Fast Low-Power Decoders for RAMs

Bharadwaj S. Amrutur and Mark A. Horowitz, *Fellow, IEEE*

***Abstract*—Decoder design involves choosing the optimal circuit style and figuring out their sizing, including adding buffers if necessary. The problem of sizing a simple chain of logic gates has an elegant analytical solution, though there have been no corresponding analytical results until now which include the resistive effects of the interconnect. Using simple *RC* models, we analyze the problem of optimally sizing the decoder chain with *RC* interconnect and find the optimum fan-out to be about 4, just as in the case of a simple buffer chain. As in the simple buffer chain, supporting a fan-out of 4 often requires noninteger number of stages in the chain. Nevertheless, this result is used to arrive at a tight lower bound on the delay of a decoder. Two simple heuristics for sizing of real decoder with integer stages are examined. We evaluate a simple technique to reduce power, namely, reducing the sizes of the inputs of the word drivers, while sizing each of the subchains for maximum speed, and find that it provides for an efficient mechanism to trade off speed and power. We then use the *RC* models to compare different circuit techniques in use today and find that decoders with two input gates for all stages after the predecoder and pulse mode circuit techniques with skewed N to P ratios have the best performance.**

***Index Terms*—Decoder circuit comparison, low power, optimal decoder structure, optimal sizing, pulsed circuits, random access memory (RAM), resistive interconnect.**

I. Introduction

THE DESIGN of a random access memory (RAM) is generally divided into two parts, the decoder, which is the circuitry from the address input to the wordline, and the sense and column circuits, which includes the bitline to the data input/output circuits. For a normal read access, the decoder contributes up to half of the access time and a significant fraction of the total RAM power. While the logical function of the decoder is simple, it is equivalent to 2^n n-input AND gates, there are a large number of options for how to implement this function. Modern RAMs typically implement the large fan-in AND operation in an hierarchical structure [18]. Fig. 1 shows the critical path of a typical three-level decode hierarchy. The path starts from the address input, goes through the predecoder gates which drive the long predecode wires and the global word driver, which in turn drives the global wordline wire and the local word drivers and finally ends in the local wordline. The decoder designer has two major tasks: choosing the circuit style and sizing the resulting gates, including adding buffers if needed. While the problem of sizing a simple chain of gates is well understood, there are no analytical results when there is *RC* interconnect embedded within such a chain. We present analytical results and heuristics to size decoder chains with intermediate *RC* interconnect. There are many circuit styles in use for designing decoders. Using simple *RC* gate delay models, we analyze these to arrive at optimal decoder structures.

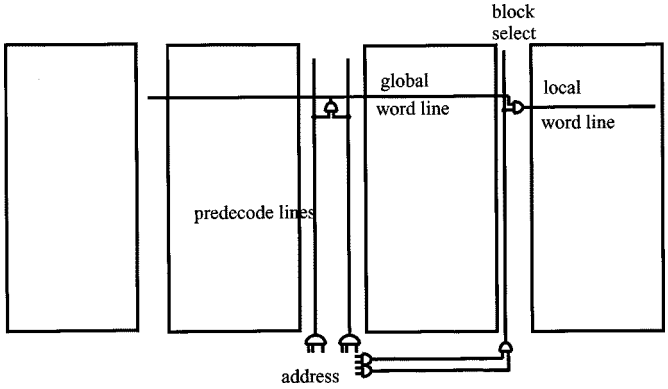

Fig. 1. Divided wordline (DWL) architecture showing a three-level decode.

Section II first reviews the approach of logical effort [9], [19], which uses a simple delay model to solve the sizing problem, and provides an estimate for the delay of the resulting circuit. This analysis allows us to bound the decoder delay and evaluate some simple heuristics for gate sizing in practical situations. Section III then uses this information to evaluate various circuit techniques that have been proposed to speed up the decode path. The decode gate delay can be significantly reduced by using pulsed circuit techniques [6]–[8], where the wordline is not a combinational signal but a pulse which stays active for a certain minimum duration and then shuts off. Fortunately, the power cost of these techniques is modest, and in some situations using pulses can reduce the overall RAM power. We conclude the paper by putting together a sketch of optimal decode structures to achieve fast and low-power operation.

II. Decoder Sizing

Estimating the delay and optimal sizing of CMOS gates is a well-studied problem. Jaeger in 1975 [1] published a solution to the inverter problem, which has been reexamined a number of times [2]–[5]. This analysis shows that for optimal delay, the delay of each stage should be the same, and the fan-out of each stage should be around 4. More recently, Sutherland and Sproull [9], [19] have proposed an approach called logical effort that allows one to quickly solve sizing problems for more complex circuits. We will adopt their approach to solve the decoder problem. The basic delay model they use is quite simple, yet it is reasonably accurate. It assumes that the delay of a gate is the sum of two terms. The first term is called the effort delay

Manuscript received November 21, 2000; revised June 28, 2001. This work was supported by the Advanced Research Projects Agency under Contract J-FBI-92-194 and by a gift from Fujitsu Ltd.

B. S. Amrutur is with Agilent Laboratories, Palo Alto, CA 94304 USA (e-mail: amrutur@stanfordalumni.org).

M. A. Horowitz is with the Computer Systems Laboratory, Stanford, CA 94305 USA.

Publisher Item Identifier S 0018-9200(01)08418-9.

and is a linear function of the gate's fan-out, the ratio of the gates's output capacitance to its input capacitance. This term models the delay caused by the gate current charging or discharging the load capacitance. Since the current is proportional to the gate size, the delay depends only on the ratio of the gate's load and its input capacitance. The second term is the parasitic delay. It models the delay needed to charge/discharge the gates's internal parasitic capacitance. Since the parasitics are proportional to the transistor sizes, this delay does not change with gate sizing or load. Thus using this model, the delay of a gate is simply $T_{\text{gate}} = k\,(C_{\text{out}}/C_{\text{in}}) + T_{\text{par}}$.

Logical effort goes one step further since it needs to optimize different types of gates in a chain. A complex gate like a static n-input NAND gate has n nMOS transistors in series, which degrades its speed compared to an inverter. Since all static n-input NAND gates will have the same topology, the constant k for all these gates will be the same and will be some k^* larger than an inverter. One can estimate k^* by using a simple resistor model of a transistor. If we further assume that the pMOS devices have 1/2 the current of an nMOS device, then a standard inverter would have an nMOS width of w and a pMOS width of $2w$. For the NAND gate to have the same current drive, the nMOS devices in this gate would have to be n times bigger, since there are n devices in series. These larger transistors cause the input capacitance for each of the NAND inputs to be $C_g(n+2)w$ compared to $C_g 3w$ for the inverter. k^* for this gate is $(n+2)/3$,[1] and is called the logical effort le of the gate. Thus, the delay of a gate is

$$T_{\text{gate}} = k_{\text{inv}}\left(le\frac{C_{\text{out}}}{C_{\text{in}}} + P_{\text{gate}}\right). \tag{1}$$

k_{inv} is delay added for each additional fan-out of an inverter, and P_{gate} is the effective added fan-out caused by the gate's parasitics. This formulation makes it clear that the only difference between an inverter and a gate is that the effective fan-out a gate sees is larger than an inverter by a factor of le. Ignoring the small difference in parasitic delays between inverters and gates, we can convert the gate sizing problem to the inverter sizing problem by defining the effective fan-out to be $le^*C_{\text{out}}/C_{\text{in}}$. Thus, delay is minimized when the effective fan-out is about 4 for each stage.

In the decode path, the signals at some of the intermediate nodes branch out to a number of identical stages, e.g., the global wordline signal in Fig. 1 splits to a number be of local word driver stages. The loading on the global wordline signal is be times the capacitance of the local word driver stage. If one focuses on a single path, the capacitance of all the other paths can be accounted for by making the effective fan-out of that stage $be^*le^*C_{\text{out}}/C_{\text{in}}$. The amount of branching at each node is called the branching effort of the node and the total branching effort of the path is the product of all the node branching efforts.

In general for a r to 2^r decode, the total branching effort of the critical path from the input or its complement to the output is

[1]Note that the actual logical effort is less than this formula since the devices are velocity saturated, and the current through two series devices is actually greater than 1/2. With velocity saturation, the transistors have to size up less than two to match the current through a single device. The theory of logical effort still holds in this case, one only needs to obtain the logical effort of each gate topology from simulation, or from more complex transistor models.

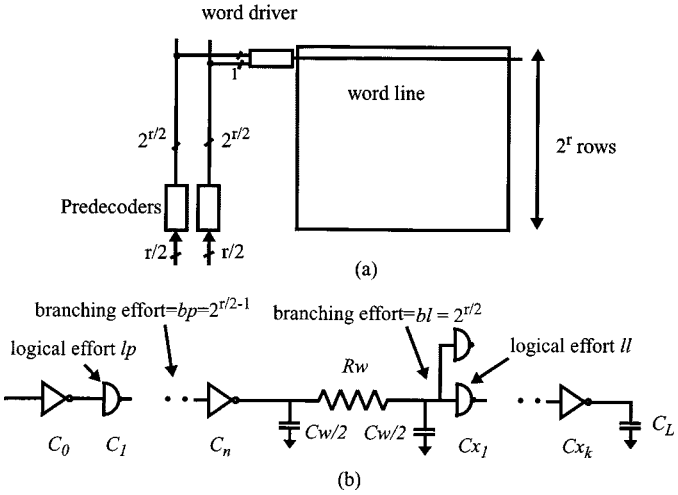

Fig. 2. (a) Schematic of small RAM with two-level decode. (b) Equivalent circuit of the critical path in the decoder. This models the predecode line which has all of its gate loading lumped at the end of the wire.

2^{r-1} since each input selects half of all the words in the RAM. The total logical effort of the path is the effort needed to build an r-input AND function. If the wire capacitance and resistance within the decoder are insignificant, then one could size all the gates in the decoder using just the total effective fan-out for each address line shown in (2). As we will see next in the context of two and three-level decoders, this is not a bad estimate when the wire delay is small.

$$\text{Effective fan-out} \approx \frac{C_{\text{load}}}{C_{\text{in}}}\cdot 2^{r-1} \cdot \text{Logical Effort}\,(r-\text{input}-\text{AND}). \tag{2}$$

A. Two-Level Decoders

Consider a design where r row address bits have to be decoded to select one of 2^r wordlines with a hierarchy of two levels. The first level has two predecoders each decoding $r/2$ address bits to drive one of $2^{r/2}$ predecode lines. The next level then ANDs two of the predecode lines to generate the wordline. This is a typical design for small embedded RAMs and is shown in Fig. 2. The equivalent critical path is shown in Fig. 2(b). Since the delay formulas only depend on the input capacitance of the gates, we use the input capacitance to denote the gate's size. We label the branching effort at the input to the wordline drivers as $bl(=2^{r/2})$, the logical effort of the NAND gate in the wordline driver as ll, and the branching effort and logical effort of the predecoder as bp and lp, respectively.

The total delay is just the sum of the delays of the gates along the decoder path, which in turn can be expressed as the sum of the effort delay plus the parasitic delay. The delay of the gate driving the wire only slightly complicates the expression:

$$\begin{aligned}\frac{D}{k_{\text{inv}}} =& \frac{C_1}{C_0} + \frac{lpC_2}{C_1}\cdots\frac{Cw + bl\cdot Cx_1}{C_n} \\ &+ \frac{RwCw}{2\alpha k_{\text{inv}}} + \frac{Rw\cdot bl\cdot Cx_1}{\alpha k_{\text{inv}}} + \frac{ll\cdot Cx_2}{Cx_1}\cdots\frac{C_L}{Cx_k}\cdots \\ &+ (n+k+1)\cdot P_{\text{gate}}\end{aligned} \tag{3}$$

where α is close to one and is a fitting parameter to convert wire resistance to delay.

Sizing the decoder would be an easy problem except for the predecoder wire's parasitic capacitance and resistance. Differentiating (3) with respect to the variables $C_1, \ldots C_n$ and $Cx_1, \ldots Cx_k$, and setting the coefficients of each of the partial differentials to zero, we get

$$\frac{C_1}{C_0} = \frac{lpC_2}{C_1} \cdots = \frac{Cw + bl \cdot Cx_1}{C_n} = f_1 \tag{4}$$

$$\left(\frac{Rw}{\alpha k_{\text{inv}}} + \frac{1}{C_n}\right) \cdot bl \cdot Cx_1 = \frac{ll \cdot Cx_2}{Cx_1} \cdots \frac{C_L}{Cx_k} = f_2. \tag{5}$$

The effective fan-out of the stages before the wire must all be the same, as must the effective fan-outs of the gates after the wire. The relation between the two fan-outs is set by the wire's parameters. The wire capacitance is part of the loading of the last gate in the first chain, and the resistance of the wire changes the effective drive strength of this gate when it drives the first gate of the second chain. The total delay can now be rewritten as

$$\frac{D}{k_{\text{inv}}} = (n+1) \cdot (f_1 + P_{\text{gate}}) + \frac{Rw \cdot Cw}{2\alpha k_{\text{inv}}} + \frac{Rw \cdot bl \cdot Cx_1}{\alpha k_{\text{inv}}} + k \cdot (f_2 + P_{\text{gate}}). \tag{6}$$

The total delay can be minimized by solving for the values for f_1, n, f_2, and k. Sizing the predecode chain is similar to sizing a buffer chain driving a fixed load and the optimal solution is to have $f_1 = f \sim 4$ as discussed in Section II. Intuitively, since the wire's parasitics will only slow the circuit down, the optimal sizing tries to reduce the effect of the wire. If the wire resistance is small, the optimal sizing will push more stages into the first subchain, making the final driver larger and reducing the effect of the wire capacitance. If the wire resistance is large, optimal sizing will push more stages into the second subchain, making the gate loading on this wire smaller, again reducing the effect of the wire. In fact, the optimal position of the wire sets n and k to try to balance the effects of the wire resistance and capacitance, such that

$$\frac{Rw}{\alpha k_{\text{inv}}} \cdot bl \cdot Cx_1 = \frac{Cw}{C_n}. \tag{7}$$

This is the same condition that is encountered in the solution for optimal placement of repeaters [22], and a detailed derivation is presented in [17]. Intuitively, if we were to make a small change in the location of the wire in the fanup chain, then if the above condition is true, the change in the delay of the driver will cancel out the change in delay of the wire. Putting (7) in (4) and (5), we find that the fan-outs of the two chains, f_1 and f_2, are the same. The constraints of a real design sometimes prevent this balance from occurring, since the number of buffers needs to be a positive, and often even, integer but we can use this optimal position of the wire to derive a lower bound on the delay.

If the wire did not exist, bl^*Cx_1/C_n would equal f, the stage effort. Since the wire exists, this ratio, f^*, will be less than f, since $(bl^*Cx_1 + Cw)/C_n$ must equal f. f/f^* is the effort cost of the wire, and can be found if the wire is optimally placed, so $f1 = f2 = f$. In that case, substituting f^* into (4) and (5) and setting them equal gives

$$\frac{Cw}{C_n} + f^* = f = \left(\frac{RwC_n}{\alpha k_{\text{inv}}} + 1\right) \cdot f^*. \tag{8}$$

Solving for f^* gives

$$f^* = f - f_w\left(\sqrt{1 + \frac{2f}{f_w}} - 1\right); \quad f_w = \frac{RwCw}{2\alpha k_{\text{inv}}} \tag{9}$$

where f_w is the wire delay measured in effective fan-out. The means that the minimal effort cost of a wire is

$$\frac{f}{f^*} = 1 + \frac{f_w}{f}\left(\sqrt{1 + \frac{2f}{f_w}} + 1\right) \tag{10}$$

and the total effort of a decoder path is

$$f^{n+k+1} = \frac{bp \cdot lp \cdot bl \cdot ll \cdot C_L}{C_0} \cdot \left(1 + \frac{f_w}{f}\left(\sqrt{1 + \frac{2f}{f_w}} + 1\right)\right). \tag{11}$$

Note here bp^*bl = total branching effort $= 2^{r-1}$ and lp^*ll = total logical effort of a r-input AND function. Hence (11) is similar to (2) except for the presence of factor dependent on the interconnect which diminishes as the intrinsic delay of the interconnect becomes negligible compared to a fan-out delay.

Once we know f^* we can also solve for C_n to find n and k.

$$f^n = \frac{bp \cdot lp \cdot Cw}{2fC_0} \cdot \left(1 + \sqrt{1 + \frac{2f}{f_w}}\right) \tag{12}$$

$$f^k = \frac{Rw \cdot bl \cdot ll \cdot C_L}{2\alpha k_{\text{inv}} f} \cdot \left(1 + \sqrt{1 + \frac{2f}{f_w}}\right) \tag{13}$$

Just like in the case of a simple buffer chain, the values of n, k will turn out to be noninteger in general and will have to be rounded to integer values. Nevertheless, the unrounded values can be used in (6) to yield a tight lower bound to the decoder delay. A useful parameter to consider is the ratio of the total input gate capacitance of the word driver to the predecoder wire capacitance, which we will call u and which equals bl^*Cx_1/Cw. We will evaluate two different heuristics to obtain sizing for real decoders which have integer number of stages. In the first heuristic H1, we keep the input gate size of Cx_1 obtained for the lower bound case, thus achieving the same gate to wire ratio u, as in the lower bound case. Since Cx_1 is fixed now, the sizing of the predecoder and the word driver chain can be done independently as in the standard buffer sizing problem. In the second heuristic H2, we will use (13) to estimate k, and then round it to the nearest even integer. We then use $f_2 = f \sim 4$ to calculate Cx_1, which fixes the predecoder problem, and it can be sized as the standard buffer chain. We also determine the optimal solution for integer number of stages by doing an exhaustive search of the variable values f_1 and f_2 between 2 to 7.5 and a small integer range of 2 to 10 for n and k. Table I compares the fan-outs, number of stages, and the delays normalized to a fan-out 4 loaded inverter and power, for the lower bound (LB), the optimal (OPT) and the heuristics H1 & H2 sizing. The energy is estimated as the sum of switching capacitances in the decoder. We see that the lower bound delay is fairly tight and

TABLE I
FAN-OUTS, DELAY, AND POWER FOR DIFFERENT SIZING TECHNIQUES IN 0.25-μm CMOS

Size	Intrinsic wire delay f_w (k_{inv})	Sizing Style	f_1	n	f_2	k	u	delay(τ_{fo4})	energy (arb Unit)	delay *energy
32x32	~0	LB	3.8	3.2	3.8	0.2	31.2	4.4	4.5	19.9
		H1	3.0	4	1.2	2	31.2	5.1	5.3	27
		H2	3.0	2	3.8	2	2.8	4.5	0.8	3.6
		OPT	3.2	2	3.6	2	3.2	4.5	0.8	3.8
64x64	~0	LB	3.8	3.9	3.8	0.7	15.4	5.6	4.8	27
		H1	3.7	4	1.6	2	15.4	6.1	5.1	30.9
		H2	3.5	3	3.8	2	2.8	5.8	1.5	8.7
		OPT	2.8	4	3.0	2	4.6	5.7	2.2	12.7
256x256	0.11	LB	3.8	4.3	3.8	2.4	3.5	7.7	6.6	50.8
		H1	4.3	4	4.9	2	3.5	8.0	6.2	49.9
		H2	3.5	5	3.8	2	5.6	7.9	9.2	72.1
		OPT	3.3	5	4.3	2	4.5	7.8	7.9	62.2
512x256	0.48	LB	3.8	4.4	3.8	3.0	1.5	8.5	7.2	61.2
		H1	4.4	4	7.3	2	1.5	9.3	6.6	61.1
		H2	3.2	6	3.8	2	5.6	9.2	17.2	158.7
		OPT	3.1	5	3.0	4	1.0	8.8	6.7	59.4
512x256	0.21	LB	3.8	4.7	3.8	2.6	2.6	8.3	9.4	77.8
		H1	4.8	4	5.7	2	2.6	8.9	8.5	75.4
		H2	3.2	6	3.8	2	5.6	8.6	17.2	147.4
		OPT	3.1	6	4.3	2	4.5	8.5	14.7	125.2

close to the optimal solution which uses only integer number of stages. Both the heuristics H1 and H2 give delay which are within 2% of the optimal solution, with H2 being slightly faster. For the large block of 512×256, with narrower wire, H1 and H2 are slower by 4%. But increasing the wire size gets them to within 2% of the optimum. We also notice that H2 consumes significantly more power for the larger sizes blocks. The critical parameter for power dissipation is u, the ratio of the word driver input gate cap to the predecoder wire cap. Larger value for u leads to more power dissipation. We will explore this aspect further in Section III. In the next section, we will look at sizing for three-level decoders.

B. Three-Level Decoder

Large RAMs typically use the divided wordline (DWL) architecture which uses an additional level of decoding, and so we next look at sizing strategies for three-level decoders. Fig. 3 depicts the critical path for a typical decoder implemented using the DWL architecture. The path has three subchains, the predecode, the global word driver and the local word driver chains. Let the number of stages in these be m, n, and k. Let bp, bg, and bl be the branching efforts of the predecoder, the inputs to the global and local word drivers, respectively, and let lp, lg, and ll be their logical efforts. For minimum delay, the fan-outs

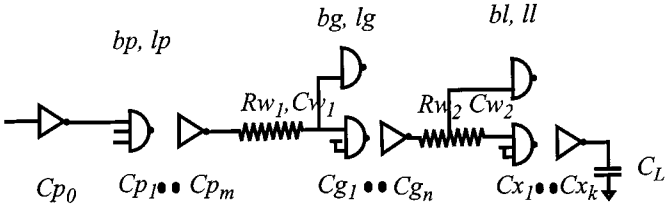

Fig. 3. Critical path for a three-level decoder.

in each of the predecoder, global word driver, and local word drivers need to be equal. We will call them f_1, f_2, and f_3, respectively. Like the two-level decoder case, if we can optimally size for the wires, all three of these fan-outs will be the same, and the detailed derivation is presented in [17]. Using this result, we can first calculate Cx_1, and then Cg_1. Using (13) as a reference, we can write the expression for k as

$$f^k = \frac{Rw_2 \cdot bl \cdot ll \cdot C_L}{2\alpha k_{\text{inv}} f} \cdot \left[1 + \sqrt{1 + \frac{2f}{f_{w2}}}\right]. \qquad (14)$$

As was done before, here f_{w2} is delay of the global wordline wire normalized to that of an inverter driving a fan-out of 4 load, i.e., $f_{w2} = Rw_2^* Cw_2/(2ak_{\text{inv}})$. This can be used to calculate the size of Cx_1 as $ll^* C_L/f^k$ to give the loading for the first two

TABLE II
FAN-OUTS, DELAY, AND ENERGY FOR THREE LEVEL DECODER IN 0.25-μm CMOS

Size	Intrinsic wire delay $f_{w1}+f_{w2}$ (in k_{inv})	Style	$f_1f_2f_3$	m,n,k	u	v	delay (τ_{fo4})	energy (arb. units)	delay*energy
1Mb	1.71	LB	3.8,3.8,3.8	4.6,2.9.2.2	1.9	1.0	11.7	49.7	582
		H1	3.4,7.3,4.5	5,2,2	1.9	2.0	12.0	46.4	559
		H2	4.2,3.8,3.8	5,2,2	2.6	4.4	12.7	90.1	1146
		OPT	3.4,3.0,3.8	5,4,2	2.6	0.8	11.8	58	683
1Mb	0.91	LB	3.8,3.8,3.8	4.7,2.8,2.0	2.7	1.5	11.2	62.1	695
		H1	3.6,6.7,3.7	5,2,2	2.7	1.5	11.4	57.9	662
		H2	4.2,3.8,3.8	5,2,2	2.6	4.4	11.6	90.1	1048
		OPT	3.4,3.0,3.4	5,4,2	3.3	1.0	11.3	65.7	741
4Mb	6.9	LB	3.8,3.8,3.8	4.8,3.8,2.9	0.8	0.4	14.8	64.3	951
		H1	3.7,2.6,7.0	5,4,2	0.8	0.4	15.1	65.2	984
		H2	3.8,3.8,3.8	5,4,2	2.6	0.6	15.5	99	1534
		OPT	3.6,3.6,2.9	5,4,4	0.5	0.3	14.9	61.2	913
4Mb	3.7	LB	3.8,3.8,3.8	4.9,3.6,2.6	1.2	0.6	13.6	76	1033
		H1	3.8,3.4,5.7	5,4,2	1.2	0.6	13.8	78	1068
		H2	3.8,3.8,3.8	5,4,2	2.6	0.6	13.8	99	1369
		OPT	3.7,3.7,4.8	5,4,2	1.7	0.5	13.7	83	1134

subchains as $C_2 + bg^*Cx_1$. Again using (12) and (13) for the predecode and global word driver chains with this output load yields the expressions for m and n as

$$f^n = \frac{bg \cdot lg \cdot Cw_2}{Cw_1} \cdot \frac{\left(1+\sqrt{1+\frac{2f}{f_{w2}}}\right)}{\left(\sqrt{\frac{2f}{f_{w1}}}+1-1\right)} \tag{15}$$

$$f^m = \frac{bplpCw_1}{2fCp_0} \cdot \left(1+\sqrt{1+\frac{2f}{f_{w1}}}\right). \tag{16}$$

Here f_{w1} is the normalized delay of the predecode wire.

As before the values of m, n, and k will not in general be integers, but can be used to calculate the lower bound (LB) on the delay. Analogous to the two-level case, we will define two additional parameters, u, the ratio of input gate cap for local word driver to the global word wire cap, and v, the ratio of input gate cap of the global word driver to the input predecoder wire cap. Sizing heuristics H1 and H2 can be extended to the three-level case. In the case of H1, we keep the ratios u and v the same as in the lower-bound computation. This fixes the input sizes of the global and word drivers and the three subchains can be sized independently as simple buffer chains. For heuristic H2, we round n, k obtained from (14) and (15) to even integers and use $f_2 = f3 = f \sim 4$. We also do an exhaustive search with integer number of stages in the three subchains to obtain the optimal solution (OPT). The results for a hypothetical 1-Mb and 4-Mb SRAM in 0.25-μm CMOS process for two different wire widths are tabulated in Table II. We observe that the lower bound is quite tight and is within a percent of the optimal solution. Unlike in the two-level case, here heuristic H1 gives better results than H2. H1 is within 2% of the optimum while H2 is within 8% of the optimum. H2 also consumes more power in general and again this can be correlated with the higher ratios for the input gate capacitance of the word drivers to the wire capacitance. Increasing wire widths to reduce wire resistance not only decreases the delay but also gets the two heuristics closer to the optimum.

Minimum delay solutions typically burn a lot of power since getting the last bit of incremental improvement in delay requires significant power overhead. We will next look at sizing to reduce power at the cost of a modest increase in delay.

C. Sizing for Fast Low-Power Operation

The main component of power loss in a decoder is the dynamic power lost in switching the large interconnect capacitances in the predecode, block select, and wordlines, as well as the gate and junction capacitances in the logic gates of the decode chain. Table III provides a breakdown of the relative contribution from the different components to the total switching capacitance for two different SRAM sizes. The total switching capacitance is the sum of the interconnect capacitances, the transistor capacitances internal to the predecoders, the gate capacitance of the input gate of the global word drivers, the transistor capacitances internal to the global word drivers, the gate capacitance of the input gate of the local word drivers, and the transistor capacitances internal to the local word driver.

TABLE III
RELATIVE ENERGY OF VARIOUS COMPONENTS OF THE DECODE PATH IN %

Size	Wire	Predecode Internal	Input of Global Word Driver	Global Word Driver Internal	Input of Local Word Driver	Local Word Driver Internal
16kb	25	24	16	13	16	6
1Mb	15	29	29	12	14	1

TABLE IV
RELATIVE DELAY OF VARIOUS COMPONENTS OF THE DECODE PATH UNDER H1 IN %

Size	Predecode	Predecode Wire	Global Word Driver	Global Word Line	Local Word Driver
16kb	56	2	26	1	15
1Mb	44	15.5	17.5	5	18

Table IV shows the relative breakdown of the total delay between the predecoder, the predecode wire, the global word driver, the global wordline, and the local word driver. The two key features to note from these tables are that the input gate capacitance of the two word drivers contribute a significant fraction to the total switching capacitance due to the large branching efforts, and that the delays of the two word drivers contribute a significant fraction to the total delay. In fact, the input gate capacitance of the two word drivers are responsible for more of the decoder power than is shown in the table, as they also impact the sizing of the preceding stages. For example, in the case of the 1-Mb SRAM, by breaking down the power dissipation in the predecoders into two components, one directly dependent on the word driver sizes and the other independent on the word driver sizes, we find that 50% of the decoder power is directly proportional to the word driver input sizes. This suggests a simple heuristic to achieve a fast low power operation will be to reduce the input sizes of the two word drivers but still size each chain for max speed. A convenient way to do this is via the parameters u and v, which represent the ratio of the input gate cap to the input wire cap. Table V shows the delay, energy, and energy–delay product for a 1-Mb RAM decoder starting from the sizing of heuristic H1 in Row 2 of Table II and gradually reducing the ratios u and v.The last entry with $u = 0.2$ and $v = 0.04$ corresponds to minimum gate sizes for the inputs of the global and local word drivers. We observe that reducing u and v leads to significant power reductions while the delay only increases modestly. In the last row, the input gate cap of the word drivers is made almost insignificant and we find that the energy reduces by nearly 50% in agreement with the finding that 50% of the decoder power under H1 is directly attributable to these sizes. The delay in the last row only increases by two gate delays (16%) when compared to H1 and can be accounted as follows. Reduction of input local word driver size by a factor of 25 $(v = 0.04)$ leads to an increase of about 2.5 gate delays in the local word driver delay. The reduction of input global word driver size by 10 $(u = 0.2)$ along with the above reduction in v, leads to an increase of one gate delay in the global word driver, while the predecode delay reduces by 0.5 gate delays.

TABLE V
DELAY AND ENERGY FOR A 1-MB SRAM DECODER FOR DIFFERENT RATIOS OF WORD DRIVER INPUT GATE CAP TO INPUT WIRE CAP

u	v	delay (τ_{fo4})	energy (arb. units)	delay * energy
1.9	1.0	12.0	46.4	559
1.0	1.0	12.1	40.9	495
1.0	0.2	12.6	33.5	421
0.2	0.2	13.1	27.7	363
0.2	0.04	14.1	24.7	348

Also because of the reduced capacitance, the wire *RC* delay decreases by about one gate delay leading to only a two gate delay increase in the total delay. The reduction in the energy delay product with reducing u and v indicates that there is a large range for efficient tradeoff between delay and energy by the simple mechanism of varying the sizes of the word driver inputs.

III. DECODER CIRCUITS

The total logical effort of the decode path is directly affected by the circuits used to construct the individual gates of the path. This effort can be reduced in two complementary ways: by skewing the FET sizes in the gates and by using circuit styles which implement the n-input logical AND function with the least logical effort. We first describe techniques to implement skewed gates in a power efficient way. We will then discuss methods of implementing an n-input AND function efficiently, and finally do a case study of a pulsed 4-to-16 predecoder.

A. Reducing Logical Effort by Skewing the Gates

Since the wordline selection requires each gate in the critical path to propagate an edge in a single direction, the FET sizes in the gate can be skewed to speed up this transition. By reducing the sizes for the FETs which control the opposite transition, the capacitance of the inputs and hence the logical effort for the gate is reduced, thus speeding up the decode path. The cost is that separate reset devices are needed to reset the output to prevent the slow reset transition from limiting the memory performance. These reset devices are activated using one of three techniques: precharge logic uses an external clock, self-resetting logic (SRCMOS) [6], [11] uses the output to reset the gate, and delayed reset logic (DRCMOS) [7], [12], [13] uses a delayed version of one of the inputs to conditionally reset the gate.

Precharge logic is the simplest to implement, but is very power inefficient for decoders since the precharge clock is fed to all the gates. Since in any cycle only a small percentage of these gates are activated for the decode, the power used to clock the reset transistors in all the decode gates can be larger than the power to change the outputs of the few gates that actually switch. SRCMOS and DRCMOS logic avoid this problem by activating the reset devices only for the gates which are active. In both these approaches, a sequence of gates, usually all in the same level of the decode hierarchy, share a reset chain. In the SRCMOS approach, the output of this gate sequence triggers

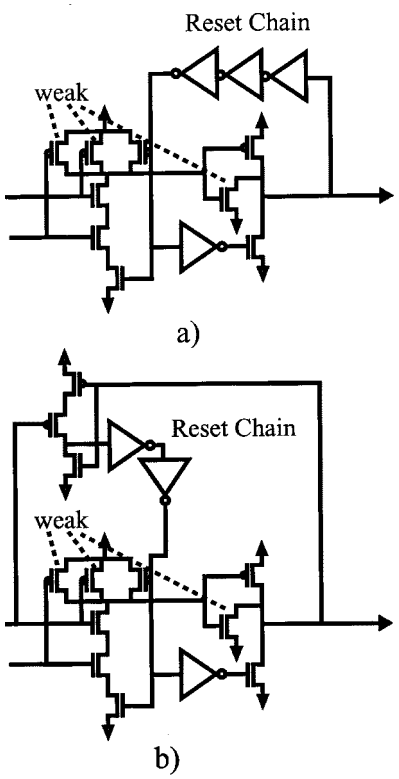

Fig. 4. SRCMOS resetting technique. (a) Self-reset. (b) Predicated self-reset.

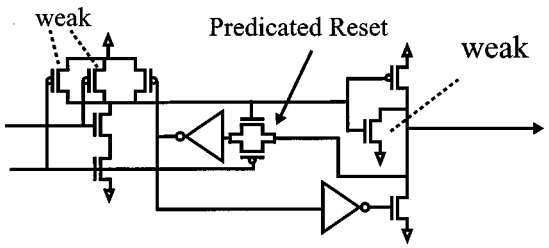

Fig. 5. A DRCMOS technique to do local self-resetting of a skewed gate.

the reset chain, which then activates the reset transistors in all the gates to eventually reset the output (Fig. 4). The output pulsewidth is determined by the delay through this reset chain. If the delay of the reset chain cannot be guaranteed to be longer than the input pulsewidths, then an extra series FET in the input is required to disconnect the pulldown stack during the reset phase, which will increase the logical effort of the gate. Once the output is reset, it travels back again through the reset chain to turn off the reset gates and get the gate ready for the next inputs. Hence, if the input pulsewidths are longer than twice the delay of going around the reset chain, special care must be taken to ensure that the gate does not activate more than once. This is achieved by predicating the reset chain the second time around with the falling input [Fig. 4(b)]. (Another approach is shown in [11].)

The DRCMOS gate fixes the problem of needing an extra series nFET in the input gate by predicating the reset chain activation with the falling input even for propagating the signal the first time around the loop (Fig. 5). (Another version is shown in [13].) Hence, the DRCMOS techniques will have the least logical effort and hence the lowest delay. The main problem with this approach is that the output pulsewidth will be larger than the input pulsewidth so only a limited number of successive levels of the decode path can use this technique before the pulsewidths will exceed the cycle time.

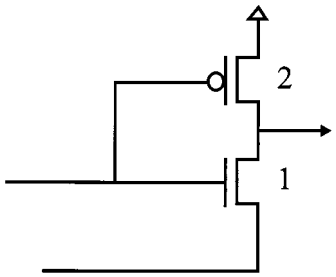

Fig. 6. Source-coupled NAND gate for a pulsed design.

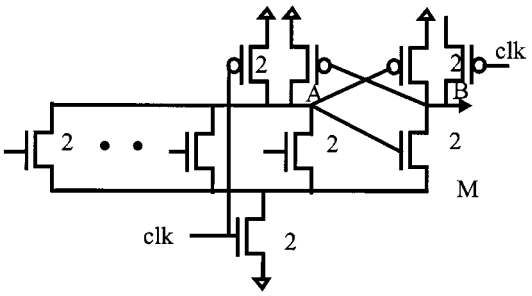

Fig. 7. NOR style decoder [7].

B. Performing an n-input AND Function With Minimum Logical Effort

The n-input AND function can be implemented via different combination of NANDs, NORs, and inverters. Since in current CMOS technologies, a pFET is at least two times slower than an nFET, a conventional NOR gate with series pFET is very inefficient and so the AND function is usually best achieved by a combination of NANDs and inverters. If we use k-input NAND gates with a logical effort of $le(k)$, then we will need $\log_k n$ levels to make the n-input NAND function, resulting in a total logical effort shown in (17).

$$\text{total effort} = le(k)^{\log_k n}. \tag{17}$$

For a conventional static style NAND gate with long channel devices, the logical effort for a k-input NAND gate is $(k+2)/3$. Using this in (17) and solving for different k, we find that the total logical effort for an n-input NAND function is minimized for $k = 2$. At the other extreme, if we use completely skewed NAND gates with short channel devices, the logical effort can be approximated by $\mathrm{sqrt}(k)/3$. Again $k = 2$ minimizes the total logical effort. Hence building the decoder out of two-input NAND gates leads to the lowest delay. An added benefit is that with two-input NAND gates, the least number of predecode capacitance is switched thus minimizing power dissipation. When the two-input NAND gate is implemented in the source-coupled style [15], [16], its logical effort approaches that of the inverter, if the output load is sufficiently small compared to the load at the source input (Fig. 6). This is true for the input stage of the word drivers.

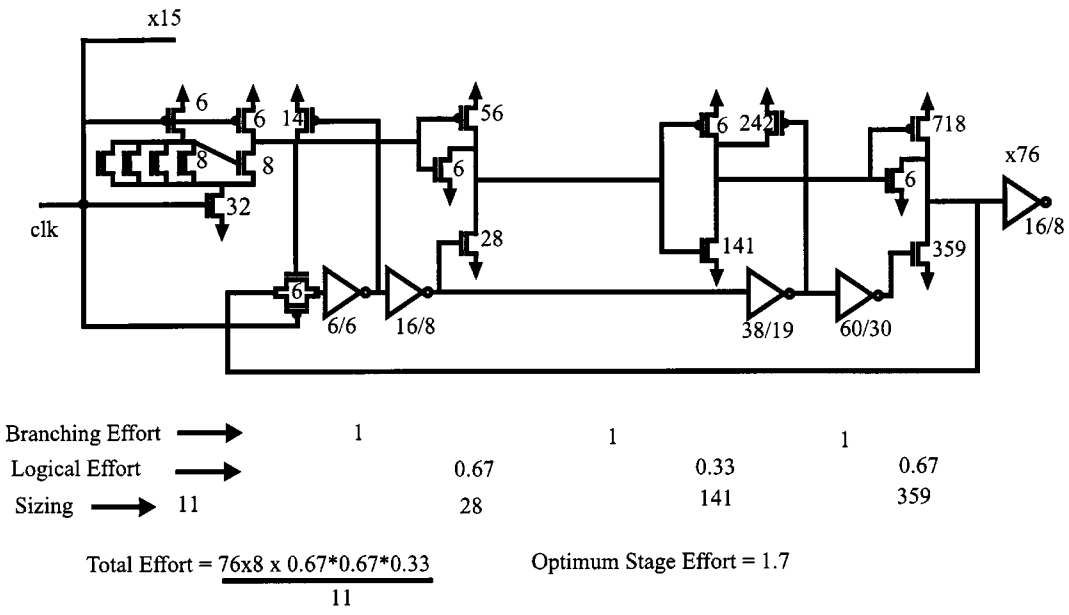

Fig. 8. NOR style 4-to-16 predecoder with maximal skewing and DRCMOS resetting.

