2

Determining a Non-Collision Data Transfer Paths in Hypercube Processors Network

Jan Chudzikiewicz and Zbigniew Zieliński
Military University of Technology
Poland

1. Introduction

Fault tolerant systems are called systems capable of performing certain tasks despite of some unfitness (Kulesza et al., 1999; Kulesza, 2000; Chudzikiewicz, 2002; Chudzikiewicz & Zielinski, 2010). One of the conditions to be met by the structure used in fault tolerant systems is redundancy of the system components, namely use of redundant structures (see definition 2). Example of a structure, which ensures adequate number of communication lines is a binary *n*-dimensional hypercube H^n structure (see definition 1). Structures of this type have large reliability (Kulesza, 2000, 2003) and large diagnostic deepness in the sense of network coherence (Kulesza, 2000; Chudzikiewicz, 2002). The hypercube structures find wide application in data processing systems, especially for building fault tolerant systems, because such structures have natural features of redundancy.

Interconnection networks with the hypercube logical structure possess already numerous applications in critical systems and still they are the field of interest of many theoretical studies. In this kind of network the faulty processor may be replaced with a spare fault free processor (e.g. after network reconfiguration) or may be eliminated from the network and the new (degraded) network continues to operate, provided that it meets certain requirements. The last kind of such network is called a soft degradation network. A system's dependability is maintained by ensuring that it can discriminate between faulty and fault-free processors. The process of identifying faulty processors is called diagnosis of the processors' network.

We assume, that processors that are determined as faulty could not be repaired or replaced with spare equipment. The elimination of the faulty processor from the network induces (in the general case) the structure of several components of consistency. If the obtained (reduced) logical structure of the network is not the working structure, then the network loses its ability to operate (this network state will be determined as the network failure).

Correct diagnosis is another condition to tolerate failures in such systems. The quality of this diagnosis is critical to restore the suitability of the system by replacing the failure units, or isolation of such elements (soft system degradation) and perform reconfiguration tasks (Wang, 1999; Kulesza, 2000; Chudzikiewicz & Murawski, 2006; Zielinski et al., 2010). This requires the use of most effective diagnosis methods (Chudzikiewicz & Zielinski, 2003; Zielinski, 2006). In the case of distributed processing systems, a methods which uses the results of mutual testing of the system elements may be used (Kulesza & Zieliński 2010; Zielinski et al., 2011).

Both, from the viewpoint of functional tasks for which the system was built as well as the implementation of a system diagnosis it is important to ensure an effective mechanism for communication between system components (Chudzikiewicz & Zielinski, 2010; Kulesza et al., 1999).

In multiprocessor systems, effective communication between processors is one of the critical elements of data processing. Processors in multiprocessor systems communicate with each other by sending messages. The problem of data transfer in hypercube systems has been widely analyzed in the literature. Among other things, Gordon and Stout present the method called by them "sidetracking" (Gordon & Stout, 1988). This method assumes that each node stores information about the reliability state of their neighbors. Information from a given node is sent by a random path which is adjacent to a faulty free node. In the case of no path adjacent to the faulty free nodes, information is blocked and sent back to the node from which it was originally sent. A disadvantage of this method is little probability to submit information for a specified number of unfit nodes and large time delay. Another method proposed by Chen is called "backtracking" (Chen & Shin, 1990). This method assumes that the information on subsequent nodes, which mediated in data transmission is stored in the transmitted data. In the case that the data reaches the node that is adjacent to the unfit nodes, the information is used to send data back to the earlier node. Disadvantage of this solution is that the redundant information is moved in transmitted data and large time delays. Both methods - "sidetracking" and "backtracking" may lead to situation where the same intermediate nodes will be used to send data from different system components that communicate with each other (pairs of nodes). This may cause significant overload of individual links, while others will have unused resources. Moreover, individual data packets can be sent over different paths and reach out customers in a different (not always consistent with the assumed) sequence. This is especially inadvisable in the case of the need to ensure the efficiency of communication e.g. video conference realization.

This chapter presents the method of the data transmission paths reconfiguration in a hypercube-type processor network. The method is based on the determining strongly and mutually independent simple chains (see definition 4) between communicating pairs of nodes, which are called – I/O ports[1]. The method assumes that each node stored information about the reliability state of the system. The implementation problem of the presented method in embedded systems has also been raised. Mechanisms based on operating systems of Windows CE class are also presented, which will facilitate the implementation of the developed method.

2. Basic definitions

Let Z^n indicate the set of *n*-dimensional binary vectors.

Let us determine:

$$(s_1,...,s_2) = \{z \in Z^n : ((s_i \neq x) \Rightarrow (z_i = s_i)) \wedge ((s_i = x) \Rightarrow (z_i = \{0,1\}))\}\ (s_i \in \{0,1,x\}, 1 \leq i \leq n),$$

[1] I/O port is a node representing an element in a real system which can communicate with external networks.

where:

x indicates the indefinite value (0 or 1),

$Z(s)$ – is a set of 0-dimensional cubes (vector set $z=(z_1,...,z_n)$ $(z_i\in\{0,1\},\ 1\le i\le n)$ of cube s $(s\in S^n)$).

Definition 1

An n-dimensional binary hypercube is the ordinary graph G $(G=<E,G>,$ $|E|=2^n,|U|=n\cdot 2^{n-1})$ with 2^n nodes, each of which is described with an adequate binary vector z $(z=(z_1,...,z_n),\ z_i\in\{0,1\},\ 1\le i\le n,\ z\in Z^n,\ |Z^n|=2^n)$ and $n\cdot 2^{n-1}$ edges connecting these nodes, which vectors that describe them are distant by 1 according to the Hamming measure.

Hereinafter the a nodes graph H^n will represent real processors, and its edges the data transmission paths between processors, which are adjacent to a specific edge.

The Hamming distance between two binary vectors $b'(\tau_i)$ and $b''(\tau_i)$, which are the poles of the chain τ_i, complies with the dependency:

$$\delta(b'(\tau_i),b''(\tau_i))=\sum_{k\in\{1,...,n\}}(b'(\tau_i)_k\oplus b''(\tau_i)_k)$$

where:

$b'(\tau_i)_k$ – the k-th element of the binary vector $b'(\tau_i)$,

$\oplus$ –modulo 2 sum.

Definition 2

Redundant network logical structure is a structure whose graph G $(G=<E,U>)$ meets the condition: $|U|\ge|E|$.

Definition 3

A chain τ with a length k $(0\le k\le 2^n)$ in H^n is called a coherent subgraph of the H^n graph if it includes $k+1$ nodes from which only two are of the first degree.

The node of the first degree chain is called the pole of this chain.

Let $Z(\tau)$ and $B(\tau)$ $(B(\tau)\subseteq Z(\tau))$ indicate the set of nodes and the poles of the τ chain respectively.

The chain τ will be presented both in the form of a subgraph $<Z(\tau)>H^n$ as well as in the form of a set $S(\tau)$ of s $(s\in S_1^n)$ 1-dimensional subcubes such that: $[s\in S(\tau)]\Leftrightarrow[\exists\, z',z''\in Z(\tau):z'+z''=s]$.

Definition 4

It is said that chains τ' and τ'' in H^n are strongly and mutually independent, if $Z(\tau')\cap Z(\tau'')=\varnothing$.

