# APPLICATION OF NEURAL NETWORKS AND SIMULATION MODELING IN MANUFACTURING SYSTEM DESIGN

**Kenneth LeCroy**
**Lucent Technologies**
**9333 S. John Young Parkway**
**Orlando, FL 32819**

**Mansooreh Mollaghasemi, Ph.D.**
**Department of Industrial Engineering**
**University of Central Florida**
**Orlando, FL 32816**

**Michael Georgiopoulos, Ph.D.**
**Department of Electrical and Computer Engineering**
**University of Central Florida**
**Orlando, FL 32816**

## ABSTRACT
We demonstrate how neural networks can be used in conjunction with simulation modeling for system design. This approach is used to achieve the opposite of what a simulation model can achieve: given a set of desired performance measures, the neural network outputs a suitable design to meet management goals. The methodology is presented using a real world application involving the Test Operation of a major semiconductor manufacturing facility.

**KEYWORDS:** Neural Networks, Simulation, System Design, Semiconductor Manufacturing.

## INTRODUCTION
Manufacturing system design is a complex process involving the gathering of data from many sources and the modeling of the interrelationships among the proposed system's entities and between the system's entities and its environment. Typical decisions that must be made during the design process include the types of parts/end items to be produced, the production rates desired, the amounts of resources (e.g., machine tools, pallets, fixtures, and material handling devices) of various types to include in the system, the system layout, and routing and scheduling decisions (Mollaghasemi and Evans, 1994).

The goal of the design process is to choose the design that yields the "best" set of values for the performance measures. This is difficult for several reasons, including the uncertainties associated with the input data to the process (e.g., product demands), the complex relationships between the design variables and the performance measures, and the multiple, conflicting performance measures that must be considered when choosing the "best" design. For example, system utilization may be increased at the expense of increased work in-process inventory (Evans, Stuckman, and Mollaghasemi, 1991).

Quantitative approaches have been proposed to solve the manufacturing design problem (see for example, Buzacott and Yao, 1986, or Kusiak, 1986). Unfortunately, mathematical programming formulations can not account for factors that contribute significantly to the performance of manufacturing systems such as machine breakdowns and resource loading conflicts.

Simulation models, on the other hand, have been used to accurately evaluate the performance of complex manufacturing systems. The popularity of simulation modeling is due to its flexibility, its ability to model systems more precisely, its capability to perform what-if analysis, and its ability to model the time dynamic behavior of systems. Simulation, however, in and of itself, is not a design tool. Simulation models can only give an evaluation (in terms of the performance measures) of a design that is input to the system. System design by simulation is therefore an ad hoc process which is highly iterative and as a result very time consuming.

The objective of this research is to demonstrate how artificial neural networks can be used in conjunction with simulation models to provide a decision support system that eliminates the trial and error

**ISBN# 0-7803-3268-7**

process from the manufacturing system design. The role of this decision support system is the opposite of what the simulation model can achieve: given a set of desired performance measures (e.g., throughput, machine utilization, cost), the decision support system outputs a suitable design (e.g., scheduling rules, the number of resources of each type) to achieve those objectives. In other words, the system suggests a well-suited alternative which accomplishes management objectives under the given set of operating conditions. Little research has been cited in the literature about the use of neural networks in conjunction with simulation models (see for example, Pierreval and Huntsinger, 1992, Kilmer and Smith, 1993 and 1994 and Chryssolouris, Lee, Pierce, and Domroese, 1990). Our research was motivated by Chryssolouris et. al. (1990) who propose the use of neural networks and simulation modeling for the design of manufacturing systems.

## METHODOLOGY

In this section, the proposed methodology is illustrated in the context of the application problem. The application involves the Test Floor Operation of a major semiconductor manufacturing facility.

### Simulating the Test Floor Operation

The first step in the development of the decision support system involved the modeling of the Test Floor Operation. The model of the system was developed using SIMAN (Pegden, Shannon, and Sadowski, 1990) general purpose simulation language. Prior to the development of the simulation model, interviews were conducted with the test floor supervisors, process analysts, and test floor operators. In addition, significant time was spent on the test floor to gain additional insight into how the test floor operated. Upon completion, the simulation model was validated to insure the accurate representation of the actual system.

### Generation of Training Data

The next step involved generating the input-output patterns from the simulation model to be used in training the neural network. The inputs to the simulation were identified as the number of testers of each type (three types of testers were selected) and the queuing strategies used to process the lots on each tester. The possible values for the number of testers were 1, 2, or 3 testers and the queuing strategies were considered to be first in first out, shortest processing time, highest demand, or lowest slack The performance measures of the simulation were identified to be the cycle time, and work-in-process (WIP).

