Information Theory Based Analysis and Design of Sorting Networks

Mohammad Reza Ghaderi karkani
mrghaderi@ece.ut.ac.ir

Nastaran Nemati
nastaran.nemati@ece.ut.ac.ir

***Abstract* – In this paper the concepts of information theory are utilized to perform the performance analysis of the sorting networks which is selected as an example of the parallel architectures. It is shown that using this method, the source of the redundancy and the short comings of the performance can be monitored specifically and an analytical proving for efficiency of designs can be presented. Also it is expected that the optimum design can be obtained at first try in design stage instead of some try and error methods. It may have more contribution in the large size and complicated approximate working architectures. More accurate bounds for the performance characteristics can be determined using this method to anticipate the time to stop attempt to increase in the performance with definite condition. Some example is presented and future works are introduced.**

I. INTRODUCTION

Although information theory was primarily developed to deal with the fundamental questions in communication theory [1] [2], it has had a much broader impact. Information theoretic concepts have found widespread applications in some different areas such as statistics, optimization, and population studies. There had been a renewed interest in the application of these concepts to some important problems in the field of computer science in past two decade [3] [4], for example an entropy based method for minimization of Sum-of-Product (SOP) expressions of switching functions was presented at [5]. Recently some attempt to apply information theory concepts to evolutionary Boolean circuit synthesis is reported too [6] [7].

In a different point of view all computation and calculation results can be considered as a kind of information resource. For example in the sorting operation of a given number of inputs, the position of each number in the sorted list can be considered as a resource of information. Shannon has presented a way to understand the information at [1] [2], which is use of information entropy as a measure of the amount of information contained within a message [6]. So there is an idea to suppose each operation, as an experiment to receive this some part of total information of the operation. Whit this consideration, it is obvious that the efficient algorithm and so architecture is the one, in which, each operation obtains the most possible information. As entropy tells us that there is a limitation in the amount of information that can be removed from a random process without information loss [6], it is concluded that if in the cost of one processor the maximum amount of information can be obtained the efficiency is already maximized. The tight bounds for performance characteristics such as cost and delay could be derived from total amount of the information divided by the maximum amount of information can be obtained by each processor. And the total delay could be calculated as number of processors multiplied with the delay of each processor. In parallel architectures and array processors the conclusion is authentic if we use the maximum possible number of processors with the maximum amount of information recovery with each processor. On of the most important note in this consideration is monitoring the information and preventing from the unwanted redundancy which means recovery of some discovered information which is useless as is clear for us and cause in less efficiency.
It is expected that the efficient algorithm and architecture can be derived from this method of observation in design steps, as we try to maximize the efficiency in each step.

In this paper a sorting network are selected as an example of the parallel architecture. Some notation and the utilized concepts of the information theory are described briefly in the next section. The efficiency and performance for sorting network are described in part two of this report. The investigation of the performance and efficiency for some conventional sorting networks is coming in part three. The proposed network designed with information theory is coming in this section too and some comparison is done with the conventional networks. The application and future works are introduced in the fourth part and in the last part the conclusion and summary are coming.

II. NOTATION AND CONCEPTS

A. *Sorting Network*

A sorting network is a circuit that receives n inputs, x_0, x_1, x_2… x_{n-1}, and permutes them to produce n outputs, y_0, y_1, y_2 …, y_{n-1}, such that the outputs satisfy $y_0 \le y_1 \le y_2 \le \ldots y_{n-1}$. We often refer to such an n-input n-output sorting network as an n-sorter [8]. An n-input comparator network, which is an acyclic circuit of comparators, is called a sorting network if it sorts all n! permutations of {0, 1, 2…, n-1} [9].

B. *2-Sorter*

We can build an n-sorter out of 2-sorter building blocks. A 2-sorter compares its two inputs and orders them at the output, putting the smaller value, before the larger value [8].

C. *The Zero–One Principle*

An n -sorter is valid if it correctly sorts all 0/1 sequences of length n [8].

