Chapter 5

PCB Implementation of PLPN-Based Bidirectional Associative Memory

5-1 Implementation Description

The present implementation is intended to simulate and implement the parallel learning-processing strategy mapped in Bidirectional Associative Memory composed of six neurons in the *X* layer and four neurons in the *Y* layer. The circuit is divided into three sub-circuits, each of which has to perform a specific task namely, the synaptic strength sub-circuit, *X* layer sub-circuit and *Y* layer sub-circuit. The associated binary words (images) are sent successively to the synaptic strength sub-circuit to generate excitatory and inhibitory synaptic strengths. Then both are multiplied by the input signal vector that is already stored in the D-Type flip flops of the *X* layer neurons output. The results are half-added separately (the inhibitory part and the excitatory part), then fully-added to be stored in the memory of an accumulator. When the associated words have all been sent, the memory of the accumulator sends its value to the comparison (four-bit magnitude comparator (7485)) component which indicates the maximum between the inhibitory sum and the excitatory sum. If the excitatory sum is greater than the inhibitory sum then the neuron output (comparator) gets '1' otherwise, it gets the value '0'.

The output of this layer is sent to the multiplication sub-circuit of the next *Y* layer where the associated binary words sent to the *X* layer are sent to the *Y* layer successively. All processes performed in the *X* layer are realised in the *Y* layer. After the *Y* layer generates its output vector, which is fed back to the *X* layer, the above mentioned cycle is repeated. The PLP strategy is realised in this implementation through the dynamical storage of the inner product of the associated binary words continuously sent to the BAM network with the stimulus vector initially stored in a layer neurons outputs. This dynamicity is assured by the clock signals of D-Type flip-flops injected in the different sub-circuits.

5-2 EEDIII CAD Designing Methodology

EEDesignerIII (EEDIII) is a PC based Computer Aided Design (CAD) software. This CAD tool was created by VISIONICS SCANDINAVIA for designing both analogue and digital circuits. It gives the facility to the designer to draw his/her schematic and route between the different components, that he/she calls for, by introducing their standard reference code. Then, the software transfers the schematic (after packaging all the components) to a layout. Before that, the designer may verify the functionality of the schematic by simulating its outputs and for that, he/she may use a simulation function. The simulation function called LSIMULATE provides at maximum 20 signal generators of clock signals with different periods and displays. After the simulation, the designer transfers the schematic to an unrouted PCB layout. Next, the designer routes between IC gates manually by the guidance of EEDIII or automatically using the CAD auto-routing facility. Along designing process, the EEDIII CAD tool offers many options and facilities such as dimensioning, error auto-detection and correction…etc.

5-3 PLP-Based Neuron Compositional Structure

The PLP-based neuron circuit is divided, based on its different functionalities, as mentioned before into the following three sub-circuits (see Figure (7)):

1. Synaptic strength sub-circuit.
2. *X* layer sub-circuit containing four neurons.
3. *Y* layer sub-circuit containing six neurons.

In the synaptic strength sub-circuit, the two types of the synaptic strength are generated for each neuron by using the XOR gates and the D Type flip-flops. The number of the positive outputs *N* (four in our case) of the D Type flip flop is equal to the number of number of bits of the input signal of the *Y* layer sub-circuit. Equal number *N* of XOR gates are connected to each D Type flip-flop positive and complementary outputs

The multiplication stage is implemented in the next layer input. The multiplication is performed, separately between the excitatory side and the inhibitory side, using AND gates. The multiplication products are sent to the half adder stage where two-bit binary words will be generated to sum them in the next full-adder stage. Using the two-bit-binary full adder, the full adder stage produces three-bit binary word representing the sum of four bit of the input signal multiplied by the corresponding synaptic strength. Finally, the binary word is sent to the accumulator stage where it is summed with the previous sums (resulted from the previous sent memories). The accumulator stage is composed of a memories serially connected with a full adder as shown in Figure (1).

