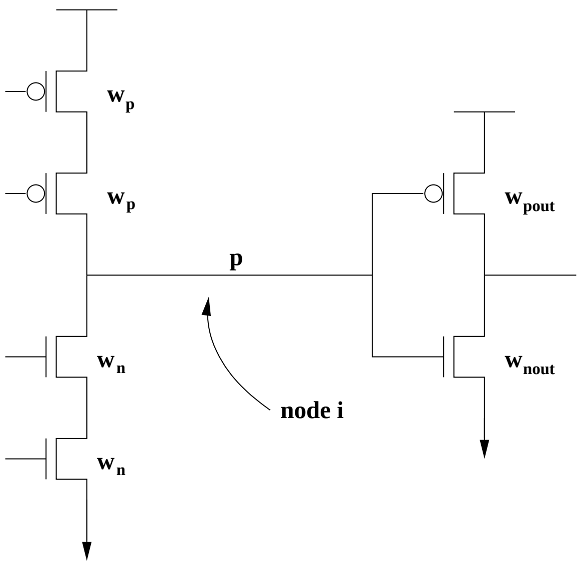

Figure 2.4: CMOS circuit fragment.

and based on Equation 2.17, the time it takes to charge node i is

$$t_r = K_t 2 \frac{w_{pout} + w_{nout} + p}{\frac{w_p}{\mu}},$$

where w_p, w_n, w_{pout}, w_{nout} are the corresponding transistor widths and p is the wire capacitance normalized in transistor widths units.

As we will see in Chapter 3, we combine E and t into a single energy-delay efficiency metric. When minimizing such a metric, the technology-dependent constants K_E and K_t drop out of the optimization. As a consequence, without loss of generality we assume $K_E = K_t = 1$.

In this context, based on Equations 2.4, 2.16, 2.17, and the argument about the total circuit delay, we formalize the energy consumption E and delay t of a CMOS circuit:

$$E = \sum_{all\ nodes\ i} \Big(\sum_{\substack{all\ gates\ j \\ connected\ to\ i}} w_j + p_i \Big), \tag{2.18}$$

with p_i the wire capacitance on node i;

$$t = \max_{all\ cycles\ k} t_k, \tag{2.19}$$

$$t_k = \sum_{\substack{all\ gates\ g \\ of\ cycle\ k}} t_g, \tag{2.20}$$

with t_g given by Equation 2.16 or 2.17.

Clearly, a simple transistor model like the one just described does not capture all the details of the nonlinear behavior of transistors. Nor does it capture the issues related to the specific fabrication technology in which a circuit is eventually implemented. However, based on [55, 56, 57], we believe that such a simple model still constitutes a good basis for making relevant predictions about the future performance of the scrutinized circuit.

Chapter 3

An Energy-Delay Efficiency Metric

In an ideal world, an energy-delay efficient VLSI computation would have both the optimal energy consumption E and the optimal delay t. This is never the case in reality. There exists a trade-off between the speed of the computation and its energy consumption; i.e., the best energy consumption and the best delay are not achievable at the same time. For the purpose of this work, we shall combine the two figures of merit of a computation—E and t—into a single metric. There are many ways E and t can be combined. In this chapter we look at several of these possibilities.

3.1 The Et^2 Metric

The first optimization metric that we consider combines energy and delay in a way that is independent of voltage. It has been argued in [1, 33] that the best such metric has the form $\Theta = Et^2$. The argument goes as follows. Consider a node i of a circuit having capacitance C_i. The energy spent to charge node i is approximately $E_i = \frac{C_i V_{dd}^2}{2}$, where V_{dd} is the power-supply voltage. The delay to switch node i, given that the transistor behavior is dominated by the saturation regime, is approximately $t_i = \frac{C_i}{K V_{dd}}$, where K is a constant that depends on the structure of the gate operating on node i. The energy consumed by a circuit can be seen as the sum of the energies consumed by each individual circuit node. Similarly, the delay of a circuit can be seen as the sum of the delays of individual circuit nodes on the critical path. As a consequence, the total energy can now be written as

$$E = k_E V_{dd}^2, \tag{3.1}$$

while the delay can be written as

$$t = \frac{k_t}{V_{dd}}, \tag{3.2}$$

where k_E and k_t are voltage-independent quantities. If we combine these last two equations we get

$$\Theta = Et^2 = k_E k_t^2, \tag{3.3}$$

which is independent of the supply voltage.

There are several simplifying assumptions made in writing the previous equations. There are several operating modes for the CMOS transistor, each with a very different relation between current and voltage. In particular, at high electric fields, the carrier velocity saturates and becomes constant; the delay becomes independent of the voltage, and $\Theta = Et^2$ becomes quadratic in the voltage. Short-circuit and leakage currents—ignored for now—could potentially affect both E and t. It is then natural to ask how constant Et^2 really is in reality.

Figure 3.1 shows the measured $\Theta = Et^2$ for the two-million–transistor asynchronous MIPS R3000 microprocessor designed at Caltech between 1995 and 1998. It was fabricated in 0.6-μm CMOS and was entirely functional on first silicon [33]. (Measurements on other fabricated chips give similar results.) The behavior below 1.3 V shows the effect of approaching the threshold voltage; in our calculations we have assumed that the threshold voltage would scale with V_{dd}, but we obviously cannot enforce this for HP's 0.6-μm process, whose threshold voltage is fixed at 0.8 V. The positive slope from 3 V and up shows the onset of velocity saturation. The nominal voltage of this process is 3.3 V; the graph shows that Et^2 varies only about 20% around its average when V_{dd} is in the range 1.5–4.9 V.

With the metric Θ at hand, if we desire to change to a particular delay target t, we adjust the voltage to meet it, and a circuit optimized for Θ would have the best E for

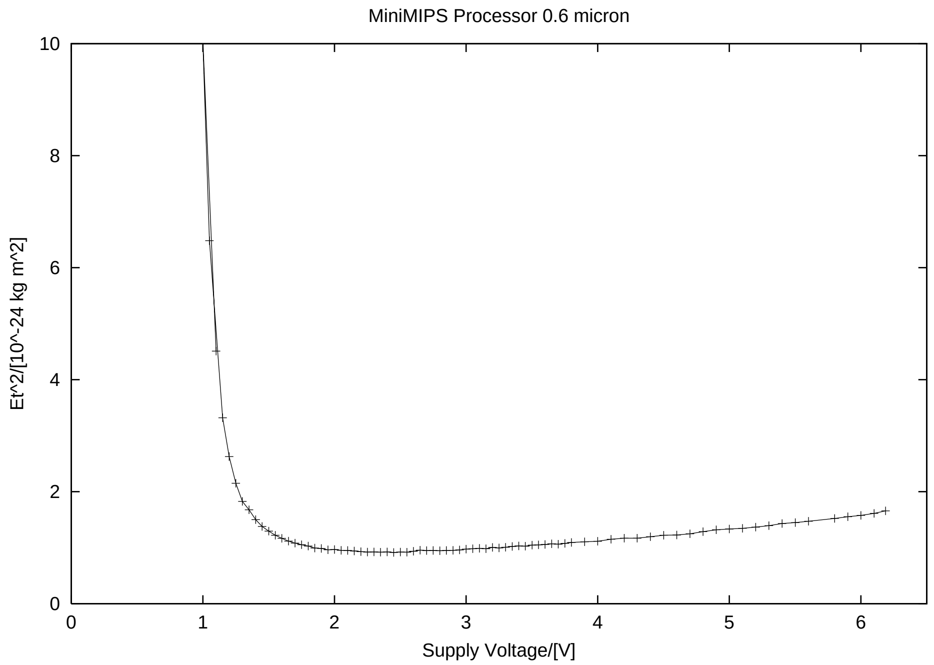

Figure 3.1: Measured Et^2 for a two-million–transistor asynchronous microprocessor.

that t. Likewise, we may choose an energy target E and get the best t instead. Thus, a voltage-independent metric Θ could express the voltage scaling trade-off between E and t.

3.2 Why Et is not the Right Metric

Several researchers [3, 4] use the energy-delay product Et as a measure of energy-delay efficiency. Given the voltage dependence of this metric, comparing designs using Et yields misleading results, as Martin has pointed out [1].

Assume the same computation can be implemented by two different circuits A and B. Furthermore, assume that $E_A = 2E_B$ and $t_A = \frac{t_B}{2}$. Using the Et metric, circuit A is as energy-delay efficient as circuit B. If we now reduce the supply voltage of circuit A by half, the new energy consumption of circuit A is $\frac{E_A}{4}$ (given that energy scales quadratically with supply voltage as in Equation 3.1), while the new delay is $t'_A = 2t_A$ (given that delay is inversely proportional to supply voltage as in Equation 3.2). As a consequence, $E'_A = \frac{E_B}{2}$ while $t'_A = t_B$.

This calculation shows that circuit A consumes only half the energy of circuit B when both circuits run with the same delay. This holds for any delay. Therefore, circuit A is clearly a more energy-delay efficient implementation than circuit B, contrary to what the Et metric indicates.

The results suggested by the previous calculation have been checked in practice. In Table 2.1, we see the results of simulating with `hspice` two different 0.6-μm implementations of an 8-bit comparator. In each case, eight single-bit comparators perform the comparison: in the "linear" comparator, the results of the single-bit comparators are merged in a linear chain; in the "log" comparator, in a binary tree. Comparing the performance of the comparators at 3.3-V V_{dd}, we see that the linear comparator is slower than the log comparator, but using the Et metric, we find that it more than makes up for its sluggishness with its lower energy consumption. On the other hand, using the Et^2 metric, we find that the log comparator is better. Which is it?

8-bit comparator	$E/[J \cdot 10^{-11}]$	$t/[s \cdot 10^{-9}]$	$Et/[Js \cdot 10^{-20}]$	$Et^2/[Js^2 \cdot 10^{-29}]$
Linear (3.3V)	25.24	3.93	99.21	389.97
Log (3.3V)	44.97	2.35	105.52	247.57
Log (2.15V)	16.52	3.93	64.97	255.59

Table 2.1: Comparison of E, t, Et, and Et^2 of two kinds of 8-bit comparators.

If we adjust the supply voltage on the log comparator down to 2.15 V, we see that we can match the delay of the linear comparator while using less energy; thus, the log comparator outperforms the linear one in both speed and energy if we are allowed to adjust the supply voltage. Even over this relatively wide range of supply voltages, Et^2 changes only by an insignificant 3.2%. This example illustrates that the Et^2 metric is more trustworthy for circuit comparisons when we are allowed to adjust the supply voltage.

3.3 The Et^n Metric

We have seen theoretical and experimental motivation for Et^2. At this point we introduce a more general energy-delay efficiency metric: Et^n with $n \geq 0$. There are many reasons why we wish to study this more general metric over Et^2, despite the voltage independence of Et^2 over a wide range. Of course, any result obtained for Et^n applies to Et^2 by simply substituting $n = 2$—nothing relevant about Et^2 would be missed by studying the metric Et^n.

We have seen in Figure 3.1 that sub-threshold current and velocity saturation cause Et^2 to be voltage dependent for extremely high and low supply voltages. If a circuit were optimized to operate near sub-threshold or velocity saturation, one might consider expending more energy for a given speed than suggested by the Et^2 metric. By the same token, sometimes we wish to optimize a circuit for speed with little regard for energy; this would correspond to our optimizing Et^n, for n much greater than two. It should be noted that optimizing *entirely* for speed is not possible because the wire parasitics cannot be *fully* overcome without using infinitely large transistors (as will be shown in Chapter 4). Similarly, if we are interested in optimizing a circuit for energy with little regard for speed, we should choose n smaller than two.

If the only available design parameter should be supply voltage and if we could adjust it up or down without bound, and the transistors should maintain the simple first-order behavior we used for justifying the Et^2 metric—that is to say, if the applicability of the Et^2 metric were perfect—then we should need no metric other than Et^2, because Et^n, for all n, would be optimized by optimizing Et^2 (as will be shown in Chapter 5). Furthermore, all components of the original system would be Et^2 optimal. However, neither of these assumptions is true. First, we know that the applicability of Et^2 is unfortunately not perfect; it is sometimes better to use Et^n with n different from 2 as the metric when the target we are striving for would make us operate an Et^2-optimal circuit outside the practically allowable and convenient range of supply voltages. Second, there are other design parameters besides supply voltage (such as micro-architecture or transistor sizing) that, even though chosen so as to globally optimize a system for Et^2, would not imply the Et^2-optimality of the system's components. Conversely, the Et^2-optimality of the components would not imply the Et^2-optimality of the whole (as will be shown in Chapter 5). In other words, it is clearly feasible to have a globally Et^2-optimal system with components optimized for Et^n with $n \neq 2$. We will explore this possibility in detail in Chapter 5.

Intuitively, the metric Et^n implies that a 1% improvement in speed is worth roughly an n% increase in energy consumption. If one values the computation delay without regard to the consumed energy, the efficiency metric index n should be increased without bound. Conversely, if one values the energy consumed by the computation without regard to the computation delay, one should let the efficiency metric index be $n = 0$. As a consequence, the metric Et^n quantifies—through the single parameter n—the entire range of optimal preferences in the trade-off between energy and delay.

Even if one were to care only about $n = 2$, working out the formulas in terms of Et^n offers a deeper insight into the mathematical structure of the expressions for energy and delay. Furthermore, some results are entirely independent of n, a very useful property that allows certain Et^n problems to be treated the same way as their simpler counterparts with $n = 0$ or n infinitely large, which are often easier to analyze.

Chapter 4

Transistor Sizing for Optimal Et^n

4.1 Introduction

In this chapter we study the problem of transistor sizing for optimal Et^n. Transistor sizing is the next step in the design process after netlist generation; given a netlist, we must choose transistor sizes that ensure correct functionality and optimal performance. More precisely, given a transistor netlist where each transistor i has width w_i and length l_i, transistor sizing finds the values of w_i and l_i that minimize Et^n, with the constraints $w_i \geq w_{min}$, $l_i \geq l_{min}$. While it is true that most layout systems demand that transistor sizes be quantized to some grid, we ignore this constraint.

Also, we can set all l_is to l_{min} and remove them from consideration since there is usually no reason to set the lengths of transistors in a digital circuit to anything other than the minimum allowed by the fabrication technology: increasing the length increases both the resistance and the capacitance and hence worsens both the energy and delay. Furthermore, we assume $w_{min} = 0$; in other words, we assume that the positive transistor widths can be made arbitrarily small.

The sized transistors of a circuit are connected to each other through wires. The capacitance of these wires leads to additional energy and delay. For delay-only optimization, which can be phrased as the minimization of the metric Et^n for very large n, the wire capacitance can be overcome by increasing transistor sizes where appropriate. One might worry that this could lead to a vicious cycle in which wires must get longer in order to go past the larger transistors. The following argument shows that this is not the case: assume that each device in the circuit is increased by a factor K by increasing its channel width by $\sqrt{K}$ and by replicating it in parallel $\sqrt{K}$ times. This process would increase the wiring cost by $\sqrt{K}$ while the device drive increased K times. In this way, for large enough K, the delay due to wiring could be made arbitrarily small. Conversely, for energy-only optimization, when $n = 0$, the transistor widths can be chosen to be minimum size, independently of the wire capacitance. In contrast to these special cases, for n small but nonzero, wire capacitance cannot be ignored or overcome in a straightforward way, and the optimal transistor sizes depend strongly on this capacitance.

Classical numerical methods, such as the conjugate gradient descent method, have been applied to the transistor-sizing problem: there exist several transistor-sizing programs that minimize power consumption while maintaining performance specifications [34, 35, 36]. While these tools can be used successfully for small circuits, they tend to be inadequate for large circuits owing to their $O(N^3)$ (or worse) runtime. Part of the problem is the size of the circuits: today's large VLSI systems have tens of millions of devices; the optimization space has a similar number of dimensions. Classical numerical methods that do not make use of special properties of CMOS circuits might not converge, or their convergence might be very much dependent on the initial values. For large problems, such as VLSI transistor sizing, an algorithm's polynomial runtime is no guarantee of usability. Techniques for reducing the solution search space and improving the initial values could potentially greatly extend the power of numerical methods for energy-delay optimization.

In the VLSI design flow, it is important to be able to assess the energy-delay performance of a circuit as early as possible, so as to avoid costly redesign. While numerical methods could ultimately yield the Et^n-optimal transistor sizes once these practical difficulties are overcome, the methods still lack the potential to assist designers in selecting among alternative circuit solutions prior to sizing or to give insight into ways to change the original circuit so as to improve the achievable optimum. For instance, a designer that wants to choose one implementation out of a family of possible ones would rather use a roughly correct analytic expression than a more accurate numerical solution, especially if the analytic expression could be parameterized so that one expression is valid for the entire family. For this reason, we first propose a simple energy-delay complexity model that abstracts transistor sizing and allows a circuit level comparison of design choices. Later in the chapter, we develop a set of analytical formulas that closely approximate the transistor sizes of a system optimized for Et^n. If the approximate solution is acceptable for the given application, the formula can be used as is (no numerical optimization is then needed); however, if more accuracy is required the formula can be used to provide a good starting point for numerical optimization. Later in the chapter, we propose an efficient iteration procedure that can further improve the accuracy of the original analytical solution. Based on these results, we introduce a novel transistor-sizing algorithm for energy-delay efficient circuits.

Transistor sizing that considers wire parasitics can be studied with the above mentioned classical numerical approaches. More recently, several specialized numerical techniques have been proposed [37, 38, 39]. On the analytical side, Cong and Koh have studied the related problem of simultaneous gate and wire optimization for optimal delay and power [41]. Cong and Koh's solution space and optimization metric are different from what we shall see in the present chapter. A different analytic approach to the transistor sizing problem, for the performance metric Et, is given by Hu [40] and another by Horowitz, Indermaur, and Gonzalez [5]. Both Hu and Horowitz *et al.* present qualitative results; they only analyze basic inverter gates. To the best of our knowledge, the present work is the first one that goes beyond such a qualitative approach, both in terms of the generality of the optimization metric and in terms of the generality of the considered circuits. A subset of the results to be presented here have been published in [43, 44, 52].

To give a concrete example of the transistor-sizing problem, let us consider a five-stage ring oscillator, as shown in Figure 4.1. The delay of this circuit is assumed to be the delay between two consecutive falling or rising transitions of a given node, i.e., using Equations 2.16, 2.17 and 2.20 of the model introduced in Chapter 2, this delay

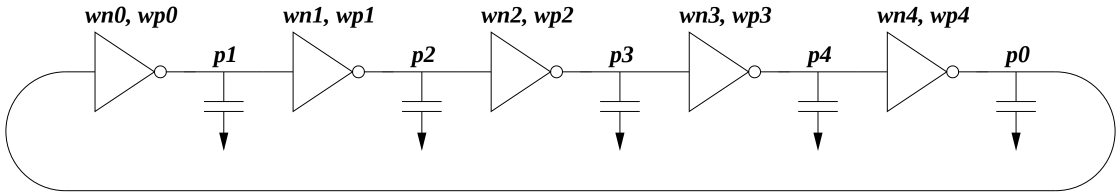

Figure 4.1: Simple example of energy-delay sizing problem.

is

$$
\begin{aligned}
t &= t_{f0}+t_{r0}+t_{f1}+t_{r1}+t_{f2}+t_{r2}+t_{f3}+t_{r3}+t_{f4}+t_{r4} \\
&= \Big(\frac{w_{n1}+w_{p1}+p_1}{w_{n0}}+\frac{w_{n1}+w_{p1}+p_1}{\frac{w_{p0}}{\mu}}+\frac{w_{n2}+w_{p2}+p_2}{w_{n1}}+\frac{w_{n2}+w_{p2}+p_2}{\frac{w_{p1}}{\mu}} \\
&+ \frac{w_{n3}+w_{p3}+p_3}{w_{n2}}+\frac{w_{n3}+w_{p3}+p_3}{\frac{w_{p2}}{\mu}}+\frac{w_{n4}+w_{p4}+p_4}{w_{n3}}+\frac{w_{n4}+w_{p4}+p_4}{\frac{w_{p3}}{\mu}} \\
&+ \frac{w_{n0}+w_{p0}+p_0}{w_{n4}}+\frac{w_{n0}+w_{p0}+p_0}{\frac{w_{p4}}{\mu}}\Big).
\end{aligned}
$$

The energy consumption can be written, using Equation 2.18 of the same model, as

$$E = (w_{n0}+w_{p0}+p_0+w_{n1}+w_{p1}+p_1+w_{n2}+w_{p2}+p_2+w_{n3}+w_{p3}+p_3+w_{n4}+w_{p4}+p_4).$$

Let us now assume, for the sake of the argument, that $p_0 = 1$, $p_1 = 4$, $p_2 = 3$, $p_3 = 7$, and $p_4 = 2$.

When optimized for delay only, i.e., $\min Et^n$ with very large n, the circuit is sized with little regard to the wire capacitances p_i [11]. The transistor widths are $\forall i : w_{ni} = \alpha/(1+\sqrt{\mu}), w_{pi} = \alpha\sqrt{\mu}/(1+\sqrt{\mu})$, where α is a constant large enough to wash away the delay contribution of the parasitics p_i. The basic principle behind such a sizing method is that the optimal delay is achieved when the sum of the falling and rising delays of all stages are equal [11]. In our case, this means $t_{f0}+t_{r0}=t_{f1}+t_{r1}=t_{f2}+t_{r2}=t_{f3}+t_{r3}=t_{f4}+t_{r4}=(1+\sqrt{\mu})^2$ and $t_{opt} = 5(1+\sqrt{\mu})^2$, independently of the p_is.

If the circuit is optimized for energy only, i.e., $\min Et^n$ with $n = 0$, the circuit is sized with little regard to the type of logic gates. The transistor widths are $\forall i$: $w_{ni} = w_{min}, w_{pi} = w_{min}$, i.e., minimum transistor widths for all devices. In this case $t_{fi} + t_{ri} = (1+\mu)(2+p_{i+1}/w_{min})$ and $E_{opt} = (\sum p_i + 10w_{min})$. When w_{min} approaches zero, $t_{fi} + t_{ri}$ increases without bound and E_{opt} approaches $\sum p_i$.

Finally, if the circuit is optimized for Et^2, the resulting widths are $w_{n0} = 6.52/(1+\sqrt{\mu})$, $w_{p0} = 6.52\sqrt{\mu}/(1+\sqrt{\mu})$, $w_{n1} = 6.82/(1+\sqrt{\mu})$, $w_{p1} = 6.82\sqrt{\mu}/(1+\sqrt{\mu})$, $w_{n2} = 7.67/(1+\sqrt{\mu})$, $w_{p2} = 7.67\sqrt{\mu}/(1+\sqrt{\mu})$, $w_{n3} = 6.07/(1+\sqrt{\mu})$, $w_{p3} = 6.07\sqrt{\mu}/(1+\sqrt{\mu})$, $w_{n4} = 5.59/(1+\sqrt{\mu})$, and $w_{p4} = 5.59\sqrt{\mu}/(1+\sqrt{\mu})$. The energy consumption is $E_{opt} = 49.67$, and the delay is $t_{opt} = 7.52(1+\sqrt{\mu})^2$, with the corresponding delays at each stage $t_{f0}+t_{r0} = 1.66(1+\sqrt{\mu})^2$, $t_{f1}+t_{r1} = 1.56(1+\sqrt{\mu})^2$, $t_{f2}+t_{r2} = 1.70(1+\sqrt{\mu})^2$, $t_{f3}+t_{r3} = 1.25(1+\sqrt{\mu})^2$, and $t_{f4}+t_{r4} = 1.35(1+\sqrt{\mu})^2$. In this case, the transistor sizes depend both on the parasitics p_i and on the type of the logic gates. Now it becomes clear that neither the equal stage delay assumption nor the minimum transistor widths assignment yield an Et^n-optimal circuit. As a consequence, a new theory is needed to relate the energy and the delay of the optimally sized circuit to the topology of the logic gates, to the wire parasitics and to the optimization index n in the Et^n metric.

4.2 Et^n-Optimal Circuits

We would like to analyze a transistor netlist as a collection of cycles—one of which is the critical cycle. Let t be the cycle time of a critical cycle. After transistor sizing is complete, as stated by Corollary 1 (to be presented later), we know that all minimal cycle times are equal, i.e., all cycles are critical. This is true for any optimally sized circuit in the absence of additional constraints (minimum-size or slew-rate constraints) on transistor sizes. Let E be the energy consumption of a chosen critical cycle. Let us further assume that E is a constant proportion of the total energy consumption; in this case, optimizing the energy E of the critical cycle optimizes the total energy of the circuit, and vice versa.

4.2.1 Properties of Circuits Optimally Sized for Et^n

4.2.1.1 Synchronization Points

In Chapter 2 we have mentioned that the operation of a logic-gate network can be represented by a directed graph. In this context, a *synchronization point* of a logic-gate network is any vertex with more than one incoming edge. Such a vertex has the property that the rising or falling transition it corresponds to will not happen until all corresponding input signals are present. In this sense, a synchronization point effectively waits for the arrival of all input signals of the corresponding pull-up/pull-down of the logic-gate. In this context, we can state the following:

Theorem 1 *In a logic-gate network optimally sized for Et^n, each synchronization point enabled to switch non-vacuously has its input signals arriving simultaneously.*

Proof. By contradiction. Assume that the sizing is Et^n-optimal, but there exists a logic gate that is enabled to switch non-vacuously and has different arrival times for its inputs. By slowing down the faster input signal, through size reduction on the paths containing it, t will not change, since the faster signal was not on the critical path. However, E will decrease resulting in a better Et^n. This is in contradiction with the optimality of Et^n. □

Theorem 1 suggests a practical way to achieve an Et^n optimum: identify all synchronization points with unequal arrival times and slow down the fast transitions by shrinking the corresponding transistors on the noncritical paths. Theorem 1 is only a necessary, not a sufficient, condition for Et^n-optimality. More precisely, the abovementioned method to reach an optimum by slowing down fast transitions does not guarantee that the resulting system is optimal for the n we were optimizing for; instead, it only guarantees the existence of an n for which the resulting system is Et^n-optimal.

We define the *minimal cycle time* to be the cycle time of a cycle in the absence of all delay constraints imposed by the other cycles of the circuit. In other words, the minimal cycle time is the delay the given cycle would have if at each of its synchronization points the other participating signals would arrive early. In this context, we can state the following:

Corollary 1 *In a logic-gate network optimally sized for Et^n, all minimal cycle times are equal.*

Proof. By contradiction. Assume cycle C_1 has a shorter minimal cycle time than all other cycles. Cycle C_1 should have at least one synchronization point in common with one other cycle of the circuit, say C_2. At this synchronization point, the input signal belonging to C_1 arrives earlier than the input signal belonging to C_2. However, this violates the simultaneous input arrival condition, stated by Theorem 1; and as a consequence, it leads to a contradiction. □

Since the cycle time of a circuit is given by the largest minimal cycle time [7], when all minimal cycle times are equal, they are also equal to the cycle time of the circuit, i.e., all cycles are critical.

4.2.1.2 Properties of Transistor Sizes in Et^n-Optimal Circuits

Property 1 *If w_i are the transistor widths that minimize Et^n for a given set of wire parasitics p_i and logic-gate topologies k_i, then αw_i, $\alpha > 0$ are the widths that minimize Et^n for the set of wire parasitics αp_i and operator topologies k_i.*

Proof. Let us recall Equations 2.18 and 2.20 that define the energy E and delay t given w_i, k_i and p_i. The energy E' of an equivalent system with transistor widths αw_i and parasitics αp_i is $E' = \alpha E$, since α factors out in the expression of the energy both in terms of the transistor widths and in terms of the parasitics. Similarly, the delay t' of an equivalent system with transistor widths αw_i and parasitics αp_i is $t' = t$, since α cancels out for each rising and falling delay. Because minimizing Et^n is equivalent to minimizing αEt^n, it follows that if the w_is minimize Et^n, then the αw_is minimize $E't'^n = \alpha Et^n$. □

Property 2 *If w_{ni} are the transistor widths that minimize Et^n for a given set of wire parasitics p_i and logic-gate topologies k_i, then w_{ni} also minimize Et^n for the set of wire parasitics p_i and logic-gate topologies αk_i.*

Proof. Similarly to what we did in the proof of Property 1, let us call E the energy and t the delay of a system that has transistor widths w_i, logic-gate topologies k_i and parasitics p_i. The energy E' of an equivalent system with logic-gate topologies αk_i is $E' = E$, since the logic-gate topologies k_i do not appear directly in the expression of the energy. Similarly, the delay t' of an equivalent system with logic-gate topologies αk_i is $t' = \alpha t$, since α factors out in the expression of each rising and falling delay. Because minimizing Et^n is equivalent to minimizing αEt^n, it follows that if the w_is minimize Et^n then the w_is also minimize $E't'^n = \alpha Et^n$. $\square$

Properties 1 and 2 will be used to motivate and prove some of the results to be presented later on in this chapter.

The next property states that there exists a general relationship, when optimizing for Et^n, between the width of the nMOS transistors and the width of the pMOS transistors of the same operator.

Property 3 *Consider a cycle of a circuit that contains both the rising and the falling transitions of a logic gate i. Assuming that logic gate i has k_{ni} nMOS transistors of width w_{ni} in series, and k_{pi} pMOS transistors of width w_{pi} in series, then when optimized for Et^n:*

$$w_{pi} = w_{ni}\sqrt{\mu\frac{k_{pi}}{k_{ni}}}.$$

Proof. Using Equation 2.18, the consumed energy E of the chosen cycle can be written as

$$E = \sum_{i=0}^{m-1}(w_{ni} + w_{pi} + p_i), \tag{4.1}$$

and using Equation 2.20, the delay of the cycle can be written as

$$\begin{aligned} t &= \sum_{i=0}^{m-1}\frac{k_{ni}f_{i+1}(w_{n(i+1)} + w_{p(i+1)} + p_{i+1})}{w_{ni}} \\ &+ \sum_{i=0}^{m-1}\frac{\mu k_{pi}f_{i+1}(w_{n(i+1)} + w_{p(i+1)} + p_{i+1})}{w_{pi}}, \end{aligned} \tag{4.2}$$

where $p_{i+1} > 0$ represents the wire parasitics at the output of logic gate i; $f_{i+1} > 0$ is the fanout of logic gate i; μ is the ratio of electron mobility to hole mobility; m is the length of the cycle, and $i \in 0..m-1$ with all indices modulo m.

The optimal solution is reached when the partial derivatives of Et^n in terms of w_{ni} and w_{pi} are zero, i.e.,

$$\frac{\partial Et^n}{\partial w_{ni}} = \frac{\partial Et^n}{\partial w_{pi}} = 0.$$

It follows that

$$\frac{f_i k_{n(i-1)}}{w_{n(i-1)}} + \frac{\mu f_i k_{p(i-1)}}{w_{p(i-1)}} - \frac{k_{ni} f_{i+1}(w_{n(i+1)} + w_{p(i+1)} + p_{i+1})}{w_{ni}^2} = -\frac{t}{nE} \tag{4.3}$$

$$\frac{f_i k_{n(i-1)}}{w_{n(i-1)}} + \frac{\mu f_i k_{p(i-1)}}{w_{p(i-1)}} - \frac{\mu k_{pi} f_{i+1}(w_{n(i+1)} + w_{p(i+1)} + p_{i+1})}{w_{pi}^2} = -\frac{t}{nE} \tag{4.4}$$

Equalizing the left-hand side of (4.3) and (4.4), we get the relationship between the pMOS transistor and nMOS transistors of a given operator:

$$w_{pi} = w_{ni}\sqrt{\mu \frac{k_{pi}}{k_{ni}}}. \qquad \square \tag{4.5}$$

Equation 4.5 is a local relationship; it does not depend on either E or t, and is independent of n—the optimization index. In other words, the ratio of pMOS transistor to nMOS transistor is independent of the exact value of n: it is the same for the entire class of energy-delay optimizations considered here.

There exist circuits with cycles that do not contain both the rising and the falling transition of their component logic gates. However, even for any of these gates there exists some other cycle in the QDI circuit that contains their rising and falling transition. In consequence, even for logic gates on cycles that Property 3 does not directly apply to, Equation 4.5 constitutes a good approximation.

Property 3 eases the way to a transistor sizing abstraction. One important consequence of Property 3 is that the number of free variables in the search space for optimum transistor sizing could be reduced roughly by half, by eliminating the free variables corresponding to either the nMOS transistors or the pMOS transistors. In particular, by eliminating the pMOS transistors from Equation 4.2 we get

$$\begin{aligned} t &= \sum_{i=0}^{m-1} f_{i+1}\left(\frac{k_{ni}}{w_{ni}} + \frac{\mu k_{pi}}{w_{pi}}\right)(w_{n(i+1)} + w_{p(i+1)} + p_{i+1}) \\ &= \sum_{i=0}^{m-1} f_{i+1}\frac{k_{ni}}{w_{ni}}\left(1+\sqrt{\mu\frac{k_{pi}}{k_{ni}}}\right)\left(w_{n(i+1)}\left(1+\sqrt{\mu\frac{k_{p(i+1)}}{k_{n(i+1)}}}\right)+p_{i+1}\right). \end{aligned} \tag{4.6}$$

Let us define

$$w_i = w_{ni} + w_{pi} = w_{ni}\left(1+\sqrt{\frac{\mu k_{pi}}{k_{ni}}}\right) \tag{4.7}$$

and

$$k_i = f_{i+1}k_{ni}\left(1+\sqrt{\frac{\mu k_{pi}}{k_{ni}}}\right)^2 \tag{4.8}$$

Equation 4.6 becomes

$$t = \sum_{i=0}^{m-1} k_i \frac{w_{i+1}+p_{i+1}}{w_i}. \tag{4.9}$$

Similarly, the expression of the energy consumption given by Equation 4.1 simplifies to

$$E = \sum_{i=0}^{m-1}(w_i + p_i). \tag{4.10}$$

For the rest of this chapter we will use these simpler formulas of E and t in the expression for Et^n.

