Proceedings of the WG '83

International Workshop on

Graphtheoretic Concepts in Computer Science

June 16–18, 1983,
Osnabrück, Fed. Rep. Germany

Edited by Manfred Nagl
and Jürgen Perl

Universitätsverlag Rudolf Trauner, Linz

On the Energy-Time-Area Cost of a Memory Access

Clark D. Thompson
Computer Science Division
573 Evans Hall
University of California
Berkeley, CA 94720

ABSTRACT

Metrics of energy, area, and time are defined for a graph-theoretic model of VLSI computation. Different "constant factors" are seen to be appropriate for different logic families. We examine seven such families: NMOS, CMOS, CMOS-SOS, I^2L, GaAs HEMT, JJ-CIL, and JJ-CS. For each family, we sketch a construction for an energy-efficient, read/write, random-access memory circuit.

1. Introduction

Complexity theorists have traditionally focussed on two metrics of the quality of a circuit design: size (the number of gates involved) and speed (the number of gates on the longest path from any input to any output). More recently, notions of circuit area and wire delay have come into play [13], due to their growing importance in VLSI design.

Very little effort, by contrast, has been put into the study of the power and energy requirements of computation. This theoretical oversight might seem strange, in view of the practical importance of the subject. Energy is a critical resource in battery-powered systems, and heat (*i.e.*, power) dissipation is an important constraint on the physical design of almost every computational system.

The theoretical difficulties of the area are easily described: there is no simple, accepted, widely-applicable model of the energetics of computation. A literature search yields scanty and contradictory results. It is hard to reconcile the assumptions in Chapter 9 of [11] (seemingly written by Chuck Seitz), with Bennett's monograph [3], with Kissin's recent paper [7], or with Lengauer and Mehlhorn's

This work was supported by the National Science Foundation Grant ECS-8110684.

article [10]. Bennett argues that, in principle, one can do any amount of computation with a small amount of energy. Seitz takes the opposite position to Bennett, maintaining that "gate switching energy" is a fundamental property of any logic technology. Seitz' assumptions are, in turn, too weak for the models of Kissin, Lengauer and Mehlhorn. Their results are motivated by the dependence of switching energy on wirelength in the CMOS technology.

This paper attempts to inform the controversy surrounding models of energy consumption. A unified model is proposed for all present-day (and a couple of futuristic) integrated-circuit technologies. When placed on this common basis, it is easy to see that there are almost as many "fundamental modes" of energy consumption as there are technologies.

Some technologies consume power at a nearly constant rate per logic gate, while others exhibit a state-dependent rate of power consumption. In either case, total energy consumption might be called "static" since very little additional "switching energy" is required.

A second major grouping of contemporary technologies has little static energy dissipation, but consume significant switching energy. As suggested by [7] and [10], switching energy might be proportional to wire length. Alternatively, it might be essentially independent of wire length [11]. (A final possibility, a Bennett-style [3] zero-energy technology, is unlikely to be seen in this century.)

The technological diversity outlined above is captured in this paper's unified model by a number of technological "constant factors." As a check on the model's completeness, constructions have been attempted for energy-efficient memory access in each technology. These are outlined in Section 4; Section 2 contains the unified model assumptions, and Section 3 lists the technological constant factors.

2. Model of Computation

In the following assumptions, greek letters are used for the technological constant factors. (There are two exceptions: δ and Δ bear their standard meaning of vertex in- and out-degree.) Sets and their elements are defined by capitalized and lower case roman letters, respectively.

1. *Sources* O_h, *sinks* I_h. A computation graph is a directed hypergraph $G = (V, H)$. A hyperedge h is denoted by an ordered pair (O_h, I_h) of vertex sets $O_h \subseteq V$, $I_h \subseteq V$. The vertices in O_h are the s*O*urces of h; the vertices in I_h are its s*I*nks.

2. *Edge fanout restrictions o_{max}, ι_{max}.* Each edge h has at least one, and at most $O(1)$, sources and sinks:

$$1 \leq |O_h| \leq o_{max}, \qquad 1 \leq |I_h| \leq \iota_{max}$$

Limits on vertex indegree δ and outdegree Δ are discussed in Assumption 10.

3. *Vertex widths λ_{gate}, $\lambda_{I/O}$.* Each vertex in a computation graph is embedded as a square region in the Euclidean plane. No two vertices overlap. The size of an embedded vertex depends upon its functionality: gates $v \in V_{gate}$ occupy λ^2_{gate} area, while I/O ports $v \in V_{I/O}$ occupy $\lambda^2_{I/O}$ area.

4. *Edge width λ_{wire}, number of wiring layers μ.* An edge is embedded as a connected set of wire segments. Each wire segment is a rectangle of width λ_{wire} and arbitrary length, placed on one of μ planar wiring layers stacked above the plane of the vertices. A wire segment on the bottom wiring layer connects to the vertices it passes over. Two wire segments are connected to each other if they pass over the same point and if they are on either the same layer or an adjacent layer. (Note that $\lceil \mu/2 \rceil$ disconnected wire segments may pass over the same point in the vertex plane. Also note that any hyperedge h can be embedded as a tree of wire segments passing over the vertices in $O_h \cup I_h$.)

5. *Total area A, maximum total area α_{max}.* The total area A of an embedded computation graph is the area of the smallest square that encloses all its vertices and wire segments. The area of this square is bounded by a technological constant: $A \leq \alpha_{max}$.

6. *Maximum edge length λ_{max}.* The total length $||h||$ of an (embedded) edge h is the sum of the lengths of its wire segments. Edge lengths are bounded by a constant: $\forall h \; ||h|| \leq \lambda_{max}$.

7. *Votes $v(t)$, signals $h(t)$.* The state of the computation graph at any time t is defined by a vector $(V(t), H(t))$ of votes $v(t)$ and signals $h(t)$ associated with each vertex v and hyperedge h. The value of a vote or signal is taken from the ternary set $\{0, 1, u\}$: logic-0, logic-1, and undetermined. (An alternative formulation, found in [10] and in state-of-the-art circuit simulators, takes signal values from a two-dimensional set of voltages V and impedances R.)

8. *Maximum size of voting equivalence class ξ_{max}, edge delay d_h, time constants τ_{gate}, τ_{wire} and τ_{fanout}, transmission line indicator ς, signal rise time r_h.*

   a. In many technologies, the delay associated with a wire can be decreased by driving that wire with a larger transistor. Such high-power drivers can be represented by several (unit-power) sources with identical voting behavior. We are thus led to the following definition of equivalence classes $C_{h,i}$ on the voting behavior of the sources for each edge h:

$$v_1, v_2 \in C_{h,i} \iff (v_1, v_2 \in O_h) \wedge (\forall t \;\; v_1(t) = v_2(t))$$

A technological limit on driving power translates into a restriction on the size of (*i.e.*, number of vertices in) any voting equivalence class:

$$\forall h, i \;\; |C_{h,i}| \le \xi_{max}$$

b. At the time of circuit construction ($t = 0$), a fixed but indeterminate delay d_h is assigned to each edge h. An edge's delay (in a worst-case analysis) is proportional to its length $||h||$ and number of sinks $|I_h|$, and inversely proportional to the size of its smallest equivalence class $C_h = \min_i |C_{h,i}| \le \xi_{max}$:

$$d_h = \tau_{gate} + \frac{||h||\tau_{wire} + |I_h|\tau_{fanout}}{C_h} (\pm 50\%)$$

(Indeterminacy is introduced into the definition of edge delays to force "realistic" design practices, *e.g.*, self-timed or clocked logic.)

c. We define r_h to be the rise time of a signal on edge h. For technologies in which wires are transmission lines, r_h is approximately equal to the gate delay τ_{gate}. We indicate this by assigning the value 1 to the 0-1 variable ς (a mnemonic is the common symbol Z for the impedance of a line). The other technological possibility is that the wires are essentially capacitive in nature (as long as their length does not exceed λ_{max}, as defined in Assumption 6). Thus

$$r_h = \begin{cases} d_h, & \text{if } \varsigma = 0 \\ \tau_{gate}, & \text{if } \varsigma = 1 \end{cases}$$

d. The value of a signal $h(t)$ is determined by the votes of its sources O_h, with delay d_h. We prevent the propagation of unreasonably-short signal pulsewidths by requiring the "election results" to be stable for at least r_h time units.

$$h(t) = \begin{cases} 1, \text{ if } \exists v \in O_h \; \forall s \in [t-d_h, t-d_h+r_h] \;\; v(s) = 1, \text{ } else \\ 0, \text{ if } \exists v \in O_h \; \forall s \in [t-d_h, t-d_h+r_h] \;\; v(s) = 0, \text{ } else \\ u \end{cases}$$

Note that this formulation allows "wire-oring": the signal on an edge becomes 1 if any of its source votes is 1 for at least r_h time.

9. *Symmetry indicator* σ.

a. Not all patterns of voting behavior are allowed in all technologies. One restriction is observed in the so-called "symmetric" technologies ($\sigma = 1$). In these technologies, the effects of logic-1 votes and logic-0 vote are symmetric, making "wire-oring" infeasible. (A system of "majority-rule" is conceivable but not observed in any present-day logic family, possibly because it would reduce noise margins.) To outlaw wire-oring, we permit just one equivalence class per edge:

$$(\sigma = 1) \Rightarrow (\forall t\ \forall h\ \forall v_1, v_2 \in O_h\quad v_1(t) = v_2(t))$$

b. A second type of restriction on allowable voting behavior arises in the asymmetric ($\sigma = 0$) technologies. We must restrict the number of high-power logic-1 votes that appear at one time on an edge, to avoid exceeding the current density limit mentioned in Assumption 8a:

$$(\sigma = 0) \Rightarrow (\forall t \forall h\ \ |\{v \in O_h\ :\ v(t) = 1\}| \leq \xi_{max})$$

10. *Logic family ϕ, power supply period τ_{supply}, I/O schedule S, external clock period $\tau_{I/O}$.*

a. A logic family ϕ is a technologically-constrained set of triples (δ, f, Δ). The first and third parts of a triple denote the indegree and outdegree of one type of gate. The second part of a triple defines a functionality, or voting behavior. A gate with the 'and' functionality, for example, is modelled by a vertex whose vote is the logical 'and' of the signals on its in-edges. As another example, the 'latch' function depends on a delayed feedback signal. Finally, the voting behavior of gates in the JJ-CIL technology depends upon the phase of their AC power supply. Thus, in the general case, the functionality f_v of a gate v has $2 + \delta_v$ parameters, and defines the gate's vote as follows:

$$v(t) = f_v(v(t - \tau_{wire}), c(t), h_1(t), h_2(t), \ldots h_{\delta_v}(t))$$

where the phase of the power supply (assumed to have a 90% duty cycle) is

$$c(t) = \begin{cases} 1, & \text{if } t < (.1 + \lfloor t/\tau_{supply} \rfloor)\ \tau_{supply} \\ 0, & \text{otherwise} \end{cases}$$

