ISSN: 2319-8753

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 6, June 2014

Design and Implementation of EPC Gen2 Using FPGA

Pallavi J [1], Kavitha T M [2]

P.G. Student, Department of electronics and communication, DBIT, Bangalore, India[1]

Assistant professor, Department of electronics and communication, DBIT, Bangalore, India[2]

ABSTRACT: Electronic Product Code or EPC Global class-I Generation-I and Generation-II are popular and globally accepted standard specification which enables the use of password protection for accessing the data stored in the memory of a RFID tag. Use of password gives one of the ultimate securities for the RFID system but these are not immune to 'hacking'. Hackers are adopting advance technologies like side attack, power analysis to detect data password. But in many cases it is very much important that data must be secured enough. In this paper, we propose a data security system integrated with this protocol based on RSA algorithm. The RSA system is widely employed in networking applications and achieves good performance and high security. We have implemented the protocol in VERILOG.A synthesizable module is presented here in this article. System efficiency is evaluated according to the implementation.

KEYWORDS— EPC Gen-2, RFID, RSA, VERILOG, RTL schematic, Simulation, Synthesis report.

I. INTRODUCTION

The global standard EPC Gen-2 protocol is the advanced version of EPC Gen-1 protocol. Some disadvantages and shortcomings of EPC Class 1 Gen1 overcome in the revised EPC Gen-2 protocol, which uses ALOHA based random anti-collision protocol called Q protocol (Q algorithm).It is of frame slotted ALOHA based probabilistic algorithm for tag anti-collision protocol with dynamic frame length.

VLSI implementations of algorithms are very important to achieve desired reliability, performance, low cost and high speed production. There are three possible types of slots; empty slot, when there is no tag to reply. Collided slot, when more than one tag to respond and the successful slot, when there is only one tag to respond and exchange information. The three conditions are depicted in Figure 1 with Q value 2. Tags are asked to choose slot number SN within the range $0 \leq SN \leq (22-1)$. So, the tags can choose 0, 1, 2 and 3 as their SN. But, if no tag is chosen, SN = 0, then the slot will be an empty and no tag will reply. When more than one tag choose SN = 0, it results a collided slot and reader modified its Q value and ask the tags to choose new SN. Successful slot, when only one tag responds to the Reader with SN = 0 and generates the 16 bit number RN16 and transmit.

5_

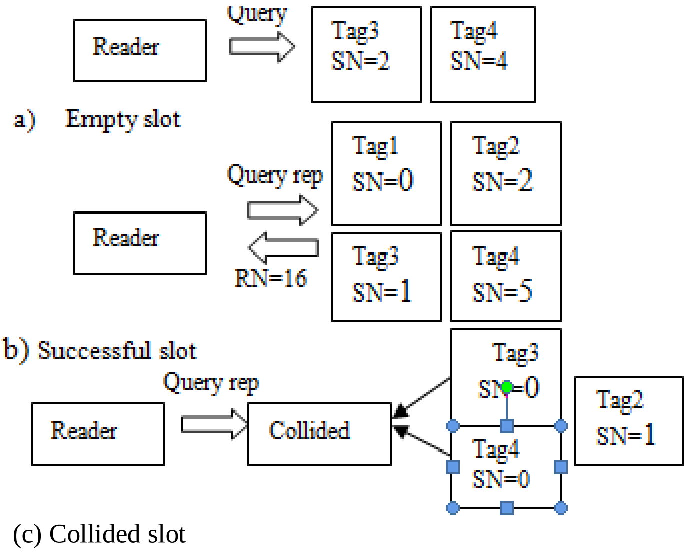


Fig1: Empty, successful and collided slots.

EPC Class 1 Gen1 and EPC Class 1 Gen2 protocols are ratified as the International standard anti-collision protocol. But in EPC code data are not encrypted, rather covered by more pseudo-random number or data. Hence attackers or hackers can decode the data in several ways. So, password security does not provide the ultimate security for important data. It is possible to discover the details personal data if anybody has the knowledge of EPC references. Side channel attacks are executed by watching the power consumption or variations of the energy output of the devices. Important data or personal information especially sensitive personal data will need an adequate level of encryption to safeguard the data. Hence instead of PCA which is used for securing the data in the existing system we are using a cryptography method in order to give high level security to the data or password. We have used RS algorithm to generate a secret code using a secret key within the EPC tag. We have designed the secret code generator and the processor both for the reader and the tag in VERILOG code. Hence an unauthorized Reader will not be able to read the data unless the key is available.

