Journal of VLSI Signal Processing 15, 247–257 (1997)
© 1997 Kluwer Academic Publishers. Manufactured in The Netherlands.

Digital Design of Discrete Exponential Bidirectional Associative Memory*

CHUA-CHIN WANG AND CHIH-LWAN FAN
Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan 80424

Received September 28, 1995; Revised May 31, 1996; Accepted June 11, 1996

Abstract. The exponential bidirectional associative memory (eBAM) was proved to be a systematically stable high-capacity memory. Considering the difficulty of the implementation of such an eBAM by analog circuits and the compactability with binary logic circuits, we adopt the digital logic methodology to design such a neural network. Besides, we also count in other factors, e.g., scalability and speed, so that the complete digital design of this neural network is feasible. In order to realize the eBAM by digital circuitry only, some special design is required such that the exponential function can be implemented without the loss of operating speed. For example, a high-speed 8-to-9 exponent value generator is required in the design. In addition, because the traditional add/sub accumulator costs too much area when the dimension of patterns is large, a cascaded increment/decrement accumulator (IDA) is proposed in the design, which can also speed up the addition or subtraction besides the saving of chip area. For the sake of area saving, regenerated IDA is also proposed to reduce the cost of chip area. At last, thorough simulations by MAGIC and IRSIM are proceeded to verify the performance of the design.

1. Introduction

After Kosko [1] proposed the *bidirectional associative memory* (BAM), many researchers threw efforts on improving its intrinsic poor capacity and implementing the BAM with hardware circuits. Among those researchers, Wang et al. [2] proposed two alternatives, *multiple training and dummy augmentation*, to enhance BAM's ability to find the global minimum; Simpson proposed an intraconnected BAM and a high-order autocorrelator [3] and Tai et al. [4] proposed a high-order BAM; Wang et al. [5] developed a *weighted learning* algorithm for BAM. However, we have pointed out that all of these improvements pay a high price of increasing the complexity of the network but only get little enhancement of the capacity [6]. Chiueh and Goodman [7] proposed an exponential Hopfield associative memory motivated by the MOS transistor's exponential drain current dependence on the gate voltage in the subthreshold region such that the VLSI implementation of an exponential function is feasible. Based upon the concept of Chiueh's exponential Hopfield associative memory, Jeng et al. proposed one kind of exponential BAM [8]. However, the energy function proposed in [8] cannot guarantee that every stored pattern pair will have a local minimum on the energy surface. Moreover, there is no capacity analysis given in [8].

After we estimated the impressive capacity of an eBAM [6], it becomes very interesting to explore the possibility of the hardware realization of such a neural network. Even though the neural networks implemented with MOS operating in the subthreshold region have the advantages of low power and compatibility with VLSI circuits, [9, 10], employing only digital logic to realize such a network still remains an option because the advantages of the digital logic design [11]. It is well known that digital logic possesses advantages of noise margin, design scalability, and less complexity. The only drawback is the exponential function used in such a network. In this paper, we first show the architecture of the eBAM, including the realization of exponential function needed in such a network, cascaded increment/decrement accumulator, and so on [11]. Then, thorough simulation of the logic

*This research was partially supported by National Science Council under grant NSC 84-0408-E-110-003.

design of the eBAM is shown in order to verify its performance.

2. Digital Design of Exponential BAM

Before proceeding the hardware implementation of the eBAM, we have to restate the framework of the eBAM neural network as a background knowledge [6].

2.1. *Theory of eBAM*

Suppose we are given M bipolar pattern pairs, which are $\{(X_1, Y_1), (X_2, Y_2), \ldots, (X_M, Y_M)\}$, where $X_i = (x_{i1}, x_{i2}, \ldots, x_{in})$, $Y_i = (y_{i1}, y_{i2}, \ldots, y_{ip})$, $X_i \neq X_j$, $i \neq j$, and $Y_i \neq Y_j, i \neq j$. We use the following evolution equations in the recall process of the eBAM.

$$y_k = \begin{cases} 1, & \text{if } \sum_{i=1}^{M} y_{ik} b^{X_i \cdot X} \geq 0 \\ -1, & \text{if } \sum_{i=1}^{M} y_{ik} b^{X_i \cdot X} < 0 \end{cases}$$

$$x_k = \begin{cases} 1, & \text{if } \sum_{i=1}^{M} x_{ik} b^{Y_i \cdot Y} \geq 0 \\ -1, & \text{if } \sum_{i=1}^{M} x_{ik} b^{Y_i \cdot Y} < 0 \end{cases} \qquad (1)$$

where b is a positive number, $b > 1$, "$\cdot$" represents the inner product operator, x_k and x_{ik} are the kth bits of X and the X_i, respectively, and y_k and y_{ik} are for Y and the Y_i, respectively. Note that X and Y in Eq. (1) are input patterns. The reasons for using an exponential scheme are to enlarge the attraction radius of every stored pattern pair and to augment the desired pattern in the recall reverberation process. We adopt the SNR (signal-to-noise ratio) approach to compute the capacity of the exponential BAM [6], $\text{SNR}_{\text{eBAM}} = \frac{2^{n-1} b^4}{2(M-1)(1+b^{-4})^{n-1}}$ where n is assumed to the min(n, p) without any loss of generality.

