Chapter 1

# $ET^2$ : A METRIC FOR TIME AND ENERGY EFFICIENCY OF COMPUTATION

Alain J. Martin
*Department of Computer Science*
*California Institute of Technology*
alain@cs.caltech.edu

Mika Nyström
*Department of Computer Science*
*California Institute of Technology*
mika@cs.caltech.edu

Paul I. Pénzes
*Department of Computer Science*
*California Institute of Technology*
penzes@cs.caltech.edu

**Abstract**

We investigate an efficiency metric for VLSI computation that includes energy, $E$, and time, $t$, in the form $Et^2$. We apply the metric to CMOS circuits operating outside velocity saturation when energy and delay can be exchanged by adjusting the supply voltage; we prove that under these assumptions, optimal $Et^2$ implies optimal energy and delay. We give experimental and simulation evidences of the range and limits of the assumptions. We derive several results about sequential, parallel, and pipelined computations optimized for $Et^2$, including a result about the optimal length of a pipeline.

We discuss transistor sizing for optimal $Et^2$ and show that, for fixed, nonzero execution rates, the optimum is achieved when the sum of the transistor-gate capacitances is twice the sum of the parasitic capacitances—not for minimum transistor sizes. We derive an approximation for $Et^n$ (for arbitrary $n$) of an optimally sized system that can be computed without actually sizing the transistors; we show that this approximation is accurate. We prove that when multiple, adjustable supply voltages are allowed, the optimal $Et^2$ for the sequential composition of components is achieved when the supply voltages are adjusted so that the components consume equal power. Finally, we give rules for computing the $Et^2$ of the sequential and parallel compositions of systems, when the $Et^2$ of the components are known.

# 1. Introduction

With energy becoming as important as time as a criterion for computational efficiency, analytical tools are needed to evaluate computations according to both criteria simultaneously. How are energy and time traded against each other in the design process? How are algorithms compared when they have different energy and time figures? A useful metric must separate the algorithmic tradeoffs of energy against time from the physical (usually electrical) tradeoffs.

We propose an efficiency metric for VLSI computation that combines energy, $E$, and time, $t$, in the form $Et^2$. The choice of this metric is based on CMOS VLSI technology: in CMOS, $Et^2$ is independent of the voltage in first approximation. Instead of attempting to optimize a circuit for both $E$ and $t$ , the designer can now optimize the design for the single metric and adjust the voltage to obtain the chosen tradeoff between $E$ and $t$ .

We prove that the $Et^2$ metric is optimal for CMOS circuits under assumptions that hold approximately in the normal range of operation. Under those assumptions, energy and delay can be freely exchanged through supply-voltage adjustment. Although the metric is not adequate over the entire range of operation for CMOS transistors, we have experimental evidence that a large class of circuits exhibit a collective behavior that is more regular than that of individual transistors. We also investigate when and why the $Et^2$ metric is inadequate; in some cases, we can use the metric $Et^n$ with $n \neq 2$ . We shall see that most results we prove for $Et^2$ generalize for $Et^n$ with $n \neq 2$ .

The objection that a metric grounded in a specific technology (CMOS) cannot be general enough for the study of algorithms can be answered by observing that the CMOS model of computation is certainly as general as the "random-access machine" model, which has been used successfully in the traditional analysis of algorithms.

The $Et^2$ metric was originally introduced for the design of an asynchronous MIPS R3000 microprocessor [4]. The arguments about the validity of the metric and the analysis of the pipeline were published by Martin [3]. The results about transistor sizing for minimal $Et^n$ have been described previously by the authors [7, 11].

## 2. Energy and Delay in VLSI Computations

Our study of energy and delay in computations is based on CMOS implementations. We consider a digital VLSI computation to be a partially ordered sequence of transitions. Each transition changes the value of a boolean variable of the computation. We consider irreversible computations only, i.e., computations in which assigning a value to a variable destroys the previous value of the variable.

In digital CMOS, a boolean variable is implemented as an electrical node whose voltage represents the present value of the variable. Each transition charges or discharges the capacitor attached to the node, bringing its voltage either to the supply voltage *Vdd* or to the ground voltage *GND*. We are interested in the *computational*, or *dynamic*, energy spent in charging and discharging the capacitances of all nodes involved in a computation. We ignore leakage energy and short-circuit energy. Since we ignore leakage, we also assume that no energy is spent maintaining the values of variables stored in registers.

Any algorithm can be implemented as a set of *production rules* [5]. A production rule is of the form: $B \to t$, where $B$ is a boolean expression, and $t$ is a single assignment of the value **true** or **false** to a boolean variable. Such an assignment is called a *transition;* in this example, the transition $t$ is performed after $B$ becomes **true**. A *logic gate* or *operator* is the physical implementation of the pair of production rules that set and reset a given variable, of the following form:

$$\begin{array}{lcl} Bu & \to & z\uparrow \\ Bd & \to & z\downarrow \end{array} \tag{1.1}$$

Let $E_{z\uparrow}$ and $E_{z\downarrow}$ be the energy spent firing the first and second production rules, respectively. (A firing is an execution of a production rule that changes the value of a variable.) The energy spent firing production rule $Bu \to z\uparrow$ is the energy dissipated charging the capacitor $C_z$ associated with the node of $z$. The total energy required for charging the capacitor up to voltage $V$ is $C_z V^2$;

half of this energy is stored in the capacitor, and half is dissipated as heat in the pull-up network connecting the capacitor to the power supply. We have that

$$E_{z\uparrow} = \frac{C_z V^2}{2}, \tag{1.2}$$

where $V$ is the power-supply voltage. When the capacitor is discharged to ground, the energy stored in the capacitor is dissipated in the pull-down network. Hence, $E_{z\downarrow} = E_{z\uparrow}$.

We shall not elaborate on the calculation of the capacitance $C_z$ beyond noting that $C_z$ depends mostly on the "load" of $z$, i.e., the topology of the logical gates of which $z$ is an input, and hardly depends on the structure of the logical gate of which $z$ is an output. In other words, the energy consumed in computing the value of $z$ does not depend on what is computed but rather on where the result of the computation is needed.