Since a wide fan-in NOR can be implemented with very small logical effort in the domino circuit style, a large fan-in NAND can be implemented doing a NOR of the complementary inputs (Fig. 7), and is a candidate for building high-speed predecoders. The rationale for this approach is that with increasing number of inputs, nFETs are added in parallel, thus keeping the logical effort a constant, unlike in a NAND gate. To implement the NAND functionality with NOR gates, Nambu *et al.* in [7] have proposed a circuit technique to isolate the output node of an unselected gate from discharging. This is reproduced in the figure. An extra nFET (M) on the output node B shares the same source as the input nFETs, but its gate is connected to the output of the NOR gate (A). When clock (clk) is low, both nodes A and B are precharged high. When clock goes high, the behavior of the gate depends on the input values. If all the inputs are low, then node A remains high, while node B discharges and the decoder output is selected. If any of the inputs are high, then node A discharges, shutting off M and preventing node B from discharging. This causes the unselected output to remain high. This situation involves a race between A and B and is fixed by using two small cross-coupled pFETs connected to A and B.

We will quantify the impact of skewing and circuit style on delay and power in the next section for a 4-to-16 predecoder.

C. Case Study of a 4-to-16 Predecoder

Let us consider the design of a 4-to-16 predecoder which needs to drive a load which is equivalent to 76 inverters of size 8. This load is typical when the predecode line spans 256 rows. We compare designs in both the series stack style and the NOR style, and for each consider both the nonskewed as well as the skewed versions. To have a fair comparison between the designs, we will size the input stage in each such that the total input loading on any of the address inputs is the same across the designs. Due to space constraints, we will only describe in detail the skewed design with NOR style gate, but report the results for the other designs. The details for the other designs can be found in [17].

TABLE VI
DELAY AND POWER COMPARISONS OF VARIOUS CIRCUIT STYLES IN 0.25-μm PROCESS AT 2.5 V. DELAY OF A FAN-OUT 4 LOADED INVERTER IS 90 PS

Circuit Style	Delay (pS)/ number of fanout 4 loaded inverters	Power (mW)
2-input NAND without skewing	316 / 3.5	1.1
2-input NAND with skewing	234 / 2.6	1.1
NOR style without skewing	284 / 3.2	1.2
NOR style with skewing (Figure 8)	202 / 2.25	1.3

Fig. 8 shows a predecoder design which uses NOR style gate and combines skewing and local resetting in the DRCMOS style. The total path effort is reduced by a factor of 2.6 compared to a skewed design which uses two-input NAND gates. A summary of delay and power for the four designs is shown in Table VI. This is the fastest design with a delay of 202 ps (2.25 fan-out 4 loaded inverters). It has about 36% lower delay than the slowest design, which is a conventional nonskewed version with two-input NAND gates. We note here that this number is almost the same as reported in [7], but we differ on to what we ascribe the delay gains. From the examples, it is clear that the major cause for delay improvement in this style is gate skewing, which buys almost 26% of the reduction as seen in Table VI. The remaining 10% gain comes from using the NOR front end. Nambu *et al.* have reversed this allocation of gains in their paper [7]. The power dissipation in the above design is kept to about 1.33 mW, because of the DRCMOS reset technique. (We include the power dissipation in the unselected NOR gates, which is not shown in the above figure for sake of clarity.)

From the table, it is apparent that skewing leads to considerable speedup at very minimal power overhead and NOR style predecoder yields the fastest design.

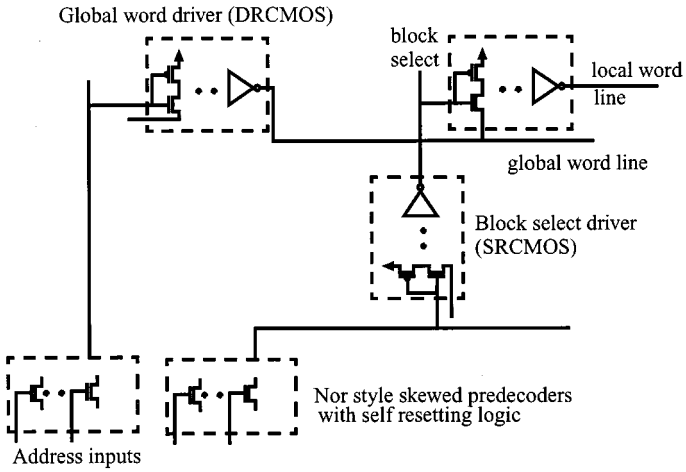

Fig. 9. Schematic of fast low power three-level decoder structure.

D. Optimum Decode Structure

Based on the discussions in Section III-A–C, we can now summarize the optimal decoder structure for fast low-power SRAMs (Fig. 9). Except for the predecoder, all the higher levels of the decode tree should have a fan-in of 2 to minimize the power dissipation, as we want only the smallest number of long decode wires to transition. The two-input NAND function can be implemented in the source-coupled style without any delay penalty, since it does as well as an inverter. This has the further advantage that under low supply voltage operation, the voltage swings on the input wires can be reduced by half and still preserve speed while significantly reducing the power to drive these lines [20], [21]. The local word driver will have two stages in most cases, and have four when the block widths are very large. In the latter case, unless the applications demand it, it will be better to repartition the block to be less wide in the interests of the wordline *RC* delay and bitline power dissipation. Skewing the local word drivers for speed is very expensive in terms of area due to the large numbers of these circuits. Bitline power can be controlled by controlling the wordline pulsewidth, which is easily achieved by controlling the block select pulsewidth. Hence, the block select signal should be connected to the gate of the input NAND gate and the global word driver should be connected to the source. Both the block select and the global wordline drivers should have skewed gates for maximum speed, and will have anywhere from two to four stages depending on the size of the memory. The block select driver should be implemented in the SRCMOS style to allow for its output pulsewidth to be controlled independently of the input pulsewidths. The global word driver should be made in the DRCMOS style to allow for generating a wide enough pulsewidth in the global wordline to allow for sufficient margin of overlap with the block select signal. Since in large SRAMs the global wordline spans multiple pitches, all the resetting circuitry can be laid out local to each driver. In cases where this is not possible, the reset circuitry can be pulled out and shared amongst a small group of drivers [7]. Predecoder performance can be significantly improved at no cost in power by skewing the gates and using local resetting techniques. The highest performance predecoders will have a NOR style wide fan-in input stage followed by skewed buffers. When this is coupled with a technique such as that presented in [7] to do a selective discharge of the output, the power dissipation is very reasonable compared to the speed gains that can be achieved. With the NOR style predecoder the total path effort becomes independent of the exact partitioning of the decode tree, which will allow the SRAM designer to choose the best memory organization based on other considerations.