There are 108 (4x3x3x3) possible combinations of the input parameters. From the 108 combinations, 40 combinations were randomly selected for training the neural network. Several replications were run at each of the 40 experimental point. A set of performance measures consisting of the cycle time, and work-in-process were collected at the end of each simulation. Note that the number of training pairs were arbitrarily selected as 40 with the idea that if the results were unsatisfactory, we could always add more training pairs to the training set.

### Training of the Neural Network

The data used for training the neural network included inputs, consisting of the two performance measures mentioned above (i.e., the output of the simulation model), and outputs, consisting of the number of each type of tester and the queuing strategies used to process the lot (i.e., the input to the simulation model). Note that the output of the simulation model is used as the input to the neural network (and vice versa) because the objective here is to do the reverse of the simulation model. That is, given a set of desired performance measures (e.g., cycle time, WIP), we want the decision support system to output a suitable design, (e.g., scheduling rules, the number of resources of each type) to achieve those objectives.

A neural network package known as XERION was used in training the data. This package was developed by Drew van Camp, Tony Plate, and Geoffrey Hinton at the University of Toronto. XERION allows the user to design, train, test, and use a host of different types of neural networks. These networks differ in architecture, learning algorithms, and types of training paradigms. In this project, a multi-layer feedforward neural network with a back-propagation learning algorithm was selected for use. This approach was selected because of its excellent track record in successfully solving a variety of application problems.

A network consisting of an input layer, a hidden layer, and an output layer was set up. A 2-10-4 architecture was selected for the neural network. These numbers signify the number of nodes in the input layer, the hidden layer, and the output layer, respectively. The number of nodes in the input and the output layers were selected to match the number of inputs and outputs respectively. The number of nodes in the hidden layer, however, was not dictated by the problem at hand and was arbitrarily selected. Note that if the number of nodes in the hidden layer are too large, the network will not generalize well while if the number of nodes in that layer is too small, the algorithm will not converge (Lippmann, 1987).

Prior to training the neural network, the input/output pairs of the training set were transformed into appropriate binary input/output pairs. In order to represent the network input values, several ranges were defined for each input. Five ranges were defined for each performance measure and each input value was defined as belonging to a certain range. In this manner, if an input value fell into one of several defined ranges, the node in the network which represented that particular range for that particular input value received a value of 1. All other nodes which represented that input variable would have a value of 0. For example, if the cycle time ranges from 1550 time units to 2050, and five ranges are defined for the cycle time with each range including 100 time units, then a value of 1800 can be represented as 00100. Similarly, the values of training output can be converted. For example, if the number of units of a resource can take on a value of 1, 2, or 3, a value of 2 can be represented as 010. Examples of the binary representation of the input and output is shown in Tables 1 and 2.

Using the above process, the input and output values of the 40 training pairs were converted into the binary format. In addition, network parameters such as training tolerance, learning rate, momentum, and initial interconnection weights were selected. For this study, a network tolerance of 10% was selected. That means the network is trained until

$$\left|\frac{\textit{Desired Output - Actual Output}}{\textit{Desired Output}}\right| < 0.1 .$$

The neural network was then trained successfully with the 40 training pairs and achieved a training tolerance of 10%.

**Table 1. Conversion of an example training input**

| Type of output | Value | Binary Representation |
|---|---|---|
| Cycle Time | 1,800 | 0 0 1 0 0 |
| WIP | 550 | 1 0 0 0 0 |

**Table 2. Conversion of an example training output**

| Type of Input | Value | Binary Representation |
|---|---|---|
| Queuing Strategy | SPT | 0 1 0 0 |
| No. of Type A-1 Testers | 1 | 1 0 0 |
| No. of Type A-2 Testers | 3 | 0 0 1 |
| No. of Type A-3 Testers | 2 | 0 1 0 |

**Testing of the Neural Network**

In order to begin the testing process, an initial vector of desired system outputs( in terms of cycle time and work-in-process) was formed. This test case was suggested by the Manager of Test Operation. Note that this vector was not used in training the network. The values of these performance measures were first converted into the binary format as shown in Table 3.