Note that voting is a zero-delay process, since gate delays were included in the definition of edge delay d_h.

b. An I/O port $v_i \in V_{I/O}$ has $\delta_v = 1$, $\Delta_v = 1$. Its voting is determined by an externally-imposed I/O schedule $S_i \in \{r_0, r_1, r_u, w_0, w_1, w_u\}^*$. Each literal in S_i indicates whether the I/O port is to read (r_0, r_1, r_u) or write (w_0, w_1, w_u) a '0', a '1', or a 'u'. The k-th literal in S_i refers to the k-th external clock period defined by $t \in ((k-1)\tau_{I/O}, k\tau_{I/O}]$, where $\tau_{I/O}$ is a technological constant. Thus, if the k-th literal is r_x, the port votes $v_i(t) = x$ during the k-th clock period. Alternatively, if the k-th literal is w_y, we say the schedule S_i is "satisfied" only if the port's in-edge h has signal $h(t) = y$, for all times t in the k-th clock period. (If the output bit for some time period is u, i.e. undetermined, we allow $h(t)$ to be any value.)

11. *Energy consumption* $E_{standby}$, $E_{1\text{-}0}$, E_{wire}, E_{sink}, E. Four modes of energy consumption are observed in physical realizations of computation graphs.

   a. A constant power dissipation of $\epsilon_{standby}/\tau_{gate}$ is associated with every gate. Total "standby" energy dissipation over the period $[t_1, t_2]$ is thus defined as

$$E_{standby} = \sum_{v \in V_{gate}} \frac{(t_2 - t_1)}{\tau_{gate}} \epsilon_{standby}$$

   b. In asymmetric (wire-or) technologies, a gate voting 1 consumes more power than a gate voting 0. We define energy $\epsilon_{1\text{-}0}$ so that the difference between these two levels of power consumption is $\epsilon_{1\text{-}0}/\tau_{gate}$. Total energy consumption in this mode is thus

$$E_{1\text{-}0} = \sum_{v \in V_{gate}} \int_{\substack{t_1 \le t \le t_2 \\ v(t) = 1}} \frac{\epsilon_{1\text{-}0}}{\tau_{gate}} dt$$

   By Assumption 8d, a gate's vote can change a signal only if it persists for at least $r_k \geq \tau_{gate}$ time. We thus employ the following (approximate) expression for $E_{1\text{-}0}$:

$$E_{1\text{-}0} = \sum_{v \in V_{gate}} \sum_{\substack{t_1/\tau_{gate} \le k \le t_2/\tau_{gate} \\ v(k\tau_{gate}) = 1}} \epsilon_{1\text{-}0}$$

   c. Each change in an edge's signal consumes energy proportional to the length of that edge. Assuming such signal changes occur at a frequency less than $1/\tau_{gate}$, we write

$$E_{wire} = \sum_{h} \sum_{\substack{t_1/\tau_{gate} \le k \le t_2/\tau_{gate} \\ h(k\tau_{gate}) \neq h((k+1)\tau_{gate})}} \|h\| \epsilon_{wire}$$

   d. Energy E_{sink}, like E_{wire}, is a form of "switching energy." In this case, the energy consumption is proportional to the number of sinks:

$$E_{sink} = \sum_{h} \sum_{\substack{t_1/\tau_{gate} \le k \le t_2/\tau_{gate} \\ h(k\tau_{gate}) \neq h((k+1)\tau_{gate})}} |I_h| \epsilon_{sink}$$

   e. The total energy consumed by a computation is $E = E_{standby} + E_{1\text{-}0} + E_{wire} + E_{sink}$.

3. Technological Parameters

	I^{2}L	JJ-CIL	NMOS	HEMT	JJ-CS	CMOS	CMOS-SOS	units
λ_{gate}	4	10	7	7	100	10	10	*um*
λ_{wire}	1	2	1	1	4	1	1	*um*
λ_{max}	10^4	10^5	10^4	10^4	10^5	10^4	10^4	*um*
$\tau_{gate}, \tau_{fanout}$	100	10	100	50	1000	100	100	*pS*
τ_{wire}	0.1	0.01	1	1	1	1	0.1	pS/λ_{wire}
τ_{supply}		1000						*ps*
$\epsilon_{standby}$	0.1	0.01	~ 0	~ 0	~ 0	~ 0	~ 0	*fJ*
$\epsilon_{1\text{-}0}$			0.1	0.1	~ 0	~ 0	~ 0	*fJ*
ϵ_{wire}					0.1	0.1	0.001	fJ/λ_{wire}
ϵ_{sink}							0.1	*fJ*
ξ_{max}	1	1	10	10	1	10	10	--
σ	0	1	0	0	1	1	1	--
ς	0	1	0	0	0	0	0	--
o_{max}	10^2	1	10^2	10^2	1	$(=\xi_{max})$	$(=\xi_{max})$	--
ι_{max}	1	1	10^4	10^4	10^4	10^4	10^4	--

Table 1. Constant factors for seven VLSI technologies (multiplicative factors for units are $f = 10^{-15}$, $p = 10^{-12}$, $n = 10^{-9}$, and $u = 10^{-6}$).

Table 1 gives approximate values for the "constant factors" of seven VLSI technologies. The values are (hopefully) correct to within a factor of 10, for circuits built with about .5 *um* line widths during the late 1980's [1, 2, 4, 5, 6, 8, 11]. Thus λ_{wire}, the minimum wire-wire spacing, is equal to 1 *um* for most of the technologies in the table. The only technologies with $\lambda_{wire} > 1$ *um* are JJ-CIL and JJ-CS, in which tree-shaped model edges must be traversed with a non-branching wire. At one or two wires per edge, this implies that the effective edge widths for JJ-CIL is between 1 and 2 *um*. And, since double-rail logic is used in JJ-CS, its λ_{wire} increases by another factor of two, to approximately 4 *um*.

An important feature of Table 1 is the diagonal structure of the entries for circuit energies $\epsilon_{standby}$, $\epsilon_{1\text{-}0}$, ϵ_{wire}, and ϵ_{sink}. When calculating total energy, contributions from entries below the diagonal can be ignored. For example, the technologies with $\epsilon_{standby} > 0$ have a nearly constant power dissipation per gate which does not increase by more than 10% when the gates change their state at maximum frequency.

Table 1 is not quite a complete list of the parameters in the model. The following are nearly constant over all technologies:

$$\alpha_{mes} = 10^9\ um^2, \quad \tau_{I/O} = 10\ nS,$$

$$\mu = 4 \text{ to } 6 \text{ layers}, \quad \lambda_{I/O} = 10^2\ um.$$

Note that, by Assumption 4, $\mu = 6$ corresponds to a three-level metal process. The other $\mu - 3$ layers are made of an insulating material, through which small square holes or "vias" are cut.

Finally, we specify the logic family ϕ of gates and latches available in each technology. For I^2L, ϕ consists of just a 1-input, 4-output inverter (all logic must be done by "wire-oring"). For NMOS and GaAs HEMT, we assume that ϕ contains a 2-input, 1-output 'nand'; a 4-input, 1-output 'nor'; and a 3-input, 1-output inverting D-latch. For all other technologies, we augment the NMOS ϕ with gates for all 2-input, 1-output boolean functions.

4. Constructions

Using the model of Section 2 and the technological parameters of Section 3, it is possible to estimate the area, time and energy performance of various circuit constructions.

In this section, we report the performance of our best constructions for large ($\sim 10^6$ bit), random-access, read-write memories in each of the seven technologies. Due to space limitations, only one of the constructions will be described in any detail.

First of all, we need a formal definition of the "N-bit memory access problem." For concreteness and simplicity, we assume that N is a power of 2, writing $n = \log_2 N = lg\ N$. We also assume the existence of a fully parallel interface that runs memory cycles as often as possible. Thus we provide n separate address lines into the memory. A new address is available on these lines every T time units, where the length T of a memory cycle is assumed to be an exact multiple of the external clock period $\tau_{I/O}$. These considerations lead to the following definition:

Memory access with cycle time T. A computation graph implements an N-bit memory access in time T if it has $n + 3$ I/O ports whose schedules obey the following restrictions. Every $T/\tau_{I/O}$ external clock periods, n address bits are read in through ports labeled a_n, a_{n-1}, ... a_1. The most significant address bit is read by a_n. Every time these address bits are read, port *we* (mnemonic: "write enable") reads a bit. If it reads a '1', the bit read by *din* ("data input") is to be written into the addressed memory location. Alternatively, if the port *we* reads a '0', a read from the addressed bit is enabled. In this case, the port *dout* ("data output") should write the correct value for the bit being accessed. (If the accessed bit has not been written into previously, *dout* may be any value.) The value written by the port *dout* must be available during the last external clock cycle of the memory cycle.

More formally, the computation graph must satisfy all I/O schedules S, where

$$S_i = \left((r_0, r_1)(r_u)^{k-1}\right)^*,$$

for all ports other than v_{dout} and where the integer k equals $T/\tau_{I/O}$. The schedule for v_{dout} is of the form

$$S_{dout} = \left((w_u)^{k-1} w_{m(S,t)}\right)^*,$$

where the values $m(S,t)$ to be written by the port are, of course, dependent upon the values written into the memory by S up to that time t.

4.1. I^{2}L

As described in Section 3, the I^{2}L logic family has just one type of gate and no latches. Lacking a latch, it seems that about 10 gates are required for each bit of memory: see Figure 1. The state of this memory cell is available on the *dout* line, whenever both x and y are at logic-1. Otherwise, $dout = 0$. The state of the memory cell is updated, to the value of *din*, whenever $x = y = we = 1$.

It is somewhat unrealistic to use a 10-gate memory cell, since an analog circuit designer could undoubtedly design a functionally-equivalent cell occupying less than one quarter of the area. We shall see, however, that the use of a smaller memory cell would not markedly decrease the area of the complete memory circuit.