2.2. *Architecture of Digital eBAM*

Without any loss of generality, our design is dedicated to 8-bit patterns. Note that the same design methodology is also applied to patterns with larger length. Referring to Fig. 1, which shows the entire architecture of the digital eBAM, we divide the design into three major parts, X & Y RAM planes with shift controllers, the 8-to-9 exponent value generator (EVG), and increment/decrement accumulators (IDA). Before we discuss the details of each block of the system, we'd like to show how this architecture corresponds to the evolution equations in Eq. (1).

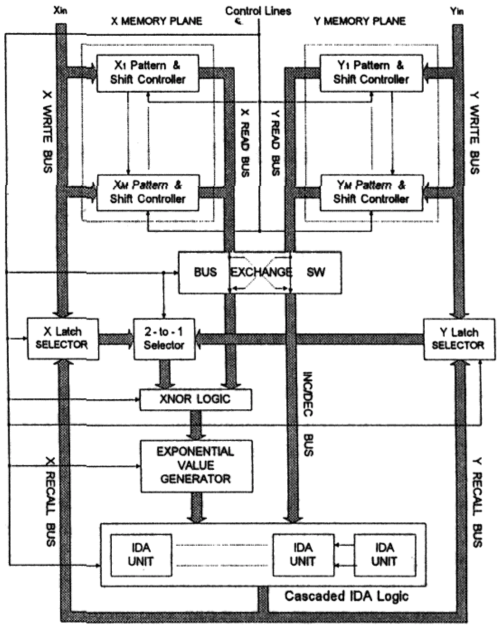


Figure 1. Architecture of eBAM.

The Control Lines are in charge of the Bus Exchange SW and the 2-to-1 Selector. They determine whether $X_i \cdot X$ or $Y_i \cdot Y$ is going to be executed. The "$\cdot$" operation is processed by XNOR gates, i.e., the XNOR logic block in Fig. 1. For instance, if the recall of Y is chosen to be processed, then Control Lines select the data on X Read Bus which are fed into XNOR logic to execute inner product with the retrieval X vector, i.e., $X_i \cdot X$. Then the output of XNOR logic block is presented to the Exponent Value Generator (EVG) to get $b^{X_i \cdot X}$.

Meanwhile, Control Lines also connect INC/DEC Bus to Y Read Bus such that Y_i will be delivered with $b^{X_i \cdot X}$ to the cascaded increment/decrement accumulator (IDA) Logic block to generate $y_{ik} \cdot b^{X_i \cdot X}$, where y_{ik} is the kth bit of $Y_i \cdot y_{ik}$ decides the sign of $b^{X_i \cdot X}$. Hence, we need at least 8 IDA units since the length of patterns is 8 bits.

After repeating M times of the same procedure, $\sum_{i=1}^{M} y_{ik} \cdot b^{X_i \cdot X}$ will be generated and then saved in Y Latch Selector through Y Recall Bus. Then, the Y

Latch Selector will trigger the 2-to-1 Selector if the recalled Y is incorrect. The 2-to-1 Selector is then toggled and starts the recall of X, which is similar to the recall of Y mentioned in the above. The entire recall process will be terminated when the newly recalled X and Y are respectively identical to the X and the Y in the last recall which are stored in the X Latch Selector and the Y Latch Selector, respectively.

2.2.1. *RAM Planes with Shift Controllers.*

The RAM planes for the digital eBAM are slightly different from the traditional DRAM or SRAM. There is no address decoder in our design because the decoder will consume a large area if the number of stored pattern pairs is large. In contrast, we use a series of shifters, or called shift controllers, to mark which RAM module will be written with data next. The block diagram of X-Y RAM plane is shown in Fig. 2. This simplifies the design complexity and increases the speed of data access. Every RAM module in Fig. 2 is composed of one pair of X and Y 8-bit vectors. In the initialization of the chip, all of the shifters' contents are set to "0", except the first RAM module. The module of which the shifter is containing "1" is allowed to fetch data from Data Write Bus or deliver data to Data Read Bus.

The detailed schematic diagram of the RAM cell and the shift controller is shown in Fig. 3. The passing of "1" is executed by N2, N3, Not1, and Not2. Note that the "1", called the permission of data access, is defined as the output of Not2. During the initialization, $\overline{Enable}$ stays at "1". Then, a positive pulse at *Initial* will turn on N1 which makes the output of Not2 low. When *Initial* returns to low, N1 is OFF. It is clear that $\overline{Enable}$ controls the flow of the permission of data access. When $\overline{Enable} = 0$, the contents of the previous stage will be passed from Din to Not1, and the data at the output of Not2 will be passed to the next stage through N3. When $\overline{Enable} = 0 \rightarrow 1$ N2 should be ON before N3

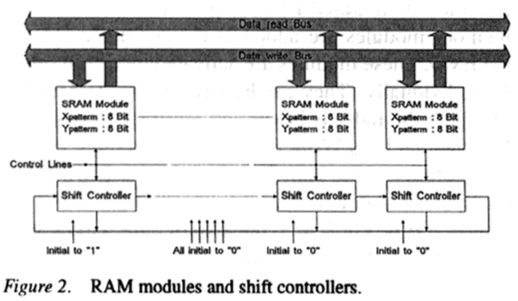


Figure 2. RAM modules and shift controllers.

