International Journal For Technological Research In Engineering
Volume 1, Issue 10, June-2014 *ISSN (Online): 2347 - 4718*

SECURE WIRELESS RECONFIGURATION OF FPGA BASED SOFT-PROCESSOR USING ARM PLATFORM

Pranoti K S[1] (PG Student), Mr. Manoj Kumar S B[2] (Asst. Prof.), Prof. Ananda Raju M B[3] (H.O.D)
[1]VLSI Design and Embedded Systems, [2, 3]Dept. of Electronics & Communication Engg.
BGSIT, BG Nagar, Karnataka, India.

Abstract: ***The use of reconfigurable logic in the field of computing has increased during the last decades. Much different architecture for reconfigurable exists today is an FPGA (Field Programmable Gate Array). In this paper the new concept called as Internet Of Things (IOT) is used which is sensitive, adaptive and responsive to human needs and it involves billions of digital devices, people, services and other physical objects which are connected, interacted and exchange information among themselves. The paper also shows the combined architecture of the ARM processor and FPGA. The reconfigured data in processor core are modified in the IOT devices with the type of wireless communication called Zigbee. This is an important feature of the sensor network. Finally, this paper proposes an embedded security framework as a feature of software/hardware co-design methodology.***

Keywords: ARM and Gumnut Processor, FPGA, Zig-Bee transceiver.

I. INTRODUCTION

ARM (Advanced RISC Machine) is the industry leading processor, which provides high performance with common RISC architecture. The ARM processor supports three states depending upon the number of bits of operation namely,

- ARM (32 bits of Operation)
- Thumb (16 bits of Operation)
- Jazella (8 bits of Operation)

ARM processor is preferred because of its load-store instruction, can access all 32 registers which are inbuilt in the processor, less energy consumption and Thumb code takes about 40% of less memory space compared to 32 bit ARM code.

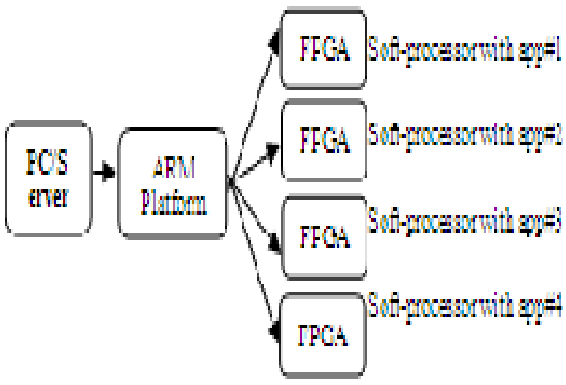

Fig 1: Basic Block Diagram

The microcontroller is used to execute the preloaded task. In this paper the microcontroller is used in a wireless sensor node for N distributed services requiring processing the data and transferring the data to a server node placed at any point in a distributed network. The distributed network requires a communication interface between the sensor node and the server. The interface can be done by both wired and wireless; in this paper, we preferred wireless because of its flexibility. The work of paper characterizes the XILINX FPGA (Spartan 3A) and ARM processor (LPC1248) of two different domains can be communicated through UART for distributed services. N number of sensor nodes can be connected to the FPGA through MIMO (Multiple Input and Multiple Output). The common issues in embedded network systems are power problem, insecure and distance coverage. This paper overcomes the above issues by using Zigbee technology, which is widely developed in wireless control and monitoring applications and it allows the communication between devices and the central computer.

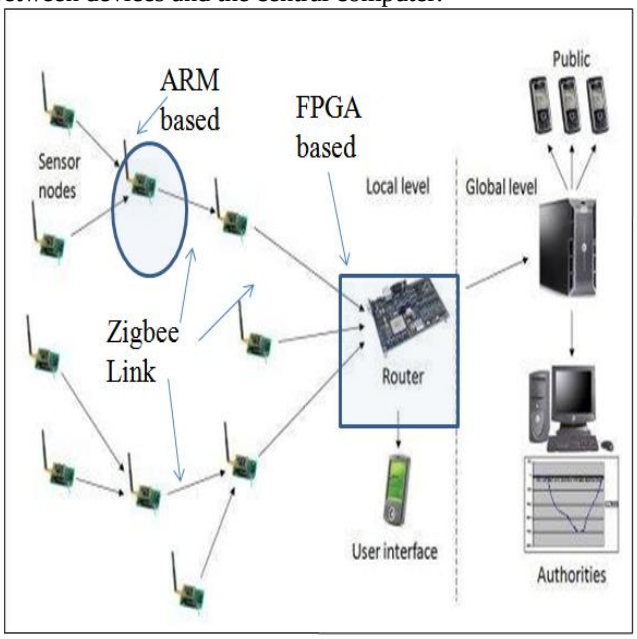

Fig 2: Wireless Sensor Network

II. OBJECTIVE

The objective of this paper is to prove the concept of communication between embedded and FPGA platform of two different domains by creating a prototype and configuring them through UART transmission and reception. And prove FPGA based soft-processors can be reconfigured in a secured way from a remote system using a wireless link.