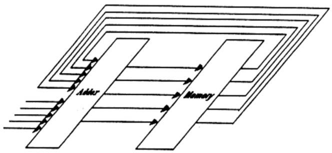

Figure (1). The Accumulator Architecture.

In the present PCB implementation, a four-bit D Type flip-flop was used as the memory and four bit full adder as the adder unit in the accumulator of the *X* layer. The last stage of the *X* sub-circuit layer is the comparators. In this hidden stage, the two sides of the precedent sub-circuit layer are unified: the excitatory side and the inhibitory side using comparator (four-bit comparator). The comparator compares between the two summations of the accumulators, the excitatory summation and the inhibitory summation. If the excitatory sum is greater than the inhibitory sum, the comparator outputs high logical signal, otherwise the output is a low logic signal. The outputs of the comparators are sent to D-Type flip-flops (four-bit flip-flop), where the states of the different neurons of the *X* layer are stored.

The forward process of the adopted neural associative memory (BAM) ends at the end of the above sub-circuit. For the backward process, the outputs of the X layer neurons are sent to the next *Y* layer sub-circuit.

The structures of the *Y* layer sub-circuit are similar to the *X* layer sub-circuit structures differing only in number of components because of the different neurons number.

5-4 The System Synchronisa tion

Synchronisation in the present context is to make the implemented circuit consistent by avoiding the overlap between the different processes as well as making the circuit as fast as possible. This synchronisation is based on the clock and the clear signals of the different sub-circuits D-Type flip-flops. The clock signals are used to pass the outputs from the precedent stage (sub-circuit) to the next one using the D Type flip-flop. The clear signals are used only by the accumulators D Type flip-flops to clear their contents in order to hold the new sums.

At the start of processing, the input binary word (stimulus) representing the initial network configuration is loaded to the *X* or *Y* layer sub-circuit outputs. The loading is performed using Quad Bilateral Switch connected to the D-Type flip-flops holding the outputs of the chosen layer.

5-4-1 Symbols Definition

In this section, the symbols of different components and signals used in the circuit synchronisation are defined.

[X]: Clock signal of the D-Type flip-flop that sends the output of the *X* layer to the input of the *Y* layer.

[Y]: Clock signal of the D-Type flip-flop that sends the output of the *Y* layer to the input of the *X* layer.

CLR_1 : Clock signal for clearing the accumulator memory in the *Y* layer sub-circuit.

CLR_2 : Clock signal for clearing the accumulator memory in the *X* layer sub-circuit.

CLK_1 : Clock signal for flipping the accumulator memory in the *Y* layer sub-circuit.

CLK_2 : Clock signal for flipping the accumulator memory in the *X* layer sub-circuit.

Pdt_dff_x: Propagation delay time of the D Type flip-flop in the *X* layer sub-circuit.

Pdt_dff_y: Propagation delay time of the D Type flip-flop in the *Y* layer sub-circuit.

Pdt_comp_x: Propagation delay time of the comparator in the *X* layer sub-circuit.

Pdt_comp_y: Propagation delay time of the comparator in the *Y* layer sub-circuit.

Pdt_add_x: Propagation delay time of the accumulator adder in the *X* layer sub-circuit.

Pdt_add_y: Propagation delay time of the accumulator adder in the *Y* layer sub-circuit.

Pdt_Dff_x: Propagation delay time of the accumulator memory of *X* layer sub-circuit.

Pdt_Dff_y: Propagation delay time of the accumulator memory of *Y* layer sub-circuit.

Pdt_in_add_x: Propagation delay time from multiplication to the adder input in *X* layer.

Pdt_in_add_y: Propagation delay time from multiplication to the adder input in *Y* layer.

5-4-2 Clocks Synchronisation

The clock signals format adopted by the EEDIII CAD tools is the following:

CLK: $L\,a, (H\,b, L\,c)$.