4.2.2 Preliminaries for Et^n-Optimal Transistor Sizing

Based on Equations 4.9 and 4.10, we formalize the sizing problem of a transistor netlist for minimal Et^n as the minimization, over the w_is, of

$$f : {R_+}^m \to R_+,\ \ f(w_0, w_1, ..., w_{m-1}) = \left(\sum_{i=0}^{m-1}(w_i+p_i)\right)\left(\sum_{i=0}^{m-1} k_{i-1}\frac{w_i+p_i}{w_{i-1}}\right)^n, \tag{4.11}$$

where $n \geq 0$, $p_i > 0$, $k_i > 0$ with $i \in 0..m-1$, $m \in N$, and all indices modulo m.

Equation 4.11 holds not only for a ring, but also for a chain of operators, as long as the parameters for the input of the chain are equal to the parameters for the output of the chain (since in this case the E and t for a chain have the same form as the ones for a ring). This is an important observation, as it makes our results for transistor sizing applicable to circuit delays both in terms of latency and cycle time. Whenever we use latency as the measure of delay, we make the assumption that the scrutinized component has its input "drive" equal to its output "drive" (i.e., no amplification). This is a reasonable assumption since most logic-gate chains are part of closed ring topologies.

We can show that the unique optimum of f is achieved when

$$\forall i : \frac{\partial f}{\partial w_i} = 0,$$

or equivalently

$$\forall i : \frac{k_{i-1}}{w_{i-1}} - \frac{k_i(w_{i+1} + p_{i+1})}{w_i^2} = -\frac{1}{n}\frac{\sum_{i=0}^{m-1} k_{i-1}\frac{w_i+p_i}{w_{i-1}}}{\sum_{i=0}^{m-1}(w_i + p_i)} = -\frac{1}{n}\frac{1}{P}, \tag{4.12}$$

where $P = E/t$ is the power consumption of the chosen cycle. If $\forall i : p_i = 0$ (no wire parasitics) and n very large (delay-only optimization), Equation 4.12 reduces to

$$k_i \frac{w_{i+1}}{w_i} = k_{i-1}\frac{w_i}{w_{i-1}},$$

which is the known condition of equal stage delays for delay-only transistor sizing [56]. If we were able to solve Equation 4.12 analytically for any p_is and k_is, we could compute the optimal w_is directly and our transistor-sizing problem would be solved. Unfortunately, this is not the case. We can compute an exact analytical solution of Equation 4.12 only for a restricted class of p_is and k_is. Using Equation 4.12, simple algebra shows that if all p_is satisfy

$$p_i = \left(1 + \frac{1}{n}\right)k_{i-1} - k_i, \tag{4.13}$$

or if all k_is satisfy

$$k_i = \frac{\sum_{j=i}^{0} \left(1+\frac{1}{n}\right)^j p_{i-j} + \sum_{j=m-1}^{i+1} \left(1+\frac{1}{n}\right)^j p_{m+i-j}}{\left(1+\frac{1}{n}\right)^m - 1}, \tag{4.14}$$

then

$$\forall i : w_i = k_i. \tag{4.15}$$

We can also express Equations 4.13 and 4.14 using an $m \times m$ transformation matrix M

$$M = \begin{pmatrix} -1 & 0 & 0 & \dots & 1+\frac{1}{n} \\ 1+\frac{1}{n} & -1 & 0 & \dots & 0 \\ 0 & 1+\frac{1}{n} & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & -1 \end{pmatrix}.$$

Equation 4.13 states that given a set of k_is, one can always pick—using the transformation matrix M—a set of p_is given by

$$(p_0, p_1, \dots, p_{m-1})^T = M(k_0, k_1, \dots, k_{m-1})^T \tag{4.16}$$

(^{T} is the matrix transposition operator) such that Equation 4.15 holds. Similarly, Equation 4.14 states that given a set of p_is, one can always pick—using M^{-1}, the inverse of the transformation matrix M—a set of k_is given by

$$(k_0, k_1, \dots, k_{m-1})^T = M^{-1}(p_0, p_1, \dots, p_{m-1})^T \tag{4.17}$$

such that Equation 4.15 holds. Equations 4.13 and 4.14, or equivalently Equations 4.16 and 4.17, yield two classes of analytically computable values of the w_is. These analytically computable values will play an important role in motivating the validity of our approximate solution of the Et^n-optimal w_is.

In the particular case of $\forall i : k_i = k$, i.e., the case of homogeneous circuits, using Equation 4.13, we get that if $\forall i : p_i = k/n$ then $\forall i : w_i = k$, or equivalently, using

Property 1, if $\forall i : p_i = p$ then

$$\forall i : w_i = np, \ \ \forall\, p, k > 0. \tag{4.18}$$

Equation 4.18 states that the transistor widths w_i of a homogeneous circuit with equal wire parasitics p, optimized for Et^n, are all equal to np, independently of k [43, 44].

4.2.3 Energy-Delay Complexity of Et^n-Optimal Circuits

In this section we establish two global properties of asynchronous circuits optimized for Et^n. First, the consumed energy (E_n) is independent, in first approximation, of the types of logic gates (NAND, NOR, C-element, etc.) used by the circuit and is solely dependent on the optimization index n and the total amount of wiring capacitance switched during computation. Second, the circuit speed (t_n) is independent, in first approximation, of the wire capacitance and depends only on the optimization index n and the types of logic gates used. These properties allow an abstract view on transistor sizing by shifting the design emphasis to the logical level of circuits. As such, they constitute the basis of our energy-delay complexity definition.

Let us define the *theoretical minimal energy* E_0 to correspond to minimizing E without regard for t; in other words, to correspond to the situation when the transistors are all zero-sized and the fixed parasitic capacitances constitute the entire E. Conversely, let us define the *theoretical minimal delay* t_∞ to correspond to minimizing t without regard for E. This delay is obtained when the transistor sizes go to infinity, i.e., when only gate capacitances contribute to E and t. Using these definitions, we can state the following theorem:

Theorem 2 *For a neighborhood $\mathcal{V}_p = [p-\eta, p+\eta]$ of $p > 0, \eta > 0$ and a neighborhood $\mathcal{V}_k = [k-\eta, k+\eta]$ of $k > 0$ where $\forall i : p_i \in \mathcal{V}_p, k_i \in \mathcal{V}_k$ and η approaches zero, the energy E_n and delay t_n of a circuit optimized for Et^n are given by*

$$E_n = (1+n)E_0, \tag{4.19}$$

$$t_n = \left(1 + \frac{1}{n}\right) t_\infty \tag{4.20}$$

where E_0 is the theoretical minimal energy (i.e., the energy due to the total switched wire capacitance) and t_∞ is theoretical minimal delay.

Proof. Let us compute the energy E_n and delay t_n for the circuit with parameters $k_i = k$ and $p_i = p$. We know from Equation 4.18 that the w_is that achieve the optimum in this case are $w_i = np$. As a consequence—using Equation 4.10—we can write

$$E_n = \sum_{i=0}^{m-1} w_i + \sum_{i=0}^{m-1} p_i = n \sum_{i=0}^{m-1} p_i + \sum_{i=0}^{m-1} p_i = (1+n) \sum_{i=0}^{m-1} p_i = (1+n)E_0.$$

Similarly, using Equation 4.9 we can write

$$\begin{aligned} t_n &= \sum_{i=0}^{m-1} k_i \frac{w_{i+1} + p_{i+1}}{w_i} = \sum_{i=0}^{m-1} k_i \frac{np_{i+1} + p_{i+1}}{np_i} \\ &= \left(1 + \frac{1}{n}\right) \sum_{i=0}^{m-1} k_i = \left(1 + \frac{1}{n}\right) m \sqrt[m]{\prod_{i=0}^{m-1} k_i} = \left(1 + \frac{1}{n}\right) t_\infty. \end{aligned}$$

Since $\forall i : \lim_{\eta \to 0} p_i = p$ and $\forall i : \lim_{\eta \to 0} k_i = k$, the values of E_n and t_n we just computed correspond to the energy and delay stated by the theorem. □

Theorem 2 states that when all p_is are close to each other and all k_is are close to each other as well, the optimal energy and delay of an Et^n-optimal circuit is given by Equations 4.19 and 4.20. While Equations 4.19 and 4.20 have been derived for only a restricted class of circuits, they are in fact good approximations for a much wider class. In the next section, we will present experimental results for this claim.

If we now revisit the example presented in the introduction of this chapter, using Equations 4.19 and 4.20, we find that the circuit optimized for Et^n consumes energy $E_n = 51$ and operates at delay $t_n = 7.5(1 + \sqrt{\mu})^2$. Both of these values are close approximations of the optimal values $E_{opt} = 49.67$ and $t_{opt} = 7.52(1 + \sqrt{\mu})^2$.

Based on Equation 4.19, the consumed energy is independent of the types of logic gates used by the circuit and is solely dependent on the optimization index and the total amount of wiring capacitance switched during computation. On the other hand, based on Equation 4.20 the circuit speed is independent of the parasitics and depends only on the optimization index and the types of logic gates used.

If n goes to infinity in Equation 4.20, then $t = t_\infty$—an expected result in case of delay-only optimization. On the other hand, if $n = 0$ in Equation 4.19 then $E = E_0$, i.e., transistors should be sized as small as possible for minimum energy consumption—another expected result. For voltage-independent energy efficiency ($n = 2$) the cycle time of the circuit should be chosen as $\frac{3}{2}t_\infty$; while the minimum energy to achieve this delay will be $3E_0$—the loss in speed is more than compensated by the energy savings.

Equations 4.19 and 4.20 provide an elegant way to analyze E and t independently at circuit level, while optimizing Et^n. In particular, the speed of an Et^n optimal system can be directly derived from the theoretical minimal delay t_∞ of the same system. Similarly, the energy consumption of the system can be directly derived from the total energy due to the switched wire capacitance E_0.

Corollary 2 *For a neighborhood $\mathcal{V}_p = [p-\eta, p+\eta]$ of $p > 0, \eta > 0$ and a neighborhood $\mathcal{V}_k = [k-\eta, k+\eta]$ of $k > 0$ where $\forall i : p_i \in \mathcal{V}_p, k_i \in \mathcal{V}_k$ and η approaches zero, the total gate capacitance of a circuit optimized for Et^n is n times the total switched wire capacitance.*

Proof. It follows directly from Equation 4.19 of Theorem 2 by recognizing that

$$E_n = \sum_{i=0}^{m-1}(w_i + p_i) = E_0 + \sum_{i=0}^{m-1} w_i = (n+1)E_0 \Rightarrow \sum_{i=0}^{m-1} w_i = nE_0 = n\sum_{i=0}^{m-1} p_i. \qquad \square$$

Corollary 2 states that the total gate capacitance of a circuit is equal to the total wire capacitance times the optimization index. In particular, a circuit optimized for Et^2 will have—on average—transistors twice as big as the same circuit optimized for Et. Corollary 2 also suggests a strong dependence of transistor sizes on wire capacitance, a dependency that is in general ignored in delay-only sizing. For Et^2 optimization, wire capacitance plays a major role and needs to be dealt with explicitly.

Experimental evidence for Corollary 2, for the case $n = 2$, is also provided by SPICE simulations of an adder published by Chandrakasan and Brodersen and summarized in Figure 4.7 of their book [3]. Their figure shows that, for the five different parasitic contributions they study, the minimum energy for a given speed (allowing supply-voltage adjustment) is achieved when the gate capacitance is very close to twice the parasitics. (They did not, however, draw the conclusion that we have reached here.)

Based on the Equations 4.19 and 4.20, we define $E_0 t_\infty^n$ to be the *energy-delay complexity* of a circuit. The optimization index n represents the chosen trade-off between energy and delay. Based on this definition, we can compare the energy-delay efficiency of two circuits implementing the same computation by comparing their energy-delay complexities, as stated by the following theorem:

Theorem 3 *Given two circuits A and B implementing the same computation, circuit A is more energy-delay efficient than circuit B—in terms of Et^n—if and only if*

$$E_{0A} t_{\infty A}^n < E_{0B} t_{\infty B}^n, \tag{4.21}$$

where E_{0A}, E_{0B} are the theoretical minimal energies and $t_{\infty A}$, $t_{\infty B}$ are the theoretical minimal delays of circuit A and circuit B, respectively.

Proof. Follows directly from Theorem 2. □

The concept of energy-delay complexity together with Theorem 3 allow an abstract view of the efficiency of a circuit—through the two figures of merit E_0 and t_∞—and facilitate a high level comparison of circuit choices. One can attach a circuit-level cost—in terms of E_0 and t_∞—to basic constructs of a high level specification language such as the CHP language [30]. Hence, the circuit level cost of any CHP program can be evaluated for energy-delay efficiency without needing to consider the circuit-level details of the given program.

Based on Theorem 2, we can formally show that optimizing Et^n for large n is equivalent to optimizing t. First, we notice that minimizing a circuit for Et^n is equivalent to minimizing the circuit for $\sqrt[n]{E}t$. Thus, we want to show that

$$\lim_{n\to\infty} \min(\sqrt[n]{E}t) = \min t.$$

Using Equations 4.19 and 4.20, we get

$$\lim_{n\to\infty} \min(\sqrt[n]{E}t) = \lim_{n\to\infty} \sqrt[n]{(1+n)E_0}\Big(1+\frac{1}{n}\Big)t_\infty = t_\infty = \min t.$$

4.2.4 Experimental Evidence on the Energy and Delay of Et^n-Optimal Circuits

As mentioned before, even though Equations 4.19 and 4.20 have been derived for only a very restricted class of circuits, they are in fact good approximations for a much wider class. We have checked the equations against the minimal Et^n obtained by applying an optimization algorithm (gradient descent) to two classes of circuits. In the first class, each circuit consisted of a ring of operators that were chosen at random with a uniform-squared distribution of parasitic capacitances; the number of transistors in series was also chosen according to such a distribution. We used real numbers for both parameters; we optimized the expression for Et^n using Equations 4.11. The range of parasitics was [1,100] in normalized units; the range of transistors in series was [1,6]. The results show that Equations 4.19 and 4.20 hold, with good accuracy, over a wide range of parasitics, logic-gate types, and circuit sizes.

The results of the simulations for circuits consisting of a ring of 100 operators are summarized in Figure 4.2. (Simulations for rings of 10 and 1000 operators show similar results.) The figure shows the mean and standard deviation of the error in the estimates of Equations 4.19 and 4.20 for a range of different optimization indices ($n \in [1, 10]$). The estimates get more dependable for larger circuits, where the random variation in operators tends to average out over the cycle. Overall, the estimates are usually within 5% of the energy and within 2% of the delay values for the actual optimum Et^n.

The second class of circuits consisted of a closed chain of connected rings of operators with parasitic capacitances and number of transistors in series chosen the same way as in the previous experiment. These results show, as well, that Equations 4.19 and 4.20 hold, with very good accuracy, over a wide range of parasitics, logic-gate

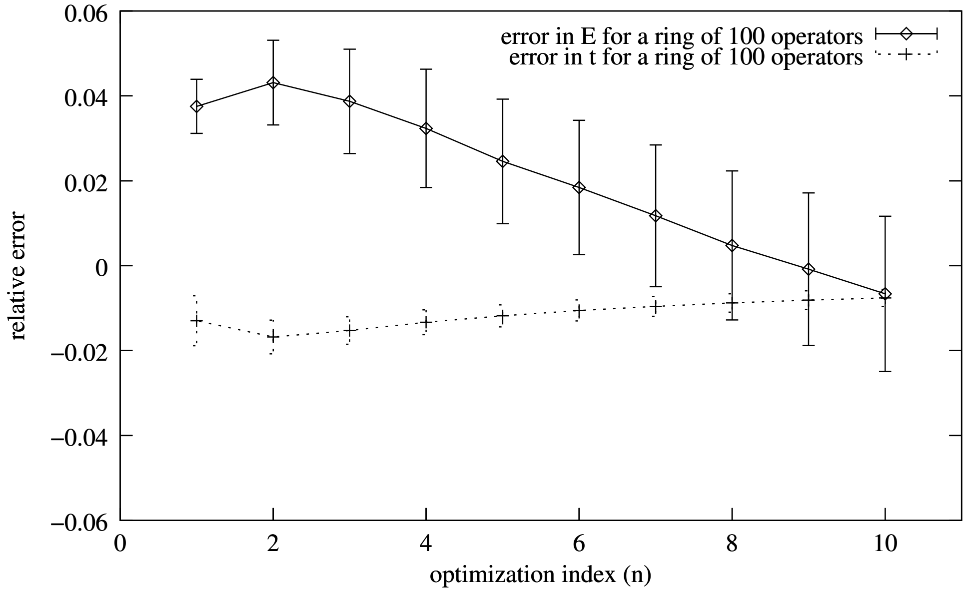

Figure 4.2: Results of simulating a ring of logic gates with gate topologies and parasitics chosen randomly.

types, circuit sizes and circuit topologies.

The results of the simulations for circuits consisting of a chain of 10 rings of 10 operators each are summarized in Figure 4.3. (Simulations for chains of 2, 4, 8, 12 and 20 rings of 4, 20 and 100 operators show similar results.) The figure shows the mean and standard deviation of the error in the estimates of Equations 4.19 and 4.20 for a range of different optimization indices ($n \in [1, 10]$). Overall, the estimates are usually good to within 8% of the energy and within 5% of the delay values for the actual optimum Et^n.

So far we have seen evidence that, on average, Equations 4.19 and 4.20 approximate the E and t of an Et^n optimal circuit well. Ideally, we would like to characterize more precisely the error between the estimated energy-delay performance and the actual minimal Et^n. We would like to bound analytically the error in E, t and Et^n. Unfortunately, this seems too much to ask. We have seen that the transistor sizes of the optimal solution are described by Equation 4.12. We are not aware of a general analytical solution to this equation. We were able to compute the solution exactly, through Equations 4.13 and 4.14, only for a very restricted class of parameters p_i and

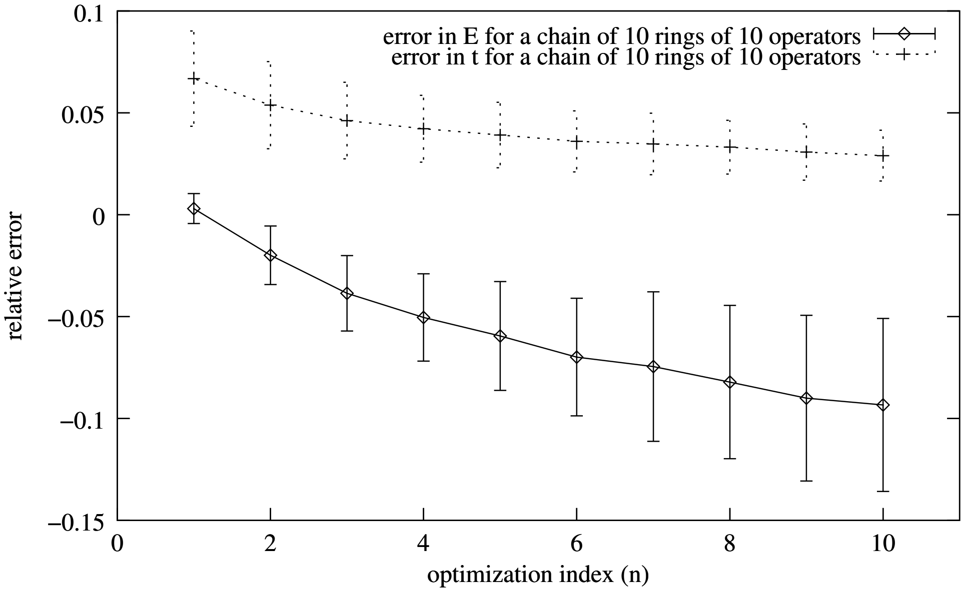

Figure 4.3: Results of simulating a chain of rings of logic-gates with gate topologies and parasitics chosen randomly.

k_i. Hence, we have to rely on numerical optimization to characterize the absolute error in formulas 4.19 and 4.20. For this, we consider a relevant case of the transistor sizing problem and, using numerical optimization, we exhaustively evaluate the errors for each circuit within the bounds imposed on p_is and k_is. Given the fact that the number of circuits to be evaluated grows very quickly with the problem size m and the number of different circuit parameters p_i and k_i, we have to limit significantly the size of the class of problems we can consider. As a consequence, we analyze a particular case of Equation 4.11 with $n = 2$, $m = 5$, $p_i \in \{1, 2, 3, 4, 5\}$ and $k_i \in \{1, 2, 3\}$. For this particular class of problems, there are $15^5 = 759375$ circuits to consider. Figures 4.4 and 4.5 show a histogram of the relative error in E, t and Et^2 between the optimal values (computed using an optimization algorithm) and the estimated values (computed using Equations 4.19 and 4.20). In general, the spread of all errors increases by increasing the ratio between $\max p_i / \min p_i$ and $\max k_i / \min k_i$. The error range observed in this experiment is particular to the problem we have considered. However, the general form of the error distribution is the same for the entire class of transistor sizing problems. Figure 4.4 shows that the error in t is more clustered

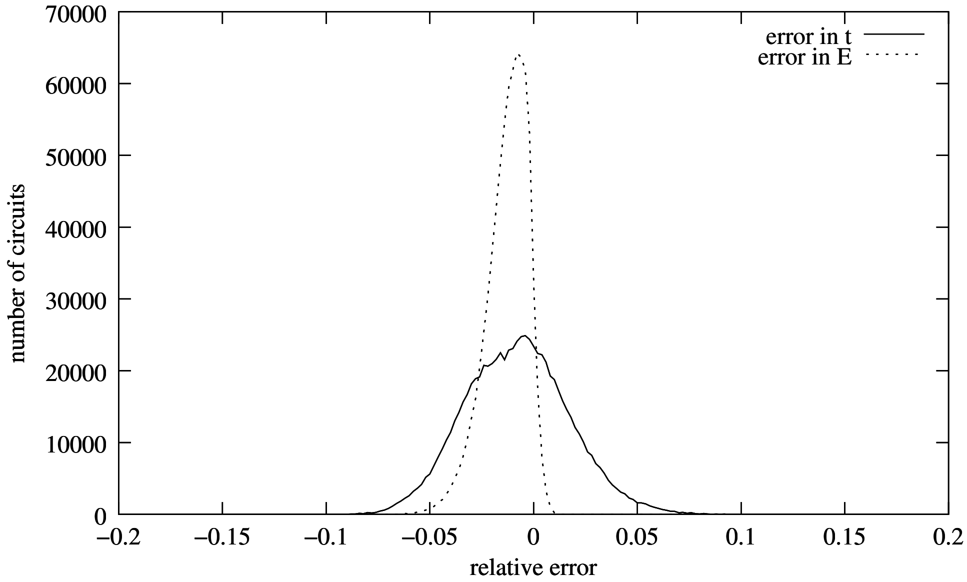

Figure 4.4: Error in E and t when exhaustively simulating a ring of random gates and parasitics.

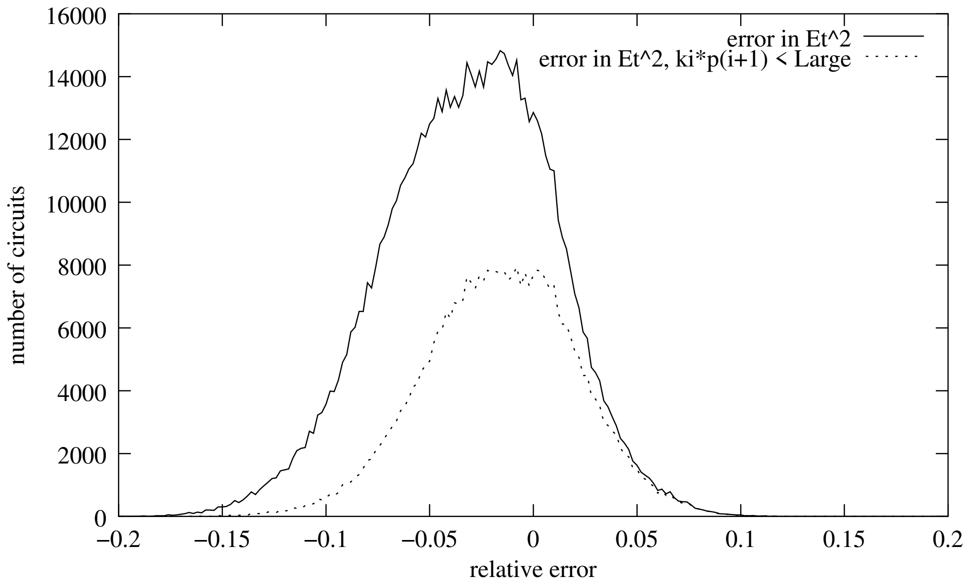

Figure 4.5: Error in Et^2 when exhaustively simulating a ring of random gates and parasitics with and without the constraint $k_i p_{i+1} < Large$.

than the error in E. In other words, the delay estimate is consistently better than the energy estimate. This is particularly beneficial, given the fact that the error in t gets amplified about linearly with n in the Et^n metric, as we observe in Figure 4.5.

In the exhaustive search through all problem instances, we have also considered circuits that would not make good designs in practice, independently of the accuracy of our performance estimate. In particular, circuits with slow gates (large k_i due to several transistors in series and/or fanout) driving long capacitive wires (large p_{i+1}) are not desirable in practice. It turns out that circuits with $k_i p_{i+1} > Large$, where $Large$ is a cut-off constant, typically underestimate the circuit performance. Eliminating such cases from the exhaustive search reduces the error spread, as shown in Figure 4.5. Such a restriction naturally eliminates unpractical circuits whose performance incidentally was estimated well, i.e., some circuits at the top of the histogram. Figure 4.5 also shows a significant shift towards zero of the average error in Et^2, when the impractical cases are eliminated.

4.2.5 The Influence of RC Delay and Intrinsic Gate Delay on Energy-Delay Complexity

The transistor model described in Chapter 2 ignores the RC delay due to wire resistance and intrinsic gate delay. For a fixed-size wire, the corresponding RC propagation delay is independent, to first order, of the transistor sizes. Similarly, the intrinsic delay of a gate, i.e., the delay due to its own internal capacitance, is largely independent of the transistor sizes as well. As a consequence, we can consider that each cycle of a circuit has an added delay t_{fix} that is independent of the transistor sizes. This delay depends only on the wire capacitance, wire resistance and logic gate topology.

We would like to know, qualitatively, how the energy-delay complexity of a circuit changes when a constant delay is added to each transition in the transistor model. For this, we consider a simple example: a ring of identical inverters of width w driving identical parasitics p. Assume that the intrinsic gate delay and the RC delay per inverter and associated output wire is $t_{stage\,fix}$. The delay of the circuit is proportional to $t = \frac{w+p}{w} + t_{stage\,fix}$ and the consumed energy is proportional to $E = w + p$. If we minimize the metric Et^n, we get a transistor width that is $w_{opt} \leq np$, with equality if and only if $t_{stage\,fix} = 0$. Hence, considering t_{fix} to be the total RC and intrinsic gate delay, we have

$$E_n \leq (n+1)E_0$$

and

$$t_n \geq \Big(1 + \frac{1}{n}\Big)t_\infty + t_{fix},$$

with equality if and only if $t_{fix} = 0$. If $t_{fix} \neq 0$, the equalities of Theorem 2 become inequalities.

Remarkably, the optimal Et^n of the circuit is within 0.5% of

$$(n+1)E_0\Big(\Big(1 + \frac{1}{n}\Big)t_\infty + t_{fix}\Big)^n,$$

when t_{fix} is less than 10% of t_∞. Thus, for small t_{fix} we can still use $(n+1)E_0$ as an estimate for the optimal energy for minimal Et^n, but we have to change the delay estimate from $\big(1 + \frac{1}{n}\big)t_\infty$ to $\big(1 + \frac{1}{n}\big)t_\infty + t_{fix}$. As a consequence, when a fix delay is added to delay t, the energy-delay complexity changes from

$$E_0 t_\infty^n \quad \text{to} \quad E_0\Big(t_\infty + \frac{n}{n+1}t_{fix}\Big)^n. \tag{4.22}$$

4.2.6 An Application of Energy-Delay Complexity

In this section we study the energy-delay efficiency of several circuit-level implementation techniques by applying the energy-delay complexity model developed in Section 4.2.3. First, we compute the energy-delay complexity of a generic pipeline stage. We investigate the influence of data encoding and horizontal pipelining on this complexity. Next, we study circuits that are not pipelined and show that their latency and energy consumption are smaller than that of an equivalent pipelined implementation. Lastly, we investigate the energy-delay complexity of interleaved circuits.

4.2.6.1 Data Encoding and Horizontal Pipelining

Asynchronous circuits use delay-insensitive data encodings, i.e., encodings in which data becomes valid only when all bits supposed to switch have done so. One of the most widely used delay-insensitive encodings is the 1ofN encoding. A 1ofN encoding represents log(N) bits. Each rail corresponding to the physical realization of a 1ofN encoding represents a different combination of the log(N) bits. It is often the case, in practice, that a designer has the freedom to decide the data encoding of the circuit being implemented. In such a case, it is useful to know the influence data encoding has on the energy-delay complexity of the circuit. Intuitively, encodings that result in fewer wires being switched tend to be more energy efficient. On the other hand, our measurements show that buffer circuits operating on dual-rail data are faster than buffer circuits operating on quad-rail data. As a consequence, it is not immediately clear how data encoding influences the energy-delay complexity of a circuit.

Horizontal pipelining is a technique used to speed-up the control distribution of asynchronous circuits. Horizontal pipelining is achieved by grouping the datapath into a set of unsynchronized circuit blocks operating in parallel (for more details please see [33]). Similarly to data encoding, the amount of horizontal pipelining, i.e., the number of bits grouped together, creates an energy-delay trade-off in the design of a pipeline stage. Larger horizontal pipeline slices share more circuitry, and as a consequence, consume less energy. On the other hand, shared circuitry adds additional synchronization to a pipeline stage, and as a consequence, the stage operates slower.

We will analyze a generic 32-bit pipeline stage. We compute the energy-delay complexity of the circuit, as a function of its data encoding and amount of horizontal pipelining. We investigate a generic pipeline stage that receives one bit of control on input-channel C and 32 bits of data on input-channel L and produces 32 bits of data on output-channel R. This circuit is described by the following high-level specification (for a detailed description of the CHP language please see [30]):

$$
\begin{aligned}
generic - stage \equiv *[&C?c;\\
&[c = 0 \longrightarrow R!f_0(L?)\\
&\,[\!]\, c = 1 \longrightarrow R!f_1(L?)\\
&]],
\end{aligned}
$$

where f_0 and f_1 are bit functions. The environment of a circuit could have a significant influence on the energy consumption and delay. For this reason, we analyze circuit *generic-stage* in a typical environment. We assume that channel L comes from an identical *generic-stage* circuit, while channel R goes to an identical *generic-stage* circuit. We limit our analysis to the two most widely used data encodings: dual-rail (1of2) and quad-rail (1of4).

We compare the energy complexities of the different circuit alternatives using Theorem 3. As a prerequisite, we need to compute the theoretical minimal energy E_0 and the theoretical minimal delay t_∞ of each alternative. We compute E_0 using `esim`, a simulator that estimates, among other things, the total switched wire capacitance of a circuit [50]. The Appendix describes this simulator in detail. For our current example, we assume a simple wiring scheme in which each circuit node has wire capacitance proportional to its fanout. We separately compute, using the *logical-effort* model [11], the minimal cycle time of the control distribution and the minimal cycle time of the circuit itself, and then, we assign the maximum of the two to the system cycle time t_∞.

