An FPGA-based Fault Tolerance Hypercube Multiprocessor DSP System

Ahmad Falih AL-Allaf

Lecturer
Dept of Computer Eng.Technical College-Mosul
Ahmadalallaf@yahoo.com

Sabah waad nayif

Assit. Lecturer
Dept. of Computer Eng.Electronics college\ Mosul university
Sab_nayif@yahoo.com

Abstract

This paper describes a new proposed architecture for tolerating faults in hypercube multiprocessor DSP system. The architecture considered employs the TMS320C40 DSP processors as processing node. The system has a single spare DSP processor assigned to each cluster (a group of four nodes). Each pair of clusters share one FPGA unit connected to every node in the two clusters plus the two spare processors. The FPGA units in the system are devoted for data routing, data distributing (in real time processing), diagnosis, system reconfiguration and expanding. Every 3D hypercube has additional spare processors connected to FPGA device of that cube. The spare nodes are used in two stages to tolerate more than one faulty node in each cluster with a low overhead and minimum performance degradation. The system makes use 50% hardware redundancy in the form of spare nodes to achieve fault tolerance. The effectiveness of interprocessor communications and the mechanism of fault detection(for one and two fault) has been successively simulated using (Xilinx Foundation F2.1i) simulator.

Keywords: Fault Tolerance, Hypercube multiprocessor, TMS320C40, FPGA, DSP processor

منظومة معالجة إشارة رقمية, متعددة المعالجات بهيكلية المكعب الفائق, متسامحة الأخطاء باعتماد دوائر ألـ FPGA

أحمد فالح محمود العلاف

مدرس
قسم هندسة الحاسبات- الكلية التقنية-الموصل
Sab_nayif@yahoo.com

صباح وعد نايف

مدرس مساعد
قسم هندسة الحاسبات كلية هندسة الالكترونيات/جامعة الموصل
Ahmadalallaf@yahoo.com

الخلاصة

يصف هذا البحث مقترح جديد لمعمارية منظومة اشارة رقمية متعددة المعالجات بهيكلية المكعب الفوقي متسامحة الاعطال. المنظومة المقترحة تستخدم معالجات الاشارة الرقمية نوع **TMS320C40** كعقد معالجة في المنظومة. تحتوي المنظومة على معالج احتياط يخصص لكل مجموعة مكونة من اربع عقد معالجة. كل مجموعتين من العقد تشتركان بدائرة **FPGA** واحدة تربط بكل عقدة من عقد المجموعتين وكذلك تربط بالمعالج الاحتياط لكل مجموعة. دوائر ال **FPGA** في المنظومة تقوم بمهام تمرير البيانات (عند العمل بالزمن الحقيقي), تشخيص العطال, اعادة التشكيل والتوسع في المنظومة.

للسماح بمعالجة اكثر من عطل ضمن نفس المجموعة, يحتوى كل مكعب ثلاثي الابعاد على معالجين احتياطيين اضافيين يربطان الى دائرة **FPGA** في المكعب. النظام المقترح يستخدم مكونات مادية اضافية بمعدل **50%** لتحقيق تسامحية الاخطاء. واخيرا فقد تم عمل محاكاة لالية نقل البيانات في النظام المقترح وكذلك لالية كشف ومعالجة الاعطال التي تحدث في المنظومة .

Received 1 March 2008 **Accepted 24 April 2009**

1. INTRODUCTION

A hypercube multiprocessor systems have several interesting features which make them useful and popular as a general purpose multiprocessor system[1]. However, dependability and, in particular, reliability are important aspects in the designs of hypercube system and other complex parallel computing systems used for a wide range of applications in science, industry, and engineering. These aspects achieved by fault tolerance that incorporates mechanisms in the design of such systems to detect and localize errors as well as mechanisms to reconfigure the system and to recover from erroneous states.

In a fault tolerance multiprocessor system designs, the common method to obtain continuous execution and sustain the same level of performance in the presence of faults is to use spare nodes and/or spare links to replace the faulty ones[2,3]. Over the past 20 years, a number of fault-tolerant designs for hypercube multiprocessor systems have been proposed. Banerjee et. al. [4] used spare nodes and spare links to perform system reconfiguration. In his scheme, two spares are assigned to each 3-dimensinal cube and the spare processors form a (d-2) cube. Upon a node failure, the faulty node is replaced with the local spare. The link connecting the spare to the faulty processor and the link connecting it to the node diagonally opposite to faulty node is disabled. The system can only tolerate a single node failure.

Alam et. al.[5,6] have proposed two schemes to tolerate faulty nodes: in the first one; a single spare node is added to each cluster of four nodes. If one of the four nodes fails the spare replaces it and inherits its address. Therefore only one fault can be handled per cluster. In the second scheme, four spare nodes are added to each pair of clusters (eight regular nodes) which allows to handle two faulty nodes in each cluster. However, a more complicated routing algorithm is needed. Q. Mohmmad [7], proposed an adaptive fault tolerant routing algorithm to route a message around the faulty node in a connected tree-hypercube system instead of using spare nodes to achieve the fault tolerant design. To enable any non faulty node to communicate with any other non faulty node, the information on component failures has to be made available to non faulty nodes to rout message around the faulty nodes. In his scheme to route a message in presence of the fault, each node needs to know the condition of its own links only.

In this paper, we present a new fault tolerant n-dimensional hypercube multiprocessor system architecture for DSP applications. The system uses TMS320C40 DSP processor as a processing node and the FPGA devices to achieve the expandability and the reconfigurability of the system and to implement other tasks. The system employs the spare nodes to tolerate more than one fault in the processing nodes with minimum communication overhead. The proposed system uses the FPGA technology instead of the old techniques that are used in design of other fault tolerant multiprocessor systems such as switching network [14], replacement circuits, or graph-based bypass connection [15].

Most of these old techniques involve incorporating large number of external switches, control processor or complicated control circuit to detect and locate errors, reconfigure the system and recover from error.

The rest of this paper is organized as follows: In the next section an overview of the proposed architecture is presented. In section 3 we explain the mechanism of fault detection and reconfigurability of the system. The routing scheme and the general consideration in the design of the FPGA unit are described in section 4. In section 5 we present a simulation using VHDL language and (*Xilinx Foundation F2.1i*) simulator for inter processor communication and fault tolerance mechanism in our proposed system. Finally, system performance and concluding remarks are discussed in section 6.

2. SYSTEM ARCHITECTURE

Two general approaches can be used in design of the fault tolerance in hypercube multiprocessor system. The first approach looks into ways of employing the healthy processors and links of the hypercube with faulty nodes, to identify embedded topologies such as lower dimension hypercube (subcubes), meshes, ..etc. required by the computations. With this approach, some performance degradation is expected, however special hardware design for fault tolerance is not necessary. Once the faulty processors are identified, the hypercube multiprocessor and the algorithm running on the system must be reconfigured to run on the available processor.

The second approach makes use of hardware redundancy in the form of spare nodes and/or links to tolerate faults and usually requires modifications in the communication hardware. However, the algorithm running in the hypercube with the faulty node need not be modified. Also in this approach, almost no performance degradation occurs. The second approach is adopted in our proposed system.

In our design, a single spare node is added to each cluster of four regular nodes. Figure 1 shows a 3 dimensional hypercube with two clusters each is a sub cube of dimension 2. In general, the hypercube of dimension (n) can be divided into $2^{(n-2)}$ clusters and each cluster is a cube of dimension (2).

The proposed scheme employs the FPGA devices to enable the spare node to communicate with the neighboring nodes through the I/O port of the FPGA. The spare node then forwards/receives its data to/from other neighboring nodes via its communication ports and the FPGA I/O ports.

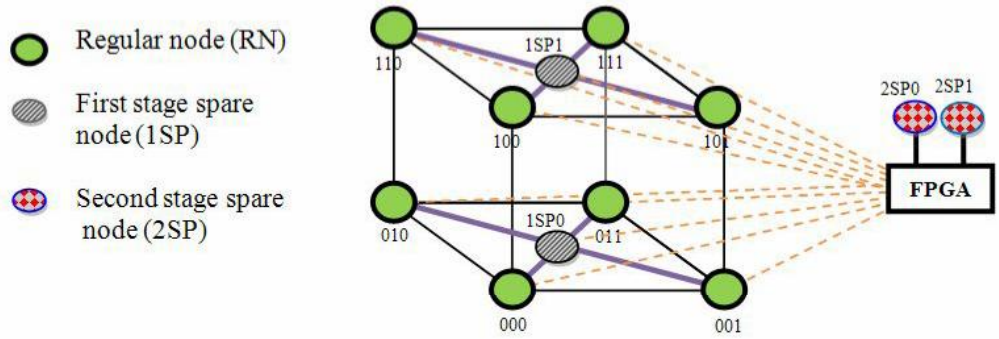

Figure 1 Proposed architecture of 3-dimensional (3D) fault tolerant hypercube system

Upon detecting a node failure, the spare node within the respective cluster logically replaces the faulty nodes and takes its address. When the processor fails in a cluster, the spare replaces it must be able to communicate with the neighboring processors in the local 3D cube as well as with the neighboring nodes in the extended hypercube. Therefore, it has to be linked to the neighboring processors in its local 3D hypercube as well as with the other adjacent nodes in the extended hypercube using FPGA units. For example, in figure 1, the spare 1SP0 may logically replace any one of the four nodes 000, 001, 010, or 011. If the faulty node is , for example, 001 then the spare 1SP0 replace it and take its address. In this case, 1SP0 communicates directly with the regular nodes 000 and 011. While it communicates indirectly with the node 101 through the FPGA I/O ports.

To tolerate multiple faults within the cluster, two additional spare nodes are added to each 3D hypercube and connected through the FPGA unit as shown in figure 1. In other words, the spare nodes in every 3 dimensional hypercube are used in two stages to handle multiple faults: at stage

one, the local spare node connected in each cluster (first stage spare node; 1SP) replaces any faulty node in that cluster. At stage two, if a fault occurs in any node of the same cluster, then one of the spares (second stage spare nodes; 2SP) connected to the FPGA with that 3D hypercube is used to replace the faulty one.

In this architecture we suggest to use the TMS320C40 DSP processors as a processing nodes which is the 4th generation DSP processor produced by Texas Instruments[8]. The TMS320C40s are dedicated to computing intensive task while the FPGA in every 3D cube is responsible, in addition to the data routing, for system expanding, reconfiguration in case of the node failure, system monitoring and real-time IO. The six built-in communication ports of the TMS320C40 are used to link the DSP processor to its adjacent-regular DSP nodes, to spare nodes and to the FPGA devices. Using TMS320C40, the processor and the communication port are integrated in one unit (not separated), therefore failure in the processor meaning failure in the communication channels that connect that processor to its adjacent nodes.

The present scheme can tolerate four faults in every 3D cube with a small performance degradation, due to communication overhead caused by the routing through the FPGA units, and the mechanism of fault detection as described next. The resultant configuration does not affect either the communication or computation algorithm already developed for the hypercube multiprocessor system. Figure 2 illustrate the architecture of 4 dimensional fault tolerance hypercube system.

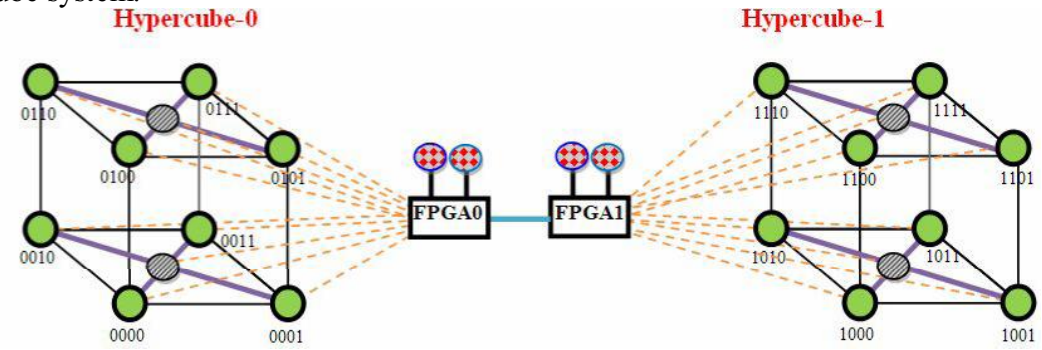

Figure 2 4-dimensional fault tolerant hypercube system

In this architecture, the adjacent nodes to node 0000, for example, are 0001, 0010, 0100 and through the I/O communication ports of FPGA0 and FPGA1 with the node 1000. The 5 dimensional hypercube system is constructed by connecting four 3 dimensional hypercubes as shown in figure 3. The FPGA units are also connected to form a cube in their own as shown in figure 5. However, in this architecture, to design n-dimensional extended hypercube we need $2^{(n-3)}$ FPGA devices.

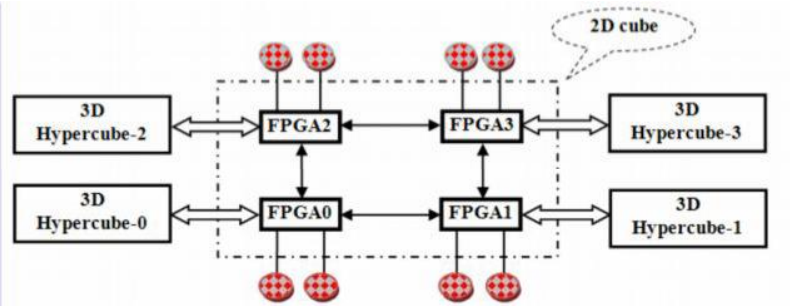

Figure 3 5 dimensional fault tolerant extended Hypercube system

3. MECHANISM OF FAULT DETECTION

The effective integration of fault tolerance into massively parallel systems requires a proper and cost effective combination of different error detection mechanisms. These mechanisms can be grouped into the following three techniques according to the level on which they are applied:

- *Self-checking* techniques in each processing node
- *Central checking* techniques for entire clusters of processing nodes
- *Distributed system-wide checking* techniques by mutual tests between nodes.

In the first techniques, the main emphasis focuses on the reliability of the computing core, as the most intensively utilized resource [9]. However, the basic checks integrated onto the chips alone proved insufficient for the construction of a reliable multiprocessor.
One of the important methods for checking the computing core is the *Watchdog processors* [10,11]. A watchdog processor is a coprocessor concurrently monitoring the program control flow either by observing the instruction fetch on the CPU bus or by checking symbolic labels explicitly sent from the main program. However, This solution to be cost effective, large multiprocessor should be based on sharing of the additional hardware between clusters of processing nodes. So the relative overhead per node can be kept low.

Another means to detect faulty nodes is given by mutual tests between nodes. The most widely used technique is based on the system-wide exchange of <I'm alive> messages (or *heartbeat*). This technique had first been implemented into the fault tolerant multiprocessors of Tandem [6].

In our proposed system and with every local 3D hypercube, the mechanism of error detection depends on using central checking technique represented by the diagnosis algorithm running in the FPGA and based on <I'm alive> mechanism. However, with respect to overall system (extended hypercube), the mechanism of fault detection can be regarded as a distributed checking mechanism. Since, every FPGA monitors the activity of their processors in a 3 dimensional hypercube connecting with it. Each processor must send <I'm alive> message to the local FPGA continuously every constant period of time. If the FPGA does not receive this message from any of the 3D hypercube processors within the predetermine time interval, then that processor is considered to be fail.

After the FPGA has detected the fault, it raises an interrupt to all the processor nodes in the system to go back one processing step and wait to reconfigure the hypercube to isolate the faulty node and replacing it with one of the spare nodes. After the system reconfiguration, the FPGA detecting the fault issues a FAIL message to all nodes in the 3 dimensional hypercube identifying which processor has failed. This message also propagates to all FPGA units in the system. And through each FPGA, the FAIL message propagates to every processor in its local 3 dimensional hypercube. The fail message contains information about the address of the faulty node and the address of the spare node replacing it. After reconfiguration, all the processors resume the processing from the stage prior the occurrence of the fault.

Every processor keep a record of the useful work performed in each period of time and in case of a processing node failure, after reconfiguration they are rolled back to a predetermined state. This predetermined state can be the point of receiving last <I'm alive> message. The spare node should containing a backup copy of the tasks running in each regular processor in the hypercube, therefore the computation module of spare node logically can replaces the computation module of faulty node.

4. ROUTING SCHEME AND FPGA CONFIGURATION

Efficient routing of message is a key to the performance of fault tolerance multiprocessor system and this is achieved by allowing fast reconfiguration in the present of the fault. Therefore the spare replacement of faulty components should result a very few changes in the system interconnections. To achieve this goal, each faulty processor should be replaced with the local spare as possible. In our proposed system, in order to enable non faulty nodes in a faulty hypercube to communicate with the spare node, it is required that each node to know only the conditions of adjacent nodes and the address of the spare node replacing the faulty node for making the correct routing path.

The FPGA provide a hardware environment in which physical logic and routing resources can be reprogrammed by configuring the device in order to perform a specific function. As a result, they provide an ideal template for dynamic circuit specialization and logic reconfiguration. The most important benefit of using the FPGA is significant reduction of the design effort compared to system specific interconnection network. FPGAs as Altera Stratix IS40 offer excellent platform for system on chips that can implement Intellectual property (IP) components as processor, hardware accelerators, memories, and communication interface…etc. The FPGA used in this system consisting of four Intellectual property (IP) blocks as shown in figure (4).

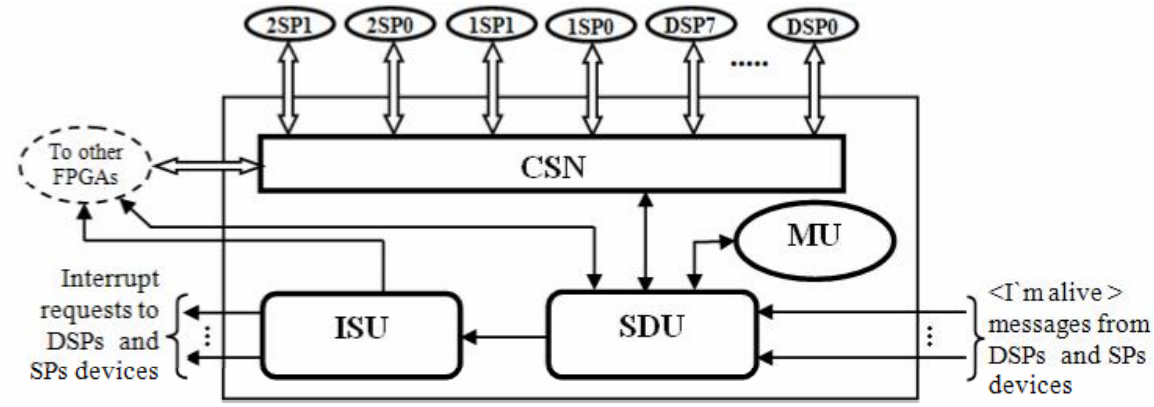

Figure 4 Architecture of the FPGA device in the proposed system

These components are: control circuit for system diagnosis (system diagnosis unit - SDU), Interrupt sending unit (ISU), crossbar switching network (CSN) and the memory unit (MU). The SDU is a control circuit responsible for overall control of the FPGA components. It can receive the <I`m alive> messages from the DSP processors sequentially. If one processor fails, then the SDU reconfigure the CSN to isolate the faulty processor and rearrange the interconnections between the processors to incorporate the appropriate spare processor. The CSN is a crossbar switching network connecting the 8-bit input/output bus of all the processors in 3D hypercube and the 1st and 2nd stage spares together. Also it connects the FPGA units in a hypercube network. In case of a failure, the configuration of the CSN can be done during the runtime. The memory unit is used to store the bit stream required to configure the CSN and all FPGAs in the system share the same configuration bit stream. Figure (5) shows the arrangement of the communication ports of the processors and the FPGA unit in the 3D hypercube through the CSN. Also the figure illustrates examples of configuration the crossbar switching network (CSN) in FPGA0 upon the failures in one, two three or four DSP processors.

Table 1 illustrates the required replacement and connection between the regular and spare nodes upon failure in one, two, three, and four DSP failures based on the examples that are given in figure 5 .

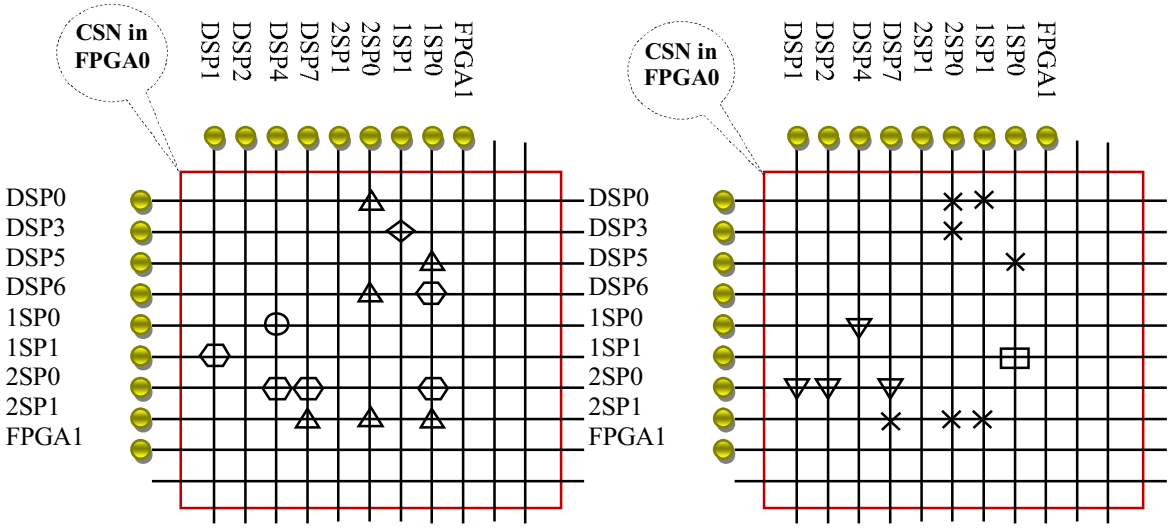

Figure 5 Configuration of the crossbar switching network (CSN) in FPGA0 upon :
a) one or three DSP failures b) two or four DSP failures

Where the following symbols are denoted to the connection through the crossbar switch if the failure(for example) in :
DSP0 *DSP7* *DSP0 and DSP4* *DSP0 and DSP3* *DSP2, DSP5 and DSP6*
DSP1, DSP2 and DSP3 *DSP1, DSP2, DSP4 and DSP6*

Table 1 Replacement and connection between the regular and spare nodes upon failure in DSP processors based on the examples that are given in figure 5 .

Faulty node(s)	Replaced nodes	Required connections through the FPGA
DSP0	1SP0 replace DSP0	DSP4 with 1SP0
DSP7	1SP1 replace DSP7	DSP3 with 1SP1
DSP0 and DSP4	1SP0 replace DSP0 and 1SP1 replace DSP4	1SP0 with 1SP1
DSP0 and DSP3	1SP0 replace DSP0 and 2SP0 replace DSP3	DSP4 with 1SP0 (DSP0, DSP2, DSP7) with 2SP0
DSP2,DSP5 and DSP6	1SP0 replace DSP2, 1SP1 replace DSP5, and 2SP0 replace DSP6	DSP6 with 1SP0 DSP1 with 1SP1 (DSP4, DSP7, 1SP0) with 2SP0
DSP1,DSP2 and DSP3	1SP0 replace DSP1, 2SP0 replace DSP2, and 2SP1 replace DSP3	DSP5 with 1SP0 (DSP0, DSP6, 2SP1) with 2SP0 (DSP7, 1SP0, 2SP0) with 2SP1
DSP1, DSP2, DSP4 and DSP6	1SP0 replace DSP1, 2SP0 replace DSP2, 1SP1 replace DSP4, and 2SP1 replace DSP6	DSP5 with 1SP0 (DSP0, DSP3, 2SP1) with 2SP0 (DSP0, 2SP1) with 1SP1 (DSP7, 1SP1, 2SP0) with 2SP1

5. SIMULATION

The simulation of the fault tolerance mechanism in the proposed systems has been accomplished using VHDL and (*Xilinx Foundation F2.1i*) software. The simulation is implemented for a 3D hypercube system, and during this simulation, the following assumptions have been considered:

-The FPGA device is considered reliable;
-The activation of the fault in a DSP processor results in the failure of service is carrying out and all its inputs and output ports;
- Each DSP fails independently from the others in the probabilistic sense;
- The time interval between two successive <I`m alive> messages (which represented by the code 5AH) is assumed to be equal to eight clocks;
-The messages consists of number of frames (each frame consists of eight bytes) are sent during the time interval between two successive <I`m alive> messages.
- In case of fault, all DSP processors return back one processing step, which is assumed the point of sending the last <I`m alive> message;
- Two or more faults appear to be simultaneous if their activations occur within the same time interval between two successive <I`m alive> messages, sequential faults occur during separate time intervals;
-The communication port of each DSP processor are named as A, B, C, D, E, and F.
- Each DSP processor connect to the crossbar switch in the FPGA through communication port-E ;
- Communication port-F in every DSP processors is devoted to send <I`m alive> messages to the FPGA device;
-Communication ports (A, B, C and D) are used for communications between the DSP processors.
-In the simulated system, processors are used without any synchronization or scheduling delays.

5.1 Simulating TMS320C40 Communication Port Protocol:

A communication port transmits each of the 32-bit words stored in its output FIFO on a byte-to-byte basis. Because the control and data lines are bidirectional, each ’C4x must have ownership of the communication port data bus before starting a word transfer. A simulated token is used to designate bus ownership: the communication port that has the token owns the communication port data bus and can transmit data. Figure (6) shows the interconnections between two TMS320C40 DSP processors[8]. One important feature of the ports is that they can work with the DMA coprocessor to transfer data without CPU intervention, allowing the CPU to perform other tasks.

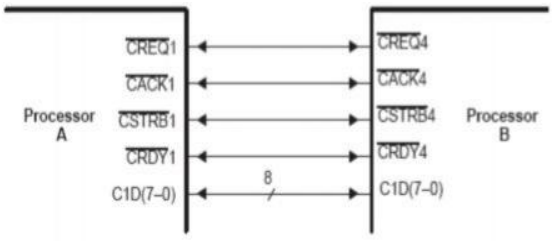

Figure 6 Communication between two DSP processors using communication ports

This simple communication interface consists of the following bidirectional control and data lines:

- CREQx — communication-port token request. A ’C4x activates this signal to request the use of the communication-port data bus.
- CACKx — communication-port token acknowledge. A ’C4x activates this signal to relinquish ownership of the communication-port data bus upon receiving a CREQx from another ’C4x.
- CSTRBx — communication-port strobe. A sending ’C4x activates this signal to indicate that it has placed a valid data byte on the communication port data bus.
- CRDYx — communication-port ready. A receiving ’C4x activates this signal to indicate that it has received a data byte via the communication port data bus.
- CxD(7–0) — communication-port data bus. This bus carries data bidirectionally, one byte at a time, between two ’C4xs or between a ’C4x and some other device.

A data transfer operation between two DSP processors using communication ports takes four basic steps to complete:

1) The CPU or DMA coprocessor of the sending DSP writes a 32-bit data word to the output FIFO (of a communication port) via a memory-mapped address.
2) The communication port then places the 32-bit data word on CxD(7–0) on a byte-to-byte basis (LS byte first), activating CSTRBx to signal the receiving communication port that the bus contains a valid data byte.
3) Upon receiving each data byte, the receiving communication port activates CRDYx to indicate that it has received the data byte.
4) After receiving the 4 bytes of a 32-bit word, the CPU or DMA coprocessor of the receiving DSP can then read the data from the input FIFO via a memory-mapped address. Each of the input and output FIFOs can buffer a maximum of eight 32-bit words. This operation was simulated using (*Xilinx Foundation F2.1i*) simulator as shown in figure 7, assuming that the token transfer operation has been asserted.

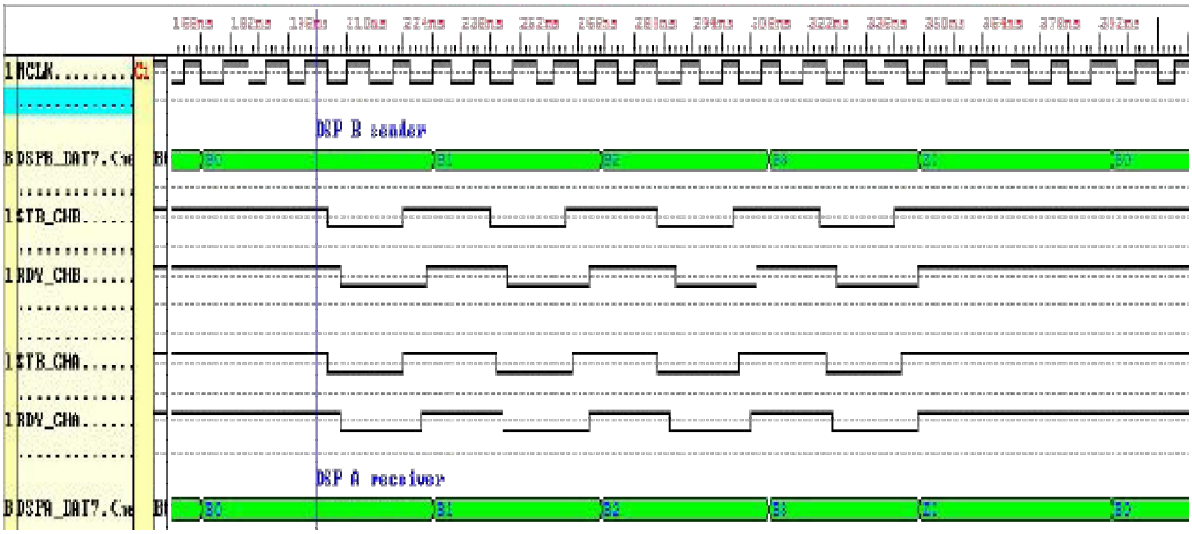

Figure 7 Simulation of data transfer operation between two DSP processors using communication ports

In next simulation cases we will eliminate the handshaking signals (CSTRBx and CRDYx) to simplify of the timing diagrams. Also we represent the message that transferred between the processors in the system as a decimal counting number. We simulate three case studies: Fault free operation, Operation with one fault, Operation with two fault.

5.2 Case Study-1: Fault Free Operation

This part of the simulation was performed to model the interprocessor communication in the proposed system in fault free operation. As shown in figure 8, DSP1(port-A) and DSP2 (port-A) they send messages to DSP0 (port-A and port-B respectively). Then DSP0 perform processing (addition) on the two received messages and send the result (through port-A) to DSP4(port-A).

5.3 Case Study-2: Operation With One Fault

This part of the simulation was performed to model the first case in table-1.The same example described in case study-1 is repeated here with the assumption of occurrence a fault in DSP0. Therefore all communication ports of DSP0 are tri stated. The fault is indicated by the absence of the <I`m alive> message of the faulty processor. As illustrated in figure 9, 1SP0 is activated and replaces DSP0 in operation. The figure also shows the activation of interrupt signal that send to all processors in the system to stop processing and go back to point of the last <I`m alive> message. Finally the address of the faulty node is appears in the <I`m alive> message.

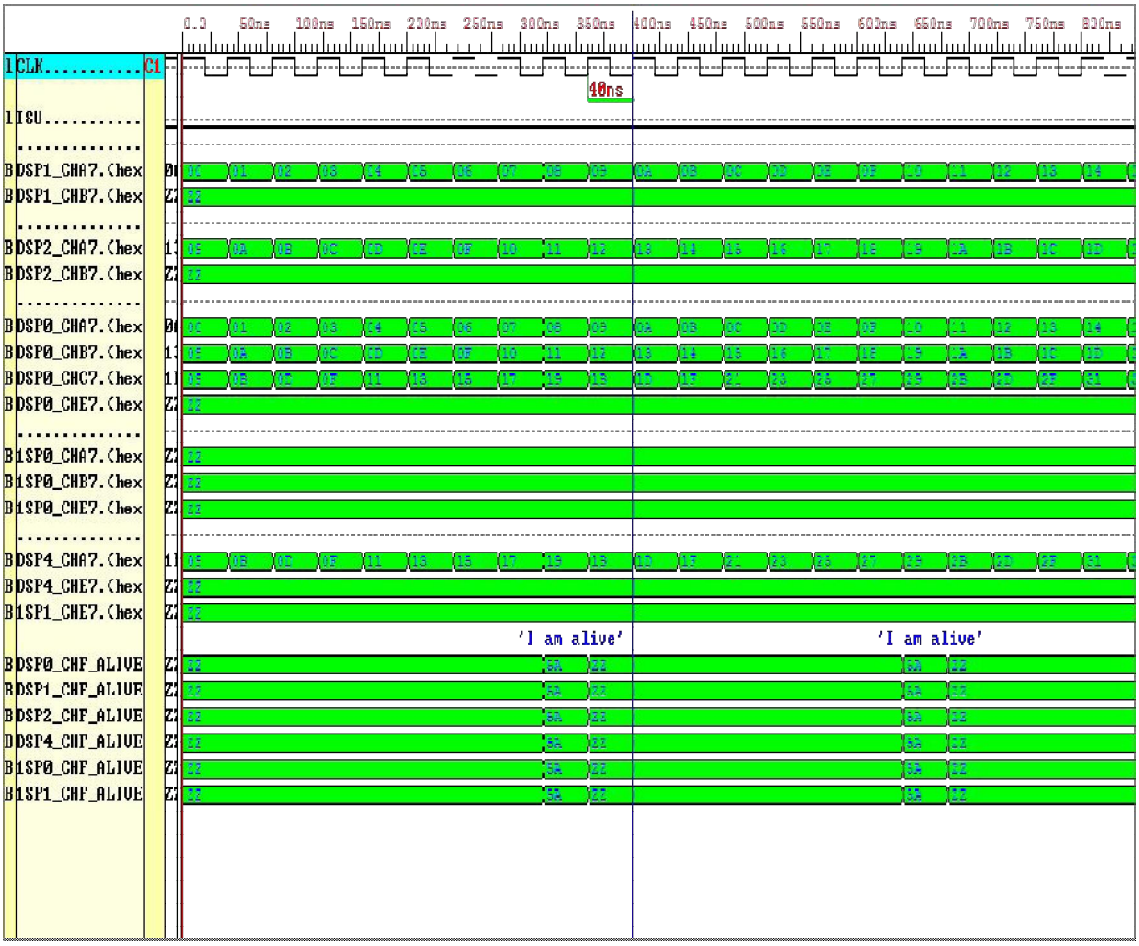

Figure 8 Simulation of data transfer in fault free operation

5.4 Case Study-3: Operation With Two Fault

This case simulate the third case in table-1(fault in DSP0 and DSP4). Again the same example described in case study-1 is repeated here. We consider in this case two situations: in the first situation (figure 10), the two faults occurs within the same period between two (successive) <I`m alive> messages (simultaneous faults). In the second situation (figure 11), the two faults occurs in different periods (sequential faults). In both situations, the spare processors (1SP0 and 1SP1) replace and resume the operation of the faulty processors DSP0 and DSP4 respectively.

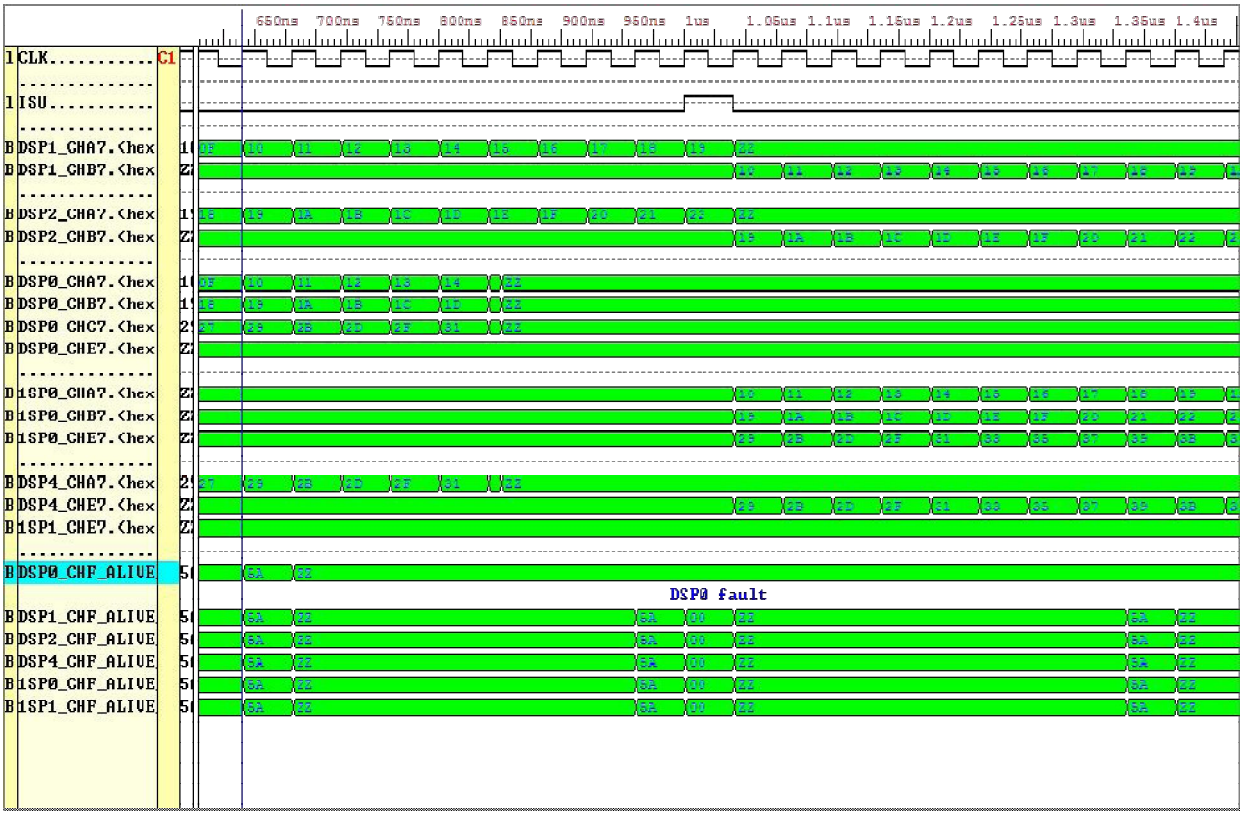

Figure 9 Simulation the data transfer with the fault of DSP0

6. PERFORMANCE DISCUSSION

A faulty hypercube need to be reconfigured to perform the computation task and communication as required by the algorithm with minimal or no performance degradation. In a fault free operation, the cost of communication overhead is constant between any two given nodes in every 3D hypercube. The occurrence of the faults and the mechanism of fault detection, reconfiguration, and to recovering the system from erroneous states causes some degradation in performance. The degradation in performance due to the fault comes from two sources: the first is caused by the overhead result from configuring the system and routing some messages through the crossbar switch in the FPGA device. However, this overhead is small and can be neglected comparing to the second source of performance degradation which is caused by the stopping the processing in all processors in the system and go back to point of receiving last <I`m alive> message.

Figure 10 Simulation the data transfer with the fault of DSP0 and DSP4(simultaneous faults)

According to our simulation, the degradation in system performance depend on the length of the message (in frames), and number of faults. For example, if the message consists of *n* frames and number of faults per message is *m*, then the degradation in system performance for the sequential faults is equal to (m/n). However, the degradation in performance for the simultaneous faults is equal to the degradation caused by one fault. Since the fault is cannot be detected until receiving <I`m alive> message.

Using the FPGA will reduce the design offers, cost, power consumption, and the delay in data routing comparing with other design techniques in implementing interconnection networks and hardware diagnosis circuits (as mentioned in section 1). In addition, using the FPGA devices, provides a very flexible system interconnection networks and allows system developments. Using the TMS320C40, the communication links are built-in with the processor, therefore, failure in the processor means the failure in the communication channels of that processor. This will reduce the complexity of fault tolerant design, since it is not required to design a spare links. The failed processor is replaced with its communication channels.

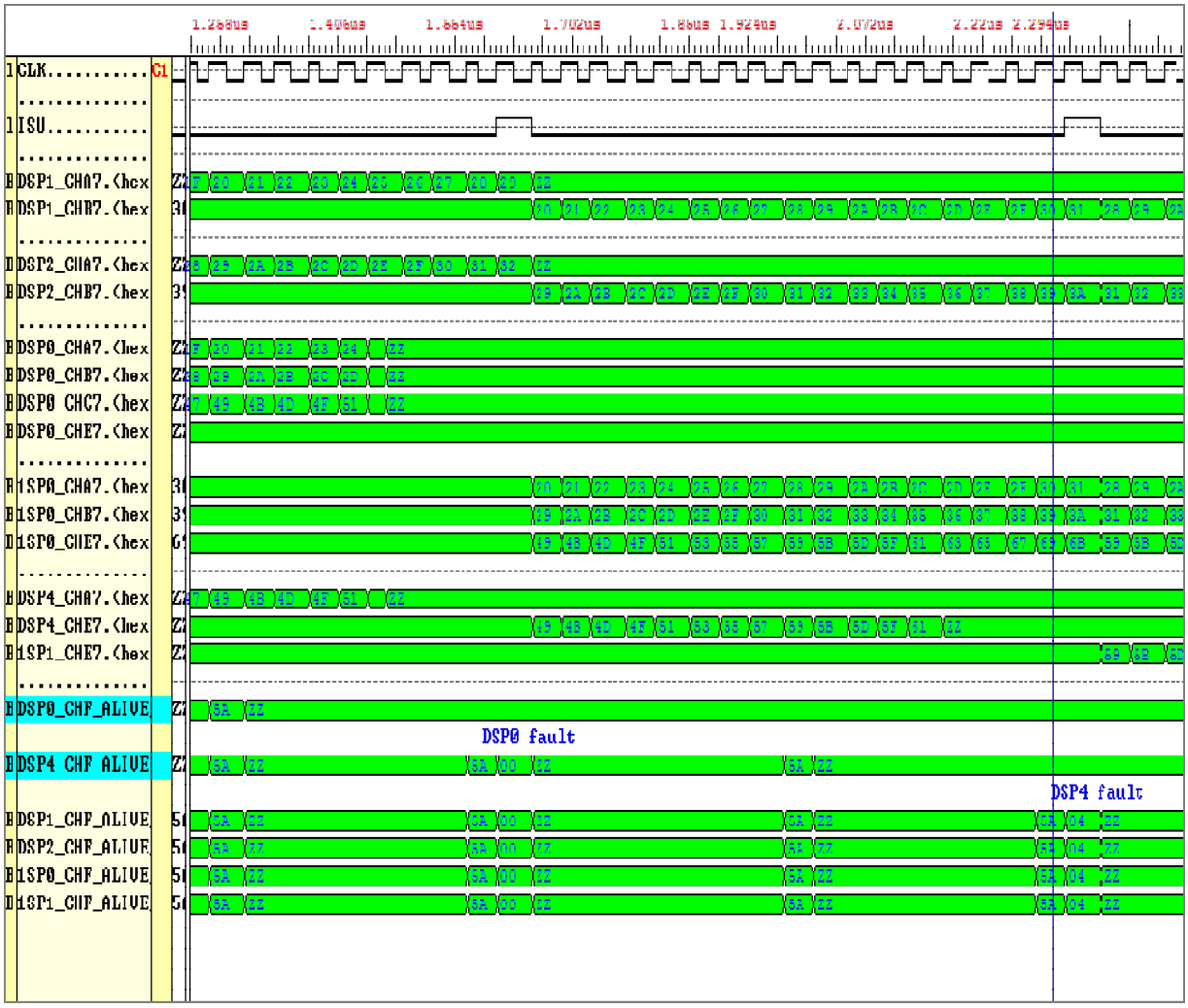

Figure 11 Simulation the data transfer with the fault of DSP0 and DSP4(sequential faults)

7. CONCLUSION

In this paper we present an architecture which uses hardware redundancy in the form of spare nodes that logically replace the faulty nodes to design a fault tolerant (n) dimensional hypercube system. In our scheme, the n-dimensional hypercube is divided into $2^{(n-i)}$ subcubes each of dimension (i), we call each of these subcubes a cluster. One spare node is assigned to each cluster. And the spare node is connected to every regular node of its cluster via I/O communication Ports.

Every node in the proposed system is TMS320C40 digital signal processor which is devoted to overall task computation. The FPGA is used to link two of the regular adjacent nodes between two different 3D hypercube in an extended hypercube. In addition to that, it is used to link the first and second stage spares to some of its adjacent regular nodes. The FPGA units in the proposed system are also connected as an (n-3) dimension cube in the extended hypercube structure.

Upon a node failure, the faulty node is assigned to one of the spares within the cluster. The links of the faulty node are then neglected and the FPGA connects the spare in that cluster is used to connect the spare to some of the neighboring nodes of the faulty node. To tolerate more than

one node failure, the FPGA connected to each 3D hypercube is used to connect two additional spares to that 3D hypercube.

The spare nodes is used in two stages: at stage one, the local spare in every cluster replaces any faulty node in that cluster, at stage two; if the failure occurs in the same cluster, then one of the second stage spares attached to the FPGA replaces the faulty node.

Compared with other proposed schemes, our approach can tolerate significantly more faulty nodes with some overhead and small performance degradation and the resultant configuration does not affect either the hypercube size, communication or computation algorithm already developed for the hypercube multiprocessor.

REFRENCES

[1] A. F. AL-ALLAF, *Parallel Processing System using TMS320C20 Digital Signal Processors,* M.Sc. theses, Engineering college, Mosul University, 1992.

[2] J. Bruck at. al. "Wildcard dimensions, coding theory and fault-tolerant meshes and hypercubes," Proceedings of the 23nd annual international symposium on fault tolerant computing, pp 260-267, July 1993.

[3] P. Banerjee, at. al."Algorithm-based fault tolerance on a hypercube multiprocessor ," IEEE trans. on computers, Vol. 39, pp 1132-1145, Sept 1990.

[4] P. Banerjee, at. al. "Design and evaluation of hardware strategies for reconfiguring hypercubes and meshes under faults," IEEE trans. on computers, Vol. 43, pp 841-848, July 1994.

[5] M. Alam and R. Melhem, " An efficient modular spare allocation scheme and its application to fault tolerant binary hypercubes," IEEE trans on parallel and distributed systems, Vol. 2, pp 117-126, Jan 1991.

[6] M. Alam and R. Melhem, " Fault tolerance and reliable routing in augmented hypercube architectures," Proceedings of the 18th conference on computer communication , pp 19-23, 1989.

[7] Q. Mohammad, " Adaptive Fault Tolerant Routing Algorithm for Tree-Hypercube Multicomputer," Journal of computer science Vol. 2, pp 124-126. 2006.

[8] TMS320C40 Digital Signal Processor use`s guide, Texas Instruments, USA, 1996.

[9] W. Hohl at. al. " Hardware support for error detection in multiprocessor systems", Microprocessor & Microsystems, Vol. 17, No. 4, 1993, pp 201-206.

[10] A. Pataricza, I. Majzik, W. Hohl and J. Honig, "Watchdog processors in parallel systems", Microprocessing and Microprogramming, Vol.39, 1993 , pp 69-74.

[11] I. Majzik, W. Hohi,] A. Pataricza and V. Sieh, "Multiprocessor checking using Watchdog processors" Computer system science & engineering, Vol. 5, 1996, pp301-310.

[12] A. Avizienis, " Toward Systematic Design of Fault-Tolerant System", IEEE computer , April 1997

[13] D. A. Rennels, "Fault-tolerant computing concepts and examples", IEEE Transactions computers, Vol. 33, No 12, pp. 1116-1129, 1984.

[14] ---"*Constant time fault tolerant algorithms for a linear array with a reconfigurable pipelined bus system", J. Parallel Distrib. Comput. 65 (2005) pp. 374 – 381*

[15] N.Tsuda **"***Fault tolerant processor array using additional bypass linking allocated by graph-node coloring*", IEEE Trans. on computers, Vol.49, No.5, May 2000.

INFLUENCE OF COOLING RATE ON THE PROPERTIES AND CHARACTERIZATIONS OF ALUMINIUM-BASED CAST PARTICULATE *IN-SITU* COMPOSITE

Abdulhaqq A. Hamid Al-dabbagh

Mechanical Eng. Dept. - College of Engineering - University of Mosul
E-Mail: abdulhaqqhamid@yahoo.com

Abstract

The current work focused on the influence of cooling rate on the microstructure, mechanical and tribological properties of cast *in-situ* composite. It was observed that the size of intermetallic phase $Mn(Al_{1-x}Fe_x)_6$ and the dendrite arm spacing (DAS) increases considerably with decreasing cooling rate of the cast ingot. Microstructural examination of these different cast *in-situ* composites shows that there is no significant difference in the size of the *in-situ* formed alumina particle. Superior mechanical properties, as indicated by ultimate tensile stress, yield stress, percentage elongation and hardness, are obtained when the *in-situ* composites are processed by cooling the cast ingot in water, resulting in refined microstructure. Higher hardness due to refined microstructure and superior mechanical properties result in decreased wear rate in cast *in-situ* composites cooled in water after casting, compared to the wear rates observed in cast ingots cooled either in air or inside furnace. Cast *in-situ* composite cooled in water after casting, shows higher coefficient of friction compared to those cooled in air or inside furnace.

Keywords Cast *In-situ* Composite, Al-Al_2O_3; Cooling Rate; Microstructure; Mechanical Properties; Dry Sliding; Wear; Friction.

تأثير معدل التبريد على خواص ومواصفات الألمنيوم المتراكب من الداخل بالجسيمات الدقيقة

د. عبدالحق عبدالقادر حامد الدباغ

قسم الهندسة الميكانيكية ـ كلية الهندسة ـ جامعة الموصل ـ موصل/العراق

الخلاصة

في هذا البحث الحالي ركزت الدراسة على تأثير معدل التبريد على التركيب ألمجهري, والخواص الميكانيكية وخواص الاحتكاك والبرئ للمادة المتراكبة. حيث وجد أن حجم الطور $Mn(Al_{1-x}Fe_x)_6$ والمسافة (DAS) يزدادان مع تناقص معدل التبريد للمسبوك. وكما أن الاختبار ألمجهري لهذه المسبوكات المختلفة بين بأنة لايوجد اختلاف في حجم دقائق الالومينا المتولدة من الداخل. وكذلك وجد أن معدل التبريد له تأثير كبير ومباشر على تركيبة وخواص المادة المتراكبة, حيث أن الخواص الميكانيكية والمتمثلة بالجهد الاقصى للفشل, وجهد الخضوع, والمطاطية, والصلادة تكون اكبر في المواد التي كان تبريدها بالماء. وبشكل عام, فالتركيبة المجهرية الصغيرة والنقية والخواص الميكانيكية العالية والتي سببها معدل التبريد العالي أدى بالتالي إلى انخفاض معدل البرئ بشكل كبير مقارنة مع المواد المتراكبة والتي معدل تبريدها اقل. وان معامل الاحتكاك أعلى في المادة المتراكبة التي معدل تبريدها عالي مقارنة مع المواد المتراكبة والتي معدل تبريدها اقل كما هو الحال في التبريد في الهواء أو في داخل الفرن.

Received 1 Oct. 2008 Accepted 12 Mar. 2009