Hardware Software Codesign of a Safety-Critical Embedded Computer System for an Automatic Endoscope

Gul N. Khan and Matthew Jin Electrical and Computer Engineering, Ryerson University, 350 Victoria Street, Toronto, Ontario, Canada M5B 2K3 gnkhan@ee.ryerson.ca

Abstract

Hardware-software codesign is presented for a safety-critical embedded computer system. The system is applied for endoscope control and navigation. The embedded architecture system provides high performance computing for real-time implementation of machine vision algorithms and fault-tolerance for patient safety. It consists of five processor cores, local memory, I/O interface and multi-port shared memory. The hardware and software system architectures are codesigned. A virtual hardware is developed to execute the application and system software tasks. The system is designed and modeled sing VHDL and Eaglei toolset. We have limited system verification to co-verification of system hardware architecture and fault-tolerance strategies. Co-verification results indicate that the system performance degrades gracefully under various fault scenarios.

Keywords: Hardware-software codesign, High performance embedded systems; Automatic endoscope.

1. INTRODUCTION

The applications of embedded computer systems range from sophisticated aircraft flight control to consumer electronics. The high volume embedded applications include home appliances, automobiles, parking systems, cellular phones, toys, and many other devices that contain some sort of μ -controller. On the other extreme are safety-critical applications like fly-bywire aircrafts, high-rise building elevators and medical instruments. The essential features of safety-critical embedded systems like endoscope are high performance, fault-tolerance and adaptability of the system. The architects of such system are facing high throughput and reliability demands that have never before been required of these systems. The system architecture has to provide fault-tolerant hardware support that can be programmed to implement software fault-tolerant strategies [1, 2].

The design of embedded system has adapted the brute force approach in the past. Hardware and software were designed separately while correctness and compatibility of two domains are sorted at the integration stage. Hardware-software integration problems lead to either large number of design iterations or non-optimal system architecture. Hardware-software codesign approach speeds up an intuitively serial design process by developing hardware and software concurrently. It is best suited to embedded system design and helps system designers to meet the design and development deadlines [3, 4]. Hardware-software codesign shortens the development cycle, minimize bugs, manage cost, and produce competitive embedded systems.

The high performance embedded system presented in this paper is employed for endoscope control. Endoscope is a medical instrument used for diagnosing UGI, colon and bronchus diseases as shown in Figure 1. We have been concentrating on the automation of colonoscopy. It requires high performance computation for real-time implementation of machine vision algorithms and faulttolerance support for patient safety. High performance computing and fault-tolerance can be accomplished by a hardware engine of heterogeneous multiple processor. During conventional colonoscopy, the consultant advances the endoscope progressively while controlling its tip by judging the direction from a stream of colon images. The deepest area in colon (lumen) corresponds to the dark region near the center of colon images as shown in Figure 2. Lumen is an important navigational landmark for colonoscope control and navigation. We have investigated a number of methods to identify the lumen in the past [5]. Magnetic 3D imaging is also being introduced to support doctors during colonoscopy procedures [6]. Attempts are also made to navigate the endoscope using collision avoidance technique [7].



Fig. 1. Conventional endoscope

We have employed standard industrial and parallel computer systems for endoscope navigation in the past

[5]. However, it is observed that a dedicated embedded system is needed for such safety-critical applications. The main objectives of our research are to develop a fault-tolerant and high performance embedded system architecture. The software architecture to be realized by the embedded system includes task scheduling, interprocessor communication, image feature extraction and building a search space representation of colon.

2. THE EMBEDDED SYSTEM

A block diagram of the endoscope embedded computer system is illustrated in Figure 3. The system consists of an embedded computer and I/O modules. Digital outputs control the endoscope tip while high-speed analog input channels acquire video/ultrasound images of colon.



Fig. 2. Typical colon images



Fig. 3. Endoscope control embedded system

2.1 Embedded Computer Architecture

The embedded computer system consists of five nodes, which are fully connected by ten dual port memories DP_{ij} as shown in Figure 4. High performance interconnection network provides efficient interprocessor communication. There are three compute processor cores (WP_i) for computational intensive tasks and two IO processor cores (IP_x) for real-time control. The **IP** processors support fault tolerance and recovery from failures in addition to task scheduling and load balancing. Both **IP** processors are capable of serving as system controller, however at a given time, one of them is designated as controller for system monitoring. The other **IP** monitors the designated controller and take over the role of controller when the designated controller fails.

The system nodes are self-checking and in the case of a failure they isolate themselves. A watchdog timer

detects a node failure while failure of **DP** memories and their interface is detected during message transfer. The processor interconnection network provides alternate routes for inter-processor communication. A number of fault-tolerant strategies are used for fault detection, containment and system recovery. The failure of node components is handled as a single fault. The interrupting capabilities of a faulty processor is disabled and its access to DP memories is also inhibited. The system controller broadcasts the failure to healthy nodes and invokes a diagnostic process. The failed node is put into service for a transient fault. Otherwise, it is kept out of the system and its tasks are re-scheduled. A node with a faulty program memory is utilized in a degraded mode by executing its critical tasks from **DP** memory blocks.



Fig. 4. Embedded processor architecture

The processor architecture supports various software and hardware fault-tolerance configurations. Three compute processors can be connected in TMR (triple modular redundant) mode while one of the **IP** processor serves as a voter. The other **IP** processor provides the input data to the TMR module as shown in Figure 5a. The architecture is flexible and one can configure both **IP** processors as a duplex voter for the TMR configuration as shown in Figure 5b.

2.2 Endoscope Control

The embedded system processor architecture can be envisaged having two partitions: computing and IO as illustrated in Figure 4. IO partition interfaces with endoscope sensors and issues tip movement commands. IO partition digitizes colon images and passed on to computing partition. Computing partition executes machine vision algorithms, detects the lumen and builds a search space representation. IO partition generates digital pulses for two motors for controlling endoscope tip in two concentric perpendicular directions. The tip direction movements are fed back to IO partition using analog input channels. Left-right and up-down tip movements are first translated into voltages that are finally converted by analog to digital conversion module. The feedback of tip movements forms a servo control mechanism where the servo loop is completed by a software process being executed by IO partition.



(a) Triple Modular redundancy







2.3 Hardware Software Codesign

Hardware-software co-design problem can be broken up into design specification, hardware/software partitioning and estimation of system performance. The design specification for describing system level behavior is a challenging problem. It needs high abstraction yet requires fine details to reduce ambiguities during synthesis. Embedded system is partitioned by using design constrains in terms of cost, power consumption, silicon size, speed, etc. After partitioning, the design comes down to hardware-software co-simulation. A cosimulation tool, Seamless uses instruction set simulator and a target processor model to create a virtual hardware environment. A more abstract approach is used by Eaglei to simulate the functional behavior of the system.

We have employed Eaglei for hardware-software coverification of the embedded system. The software execution environment interacts with a VHDL based hardware model having Virtual Software Processor (VSP). VSPs are functional description of μ -processor cores divided into two halves that communicate with the hardware and software design entities. It allows the fetch/execute cycles to be handled by the software application running at the workstation speed, thereby freeing the HDL simulator to handle rest of the hardware on demand from the software application. Two types of processor cores: Oak DSP for computing and ARM7TDMi for IO partition. Other hardware modules are implemented in VHDL. System and application software tasks are coded in C language.

The main components of the IO and compute nodes include a virtual software processor, watchdog timer, private and dual-port memory interfaces. IP processors also contain processor-timer one each for the rest of system nodes that helps IP nodes to monitor other system nodes. A system status is kept at each node in a 4x5 2-D register. The contents of status register, kept at IP5 are shown in Table 1. The 'Available' column field when set to '1' represents the node availability. The "Busy" field when set indicates a busy node that is unable to accept new tasks. The 'Reset' field when set implies that the node is currently being diagnosed. The 'TaskID' field provides the tasks details a node is executing. The node status register given in table indicates that IP4, WP1 and WP3 are available, DP15 is faulty; WP1 and WP3 are processing tasks 3 and 2 respectively.

Table 1. Status Register of IP5

System Node	Available (1/0)	Busy (1/0)	Reset (1/0)	Availability Com-module (1/0)	Task ID
IP4	1	1	0	0	-
WP1	1	1	0	1	3
WP2	1	0	1	0	-
WP3	1	1	0	0	2

3. TESTING AND VERIFICATION

A subset of endoscope control application is selected to verify fault-tolerance features of system architecture. It consists of a suit of matrix manipulation algorithms that are the basic building block of image analysis. The application is partitioned into tasks to be executed by compute processors. IP node provides support for faulttolerance, task scheduling and endoscope control. A dedicated process that executes concurrently along with application tasks facilitates fault-injection. We present the verification results of a few representative failure scenarios. One of the healthy IP nodes (IP4) becomes the system controller while the other takes the role of backup controller (IP5) at startup. Figure 6 shows the verification of system startup where each node checks itself and test the integrity of its DP memories.

A permanent fault is simulated by injecting a fault in WP1 node, in the next fault-scenario being presented in Figure 7. The controller (IP4) detects the node failure, activates the corresponding processor-timer while resetting the faulty node. After a time-out, if WP1 cannot recover, it is declared faulty and controller removes it from the system. The application tasks being executed by the faulty nodes are re-assigned to other healthy nodes. However, if the faulty node recovers within a time-out, system controller puts the recovered node into the system. Inter-processor communication is the key to achieving high performance from a multiple processor system. The embedded processor system is tested and verified to handle the failure of DP memories and their interface. The results are presented for DP14 or its interface failure with WP1 and/or WP4 nodes. A fault is injected in DP14 memory module. WP1 detects the fault successfully during a message transfer as given in Figure 8. It reports the failure to IP4 by using DP24 and DP12 modules.

A temporary fault is also injected in the system controller (IP4) to verify the fault-tolerance for IP nodes. Figure 9 presents the fault detection and system recovery from the system controller failure. The backup controller (IP5) monitors IP4 and detects its failure. Then it establishes itself as a new system controller and starts diagnosing IP4.

DDD: WP1 main 2> < /dev/pts - DDD: WP2 main		2> < /dev/pts/11 · 🔄	- DDD: WP3main 2> < /dev/pts/1 · _			
Choice : 1 Establish connection with DP14. Establish connection with DP15. Establish connection with DP12. Establish connection with DP13. Plose choose one of the following option 1. Initialize MP1	Operating System of WP2 Please choose one of t 1. Initialise WP2 2. Permanent Fault 3. Temporary Fault 4. Exit Choice : 1	the following option	Operating System of MP3 Please choose one of the following option 1. Initialise MP3 2. Persament Fault 3. Temporary Fault 4. Choice : 1			
2. Persament Fault 3. Temporary Fault 4. Exit IPS ready to receive jab IP4 ready to receive jab		th DP24. th DP25. th DP12. th DP23.	Establish connection with DP35. Establish connection with DP35. Establish connection with DP35. Establish connection with DP33. Please choose one of the following option 1. Initialise MP3 2. Personent Fault 3. Temporary Fault 4. Exit Choice :			
		the following option				
2. Temporary Fault 4. Exit Choice :	IPS ready to receive job IP4 ready to receive job		TPS ready to receive job TP4 ready to receive job[
DDD: IP4main 2> < /dev/pts/7 > 4. Teaporary Fault 5. Exit Choice : 1 Establish connection with DP14. Establish connection with DP34. Establish connection with DP34. Establish connection with DP35.	/dev/pts/7	DDD: IPSmain 2> < /dev/pts/15 > /dev/pts/15 < Establish connection with DP35. Establish connection with DP45. Please choose one of the following option 1. Initialise IP5 2. Read Do File 3. Permanent Fault 4. Topporary Fault				
Please choose one of the following option 1. Initialise 194 2. Read John File 3. Permanent Fault 4. Temperary Fault 5. Temperary Fault Choice : 1P5 ready to receive job		5. Exit Choice : IP4 ready to receive job Please choose one of the following option 1. Initiation IPS 2. Read Job File 3. Persament Fault 4. Temporary Fault 5. Exit				
MP1 is ready to recieve job[]		Choice : []	والبهي الماليجي ويجاجبها والمتعادين والمستحد المالي			

Fig. 6. Verification of system startup



Fig. 7. System recovery from permanent fault in WP1 node

4. CONCLUDING REMARKS

The embedded computer architecture is modeled using Eagle*i* tool set that provides virtual software processors to build a multiprocessor virtual hardware system. Embedded system software is executed for hardware software co-verification. The system architecture is improved to fulfill the application requirements. Hardware-software co-verification results indicate that embedded system degrades gracefully for different fault scenarios. The system would require only two fault-free communicating nodes (one in each partition) for the system to be considered operational. In the extreme case, even one healthy IP node keeps the embedded system operational at lowest level of performance. The main goal of this research is to design and develop an SOC level high performance embedded system. The system will be co-verified using Seamless from the hard real-time point of view.

- DDD: WP1 main 2> < /dev/pts/9	- DDD: WP2main 2:	> < /dev/pts/1 · _	DDD: WP3main 2> < /dev/pts/1		
Plesse choose one of the following option 1. Initialise WP1 2. Personent Fault 3. Tesporary Fault 4. Exit Choico : WP2 ready to receive job Plesse choose one of the following option 1. Initialise WP1 3. Temporary Fault 4. Exit Choice : WP3 ready to receive job Please choose one of the following option 1. Initialise WP1 2. Persament Fault 3. Temporary Fault 3. Temporary Fault 5. Tempora	 4. Exit Choice: IP5 ready to receive job IP4 ready to receive job WP1 ready to receive job Please choose one of th 1. Initialise WP2 Choice: WP3 ready to receive job Please choose one of th 1. Initialise WP2 2. Permanet Fault 3. Themporary Fault 3. Themporary Fault Choice: I 	<pre>> > </pre>			
DDD: IP4rnain 2> < /dev/pts/7> WP3 is ready to receive job Plasse choose one of the following option 1. Initiation IP4 2. Read Job file 3. Permanent Fault 4. Temporary Fault 5. Exit Choice : 4 Plasse choose one of the following option 1. Initiaties IP4 2. Read Job file 3. Permanent Fault 3. Perman	/dev/pts/7	DDD: IPSmain 2> < /dev/pts/15 > /dev/pts/15 < Please choose one of the following option Initialise JPS Rond Joh File Permanent Fault Temporary Fault Sexit Choice : WP3 ready to receive job Please choose one of the following option Initialise JPS Rond Joh File Permanent Fault Temporary Fault			
4. Temporary Fault S. Exit Choice : IP4 is being reset by IP5 IP4 is in reset mode()		5. Exit Choice : IPS -> Reset IP4 Bo-initialise IP4 Establishing as System Controllen			

Fig. 8. Communication module DP14 failure and system recovery

— DDD: WPI main 2> < /dev/pts/9:	— DDD: WP2main 2> < /de	ev/pts/11	— DDD: WP3main 2> < /dev/pts/1	— DDD: WPI main 2> < /dev/pts/9	DDD: WP2main 2:	<th>— DDD: WP3main 2> < /dev/pts/1 ·</th>	— DDD: WP3main 2> < /dev/pts/1 ·	
MP1 -> ComModule 14 Not Functioning	Choice : Transfer Hatrix Data to MP1		Result is 10	Please choose one of the following option	4, Exit Choice :		Establish connection with DP13,	
Send DP14 Error Message to MP2			Send Result Back to IP4	2. Permanent Fault			Electronic connection with west	
Please choose one of the following option Please choose one of the		aving option Please choose one of the following option		4. Exit	1-3 ready to receive jus		1. Initialize WP3	
2. Permanent fault	2. Permanent Fault		2. Permanent fault	Charles 1	1P4 ready to receive job		2. Permanent Fault 3. Temporary Fault	
3. Temporary Fault 4. Exit	 Temporary Fault Exit 		3. Temporary Fault 4. Exit	W2 ready to receive job	my1 ready to receive job	22 C D	4. Ent Outce :	
Choice :	Chaice :		Choice :	Please choose one of the following option 1. Initialise WP1	Please choose one of th 1. Initialise WP2	fellowing option	1P5 ready to receive 3sb	
Receive Task 4 from MP2	ve Taik 4 free W2 Biochive matrix result free W9 to 54 Heault Eack to W2 Send Result Eack to 104 through W2 es choose one of the following option escenant Rain escenant Rain esce		Receive Task 1 from IP4	2. Persament fault 3. Temporary fault 4. Exit Choice 1	2. Permanent Fault 1. Tegoriny Fault dutics : UPD ready to receive job Please choose one of the fallowing option 1. Initialise W2 2. Permanent Fault 3. Tegoring Fault 3. Tegoring Fault 3. Tegoring Fault 3. Tegoring Fault		IP4 ready to receive job	
Result is 04			Result is 30				WP1 ready to receive tab	
Send Result Back to MP2			Send Recult Back to IP4	MP3 ready to receive job			MP2 ready to receive job	
Please choose one of the following option 1. Initialise WPI 2. Persament Fault 3. Temporary Fault 4. Exit Choice : []			Please choose one of the following option 1. Initialise M02 2. Persament Fault 3. Teapony Fault 4. Ent Choice : []	Please choose use of the following option 1. Initialize MP1 2. Persanet Fault 3. Temporary Fault Choice : D			Please choose one of the following option 1. Initialize #01 2. Personent Fault 3. Temporary Fault Contex - D	
- DDD: IP4main 2> < /dev/pts/7 > /dev/	/pts/7	DDD: IPS main 2	<pre>2> < /dev/pts/15 > /dev/pts/15</pre>	DDD: IP4main 2> < /dev/pts/7>	/dev/pts/7	DDD: IP5 mat	n 2> < /dev/nts/15 > /dev/nts/15	
Sand Task 3 to 403	5.	Exit:		Not allow to send job as status is not System C	ontroller	Please choose one of	the following option	
Baralas DB14 Front menunan		It Received is 24		Not allow to send job as status is not System C	Not allow to send job as status is not System Controller		2. Read Job File	
Task hold by MC1 is 1		Please choose one of the following option 1. Initialize IPS 2. Read Top File		Not allow to send job as status is not System C	Not allow to send job as status is not System Controller		4. Tesporary Fault 5. Exit Chaice 1	
Undate the Backup Controller the Benuilt Becalved				Not allow to send job as status is not System C	Not allow to send job as status is not System Controller			
TagkID - 3, Result - 10		Permanent Fault		Not allow to send job as status is not System C	Not allow to send job as status is not System Controller.		Running Send Status Process	
Update the Backup Controller the Result Received: S. Exit		Exit		Not allow to send job as status is not System C	Not allow to send job as status is not System Controller		Please choose one of the following option	
Contrar - 1, monte - 14 Good Task & to 401 through 401		Mot allow to send job as status is not System		etroller 2. Read Jub File				
fand Task 1 to 101		the second and all	the following action	Please choose one of the following option	Flease choose one of the following option 1. Initialize IP4 2. Read Job file		4. Tesporary Fault 5. Doit Choice 1 2	
Send talk i to WJ		Initialize IP5	the remaining option	2. Read Job file				
TaskID = 1, Result = 30		2. Permanent Fault		4. Tesporary Fault		Enter the name of the first matrix file: matrixim.txt		
Update the Backup Controller the Result Received: 5. Eait TaskID - 4, Result - 84 [] Choice : []		A, Dit Obsice : []			inter the mase of the 2nd matrix file: matrixib.txt			

Fig. 9. Temporary fault at system controller IP4

Acknowledgements

This research is supported by a grant from NSERC Canada. The authors would also like to thank CMC for providing hardware software co-design CAD tools.

References

- [1] Gul N. Khan, "Fault-tolerance evaluation of a high performance embedded computer system," *SAS Research Link*, vol. 4/99, Singapore, pp. 4-5, January 1999.
- [2] J. C. Laprie, J. Arlat, C. Beounes and K. Kanoun, "Definition and analysis of hardware and software faulttolerant architectures," *IEEE Computer*, vol. 23, no. 7, pp. 39-61, 1990.
- [3] M. Chiodo, P. Giusto, A. Jurecska, H. C. Hsieb, A. S. Vincentelli and L. Lavagno, "Hardware-software codesign of embedded systems," *IEEE Micro*, vol. 14, no. 4, pp. 26-36, August 1994.

- [4] F. Slomka, M. Dorfel, R. Munzenberger and R. Hofmann, "Hardware/software codesign and rapid prototyping of embedded systems," *IEEE-Design and Test of Computers*, vol.17, no.2, pp.28-38, April-June 2000.
- [5] Gul N. Khan and Duncan F. Gillies, "Vision based navigation system for an endoscope," *Image and Vision Computing*, vol. 14, no. 10, pp. 763-772, 1996.
- [6] G. D. Bell, R. S. Rowland, M. Rutter, M. Abu-Sada, S. Dogramadzi and C. Allen, "Colonoscopy aided by magnetic 3D imaging: Assessing the routine use of a stiffening sigmoid overtube to speed up the procedure," *Medical and Biological Engineering and Computing*, vol. 37, no. 5, pp.605-611, Sept. 1999.
- [7] S. D'Attanasio, O. Tonet, G. Megali, M. C. Carrozza and P. Dario, "A semi-automatic handheld mechatronic endoscope with collision-avoidance capabilities," in Proc. *IEEE International Conference on Robotics an Automation*, pp. 1586-1591, 2000.