L and H means low logical level '0' and the high logical level '1' respectively and a, b and c are the multiples of the basic time unit representing the duration of logical level mentioned beside them. The bracket in the CLK representation represents the repeated part of the clock, whereas the first period ($L\,a$) is the initial logic level of the clock and its duration (see Figure (2).

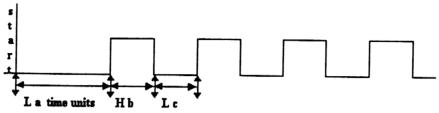

Figure (2). The diagram of the generated clock signal in the EEDIII simulator.

Adopting the above clock format for the different clocks used in the processing of the circuit and their relation with the different propagation delays of the involved components, the following relations are deduced. The suggested relations are deduced by minimising the setup time of each of the different involved stages components.

[X]: $L\,x_1, (H\,x_2, L\,y_3)$.

[Y]: $L\,x_1, (H\,y_2, L\,y_3)$.

$\mathrm{CLR}_1 : L\,c_{11}, (H\,c_{12}, L\,c_{13})$.

$\mathrm{CLR}_2 : L\,c_{12}, (H\,c_{22}, L\,c_{23})$.

$\mathrm{CLK}_1 : L\,k_{11}, ([H\,k_{11}], [L\,k_{13}])$.

$\mathrm{CLK}_2 : L\,k_{21}, ([H\,k_{22}], [L\,k_{23}])$

x_1: Optional duration but it depends on the switcher, i.e. the clock signal of the X layer D Type flip-flops has to get the high logical level '1',directly after the low edge of the switcher clock.

x_2 = Pdt_in_add_y + Pdt_add_y (Start of CLK$_1$) + Pdt_Dff_y.

$x_3 = y_2$ + Pdt_comp_y + Pdt_comp_x.

$y_1 = x_1 + x_2$ + Pdt_comp_x.

y_2 = Pdt_in_add_x + Pdt_add_x (Start of CLK$_2$) + Pdt_Dff_x.

$y_3 = x_3 + x_2 - y_2$.

$c_{11} = x_1$ + Pdt_in_add_y.

c_{12} = Pdt_add_y + Pdt_Dff_y.

$c_{13} = x_2 + x_3 - c_{12}$.

$c_{21} = y_1$ + Pdt_in_add_x.

c_{22} = Pdt_add_x (Start CLK$_2$) + Pdt_Dff_x.

$c_{23} = y_2 + y_3 - c_{22}$.

$k_{11} = c_{11}$ + Pdt_add_y (start CLK$_1$).

k_{12} = Pdt_Dff_y.

$k_{13} = c_{12} + c_{13} + k_{12}$.

$k_{21} = c_{21}$ + Pdt_add_x (Start CLK$_2$).

k_{22} = Pdt_Dff_x.

$k_{23} = c_{22} + c_{23} - k_{22}$.

In the above relations, the start of the clock means the first positive edge of the clock or starting the periodic term in the representation of the mentioned clock.

5-5 Timing of the Increased Stored Images

To increase the stored associated-images by one pair, a period is to be added to the high and low logic level periods of each of the above clock signals. The condition of this addition concerns parameters *k* and *c* having their second index equal to '2' except those concerning CLK_1 and CLK_2. These time periods depends also on the layers as mentioned bellow:

- For *Y* layer the period is: Pdt_add_y + Pdt_Dff_y.
- For *X* layer the period is Pdt_add_x + Pdt_Dff_x.

However, these two periods are generally equal because the devices are usually similar. Finally, in CLK_1 and CLK_2 the symbol [[.], [.]] means that the quantity is to be repeated one or many times depending on the number of stored memories.

5-6 Circuit Expansion Hardware Cost

In this section, the cost of the construction of the present ANN circuit (based on the PLP strategy) using the number of gates as a unite scale, is proposed. The adoption of the gates as a unit scale for the estimation is justified by the alternative implementation using the programmable logic devices (PLD) such as PAL, PLA, GAL, PEEL...etc. If this alternative is adopted the cost, in the implemented-components number point of view, will drop drastically because of the PLD relatively high integration.