IV. Summary

We found that the optimum fan-out for the decoder chain with *RC* interconnect is about 4, just as in the case of a simple buffer chain. As in the simple buffer chain, supporting a fan-out of 4 often requires a noninteger number of stages in the chain. Nevertheless, this result can be used to arrive at a tight lower bound on the delay of a decoder. We examined two simple heuristics for sizing of a real decoder with integer stages. In one, the number of stages in the various subchains are rounded values based on the formulae for the lower-bound computation. The fan-outs in the word driver chains are then kept around 4. This heuristic does well for small RAMs with two-level decoders. In the second heuristic, the input sizes of the word drivers are kept the same as in the lower-bound computation. This heuristic does well for larger blocks and three-level decoders. Reducing wire delay by wire sizing brings the delays of both the heuristics within a few percent of the optimum. High-speed designs burn a lot of power. We show that varying the sizes of the inputs of the word drivers, while sizing each of the subchains for maximum speed, provides for a simple mechanism to efficiently trade off speed and power.

We examined a number of circuit styles for implementing the AND function of the decoder. We found that a decoder hierarchy with a fan-in of 2 provides the optimal solution both in terms of speed and power. A detailed analysis of pulse mode gates shows that they are the most energy efficient. Finally, we put together all the results from our analysis and sketch out the optimal decoder structure for fast low-power RAMs.

References

[1] R. C. Jaeger, "Comments on "An optimized output stage for MOS integrated circuits"," *IEEE J. Solid State Circuits*, vol. SC-10, pp. 185–186, June 1975.

[2] C. Mead and L. Conway, *Introduction to VLSI Systems*. Reading, MA: Addison-Wesley, 1980.

[3] N. C. Li *et al.*, "CMOS tapered buffer," *IEEE J. Solid State Circuits*, vol. 25, pp. 1005–1008, Aug. 1990.

[4] J. Choi *et al.*, "Design of CMOS tapered buffer for minimum power-delay product," *IEEE J. Solid State Circuits*, vol. 29, pp. 1142–1145, Sept. 1994.

[5] B. S. Cherkauer and E. G. Friedman, "A unified design methodology for CMOS tapered buffers," *IEEE J. Solid State Circuits*, vol. 3, pp. 99–111, Mar. 1995.

[6] T. Chappell *et al.*, "A 2-ns cycle, 3.8-ns access 512-Kb CMOS ECL SRAM with a fully pipelined architecture," *IEEE J. Solid State Circuits*, vol. 26, pp. 1577–1585, Nov. 1991.

[7] H. Nambu *et al.*, "A 1.8 ns access, 550 MHz 4.5 Mb CMOS SRAM," in *1998 IEEE Int. Solid State Circuits Conf., Dig. Tech. Papers*, pp. 360–361.

[8] G. Braceras *et al.*, "A 350-MHz 3.3-V 4-Mb SRAM fabricated in a 0.3-μm CMOS process," in *1997 IEEE Int. Solid State Circuits Conf. Dig. Tech. Papers*, pp. 404–405.

[9] I. E. Sutherland and R. F. Sproull, "Logical effort: Designing for speed on the back of an envelope," *Advanced Res. VLSI*, pp. 1–16, 1991.

[10] HSPICE, Meta-Software, Inc, 1996.

[11] H. C. Park *et al.*, "A 833-Mb/s 2.5-V 4-Mb double-data-rate SRAM," in *1998 IEEE Int. Solid State Circuits Conf. Dig. Tech. Papers*, pp. 356–357.
[12] B. Amrutur and M. Horowitz, "A replica technique for wordline and sense control in low-power SRAMs," *IEEE J. Solid State Circuits*, vol. 33, pp. 1208–1219, Aug. 1998.
[13] R. Heald and J. Holst, "A 6-ns cycle 256-kb cache memory and memory management unit," *IEEE J. Solid State Circuits*, vol. 28, pp. 1078–1083, Nov. 1993.
[14] K. Nakamura *et al.*, "A 500-MHz 4-Mb CMOS pipeline-burst cache SRAM with point-to-point noise reduction coding I/O," in *1997 IEEE Int. Solid State Circuits Conf. Dig. Tech. Papers*, pp. 406–407.
[15] K. Sasaki *et al.*, "A 15-ns 1-Mbit CMOS SRAM," *IEEE J. Solid State Circuits*, vol. 23, pp. 1067–1071, Oct. 1988.
[16] M. Matsumiya *et al.*, "A 15-ns 16-Mb CMOS SRAM with interdigitated bit-line architecture," *IEEE J. Solid State Circuits*, vol. 27, pp. 1497–1502, Nov. 1992.
[17] B. Amrutur, "Fast Low Power SRAMs," Ph.D. dissertation, Computer Systems Laboratory, Stanford University, Stanford, CA, 1999.
[18] O. Minato *et al.*, "2 K $\times$ 8 bit Hi-CMOS static RAMs," *IEEE J. Solid State Circuits*, vol. SC-15, pp. 656–660, Aug. 1980.
[19] I. Sutherland *et al.*, *Logical Effort: Designing fast CMOS circuits*, 1st ed. San Mateo, CA: Morgan Kaufmann, 1999.
[20] T. Mori *et al.*, "A 1-V 0.9-mW at 100-MHz 2-k^* 16-b SRAM utilizing a half-swing pulsed-decoder and write-bus architecture in 0.25-μm dual-Vt CMOS," in *1998 IEEE Int. Solid State Circuits Conf. Dig. Tech. Papers*, pp. 354–355.
[21] K. W. Mori *et al.*, "Low-power SRAM design using half-swing pulse-mode techniques," *IEEE J. Solid State Circuits*, vol. 33, pp. 1659–1671, Nov. 1998.
[22] H. B. Bakoglu and J. D. Meindl, "Optimal interconnects for VLSI," *IEEE Trans. Electron. Devices*, vol. ED-32, pp. 903–909, May 1985.

Bharadwaj S. Amrutur received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology, Mumbai, India, in 1990, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1994 and 1999, respectively.

He is currently a Member of Technical Staff with Agilent Laboratories, Palo Alto, CA, working on high-speed serial interfaces.

Mark A. Horowitz (S'77–M'78–SM'95–F'00) received the B.S. and M.S. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, and the Ph.D. degree from Stanford University, Stanford, CA.

He is Yahoo Founder's Professor of Electrical Engineering and Computer Sciences and Director of the Computer Systems Laboratory at Stanford University. He is well known for his research in integrated circuit design and VLSI systems. His current research includes multiprocessor design, low-power circuits, memory design, and high-speed links. He is also co-founder of Rambus, Inc., Mountain View, CA.

Dr. Horowitz received the Presidential Young Investigator Award and an IBM Faculty Development Award in 1985. In 1993, he was awarded Best Paper at the International Solid State Circuits Conference.