The delay $t_{z\uparrow}$ for firing $z\uparrow$ is the ratio of the final electrical charge $Q_z$ on $C_z$ to the current $i_{z\uparrow}$ available for charging $C_z$:

$$t_{z\uparrow} = \frac{Q_z}{i_{z\uparrow}}, \tag{1.3}$$

with $Q_z = C_z V$. The current $i_{z\uparrow}$ is the current flowing in the transistor network connecting the constant power-supply to $z$ when and only when $Bu$ holds; similarly for the delay $t_{z\downarrow}$ and current $i_{z\downarrow}$.

In general, the transistor current is difficult to analyze. Let us look first at one single nMOS-transistor as pull-down network. (The analysis for a pMOS transistor as pull-up network is similar.) We assume that the transistor is above threshold ($V_{gs} > V_t$), and not in velocity saturation. Then, the current is either the saturation current, $I_s$, when $V_{ds} \geq V_{gs} - V_t$; or it is the linear current, $I_l$, when $V_{ds} < V_{gs} - V_t$, where $V_{gs}$ and $V_{ds}$ are the gate-to-source and drain-to-source voltages of the transistor, respectively, and $V_t$ is the threshold voltage.

The formulas for $I_s$ and $I_l$ are well-known:

$$I_l = k\left(2(V_{gs} - V_t)V_{ds} - V_{ds}^2\right) \tag{1.4}$$

$$I_s = k(V_{gs} - V_t)^2 \,. \tag{1.5}$$

If we assume that the voltages $V_{gs}, V_{ds}, V_t$ vary proportionally to the supply voltage $V$, then both $I_l$ and $I_s$ depend quadratically[1] on $V$, and therefore the

[1] Of course, $V_{gs}$ and $V_{ds}$ must vary by quite a different mechanism from the one governing $V_t$: $V_{gs}$ and $V_{ds}$ can vary "automatically" as a result of changing *Vdd*, whereas $V_t$ must be set at the time of fabrication. The main reason that the proportional variation breaks down is that it is in practice impossible to scale $V_t$ with *Vdd* because that would lead to unacceptably large leakage currents at the low end of the scale.

current $i_{z\downarrow}$ discharging $Q_z$ is of the form $i_{z\downarrow} = K_{z\downarrow}V^2$ . Similarly, for a pull-up network, we have $i_{z\uparrow} = K_{z\uparrow}V^2$ . Hence, we have for the delay $t_{z\uparrow}$ that

$$t_{z\uparrow} = \frac{C_z}{K_{z\uparrow}V} \; . \tag{1.6}$$

Combining the expressions for delay and for energy, we see that the expression $E_z t_z^2$ is independent of $V$.

Under certain restrictions, it is possible to extend the result that the current is quadratic in $V$ to cover the arbitrary composition of pullup and pulldown networks. Papadantonakis has proved this result for a class of circuits called "smooth circuits" [8]. A smooth circuit is a network of transistors in which each node has a capacitance to ground, the power supplies are modeled as large capacitors, and again the threshold voltage is assumed to scale with the supply voltage.

If we assume that a CMOS circuit is a reasonable approximation of a smooth circuit, we can assume that the quadratic relation between currents and supply voltage $V$ holds, and therefore that the delays are inversely proportional to $V$. For those circuits, $Et^2$ is independent of $V$, where $E$ is the dynamic energy dissipated by a computation, and $t$ represents either the latency or the cycle time of the computation. We shall return to the limitations of this assumption.

## 3. Comparing Algorithms for Energy and Delay

Given two algorithms $A$ and $B$, with energy and delay $(E_A, t_A)$ and $(E_B, t_B)$, how do we compare them for energy and delay? In evaluating the time efficiency of a computation, we may be interested in either one of two delay parameters: the latency and the cycle time. For an algorithm computing the function $F$, the latency is the delay between the input of parameter $x_i$ and the output of $F(x_i)$, averaged over all values of $i$. The cycle time is the delay between the input of parameter $x_i$ and the input of the next parameter $x_{i+1}$, again averaged over all values of $i$.

### 3.1 Why *Et* is not the Right Metric

The energy-delay product $Et$ is often used for comparing designs, but it is not usually an acceptable metric, as we shall presently demonstrate.

Let us assume that we have two circuits, $A$ and $B$, that compute the same thing in two different ways. Assume $E_A = 2E_B$ and $t_A = \frac{t_B}{2}$. Then, according to the $Et$ metric, $A$ and $B$ are equally good. But let us reduce the supply voltage of $A$ by half. Let $(E'_A, t'_A)$ be the new values of energy and delay for $A$. Given the dependence of energy and delay on voltage, Equations 1.2 and 1.6, we have that

$$E'_A = \frac{E_A}{4}, \tag{1.7}$$
$$t'_A = 2t_A, \tag{1.8}$$

which gives

$$E'_A = \frac{E_B}{2}, \tag{1.9}$$
$$t'_A = t_B. \tag{1.10}$$

Hence, $A$ now has the same delay as $B$ but at only half the energy, and therefore $A$ is a better implementation than $B$, contrary to what the $Et$ metric indicates.

These results are borne out in practice [10]. In Table 1.1, we see the results of simulating two different implementations of an eight-bit comparator with the simulator HSPICE. 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 *Vdd,* 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?

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 percent. This example illustrates that the $Et^2$ metric is more trustworthy for circuit comparisons when we are allowed to adjust the supply voltage.

## 3.2 The Θ Metric

Let us now ignore the lower and upper bounds imposed on the voltage by the technology and assume that we can always trade $E$ and $t$ against each other through voltage adjustment. Suppose that under these conditions there exists a function $\Theta(E, t)$ with the properties:

**Property** $\Theta 1$**.** $\Theta$ is monotonically increasing in $E$ and $t$,

**Property** $\Theta 2$**.** $\Theta$ is independent of $V$.

| 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 (3.3 V) | 25.24 | 3.93 | 99.21 | 389.97 |
| Log (3.3 V) | 44.97 | 2.35 | 105.50 | 247.57 |
| Log (2.15 V) | 16.52 | 3.93 | 64.97 | 255.59 |

*Table 1.1.* Comparison of $E$, $t$, $Et$, and $Et^2$ of two kinds of 8-bit comparators. Simulations with HSPICE using parameters for HP's 0.6-$\mu$m CMOS process (via MOSIS).