An example of hypercube structure H^4 is shown in Figure 1. This structure is characterized by $|E|=2^4=16, |U|=4\cdot 2^{4-1}=32$. In parentheses in Figure 1 the binary label values assigned to individual nodes are given. The set Z^n of nodes is of the form as below:

$$Z^n=\{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}$$

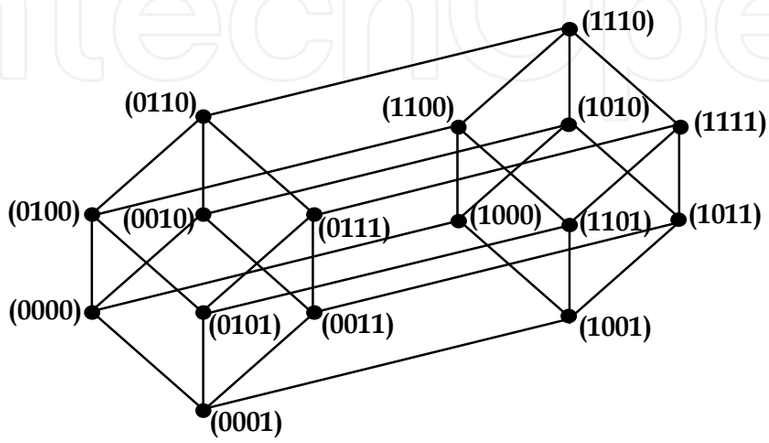


Fig. 1. An example of the H^4 structure.

Damage to the processor in the system described by the H^n graph and a lack of interchangeability causes the creation of a working structure, which is a partial subgraph of the graph H^n. An example of this type of structure is the structure shown in Figure 2, which is a partial subgraph of the graph H^4 shown in Figure 1. The processors labeled 0111 and 1000 are damaged.

3. The method of determining non-collision paths in a cube-type structure

The method of determining non-collision paths in hypercube structures is based on determination of simple chains between the nodes representing processors, which want to communicate with each other. An example of such a structure is shown in Figure 3.

Suppose that in the present structure the nodes from the E' set (nodes: 0000, 0010, 0100) and E'' set (nodes: 0011, 1011, 1110) $((E', E'' \subset E) \wedge (E' \cap E'' = \varnothing))$ represent processors, which are connected to I/O ports. Sending data from the processor represented by a node from the E' set to the processor represented by a node from the E'' set, requires a mediation of processors represented by nodes from the E''' set $(E''' = E \setminus (E' \cup E''))$.

Let us accept the following assumptions:

- minimum cost to send data – interpreted as the minimum number of elements in the transmission of intermediary data;
- possibility of implementing parallel data transfer between several pairs of processors – each pair communicates through independent pathways.

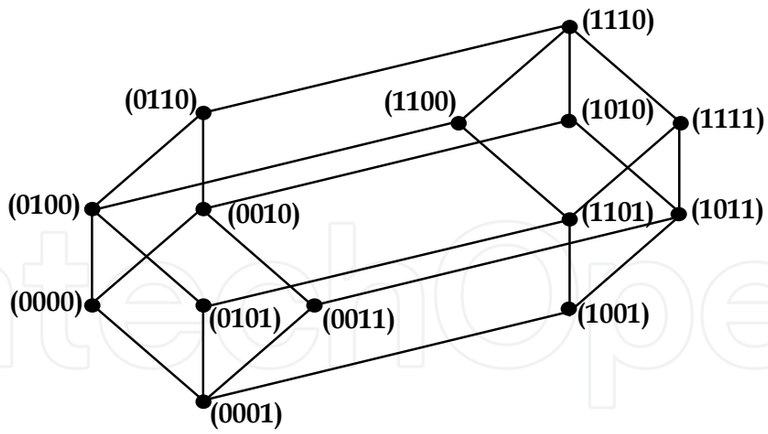


Fig. 2. An example of a partial subgraph of the structure shown in Figure 1.

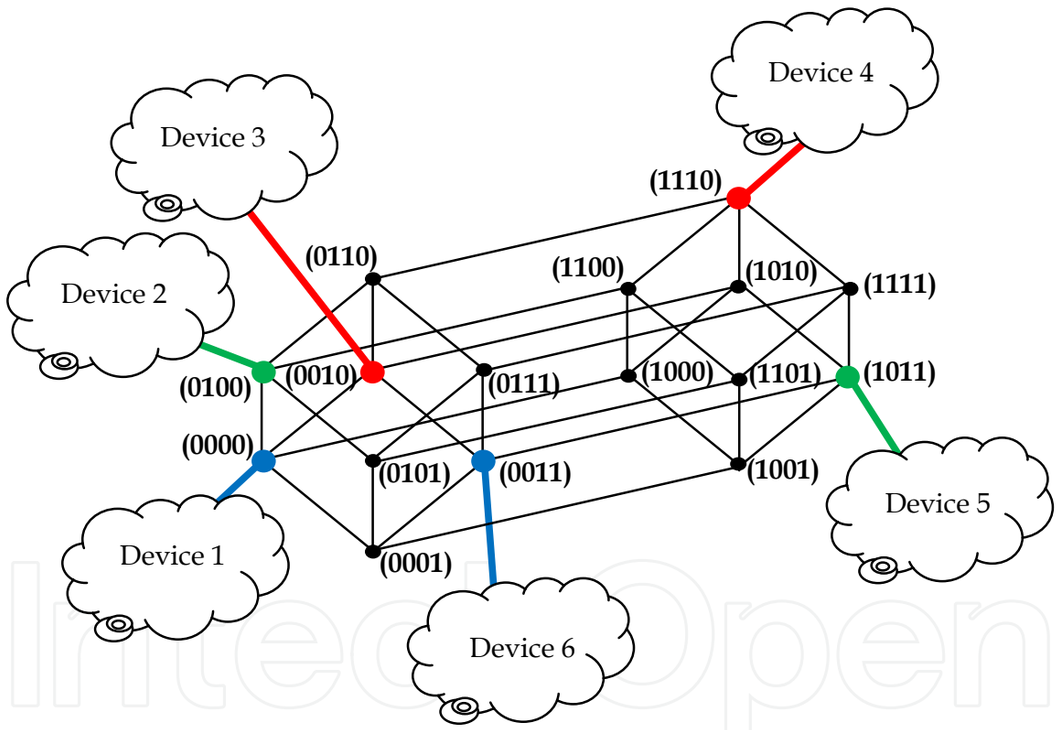


Fig. 3. An example of the H^4 structure with indicated I/O ports.

Determining connections between the nodes $e'\,(e' \in E')$ and $e''\,(e'' \in E'')$ means determination of the shortest chain (see definition 3) between these nodes. Implementation of parallel transmission between nodes from the set E' and the nodes from the set E'' requires calculation of strongly and mutually independent chains between specific nodes.

The final result of the method is to determine all paths between elements, which at the given moment intend to exchange data in such a way so that they do not interfere with other transmissions.

The proposed method is implemented in two phases. In the first phase, all possible simple chains between nodes that want to implement data exchange are determined. Determined for a specific pair of nodes, simple chains can't contain other nodes that are I/O ports.

In the second phase, from the set of simple chains, strongly and mutually independent chains are determined for pairs of nodes that communicate with each other. The method to determine the simple chain uses the algorithm based on the adjacency binary matrix. The algorithm determining data transmission paths between node pairs is given below and the adjacency binary matrix for the structure from Figure 3 is shown in Figure 4.