Figure 4.6 shows the minimal energy E_0 and minimal cycle time t_∞ as a function of the data encoding and amount of horizontal pipelining. Figure 4.6 shows that, for the same amount of horizontal pipelining, circuits using quad-rail encoding consume less energy. The speed of a circuit using small groups of bits in the horizontal pipelining is limited by the speed of the control distribution, both for dual-rail and quad-rail implementations. On the other hand, for slices of 4 bits or more, the control distribution cycle is not critical anymore. In this case, for the same amount of horizontal pipelining, the quad-rail (1of4) implementation is slightly faster than the dual-rail (1of2) implementation. In conclusion, under the setup of our experiment, for the

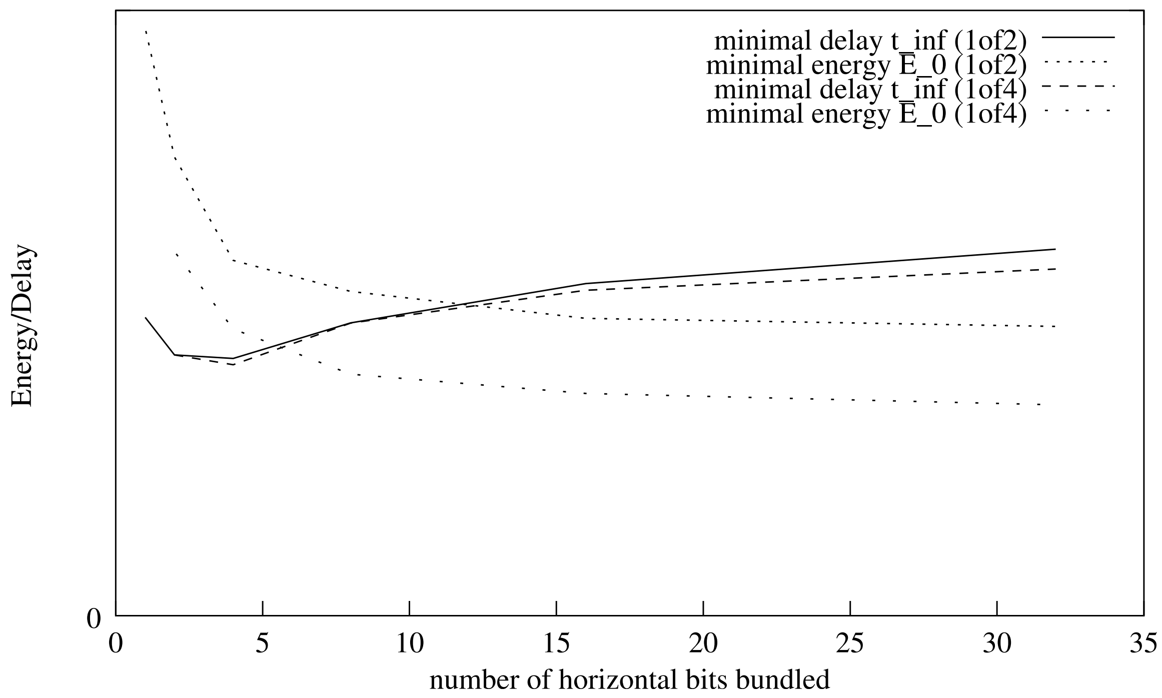

Figure 4.6: E_0 and t_∞ for a generic pipeline stage for dual-rail and quad-rail encoding.

same amount of horizontal pipelining, the energy-delay complexity of a quad-rail implementation is always better than that of a dual-rail implementation, independently of the optimization index n.

Figure 4.7 shows the energy-delay complexity of process *generic-stage* for $n = 2$. Figure 4.7 shows that the optimal amount of horizontal pipelining is achieved for bundles of 4 bits, both for dual-rail and quad-rail implementations. As a consequence, we conclude that the optimal Et^2 is achieved for horizontal pipelining in bundles of 4 bits.

The logic gate network we used so far in our implementation did not share any transistors. We would like to quantify the influence of transistor sharing on energy-delay complexity. To do this, we recompute E_0 and t_∞ for process *generic-stage* implemented quad-rail, in bundles of 4 bits, using maximal transistor sharing. As expected, transistor sharing improves both E_0 and t_∞. The smaller number of transistors reduces the fanout in the expression of t_∞, improving it by about 10%. The smaller number of wires decreases E_0 by about 25%. The total improvement in Et^2

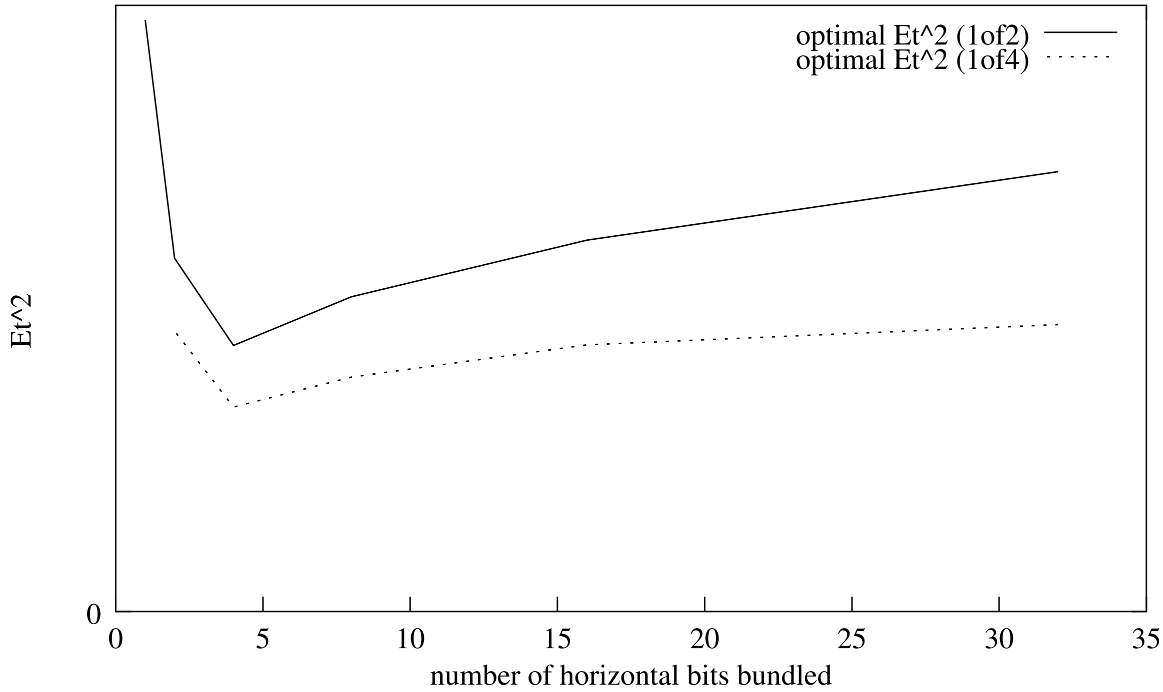

Figure 4.7: Et^2 of a generic pipeline stage for dual-rail and quad-rail encoding.

is about 40%.

4.2.6.2 Unpipelined circuits

An unpipelined circuit can be obtained from a sequence of pipeline stages by eliminating the internal communication channels. In practice, this amounts to eliminating internal output completions and internal right enable generations, while keeping the precharge logic structure. A generic unpipelined circuit is shown in Figure 4.8. The operation of this circuit is as follows. In the set phase, when input l arrives and the enable signal en is high, the first pull-down fires and a data value is produced and passed on to the next pull-down stage. In parallel, the enable signal and the input validity are passed on to the next stage, as well. The enable signal en is buffered in order to reduce its fanout and to balance out the signal arrival delay at the precharge gates. Eventually, the computed data, together with the completion signals, propagate to the last logic stage. This last stage behaves like a standard pipeline stage: it

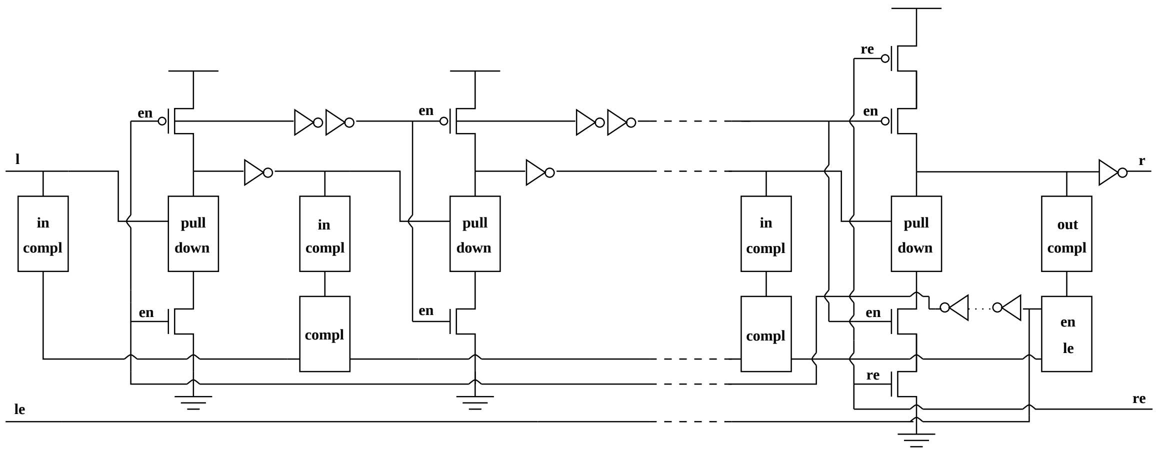

Figure 4.8: Generic unpipelined circuit.

produces an output when the environment is ready, and it generates the enable signal *en* and the left enable signal *le*. In the reset phase, the input neutrality, together with the neutrality of the internal signals, propagates to the last stage which then re-enables the circuit for a new computation cycle.

We notice that, since the circuits implementing the internal handshakes were eliminated, all precharge logic stages but the last, have one fewer transistor in series. Furthermore, since all output completions but the last, were eliminated, the fanout of the precharge signals is reduced (asynchronous circuits use output completion to check the validity and neutrality of the data channels connecting circuit blocks). These two effects reduce the latency of an unpipelined circuit, when compared to an equivalent pipelined implementation. We can quantify this reduction using the logical effort model. We consider the previously mentioned *generic-stage* process as a generic example. The ratio between the pipelined and unpipelined latencies of a k-logic-stage circuit is $r^{\frac{1}{2}(1-\frac{1}{k})}$, where r is a constant that depends on the data encoding and can be compute, for example, using the logical effort model. Numerically, $r = 2.59$ for dual-rail and $r = 3.19$ for quad-rail. In the limit when k goes to infinity, the latency ratio becomes $\sqrt{r}$. In other words, the limit on the latency speed-up due to unpipelining is $1 - 1/\sqrt{r}$: 38% for 1of2 encoding and 44% for 1of4 encoding.

Because an unpipelined circuit has fewer transistors than its pipelined counterpart,

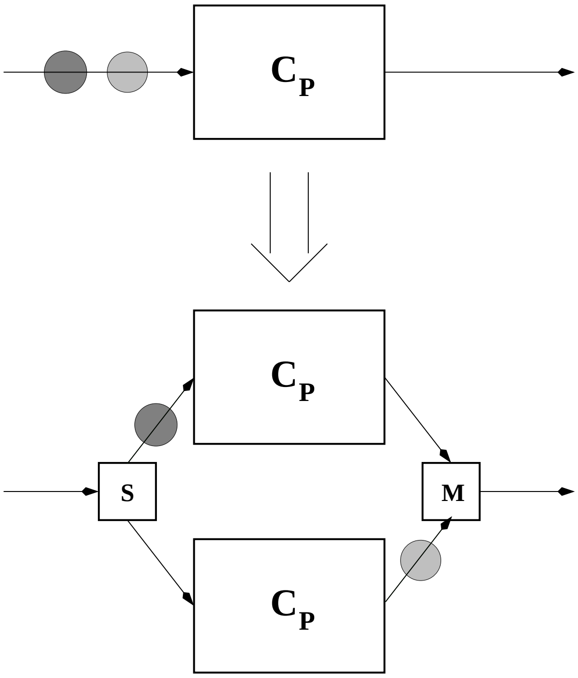

Figure 4.9: Generic interleaved circuit.

its minimal energy E_0 is reduced. The ratio of the minimal energy of a pipelined to that of an unpipelined implementation is $\frac{k}{0.87k+0.13}$ for 1of2 encoding, and $\frac{k}{0.81k+0.19}$ for 1of4 encoding, independently of the amount of horizontal pipelining. In the limit when k goes to infinity, the minimal energy E_0 is reduced by about 13% for 1of2 encoding and by about 19% for 1of4 encoding.

Considering the delay of the circuit to be its latency, on the limit, the Et^2 improvement is about 66% for 1of2 encoding and about 75% for 1of4 encoding.

4.2.6.3 Interleaved pipeline stages

Consider a program P implemented by circuit C_P. An interleaved implementation of P can be obtained by connecting in parallel two identical circuits C_P through a split circuit S and a merge circuit M, as shown in Figure 4.9. The split and merge circuits alternately communicate with the C_P circuits, i.e., one data element is processed by the top C_P circuit while the next data element is processed by the bottom C_P circuit. Given such an operation, the throughput of the interleaved C_P circuit can be reduced to half the throughput of the split and merge circuits, without affecting the overall throughput of the system. The reduced throughput allows a more energy-efficient implementation of C_P. However, all this comes with added energy and latency costs due to the split and merge cells. As a result, it is not immediately clear if interleaved circuits are more energy-delay efficient than non-interleaved circuit implementations.

We quantify the abovementioned trade-off using the generic process *generic-stage*. We choose the implementation of *generic-stage* that has a throughput about half that of the interleaving units. This ensures that the *generic-stage* processes do not operate faster than they have to. Under this setup, we can compare the energy-delay complexity of the interleaved circuit against a standard pipelined implementation. The ratio between the two complexities, i.e., $(E_{0\,interleaved} t^2_{\infty\,interleaved})/(E_{0\,pipelined} t^2_{\infty\,pipelined})$, is 1.34 for $k = 1$, 0.93 for $k = 2$, 0.79 for $k = 3$, 0.72 for $k = 4$, 0.68 for $k = 5$, 0.65 for $k = 6$, and 0.63 for $k = 7$. These results show that, with the exception of a single pipeline stage ($k = 1$), the interleaving results in a better Et^2. In the limit when k goes to infinity, the Et^2 of the interleaved system is 48% better than the equivalent pipelined version.

The energy-delay efficiency of a system using interleaved elements can be further increased by replacing the pipelined C_P circuits by equivalent unpipelined implementations. In this case, $(E_{0\,interleaved} t^2_{\infty\,interleaved})/(E_{0\,pipelined} t^2_{\infty\,pipelined})$ is 1.22 for $k = 1$, 0.84 for $k = 2$, 0.67 for $k = 3$, 0.58 for $k = 4$, 0.55 for $k = 5$, 0.52 for $k = 6$, and 0.49 for $k = 7$. These results shows that the energy-delay efficiency is even higher when the interleaved circuits are unpipelined.

It should be noted that the energy cost of the interleaving circuits is relatively high. It is comparable to the energy cost of a regular pipeline stage. As a result, the energy-delay improvement due to interleaving is significantly smaller than the theoretical improvement resulting from zero cost split/merge devices [1]. Moreover, to keep up any energy-delay improvement given by the interleaving technique, one has to be able to run the entire system at the increased speed of the split/merge circuits. Furthermore, the system needs to tolerate the latency added by these circuits. These requirements might be difficult to meet in a system that has feedback in its operation, such as a microprocessor. On the other hand, they should be easy to achieve in a system with feed-forward operation, such as a DSP.

4.2.7 Et^n-Optimal Transistor Sizes

So far we have analyzed the total energy and delay of an Et^n-optimal system, without saying much about the actual transistor sizes that achieve this optimum. One can obviously find these transistor sizes by numerically optimizing Et^n with respect to the transistor sizes. In this subsection we develop a set of simple analytical formulas that approximates the transistor sizes of a system optimized for Et^n. Such a formula has two main uses: first, if a rough estimate is enough for the given application, the formula can be used as is (no numerical optimization is then needed); second, if accuracy is required, the formula can provide a good starting point for numerical optimization.

We start by proposing an approximate formula for the transistor sizes w_i that optimize f, in Equation 4.11, when $\forall i : k_i = k$. We then extend this formula to the case when the k_is are not equal to each other.

4.2.7.1 Homogeneous Circuits

For the case when $\forall i : k_i = k$, we propose an approximate solution of the w_is, of the following form:

$$w_i = \alpha_1 p_{i+1} + \alpha_2 p_{Avg}, \tag{4.23}$$

where

$$p_{Avg} = \frac{1}{m} \sum p_i \tag{4.24}$$

and α_1 and α_2 will be determined later. First, let us motivate Equation 4.23. Based on Property 2, we know that finding the w_is when $\forall i : k_i = k$ is equivalent to finding the w_is when $\forall i : k_i = 1$. In other words, the value of the w_is is independent of the k_is, when all k_is are equal. Conversely, based on Property 1, we know that the w_is scale linearly with the p_is. This suggests that the w_is should not have terms that are independent of the p_is. Based on our experience of sizing, we know that—while the transistor sizes of gate i depend mostly on p_{i+1}—the effect of a particular p_i gets distributed to some degree to all other gates. As a consequence, we would like Equation 4.23 to depend linearly on both p_{i+1} and some average of all other p_is and one such choice is $\alpha_1 p_{i+1} + \alpha_2 p_{Avg}$. We use the arithmetic mean for p_{Avg} since the p_is correspond physically to wire capacitances that are manipulated additively both in terms of delay and energy. With these clarifications in mind, we state the following:

Theorem 4 *For a neighborhood $\mathcal{V}_p = [p-\eta, p+\eta]$ of $p > 0, \eta > 0$, the values of α_1 and α_2 that minimize Et^n given the w_is of the form defined by Equation 4.23, where $\forall i : p_i \in \mathcal{V}_p$, $k_i = k > 0$ and η approaches zero, are*

$$\alpha_1 = \frac{\frac{1}{2}}{\frac{1}{n} + \frac{m}{m-1}} \quad \textit{and} \quad \alpha_2 = n - \frac{\frac{1}{2}}{\frac{1}{n} + \frac{m}{m-1}}.$$

Proof. We start with the observation that for $\forall i : p_i = p \in \mathcal{V}_p$, $k_i = k$ Equation 4.18 implies $\forall i : w_i = np$. Using the same values in Equation 4.23 yields $\forall i : w_i = \alpha_1 p + \alpha_2 p$. By combining these two relations it follows that $np = \alpha_1 p + \alpha_2 p \Rightarrow \alpha_2 = n - \alpha_1$.

Next, we focus on computing α_1. We notice that

$$\begin{aligned} E &= \sum_{i=0}^{m-1}(w_i + p_i) = \sum_{i=0}^{m-1}\Big(\alpha_1 p_{i+1} + (n-\alpha_1)p_{\mathrm{Avg}} + p_i\Big) \\ &= \alpha_1 \sum_{i=0}^{m-1} p_i + (n-\alpha_1) m p_{\mathrm{Avg}} + \sum_{i=0}^{m-1} p_i = (n+1)\sum_{i=0}^{m-1} p_i = \mathit{constant}. \end{aligned}$$

As a consequence,

$$\min\Big(t = \sum_{i=0}^{m-1} k_{i-1}\frac{w_i + p_i}{w_{i-1}}\Big) \Rightarrow \min f.$$

In other words, it is enough to compute α_1 that minimizes the simpler expression t instead of f. We minimize t as a function of α_1 by substituting w_i given by Equation 4.23 into the expression of t. One way to do this computation is to assign $p_0 = p(1 + m\epsilon)$ and $p_i = p, i \in 1..m-1$, where $\epsilon = \eta/(mp)$. Clearly, the chosen p_is belong to the neighborhood $\mathcal{V}_p$ of p. From the choice of the p_is it also follows that $p_{Avg} = p(1+\epsilon)$. As a consequence,

$$\begin{aligned} w_i &= \alpha_1 p + (n-\alpha_1)(1+\epsilon)p,\ \forall i \in 0..m-2 \\ w_{m-1} &= \alpha_1(1+m\epsilon)p + (n-\alpha_1)(1+\epsilon)p \end{aligned}$$

and we may write

$$\begin{aligned} t &= \frac{\alpha_1 p + (n-\alpha_1)(1+\epsilon)p + p}{\alpha_1(1+m\epsilon)p + (n-\alpha_1)(1+\epsilon)p} + (m-2)\frac{\alpha_1 p + (n-\alpha_1)(1+\epsilon)p + p}{\alpha_1 p + (n-\alpha_1)(1+\epsilon)p} \\ &+ \frac{\alpha_1(1+m\epsilon)p + (n-\alpha_1)(1+\epsilon)p + p}{\alpha_1 p + (n-\alpha_1)(1+\epsilon)p}. \end{aligned}$$

Minimizing t with respect to α_1, i.e., solving $\frac{dt}{d\alpha_1} = 0$, we obtain a quadratic equation in α_1 with coefficients that depend on n, m and ϵ. This equation has the positive root

$$\alpha_1 = n(1+\epsilon)\frac{-(1+\epsilon)(n+\frac{m-1}{m}) + \sqrt{(n\epsilon + n + \frac{m-1}{m})(n\epsilon + n + \frac{m-1}{m} - \epsilon + m\epsilon)}}{\epsilon\Big(\frac{(m-1)^2}{m} + (mn - 2n - \frac{m-1}{m})(1+\epsilon)\Big)}.$$

We would like to determine α_1 when η goes to zero or equivalently when ϵ goes to zero. It turns out that $\epsilon = 0$ creates a $\frac{0}{0}$ indeterminacy in the expression of α_1; thus, we need to use l'Hospital's rule, which yields $\alpha_1 = \frac{\frac{1}{2}}{\frac{1}{n}+\frac{m}{m-1}}$ and $\alpha_2 = n - \alpha_1 = n - \frac{\frac{1}{2}}{\frac{1}{n}+\frac{m}{m-1}}$.
□

If the size of the problem is large, $\frac{m}{m-1}$ is approximately one, which implies $\alpha_1 = \frac{n}{2(1+n)}$ and $\alpha_2 = \frac{n(1+2n)}{2(1+n)}$, thus $\frac{\alpha_2}{\alpha_1} = 1 + 2n$. What is particularly surprising about Equation 4.23 is that the width of a logic-gate depends far more strongly (5 times more for Et^2 optimization) on the *average* parasitic load ($\alpha_2 = 5/3$) than it does on the load on that *particular* operator ($\alpha_1 = 1/3$). Furthermore, on the limit when n goes to zero $\alpha_1 = \alpha_2 = 0$ which implies that $\forall i : w_i = 0$, regardless of the p_is. In other words, for energy-only optimization, Equation 4.23 yields minimum-size transistors, as one might expect.

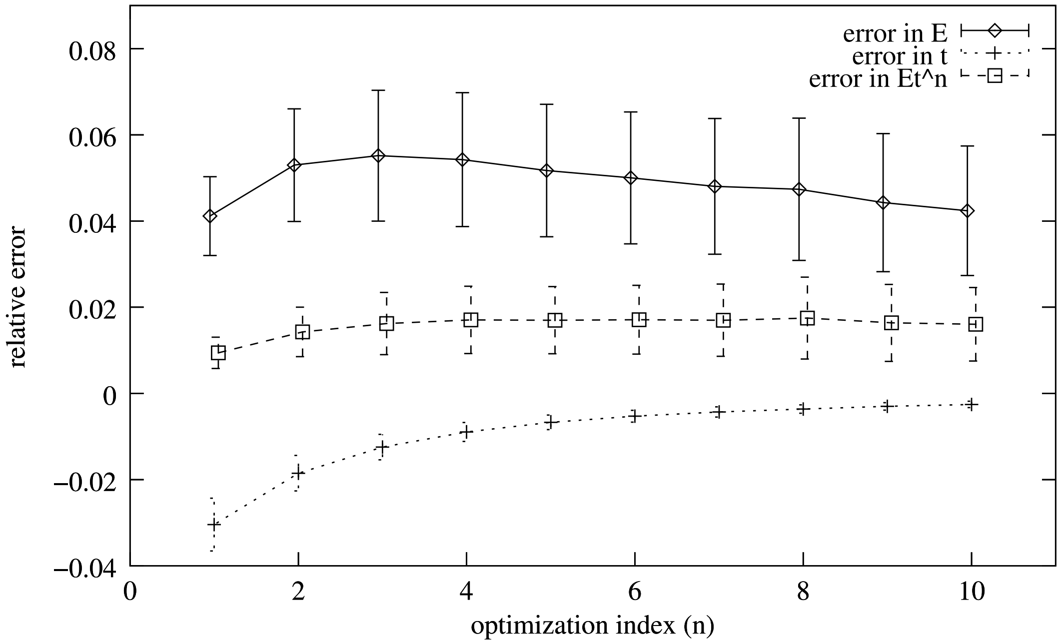

Figure 4.10: Accuracy in E, t and Et^n of Equation 4.23 with α_1 and α_2 given by Theorem 4.

Theorem 4 yields the optimal values of α_1 and α_2 in a small neighborhood of p, or equivalently when the p_is are close to each other. We now want to check if the form of the w_is given by Equation 4.23 and Theorem 4 yields a practical approximation of the w_is when $\forall i : k_i = k$ and p_is are no longer close to each other. Again, we rely on a numerical optimizer to compute the error between the optimal and the predicted $f = Et^n$ for a given n, m and a set of p_is. We varied m over the range $[2, 1000]$, n over the range $[1, 10]$ and used three different distributions (uniform, uniform-squared, uniform-cubed) for p_i in the range $[1, 100]$. The observed errors are practically independent of the problem size m and the distribution chosen for the p_is; the errors only depend on n. Figure 4.10 shows the error in E, t and $f = Et^n$ for $m = 31$, $n \in [1, 10]$, $k_i = 1$ and $p_i \in [1, 100]$ chosen randomly through a uniform-squared distribution. The average error in E is between 4.1% and 5.5%, the average error in t is between -3.0% and -0.3%, and the average error in $f = Et^n$ is between 1.0% and 1.7%.

4.2.7.2 Nonhomogeneous Circuits (First Form)

The formula resulting from Theorem 4 yields excellent results when all k_is are equal. Let us extend it to incorporate the case when the k_is are no longer all equal. To do this, we assume that the cumulative effect of the p_is and the k_is on the w_is can be viewed as the product of the individual effect of the p_is (wire capacitances) on the w_is and the individual effect of the k_is (gate topology) on the w_is. Hence, we propose an approximate solution of the w_is of the following form:

$$w_i = (\alpha_1 p_{i+1} + \alpha_2 p_{Avg})\ r_i(k_0, k_1, ..., k_{m-1}) \tag{4.25}$$

where α_1 and α_2 are given by Theorem 4, while the functions r_i will be determined later. When all gates are identical, i.e., when $\forall i : k_i = k$, we know from Equation 4.18 that the w_is are independent of the k_is. For this reason, we choose $r_i(k_0, k_1, ..., k_{m-1})$ such that $\forall k :\ r_i(k, k, ..., k) = 1$.

Based on our experience of delay-only transistor sizing, we know that—while the transistor sizes of gate i depend strongly on k_i—the effect of a particular k_i gets distributed to some degree to all other gates. As a consequence, we would like r_i to depend on both k_i and some average of all other k_is. We use the geometric mean $k_{Avg} = \sqrt[m]{\prod k_i}$ as an average of the k_is, since it has physical meaning—it is proportional to the theoretical minimal delay t_∞ of the cycle. In this context, we introduce the following theorem:

Theorem 5 *For a neighborhood $\mathcal{V}_p = [p-\eta, p+\eta]$ of $p > 0, \eta > 0$ and a neighborhood $\mathcal{V}_k = [k-\eta, k+\eta]$ of $k > 0$, the values of α_1, α_2, β_1 and β_2 that minimize Et^n given the w_is of the form defined by Equation 4.25 with $r_i(k_0, k_1, ..., k_{m-1}) = \beta_1 \frac{k_i}{k_{Avg}} + \beta_2$, where $\forall i : p_i \in \mathcal{V}_p$, $k_i \in \mathcal{V}_k$, and η approaches zero, are*

$$\alpha_1 = \frac{\frac{1}{2}}{\frac{1}{n} + \frac{m}{m-1}}, \alpha_2 = n - \frac{\frac{1}{2}}{\frac{1}{n} + \frac{m}{m-1}},\ \textit{and}\ \ \beta_1 = \beta_2 = \frac{1}{2}.$$

Proof. By substituting $\forall i : k_i = k \in \mathcal{V}_k$ the theorem reduces to Equation 4.23, for which Theorem 4 states the required values for α_1 and α_2. Thus, we need to focus only on β_1 and β_2. In order to compute these values, we apply a similar strategy to the one in the proof of Theorem 4: we eliminate α_1 and α_2 by choosing $\forall i : p_i = p$ and then compute β_1 and β_2 to minimize $f = Et^n$ around a small neighborhood $\mathcal{V}_k$ of k. If we substitute $\forall i : k_i = k \in \mathcal{V}_k$, Equation 4.13 yields $w_i = np$. On the other hand, from the hypothesis of Theorem 5 $w_i = np(\beta_1 + \beta_2)$. Combining these last two relations, it follows that $\beta_2 = 1 - \beta_1$.

We now pick $k_i = k, i \in 0..m-2$ and $k_{m-1} = k(1 + m\epsilon)$ where $\epsilon = \eta/(mk)$. Because we are computing β_1 and β_2 around $k_i \in \mathcal{V}_k$, we use the computationally simpler arithmetic mean instead of the geometric mean in the expression of k_{Avg} (on the limit when η approaches zero, the two means are equal). This choice of k_{Avg} allows us to minimize $t = \sum_{i=0}^{m-1} k_{i-1} \frac{w_i + p_i}{w_{i-1}}$ instead of f because E is constant. With the specified values of k_i, $k_{Avg} = k(1 + \epsilon)$ and

$$
\begin{aligned}
w_i &= np\Big(1 - \frac{\beta_1 \epsilon}{1+\epsilon}\Big), \ \forall i \in 0..m-2, \\
w_{m-1} &= np\Big(1 + (m-1)\frac{\beta_1 \epsilon}{1+\epsilon}\Big).
\end{aligned}
$$

It follows that

$$
\begin{aligned}
t &= k(1+m\epsilon)\frac{np\Big(1 - \frac{\beta_1\epsilon}{1+\epsilon}\Big) + p}{np\Big(1 + (m-1)\frac{\beta_1\epsilon}{1+\epsilon}\Big)} + k(m-2)\frac{np\Big(1 - \frac{\beta_1\epsilon}{1+\epsilon}\Big) + p}{np\Big(1 - \frac{\beta_1\epsilon}{1+\epsilon}\Big)} \\
&+ k\frac{np\Big(1 + (m-1)\frac{\beta_1\epsilon}{1+\epsilon}\Big) + p}{np\Big(1 - \frac{\beta_1\epsilon}{1+\epsilon}\Big)}.
\end{aligned}
$$

Minimizing t with respect to β_1, i.e., solving $\frac{dt}{d\beta_1} = 0$, we obtain a quadratic equation in β_1 with coefficients that depend on m and ϵ. This equation has the positive root

$$
\beta_1 = \frac{(1+\epsilon)(1 + \epsilon - \sqrt{1 + m\epsilon})}{\epsilon(2 - m + \epsilon)}.
$$

Once again, $\epsilon = 0$ creates a $\frac{0}{0}$ indeterminacy in the expression of β_1; thus, we need to use l'Hospital's rule, which yields $\beta_1 = \frac{1}{2}$ and $\beta_2 = 1 - \beta_1 = \frac{1}{2}$. $\square$

If we substitute the values of α_1, α_2, β_1 and β_2 given by Theorem 5 into Equation 4.25, we get the following expression:

$$w_i = \frac{1}{2}\left(\frac{\frac{1}{2}}{(\frac{1}{n} + \frac{m}{m-1})}p_{i+1} + \left(n - \frac{\frac{1}{2}}{(\frac{1}{n} + \frac{m}{m-1})}\right)p_{Avg}\right)\left(\frac{k_i}{k_{Avg}} + 1\right). \tag{4.26}$$

Theorem 5 yields the optimal values of α_1, α_2, β_1 and β_2 when the p_is are in a close neighborhood of p, and the k_is are in a close neighborhood of k. We would like to verify now how good these values are in minimizing Et^n when the p_is and the k_is are no longer close to each other. We again use a numerical optimizer to compute the error between the optimal and the estimated $f = Et^n$ for a given n, m and a set of p_is and k_is. We varied m in the range $[2, 1000]$, n in the range $[1, 10]$ and used three different distributions (uniform, uniform-squared, and uniform-cubed) for p_i in the range $[1, 100]$ and k_i in the range $[1, 3.3]$ (if we assume k_{ni} in the range $[1, 6]$ and k_{pi} in the range $[1, 2]$, then with $\mu = 2.5$ we get k_i in the range $[6.66, 21.95]$ or equivalently, using Property 2, k_i in the range $[1, 3.3]$). Same as for Equation 4.23, the observed errors are practically independent of the problem size m and the distribution chosen for the p_is and the k_is; the errors only depend on n. Figure 4.11 shows the error in E, t and $f = Et^n$ for $m = 31$, $n \in [1, 10]$ and $p_i \in [1, 100]$, $k_i \in [1, 3.3]$ chosen randomly through a uniform-squared distribution. The average error in E is between 1.7% and 6.1%, the average error in t is between -3.4% and 1.7%, and the average error in $f = Et^n$ is about 3.3% for $n = 2$, but increasing about linearly with n, owing to the error amplifying artifact of Et^n (if $t = t_0(1 + \Delta) \Rightarrow t^n \approx t_0^n(1 + n\Delta)$ for small Δ).