**Theorem 1** *Given two computations $A$ and $B$ with corresponding $\Theta_A$ and $\Theta_B$:*

- *If $\Theta_A < \Theta_B$ then $A$ is more delay-efficient than $B$ when $A$ and $B$ use equal energy and $A$ is more energy-efficient than $B$ when $A$ and $B$ have the same delay.*

- *If $\Theta_A = \Theta_B$ then $A$ is equivalent to $B$ with respect to energy when their delays are the same.*

**Proof:** Through supply-voltage adjustment, we can equalize either the energy or the delay of the two computations. Let us arbitrarily choose to equalize the delays: $t_A = t_B$. Because of Property $\Theta$2, the $\Theta$s have not changed.

We can now compare the two computations, thanks to Property $\Theta$1:

- $(\Theta_A < \Theta_B) \Rightarrow (E_A < E_B)$, i.e., $A$ is better than $B$,

- $(\Theta_A = \Theta_B) \Rightarrow (E_A = E_B)$, i.e., $A$ and $B$ are equally good.

Hence, for any chosen delay $t$, $A$ is more energy-efficient than $B$. Likewise, for any chosen energy $E$, $A$ is more time-efficient than $B$. $\square$

## 3.3 The $Et^2$ Metric

Any expression in $E$ and $t$ that is monotonically increasing in $E$ and $t$ and that is independent of $V$ can be used as complexity metric $\Theta$. We have shown in Section 2 that, in CMOS technology, the following definition for $\Theta$ is valid:

$$\Theta \stackrel{\text{def}}{=} Et^2 \; . \tag{1.11}$$

Henceforth, $\Theta$ will always mean $Et^2$ . If we now return to the example at the beginning of Section 3.1, we compare the two computations $A$ and $B$ by comparing their $\Theta$s:

$$\Theta_A = E_A t_A^2 \tag{1.12}$$

$$\Theta_A = (2E_B)\left(\frac{t_B}{2}\right)^2 \tag{1.13}$$

$$\Theta_A = \frac{\Theta_B}{2}\,. \tag{1.14}$$

Hence, we can conclude that $A$ is twice as $\Theta$-efficient as $B$. For equal delays, $E_A = \frac{E_B}{2}$ . For equal energies, $t_A = \frac{t_B}{\sqrt{2}}$ .

## 3.4 $Et^2$ Measurements

How constant is $Et^2$ in reality? There are several operating modes for the CMOS transistor, each with a very different relation between current and voltage. In particular, at high electric field, the carrier velocity saturates and becomes constant; the delay becomes independent of the voltage, and $Et^2$ becomes quadratic in the voltage. Figure 1.1 shows the measured $Et^2$ for the two-million–transistor asynchronous MIPS R3000 microprocessor designed at Caltech between 1996 and 1998. It was fabricated in 0.6-$\mu$m CMOS and was entirely functional on first silicon [4]. (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 *Vdd,* but we obviously cannot enforce this for HP's 0.6-$\mu$m 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 in the range 1.5–4.9 volts *Vdd.*

# 4. The $\Theta$-Efficiency of Designs

In this section, we use the $\Theta$ metric to determine when two standard design transformations—parallel composition and pipelining—improve the efficiency of a design compared with sequential execution.

## 4.1 The $\Theta$-Efficiency of Parallelism

Given a collection of independent tasks, when does the parallel execution of the tasks improve (i.e., reduce) the $\Theta$ of the computation? For simplicity, consider $m$ identical tasks each consuming energy $E/m$ and using delay $t/m$ to complete.

If the $m$ tasks are executed sequentially, the total energy is $E$ and the total delay (execution time) is $t$, giving a $\Theta_0$ of $Et^2$.

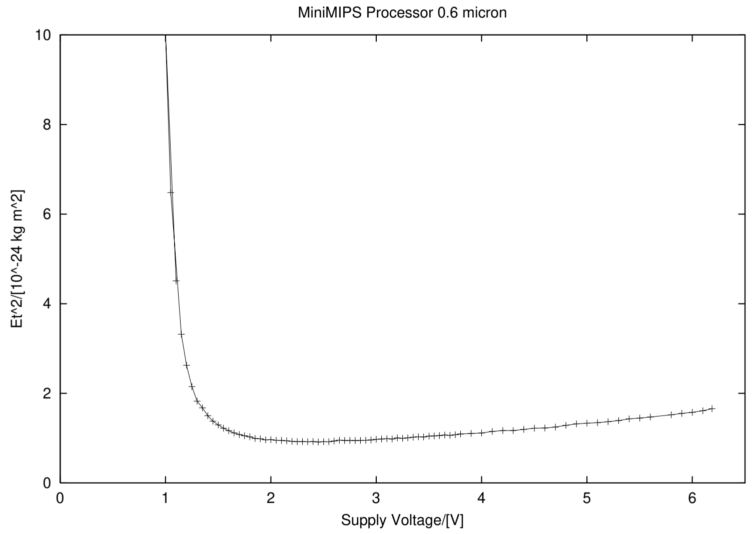


*Figure 1.1.* Measured $Et^2$ for a two-million–transistor asynchronous microprocessor.

Now, let us consider what happens when the $m$ tasks run in parallel. First, we ignore the cost of the *split* circuitry that may be needed to distribute the control (and possibly data) to all tasks, and the cost of the *merge* circuitry that may be needed to gather the completion signal (and possibly some results) from all tasks. In that simple case, the total energy is still $E$, but the total delay is now reduced to $t/m$, giving a $\Theta_{par}$ of $(Et^2)/m^2$. Hence, the improvement $\Theta_{par}/\Theta_0 = 1/m^2$: parallelism reduces the $\Theta$ of the computation by a factor $m^2$.

Assuming we can vary the voltage of the new design so as to make the delay equal to the delay of the sequential design, then under the invariance of $Et^2$, the parallel transformation decreases the energy consumption by a factor $m^2$. (For large $m$, it may in practice be impossible to scale down the voltage by a factor $m$, and therefore it may be impossible to exploit all the potential energy improvement of parallelism.)