II. RELATED WORK

[1] With reference to the paper **Analysis of Tree Algorithms for RFID Arbitration** the nee Search (or splitting) method introduced by Capetanakis for use in conventional multi-access systems can also be applied to RFID arbitration. This paper performs a transient analysis of this, and related methods. Radio Frequency Identification (RFID) systems have emerged as a viable and affordable alternative for tagging and/or labelling small to large quantities of items. The arbitration problem is that of identifying all 'tags' present in the current environment (i.e. reading the labels). This is a multi-access communication problem in which contention for the communication medium is transient in nature. Important measures of performance include the time required to identify the tags, and the power consumed by the tags. The first of these is proportional to the number of time slots needed to complete the arbitration process, and the second is proportional to the total number of times the tags are required to transmit during this process. We analyze a family of nee Search algorithms with respect to these two measures. Previous analysis on the average number of time slots reveals a linear dependence on the number of tags. We extend this work by developing expressions for the decomposition of these slots into three basic types: zero, one and multiple reply slots. Original expressions are also developed for the total number of tag replies.

[2]With the analysis of the paper **A Programmable Hardware Cellular Automaton: Example of Data Flow Transformation** we present an IP-core called PHCA for Programmable Hardware Cellular Automaton (CA). PHCA is a hardware implementation of a general purpose cellular automaton entirely programmable. The heart of this structure is a PE array with reconfigurable side links allowing implementing a 2-D CA or a 1-D CA. As an illustration of a PHCA program we present the implementation of a symmetric cryptography algorithm called ISEA for Ising Spin Encryption Algorithm. Indeed ISEA is based on a 2-D Ising spin lattice presenting random series of disordered spin configurations. The main idea of ISEA is to use the disordered spin configurations to encrypt data. Cellular automata

were originally introduced by von Neumann for studying self-reproduction in biological systems. Then they have been used for language recognition and modeling of physical systems. The mathematical properties of cellular automata were also studied. Nowadays the concept of quantum cellular automata is defined everywhere.

[3]According to the paper **Query Tree-Based Reservation for Efficient RFID Tag Anti-Collision** Since the tree based RFID tag anti-collision protocol achieves 100% read rate, and the slotted ALOHA based tag anti-collision protocol allows simple implementation and good performance in small amount of tags, we consider how to take the advantages of each algorithm. This paper presents query tree based reservation for efficient RFID tag anti-collision. According to the simulation results, the proposed protocol requires less time consumption for tag identification than the present tag anti-collision protocols implemented in EPC Class 0, Class 1, and Class 1 Gen. **2.** FID systems are coming the most promising technologies used for contactless object identification. One of the research areas in RFID systems is a tag anti-collision protocol; how to reduce identification time with a given number of tags in the field of an RFID reader. For fast tag identification, anti-collision protocols, which reduce the frequency of collisions occurred and identify tags independent of occurring collisions, are required. Moreover, tag anti-collision protocols are more important than reader anticollision protocols because of the incompetence of tags in low-cost passive RFID systems. There are two types of tag anti-collision protocols for RFID systems: deterministic methods and probabilistic methods The deterministic methods are tree based protocols The tree based protocols keep splitting the group of colliding tags into two subgroups until all tags are identified. The probabilistic methods are based on slotted ALOHA. The slotted ALOHA based protocols reduce the probability of occurring tag collisions how tags respond at the different time. This paper considers how to improve the performance of the RFID tag anti-collision protocol by applying the characteristics of EPC Class 1 Gen2 protocol to the query tree algorithm.