4.2.7.3 Nonhomogeneous Circuits (Second Form)

The main intended use of Equation 4.25 in energy-delay efficient design is to find approximate transistor sizes when $n \approx 2$, i.e., when voltage scaling is a design parameter. As Figure 4.11 shows, Equation 4.26, i.e., a particular case of Equation 4.25, does this reasonably well—i.e., within a few percent of the optimum. On the other hand, one might want to use such a sizing formula for large n as well—i.e., predomi-

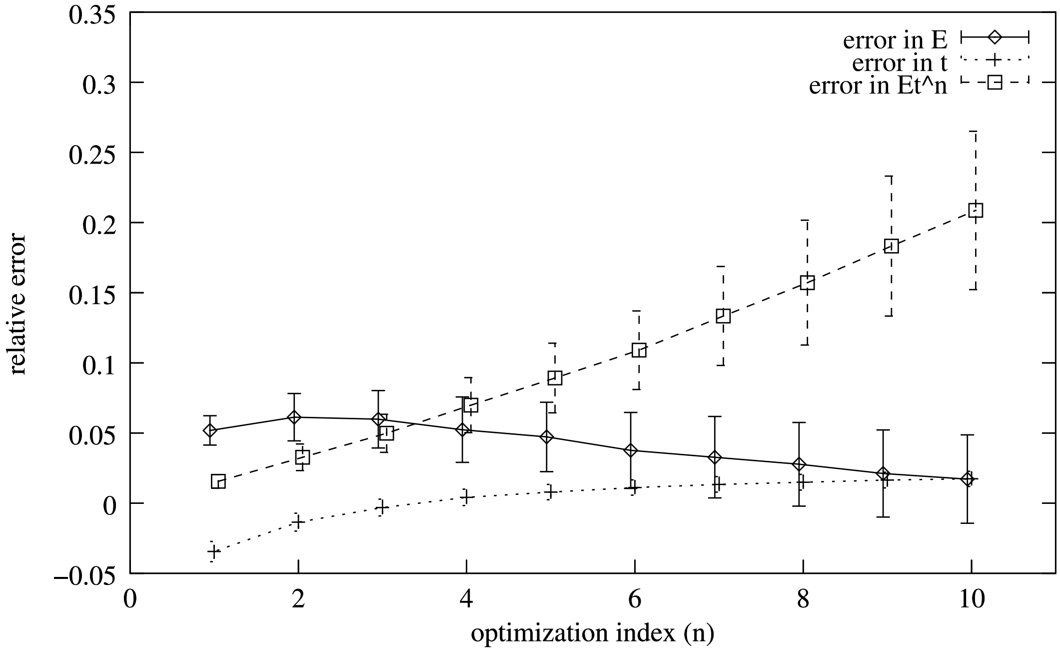

Figure 4.11: Accuracy in E, t and Et^n of Equation 4.26.

nantly delay-only optimization. Getting a close approximation of $f = Et^n$ when n is large requires a very good delay estimate, since even a small error Δ in t gets linearly amplified to $n\Delta$ in $f = Et^n$. For this reason, we study the behavior of Equation 4.25 and the delay estimate resulting from it, when n goes to infinity.

For now, consider a simpler problem, namely finding the transistor widths $w_{\infty i}$ that minimize t given by Equation 4.9. This is a special case of the Et^n optimization problem when n goes to infinity. If we apply the inequality between the arithmetic mean and the geometric mean to the terms of Equation 4.9 that are independent of the p_is, we get

$$t \geq m \sqrt[m]{\prod_{i=0}^{m-1} k_i} + \sum_{i=0}^{m-1} k_i \frac{p_{i+1}}{w_i}, \tag{4.27}$$

with equality if and only if $\forall i : k_i \frac{w_{i+1}}{w_i} = \sqrt[m]{\prod k_i} = k_{Avg}$. The second term in the right-hand side of Equation 4.27 is minimized when the w_is are infinitely large. As a consequence, the optimal delay $t_\infty = mk_{Avg}$ is reached for transistor widths $w_{\infty i} = \omega \frac{k^i_{Avg}}{k_0 k_1 \ldots k_{i-1}}$, where ω is an infinitely large scaling factor. In particular, the $w_{\infty i}$s have the property that

$$\forall i : \frac{w_{\infty(i+1)}}{w_{\infty i}} = \frac{k_{Avg}}{k_i}. \tag{4.28}$$

We would like the w_is given by Equation 4.25 to reduce to the $w_{\infty i}$s, for large n. This condition would guarantee that the delay estimate resulting from Equation 4.25 is optimal for large n. However, we cannot express this condition properly simply by $\lim_{n\to\infty} w_i = w_{\infty i}$, since both terms are infinitely large ($\lim_{n\to\infty} \alpha_2 = \infty \Rightarrow \lim_{n\to\infty} w_i = \infty$). Instead, we express the requirement on the w_is as a condition on the ratio of transistor widths. More precisely, we require

$$\lim_{n\to\infty} \frac{w_{i+1}}{w_i} = \frac{w_{\infty(i+1)}}{w_{\infty i}}. \tag{4.29}$$

Using α_1 and α_2 given by Theorem 5, we have

$$\begin{aligned} \lim_{n\to\infty} \frac{w_{i+1}}{w_i} &= \lim_{n\to\infty} \frac{(\alpha_1 p_{i+2} + \alpha_2 p_{\mathrm{Avg}}) r_{i+1}(k_0, k_1, \ldots, k_{m-1})}{(\alpha_1 p_{i+1} + \alpha_2 p_{\mathrm{Avg}}) r_i(k_0, k_1, \ldots, k_{m-1})} \\ &= \lim_{n\to\infty} \frac{(\alpha_1 p_{i+2} + \alpha_2 p_{\mathrm{Avg}})}{(\alpha_1 p_{i+1} + \alpha_2 p_{\mathrm{Avg}})} \lim_{n\to\infty} \frac{r_{i+1}(k_0, k_1, \ldots, k_{m-1})}{r_i(k_0, k_1, \ldots, k_{m-1})} \\ &= \lim_{n\to\infty} \frac{r_{i+1}(k_0, k_1, \ldots, k_{m-1})}{r_i(k_0, k_1, \ldots, k_{m-1})}; \end{aligned}$$

thus Equation 4.29 can be restated in terms of the r_is as

$$\lim_{n\to\infty} \frac{w_{i+1}}{w_i} = \lim_{n\to\infty} \frac{r_{i+1}(k_0, k_1, \ldots, k_{m-1})}{r_i(k_0, k_1, \ldots, k_{m-1})} = \frac{w_{\infty(i+1)}}{w_{\infty i}}. \tag{4.30}$$

Condition 4.30 guarantees that the delay estimate resulting from Equation 4.25 is optimal when n increases without bound. An obvious choice of the r_is that fulfill Equation 4.30 is $\forall i : r_i(k_0, k_1, \ldots, k_{m-1}) = \beta w_{\infty i}$, where $\beta > 0$ is a constant scaling factor. The role of β is to normalize the w_is to the right energy level; its optimal value is stated by the following theorem:

Theorem 6 *For a neighborhood $\mathcal{V}_p = [p-\eta, p+\eta]$ of $p > 0, \eta > 0$ and a neighborhood $\mathcal{V}_k = [k-\eta, k+\eta]$ of $k > 0$, the values of α_1, α_2, β that minimize Et^n given the w_is of* *the form defined by Equation 4.25 with* $r_i(k_0, k_1, ..., k_{m-1}) = \beta w_{\infty i}$, *where* $\forall i : p_i \in \mathcal{V}_p$, $k_i \in \mathcal{V}_k$, *and* η *approaches zero, are*

$$\alpha_1 = \frac{\frac{1}{2}}{\frac{1}{n} + \frac{m}{m-1}}, \alpha_2 = n - \frac{\frac{1}{2}}{\frac{1}{n} + \frac{m}{m-1}}, \text{ and } \beta = \frac{S_2}{2}\left(\left(1 - \frac{1}{n}\right) + \sqrt{\left(1 - \frac{1}{n}\right)^2 + \frac{4}{nS_1S_2}}\right),$$

where

$$S_1 = \frac{1}{m}\sum_{i=0}^{m-1} w_{\infty i} \text{ and } S_2 = \frac{1}{m}\sum_{i=0}^{m-1}\frac{1}{w_{\infty i}}.$$

Proof. The part stating the values of α_1 and α_2 is proved exactly as it was proved for Theorem 5. For the second part, we assume $\forall i : p_i = p$ and find β that minimizes Et^n. With the notations given in the theorem, $E = mp(1 + nS_1\beta)$ and $t = mk_{Avg}\left(1 + \frac{S_2}{n\beta}\right)$. To optimize Et^n function of β, we compute

$$\frac{d(Et^n)}{d\beta} = 0 \Rightarrow \beta = \frac{S_2}{2}\left(\left(1 - \frac{1}{n}\right) + \sqrt{\left(1 - \frac{1}{n}\right)^2 + \frac{4}{nS_1S_2}}\right). \quad \square$$

Given the value of β from Theorem 6, we have that $\forall n \geq 0 : \frac{1}{S_1} \leq \beta \leq S_2$ with $\beta = \frac{1}{S_1}$ if n approaches zero and $\beta = S_2$ if n becomes infinitely large. If we choose $\beta = \frac{1}{S_1}$, the error in E is reduced by bringing E close to the energy optimum $(1+n)E_0$ given by Equation 4.19, while if we choose $\beta = S_2$, the error in t is reduced by bringing t close to the delay optimum $(1 + \frac{1}{n})t_\infty$ given by Equation 4.20. If we choose β from Theorem 6, the error in Et^n is minimized.

The formula resulting from Theorem 6 works extremely well in practice for small m, i.e., it keeps the error in Et^n very low for the entire range of n, including large n. However, for large m the accuracy of the formula deteriorates somewhat owing to the fact that E becomes consistently overestimated, while the estimate in t stays very accurate. This is a consequence of the choice of the r_is, where we have intentionally favored the accuracy of the delay estimation. For large ms, the difference between $\frac{1}{S_1}$ and S_2 becomes large enough that the resulting β pulls E noticeably away from the optimum $(1 + n)E_0$.

Figure 4.12 shows the error in E, t and $f = Et^n$ for the approximation given

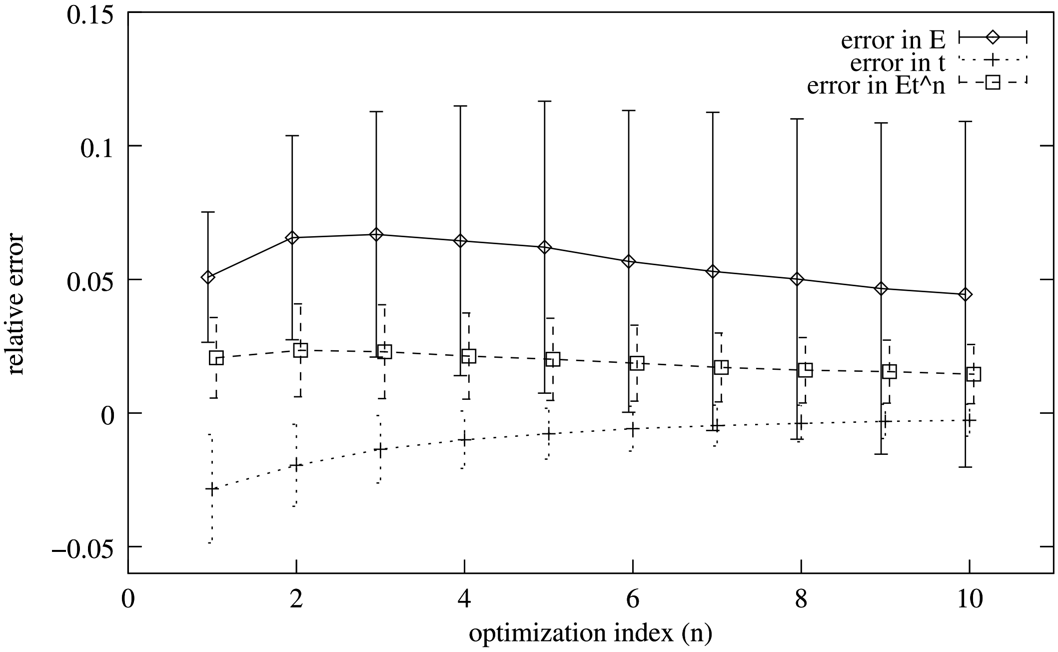

Figure 4.12: Accuracy in E, t and Et^n of the approximation given by Theorem 6.

by Theorem 6 for $m = 9$ (a circuit with 18 transitions per cycle), $n \in [1, 10]$ and $p_i \in [1, 100]$, $k_i \in [1, 3.3]$ chosen randomly through a uniform-squared distribution. The average error in E is between 4.4% and 6.7%, the average error in t is between -0.2% and -2.8%, and the average error in Et^n is between 1.4% and 2.3% for the entire range of n. It is interesting to point out that for $n = 100$, the average error in E is about 1.2%, the average error in t is about -0.003%, and the average error in $f = Et^n$ is about 0.5%.

For clarity, Theorems 4, 5, and 6 were formulated to refer to the transistor sizing problem of a single-cycle system. However, these theorems can easily be extended to multi-cycle systems. We extend formula 4.23, and as a consequence Theorem 4, to multi-cycle systems by redefining p_{Avg} for each gate i to be the average parasitic of all cycles gate i is part of. Theorem 5 extends to multi-cycle systems by substituting mk_{Avg} with t_∞ (the minimum achievable delay of the circuit). Given the definitions of $w_{\infty i}$s and p_{Avg}, Theorem 6 generalizes straightforwardly to multi-cycle systems, with the only change that m—in the expression of S_1 and S_2—represents the total number of transistors in the considered circuit, not just the ones on a given cycle.

Remembering the derivation of section 4.2.1.2, the values of the w_is are per operator i; but they can be transformed into the effective nMOS and pMOS transistor sizes directly, using Equations 4.5, 4.7 and 4.8.

4.2.8 An Iterative Approach to Et^n-Optimal Transistor Sizing

With the help of Theorems 5 and 6, we can compute approximate transistor sizes of an Et^n-optimal circuit. As we have seen, the approximate solution yields energy and delay values within a few percent of the optimum. However, if the accuracy of such a solution is not acceptable for the given application, one can employ an iterative procedure to “fine tune” the initial transistor sizes.

Using Equation 4.12, we can compute w_i—for a fixed i—as a function of the other ws. More precisely, if we define

$$a_2 = \frac{b_1 + nb_0b_2}{(n+1)b_2}, \; a_1 = \frac{(-n+1)b_3}{(n+1)b_2}, \; \text{ and } \; a_0 = \frac{-nb_0b_3}{(n+1)b_2},$$

where

$$b_0 = \sum_{i=1, i\neq j}^{m} w_j + \sum_{i=1}^{m} p_j,$$

$$b_1 = \sum_{i=1, i\neq j, i\neq j+1}^{m} k_{j-1}\frac{w_j + p_j}{w_{j-1}} + k_{i-1}\frac{p_i}{w_{i-1}},$$

$$b_2 = \frac{k_{i-1}}{w_{i-1}},$$

and

$$b_3 = k_i(w_{i+1} + p_{i+1});$$

we can compute, using Cardano's formula, w_i as the positive solution to the cubic equation

$$w_i^3 + a_2w_i^2 + a_1w_i + a_0 = 0. \tag{4.31}$$

Equation 4.31 has three solutions, which could be either all real, or one real and two imaginary. If there is only one real solution, noting that $a_0 < 0$, that solution has to be positive. If all solutions are real, they are either all positive or one positive and two negative. But, noting that $a_1 \leq 0$ if $n \geq 1$ implies that not all solutions are positive. In conclusion, Equation 4.31 has a single positive solution for $n \geq 1$, and that is the solution we are interested in.

The iterative procedure starts with an initial solution and then repetitively computes each w_i as the positive solution of Equation 4.31 with coefficients computed from the current value of all other ws. It is easy to see that such a procedure converges to the Et^n-optimal solution. First, the recomputed value of w_i yields a better Et^n than the pre-iteration value. This is because $\frac{\partial Et^n}{\partial w_i} = 0$, i.e., the new w_i is Et^n-optimal when all other ws are fixed at their current value. Second, the Et^n optimization problem is convex in the ws, hence a local minimum reached by the iteration procedure is indeed the global minimum.

To fully appreciate the benefit of the proposed iterative procedure when applied to the initial solution given by Theorems 5 or 6, we exhaustively analyze a particular case of Equation 4.11 with $n = 2$, $m = 5$, $p_i \in \{1, 2, 3, 4, 5\}$ and $k_i \in \{1, 2, 3\}$. Figure 4.13 shows a histogram of the relative error in Et^2 between the optimal values (computed with an optimization algorithm) and the estimated values based on Theorem 6, and also between the optimal values and the values computed by one step of the iteration procedure starting with the approximate solution given by Theorem 6. One step of iteration assigns one new value to each w_i. We observe that the already small maximal error of the original sizing formula is reduced about ten-fold by a single step of the iteration procedure. Of course, one can repeat the same procedure and get an even smaller error. However, this second step does not have the same impact on reducing the error as the first step had. Given that the transistor sizes of a real circuit are integer multiples of a technology-dependent constant, there is not much point in trying to find the zero-error solution. That solution is unlikely to be implementable in practice, since it will likely have non-integer components.

We have done several experiments in which we tested the dependence of the it-

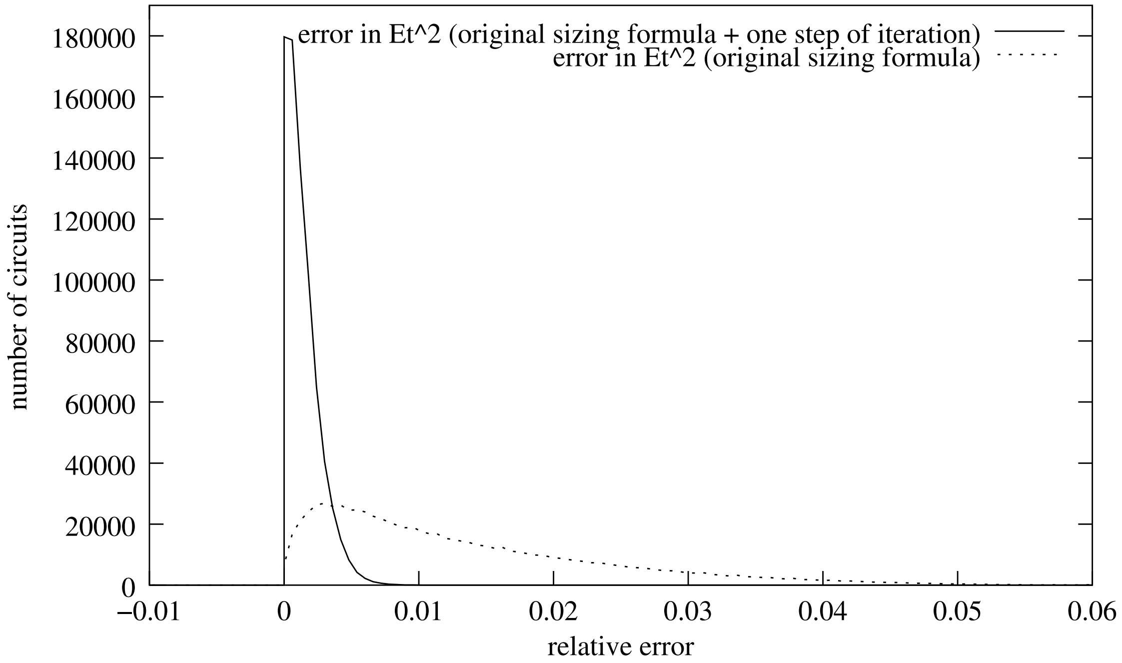

Figure 4.13: Error in Et^2 when exhaustively simulating an entire class of circuits.

eration procedure on the initial starting point. We have found that the applicability of the method strongly depends on the initial solution's proximity to the optimal solution. Without a good initial solution like the one given by Theorem 5 or 6, the method nevertheless converges to the optimum eventually. However, the first step of iteration yields a solution that has an error spread two orders of magnitude greater than the solution resulting from the first step of the iteration on the good initial solution.

4.2.9 An Algorithm for Et^n-Optimal Transistor Sizing

As we have seen, the transistor sizes w_i of a system optimized for Et^n depend heavily on the wire parasitics p_i. However, these parasitics are not known to begin with, since they are meant to connect the transistors whose dimensions are yet to be found. This circular structure makes transistor sizing a chicken and egg problem—which one was first?

A two-phase algorithm solves the problem of the unknown parasitics. In the first phase, given the transistor netlist, each wire is assigned an initial wiring cost. The more is known about the structure of the transistor netlist and about a future floorplan, the more accurate such an assignment will be. Based on these initial wire parasitics, we can then compute an initial estimate for the w_is with the formulas established by Theorems 5 and 6.

In the second phase, the pre-sized transistor netlist is wired up and the actual wire capacitances are extracted from the layout. With these new parasitics, we recompute the transistor widths w_i. Finally, we may fine-tune the solution by iterating once as described in Section 4.2.8.

If the accuracy of the solution is still not acceptable, a third phase can be added to the transistor-sizing algorithm. In this last phase, the approximate w_is are used as the starting point of a numerical optimizer. Given the proximity of the current solution to the optimum, such an optimization should converge quickly. In this last phase, a more accurate transistor model (e.g., a BSIM model) can be employed, so as to bridge the gap between the simplified transistor model used to infer Theorem 5 and 6, and the actual transistor behavior.

4.2.10 A Transistor-Sizing Example

In this section we present a simple optimization example in which we show how the transistor sizing formulas of section 4.2.7 are applied to a concrete circuit. The circuit we consider is the serial composition of several identical dual-rail weak-condition half-buffers [10]. This circuit is an asynchronous buffer that passes one bit, encoded as a dual-rail code, from left to right. The corresponding circuit diagram is shown in Figure 4.14. The cycle time of this circuit is determined by the speed of the four-phase handshake between two consecutive buffers. It can be written as

$$\begin{aligned} t &= 2\frac{w_{n2}+w_{p2}+p_1+w_{n3}+w_{p3}+p_2}{w_{n1}} + 2\mu\frac{w_{n2}+w_{p2}+p_1+w_{n3}+w_{p3}+p_2}{w_{p1}} \\ &+ 2\frac{w_{n4}+w_{p4}+p_3}{w_{n3}} + \mu\frac{w_{n4}+w_{p4}+p_3}{w_{p3}} \end{aligned}$$

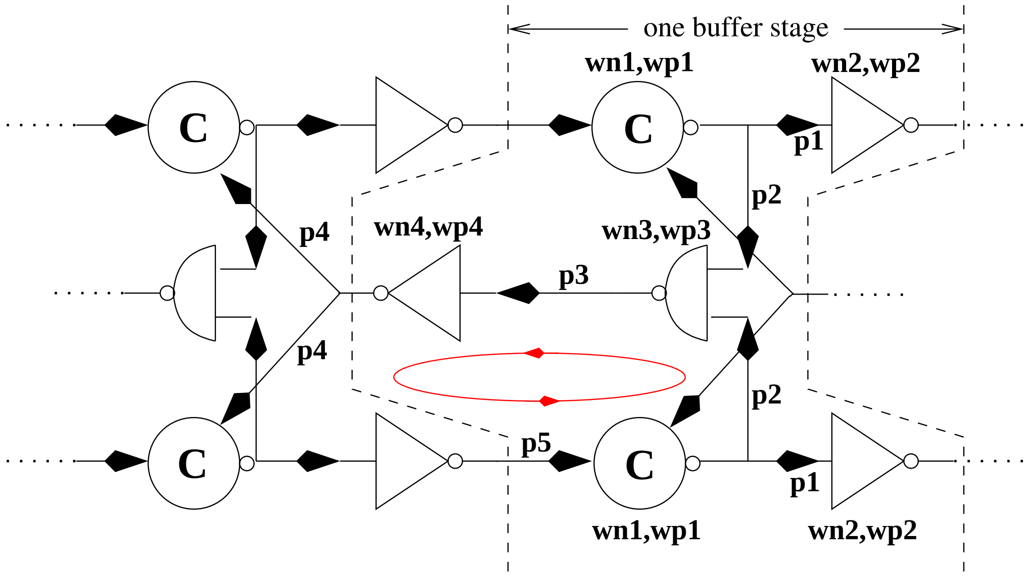

Figure 4.14: Optimal transistor sizing of a chain of buffers.

$$\begin{aligned}
&+ \quad \frac{2(w_{n1}+w_{p1}+p_4)}{w_{n4}} && + \mu\frac{2(w_{n1}+w_{p1}+p_4)}{w_{p4}} \\
&+ \quad 2\frac{w_{n2}+w_{p2}+p_1+w_{n3}+w_{p3}+p_2}{w_{n1}} && + 2\mu\frac{w_{n2}+w_{p2}+p_1+w_{n3}+w_{p3}+p_2}{w_{p1}} \\
&+ \quad \frac{w_{n1}+w_{p1}+p_5}{w_{n2}} && + \mu\frac{w_{n1}+w_{p1}+p_5}{w_{p2}}.
\end{aligned}$$

where for this example we assume $\mu = 3$. Similarly, the total energy consumption per operation can be written as

$$\begin{aligned}
E &= w_{n2}+w_{p2}+p_1+w_{n3}+w_{p3}+p_2+w_{n4}+w_{p4}+p_3 \\
&+ w_{n1}+w_{p1}+p_5 \qquad\qquad + 2(w_{n1}+w_{p1}+p_4).
\end{aligned}$$

If we optimize this circuit only for t when $\forall i : p_i = 0$, the minimal delay $t_\infty = 79.06$ is reached for

$$\begin{aligned}
w_{\infty n1} &= 2.75\omega, \quad w_{\infty p1} = 4.76\omega, \\
w_{\infty n2} &= 1.38\omega, \quad w_{\infty p2} = 2.38\omega,
\end{aligned}$$

$$w_{\infty n3} = 1.86\omega, \quad w_{\infty p3} = 2.27\omega,$$

$$w_{\infty n4} = 2.50\omega, \quad w_{\infty p4} = 4.34\omega,$$

where $\omega > 0$ is an arbitrary constant.

For the case of Et^n optimization, assume $p_1 = 3$, $p_2 = 1$, $p_3 = 1.5$, $p_4 = 2$, and $p_5 = 3$. We first compute all the constants needed to apply the proposed sizing formulas. The length of the original cycle is 10, but after operator reduction, i.e., the elimination of pMOS transistors using Equation 3, it becomes 5, i.e., $m = 5$. Using this value of m, we can compute α_1 and α_2 for any n. Then,

$$\begin{aligned}
p_{avg} &= \frac{1}{m}(p_1 + p_2 + p_3 + p_4 + p_5) = 2.1, \\
S1 &= \frac{1}{m}\Big(w_{\infty n1} + w_{\infty p1} + w_{\infty n2} + w_{\infty p2} + 2(w_{\infty n1} + w_{\infty p1}) \\
&+ w_{\infty n3} + w_{\infty p3} + w_{\infty n4} + w_{\infty p4}\Big) = 7.45, \\
S2 &= \frac{1}{m}\Big(\frac{1}{w_{\infty n1} + w_{\infty p1}} + \frac{1}{w_{\infty n2} + w_{\infty p2}} + \frac{2}{w_{\infty n1} + w_{\infty p1}} \\
&+ \frac{1}{w_{\infty n3} + w_{\infty p3}} + \frac{1}{w_{\infty n4} + w_{\infty p4}}\Big) = 0.21.
\end{aligned}$$

Using these values of S_1 and S_2, we can compute β for any n. We can now directly compute the Et^n-optimal nMOS and pMOS transistor widths as

$$\begin{aligned}
w_{n1} &= (\alpha_1 \frac{p_1 + p_2}{2} + \alpha_2 p_{avg}) r_{n1}, \\
w_{p1} &= (\alpha_1 \frac{p_1 + p_2}{2} + \alpha_2 p_{avg}) r_{p1}, \\
w_{n2} &= (\alpha_1 p_5 + \alpha_2 p_{avg}) r_{n2}, \\
w_{p2} &= (\alpha_1 p_5 + \alpha_2 p_{avg}) r_{p2}, \\
w_{n3} &= (\alpha_1 p_3 + \alpha_2 p_{avg}) r_{n3}, \\
w_{p3} &= (\alpha_1 p_3 + \alpha_2 p_{avg}) r_{p3}, \\
w_{n4} &= (\alpha_1 p_4 + \alpha_2 p_{avg}) r_{n4},
\end{aligned}$$

$$w_{p4} = (\alpha_1 p_4 + \alpha_2 p_{avg}) r_{p4},$$

where

$$\begin{aligned}
r_{n1} &= \frac{1}{2}\Big(\frac{4(1+\sqrt{\mu})^2 m}{t_\infty} + 1\Big)\frac{1}{1+\sqrt{\mu}},\\
r_{p1} &= r_{n1}\sqrt{\mu},\\
r_{n2} &= \frac{1}{2}\Big(\frac{(1+\sqrt{\mu})^2 m}{t_\infty} + 1\Big)\frac{1}{1+\sqrt{\mu}},\\
r_{p2} &= r_{n2}\sqrt{\mu},\\
r_{n3} &= \frac{1}{2}\Big(\frac{2(1+\sqrt{\frac{\mu}{2}})^2 m}{t_\infty} + 1\Big)\frac{1}{1+\sqrt{\frac{\mu}{2}}},\\
r_{p3} &= r_{n3}\sqrt{\frac{\mu}{2}},\\
r_{n4} &= \frac{1}{2}\Big(\frac{2(1+\sqrt{\mu})^2 m}{t_\infty} + 1\Big)\frac{1}{1+\sqrt{\mu}},\\
r_{p4} &= r_{n4}\sqrt{\mu},
\end{aligned}$$

if we use Theorem 5, or $r_{ni} = \beta w_{ni}$ and $r_{pi} = \beta w_{pi}$ if we use Theorem 6.

We have verified the accuracy of the proposed transistor sizes using a numerical optimizer. We have computed the error between the optimal and the predicted E, t and Et^n for $n \in [1, 10]$. For the first formula (given by Theorem 5), the average error in E is between 1.6% and 3.6%, the average error in t is between -2.5% and 1.5%, and the average error in Et^n is about 2.4% for $n = 2$, but, as in Figure 4.11, increasing about linearly with n. (for $n = 10$ the error in Et^n becomes 18.2%). For the second formula (given by Theorem 6), the average error in E is between 1.1% and 6.7%, the average error in t is between -2.1% and -0.7%, and the average error in Et^n is between 0.2% and 0.9%. Both the magnitude and general shape of the error functions resulting from using the sizing formulas in this example are consistent with the error behavior seen in Figures 4.11 and 4.12.

If the example circuit is optimally sized (using a numerical optimizer) predom-

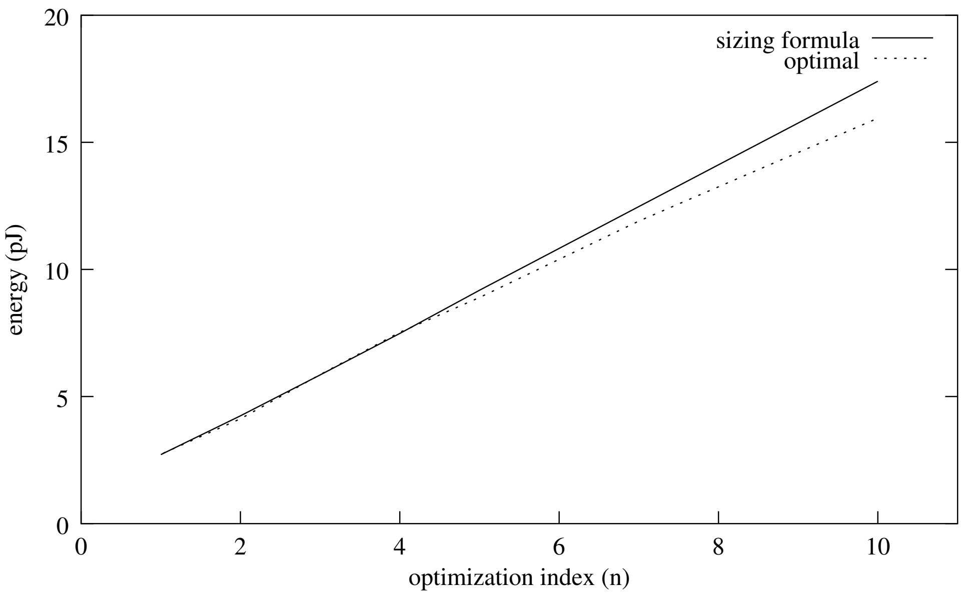

Figure 4.15: Circuit energy using estimated and optimal transistor sizes.

inantly for speed ($n = 20$), its energy consumption is $E_{opt} = 291.75$ and its delay is $t_{opt} = 83.04$. On the other hand, if the circuit is optimized for $n = 2$ using the proposed sizing formulas, the resulting energy and delay values are $t_2 = 119.71$, $E_2 = 40.65$ (first form) and $t_2 = 117.92$, $E_2 = 40.99$ (second form). The energy consumption is reduced by 86%, while the delay is increase by 40%; as a consequence, the Et^2 is improved by 350%.

The transistor sizes inferred in this example have been checked in practice. We have simulated in `hspice`, using the HP's 0.6μm process, both the circuits resulting from using the optimal transistor sizes and the circuits resulting from using the transistor sizing formula, for values of n in the range $[1, 10]$. The optimal transistor sizes were obtain using an iterative method similar to Powell's method. The iteration procedure started with the optimal analytical solution, based on the simple transistor model, and produced an optimal `hspice` solution. Figures 4.15 and 4.16 show the energy and the delay of these circuits as a function of n, simulated in `hspice`. The accuracy of these results is consistent with the accuracy of the theoretically predicted results.