Figure 2 shows a floor plan of our I^{2}L memory design. The upper rectangular box decodes the top $1/2 \lg N = n/2$ address bits into $\sqrt{N}$ *column select* lines. It also fans out the *din* bit into $\sqrt{N}$ lines, one per column in the $(\sqrt{N} \times \sqrt{N})$-cell memory array.

The left-hand rectangular box performs a similar function, decoding the least significant address bits onto $\sqrt{N}$ *row select* lines, and fanning out the *we* signal.

The data output bit, *dout*, is obtained by or-ing together the outputs of all

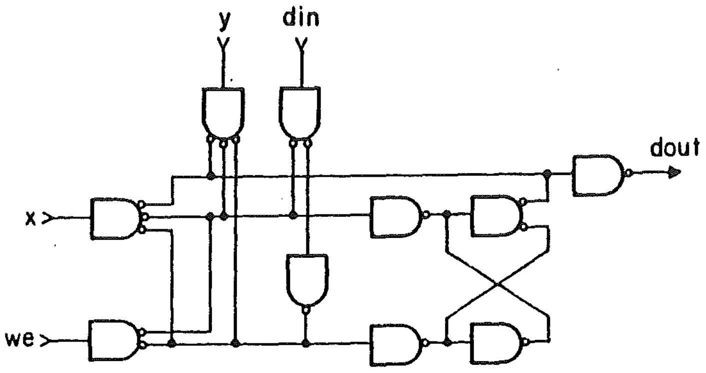

Figure 1. An I^2L memory cell.

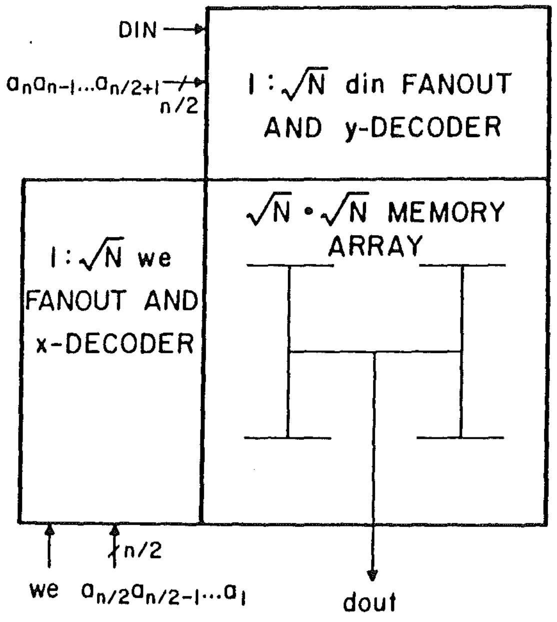
DIN
$a_n a_{n-1} \ldots a_{n/2+1}$
n/2
$1 : \sqrt{N}$ din FANOUT AND y-DECODER
$\sqrt{N} \cdot \sqrt{N}$ MEMORY ARRAY
$1 : \sqrt{N}$ we FANOUT AND x-DECODER
n/2
we
$a_{n/2} a_{n/2-1} \ldots a_1$
dout

Figure 2. Floor plan for I^2L memory.

the memory cells. This is done in an "H-tree" pattern: the data outputs of a square of $o_{max} = 100$ adjacent memory cells are wire-ored to form the input signal for an inverter. A second inverter produces a positive-true signal, 100 of which can be wire-ored in the next level of the tree. The fanin tree is thus $2 \lceil \log_{100} N \rceil$ gates deep.

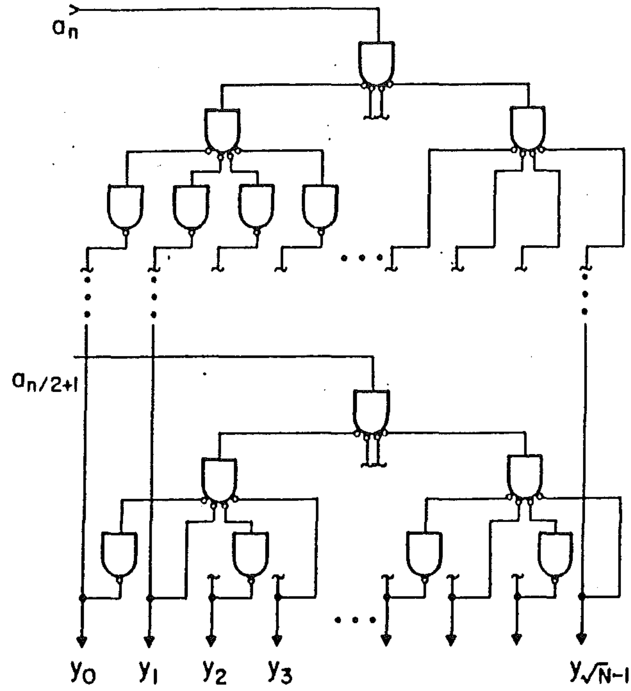

Figure 3. An I^2L *y*-decoder.

Figure 3 shows a construction for a $n/2$-bit to $\sqrt{N}$-line decoder. Note that multiple sinks are not allowed in I^2L (since $\iota_{max} = 1$). The only way to fan out a signal is to use a 4-output inverter, so $\log_4 \sqrt{N} = n/4$ levels of gating are required. Decoding is trivial, using $n/2$-way wire-ors. All in all, we need slightly fewer than $n/2\ \sqrt{N} = 1/2\ \sqrt{N}\ lg\ N$ gates to construct the circuit of Figure 3. One additional fanout tree (containing about $1/3\ \sqrt{N}$ gates) is needed for *din* or *we* fanout, completing the construction of the rectangular boxes in Figure 2.

Figure 4 indicates why I^2L is not particularly suited for memory circuits. Since wire fanout is not allowed, $1/3\ \sqrt{N}$ gates are required for each row of the memory array, to fanout the *row select* signal. Another $1/3\ \sqrt{N}$ gates per row

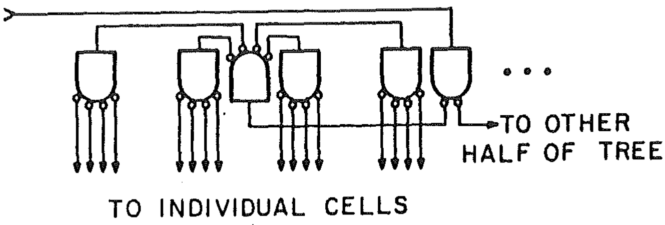

Figure 4. Row fanout tree for I^2L.

are needed for the *we* lines.

The tree fanout construction of Figure 4 was chosen to minimize total gate delay, at some expense in additional wire area. It requires one gate position for every two memory cells, so we need only allocate one gate in each memory cell for the two row fanout trees. (Alternatively, we could have fanned out the signals in a linear, left-to-right fashion, allocating two gate positions every three cells. This would result in a worst-case access time exceeding $1/3 \sqrt{N}\ \tau_{gate}$, significantly worse than what we obtain with a fanout tree.)

The row-row spacing of the memory array can now be estimated. The total number of gates per cell is 10 (for the D-latch of Figure 1) plus 1 (for the two row fanout trees) plus 1 (for the fanout of *column select* and *din* lines). In addition, a gap of λ_{gate} must be left between every 10 rows, to make room for the "H-tree" fanin of *dout*. Thus the total height or width of the $\sqrt{N} \times \sqrt{N}$ array is about $4\sqrt{N}\ \lambda_{gate}$, if we assume that we have enough room to fit the wiring.

Turning to the wiring area, it is apparent that the row fanout trees (Figure 4) could easily be a critical constraint. In fact, the row and column select lines of our memory circuit form a "mesh-of-trees" graph, requiring $\Omega(N\ lg^2 N\ \lambda_{wire}^2)$ area [9].

Considering Figure 4 more carefully, we find that we must allocate $(n/2)$ wiring tracks per row for each of the two row fanout trees. In addition, we should allocate about 5 horizontal tracks for wiring internal to each memory cell, and 1 track for the *dout* fanin. We thus need $n+6$ horizontal tracks per cell. In the vertical direction, the count is the same, so our row-row spacing must be at least $(2n+12)/(\lfloor\mu/2\rfloor)\ \lambda_{wire}$. For the case that $N = 10^6$ and $\mu = 4$, this evaluates to a cell "pitch" of 16 *um*.

According to the parameters given in Section 3 for I^2L, $\lambda_{gate} = 4$ *um*. Thus our (4 gate by 4 gate) $= (16\ um)^2$ memory cells are just large enough to accomodate the fanout wiring for $N \leq 10^6$ and $\mu = 4$. If we used smaller latches, or chose a larger N, we would have to increase μ to avoid "wasting" area on wiring.

To calculate the total area A of the memory, we must consider the decoders as well as the memory array. For $N \sim 10^6$, however, the area required by the decoders is negligible. The layout suggested by Figure 3 would be only $(lg^2 \sqrt{N})\,\lambda_{gate}$ high, and this height could be further reduced to $(lg \sqrt{N})\,\lambda_{gate} + (lg^2 \sqrt{N})\,\lambda_{wire}$ by using the tree-on-a-line idea of Figure 4. Thus we can report a total area of about $16N\,\lambda^2_{gate} \sim 2.6\ cm^2$ for $N \sim 10^6$ and $\mu \geq 4$.

A time analysis of this construction is fairly straightforward. We split the total circuit delay into the three components appearing in the Assumption 8b's definition of d_h: $\sum d_h = T_{gate} + T_{wire} + T_{fanout}$. The longest delay path through the memory is from any address bit, through the decoder, through a row (or column) fanout tree, through a memory cell, and then out the *dout* fanin tree. Total number of gates encountered on this path is about $8 + 1/2\ lg\ N$. If $N \sim 10^6$, $T_{gate} \sim 18\ \tau_{gate} \sim 1.8\ ns$.

Since C_h is identically 1 in I^2L, T_{wire} is just the length of the longest path through the circuit times τ_{wire}. (C_h is upper-bounded by ξ_h and lower-bounded by 1.) This length is about 3/4 of the perimeter of the memory: the decoder contributes one side-length, as do the row fanout tree and the *dout* fanin tree. Total wirelength is thus about $3\sqrt{A}\ \lambda_{gate} = 48\sqrt{N}\ \lambda_{wire}$, yielding a delay term of $T_{wire} \sim 48\sqrt{N}\ \tau_{wire} \sim 4.8\ ns$ for $N \sim 10^6$.

