IJE TRANSACTIONS A: Basics Vol. 27, No. 1, (January 2014) 101-112

International Journal of Engineering

Journal Homepage: www.ije.ir

On Feasibility of Adaptive Level Hardware Evolution for Emergent Fault Tolerant Communication

Y. Baleghi Damavandi*[a], K. Mohammadi[b], A. Upegui[c], Y. Thoma[d]

[a] *Department of Electrical & Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran*
[b] *Department of Electrical Engineering, Iran University of science & Technology, Tehran, Iran*
[c] *HEPIA - HES-SO, University of Applied Sciences of Western Switzerland, Geneva, Switzerland*
[d] *HEIG-VD - HES-SO, University of Applied Sciences of Western Switzerland, Yverdon-les-Bains, Switzerland*

PAPER INFO

Paper history:
Received 28November2012
Accepted in revised form 14 September 2013

Keywords:
Evolvable Hardware
Co-evolution
Genetic Algorithm
Emergent Communication

ABSTRACT

A permanent physical fault in communication lines usually leads to a failure. The feasibility of evolution of a self-organized communication is studied in this paper to mitigate this problem. In this case a communication protocol may emerge between blocks and also can adapt itself to environmental changes like physical faults and defects. In spite of the faults, blocks may continue to function, since a self-organized nature can provide self-healing capabilities. In the present paper, Evolvable Hardware is to create such a fault tolerant communication without any predefined protocol using a GA algorithm. Evolvable Hardware is a concept that aims the application of evolutionary algorithms to hardware design. The feasibility of this idea is studied in simulation of two reconfigurable blocks that are intended to transmit video streams through their communication lines. Permanent physical faults are induced in the communication lines between Evolvable Hardware blocks. Although the results show the emergence of fault tolerant protocols among Evolvable Hardware blocks without human intervention, there are some limitations in functional and gate level evolution of the blocks. Thus, a new adaptive approach is presented in this paper to defeat some limitations like the stalling effect of GA in faulty conditions.

doi:*10.5829/idosi.ije.2014.27.01a.13*

1. INTRODUCTION

In a modular system, it is very important to have a correct communication among the modules. A pre-designed protocol or language, if well-developed, would usually result in an optimized fulfillment of the purpose of communication. If it faces unpredicted defects in a varying environment, it may lose its efficiency. For example a short circuit between any of the communication lines of GPIB, IDE, ISA, PCI or LPT (IEEE 1284) protocols will undoubtedly lead to a failure. The problem addressed here, refers to the incapability of current communication protocols to endure the permanent physical faults.

This paper investigates the feasibility of a biologically inspired approach as a solution of the mentioned problem. In nature, the communicating blocks should have the following properties for the emergence of a language:

- Be intelligent enough to establish a language, and
- Be able to produce the correct physical signals.

Being intelligent enough is interpreted as being knowledge-based, purposeful and rational [1]. The first property (required intelligence) has been investigated by many researchers [2, 3], most of them using software blocks. However, when confronting the hardware approach, which is discussed in this paper, the second ability (producing correct signals) comes into sight. In human language, if not able to convey the meanings, one may shout, rephrase, and use gestures.... Here, the question arises that for the corresponding problem in hardware blocks (like the mentioned short circuit in Parallel communication lines) is it possible for the block to autonomously acquire the flexibility to explore other ways or tools for communication? The present paper reports and analyses some experiments to answer this question.

*Corresponding Author Email: y.baleghi@nit.ac.ir (Y. Baleghi Damavandi)

The blocks selected for simulation are Evolvable Hardware (EHW) modules, of which an introduction will be put forth in the following section. These blocks had been previously observed to establish a primitive self-organized communication [4] and thus assumed to be intelligent enough for this application.

The experiment selected for feasibility study is:

Without any predefined protocol the EHW blocks are set to reach the best fitness in an evolutionary process if they can transmit the video stream data using an emerged encryption. The benefit of this self-organized communication reveals itself when a permanent physical defect happens in communication lines.

While performing these experiments, the EHW blocks can be evolved in gate level [5] or functional level [6]. Functional level evolution is limited in the diversity of circuits that are searched, but is faster in finding simple solutions; however, the gate level evolution can be very slow, but has no limitation in the problem space and diversity of circuits that are searched. A new adaptive level evolution is proposed here for EHW that has led to emergence of communication in bridging fault conditions. This paper is organized as follows: Section 2 gives an introduction of EHW and then describes these blocks with both gate level and functional level evolution. The adaptive level evolution is presented in this section. The simulation setup is described in Section 3 and the results are discussed in Section 4 for each type of evolution level. A discussion is put forth in Section 5 to compare the evolved communication with the present error correction codes and the paper concludes and suggests the future work in Section 6.

2. EHW BLOCKS

EHW is a technique that targets the application of evolutionary algorithms to hardware design. EHW can adapt itself to unknown environments based on reconfigurable hardware features (e.g. FPGA). The work presented in this paper uses genetic algorithms as the evolutionary mechanism to search the goal architecture for hardware. The hardware architecture data is converted to chromosomes, where its fitness value will be derived in comparison with the target function, after the evolutionary process. EHW combines the powerful optimization tool of evolutionary algorithms with the flexibility of programmable devices, thereby providing a natural framework for reconfiguration. This framework has attracted interests in using EHW for fault-tolerant systems [7] because reconfiguration can effectively deal with hardware faults whenever it is impossible to provide spares and human intervention is restricted.

2. 1. Gate Level Evolution The purpose of this approach is to use EHW for the evolution of self-organized communication on the gate level. In addition, a sequential circuit is required inside the communicating EHW modules to transmit the data over their communication lines. To describe EHW with a chromosome, a form of finite state machine circuit as shown in Figure 1 [8] is used. This structure simplifies the definition of EHW chromosome to the combinational circuit.

In the co-evolutionary process, two hardware modules are supposed to be automatically designed (evolved). Here, hardware means the VHDL code of the EHW, which describes the EHW module. This description is derived from the genome and the predetermined architecture. The simulated architecture of EHW is inspired from a kind of PLD structure and is depicted in Figure 2.A chromosome with a fixed size has been used to describe EHW blocks. To represent the hardware block, the process of programming a PLD was simulated. First a *Testbed.Vhd* file was created that described all of the wired AND and wired OR connections in a PLD that have the potential to be programmed by burning the fuses. All these connections are disabled by *commenting*. The new binary chromosome is then interpreted in a way that a 1 means to *uncomment* the corresponding line where a 0 means to leave it disabled. With this method, the programming of fuses is simulated. Figure 3 illustrates this procedure.

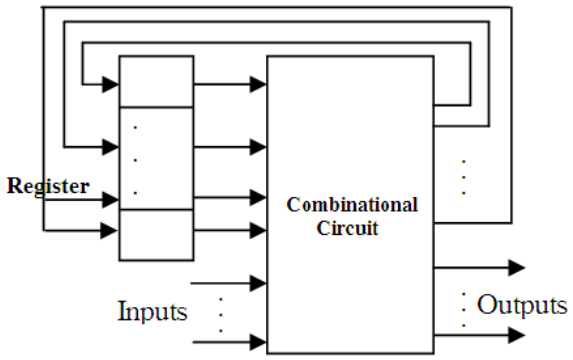

Figure 1. Representation of a sequential circuit with registers and combinational circuit

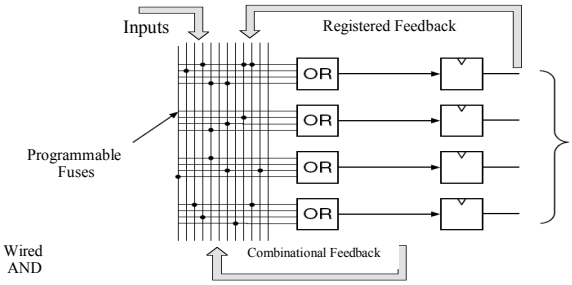

Figure 2. A kind of PLD architecture for description of the evolved hardware

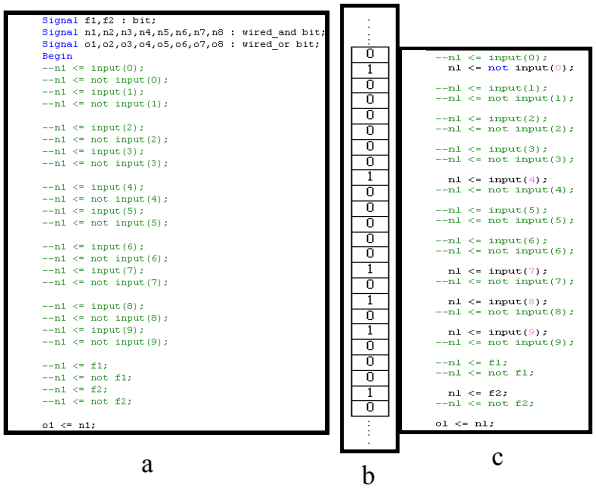

Figure 3. a. Part of *Testbed.Vhd* code that is simulating a raw architecture of a PLD. b, c. Part of a chromosome and its corresponding PLD configuration

Gate level evolution was simulated using the described method; however, other methods like functional level evolution are utilized to enhance the efficiency of EHW in improvising a communication as follows.

2. 2. Functional Level Evolution The size of circuits generated by gate level evolution can rarely be very large since GA is not scalable. It means that the size of the problem highly influences the performance of this algorithm and hence takes a long time to evolve large circuits [9]. This makes it difficult to use gate level EHW to produce hardware functions which are useful for practical applications.

If hardware is genetically synthesized from high level hardware functions instead of primitive gates, more useful hardware functions can be obtained in this way. Functional level EHW can be more useful for practical applications than gate level EHW [6].

To attain the goal of creating a communication between evolvable hardware blocks without the deficiencies of a PLD based gate level evolution -as mentioned in the previous subsection- one has to describe some building blocks that have determined functions.

While selecting or designing the blocks, we can reinforce the GA capabilities with the knowledge of a circuit designer to achieve a proper circuit in a shorter time. In this condition, what is lost is some of the high-range compatibility of GA to search all combinations of circuits. This is the point that will be discussed in the following section.

The building blocks that are to be used in EHW blocks for this application are: *Multiplexers*, *Decoders*, *Scramblers* and *Expanders*. The *Multiplexers* are to determine a 4-bit output from a combination of two data lines of the same length. The *decoder* block decodes a 4-bit input. A *scrambler* is defined here to have 4 input bits and it can scramble the order of these bits in the output and thus can have 24 configurations. How to scramble the bits will be defined in the dedicated part of the chromosome for this block. The *expander* block selects one of its four inputs to be copied in all of four output bits of the block. This block can be fully defined in a 2 bit part of the chromosome, since it has only 4 possible configurations. All of the four introduced blocks can have several configurations for different functions that are strictly defined in the chromosome which is generated by GA process of EHW. These blocks are located in a 2 dimensional space of 6*8 blocks and can have different types of routings simultaneously which is again defined in the chromosome.

It is clear that these blocks are designed with a preview and estimation of the requirements of a fault tolerant emergent communication. For instance a *multiplexer* can select one part of the input bits to be transferred while some transmission lines are damaged. An expander can select a bit to be transmitted by, at least, one safe line.

2. 3. Adaptive Level Evolution Considering the shortcomings of either of the functional and gate level approaches, another method is suggested here to be used in EHW evolution. This method is introduced on the basis of the functional evolution approach in the first hierarchy.

While all functional blocks will be used as mentioned in the previous subsection, all of them have a defined gate level structure. This structure can only be changed in a special process. In this process the stalling effect is monitored, and if it exceeds a threshold, the gate level structure of function can be changed in the second hierarchy as shown in Figure 4. The problem of stalling effect in fitness is related to the no improvement in the fitness values during the evolutionary process [9].

Here a variable length chromosome is required, because in the functional level evolution mentioned before, each block is just defined by its dedicated part in the chromosome behaviorally. This means that the chromosome just defines its behavior, not the hardware structure. But when it goes to the second hierarchy, the hardware structure of the block needs to be determined. For instance, the *expander* block that was defined in the previous subsection could be defined by a 2-bit part of the chromosome because it just needs to determine one of its 4 inputs to be expanded in all of its 4 outputs. But, as a combinational logic in the gate level hierarchy, a 4-input/4-output truth table (which can describe an expander) needs 64 bits in the chromosome to be fully defined. Thus, the chromosome is required to be expanded.

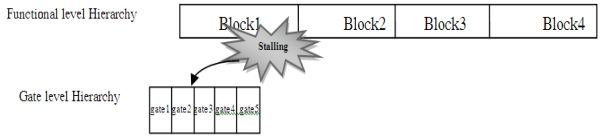

Figure 4. An illustration of a chromosome of EHW encountering a stalling that causes transition from functional level hierarchy to gate level hierarchy

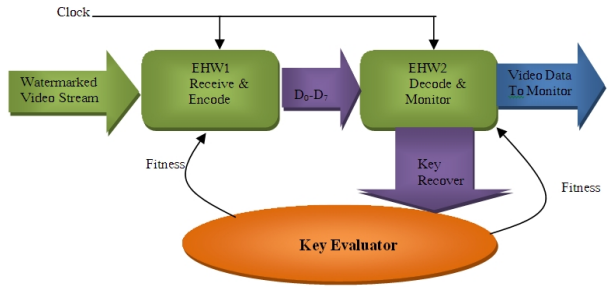

Figure 5. System diagram of gate level EHW-based frame grabber

That is why the Adaptive Level Evolution of EHW (ALE-EHW) needs a Variable Chromosome Length Genetic Algorithm (VCL-GA).

Considering the relationship between the GA time consuming feature and its chromosome length, one may deduce that the suggested approach will be more time consuming because it may enlarge the chromosome, but it must be noted that the chromosome will expand only if a stalling effect is reported, i.e. when there is no other way to find a good solution.

3. SIMULATION SETUP

To analyze the feasibility of the evolutionary approaches of different EHW levels to build up or modify a communication, a benchmark should be proposed. However, there is not a standard benchmark which is used by many references in this issue, so a video transmission case between two electronic boards is proposed in 3.1. The type of transmitted data (video data) does not seem to be very important; however, in this case the high amount of redundancy (like psycho-visual redundancy) gives the opportunity to use no-reference quality assessment for fitness evaluation process that is described in detail in 3.2.

3. 1. Video Interfacing as a Benchmark Based on the work referenced in [10], encouraging starts with simple serial adders, this section proposes a real world application to demonstrate the potentials of the self-organized communication with EHW blocks.

Like a frame grabber card which interfaces a video stream to the computer, two electronic boards are assumed to contain EHW modules to send and receive the video data. The video transmission task is implicitly defined for them so that a correct transmission will result in the best fitness in the evaluation process. Initially for transmission of each color, 8 ports of each board are connected to the other board for possible communication.

Figure 5 shows the system diagram to simulate the EHW modules communicating to each other to send/receive and monitor a watermarked video stream through an 8-bit parallel line D_0-D_7.

If a successful transmission is constructed by EHW module, then a fault simulation will be applied in the second step. In this stage, one (or more) communication line is subjected to a faulty condition based on a fault model (see 4.1.2). In this case, other parts (like EHW modules) are assumed to be fault-free.

3. 2. Fitness Issue Genetic algorithm as the optimization engine of EHW needs a fitness function. The required fitness is derived by video assessment using watermarked video stream in this work. A constant key is watermarked in every frame of the video stream. This method is based on digital watermark technology. Invisible reference information (watermarked key) is embedded into the video sequence, which may be corrupted during transmission and can be retrieved from the decoded video at the destination key evaluator. By the comparison of retrieved watermark with the original watermark stored in the destination, we are able to assess how much video sequence quality degrades during the transmission, so as to achieve our goal to assess the video quality without reference [11].

The evaluator will monitor recovered key from video data and will return its hamming distance (as the fitness) from the predetermined original key. This value will be supposed as the fitness value in genetic algorithm which determines the new configuration of EHW modules. Using the mentioned fitness function, each of the EHWs has to adapt, to score the best, in a co-evolutionary process. The key recovery process assures that video data has been recovered successfully. This way has been used for no-reference video quality assessment.

4. SIMULATION AND RESULTS

With the 3 types of aforementioned EHW blocks, 3 experiments have been performed and the results are analyzed independently. While the gate level and functional level evolutions show some good results to achieve the goal of the paper that is emergence of a communication between EHW blocks, the proposed method of adaptive level evolution shows better performance to deal with the stalling effect.

4. 1. Gate Level Evolution Gate level evolution was simulated in *Matlab* 7.5 software between two EHW modules and finally resulted in a correct transmission of video data. As shown in Figure 6, the input image is captured by the image acquisition toolbox of *Matlab* 7.5 software. The *link for modelsim* toolbox is also used to interface the image data from *matlab* to the *vhdl* code that is simulating the EHW modules (see 2.1).

4. 1. 1. Video Transmission The stream format is in YCbCr color space which has three 8-bit parameters to represent each pixel [12]. Thus, there are 24 lines as inputs to EHW1 and the same number of lines as outputs from it. A truth table that describes such a logic system needs: 2^{24}x24=402,653,184 bits of the chromosome for genetic algorithm that looks huge!
A way to reduce this chromosome length is to divide the unrelated bits and use individual EHW modules to encode each of them. That is why the EHW1 consists of 3 independent truth table modules, each one with 8 inputs and 8 outputs. This results in 2^8x8=2048 bits for each module that shows a good reduction, even when taking account of all three modules i.e. 6,144 bits.

The above mentioned structure, in an evolutionary process, produces the signals of Figure 7. EHW2 has the same raw architecture of EHW1, and is to co-evolve with it to reach the fitness of decoding the video. The evaluation process takes place after the output of EHW2 is ready and the original key is compared with the retrieved watermarked data.

The genetic algorithm used for this process had the following parameters: 100 random individuals for initial population, two-point crossover and mutation operators with Gaussian function, tournament selection mechanism and forward migration.

The D_0-D_7 interface signals as shown in Figure 7 will be generated by EHW1 and interpreted by EHW2 that can represent an emerging common language between the two modules. The color pixels are transmitted in YCbCr format where each pixel is defined by luminance, blue and red chrominance respectively.

The input image has 320x240 pixels per frame. Figure 7 shows the first 20 pixels of the first line of the image that is transferred by EHW1. As illustrated, this process needs 52 pixel clocks i.e. 1.6 times redundancy in comparison with the original data. This redundancy however is time consuming here, but as will be shown later, it will be useful for fault tolerance.

Finally, after 107 generations, the video stream was successfully transmitted and monitored at the destination. An example of transmitted image is shown in Figure 6. This figure shows the correct retrieval after the evolution time.

4. 1. 2. Fault Tolerant Communication In this subsection the idea of fault tolerance in EHW modules communication is put to test. For this purpose the previously described model will face different types of faults in the communication lines. Since the EHW modules are two electronic boards that communicate with each other via a connector, the fault model can contain related cases. A typical fault affecting the mentioned connections is a short between a pin and the power supply line in the connector. Considering the type of these pins, one of the following logical faults may happen. *Stuck at 1/0:* A short between ground or power and the signal line can make the signal remain at a fixed voltage level. The corresponding logical fault consists of the signal being stuck at a fixed logic value *v* *(v ε {1,0})*, and it is denoted by *s-a-v*. Since there are some power and usually more ground pins in such a connection, a short between them and the signal lines will cause *s-a-1/0* faults and thus they are set in the fault model.

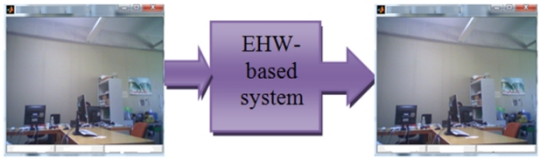

Figure 6. Interface signals produced by EHW1 to transmit the first 20 pixels of first line

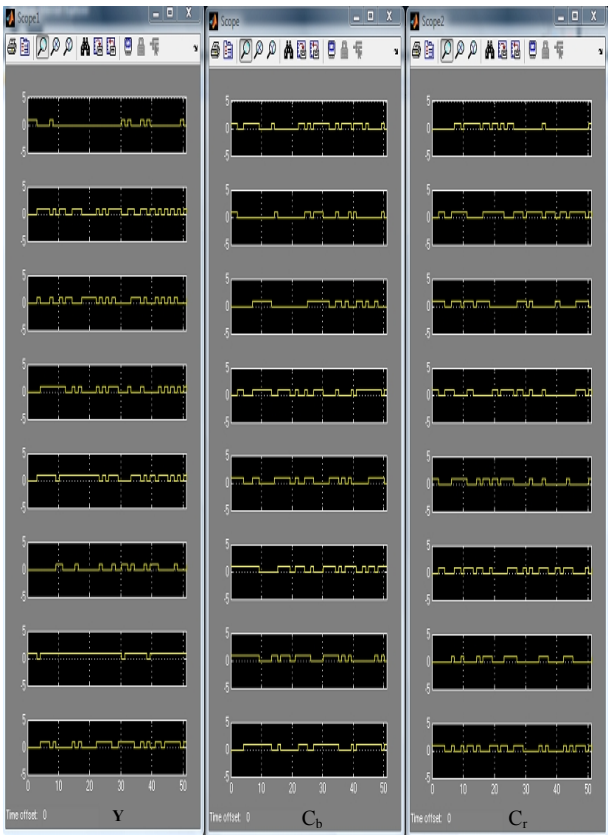

Figure 7. Interface signals produced by EHW1 to transmit the first 20 pixels of first line

In many cases, the effect of an *open* on a signal line with only one fan-out is to make the input that has become unconnected due to the open, assume a constant logic value and hence appear as a stuck fault [13]. Supposing a shielded, low length connection which brings about negligible radiation noise and delay faults, the introduced model can cover many kinds of physical faults that may happen in the EHW communication line. The main advantage of this system is gaining an intrinsic fault tolerance. The varying nature of EHW gives it an intrinsic fault tolerance [10]. Whenever a fault happens at communication lines that damages the video data, the key cannot be recovered correctly and it will lead to a degradation of the fitness. This will result in a change in genome, and consequently the hardware. The evolutionary loop will continue until it reaches the best fitness, i.e. the key and video data are recovered correctly. The video and D_0-D_7 signals in faulty condition are depicted in Figure 8 as an example of an *s-a-0* in Y line and *s-a-1* in C_b and C_r lines which all happened on D_0 line. When D_0 experience a stuck at 0/1 fault (shorts to GND/V_{supply}) the system will try to transfer the data without D_0 line. This may be more time-consuming, but what is gained instead of the time drawback is a correct function under faulty condition.
The signals in Figure 8 are again for conveying 20 pixel colors but with more pixel clocks, i.e. 80. In the Single Stuck Fault (SSF) simulations performed here, all combinations of $D_0 \ldots D_7$ from YC_bC_r channels are put to test and the results show an insensitivity of the system to fault occurrence in 4^{th} line of C_b! It shows a redundancy in this encryption (1.6 times) that made line 4 insensitive of fault. The same behavior is reported in EHW-based robot controllers [14]. In spite of the good results mentioned here, the system was not able to reach a good solution for bridging/multiple fault condition, i.e. when more communication lines fail to function correctly.

4. 2. Functional Level Evolution In Contrast to the gate level EHW introduced before, the functional level evolution experiments are performed intrinsically on the *Ubichip*.

Ubichip is a custom reconfigurable electronic device, capable of implementing bio-inspired circuits featuring growth, learning and evolution [15]. The *ubichip* is developed in the framework of *Perplexus*[16], a European project that aims to develop a scalable hardware platform made of bio-inspired custom reconfigurable devices for simulating large-scale complex devices.

The reconfigurable array part of *ubichip* is composed of a matrix of *ubicells* as shown in Figure 9. They have a 4-bit granularity. To set the stage to implement the communication of EHW blocks, two 4x3 parts of cells were determined as EHW1 & EHW2 that are to create the communication. A cell that consists of 4 Look Up Tables (LUTs) and 4 flip flops can simulate a 4 bit communication line (The communication line cell in the middle in Figure 9). The indicated input/output cells provide the input and output 8 bits streams to check the fitness value of the hardware.

A *Matlab* program was developed that provided the input test streams, the new configuration bits from the chromosome and commanded to run and checked the output stream to evaluate the fitness. Each *ubicell* can have a different structure. In this experiment, the cells consist of 4 independent LUTs. As described in Section 2.2, the individuals of the EHW blocks consisted of 4 main block functions that could vary their behavior using the dedicated part of the chromosome.

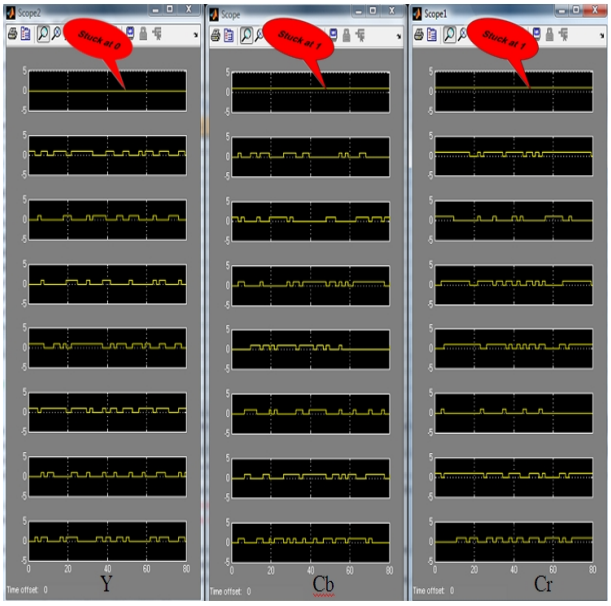

Figure 8. Interface signals between gate level EHW modules encountering faults

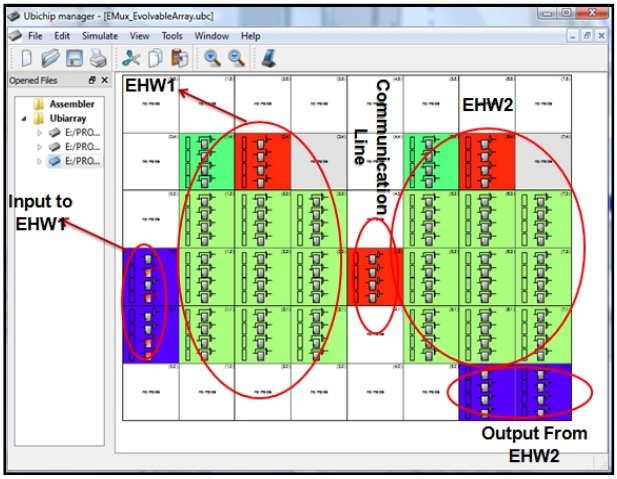

Figure 9. Functional level intrinsic EHW communication using *ubimanager* to evolve on the *ubichip*

Using the capabilities of this hardware, they also could be just a routing from each of their 8 neighborhoods to the other that was given a less probability in the GA process. The first 2 cells of each EHW (in the first row) are dedicated to the clock generator and its routing and have a fixed structure. The next cell is a control unit which provides the *select* signal for *Multiplexer* units. 48 bits define the truth tables of this unit in the chromosome. The remaining 3x3 part of cells is described by the rest of the chromosome. Since there are only 4 functional blocks, the first 2-bit part of the chromosome for a block describes the type of individual. The behavior of the block, like the order of selection in a multiplexer is defined by the next 5 bits. The next 16 bits define the routing from the block. The block can let all of the neighbor cells to access its output, or it can transmit one of its inputs to any of the neighbor cells. That is why so many bits are required for determining the routing structure. The last 4 bits are to define which of the neighbor cells to be selected as the input. Thus, the total bits to define one block are 27 bits. The sum of 9 cells of one EHW and the control unit bits is 291 long bit chromosome. The chromosome structure is described in Figure 10.

The same evaluation system that was set for the gate level evolution can be set in this configuration, because the EHW parts are only involved in the bit transmission process and the other parts (like fitness evaluation, ..) are performed in the same *Matlab* program, and so other details are the same and not repeated in this subsection.

The tests resulted in a configuration that was able to transmit the bit streams through the simulated communication lines. The obvious stalling effect happened when the experiment was being performed in the presence of faults without any success. The GA process that resulted in the emergence of the communication in non faulty condition is illustrated in Figure 11. Figure 11 also illustrates the best, mean and worst fitness and the average distance between individuals in each generation, and the percentage of stalling generations in comparison with the total number of generations.

Corresponding to the structure of Figure 9, the simplified evolved circuit is shown in Figure 12. A single path in this schematic may pass from 2 or 3 cells in the real evolved circuit. The control unit is a sequencer that provides the select signal for the *Multiplexers* to be able to transmit each 4-bit portion of data from the communication cells alternatively. Because of the 4-bit granularity of *ubichip*, all buses in Figure 12 are 4-bit wide and the communication cells provide the same input for Mux2 and Mux3 from two separate paths. In the gate level and adaptive level evolution approaches, the long chromosome cannot be easily translated to a circuit that can be drawn in this paper.

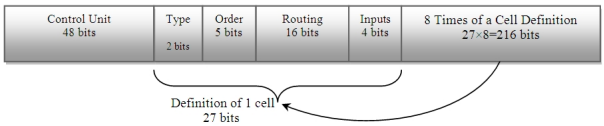

Figure10. Chromosome structure in functional level evolution

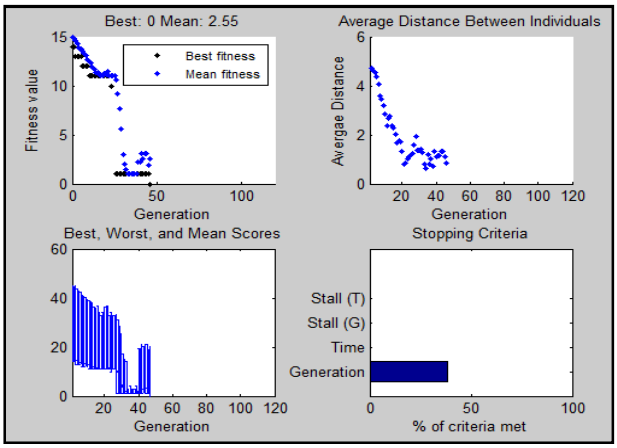

Figure 11. Genetic algorithm process for functional level EHW modules

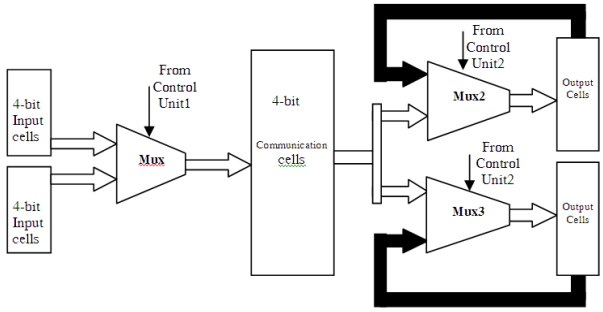

Figure 12. The evolved circuit in *ubichip* in functional level evolution

Even in non-faulty condition, the stalling effects can be observed in Figure 11 during the GA process. This may indicate that a functional level evolution is more likely to experience stalling effects however they can reach best fitness faster than the gate level evolution in case of finding an answer. The better speed of functional evolution (42 generations) method can be shown by comparing the gate level evolution No. of generations (107 generations) to reach the best fitness. On the other hand, the stalling effect is not very obvious in gate level evolution. However, the same effect prevented the functional level evolution to reach at least one solution for faulty condition in EHW communication and thus we can say it failed to reach a good fitness. These results could be foreseen from the structure of functional level evolution. Since the blocks are selected by a circuit designer with a special solution in his mind, this solution may be found easily but the

diversity of the problem space is reduced and the compatibility that is a common feature of bio-inspired algorithms is lost. That may be why the functional level evolution was not capable of finding a good solution for faulty communication, while there existed a solution with the present blocks!

4. 3. Adaptive Level Evolution Considering the features of the two previous methods, for adaptive level evolution that is put forth, the theory is to have the speed of functional level evolution while keeping the compatibility to find further solutions in a more diverse space like gate level EHW blocks. The block individuals were specified the same as functional blocks, but with the ability to change the internal hardware structure as it is described in 2.3. The experiment was an extrinsic EHW simulation. The GA would break through the chromosome to define a new block when the stalling effect exceeds 10 generations of fixed best fitness. The 10 generations threshold was empirically chosen since usually no improvement was observed after 10 generations of stalling. Using this approach, the system was able to reach the best fitness in all faulty conditions (that logically had a possible solution). Of course, the process was very time consuming because sometimes the chromosome is enlarged more than 10 times, but what is gained is a correct answer which can let a communication survive in a very harsh environment. An important issue is about the analysis of the encrypted data. This data encryption has some redundancy. This redundancy increases when the complexity of the evolved circuit increases. This redundancy can be interpreted like this: A single byte is represented by more than one byte and thus its transmission takes more clocks. Assuming that the YCbCr colors are being transferred, Table 1 shows that the normal YCbCr codes are mapped to 2-byte codes that are named evolved Y_e, Cb_e, Cr_e. This table shows the corresponding 2 bytes codes that are evolved in the adaptive level evolution for communication between EHW blocks. Table 1 shows that the package length of the color data has been duplicated (one byte is added in correspondence of each byte); however, 1.6 data redundancy was formed after this evolution (As it was in section 4.1.1 where 20 pixels were transferred by 52 clocks). That is because some other bytes were repeatedly found in the data stream that can be interpreted as synchronization bytes. While analyzing the evolved encryption, this repeated stream of bytes was observed in every line. This was surprisingly familiar to the SAV (start of active video) and EAV (end of active video) bytes that are being used in the designed standards like ITU656 [17]. Figure 14 shows one of these evolved SAV_e and EAV_e streams. These bytes help the receiver side to synchronize each line with the sent data.

TABLE 1. Some of normal YCbCr (A pixel data format for transmitting color pixels) codes and their corresponding evolved codes that are transmitted via communication line

Y	Cb	Cr	Y_e	Cb_e	Cr_e
112	32	212	143-127	72-234	168-45
203	156	107	232-190	124-117	203-105
102	214	45	132-152	208-167	20-13
0	89	20	0-0	67-234	214-98
53	217	135	67-90	232-240	117-15
187	245	112	89-0	230-15	103-76
76	208	145	123-45	216-98	129-65

TABLE 2. Fault analysis in Hierarchical level evolution of EHW blocks. D_0-D_7represent the parallel communication lines corresponding to Figure 5. S@0-1 shows the type of injected fault according to subsection 3.1.3. "No Fault" represents simulation without existence of faults.

S@0	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**	**No Fault**
Generations	142	163	125	155	135	147	138	151	115
Redundancy	4	4	4	4	4	4	4	4	1.6
Bridging AND	D0D1	D0D2	D0D3	D0D4	D0D5	D0D6	D0D7		
Generations	176	138	185	193	168	185	177		
Redundancy	4	4	4	4	4	4	4		
S@1	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**	**No Fault**
Generations	136	142	148	183	165	109	122	115	115
Redundancy	4	4	4	4	4	4	4	4	1.6
Bridging OR	D0D1	D0D2	D0D3	D0D4	D0D5	D0D6	D0D7		
Generations	176	168	122	189	178	197	177		
Redundancy	4	4	4	4	4	4	4		

TABLE 3. Comparison of the 3 types of evolutions properties that are simulated in this work

	Gate Level Evolution	Functional level evolution	Hierarchical level evolution
Average No. of generations to emerge a communication (without fault)	107	42	115
Ability to recover from single stuck faults	Yes	No	Yes
Average No. of generations to emerge a communication in single fault situation	120	No	142*
Ability to recover from bridging faults	No	No	Yes
Average No. of generations to emerge a communication in bridging fault situation	No	No	174*
Ability to recover from multiple faults	No	No	No

*The average No. of generations are rounded

Figure 13. Evolved SAV_e and EAV_e in Hierarchical level EHW communication protocol

Since the adaptive level evolution approach was able to tolerate more faults, the fault model could be expanded to wider types of faults than mentioned in 4.1.2. The faults added to the model are the bridging OR/AND faults as follows:

Bridging AND/OR: A short between two signal lines in the connector creates a new logic function. The logical fault representing such a short is referred to as a bridging fault. According to the function introduced by a short we distinguish between AND bridging faults and OR bridging faults. For example, if the signals are derived by open collector gates, a short between them will cause AND function, where the ECL ones will lead to OR bridging faults.

Table 2 shows the single and bridging faults that were injected to the adaptive level evolution EHW blocks, their generations and the amount of redundancy that is added to the evolved encryption. The number of redundancy means how many times the transferred data has increased in comparison with the standard form of the data of one pixel.

5. DISCUSSION

Considering the 3 types of simulations that are introduced in Section 4, the discussion can be categorized into 3 following categories.

The first category presents a comparison between the 3 aforementioned simulations. The second category compares the whole approach of this paper with other fault tolerant protocols; however, in the third subsection, the outcomes and circuits of the proposed approach are analyzed.

5. 1. Comparison of Experiments All the 3 experiments have the same inputs, outputs and targets that are described in Section 3, but as it is implied in Section 4, the gate level and adaptive level evolution experiments are extrinsic EHW simulations and their evolutionary loops are simulated in a software environment; however, the functional level experiment is implemented in Ubichip.

Basically, using extrinsic or intrinsic EHW does not affect the results of the experiments; however, as it can be predicted, the intrinsic EHW experiment was less time-consuming thanks to the appropriate architecture of Ubichip. Of course, this is just about the time that it takes for the period of each generation and should not be mistaken for the number of generations that is totally independent of the hardware platform.

Apart from the implementation details, the 3 types of experiments resulted in simple communications that were different in fault tolerance and time-consuming properties. Table 3 classifies the simulations using these properties.

A comparison between the proposed methods shows that the gate level evolution is able to find a solution when 1 single fault happens in the communication lines, but all solutions (even the non faulty conditions) are very time consuming.

The functional level evolution observes the emergence of communication much faster than a gate level one, but it is absolutely incapable to find a solution for faulty conditions that need a more complex structure for communication.

The adaptive level evolution can find a simpler structure for non faulty communication not much slower than the gate level one, while it is able to create a communication in faulty conditions which is a unique property for bridging fault lines in this experiment. However, in multiple faults situation (more than one simultaneous stuck fault) none of the mentioned approaches were able to find a solution in a reasonable time.

5. 2. Comparison with other Methods To compare the present conventional methods that can be used for handling bus errors with the proposed

approach, it is required to classify the common methods from the fault tolerance point of view, which is as follows:

I. **Using Error Correction Codes** In a predesigned fault tolerant code like hamming code, 3 bits are required to be added to a 4 bit data to immunize it from single bit fault, and thus 3 parallel lines are required as the redundant hardware. However, in the proposed method, the number of communication lines are fixed and cannot be changed (A clear situation can be space applications, where human intervention may be impossible and the type of fault may be unanticipated too). The only way to tolerate such a fault is to change the hardware structure of sender which is the unique capability of this variant encryption.

II. **Permanent Fault Tolerance** Consider that many of the communication protocols avoid using error correction codes because of their heavy redundancy. They usually try to just detect the error and resend the data in case of a failure. In this way, they can just cover the transient faults, while the proposed communication can cover the permanent faults too.

III. **Rerouting in Networks** The objective of a fault-tolerant routing strategy is to get a message from source to destination despite a subset of the network being faulty. The basic idea is simple: if no shortest or most convenient path is available because of link or node failures, reroute the message through other paths to its destination [18]. There are many rerouting strategies, but in all of them the size and the structure of the packet of data remain unaffected. However, in the proposed approach, the size or structure of the data may be broken (using a kind of evolved encryption) to use the correct communication lines.

IV. **Conventional Fault Recovery Process** The classic fault tolerant algorithms contain fault detection, location and recovery. For example, in classic fault detection, a Test Pattern Generator is required. However, in this approach, the evolutionary process does not investigate the origin of fault but just searches an encryption that can successfully transmit the data.

Another important issue is the way of recovery from fault. Using the conventional method, the designer should think for a way of recovery from each fault (for example, a bridging fault between the lines) and save it for possible operation in the system. However, in the proposed method, this is the task for EHW modules. It is notable that the proposed communication is a variable encryption which can vary in different conditions. For example, if no fault happens, the encryption will not restructure, and can work in its most optimum mode.
On the other hand, the advantage of EHW-based communication is the ability to tolerate unanticipated faults [19].

5. 3. Analysis of Results A systematic overview of the evolved systems in all 3 experiments shows that what is finally achieved by the EHW modules is a special type of serializer for EHW1 and deserializer for EHW2. This may seem to be the evolution of simple digital blocks, but in a system that human intervention is not possible, even such modifications can be useful.

It can be supposed that in fault-free condition a parallel communication is set that is fast, but when encountering faults, the evolutionary process tends to use less communication lines or encrypt the data (especially in bridging fault) which leads to a semi-serial communication that is more time-consuming, but still correct. So, the only application that makes this experiment reasonable is an unmanned application. Apart from the previous application, the emergence of a communication without any predefined protocol (like what occurred in no-fault conditions) between EHW modules may be an interesting phenomenon; however, a real world application cannot be imagined for it in near future, because it is not optimized in comparison with other designed protocols (except that it has no designer and thus no license is needed to use it!)

Finally, as a feasibility study, the experiments and results are studied for this particular case to show that the emergence of a communication is just possible between evolvable hardware blocks.

6. CONCLUSION AND FUTURE WORK

Evolvable Hardware is an appropriate test bed for implementing the idea of self organized communication. Perhaps it is because of the role of evolution in creating the first primitive languages in the real world. The presented experiments on the basis of EHW led to a compatible varying communication that has the following preference to other protocols:

Most of the communication protocols in computer peripheral devices are not immune against the **permanent** faults that may happen in the communication channel. For example one can assume a stuck at 1 fault in one of the communication lines of GPIB, IDE, ISA, PCI or LPT (IEEE 1284) which undoubtedly will lead to a failure.

But, the evolved encryption in this experiment is capable of trying other structures to send the data and gain the best fitness. This structure may vary from each fault to another and even each generation to another one. It also may be very time consuming and not optimized to evolve to transmit the data in this way but

it still can function correctly where the other protocols absolutely fail.

Comparing this varying encryption with other encryption, it seems to be a good option when human intervention is not possible in faulty condition like the space applications.

In the process of evolving the blocks to organize such a communication, the traditional approaches of gate level and functional level evolutions face some limitations where the proposed adaptive level evolution shows to be more efficient.

Considering the features of the gate level and functional level evolution EHW blocks, the adaptive level evolution that is presented in this paper, features the speed of functional level evolution while keeping the compatibility to find further solutions in a more diverse space like gate level EHW blocks.

In this experiment one module was predefined as sender and the second one as receiver. An extension can be a simultaneous role of sender/receiver for both EHW modules which is in future work.

Another task from the list of future work is evolving the physical characteristics. Physical layer characteristics like voltage level and clock frequencies are supposed to be fixed in this experiment. Using reconfigurable analog devices [19] these physical characteristics can be evolved to gain more possible fault tolerance.