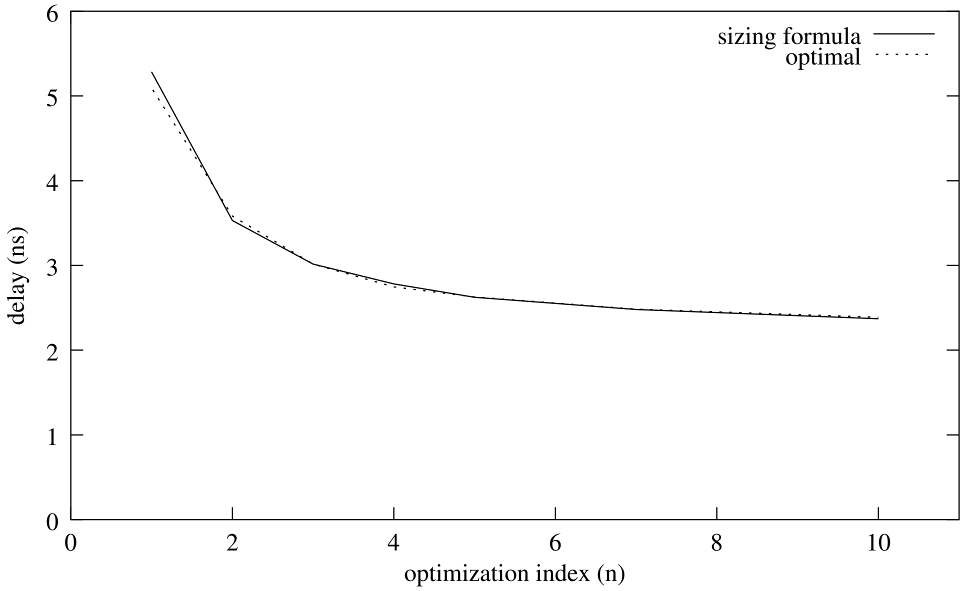

Figure 4.16: Circuit delay using estimated and optimal transistor sizes.

4.3 An Abstract View on Transistor Sizing

If we eliminate n from Equations 4.19 and 4.20, we arrive at the following relation between the minimum energy and the minimum delay of a circuit at a fixed voltage:

$$E_n = \frac{E_0 t_n}{t_n - t_\infty} \ . \tag{4.32}$$

Based on this expression, we can define an antimonotonic *minimum-energy function* $E(t)$ that describes the effect of transistor sizing on the minimum energy required for a system to run at a given t at a fixed voltage. (Tierno has previously used a similar energy function [48].) If we rewrite Equation 4.32 with E a function of t, we get the following function:

$$E(t) = \frac{E_0 t}{t - t_\infty} \ . \tag{4.33}$$

It is easy to prove that Equation 4.33 satisfies the above definition of the minimum-energy function. In the Chapter 5 we will extend the notion of *minimum-energy function* and we will use Equation 4.33 to infer properties of composed systems.

If we consider that the RC delay and the intrinsic gate delay are non-zero, i.e.,

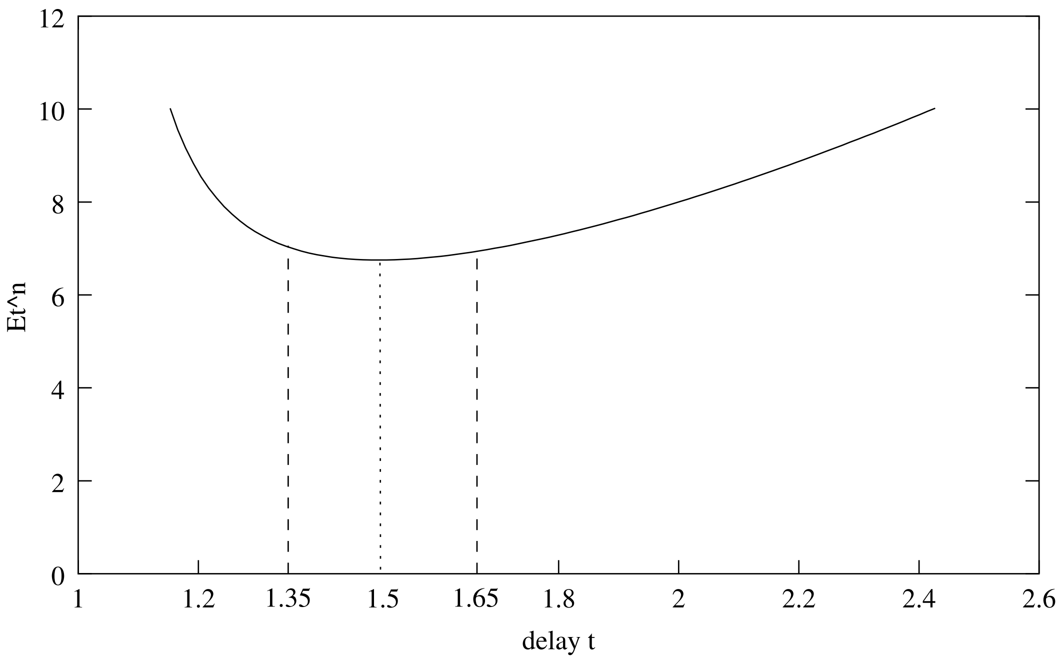

Figure 4.17: The Et^2 curve around the optimum.

$t_{fix} \neq 0$, Equation 4.33 becomes

$$E(t) = \frac{E_0(t - t_{fix})}{t - t_{fix} - t_\infty} \; , \tag{4.34}$$

or equivalently

$$E(t + t_{fix}) = \frac{E_0 t}{t - t_\infty} \; . \tag{4.35}$$

Another way to rewrite Equations 4.19 and 4.20 was suggested by Alain Martin, in the form

$$\Delta E \Delta t = E_0 t_\infty \tag{4.36}$$

where $\Delta E = E_n - E_0$ and $\Delta t = t_n - t_\infty$.

Using Equation 4.33, we can write Et^n as $f(t) = \frac{E_0 t^{n+1}}{t - t_\infty}$. Figure 4.17 shows a plot of f for $n = 2$, $E_0 = 1$, and $t_\infty = 1$. One can notice that around the point of optimum $t_{opt} = (1 + \frac{1}{n})t_\infty = \frac{3}{2}$, function f is flat. More precisely, if $t = 0.9 * t_{opt} = 1.35 \Rightarrow f(t) \approx 1.04 * f_{opt}$, while if $t = 1.1 * t_{opt} = 1.65 \Rightarrow f(t) \approx 1.02 * f_{opt}$.

The observation on the flatness of the Et^n curve around the minimum should not be misinterpreted. In particular, this flatness does not imply that transistor sizing has little impact on the energy-delay efficiency of the final circuit. What it really means is that, assuming perfect sizing for a given target delay, i.e., minimal E for a given t, the exact choice of t around the optimum t_{opt} has little effect on Et^n. This is because what would be lost by, for example, running the circuit at t slightly higher than t_{opt} would be partly compensated by the savings in E. On the other hand, if the circuit does not run at minimal E given t, even a few unproperly sized transistors can compromise the final outcome. One can see this by considering a circuit with many concurrent cycles, originally all sized to run at the same speed. If a few transistors on one of these cycles were to be shrunk, the total energy E would not improve much, since the global weight in E of these transistors is small. However, the delay of the circuit t can go up significantly. As a result, the Et^n metric could increase dramatically.

4.4 Improvement in Et^2 due to Transistor Sizing

In theory, sizing for delay-only requires n going to infinity. In practice, n is chosen big, but finite. This is because the delay contribution of wire parasitics is not quite zero, since that would result in impractically large transistors. While the MiniMIPS microprocessor (an asynchronous version of a MIPS R3000) [33] was designed and sized for high speed, we estimate its optimization index n to be between 10 to 20. If the same design had been optimized for Et^2, the expected energy improvement would have been $\frac{11}{3}$ to $\frac{21}{3}$, while the speed slow-down between $\frac{7}{10}$ to $\frac{11}{15}$ of the original. This would have resulted in an overall Et^2 improvement of 200% to 350%. We would expect this improvement everywhere on the chip except for the cache core cells which are sized based on different considerations. The power consumption would decrease by 80% to 90%.

Chapter 5

Energy-Delay Optimization

5.1 Introduction

In Chapter 3 we have discussed the merits of the Et^2 metric proposed as an efficiency metric for VLSI computation [1], and we have motivated the introduction of the more general Et^n metric. In this chapter, we show that the Et^n metric for the energy-delay efficiency index $n \geq 0$ can capture *any* optimal trade-off—as will be defined later, not only the trade-off through voltage scaling, between the energy and the delay of a computation. For example, any problem of minimizing the energy of a system for a given target delay can be restated as minimizing Et^n for a certain n. A subset of the results to be presented here, have been published in [43, 44, 51].

In general, for a VLSI computation implementing a *given* algorithm, the faster the computation, the more energy it consumes. This observation points to the existence of a trade-off between the *good*ness of one property (delay) versus the *bad*ness of the other (consumed energy). The goal of our efficiency metric is to quantify such a trade-off. Certainly, there are computations that are both slower (*bad*) and consume more energy (*bad*) than some base case; however, those computations are not interesting implementations and will not be considered. Moreover, there are computations that are both faster (*good*) and more energy efficient (*good*) than some base case (for example the implementation of a transistor network in a newer technology). Again, these cases are of no interest to us since one will always prefer the *good-good* case and there is no trade-off. Later in this chapter we formalize the notion of trade-off and

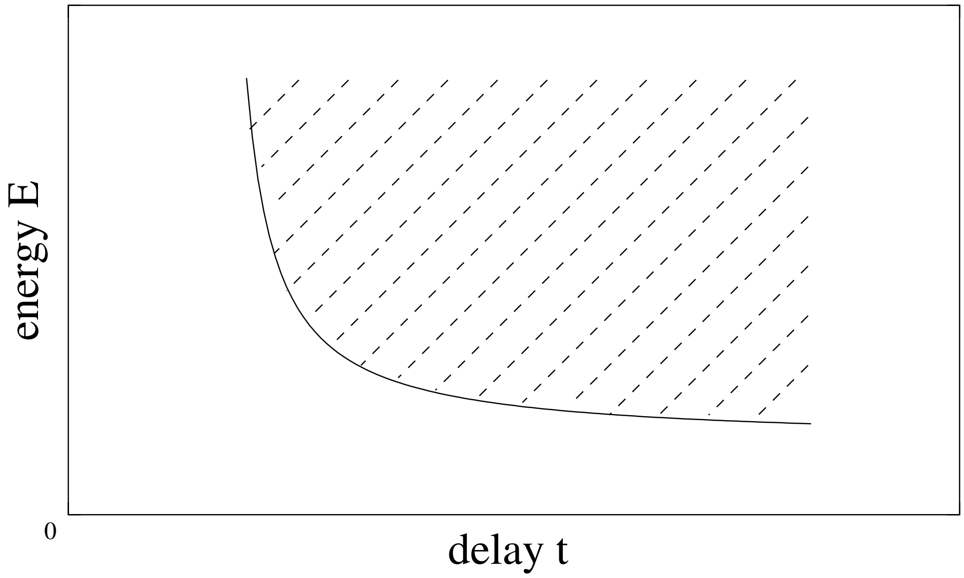

Figure 5.1: Optimal energy-delay trade-off.

the design parameters this trade-off applies to.

The *feasible energy-delay region* of a computation is defined by the set of (t, E) pairs with the property that, under the available design parameters, there exists a circuit implementation that operates with delay t and consumes energy E. In Figure 5.1, the shaded area represents such a feasible energy-delay region, when the design-parameter space is continuous. Any point on the boundary of the feasible energy-delay region represents an *optimal energy-delay trade-off* of the given computation, under the available design parameters. In case the design parameter space is noncontinuous, the boundary of a feasible energy-delay region is defined by a step function. More precisely, given the set of feasible energy-delay pairs (t_i, E_i), $i \in 0..m-1$, such that $t_i < t_{i+1}$ and $E_i > E_{i+1}$, the boundary of the feasible energy-delay region is defined by the function

$$E(t) = \begin{cases} \infty & \text{if } t < t_0 \\ E_i & \text{if } t_i \leq t < t_{i+1} \\ E_{m-1} & \text{if } t_{m-1} \leq t \end{cases}$$

A VLSI computation can be made slower or faster in several ways: at high level, by using a different architecture, and at low level, by choosing a different supply voltage or different device parameters of the transistor network. In general, any of those choices amounts to trading delay for energy and vice versa. For example, by operating a circuit at a higher supply voltage its delay decreases, while its energy consumption increases. Conversely, by operating the same circuit at a lower supply voltage, its delay increases, while its energy consumption decreases. Thus, voltage scaling is one way to trade delay for energy.

With an efficiency metric at hand, one can define the corresponding optimization problem as of finding a set of parameters in the available parameter space that optimizes the given metric. The parameter space is defined by the freedoms available to the designer. For example, if the operating voltage of the design can be varied, then voltage is part of the parameter space. On the other hand, if operating voltage is fixed, then it is not part of the parameter space. Throughout this text, when we are referring to the efficiency metric, we are implicitly referring to the corresponding optimization problem as well.

We define the efficiency metric in two seemingly different ways, and then we show how these definitions indeed lead to the same trade-off between energy and delay. Later in the chapter, we quantify the *good*ness of parallel and sequential VLSI computations using the efficiency metric. As an example, we look at the energy-delay efficiency of circuits optimized through transistor sizing and voltage scaling. We bound the energy and delay of the optimized circuits and we give necessary and sufficient conditions under which these bounds are reached. We also give necessary and sufficient conditions under which components of a design can be optimized independently, so as to yield global optimum when composed.

5.2 The Et^n Efficiency Metric

As mentioned in the introduction of this chapter, we are interested in defining an efficiency metric over a set of design parameters, parameters that create a trade-off between energy and delay. More precisely, if we define two functions one for energy $\mathcal{E}(\star) > 0$ and one for delay $\mathcal{T}(\star) > 0$, we are interested in studying them on the domain D that has the property that if $v, v + dv \in D$, $dv \neq 0$ then $\big(\mathcal{E}(v + dv) - \mathcal{E}(v)\big)\big(\mathcal{T}(v + dv) - \mathcal{T}(v)\big) < 0$; in other words, we are interested in the domain where evaluating $\mathcal{E}$ and $\mathcal{T}$ for a point $v + dv$ different from v results in increasing $\mathcal{E}$ while decreasing $\mathcal{T}$ or vice versa. Specifically, we are not interested in domains where $\big(\mathcal{E}(v + dv) - \mathcal{E}(v)\big)\big(\mathcal{T}(v + dv) - \mathcal{T}(v)\big) \geq 0$; since then there is no trade-off and the optimization becomes trivial.

We do not require D to be continuous. It is important that our functions are general enough to be definable on noncontinuous domains. This allows us to use them to reason about noncontinuous parameter spaces like different architectural implementations of a given algorithm or different decompositions of a high-level circuit specification. For example, if we want to evaluate an architectural trade-off between adders, the union of each different adder architecture (ripple-carry, carry-lookahead, carry-save, etc.) can form the domain D.

The first form we propose for an efficiency metric combines the energy consumed by the computation, and the delay (cycle time or latency) of the computation, in the form $\Theta_n(v) : D \to R_+, \Theta_n(v) = \mathcal{E}(v)\mathcal{T}(v)^n$, $n \geq 0$. When the domain D of variable v is clear or irrelevant, we will omit explicitly using v in Θ_n, $\mathcal{E}$ and $\mathcal{T}$. Furthermore, when we will use the value of $\mathcal{E}$ or $\mathcal{T}$ evaluated at a specific point v_0 that follows from the context, we will use E and t as a shorthand for $\mathcal{E}(v_0)$ and $\mathcal{T}(v_0)$, respectively.

It has been argued in [1] that Et^2—or with our notation Θ_2 $(n = 2)$—is independent, in first approximation, of the supply voltage. In other words, for $v = (..., V, ...) \in D$ (v includes the supply voltage V), $\Theta_2(..., V_1, ...) = \Theta_2(..., V_2, ...)\ \forall\ V_1, V_2$. Practically this means that, away from velocity saturation and threshold voltages, energy and delay can be freely exchanged through supply-voltage adjustment (within the feasible voltage range) while Θ_2 remains constant. We shall point out that if Θ_n is constant under the variation of a certain design parameter, E and t cannot be determined uniquely (as is the case with voltage scaling).

5.3 A Minimum-Energy Function

We can further refine the energy function $\mathcal{E}(\star)$ and the delay function $\mathcal{T}(\star)$ by defining two implicit functions: energy as a function of delay and delay as a function of energy. More precisely, we introduce a single-variable antimonotonic function by defining a *minimum-energy function* $E(t) : R_+ \rightarrow R_+$ that describes the minimum energy required for a system to run at a given delay t. Similarly, we introduce a single-variable antimonotonic function by defining a *minimum-delay function* $t(E) : R_+ \rightarrow R_+$ that describes the minimum delay of a system that consumes energy E. Through these two functions, we have abstracted away the original domain D of $\mathcal{E}(\star)$ and $\mathcal{T}(\star)$; however, it should be noted that the choice of D affects the expressions of $E(t)$ and $t(E)$. Furthermore, these two functions depend at high level on the particular computation being implemented and at low level on the circuits and device parameters used.

It should be noted that both of these functions are well defined (in the mathematical sense). In particular, for the minimum-energy function even though there could be several ways to achieve a delay t, yielding several—possibly different—energy values E, by picking the smallest of them we force this relation to take a unique value for each input t, and thus become a well-defined function. A similar argument applies to the minimum-delay function.

The related optimization problem consists of finding these functions (or their value) over parts or the entire domain of definition.

It can be shown that the minimum-energy function and minimum-delay function represent the same implicit relation between E and t. More precisely, the minimum-energy function and the minimum-delay function are the inverse of each other, i.e., $E \circ t = t \circ E = I$, where I is the identity function. It turns out that the minimum-energy function lends itself better to mathematical manipulation, given the fact that many compositional properties result in relations in terms of the delay t. For this reason, we will use only the minimum-energy function in our reasoning, but one should remember that the same argument can be stated in terms of the minimum-delay function.

In the next section, we give two examples of a minimum-energy function and of a minimum-delay function for two particular types of optimizations.

5.3.1 A Minimum-Energy Function for Transistor Sizing

As studied in Chapter 4, transistor sizing provides a set of transistor dimensions for a circuit such that a given metric is optimized. It has been shown in Chapter 4 that for optimal transistor sizing for Et^n, the consumed energy is

$$E \approx (1+n)E_0$$

and the delay is

$$t \approx \left(1+\frac{1}{n}\right)t_\infty,$$

where E_0 is the energy due to the total switched wire capacitance of the circuit and t_∞ is the lower bound on the achievable delay of the circuit. It was also pointed out that if we rewrite these two equations with E a function of t—by eliminating n—we get the following function:

$$E(t) = \frac{E_0 t}{t - t_\infty} \ . \tag{5.1}$$

Similarly, one can express t as function of E and get the minimum-delay function for optimal transistor sizing

$$t(E) = \frac{t_\infty E}{E - E_0}. \tag{5.2}$$

In the context of transistor sizing, we define the *asymptotic power* as

$$\hat{P} = \frac{E_0}{t_\infty}.$$

The energy and delay used in defining the asymptotic power are of course not simultaneously attainable; yet the asymptotic power is related to the actual circuit power. More precisely, the power consumption of a circuit optimized for Et^n through

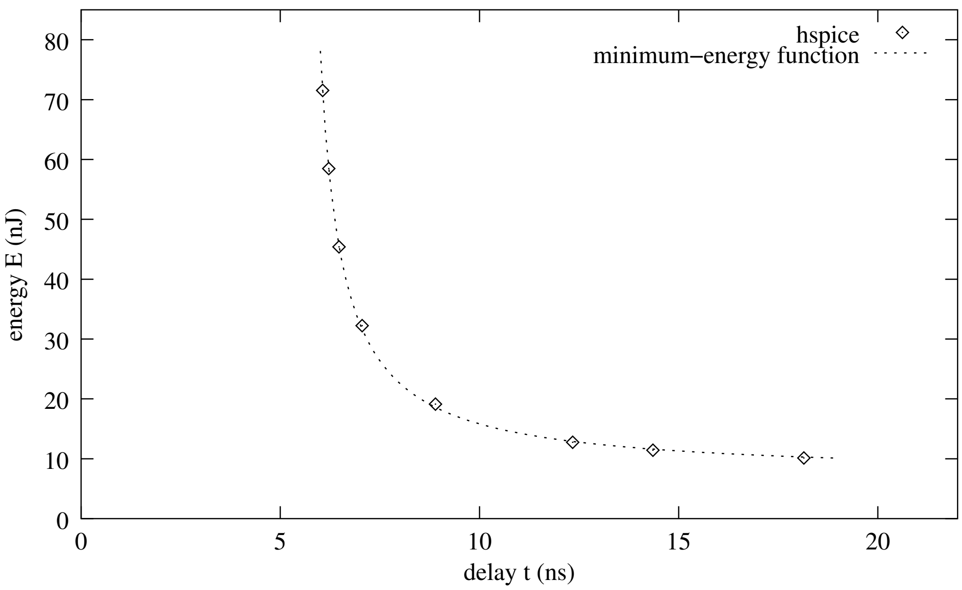

Figure 5.2: The minimum-energy function in practice.

transistor sizing is

$$P = \frac{E}{t} = n\frac{E_0}{t_\infty} = n\hat{P}. \tag{5.3}$$

This relationship shows that the power consumption increases linearly with the optimization index n. In particular, the power consumption of a circuit optimized for Et is half that of the same circuit optimized for Et^2. Equation 5.3 also relates the optimization index n to the ratio of the actual power consumption to the asymptotic power of the circuit.

The validity of Equation 5.1 has been checked in practice. A ring of operators optimized for different values of n was considered as a test vehicle. The results given by the minimum-energy function were compared against `hspice` simulations conducted in HP's 0.6μm process. The results of this experiment are shown in Figure 5.2.

5.3.2 A Minimum-Energy Function for Voltage Scaling

Consider a system in which the supply voltage is the only optimization parameter. In this case, assuming $E \approx CV^2$, $t \approx k/V$ and $\Theta = Ck^2$,

$$E(t) = \frac{\Theta}{t^2} \tag{5.4}$$

and

$$t(E) = \frac{\sqrt{\Theta}}{\sqrt{E}}. \tag{5.5}$$

It should be noted that through the single parameter t—using the minimum-energy function—one can quantify the optimal energy-delay trade-off.The same holds for the single parameter E using the minimum-delay function. But this outcome was already achieved by the Et^n metric. For this reason, we would like to know if these new functions are fundamentally different from our previous Et^n metric. More precisely, if a system were to be optimized using one of these functions or the Et^n metric, would that result in different values of the optimal E and t?

5.4 Metric Equivalence

The answer to the previous question is given by the following:

Theorem 7 *Given an energy-delay optimization of a computation, the problem specified as "find $E_0 = \min E$ given t_0" is equivalent to "find the values of E and t that minimize Et^{n_0} for $n_0 = -\frac{t_0}{E_0}\frac{dE}{dt}(t_0)$"—when such a solution is well defined and unique. Similarly, the problem specified as "find $t_0 = \min t$ given E_0" is equivalent to "find the values of E and t that minimize Et^{n_0} for $n_0 = -\frac{t_0}{E_0}\frac{1}{\frac{dt}{dE}(E_0)}$"—when such a solution is well defined and unique.*

Proof. We prove the equivalence of the two statements by showing that one implies the other and vice versa. First, assume that we are solving "find $E_0 = \min E$ given t_0". Minimizing Et^n for the given t_0 implies—for any n—finding the minimum E given t_0—which in this case is E_0. Second, assume we are solving "find the values of

E and t that minimize Et^{n_0} for $n_0 = -\frac{t_0}{E_0}\frac{dE}{dt}(t_0)$". With the help of the minimum-energy function, we can write Et^n as a single-variable function in t. This function is minimized—given that the minimum-energy function is antimonotonic—where

$$\frac{d(Et^n)}{dt} = 0 \Rightarrow \frac{dE}{dt}{t'_0}^n + nE'_0{t'_0}^{n-1} = 0 \Rightarrow \frac{t'_0}{E'_0}\frac{dE}{dt} + n = 0,$$

but by hypothesis

$$n = n_0 = -\frac{t_0}{E_0}\frac{dE}{dt}(t_0) \Rightarrow \frac{t'_0}{E'_0}\frac{dE}{dt}(t'_0) = \frac{t_0}{E_0}\frac{dE}{dt}(t_0).$$

Thus, we found E'_0 and t'_0 that optimize Et^{n_0} such that $\frac{t'_0}{E'_0}\frac{dE}{dt}(t'_0) = \frac{t_0}{E_0}\frac{dE}{dt}(t_0)$. Clearly, E_0 and t_0 are solutions of this equality. However, by hypothesis, the solution to the minimization problem is unique $\Rightarrow t'_0 = t_0 \Rightarrow E'_0 = E(t'_0) = E(t_0) = E_0$ as well. Thus, when optimizing Et^n with $n = n_0$, if a unique solution exists, we find it to be the required E_0 and t_0. $\square$

The requirement on the solution to be well defined requires $E(t)$ or $t(E)$ to be differentiable around t_0 and E_0, respectively. In particular, this requirement is not met in the case of a noncontinuous parameter space. In such a situation n_0 is no longer unique, but corresponds to an interval of values, all yielding the same energy-delay pair.

The uniqueness of the solution minimizing Et^{n_0} is important for un-ambiguously determining E_0 and t_0. It could be the case that there are several (E,t) pairs—including (E_0,t_0)—that minimize Et^{n_0}. In particular, the metric Et^{n_0} accepts infinitely many (E,t) pairs as solution if $E(t) = ct^{-k}$, $c > 0$, $k > 0$. If more than one solution exists, finding the solution pair (E_0,t_0) reduces to choosing from the solution pairs (E,t) the one that has $t = t_0$.

Theorem 7 tells us that, for a given system to be optimized in terms of both E and t, one can pose the optimization problem either in terms of an energy-delay efficiency index n, or a desired delay target t and obtain as result the same optimal values of E and t. This seemingly harmless result has the great benefit of allowing the application of the results developed for Et^n optimization in Chapter 4 and [44] to other types of energy-delay optimizations—optimizations where either the target energy or the target delay are fixed. As a concrete example consider finding the optimal transistor sizes of a circuit, so as to achieve delay t_0 for minimal energy. Given t_0, one can find the corresponding energy-delay optimization index n_0. With n_0 at hand—using the formulas developed in Chapter 4 for optimal Et^n transistor sizing—one can directly generate the transistor sizes that achieve delay t_0 for minimal energy consumption.

In the next section, we apply the concept of metric equivalence to the parallel and sequential composition of circuits.

5.5 Composition

It is often the case, in practice, that one wishes to decompose the design of a complex system into a set of relatively independent subsystems, which then can be independently designed and implemented. If the optimization problem is defined globally using any of the parameters n, t or E, it is not immediately clear how subsystems of the original design should be optimized in terms of n, t or E, so as to achieve global minimum when the subsystems are composed.

The two major composition techniques used in VLSI design are parallel composition and sequential composition. In the following, we show how the energy-delay efficiency metrics have to be applied to subcomponents, so as to yield global minimum when composed in parallel or serially.

We will assume that each subsystem S_i has its own optimization index n_i (to be determined), and its own minimum-energy function $E_i(t)$.

5.5.1 Parallel Composition

Let us consider the parallel composition of m subsystems S_i. Let us assume a computation that runs in parallel all S_is to completion before starting a new computation. We want to know at what t_i to run S_i or which n_i to optimize S_i for, so as to obtain the best E for a given t or to minimize Et^n for a given n, respectively.

Let us consider the first case, i.e., when we would like to find the minimal E for a given t. Knowing that S_i will complete after delay $t = \max_{1 \leq i \leq m}(t_i)$, there is no reason to run any of the subsystems faster than t, in other words $t_i = t$, $\forall i \in 1..m$. Under these circumstances, the energy consumption of S_i is $E_i(t)$ and the total energy consumption is $E = \sum_{i=1}^{m} E_i(t)$. Using Theorem 7, we can determine

$$n = -\left(\sum_{i=1}^{m} \frac{dE_i(t)}{dt}\right) \frac{t}{\sum_{i=1}^{m} E_i(t)}$$

and

$$n_i = -\frac{dE_i(t)}{dt} \frac{t}{E_i(t)}.$$

On the other hand, if n is given—noting again that $t_i = t$, $\forall i \in 1..m$—we can use the minimum-energy functions of S_i to write Et^n as a single-variable expression in t. If this single-variable function is continuous and differentiable, we find its minimum using the methods of mathematical analysis. If Et^n is not continuous—because the underlying domain is not continuous—one can still find the minimum by enumeration. Once the point of minimum is known, all other unknowns can be determined the same way as in the previous case.

In the following, we will consider two examples of energy-delay optimization in the context of parallel composition.

5.5.1.1 Parallel Composition and Transistor Sizing

The first example we look at is the energy-delay optimization through transistor sizing of a system consisting of the parallel composition of m subsystems S_i. Given the nature of the optimization parameter (transistor sizing), the minimum-energy function of S_i is $E_i(t) = E_{0i}t/(t - t_{\infty i})$, where E_{0i} is the energy due to the total switched wire capacitance of subsystem S_i and $t_{\infty i}$ is the lower bound on the achievable delay of subsystem S_i.

Lets consider the first instance of the optimization problem, namely when t is given and we want to find the minimum E that achieves this t. Based on the previous discussion on parallel composition, using the minimum-energy functions we can compute E directly as

$$E = \sum_{i=1}^{m} E_i(t) = \sum_{i=1}^{m} \frac{E_{0i}t}{t - t_{\infty i}} \ . \tag{5.6}$$

Then, we find

$$n = -t\frac{\sum_{i=1}^{m} E_i'(t)}{\sum_{i=1}^{m} E_i(t)} = \frac{\sum_{i=1}^{m} \frac{E_{0i}t_{\infty i}}{(t-t_{\infty i})^2}}{\sum_{i=1}^{m} \frac{E_{0i}}{t-t_{\infty i}}} \tag{5.7}$$

and

$$n_i = -t\frac{E_i'(t)}{E_i(t)} = \frac{t_{\infty i}}{t - t_{\infty i}} \ . \tag{5.8}$$

On the other hand, if we are given n and asked to find E and t that optimize Et^n, we use Equation 5.6 to write

$$Et^n = \left(\sum_{i=1}^{m} \frac{E_{0i}t}{t - t_{\infty i}}\right) t^n$$

and then minimize this single-variable function of t. It follows that,

$$\min\,(Et^n) \Rightarrow \frac{d(E(t)t^n)}{dt} = 0 \Rightarrow \sum_{i=1}^{m} \frac{E_{0i}t_{\infty i}}{(t - t_{\infty i})^2} = n\left(\sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}}\right).$$

As a consequence, t can be computed as the solution to this $2m-1$ order polynomial equation. With the computed t, E and E_i follow. Lastly, we would like to find what n_i to optimize S_i for, so as to yield global Et^n optimality. We can obtain this by computing n_i directly using Equation 5.8.

With the help of the minimum-energy function and the energy-delay efficiency index, we can infer several properties of parallel composition optimized through transistor sizing without the need to solve a $2m-1$ order polynomial equation to compute t. These properties are presented next.

Property 1 *For the parallel composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$\left(1 + \frac{1}{n_i}\right)t_{\infty i} = \left(1 + \frac{1}{n_j}\right)t_{\infty j} \ \forall i, j \in 1..m.$$

Proof. It follows directly from formula $t_i = (1 + 1/n_i)t_{\infty i}$ for S_i and the fact that $t_i = t_j\ \forall i, j \in 1..m$. $\square$

Property 1 relates the optimization indexes of the components to their respective t_∞s and will be used to prove other properties.

Theorem 8 *For the parallel composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$n = n_i\ \forall i \in 1..m \iff t_{\infty i} = t_{\infty j}\ \forall i, j \in 1..m.$$

Proof. Follows directly form Property 1. $\square$

Theorem 8 tells us that the parallel components of a system can be optimized independently for Et^n, yielding the global optimum when composed, if and only if all $t_{\infty i}$s are equal. Otherwise, even if one is globally optimizing for n, locally one needs to be able to optimize for $n_i \neq n$.

Theorem 9 *For the parallel composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$E = \frac{1}{n}\sum_{i=1}^{m} n_i E_i$$

or equivalently

$$\sum_{i=1}^{m} (n_i - n)(1 + n_i)E_{0i} = 0$$

or equivalently

$$P = \frac{1}{n}\sum_{i=1}^{m} n_i^2 \hat{P}_i.$$

Proof. From Property 1 we have

$$n_i = \frac{1}{\left(1 + \frac{1}{n_1}\right)\frac{t_{\infty 1}}{t_{\infty j}} - 1}$$

$$\Rightarrow \frac{dn_i}{dn_1} = \frac{\frac{1}{n_1^2}\frac{t_{\infty 1}}{t_{\infty i}}}{\left(\left(1+\frac{1}{n_1}\right)\frac{t_{\infty 1}}{t_{\infty i}}-1\right)^2} = \frac{n_i^2}{n_1^2}\frac{t_{\infty 1}}{t_{\infty i}}.$$

One can find n_1 that minimizes $f(n_1) = Et^n$, where $E = \sum_{i=1}^{m}(n_i+1)E_{0i}$ and $t = (1+\frac{1}{n_1})t_{\infty 1}$ as follows:

$$\begin{aligned}
\frac{df}{dn_1} = 0 \Rightarrow nE &= n_1(n_1+1)\Big(E_{01} + \sum_{i=2}^{m} E_{0i}\frac{dn_i}{dn_1}\Big) \\
&= n_1(n_1+1)\Big(E_{01} + \sum_{i=2}^{m} E_{0i}\frac{n_i^2}{n_1^2}\frac{t_{\infty 1}}{t_{\infty i}}\Big) \\
&= n_1(n_1+1)E_{01} + \frac{n_1+1}{n_1}t_{\infty 1}\sum_{i=2}^{m} E_{0i}\frac{n_i^2}{t_{\infty i}} \\
&= n_1(n_1+1)E_{01} + \sum_{i=2}^{m} n_i(n_i+1)E_{0i} \\
&= \sum_{i=1}^{m} n_i(n_i+1)E_{0i} \\
\Rightarrow E &= \frac{1}{n}\sum_{i=1}^{m} n_i(n_i+1)E_{0i}
\end{aligned}$$

or using formula $E_i = (1+n_i)E_{0i}$ we have

$$E = \frac{1}{n}\sum_{i=1}^{m} n_i E_i. \tag{5.9}$$

Equation 5.9 can be rewritten using the formula $E_i = (1+n_i)E_{0i}$ as

$$\begin{aligned}
\sum_{i=1}^{m} n_i(1+n_i)E_{0i} &= n\sum_{i=1}^{m}(1+n_i)E_{0i} \\
\Rightarrow \sum_{i=1}^{m}(n_i-n)(1+n_i)E_{0i} &= 0. \qquad (5.10)
\end{aligned}$$

Similarly, the total power of the system can be written as

$$P = \frac{E}{t} = \frac{\frac{1}{n}\sum_{i=1}^{m} n_i E_i}{t} = \frac{1}{n}\sum_{i=1}^{m} n_i P_i = \frac{1}{n}\sum_{i=1}^{m} n_i^2 \hat{P}_i. \qquad \square$$

Theorem 9—in its first form—relates the total consumed energy, as defined by Equation 5.6, to the optimization indexes of the components and their respective energies, or—using the second form—it relates the optimization indexes to the minimal energies E_{0i} of the components. The last form of Theorem 9 relates the total power of the system to the optimization indexes and asymptotic powers of its components.