In any I^2L circuit T_{fanout} is always equal to T_{gate}, since $\iota_{max} = 1$. The total delay for our memory is thus about $8.4\ ns$ for $N \sim 10^6$. Since $\tau_{I/O} = 10\ ns$, our memory should be able to satisfy an I/O schedule with $T = 2\ \tau_{I/O}$. Address and *din* lines would be valid for the first clock period of a memory cycle; *dout* would be valid on the second. (Since the d_h values in our model are indeterminate, not all memory circuits built according to our design would satisfy this schedule. It would be interesting to build a probabilistic model that captured the effects of process fluctuations and predicted the observed spread in access times among "identical" memory circuits.)

Total energy E per access is just the number of gates ($\sim 12N$) times (T/τ_{gate}) times $\epsilon_{standby}$, or about $240\ nJ$ when $N \sim 10^6$. Power consumption is thus $(240\ nJ)/(20\ ns) = 12\ Watt$, a very high figure for a $2.6\ cm^2$ chip. If our value for $\epsilon_{standby}$ is correct, I^2L gates will have to be more widely spaced than $\lambda_{gate} = 4\ um$. Note that decreasing the number of gates per I^2L latch from 10 to, say, 4 would result in a memory circuit of similar size and speed, but with half the power consumption per unit area.

In summary, we have proved the following.

THEOREM 4-1. An I^2L memory can be constructed with

$$N \sim 10^6, \qquad A = max\left\{16N\,\lambda_{gate}^2,\; N\left((5 + lg\sqrt{N})/\lfloor \mu/2 \rfloor\,\lambda_{wire}\right)^2\right\},$$

$$E = 12NT\,\epsilon_{standby}, \qquad T = \left(\left\lceil \frac{(2\;lg\;N + 5)\;\tau_{gate} + 3\sqrt{A}\;\tau_{wire}}{\tau_{I/O}} \right\rceil + 1\right)\tau_{I/O}.$$

4.2. Other technologies

Due to space limitations, we merely quote the area-time-energy performance of our best constructions to date for the other six technologies listed in Table 1.

THEOREM 4-2. A JJ-CIL memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 6N\,\lambda_{gate}^2,$$

$$E = 6NT\,\epsilon_{standby}, \qquad T = \left(\left\lceil \frac{9\;lg\;N\;\tau_{gate} + 6\sqrt{A}\;\tau_{wire}}{\tau_{I/O}} \right\rceil + 1\right)\tau_{I/O}.$$

THEOREM 4-3. An NMOS or GaAs HEMT memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 11N\,\lambda_{gate}^2,$$

$$E = 5N\,\epsilon_{1\text{-}0}, \qquad T = 9\;lg\;N\;\tau_{gate} + 6\sqrt{A}\;\tau_{wire}.$$

(The energy figure given above is for the worst case, when all memory cells store a '0'. We have designed our memory so that less than half of the gates are in their high-power logic-1 output state, at any given time.)

THEOREM 4-4. A JJ-CS or CMOS memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 6N\,\lambda_{gate}^2,$$

$$E = 3\sqrt{A}\,lg\;N\;\epsilon_{wire}, \qquad T = 9\;lg\;N\;\tau_{gate} + 6\sqrt{A}\;\tau_{wire}.$$

THEOREM 4-5. A CMOS-SOS memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 6N\,\lambda_{gate}^2,$$

$$E = 1/2\;lg^2 N\;\epsilon_{sink} + 3\sqrt{A}\,lg\;N\;\epsilon_{wire}, \qquad T = 9\;lg\;N\;\tau_{gate} + 6\sqrt{A}\;\tau_{wire}.$$

Note that E_{wire} dominates E_{sink} in this construction, despite the fact that $\epsilon_{sink} >> \epsilon_{wire}$. Thus it is not accurate to describe all CMOS-SOS circuitry as consuming energy proportional to E_{sink}.

Using bit-serial address decoding on a minimax-edgelength version of the H-tree [12], it is possible to reduce E_{wire} to $O(\sqrt{A})$. We are presently trying to evaluate the constant factors in this construction, to see if it is of practical interest.

5. Conclusions

This report covers the first phase of an ambitious project, to develop a universal "constant factor" model of VLSI energy, area and time. Many troublesome areas have been identified:

1. The model is only applicable to gate-level design. Pass-transistor and other forms of switch-type logic design are not modelled adequately.
2. No pseudo-static memory cells are allowed. Thus we cannot as yet discuss the "dynamic" memory designs that currently achieve the best density and energy figures.
3. The constants in Table 1 are undoubtedly inaccurate, and would benefit from a more careful study. (The most pressing question is whether there exist values for these constants that would accurately predict area, time, and energy. If not, parameters must be added to the model. Alternatively, we may be able to eliminate one or more parameters without jeopardizing our goal of attaining "factor-of-ten" accuracy.)
4. At least one bipolar technology which allows wire fanout (such as ECL) should be added to Table 1. As it stands, bipolar gets very short shrift, despite its current importance in high-speed static memory designs.
5. It would be very hard to make lower bound arguments on the basis of the model, especially in view of the unrealistic restrictions mentioned in items 1 and 2 above. Once the model is extended, the arguments of [10] and [7] could be employed to prove lower bounds on CMOS-like technologies. Such lower bounds would presumably be valid, but not tight, for other present-day technologies.

Despite these problems, we have succeeded in developing a circuit model that is capable of handling most of today's technological diversity. We have applied this model to the problem of designing megabit memories, and we have analyzed the area, time and energy performance of circuit constructions in various technologies.

Acknowledgment

This research could not have proceeded without Prabhakar Raghavan's comments and criticisms on innumerable revisions of the model. I'd also like to acknowledge Vallath Nandarkumar, for his aid with the JJ-CIL and JJ-CS technologies, and Thomas Lengauer, for a helpful discussion on theoretical modelling. Finally, I'm grateful for the hospitality shown me during and after WG'83, especially by J. Ebert, T. Lengauer, and M. Nagl.

References

1. Masayuki Abe, Takashi Mimura, Naoki Yokoyama, and Hajime Ishikawa, "New Technology Towards GaAs LSI/VLSI for Computer Applications," *IEEE Trans. on Electron Devices*, vol. ED-29, no. 7, pp. 1088-1093, July 1982.
2. W. Anacker, "Computing at 4 degrees Kelvin," *IEEE Spectrum*, pp. 26-37, May 1979.
3. Charles H. Bennett, "The Thermodynamics of Computation -- a Review," *International Journal of Theoretical Physics*, vol. 21, 1982.
4. S. A. Evans, "Scaling I^2L for VLSI," *IEEE Journal of Solid-State Circuits*, vol. SC-14, no. 2, pp. 318-326, April 1979.
5. T. R. Gheewala, "Design of 2.5-Micrometer Josephson Current Injection Logic (CIL)," *IBM J. Res. Develop.*, vol. 24, no. 2, pp. 130-142, March 1980.
6. P. Guéret, A. Moser, and P. Wolf, "Investigations for a Josephson Computer Main Memory with Single-Flux-Quantum Cells," *IBM J. Res. Develop.*, vol. 24, no. 2, pp. 155-166, March 1980.
7. Gloria Kissin, "Measuring Energy Consumption in VLSI Circuits: a Foundation," in *Proc. 14th Annual ACM Symp. on Theory of Computing*, pp. 99-104, May 1982.
8. H. Ko and T. Van Duzer, "Miniaturization of Josephson Current Injection (CIL) Logic Circuits," in *Int'l Conf. on Computer Design, VLSI in Computers*, New York, October 1983.
9. F. T. Leighton, "Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI (Ph. D. Dissertation)," MIT/LCS/TR-724, M.I.T. Lab for Computer Science, June 1982.
10. Thomas Lengauer and Kurt Mehlhorn, "On the Complexity of VLSI Computations," in *VLSI Systems and Computations*, ed. H. T. Kung, Bob Sproull, Guy Steele, pp. 89-99, Computer Science Press, October 1981.
11. C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.
12. M. S. Paterson, W. L. Ruzzo, and L. Snyder, "Bounds on Minimax Edge Length for Complete Binary Trees," in *Proc. 13th Annual ACM Symp. on Theory of Computing*, pp. 293-299, May 1981.
13. C. D. Thompson, "A Complexity Theory for VLSI," Ph. D. Dissertation, CMU-CS-80-140, Computer Science Dept., Carnegie-Mellon University, August 1980.

TO APPEAR IN PROC '83
GRAPHTHEORETIC WORKSHOP

On the Energy-Time-Area Cost of a Memory Access

Clark D. Thompson
Computer Science Division
573 Evans Hall
University of California
Berkeley, CA 94720

ABSTRACT

Metrics of energy, area, and time are defined for a graph-theoretic model of VLSI computation. Different "constant factors" are seen to be appropriate for different logic families. We examine seven such families: NMOS, CMOS, CMOS–SOS, I^2L, GaAs HEMT, JJ–CIL, and JJ–CS. For each family, we sketch a construction for an energy-efficient, read/write, random-access memory circuit.

1. Introduction

Complexity theorists have traditionally focussed on two metrics of the quality of a circuit design: size (the number of gates involved) and speed (the number of gates on the longest path from any input to any output). More recently, notions of circuit area and wire delay have come into play [13], due to their growing importance in VLSI design.

Very little effort, by contrast, has been put into the study of the power and energy requirements of computation. This theoretical oversight might seem strange, in view of the practical importance of the subject. Energy is a critical resource in battery-powered systems, and heat (*i.e.*, power) dissipation is an important constraint on the physical design of almost every computational system.

The theoretical difficulties of the area are easily described: there is no simple, accepted, widely-applicable model of the energetics of computation. A literature search yields scanty and contradictory results. It is hard to reconcile the assumptions in Chapter 9 of [11] (seemingly written by Chuck Seitz), with Bennett's monograph [3], with Kissin's recent paper [7], or with Lengauer and Mehlhorn's

This work was supported by the National Science Foundation Grant ECS-8110684.

article [10]. Bennett argues that, in principle, one can do any amount of computation with a small amount of energy. Seitz takes the opposite position to Bennett, maintaining that "gate switching energy" is a fundamental property of any logic technology. Seitz' assumptions are, in turn, too weak for the models of Kissin, Lengauer and Mehlhorn. Their results are motivated by the dependence of switching energy on wirelength in the CMOS technology.