Let us denote:

$Ł(z', z'')$ - a set of chains connecting nodes z' and z'',

$Z(\tau_j)$ - a set of nodes that create chain τ_j,

$B(\tau_j)$ - a set of poles of chain τ_j,

W – a set of pairs of poles among, which simple chains $W = \{(z(e'), z(e'')) : (e', e'' \in E) \wedge (z(e'), z(e'') \in B(\tau))\}$ will be determined;

P – a set of strongly and mutually independent chains connecting communicating pairs of nodes.

Step 1. Select an unselected node as initial pole z' from the set W with the smallest label. As the end pole, select the node z'', so that: $(z', z'') \in W$.

Step 2. Determine the set $Ł(z', z'')$ of chains connecting nodes z' and z'', so that:

$$Ł(z', z'') = \{\tau : (Z(\tau) \setminus B(\tau)) \cap W = \varnothing\}.$$

If the set of chains is determined for all pairs of the set W go to step 3, otherwise go to step 1.

Step 3. Take the chain τ from the chain set $Ł(z', z'')$ for $(z', z'') \in W$. $Ł(z', z'') = Ł(z', z'') \setminus \tau$.

Step 4. Step 4Add the selected chain to set P, if:

$$((B(\tau) \neq B(\tau_i)) \wedge (Z(\tau) \neq Z(\tau_i))) \quad \forall \tau_i \in P, i = \{1,...,|P|\}).$$

If the condition is met go to step 5.

If the condition is not met and $Ł(z', z'') \neq \varnothing$ go to step 3.

If the condition is not met and $Ł(z', z'') = \varnothing$ go to step 5.

Step 5. If $|P| = |W|$ the set of chains for all pairs $(z', z'') \in W$ is determined. Go to step 6.

If $|P| \neq |W|$ determine the next pair $(z', z'') \in W$. Go to step 3.

Step 6. The end of the algorithm.

On the adjacency matrix from Figure 4 colors mark rows and columns corresponding to the I/O ports.

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 10	1011 11	1100 12	1101 13	1110 14	1111 15
0000	**0**		1	1		1				1							
0001	**1**	1			1		1				1						
0010	**2**	1			1			1				1					
0011	**3**		1	1					1				1				
0100	**4**	1					1	1						1			
0101	**5**		1			1			1						1		
0110	**6**			1		1			1							1	
0111	**7**				1		1	1									1
1000	**8**	1									1	1		1			
1001	**9**		1							1			1		1		
1010	**10**			1						1			1			1	
1011	**11**				1						1	1					1
1100	**12**				1					1					1	1	
1101	**13**					1					1			1			1
1110	**14**						1					1		1			1
1111	**15**							1					1		1	1	

Fig. 4. Adjacency matrix for the structure shown in Figure 3.

For illustration of the algorithm let us trace designation of a simple chain between nodes: 0100 and 1011. In the first step the algorithm has appointed the node 0101 moving along the 4-th column to 5 row of this matrix. This is shown in Figure 5.

	0000 0	0001 1	0010 2	0011 3	0100 4	0101 5	0110 6	0111 7	1000 8	1001 9	1010 10	1011 11	1100 12	1101 13	1110 14	1111 15
0000	**0**															
0001	**1**					1				1						
0010	**2**															
0011	**3**															
0100	**4**					1	1						1			
0101	**5**	1			1			1						1		
0110	**6**				1			1								
0111	**7**					1	1									1
1000	**8**									1	1		1			
1001	**9**	1							1			1		1		
1010	**10**								1			1				
1011	**11**									1	1					1
1100	**12**				1				1					1		
1101	**13**					1				1			1			1
1110	**14**															
1111	**15**							1				1		1		

Fig. 5. Illustration of the first step of the algorithm. The matrix does not contain others I/O ports.

In the second step the algorithm has appointed 0001 node moving along the 5th row of the matrix. This is shown in Figure 6.

The algorithm in the next steps, alternating moving along columns and rows has appointed simple chain linking nodes: 0100 and 1011. This is shown in Figure 7.

The algorithm in six steps, has appointed the single simple chain linking nodes: : 0100 and 1011 the following form: {0100, 0101, 0001, 1001, 1000, 1010, 1011}.

For the structure from Figure 3 the algorithm determined sets of simple chains $Ł(z', z'')$ as shown in Table 1.

In the second phase of the method from the set of a simple chains, as shown in Table 1, for each pair of I/O ports, will be chosen the shortest simple chains allowing for the implementation of collision-free data transfer, as shown in Figure 8.

Let us consider the case when nodes: 0111 and 1000 are damaged. According to the presented method the new configuration will be determined by choosing from the set shown in Table 1 simple chains, which do not contain damaged nodes. The algorithm assigned new sets of simple chains $Ł(z', z'')$ shown in Table 2. Figure 9 shows the network configuration rejecting the unfit nodes: 0111 and 1000 and allows implementation of the collision-free data transfer.

		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0000	**0**																
0001	**1**						**1**				**1**						
0010	**2**																
0011	**3**																
0100	**4**						**1**	**1**						**1**			
0101	**5**		**1**			**1**			~~1~~						~~1~~		
0110	**6**					~~1~~			**1**								
0111	**7**						**1**	**1**									**1**
1000	**8**										**1**	**1**		**1**			
1001	**9**		**1**							**1**			**1**		**1**		
1010	**10**									**1**			**1**				
1011	**11**										**1**	**1**					**1**
1100	**12**					~~1~~				**1**					**1**		
1101	**13**						**1**				**1**			**1**			**1**
1110	**14**																
1111	**15**								**1**				**1**		**1**		

Fig. 6. Illustration of the second step of the algorithm.

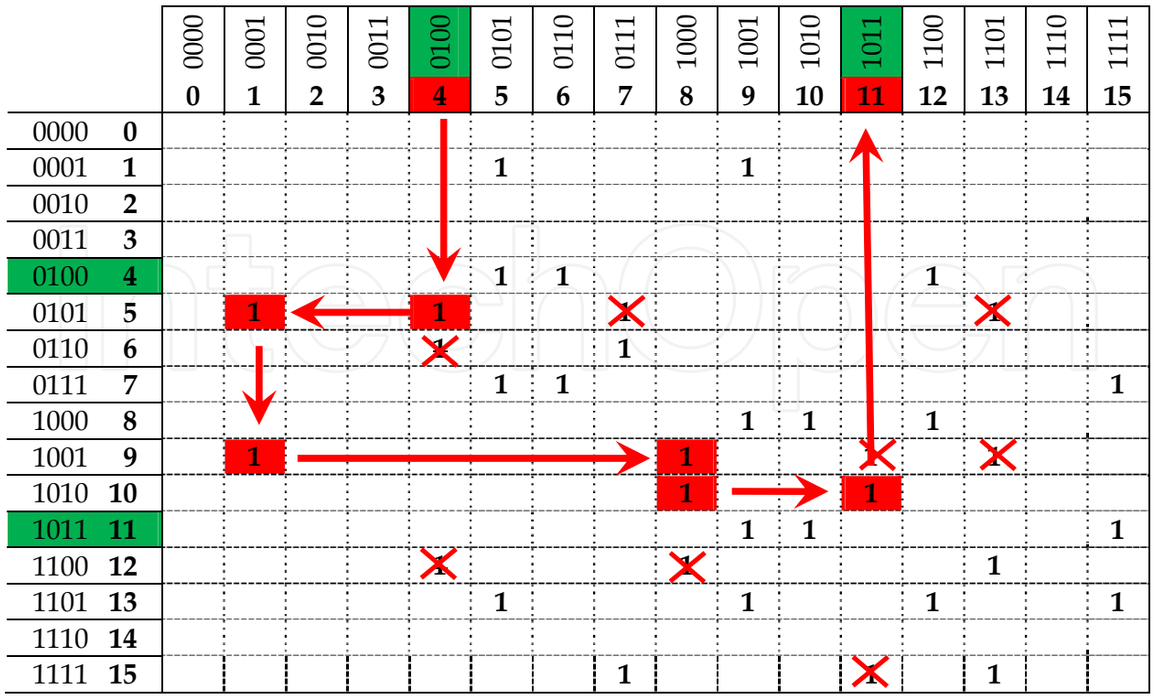


Fig. 7. Illustrate appointed by the algorithm the single simple chain between nodes: 0100 and 1011.