Consider two subsystems S_1 and S_2 ($m = 2$). We can show that, for any given $n_1 > 0$, $n_2 > 0$ and $n > 0$, if $sign(n_1 - n) = sign(n - n_2)$, one can always construct a parallel system composed of subsystems S_1 and S_2, such that subsystem S_1 is optimized for Et^{n_1}, subsystem S_2 for Et^{n_2} while the entire system is optimized for Et^n. For example, if $n_1 = 1$, $n_2 = 3$ and $n = 2$ then one corresponding system could have $t_{\infty 1} = 2$, $E_{01} = 2$, $t_{\infty 2} = 3$, and $E_{02} = 1$.

Theorem 10 *For the parallel composition of m systems S_i (E_{0i}, $t_{\infty i}$), if the composed system is optimized for Et^n through transistor sizing, then*

$$E \leq (n+1) \sum_{i=1}^{m} E_{0i}$$

with equality if and only if all $t_{\infty i}$s are equal.

Proof. The optimal Et^n of this composed system is reached for E and t that satisfy

$$\frac{d(E(t)t^n)}{dt} = 0 \ ,$$

which is achieved when

$$(n+1) \sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} = t \sum_{i=1}^{m} \frac{E_{0i}}{(t - t_{\infty i})^2} \ . \qquad (5.11)$$

We may now invoke the Cauchy-Schwarz inequality $\left(\sum l_i r_i\right)^2 \leq \sum l_i^2 \sum r_i^2$, where equality holds if and only if l_i/r_i has the same value for all i. If we substitute $l_i \leftarrow \frac{\sqrt{E_{0i}}}{t - t_{\infty i}}$ and $r_i \leftarrow \sqrt{E_{0i}}$, we get that

$$\left(\sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \right)^2 \leq \sum_{i=1}^{m} \frac{E_{0i}}{(t - t_{\infty i})^2} \sum_{i=1}^{m} E_{0i} \qquad (5.12)$$

with equality if and only if all $t_{\infty i}$s are equal. Using Equation 5.11, we replace $\sum \frac{E_{0i}}{(t-t_{\infty i})^2}$ with $\frac{(n+1)}{t} \sum \frac{E_{0i}}{t-t_{\infty i}}$ in Equation 5.12, and we get the following result:

$$\left(\sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \right)^2 \leq \frac{(n+1)}{t} \sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \sum_{i=1}^{m} E_{0i} \ .$$

By Equation 5.6 then,

$$E(t) = t \sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \leq (n+1) \sum_{i=1}^{m} E_{0i} \ .$$

And therefore

$$E \leq (n+1) \sum_{i=1}^{m} E_{0i} \ . \qquad \square$$

In Theorem 10, equality holds if and only if all $t_{\infty i}$s are equal; in this situation, we also have that $E_i = (n+1)E_{0i}$. In practice, generally all bits within a datapath pipeline are identical and different datapath pipelines have similar structure, thus it could be assumed that—for most well-designed circuits—the cycles formed by these bits have very similar (or identical) t_{∞}s. So, we should expect that usually $E \approx (n+1) \sum E_{0i}$. The existence of some potentially faster cycles (due possibly to buffers or fast control) will not have a significant impact on the global speed and energy of the system.

Let us consider a numerical example to illustrate Theorem 10. If $n = 2$, $m = 2$, $t_{\infty 1} = 1$, $t_{\infty 2} = 1.2$ and $E_{01} = E_{02} = 10$ then $t = 1.70$ and $E = 58.37$ ($E = E_1 + E_2 = 24.31 + 34.06$). Notice that $(1 + \frac{1}{n})t_{\infty 1} = 1.5$, $(1 + \frac{1}{n})t_{\infty 2} = 1.8$, $(n+1)E_{01} = 30$ and $(n+1)E_{02} = 30$. Thus, the optimal running speed of the system is between $(1+\frac{1}{n})t_{\infty 1}$ and $(1 + \frac{1}{n})t_{\infty 2}$ (as claimed by the next theorem). The way t is reached is by running the faster system S_1 slower than its own speed target $(1+\frac{1}{n})t_{\infty 1}$ — thus saving energy (from $(n+1)E_{01} = 30$ to $E_1 = 24.31$), and running the slower system S_2 faster than its own speed target $(1 + \frac{1}{n})t_{\infty 2}$ — thus spending more energy (from $(n+1)E_{02} = 30$ to $E_2 = 34.06$). What Theorem 10 is saying is that the energy trade-off between the slow and the fast systems is done such that only part of the energy saved by slowing down S_1 is spent on speeding up S_2; i.e., $(n+1)E_{01} + (n+1)E_{02} = 60$ is always greater that $E = 58.37$.

Theorem 11 *For the parallel composition of m systems S_i (E_{0i}, $t_{\infty i}$), if the composed system is optimized for Et^n through transistor sizing and there exists $j \in 1..m$ such that $|\frac{E_{0i}}{E_{0j}}| < \varepsilon$, $\forall\ \varepsilon > 0$ and there exists $\delta > 0$ such that $|(1+\frac{1}{n})t_{\infty j} - t_{\infty i}| > \delta$, $\forall$ $i \in 1..m, i \neq j$ then, $t = (1+\frac{1}{n})t_{\infty j}$ and $E = (n+1)E_{0j}$.*

Proof. Assume $\zeta > 0$ such that $|t - t_{\infty j}| > \zeta$, $\forall\ i \in 1..m$. Then, by dividing both sides of Equation 5.11 with E_{0j} and choosing a small enough $\varepsilon > 0$ we can cancel all terms except two; it follows that

$$(n+1)\frac{E_j}{t - t_{\infty j}} = t\frac{E_{0j}}{(t - t_{\infty j})^2} \Rightarrow t = \frac{n+1}{n}t_{\infty j}.$$

By writing Equation 5.6 as

$$E = E_{0j}\left(t\sum_{i=1}^{m}\frac{E_{0i}}{E_{0j}}\frac{1}{t - t_{\infty i}}\right)$$

for a small enough $\varepsilon > 0$, we get $E = E_{0j}t\frac{1}{t-t_{\infty j}}$ and using $t = (1+\frac{1}{n})t_{\infty j}$ it follows that $E = (n+1)E_{0j}$. We still have to validate our original assumption about ζ. By hypothesis, $\exists\ \delta > 0$ such that $|(1+\frac{1}{n})t_{\infty j} - t_{\infty i}| > \delta$; thus, for the t we found $\exists\ \delta > 0$ such that $|t - t_{\infty i}| > \delta\ \forall\ i \in 1..m$. By choosing $\delta = \zeta$ our initial assumption is verified and the proof is complete. $\square$

Theorem 11 tells us that the composed system runs close to the target speed $\frac{n+1}{n}t_{\infty j}$ of the component (S_j) that consumes the most energy E_{0j} and that the overall energy consumption is close to $(n+1)E_{0j}$. In practice, as already mentioned, generally all bits within a datapath pipeline are identical and different datapath pipelines have similar structure, thus it could be assumed that the cycles formed by these bits have very similar (or identical) t_∞s. These t_∞ cycles will generate a dominant term in the energy expression (since most of the energy is consumed in the datapath) and will bound the optimal running speed of the system to $(1+\frac{1}{n})t_\infty$ and its energy consumption to $(n+1)E_{0j}$. Theorem 11 allows us to use—under certain circumstances—the simpler Equations 4.19 and 4.20 in our global performance analysis.

Theorem 12 *For the parallel composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$\max\Big(\max_{i\in 1..m} t_{\infty i}, \Big(1+\frac{1}{n}\Big)\min_{i\in 1..m} t_{\infty i}\Big) \le t \le \Big(1+\frac{1}{n}\Big)\max_{i\in 1..m} t_{\infty i}$$

with equality if and only if all $t_{\infty i}$s are equal.

Proof. By contradiction. Assume that

$$t < \frac{n+1}{n} t_{\infty i}\ \forall i \in 1..m \Rightarrow (n+1)\frac{E_{0i}}{t - t_{\infty i}} > t\frac{E_{0i}}{(t-t_{\infty i})^2}\ \forall i \in 1..m$$

$$\Rightarrow (n+1)\sum_{i=1}^{m}\frac{E_{0i}}{t-t_{\infty i}} > t\sum_{i=1}^{m}\frac{E_{0i}}{(t-t_{\infty i})^2} \Rightarrow \text{contradiction with Equation 5.11.}$$

Similarly, if we assume $t > \frac{n+1}{n} t_{\infty i}\ \forall i \in 1..m$, then

$$(n+1)\sum_{i=1}^{m}\frac{E_{0i}}{t-t_{\infty i}} < t\sum_{i=1}^{m}\frac{E_{0i}}{(t-t_{\infty i})^2} \Rightarrow \text{contradiction with Equation 5.11.}$$

As a result, $\exists\ i, j \in 1..m$ such that $\frac{n+1}{n} t_{\infty i} \le t \le \frac{n+1}{n} t_{\infty j}$ but $\frac{n+1}{n}\min_{i\in 1..m} t_{\infty i} \le \frac{n+1}{n} t_{\infty i}\ \forall i \in 1..m$ and $\frac{n+1}{n} t_{\infty j} \le \frac{n+1}{n}\max_{i\in 1..m} t_{\infty i}\ \forall i \in 1..m \Rightarrow \frac{n+1}{n}\min_{i\in 1..m} t_{\infty i} \le t \le \frac{n+1}{n}\max_{i\in 1..m} t_{\infty i}$. Furthermore, given the domain of $E(t)$, $\max_{i\in 1..m} t_{\infty i} \le t \Rightarrow$

$$\max\Big(\max_{i\in 1..m} t_{\infty i}, \frac{n+1}{n}\max_{i\in 1..m} t_{\infty i}\Big) \le t \le \frac{n+1}{n}\max_{i\in 1..m} t_{\infty i}. \qquad \square$$

In Theorem 12, equality holds if and only if all $t_{\infty i}$s are equal; in this situation, we also have that $t = (1+\frac{1}{n})t_\infty$. Theorem 12 bounds the optimal running speed of a circuit between its scaled $(1+\frac{1}{n})\times$ slowest cycle ($\min_{i\in 1..m} t_{\infty i}$) and fastest cycle ($\max_{i\in 1..m} t_{\infty i}$). If those cycles are close to each other—as is the case in a balanced design—both bounds on t are tight. If n increases without bound then $\max_{i\in 1..m} t_{\infty i} \le t \le \max_{i\in 1..m} t_{\infty i} \Rightarrow t = \max_{i\in 1..m} t_{\infty i}$, i.e., the delay of a circuit optimized for speed only is limited by the delay of its critical cycle; an expected result for speed-only optimization.

Based on Theorems 10 and 12, we can find an upper bound on the minimum Et^n, as suggested by the following theorem:

Theorem 13 *For the parallel composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$\min(Et^n) \leq (n+1)\Big(\sum_{i=1}^{m} E_{0i}\Big)\bigg(\Big(1+\frac{1}{n}\Big)\max_{i\in 1..m} t_{\infty i}\bigg)^n$$

with equality if and only if all $t_{\infty i}$s are equal.

Proof. Follows directly form Theorem 10 and Theorem 12.

As mentioned earlier, in a well-designed system, the $t_{\infty i}$s are close to each other; therefore, the upper bound given by Theorem 13 is tight and can be used as a good approximation of the actual minimal Et^n.

Theorem 14 *For the parallel composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$P \leq n\sum_{i=1}^{m} \hat{P}_i$$

with equality if and only if all $t_{\infty i}$s are equal.

Proof. We invoke again the Cauchy-Schwarz inequality, substitute $l_i \leftarrow \frac{\sqrt{E_{0i}t_{\infty i}}}{t-t_{\infty i}}$ and $r_i \leftarrow \sqrt{\frac{E_{0i}}{t_{\infty i}}}$ and we get that

$$\left(\sum_{i=1}^{m} \frac{E_{0i}}{t-t_{\infty i}}\right)^2 \leq \sum_{i=1}^{m} \frac{E_{0i}t_{\infty i}}{(t-t_{\infty i})^2} \sum_{i=1}^{m} \frac{E_{0i}}{t_{\infty i}} \tag{5.13}$$

with equality if and only if all $t_{\infty i}$s are equal. Equation 5.13 can be written as

$$\sum_{i=1}^{m}\left(\frac{t_{\infty i}^2}{(t-t_{\infty i})^2}\frac{E_{0i}}{t_{\infty i}}\right) \leq \left(\frac{\sum_{i=1}^{m}\frac{E_{0i}t_{\infty i}}{(t-t_{\infty i})^2}}{\sum_{i=1}^{m}\frac{E_{0i}}{t-t_{\infty i}}}\right)^2 \sum_{i=1}^{m}\frac{E_{0i}}{t_{\infty i}}. \tag{5.14}$$

Using Equations 5.7 and 5.8, Equation 5.14 becomes

$$\sum_{i=1}^{m} n_i^2 \hat{P}_i \leq n^2 \sum_{i=1}^{m} \hat{P}_i. \tag{5.15}$$

Then, using the third form of Theorem 9, we get

$$P \leq n \sum_{i=1}^{m} \hat{P}_i, \tag{5.16}$$

with equality if and only if all $t_{\infty i}$s are equal. □

Theorem 14 bounds the total power consumption of the system, as a function of n and the sum of asymptotic powers. This bound is tight as long as the $t_{\infty i}$s are close to each other, which is the case in a well designed circuit.

5.5.1.2 Parallel Composition and Voltage Scaling

The second example we look at is the energy-delay optimization through voltage scaling of a system consisting of the parallel composition of m subsystems S_i. Given the nature of the optimization parameter (voltage scaling), the minimum-energy function of S_i is $E_i(t) = \Theta_i/t^2$, where Θ_i is a voltage independent constant.

Lets consider the first instance of the optimization problem, namely when t is given and we want to find the minimum E that achieves this t. Based on the previous discussion on parallel composition, using the minimum-energy function we can compute E directly as

$$E = \sum_{i=1}^{m} E_i(t) = \sum_{i=1}^{m} \Theta_i/t^2.$$

Then, we can find

$$n = -t\frac{\sum_{i=1}^{m} dE_i/dt}{\sum_{i=1}^{m} E_i} = t\frac{2\sum_{i=1}^{m} \Theta_i/t^3}{\sum_{i=1}^{m} \Theta_i/t^2} = 2,$$

and

$$n_i = -t\frac{dE_i/dt}{E_i} = t\frac{2\Theta_i/t^3}{\Theta_i/t^2} = 2;$$

in other words

$$n = n_i = 2.$$

This property tells us that when only voltage scaling is applied to optimize a parallel composition, it is sufficient to be able to optimize only for $n = 2$.

On the other hand, if we are given n and asked to find E and t that optimize Et^n, we write

$$Et^n = \Big(\sum_{i=1}^{m} \Theta_i/t^2\Big)t^n = \Big(\sum_{i=1}^{m} \Theta_i\Big)t^{(n-2)}$$

and then minimize this single-variable function of t. We notice that if $n = 2$ then $Et^n = \sum_{i=1}^{m} \Theta_i = constant$. This implies that the metric Et^n is optimal for any voltage. One can compute E and E_i by picking any t. Let us define t_{min} and t_{max} the smallest and the largest delays such that for $t \in [t_{min}, t_{max}]$ the relations $E_i(t) = \Theta_i/t^2$ hold in practice. If $n \neq 2$ and $t \in [t_{min}, t_{max}]$ then $\min Et^n \Rightarrow t = t_{min}$ if $n > 2$ and $t = t_{max}$ if $n < 2$. In other words, if $n > 2$ one will run the system at the highest possible operation voltage, while if $n < 2$ one will run the same system at the lowest possible operation voltage. Using this t, E and E_i follow. Lastly, it follows that we need $n = n_i = 2$ to yield global Et^n optimality.

5.5.2 Sequential Composition

Let us now consider the sequential composition of m subsystems S_i. Let us assume a sequential computation that runs S_1 to completion, then S_2 to completion, all the way to the completion of S_m; we assume the delay between the end of S_i and the start of S_{i+1} to be negligible. Again, we want to know at what t_i to run S_i or which n_i to optimize S_i for, so as to obtain the best E for a given t or to minimize Et^n for a given n, respectively.

Theorem 15 *For the sequential composition of m systems S_i, if the composed system is optimized for minimum energy E given a delay t or for Et^n, then*

$$\frac{dE_i(t_i)}{dt_i} = \frac{dE_j(t_j)}{dt_j} \;\forall i, j \in 1..m.$$

Proof. When the minimal energy E is to be computed for a fixed t, given that $t = \sum_{i=1}^{m} t_i$, we want to minimize

$$E(t_2, t_3, ..., t_m) = E_1(t - t_2 - t_3... - t_m) + E_2(t_2) + ... + E_m(t_m).$$

This function reaches its minimum where

$$\begin{aligned}
\frac{dE}{dt_i} = 0 \ \forall i \in 2..m \ \Rightarrow \ & \frac{dE_1(t - t_2 - t_3... - t_m)}{dt_i} + \frac{dE_i(t_i)}{dt_i} = 0 \\
\Rightarrow \ & \frac{dE_1(t - t_2 - t_3... - t_m)}{d(t - t_2 - t_3... - t_m)} \frac{d(t - t_2 - t_3... - t_m)}{dt_i} + \frac{dE_i(t_i)}{dt_i} = 0 \\
\Rightarrow \ & -\frac{dE_1(t_1)}{dt_1} + \frac{dE_i(t_i)}{dt_i} = 0 \\
\Rightarrow \ & \frac{dE_1(t_1)}{dt_1} = \frac{dE_i(t_i)}{dt_i} \ \forall i \in 2..m
\end{aligned}$$

When the optimization is specified in terms of n, we can write the latency of the composed system as $t = \sum_{i=1}^{m} t_i$, and its energy as $E = \sum_{i=1}^{m} E_i(t_i)$. Thus, we want to minimize

$$f(t_1, t_2, ..., t_m) = \left(\sum_{i=1}^{m} E_i(t_i)\right)\left(\sum_{i=1}^{m} t_i\right)^n.$$

f reaches its minimum where

$$\frac{\partial f}{\partial t_i} = 0 \ \forall i \in 1..m \Rightarrow \frac{dE_i(t_i)}{dt_i}\left(\sum_{i=1}^{m} t_i\right) + n\left(\sum_{i=1}^{m} E_i(t_i)\right) = 0 \ \forall i \in 1..m$$

$$\Rightarrow \frac{dE_i(t_i)}{dt_i} = -\frac{n\left(\sum_{i=1}^{m} E_i(t_i)\right)}{\sum_{i=1}^{m} t_i} \ \forall i \in 1..m \Rightarrow \frac{dE_i(t_i)}{dt_i} = \frac{dE_j(t_j)}{dt_j} \ \forall i, j \in 1..m. \qquad \square$$

Theorem 15 is a very general result; it holds for any minimum-energy function $E(t)$ (as defined earlier) and any optimization index n. It extends to the more general case of sequential composition where each subsystem S_i is used repetitively with probability p_i.

Using Theorem 15, one can determine t_i, E_i and E. If n is not given, it can be determined from $n = -\frac{dE_i(t_i)}{dt_i} \frac{\sum_{i=1}^{m} t_i}{\sum_{i=1}^{m} E_i(t_i)}$ while $n_i = -\frac{dE_i(t_i)}{dt_i} \frac{t_i}{E_i(t_i)}$.

In the following, we will consider two relevant examples of energy-delay optimization in the context of sequential composition.

5.5.2.1 Sequential Composition and Transistor Sizing

The first example we look at is the energy-delay optimization through transistor sizing of a system consisting of the sequential composition of m subsystems S_i. Given the nature of the optimization parameter (transistor sizing), the minimum-energy function of S_i is $E_i(t) = E_{0i}t/(t - t_{\infty i})$, where E_{0i} is the energy due to the total switched wire capacitance of subsystem S_i and $t_{\infty i}$ is the lower bound on the achievable delay of subsystem S_i.

Property 2 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n, then*

$$\frac{t_i - t_{\infty i}}{\sqrt{E_{0i}t_{\infty i}}} = \frac{t_j - t_{\infty j}}{\sqrt{E_{0j}t_{\infty j}}} \ \forall i, j \in 1..m.$$

Proof. Follows directly from Theorem 15. $\square$

Property 2 relates the individual delays t_is to each other. Assume the energy-delay optimization is defined in terms of n. Using Property 2, we can express all t_is function of t_1, for example, as

$$t_i = t_{\infty i} + \frac{t_1 - t_{\infty 1}}{\sqrt{E_{01}t_{\infty 1}}}\sqrt{E_{0i}t_{\infty i}} \ \ \forall i \in 1..m.$$

As a consequence, we can write

$$t = \sum_{i=1}^{m} t_i = \sum_{i=1}^{m} t_{\infty i} + \frac{t_1 - t_{\infty 1}}{\sqrt{E_{01}t_{\infty 1}}} \sum_{i=1}^{m} \sqrt{E_{0i}t_{\infty i}} \tag{5.17}$$

and

$$E = \sum_{i=1}^{m} E_i(t_i) = \sum_{i=1}^{m} \frac{E_{0i}t_i}{t_i - t_{\infty i}} = \sum_{i=1}^{m} E_{0i} + \frac{\sqrt{E_{01}t_{\infty 1}}}{t_1 - t_{\infty 1}} \sum_{i=1}^{m} \sqrt{E_{0i}t_{\infty i}} \ . \tag{5.18}$$

It follows that Et^n can be written as a single-variable function and minimized for t_1. From the proof of Theorem 15, it follows that t_1 has to fulfill the equation $\frac{dE_1(t_1)}{dt_1} = -\frac{nE}{t}$. This yields t_1 as the solution to a second order polynomial equation. Then, all other t_is and E_is are computable.

If the energy-delay optimization is specified in terms of t then, using again the proof of Theorem 15, we can find the t_is as the solution to the equation $\frac{dE_i(t_i)}{dt_i} = -\frac{nE}{t}$. It follows that

$$t_i = t_{\infty i} + \frac{t - \sum_{i=1}^{m} t_{\infty i}}{\sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}} \sqrt{E_{0i} t_{\infty i}} \;\; \forall i \in 1..m. \tag{5.19}$$

As a consequence, all the other unknowns can be computed; in particular, $n_i = t_{\infty i}/(t_i - t_{\infty i})$.

For the rest of this subsection, we present a set of simple and intuitive properties of sequential composition of circuits optimized for Et^n through transistor sizing.

Theorem 16 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n, then*

$$P = \frac{n_i^2}{n} \hat{P}_i \;\forall i \in 1..m.$$

Proof. The equality among the $\frac{n_i^2}{n}\hat{P}_i$ terms follows from Property 2 using equations $t_i = (1 + 1/n_i)t_{\infty i}$. As a consequence, we can substitute $n_i = n_j\sqrt{\frac{\hat{P}_j}{\hat{P}_i}}$, for any fixed $j \in 1..m$, in $f = Et^n$ where $E = \sum_{i=1}^{m} (n_i + 1)E_{0i}$ and $t = \sum_{i=1}^{m} (1 + \frac{1}{n_i})t_{\infty i}$. f reaches its minimum when

$$\begin{aligned}
\frac{df}{dn_j} = 0 \;\; &\Rightarrow \;\; \left(\sum_{i=1}^{m} n_i E_{0i}\right)\left(\sum_{i=1}^{m}(1 + \frac{1}{n_i})t_{\infty i}\right) = n\left(\sum_{i=1}^{m}(n_i + 1)E_{0i}\right)\left(\sum_{i=1}^{m} \frac{t_{\infty i}}{n_i}\right) \\
&\Rightarrow \;\; n\frac{\sum_{i=1}^{m}(n_i + 1)E_{0i}}{\sum_{i=1}^{m}(1 + \frac{1}{n_i})t_{\infty i}} = \frac{\sum_{i=1}^{m} n_i E_{0i}}{\sum_{i=1}^{m} \frac{t_{\infty i}}{n_i}} = \frac{\sum_{i=1}^{m} n_i \frac{E_{0j} t_{\infty i}}{t_{\infty j}} \frac{n_j^2}{n_i^2}}{\sum_{i=1}^{m} \frac{t_{\infty i}}{n_i}} \\
&\Rightarrow \;\; n\frac{E}{t} = n_j^2 \frac{E_{0j}}{t_{\infty j}} \;\forall j \in 1..m \\
&\Rightarrow \;\; P = \frac{n_i^2}{n}\hat{P}_i \;\forall i \in 1..m. \qquad \square
\end{aligned}$$

Theorem 16 is the equivalent, for sequential composition, of the third form of Theorem 9. Theorem 16 relates the total consumed power of the system to the optimization indexes and asymptotic powers of its components.

Theorem 17 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$n = n_i \ \forall i \in 1..m \iff \hat{P}_i = \hat{P}_j \ \forall i,j \in 1..m$$

Proof. Follows directly from Property 16. □

Theorem 17 is the equivalent, for sequential composition, of Theorem 8. Theorem 17 tells us that the sequential components of a system can be optimized independently for Et^n, yielding global optimum when composed, if and only if all $\hat{P}_i$s are equal. Otherwise, even if one is globally optimizing for n, locally one needs to be able to optimize for $n_i \neq n$.

Property 3 *For the sequential composition of m systems $S_i(E_{0i},t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$\left(\sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}\right)^2 \leq \sum_{i=1}^{m} E_{0i} \sum_{i=1}^{m} t_{\infty i}$$

with equality if and only if all $\hat{P}_i$s are equal.

Proof. We invoke again the Cauchy-Schwarz inequality $(\sum l_i r_i)^2 \leq \sum l_i^2 \sum r_i^2$, where equality holds if and only if l_i/r_i has the same value for all i. If we substitute $l_i \leftarrow \sqrt{E_{0i}}$ and $r_i \leftarrow \sqrt{t_{\infty i}}$, we get that

$$\left(\sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}\right)^2 \leq \sum_{i=1}^{m} E_{0i} \sum_{i=1}^{m} t_{\infty i}$$

with equality if and only if all $\hat{P}_i = \frac{E_{0i}}{t_{\infty i}}$ are equal. □

Property 3 will be used in the proof of Theorems 18 and 19.

Theorem 18 *For the sequential composition of m systems $S_i(E_{0i},t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$E \leq (n+1)\sum_{i=1}^{m} E_{0i}$$

with equality if and only if all $\hat{P}_i$s are equal.

Proof. Using Equations 5.17 and 5.18, we can write a second-order polynomial equation

$$nx^2\sum_{i=1}^{m} E_{0i} + (n-1)x\sqrt{E_{01}t_{\infty 1}}\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}} - E_{01}t_{\infty 1}\sum_{i=1}^{m} t_{\infty i} = 0,$$

where $x = t_1 - t_{\infty 1}$. It follows that

$$x = \frac{-(n-1)\sqrt{E_{01}t_{\infty 1}}\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}} + \sqrt{\Delta}}{2n\sum_{i=1}^{m} E_{0i}},$$

where

$$\Delta = (n-1)^2 E_{01}t_{\infty 1}\left(\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}\right)^2 + 4nE_{01}t_{\infty 1}\sum_{i=1}^{m} E_{0i}\sum_{i=1}^{m} t_{\infty i}.$$

However, using Property 3, we have

$$\begin{aligned}\Delta &\geq (n-1)^2 E_{01}t_{\infty 1}\left(\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}\right)^2 + 4nE_{01}t_{\infty 1}\left(\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}\right)^2 \\ &= (n+1)^2 E_{01}t_{\infty 1}\left(\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}\right)^2\end{aligned}$$

with equality if and only if all $\hat{P}_i$s are equal. It follows that

$$x \geq \frac{\sqrt{E_{01}t_{\infty 1}}\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}}{n\sum_{i=1}^{m} E_{0i}}.$$

With this result in mind, using again Equation 5.18, we can write the consumed energy as

$$E = \sum_{i=1}^{m} E_{0i} + \frac{\sqrt{E_{01} t_{\infty 1}}}{x} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}} \leq (n+1) \sum_{i=1}^{m} E_{0i}. \qquad \square$$

Theorem 18 is the equivalent, for sequential composition, of Theorem 10. In Theorem 18, equality holds if and only if all $\hat{P}_i$s are equal; in this situation, we also have that $E_i = (n+1)E_{0i}$. Given that Theorems 10 and 18 have the same form, it follows that for any parallel-sequential composition of circuits a property of the same form holds.

Theorem 19 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$t \leq \left(1 + \frac{1}{n}\right) \sum_{i=1}^{m} t_{\infty i}$$

with equality if and only if all $\hat{P}_i s$ are equal.

Proof. Using Theorem 15, we get

$$\frac{E_{0i} t_{\infty i}}{(t_i - t_{\infty i})^2} = n \frac{E}{t} \;\; \forall i \in 1..m.$$

With the notation $x_i = t_i - t_{\infty i}$ and $\alpha = n\frac{E}{t}$ it follows that

$$\frac{E_{0i} t_{\infty i}}{x_i^2} = \alpha \;\; \forall i, j \in 1..m \Rightarrow x_i = \frac{\sqrt{E_{0i} t_{\infty i}}}{\sqrt{\alpha}}.$$

Notice that

$$\alpha = n\frac{E}{t} = n\frac{\sum_{i=1}^{m} E_i}{\sum_{i=1}^{m} t_i} = n\frac{\sum_{i=1}^{m} E_{0i} + \sum_{i=1}^{m} \frac{E_{0i} t_{\infty i}}{x_i}}{\sum_{i=1}^{m} t_{\infty i} + \sum_{i=1}^{m} x_i} = n\frac{\sum_{i=1}^{m} E_{0i} + \sqrt{\alpha} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}}{\sum_{i=1}^{m} t_{\infty i} + \frac{1}{\sqrt{\alpha}} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}}.$$

If we call

$$x = \frac{\sum_{i=1}^{m} t_{\infty i}}{\sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}} \quad \text{and} \quad y = \frac{\sum_{i=1}^{m} E_{0i}}{\sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}},$$

we know from Property 3 that $xy \geq 1$, with equality if and only if all $\hat{P}_i$s are equal. Using these new notations, we can rewrite the expression of α as

$$\alpha = n\frac{y+\sqrt{\alpha}}{x+\frac{1}{\sqrt{\alpha}}} \Rightarrow y = \frac{\alpha}{n}\left(x+\frac{1}{\sqrt{\alpha}}\right) - \sqrt{\alpha}.$$

Thus,

$$xy \geq 1 \Rightarrow x\left(\frac{\alpha}{n}\left(x+\frac{1}{\sqrt{\alpha}}\right) - \sqrt{\alpha}\right) \geq 1 \Rightarrow \left(x - \frac{n}{\sqrt{\alpha}}\right)\left(x+\frac{1}{\sqrt{\alpha}}\right) \geq 0.$$

However, $x + \frac{1}{\sqrt{\alpha}} > 0$, since all variables are positive. It follows than that

$$\begin{aligned}
x - \frac{n}{\sqrt{\alpha}} \geq 0 \quad &\Rightarrow \quad x \geq \frac{n}{\sqrt{\alpha}} \Rightarrow \frac{\sum_{i=1}^{m} t_{\infty i}}{\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}} \geq \frac{n}{\sqrt{\alpha}} \\
&\Rightarrow \quad \frac{1}{n}\sum_{i=1}^{m} t_{\infty i} \geq \frac{\sum_{i=1}^{m}\sqrt{E_{0i}t_{\infty i}}}{\sqrt{\alpha}} = \sum_{i=1}^{m} x_i \\
&\Rightarrow \quad \left(1+\frac{1}{n}\right)\sum_{i=1}^{m} t_{\infty i} \geq \sum_{i=1}^{m} t_{\infty i} + \sum_{i=1}^{m} x_i = \sum_{i=1}^{m} t_i = t. \qquad \square
\end{aligned}$$

Theorem 19 is the equivalent, for sequential composition, of Theorem 12. In Theorem 19, equality holds if and only if all $\hat{P}_i$s are equal; in this situation, we also have that $t_i = (1+\frac{1}{n})t_{\infty i}$. The same way as Theorems 10 and 18 give an upper bound on the energy E for a parallel-sequential composition, Theorems 12 and 19 give an upper bound on the delay t of the same composition.