This paper attempts to inform the controversy surrounding models of energy consumption. A unified model is proposed for all present-day (and a couple of futuristic) integrated-circuit technologies. When placed on this common basis, it is easy to see that there are almost as many "fundamental modes" of energy consumption as there are technologies.

Some technologies consume power at a nearly constant rate per logic gate, while others exhibit a state-dependent rate of power consumption. In either case, total energy consumption might be called "static" since very little additional "switching energy" is required.

A second major grouping of contemporary technologies has little static energy dissipation, but consume significant switching energy. As suggested by [7] and [10], switching energy might be proportional to wire length. Alternatively, it might be essentially independent of wire length [11]. (A final possibility, a Bennett-style [3] zero-energy technology, is unlikely to be seen in this century.)

The technological diversity outlined above is captured in this paper's unified model by a number of technological "constant factors." As a check on the model's completeness, constructions have been attempted for energy-efficient memory access in each technology. These are outlined in Section 4; Section 2 contains the unified model assumptions, and Section 3 lists the technological constant factors.

2. Model of Computation

In the following assumptions, greek letters are used for the technological constant factors. (There are two exceptions: δ and Δ bear their standard meaning of vertex in- and out-degree.) Sets and their elements are defined by capitalized and lower case roman letters, respectively.

1. *Sources* O_h, *sinks* I_h. A computation graph is a directed hypergraph $G = (V, H)$. A hyperedge h is denoted by an ordered pair (O_h, I_h) of vertex sets $O_h \subseteq V$, $I_h \subseteq V$. The vertices in O_h are the s*O*urces of h; the vertices in I_h are its s*I*nks.

2. *Edge fanout restrictions o_{max}, ι_{max}.* Each edge h has at least one, and at most $O(1)$, sources and sinks:
$$1 \leq |O_h| \leq o_{max}, \qquad 1 \leq |I_h| \leq \iota_{max}$$
Limits on vertex indegree δ and outdegree Δ are discussed in Assumption 10.

3. *Vertex widths λ_{gate}, $\lambda_{I/O}$.* Each vertex in a computation graph is embedded as a square region in the Euclidean plane. No two vertices overlap. The size of an embedded vertex depends upon its functionality: gates $v \in V_{gate}$ occupy λ^2_{gate} area, while I/O ports $v \in V_{I/O}$ occupy $\lambda^2_{I/O}$ area.

4. *Edge width λ_{wire}, number of wiring layers μ.* An edge is embedded as a connected set of wire segments. Each wire segment is a rectangle of width λ_{wire} and arbitrary length, placed on one of μ planar wiring layers stacked above the plane of the vertices. A wire segment on the bottom wiring layer connects to the vertices it passes over. Two wire segments are connected to each other if they pass over the same point and if they are on either the same layer or an adjacent layer. (Note that $\lceil \mu/2 \rceil$ disconnected wire segments may pass over the same point in the vertex plane. Also note that any hyperedge h can be embedded as a tree of wire segments passing over the vertices in $O_h \cup I_h$.)

5. *Total area A, maximum total area α_{max}.* The total area A of an embedded computation graph is the area of the smallest square that encloses all its vertices and wire segments. The area of this square is bounded by a technological constant: $A \leq \alpha_{max}$.

6. *Maximum edge length λ_{max}.* The total length $||h||$ of an (embedded) edge h is the sum of the lengths of its wire segments. Edge lengths are bounded by a constant: $\forall h \; ||h|| \leq \lambda_{max}$.

7. *Votes $v(t)$, signals $h(t)$.* The state of the computation graph at any time t is defined by a vector $(V(t), H(t))$ of votes $v(t)$ and signals $h(t)$ associated with each vertex v and hyperedge h. The value of a vote or signal is taken from the ternary set $\{0, 1, u\}$: logic-0, logic-1, and undetermined. (An alternative formulation, found in [10] and in state-of-the-art circuit simulators, takes signal values from a two-dimensional set of voltages V and impedances R.)

8. *Maximum size of voting equivalence class ξ_{max}, edge delay d_h, time constants τ_{gate}, τ_{wire} and τ_{fanout}, transmission line indicator ς, signal rise time r_h.*

   a. In many technologies, the delay associated with a wire can be decreased by driving that wire with a larger transistor. Such high-power drivers can be represented by several (unit-power) sources with identical voting behavior. We are thus led to the following definition of equivalence classes $C_{h,i}$ on the voting behavior of the sources for each edge h:

$$v_1,v_2 \in C_{h,i} \iff (v_1,v_2 \in O_h) \wedge (\forall t \quad v_1(t) = v_2(t))$$

A technological limit on driving power translates into a restriction on the size of (*i.e.*, number of vertices in) any voting equivalence class:

$$\forall h,i \quad |C_{h,i}| \leq \xi_{max}$$

b. At the time of circuit construction ($t = 0$), a fixed but indeterminate delay d_h is assigned to each edge h. An edge's delay (in a worst-case analysis) is proportional to its length $||h||$ and number of sinks $|I_h|$, and inversely proportional to the size of its smallest equivalence class $C_h = \min_i |C_{h,i}| \leq \xi_{max}$:

$$d_h = \tau_{gate} + \frac{||h||\tau_{wire} + |I_h|\tau_{fanout}}{C_h} (\pm 50\%)$$

(Indeterminacy is introduced into the definition of edge delays to force "realistic" design practices, *e.g.*, self-timed or clocked logic.)

c. We define r_h to be the rise time of a signal on edge h. For technologies in which wires are transmission lines, r_h is approximately equal to the gate delay τ_{gate}. We indicate this by assigning the value 1 to the 0-1 variable ς (a mnemonic is the common symbol Z for the impedance of a line). The other technological possibility is that the wires are essentially capacitive in nature (as long as their length does not exceed λ_{max}, as defined in Assumption 6). Thus

$$r_h = \begin{cases} d_h, & \text{if } \varsigma = 0 \\ \tau_{gate}, & \text{if } \varsigma = 1 \end{cases}$$

d. The value of a signal $h(t)$ is determined by the votes of its sources O_h, with delay d_h. We prevent the propagation of unreasonably-short signal pulsewidths by requiring the "election results" to be stable for at least r_h time units.

$$h(t) = \begin{cases} 1, \text{ if } \exists v \in O_h \ \forall s \in [t-d_h, t-d_h+r_h] \quad v(s) = 1, \textit{ else} \\ 0, \text{ if } \exists v \in O_h \ \forall s \in [t-d_h, t-d_h+r_h] \quad v(s) = 0, \textit{ else} \\ u \end{cases}$$

Note that this formulation allows "wire-oring": the signal on an edge becomes 1 if any of its source votes is 1 for at least r_h time.

9. *Symmetry indicator σ.*

a. Not all patterns of voting behavior are allowed in all technologies. One restriction is observed in the so-called "symmetric" technologies ($\sigma = 1$). In these technologies, the effects of logic-1 votes and logic-0 vote are symmetric, making "wire-oring" infeasible. (A system of "majority-rule" is conceivable but not observed in any present-day logic family, possibly because it would reduce noise margins.) To outlaw wire-oring, we permit just one equivalence class per edge:

$$(\sigma = 1) \implies (\forall t\ \forall h\ \forall v_1, v_2 \in O_h\ \ v_1(t) = v_2(t))$$

b. A second type of restriction on allowable voting behavior arises in the asymmetric $(\sigma = 0)$ technologies. We must restrict the number of high-power logic-1 votes that appear at one time on an edge, to avoid exceeding the current density limit mentioned in Assumption 8a:

$$(\sigma = 0) \implies \left(\forall t \forall h\ \ |\{v \in O_h\ :\ v(t) = 1\}| \le \xi_{max}\right)$$

10. *Logic family ϕ, power supply period τ_{supply}, I/O schedule S, external clock period $\tau_{I/O}$.*

a. A logic family ϕ is a technologically-constrained set of triples (δ, f, Δ). The first and third parts of a triple denote the indegree and outdegree of one type of gate. The second part of a triple defines a functionality, or voting behavior. A gate with the 'and' functionality, for example, is modelled by a vertex whose vote is the logical 'and' of the signals on its in-edges. As another example, the 'latch' function depends on a delayed feedback signal. Finally, the voting behavior of gates in the JJ–CIL technology depends upon the phase of their AC power supply. Thus, in the general case, the functionality f_v of a gate v has $2 + \delta_v$ parameters, and defines the gate's vote as follows:

$$v(t) = f_v\left(v(t - \tau_{wire}), c(t), h_1(t), h_2(t), \ldots h_{\delta_v}(t)\right)$$

where the phase of the power supply (assumed to have a 90% duty cycle) is

$$c(t) = \begin{cases} 1, & \text{if } t < \left(.1 + \lfloor t/\tau_{supply} \rfloor\right) \tau_{supply} \\ 0, & \text{otherwise} \end{cases}$$

Note that voting is a zero-delay process, since gate delays were included in the definition of edge delay d_h.

b. An I/O port $v_i \in V_{I/O}$ has $\delta_v = 1$, $\Delta_v = 1$. Its voting is determined by an externally-imposed I/O schedule $S_i \in \{r_0, r_1, r_u, w_0, w_1, w_u\}^*$. Each literal in S_i indicates whether the I/O port is to read (r_0, r_1, r_u) or write (w_0, w_1, w_u) a '0', a '1', or a 'u'. The k-th literal in S_i refers to the k-th external clock period defined by $t \in ((k-1)\tau_{I/O}, k\tau_{I/O}]$, where $\tau_{I/O}$ is a technological constant. Thus, if the k-th literal is r_x, the port votes $v_i(t) = x$ during the k-th clock period. Alternatively, if the k-th literal is w_y, we say the schedule S_i is "satisfied" only if the port's in-edge h has signal $h(t) = y$, for all times t in the k-th clock period. (If the output bit for some time period is u, i.e. undetermined, we allow $h(t)$ to be any value.)

