A Ternary Content Addressable Search Engine

JON P. WADE, MEMBER, IEEE, AND CHARLES G. SODINI, MEMBER, IEEE

Abstract—**This paper describes the design, implementation, and experimental results for a ternary content addressable search engine chip, known as the Database Accelerator (DBA). The DBA chip architecture is presented, and it is well suited to serve as a coprocessor for a variety of logical search applications. The core of the DBA system is composed of novel high-density content addressable memory (CAM) cells capable of storing three states. The design of these cells and their support circuitry is described. The CAM cell and support circuitry were fabricated and their operation confirmed. The circuit implementation of the DBA data path is described with particular emphasis on the optimization of the multiple response resolver. The timing and control methodology, which simultaneously satisfies the complexity, speed, and robustness requirements of the DBA chip, is reported. Finally, experimental DBA chip results are presented; these results verify the full functionality and testability of the design.**

I. INTRODUCTION

THE COMPUTATIONAL speed of a conventional von Neumann computer architecture is limited primarily by the data-path bottleneck between the memory and the central processing unit (CPU). One solution to this problem is to eliminate the separation between the CPU and memory by moving the processor functions directly into the memory. For these purposes, content addressable memory (CAM) is an ideal way to blur the distinction between memory and processing as each memory cell is a processing unit capable of performing the EXCLUSIVE-NOR (equivalence) function.

Although silicon implementations of CAM have existed since 1966 [1], a substantial amount of CAM has yet to be found in conventional computer systems. A recent resurgence in CAM research has resulted in a number of different architectures for a variety of uses. A universal computing architecture [2], which was implemented using a chip consisting of 8 kbits of CAM and the necessary support logic, has been presented [3]. Many special-purpose architectures for character string search [4], [5], pattern inspection [6], and relational searches for artificial intelligence (AI) processors [7]–[11] have also resulted in high-density CAM chips. Each of these chips was designed using static or pseudostatic CAM cells capable of storing the binary states of ONE and ZERO.

None of the aforementioned CAM implementations exploits the potential of high-density ternary CAM in an architecture capable of efficiently performing general logical searches. The Database Accelerator (DBA) chip is an attempt to address this issue by combining a high-density CAM array with a powerful single-instruction, multiple-data (SIMD) parallel processor to create a component useful in many database search applications. The DBA chip can be used as a conventional CAM with arbitrary word lengths and depths supporting arithmetic comparison, approximate matching, and other relational operations. There are a variety of additional applications that exploit the architecture of the DBA chip [12].

The next section describes the advantages of ternary CAM. The overall DBA architecture is presented in Section III. Section IV discusses the implementation of the DBA chip, including the CAM array, parallel processor, and control logic. A test methodology is proposed in Section V. Finally, experimental results and conclusions are given in Section VI.

II. TERNARY CAM

The flexibility of CAM can greatly be extended through the use of trits (*ter*nary dig*its*). Trits consist of the binary logical values ZERO and ONE with the addition of "X" (DON'T CARE). The "X" state is a convenient way of denoting the set consisting of the logical elements ZERO and ONE [13]. Thus, the possible ternary values (denoted with the subscript T) are composed of sets of the usual binary values (denoted with subscript B), as shown in Fig. 1(a). For example, the trit strings "XX0" and "0XX" represent the binary sets $\{000, 010, 100, 110\}$ and $\{000, 001, 010, 011\}$, respectively.

Two useful ternary operations that follow from this definition, namely union and intersection, are shown in Fig. 1(b). With this convention, searches are equivalent to checking for intersections on a trit-by-trit basis between

Manuscript received July 25, 1988; revised December 26, 1988. This work was jointly supported by grants from Analog Devices Inc., Digital Equipment Corp., Harris Semiconductor Corp., IBM Corp., NCR Corp., Raytheon Co., and Sanders Assoc. Inc. In addition, circuit fabrication and other support has been provided by DARPA under Grant N00014-80-C-0622.

J. P. Wade is with Thinking Machines Corporation, Cambridge, MA, 02142.

C. G. Sodini is with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

IEEE Log Number 8928782.

$$0_T \equiv \{0_B\}$$
$$1_T \equiv \{1_B\}$$
$$X_T \equiv \{0_B, 1_B\}$$

(a)

∪	0	1	X
0	0	X	X
1	X	1	X
X	X	X	X

∩	0	1	X
0	0	∅	0
1	∅	1	1
X	0	1	X

$$\text{match}(t,s) \equiv \begin{cases} 1 & \text{if } (t \cap s) \neq \emptyset \\ 0 & \text{otherwise.} \end{cases}$$

(b)

Fig. 1. (a) Representation of binary values with ternary values; and (b) two useful ternary operations, union and intersect.

stored and presented data. For example, the ternary values ZERO and ONE do not intersect and thus do not match. However, the ternary value "X" intersects the ternary values ZERO and ONE and itself and thus matches all these values, hence the name DON'T CARE. Obviously, the ternary values ZERO and ONE intersect themselves and thereby match themselves. Parallel match operations represent the ANDed result of many trit-by-trit intersect operations. Thus, two trit strings match when each trit in the search data intersects the associated trit in the stored data.

Presented DON'T CARE's provide a useful means of removing trit columns in the CAM array from comparison. In this way, CAM words can be broken into fields such that only the parts of interest are queried. The DON'T CARE's can also be used to mask off bit positions for bit-serial algorithms. CAM's have traditionally implemented this feature using separate data and mask registers to present the ZERO, ONE and "X" states. Ternary CAM allows for the storage of these three states as well. Stored DON'T CARE's are extremely useful for providing compact representation of sets of bit strings. In this way, the storage of a single ternary string that represents a set containing a large number of binary strings may result in a significant savings of memory space.

III. Architecture

The DBA chip consists of two major sections. The first is the CAM array that is used for associative operations. The search results are made available to the second part, which consists of many simple, general-purpose, single-bit computing elements. These computing elements, referred to as finite state machines (FSM's), contain a small number of registers with linear nearest-neighbor communication and a single-bit function generator. The data path is used to process the results of match operations to facilitate the use of arbitrary word lengths and complex search algorithms. A selector is used to determine which lines are active in subsequent DBA instructions. The selector input consists of a string of ZERO's, ONE's, and "X's." By allowing DON'T CARE's as inputs to the select logic, it is possible to activate arbitrary binary blocks of the DBA for participation in subsequent instructions. Finally, control logic and I/O latches are used to control the operation of the DBA functional blocks and provide an interface with the overall DBA system. Fig. 2 shows a block diagram of the DBA, with the major blocks being described in Table I.

The DBA architecture can be viewed as an SIMD system, where each processing element contains a word of CAM connected to a 1-bit-wide data path and a global readout data bus. Fig. 3, which is a horizontal cut across the block diagram of Fig. 2, shows in greater detail how the various functional blocks are interconnected. This horizontal cut is referred to as a DBA *line*. In particular, this figure shows the internal details of the FSM.

