# DESIGN OF AUTOMOTIVE'S COMPLEX ELECTRONIC SYSTEM BASED ON SOM CLUSTERING

**Stephan Brummund, Philipp Nenninger, Simmi Saxena, Uwe Kiencke**
Universität Karlsruhe (TH)
Institute of Industrial Information Technology
Karlsruhe. Germany
**brummund@iiit.uni-karlsruhe.de, nenninger@iiit.uni-karlsruhe.de,**
**kiencke@iiit.uni-karlsruhe.de, simmi_saxena@yahoo.co.in**

**Abstract -** *This paper presents an approach of designing complex electronic system of automotive by clustering control units using self organising maps (SOMs). SOMs are neural networks based on competitive learning in an unsupervised manner. The communication between various control units is modeled and taken as input for the SOM. Control units are combined together later on the basis of clusters obtained from the trained SOM. A different approach of automatic determination of the units belonging to various clusters based on image processing is also proposed.*
*The proposed architecture is further verified using a CAN-bus model.*

**Key words - electronic control units, clustering, som, max-plus algebra, CAN-bus**

## 1 Introduction

Due to the enormous increase in the number of Electronic Control Units in automobiles over the past few years for controlling various electrical and electromechanical functions, the design process of its electronic system has become highly complex. The use of automated tools for design, implementation and testing is therefore increasing. At the moment ECU is developed with its own internal system structure and there is little integration among different ECUs in this system architecture. Each ECU acts autonomously and is almost independent of all other ECUs in the network as shown in figure 1. Information exchange between ECUs is performed via a communication system with relatively low communication speed, typically with a Controller Area Network (CAN) up to 500 kbit/s [2].

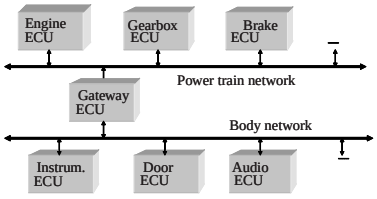



Figure 1: Typical Automotive Electronic Architecture

In future systems, complexity will be further transferred from mechanical domain to software and

electronics [3]. But it is expected that the number of ECUs will not grow drastically, simply because it becomes too expensive and heavy to have separate hardware (HW) for each functionality and also the available space in the vehicle is limited. Therefore, ways must be found to integrate software (SW) modules that participate in the implementation of different functionality in the same ECU. Next-generation functions require a shift from loosely coupled ECUs to an integrated electronic architecture to reduce the overall bus load. The basic requirement is modularization of the entire system. When developing distributed systems, engineers face the trade-off between partitioning the system early in the development process in order to be able to independently develop the subsystems while losing important degrees of freedom and developing a large monolithic system postponing the partitioning until the end of the design process. Tools are not yet widely present in the area of the transformation of a complex, monolithic system given in a design tool such as Simulink into a distributed system of several controllers connected by a communication bus. Loosely coupled ECUs are state-of-the-art. In order to efficiently partition a given automotive electronic system either during the development process or dynamically during run-time, a clear notion of atomic functional objects ($FOs$) is needed. The $FO$ is a smallest distributable function on an ECU. The modeling of the temporal behavior of distributed electronic systems in $n$-dimensional space using Max-Plus algebra has been demonstrated in [2] . In recent years, research has been focusing on the partitioning of a given system at the end of the development process rather than partitioning a system solely based on its specifications. Thus the fundamental problem in the automotive's electronic system design is in distributing a pool of functions or $FO$ over the target architecture to satisfy cost, safety, and real-time operating requirements. SOMs are neural networks based on unsupervised learning, which is also the demand of problem domain. Thus partitioning of the basic architecture in the design of complex electronic systems, by allocation of various $FOs$ to different hardware units is done based on SOM clustering. If the number of units on which $FOs$ are to be mapped is fixed, the problem can as well be attacked as a partitioning problem. But then, this will decrease the efficiency of the design. Therefore the problem can be better attacked by the clustering approach where the number of partitions in which the data is to be divided is also decided by the algorithms itself.

# 2 Self Organising Maps

## 2.1 Introduction

Self Organising Maps or SOMs are one of the most realistic models of the biological brain function. SOMs are neural networks introduced by Kohonen [5] and are among the best-known unsupervised learning ANN (Artificial Neural Networks). SOMs belong to class of vector coding algorithm. It possess the properties of vector quantization and topological mapping.

## 2.2 The Components

### 2.2.1 Input Layer of the SOM

Each input vector $x$ of SOM, represents a data point in this $n$-dimensional input space $X$. The input layer of SOM consists of as many neurons as the number of features in the input vector. The idea of the self-organizing maps is to project the $n$-dimensional data into something that can be better understood visually.

### 2.2.2 Output Layer of the SOM

The output layer of SOM consists of an array of neurons. Let $O(t)$ represent the output space with dimension always less than that of input space. The number of neurons in the output layer of SOM is still a matter of research. A lesser number of neurons may lead to inferior results but will have fast processing speed. However a large number of output neurons increases the efficiency of the algorithm but will decrease the speed and may also lead to an unclustered map.

### 2.2.3 Weights

Weights represent connections from input neurons to the output neurons and are the most important part of SOM structure. A weight vector $c_{i,j}(t)$ represents a vector connecting all the input neurons to the $(i,j)_{th}$ unit in the output layer. Initial value of the weight vector can be chosen in various ways.

- Random Initialization
- Initialization using input vectors
- Gradient Initialization

## 2.3 The Algorithm

The algorithm is a competitive learning algorithm, which means learning is enforced by competition among the neurons [4]. Competitive learning is an adaptive process in which the neuron which best matches a given input will be rewarded by becoming more like the input. Nearby neurons in the output map gradually become more sensitive or get specialized to certain categories of input data. Training SOMs is basically mapping of each input vector $x$ of the input space $X$ to a neuron in the output space $O(t)$. If we have $M$ vectors of $n$-dimension to be clustered then an input vector is one among these $M$ vectors chosen randomly.

The algorithm consists of the following steps:

1. Choose an input vector $x_m$ from the input space
2. For each input vector $x_m$ find the best matching unit (BMU) $o_{max}(t)$ from the output layer $O(t)$. Usually an Euclidean metric is used to determine the BMU $o_{max}(t)$. If $c_{max}(t)$ represents the weight vector connecting input neuron layer to the BMU $o_{max}(t)$ in the output layer $O(t)$ then

$$c_{max}(t) \quad = \quad \min_{\substack{0\le i\le a \\ 0\le j\le b}} \left\{ \left\| x_m - c_{i,j}(t) \right\| \right\} = \min_{\substack{0\le i\le a \\ 0\le j\le b}} \left\{ \sqrt{\sum_n \left(x_m - c_{i,j}(t)\right)^2} \right\}$$

3. For each node of the output layer, adjust its weight vector according to

$$c_{i,j}(t+1) = \left((x_m - c_{i,j}(t)) \cdot \alpha(t) \cdot F_{c_{max}}(t)\right) + c_{i,j}(t); \\ \forall\, i \le a \quad and \quad \forall\, j \le b$$

   Where,

(a) $\alpha(t)$ is known as 'Adaptation Coefficient' or 'Gain factor'and its value decreases monotonically with $t$ as:

$$\alpha(t) = \dot{\alpha}_0(1 - t/T)$$

(b) The $F_{c_{max}}(t)$ represents the neighborhood kernel of $o_{max}(t)$

$$F_{c_max}(t) = \exp\left(-\frac{\|r_{max} - r_{i,j}\|^2}{2 \cdot \sigma^2(t)}\right)$$

Here,

i. To insure that the clusters do not blur in the later stages of the clustering process

$$\sigma(t) = \dot{\sigma}_0(1 - t/T)$$

ii. The $r_{max}$ and $r_{i,j}$ represent the positions of the BMU and the $(i,j)_{th}$ unit on the output grid and $||r_{max} - r_{i,j}||$ represents the distance between them.

Thus neighborhood kernel is both a function of $t$ and the distance of the neuron from the BMU. It decreases monotonically with $t$ and with increasing distance. Thus we have,

$$0 \leq F_{c_{max}}(t) \leq 1 \quad \forall\, c,t$$

This is to ensure that the areas with neurons which are not well adapted to the current input vector should not get surrounded by well adapted areas, thus breaking the cluster. For the BMU where $c = c_{max}$, to completely adapt to the input vector

$$F_{c_{max}}(t) = 1 \quad \forall\, t$$

4. Repeat all the 3 steps above for all the $M$ input vectors, i.e.

$$\forall\, m \leq M$$

5. Repeat the four steps above for all the $T$ iterations, i.e.

$$\forall\, t \leq T$$

# 3 Clustering of Functional Objects

Several visualization techniques are used to extract the data information from the clusters of the trained SOM. The most commonly used is the U-Matrix or Unified Distance Matrix method [4]. Another visualization method known as Histogram method is used to display the number of hits on each map unit. Based on the visualization, clusters can be selected manually and the data points belonging to a particular cluster can then be extracted. However this approach is tedious and more prone to errors as determination of cluster boundaries is totally based on personal perspectives. Therefore the process should be automated.
For determining cluster boundaries automatically, a different approach is being attempted. Both the U-Matrix data and the Histogram data along with some image processing techniques are used to make process efficient and faster. The automatic evaluation of clusters and determination of the data points belonging to them, (which is also essential for the automated design tool) could be done as follows:

1. Obtain U-Matrix and Histogram of the trained SOM.
2. Use edge closing filter over the U-Matrix to make the cluster boundaries appear more sharper.
3. Determine a suitable threshold $\theta$ for this image, which can represent the cluster boundaries reasonably well. Obtain a binary image $b$ of the U-Matrix filtered gray scale image, using this threshold $\theta$, such that

$$b_{ij} = \begin{cases} 1 & \text{if } u_{ij} >= \theta, \\ 0 & \text{if } u_{ij} < \theta \end{cases}$$

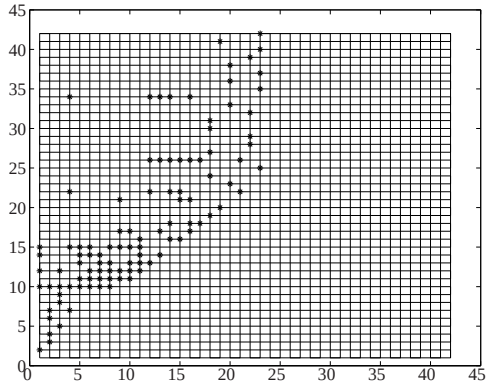
4. With the assumption that the neurons of the output layer having more than 1 hit have a higher probability of being a cluster center, arrange the elements of the Histogram data Matrix in the descending order of its value. Store the co-ordinates of these elements in the same order in another vector $V$ of length less than or equal to $M$ (where $M$ represents the total number of input vectors). This is to make the search faster.
5. Select a point serially from the vector $V$, which is of course a co-ordinate (i,j) and check whether it lies within a cluster or a valley.

$$\begin{cases} \text{if } b_{ij} = 1; & \text{point } (i,j) \text{ lies outside the obtained clusters} \\ \text{if } b_{ij} = 0; & \text{point } (i,j) \text{ lies inside the obtained clusters} \end{cases}$$

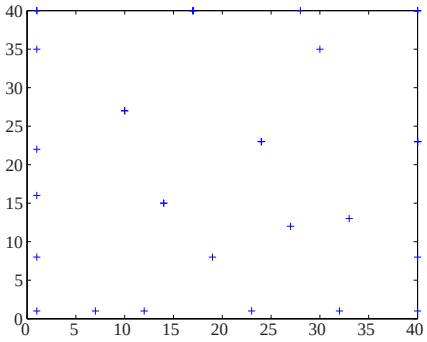
   This is to ensure that the search is limited to obtained clusters.
6. In case the data point lies within a cluster, treat this point as cluster center and start the search of other data points lying inside the same cluster. As soon as any data point is encountered, include it in the list of points lying in this cluster and remove this particular point from $V$ at the same time. This is done to make the search process efficient.
7. Repeat above two steps for all co-ordinate points in $V$.

# 4 Experimental Results

The above clustering algorithms have been attempted for the clustering of functional objects ($FOs$) in the design of complex ECUs. The system consists of 42 $FOs$ which are to be clustered on the basis of the communication between them. In order to reduce the load on CAN-bus of the system, proper clusters are needed instead of random ones [1]. The communication between these ECUs have been modeled in the form of binary matrix, where a 1 represents presence of communication and a 0 represents absence of communication between the corresponding $FOs$. A map size of $40 \times 40$ was used. Both the random and gradient initialization were tried, but gradient initialization seems more promising. An adaptation coefficient of 0.95 and initial neighborhood radius of 20 was used.
The figure 2(a) represents the experimental data where, $'*'$ represents communication line between the corresponding $FOs$. The SOM is trained using the algorithm of section 2.3. Figure 2(b) shows the hits on the output layer neurons of the trained map.
The U-Matrix representation of the trained SOM of figure 2(b), which represents the relative difference between the nearby weight vectors is shown in figure 3(a). The histograms of the trained SOM of figure 2(b) is shown in figure 3(b), which represents the number of hits on each neuron on the output layer.

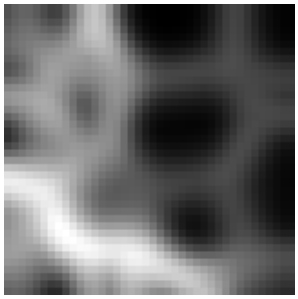



(a) Electronic System to be designed

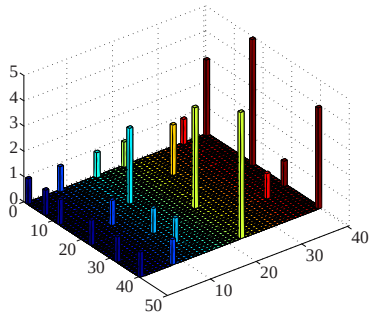



(b) Hits on SOM Output layer

Figure 2:



(a) U-Matrix Visualization




(b) Histogram Representation

Figure 3:

## 4.1 Image Processing

As explained in section 3, the information regarding various clusters could be obtained using image processing from the trained SOM. An egde closing filter is used to obtain sharp cluster boundaries from the U-Matrix data. The results are shown in figure 4(a). Further a binary image is obtained using a suitable threshold value as explained in section 3 to make the search of input vectors automatic in the formed clusters. The result is shown in figure 4(b).

Finally the steps 3 to 6 of section 3 have been performed on the binary matrix of figure 4(b) and various input vectors belonging to same cluster is being obtained. The clustered data in reference of the unclustered system of figure 2(a) is shown in figure 5. Here similar shaped marks on the communication network indicates clustered input vectors. For example, $39^{th}$, $32^{th}$, $29^{th}$ and $28^{th}$ $FOs$ communicating with $22^{nd}$ $FO$ is found to be clustered and thus is being represented by a similar mark in figure 5. Similarly information about other clustered input vectors can be obtained from this figure.

(a) Filtered U-Matrix (b) Image after Thresh-holding

Figure 4:

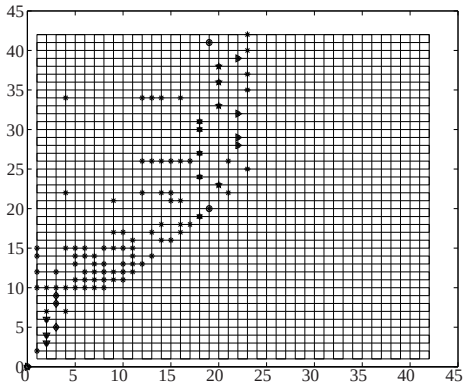
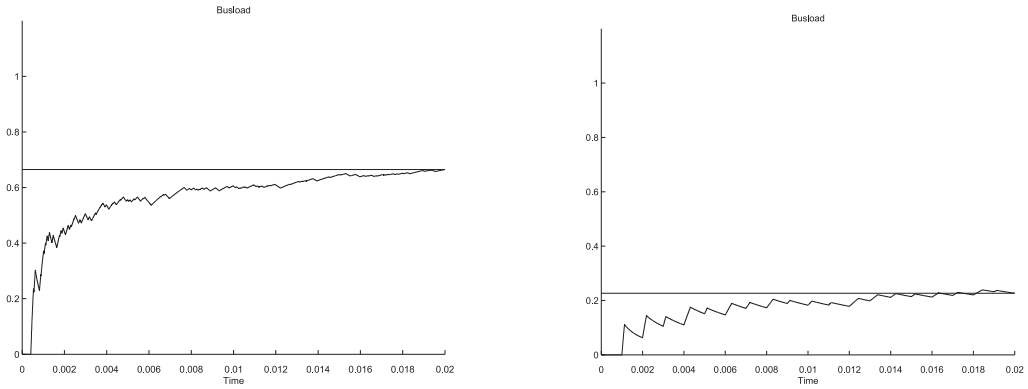



Figure 5: Proposed Architecture after clustering

## 4.2 Verification Using CAN-Bus Model

CAN is a serial bus system used for communication in automobile. Results in terms of communication load on the CAN bus have been analyzed. The communication among various elements has been already modeled in terms of array of binary and weighted vectors. A weighted entry in the communication matrix represents actual normalized communication load. The ECU network consists of 42 nodes and 101 connections running among them. If these communication lines were modeled in the form of binary vectors then we will have an average of 2.4 connections per node. These vectors are clustered using the SOM. With the aid of SOM algorithm out of these 42 $FOs$, 26 $FOs$ get clustered to 7 clusters (and the remaining 16 got mapped to VALLEYS between the clusters), thus leaving behind 23 (16+7) units with 82 connections. This leads to an average of 1.95 connections per node. In order to verify the results, a model of the CAN-bus is being used.




(a) Bus load of unclustered system (b) Bus load of clustered system

Figure 6: Can-bus load

# 5 Conclusion

The approach of SOMs provides better results than random clustering as the algorithm itself has independence of determining the number of clusters. Further experiments were done on the weighted matrix as well, but the approach could not give any more significant information.
As can be seen from figure 6 CAN bus load has reduced considerably. As shown in figure 6(a), the bus load of the unclustered system is 67.5 percent of the total bus load capacity. However, it has decreased to 22 percent in figure 6(b) when tested with the clustered system. Thus it can be seen from figure 6 that the CAN bus load has reduced considerably.
In this paper, an approach for designing complex electronic systems is presented. The communication behavior of the network is presented in the form of binary and weighted matrix. However binary matrix approach has given faster and better solutions. Thus the approach of clustering the *FOs* on the basis of communication between them could be tried in future system design. The result of clustering can be used to determine those functions which could be mapped together in hardware domain to get reduced bus load.

# References

[1] P. Nenninger (2003): *Automatisierter Entwurf und Analyse verteilter Realzeitsysteme in einem CAN-basierten Steuergerätenetzwerk*, Universität Karlsruhe (TH).

[2] P. Nenninger, T. Rambow, U. Kiencke (2004): *Automatic Model Based Partitioning of Distributed Automotive Electronic Systems*, Proceedings of the SAE World Congress.

[3] R. Isermann, R. Schwarz (2002): *FAULT-TOLERANT Drive-by-Wire Systems*, IEEE Control Systems Magazine.

[4] M. Endo, M. Ueno, T. Tanabe, M. Yamamoto (2000): *Clustering Method using Self Organising Map*, IEEE.

[5] T. Kohonen (1995): *Self-Organizing Maps*, Springer, Berlin.