11. *Energy consumption* $E_{standby}$, E_{1-0}, E_{wire}, E_{sink}, E. Four modes of energy consumption are observed in physical realizations of computation graphs.

    a. A constant power dissipation of $\epsilon_{standby}/\tau_{gate}$ is associated with every gate. Total "standby" energy dissipation over the period $[t_1, t_2]$ is thus defined as

$$E_{standby} = \sum_{v \in V_{gate}} \frac{(t_2 - t_1)}{\tau_{gate}} \epsilon_{standby}$$

    b. In asymmetric (wire-or) technologies, a gate voting 1 consumes more power than a gate voting 0. We define energy ϵ_{1-0} so that the difference between these two levels of power consumption is $\epsilon_{1-0}/\tau_{gate}$. Total energy consumption in this mode is thus

$$E_{1-0} = \sum_{v \in V_{gate}} \int_{\substack{t_1 \le t \le t_2 \\ v(t) = 1}} \frac{\epsilon_{1-0}}{\tau_{gate}} dt$$

    By Assumption 8d, a gate's vote can change a signal only if it persists for at least $r_h \geq \tau_{gate}$ time. We thus employ the following (approximate) expression for E_{1-0}:

$$E_{1-0} = \sum_{v \in V_{gate}} \sum_{\substack{t_1 / \tau_{gate} \le k \le t_2 / \tau_{gate} \\ v(k\tau_{gate}) = 1}} \epsilon_{1-0}$$

    c. Each change in an edge's signal consumes energy proportional to the length of that edge. Assuming such signal changes occur at a frequency less than $1/\tau_{gate}$, we write

$$E_{wire} = \sum_{h} \sum_{\substack{t_1 / \tau_{gate} \le k \le t_2 / \tau_{gate} \\ h(k\tau_{gate}) \ne h((k+1)\tau_{gate})}} ||h|| \epsilon_{wire}$$

    d. Energy E_{sink}, like E_{wire}, is a form of "switching energy." In this case, the energy consumption is proportional to the number of sinks:

$$E_{sink} = \sum_{h} \sum_{\substack{t_1 / \tau_{gate} \le k \le t_2 / \tau_{gate} \\ h(k\tau_{gate}) \ne h((k+1)\tau_{gate})}} |I_h| \epsilon_{sink}$$

    e. The total energy consumed by a computation is $E = E_{standby} + E_{1-0} + E_{wire} + E_{sink}$.

3. Technological Parameters

	I^{2}L	JJ–CIL	NMOS	HEMT	JJ–CS	CMOS	CMOS–SOS	units
λ_{gate}	4	10	7	7	100	10	10	*um*
λ_{wire}	1	2	1	1	4	1	1	*um*
λ_{max}	10^4	10^5	10^4	10^4	10^5	10^4	10^4	*um*
$\tau_{gate}, \tau_{fanout}$	100	10	100	50	1000	100	100	*pS*
τ_{wire}	0.1	0.01	1	1	1	1	0.1	pS/λ_{wire}
τ_{supply}		1000						*ps*
$\epsilon_{standby}$	0.1	0.01	$\sim$0	$\sim$0	$\sim$0	$\sim$0	$\sim$0	*fJ*
ϵ_{1-0}			0.1	0.1	$\sim$0	$\sim$0	$\sim$0	*fJ*
ϵ_{wire}					0.1	0.1	0.001	fJ/λ_{wire}
ϵ_{sink}							0.1	*fJ*
ξ_{max}	1	1	10	10	1	10	10	--
σ	0	1	0	0	1	1	1	--
ς	0	1	0	0	0	0	0	--
o_{max}	10^2	1	10^2	10^2	1	$(=\xi_{max})$	$(=\xi_{max})$	--
ι_{max}	1	1	10^4	10^4	10^4	10^4	10^4	--

Table 1. Constant factors for seven VLSI technologies (multiplicative factors for units are $f = 10^{-15}$, $p = 10^{-12}$, $n = 10^{-9}$, and $u = 10^{-6}$).

Table 1 gives approximate values for the "constant factors" of seven VLSI technologies. The values are (hopefully) correct to within a factor of 10, for circuits built with about .5 *um* line widths during the late 1980's [1, 2, 4, 5, 6, 8, 11]. Thus λ_{wire}, the minimum wire-wire spacing, is equal to 1 *um* for most of the technologies in the table. The only technologies with $\lambda_{wire} > 1$ *um* are JJ–CIL and JJ–CS, in which tree-shaped model edges must be traversed with a non-branching wire. At one or two wires per edge, this implies that the effective edge widths for JJ–CIL is between 1 and 2 *um*. And, since double-rail logic is used in JJ–CS, its λ_{wire} increases by another factor of two, to approximately 4 *um*.

An important feature of Table 1 is the diagonal structure of the entries for circuit energies $\epsilon_{standby}$, ϵ_{1-0}, ϵ_{wire}, and ϵ_{sink}. When calculating total energy, contributions from entries below the diagonal can be ignored. For example, the technologies with $\epsilon_{standby} > 0$ have a nearly constant power dissipation per gate which does not increase by more than 10% when the gates change their state at maximum frequency.

Table 1 is not quite a complete list of the parameters in the model. The following are nearly constant over all technologies:

$$\alpha_{max} = 10^9\ um^2, \quad \tau_{I/O} = 10\ nS,$$

$$\mu = 4 \text{ to } 6 \text{ layers}, \quad \lambda_{I/O} = 10^2\ um.$$

Note that, by Assumption 4, $\mu = 6$ corresponds to a three-level metal process. The other $\mu - 3$ layers are made of an insulating material, through which small square holes or "vias" are cut.

Finally, we specify the logic family ϕ of gates and latches available in each technology. For I^{2}L, ϕ consists of just a 1-input, 4-output inverter (all logic must be done by "wire-oring"). For NMOS and GaAs HEMT, we assume that ϕ contains a 2-input, 1-output 'nand'; a 4-input, 1-output 'nor'; and a 3-input, 1-output inverting D-latch. For all other technologies, we augment the NMOS ϕ with gates for all 2-input, 1-output boolean functions.

4. Constructions

Using the model of Section 2 and the technological parameters of Section 3, it is possible to estimate the area, time and energy performance of various circuit constructions.

In this section, we report the performance of our best constructions for large ($\sim 10^6$ bit), random-access, read-write memories in each of the seven technologies. Due to space limitations, only one of the constructions will be described in any detail.

First of all, we need a formal definition of the "N-bit memory access problem." For concreteness and simplicity, we assume that N is a power of 2, writing $n = \log_2 N = lg\ N$. We also assume the existence of a fully parallel interface that runs memory cycles as often as possible. Thus we provide n separate address lines into the memory. A new address is available on these lines every T time units, where the length T of a memory cycle is assumed to be an exact multiple of the external clock period $\tau_{I/O}$. These considerations lead to the following definition:

Memory access with cycle time T. A computation graph implements an N-bit memory access in time T if it has $n + 3$ I/O ports whose schedules obey the following restrictions. Every $T/\tau_{I/O}$ external clock periods, n address bits are read in through ports labeled a_n, a_{n-1}, ... a_1. The most significant address bit is read by a_n. Every time these address bits are read, port *we* (mnemonic: "write enable") reads a bit. If it reads a '1', the bit read by *din* ("data input") is to be written into the addressed memory location. Alternatively, if the port *we* reads a '0', a read from the addressed bit is enabled. In this case, the port *dout* ("data output") should write the correct value for the bit being accessed. (If the accessed bit has not been written into previously, *dout* may be any value.) The value written by the port *dout* must be available during the last external clock cycle of the memory cycle.

More formally, the computation graph must satisfy all I/O schedules S, where

$$S_i = \left((r_0, r_1)(r_u)^{k-1}\right)^*,$$

for all ports other than v_{dout} and where the integer k equals $T/\tau_{I/O}$. The schedule for v_{dout} is of the form

$$S_{dout} = \left((w_u)^{k-1} w_{m(S,t)}\right)^*,$$

where the values $m(S,t)$ to be written by the port are, of course, dependent upon the values written into the memory by S up to that time t.

4.1. I^{2}L

As described in Section 3, the I^{2}L logic family has just one type of gate and no latches. Lacking a latch, it seems that about 10 gates are required for each bit of memory: see Figure 1. The state of this memory cell is available on the *dout* line, whenever both x and y are at logic-1. Otherwise, $dout = 0$. The state of the memory cell is updated, to the value of *din*, whenever $x = y = we = 1$.

It is somewhat unrealistic to use a 10-gate memory cell, since an analog circuit designer could undoubtedly design a functionally-equivalent cell occupying less than one quarter of the area. We shall see, however, that the use of a smaller memory cell would not markedly decrease the area of the complete memory circuit.

Figure 2 shows a floor plan of our I^{2}L memory design. The upper rectangular box decodes the top $1/2\ lg\ N = n/2$ address bits into $\sqrt{N}$ *column select* lines. It also fans out the *din* bit into $\sqrt{N}$ lines, one per column in the $(\sqrt{N} \times \sqrt{N})$-cell memory array.

The left-hand rectangular box performs a similar function, decoding the least significant address bits onto $\sqrt{N}$ *row select* lines, and fanning out the *we* signal.

The data output bit, *dout*, is obtained by or-ing together the outputs of all

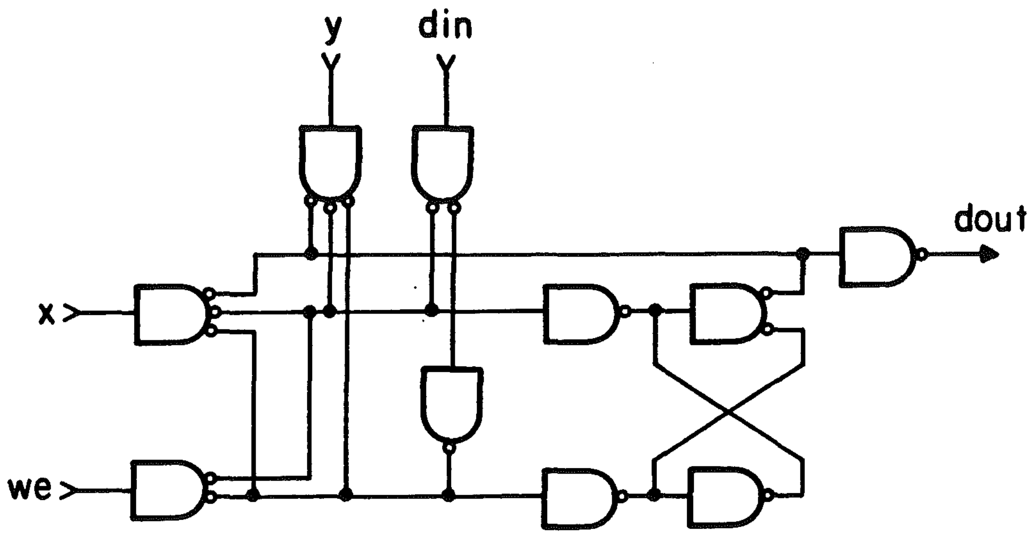

Figure 1. An I^{2}L memory cell.