Ł(0000, 0011)	Ł(0010, 1110)	Ł(0100, 1011)
0 ; 1 ; 3	2 ; 6 ; 7 ; 5 ; 1 ; 9 ; 8 ; 10 ; 14	4 ; 5 ; 1 ; 9 ; 8 ; 10 ; 11
0 ; 1 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 1 ; 9 ; 8 ; 12 ; 13 ; 15 ; 14	4 ; 5 ; 1 ; 9 ; 8 ; 12 ; 13 ; 15 ; 11
0 ; 1 ; 5 ; 13 ; 15 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 1 ; 9 ; 8 ; 12 ; 14	4 ; 5 ; 1 ; 9 ; 11
0 ; 1 ; 9 ; 8 ; 12 ; 13 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 1 ; 9 ; 13 ; 12 ; 8 ; 10 ; 14	4 ; 5 ; 1 ; 9 ; 13 ; 12 ; 8 ; 10 ; 11
0 ; 1 ; 9 ; 8 ; 12 ; 13 ; 15 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 1 ; 9 ; 13 ; 12 ; 14	4 ; 5 ; 1 ; 9 ; 13 ; 15 ; 11
0 ; 1 ; 9 ; 13 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 1 ; 9 ; 13 ; 15 ; 14	4 ; 5 ; 7 ; 15 ; 11
0 ; 1 ; 9 ; 13 ; 15 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 13 ; 9 ; 8 ; 10 ; 14	4 ; 5 ; 7 ; 15 ; 13 ; 9 ; 8 ; 10 ; 11
0 ; 8 ; 9 ; 1 ; 3	2 ; 6 ; 7 ; 5 ; 13 ; 9 ; 8 ; 12 ; 14	4 ; 5 ; 7 ; 15 ; 13 ; 9 ; 11
0 ; 8 ; 9 ; 1 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 13 ; 12 ; 8 ; 10 ; 14	4 ; 5 ; 7 ; 15 ; 13 ; 12 ; 8 ; 9 ; 11
0 ; 8 ; 9 ; 1 ; 5 ; 13 ; 15 ; 7 ; 3	2 ; 6 ; 7 ; 5 ; 13 ; 12 ; 14	4 ; 5 ; 7 ; 15 ; 13 ; 12 ; 8 ; 10 ; 11
0 ; 8 ; 9 ; 13 ; 5 ; 1 ; 3	2 ; 6 ; 7 ; 5 ; 13 ; 15 ; 14	4 ; 5 ; 13 ; 9 ; 8 ; 10 ; 11
0 ; 8 ; 9 ; 13 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 15 ; 13 ; 5 ; 1 ; 9 ; 8 ; 10 ; 14	4 ; 5 ; 13 ; 9 ; 11
0 ; 8 ; 9 ; 13 ; 15 ; 7 ; 3	2 ; 6 ; 7 ; 15 ; 13 ; 5 ; 1 ; 9 ; 8 ; 12 ; 14	4 ; 5 ; 13 ; 12 ; 8 ; 9 ; 11
0 ; 8 ; 9 ; 13 ; 15 ; 7 ; 5 ; 1 ; 3	2 ; 6 ; 7 ; 15 ; 13 ; 9 ; 8 ; 10 ; 14	4 ; 5 ; 13 ; 12 ; 8 ; 10 ; 11
0 ; 8 ; 12 ; 13 ; 5 ; 1 ; 3	2 ; 6 ; 7 ; 15 ; 13 ; 9 ; 8 ; 12 ; 14	4 ; 5 ; 13 ; 15 ; 11
0 ; 8 ; 12 ; 13 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 15 ; 13 ; 12 ; 8 ; 10 ; 14	4 ; 6 ; 7 ; 5 ; 1 ; 9 ; 8 ; 10 ; 11
0 ; 8 ; 12 ; 13 ; 9 ; 1 ; 3	2 ; 6 ; 7 ; 15 ; 13 ; 12 ; 14	4 ; 6 ; 7 ; 5 ; 1 ; 9 ; 8 ; 12 ; 13 ; 15 ; 11
0 ; 8 ; 12 ; 13 ; 9 ; 1 ; 5 ; 7 ; 3	2 ; 6 ; 7 ; 15 ; 14	4 ; 6 ; 7 ; 5 ; 1 ; 9 ; 11
0 ; 8 ; 12 ; 13 ; 15 ; 7 ; 3	2 ; 6 ; 14	4 ; 6 ; 7 ; 5 ; 1 ; 9 ; 13 ; 12 ; 8 ; 10 ; 11
0 ; 8 ; 12 ; 13 ; 15 ; 7 ; 5 ; 1 ; 3	2 ; 10 ; 8 ; 9 ; 1 ; 5 ; 7 ; 6 ; 14	4 ; 6 ; 7 ; 5 ; 1 ; 9 ; 13 ; 15 ; 11
	2 ; 10 ; 8 ; 9 ; 1 ; 5 ; 7 ; 15 ; 13 ; 12 ; 14	4 ; 6 ; 7 ; 5 ; 13 ; 9 ; 8 ; 10 ; 11
	2 ; 10 ; 8 ; 9 ; 1 ; 5 ; 7 ; 15 ; 14	4 ; 6 ; 7 ; 5 ; 13 ; 9 ; 11
	2 ; 10 ; 8 ; 9 ; 1 ; 5 ; 13 ; 12 ; 14	4 ; 6 ; 7 ; 5 ; 13 ; 12 ; 8 ; 9 ; 11
	2 ; 10 ; 8 ; 9 ; 1 ; 5 ; 13 ; 15 ; 7 ; 6 ; 14	4 ; 6 ; 7 ; 5 ; 13 ; 12 ; 8 ; 10 ; 11
	2 ; 10 ; 8 ; 9 ; 1 ; 5 ; 13 ; 15 ; 14	4 ; 6 ; 7 ; 5 ; 13 ; 15 ; 11
	2 ; 10 ; 8 ; 9 ; 13 ; 5 ; 7 ; 6 ; 14	4 ; 6 ; 7 ; 15 ; 11
	2 ; 10 ; 8 ; 9 ; 13 ; 5 ; 7 ; 15 ; 14	4 ; 6 ; 7 ; 15 ; 13 ; 5 ; 1 ; 9 ; 8 ; 10 ; 11