Theorem 20 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$\frac{P_i}{\sqrt{\hat{P}_i}} = \frac{P_j}{\sqrt{\hat{P}_j}} \;\forall i,j \in 1..m.$$

Proof. From Property 2, it follows that

$$\frac{\sqrt{E_{0i}t_{\infty i}}}{t_i - t_{\infty i}} = \frac{\sqrt{E_{0j}t_{\infty j}}}{t_j - t_{\infty j}} \;\forall i,j \in 1..m \quad \Rightarrow \quad \frac{E_i}{t_i}\frac{\sqrt{E_{0i}t_{\infty i}}}{E_{0i}} = \frac{E_j}{t_j}\frac{\sqrt{E_{0j}t_{\infty j}}}{E_{0j}} \;\forall i,j \in 1..m$$

$$\Rightarrow \frac{P_i}{\sqrt{\frac{E_{0i}}{t_{\infty i}}}} = \frac{P_j}{\sqrt{\frac{E_{0j}}{t_{\infty j}}}}\ \forall i,j \in 1..m \ \Rightarrow\ \frac{P_i}{\sqrt{\hat{P}_i}} = \frac{P_j}{\sqrt{\hat{P}_j}}\ \forall i,j \in 1..m. \qquad \square$$

Theorem 20 tells us that when optimizing through transistor sizing, circuits composed sequentially should be designed so as to make their power usage proportional to the square root of their *asymptotic power*.

The achievable lower bound through transistor sizing of a sequential composition is given by the next two relations. While Property 4 gives an exact minimum for the special case of $n = 1$, Theorem 21 gives an upper bound on this minimum for any n.

Property 4 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et through transistor sizing, then*

$$\sqrt{\min(Et)} = \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}} + \sqrt{\left(\sum_{i=1}^{m} E_{0i}\right)\left(\sum_{i=1}^{m} t_{\infty i}\right)}.$$

Proof. Using Equations 5.17 and 5.18, we can write Et function of $x = t_1 - t_{\infty 1}$ as

$$Et = \left(\sum_{i=1}^{m} E_{0i} + \frac{\sqrt{E_{01} t_{\infty 1}}}{x} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}\right)\left(\sum_{i=1}^{m} t_{\infty i} + \frac{x}{\sqrt{E_{01} t_{\infty 1}}} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}\right)$$

If now we open up the parenthesis and apply the inequality *arithmetical mean ≥ geometrical mean* it follows that

$$Et \geq \left(\sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}} + \sqrt{\left(\sum_{i=1}^{m} E_{0i}\right)\left(\sum_{i=1}^{m} t_{\infty i}\right)}\right)^2$$

with equality if and only if

$$\left(\sum_{i=1}^{m} t_{\infty i}\right) \frac{\sqrt{E_{01} t_{\infty 1}}}{x} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}} = \left(\sum_{i=1}^{m} E_{0i}\right) \frac{x}{\sqrt{E_{01} t_{\infty 1}}} \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}}$$

or

$$x = t_1 - t_{\infty 1} = \sqrt{\frac{E_{01} t_{\infty 1} \sum_{i=1}^{m} t_{\infty i}}{\sum_{i=1}^{m} E_{0i}}}.$$

Similarly,

$$t_i = t_{\infty i} + \sqrt{\frac{E_{0i} t_{\infty i} \sum_{i=1}^{m} t_{\infty i}}{\sum_{i=1}^{m} E_{0i}}} \quad \forall i \in 1..m. \tag{5.20}$$

In consequence, using the values determined by Equation 5.20, the inequality becomes equality:

$$\sqrt{Et} = \sum_{i=1}^{m} \sqrt{E_{0i} t_{\infty i}} + \sqrt{\left(\sum_{i=1}^{m} E_{0i}\right)\left(\sum_{i=1}^{m} t_{\infty i}\right)}. \qquad \square$$

Theorem 21 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$\min(Et^n) \leq (n+1)\left(\sum_{i=1}^{m} E_{0i}\right)\left(\left(1+\frac{1}{n}\right)\left(\sum_{i=1}^{m} t_{\infty i}\right)\right)^n$$

with equality if and only if all $\hat{P}_i$s are equal.

Proof. Follows directly from Theorem 18 and 19.

Theorem 21 is the equivalent, for sequential composition, of Theorem 13. The given upper bound is, in practice, a tight bound and due to the flatness of the Et^n metric around the optimum it is a good approximation of the absolute minimum.

Theorem 22 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through transistor sizing, then*

$$n \min_{i \in 1..m} \hat{P}_i \leq P \leq n \max_{i \in 1..m} \hat{P}_i$$

with equality if and only if all $\hat{P}_i$s are equal.

Proof. Using Theorem 16, we can show that the inequality stated by the theorem is equivalent to

$$\min_{i \in 1..m} n_i \leq n \leq \max_{i \in 1..m} n_i. \tag{5.21}$$

We prove Equation 5.21 by contradiction. Assume first that $\forall i \in 1..m : n_i < n$. Substituting $n_i = t_{\infty i}/(t_i - t_{\infty i})$, it follows that

$$\forall i \in 1..m : \left(1+\frac{1}{n}\right)t_{\infty i} < t_i \;\Rightarrow\; \left(1+\frac{1}{n}\right)\sum_{i=1}^{m} t_{\infty i} \leq \sum_{i=1}^{m} t_i = t.$$

But this is in contradiction with Theorem 19. As a consequence, there exists $i \in 1..m$ such that $n_i \geq n$; thus $\max_{i \in 1..m} n_i \geq n$. Assume now that $\forall i \in 1..m : n_i > n$. Substituting $n_i = E_i/E_{0i} - 1$, it follows that

$$\forall i \in 1..m : E_i \geq (n+1)E_{0i} \;\Rightarrow\; E = \sum_{i=1}^{m} E_i \geq (n+1)\sum_{i=1}^{m} E_{0i}.$$

But this is in contradiction with Theorem 18. As a consequence, there exists $i \in 1..m$ such that $n_i \leq n$; thus $\min_{i \in 1..m} n_i \leq n$. As a consequence, we have shown that $\min_{i \in 1..m} n_i \leq n \leq \max_{i \in 1..m} n_i$, which is equivalent with the claim of the theorem. □

Theorem 22 bounds the power consumption of the sequential circuit between n times the smallest and the largest asymptotic powers of the components.

While it is rather obvious what it means to have all $t_{\infty i}$s equal for parallel composition, it is not immediately clear what all $\hat{P}_i$s equal imply. Consider two pipeline stages S_1 and S_2 composed sequentially, and assume that S_1 operates on N_1 bits while S_2 operates on N_2 bits. Further assume—for simplicity—that, per bit, the minimal energies and the minimal delays are the same for both pipeline stages, respectively. In other words, $E_{01} = N_1 E_0$, $t_{\infty 1} = t_\infty$ and $E_{02} = N_2 E_0$, $t_{\infty 2} = t_\infty$. This assumption is reasonable for pipelines with comparable per-bit-complexity and similar latency—so as to operate in the same clock domain. When we compute the asymptotic powers of S_1 and S_2 we get that $\hat{P}_1 = N_1 \frac{E_0}{t_\infty}$ and $\hat{P}_2 = N_2 \frac{E_0}{t_\infty}$. For these two values to be equal—as required for equality in Theorems 18, 19 and 21—we need to have $N_1 = N_2$, i.e., the number of bits each pipeline operates on should be the same. This suggests that the bounds are tighter for pipeline chains that average out more evenly the number of bits operated on in each individual stage.

Theorem 13 together with Theorem 21 provide an upper bound to the energy-delay efficiency of any parallel-sequential composition of circuits. Furthermore, they suggest a practical way to improve the energy-efficiency of these circuits by reducing the E_{0i}s and $t_{\infty i}$s. Transistor sizing is not able to change the $t_{\infty i}$s or the E_{0i}s, since they depend on other variables than transistor sizes—such as circuit micro-architecture, supply voltage, and fabrication technology. Thus, improving these other factors will ultimately impact the efficiency of the final design. In particular, it should be noted that E_{0i} depends on the wiring of system S_i; thus, compact hand layout or good layout tools can make a difference on the energy-efficiency of circuits. Similarly, $t_{\infty i}$s can be directly improved by a proper choice of transistor netlist topology.

5.5.2.2 Sequential Composition and Voltage Scaling

The second example we look at is the energy-delay optimization through voltage scaling of a system consisting of the sequential composition of m subsystems S_i. Given the nature of the optimization parameter (voltage scaling), the minimum-energy function of S_i is $E_i(t) = \Theta_i/t^2$, where Θ_i is a voltage independent constant. We can show the following:

Theorem 23 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through voltage scaling, then all P_is are equal.*

Proof. Using Theorem 15 with $E_i(t_i) = \frac{\Theta_i}{t_i^2}$ $\forall i \in 1..m$ we get

$$\frac{\Theta_i}{t_i^3} = \frac{\Theta_j}{t_j^3} \ \forall i, j \in 1..m \Rightarrow \frac{E_i}{t_i} = \frac{E_j}{t_j} \ \forall i, j \in 1..m \Rightarrow P_i = P_j \ \forall i, j \in 1..m. \qquad \square$$

For $n = 2$ this correlation was first suggested by Mika Nyström [45]. Theorem 23 tells us that if the used optimization is voltage scaling, circuits composed sequentially and optimized for Et^n should be designed so as to equalize their power use. With the definition of t_{min} and t_{max} given in Section 5.5.1.2, we can state the following theorem:

Theorem 24 *For the sequential composition of m systems $S_i(E_{0i}, t_{\infty i})$, if the composed system is optimized for Et^n through voltage scaling and if $t \in [t_{min}, t_{max}]$, then*

$$\text{if } n > 2 \ \ \min Et^n = \left(\sum \sqrt[3]{\Theta_i}\right)^{(n+1)} \left(\frac{t_{min}}{\sqrt[3]{\Theta_1}}\right)^{(n-2)},$$

$$\text{if } n < 2 \ \ \min Et^n = \left(\sum \sqrt[3]{\Theta_i}\right)^{(n+1)} \left(\frac{t_{max}}{\sqrt[3]{\Theta_1}}\right)^{(n-2)},$$

$$\text{if } n = 2 \ \ \min(Et^2) = \Big(\sum_{i=1}^{m} \sqrt[3]{\Theta_i}\Big)^3$$

or equivalently

$$\sqrt[3]{\min\left(Et^2\right)} = \sum_{i=1}^{m} \sqrt[3]{\Theta_i}\ .$$

Proof. Using Theorem 15 with $E_i(t_i) = \frac{\Theta_i}{t_i^2}$ we get

$$\frac{\Theta_i}{t_i^3} = \frac{\Theta_j}{t_j^3}\ \forall i,j \in 1..m \Rightarrow \frac{t_i}{\sqrt[3]{\Theta_i}} = \frac{t_j}{\sqrt[3]{\Theta_j}}\ \forall i,j \in 1..m.$$

Thus,

$$\min Et^n = \min\Big(\sum_{i=1}^{m} E_i(t_i)\Big)\Big(\sum_{i=1}^{m} t_i\Big)^n = \Big(\sum_{i=1}^{m} \sqrt[3]{\Theta_i}\Big)^{(n+1)} \min\Big(\frac{t_1}{\sqrt[3]{\Theta_A}}\Big)^{(n-2)}.$$

If $n > 2$ $\min Et^n$ is reached for t_{min} (highest feasible voltage), while if $n < 2$ $\min Et^n$ is reached for t_{max} (lowest feasible voltage). If $n = 2$ then $\min Et^2 = \left(\sum_{i=1}^{m} \sqrt[3]{\Theta_i}\right)^3$ or $\sqrt[3]{\min Et^2} = \sum_{i=1}^{m} \sqrt[3]{\Theta_i}$.

For $n = 2$ this theorem was first proved by Karl Papadantonakis [46]. Theorem 24 gives a lower bound on the achievable optimum for sequential composition using voltage scaling.

5.6 Conclusions

In this chapter we have shown that the energy-delay efficiency metric Et^n captures *any* optimal trade-off between the energy and the delay of the computation. We have presented another—seemingly different—way to capture the energy-delay trade-off, and we have shown that these two forms ultimately yield the same circuit solution.

We applied this new concept to the parallel and sequential composition of circuits in general and in particular to circuits optimized through transistor sizing and voltage scaling. We gave necessary and sufficient conditions under which subcomponents of a design can be optimized independently so as to yield global optimum when composed. We bounded the delay and energy of the optimized circuit and we gave necessary and sufficient conditions under which these bounds are reached. When applied to transistor sizing, we found that circuits composed sequentially should be designed so as to make their power usage proportional to the square root of their asymptotic power. When applied to voltage scaling, we found that circuits composed sequentially should be designed so as to equalize the power usage of each component. Many of the results inferred for parallel and sequential composition apply directly to the more general parallel-sequential composition of circuits.

We have demonstrated the utility of the minimum-energy function and its capacity to capture high level compositional properties of circuits. The use of the minimum-energy function gave us practical insight into ways to improve the overall energy-delay efficiency of the studied design.

Chapter 6

Conclusions and Future Work

This thesis has developed a *circuit-level theory of energy-delay complexity* of asynchronous circuits. Within the considered framework, the energy-delay efficiency of a circuit was characterized using the metric Et^n, where E is the energy consumed by the computation, t is the delay of the computation and n is a positive number that reflects a chosen trade-off between energy and delay. Based on theoretically and experimentally, it was argued that for a circuit optimized for minimal Et^n for a fixed n, the consumed energy E_n is

$$E_n \approx (n+1)E_0 \tag{6.1}$$

and the delay t_n is

$$t_n \approx \left(1 + \frac{1}{n}\right)t_\infty, \tag{6.2}$$

where E_0 is the theoretical minimal energy (i.e., the energy due to the total switched wire capacitance) and t_∞ is theoretical minimal delay [43]. Based on Equation 6.1, the consumed energy is independent of the types of gates used by the circuit and is solely dependent on n and the amount of wiring capacitance switched during computation. On the other hand, based on Equation 6.2 the circuit speed is independent of the wire capacitance and depends only on n and the types of gates used.

Based on Equations 6.1 and 6.2, the energy-delay complexity of a circuit was defined as $E_0 t_\infty^n$. Using this definition, we can compare at an abstract level the energy-delay efficiency of different circuits implementing the same computation, without going through the costly implementation steps of transistor sizing, layout, and electrical simulation.

Equations 6.1 and 6.2 allow us to estimate the energy and delay of an Et^n-optimal system. However, they do not say much about the actual transistor sizes that achieve this optimum. Therefore, a simple analytical formula was developed that approximates, to within a few percent, the actual optimal transistor sizes [52]. Such a formula has two main uses: first, if a rough estimate is enough for the given application, the formula can be used as is (no numerical optimization is then needed); second, if accuracy is important, the formula can provide a good starting point for numerical optimization. An efficient iterative procedure was also studied. Such a procedure can further improve the original analytical transistor sizing solution. Based on these results, a novel transistor-sizing algorithm for energy-delay efficiency was introduced.

One way to capture the trade-off between the consumed energy and the delay of a circuit is through the Et^n metric. It was shown that there is an equivalent way to characterize the same trade-off through the minimum-energy function, i.e., the function that describes the minimum energy required for a system to run at a given speed [51]. The theory of minimum-energy functions was applied to the parallel and sequential composition of circuits in general, and in particular to circuits optimized through transistor sizing and voltage scaling. These case studies have demonstrated the utility of the minimum-energy function in capturing high level compositional properties of circuits. The use of the minimum-energy function yields practical insight into ways to improve the overall energy-delay efficiency of circuits.

6.1 Future Work

The circuit level energy-delay complexity model utilizes surprisingly little knowledge about the actual circuit; it requires only E_0 and t_∞. While the current work has focused on accurately computing these values at circuit level, it is conceivable that a good estimate of E_0 and t_∞ can be achieved at a higher level, such as the CHP level. Given a generic wiring model and a circuit template, the minimal energy and minimal delay of the basic CHP constructs could be computed. Then, these constructs could be used to estimate the energy and delay of any CHP program. This early knowledge about the expected circuit performance can help guide the designer, or the tool assisting the designer [53], to more efficiently explore the available design space. Given the bottom-up approach of the suggested CHP energy and delay estimation, it would be interesting to see how the energy estimates match the energy estimates inferred, through a top-down approach, by Tierno in [6].

On the practical side, the energy-delay complexity suggests that Et^2 optimal circuits should be sized to run about 50% slower than an equivalent circuit optimized mostly for speed. The increase in delay should be more than compensated by the energy savings and the overall Et^2 should significantly decrease. This conclusion has been inferred based on a widely accepted but relatively simple transistor model [56, 57]. It would be interesting to see how such a prediction plays out in practice, given the ever-growing concern about modeling submicron fabrication technologies. Currently, we are designing two complex chips targeted for energy-delay efficiency. The outcome of these projects should clarify the large-scale practical applicability of our theoretical results.

Epilog

[6.52] We feel that even if all possible *scientific questions be answered, the problems of life have still not been touched at all. Of course there is no question left, and just this is the answer.*
[6.54] My propositions are elucidatory in this way: he who understands me finally recognizes them as senseless, when he has climbed out through them, on them, over them. (He must so to speak throw away the ladder, after he has climbed up on it.)
[7] Whereof one cannot speak, thereof one must be silent.

Ludwig Wittgenstein

Bibliography

[1] Alain J. Martin, *Towards an Energy Complexity of Computation.* Information Processing Letters, 77, 2001.

[2] Alain J. Martin, *Remarks on Low-power Advantages of Asynchronous Circuits.* ESSCIRC'98: Low-Power Systems on a Chip, September 1998.

[3] Anantha P. Chandrakasan, Robert W. Brodersen, *Low Power Digital CMOS Design.* Kluwer Academic Publishers, 1995.

[4] R. Gonzalez, M. Horowitz, *Supply and threshold voltage scaling for low power CMOS.* IEEE Journal of Solid-State Circuits, August 1997.

[5] M. Horowitz, T. Indermaur, R. Gonzalez, *Low-power digital design.* Symposium on Low Power Electronics, October 1994, pages 8-11.

[6] José A. Tierno, *An Energy-Complexity Model for VLSI Computations.* Ph.D. thesis, Caltech, 1995.

[7] Steven M. Burns, *Performance Analysis and Optimization of Asynchronous Circuits.* Ph.D. thesis, Caltech, 1991.

[8] Tak Kwan Lee, *A General Approach to Performance Analysis and Optimization of Asynchronous Circuits.* Ph.D. thesis, Caltech, 1995.

[9] Ted Williams, *Latency and Throughput Tradeoffs in Self-Timed Asynchronous Pipelines and Rings.* Stanford Tech. Report, CSL-TR-90-431, May 1990.

[10] Andrew M. Lines, *Pipelined Asynchronous Circuits.* M.S. thesis, Caltech, 1995.

[11] Ivan Sutherland, Bob Sproull, David Harris, *Logical Effort: Designing Fast CMOS Circuits.* Morgan Kaufmann Publishers, 1999.

[12] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest *Introduction to Algorithms.* McGraw-Hill, 1992.

[13] Eugene L. Lawler, *Combinatorial Optimizations: Networks and Matroids.* Hold, Rinehart and Winston, 1976.

[14] Matt Hanna, Eitan Grinspun, *A Production Rule Simulation.* Caltech Computer Science Technical Report, 2000

[15] A. Shen, A. Ghosh, S. Devadas, K. Keutzer, *On average power dissipation and random pattern testability of CMOS combinational logic networks.* IEEE/ACM Int. Conf. Computer-Aided Design, Santa Clara, CA, Nov. 8-12, 1992, pp 402.

[16] M. A. Cirit, *Estimating dynamic power consumption of CMOS circuits.* IEEE Int. Conf. Computer-Aided Design, pp. 534-537, 1987.

[17] F. Najm, R. Burch, P. Yang, I. Hajj, *CREST- A current estimation tool for CMOS circuits.* IEEE Int. Conf. Computer-Aided Design, Santa Clara, CA Nov. 7-10, 1988, pp.204-207.

[18] C. Y. Tsui, M. Pedran, A. M. Despain, *Efficient estimation of synamic power consumption under a real delay model.* IEEE Int. Conf. Computer-Aided Design, Satan Clara, CA, Nov 7-11, 1993, pp. 224-228.

[19] C. M. Huizer, *Power dissipation analysis of CMOS VLSI circuits by means of switch-level simulation.* IEEE Europ. Solid-State Computer Conf, Grenoble, France, 1990, pp. 61-64.

[20] R. Burch, F. Najm, P. Yang, T. Trick, *McPower: A Monte Carlo approach to power estimation.* IEEE/ACM Int. Conf. Computer-Aided Design, Santa Clara, CA, Nov 8-12, 1992, pp. 90-97.

[21] F. Najm, *Transition density, a stochastic measure of activity in circuits.* 28th ACM/IEEE Design Automation Conference, San Francisco, CA, June 17-21, 1991, pp. 644-649.

[22] F. Najm, *Transition density: a new measure of activity in digital circuits.* IEEE Trans. Computer-Aided Design, Vol. 12, No. 2, pp. 310-323, Feb. 1993.

[23] H. J. M. Veendrick, *Short-Circuit Dissipation of Static CMOS Circuitry and Its Impact on the Design of Buffer Circuits.* IEEE Journal of Solid-State Circuits, Vol. SC-19,No. 4, August 1984, pp. 468-473.

[24] T. Sakurai, A. R. Newton, *Alpha-Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas.* IEEE Journal of Solid-State Circuits, Vol. 25, No. 2, April 1990.

[25] T. Sakurai, A. R. Newton, *Delay Analysis of Series-Connected MOFSET Circuits.* IEEE Journal of Solid-State Circuits, Vol. 26, No. 2, Feb. 1991.

[26] N. Hedenstierna, K. O. Jeppson, *CMOS Circuit Speed and Buffer Optimization.* IEEE Transactions on Computer-Aided Design, Vol. CAS-6, No. 2, March 1987.

[27] S. R. Vemuru, N. Scheinberg, *Short-Circuit Power Dissipation Estimation for CMOS Logic Gates.* IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 41, No. 11, November 1994.

[28] L. Bisdounis, S. Nikolaidis, O. Koufopavlou, *Propagation Delay and Short-Circuit Power Dissipation Modeling of the CMOS Inverter.* IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, Vol. 45, No. 3, March 1998.

[29] S. Sze, *Physics of Semiconductor Devices.* Wiley, New York, 1981.

[30] Alain J. Martin, *Synthesis of Asynchronous VLSI Circuits.* Formal Methods for VLSI Design, ed. J. Staunstrup, North-Holland, 1990.

[31] Alain J. Martin. *Synthesis of Asynchronous VLSI Circuits.* Report CS–TR–93–28, California Institute of Technology, 1993.

[32] Paul I. Pénzes, *The Design of High Performance Asynchronous Circuits for the Caltech MiniMIPS Processor.* M.S. thesis, Caltech, 1999 (CSTR:2002.001).

[33] Alain J. Martin, Andrew Lines, Rajit Manohar, Mika Nyström, Paul I. Pénzes, Robert Southworth and Uri Cummings. *The Design of an Asynchronous MIPS R3000 Microprocessor.* Proceedings of the 17th Conference on Advanced Research in VLSI, IEEE Computer Society Press, p164-181, 1997.

[34] P. E. Gill, W. Murray, M. H. Wright, *Practical Optimization.* Academic Press, 1981.

[35] D. P. Marple, *Transistor size optimization in the Tailor layout system.* Design Automation Conference, 1989, pp. 43-48.

[36] J. P. Fishburn, A. E. Dunlop, *TILOS: A posynomial approach to transistor sizing.* Proceedings of the 1985 International Coference on Computer-aided Design, Nov. 1985, pp. 326-328.

[37] B. Hoppe, G. Neuendorf, D. Schmitt-Landsiedel, W. Specks, *Optimization of high-speed CMOS logic circuits with analytical models for signal delay, chip area, and dynamic power dissipation.* IEEE Tran. on Computer-Aided Design, 9(3):236–247, 1990.

[38] Y. Tamiya, Y. Matsunaga, M. Fujita, *LP based Cell Selection with Constraints on Timing, Area and Power Consumption.* in Proc. ICCAD Conf., Nov. 1994.

[39] G. Chen, H. Onodera, K. Tamaru, *An Iterative Gate Sizing Approach with Accurate Delay Evaluation.* Proc. IEEE Int'l. Conf. on CAD, 1995.

[40] Chenming Hu, *Device and Technology Impact on Low Power Electronics.* Low Power Design Methodologies, Kluwer Academic/Plenum Publishers, 1996.

[41] J. Cong, C. K. Koh, *Simultaneous Driver and Wire Sizing for Performance and Power Optimization.* IEEE Trans. on VLSI Systems, pp. 408-425, December 1994.

[42] Maria del Mar Hershenson, Stephen Boyd, Thomas H. Lee, *CMOS Operational Amplier Design and Optimization via Geometric Programming*, 1997.

[43] Paul I. Pénzes, Alain Martin, *Global and Local Properties of Asynchronous Circuits Optimized for Energy Efficiency.* IEEE Workshop on Power Management for Real-time and Embedded Systems, Taipei, Taiwan, May 29th, 2001 (CSTR:2002.002).

[44] Alain Martin, Mika Nyström, Paul I. Pénzes, *ET^2: A Metric for Time and Energy Efficiency of Computation.* Power-Aware Computing, Kluwer Academic/Plenum Publishers, 2001 (CSTR:2001.007).

[45] Mika Nyström, *Et^2 and Multi-voltage Logic.* Caltech Technical Report, April 1995.

[46] Karl Papadantonakis, *Hierarchical Voltage Scaling for Et^2*. Caltech Computer Science Technical Report 2001.005, July 2001.

[47] John L. Hennessey, David A. Patterson. *Computer Architectures: A Quantitative Approach.* Morgan Kaufmann, 1990.

[48] José A. Tierno, *An Energy-Complexity Model for VLSI Computations*, Ph.D. Thesis, California Institute of Technology, 1995.

[49] L. W. Nagel, D. O. Pederson, *Simulation Program with Integrated Circuit Emphasis.* In Proceedings of the 16th Midwest Symposium Circuit Theory, Waterloo, Canada, 1973. 23.

[50] Paul I. Pénzes, Alain J. Martin, *An Energy Estimation Method for Asynchronous Circuits with Application to an Asynchronous Microprocessor.* Design Automation and Test in Europe, Paris, France March 4-8, 2002.

[51] Paul I. Pénzes, Alain J. Martin, *Energy-Delay Efficiency of VLSI Computations.* 12th Great Lakes Symposium on VLSI, New York, USA April 18-19, 2002.

[52] Paul I. Pénzes, Mika Nyström, Alain J. Martin, *Transistor Sizing of Energy-Delay–Efficient Circuits.* Caltech Computer Science Technical Report 2002.003, April 2002.

[53] Catherine G. Wong, Alain J. Martin. *Data-driven Process Decomposition For the Synthesis of Asynchronous Circuits.* Proc. IEEE Conference on Electronic Circuits and Systems, 2001.

[54] Kaveh Shakeri, James D. Meindl, *A Compact Delay Model for Series-Connected MOSFET's.* 12th Great Lakes Symposium on VLSI, New York, USA April 18-19, 2002.

[55] J. Rubenstein, P. Penfield, and M. A. Horowitz, *Signal delay in RC tree networks.* IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems 2 (1983), no. 3, 202-211.

[56] C. Mead, L. Conway, *Introduction to VLSI systems.* Addison Wesley, 1980.

[57] N. H. Weste, K. Eshraghian, *Principles of CMOS VLSI Design.* Addison Wesley, 1994.

Appendix A

An Energy Estimation Method for Asynchronous Circuits with Application to an Asynchronous Microprocessor

A.1 Abstract

This appendix presents a simulator operating on a logical representation of an asynchronous circuit that gives energy estimates within 10% of electrical (`hspice`) simulation. Our simulator is the first such tool in the literature specifically targeted to efficient energy estimation of QDI asynchronous circuits.

As an application, we show how the simulator has been used to accurately estimate the energy consumption in different parts of an asynchronous MIPS R3000 microprocessor. This is the first energy breakdown of an asynchronous microprocessor in the literature.

A.2 Introduction

The increasing power consumption of modern VLSI systems requires better design methods to save energy. To guide the designer, energy estimation methods are needed to understand where and how the energy is consumed in a circuit. Clearly, there is a trade-off between the accuracy of the energy estimation and the level of detail at which the circuit is analyzed or simulated. The more detailed the description, the more accurate the simulation or analysis will be. But on the other hand, the more time consuming it will be. Also, the designer wants to make decisions as early as possible in the design process so as to avoid costly design backtracking.

This appendix presents a simulator operating on a logical representation of the design—called "production-rules" (PRs)—that gives energy estimates within 10% of electrical (`hspice`) simulation. The PRs of a design can be either extracted from the layout or synthesized from a high-level description. Similarly, the physical parameters needed for evaluating the energy can be extracted from the layout or computed directly.

The simulator and the estimation method have been developed for the design of asynchronous circuits. Asynchronous circuits do not use clocks. Communication and synchronization among the units are implemented by handshake protocols. The particular type of asynchronous circuits this method is based on are called quasi-delay-insensitive, or QDI [30]. QDI circuit use no timing assumption on the delays of the operators and wires, with the exception of some forks, called *isochronic forks*, in which the delays on the different branches of the fork are assumed to be similar. QDI circuits are the most conservative asynchronous circuits in terms of the use of delays. But they are also the most robust to physical parameters variations because the circuit's dependence on delays is minimal. QDI circuits are interesting in the context of energy estimation because, first, they are inherently energy-efficient—absence of a global clock, locality of activity, automatic shut-off of inactive parts, absence of spurious transitions (glitches), and second, the data encoding required for QDI circuits reduces data-dependent variation in switching activity, hence making the energy estimation more accurate. Our simulator is the first such tool in the literature specifically targeted to efficient energy estimation of QDI circuits.

The appendix is organized as follows. We first introduce PRs as both a programming language and a logical model for transistor networks. We present previous work in energy estimation and focus on some of the resulting difficulties such as glitches and data dependencies. We then show how energy estimation in the QDI asynchronous context is simpler and more accurate than in the synchronous context. We describe the `esim` simulator that executes a PR set. The simulator can be used for logical, timing, and energy simulation. We then describe the model used for estimating energy and the resulting accuracy—when applied to QDI circuits.

As an application, we show how the simulator has been used to estimate the energy consumption in the different parts of the MiniMIPS [33], an asynchronous MIPS R3000. This is the first known energy breakdown of an asynchronous microprocessor in the literature.

A.3 Production Rules as a Model for CMOS circuits

A "production-rule" (PR) is a construct of the form $G \mapsto S$, where S is a simple boolean assignment and G is a boolean expression called the guard of the PR [30]. For example, a *nand* gate with inputs x and y and output z is implemented by the two PRs: $x \wedge y \mapsto z \downarrow$ and $\neg x \vee \neg y \mapsto z \uparrow$. The PRs that set and reset the same variable, like $\neg g1 \mapsto z \uparrow$, $g2 \mapsto z \downarrow$ are implemented as one operator ($g1$ corresponds to the pull-up, while $g2$ to the pull-down circuitry of node z).

We use a PR set as the logical representation of QDI circuits. PRs closely correspond to the intended CMOS implementation of the circuit. The PR set of a given QDI circuit could be either synthesized from a high level representation or could be directly generated by the designer.

A.4 Energy Estimation

A.4.1 Previous work

The estimation of energy consumption using a simulator is complicated by the *pattern-dependence* problem. In a synchronous circuit, the switching activity of a node depends on the current data being processed and also on the initial state of that node.

Furthermore, during a clock cycle, a node might switch several times before settling to the steady state value. This extra switching activity (glitches) contributes to the total energy dissipation. The extra energy added by glitches is typically 20%, but could be up to 70% of the total energy (in combinational adders) [15].

In general, the approach taken to alleviate the *pattern-dependence* problem is to assume some simplifying conditions about the input space and to attach user specified probabilities of transition [16] to the input nodes; probabilities that get propagated to all nodes of the circuit. In these methods two different assumptions might be made: a *spatial independence*, in which case input signals on the same clock cycle are assumed to be non-correlated and a *temporal independence*, in which case the value of the same signal in two consecutive clock cycles is assumed independent. Once the transition probabilities are computed using probabilistic [18] or statistical [7] techniques, the energy can be computed as $E_{av} = \frac{1}{2}V_{dd}^2 \sum_{i=1}^{m} C_i P(x_i)$, where C_i is the total capacitance at node x_i, $P(x_i)$ is the transition probability at node x_i and m is the total number of circuit nodes that are outputs of logic gates (the power can be computed as $P_{av} = \frac{E_{av}}{T}$, where T is the clock period). However, this estimation is a lower bound on the consumed energy, since there will be at most one transition per clock cycle for a given node; thus, glitching energy is not included. Some methods [22] use *transition density* instead of *transition probability* of a node to capture the glitching activity of the circuit.

All these methods are still pattern-dependent to some extent, except that now the user must supply information (in terms of probabilities) about a typical behavior at the circuit input. Furthermore, these techniques use simplified delay models which make the estimation of glitching energy (in particular) more prone to error.

A.4.2 Energy estimation in the asynchronous context

Our method of energy estimation is also based on simulation [14]; however, as opposed to the synchronous design case, our approach practically is *not* input *pattern-dependent* due to the QDI circuit design methodology. In the following, we describe why we believe our energy simulation method of QDI circuits yields superior accuracy and performance when compared to simulators of synchronous circuits of the same complexity.