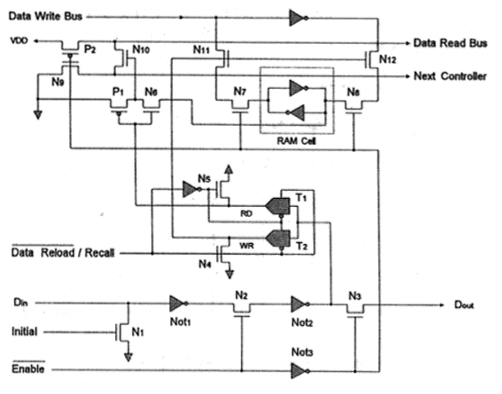


Figure 3. Schematic diagram of the shifter.

is OFF such that the passing of permission will be correct. Hence, we have to make the delay through Not3 less than that through Not2. If the shifter contains "0", then N6, N11, N12, and N10 are OFF. Then the RAM module can not be either written into or read from. It is virtually isolated from the buses.

When the shifter contains "1", $\overline{Data\ Reload/Recall}$ will decide whether it is a write or read operation:

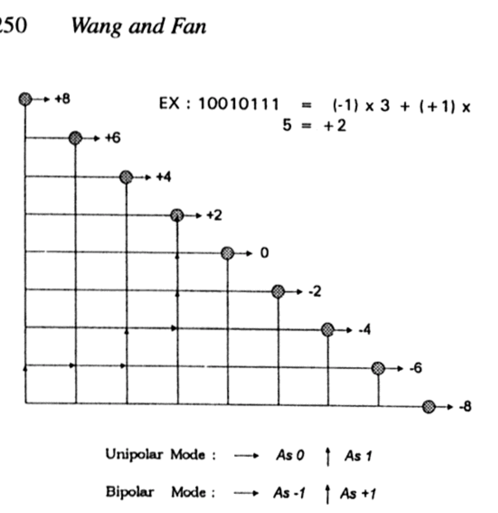
write operation. $\overline{Data\ Reload/Recall} = 0$. N4 is OFF while N5 is ON. RD is grounded. T1 is OFF and T2 is ON. WR is connected to the output of Not2 such that it is high. Thus, N11, N12, P1 are ON, while N6 is OFF and then N10 is OFF. The data on Write Bus will be latched into RAM cell through N7 and N8.

read operation. $\overline{Data\ Reload/Recall} = 1$. N5 is OFF and N4 is ON. WR, thus, is grounded to be "0". Besides, T1 is ON and T2 is OFF. Then RD is connected to the output of Not2 to be "1". Meanwhile N11, N12, P1 are OFF, and N6 is ON. The gate of N10 is connected to the inverting port of RAM cell. Data Read Bus is precharged when $\overline{Enable}$ is high. The data will be verified on the bus through N9 and N10 when $\overline{Enable}$ goes low.

Basing on the simulation results by MAGIC layout and IRSIM, both the read operation and the write operation can be finished in 5 ns.

2.2.2. *Exponent Value Generator (EVG).*

The exponential function is the core of the eBAM, because it drastically enlarges the signal-to-noise ratio [6]. The range of the exponent of the inner product to two 8-bit



EX : 10010111 = (-1) x 3 + (+1) x 5 = +2
+8
+6
+4
+2
0
-2
-4
-6
-8
Unipolar Mode : → As 0 ↑ As 1
Bipolar Mode : → As -1 ↑ As +1

Figure 4. Path finding of EVG.

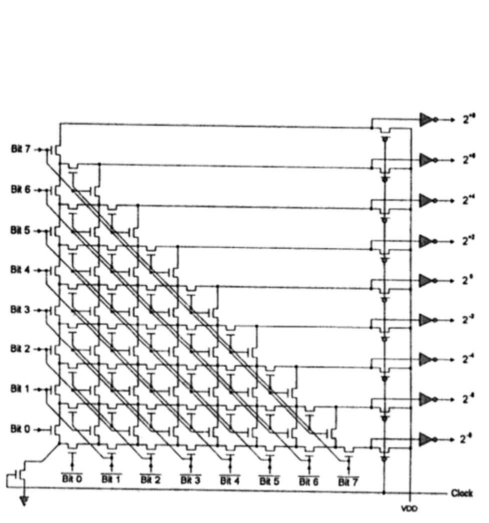


Figure 5. Schematic diagram of EVG (radix = 2).

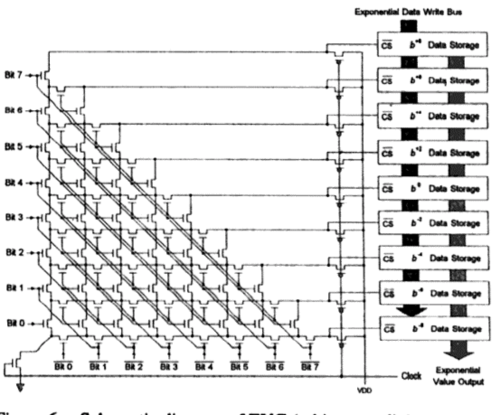


Figure 6. Schematic diagram of EVG (arbitrary radix).

bipolar vectors will be 2^{-8} to 2^{+8}. If the direct implementation of such a digital circuit by either SOP or POS form is adopted, we can foresee a huge and slow circuit will appear. This is why we discard this direct implementation.

Since our design is dedicated for 8-bit data vectors, the result of the inner product of the exponent, e.g., $X \cdot X_i$, can only be one of the following integers, $-8, -6, -4, -2, 0, 2, 4, 6, 8$. That is, there are 9 possibilities for the exponent. This indicates a 8-to-9 exponent value generator is required. The result of such a value generator is only one "H" (high or "1") out of the 9 possible outputs and the rest 8 outputs are all "L" or "0". The algorithm is shown in Fig. 4. The architecture of the generator is the lower left half of a 8×8 box. If one bit of the bitwise XORed result of X and X_i is 0, then move horizontally one grid to the right; if it is 1, then move upwardly one grid. The procedure starts from MSB to LSB or from LSB to MSB of the inner product. An example is also shown in Fig. 4. For instance, if the result of the inner product is (1 0 0 1 0 1 1 1), then the output for +2 should be high, and the rest are all 0. The circuit of such a generator is shown in Fig. 5.