**Theorem 2** *The parallel composition of $m$ identical tasks without overhead gives an $Et^2$ reduction of $m^2$ compared with sequential execution, and a potential energy reduction of $m^2$ if the voltage can be reduced by a factor $m$.*

The situation is more complicated if the tasks are different. Let us assume that each task $A_i$ now uses energy $E_i$ and delay $t_i$, such that we still have $\sum E_i = E$ and $\sum t_i = t$. The parallel composition still uses energy $E$, but the delay is now $\max(t_i)$; let us call the task with maximal delay $A_{max}$ and its delay $t_{max} = \max(t_i)$. Remember that we are assuming that the voltage of each task can be adjusted freely. Then it is clear that, for each task except $A_{max}$, the early termination of the task amounts to a waste of energy since every task but $A_{max}$ can be slowed down to $t_{max}$ without affecting the delay of the whole computation, but with an energy improvement corresponding to the voltage decrease.

According to the above analysis, parallelism always improves $Et^2$ if we ignore any overhead it introduces. Let us now examine the case when the cost of the *split* and *merge* circuitries cannot be ignored. We are looking at the simple case where we split the original task into two parallel tasks with the help of just one binary *split* and one binary *merge.* We assume that both the split and merge have an energy and delay that are a fraction $k$ of the energy and delay of the original task:

$$E_{split} = E_{merge} = kE \ , \tag{1.15}$$

$$t_{split} = t_{merge} = kt \ . \tag{1.16}$$

With the added overhead, the energy $E_{par}$ and delay $t_{par}$ of the parallel execution become:

$$E_{par} = E + 2kE \ , \tag{1.17}$$

$$t_{par} = t/2 + 2kt \ , \tag{1.18}$$

which means that

$$\Theta_{par} = \Theta_0(1 + 2k)(1/2 + 2k)^2 \ . \tag{1.19}$$

The ratio $\Theta_{par}/\Theta_0$ is less than 1 only for $k < 0.18$ . In other words, binary parallel composition using a split and a merge with the above characteristics decreases $\Theta$ only when the task to be parallelized is at least $5.6$ times as expensive as a split or merge.

As a concrete example, the authors have investigated the possibility of improving the performance of a 32-bit four-stage carry-lookahead adder by interleaving two identical adders. For this type of circuit, k is empirically found to be approximately $0.25$ (the split and merge networks are about as expensive as one adder stage). Therefore splitting the adder in this particular way does not help.

## 4.2 The Θ-Efficiency of Pipelining

Now let us consider a task $S$ that repeatedly evaluates the function $F$ for a sequence of parameter values: $S$ receives a parameter value $x$ from the environment, evaluates $F(x)$, sends the result to the environment, and repeats the cycle for a next parameter value.

Pipelining is the transformation that replaces the single task $S$ with a chain $P$ of $m$ tasks $P_j$ computing functions $f_j$, called the "stages" of the pipeline. Each stage $P_j$ behaves exactly like the original task, except that it computes $f_j$ instead of $F$, and its environment is different: $P_j$ receives its parameters from $P_{j-1}$ if $j > 0$, or from the environment of $S$ if $j = 0$ ; it sends its results to $P_{j+1}$ if $j < m - 1$, or to the environment of $S$ if $j = m - 1$ .

The $f$s are chosen such that $F = f_{m-1} \circ \ldots \circ f_1 \circ f_0$, with $0 < m$.

In this example, we are interested in the cycle time and the energy consumed by one cycle. We first determine the energy $E_1$ and cycle time $t_1$ for the computation of one $F(x)$ by task $S$.

There are two parts to the activity of a cycle: the computation of $F$ and the communication overhead (receiving parameters and sending results). Let $E$ be the energy and $t$ the time to compute $F$. We assume that the energy of the communication overhead is $kE$ and that the delay is $kt$. Putting the pieces together, we get:

$$
\begin{aligned}
E_1 &= E + kE, && (1.20)\\
t_1 &= t + kt, && (1.21)\\
\Theta_1 &= Et^2(1+k)^3 \,. && (1.22)
\end{aligned}
$$

Now, want to choose the length of the pipeline, $m$, and the functions $f$ so as to minimize the $\Theta$ of the pipeline in terms of the energy and cycle time for computing one $F(x)$.

The cycle time of the pipeline is the cycle time of the slowest stage. Hence, we should choose the functions $f$ such that all stages have the same cycle time. For all $j$:

$$t_j = \frac{t}{m} + kt \,. \qquad (1.23)$$

But the stages do not need to have the same energy; let us say that each stage consumes $G_j$, with $(\sum_j G_j) = E$. For simplicity, let us assume that the communication overhead for the entire non-pipelined implementation is paid by *each stage* of the pipelined implementation. Under these assumptions, the total energy for stage $j$ is

$$E_j = G_j + kE \,. \qquad (1.24)$$

We can now compute the energy $E_m$ and the cycle time $t_m$ for computing one $F(x)$:

$$
\begin{aligned}
E_m &= \sum_{j=0}^{m-1} E_j && (1.25)\\
t_m &= t_j \qquad \text{for all } j, && (1.26)
\end{aligned}
$$

which gives:

$$E_m = E + kmE \; , \tag{1.27}$$

$$t_m = \frac{t}{m} + kt \; . \tag{1.28}$$

Let $\Theta_m$ be the $\Theta$ of the pipeline. By definition, $\Theta_m = E_m t_m^2$ , i.e.,

$$\Theta_m = Et^2 \frac{(1+km)^3}{m^2} \; . \tag{1.29}$$

We can express the improvement in $\Theta$ compared with the non-pipelined case as the ratio of the two $\Theta$s:

$$\frac{\Theta_m}{\Theta_1} = \frac{1}{m^2} \frac{(1+km)^3}{(1+k)^3} \; . \tag{1.30}$$

The ideal case $k = 0$ (no overhead) gives an improvement

$$\frac{\Theta_m}{\Theta_1} = \frac{1}{m^2} \; , \tag{1.31}$$

with $E_m = E$ and $t_m = t/m$. Although it looks like we have gained nothing in energy, in fact we can save up to a factor $m^2$ in energy if we equalize the cycle time to that of the non-pipelined case by adjusting the supply voltage.

For $k > 0$, the optimal improvement is achieved for

$$\frac{d}{dm}\left(\frac{\Theta_m}{\Theta_1}\right) = 0 \; , \tag{1.32}$$

i.e., for $mk = 2$.