	2 ; 10 ; 8 ; 9 ; 13 ; 12 ; 14 2 ; 10 ; 8 ; 9 ; 13 ; 15 ; 7 ; 6 ; 14 2 ; 10 ; 8 ; 9 ; 13 ; 15 ; 14 2 ; 10 ; 8 ; 12 ; 13 ; 5 ; 7 ; 6 ; 14 2 ; 10 ; 8 ; 12 ; 13 ; 5 ; 7 ; 15 ; 14 2 ; 10 ; 8 ; 12 ; 13 ; 9 ; 1 ; 5 ; 7 ; 6 ; 14 2 ; 10 ; 8 ; 12 ; 13 ; 9 ; 1 ; 5 ; 7 ; 15 ; 14 2 ; 10 ; 8 ; 12 ; 13 ; 15 ; 7 ; 6 ; 14 2 ; 10 ; 8 ; 12 ; 13 ; 15 ; 14 2 ; 10 ; 8 ; 12 ; 14 2 ; 10 ; 14	4 ; 6 ; 7 ; 15 ; 13 ; 5 ; 1 ; 9 ; 11 4 ; 6 ; 7 ; 15 ; 13 ; 9 ; 8 ; 10 ; 11 4 ; 6 ; 7 ; 15 ; 13 ; 9 ; 11 4 ; 6 ; 7 ; 15 ; 13 ; 12 ; 8 ; 9 ; 11 4 ; 6 ; 7 ; 15 ; 13 ; 12 ; 8 ; 10 ; 11 4 ; 12 ; 8 ; 9 ; 1 ; 5 ; 7 ; 15 ; 11 4 ; 12 ; 8 ; 9 ; 1 ; 5 ; 13 ; 15 ; 11 4 ; 12 ; 8 ; 9 ; 11 4 ; 12 ; 8 ; 9 ; 13 ; 5 ; 7 ; 15 ; 11 4 ; 12 ; 8 ; 9 ; 13 ; 15 ; 11 4 ; 12 ; 8 ; 10 ; 11 4 ; 12 ; 13 ; 5 ; 1 ; 9 ; 8 ; 10 ; 11 4 ; 12 ; 13 ; 5 ; 1 ; 9 ; 11 4 ; 12 ; 13 ; 5 ; 7 ; 15 ; 11 4 ; 12 ; 13 ; 9 ; 1 ; 5 ; 7 ; 15 ; 11 4 ; 12 ; 13 ; 9 ; 8 ; 10 ; 11 4 ; 12 ; 13 ; 9 ; 11 4 ; 12 ; 13 ; 15 ; 7 ; 5 ; 1 ; 9 ; 8 ; 10 ; 11 4 ; 12 ; 13 ; 15 ; 7 ; 5 ; 1 ; 9 ; 11 4 ; 12 ; 13 ; 15 ; 11

Table 1. Sets of simple chains determined by algorithm for the structure from Figure 3.

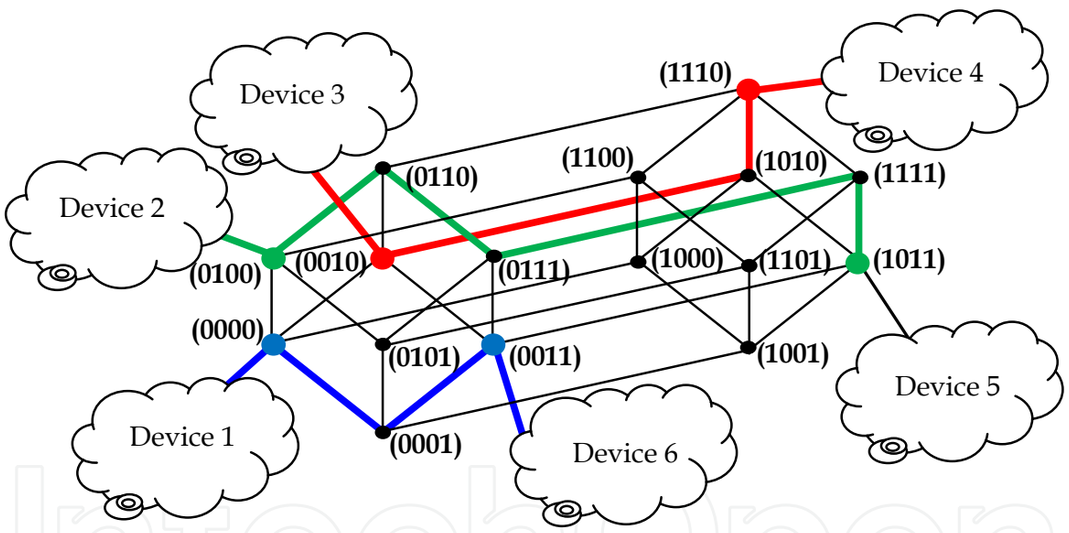


Fig. 8. An example of network configuration that allows collision-free communication between I/O ports.

$Ł(0000, 0011)$	$Ł(0010, 1110)$	$Ł(0100, 1011)$
0 ; 1 ; 3	2 ; 6 ; 14 2 ; 10 ; 14	4 ; 5 ; 1 ; 9 ; 11 4 ; 5 ; 1 ; 9 ; 13 ; 15 ; 11 4 ; 5 ; 13 ; 9 ; 11 4 ; 5 ; 13 ; 15 ; 11 4 ; 12 ; 13 ; 5 ; 1 ; 9 ; 11 4 ; 12 ; 13 ; 9 ; 114 ; 12 ; 13 ; 15 ; 11

Table 2. The sets of simple chains without damaged nodes (0111, 1000).

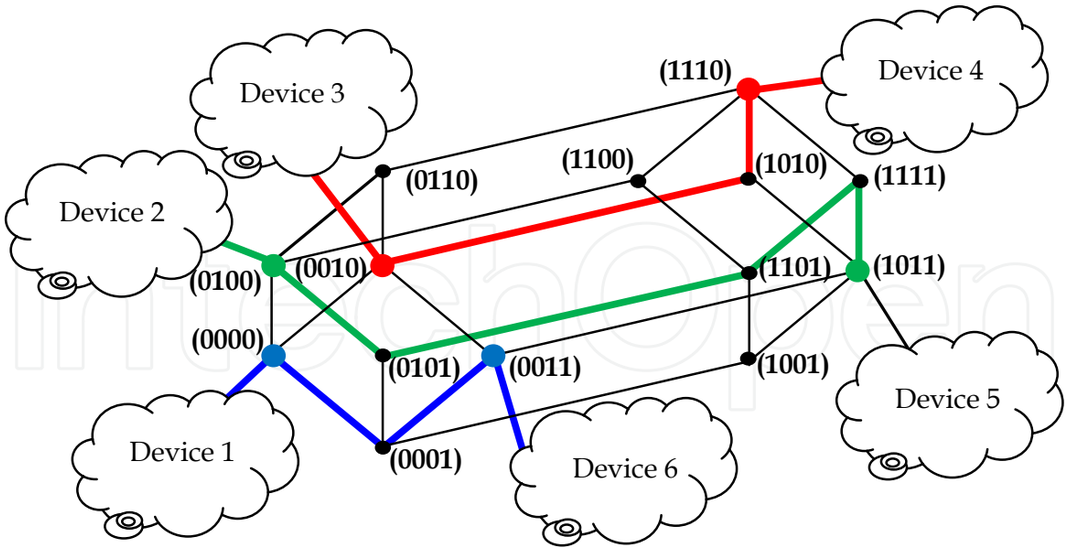


Fig. 9. The sets of simple chains without damaged nodes (0111, 1000).