The DBA chip is organized as 64 lines, each having the structure shown in Fig. 3. Each line consists of a 32-trit word of CAM, a latch to hold the result of match operations, a 1-bit Boolean function generator (FG), four 1-bit registers (R_{0-3}), a selector that controls participation in different operations, and a priority encoder (PE) and latch that is used to serialize the output when more than one line has been matched. The whole system is organized as a set of simple 1-bit data paths that are connected in a linear nearest-neighbor fashion. The FSM takes one bit of input data from one of the current, previous, or next-line register files (A-Source); one bit of data from the current register file or Match latch (B-Source); computes 1 of 16 possible Boolean functions with the function generator (FG); and stores the result in the PE latch and current register file. To continue the A-Source communication scheme across chip boundaries, additional data paths are included to provide access to the adjacent registers of the previous and next DBA chips. In this way, DBA chips can be cascaded into an arbitrarily large system.

The DBA chip supports a minimal set of six different instructions: *Refresh*, *Write*, and *Match* control the CAM array; *Operate* activates the SIMD processor, *Load-Select* selects the active DBA lines, and *Read-Out* allows for the output of data. Sequences of these instructions enable the DBA chip to perform arbitrarily complex matching operations. Ternary data are encoded by two binary bits, *mask* and *data*, which are internally decoded within the DBA chip as necessary. The internal state of the DBA chip is composed of the state of the CAM array; Selector, Match, and PE latches; and FSM file registers. Two outputs are provided from the DBA: *RSP*, a signal that indicates the presence of any valid responders from the most recent Match or Operate instruction, and the address of the lowest valid responder. Except for the Refresh instruction, these instructions all change the state of the DBA and may change the state of the output data. The instruction set is summarized in Table II.

Since the data path has access to the previous and next lines, it is possible to perform match operations on arbitrarily wide words. For example, consider a 64-trit match operation in which each 64-trit word is stored in two successive 32-trit words of a DBA chip. First, the even lines of the DBA are selected, and the even match information is presented. Then, an Operate instruction is used to move the match results to the odd-line register files. The

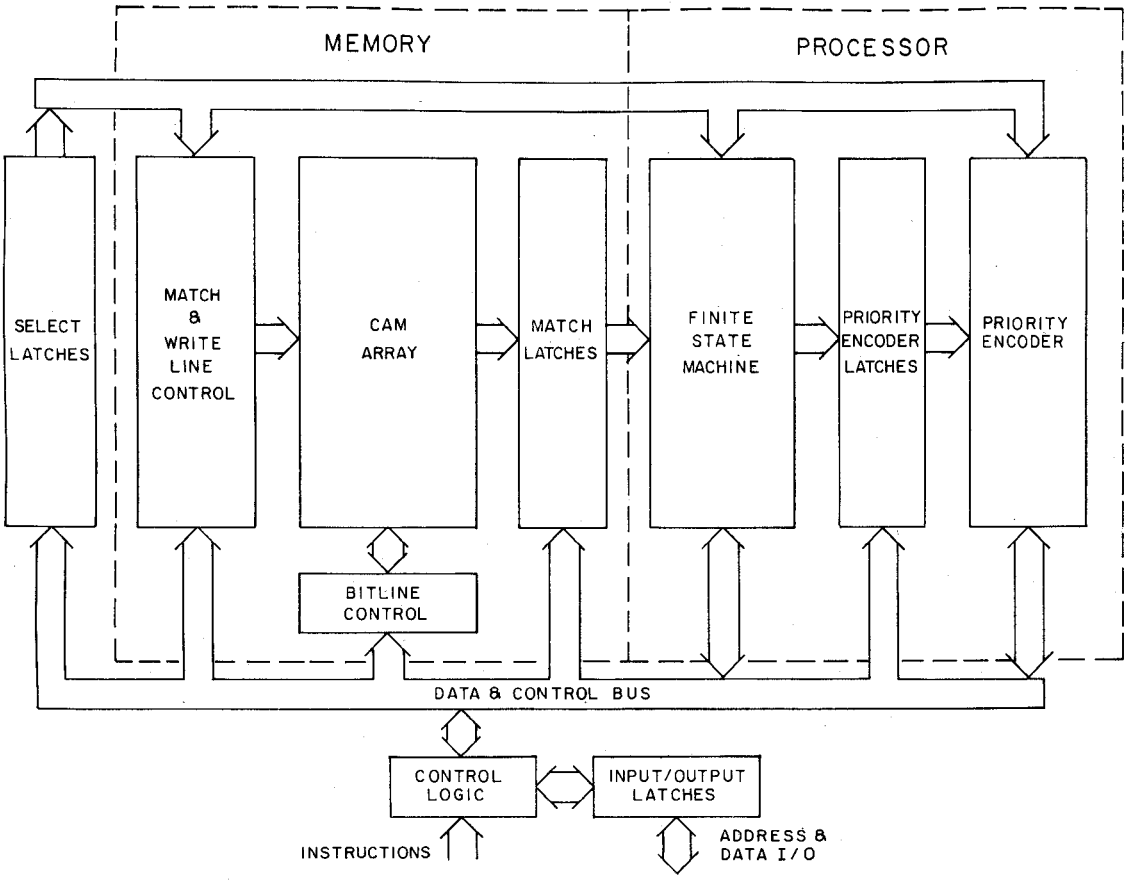


Fig. 2. Block diagram of DBA chip architecture.

TABLE I
DESCRIPTION OF DBA FUNCTIONAL BLOCKS

- *Input/Output Latches* - Latch data being transferred to/from the DBA.
- *Control Logic* - Controls the operation of the DBA, manipulates control signals, coordinates the actions of the different parts, controls the sequence of operations, and runs the I/O interface.
- *Select Latches* - Hold the 'select' information, which specifies the lines of the DBA that will participate in subsequent instructions.
- *CAM Array* - Contains 64 words by 32-trits of content addressable memory.
- *Match Latches* - Hold the results of the Match instruction.
- *Match/Write Line Control* - Controls the CAM cell Match and Write lines.
- *Bit Line Control* - Controls the CAM cell bit lines.
- *Finite State Machines* - Are the components of the SIMD parallel processor used to process Match results.
- *Priority Encoder Latches* - Hold information to be output from the DBA.
- *Priority Encoder* - Encodes the information in the priority encoder latches for output.

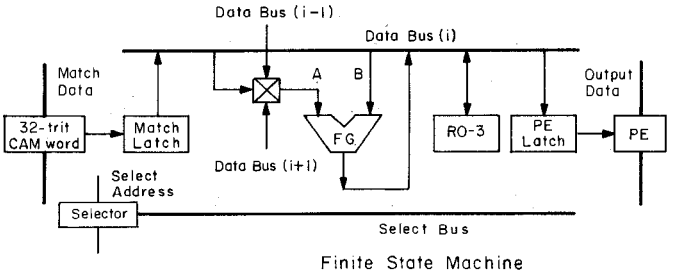


Fig. 3. Functional diagram of DBA data path.

odd lines are selected, and the odd match information is presented. The final 64-trit match result is found by using an Operate instruction to AND together the even and odd-line match results.

Another class of complex match operations is similarity search, which is extremely useful in a variety of image and speech processing and relational database applications. One particular type of similarity search, known as the *nearest-hamming-distance* search, involves finding all those objects that differ in the least number of attributes from a presented target. This search can be implemented by serially presenting target attributes, while masking all other fields with "X's," and counting the number of responders. Since counting requires an adder for each stored object, this is an arithmetic search. Although the DBA chip is not specifically designed to handle such algorithms, the FSM can be used to implement the necessary 3-bit counters. In this case, a maximum of eight attributes each consisting of four trits can be stored on a single DBA line without potentially overflowing the four FSM registers (one register is needed for temporary storage on all but the last count). Since the number of necessary registers grows logarithmically with the number of counts, sharing registers between two lines allows for objects containing a minimum of 64 1-bit attributes. As most foreseeable appli-

TABLE II
SUMMARY OF DBA INSTRUCTION SET

1. **Refresh**—Refreshes a word of dynamic CAM.
   - *Input*—6-trits of refresh-address.
   - *Output*—Null
   - *State*—Null
2. **Write**—Writes ternary data into the selected words of the DBA.
   - *Input*—32-trits of write-data.
   - *Output*—Null
   - *State*—Input data stored in selected CAM words.
3. **Match**—Matches presented ternary data against the selected words in the DBA.
   - *Input*—32-trits of match-data.
   - *Output*—$\overline{RSP}$
   - *State*—Match results stored in Match and Priority Encoder latches. Results in unselected lines are undefined.
4. **Operate**—Executes a 3-operand instruction in the finite state machines of the selected lines in the DBA.
   - *Input:*
     - 4-bits of A-Source
     - 3-bits of B-Source
     - 3-bits of Destination
     - 4-bits of Function-Generator Specifier
   - *Output*—$\overline{RSP}$
   - *State*—FG results stored in Priority Encoder latches.
5. **Load-Select**—Loads a ternary address into DBA's Select latches. The line or lines specified by the contents of the DBA's Selector latch are termed the 'selected' lines and are active in all subsequent instructions.
   - *Input*—6-trit selector data.
   - *Output*—$\overline{RSP}$
   - *State*—Input data stored in Selector latches.
6. **Read-Out**—Outputs the address of the lowest set Priority Encoder latch.
   - *Input*—Null
   - *Output*
     - $\overline{RSP}$
     - 6-bits of Responder address.
   - *State*—Reset lowest set Priority Encoder latch.

Note: Unless otherwise stated, internal state and outputs remain unchanged between instructions.

cations fit within these constraints, four FSM registers per line are considered to be sufficient for the DBA.

IV. IMPLEMENTATION

A. CAM Array

The core of the DBA is the CAM array. The five-transistor dynamic CAM cell [14], shown in Fig. 4(a), is used because of its small size and ability to store three states. The stored DON'T CARE state is necessary in a number of applications. These transistors are configured as an EXCLUSIVE-NOR gate that discharges the match line when the contents of the cell differ from the bit lines. The diode-connected transistor M_D is included in the circuit to eliminate "sneak" current paths between adjacent cells. In this CAM cell, data are stored dynamically as charge on the gates of two transistors. The 0-1 and 1-0 states represent a logical ZERO and ONE, respectively, whereas a 0-0 state represents a DON'T CARE. A cell in the DON'T CARE state is unable to discharge the match line.

Data are stored dynamically on the gates of the M_S transistors with access provided by the M_W transistors. Writing is accomplished by choosing a particular word, raising its write line, and driving the desired logic levels on the bit-1 and bit-0 lines. The M_W transistors are turned ON, which allows the gates of the M_S transistors to be charged or discharged by the bit lines. The DON'T CARE state is stored by discharging the gates of both the M_S transistors.

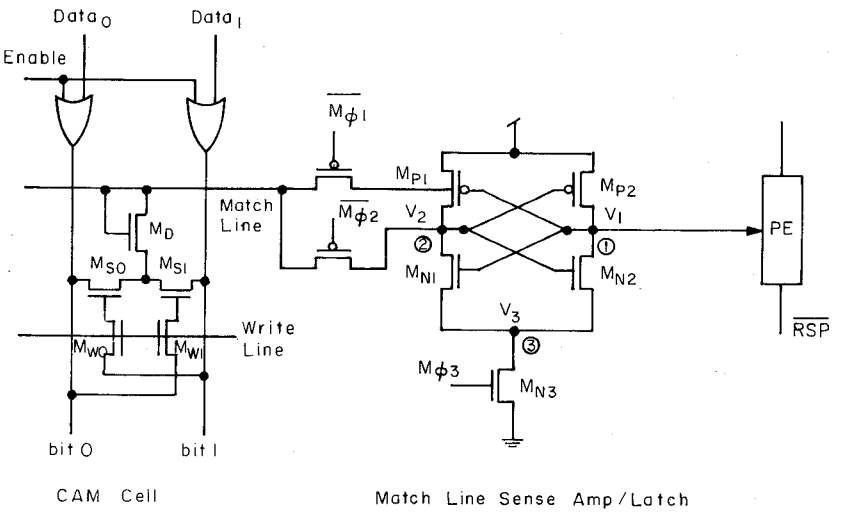
The Match operation is accomplished by holding the match line at a high potential while the bit lines are moved to their desired potentials. If a bit line is lowered on a branch with an ON M_S transistor, current flows from the match line, and a mismatch is detected. For example, if M_{S1} has charge stored on its gate, and bit-1 line is lowered, current flows from the match line. If no current flows from any of the bits in a word, a match is detected.

There are two possible methods of performing an addressed Read operation with this cell. The cell can be read actively by keeping the write line at ground and raising the potential of the match line. Current will flow on a bit line if it is connected to a branch with an ON M_S transistor. This cell can also be read in the same manner as a one-transistor dynamic RAM cell. In this case, the write line is raised, and the storage charge is dumped on the bit lines, which are then sensed. If only ONE's and ZERO's are stored, a differential signal exists on the bit lines, and the charge sense scheme is practical. However, since the DON'T CARE state is stored with two low values, a differential signal does not exist, and thus, an inherently more difficult single-ended sensing must be performed. Therefore, the active read method is used on the DBA chip. Refresh is accomplished using word-by-word Read and Write operations.

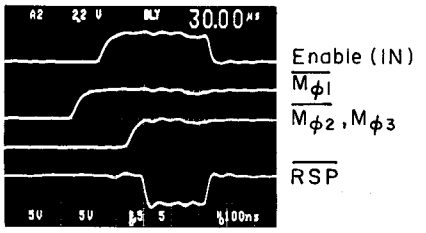
This cell has the advantages and disadvantages that are characteristic of dynamic memory designs. The major advantage of this design is that it is more dense than static designs and is capable of storing three logical states. However, this improvement in area and functionality is accompanied by the need to refresh the CAM array. The Refresh instruction is optimized so that all steps necessary for refresh (selection of a refresh line, read, and write-back) are internally timed and may be executed at the same rate as a Write instruction. Therefore, with a conservative maximum data retention time of 1 ms, less than 5 percent of the available DBA cycles are needed for refresh.

The sensing requirements for this CAM cell are much different that those of traditional one-transistor DRAM cells. The match signal originates from a capacitive load that is discharged from a precharged potential down to within a threshold voltage above ground. This large voltage swing is possible since charge is stored on the gate of a transistor, which results in a nondestructive current driven Match operation. Due to precharge time constraints, the exact initial potentials on the match lines are not known. In addition, bit-line capacitive coupling introduces additional noise, which could lead to an erroneous mismatch signal. Another complication is that the sensing must be done on a single-ended basis.