Referring to Fig. 5, the NMOS pass transistor logic is employed to realize such a generator. The Clock will precharge the output when it is low. Bit0, ..., Bit7 are the bits of the result of the inner product. Thus, those bits which are "1" will turn on its corresponding NMOS pass transistor. This makes sure that any 8-bit vector will create a path from the ground at the lower left corner to one of the outputs. The inverters at the righthand side are used to invert the ground to Vdd, i.e., "1". Hence, we can generate a number which is a power of 2.

According to the derivation in [10], the SNR will increase dramatically if the radix b increases. Certainly we like to implement a large b in the eBAM if it is feasible. A modified EVG is shown in Fig. 6. Nine memory modules are added at the righthand side of the EVG. These modules are activated by a chip select $\overline{CS}$ individually. They can be written with larger data of which the radix is larger and arbitrary through the Exponential Data Write Bus. In reality, though we can use arbitrary radix, e.g., $b = 3, 4, 5, \ldots$, it will need more accumulators if b is not the power of 2. For instance, if $b = 3$ or $b = 5$, the area of chip will be very huge. Hence, the best choice of radix is the powers of 2, e.g., 2, 4, 8, 16, ...

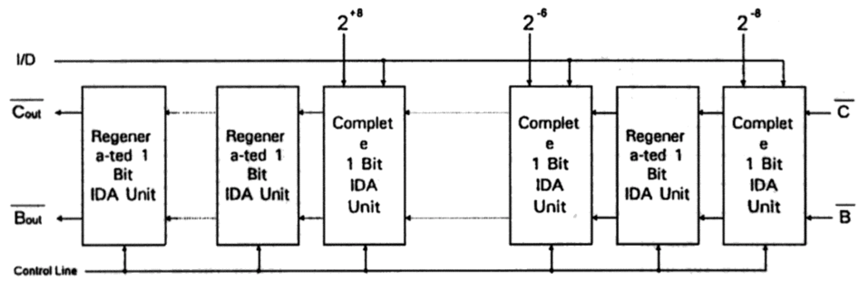


Figure 7. Block diagram of cascased IDA.

The simulation results by MAGIC layouts and IRSIM tools show that the generation of correct exponent value given a 8-bit vector can be finished in 5 ns.

2.2.3. *Increment/Decrement Accumulator (IDA).*

Since the eBAM uses summations in Eq. (1), e.g., $\sum_{i=1}^{M} y_{ik} b^{X_i \cdot X}$, adders or accumulators are needed. The y_{ik}s are either 0 or 1 in the binary system, or -1 or $+1$ in the bipolar system. There are two reasons for us to adopt accumulators instead of traditional adders. First, the adders will occupy a large chip area when the bit length of the vectors increases. For instance, if carry look ahead adders are used, the area will be large and the delay will be very long because the sum in Eq. (1) will be about 20 bits. Besides, a 2's complement conversion circuit is required to carry out the addition for $y_{ik} = -1$. Second, according to Eq. (1), what we need is the sign of the overall sum, not the exact numerical value of the overall sum. We can take advantage of this reason to reduce the complexity of the design.

We propose a special design to resolve this problem, as shown in Fig. 7 which are composed of cascaded complete increment/decrement accumulators (IDA) and regenerated IDAs. In every IDA unit, C_{out} denotes the carry, while B_{out} denotes the borrow. Because both the carry and the borrow are saved before the next operation instead of sending to the next unit right away, the IDA can be deemed as a two-stage pipelined accumulator. The *Bit* is the output from EVG. That is, the *Bit* denotes $2^{+8}, 2^{+6}, \ldots,$ or 2^{-8} to one IDA. I/D denotes the y_{ik} (or x_{ik}). If *Bit* $= 0$, then I/D has no impact on the result. If *Bit* $= 1$ and $I/D = 1$, it is an increment operation, i.e., $+1$; if *Bit* $= 1$ and $I/D = 0$, it is a decrement operation, i.e., -1. Since the input to $2^{+7}, 2^{+5}, \ldots, 2^{-7}$ is always 0, the IDAs for these bits can be simplified. The simplified IDA is called regenerated IDA. The schematic circuit diagrams for the complete IDA and the regenerated IDA will be studied later in the text.

D	C	B	Bit	Inc/Dec		Dnext	Cnext	Bnext
0	0	0	0	0		0	0	0
0	0	0	0	1		0	0	0
0	0	0	1	0		1	0	1
0	0	0	1	1		1	0	0
0	0	1	0	0		1	0	1
0	0	1	0	1		1	0	1
0	0	1	1	0		0	0	1
0	0	1	1	1		0	0	0
0	1	0	0	0		1	0	0
0	1	0	0	1		1	0	0
0	1	0	1	0		0	0	0
0	1	0	1	1		0	1	0
0	1	1	0	0		0	0	0
0	1	1	0	1		0	0	0
0	1	1	1	0		1	0	1
0	1	1	1	1		1	0	0
1	0	0	0	0		1	0	0
1	0	0	0	1		1	0	0
1	0	0	1	0		0	0	0
1	0	0	1	1		0	1	0
1	0	1	0	0		0	0	0
1	0	1	0	1		0	0	0
1	0	1	1	0		1	0	1
1	0	1	1	1		1	0	0
1	1	0	0	0		0	1	0
1	1	0	0	1		0	1	0
1	1	0	1	0		1	0	0
1	1	0	1	1		1	1	0
1	1	1	0	0		1	0	0
1	1	1	0	1		1	0	0
1	1	1	1	0		0	0	0
1	1	1	1	1		0	1	0

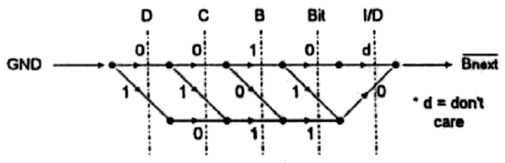
Figure 8. Truth table of the complete IDA unit.