III. SYSTEM AND DESIGN

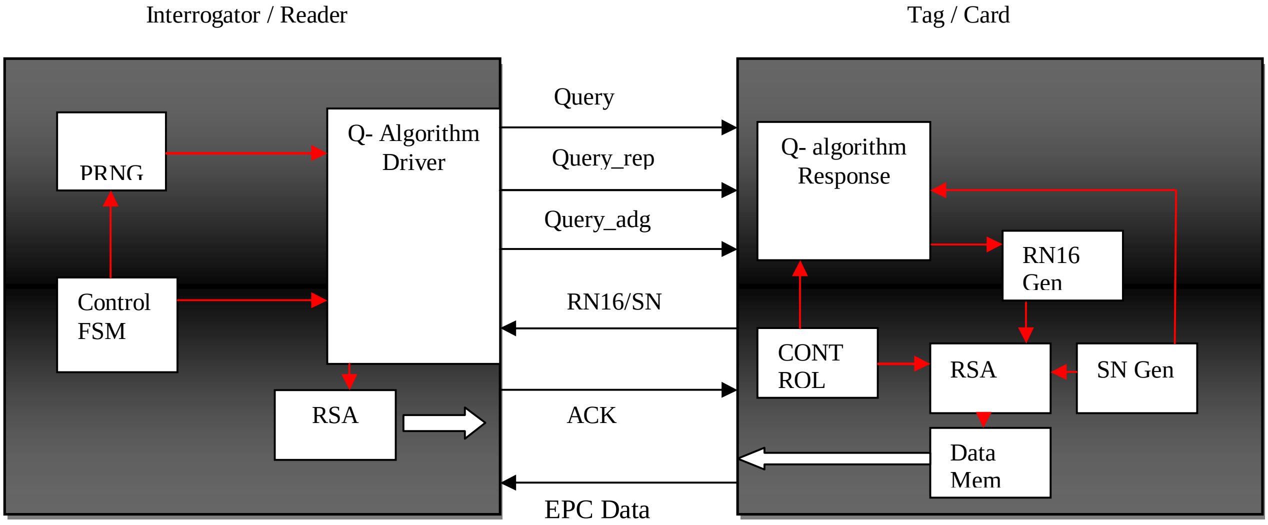


Fig 2: Block diagram of EPC

As soon reset is released the FSM triggers the Q algorithm block of the interrogator module to start the communication with reader block. Parallely it also enables the PRNG block to generate the SN numbers to be able to compare the SN received from reader module. Once the Tag or reader is identified the RSA module will be enabled by control FSM, the RSA module starts decoding the received data based on RSA decryption algorithms. The reader module will also simultaneously starts the control FSM as soon reset is released. The control FSM triggers the Q algorithm response block to respond the interrogator also it triggers the SN Gen module to generate the Random SN number which will be transmitted to interrogator as soon tag is identified, FSM triggers the RSA block, RSA block on request from interrogator reads the data from the Data Memory and then Encrypts it based on the RSA algorithm with private and Public key and then it will be transmitted on the data line.

PRNG MODULE :The architecture of the PRNG block is given in Fig 3. For the whole circuit, there are five input signals. Those are the input signal to be shifted into the serial-to-parallel shift register as the input seed for RNG to generate the random numbers (Inbit), the clock pulse for PRNG and shift register (Clk), the reset and activate signal for PRNG (Reset), the reset signal for shift register (Reset_shift), and the enable signal for the register (Enable_reg). The output pins are the 16-bit random number output signal (Outbit) and the read enable signal for the next stage (Enable_read).

The PRNG circuit consists primarily of three blocks of circuitries: a 16-bit Serial-to-Parallel shift register, a 16-bit register, and a PRNG module which takes in an input seed of 16-bit width and generates a random 16-bit number using the Park-Miller Algorithm. The serial-to-parallel shift-in register is used to set the initial states of the 16bit RNG. It shifts in the single bit inputs “Inbit‟ to form a 16-bit input seed under the control of the “Reset_shift‟ signal. After initialization, at every rising edge of the “Clk‟ signal, if the “Enable_reg‟ signal is high, the 16 bit register will take in the value of the outputs of the shift register and the PRNG module will update its state according to the output of the multiplexer.

-

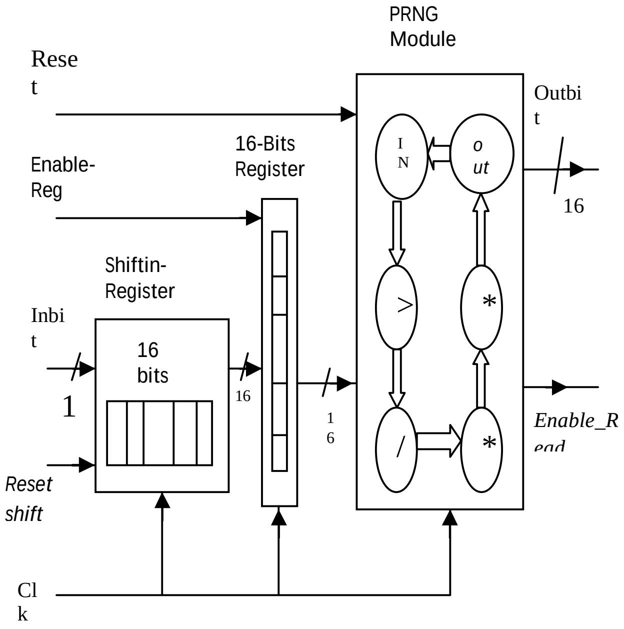


Fig3: Block diagram of PRNG

RSA : Our proposed method uses the RSA algorithm compared to PCA method proposed in referred papers. Below are details of RSA algorithm and implementation. The entire RSA cryptosystem is divided in to three parts: key generation, encryption and decryption. The design of each module required to build the RSA cryptosystem is described in this chapter. The implementation block diagram of RSA Cryptosystem is shown in fig 4.