CMOS circuits have three main sources of energy dissipation: dynamic currents (due to charge and discharge of capacitors), short-circuit currents and leakage currents. Some existing simulation tools include the energy dissipation due to short-circuit currents. We have found that—for the types of QDI circuits we are interested in—they are not an important source of energy dissipation. Leakage currents are due to the subtreshold behavior of the CMOS transistor. At the current state of technology they contribute a small fraction to the total energy consumption. For these reasons, for the purpose of this work we consider the contribution of both short-circuit currents and leakage currents to be null.

As in many other energy estimation tools, we focus our attention to the dynamic energy consumption

$$E_{dynamic} = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{m} C_i n_i \tag{A.1}$$

where C_i is the total capacitance at node i, n_i is the *transition count* of node i during the measured period of time and m is the total number of circuit nodes. The main difficulty with this formula for synchronous systems comes from estimating n_i. In the following, we will show how our method for estimating energy in QDI circuits deals with this difficulty.

A.4.3 Energy estimation error due to glitching

As mentioned earlier, energy simulation of synchronous circuits is complicated by spurious transitions (glitches). The main reason is that glitches are highly delay dependent; any attempt to properly estimate the energy due to them has to include an accurate timing model. For QDI circuits, the design methodology explicitly avoids glitches by enforcing the monotonicity of signal transitions [30]. As a consequence, the energy consumed due to glitches is null and there is no need to complicate the energy simulator with a timing model.

A.4.4 Energy estimation error due to *spatial dependence* (input correlation)

In a QDI circuit, data is encoded as dual-rail or 1-of-N signals. In general, we make no assumption about which rails of a data channel (as opposed to control channel) would be exercised more often. As a result, the corresponding CMOS circuitry is symmetric—in terms of transistor sizes—for each data rail within a channel. Furthermore, each activated node transitions exactly twice (not all signals become active, since some of them are inherently mutually exclusive). For these reasons, these types of circuits consume the same amount of energy, independently on the data being processed. Most of the QDI circuits we are interested in are of this type.

One important exception are adders. Consider a carry-lookahead adder. In such a circuit, the propagate rails have more load than the kill and generate rails, since—for uniform input distribution—a transition on the propagate rail is more likely. This effect generates some input pattern dependence on the consumed energy. However, when put in the realistic context of the QDI design (input/output completions, data/control acknowledges, internal enable) this dependence is very weak. In case of an `add` instruction of the MiniMIPS [33], the difference between the worst case (all propagates) and best case (no propagates) energy dissipation is 1.53% of the average case; correspondingly, for an `addi` instruction the same relative difference is 1.49%.

For control channels, the dependence could be stronger. This is expected, since different values of the control might activate different parts of the circuit. Consider an adder again. In case of a regular add instruction, the two inputs used are coming from the register file; in the case of an add immediate instruction one input comes from the register file, and the other from the immediate bus, and (depending on the implementation) this could take a different amount of energy. For these reasons, the energy corresponding to each distinct control signal of a given circuit block has to be estimated individually.

A.4.5 Energy estimation error due to *temporal dependence* (cycle-to-cycle dependency)

In a synchronous circuit, the switching of a signal depends not only on the data value on the current cycle, but also on the previous value of that node, causing a cycle-to-cycle dependency.

For a QDI circuit, each node that will be active during an operation will go through exactly two transitions (a charging and a discharging transition). By the end of the cycle, each node will be in the same state as it was at the beginning of the cycle—independently of the performed operation. This property eliminates cycle-to-cycle dependencies altogether.

There is one important exception, namely the case of writing state variables implemented as flip-flops. If the state bit has the same value as the value to be written to it, the bit will not flip. On the other hand, if the two values are different, the bit will transition and as a result energy will be consumed. Thus, depending on the initial condition of the state variable, an energy consuming event might or might not occur, causing a temporal dependence.

One way to deal with this problem is to simulate the system for the worst (all bits changing) and best (no bits changing) case and take as energy estimate the average of the two numbers. In practice, when such a flip-flop is integrated in its actual QDI environment, its energy contribution becomes small compared to the total energy dissipation. For example, the relative difference between the worst case and best case write to a register in the register file of the MiniMIPS is 6.44% of the average case. Thus, if it is assumed that on average 50% of the state bits change the worst case error will be $\pm 3.22\%$.

All in all, the energy variability due to temporal or spatial dependence of input data is localized and relatively small; thus, in general it can be ignored.

A.5 The `esim` simulator

A.5.1 Energy model used by `esim`

The `esim` energy simulator estimates the dynamic energy consumption of a QDI circuit using Equation A.1.

The general operation of the simulator is as follows. Once the simulator is launched on a closed PR set, all nodes are initialized and assigned an energy `weight`. At any time, the ready queue contains all PRs that are ready to be fired, i.e. whose guards are true. A step of the simulator consists of taking one PR out of the ready queue and firing it, i.e. changing the value of the variable the PR operates on. This firing may cause other PRs to become enabled, in which case their PRs are added to the ready queue. Which PR is removed from the ready queue on a firing depends on the timing settings used (timed or random) to initialize the simulator. If the PR just taken off the ready queue is in the scope of the monitored set, its assigned energy weight is added to a counter that keeps track of the total energy consumed. Once the simulation is terminated, the energy counter contains the total energy consumed by the monitored region of the circuit.

There are two unknowns in Equation (1): the node capacitance C_i and the transition count n_i. The node capacitance C_i is determined statically—as will be explain later—when the simulator is launched. The transition count n_i is being taken care of automatically by the simulator: for each actual transition, the corresponding weight is added to an energy counter. Since our simulator handles transitions explicitly, each and every energy consuming signal switching is recorded.

As shown in Figure A.1, each node in a PR set could have the following capacitive components: source/drain diffusion capacitance due to the PR's pull-ups and pull-downs (C_d), wiring capacitance (C_w), capacitance due to a staticizer ("keeper") if the node is state-holding (C_s) and gate capacitance due to the transistors the node is driving (C_g). There are other capacitive nodes present in the actual circuit that are not represented at PR level. They are mainly internal nodes of transistor chains. We choose to ignore the energy dissipated to charge and discharge these internal nodes.

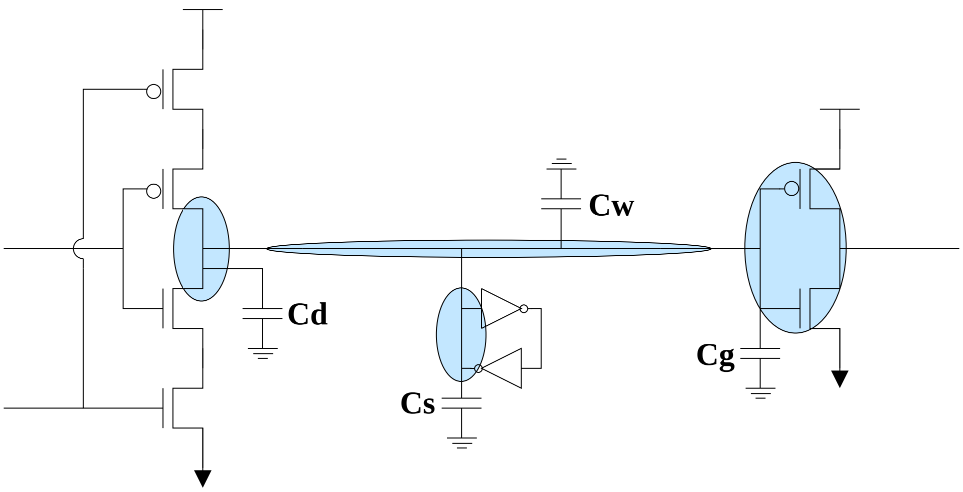

Figure A.1: `esim` capacitance model

Depending on the level of detail of the simulated design, each capacitive component may or may not be available. If only the unsized PRs are available, all capacitive terms are ignored except C_g that is estimated to be proportional to the actual (i.e. considering gate sharing) fanout of the corresponding node.

If a sized but unwired PR set is available for simulation, all capacitive components except C_w can be computed with good accuracy. The dynamic energy consumed by a node i can be written as $E = E_{C_g} + E_{C_d} + E_{C_s} = K * (\sum transistor\ width)$ where the technology-dependent constant K can be computed directly or calibrated using `hspice`. When `esim` is first run on a PR set, the value of $(\sum transistor\ width)$ is computed for each node and assigned to its corresponding `weight` variable in the data structure.

If a wired-up layout exists, the wire information can be added as explicit PRs into the original PR set. Given the capacitance of the wires (computed by the layout extractor) one has to transform these values into *transistor width* units. Once this is done, wires are treated by `esim` exactly as any other PR pair.

There are several energy dissipating sources `esim` is ignoring. Possibly the most important one is the energy consumed due to leakage currents. Some researchers have reported the increased importance of this energy dissipation source for submicron technologies. Energy consumption due to short-circuit currents is ignored as well. Our evaluation shows that in typical QDI circuits the energy dissipated due to short-circuit currents is no more than 1.5% of the total energy consumption. There are two elements relating to dynamic energy consumption that are ignored. First, energy due to charge-sharing is not captured, and second, the energy spent on charging and discharging internal nodes of transistor stacks are not accounted for. Our `hspice` simulations show that these dynamic energy dissipation sources can be safely ignored for most of the circuits we are interested in (in current technology).

A.5.2 Timing model used by `esim`

`esim` does not need a timing model for energy estimation. For this reason, the simulator does not explicitly compute the timing information of the PRs. However, if timing estimate is needed, the timing of each PR can be passed to `esim` using *after delay* rules; for example: *after* 123 $a \wedge\ b \mapsto \_c \downarrow$ i.e., $\_c$ will fire 123 time units after a and b become true. If no *after rules* are present, either random or unit timing is assumed depending on the simulation settings. The after rules can be assigned by a preprocessing phase using either a simple τ-model estimation or using an `hspice` characterization table. The same after rules can be used to assign RC delays to wires. There is extensive research done in accurately predicting the timing behavior of transistor netlists; however, for many of our experiments, it is sufficient to use a simple timing model based on counting transitions on the critical path, i.e. using `esim` with unit timings.

A.5.3 Accuracy and speed of `esim`

We have evaluated the accuracy of `esim` in estimating energy on several test circuits ranging from 192 to 2584 transistors, by comparing the results to the `hspice` results recorded on the same circuits. We have done our evaluation in the HP 0.6μ CMOS technology. We have evaluated different levels of detail in implementing the node capacitance computation. In the first alternative, we have included only the gate capacitance C_g in the energy computation. For the tested circuits the error ranges from 16.46% to 22.28%. In the second alternative, we have added the source/drain diffusion capacitance C_d together with C_g and the resulting error was reduced to the range of 2.17% to 14.96%. In the last alternative, we have added the energy consumption due to staticizers C_s together with C_g and C_d. The conclusion was that if C_g, C_d and C_s were included, the resulting error was in the range of 1.45% to 7.25%. From these results we have concluded that if the terms C_g, C_d and C_s are available to the simulation, the error in energy estimate between `hspice` and `esim` is less than 10%. The salient assumption of this claim is that wire capacitance (C_w) is an order of magnitude smaller than gate capacitance for the considered design. This assumption is true for most of the circuits sized for high speed, such as the MiniMIPS microprocessor. However, if this assumption is violated, wires have to be considered explicitly through PRs.

The discrepancy is due to all other sources of energy consumption that were ignored, mainly the wire capacitance C_w—not included in the simulated PR set.

The speed of `esim` is outstanding: more than three orders of magnitude faster than `hspice`. One instruction of the MiniMIPS (more than 2 million nodes) can be simulated in less than 1 second on a Pentium III Xeon 550MHz.

A.6 Energy Estimation of the MiniMIPS

In this section, we apply the `esim` simulator to obtain a breakdown of the energy consumption of the MiniMIPS. We believe that all the figures obtained by these simulations are no more than 10% smaller than the actual energy consumption figures of the fabricated chip. The MiniMIPS processor was fabricated in an HP 0.6μ CMOS process; the quoted numbers refer to this technology. Given the technology independent PR representation of QDI circuits, it is straight-forward—as long as the assumptions about leakage and short-circuit currents hold—to extend the simulation tool to other technologies by simply recomputing the constant K for the target technology.

We have divided the processor into 15 functional units, shown in Figure A.2. These units are: Icache (4K pipelined instruction cache core), ICCTRL (instruction cache control, tag comparison), PRE (instruction decode), ITOSS (mispredicted instruction filter used for branch prediction), fetch (pc computation), register file (32 32-bit registers with locking, one-entry register bypass), source bus (instruction bus, source operand bus, immediate bus), adder, shifter, fblock (logical-function block), muldiv (array multiplier, iterative divider), DCCTRL (data cache control, exception pc queue, CP0 registers), Dcache (4K pipelined data cache core), result bus, WB (writeback unit).

In the following subsection, we describe the behavior and the associated energy breakdown of some generic groups of instructions.

A.6.1 `nop`

On a `nop`, the instruction is looked up in the instruction cache (Icache) based on the pc received from the fetch. Once the instruction cache control (ICCTRL) determines that the cache entry is valid, the instruction is sent to the decode (PRE). If the instruction is not a mispredicted branch, it passes through a filter (ITOSS) and is allowed execution. In parallel, the fetch is notified by the decode about the nature of the instruction such that it can compute the next pc. The register file is also notified about the `nop` instruction. In order to properly restore processor state in case of an exception or interrupt, the writeback unit (WB) is notified and the program counter is saved through the epc queue which is part of the data memory system (DCCTRL).

The total energy consumed by a `nop` executing from cache is 20.63nJ—the corresponding energy breakdown is shown in Figure A.3 (an instruction cache miss adds 23.28nJ to the currently executing instruction). About half of the energy goes into bringing the instruction from the cache, and about one quarter of it goes into computing the next pc, and decoding and filtering the instruction. The epc queue ($\subset$ DCCTRL) consumes a relatively high percentage of the total energy due to the high amount of buffering needed to perform its function. The writeback unit consumes

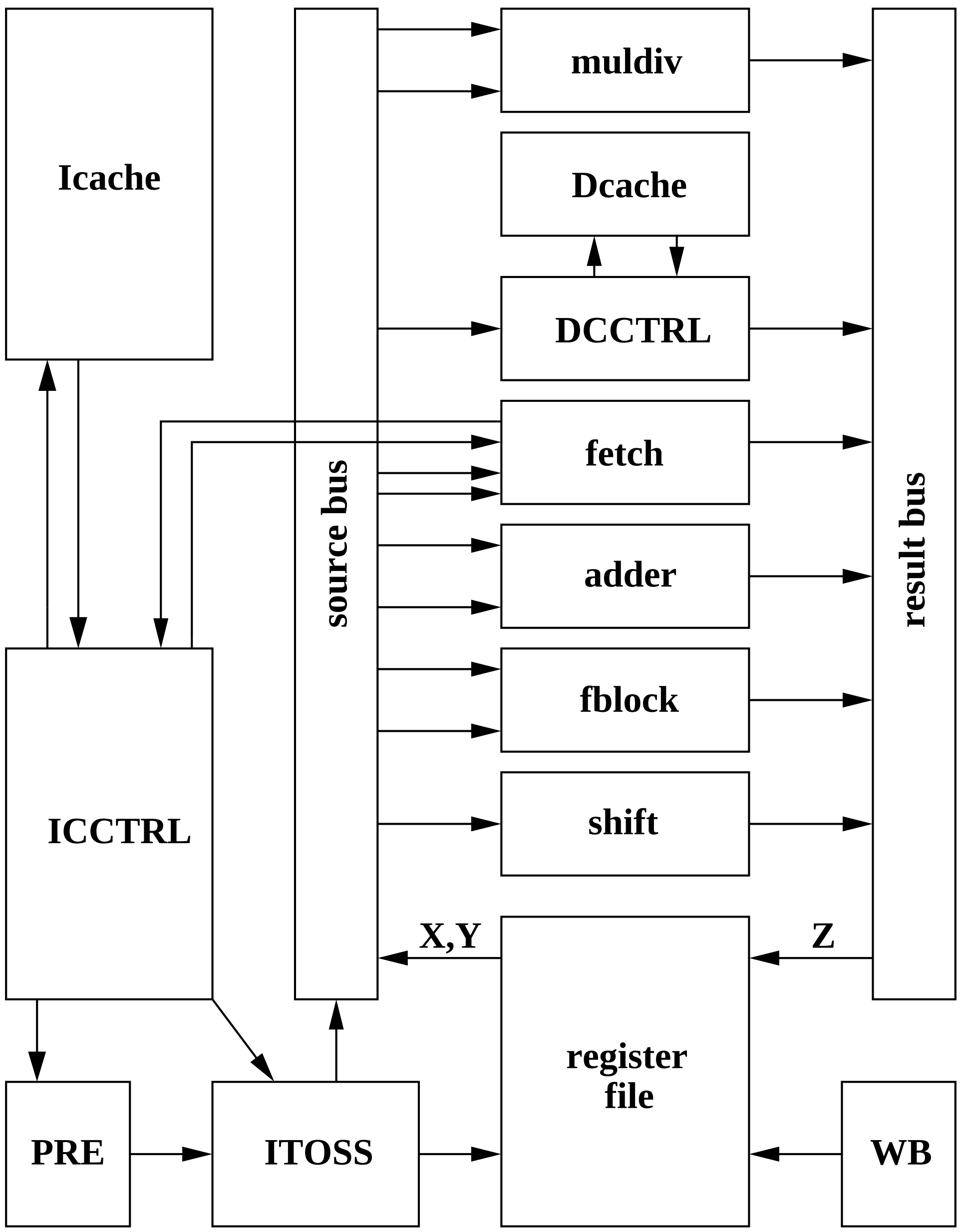

Figure A.2: Block diagram of the MiniMIPS

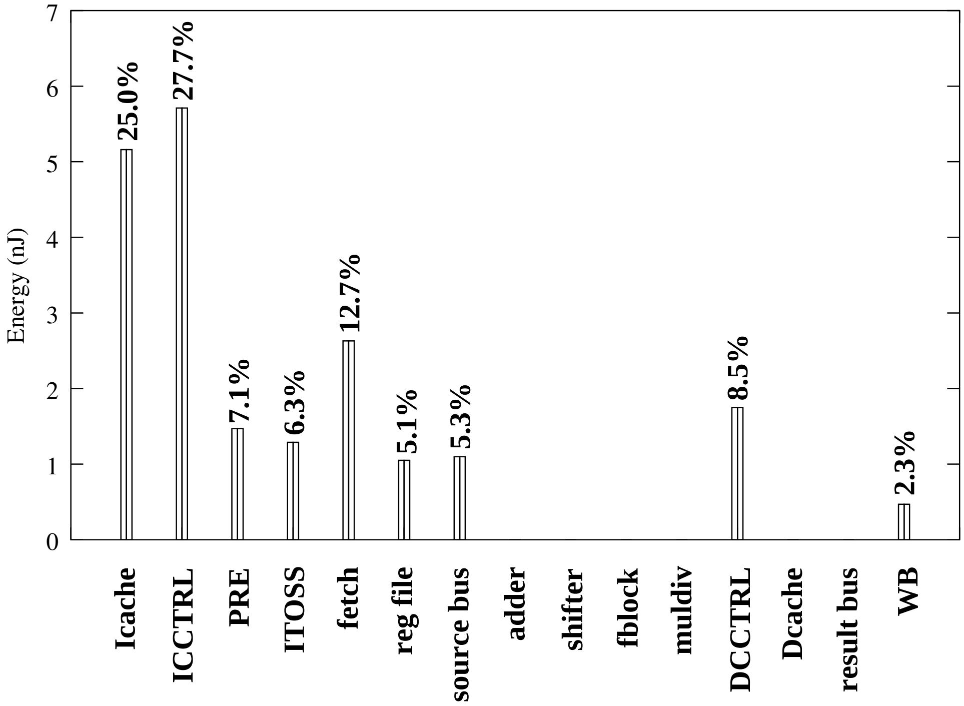

Figure A.3: Energy of a `nop` instruction

relatively little energy, not only for `nop`'s but for any other instruction. As previously mentioned, units not involved in the execution of the current instruction consume no energy; for a `nop` this is the case for 6 out of the 15 blocks of the processor. Even the units consuming energy might not consume their peak energy—an example being the register file which consumes only 1.10nJ while its peak energy consumption is 9.15nJ.

A.6.2 Arithmetic instructions

There are 4 different execution units in the MiniMIPS: the adder-subtracter, the logical-function block, the shifter and the multiplier-divider. Each instruction executed by one of these units is performed conceptually as follows. Once the instruction is launched for execution, the decode requests the register operands from the register file. Upon arrival of the operands, the execution unit performs the requested operation, and delivers the result to the register file on the result bus (except for the muldiv which writes the result to its local Hi/Lo registers). If the execution unit could potentially raise an exception (adder) the exception information is sent to the writeback. The operation result is passed to the register file bypass and eventually written back to the register file core once the writeback allows it.

In order to determine the energy cost of a given arithmetic instruction, it is necessary to know where the operands are coming from within the register file: bypass, core or both. As a results, depending on the context, the same instruction could consume different amounts of energy. We have determined the energy cost of the following individual operations:

operation	bypass	core	total
read from bypass	0.54nJ	—	0.54nJ
read from core	0.74nJ	2.11nJ	2.85nJ
write	1.53nJ	0.82nJ	2.35nJ
reg. file control	—	—	1.10nJ

Based on these individual results, a register file operation could take anywhere from 1.64nJ (reading one operand from the bypass) to 9.15nJ (reading two operands from the core and writing back one result).

The arithmetic instructions—based on their energy consumption—can be divided into (8) groups, group A: `add`, `addu`, `sub`, `subu`, `slt`, `sltu`, group B: `addi`, `addiu`, `slti`, `sltiu`, group C: `sll`, `sra`, `srl`, group D: `sllv`, `srav`, `srlv`, group E: `and`, `nor`, `or`, `xor`, group F: `andi`, `ori`, `xori`, group G: `mult`, `multu` and finally group H: `div`, `divu`. The breakdown of the energy consumption of each group of instruction is shown in Figure A.4; the energy consumed by the muldiv unit was reduced by 100× to fit the graph.

While the energy spent in manipulating the instruction (cache, decode, fetch, epc queue, WB) is the same as for a `nop`, the cost of executing the instruction is different. For example, the total energy consumed by an `add` reading both operands from the register core is 34.33nJ, reading one operand from the register core and the other from the bypass is 32.02nJ and finally, reading both operands from the bypass is 30.21nJ. The energy consumed by the adder is less that 10% of the total energy of a group A

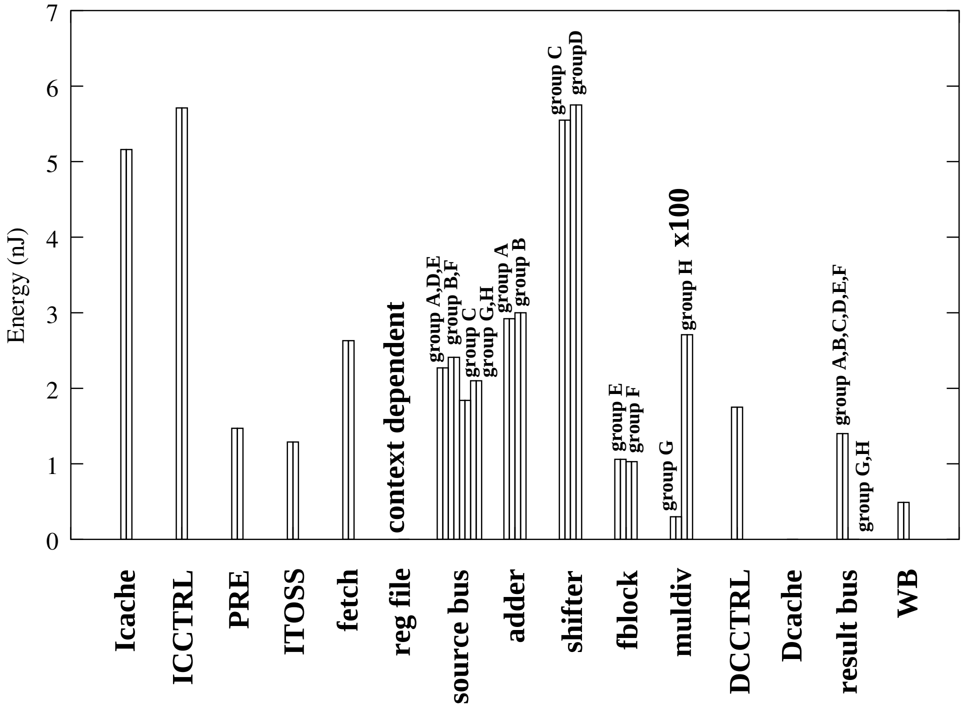

Figure A.4: Energy of an arithmetic instruction

or B instruction. The shifter consumes about 2 times, and the fblock about 1/3 of the adder's energy. The multiply part of the muldiv consumes about 10 times more energy than the adder, while the divider part of the muldiv consumes about 90 times more than the adder.

A.6.3 Control-flow instructions

There are two kinds of control-flow instructions in the MiniMIPS: unconditional jumps and conditional branches. An unconditional jump computes the pc of the next instruction either by concatenating the high order 4-bits of the delay slot's address, 26-bits of immediate and 2 zero bits (`j, jal`) or directly from a register (`jr, jalr`).

A conditional branch has the branch target address computed from the sum of the delay slot's address and a 16-bit offset, shifted left two bits and sign-extended to 32-bits. Depending on the type of the conditional branch, the contents of two registers or one register and zero are compared. In the MiniMIPS, each conditional branch

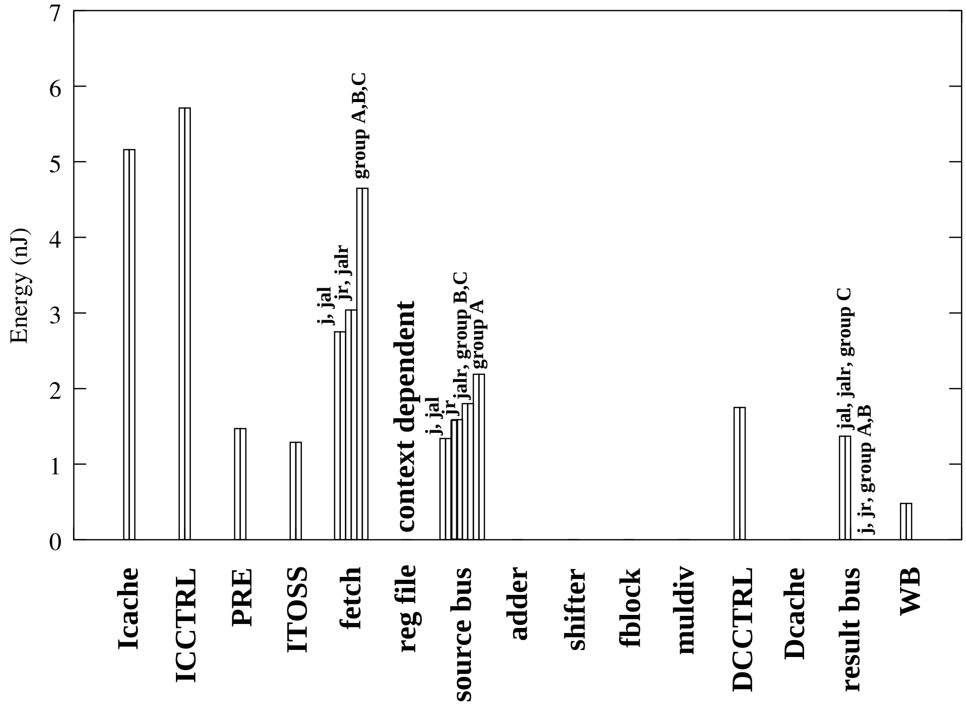

Figure A.5: Energy of control-flow instruction

is predicted taken if it is a backward branch or not taken if it is a forward branch. There are different energy costs for a correctly predicted branch and a mispredicted branch. Conditional branches—based on their energy consumption—can be divided into 3 groups, group A: `beq`, `bne`, group B: `bgez`, `bgtz`, `blez`, `bltz`, and group C: `bgezal`, `bltzal`. Figure A.5 shows the energy breakdown for unconditional jumps and correctly predicted conditional branches. Conditional branches are comparable in energy consumption to less costly (E, F group) arithmetic instructions.

In case of a mispredicted branch, the instruction is canceled in the ITOSS unit and the fetch resends the—now correct—pc to the instruction cache. In this way, only the instruction fetching consumes energy (including the epc queue), while the rest of the processor does not. Once the correct instruction is brought back from the cache, the execution of the branch proceeds as in the correctly predicted case, except now the fetch consumes 6.04nJ for all conditional branches. The total energy cost of branch misprediction is 21.49nJ, just a bit more than that of a `nop`.

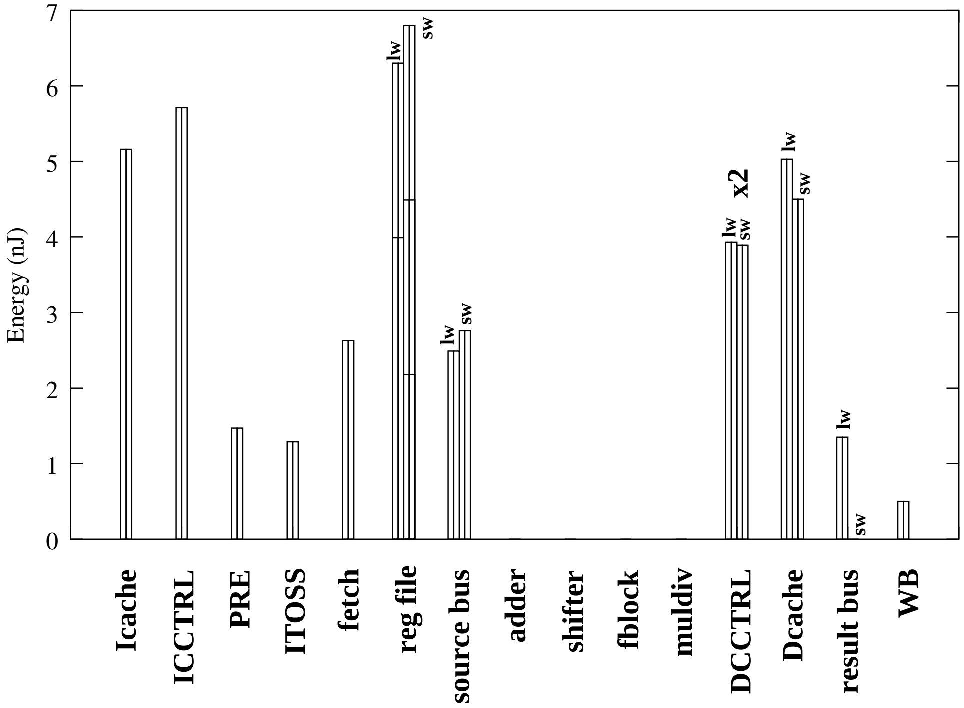

Figure A.6: Energy of load/store instruction

A.6.4 Load/store instructions

Another group of instructions are the load/store's. The MiniMIPS implements only word operations to/from memory: load word (`lw`) and store word (`sw`). Their energy breakdown is shown in Figure A.6. A data cache miss adds 30.86nJ to the current load instruction.

A.6.5 `esim` agreement with lab data

Thanks to Mika Nyström, we have collected lab data of the total energy consumed by the fabricated chip running certain instructions. The next table compares these results with the results estimated using `esim`.

instr	lab	`esim`	error
`nop`	22.71nJ	20.63nJ	9.16%
`addiu r2, r1, 0`	34.06nJ	31.57nJ	7.31%
`sll r1, r0, 0`	34.85nJ	33.58nJ	3.64%
`or r2, r1, r1`	33.78nJ	32.32nJ	4.32%
`ori r2, r1, 0`	31.04nJ	29.60nJ	4.64%
`mult r1, r1`	61.19nJ	56.77nJ	7.22%
`lw r2, 0(r1)`	44.87nJ	41.51nJ	7.48%

This comparison shows that the error between lab data and `esim` is as expected—given the accuracy of `hspice`—within 10%.

The application of `esim` to the PR set of the MiniMIPS gave us important insights in the specifics of energy consumption in an asynchronous processor. This knowledge will be useful for the contemplated redesign of the MiniMIPS.

A.7 Conclusion

We have presented a method and a simulator for the accurate energy estimation of QDI circuits.

The method is based on the "production-rule" (PR) notation, which gives an accurate logical description of the circuit. The PR set of a system can be either synthesized from a high-level description or extracted from layout. The `esim` simulator is both simple and efficient and can be parallelized easily. It can be used for logical, timing, energy, or combined Et^2 [33] simulation.

The models for capacitance and delays can be as simple or as sophisticated as necessary, and can be either estimated from a high-level description or extracted from layout. The accuracy of the physical modeling of switching activity is greatly improved by the absence of glitches and other beneficial properties of QDI circuits. Of course, estimating interconnect capacitances accurately remains a problem.

The method has been successfully applied to the energy estimation of a complete asynchronous 32-bit MIPS microprocessor, demonstrating the efficiency of the method for the simulation of large systems. The results also shed light on the energy budget of such a processor. In particular, it shows that 90% of the energy is consumed in communication and only 10% in actual execution of instructions.

These results and the accurate energy breakdown will be useful for the contemplated redesign of the asynchronous MIPS, which will be optimized for Et^2. We expect greatly improved energy consumption compared to the present design.