Referring to Fig. 8, the entire truth table for a complete IDA unit is shown. Note that D is the current data of the unit, C is the saved carry, B is the saved borrow, *Bit* is the output of the corresponding bit from EVG, and *Inc/Dec* is the I/D. The generated output signals are, respectively, D_{next}, C_{next}, and B_{next}. For the sake of simplicity, we decompose the IDA truth table into small truth tables for D_{next}, C_{next} and B_{next}, respectively, so that the design will be easier.

B_{next}: The truth table for $B_{\text{next}} = 1$ is shown at the top of Fig. 9. If the conventional SOP or POS form is used to implement the circuit, the area will be large

D	C	B	Bit	I/D
0	0	0	1	0
0	0	1	0	0
0	0	1	0	1
0	0	1	1	0
0	1	1	1	0
1	0	1	1	0

Truth table of B_{next} of the complete IDA unit



Paths of B_{next} of the complete IDA unit

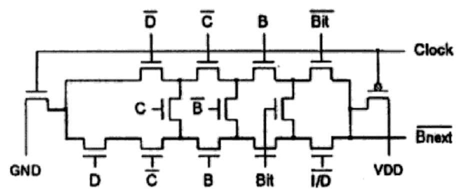
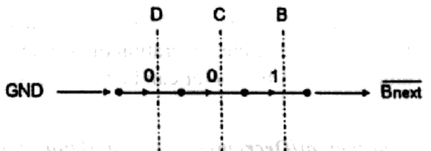


Figure 9. Schematic diagram of B_{next} of the complete IDA unit.

D	C	B	Bit	I/D
~~0~~	~~0~~	~~0~~	~~1~~	~~0~~
0	0	1	~~0~~	~~0~~
0	0	1	~~0~~	~~1~~
~~0~~	~~0~~	~~1~~	~~1~~	~~0~~
~~0~~	~~1~~	~~1~~	~~1~~	~~0~~
~~1~~	~~0~~	~~1~~	~~1~~	~~0~~

Truth table of B_{next} of the regenerated IDA unit



Paths of B_{next} of the regenerated IDA unit

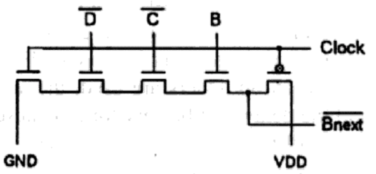


Figure 10. Schematic diagram of B_{next} of the regenerated IDA unit.

and the speed will be slow. We adopt a path finding algorithm which is similar to the pass transistor logic to implement such a function, as shown in Fig. 9. The basic idea is if the input signals, D, C, B, *Bit*, *I/D*, satisfy one of the rows in the truth table for B_{next}, they must open a path from GND to $\overline{B_{next}}$. The schematic circuit is also shown in Fig. 9.

If $Bit = 0$ which implies a regenerated B_{next}, the truth table then can be simplified as shown at the top of Fig. 10. Those rows with $Bit = 1$ are deleted. Consequently, I/D will be "don't care" in this regard. The path diagram and the circuit diagram for the regenerated B_{next} is also shown in Fig. 10.

C_{next}: The truth table for $C_{next} = 1$ is shown at the top of Fig. 11. This table is also extracted from the table in Fig. 8. The path finding algorithm again is applied to the design of the circuit, which is also shown in Fig. 11.

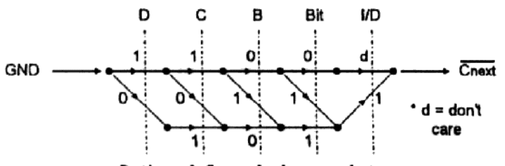
If $Bit = 0$, then we can have a simplified C_{next} function. The reduced truth table, the path finding diagram and the circuit of the regenerated C_{next} are shown in Fig. 12.

D_{next}: It is similar to the B_{next} and C_{next}. The truth table, the path diagram, and the circuit diagram of D_{next} are shown in Fig. 13, while the regenerated D_{next}'s is in Fig. 14.

