Vol 06, Article 07609; July 2015
http://ijves.com

International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

# COMMUNICATION PROTOCOL FOR NETWORKED EMBEDDED SYSTEM

[1]PRADIP RAM SELOKAR, [2]P. T. KARULE
[1]Assistant Professor, Department of Electronics & Communication Engineering, Shri Ramdeobaba College of Engineering & Management, Nagpur – 13
[2]Professor, Department of Electronics Engineering, Yashwantrao Chavan College of Engineering, Nagpur.
pradip.selokar@gmail.com, ptkarule@rediffmail.com

**ABSTRACT**

*In the developed system ARM9 is a master and Two ARM7s are slaves. The peripherals are being controlled by two ARM7 boards. The Peripherals are connected to the ARM7 through Complex Programmable Logic Device (CPLD). The CPLD is in turn connected to the ARM7 using Serial Peripheral Interface (SPI). The ARM7 boards collect the information from the peripherals and send it to the ARM9 board. The communication between ARM7 and ARM9 is via UART (Universal Asynchronous Receiver Transmitter) over CAN (Controller Area Network).*
*The ARM9 board has got the software intelligence. The ARM9 behaves as a master and two ARM7 boards behave as slaves. Being master ARM9 passes tokens to ARM7 which in turn returns (Acknowledges) the token.*
*The ARM9 is further connected to Proxy via Ethernet. The proxy is further connected to the service platform (server) via Ethernet. So subsequently any decisions at any stage can be changed at server level. Further these commands can be passed on to ARM9 which in turn controls the peripherals through ARM7.*

- *The system which we have developed consists of ARM9 as a master, Two ARM7 as Slaves. The communication between ARM9-ARM7 is via UART over a CAN.*
- *Each ARM7 further communicates serially (RS232) with the two 8051 Microcontroller nodes.*
- *Thus a networked Embedded System is developed wherein the serial data is brought over Ethernet.*

## 1 Introduction

The network topologies are categorized in basic types as Bus, Ring, Star, Tree and Mesh. Hybrid of two or more of these basic topologies can be combined together to achieve an efficient communication link. An attempt to design the token based communication protocol between one ARM9 (AT91SAM9260) board and two ARM7 (AT91SAM7S256) boards was made successfully. In this communication link the ARM9 acts as Master and two ARM7 act as Slaves. The details of the developed communication protocol are given herein.
The main constrain for setting up this communication link was TIME. i.e. peripheral should get the quicker response from the server. Often the performance in embedded devices is limited by the firmware which runs over it. This work carried out presents fundamentals of design of firmware for two ARM devices, visually ARM7 and ARM9 including fully fledged communication protocol implementation at device level and the results thereof.
Traditionally systems on devices have been designed using ATMEL 89C51 Microcontroller, PIC Microcontroller etc. These microcontrollers support the communication using RS-232 or RS-485. Here the ARM Microcontrollers are used. The ARM microcontrollers are having fast execution speed (MIPS – million instructions per second), thereby meeting the basic constrain (TIME) of this control system. The results indicate that the designed system meet the basic time constrain criterion.

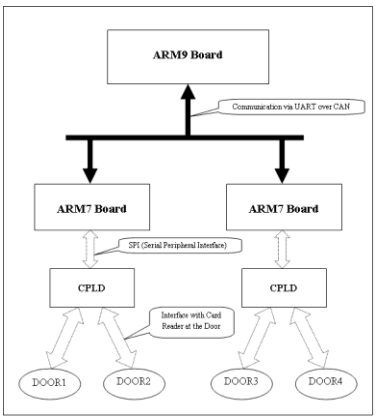


*Figure 1: Block Diagram*

In the simplest form it can be explained as, half duplex communication has been established between server and peripherals. In one direction the data collected by peripherals is transmitted to server. In other direction the command from server is transmitted to the peripherals. Figure 1 gives the block diagrammatic view of the System.

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

## 2 COMMUNICATION PROTOCOL

In the communication link between ARM9–ARM7, the ARM9 acts as a master and the two ARM7s are slaves. Being master ARM9 initiates the communication i.e. it passes the first token to one of the ARM7, receives back acknowledgement, then passes the next token to other ARM7, receives back acknowledgement and so on. It is clear from the figure 5.1 that the two ARM7s and ARM9 are connected in a star topology fashion where ARM9 is at the centre of the star. The physical communication link between them is of multidrop kind. The following algorithm explains the basic token based communication protocol and figure 2 gives its flowchart implementation.

### 2.1 Basic Token based Communication Protocol Algorithm

1. ARM9 begins the communication by passing first token to any one ARM7 say ARM7_1.
2. ARM7_1 acknowledges back this token. Along with the acknowledgement token it also sends the data collected from peripheral (if any).
3. After receiving back the acknowledgement from ARM7_1 for the last token sent, ARM9 stores the data received (if any) in the RECEIVE QUEUE and now sends the next token to other ARM7 say ARM7_2.
4. ARM7_2 acknowledges back this token. Along with the acknowledgement token it also sends the data collected from peripheral (if any).
5. At the ARM9 side the ARM7-ARM9 communication program runs on a separate THREAD and access authorization program runs on a separate THREAD within a single PROCESS. The synchronization of different threads is done using MUTEX.
6. The access authorization program continuously keeps reading the RECEIVE QUEUE, if there is a data in RECEIVE QUEUE it processes it and writes the reply command in the SEND QUEUE of corresponding ARM7 (say SEND_QUEUE_1 for ARM7_1 and SEND_QUEUE_2 for ARM7_2).
7. Next time when it is the turn of particular ARM7 to be sent token by ARM9, the ARM7-ARM9 communication program reads from the corresponding SEND QUEUE and along with the token it sends command to that ARM7.
8. Based on the received command the ARM7 acts accordingly on the peripheral.
9. The alternate exchange of tokens between ARM7s and ARM9 goes on indefinitely (whenever there is no data to be sent from either side, a dummy token with prescribed Header and Footer is sent) to ensure a persistent communication link.

## 3 COMMUNICATION DETAILS 3.1 UART Initialization

- The ttyS4 port (UART3) of ARM9 is opened using the file operation. The file descriptor is used to access this port.
- While initializing UART following settings are done through the source code,
  – Baud Rate: 115200
  – Data bits: 8 bits
  – Parity: None
  – Stop Bit: 1 bit
  – Hardware flow Control: No
- The UART0 of ARM7 is accessed using its base address (AT91_BASE_US0) and similar settings are done through the source code.
- A communication link between these two ports (ttyS4 of ARM9 and UART0 of ARM7 is set up. The physical medium is CAN bus.

This is how ARM7–ARM9 physical communication link was setup. Initially there wasn't any prescribed communication packet format.

The very first attempt to send data over this communication link was done by transmitting just the single character (1 byte) from ARM7 to ARM9 using 'putchar' function in C. This byte was received on the ARM9 side using 'getchar' function in C. This was successful. Later on communication with single byte but both transmitting and receiving was tried successfully over this communication link. At this stage a functional communication link was established. Now the next task was to decide the format of the communication packet.

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

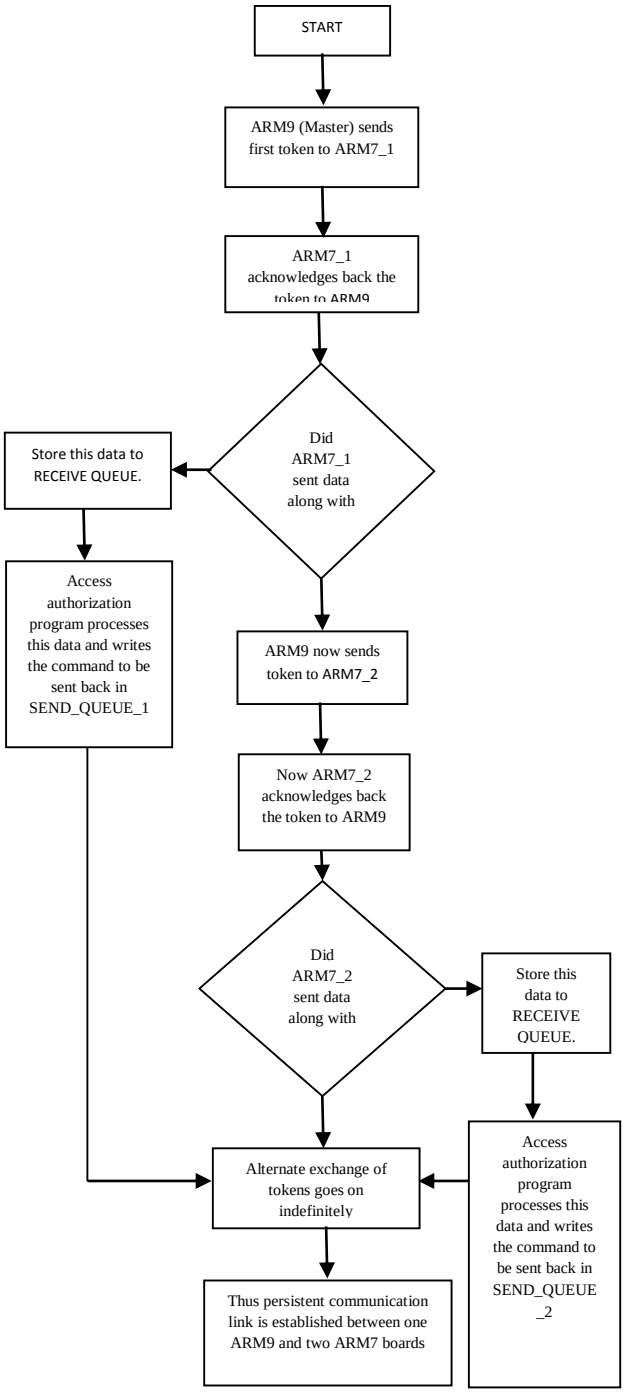


*Figure 2: Flowchart for Token Based Communication Protocol*

### 3.2 Packet Formation

Table 1 lists the prescribed packet format for the ARM7–ARM9 communication.

| 2 Bytes | 1 Byte | 2 Bytes | 1 Byte | 1 Byte | | 1 Byte | 1 Byte | 1 Byte | 1 Byte | 2 Bytes |
|---|---|---|---|---|---|---|---|---|---|---|
| Start of packet | Address of board | Sequence Number | Number of Bytes | Packet Type | DATA | Sensor Status | Relay Status | Checksum | ACK/NACK | End of Packet |
| 0,1 | 2 | 3,4 | 5 | 6 | Max 256 Bytes | N-6 | N-5 | N-4 | N-3 | N-2, N-1 |

*Table 1: ARM7–ARM9 Communication – Prescribed Packet Format*

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

The communication packet is received and transmitted in following sequence of bytes.

1. Start of Packet 1: 0x0B
2. Start of Packet 2: 0x0E
3. Address of Board
– ARM9: 0x00
– ARM7_1: 0x01
– ARM7_2: 0x02
4. Sequence Number 1
5. Sequence Number 2
6. Number of Bytes in Data
7. Packet Type
8. Data
9. Sensor Status
10. Relay Status
11. Checksum
12. ACK/NACK
13. End of Packet 1: 0xB0
14. End of Packet 2: 0xE0

**3.3 Communication Data**

Figure 3 gives the details of the communication data that has to be exchanged between ARM7s and ARM9.

- **Command:** Command is given at the server in reply to the data received from peripheral, which is received by ARM9 via proxy. The ARM9 sends this command to ARM7, which in turn handle the peripheral accordingly.

Command can also be sent from server to ARM7 without actually receiving the data from the peripheral.

- **Firmware:** For development purpose the firmware to the ARM7 can be downloaded using J-link. But in a final product ARM7 firmware is expected to be downloaded from the server via ARM9 using ARM7–ARM9 communication link.

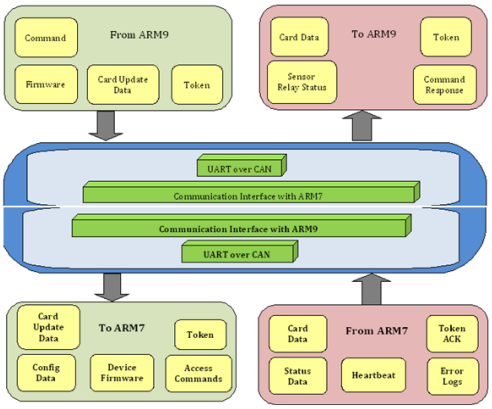


*Figure 3: ARM7–ARM9 Communication Details*

- **Token:** Token is the dummy communication packet which contains zero bytes of data. Token is continuously exchanged between ARM9 and ARM7. This is essential for sustaining the communication. Each microcontroller acknowledges the other's token.
- **Sensor Relay Status:** ARM7 reads the sensor and relay status from the peripheral and sends it to ARM9. Using the current relay status ARM9 comes to know about the peripheral.
- **Command Response:** ARM7 replies to the command received from ARM9. It sends back the operation performed on the peripheral according to command.
- **Configuration Data:** Configuration data is sensor, relay configuration as received from ARM9 (such as relay activation delay, relay on time, Sensor Ideal condition etc.).
- **Heart Bit:** The ARM7 board continuously keeps sending the signal to ARM9 to indicate whether it is alive in the communication.

**3.4 Checksum**

For both, the data received in the packet and the data to be transmitted in the packet the 8 bits checksum is calculated on both the sides ARM7 and ARM9. The logic used for calculating the checksum is XORing the data in the packet byte by byte.

Based on the checksum criterion the integrity of the data received in the packet is checked i.e. the data is said to be error free if the received checksum (Checksum field in the packet) and the calculated checksum are matching. The data is added to the receive queue only if its integrity is validated.

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

### 3.5 ACK/NACK

If the integrity of the received data is validated on the basis of checksum, the ACKNOWLEDGEMENT (ACK) is sent back by piggybacking in the next packet, otherwise NO_ ACKNOWLEDGEMENT (NACK) is sent. Whenever NACK is received the packet is resend.

The packet resending is performed for maximum of three times. If the problem still persists an event is send to the proxy letting it know about the problem in ARM7–ARM9 communication link. In turn the server will come to know about the communication problem.

For resending of the data, whenever packet is sent its copy is written in the BACKUP SEND QUEUE. If NACK is received for the last sent packet, the same packet is read from the BACKUP SEND QUEUE and resend. However if the ACK is received for the last sent packet, the BACKUP SEND QUEUE is read and it is freed.

### 3.6 Communication Fail over Mechanism

The communication fail over mechanism refers to when one of the ARM7 board is not working the communication should go on with the other one. Also when the failed ARM7 board gets alive it should be possible to reconnect it in the ongoing communication, which is something like Hot Plugging.

When it is possible for the ARM9 to read from both the ARM7 boards (one by one) before the receiver timeout (50 ms) occurs, the condition is error free i.e. both the ARM7 boards are working properly.

If the receiver timeout occurs before the byte could be read from particular ARM7 board then there is an error in reading from that particular board. This board is then declared FAILED. When the ARM9 senses the failed status of one of the ARM7 boards it switches for the other one which is ACTIVE.

After every 5 seconds the ARM9 keeps sensing for the FAILED ARM7 board, if now it is available (Reply is received for the dummy packet sent to it) it's status is changed to ACTIVE. If still it is not available the communication goes on with the previous Active ARM7 board. Figure 4 explains the communication fail over logic in the form of flow chart.

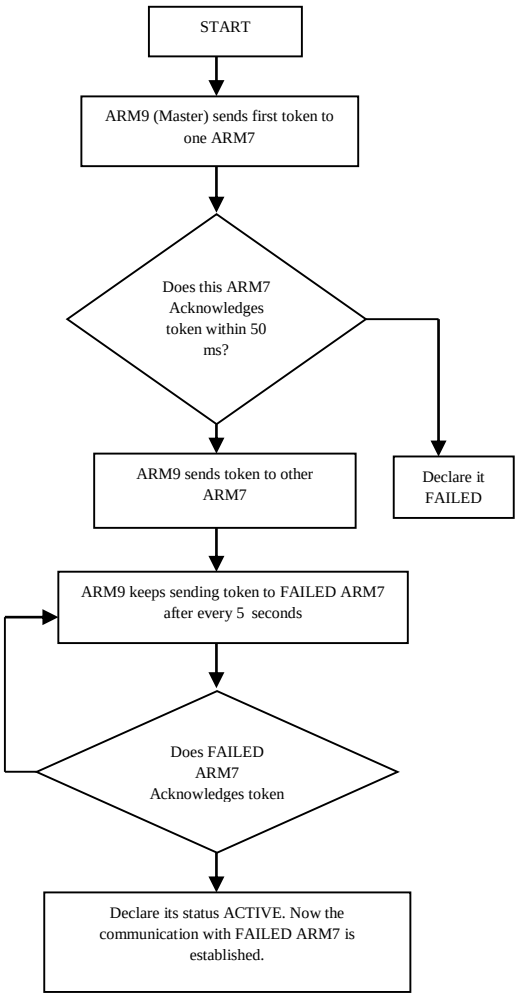


*Figure 4: Flowchart: ARM7–ARM9 Communication Fail over logic*

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

### 3.7 Communication Fail LED

This is the red color LED available on the ARM7 board which glows only when the communication is not working. It gives the physical indication of communication link being working. The communication fail LED is turned on from the main program at boot up. As soon as ARM7 receives the last byte in the very first packet it is turned off, meaning communication fail LED is turned off as soon as communication link is established with ARM9.