For these reasons, a sampled time differential scheme, shown in Fig. 4(a), is used. The Match operation proceeds



(a)



(b)

Fig. 4. (a) Circuit diagram of CAM cell and match-line sense amplifier; and (b) I/O waveforms in Match operation. Timing was externally generated with a 100-ns cycle time.

as follows. Initially, the bit-0 and bit-1 lines are at a high potential. $\overline{M_{\phi 1}}$, $\overline{M_{\phi 2}}$, and $M_{\phi 3}$ are at a low potential, and the match line is precharged to a high potential. $\overline{M_{\phi 1}}$ is raised to a high potential, thereby sampling the match-line potential on one side of the sense amplifier latch. Then, the bit lines are moved to their desired potentials. After sufficient time, $\overline{M_{\phi 2}}$ is raised, thus sampling the new match-line potential. Finally, $M_{\phi 3}$ is raised, thereby latching the results. Although this is a differential scheme, the signal source is inherently single-ended such that common-mode noise introduced between the initial and final sampling of the match-line potential introduces the need for additional noise margin.

The W/L ratios of the two nMOS transistors in the sense amplifier latch M_{N1} and M_{N2} are mismatched to provide differential drive when the match line is not discharged. If the W/L ratio of M_{N2} is made larger than that of M_{N1}, an appropriate noise margin is introduced. This noise margin, or offset voltage, allows for a convenient trade-off between speed and noise sensitivity. For example, a large noise margin will guarantee that match-line noise will not cause an erroneous mismatch but increases the amount of time that is required for match-line discharge.

Although the magnitude of the noise margin is a complex function of device geometries, switching waveforms and initial voltage conditions, it is possible to derive approximate expressions that may be used for its estimation. Using first-order MOS $I-V$ relationships, the noise margin for the mismatched cross-coupled sense amplifier can be found to be [15]

$$V_{NM} \simeq (\beta - 1)(V_{DD} - V_{TN} - V_{3i}) \tag{1}$$

where

$$\beta = \left[\frac{(W_2/L_2)}{(W_1/L_1)}\right]^{1/2} \tag{2}$$

V_{3i} is the initial voltage on node 3 and $W_{1,2}$ and $L_{1,2}$ are the widths and lengths of the $M_{N1,2}$ transistors, respectively.

In the current DBA chip design, SPICE simulations showed that the match- and bit-line signals develop at a rate of 25 mV and 5 mV/ns, respectively. The same sensing scheme with reversed sensed signal polarities is used to sense the bit lines during the Read operation. However, since the match-to-bit-line capacitive coupling is quite small, the noise margin can be reduced from the match-line sense case. With the present chip timing, the total match- and bit-line signals can be expected to be approximately 750 and 300 mV, respectively. For the match-line sense amplifier, a β of 1.4 was found to be reasonable. With a threshold voltage of 0.8 V and a precharged voltage of 5.0 V, the noise margin was found to

be 400 mV. The bit-line sense amplifier uses a β of 1.1 for a noise margin of 200 mV. These values, calculated using (1) and (2), were found to agree with SPICE simulations. Experimental results for the Match operation performed on an externally timed 64-word by 32-trit CAM array, integrated in a 3-μm CMOS process, are shown in Fig. 4(b).

B. Finite State Machine

The FSM is severely restricted by the vertical pinch of the CAM array and overall area constraints. The vertical pitch constraint is very important because it determines the feasibility of various FSM topologies. The FSM organization, shown in Fig. 3, uses single-port registers and a single time multiplexed data path to satisfy the vertical pitch constraints. In this topology, there is a function generator and four single-port registers per line. The function generator has two inputs: the A- and B-source. The A-source has access to the four registers in the current, next, and previous lines. The B-source has access only to the local data composed of the registers and match latch in the current line. Since data might be sent across the chip boundaries for the A-source, a single data bus is used to transmit the available local information to the FG, while the interchip, or nonlocal, prefetch takes place. No performance penalties are incurred with off-chip communication. The FSM topology requires the following three data transfers and computation during each Operate instruction: 1) prefetch off-chip A-source data and read B-source data, 2) read A-source data, 3) execute FG operation, and 4) write FG result to PE latch and current-line register destination.

The function generator is a four-to-one multiplexer that selects its output from the function select data dependent on the input state. The function generator is designed using four columns of two series-connected pass transistors. Regardless of the values of the A- and B-sources, only one column of pass transistors is ON at a time, and only one of the control signals is passed to the output. Since both n- and p-channel pass transistors are used, this output may experience a threshold drop below or above the potential of the maximum or minimum input, respectively. For this reason, the function generator uses precharged input and output buffers.

C. Priority Encoder

One of the more difficult issues to resolve in the design of a general CAM architecture is the need to encode multiple match responses. The problem of priority encoding, also known as multiple response resolution, is a general one and is found in many different applications. In the DBA chip, a PE is used to encode and output the location of responders from the Match and Operate instructions. Since the PE determines the maximum rate at which the DBA chip can output data, its design is critical to overall system performance.

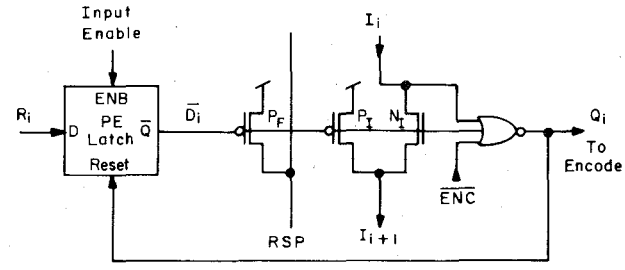


Fig. 5. Block diagram of a single line of the PE.

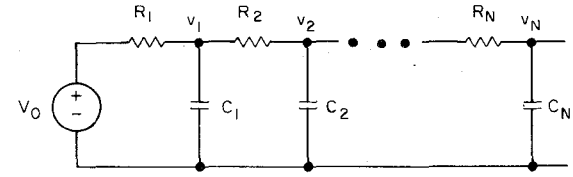


Fig. 6. Diagram of an *RC* ladder network.

The PE used in the DBA chip is a modification of a ripple chain. The simplicity of this technique, its ease of fabrication, and potential for improvement outweigh the advantages of the other priority encoding techniques. Therefore, the ripple chain design is the most attractive choice based on relevant area-performance constraints.

A diagram of a single line of the priority encoding system, which has been utilized in an integrated CAM system [9], is shown in Fig. 5. This system is composed of a PE latch, *RSP* and ripple inhibit chain, and a standard binary encoder. The operation of this system proceeds as follows. The inputs to the priority encoder are first stored in the PE latch, while the *RSP* and inhibit lines are precharged to a low potential. If there are any responders, the *RSP* line is pulled up to a high potential. The *RSP* line serves as an N-input OR gate indicating whether there are any responders among the inputs. If there is a responder on this line, the P_I transistor turns ON, thereby raising the potential of the inhibit$_{i+1}$ line. This will not affect the above inhibit lines since the N_I transistor is turned OFF by the input. If this is the uppermost responder, the inhibit$_i$ line is still at a low potential, and the output of the NOR gate will go high when it is enabled. Only the output of one of the NOR gates can be high at any one time. The outputs of the NOR gates are then sent to a binary encoder, which returns the binary address of the uppermost responder. If the locations of additional responders are desired, the PE latch is enabled, which allows the Reset signal to clear the current uppermost responder, and the PE cycle is repeated. This process can be continued until the *RSP* line signals that there are no more responders.

The performance of this system is limited by the speed of the inhibit chain. If no lookahead techniques are used, the speed of this system is related to the delay through N-unit inhibit pass transistors, where N is the number of PE inputs. This inhibit pass chain can be modeled as a lumped *RC* ladder network, as shown in Fig. 6. It can be shown that for a given input voltage V_0 and output voltage v_N the time T_D that is necessary to achieve that value is

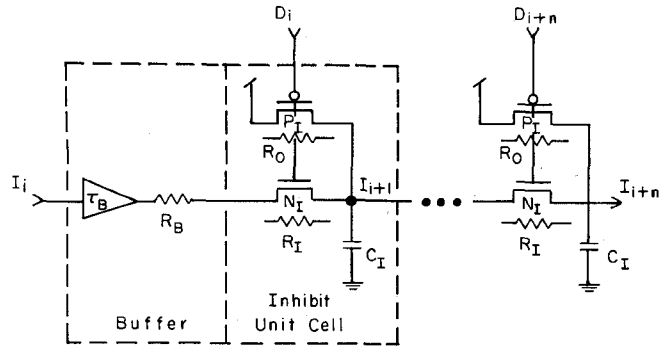


Fig. 7. Diagram of a buffered inhibit chain.

bounded by the expression [16]

$$\frac{v_N}{V_0}\tau_E \leq T_D \leq \frac{V_0}{V_0 - v_N}\tau_E \tag{3}$$

where

$$\tau_E \equiv \sum_{j=1}^{N} C_j \sum_{i=1}^{j} R_i. \tag{4}$$

The term τ_E has the dimensions of time and is equal to the first-order moment of the impulse response. This network delay constant can easily be calculated from the network component values and is instrumental in generating upper and lower bounds to the network step response behavior. In the following discussion, the priority encoder network performance is optimized in terms of τ_E.

The delay of the inhibit chain is directly related to the resistance of the N_I pass transistor and the capacitance of inhibit line, as shown in Fig. 7. In the worst case, only the top and bottom inputs respond, and the inhibit signal must pass through the entire inhibit chain. The maximum inhibit chain delay, τ_I, can be expressed as

$$\tau_I = NR_0C_I + \frac{N(N-1)}{2}R_IC_I \tag{5}$$

which for large N is given as

$$\tau_I \approx \frac{N^2}{2}R_IC_I. \tag{6}$$

The total delay is proportional to the square of the number of inhibit unit stages.

The performance of this structure can be improved by breaking the chain into n-unit sections and inserting a buffer between each section, as shown in Fig. 7. The buffers are characteracterized as having a delay of τ_B and an output resistance of R_B. The maximum inhibit chain delay now becomes

$$\tau_I = \left[N_SR_0C_I + \frac{N_S(N_S-1)}{2}R_IC_I\right] + \left[(n-1)\left(N_SR_BC_I + \frac{N_S(N_S+1)}{2}R_IC_I\right)\right] + [(n-1)\tau_B] \tag{7}$$

where $N_S \equiv N/n$ is the number of inputs per section. The three terms relate to the delay associated with the first section in the inhibit chain, the rest of the inhibit sections, and the delay of the buffer, respectively. For $n, N \gg 1$ (7) becomes

$$\tau_I \approx \frac{1}{n}\left(\frac{N^2}{2}R_IC_I\right) + N\left(R_B + \frac{R_I}{2}\right)C_I + n\tau_B. \tag{8}$$

In this expression, the first term is equal to the result in (6) reduced by the factor $1/n$. Therefore, if the additional delay terms in (8) are designed to be small, the use of buffered stages greatly enhances the performance of the inhibit chain. Obviously, as the number of sections is increased, there are diminishing returns to performance since the buffer delay begins to become important. The performance of this inhibit chain can be optimized by varying the number of sections and geometries of the transistors in each. Setting the partial derivative of (8) to zero with respect to n indicates that delay is minimized when

$$n_I = N\left(\frac{R_IC_I}{2\tau_B}\right)^{1/2} \tag{9}$$

where n_I is the optimum number of unit inhibit sections. Equation (9) shows that the optimum number of sections has a linear dependence on the total number of inputs and a square root dependence on the ratio between the unit delay of the inhibit chain and the buffer delay.

Another technique that can be used to improve the performance of this structure is to use lookahead so that the inhibit signal no longer has to propagate through all n inhibit sections. In this way, a long linear structure can be replaced by a branched tree-like structure that has less latency. Since the DBA chip generates a fast *RSP* signal, this signal is used to generate all the necessary lookahead information. This is done by breaking the *RSP* chain into buffered sections such that each section is isolated from those below and thus sends the proper signal to the appropriate inhibit chains. The *RSP* chain now looks much like the inhibit chain except that its inputs consist of a logical-OR of several PE inputs.

The delay of the *RSP* chain can be related directly to the resistance of the P_F pull-up transistor and to the capacitance of the *RSP* line, as shown in Fig. 8. The worst-case *RSP* delay occurs when there is only one responder. However, since there are no series pass transistors, the location of the single responder has negligible effect on the delay. The maximum *RSP* chain delay for N inputs can be expressed as

$$\tau_F = NR_0C_F \tag{10}$$

where R_0 is the resistance of pull-up transistor P_F, and C_F is the unit capacitance of the *RSP* chain. Unlike the inhibit chain the *RSP* chain delay varies linearly with the number of *RSP* inputs.

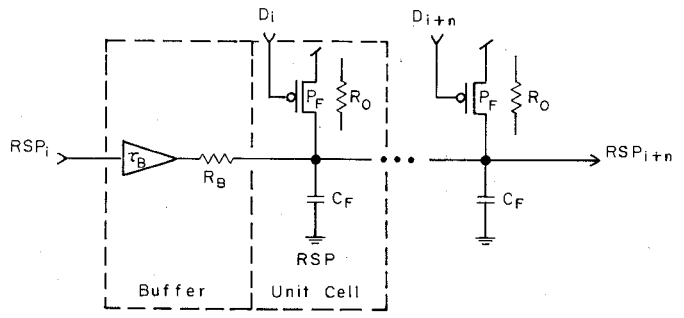


Fig. 8. Diagram of a buffered *RSP* chain.

Although the proposed lookahead scheme requires breaking the *RSP* chain into smaller buffered sections, it is possible that the performance of this structure can be improved as well. A buffered *RSP* section is shown in Fig. 8. The maximum delay now is position dependent and occurs when the top input is the only responder. The maximum *RSP* chain delay now becomes

$$\tau_F = N_S R_0 C_F + (n-1) N_S R_B C_F + (n-1)\tau_B. \quad (11)$$

The three terms are related to the delay associated with the first section in the *RSP* chain, the rest of the *RSP* sections, and the delay of the buffers, respectively. Rearranging (11) gives

$$\tau_F = \frac{R_B}{R_0}(NR_0C_F) + \frac{N}{n}(R_0 - R_B)C_F + (n-1)\tau_B. \quad (12)$$

In this expression, the first term is identical to that of (10) scaled by the factor R_B/R_0. Therefore, if R_B can be made much less than R_0, this reduction may be sufficient to offset the two additional delay terms in (12), and the overall *RSP* chain performance is improved. In practice, the output resistance of the buffer amplifiers can be made much lower than the resistance of the pull-up devices without significantly loading down the *RSP* line. Setting the partial derivative of (12) to zero with respect to n shows that delay is minimized when

$$n_F = \left(\frac{N(R_0 - R_B)C_F}{\tau_B}\right)^{1/2} \quad (13)$$

where n_F is the optimum number of unit *RSP* sections. Equation (13) shows that the optimum number of sections has a square-root dependence on the number of inputs and the ratio between the improvement in the unit delay of the *RSP* chain and the buffer delay.

The prioritizer topology that is implemented in the DBA chip is composed of buffered inhibit chains that use buffered *RSP* chains for lookahead, as shown in Fig. 9. Since the optimal number of inhibit sections is less than that of the *RSP* chain, the overall system contains multistage inhibit sections coupled with longer *RSP* sections.

The prioritizer topology can be characterized by the worst-case inhibit and *RSP* chain delays. The worst case for the *RSP* occurs when there is only one responder

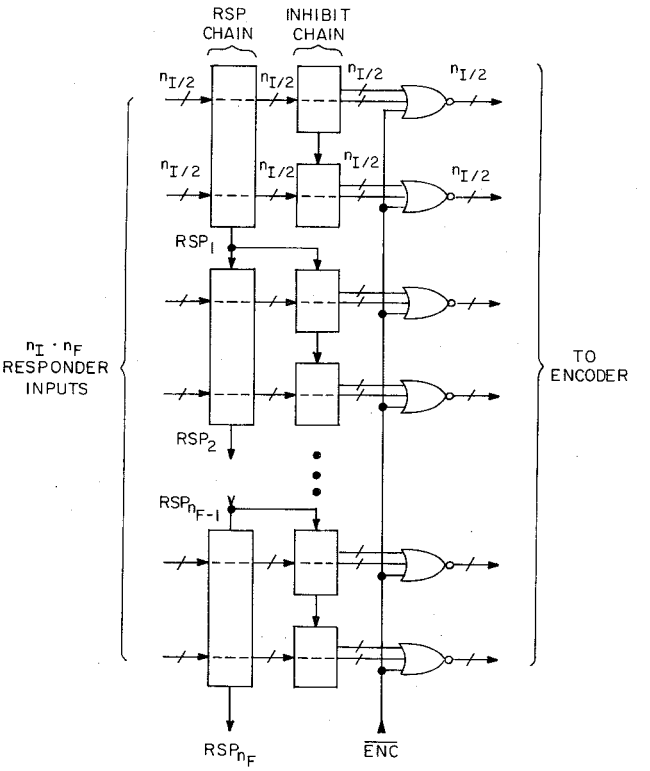


Fig. 9. Block diagram of PE topology.

whose location is in the top *RSP* section and is given by (12). However, the maximum inhibit chain latency occurs when there are two responders: one at the very top and the other at the bottom of the chain. In this case, the inhibit signal must propagate through all but one of the *RSP* sections and through a number of inhibit sections containing a total input number one less than in the last *RSP* section. The maximum inhibit delay can be given as

$$\tau_I = (n_F - 1)\tau_{F0} + (n_I - 1)\tau_{I0} \quad (14)$$

where n_F is the number of *RSP* sections, n_I is the number of inhibit sections per *RSP* section, τ_{F0} is the delay per *RSP* section, and τ_{I0} is the delay per inhibit section. If the number of inputs per section is varied along the chains, it is not possible to determine a general expression for the worst-case inhibit delay. In general, the maximum inhibit signal latency is longer than that of the *RSP* chain. Therefore, the inhibit worst-case delay is a reasonable prioritizer figure of merit.

DBA chip and system requirements have a large impact on PE performance requirements. First, the current DBA chip consists of a single array of 64 lines. The inhibit signal latency must be minimized so that the DBA chip is capable of generating a new responder address on each master clock cycle. Furthermore, the DBA system has a hierarchical prioritization protocol based on DBA chip *RSP* outputs. Therefore, the overall goal is to optimize the PE to simultaneously meet *RSP* signal and responder address generation requirements.

Although the following circuits were designed with these criteria in mind, they are quite general and can be used

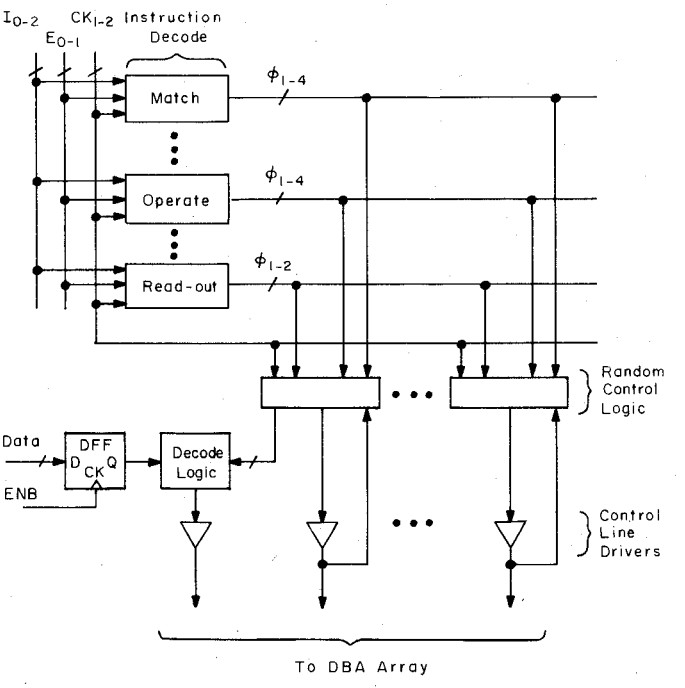
under many other optimizing constraints. From (9), the optimum number of inhibit sections is 16, with $\tau_B \approx 8R_IC_I$ and $N = 64$. Independent optimizations using SPICE supported this selection of inhibit chain section number. In addition, from (13), the optimum number of *RSP* sections is between three and four, with $\tau_B \approx 4R_0C_F$ and $R_0 \approx 4R_B$. However, as the *RSP* chain delay was found to be rather insensitive to the number of sections, inhibit delay and layout were improved with the use of eight *RSP* chain sections.

D. Timing and Control

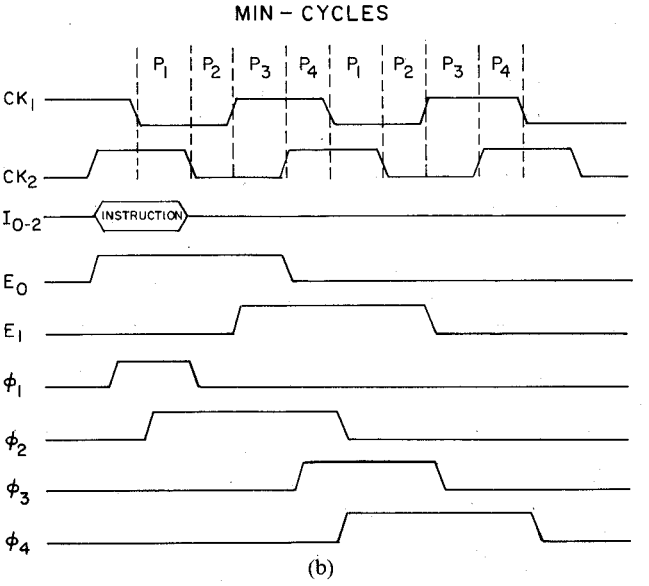
There are three general requirements affecting the design of the control logic: complexity, speed, and robustness. The control requirements are necessarily complex due to the diversity of instructions performed by the DBA chip and the precharged logic that comprises its data path. There are approximately 100 control signals that must be generated with a variety of durations and edges. An additional constraint is that the DBA chip must behave predictably in pathological situations in which the user inputs an incorrect sequence of min-cycle enables and instruction codes. The speed requirement is to minimize latency. Since most DBA algorithms depend on the state of the $\overline{RSP}$ signal, it is important to generate that signal as quickly as possible. For this reason, the overall DBA chip is not organized as a pipelined machine. Finally, the control methodology must be robust so that the DBA data path can function without error under the most perverse possible operational conditions and processing corners. Regardless of potential clock and signal skew, critical signal edge registration must be maintained.

A methodical approach is needed to address the complexity issues described above. This rules out the use of an asynchronous finely tuned delay approach that appears in some memory designs. However, synchronous techniques using look-up tables for each timing edge result in unacceptable speed performance. Although this approach is acceptable in many processor designs implemented with a standard logic family, much of the performance potential of the precharged DBA data path would be sacrificed. Therefore, the control logic for the DBA chip utilizes both synchronous and asynchronous techniques. Next, there is the question of robustness. To reduce the number of globally routed signals and potential skew problems, critical timing edges are determined locally while error handling and basic instruction decode are handled at a global level. However, the control logic must be partitioned so that these critical and noncritical elements can be separated.

Fig. 10(a) shows a block diagram of the proposed control methodology. Instruction I_{0-2} and enable E_{0-1} information is first sent to the instruction decode blocks. Since most of the timing edges can be made to lie on one of four min-cycle transitions, these decoder control blocks each output four bits of signal information from which any desired waveform can be constructed. The decoder control



(a)



(b)

Fig. 10. (a) Block diagram of DBA timing and control logic; (b) example control and timing waveforms.

blocks also perform asynchronous error detection so that erroneous instruction sequences place the DBA chip in the wait state until the next valid instruction sequence. The decoder control outputs, two overlapping clocks, and input data are the only globally routed signals. However, the decoder control outputs and input data are routed asynchronously so that they are valid before being gated locally by the master clocks.

Two externally generated overlapping master clocks CK_{1-2} and on-chip random combinational logic are used to locally generate the actual timing signals. The relationship between typical instruction, control, instruction decode output bits, and min-cycle phases is shown in Fig.

10(b). Some of the input data is further decoded by additional decode logic, which is shown schematically in Fig. 10(a). Since the input data and decoder outputs are both gated by the two master clocks, these signals are not vulnerable to skew problems.

Finally, the timing of the various instructions is arranged in such a way that there are few critical edges or precedence relationships that must be maintained. In the case in which potential race conditions exist, self-timing is used to guarantee the proper precedence relationship. For example, if one timing signal must precede another and both are generated from the same clock edge, the state of the first one is used to enable or disable state changes in the second one. In this way, some control signals in the DBA array are fed back and used as inputs to the random combinational logic to ensure proper control signal precedence. Therefore, although it is good practice to minimize clock skew within the chip, it is not essential with this timing methodology. This timing approach simultaneously satisfies the complexity, speed, and robustness requirements of the DBA data path.

V. Testing

The DBA chip was designed for testability using a variety of techniques. The chip is logically partitioned into four separately testable sections using dynamic shift registers. These sections include the CAM array, FSM, PE, and control logic. This partitioning effectively divides the DBA chip into easily tested sections. Static latches are included with the control logic scan path, which makes it especially useful since it allows for the complete or partial external control of the DBA chip. The following test procedure efficiently checks the entire data path for single transistor-level stuck-at faults.

Testing proceeds from lowest to highest levels of circuit complexity while taking full advantage of previously tested sections. The initial set of tests verify the correct operation of the basic DBA chip support circuitry. These tests are useful in screening die at the wafer probe level for functionality. First, a dc check is performed to check for power shorts. Then the scan-path shift registers are checked for functionality. The control logic shift register is used to check the timing and control signals associated with each DBA instruction.

The final set of tests verify the functionality of the CAM array and processor sections of the DBA chip. First, the PE is checked using the PE scan path as a responder source. Once the PE is checked, these results are used to test the selector. At this point, the FSM is analyzed by separate register and function generator tests. The match scan path is used as an input source for the function generator. The last section to test is the CAM array. The bit-line scan path is used to check the read portion of the Refresh instruction and to verify that input data are correctly decoded and latched. Match operations are performed to test the functionality of the overall CAM array.

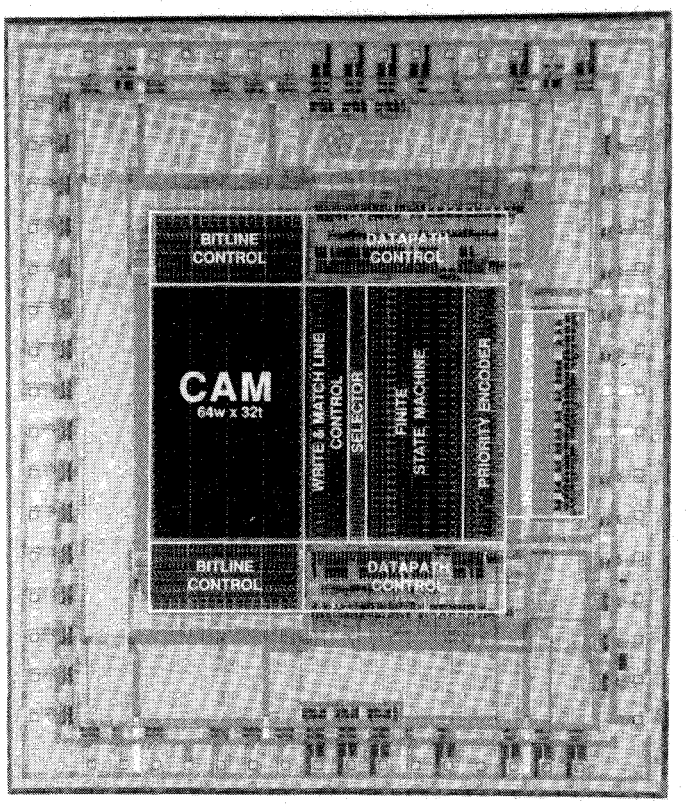


Fig. 11. Photomicrograph of the DBA chip implemented with a 3-μm two-level metal process.

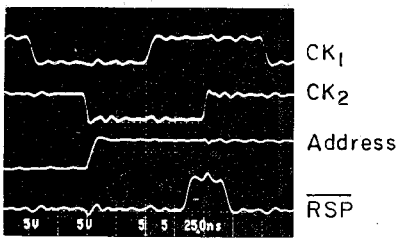


Fig. 12. I/O pulse waveforms in the Read-Out instruction at a 5-MHz clock rate.

VI. Experimental Results

The DBA chip was fabricated through MOSIS using a 3-μm CMOS, two-level metal process. A photomicrograph of the DBA chip with labeled functional blocks is shown in Fig. 11. The active area of the chip is composed of approximately 20-percent CAM cells, 25-percent CAM control logic, 30-percent processor data path, 20-percent data-path control, and 5-percent routing. With scaled design rules, the number of words per array is increased, and this will significantly reduce the relative area consumed by control logic.