An IDA's Circuit Diagram. After every individual part of the IDA unit is studied, the circuit for a IDA unit, complete or regenerated, is illustrated in Fig. 15. When $\overline{Initial} = 0$, B_{next}, C_{next}, and D_{next} precharged to 1 when *Clock* is high. Meanwhile, every individual operation is executed when $Clock = 1$. If $Clock = 0$, then N4, N5, and N6 are ON while N1, N2, N3, N7, N8, and N9 are OFF so that the generated B_{next}, C_{next}, and D_{next} are locked between inverters. They will be passed to the next stage during the next clock cycle. The complete IDA units and the regenerated IDA units are alternatively cascaded in the IDA for bits $2^{+8}, \ldots, 2^{-8}$ as shown in Fig. 7. Another three more regenerated IDA units are added ahead of the unit for 2^{+8} so as to prevent

D	C	B	Bit	I/D
0	1	0	1	1
1	0	0	1	1
1	1	0	0	0
1	1	0	0	1
1	1	0	1	1
1	1	1	1	1

Truth table of C_{next} of the complete IDA unit



Paths of C_{next} of the complete IDA unit

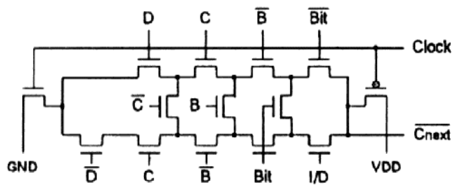
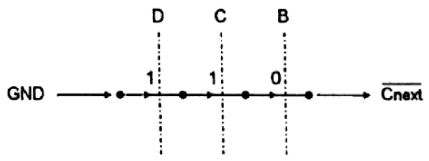


Figure 11. Schematic diagram of C_{next} of the complete IDA unit.

D	C	B	Bit	I/D
~~0~~	~~1~~	~~0~~	~~1~~	~~1~~
~~1~~	~~0~~	~~0~~	~~1~~	~~1~~
1	1	0	~~0~~	~~0~~
1	1	0	~~0~~	~~1~~
~~1~~	~~1~~	~~0~~	~~1~~	~~1~~
~~1~~	~~1~~	~~1~~	~~1~~	~~1~~

Truth table of C_{next} of the regenerated IDA unit



Paths of C_{next} of the regenerated IDA unit

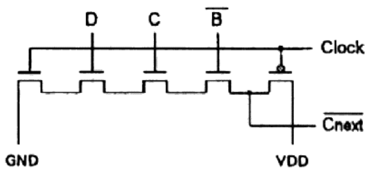


Figure 12. Schematic diagram of C_{next} of the regenerated IDA unit.

the possible overflow problem. That is, in our implementation, a total of 20 stages are used. The final result is the sign of the MSB according to Eq. (1). The *Clock* and $\overline{Clock}$ control the 2-phase operation of the IDA; in Phase 1, the logic operation is completed; and then in Phase 2, the current data is updated with new data.

Again we utilize the MAGIC layouts and IRSIM to verify that the delay of each IDA will be no longer than 5 ns. That is, the sign of MSB will be correctly derived in 100 ns.

2.3. Flow Control of the eBAM Chip

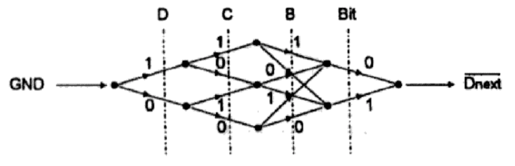
Lots of mentioned units utilize the precharge scheme to increase the operating speed. The simulation results of individual units show that 5 ns could be the best choice of clock period. Thus, if the duty cycle of this clock period is set to be 50%, then we have 2.5 ns to precharge and 2.5 ns to verify. In addition, we also hope to further improve the speed to the chip by parallel processing different units which are mutually independent. Then, at least three clocks with different delays are needed.

System Clock. The system clock is the main clock of the chip, which is provided externally. Its task is to precharge during the first half of a clock cycle and to verify during the second half.

First Delayed Clock. The first delayed clock lags behind the system clock by a total duration of the maximum verified output delay of the RAM module's shift controller and the delays caused by the pass transistor and the XNOR gate. This delay is sufficiently long enough to place the inner product result at the inputs of EVG. The first half cycle precharges the output of

D	C	B	Bit	I/D
0	0	0	1	D
0	0	1	0	D
0	1	0	0	D
0	1	1	1	D
1	0	0	0	D
1	0	1	1	D
1	1	0	1	D
1	1	1	0	D

Truth table of D_{next} of the complete IDA unit



Paths of D_{next} of the complete IDA unit

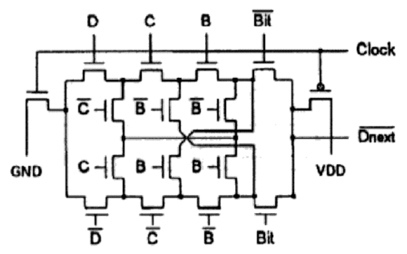
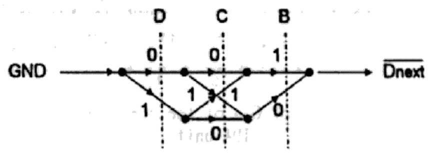


Figure 13. Schematic diagram of D_{next} of the complete IDA unit.

D	C	B	Bit	I/D
~~0~~	~~0~~	~~0~~	~~1~~	~~D~~
0	0	1	~~0~~	~~D~~
0	1	0	~~0~~	~~D~~
~~0~~	~~1~~	~~1~~	~~1~~	~~D~~
1	0	0	~~0~~	~~D~~
~~1~~	~~0~~	~~1~~	~~1~~	~~D~~
~~1~~	~~1~~	~~0~~	~~1~~	~~D~~
1	1	1	~~0~~	~~D~~

Truth table of D_{next} of the regenerated IDA unit



Paths of D_{next} of the regenerated IDA unit

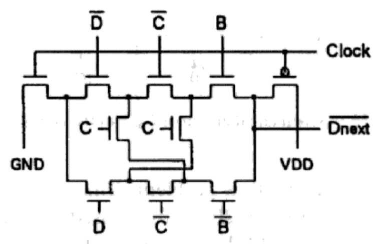


Figure 14. Schematic diagram of D_{next} of the regenerated IDA unit.