7. REFERENCES

1. Michael, P. G. and Huhns, N., "The emergence of language among autonomous blocks", ***IEEE Internet Computing Magazine***, (2000), 90-92.
2. Neubauer, "Emergence in a multi-block simulation of communicative behavior", ***Publications of the Institute of Cognitive Science***, Vol. 11, No., (2004), 19-4.
3. Thangavelautham, J., Barfoot, T. D. and D'eleuterio, G. M., Coevolving communication and cooperation for lattice formation tasks, in Advances in artificial life., Springer. (2003) 857-864.
4. Damavandi, Y. B. and Mohammadi, K., "Co-evolution for communication: An ehw approach", ***J. UCS***, Vol. 13, No. 9, (2007), 1300-1308.
5. Lee, J. and Sitte, J., "Gate-level morphogenetic evolvable hardware for scalability and adaptation on fpgas", in Adaptive Hardware and Systems, AHS First NASA/ESA Conference on, IEEE. (2006), 145-152.
6. Higuchi, T., Murakawa, M., Iwata, M., Kajitani, I., Liu, W., and Salami, M., "Evolvable hardware at function level", in Evolutionary Computation, International Conference on, IEEE. (1997), 187-192.
7. Hereford, J. M., "Fault-tolerant sensor systems using evolvable hardware", ***Instrumentation and Measurement, IEEE Transactions on***, Vol. 55, No. 3, (2006), 846-853.
8. Nelson, V. P., Nagle, H. T., Carroll, B. D. and Irwin, J. D., "Digital logic circuit analysis and design", Prentice-Hall, Inc., (1995).
9. Stomeo, E., Kalganova, T. and Lambert, C., "Generalized disjunction decomposition for evolvable hardware", ***Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on***, Vol. 36, No. 5, (2006), 1024-1043.
10. BaleghiDamavandi, Y. and Mohammadi, K., "Fault tolerance in co-evolutionary communication of ehw modules", ***Computers & Mathematics with Applications***, Vol. 57, No. 11, (2009), 1730-1735.
11. Fu-zheng, Y., Xin-dai, W., Yi-lin, C. and Shuai, W., "A no-reference video quality assessment method based on digital watermark", in Personal, Indoor and Mobile Radio Communications, 2003. 14th IEEE Proceedings on, IEEE. Vol. 3, (2003), 2707-2710.
12. Damavandi, Y. B. and Mohammadi, K., "Speed limit traffic sign detection and recognition", in Cybernetics and Intelligent Systems, 2004 IEEE Conference on, IEEE. Vol. 2, (2004), 797-802.
13. Abramovici, M., Breuer, M. A. and Friedman, A. D., "Digital systems testing and testable design", Computer science press New York, Vol. 2, (1990).
14. Greenwood, G. W. and Tyrrell, A. M., "Introduction to evolvable hardware: A practical guide for designing self-adaptive systems", Wiley. com, Vol. 5, (2006).
15. Upegui, A., Thoma, Y., Sanchez, E., Perez-Uribe, A., Moreno, J. M., and Madrenas, J., "The perplexus bio-inspired reconfigurable circuit", in Adaptive Hardware and Systems, AHS 2007. Second NASA/ESA Conference on, IEEE, (2007), 600-605.
16. Sanchez, E., Perez-Uribe, A., Upegui, A., Thoma, Y., Moreno, J. M., Napieralski, A., Villa, A., Sassatelli, G., Volken, H., and Lavarec, E., "Perplexus: Pervasive computing framework for modeling complex virtually-unbounded systems", in Adaptive Hardware and Systems, Second NASA/ESA Conference on, IEEE. (2007), 587-591.
17. BT, I.-R. R., "656-1: Interfaces for digital component video signals in 525-line and 625-line television systems operating at the 4: 2: 2 level of recommendation 601", ***Further reading***, (1998).
18. Koren, I. and Krishna, C. M., "Fault-tolerant systems", Morgan Kaufmann, (2010).
19. Greenwood, G. W., Hunter, D. and Ramsden, E., "Fault recovery in linear systems via intrinsic evolution", in Evolvable Hardware, Proceedings. NASA/DoD Conference on, IEEE., (2004), 115-122.

On Feasibility of Adaptive Level Hardware Evolution for Emergent Fault Tolerant Communication

Y. Baleghi Damavandi[a], K. Mohammadi[b], A. Upegui[c], Y. Thoma[d]

[a] *Department of Electrical & Computer Engineering, Babol Noshirvani University of Technology, Babol, Iran*
[b] *Department of Electrical Engineering, Iran University of science & Technology, Tehran, Iran*
[c] *HEPIA - HES-SO, University of Applied Sciences of Western Switzerland, Geneva, Switzerland*
[d] *HEIG-VD - HES-SO, University of Applied Sciences of Western Switzerland, Yverdon-les-Bains, Switzerland*

P A P E R I N F O

Paper history:
Received 28November 2012
Accepted in revised form 14 September 2013

Keywords:
Evolvable Hardware
Co-evolution
Genetic Algorithm
Emergent Communication

چکیده

یک نقص فیزیکی دائمی در خطوط ارتباطی بین دو عامل سخت‌افزاری معمولاً می‌تواند به عدم عملکرد صحیح منجرشود. در این مقاله به مطالعه‌ی امکان‌پذیری تکامل یک ارتباط خودسازمان‌ده برای غلبه بر مشکل ذکر شده پرداخته می‌شود. در این نوع ارتباط یک پروتکل تکاملی، به صورت خودبه‌خودی بین عامل‌ها تکامل می‌یابد که می‌تواند خود را با تغییرات محیطی مثل نقص‌های فیزیکی وفق دهد. برای تولید چنین ارتباطی در این پژوهش از سخت‌افزارهای تکامل‌پذیر استفاده شده است. سخت‌افزار تکامل پذیر با استفاده از الگوریتم‌های تکاملی در طراحی و تعیین ساختار مجدد سامانه‌های سخت‌افزاری پدیدمی-آید. عامل‌های سخت‌افزاری به‌کار رفته درابتحقیق ازالگوریتم ژنتیک برای ایجاد یک ارتباط تحمل‌پذیرخطا استفاده می‌کنند. امکان‌پذیری این ایده با شبیه‌سازی آزمون انتقال تصویر بین دو عامل سخت‌افزاری مورد مطالعه قرارگرفته است. در این آزمون نقص‌های فیزیکی دائمی به خطوط ارتباطی بین عامل‌های سخت‌افزار تکامل‌پذیر تزریق شده‌اند. نتایج اولیه‌ی‌شبیه‌سازی نشان‌دهنده‌ی تولید یک ارتباط خود سازمان‌ده و مقاوم در برابر خطا بین عامل‌های سخت‌افزاری است، اگر چه در سطوح گیت و کارکردی محدودیت‌هایی برای بازیابی در برابر خطا وجود دارد. برای رفع این محدودیت‌ها، یک ره‌یافت وفقی برای استفاده در سخت‌افزارهای تکامل‌پذیر ارائه شده است که برخی از مشکلات مانند اثر ایستایی در الگوریتم ژنتیک را بهبود داده است.

doi:*10.5829/idosi.ije.2014.27.01a.13*