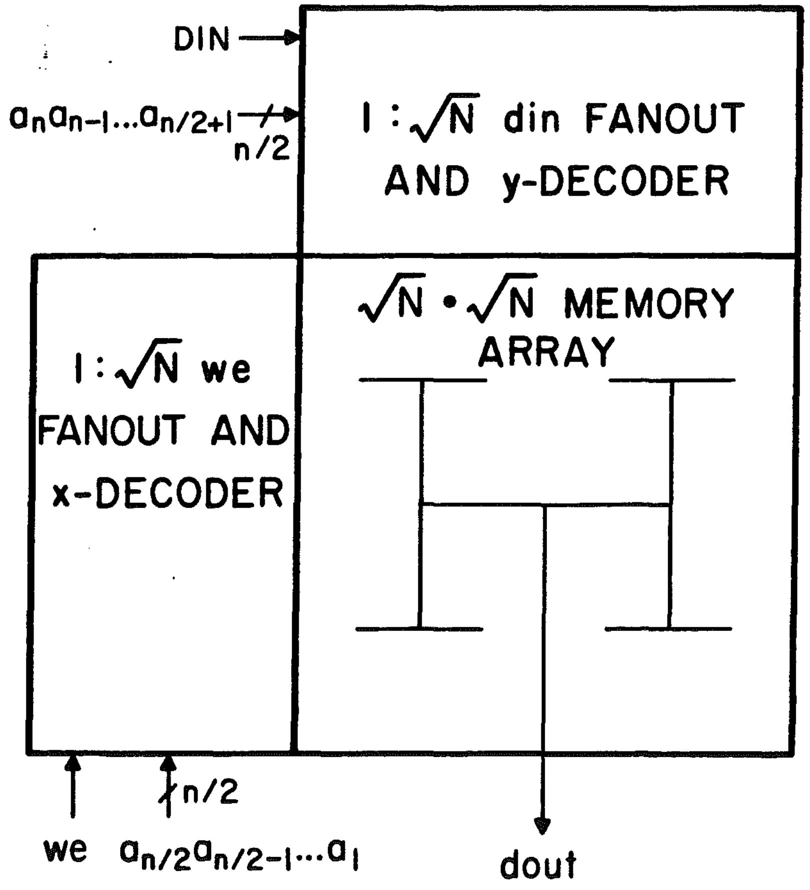
DIN
$a_n a_{n-1} \dots a_{n/2+1}$
n/2
$1 : \sqrt{N}$ din FANOUT AND y-DECODER
$\sqrt{N} \cdot \sqrt{N}$ MEMORY ARRAY
$1 : \sqrt{N}$ we FANOUT AND x-DECODER
n/2
we
$a_{n/2} a_{n/2-1} \dots a_1$
dout

Figure 2. Floor plan for I^{2}L memory.

the memory cells. This is done in an "H-tree" pattern: the data outputs of a square of $o_{max} = 100$ adjacent memory cells are wire-ored to form the input signal for an inverter. A second inverter produces a positive-true signal, 100 of which can be wire-ored in the next level of the tree. The fanin tree is thus $2 \lceil \log_{100} N \rceil$ gates deep.

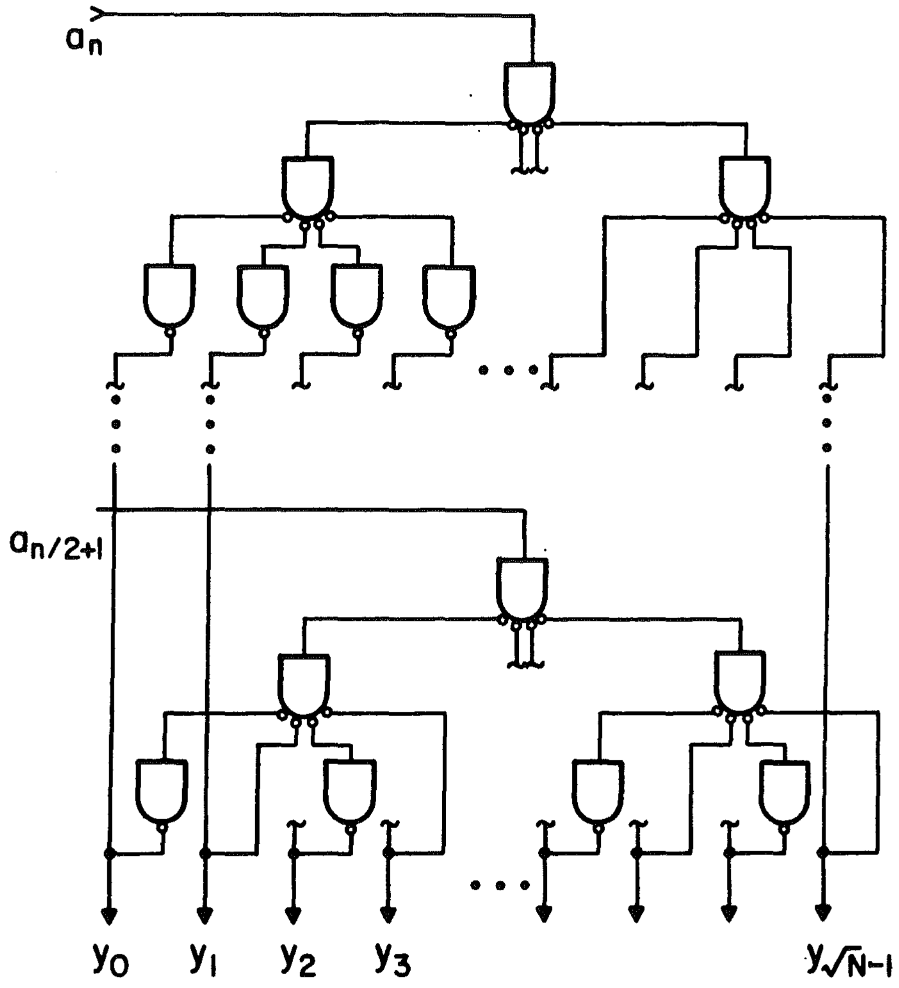

Figure 3. An I^2L *y*-decoder.

Figure 3 shows a construction for a $n/2$-bit to $\sqrt{N}$-line decoder. Note that multiple sinks are not allowed in I^2L (since $\iota_{max} = 1$). The only way to fan out a signal is to use a 4-output inverter, so $\log_4 \sqrt{N} = n/4$ levels of gating are required. Decoding is trivial, using $n/2$-way wire-ors. All in all, we need slightly fewer than $n/2 \sqrt{N} = 1/2 \sqrt{N} \; lg \; N$ gates to construct the circuit of Figure 3. One additional fanout tree (containing about $1/3 \sqrt{N}$ gates) is needed for *din* or *we* fanout, completing the construction of the rectangular boxes in Figure 2.

Figure 4 indicates why I^2L is not particularly suited for memory circuits. Since wire fanout is not allowed, $1/3 \sqrt{N}$ gates are required for each row of the memory array, to fanout the *row select* signal. Another $1/3 \sqrt{N}$ gates per row

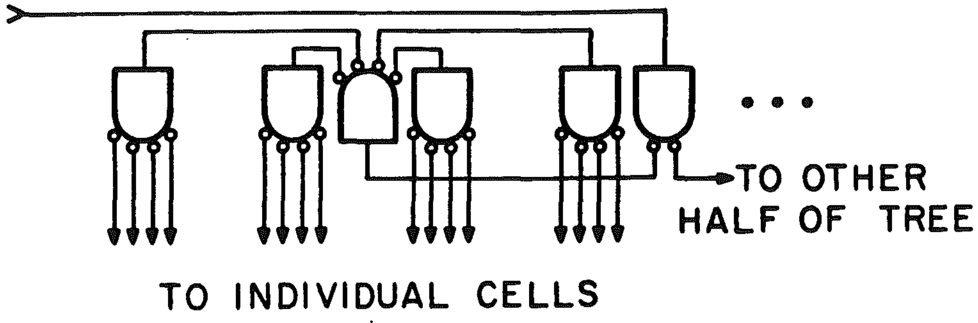

Figure 4. Row fanout tree for I^{2}L.

are needed for the *we* lines.

The tree fanout construction of Figure 4 was chosen to minimize total gate delay, at some expense in additional wire area. It requires one gate position for every two memory cells, so we need only allocate one gate in each memory cell for the two row fanout trees. (Alternatively, we could have fanned out the signals in a linear, left-to-right fashion, allocating two gate positions every three cells. This would result in a worst-case access time exceeding $1/3 \sqrt{N}\ \tau_{gate}$, significantly worse than what we obtain with a fanout tree.)

The row-row spacing of the memory array can now be estimated. The total number of gates per cell is 10 (for the D-latch of Figure 1) plus 1 (for the two row fanout trees) plus 1 (for the fanout of *column select* and *din* lines). In addition, a gap of λ_{gate} must be left between every 10 rows, to make room for the "H-tree" fanin of *dout*. Thus the total height or width of the $\sqrt{N} \times \sqrt{N}$ array is about $4\sqrt{N}\ \lambda_{gate}$, if we assume that we have enough room to fit the wiring.

Turning to the wiring area, it is apparent that the row fanout trees (Figure 4) could easily be a critical constraint. In fact, the row and column select lines of our memory circuit form a "mesh-of-trees" graph, requiring $\Omega(N\ lg^2 N\ \lambda_{wire}^2)$ area [9].

Considering Figure 4 more carefully, we find that we must allocate $(n/2)$ wiring tracks per row for each of the two row fanout trees. In addition, we should allocate about 5 horizontal tracks for wiring internal to each memory cell, and 1 track for the *dout* fanin. We thus need $n+6$ horizontal tracks per cell. In the vertical direction, the count is the same, so our row-row spacing must be at least $(2n+12)/(\lfloor \mu/2 \rfloor)\ \lambda_{wire}$. For the case that $N = 10^6$ and $\mu = 4$, this evaluates to a cell "pitch" of 16 *um*.

According to the parameters given in Section 3 for I^{2}L, $\lambda_{gate} = 4$ *um*. Thus our (4 gate by 4 gate) = (16 *um*)^2 memory cells are just large enough to accomodate the fanout wiring for $N \leq 10^6$ and $\mu = 4$. If we used smaller latches, or chose a larger N, we would have to increase μ to avoid "wasting" area on wiring.