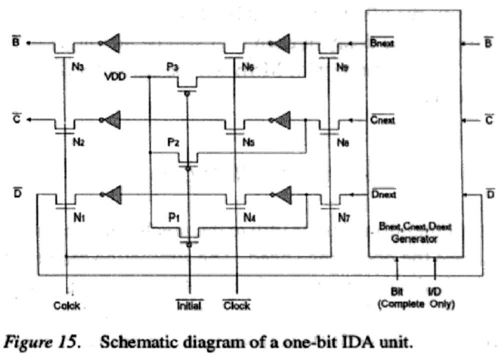


Figure 15. Schematic diagram of a one-bit IDA unit.

the EVG, and the second half verifies the output of the EVG.

Second Delayed Clock. The second delayed clock lags behind the system clock by a total duration of the delayed amount of the first delayed clock and the maximum verified output delay of the IDA unit, which is long enough for the correct output of the EVG to be placed at the input of the IDA units. The first half cycle precharges the outputs of the IDA units, and the second half verifies the outputs.

3. Simulation Analysis

Our digital eBAM chip is designed by MAGIC in 0.8 μm CMOS technology, as shown in Fig. 16. The

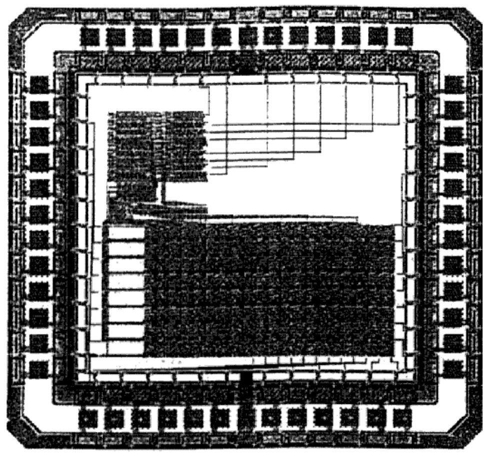


Figure 16. Chip layout of the eBAM.

area of the chip is 1400 × 1400 μm^2. The IRSIM of MAGIC is used to verify the performance of the chip. Figure 17 is a timing diagram of recalling a stored pattern pair when the given retrieval pattern has no bit error, where the "INIT−" is the initialization signal, "BX8−" to "BX8+" are the exponent values, "ID0" to "ID7" are the outputs of the IDA units, and "SGN0" to "SGN7" are the final recalled pattern. Other signals in Fig. 17 include "DR−" which loads in pattern pairs when low and start the recall when high, and "CLK−" is the system clock. The time to recall the correct pattern is 45 ns which is from the positive edge of "DR−" to the appearance of "SGN0", ..., "SGN7". In contrast, if the retrieval pattern has one bit error, the result is shown in Fig. 18. The time to recall the corresponding pattern pair is 55 ns. In addition, we conclude the approximate time to recall the first pattern pair in this chip is about (5 ns) × (no. of pattern pairs) + (5 ns) × (20 stages of pipeline transfer).

Compared with another implementation of eBAM by analog current-mode circuits [10], which takes at least about 2 × 150 ns to have a correct recall, the digital design is much faster. Besides, every unit in this digital design in verified to be functionally standalone. It implies that this design methodology can be applied to patterns with a large dimension. Though the area of the digital design in this work is slightly larger than that of [10], which is 880 × 1280 μm in 0.8 μm CMOS technology, it is more insensitive to the presence of noise because there is no analog unit in this digital design.

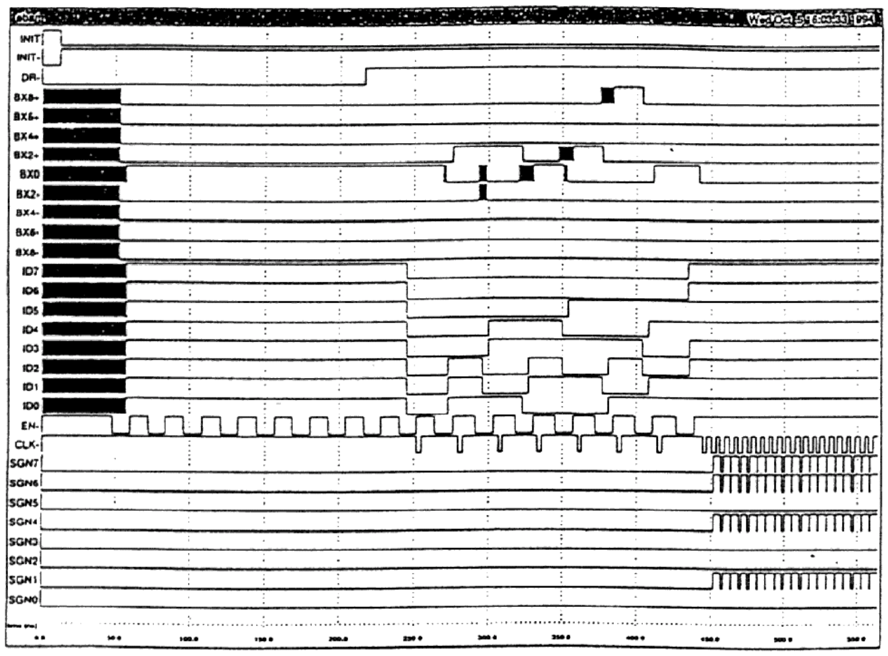


Figure 17. Timing diagram of recall without bit error.