5-6-1 *X* Layer Sub-Circuit

The consumed components estimation is based on circuit containing one *X* layer sub-circuit neuron connected to '*B*' neurons in the *Y* layer sub-circuit. The assumption that *B* is a pair number is considered. The gates cost of this cub-circuit is defined by the following sum:

[(2 × AND) + (AND + XOR)] × *B* + (2 × 2-bit-binary words adder) × (B_4's) + (2 × accumulator) + comparator + 1-input D Type flip-flop.

2 × AND are the logical multiplication gates between the input signal and both kinds of the synaptic strengths (excitatory and inhibitory).

AND + XOR are the gates needed for the 2 bits summation from the previous stage.

2 × 2-bit-binary-adder are the half adders in the first stage to sum the 2-bit-binary words entering the accumulator.

The accumulator is a cascade of a memory and an adder. In the present circuit, the first is a 4-input D Type flip-flop whereas the latter is a 4-bit-binary adder.

2 × accumulator: two accumulators are needed for every synaptic strength, one accumulator is for the excitatory synapse while the other for the inhibitory.

B_4's = is defined as follow:

$$\begin{cases} \text{If } B = n \times 4 \text{ and } (n \in \aleph) \text{ then } B\_4\text{'s} = n. \\ \text{If } B = n \times 4 + p \text{ and } (n, p \in \aleph) \text{ then } B\_4\text{'s} = n + 1. \end{cases}$$

1-input D Type flip-flop: is a D Type flip-flop of one input [4] bit (as there is one *X* layer neuron) for storing of the present layer output to send it to the next layer.

[4] If the *X* layer has '*A*' neurons, this memory device is required to be *A*-input D Type flip-flop.

5-6-2 *Y* Layer Sub-Circuit

The consumed components estimation is based on circuit containing one *Y* layer sub-circuit neuron connected to '*A*' neurons in the *X* layer sub-circuit. The assumption that *B* is a pair number is considered. The gates cost of this cub-circuit is defined by the following sum:

[(2 × AND) + (AND + XOR)] × *A* + (2 × 2-bit-binary words adder) (A_4's) + (2 × accumulator) × + comparator + 1-input D Type flip flop.

2 × AND are the logical multiplication gates between the input signal and both kinds of the synaptic strengths (excitatory and inhibitory).

AND + XOR are the gates needed for the 2 bits summation from the previous stage.

2 × 2-bit-binary-adder are the half adders in the first stage to sum the 2-bit-binary words entering the accumulator.

Accumulator: is a cascade of a memory and an adder. In the present circuit, the first is a 4-input D Type flip-flop whereas the latter is a 4-bit-binary adder.

2 × accumulator: two accumulators are needed for every synaptic strength, one accumulator is for the excitatory synapse and the other for the inhibitory.

A_4's = is defined as follow:

$$\begin{cases} \text{If A} = n \times \text{four and } (n \in \aleph) \text{ then A\_4's} = n. \\ \text{If A} = n \times 4 + \text{p and } (\text{p} \in \aleph) \text{ then A\_4's} = n + 1. \end{cases}$$

1-input D Type flip-flop: is a D Type flip-flop of one input bit [5] (as there is one *Y* layer neuron) for storing of the present layer output to send it to the next layer.

[5] If the *Y* layer has *B* neurons, this memory device is required to be *B*-input D Type flip-flop.

5-6-3 Synaptic Strength Sub-circuit

For the hardware cost estimation of this sub-circuit, *Y* layer is considered containing '*B*' neurons to be the input of the PLP-memories flow. In addition, the *X* layer is proposed to contain '*A*' neurons. The synaptic strength sub-circuit cost estimation is the following:

(B-input D Type flip-flop) $\times A + A \times B \times$ XOR + (B'tuple bilateral switch) + (A'tuple bilateral switch).