In the optimal case

$$E_m = 3E \; , \text{ and} \tag{1.33}$$

$$t_m = \frac{3}{2} kt \; , \tag{1.34}$$

whence we derive the following theorem on the optimal length of a pipeline.

**Theorem 3** *The $\Theta$-optimal pipeline requires an energy per computation step that is 3 times the energy required for computing $F$. It has a cycle time that is $3/2$ the overhead's cycle time.*

Let us compute the optimal pipeline improvement as a function of the overhead ratio $k$, ($m = \frac{2}{k}$). We get the following result:

$$\frac{\Theta_m}{\Theta_1} = \frac{27}{4} \frac{k^2}{(1+k)^3} \; . \tag{1.35}$$

The result shows that the pipeline is very sensitive to the communication overhead. For an overhead ratio of one (which obtains when the pipelining communication is as costly as the operation itself), the pipeline offers practically no gain in $Et^2$.

* *

In the second part of this chapter, we examine the relationship between $E, t$, and the two physical parameters that a designer (usually) can adjust: the supply voltage and the transistor widths. First, we address the issue of optimizing $Et^2$ as a function of the transistor widths. Secondly, we introduce the notion of *minimum-energy functions* $E(t)$ to express the dependence of $E$ and $t$ on each of the two physical parameters. We use those functions for deriving a number of important results about the sequential and parallel compositions of systems.

## 5. Transistor Sizing for Optimal Θ

The task of adjusting the transistor widths of a circuit is called "transistor sizing," or "circuit sizing." We are interested in sizing transistors so as to minimize $\Theta$. Both capacitance $C_z$ and current $i_{z\uparrow}$ in Equations 1.2 and 1.3 for $E$ and $t$ depend on the size of the transistors. The capacitance contributed by transistors increases linearly with the transistor widths, but the current also increases linearly. Hence, it is not immediately clear how transistors should be sized to optimize the $\Theta$ of a circuit.

We shall find that, in a $\Theta$-optimal circuit, the transistors are sized such that the total transistor capacitance is approximately twice the total parasitic capacitance. As we shall see, the result is exact only for a restricted class of circuits; nevertheless, it is a good approximation for most circuits.

Let $E$ be the total energy of the computation: it is the sum of all energy spent exercising the nodes of the computation. Assume, without loss of generality, that there are exactly two transitions corresponding to each node. (This amounts to "unrolling" into several nodes the nodes of the circuit that see more than two transitions and ignoring the nodes that never transition.) Let $t$ be the cycle time of a critical cycle. We assume that the circuit is designed so that all cycles are critical; this is true in many well designed circuits, and it is true for any optimally sized circuit in the absence of additional constraints (e.g., minimum-size or slew-rate constraints) on transistor sizes.

We distinguish between two types of capacitances attached to a node $j$: first, the "gate" (or transistor) capacitance $c_j$ contributed by the transistors of the operators to which node $j$ is connected, and secondly, the "parasitic" capacitance $p_j$ contributed by the wires connecting node $j$ to other operators. We assume that $p_j$ is fixed and that we can change $c_j$ as we please by adjusting the transistors' widths.

We first consider scaling all the transistors in the circuit by the same factor: we want to determine the global scaling factor $w$ to be applied to all transistors' widths that achieves the lowest $Et^2$. We have

$$E = \sum_{j \in N} (c_j + p_j) V^2, \tag{1.36}$$

where $N$ is the set of all nodes on the chosen critical cycle, i.e.,

$$E = (C + P) V^2 \ , \tag{1.37}$$

with $C \stackrel{\text{def}}{=} \sum_{j \in N} c_j$ and $P \stackrel{\text{def}}{=} \sum_{j \in N} p_j$ .

For the sake of simplicity, we assume that upgoing and downgoing transitions on a given node have the same delay. For the cycle time $t$ of a critical cycle, $t = 2(\sum_{j \in N} t_j)$ where $t_j$ is a transition of node $j$. We use Equation 1.6 to compute all $t_j$s

$$t = 2 \sum_{j \in N} \frac{(c_j + p_j)}{k_j V} \ . \tag{1.38}$$

We first simplify the problem and assume that the circuit is "homogeneous," i.e., that all gates are identical, and hence that $\forall j : j \in N : k_j = Kw$. We get that

$$t = 2 \frac{\sum_{j \in N} (c_j + p_j)}{KwV} \ , \tag{1.39}$$

i.e.,

$$t = 2 \frac{C + P}{KwV} \ . \tag{1.40}$$

By definition of the global sizing factor $w$, we have $C = wC_{min}$, and therefore we can eliminate $w$ from the expression of $t$:

$$t = \mathcal{K} \frac{C + P}{CV} \ , \tag{1.41}$$

$$Et^2 = \mathcal{K}^2 \frac{(C + P)^3}{C^2} \ ; \tag{1.42}$$

where $\mathcal{K} \stackrel{\text{def}}{=} 2C_{min}/K$. It is easy to check that $\frac{\partial}{\partial C}(Et^2) = 0$ for $C = 2P$. Hence the theorem:

**Theorem 4** *For a homogeneous circuit, the minimal $Et^2$ is achieved when the total gate capacitance is twice the total parasitic capacitance.*

## 5.1 Using $Et^n$ With $n \neq 2$

Sometimes, we may want to optimize $Et^n$ for $n \neq 2$ when using a $\Theta$-optimal circuit would not be possible because the required delay or energy would result in a supply voltage outside the practically possible range. Roughly speaking, when we perform $Et^n$ optimization we mean that we consider a 1% improvement in speed is worth an $n\%$ increase in energy. For example, for a circuit operating in velocity saturation, we might have expend twice the energy for a 10% speed improvement. In that case, we should optimize for $Et^n$ with $n \approx 5$ .

There is another reason to examine $Et^n$, besides extreme supply voltages (and mathematical insight). Even though a large system may be optimized for $Et^2$, components of that system may not individually be optimized for $Et^2$. For example, speeding up critical paths while lowering the $Et^2$ of these paths may make the entire design run faster and actually improve $Et^2$ for the entire design. If multiple supply voltages are allowed, then Theorem 1 applies to each component of the system, so each component is optimally characterized by $Et^2$. But multiple supplies are impractical; instead we can use $Et^n$ optimization for the different paths, with a larger $n$ for the more critical paths and a smaller $n$ for the less critical paths.

