COST AND PERFORMANCE OF VLSI COMPUTING STRUCTURES

by

Carver A. Mead and Martin Rem

Technical Report #1584
(Ten Figures)

April 11, 1978

Computer Science

California Institute of Technology

Pasadena, California 91125

Silicon Structures Project

sponsored by

Burroughs Corporation, Digital Equipment Corporation,

Hewlett-Packard Company, Honeywell Incorporated,

International Business Machines Corporation,

Intel Corporation, Xerox Corporation,

and the National Science Foundation

Cost and Performance of VLSI Computing Structures

Carver Mead and Martin Rem
California Institute of Technology
Pasadena, CA 91125

Abstract

Using VLSI technology, it will soon be possible to implement entire computing systems on one monolithic Silicon chip. What will the nature of such systems be? How will they be designed? What will be their cost and performance?

Conducting paths are required for communicating information throughout any integrated system. The length and organization of these communication paths places a lower bound on the area and time required for system operations. Optimal designs can be achieved in only a few of the many alternative structures. Two illustrative systems are analyzed in detail: A RAM based system and an associative system. It is shown that in each case an optimum design is possible, using the area - time product as a cost function.

This work was supported in part by BMD under Contract No. DASG60-77-C-0097, and the Office of Naval Research No. N00014-16-C-0367. California Institute of Technology, Computer Science Department Contribution No. 1584

1. Introduction

The silicon integrated circuit technology is evolving continuously toward smaller elementary devices and denser, more complex functions on each single silicon chip. It appears that new processing and lithographic techniques will make possible the fabrication of chips containing 10^7 or 10^8 individual transistors. One such chip will contain more function than todays largest computers. A large amount of effort has been put into fabrication questions, and much more effort will be required to reach the practical limits of device compactness. However, there is at present essentially no theoretical basis for optimizing the overall organization of systems implemented in this technology.

Conventional complexity theory is inadequate because its measure of cost is the number of steps of a sequential machine. No account is taken of the size of the machine (and hence the time required for each step). Possible concurrency is ignored, thereby ruling out the most important potential contribution of the silicon technology. Traditional switching theory is also inadequate. While it provides a beautiful formalism for describing elementary logic functions, its optimization methods concern themselves with logical operations rather than communication requirements. Even in current integrated circuits, the wires required for communicating information accross the chip account for most of the area and driving these wires accounts for most of the time delay. In very large scale integrated systems, the situation becomes even more extreme. In this paper, we describe a method by which the conceptual organization of a large chip can be analyzed, and a lower bound placed on its size and cycle time before a detailed design is undertaken. The results of this analysis suggest rather general guidelines for the organization of all large integrated systems.

2. Metrics of Space and time

2.1. Physical Properties

Devices used to construct monolithic silicon integrated circuits are universally of the charge controlled type. A charge Q placed on the control electrode (gate, base, etc.) results in a current $I = \frac{Q}{\tau}$ flowing through the device. The transit time τ is the time required for charge carriers to move through the active region of the device.

All times in an integrated system can be formulated as simple multiples of τ. For one transistor to drive another identical to it, a charge Q must flow through its active region, requiring time τ. If the capacitance C_L of the load being driven is K times the gate capacitance C_g of the driving transistor, a time $K\tau = \frac{C_L}{C_g}\tau$ is required.

2.2. Linear versus hierarchical structures

In large integrated systems it is necessary to communicate information throughout the entire system. As an example, a bit of information stored on the gate of a minimum size transistor in a random access memory must be communicated to the memory bus of a CPU. Since there are many words of data in the memory, there are many possible sources for each wire in the memory bus. Fig. 1 illustrates two possible approaches to organizing such a bus. In the first approach, a transistor associated with each bit drives the bus wire directly. If the bus wire has a capacitance C_w, the time required to drive the bus wire is $t = \tau \frac{C_w}{C_g}$. In a typical computer memory C_w is many orders of magnitude larger than C_g, and the delay introduced by such a scheme is very long. Since C_w is proportional to the length of the wire, it is also proportional to S, the number of driver transistors connected to the wire.

$$t = \tau S \tag{1}$$

A second scheme is shown in Fig. 1b. Here each transistor drives a wire only long enough to reach its neighbor. Each such wire is connected to the gate of a transistor twice as large as the transistor driving it. The arrangement is repeated upward until the top level where all sources have a path to the bus. In this scheme the delay in driving the lowest level wire is 2τ (assuming the primary capacitance is due to the gate of the larger transistor). The delay introduced by the wires at each level is the same, since each driver transistor is twice as large as those driving it. Hence the delay in driving the bus line is $2\tau N$ where N is the number of levels in the structure. Since there are $S = 2^N$ transistors at the lowest level, the delay may be written

$$t = 2\tau \log_2 S \qquad (2)$$

Comparing (2) and (1), we see that for large S the delay has been made much shorter by using a hierarchial structure.

2.3. A cost criterion

A hierarchy such as that shown in Fig. 1b may be built using any integral number α of transistors driving each wire. The driver transistors will in general be α times the size of those driving them. The delay for such a structure is $t = \alpha\tau \log_\alpha S = \tau \frac{\alpha}{\log\alpha} \log S$. All system delays are thus proportional to $\tau \log S$, with a penalty factor $\frac{\alpha}{\log\alpha}$ dependent upon the branching ratio of the hierarchy. This delay is plotted in Fig. 2, normalized to its minimum value which is attained at α=e.

While dramatic improvements in the performance of integrated structures can be achieved by a hierachical organization, a penalty is always paid in the area required for wires. In the simple case shown, a bus requiring one wire when driven directly requires $\log_\alpha S$ wires when organized as a hierarchy. For this reason it is not possible to optimize a design without a cost function involving both area and time. In this paper we will use the area-time product as our basic cost function. For the above simple example, the cost function is area $*$ time $= \tau(\log S)^2 \dfrac{\alpha}{(\log\alpha)^2}$. The cost is minimized for $\alpha = e^2 \approx 7.4$.

2.4. Hierarchical computing systems

The analysis given above suggests a very general structure for computing systems. Lowest level cells are grouped together into modules in such a way that α cells drive their outputs onto an output wire. Each output wire is connected to a driver transistor which is α times as large as those driving the wire. Modules are grouped in such a way that α of those modules drivers are connected to an inter-module communication wire. This wire in turn is connected to a driver transistor α^2 times as large as the lowest level transistors. This process is continued until the appropriate size system has been realized.

3. Random Access Memory

In this section we discuss the cost and performance of a random access memory (RAM) of S words of logS bits each. As the unit of length we employ the minimum distance of two conducting paths. For the unit of time we choose the time it takes a basic element to charge a wire of unit length plus another transistor like itself. One unit of time is thus slightly larger than the transit time of a transistor.

3.1. Organization of the RAM

We organize the RAM in a hierarchical fashion. The elements of level 0 are the bits themselves, each bit consisting of two crossing wires: a select wire and a data wire. When the select wire is signalled it puts its contents on the data wire. We group α^2 bits into an $\alpha \times \alpha$ square to form a module of level 1. If the width of an element (a bit) is b_0 the elements have to drive wires of length αb_0. A module on level 1 consists of an array of crossing select and data wires, constituting the α^2 bits of level 0, and some additional logic and wires at the side. We group again α^2 of these modules into a square to form a module of level 2, etc. Fig. 3 shows three levels of the hierarchy for $\alpha=4$.

To study the memory in more detail we look at a module of level i (Fig. 4). We describe how one extracts one of its α^{2i} bits. In order to select one bit of storage $2i\log\alpha$ address wires are required. We run $i\log\alpha$ of them, called the row address wires, vertically along the side of the module and the other $i\log\alpha$, the column address wires, horizontally. Its α^2 submodules are organized into α rows of α submodules each. When the select wire of the module is asserted $\log\alpha$ of the row address wires are used, by the decoder, to select one of the α rows of submodules; the select wire running through that row is asserted. The other $(i-1)\log\alpha$ row address wires are run horizontally into each of the α rows of submodules, where they serve as column address wires for the submodules. Of the $i\log\alpha$ column address wires $(i-1)\log\alpha$ are run vertically into each of the α columns of submodules, where they serve as row addresses. The other $\log\alpha$ address wires are used by the multiplexor to select one of the α data wires coming out of the columns of submodules. The signal on the selected data wire is driven onto the data wire of the module itself.