(B-input D Type flip-flop) $\times A + A \times B \times$ XOR: are the gates needed for the generation of the excitatory and inhibitory synaptic strengths.

(B'tuple bilateral switch) + (A'tuple bilateral switch): B'tuple and A'tuple bilateral switch is a bilateral switch, which got '*B*' and '*A*' inputs and outputs respectively. For the control of which association is to be loaded in the network [6]. In the case of dealing with more than four associations, the selector circuit which select the aimed association to be sent to the network is to be redesigned, and then it may differ in its architecture. For this reason, the estimation cost of this part is not considered.

In the following, *X* layer sub-circuit and *Y* layer subcircuit are assumed to have '*N*' *neurons* and '*M*' neurons successively. If one more neuron is added to the *X* layer, one more D type flip-flop has to be added to the synaptic strength sub-circuit beside *M* gates of XOR. However, if one neuron is added to the *Y* layer, *N* gates of XOR have to be added to the synaptic strength sub-circuit.

5-7 Components Consideration

One of the critical parameters of the present implementation is the gates fan-out limit.

[6] In our circuit, another bilateral switchers have been added when trying to store four associations, cascaded with the above mentioned bilateral switchers for the control associations sending.

This limit is required to be large for some components that are shared between the sub-circuits and inside each of them. The components, which are required to have large fan-out limit, are the following:

1. D Type flip-flop used in the Synaptic Strength Sub-circuit.
2. 2 bit-binary adder.
3. Accumulator.
4. D Type flip-flop used for the transmission of the outputs of each layer to the other.

In the case of a low fan-out of the above-mentioned components, the use of buffers for the amplification the shared signals, is unavoidable.

5-8 The Circuit Estimated C oast Diagrams

The following diagrams are an estimated cost of the PCB implementation of the PLP based BAM versus the number of implemented neurons in any layer.

Figure (3) represents the estimated cost of the AND logic gates. Figure (4) is the estimated cost of the XOR logic gates. Figure (5) is the general cost of the two-bit-binary adders. Figure (6) is the general cost of the accumulators equal to the cost of the comparator components.

5-9 Conclusion

The implemented PLP based BAM circuit, composed of six neurons on one layer and four neurons on the other layer, was studied. Figure (8) shows the PCB realisation of this circuit and Figure (9) shows the simulation of the PCB implementation with its different signals and buses.