III. SYSTEM DESCRIPTION

Imagine that the unmanned vehicle wants to be reconfigured from the remote location, it's very difficult to reconfigure the unmanned vehicle with wired communication in the IOT network so to overcome with this the wired communication is replaced with wireless communication.

The detailed block diagram of the whole system is as shown in Fig 3

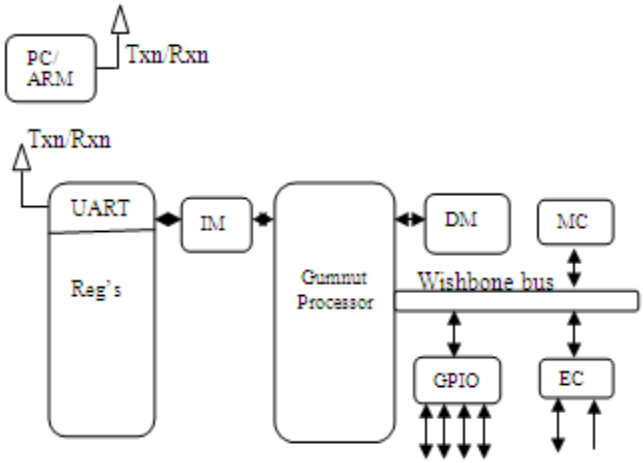

Fig 3: Detailed Block Diagram

IM - Instruction Memory
DM - Data Memory
MC - Math Co-processor
GPIO - General Purpose Input/output
EC - External Communication

A. Gumnut Processor

The gumnut has an instruction memory of up to 4096 18 bit instructions (using 12 bit addresses) and a data memory of 256 bytes (using 8 bit addresses). When the CPU of a system resets it clears the PC to 0 and starts executing instructions within the CPU. There are eight general purpose registers r0 to r7 where register r0 is special that it is hard wired to have the value 0, and any updates to it is ignored. The CPU also has two single-bit conditional-code registers called Z (zero) and C (carry). They are set to 1 or cleared to 0 depending on the result of certain instructions.

Below Table 1 shows the instruction set of Gumnut processor,

Table 1: Instruction Set of Gumnut Processor

Arithmetical and logical instructions	
add *rd, rs, op2*	Add *rs* and *op2*, result in *rd*
addc *rd, rs, op2*	Add *rs* and *op2* with carry, result in *rd*
sub *rd, rs, op2*	Subtract op2 from rs, result in rd
subc rd, rs, op2	Subtract op2 from rs with carry, result in rd
and rd, rs, op2	Logical AND of *rs* and *op2*, result in *rd*
or rd, rs, op2	Logical OR of rs and op2, result in rd
xor rd, rs, op2	Logical XOR of *rs* and *op2*, result in *rd*
mask *rd, rs, op2*	Logical AND of rs and NOT op2, result in rd

Shift Instructions	
shl *rd, rs, count*	Shift *rs* value left *count* places, result in *rd*
shr *rd, rs, count*	Shift *rs* value right *count* places, result in *rd*
rol *rd, rs, count*	Rotate *rs* value left *count* places, result in *rd*
ror rd, rs, count	Rotate *rs* value right *count* places, result in *rd*

Memory and I/O instructions	
ldm rd, (rs) ± offset	Load to rd from memory
stm *rd*, (*rs*) ± *offset*	Store to memory from rd
inp rd, (rs) ± offset	Input to rd from input controller register
out rd, (rs) ± offset	Output to output controller register from rd

Branch instructions	
bz ±*disp*	Branch if Z is set
bnz ±disp	Branch if Z is not set
bc ±disp	Branch if C is set
bnc ±*disp*	Branch if C is not set

Jump instructions	
jmp *addr*	Jump to *addr*
jsb *addr*	Jump to subroutine at *addr*

Miscellaneous instructions	
ret	Return from sub-routine
reti	Return from interrupt
Enai	Enable interrupts
Disi	Disable interrupts
Wait	Wait for interrupts
Stby	Enter low-power standby mode

B. Gumnut Bus Interface

The gumnut microcontroller core uses Wishbone buses to connect to the instruction memory, the data memory, and the input/output controller port, as shown in Fig 4. Each of buses uses classic single-read and single-write bus cycles.

C. UART (Universal Asynchronous Receiver/Transmitter)

UART based data transmission is an asynchronous form of data transmission. Serial data transmission in UART does not require a clock signal to synchronize transmitting and receiving end. Instead, it relies upon the pre-defined agreement between transmitting and receiving device by making identical serial communication settings between transmitter and receiver.

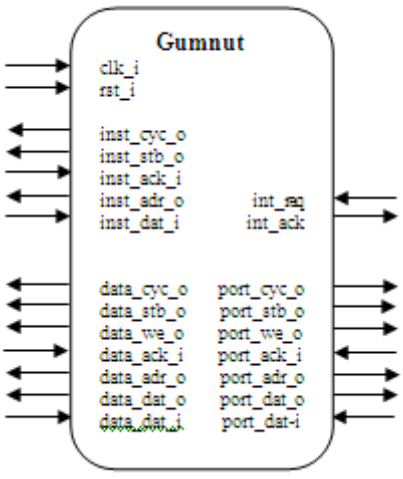

Fig 4: Wishbone Bus Connections on the Gumnut Core

Basic Operation of UART

To facilitate proper communication between transmitter and receiver, the 'transmit line' of the data sending device should be connected to the 'receiving line' of the receiving device and vice-versa as shown in Fig 5.

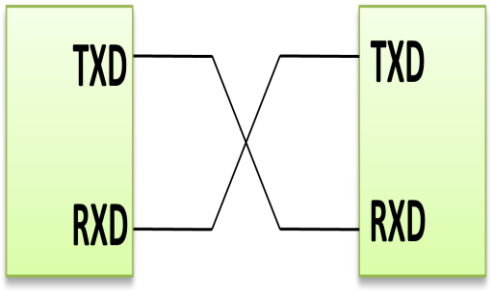

Fig 5: Basic UART

The basic UART data packet is as shown in Fig 6. This uses one start bit, 8 bits of data, no parity and one stop bit. Thus, it takes 10 bits to transmit a byte of data. The receiver synchronizes its internal clock to that of the transmitter's at the beginning of every data packet received.

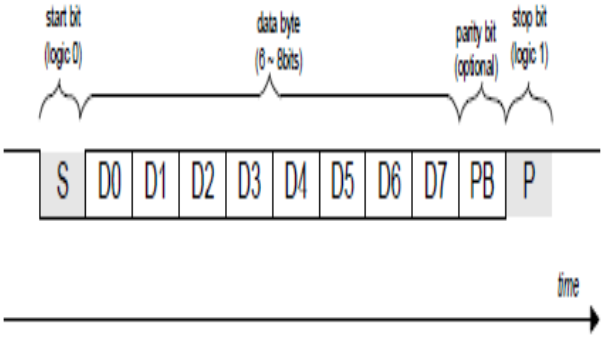

Fig 6: UART Data Packet Format

In this paper ARM LPC2148 provides 16byte receiver and transmitter FIFO's on two UART communication, the use of sendchar () and putchar () pre-defined C statements and configuring some UART registers will enable the UART transmission and reception.

Below Figure's shows the different bit configuration to enable the UART0 in ARM processor.

		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	
U0LCR	Line Control Register	DLAB	Set Break	Stick Parity	Even Par.Selct.	Parity Enable	No. of Stop Bits	Word Length Select		R/W

Bit	Symbol	Value	Description	Reset value
1:0	Word Length Select	00	5 bit character length	0
		01	6 bit character length	
		10	7 bit character length	
		11	8 bit character length	
7	Divisor Latch Access Bit (DLAB)	0	Disable access to Divisor Latches.	0
		1	Enable access to Divisor Latches.	