D. *Selection and Merging Network*

A selection network is a comparator network that classifies the input values into two groups such that all values in the first group are smaller than all those in the other group. A merging network is a comparator network that merges two sorted lists [9].

E. *Probabilistic Networks*

Comparator networks that correctly sort, select, or merge almost all input permutations are called probabilistic networks because they output the correct result with high probability on a random input permutation [9].

F. Network of Success Probability at least 1-ε

An n-input network N is called a sorting network of success probability at least *1-ε* if it correctly sorts *(1-ε)n!* of n! permutations of input [9].

An n-input network N is called an (n, k)-selection network of success probability at least *1-ε* if the output wires of N can be partitioned into two sets L and S such that on at least *(1-ε)n!* permutations, N outputs the k smallest input values on wires in S, and the n- k largest input values on wires in L [9].

An (m_0, m_1)-input network N is called an (m_0, m_1)-merging network of success probability at least *1-ε* if N merges at least $(1-\varepsilon)C(m_{0+} m_1, m_0)$ of the permutations under consideration [9].

G. Cost and Delay

The cost defined as the total number of 2-sorter blocks used in the design and the delay means the number of 2-sorters on the critical path from input to output [8].

H. The Entropy of an Information Source

Considering a discrete source of the finite state, in which for each possible state *i* there will be a set of probabilities $p_i(j)$ of producing of the various possible symbols *j*. Thus there is an entropy H_i for each state. The entropy of the source will be defined as the average of these H_i weighted in accordance with the probability of occurrence of the state in question:

$$H = \sum_i P_i H_i = -\sum_{i,j} P_i p_i(j) \log p_i(j) \qquad (1.1)$$

H is also approximately the logarithm of the reciprocal probability of a typical long sequence divided by the number of symbols in the sequence [1].

III. EFFICIENCY IN SORTING NETWORKS

The efficiency in parallel architectures is defined as [8]:

$$E(p) = \frac{T(1)}{pT(p)} \qquad (1.2)$$

Where the *E(p)* is the efficiency of a p processor network, the *T(1)* is total operation time which takes for a single processor to do a task and the *T(p)* is the total time which takes for *p* processor to do the same task in parallel. We are always interested in efficiency from the equation above to become equal to 1, which means the total operation time with *p* parallel processor takes *1/p* of time with a single processor. According to Amdahl's law as always a small fraction of inherently sequential or unparallelizable computation severely limits the speed-up that can be achieved with p processors [8]; we may never reach such efficiency in practice. The sequential computation limitation may be overcame using pipeline-like use of the parallel architecture and feed the architecture with large sequence of task to execute. This limitation may become negligible when the number of execution grows increasingly.

About the unparallelizable fraction, which some time is called redundancy, it seems clearer description is needed. We are going to investigate this fraction of unparallelizability through concepts of information theory to minimize this fraction, if there was any possibility, to obtain more efficiency and speedup.

The redundancy in parallel architectures is described as below:

$$R(p) = \frac{W(p)}{W(1)} \qquad (1.3)$$

Where the *R(p)* is redundancy the *W(p)* is total number of unit operations performed by the p processors; this is often referred to as computational work or energy, and the *W(1)* is the number of operation for one processor [8]. The ideal value for redundancy is 1 which means there won't be any need to do more task when we use a parallel architecture instead of a single processor. As the utilization is equal to redundancy multiplied by efficiency [8], in constant value of utilization any decrease in redundancy may considered as an increase in efficiency.

Suppose we have a sorting network whit the size of n. As previously described the whit the given sequence of n object the sorted result would be one of the n! permutations of this set of object. Through the concepts of the information theory the total information of this source of sorting operation calculated as below:

$$I(S_n) = -\log P(S_n) = -\log \frac{1}{n!} = \log n! \qquad (1.4)$$

Where the $I(S_n)$ is the total information of sorted sequence of n input objects in bit unit and so the base of logarithm is 2. It suggests that, if we absolutely earn one bit information in each processing (comparison) the lower bound of cost can be considered at the same value of the total information, *Log (n!)* which is strictly less than *n Log (n)* so the cost for n-sorting network would be *Ω(Log (n!))* which is more accurate lower bound than the classical size of *Θ(n Log(n))* resulted from Ajtai, Komlos, and Szemeredi with *O(n Log(n))* or *Θ(n $Log^2(n)$)* resulted from Batcher with *O(n $Log^2(n)$)* [8]. As the earned information of each comparison never exceeds 1, the above statement will remain truthful. Anyway the ideal sorting network should have 2-sorters, each of them deliver 1 bit information in each comparison. If we consider that we could do such comparison in each step, the minimum time required for sort a sequence of n object could be calculated trough *2Log(n!)/n* when in each step *n/2* comparison can be performed and the so the maximum information in each step equal to *n/2* and the total delay can be calculated from total information divided by *n/2*. From the above statement the lower band of delay for such n-sorting network is *2Log(n!)/n*. It means that the delay for such network is form *Ω(Log(n!)/n)* which is more accurate than the classical delay of *Θ(Log(n))* resulted from Ajtai, Komlos, and Szemeredi with *O(Log(n))*, or *Θ($Log^2(n)$)* resulted from Batcher with *O($Log^2(n)$)* [8].

We should notice when we speak about earning information trough each comparison or experiment we mean the new information. The obvious fraction of each output has no information. So if we arrange the 2-sorters such as the result of comparison would be predictable the output may contain less than one bit information. Obviously in two cascaded 2-

sorter the result of the second 2-sorter are fully predictable and so the output of the second stage contains no information. In another word the more the unpredictable output the more the information will earned. In a 2- sorter which has two output of sweep or not to sweep the information will be maximized and equal to one if inputs have identical chance to be greater than each other. It means the entropy maximizes if the probability of each output be equal to 0.5. It could be observable from equation below:

$$H(2) = -(p\log p + (1-p)\log(1-p))$$
$$H(2) = 1 \Rightarrow p = 1-p \Rightarrow p = \frac{1}{2} \quad (1.5)$$

Through the above statements, until there have been identical pair which have same chance to be greater than each other to compare we can walk on lower bound of size (cost) and delay (time). It is always practical in preliminary steps; unfortunately this condition will expire in further steps and sometime we forced to do a comparison which is somewhat predictable and so the information less than one bit is earned. It is the limitation, some time called inherently unparallelizem fraction of computation or redundancy in sorting network. In simpler word there would be a condition that there is no difference between using some processor in parallel or ignore some of them as they may earn a little or no information. There are some techniques to restrict the impact of this limitation but never eliminate it. Accordingly to reach the maximum efficiency it is sufficient to earn the higher information in each step considering the effect of comparisons in this step on the condition of next steps. If we follow this model the design will be optimum and cost and time efficient. The only requirement is calculate the entropy of each process correctly and try to maximize it.

IV. INVESTIGATING EFFICIENCY IN SORTING NETWORKS

One of the most famous sorting networks is Batcher Even-odd sorting network. A schematic of an 8-sorter of the Batcher are coming in Fig. 1. The compressed form of the Batcher even-odd sorting network is shown in Fig. 2.

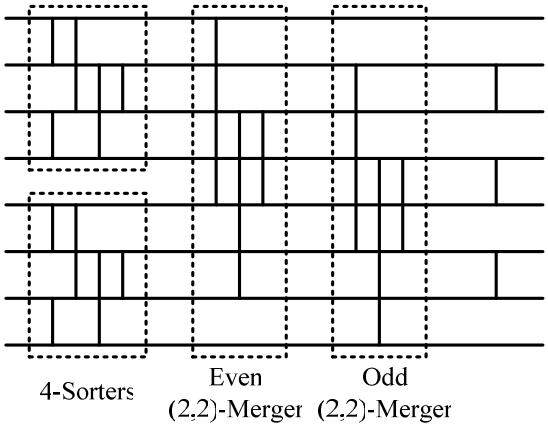

Fig. 1. An 8-input Batcher's even–odd merge sorting network.

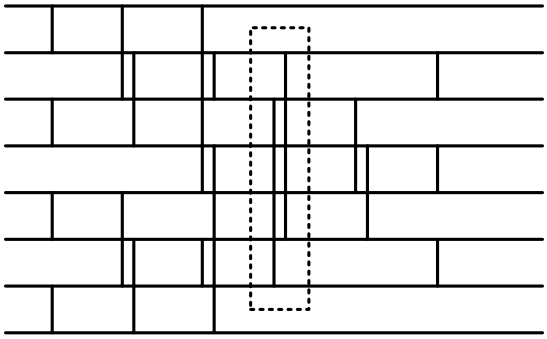

Fig. 2. An 8-input Batcher's even–odd merge sorting network in compressed form.

To investigating the efficiency of the sorting networks the model of Fig. 3 is utilized. The model shows the output status of each step according to previous comparison. The black block means that the status of this block is not clear in comparison with the (*i*,*i*) block yet. As the 2-sorters transfer the greater input to upper output the white block upper than block (*i*,*i*) means this block is obviously greater than the block (*i*,*i*) and the white block lower than block (*i*,*i*) means this block is obviously less than the block (*i*,*i*). In each comparison the earned information cause all or part of some blocks become clear. In the case of partly clearance the portion of clearance is written in the block. Suppose in one comparison object in row *i*, a_i, compared to object in row *j*, a_j, when $i>j$, to now which block become clear the statements below should be considered:

$$\begin{aligned} a_k < a_i < a_l \\ a_m < a_j < a_n \end{aligned} \quad (1.6)$$

$$\begin{aligned} a_i > a_j \Rightarrow \begin{cases} a_k, a_m < a_i < a_l \\ a_m < a_j < a_n, a_l \end{cases} \\ a_i < a_j \Rightarrow \begin{cases} a_k < a_i < a_n, a_l \\ a_k, a_m < a_j < a_n \end{cases} \end{aligned} \quad (1.7)$$

If the probability of $a_i > a_j$ equals to p and so the probability of $a_i < a_j$ equals to *1-p* the equations above simplifies as below:

$$\begin{aligned} a_i > a_j \Rightarrow \begin{cases} pa_k, pa_m < pa_i < pa_l \\ pa_m < pa_j < pa_n, pa_l \end{cases} \\ a_i < a_j \Rightarrow \begin{cases} (1-p)a_k < (1-p)a_i < (1-p)a_n, (1-p)a_l \\ (1-p)a_k, (1-p)a_m < (1-p)a_j < (1-p)a_n \end{cases} \end{aligned}$$

(1.8)

$$\begin{cases} a_k, a_m < G = pa_i + (1-p)a_j < pa_l, (1-p)a_n \\ pa_m, (1-p)a_k < L = (1-p)a_i + pa_j < a_n, a_l \end{cases}$$

(1.9)

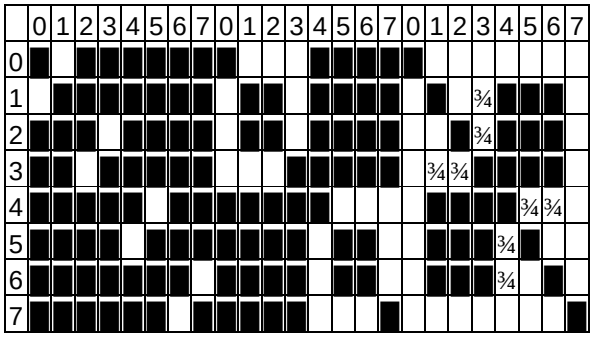

Fig. 3. Utilized model of the sorting network.

The equations above divide the two columns of *i* and *j* to four regions, upper than *i*, lower than *i*, upper than *j* and lower than *j*. The equation (1.9) says that in the lower side of column *i* the clear blocks are the lower clear blocks of column *i* and the lower clear blocks of column *j*. In the upper side of the column *j* the clear blocks are the upper clear blocks of column *i* and the upper clear blocks of column *j*. In the upper side of the column *i* the clear blocks are the upper clear blocks of column *i* multiplied by p and the upper clear blocks of column *j* multiplied by *1-p*. In the lower side of the column *j* the clear blocks are the lower clear blocks of column *i* multiplied by *1-p* and the lower clear blocks of column *j* multiplied by *p*. The block *i* in the column *j* and the block *j* in the column *i* will become clear. After all columns become compared the total chart most has diagonal and antidiagonal symmetry so if there isn't any symmetry we try to perform that and make some more block empty.

To investigate the 8-input Batcher even-odd sorting network consider the compressed form of the Fig. 2. The modeling of this 8-sorting network is coming in the Fig. 5. As previously described each stage is shown in a 8x8 table. For each of the 6 step the final form is presented. As the input *n=8* the total information is *Log(8!)=15.3 bit* as in each step total amount of 4 comparison is possible, ideally, the minimum delay is 15.3/4=4. It will observed that this ideal lower bound is newer reached as some unbalance conditions occur. We know from the entropy formula that the maximum amount of entropy occurs when the balanced data being compare in each 2-sorter. With the balanced data we mean two inputs with the same probability of being greater than each other. In the first step four comparisons with the balanced condition is existed. So four bit information is earned. In the next steps there are

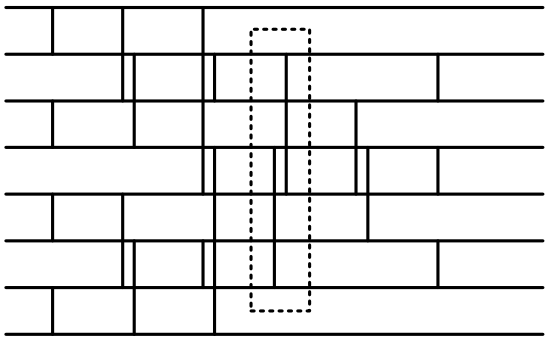

Fig. 4. An 8-input Batcher's even–odd merge sorting network.

TABLE I
THE COMPARISON BETWEEN TWO SORTING NETWORKS

Step	Batcher Odd-Even Sorting Network			Proposed Sorting Network		
	Comp.	Entropy of step		Comp.	Entropy of step	
1	0-1	1	4	0-1	1	4
	2-3	1		2-3	1	
	4-5	1		4-5	1	
	6-7	1		6-7	1	
2	0-2	1	4	0-2	1	4
	1-3	1		1-3	1	
	4-6	1		4-6	1	
	5-7	1		5-7	1	
3	0-4	1	4	0-4	1	4
	1-2	1		1-2	1	
	3-7	1		3-7	1	
	5-6	1		5-6	1	
4	1-5	1	2	1-4	1	2
	2-6	1		3-6	1	
5	2-4	≈1	≈2	2-4	≈1	≈2
	3-5	≈1		3-5	≈1	
6	1-2	≈0	≈1	1-2	≈0	≈1
	3-4	≈1		3-4	≈1	
	5-6	≈0		5-6	≈0	
Total	Size=19	≈17		Size=19	≈17	

similar condition until fourth step in which the balanced condition for the row 0 and 7 doesn't exist as the largest and smallest value is founded in third step. In this condition there are different conditions that have competitive amount of entropy. From this step a proposed network is formed to compare with Batcher even-odd sorting network. Fig. 4 shows the proposed network configuration. The modeling of this 8-sorting network is coming in the Fig. 6. The comparison between Fig. 5. and Fig. 6. shows that the different situation of the proposed network, results in different condition for future steps. The final results are coming in Table I. the results shows that although the probabilistic model shows difference in results there is no difference between two methods and the maximization of entropy results in the same performance and there is no need to try and error to find the optimum design. In other word the optimum design will forms absolutely if the mentioned rules be utilized correctly. It may show not such importance in this small problem but for larger number of inputs even the test of correctness of a design may become impossible. In order to test of the proposed network in this work a software simulator is performed in C++ which generate all *8!=40320* different permutations of {1,2,3…,8} and test the sorted sequence to ensure the correctness of the sorting function. This test takes few seconds to be run on a PC. Suppose we are going to test a 64-sorter with 64! permutation of input it may take more than 10^{79} years to be run on such PC. Another way to test reliability of sorting networks is trough the Zero–One principle and test all 0/1 sequences of length *n*. Although Zero–One principle may simplify the reliability test of an absolute sorting network, for the approximate sorting

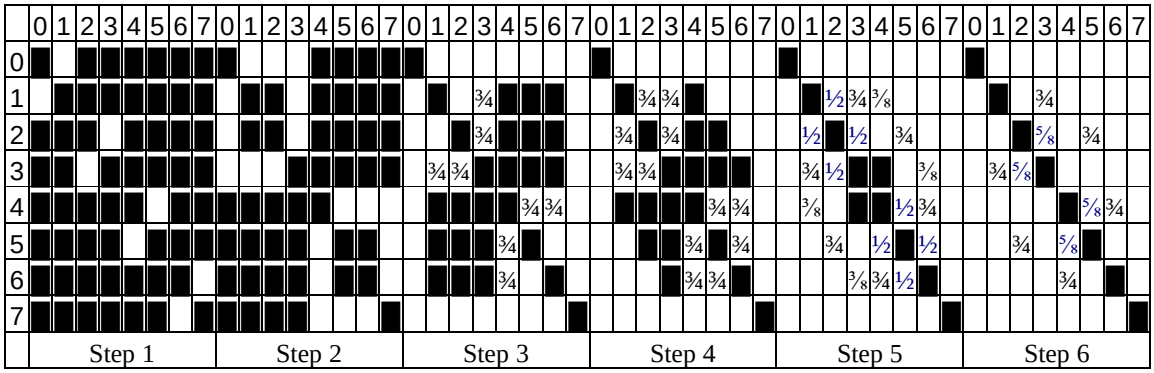

Fig. 5. The modeling of 8-input Batcher's even-odd sorting network.

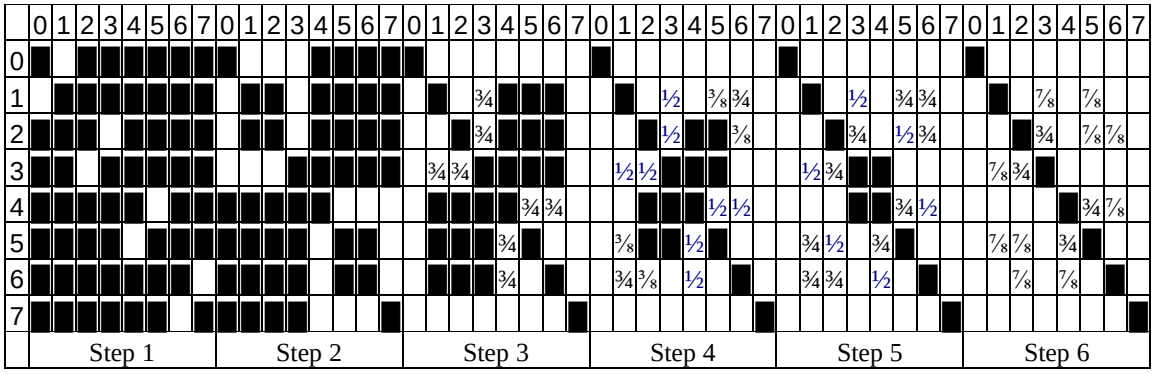

Fig. 6. The modeling of 8-input proposed (Semi-Batcher even-odd) sorting network.

network or network of success probability at least *1-ε* according to definition all *n!* permutations most be tested, so the analytical methods such as information theory based analysis of the networks may shows more benefit. Some probabilistic sorting networks are designed and tested trough this method which has less cost and time. The description about this designs is beyond the scope of this paper.

V. FUTURE WORKS

Although near optimum sorting networks are existed from several decades ago, proposing new, simple and efficient merge sort technique which has several advantages on performance over previous merge sort technique is still underway [10]. Recently an increasing interest in dynamic scheduling is shown specially in dynamic load balancing in parallel merge sort [11]. This idea that if some information from data sequence was predictable, the total amount of information which we most earn to sort the sequence will reduce, encourage us to design a reconfigurable sorting network whit the lower total cost and delay. This network can even approximately sort the sequence of data, which may be reliable in some applications in which a limited tolerance is predicted and is negligible.

As inherently these problems engage with probabilistic characteristic of the phenomena, it is expected that the information theory based dynamic architecture design become a hot solution for this problems. So if the basis of architecture and hardware design reinforced with the concepts of the information theory, a great contribution is predictable.

VI. CONCLUSION

A probabilistic model has been introduced to investigating efficiency in parallel sorting networks, utilizing the concepts of information theory. Performance analysis of Batcher even-odd sorting networks with eight inputs has been done using this method. A semi-Batcher sorting network has been proposed to show that there is no need to try and error in complete design to find the optimum design using the proposed method of design and analysis. It has been shown that using this method, the source of the redundancy and unparallelizem can be monitored specifically and an analytical proving for efficiency of designs can be presented. More contribution in the large size and complicated especially approximate working architectures design and test has been shown. More accurate bounds for the performance characteristics have been determined. Some examples have been introduced trough future works in which this method may show more elegance.

ACKNOWLEDGMENT

The authors would like to thank Prof. S. M. Fakhraei for his supportive lectures at University of Tehran, specially the Parallel Processing, and his valuable guidance to perform this work.

REFERENCES

[1] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, July- 1948.
[2] C. E. Shannon, "A mathematical theory of communication, Part 2." *Bell SW.Tech. J.*, vol. 27, pp. 623-*656*, Oct. 1948.
[3] C. R. P. Hartmann, P. K. Varshney, K. G. Mehrotra, and K. L. Gerberich "Application of Information Theory to the Construction of Efficient Decision Trees," *IEEE Transactions on Information Theory*, VOL. IT-28, NO. 4, JULY 1982
[4] A. M. KabakGioijlu, P. K. Varshney, and C. R. P. Hartmann, "Application of information theory to switching function minim isat ion," *IEE Proceedings.* Vol 1.37. Pt. E. No. 5. September 1990.
[5] S. N. Yanushkevich, V. P. Shmerko, R. S. Stankovie, P. Dziurzanski, and D. V. Popel, "Experimental Verification of the Entropy Based Method for Minimization of Switching Functions on Pseudo Ternary Decision Trees," *TELSIKS'99*, 13-15. October 1999, NiS, Yugoslavia.
[6] A. H. Aguirre, and C. C. Coello, "Gate-level Synthesis of Boolean Functions using Information Theory Concepts," *Proceedings of the Fourth Mexican International Conference on Computer Science*, 2003.
[7] A. H. Aguirre, and C. C. Coello, "Fitness Landscape and Evolutionary Boolean Synthesis using Information Theory Concepts," *Proceedings of The 2003 NASA/Dod Conference on Evolvable Hardware*, 2003.
[8] B. Parhami, *Introduction to Parallel Processing Algorithms and Architectures.* Kluwer Academic Publishers , p.19, p.103-143, 2003.
[9] Leighton, T., Y. Ma, and T. Suel, "On Probabilistic Networks for Selection, Merging, and Sorting," *Theory of Computing Systems*, Vol. 30, pp. 559–582, 1997.
[10] N. Hossain, Md. G. Rabiul Alma, Md. Amiruzzaman, S. M. Moinul Quadir, "An Efficient Merge Sort Technique that Reduces both Times and Comparisons," *International Conference on Information and Communication Technologies: From Theory to Applications*, 2004.
[11] D. G. Armando, N. Marcelo, C. Franco, and D. G. Laura, "Dynamic Load Balance in Parallel Merge Sorting over Homogeneous Clusters," *Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA'05)*,2005.