4. Implementation of the method of determining non-collision paths in Windows CE

The processor network is based on S3C2440 processor on S3C2440SBC board. This is a 32-bit RISC processor. It is compatible with the Harvard Architecture model, characterized by a separate cache for commands (16KB) and data (16KB). It is equipped in: Memory Management Unit and Internal Advanced Microcontroller Bus Architecture. The family ARM920T processor is chosen (S3C2440 processor belongs to it), because of the possibility of installing Windows Embedded CE operating system on S3C2440SBC board. S3C2440SBC board provides a rich set of communication interfaces: three RS-232 interfaces, four USB 2.0 and one RJ-45 Ethernet. In the developed model of processor network the RS 232 interface is used to implement communication between the processor modules while the Ethernet interface is used for communication with external elements in relation to the network of processors.

To implement communication through the Ethernet interface the mechanism uses NDIS network drivers. Network driver interface specification is implemented in Windows® as a library, which defines interfaces between different layers of drivers and separates hardware drivers (low level) from upper layer drivers such as transport layer (Phung, 2009). NDIS also stores information on the status and parameters of the network drivers including indicators for functions, handlers and other values.

NDIS distinguishes the following types of drivers (see Figure 10):

- Miniport driver;
- Intermediate driver;
- Protocol driver.

The protocol driver is the highest in the stack of the NDIS driver and at the same time it is the lowest located component in an implemented network protocol. The protocol driver

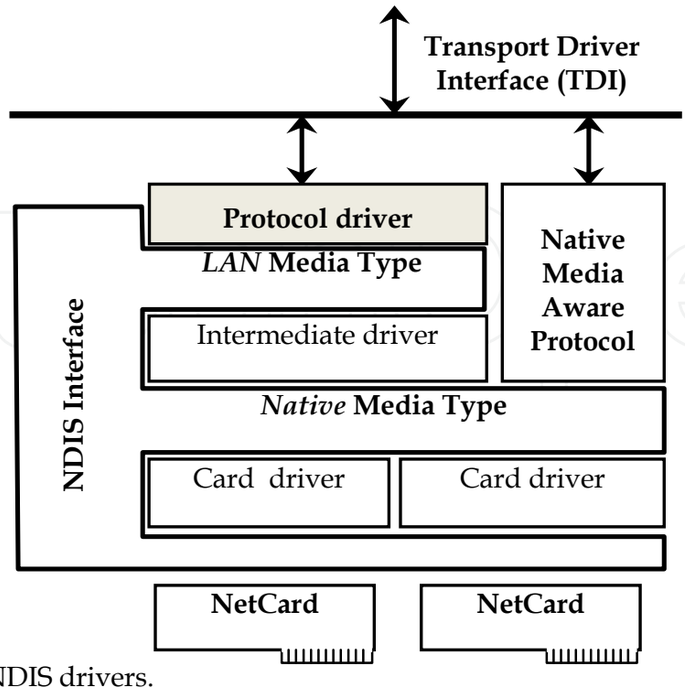


Fig. 10. Types of NDIS drivers.

allocates suitable memory area for the packet, copies data from the application to the prepared packet and - by calling the NDIS function - sends it to the network adapter. It also creates an interface for incoming data from the network adapter and sends them to the application.

Cooperation with other elements of the system is implemented using *ProcolXxx* functions, which constitute the interface for drivers situated lower in the stack. The protocol driver works with situated lower miniport or intermediate drivers in the stack that export a set of *MiniportXxx* functions. Transfer of packets by this driver is realized through the NDIS library by calling the appropriate functions. For example, functions *NdisSend* and *NdisSendPackets* can be used for sending packets.

The network software architecture with division on software layers is shown in Figure 11 (Zieliński et al., 2011).

In the operating system layer it is included a software layer which enables direct access to communication interfaces. In the communication software layer the dynamic library (*.dll) was realized to make available SEND() and RECIVE() functions. These functions enable sending and receiving messages in homogeneous manner - independently of physical interface.

The SEND() function makes it also possible to send broadcast messages that are used for broadcasting a new configuration of a degraded network structure. In the "Network Reconfiguration software" module the method of simple chains determining is implemented which is presented in Section 3. The structure of communication software layer is shown in Figure 12.

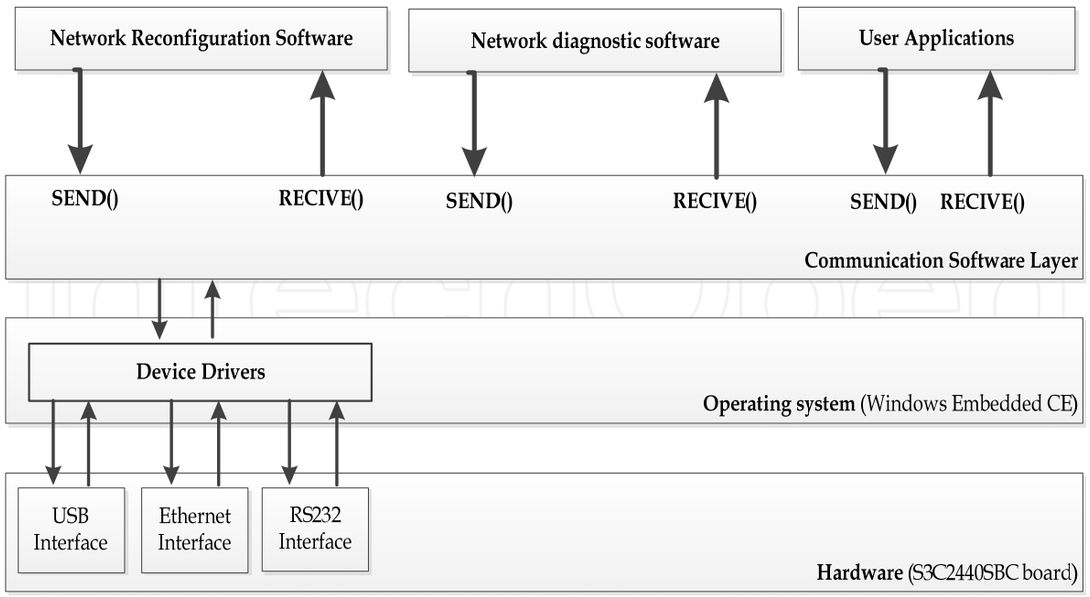


Fig. 11. The Network Software System Architecture.