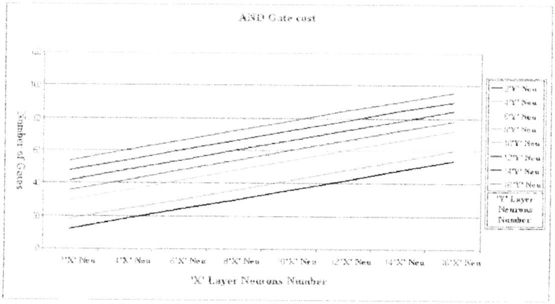

Figure (3). Estimated-Cost Diagram of the AND gates

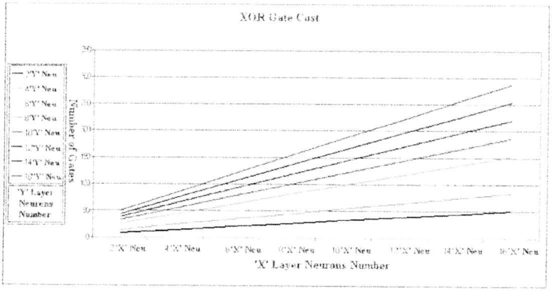

Figure (4). Estimated-Cost Diagram of the XOR gates

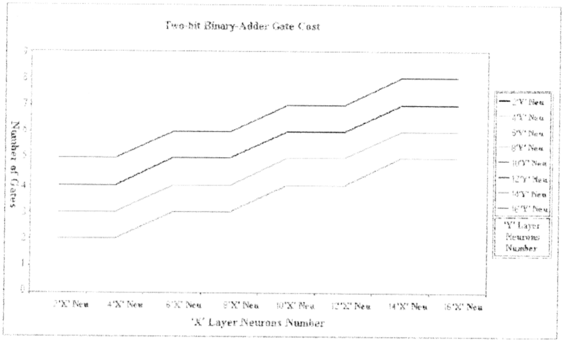

Figure (5). Estimated-Cost Diagram of Two-bit Binary Adder gates

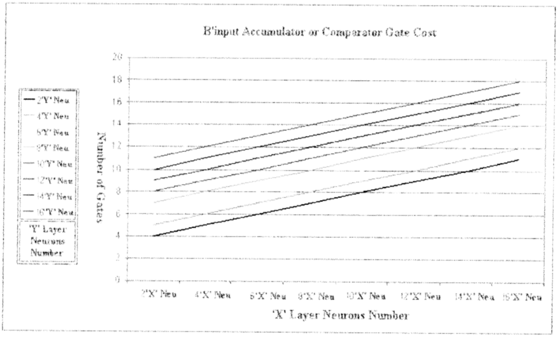

Figure (6). Estimated-Cost Diagram of the Accumulator gates

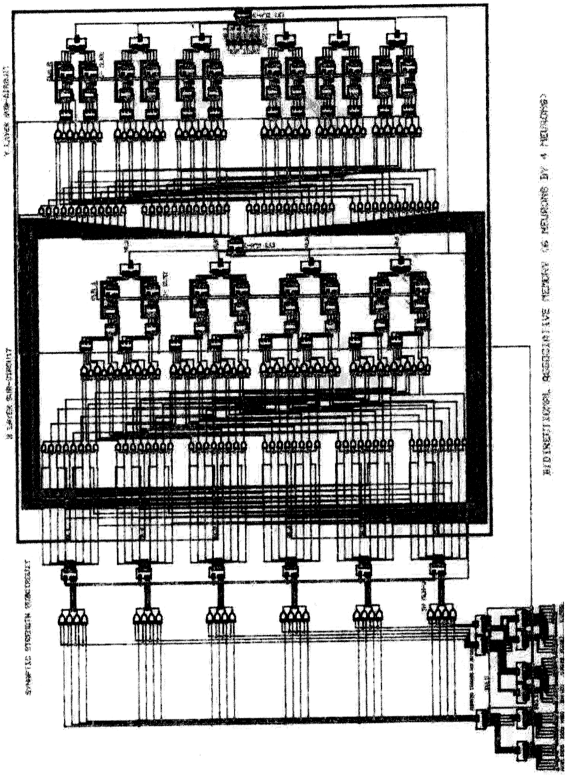
Y LAYER SUB-CIRCUIT
X LAYER SUB-CIRCUIT
SYNAPTIC STRENGTH SUBCIRCUIT
BIDIRECTIONAL ASSOCIATIVE MEMORY (6 NEURONS BY 4 NEURONS)

Figure (7). EED3 Schematic of the BAM 's PLPN-based Circuit
(6 *X* layer neurons to 4 *Y* layer neurons)

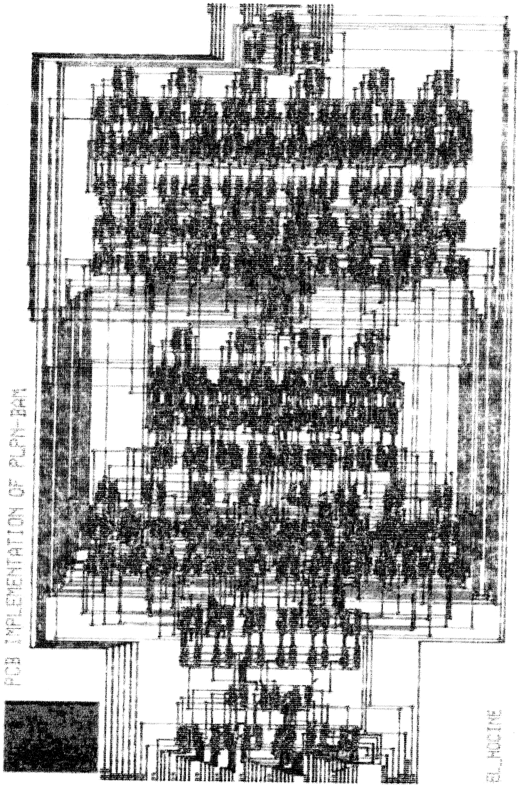

Figure (8). PCB implementation of the PLPN-based BAM Using EED3
(6 *X* layer neurons by 4 *Y* layer neurons)

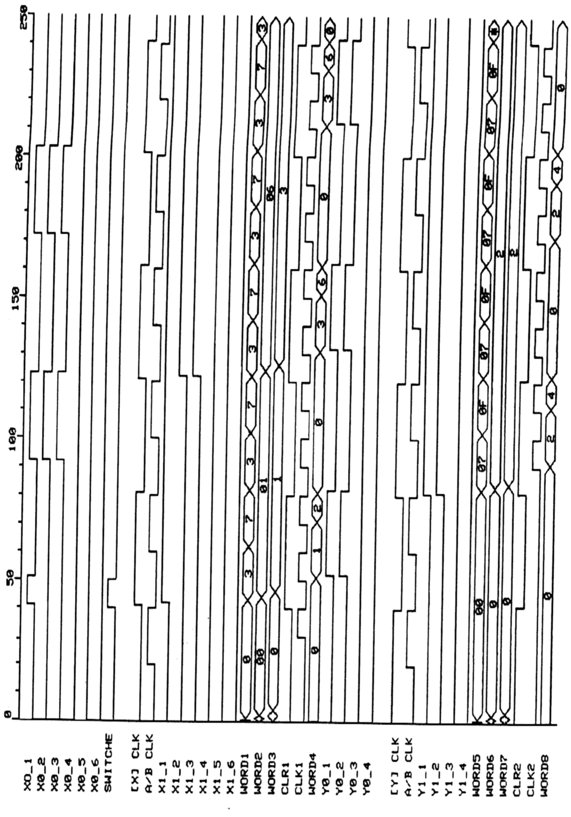

Figure (9). EED3 Simulation of the BAM 's PCB Implementation
(6 X layer neurons by 4 Y layer neurons)

The bad point the PCB technique is the dimension of is layout, which is 70.0 *cm* × 33.0 *cm*. This large consumed area (IC gates too) proves the technique limitation to implement digital neural networks. Consequently, there is no implementation of ANN using the simple IC gates. Furthermore, because of the large area of the PCB, the simple CAD tool such as EEDIII gets its capabilities very limited for the design of artificial neural network circuits. However, the hardware inconvenience may but not totally, be solved by the use of the PLD devices such as PAL, PLA, PEEL …etc. The best medium to implement neural networks is without doubt the Application Specific Integrated Circuit *ASIC* using advanced technologies such as VLSI or ULSI. Another method, but less expensive for the implementation of ANN which is very promising is the programming of Field Programmable Gate Arrays, known as *FPGA*s which can support the complexity of the hardware intended for the implementation of neural networks. The programming of the FPGA is done by the use of a specific programming language known as Hardware Description Language *HDL* designed to simplify the task of the designer implementing complex circuits. Therefor, the use of the HDLs will help ANN designers to overcome the two major problems: the design modelling and the hardware implementation.

In the next chapters, other ANN implementations but more economic and flexible are implemented using one of the most powerful HDL languages namely VHDL with different electronic architectures.