As an example of $Et^n$ optimization, we generalize Theorem 4 for all $n$:

$$Et^n = \mathcal{K}^n \frac{(C+P)^{n+1}}{C^n V^{n-2}} , \tag{1.43}$$

and we find that $\frac{\partial}{\partial C}(Et^n) = 0$ for $C = nP$. Hence the theorem:

**Theorem 5** *For a homogeneous circuit, the minimal $Et^n$ is achieved when the total gate capacitance is $n$ times the total parasitic capacitance.*

## 5.2 Optimal Energy and Cycle Time

We have seen that, for a ring of identical operators, $E$ and $t$ are of the following form:

$$E = (C+P)V^2 \tag{1.44}$$

$$t = \mathcal{K}\frac{C+P}{CV} . \tag{1.45}$$

When optimizing for $Et^n$ by transistor sizing, we have established that the minimum is achieved for $C = nP$, to which correspond an energy $E_n$ and a cycle time $t_n$, with

$$E_n = (n+1)PV^2 , \tag{1.46}$$

and

$$t_n = \mathcal{K}\frac{n+1}{n}\frac{1}{V} \ . \tag{1.47}$$

Two interesting quantities are $E_0$ and $t_\infty$: $E_0 = PV^2$ , and $t_\infty = \mathcal{K}/V$ . By definition of $n$ , $E_0$ is the theoretical minimal energy, corresponding to minimizing $E$ without regard for $t$; it corresponds to the situation when the transistors are all zero-sized and the fixed parasitic capacitances constitute the entire $E$. Conversely, $t_\infty$ is the theoretical minimal cycle time corresponding to minimizing $t$ without regard for $E$. It is obtained when $C$ goes to infinity, i.e., when only gate capacitances contribute to $E$ and $t$. We may eliminate $V$ from Equations 1.46 and 1.47 and restate our results in terms of $E_0$ and $t_\infty$; thus,

$$E_n = (n+1)E_0 \ , \tag{1.48}$$

and

$$t_n = \frac{n+1}{n}t_\infty \ . \tag{1.49}$$

In particular

$$E_2 = 3E_0 \ , \tag{1.50}$$

$$t_2 = \frac{3}{2}t_\infty \ . \tag{1.51}$$

**Theorem 6** *The cycle time for optimal $Et^2$ is 3/2 the theoretical minimal cycle time at that supply voltage. The energy for optimal $Et^2$ is three times the theoretical minimal energy at that supply voltage.*

If we eliminate $n$ from Equations 1.48 and 1.49, we arrive at the following relation between the minimum energy and the minimum delay of a single-cycle circuit at a fixed voltage:

$$E_n = \frac{E_0 t_n}{t_n - t_\infty} \ . \tag{1.52}$$

## 5.3 A Minimum-Energy Function

We can define an antimonotonic *minimum–energy-consumption function* or *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 [12].) If we rewrite Equation 1.52 with $E$ a function of $t$, we get the following function:

$$E(t) = \frac{E_0 t}{t - t_\infty} \ . \tag{1.53}$$

It is easy to prove that Equation 1.53 satisfies the above definition of the minimum-energy function.

## 5.4 Experimental Evidence

Even though Equations 1.48 and 1.49 have been derived for only a very restricted class of circuits, they are in fact good approximations for a much wider class. The authors have checked the equations against the minimal $Et^n$ obtained by applying an optimization algorithm (gradient descent) to a class of circuits. The circuits, each consisting of a ring of operators, 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. The authors used real numbers for both parameters; they optimized the expression for $Et^n$ using Equations 1.36 and 1.38 for $E$ and $t$. The range of parasitics was [1,100] in normalized units; the range of transistors in series was [1,6]. The results show that Equations 1.48 and 1.49 hold, with very good accuracy, over a wide range of parasitics, logic-gate types, and circuit sizes.

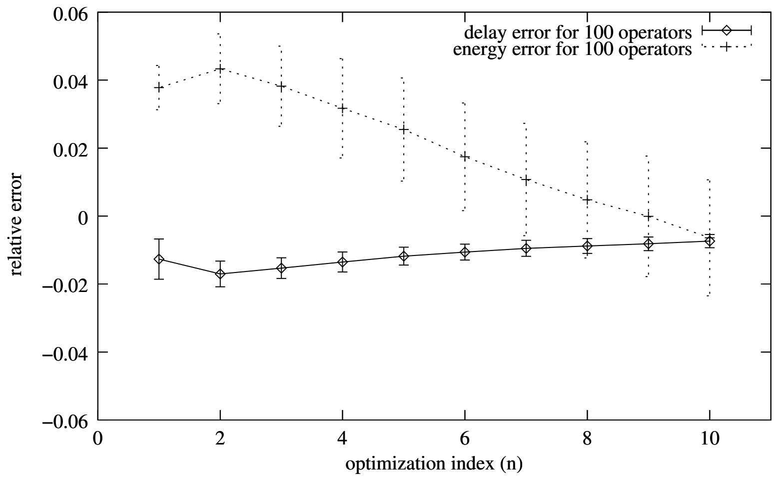


*Figure 1.2.* Results of simulating random circuits.

The results of the simulations for circuits consisting of a ring of 100 operators are summarized in Figure 1.2. (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 1.48 and 1.49 for a range of different optimization indices ($n \in 1..10$ in $Et^n$). 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 good to within five percent of the energy and delay values for the actual optimum $Et^n$.

## 5.5 Multi-cycle Systems

Let us now consider a system composed of $m$ subsystems $S_i$ ($E_{0i}$, $t_{\infty i}$) executing in parallel; each subsystem $S_i$ has minimum-energy function

$$\frac{E_{0i}t}{t - t_{\infty i}}\ .$$

These subsystems can be chains or rings of arbitrary logic gates, since our experiment shows that Equations 1.48 and 1.49 adequately describe the minimum energy and delay of a large class of circuits. Let us assume that the subsystems are synchronized so that all $t$s are equal. As a consequence, the minimum-energy function for the composed system is

$$E(t) = t\sum_{i=1}^{m}\frac{E_{0i}}{t - t_{\infty i}}\ . \tag{1.54}$$

**Theorem 7** *For a system composed of $m$ subsystems $S_i$ ($E_{0i}$, $t_{\infty i}$) as specified above, if the system is optimally sized for $Et^n$ then*

$$E_n \leq (n+1)\sum_{i=1}^{m} E_{0i} \tag{1.55}$$

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

(Note that Equation 1.48 is simply the special case of Theorem 7 that holds when all $t_{\infty i}$s are equal.)

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

$$\frac{d(E(t)t^n)}{dt} = 0\ , \tag{1.56}$$

which is achieved when

$$(n+1)\sum_{i=1}^{m}\frac{E_{0i}}{t - t_{\infty i}} = t\sum_{i=1}^{m}\frac{E_{0i}}{(t - t_{\infty i})^2}\ . \tag{1.57}$$

We may now invoke 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 $l_i \leftarrow \frac{\sqrt{E_{0i}}}{t - t_{\infty i}}$ and $r_i \leftarrow \sqrt{E_{0i}}$, we get that

$$\left(\sum_{i=1}^{m}\frac{E_{0i}}{t - t_{\infty i}}\right)^2 \leq \sum_{i=1}^{m}\frac{E_{0i}}{(t - t_{\infty i})^2}\sum_{i=1}^{m} E_{0i} \tag{1.58}$$

with equality if and only if all $t_{\infty i}$s are equal. Using Equation 1.57, 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 1.58, and we get the following result:

$$\left( \sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \right)^2 \leq \frac{(n+1)}{t} \sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \sum_{i=1}^{m} E_{0i} \ . \tag{1.59}$$

By Equation 1.54, then,

$$E(t) = t \sum_{i=1}^{m} \frac{E_{0i}}{t - t_{\infty i}} \leq (n+1) \sum_{i=1}^{m} E_{0i} \ . \tag{1.60}$$

And therefore

$$E_n \leq (n+1) \sum_{i=1}^{m} E_{0i} \ . \qquad \Box \tag{1.61}$$

In Theorem 7, equality holds if and only if all $t_{\infty i}$ s are equal; in this situation, we also have that $t = \frac{n+1}{n} t_\infty$ . This is a generalization of Equations 1.48 and 1.49 to multi-cycle systems. As we have already pointed out, the $t_{\infty i}$s are likely to be close to each other in most well designed circuits, so we should expect that usually $E(t) \approx (n+1) \sum E_{0i}$ . In other words, in a multi-cycle system optimally sized for $Et^2$ , the gate capacitance is (close to) twice the parasitic capacitance.

We have shown that the $C = nP$ result is correct for a ring of operators. We previously observed that if a dominant term exists then $C = nP$ is approximately correct for general circuits. We have experimental evidence that the relation is true for a large class of multi-cycle systems. Such evidence is also provided by SPICE simulations of an adder published by Chandrakasan and Brodersen and summarized in Figure 4.7 of their book [2]. 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.)

## 6. Power Rule for Sequential Composition

Let us now consider the sequential composition of two systems $A$ and $B$. Let us assume a sequential computation that runs $A$ to completion and then $B$ to completion; we assume the delay between the end of $A$ and the start of $B$ to be negligible. We want to know at what $t_A$, $t_B$ to run $A$ and $B$ so as to optimize the $Et^n$ of the sequential composition.

We now recall the concept of a minimum-energy function introduced by Equation 1.53. Equation 1.53 applies to the specific transformation of changing transistor sizes; i.e., it describes what the minimum energy of a circuit will be when that circuit is sized to achieve a certain performance. We are no longer limiting our discussion to the effects of transistor sizing, so we can allow other transformations to be used as a basis for this $E(t)$ .

**Theorem 8** *For the sequential composition of two systems $A$ and $B$, if the composed system is optimized for $Et^n$, then*

$$\frac{dE_A(t_A)}{dt_A} = \frac{dE_B(t_B)}{dt_B} \ . \tag{1.62}$$

**Proof.** The latency of the composed system is $t = t_A + t_B$, while its energy is $E = E_A(t_A) + E_B(t_A)$. Hence we are minimizing

$$f(t_A, t_B) \stackrel{\text{def}}{=} (E_A(t_A) + E_B(t_B))(t_A + t_B)^n. \tag{1.63}$$

If we set the partial derivatives of $f$ with respect to $t_A$ and $t_B$ equal to zero, we obtain

$$\frac{dE_A(t_A)}{dt_A} = -\frac{n(E_A(t_A) + E_B(t_B))}{t_A + t_B}$$

and

$$\frac{dE_B(t_B)}{dt_B} = -\frac{n(E_A(t_A) + E_B(t_B))}{t_A + t_B} \ ,$$

from which it is clear that Equation 1.62 holds. □

Theorem 8 holds for any minimum-energy function $E(t)$ and any value of the optimization index $n$.

If we now vary the supply voltages of the components $A$ and $B$ of the sequential composition so as to optimize $Et^2$ , we have the following theorem:

**Theorem 9** *For the sequential composition of two systems $A$ and $B$ with power consumptions $P_A$ and $P_B$, respectively, if the composed system is optimized for $Et^2$ by adjusting the supply voltages of the components, then*

$$P_A = P_B \ . \tag{1.64}$$

**Proof.** Let us define $\Theta_A = E_A t_A^2$, $\Theta_B = E_B t_B^2$; as we have established, $\Theta_A$ and $\Theta_B$ are voltage independent. Using Theorem 8 with $E_A(t_A) = \Theta_A / t_A^2$ and $E_B(t_B) = \Theta_B / t_B^2$, we get:

$$\frac{\Theta_A}{t_A^3} = \frac{\Theta_B}{t_B^3} \ . \tag{1.65}$$

Hence,

$$\frac{E_A}{t_A} = \frac{E_B}{t_B} \ . \qquad \square \tag{1.66}$$

In other words, circuits composed sequentially in a $\Theta$-optimal way should have their supply voltages adjusted so as to equalize their power use. (If the circuits are themselves $\Theta$-optimal, then equalizing their power is a necessary and sufficient condition for making the composition $\Theta$-optimal.)

## 7. $\Theta$-Rules for Parallel and Sequential Compositions

Finally, let us consider the parallel and sequential composition rules for $Et^2$ , assuming that we have the freedom of independently adjusting the supply voltages of the composed systems. Given are two systems $A$ and $B$ with latencies $t_A$ and $t_B$, energies $E_A$ and $E_B$; we have $\Theta_A = E_A t_A^2$ and $\Theta_B = E_B t_B^2$ .