If we wish to have a memory of S words with N+1 levels (level 0 through N) we choose $N = \frac{\log S}{2\log\alpha}$, or $S = \alpha^{2N}$. This gives a hierarchical structure with S bits from which we can extract one bit at a time. If we want the word length to be logS we employ logS of these structures in parallel: to select one word we select one bit in each of the logS hierarchies.

3.2. Area of the RAM.

Fig. 4 allows us to compute the size of a RAM. Let L_i denote the width of a module of level i, then we have the following recurrence relation:

$$L_0 = b_0$$
$$L_i = i\log\alpha + 1 + \log\alpha + \alpha . L_{i-1}$$

The solution to the above relation is

$$L_i = \alpha^i b_0 + \frac{\alpha^i-1}{\alpha-1} + \left(\frac{2\alpha^{i+1}-\alpha^i-\alpha}{(\alpha-1)^2} - \frac{i+1}{\alpha-1}\right)\log\alpha .$$

Rather than the width itself we are interested in the width per bit. In one direction, horizontal or vertical, module i has α^i bits; we therefore compute $\frac{L_i}{\alpha^i}$.

$$\frac{L_i}{\alpha^i} = b_0 + \frac{1}{\alpha-1} + \frac{2\alpha-1}{(\alpha-1)^2}\log\alpha - \frac{1}{(\alpha-1)\alpha^i}\left[\left(\frac{\alpha}{\alpha-1} + 1+i\right)\log\alpha + 1\right] \qquad (3)$$

An interesting property of the width per bit, as expressed by (3), is that its limit for i → ∞ is finite.

$$\lim_{i\to\infty} \frac{L_i}{\alpha^i} = b_0 + \frac{1}{\alpha-1} + \frac{2\alpha-1}{(\alpha-1)^2} \log\alpha \qquad (4)$$

This means that the width per bit $\frac{L_i}{\alpha^i}$ is bounded from above by (4) independent of the number of levels of a RAM. Expression (3) converges in an exponential fashion towards its limit: for small values of i (3) is already very close to (4). We, therefore, use (4) as the width per bit for a RAM; its square is then the area per bit. By dividing the area per bit by the bit area b_0^2 we obtain the total area per bit area for a RAM. Fig. 5 shows this quotient as a function of α for four different values of b_0. It gives the overhead factor in the area that is due to the wires. For a memory of 64K bits with N=2 α should be 16. Expression (4) is then equal to $b_0 + 0.6$. This shows that in 2-level 64K dynamic MOS memories, for which b_0 lies between 1 and 2, roughly half of the area will be occupied by wires.

One may wonder why we have not discussed the area that is consumed by the wires for power and ground. The reason for this is that these wires can be thought of as increasing only the width b_0 of each bit; they do this by an amount that is roughly independent of α, as is shown in the following analysis.

For simplicity we assume that the wires for power and ground run in opposite directions, say parallel to the data and select wires. We compute how much one of them contributes to the width of a module i. The width of a power or ground wire is proportional to the number of bits served by it.

Let the width at the highest level be u, given S and the design of the lowest level memory cell this parameter is easy to compute. The width of the wire in a module on level i is proportional to the current it must supply and is hence $u \frac{\alpha^{2i}}{\alpha^{2N}}$. In one direction, horizontal or vertical, there are $\frac{\alpha^N}{\alpha^i}$ such modules. The total contribution of all modules on level i is thus $u \frac{\alpha^i}{\alpha^N}$. Taking the sum of this expression for i=0,1,...,N yields $\frac{u}{\alpha^N} \frac{\alpha^{N+1}-1}{\alpha-1} \approx u \frac{\alpha}{\alpha-1}$. There are $\sqrt{S}$ bits in one direction, the increase of the bit width, due to power and ground, is therefore

$$\frac{u}{\sqrt{S}} \frac{\alpha}{\alpha-1},$$

which is roughly equal to $\frac{u}{\sqrt{S}}$.

We are interested in the optimal choice of α, but to make that choice we will have to look at the access time, which also depends on α, as well.

3.3 Access time of the RAM

Each element of level 0 drives a wire of length αb_0 to reach the periphery of its module on level 1, this takes time αb_0. Each module on level 1 drives in the same amount of time a wire that is α times longer to reach the periphery of its module on level 2, etc. With N being the level of the highest module, the time required to extract one bit of storage adds up to αb_0 N. We use this figure as the access time. For a RAM of S words the access time is then $\alpha b_0 \frac{\log S}{2\log\alpha}$.

3.4. The cost of the RAM.

We take the product of the area and the access time as the cost function of the RAM. A RAM of S words of logS bits each has the following area-time product.

$$\left(b_o + \frac{1}{\alpha-1} + -\frac{2\alpha-1}{(\alpha-1)^2}\log\alpha\right)^2 \frac{\alpha b_o}{2\log\alpha} \; S\log^2 S \qquad (5)$$

Fig. 6 shows (5), normalized with respect to $S\log^2 S$, as a function of α for different values of b_o. One notices that for increasing bit sizes the branching ration of the hierarchy should decrease. Static memories should therefore have a smaller α than dynamic ones. For dynamic MOS memories the optimal choice for α lies between 8 and 16, for static MOS memories ($b_o \approx 4$) between 4 and 8. One may speculate that "smart memories", structures in which part of the processing task is distributed over the memory cells, will have small branching ratios and hence relatively deep hierarchies.

4. Content Addressable Memory

The basic elements of the RAM were bits. The content addressable memory (CAM) is an example of a word organized memory. We consider a "pure" CAM. It consists of words of w bits each. We access a word by applying w bits of data to the system. We assume that there is only one word in the memory with that contents, and the address of that word is produced by the memory.

4.1. Organization of the CAM.

The basic elements are the bits each of width b_1. The bits do not constitute the modules of level 0: The modules on level 0 of the hierarchy consist of αw words of w bits each. (See Fig. 7b) The w data bits are run via parallel wires vertically through the module. Out of each word comes one horizontal match wire going to the right. A word asserts its match wire if each data bit received is equal to the corresponding bit stored. There are αw words in a module of level 0, the address of the matching word leaves the module via the $\log \alpha w$ address wires.

The above organization of a module of level 0 has one defect: it would require the individual bits of storage to drive wires of length wb_1, which may be greater than the desired αb_1, to reach the address wires. In section 2 we discussed that this type of communication should be achieved by a hierarchy. We therefore organize the driving of the match wire by the w bits in a word in the same manner as shown in section 2.

Each word is chopped up into $\frac{w}{\alpha}$ subwords of α bits each (Fig. 7a). Each of the $\frac{w}{\alpha}$ subwords sends a signal to a "match tree" which has a branching ratio of α and delivers, via $\log_\alpha w$ levels, the logical product of its inputs. The top node of the match tree can drive a wire of length $b_1\alpha^{\log_\alpha w} = b_1 w$, the length of a word in the memory. Therefore, the word itself can drive a wire of length $b_1\alpha w$ and we may group together αw words into module 0 (Fig. 7b). Notice that the module's length is roughly equal to α times its width. This will be true for modules on higher levels as well.

We now describe a module of level i (Fig. 8). It contains $w\alpha^{4i+1}$ words and consists of α^4 submodules of level i-1, grouped into α^2 rows of α^2 submodules each. Each such row contains, besides the α^2 submodules, w data wires to transport the data to each of the submodules and $\log w\alpha^{4i-1}$ outcoming address wires to transport to the right the address of the matching word. Each submodule has $w\alpha^{4i-3}$ words and hence one row contains $w\alpha^{4i-1}$ words which explains the number of address wires. A module on level i has α^2 of these rows and thus requires $\log w\alpha^{4i+1}$ outcoming address wires; they are placed to the right of the rows.

In the CAM we have α^4 submodules per module, in the RAM only α^2. This is only a seeming difference: in the CAM we have, for simplicity, combined two steps in the hierarchy; we have however maintained our multiplication factor α for the wire lengths. L_{i-1}, the length of a module of level i-1, is roughly equal to α times W_{i-1}, the width of a module of level i-1. Therefore, module i-1 can already drive wires of length αW_{i-1}. As a consequence, we can put α^2 submodules into one row as this would only require the driving of wires of length $\alpha^2 W_{i-1}$ in each row. But then we can, and this is the second step, combine α^2 rows as this would require the driving of wires of length about $\alpha^2 L_{i-1}$, which is roughly equal to $\alpha^3 W_{i-1}$.

4.2. Area of the CAM

We compute the length and the width separately. For the length L_i of a module on level i we have the relation (cf. Figs. 7b and 8)

$$L_o = \alpha w \left(b_1 + \frac{\log w}{\log\alpha}\right)$$

$$L_i = \alpha^2 \; (w + L_{i-1} + \log w\alpha^{4i-1}).$$

The solution to this recurrence relation is

$$L_i = \alpha^{2i+1} w(b_1 + \frac{\log w}{\log\alpha}) + (w + \log w) \; \frac{\alpha^{2i+2}-\alpha^2}{\alpha^2-1}$$

$$+ \left(\frac{4\alpha^{2i+2}-4\alpha^2}{(\alpha^2-1)^2} \; + \; \frac{3\alpha^{2i+2}-4i\alpha^2-3\alpha^2}{\alpha^2-1}\right)\log\alpha$$

A module on level i has $w\alpha^{2i+1}$ bits in the vertical direction. The length per bit is therefore $\frac{L_i}{w\alpha^{2i+1}}$. This has the following limit for $i \to \infty$

$$b_1 + \frac{\log w}{\log\alpha} \; + \; \frac{\alpha(w+\log w+3\log\alpha)}{w(\alpha^2-1)} \; + \frac{4\alpha\log\alpha}{w(\alpha^2-1)^2} \; . \qquad (6)$$

As in the case of the RAM, $\frac{L_i}{w\alpha^{2i+1}}$ is already very close to the limit for small values of i; the rate of convergence is again exponential. We use (6) as the length per bit of a CAM.

We find for the width W_i of a module on level i the following recurrence relation. (cf. Figs. 7b and 8)

$$W_o = \frac{w}{\alpha} \; (\alpha b_1+1) + \log\alpha w$$

$$W_i = \alpha^2 W_{i-1} + \log w\alpha^{4i+1}$$

Its solution is

$$W_i = \alpha^{2i} w \, (b_1 + \frac{1}{\alpha}) + \frac{\alpha^{2i+2}-1}{\alpha^2-1} \log w$$

$$+ \left(\frac{4\alpha^{2i+2}-4\alpha^2}{(\alpha^2-1)^2} + \frac{\alpha^{2i+2}-4i-1}{\alpha^2-1} \right) \log\alpha .$$

In the horizontal direction there are $w\alpha^{2i}$ bits. The width per bit $\frac{W_i}{w\alpha^{2i}}$ has as its limit for $i \to \infty$

$$b_1 + \frac{1}{\alpha} + \frac{\alpha^2 \log\alpha w}{w(\alpha^2-1)} + \frac{4\alpha^2 \log\alpha}{w(\alpha^2-1)^2} \quad . \qquad (7)$$

We take the product of (6) and (7) as the area per bit.

By dividing the area per bit by the bit area b_1^2 we obtain the total area per bit area for a CAM. Fig. 9 shows this quotient for $w = 32$ as a function of α for different values of b_0.

If we compare Figs. 5 and 9 we notice that for small values of α the wires in the CAM cause less overhead in area than those in the RAM. For large values of α it is the RAM that enjoys a smaller overhead in area. For equal bit sizes, i.e. with $b_0 = b_1$, the area overhead factor for the RAM and the CAM are about equal at $\alpha = 8$.

As in the RAM we can compute by how much we should increase the bit width b_1 if we wish to take power and ground into account. We leave it as an exercise for the readers to convince themselves that both power and ground give an increase of $u \frac{\alpha^2}{\alpha^2-1}$ to the length and the width of the CAM. This is even closer to u than in the case of the RAM. If we wish to ammortize this amount over the bits, the bit width b_1 should be incremented by $\frac{2u}{\sqrt{S\ w}} \frac{\alpha^2}{\alpha^2-1}$ for a CAM of S words of w bits each.

4.3 Access time of the CAM

For the access time we take the time required to extract the address of the matching word of data from a memory of S words. With the highest level being level N we have $S = w\alpha^{4N+1}$ or

$$N = \frac{\log S - \log w}{4\log\alpha} - \frac{1}{4}$$

A word of storage has a response time of $\frac{\log w}{\log\alpha}\alpha b_1$; for a module of level 0 this becomes $\left(\frac{\log w}{\log\alpha} + 1\right)\alpha b_1$. Each new level of the hierarchy multiplies the wire lengths by a factor α^2 and hence requires an additional time of $2\alpha b_1$. For N levels we find hence

$$\begin{aligned} \text{access time} &= (2N + \frac{\log w}{\log\alpha} + 1)\alpha b_1 \\ &= (\frac{\log S + \log w}{2\log\alpha} + \frac{1}{2})\alpha b_1 \end{aligned} \qquad (8)$$

4.4. The cost of the CAM.

We take again the product of the area and the access time as the cost function. For a CAM of S words of w bits each formulae (6), (7) and (8) yield the cost function

$$\left(b_1 + \frac{\log w}{\log\alpha} + \frac{\alpha(w + \log w + 3\log\alpha)}{w(\alpha^2-1)} + \frac{4\alpha\log\alpha}{w(\alpha^2-1)^2}\right) *$$

$$\left(b_1 + \frac{1}{\alpha} + \frac{\alpha^2\log\alpha w}{w(\alpha^2-1)} + \frac{4\alpha^2\log\alpha}{w(\alpha^2-1)^2}\right)\left(\frac{\log S + \log w}{2\log\alpha} + \frac{1}{2}\right)\alpha b_1 w\ S.$$

Fig. 10 shows the cost function as a function of α for a CAM of 65K words of 32 bits each. The curves are fairly independent of the choice of w provided we choose w great enough, say $w \geq 16$. A change in S will basically move the curves only up and down, it will not affect the positions of their minima.

We notice again that increasing the bit size will decrease the optimal choice of α. Comparing Figs. 6 and 10 we see that content addressable memories should have smaller branching ratios than random access memories. For $b_1 = 4$, which seems a reasonable figure, the optimal choice of α is 4.

5. Conclusion

We have presented a general method for analyzing the cost and performance of recursively defined VLSI structures. Parameters of any such structure may be optimized with respect to time, area, or some combination of the two. While we have chosen the area-time product, it is clear that some other choice may be appropriate for any given application.

The results of this study indicate that as more processing is available in each module at level zero, the optimal value of α will decrease. A system with $\alpha = 4$ would seem to be appropriate for memories in which substantial processing is comingled with storage.

Very general arguments were used to generate the basic recursive structure. For that reason it appears that a very large fraction of VLSI computing structures will be designed in this way. We have discussed two examples, one in which the basic elements were bits of storage and one with words of storage at the lowest level. They gave rise to rather different recursive structures. The way in which their area and time measures were established should make it clear how to apply these techniques to other recursively defined computing structures.

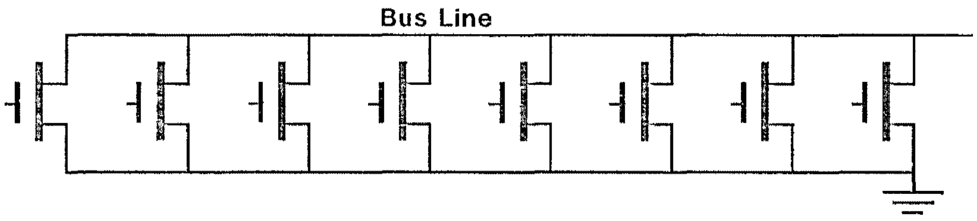

Fig 1a

A bus driven directly by memory cells

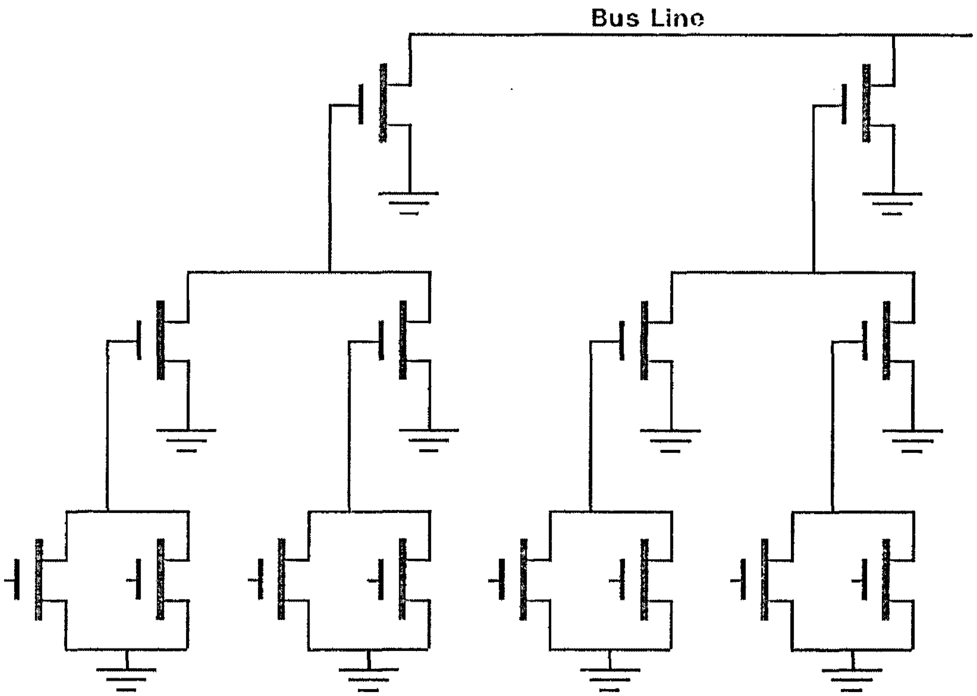

Fig 1b

A bus driver tree

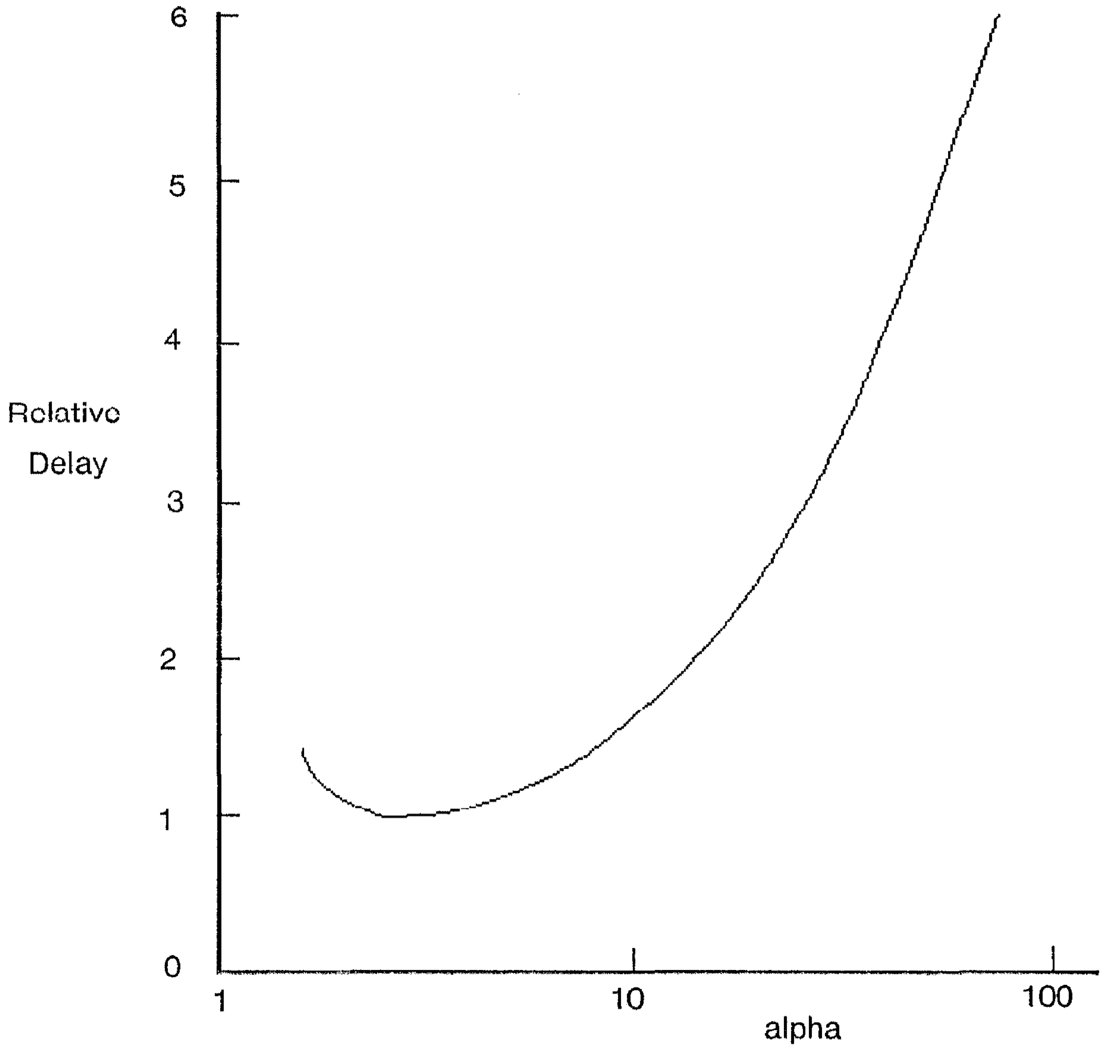

Fig. 2
Delay of a hierarchical structure as a function of alpha

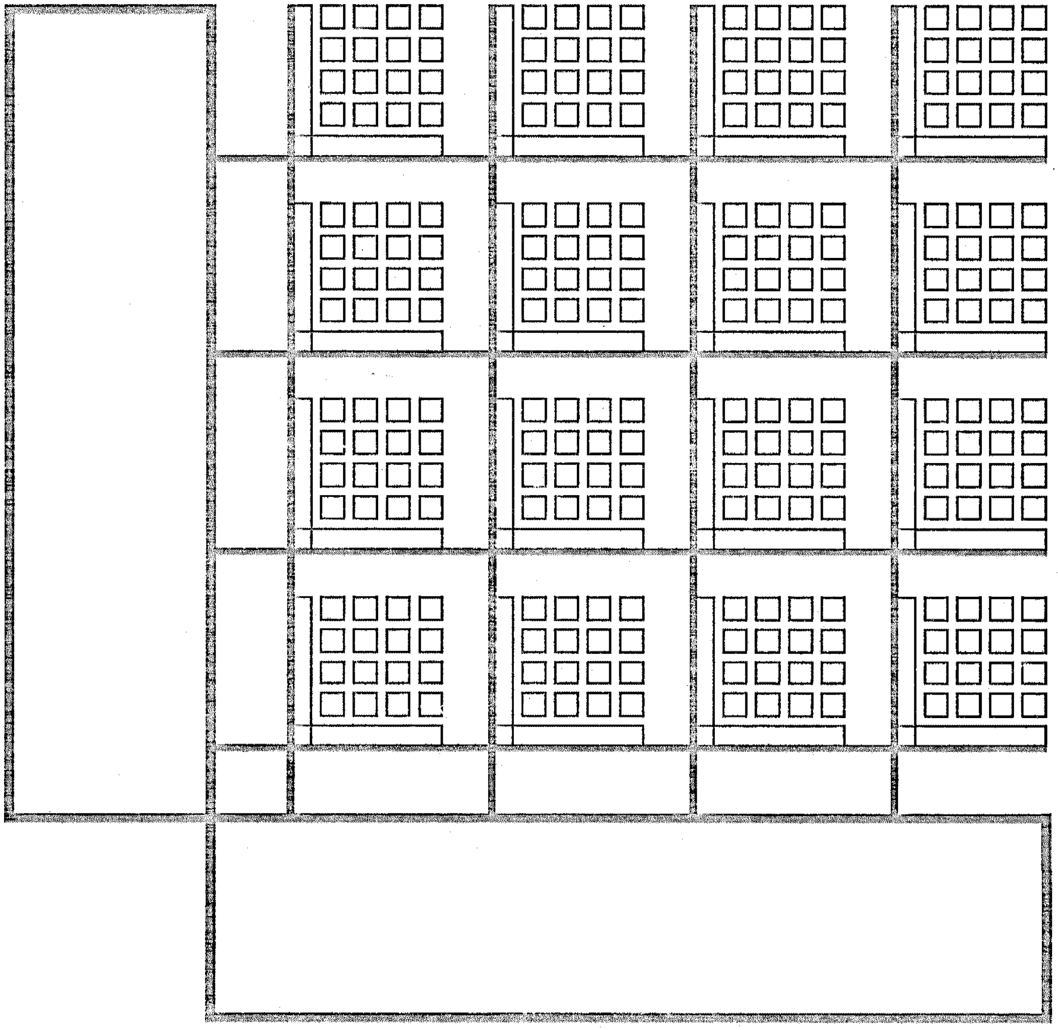

Fig. 3

Three levels of a memory hierarchy for α =4

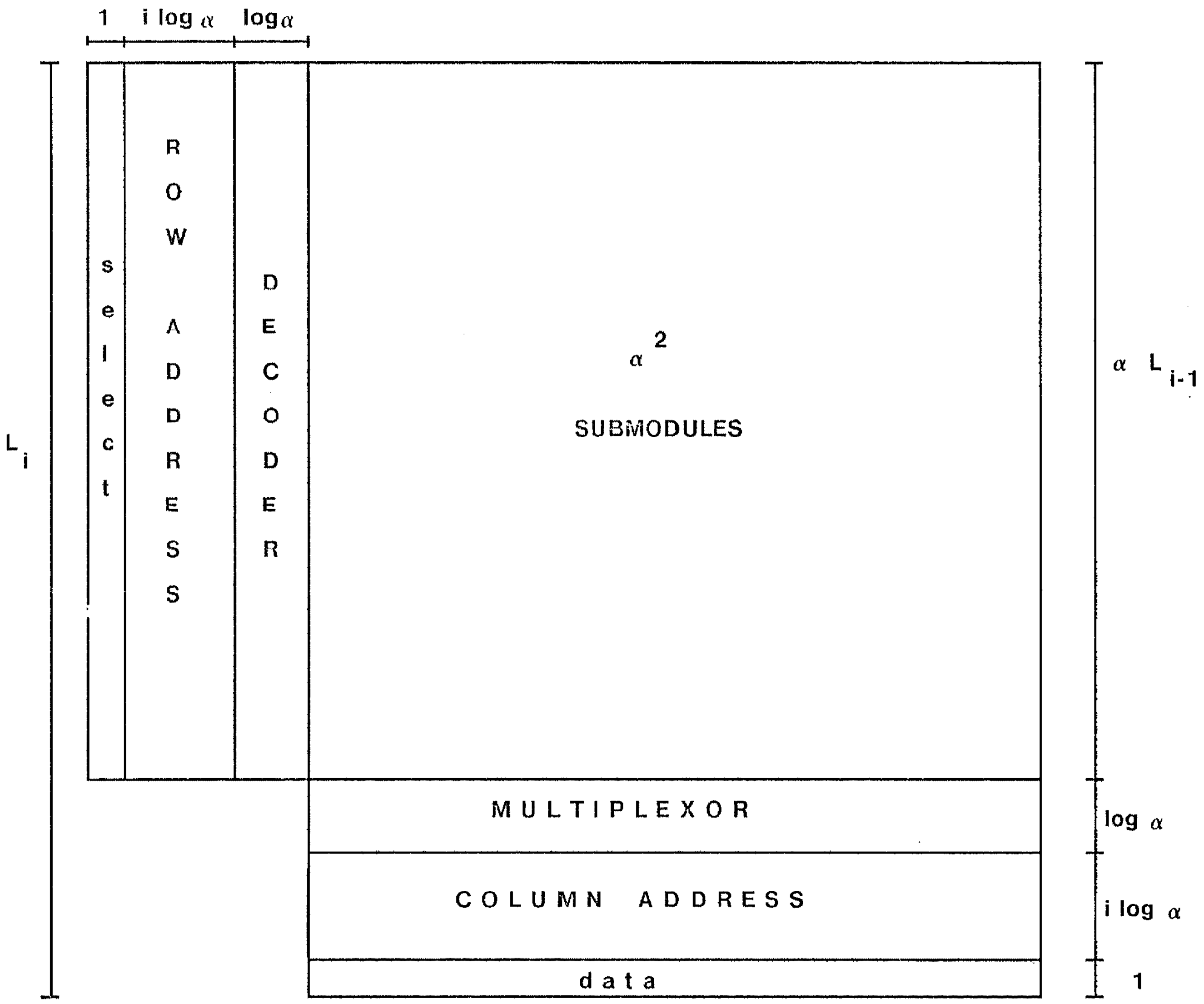

Fig. 4

A RAM module of level i (i > 0)

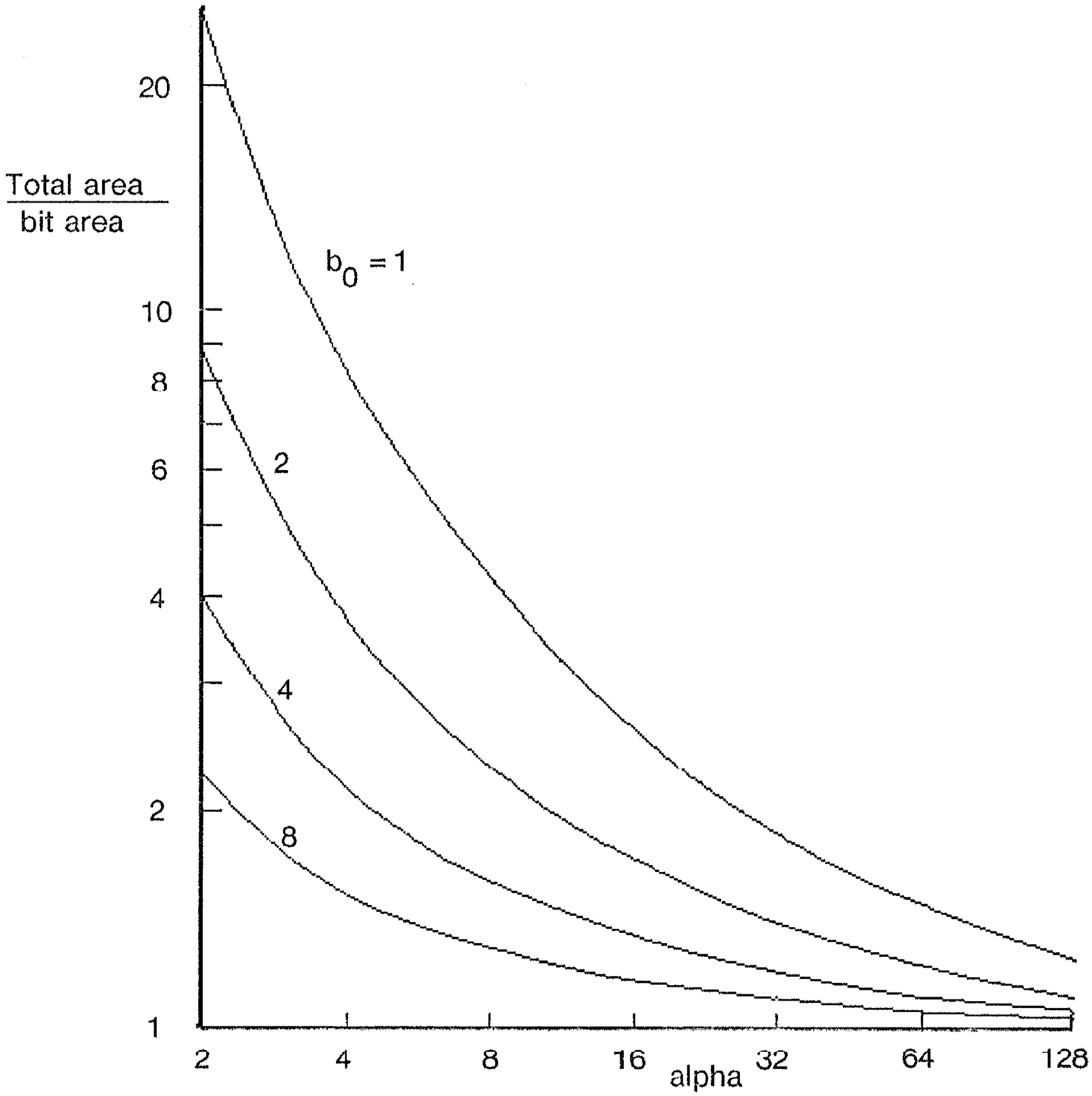

Fig. 5 Total area per bit of a RAM as a function of alpha

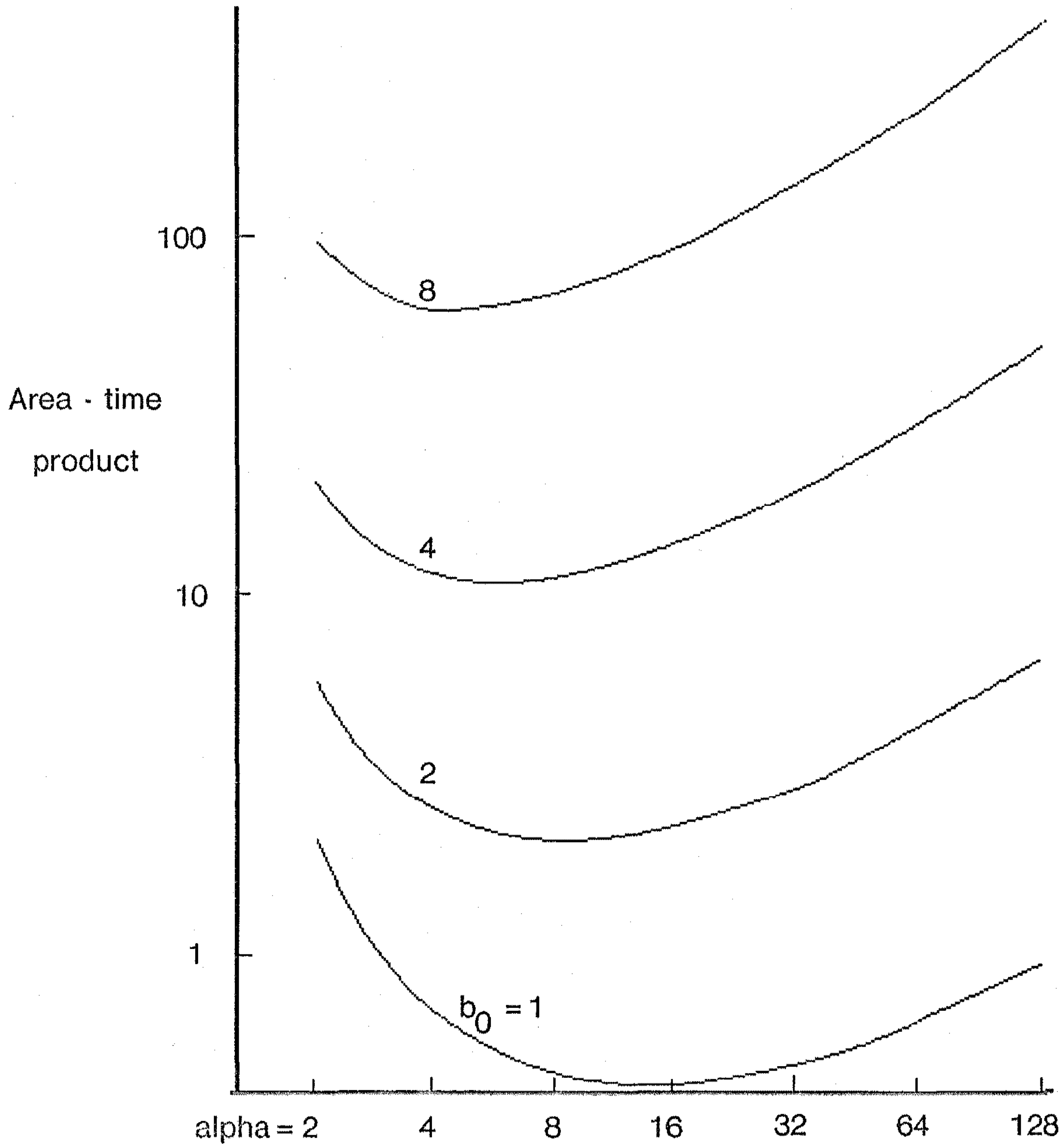

Fig. 6 Area-time product of a RAM as a function of alpha

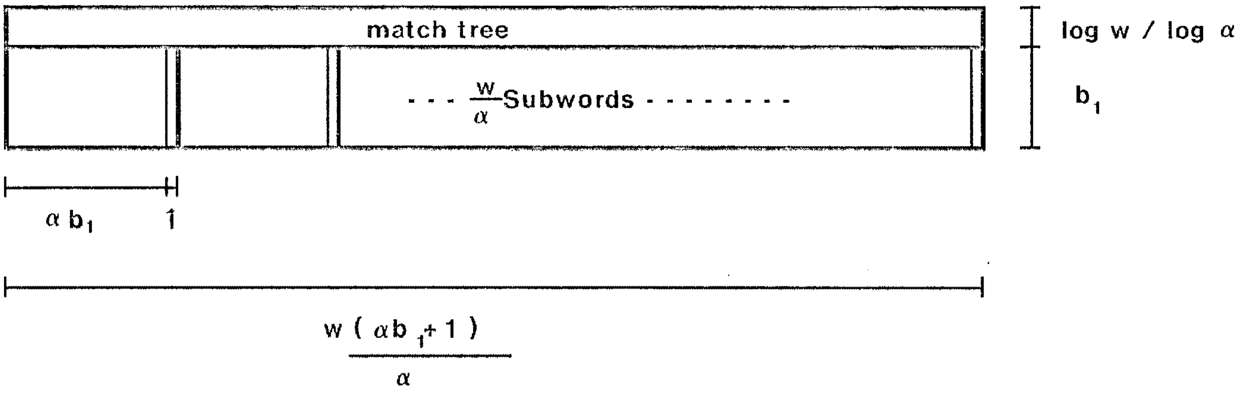
match tree
- - - $\frac{w}{\alpha}$ Subwords - - - - - - - -
log w / log α
b_1
αb_1
1
$\frac{w(\alpha b_1 + 1)}{\alpha}$