**Table 3. Desired system outputs used for testing the neural network**

| System Output | Desired Output | Binary Representation |
|---|---|---|
| WIP | 750 | 0 1 0 0 0 |
| Cycle Time | 1,700 | 0 1 0 0 0 |

These desired outputs were then fed as input to the trained network and suggested levels of simulation input were obtained. The decision support system predicted the following values to achieve the desired output:

**Table 4. Simulation inputs suggested by the neural network**

| Simulation Input | Suggested Value |
|---|---|
| Queuing Strategy | Lowest Slack |
| No. of A-1 Testers | 2 |
| No. of A-2 Testers | 1 |
| No. of A-3 Testers | 3 |

The next step was to test the accuracy of the proposed approach. In order to achieve this, the "suggested" vector of input values shown in Table 4, was fed back into the simulation model and the simulation was run. The resulting average performance measures from running the simulation. model are displayed in Table 5.

**Table 5. Comparison of the actual and desired system outputs**

| Performance Measure | Desired Output | Simulation Output |
|---|---|---|
| WIP | 700 | 749 |
| Cycle Time | 1,700 | 1,632 |

The output from the simulation was then compared against the desired output and an error was calculated in order to assess the performance of the network. The error was computed as:

$$Error = \sum_{i=1}^{n} \left| \frac{Desired\ Output - Actual\ Output}{Range} \right|$$

where

$n$ = number of outputs, and

*Range* = the span of values covered by a single node for a given output.

The following example illustrates the calculation of the error associated with the test case shown in Table 5:

$Error$ = |749-700| /100 + |1632-1700| /100 ,

$Error$ = 1.17.

Currently, in order to meet production goals, supervisors make recommendations as to how to run the system. To assess the usability of the neural network as a decision making tool, the performance of the neural network was compared with best guesses by shop floor supervisors who do this type of analysis on a day to day basis. The problem was posed to two supervisors. Each person suggested a configuration to achieve the manager's objectives. The supervisor's recommended configuration was simulated and mean performance measures were obtained. An error was then computed for each suggested solution to assess the proximity of each solutions to the desired output. The comparison of these solutions are shown in Table 6.

**Table 6. Comparison of Neural Networks and other Methods**

| | WIP | Cycle Time | Error |
|---|---|---|---|
| Target | 700 | 1,700 | |
| | | | |
| Neural Network | 749 | 1,632 | 1.17 |
| Best Guess 1 | 840 | 1,790 | 2.30 |
| Best Guess 2 | 778 | 1,823 | 2.01 |

As it is shown in the above table, the neural network solution resulted in the lowest error. In addition to the test case shown above, 20 other cases were tested. The neural network performed best in 11 of the 20 test cases. Additionally, the times that the network did not perform the best, it was a close second to the best solution.

## CONCLUSIONS

We have presented preliminary results from the integration of neural networks and simulation modeling for the design of manufacturing systems. This approach is demonstrated using a real world problem. Given the results of this research, the use of neural networks with simulation modeling shows a great deal of potential as a tool for system design. We have shown that the methodology can be used with problems where multiple inputs and objectives are present.

## REFERENCES

Buzacott, J. A., and Yao, D. D. (1986) Flexible manufacturing systems: A review of analytical models. Management Science, Vol. 32, 890-905.

Chryssolouris, G., Lee, M., Pierce, J., and Domroese, M. (1990) Use of neural networks for the design of manufacturing systems. Manufacturing Review, Vol. 3, No. 3, 187-194.

Evans, G. W., Stuckman, B., and Mollaghasemi, M. (1991) Multicriteria optimization of simulation models. Proceedings of the Winter Simulation Conference, 894-900.

Kilmer R. A. and Smith A. E. (1994) Neural networks as a metamodeling technique for discrete event stochastic simulation. Proceedings of Artificial Neural Networks in Engineering, St. Louis, Missouri.

Kilmer R. A. and Smith A. E. (1993) Using artificial neural networks to approximate a discrete event stochastic simulation model. Intelligent Engineering Systems Through Artificial Neural Networks, Volume 3, ASME Press, New York, 333-341.

Kusiak, A. (1986) Application of operational research models and techniques in flexible manufacturing systems. European Journal of Operational Research, Vol. 24, 336-345.

Lippmann, R. P. (1987) An Introduction to Computing with Neural Nets. IEEE ASSP Magazine.

Mollaghasemi, M. and Evans, G. W. (1994) Multicriteria design of manufacturing systems through simulation optimization. IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 9, 1407-1411.

Pegden, C. D., Shannon, R. E., and Sadowski, R. S. (1990) Introduction to Simulation Using SIMAN. Second Edition, McGraw-Hill, New York.

Pierreval, H. and Huntsinger R. C. (1992) An investigation on neural network capabilities as simulation metamodels. Proceedings of the 1992 Summer Computer Simulation Conference, 413-417.