In the system interrupt of ARM7 (PITC – Periodic Interval Timer Controller), which is of 50 ms the communication fail count is incremented. After reception of every packet this count is made 0. If the fail count equals 20 (meaning there has not been exchange of packet between ARM7 and ARM9 for the last 1 second), the Communication fail LED is turned on. This indicates the sudden failure in the ARM7–ARM9 communication link.

## 4 RESULTS

The performance of the ARM7–ARM9 communication link is measured in terms of time, the basic constrain while designing the protocol. The measurement of time is carried out with the help of digital storage oscilloscope (DSO). The waveforms shown here are actually captured screenshots of DSO.

### 4.1 TIME MEASUREMENT PROCEDURE

Time duration for the empty communication packet (0 bytes of data) transmission and its acknowledgement reception has been calculated by placing the DSO probe at one end of communication fail LED on ARM7 board. The LED status is toggled (OFF→ON, ON→OFF) after reception of every consecutive packet to get the square wave as shown in figure 5 and figure 6.

Time duration for the communication with data has been calculated by placing the DSO probe at one end of communication fail LED on ARM7 board. The LED is turned ON when the peripheral interrupt comes and it is turned OFF when the reply to the request sent to ARM9 is received back to get the waveforms as shown in figures 7 and 8.

### 4.2 WAVEFORM RESULTS

Figure 5 shows the waveform when both the ARM7s are connected in communication link and no data is transmitted (Empty packet communication). From the waveform obtained it is clear that single packet exchange takes 20 ms (DSO time/division = 10 ms/division).

Figure 6 shows the waveform when only one ARM7 is connected in communication link and no data is transmitted (Empty packet communication). From the waveform obtained it is clear that single packet exchange takes 10 ms (DSO time/division = 10 ms/division).

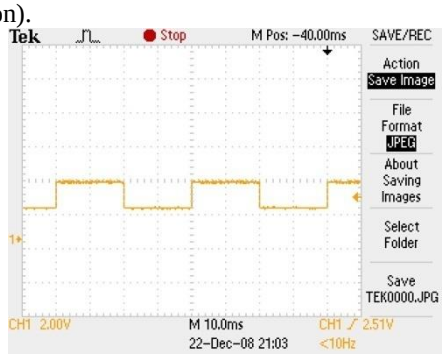


Figure 5: *Waveform – Empty packet communication when **both** the ARM7s connected (Communication Delay = 20 ms)*

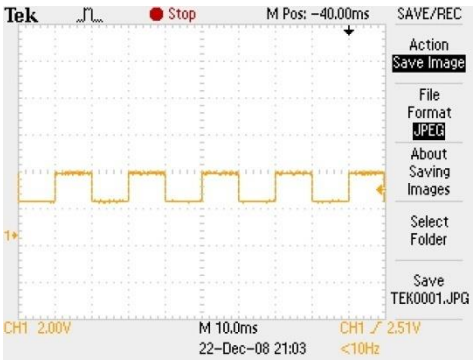


*Figure 6: Waveform – Empty packet communication when **only one** ARM7 connected (Communication Delay = 10 ms)*

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556


*Figure 7: Waveform – communication, when number of data bits are 26 (Communication Delay = 80 ms)*

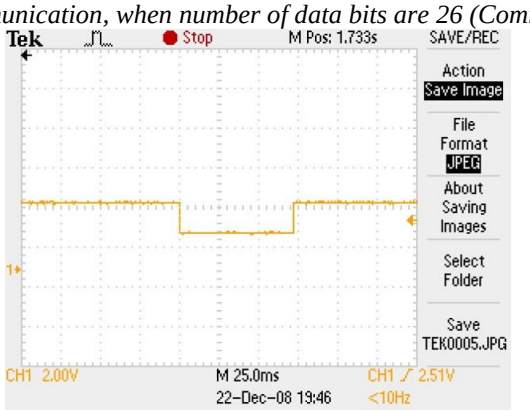


*Figure 8: Waveform – communication, when number of data bits are 44 (Communication Delay = 72.5 ms)*