First, consider the parallel composition of the two systems. We want to compute the minimum $\Theta_{A||B}$ as a function of $\Theta_A$ and $\Theta_B$ .

The minimum $\Theta_{A||B}$ is achieved when $t_A = t_B$ (see Section 4.1). With $E_{A||B} = E_A + E_B$ and $t_{A||B} = t_A = t_B$, we get $\Theta_{A||B} = E_A t_A^2 + E_B t_B^2$ . Hence the theorem:

**Theorem 10** *For two systems $A$ and $B$ composed in parallel,*

$$\Theta_{A||B} = \Theta_A + \Theta_B \ . \tag{1.67}$$

Now consider the two systems composed in sequence as in Section 6. As in the previous example, we are given $\Theta_A = E_A t_A^2$ and $\Theta_B = E_B t_B^2$, and we wish to determine the optimal $\Theta_{A;B}$ as a function of $\Theta_A$ and $\Theta_B$ . We have:

$$\Theta_{A;B} = (E_A + E_B)(t_A + t_B)^2 \ , \tag{1.68}$$

i.e.,

$$\Theta_{A;B} = \left( \frac{\Theta_A}{t_A^2} + \frac{\Theta_B}{t_B^2} \right) (t_A + t_B)^2 \ . \tag{1.69}$$

Since the optimal total $Et^2$ is independent of global scaling of $t$, it is sufficient to determine a single parameter defined as follows: $\alpha = t_A/t_B$. From Equation 1.65, we have $\alpha = \sqrt[3]{\frac{\Theta_A}{\Theta_B}}$. For this $\alpha$ , we compute the optimal $\Theta = (\sqrt[3]{\Theta_B} + \sqrt[3]{\Theta_A})^3$ :

**Theorem 11** *For two systems $A$ and $B$ composed in sequence,*

$$\sqrt[3]{\Theta_{A;B}} = \sqrt[3]{\Theta_A} + \sqrt[3]{\Theta_B} \ . \tag{1.70}$$

The $\Theta$-rules are very useful for computing the $\Theta$ of a computation as we did for parallelism and pipelining in Section 4. The industrious reader that goes through the exercise of using the $\Theta$-rules to compute the same $\Theta_{par}$ that was computed in Section 4 is in for a surprise: the result is different!

Using the $\Theta$-rules, we get $\Theta_{par} = \Theta_0(2k + \frac{1}{2^{2/3}})^3$ , which is smaller than the result of Equation 1.19. The reason is subtle but important. In the first computation, we postulate that $E_{split} = E_{merge} = kE$ and $t_{split} = t_{merge} = kt$ . By fixing the $E$ and $t$ ratios, we implicitly assume that the voltages of the different components are identical. The computation using the $\Theta$-rules just assumes that $\Theta_{split} = \Theta_{merge} = k^3\Theta_0$ . Hence the voltages of the split and merge can be adjusted independently, which leads to a lower $\Theta_{par}$ .

(An analysis of the $\Theta$-efficiency of parallelism can be found in a companion paper [6].)

## 8. Summary and Conclusion

In this chapter, we have seen that $Et^2$ constitutes an excellent metric for comparing computations for energy and delay efficiency when the physical behavior is that of CMOS VLSI circuits. We started by observing that the $Et^2$ metric for a CMOS circuit is independent of the supply voltage, as long as we can scale the threshold voltage linearly with the supply voltage and as long as we stay away from velocity saturation.

We showed that when supply-voltage adjustment is allowed, the popular $Et$ metric is inferior. Following along these lines, we established that any metric with certain properties (we called such a metric $\Theta$) could be used to compare designs independently of the voltage. As long as the required speed or energy lies within the threshold-voltage to velocity-saturation range of the implementations, we saw that the implementation with the better $\Theta$ is better for *any* desired speed or energy consumption. We showed that $Et^2$ is to first order a metric with the required properties. We then applied the metric to various circuit transformations, namely pipelining and parallelism. We also applied the metric to transistor sizing and were able to show that the optimal sizing for energy efficiency is not what is commonly used (minimal sizes). Finally, we established rules for computing the $\Theta$ of the sequential and parallel compositions of systems.

Overall, $Et^2$ is a very useful efficiency metric for designing CMOS VLSI circuits. Time and experience will show how applicable it is to other computations.

## Acknowledgments

Acknowledgment is due to Karl Papadantonakis, Martin Rem, and Catherine Wong for their comments and criticisms. The research described in this paper was sponsored by the Defense Advanced Research Projects Agency and monitored by the Air Force.

## References

[1] Carver A. Mead and Lynn Conway. *Introduction to VLSI Systems*, Addison-Wesley, Reading MA, 1980.

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

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

[4] Alain J. Martin, Andrew Lines, Rajit Manohar, Mika Nyström, Paul Penzes, Robert Southworth, Uri Cummings, and Tak Kwan Lee. The Design of an Asynchronous MIPS R3000 Microprocessor. *Proceedings of the 17th Conference on Advanced Research in VLSI*, IEEE Computer Society Press, 164-181, 1997.

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

[6] Alain J. Martin and Mika Nyström. The $Et^2$-Efficiency of Parallelism, Caltech Technical Report, October 2001.

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

[8] Karl Papadantonakis. A Theory of Constant $Et^2$ CMOS Circuits, Caltech Computer Science Technical Report 2001.004, July 2001.

[9] Karl Papadantonakis. Hierarchical Voltage Scaling for $Et^2$ Optimization of CMOS Circuits, Caltech Computer Science Technical Report 2001.005, July 2001.

[10] Paul I. Pénzes. *Energy-delay Efficiency of Asynchronous Circuits*, Ph.D. Thesis (in preparation), California Institute of Technology, 2002.

[11] Paul I. Pénzes and Alain Martin. Global and Local Properties of Asynchronous Circuits Optimized for Energy Efficiency. *IEEE Workshop on Power Management for Real-time and Embedded Systems*, May 2001.

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

[13] José A. Tierno and Alain J. Martin. Low-Energy Asynchronous Memory Design. *Proceedings of International Symposium on Advanced Research in Asynchronous Circuits and Systems*, IEEE Computer Society Press, pp. 176–185, 1994.