Fig 7: UART0 Line Control Register

Bit	Symbol	Description	Reset value
7:0	DLL	The UART0 Divisor Latch LSB Register, along with the U0DLM register, determines the baud rate of the UART0.	0x01

Bit	Symbol	Description	Reset value
7.0	DLM	The UART0 Divisor Latch MSB Register, along with the U0DLL register, determines the baud rate of the UART0.	0x00

$$UART0_{baudrate} = \frac{PCLK}{16 \times (256 \times U0DLM + U0DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Fig 8: UART0 Divisor Latch Register

		BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0	
U0LSR	Line Status Register	RXFIFO Error	TEMT	THRE	BI	FE	PE	OE	DR	RO

5	Transmitter Holding Register Empty (THRE))	THRE is set immediately upon detection of an empty UART0 THR and is cleared on a U0THR write.	1
		0 U0THR contains valid data.	
		1 U0THR is empty.	

Fig 9: UART0 Line Status Register

Pin	Type	Description
RXD0	Input	Serial Input. Serial receive data.
TXD0	Output	Serial Output. Serial transmit data.

Fig 10: UART0 Pin Details]

D. Math Co-Processor

The Math co-processor is also called as the numeric co - processor or Floating point co-processor and is defined as the mathematical circuit that performs high speed floating point operations and is generally built in computer chip. It helps to increase the overall speed and efficiency of a computer.

IV. HARDWARE AND SOFTWARE REQUIREMENTS

Table 2: Co-design Requirements

Wireless link	Zigbee
Soft-processor	Pico Blaze
FPGA	Spartan Processor
Microcontroller	ARM Processor
PC	Custom Atom Processor Based/Regular
Security Algorithm	Modified/Humming Bird
Operating System	Linux/Windows

V. IMPLEMENTATION

The work is characterized on proving the concept for wireless communication between two dissimilar devices using UART as a mode of communication. Fig 11 shows the general block diagram of the design with an ARM board placed on the transmission side and the FPGA board placed at the receiver side. The Zigbee transceivers are interfaced with the RS-232 connector to each of the boards to establish a wireless communication. The UART is the mode of serial communication programmed to transfer the instructions from the transmission side (ARM) to the receiving side (FPGA).

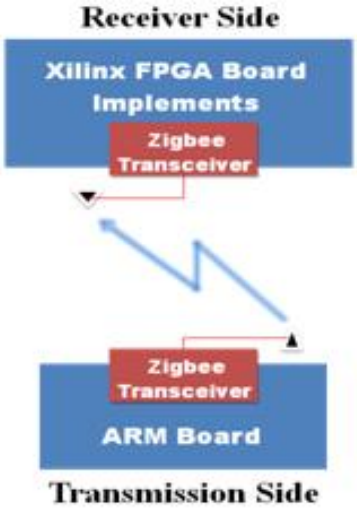

Fig 11: General Block Diagram

Transmitter Setup

The transmitter setup is as shown in Fig 12. The instruction to be transmitted is written as a string pointing to a character in a while loop. The hex file is dumped onto the processor and whenever the reset button on the board is pressed the UART0 channel configured will transmit the string of instruction wirelessly through the Zigbee transceiver to the receiving side.

Example: XXYX [50800] [158C2] [005F2] Y

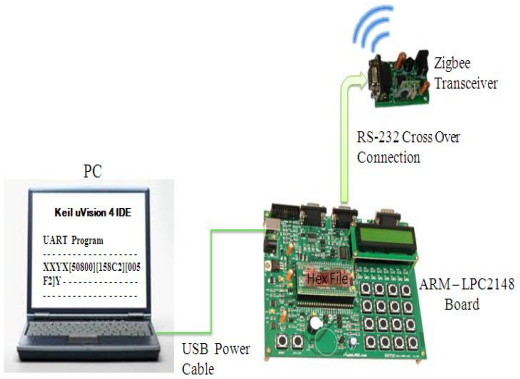

Fig 12: Transmission Setup

Receiver Setup

The receiver setup is as illustrated in the Fig 13, where the UART soft core is interfaced with the 8 bit soft core processor to provide connection to the ZigBee transceiver and the LED's on the board.

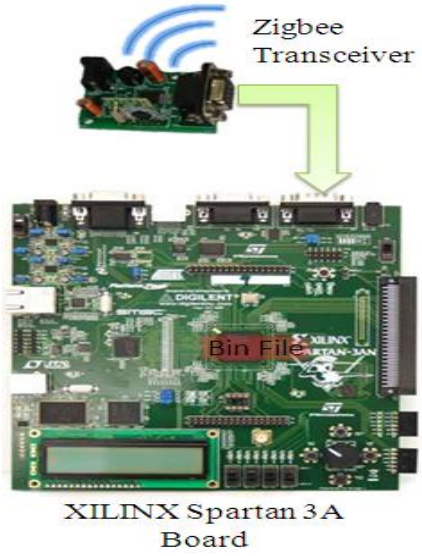

Fig 13: Receiver Setup

The detailed block diagram of receiver side is as shown in Fig 14, the main module Wiconfig_proc_systm. V provides the external communication to the ZigBee through Zigbee_rx wire. The three sub modules, namely instruction_capture. V, im.v and led_gpio. v are being programmed.

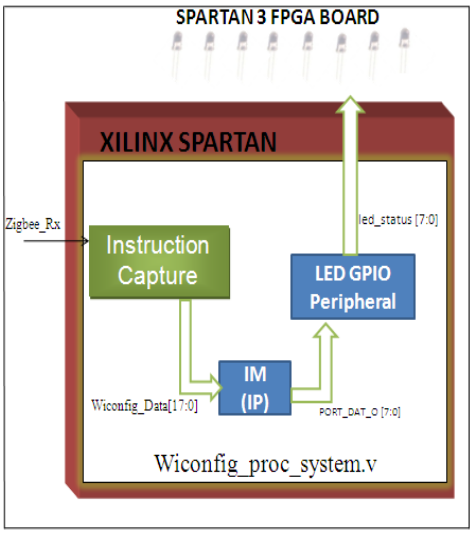

Fig 14: Detailed Receiver Block Diagram

VI. SCHEMATIC, SYNTHESIS AND SIMULATION

The FPGA used is XILINX Spartan 3A (Family), XC3S700 (Device), 5FG484 (Package), -5 (Speed Grade).

RTL Schematic

The Fig 15 is the complete design, RTL schematic with led_status [7:0] is the output to the 8 LED's and the signals zigbee_tx & Zigbee_rx are used for the interface to the external ZigBee transceiver. The baud_clock is the internal clock generated for the UART communication. The wiconfig_proc_system. v is a top wrapper module consists of

couple of sub-modules.

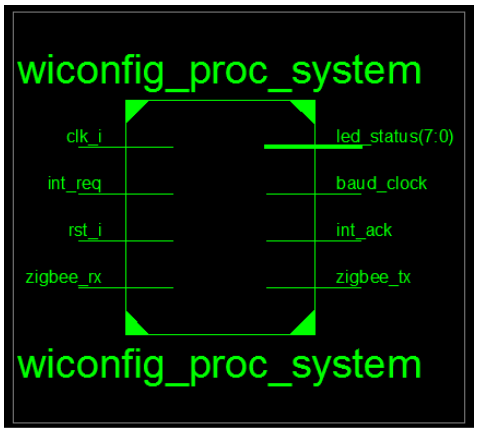

Fig 15: Design of RTL Schematic

Synthesis

The device utilization summary of the complete Verilog code is tabulated in the below Table 3. It can be observed that about only 2 % of the flops utilized thus resulting in the less consumption of the resources.

Table 3: Total Device Utilization

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	371	5888	6%
Number of Slice Flip Flops	303	11776	2%
Number of 4 input LUTs	611	11776	5%
Number of bonded IOBs	15	372	4%
Number of BRAMs	5	20	25%
Number of GCLKs	2	24	8%

Simulation Results

Simulation is done using a XILINX ISIM simulator, which is inbuilt simulator in XILINX project navigator. The Fig 16 shows the simulation results of the instructions being transferred. It can be observed that the ASCII value X is turning on the led [7] as high initially and rest other led's as zero. The idata [7:0] is the 8 bit data that is being received by the receiver UART.

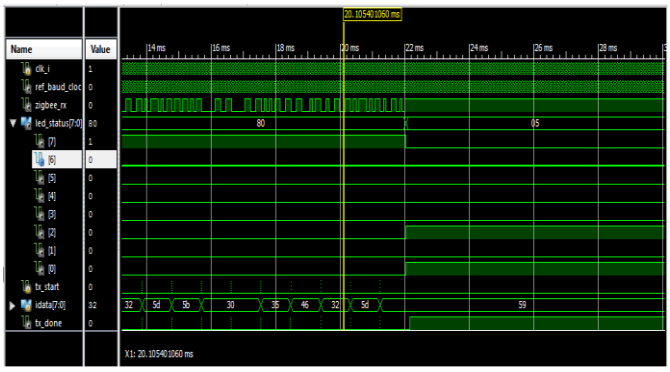

Fig 16: Simulation Result of led[7]

It can be observed in the Fig 17 that after a period of time the 8 LED's are being placed with a value of 0x05 on LED [0] & LED [2]. Whereas the LED [7] is being turned off as the ASCII Value Y turns off. Tx_done is set one soon after the 0x05 is placed on LED's.