Fig. 7a
One word of storage in the CAM

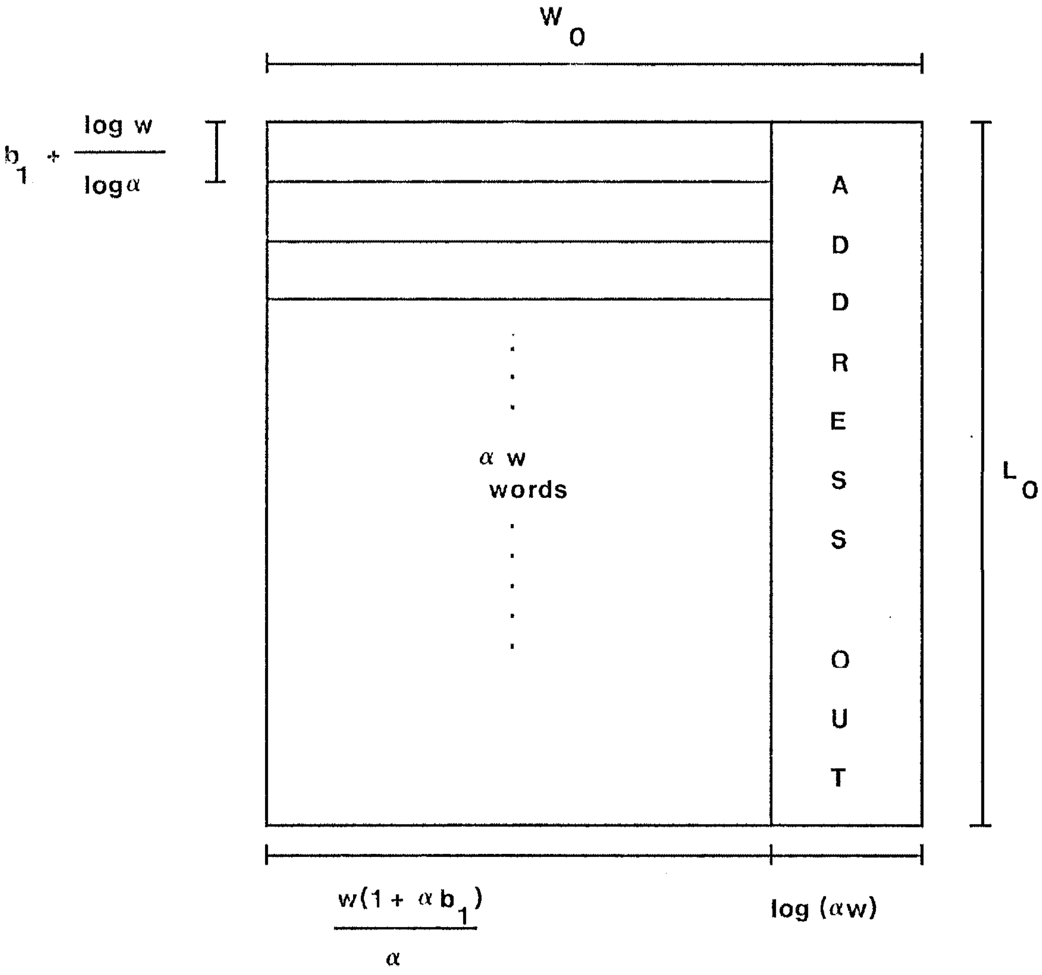
W_0
$b_1 + \frac{\log w}{\log\alpha}$
α w
words
A
D
D
R
E
S
S
O
U
T
L_0
$\frac{w(1 + \alpha b_1)}{\alpha}$
log (αw)

Fig. 7 b
A CAM Module of level zero

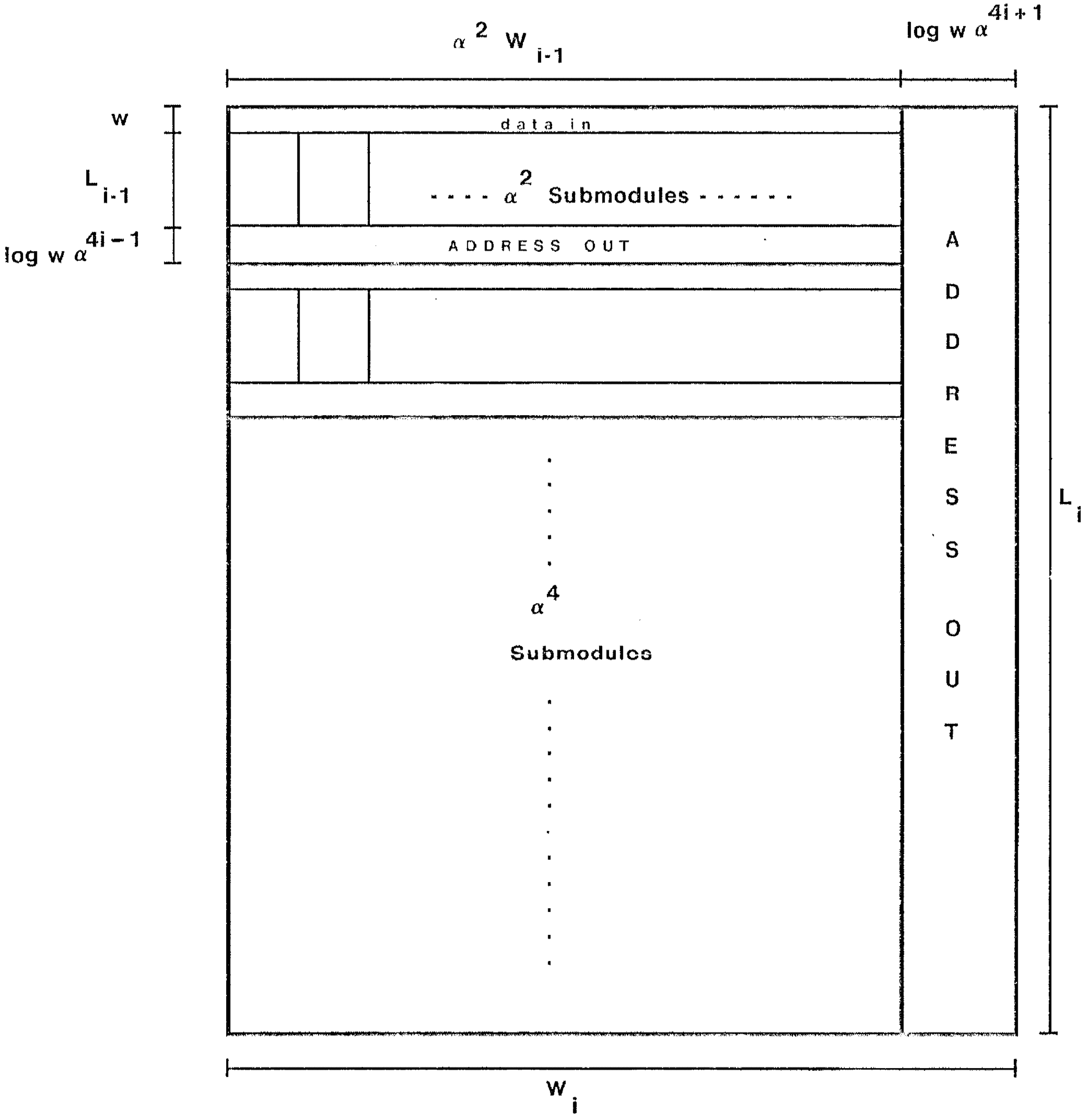

Fig. 8
A CAM Module of level i (i > 0)