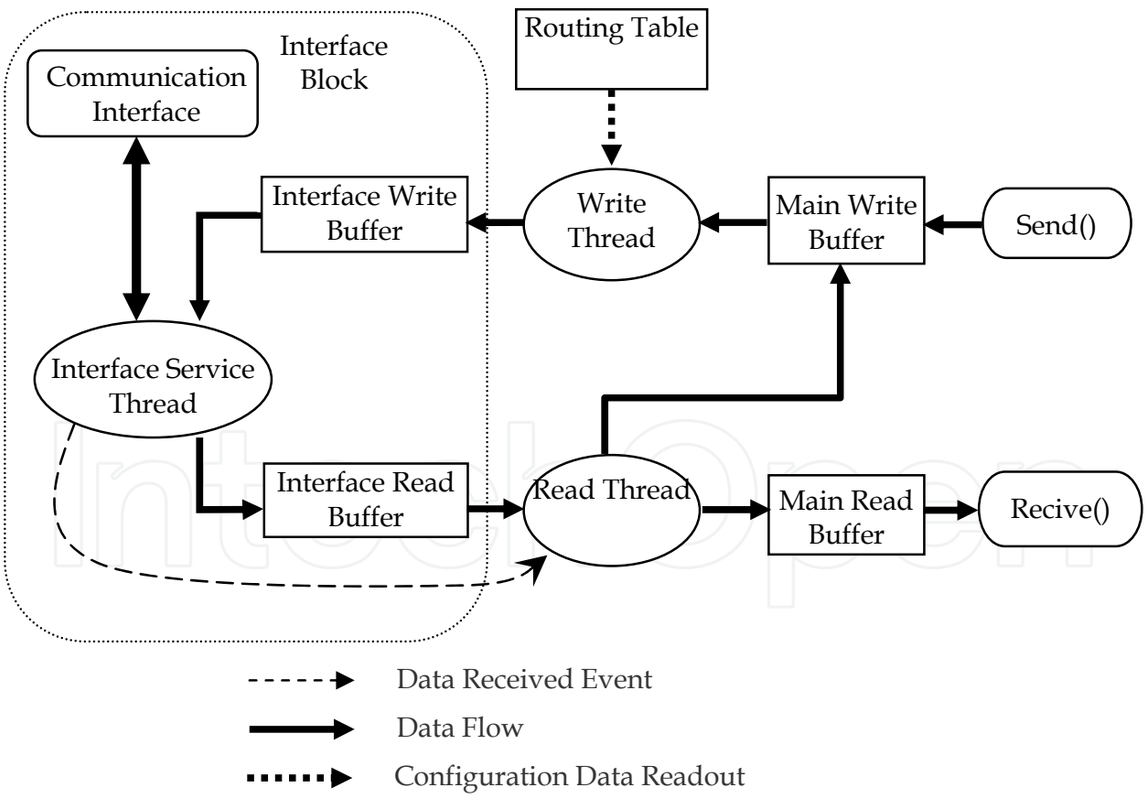


Fig. 12. Block diagram of the software communication layer.

Interface block includes hardware and software components that support a single interface. The notion of interface determines both a network card, as well as RS232 or USB 2.0. The

number of blocks depends on the number of active interfaces. Identification of the active interface is realized at the stage of initial system configuration.

Write thread performs operations of: data download from main write buffer, adding additional information to a data packet and forwarding the package to the interface write buffer. Choosing an interface to be used to send a package is realized based on destination host ID and Routing Table.

Read thread performs operations of: data download from the interface read buffer and forwarding the data to the Main Read Buffer. Data read is performed after receiving an event – "data ready" generated by the handler of thread interface. In the case of a broadcast message, this data is copied back to the Main Write Buffer. This allows sending data to other network elements.

5. Conclusions

Designed method of data paths reconfiguring allows determining parallel data paths in degradable hypercube processor network. In view of the searching independent parallel data paths transmission in a distributed manner and small computational overhead the method may be used in the systems with high performance requirements and due to possibility of adapting solutions to the current reliability state also in fault tolerant systems.

The method was implemented in the experimental 4-dimensional hypercube processor network. The photo of this network with running modules is shown in Figure 12.

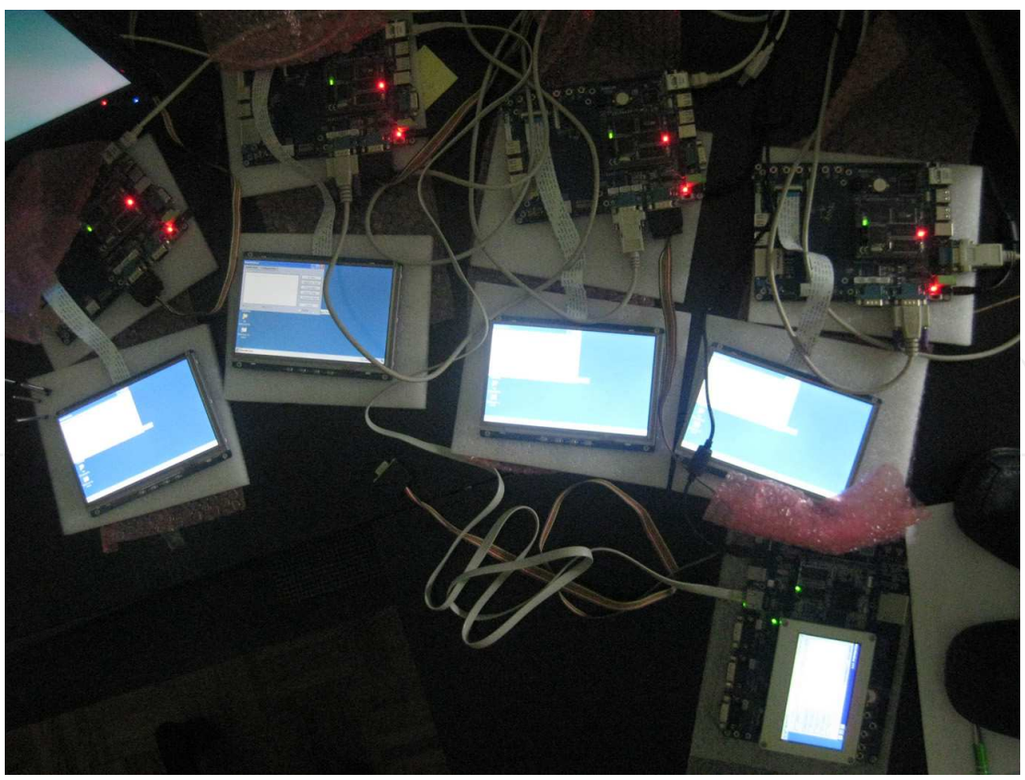


Fig. 13. The experimental network operating.

The appearance of the user interface shown in Figure 14. The procedure of diagnosing comparison method is periodically run in that network. After identifying damaged processors the reconfiguration algorithm is run on the set of fault-free processors. Currently, are realized adaptive tests of the experimental network model. Preliminary results of these tests show that the diagnostic messages are no more than 10% of all traffic in the network and time of the structure reconfiguration does not exceed 100 ms. Further work are directed to determine a degradation characteristics of the 4-dimensional hypercube processor network. Analytical methods were determined. Sets and images of all non-labeled coherent structures of order p where $p \in \{6,...,16\}$ and the powers of sets of labeled structures was determined by analytical methods. On this basis, it will be possible to build software tool to determine the cycle of life of such a network. The life cycle of the network can be expressed as a probability that the network keeps communication skills between defined sets of I/O ports and certain diagnostic properties after damaging the k processors $(k \in \{1,...,10\})$. Knowing degradation characteristics will allow selection of the best exploitation strategies of the network. The strategy consists of selecting the optimal (in the sense of communications capabilities, i.e. number of parallel data transmission paths) new working network structure and selection of a rational diagnosis method and tests.