VII. CONCLUSION

The following work characterized under the Xilinx FPGA (Spartan 3A) and ARM processor (LPC2148) of two different domains communicating through UART for distributed services. Since ARM is a leading industry processor, which provides better interoperability service for all the intercommunication process than other embedded processor.

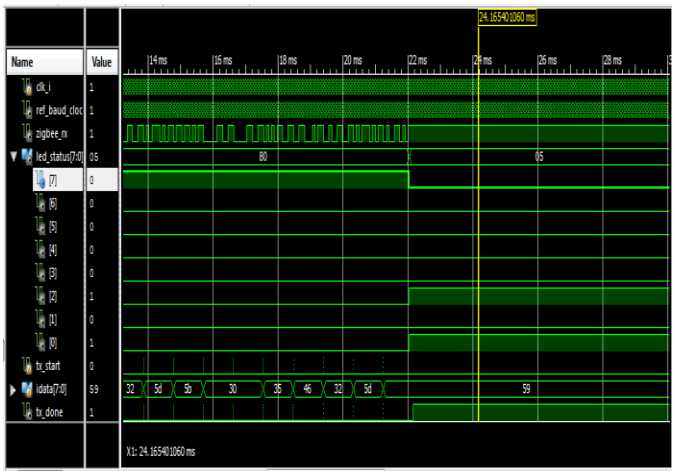

Fig 17: Simulation Result of led [0] & led [2]

In industry, n-number of processors are used to perform various functions like monitoring, controlling and commanding. So the interconnection between more than one processor using ARM will lead and enhance the system performance of the sensor network. By using interconnection between more than two processors, the utilization rate of all the processors in the network is increased. Also the delay between the processor executions is reduced. In order to achieve perfect co-ordination between the processors in the network the total system load are drastically reduced. Hence the processor failure due to practical reasons is drastically reduced. Thus the overall system efficiency is improved.

VIII. APPLICATIONS

- Wireless sensor network
- Mobile base stations
- Space applications

IX. FUTURE SCOPE

N number of processors could be connected as a network and each processing device can be connected to the FPGA through MIMO (multiple inputs and multiple outputs).

REFERENCES

[1] Rolf H. Weber, "Internet of Things- New security and privacy challenges", Computer Law & Security Review, Volume 26, Issue 1 January 2010, pages 23-30

[2] Srivaths Ravi, Anand Raghunathan, Paul Kocher, Sunil Hattangady, "Security in embedded systems: Design challenges", August 2004, Transactions on Embedded Computing System (TECS), vol 3 Issue 3, ACM

[3] Mutthew Eby, Jan Werner, Gabor Karsai, Akos Ledeczi, "Embedded systems security co-design", April 2007, SIGBED Review, Vol 4 Issue 2 Publisher ACM

[4] Tiri, K and Verbauwhede,"Design method for constant power consumptionof differential logic circuits", In proceedings of the Conference on design, Automation and test in Europe-vol 1 Design, automation and test in Europe. IEEE Computer society, Washington DC, 628-633

[5] T. Kernis, W.P. Marnane E.M Popovici: An implementationof a flexible Secure Elliptic Curve Cryptography Processor. Distinguished Paper. International Workshop on Applied Reconfigurable Computing ARC 2005, Proceedings, pp.22-30, IADIS press

[6] Fons, M; Fons, F;Canto, E;"Embedded security: New trends in personal recognition systems"; Microelectronics and electronics conference, 2007.RME.Ph.D. Research in 2-5 july 2007

[7] Qu Lei, Liu Shengde & Hu Xianbin "ZigBee Technology and Application", 2007

[8] I. F. Akyildiz, W.Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. Computer networks, (38):393-422,2001

[9] B.NEFZI and Y. Q. song. Performance analysis and improvement of zigbee routing protocol. In 7th IFAC International conferenceon fieldbuses & Networks in industrial &Embedded systems, Pages 351-369, France.2007

[10] C. Perkins and E.B. Royer. Ad-hoc On-demand distance vector (AODV) Routing, RFC3561. IETF MANET Working Group, 2003.

[11] ZigBee Standards Organization, ZigBee 2006 specification, 2006.