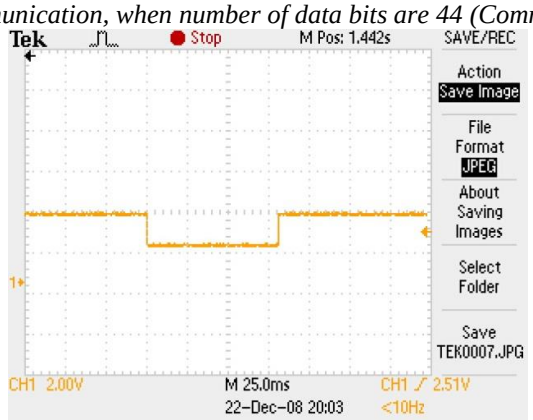


*Figure 9: Waveform – communication, when number of data bits are 30 (Communication Delay = 80 ms)*

***COMPARISON TABLE***

*Table 7.1 Communication Time Delay for Different Communication Situations*

| Master (ARM9) | Slaves (ARM7) | Number of Data bits | Communication Time Delay (in milli seconds) |
|---|---|---|---|
| 1 | 1 | Communication without data | 10 |
| 1 | 2 | Communication without data | 20 |
| 1 | 2 | 26 bits | 80 |
| 1 | 2 | 44 bits | 72.5 |
| 1 | 2 | 30 bits | 80 |

**5 Future Scope**

The future scope of the developed control system includes the firmware upgrade of ARM7 boards to which external peripherals are connected. Firmware upgrade enables the system to be modified of its functionality as

Vol 06, Article 07609; July 2015
http://ijves.com
International Journal of VLSI and Embedded Systems-IJVES
ISSN: 2249 – 6556

and when needed. As a part of security the Signature to the firmware of ARM7 has been incorporated. The signature enables the authenticity of the new firmware to be validated at the time of Firmware Upgrade.

**REFERENCES**

[1] Jinxue Zhang, Ming Zhang, (2009) "Research and Design of Embedded Tank Car Monitoring System Based on ARM9", *IEEE Computer Society*, Second International Symposium on Computational Intelligence and Design

[2] Shaoke Chen and Shaojun Jiang, (June 6-7, 2009) "Design of Embedded Network Interface Controller Based on ARM9 and ARMLinux", *Applied Computing, Computer Science, and Advanced Communication First International Conference on Future Computer and Communication*, FCC 2009, Wuhan, China, Proceedings – Springerlink.

[3] Mike Meyerstein, Inhyok Cha and Yogendra Shah, (June 3-5, 2009) "Security Aspects of Smart Cards vs. Embedded Security in Machine-to-Machine (M2M) Advanced Mobile Network Applications", *Security and Privacy in Mobile Information and Communication Systems. First International ICST Conference*, MobiSec 2009, Turin, Italy, Revised Selected Papers – Springerlink.

[4] Ou Qingvu, Huang Kai and Wu Xiaoping, (2009) "Research on the Embedded Security Architecture Based on the Control Flow Security", *IEEE Computer Society* : Second International Workshop on Computer Science and Engineering.

[5] Chandrasekaran, S. Rajendran, J. Annamalai, (2008) "Data Driven Security Alarm Model for Embedded Applications", Computing, Communication and Networking, ICCCn 2008. *International Conference – IEEE xplore digital library*.

[6] Guy Gogniat, Tilman Wolf and others, (February 2008) "Reconfigurable hardware for high-security/high-performance embedded systems: the SAFES perspective", *IEEE Transactions* on Very Large Scale Integration (VLSI) Systems, Volume 16 , Issue 2

[7] Ted Huffmire, Brett Botherton and others, (November-December 2008) "Managing Security in FPGA based embedded Systems", (vol. 25 no. 6), *IEEE CS digital Library*.

[8] R. Vaslin, G. Gogniat, J.-P. Diguet, R. Tessier, and W. Burleson. (200) "A security approach for off-chip memory in embedded microprocessor systems". *Elsevier Journal of Microelectronics and Microprocessors*.

[9] T. Eisenbarth, T. Guneysu, C. Paar, A.-R. Sadeghi, D. Schellekens, and M. Wolf. (November 2007) "Reconfigurable trusted computing in hardware". In Proceedings of the *ACM Workshop* on Scalable Trusted Computing.

[10] R. Elbaz, L. Torres, G. Sassatelli, P. Guillemin, M. Bardouillet, and A. Martinez. (July 2006), "A parallelized way to provide data encryption and integrity checking on a processor-memory bus". In Proceedings of *theIEEE/ ACMInternational Design Automation Conference*.