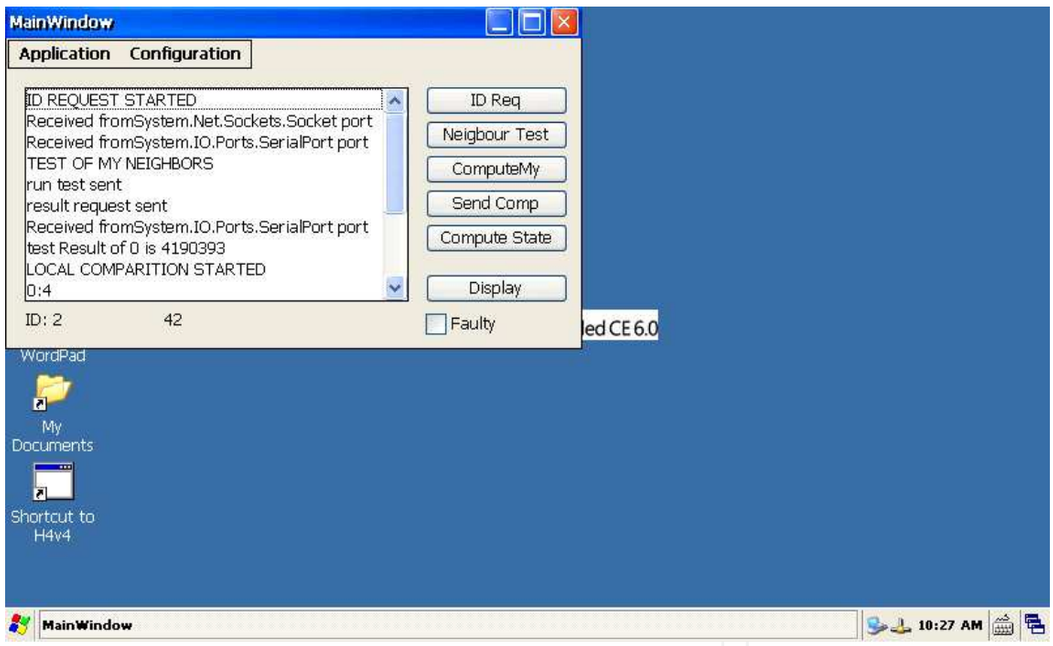


Fig. 14. Application User Interface.

6. References

Chen M.-S., Shin K.G. (1990). Depth-first search approach for faulttolerant routing in hypercube multicomputers, *IEEE Transactions Parallel and Distributed Systems*, vol. 1, no. 2, (April 1990), str. 152-159, ISSN: 1045-9219

Chudzikiewicz J. (2002), *Sieci komputerowe o strukturze logicznej typu hipersześcianu*; Institute of Automatics and Robotics, Faculty of Cybernetics, Military University of Technology, ISBN 83-916753-0-0, Warsaw, Poland

Chudzikiewicz J., Zieliński Z. (2003). Wyznaczanie *m*-diagnozowalnych struktur typu PMC w systemach o zwiększonej odporności na uszkodzenia, *Materiały X Konferencji SCR' 2003*, Ustroń, September 2003

Chudzikiewicz J., Murawski K. (2006). Determining A Non Collision Data Transfer Path In Hypercube Telecomunication Network, *Diagnostyka*, No. 3 (39), (2006), pp. 131-136, ISSN 641-6414

Chudzikiewicz J, Zieliński Z. (2010). Reconfiguration of a processor cube-type Network, *PRZEGLĄD ELEKTROTECHNICZNY (Electrical Review)*, NR 9, pp. 139-145, ISSN 0033-2097

Gordon J.M., Stout Q.F. (1988). Hypercube message routing in the presence of faults, *Proc. Third Conf. on Hypercube Concurrent Computers and Applications*, vol. 1, (Jan. 1988), str. 318-327

Kulesza R., Zieliński Z., Chudzikiewicz J. (1999). Reconfiguration of a ring structure in a hypercube computer network with faulty links; *International Conference on Technical Diagnostics 9th IMECO TC-10*, pp. 159-164, Wroclaw 1999.

Kulesza R. (2000). *Podstawy diagnostyki sieci logicznych i komputerowych (Secend edition)*, Institute of Automatics and Robotics, Faculty of Cybernetics, Military University of Technology, ISBN 83-9127747-6-4, Warsaw, Poland

Kulesza R. (2003). Struktury samodiagnozowalne w systemach cyfrowych, In: *Bulletin of Institute of Automatic and Robotics*, No 18, (2003), pp. 19-31, ISSN 1427-3578

Kulesza R., Zieliński Z. (2010). The life period of the hypercube processors' network diagnosed with the use of the comparison method, Monographs On System Dependability, In: *Monographs of System Dependability. Technical Approach to Dependability,* Sugier J., Mazurkiewicz J., Walkowiak T., Zamojski W., pp. 65-78, Oficyna Wydawnicza Politechniki Wrocławskiej, ISBN 978-83-7493-528-9, Wrocław

Phung S. (2009). *Professional Windows® Embedded CE 6.0*, Wiley Publishing, Inc., ISBN: 978-0-470-37733-8, USA, Canada

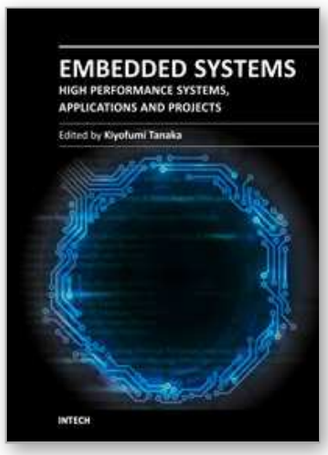
Sengupta A., Dahbura A.T. (1992). On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach, *IEEE Transactions on Computers*, vol. 41, no. 11, (November 1992), pp. 1386–1396

Wang D. (1999). Diagnosability of Hypercubes and Enhanced Hypercubes under the Comparison Diagnosis Model*, IEEE Transactions on Computers*, vol. 48, no. 12, (December 1999), pp. 1369–1374

Zielinski Z. (2006). The Simulation Model Of Distributed Network System Diagnostic Procedures, *Diagnostyka*, No. 3 (39), (2006), pp. 209-214, ISSN 641-6414

Zielinski Z., Chudzikiewicz J., Arciuch A., Kulesza R. (2011). Sieć procesorów o łagodnej degradacji i strukturze logicznej typu sześcianu 4-wymiarowego, In: *Projektowanie i implementacja systemów czasu rzeczywistego,* Trybus L., Samolej S., pp. 219-232, Wydawnictwo komunikacji i Łączności, ISBN 978-83-206-1822-8, Warszawa

Zieliński Z., Kulesza R., Strzelecki Ł. (2011). Diagnosability characterization of the 4-dimensional cube type soft degradable processors' network, In: *Monographs On System Dependability – Problems of Dependability and Modeling,* Mazurkiewicz J., Sugier J., Walkowiak T., Michalska K., pp. 283–296, Oficyna Wydawnicza Politechniki Wrocławskiej, ISSN 978-83-7493-612-5, Wrocław



Embedded Systems - High Performance Systems, Applications and Projects

Edited by Dr. Kiyofumi Tanaka

ISBN 978-953-51-0350-9
Hard cover, 278 pages
Publisher InTech
Published online 16, March, 2012
Published in print edition March, 2012

Nowadays, embedded systems - computer systems that are embedded in various kinds of devices and play an important role of specific control functions, have permeated various scenes of industry. Therefore, we can hardly discuss our life or society from now onwards without referring to embedded systems. For wide-ranging embedded systems to continue their growth, a number of high-quality fundamental and applied researches are indispensable. This book contains 13 excellent chapters and addresses a wide spectrum of research topics of embedded systems, including parallel computing, communication architecture, application-specific systems, and embedded systems projects. Embedded systems can be made only after fusing miscellaneous technologies together. Various technologies condensed in this book as well as in the complementary book "Embedded Systems - Theory and Design Methodology", will be helpful to researchers and engineers around the world.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen