A&QT-R 2004 (THETA 14)
2004 IEEE-TTTC – International Conference on Automation, Quality and Testing, Robotics
May 13 – 15, 2004, Cluj-Napoca, Romania

USING NEURAL NETWORKS IN INTEGRATED NUMERICAL CIRCUITS AND SYSTEMS

Remus Joldeş

Departament of Informatics, „1 Decembrie 1918" University of Alba Iulia, Nicolae Iorga Street, 11-13, Cod 510003, Alba Iulia, Romania, rjoldes@uab.ro

Abstract. The paper presents main aspects regarding the use of neural networks in integrated numerical circuits and systems viewed as finite state systems FSS. Beyond the theoretical aspects as concerns the use of neural networks in numerical systems, a method having the role of design guide is also presented for the design and achievement of such FSS, in this new approach.

Keywords: neural networks, finite state systems, design methods for finite state systems, simulation using neural networks.

1. Premises

The issue of digital circuits and systems is generally treated by the general theory of finite state systems (FSS). Based on this theory we will treat the numerical integrated circuits and systems by using neural networks for their simulation.

All types of numerical systems can be obtained from the sequential logic system presented in figure 1.

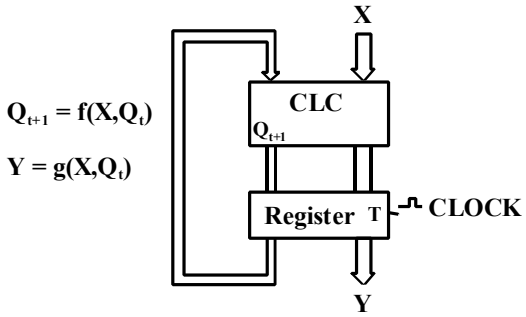

Figure 1. The Logical Sequential System

The functional behavior of this sequential logic system can be described by two functions **f and g**, named characteristic functions. These functions show the evolution

of **Y** outputs and the states of the system **Q** at the t+1 moment, depending on the system's state at the moment **t** and the system's inputs **X**.

The system's characteristic functions can be explicitly expressed by: graphs, transaction tables, flow charts, wave form and their translations (each with its own theoretical nuances and/or practical approach). By using neural networks in the study of such systems, the combinative hazard is actually excluded. The behavior of the combinative logic circuit **CLC** is not time depending and it can be described by Boolean functions or binary algebra functions.

Boolean function can be explicitly expressed by logic tables of truth canonical forms **S** or **P**, Veitch-Karnaugh diagrams, Quinne Mc Cluskey etc. The most difficult problem in the classical paradigm of this systems' approach is the minimization of the electronic system for the implementation of the given system, minimization which means to find the minimum functional system equivalent with the given one. In the case of the approach by neural paradigm the problem of minimization is implicitly solved by the conceptual philosophy of the neural networks, itself. In the classical paradigm applied to the approach of these systems the following minimization methods are used:

- Analytical method by using theorems of the Boolean algebra, starting from the logic expression of the characteristic functions and application of the Boolean theorems until the minimized forms are obtained;
- Reduction of the system's state space by the partitioning of space into equivalence classes;
- Minimization of the combinative logic circuit associated to the system by one of the graphical methods of the Veitch-Karnaugh or Quinne Mc Cluskey diagrams.

In the classical paradigm of these circuits' approach, starting from the general theory of the finite state systems FSS, the mathematical modeling conduct to:

- Classification of the digital systems;
- Mathematical description of the digital systems' behavior;
- Establishment of the general design methods.

In general, any FSS can be mathematically descripted by a quintuples like:

$$S = (X, Y, Q, f, g)$$, where:

- $X\{x_1, x_2, ..., x_p\}$ - represents the set of input variables in the FSS ;
- $Y\{y_1, y_2, ..., y_m\}$ - represents the set of output variables of the FSS;
- $Q\{q_1, q_2, ..., q_n\}$ - represents the set of states of the FSS;
- $f : XxQ \to Q$ - represents the transition function of the states, this function defines the modifying process of the states in a FSS, being dependent on the inputs and the system's former state;
- $g : XxQ \to Y$ - represents the transition function of the outputs. In the Mealy model, g **function** shows that the outputs are modified according to the input functions of the FSS and the former state of the system. In Moore model, g function shows that the outputs depend only on the former states, that is : $g : Q \to Y$. There is a rigorously, mathematically demonstrated theorem which says that the two models (Mealy and Moore) are equivalent.

The system described by this quintuple is maned with finite states whilst X, Y and Q are finite therefore discrete. The realm of time which doesn't explicitly appear in the definition is also formed by the se of integer numbers $T\{t_1, t_2, ..., t_{N+1}\}$ where : $t_1 = 0$, $t_2 = 1$, $t_3 = 2$, …, $t_{N+1} = N$ and which correspond to the appearance of synchronizations impulses of the tact generator.

The characteristic functions f and g define the evolution of the FSS in time. These functions are represented by: transaction tables, transaction graphs and flow charts.

Transaction tables contain input variables in columns and components of the state matrix in lines. At the intersection of each column and line, the state and the next output are written, that is the value of function f and the value of function g separated by a comma, as in figure 2.

Using these transition tables the transition graphs and the evolution flow charts of the FSS can be drawn. FSS are completely described by these transition tables. In the case of the transition graphs, $q_1, q_2, ..., q_n$ states will represent the nodes of the transition graph and the x_i, Y_j tuple will represent the graph's arches and the state variables which will change the system's state.

Q/X	x_1	x_2	…	x_i	x_{i+1}	…	x_p
q_1		q_1, Y_1	…	q_j, Y_i		…	
q_2	q_2, Y_4		…			…	
…	…	…	…	…	…	…	…
q_j			…		q_1, Y_{i+1}	…	
q_{j+1}			…	q_{j+1}, Y_j		…	
…	…	…	…	…	…	…	…
q_n	q_1, Y_1		…			…	

Figure 2. Transition table

In this way the transition graphs contain the states of the FSS in nodes and the arches represent the determinations of inputs and outputs as in figure 3. For example, if S system is in q_1 state and the x_2 input has been applied, system S remains in the same state, if x_i input has been applied the system S passes into state q_j and if x_3 input has been applied the system passes in state q_3, state q_1 is reached when the system is in state q_n and the applied input is x_1 , etc.

The transition flow charts of states (figure 4) contain in rectangles the states of system and the outputs of the system adjoining. The inputs x_i are the testing conditions which establish the system's evolution.

Thus if the system is in state q_1 and has x_2 as input, it will remain in state q_1 and output Y_1, if in state q_1, x_3 has been applied as input the system passes in state q_2 having the output Y_4, and if in state q_1 , x_i is applied as input the system passes in state q_j having the output Y_i and so on.

The classical paradigm of the approach of the FSS shows that such a system can be mathematically described by : $S = (X, Y, M)$, where:

- X - represents the input space of the system;
- Y - represents the output space of the system;

- $M : X \rightarrow Y$ - represents the input-output function of the system (in case of electronic systems, function M is explicitly expressed by the wave forms obtained by oscillography).

The definition by quintuples is extremely useful in the analyses and synthesis of systems, and the definition by triplets in experiments and system diagnosis.

2. Simulation of numerical systems by using neural networks

Due to the perfect functional similitude between neural networks of feed-forward type and numerical systems we shall see if it is possible to simulate numerical circuits and systems by using neural networks. Based on our research and the studied bibliography we have concluded that the fact is completely achievable. For the general case we propose the following simulating network of the integrated numerical circuits and systems presented in figure 5.

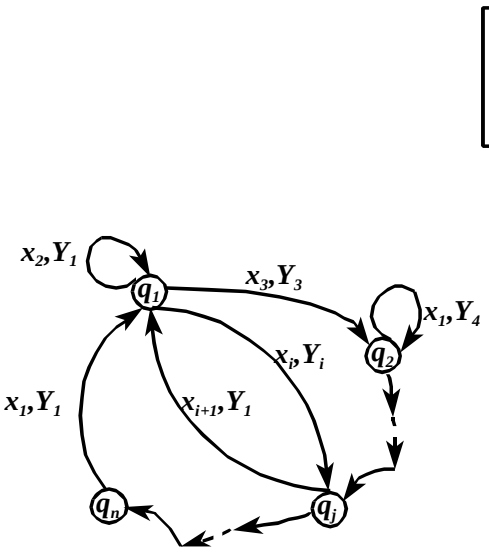

Figure 3. Transition graph of a FSS

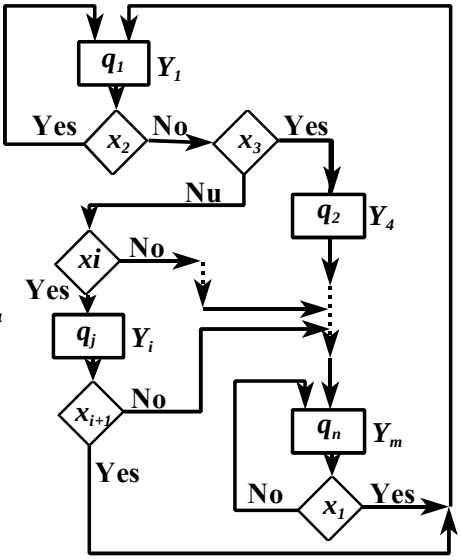

Figure 4. Transition flow chart of states

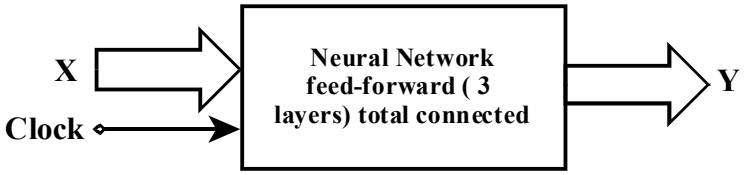

Figure 5. Neural network for the simulation of numerical circuits and systems

The neural networks inputs are represented by the set of input variables in system FSS $X\{x_1, x_2, ..., x_p\}$ and by the **Clock**. The presence of the tact signal is absolutely required for the synchronization of the stimulated system's state change. The neural networks outputs are represented by the set of output variables of the FSS, that is $Y\{y_1, y_2, ..., y_m\}$.

For the network's training, specialized software, known in the literature, can be used, software which doesn't occur any problem in exploitation.

After the network has been well trained and checked the next phase is the hardware implementation and testing in the new functional conditions. As regards the steps of design we propose the following design method:

- **Step 1**: Establishment of the truth table for the description of the FSS functioning and graphical achievement of the transition flow chart of system's states.
- **Step 2**: Selection of the neural network of feed-forward type, totally connected, with 3 layers, establishment of neurons in the input layer, output layer and the hidden layer. The number of neurons in the input player has to be equal with **p+1** ($x_1, x_2, ..., x_p$ and tact), in the output layer should be **m** ($y_1, y_2, ..., y_m$), and for the hidden layer we have concluded that the number is: **(p+m+1)*2/3**.
- **Step 3**: Training of the chosen neural network by using specialized software e.g. Mathlab or WinNN.
- **Step 4**: Checking of the correct functioning of the obtained network. Checking should be done as regards correct functioning, speed and tolerance at the breaking of a bound. If problems occur, the number of neurons in the hidden layer should be changed (increased or decreased) and step 3 should be redone.
- **Step 5**: Hardware implementation of the neural network which stimulates the FSS.

3. Tests and practical applications

For practical application I used an older project that was applied at IMMN Zlatna Factory. This real application was developed by a team lead by myself. The logical diagram for the function of this FFS is presented in figure 6.

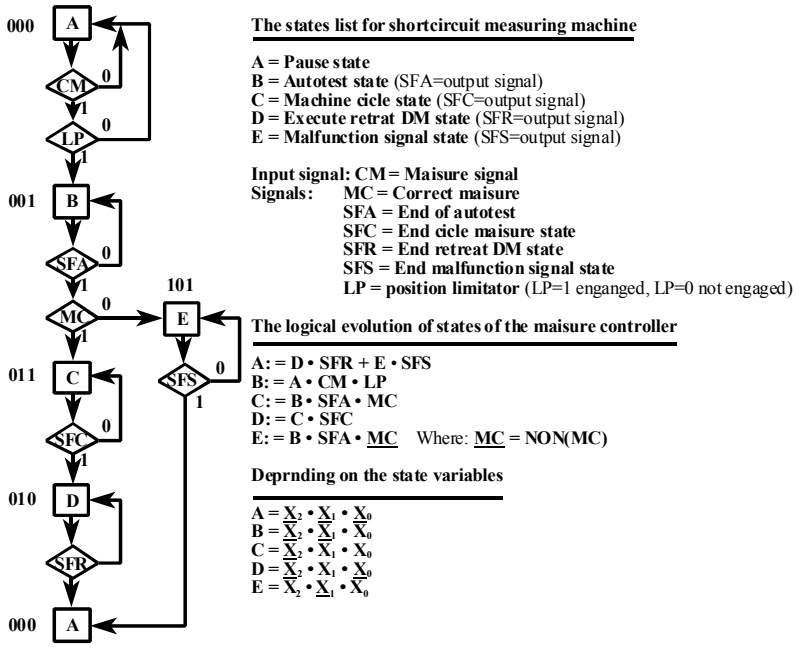

Figure 6. The logical diagram for practical example FSS

For this particular case, the neural network must accept the input of the following binary variables CM, LP, SFA, MC, SFC, SFS, and SFR and to output the following binary variables A, B, ... , E. The neural network behaves like a classical associative memory, who's training does not constitute a problem and which can be trained using Standard products (E.g. MathCad or WinNN). Our results can be obtained very easily, because after setting the truth table, the network's training with any of the above mentioned products takes under a minute. The network's property of associative memory excludes the output of any wrong variable. The input and output variables of the truth table constitute the training input and output patterns.

For each of the states of system the corresponding logical diagram can be created and also it's truth table for the corresponding sub-states. For the implementation of the FSS with the use o neural networks, a neural network of feed-forward type was used witch three layers total connected. This network has 17 input neurons (3 for the states generator, 2 for B state, 3 for C state, 2 for D state, 1 for D state, 1 for E state, 1 for clock generator and 5 for the position limits), 12 neurons in the hidden layer and 9 neurons in the output layer (3 for the controller engine command, 4 for the measure controller command, 1 for display the read values and 1 for permission of relocation of DM). The training of the neural network was done by the use of WinNN product.

4. Conclusions

We consider that due to the advantages of the neural networks, more and more electronic circuits, which are classically treated can have neural implementations, which are more simple and more reliable. This approach can conduct to the achievement of complex neural finite state automates, as first stage in the achievement of neural computing systems. The originality of the current paper resides in announcing the new method for the design which is represented by the five steps (paragraph 2) and which can be applied in the new paradigm for FSS design.

References

[1] PARK YOUNG-MOON, CHOI MYEON-SONG, LEE KWANG Y.: *An Optimal* ***Tracking Neuro-Controller for Nonlinear Dynamic Systems***, IEEE Transactions on Neural Network Vol.7, No. 5, September 1996, p. 1099-1110.

[2] JOLDEŞ R., ILEANĂ I., ROTAR C.: ***Utilization of neural networks in the simulation of combinational logical circuits***, Acta Universitatis Apulensis, Mathematics-Informatics, Alba Iulia, No. 3, 2002, p. 65-68.

[3] BIENNIER F.: ***Connexionisme: Principes et elements theoretiqe***, p. 7-49, I. N. S. A. Lyon, 1994.

[4] AMARI SHUN-ICHI: ***Mathematical Foundations of Neurocomputing,*** Proceding of the I.E.E.E., Vol.78, No. 9, September 1990, p. 1443-1463.

[5] GELEMBE, E.: ***Theory of the Random Neural Network Model***, p. 3-20, in vol. NEURAL NETWORK: Advances and Applications, Editor: Gelenbe, E., North-Holand, 1991.