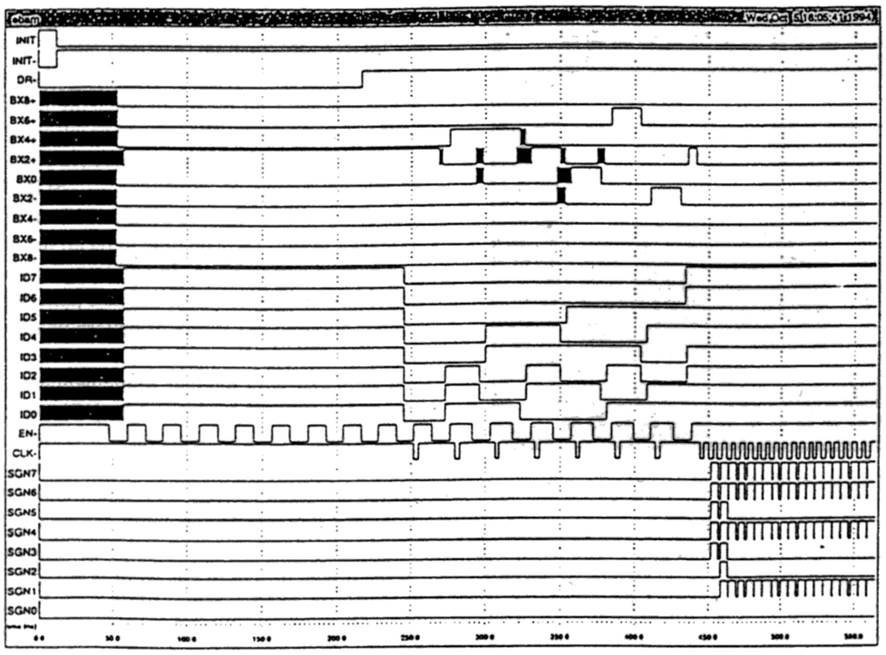


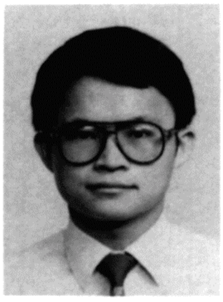
Figure 18. Timing diagram of recall with one-bit error.

4. Conclusion

The eBAM has been proved to be a high capacity associative memory which is worthy of hardware implementation. Our work has proven that the complete digital realization is a feasible possibility. The design is scalable because the design methodology is the same. The exponent value generator and the pipeline IDA units are the key factors to make the digital design possible. Though the chip area is not economical, it is still in a reasonable range. In contrast, the speed of the chip is fairly fast.

References

1. B. Kosko, "Bidirectional associative memory," *IEEE Trans. System Man Cybernet*, Vol. 18, No. 1, pp. 49–60, Jan./Feb. 1988.
2. Y.-F. Wang, J.B. Cruz, Jr., and J.H. Mulligan, Jr., "Two coding strategies for bidirectional associative memory," *IEEE Trans. Neural Network*, Vol. 1, No. 1, March 1990.
3. P.K. Simpson, "Higher-ordered and intraconnected bidirectional associative memory," *IEEE Trans. Systems Man Cybernetics*, Vol. 20, No. 3, May/June 1990.
4. H.M. Tai, C.H. Wu, and T.L. Jong, "High-order bidirectional associative memory," *Electron. Lett.*, Vol. 25, pp. 1424–1425, 1989.
5. T. Wang, X. Zhuang, and X. Xing, "Weighted learning of bidirectional associative memories by global minimization," *IEEE Trans. on Neural Networks*, Vol. 3, No. 6, pp. 1010–1018, Nov. 1992.
6. C.-C. Wang and H.-S. Don, "An analysis of high-capacity discrete exponential BAM," *IEEE Trans. on Neural Networks*, Vol. 6, No. 2, pp. 492–496, March 1995.
7. T.D. Chiueh and R.M. Goodman, "High-capacity exponential associative memory," *Proc. IJCNN*, Vol. I, pp. 153–160, 1988.
8. Y.-J. Jeng, C.-C. Yeh, and T.-D. Chiueh, "Exponential bidirectional associative memories," *IEEE Electronics Letters*, Vol. 26, No. 11, pp. 717–718, May 1990.
9. T.D. Chiueh and R.M. Goodman, "Recurrent correlation associative memories," *IEEE Trans. on Neural Networks*, Vol. 2, No. 2, pp. 275–284, 1991.
10. C.-C. Wang and J.-M. Wu, "Analysis and current-mode implementation of asymptotically stable exponential bidirectional associative memory," *1995 IEEE Inter. Sym. on Circuits & Systems*, pp. 421–424, May 1995.
11. C.-C. Wang and C.-L. Fan, "Digital realization of exponential bidirectional associative memory with dynamic storage," *Proc. of 1995 Workshop on Computer Applications*, Taichung, April 1995, pp. 85–89.
12. C.A. Mead, *Analog VLSI and Neural Systems*, Addison-Wesley, MA, (Reading), 1989.



Chua-Chin Wang was born in Taiwan in 1962. He received the B.S. degree in electrical engineering from National Taiwan University in 1984, and the M.S. and Ph.D. degree in electrical engineering from State University of New York in Stony Brook in 1988 and 1922, respectively. He is currently an Associate Professor in the Department of Electrical Engineering, National Sun Yat-Sen University, Taiwan. His recent research interests include low power and high speed logic circuit design, VLSI design, neural networks, and fuzzy logic.