As a result of the thorough test methodology, fully functional DBA chips were fabricated, tested, and are being integrated into a DBA system. Typical waveforms of I/O pulses in a Read-Out instruction are shown in Fig. 12. Clocking is accomplished using two 5-MHz clocks CK_1 and CK_2, which are out of phase by 90°. The new responder address data are generated on the falling edge of CK_1

TABLE III
DBA CHIP SPECIFICATIONS

Memory structure	64w × 32trits
Number of instructions	6
Minimum cycle time	200nsec
Power dissipation	< 100mW (5 MHz)
Total device count	30,000
Equivalent chip size (custom pad frame)	4.5 × 5.3 mm
Process technology	3μm CMOS, 2-layer metal

and are valid by the rising edge of CK_1. The rising edge of CK_1 initiates the precharge of $\overline{RSP}$, which is evaluated on the rising edge of CK_2. The new $\overline{RSP}$ signal is valid by the falling edge of CK_1. In this case, $\overline{RSP}$ is discharged, which signals the existence of additional responders whose addresses may be output in subsequent Read-Out instructions. The performance of the DBA chip is summarized in Table III.

VII. CONCLUSION

The DBA chip architecture is designed to complement the processing power of high-density ternary CAM and be a general platform from which to investigate CAM applications. This architecture permits complex search operations with arbitrary word lengths and file depths. The DBA architecture was implemented with a variety of novel circuit techniques for high density and performance. The DBA data path is designed to enable a vertical pitch match with the CAM array, thus allowing for high-density implementation. Since the priority encoder presents a potential serial bottleneck to parallel search operation, this circuitry has been carefully optimized. Finally, a hybrid control methodology is employed on the DBA chip to reduce response latency. Increasing levels of circuit integration have intensified the need of design for testability. Significant effort has been placed on the optimal placement of scan paths to facilitate the testing of the DBA chip. In addition, control state latches are included to verify the functionality of the control logic and, if necessary, permit data-path control by an external source. In conclusion, the design and implementation of the Database Accelerator chip has resulted in significant advances in the following areas of content addressable memory research: CAM chip architectures, CAM cell and sense circuitry design, and support circuit optimizations.

REFERENCES

[1] R. Igarashi, T. Kurosawa, and T. Yaita, "A 150-nanosecond associative memory using integrated MOS transistors," in *ISSCC Dig. Tech. Papers*, 1966, pp. 104–105.

[2] R. M. Lea, "SCAPE: A VLSI chip architecture for image processing," in *Microarchitecture of VLSI Components*, P. Antognetti, F. Anceau, and J. Vuillemin, Eds. Boston: Martinus Nijhoff, 1985, pp. 171–187.

[3] I. P. Jalowiecki and R. M. Lea, "A 256-element associative parallel processor," in *ISSCC Dig. Tech. Papers*, 1987, pp. 196–197.

[4] H. Yamada, M. Hirata, H. Nagai, and K. Takahashi, "A high-speed string-search engine," *IEEE J. Solid-State Circuits*, vol. SC-22, no. 5, pp. 829–834, Oct. 1987.

[5] M. Hirata, H. Yamada, H. Nagai, and K. Takahashi, "A versatile data string-search VLSI," *IEEE J. Solid-State Circuits*, vol. 23, no. 2, pp. 329–335, Apr. 1988.

[6] S.-I. Chae, J. T. Walker, C.-C. Fu, and R. F. Pease, "Content-addressable memory for VLSI pattern inspection," *IEEE J. Solid-State Circuits*, vol. 23, no. 1, pp. 74–78, Feb. 1988.

[7] E. L. Brunvand, "Content addressable memory for symbolic processing systems," Master's thesis, Univ. of Utah, Salt Lake City, Aug. 1984.

[8] T. Ogura, S.-I. Yamada, and T. Nikaido, "A 4-kbit associative memory LSI," *IEEE J. Solid-State Circuits*, vol. SC-20, no. 6, pp. 1277–1282, Dec. 1985.

[9] H. Kodata, J. Miyake, Y. Nishimichi, H. Kudo, and K. Kagawa, "An 8kb content-addressable and reentrant memory," in *ISSCC Dig. Tech. Papers*, Feb. 1985, pp. 42–43.

[10] T. Ogura, S. Yamada, and J. Yamada, "A 20kb CMOS associative memory LSI for artificial intelligence machines," in *Proc. IEEE Int. Conf. Comput. Des.*, 1986.

[11] J. V. Oldfield, "Logic programs and an experimental architecture for their execution," *Proc. Inst. Elec. Eng.*, vol. 133, pp. 163–167, May 1986.

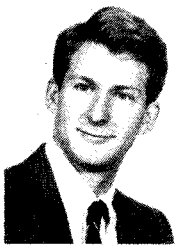
[12] C. G. Sodini *et al.*, "The MIT database accelerator: A novel content addressable memory," in *Wescon Dig. Tech. Papers*, 1986.

[13] F. P. Herrmann, "Ternary logic and algorithms," M.I.T. SMP internal document, Jan. 1988.

[14] J. P. Wade and C. G. Sodini, "Dynamic cross-coupled bitline content addressable memory cell for high density arrays," in *IEDM Tech. Dig.*, Dec. 1985.

[15] J. P. Wade, "An integrated content addressable memory system," Ph.D. dissertation, Mass. Inst. of Technology, Cambridge, MA, May 1988.

[16] P. Jr. Penfield and J. Rubinstein, "Signal delay in RC tree networks," in *Proc. Caltech Conf. VLSI*, Jan. 1981, pp. 269–283.



Jon P. Wade (S'83–M'88) received the S.B., S.M., E.E., and Ph.D. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology Cambridge, between 1984 and 1988. His doctoral research included the design and development of content addressable parallel processor applications, architectures, and circuitry.

He was a member of the technical staff at Texas Instruments Central Research Laboratories from 1980 to 1984, where he was involved in the modeling and design of GaAs monolithic microwave power amplifiers. He is currently employed by Thinking Machines Corporation, Cambridge, MA, where he is engaged in advanced parallel processor design.

Charles G. Sodini (S'80–M'82) was born in Pittsburgh, PA, in 1952. He received the B.S.E.E. degree from Purdue University, Lafayette, IN, in 1974, and the M.S.E.E. and Ph.D. degrees from the University of California, Berkeley, in 1981 and 1982, respectively.

He was a member of the technical staff at Hewlett-Packard Laboratories from 1974 to 1982, where he worked on the design of MOS memory and later on the development of MOS devices with very thin gate dielectrics. He joined the faculty of the Massachusetts Institute of Technology, Cambridge, in 1983, where he is currently an Associate Professor in the Department of Electrical Engineering and Computer Science. His research interests are focused on IC fabrication, device modeling, and device-level circuit design, with emphasis on analog and memory circuits.

Dr. Sodini held the Analog Devices Career Development Professorship of MIT's Department of Electrical Engineering and Computer Science and was awarded the IBM Faculty Development Award from 1985 to 1987. He has served on a variety of IEEE Conference Committees including the International Electron Device Meeting, where he is the 1989 General Chairman.