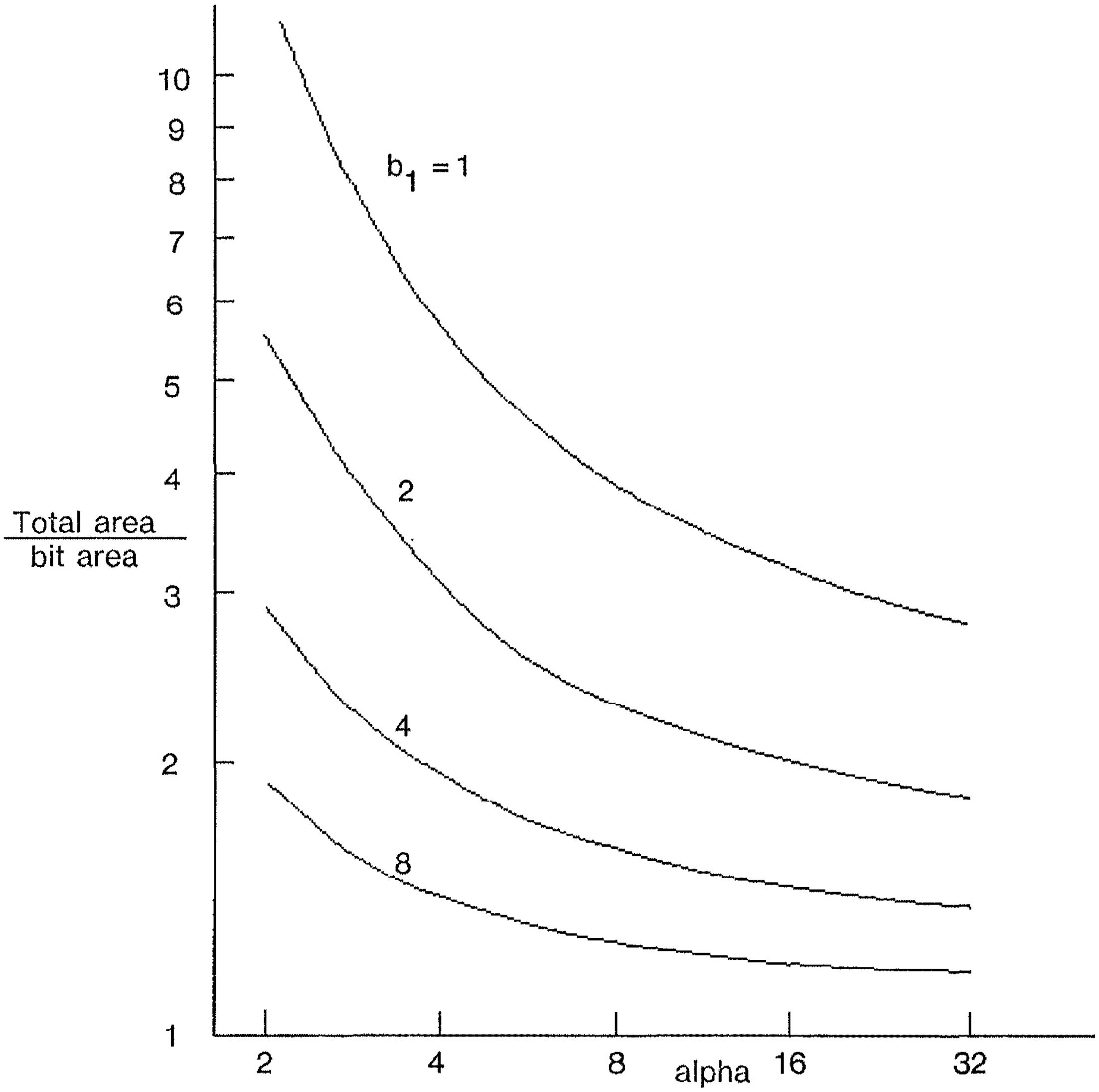

Fig. 9 Total area per bit as a function of alpha for a CAM with word length 32

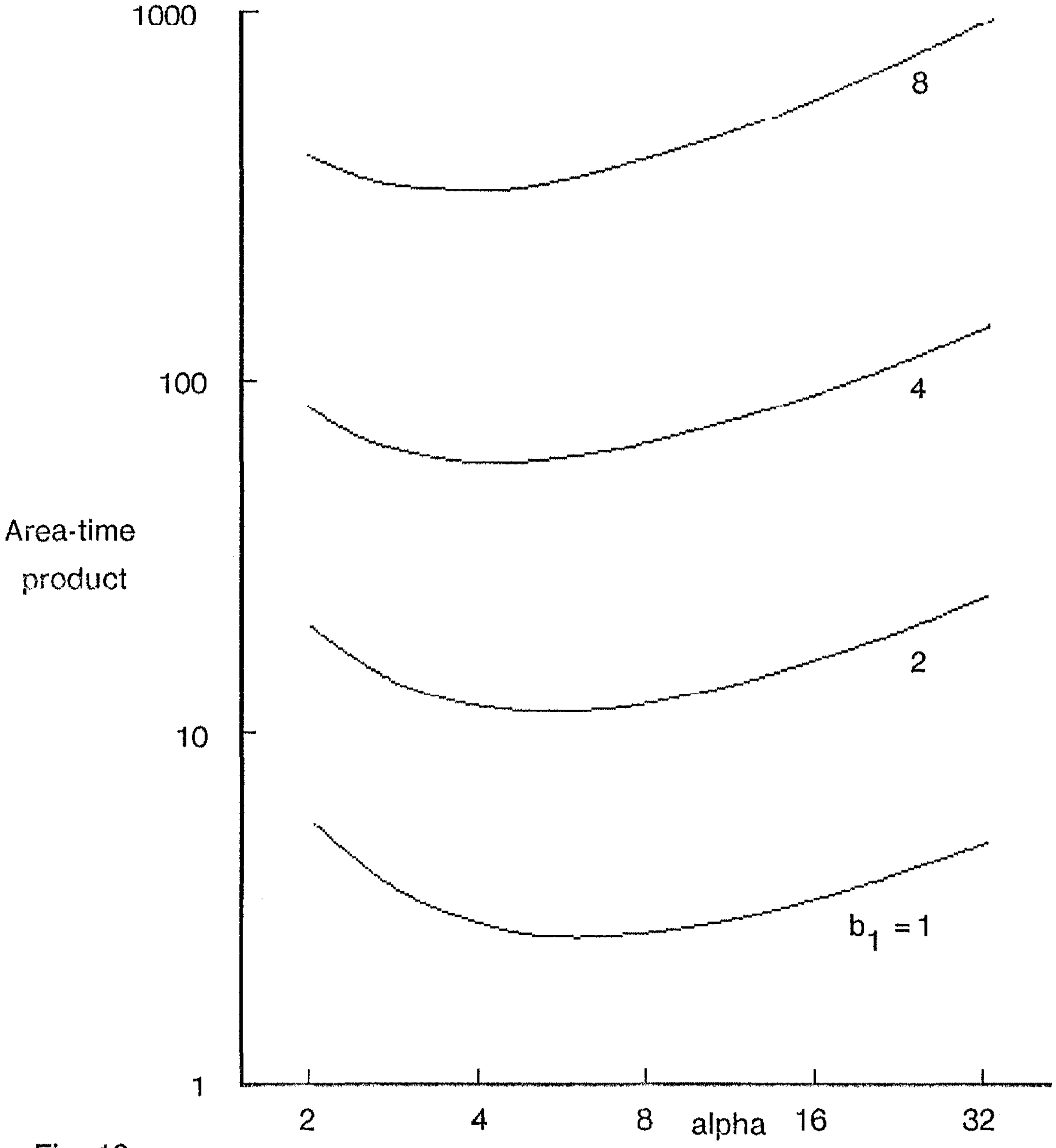

Fig. 10 Area-time product for a CAM of 65k 32 bit words