To calculate the total area A of the memory, we must consider the decoders as well as the memory array. For $N \sim 10^6$, however, the area required by the decoders is negligible. The layout suggested by Figure 3 would be only $(lg^2 \sqrt{N})\, \lambda_{gate}$ high, and this height could be further reduced to $(lg \sqrt{N})\, \lambda_{gate} + (lg^2 \sqrt{N})\, \lambda_{wire}$ by using the tree-on-a-line idea of Figure 4. Thus we can report a total area of about $16N\, \lambda_{gate}^2 \sim 2.6\ cm^2$ for $N \sim 10^6$ and $\mu \geq 4$.

A time analysis of this construction is fairly straightforward. We split the total circuit delay into the three components appearing in the Assumption 8b's definition of d_h: $\sum d_h = T_{gate} + T_{wire} + T_{fanout}$. The longest delay path through the memory is from any address bit, through the decoder, through a row (or column) fanout tree, through a memory cell, and then out the *dout* fanin tree. Total number of gates encountered on this path is about $8 + 1/2\ lg\ N$. If $N \sim 10^6$, $T_{gate} \sim 18\ \tau_{gate} \sim 1.8\ ns$.

Since C_h is identically 1 in I^{2}L, T_{wire} is just the length of the longest path through the circuit times τ_{wire}. (C_h is upper-bounded by ξ_h and lower-bounded by 1.) This length is about 3/4 of the perimeter of the memory: the decoder contributes one side-length, as do the row fanout tree and the *dout* fanin tree. Total wirelength is thus about $3\sqrt{A}\ \lambda_{gate} = 48\sqrt{N}\ \lambda_{wire}$, yielding a delay term of $T_{wire} \sim 48\sqrt{N}\ \tau_{wire} \sim 4.8\ ns$ for $N \sim 10^6$.

In any I^{2}L circuit T_{fanout} is always equal to T_{gate}, since $\iota_{max} = 1$. The total delay for our memory is thus about $8.4\ ns$ for $N \sim 10^6$. Since $\tau_{I/O} = 10\ ns$, our memory should be able to satisfy an I/O schedule with $T = 2\ \tau_{I/O}$. Address and *din* lines would be valid for the first clock period of a memory cycle; *dout* would be valid on the second. (Since the d_h values in our model are indeterminate, not all memory circuits built according to our design would satisfy this schedule. It would be interesting to build a probabilistic model that captured the effects of process fluctuations and predicted the observed spread in access times among "identical" memory circuits.)

Total energy E per access is just the number of gates ($\sim 12N$) times (T/τ_{gate}) times $\epsilon_{standby}$, or about $240\ nJ$ when $N \sim 10^6$. Power consumption is thus $(240\ nJ)/(20\ ns) = 12\ Watt$, a very high figure for a $2.6\ cm^2$ chip. If our value for $\epsilon_{standby}$ is correct, I^{2}L gates will have to be more widely spaced than $\lambda_{gate} = 4\ um$. Note that decreasing the number of gates per I^{2}L latch from 10 to, say, 4 would result in a memory circuit of similar size and speed, but with half the power consumption per unit area.

In summary, we have proved the following.

THEOREM 4-1. An I^{2}L memory can be constructed with

$$N \sim 10^6, \qquad A = max\left\{16N\,\lambda^2_{gate},\ N\left((5 + lg\sqrt{N})/\lfloor\mu/2\rfloor\,\lambda_{wire}\right)^2\right\},$$

$$E = 12NT\,\epsilon_{standby}, \qquad T = \left(\left\lceil\frac{(2\,lg\,N + 5)\,\tau_{gate} + 3\sqrt{A}\,\tau_{wire}}{\tau_{I/O}}\right\rceil + 1\right)\tau_{I/O}.$$

4.2. Other technologies

Due to space limitations, we merely quote the area-time-energy performance of our best constructions to date for the other six technologies listed in Table 1.

THEOREM 4-2. A JJ–CIL memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 6N\,\lambda^2_{gate},$$

$$E = 6NT\,\epsilon_{standby}, \qquad T = \left(\left\lceil\frac{9\,lg\,N\,\tau_{gate} + 6\sqrt{A}\,\tau_{wire}}{\tau_{I/O}}\right\rceil + 1\right)\tau_{I/O}.$$

THEOREM 4-3. An NMOS or GaAs HEMT memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 11N\,\lambda^2_{gate},$$

$$E = 5N\,\epsilon_{1\text{-}0}, \qquad T = 9\,lg\,N\,\tau_{gate} + 6\sqrt{A}\,\tau_{wire}.$$

(The energy figure given above is for the worst case, when all memory cells store a '0'. We have designed our memory so that less than half of the gates are in their high-power logic-1 output state, at any given time.)

THEOREM 4-4. A JJ–CS or CMOS memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 6N\,\lambda^2_{gate},$$

$$E = 3\sqrt{A}\,lg\,N\,\epsilon_{wire}, \qquad T = 9\,lg\,N\,\tau_{gate} + 6\sqrt{A}\,\tau_{wire}.$$

THEOREM 4-5. A CMOS–SOS memory can be built with the following parameters:

$$N \sim 10^6, \qquad A = 6N\,\lambda^2_{gate},$$

$$E = 1/2\,lg^2N\,\epsilon_{sink} + 3\sqrt{A}\,lg\,N\,\epsilon_{wire}, \qquad T = 9\,lg\,N\,\tau_{gate} + 6\sqrt{A}\,\tau_{wire}.$$

Note that E_{wire} dominates E_{sink} in this construction, despite the fact that $\epsilon_{sink} >> \epsilon_{wire}$. Thus it is not accurate to describe all CMOS–SOS circuitry as consuming energy proportional to E_{sink}.

Using bit-serial address decoding on a minimax-edgelength version of the H-tree [12], it is possible to reduce E_{wire} to $O(\sqrt{A})$. We are presently trying to evaluate the constant factors in this construction, to see if it is of practical interest.

5. Conclusions

This report covers the first phase of an ambitious project, to develop a universal "constant factor" model of VLSI energy, area and time. Many troublesome areas have been identified:

1. The model is only applicable to gate-level design. Pass-transistor and other forms of switch-type logic design are not modelled adequately.
2. No pseudo-static memory cells are allowed. Thus we cannot as yet discuss the "dynamic" memory designs that currently achieve the best density and energy figures.
3. The constants in Table 1 are undoubtedly inaccurate, and would benefit from a more careful study. (The most pressing question is whether there exist values for these constants that would accurately predict area, time, and energy. If not, parameters must be added to the model. Alternatively, we may be able to eliminate one or more parameters without jeopardizing our goal of attaining "factor-of-ten" accuracy.)
4. At least one bipolar technology which allows wire fanout (such as ECL) should be added to Table 1. As it stands, bipolar gets very short shrift, despite its current importance in high-speed static memory designs.
5. It would be very hard to make lower bound arguments on the basis of the model, especially in view of the unrealistic restrictions mentioned in items 1 and 2 above. Once the model is extended, the arguments of [10] and [7] could be employed to prove lower bounds on CMOS-like technologies. Such lower bounds would presumably be valid, but not tight, for other present-day technologies.

Despite these problems, we have succeeded in developing a circuit model that is capable of handling most of today's technological diversity. We have applied this model to the problem of designing megabit memories, and we have analyzed the area, time and energy performance of circuit constructions in various technologies.

Acknowledgment

This research could not have proceeded without Prabhakar Raghavan's comments and criticisms on innumerable revisions of the model. I'd also like to acknowledge Vallath Nandarkumar, for his aid with the JJ–CIL and JJ–CS technologies, and Thomas Lengauer, for a helpful discussion on theoretical modelling. Finally, I'm grateful for the hospitality shown me during and after WG'83, especially by J. Ebert, T. Lengauer, and M. Nagl.

References

1. Masayuki Abe, Takashi Mimura, Naoki Yokoyama, and Hajime Ishikawa, "New Technology Towards GaAs LSI/VLSI for Computer Applications," *IEEE Trans. on Electron Devices*, vol. ED-29, no. 7, pp. 1088-1093, July 1982.
2. W. Anacker, "Computing at 4 degrees Kelvin," *IEEE Spectrum*, pp. 26-37, May 1979.
3. Charles H. Bennett, "The Thermodynamics of Computation -- a Review," *International Journal of Theoretical Physics*, vol. 21, 1982.
4. S. A. Evans, "Scaling I^2L for VLSI," *IEEE Journal of Solid-State Circuits*, vol. SC-14, no. 2, pp. 318-326, April 1979.
5. T. R. Gheewala, "Design of 2.5-Micrometer Josephson Current Injection Logic (CIL)," *IBM J. Res. Develop.*, vol. 24, no. 2, pp. 130-142, March 1980.
6. P. Guéret, A. Moser, and P. Wolf, "Investigations for a Josephson Computer Main Memory with Single-Flux-Quantum Cells," *IBM J. Res. Develop.*, vol. 24, no. 2, pp. 155-166, March 1980.
7. Gloria Kissin, "Measuring Energy Consumption in VLSI Circuits: a Foundation," in *Proc. 14th Annual ACM Symp. on Theory of Computing*, pp. 99-104, May 1982.
8. H. Ko and T. Van Duzer, "Miniaturization of Josephson Current Injection (CIL) Logic Circuits," in *Int'l Conf. on Computer Design, VLSI in Computers*, New York, October 1983.
9. F. T. Leighton, "Layouts for the Shuffle-Exchange Graph and Lower Bound Techniques for VLSI (Ph. D. Dissertation)," MIT/LCS/TR-724, M.I.T. Lab for Computer Science, June 1982.
10. Thomas Lengauer and Kurt Mehlhorn, "On the Complexity of VLSI Computations," in *VLSI Systems and Computations*, ed. H. T. Kung, Bob Sproull, Guy Steele, pp. 89-99, Computer Science Press, October 1981.
11. C. Mead and L. Conway, *Introduction to VLSI Systems*, Addison-Wesley, 1980.
12. M. S. Paterson, W. L. Ruzzo, and L. Snyder, "Bounds on Minimax Edge Length for Complete Binary Trees," in *Proc. 13th Annual ACM Symp. on Theory of Computing*, pp. 293-299, May 1981.
13. C. D. Thompson, "A Complexity Theory for VLSI," Ph. D. Dissertation, CMU-CS-80-140, Computer Science Dept., Carnegie-Mellon University, August 1980.