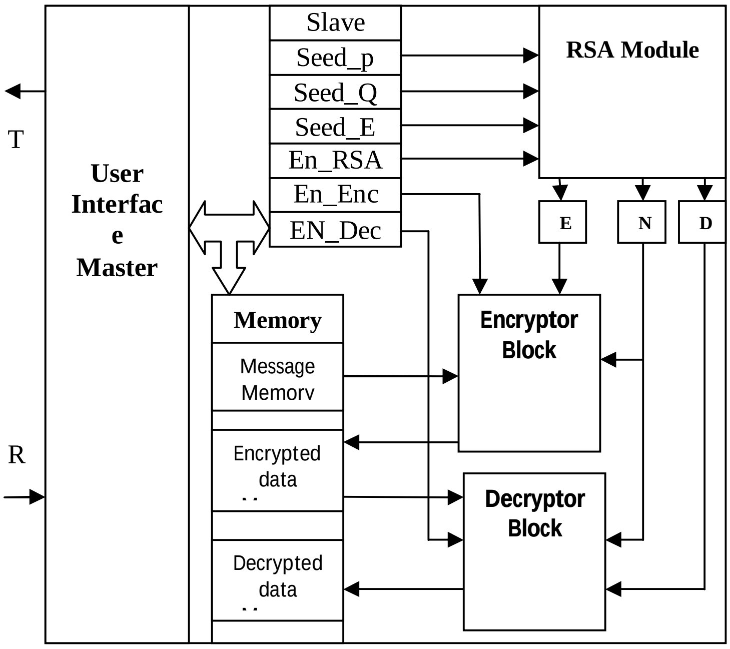


Fig 4: Implementation block diagram of RSA Cryptosystem

As shown in the fig 4 seed values required for the RSA module and En_RSA, En_Enc, En_Enc are stored in the slave register. RSA module generates public key “E‟, private key “D‟ and the modulus “N‟ by using random number generator, prime number detector, Booth multiplier and Extended Euclidean Algorithm. The implementation block diagram of the RSA module is shown in fig 4. When the En_enc signal is toggled Encryptor takes the message, private key E‟, modulus N‟ and generates cipher text which is stored in the Encrypted data memory. When the En_dec signal is toggled Decryptor takes the cipher text, public key “E‟, modulus “N‟ and generates decrypted data which is stored in the decrypted data memory. These are connected to the User Interface Master in which data are received and transmitted to the user.

Encryption : A critical requirement for the proper functioning of the RSA algorithm is that the message M must be represented as an integer in the range [0, n-1] (where n is, as usual, the RSA modulus). Our application converts text messages to integers by using a simple mapping of every character to its ASCII code and encrypts two characters at a time. For an RSA encryption scheme with the modulus size of 1024 bits, we have seen that about 100characters can be encrypted at once. Lesser number of characters cause encryption and (especially) decryption to take significantly longer, whereas higher number of characters often violate the condition that the message M must lie in the interval [0, n-1]. The encryption works using the public key (e, n). For any input message “m”, the encrypted message “c” would be calculated is using:

C=Me mod n

In above calculation “e” can be a very large number. In this project, “e” is a sixteen bit number. If the Me is calculated first them for a very small value of “e” then resultant number can be very large and cannot be stored in the memory.

The message “M” will be sent multiplier, at first multiplicand and multiplier will be same as message “M” then the product is divided by value of “N” then the modulus of it will be sent back to multiplier if the counter value is less than the “E” value or else the modulus is considered as encrypted data. Once counter value reaches to the value of “E” it also toggles a “done” signal to indicate that the encryption process is over.

Decryption: As the user chooses to decrypt an encrypted file, the decryption part is invoked. The operation of this routine is quite straightforward. From the file to decrypt (the path to which is input from the user), the function processes each encrypted integer. It does so by computing the value of cd mod n and stores the decrypted part. Of course, the values of d and n are read in earlier. Here M however contains the integer representation of the information i.e. it is a string of numbers where each 2 character sequence signifies the ASCII code of a particular character. An inverse mapping to the relevant character is carried out and the information, now as was in the original file is displayed on the standard output. The decryption process is over once all the integers (cipher text) in the encrypted file are processed and decrypted in the described manner.

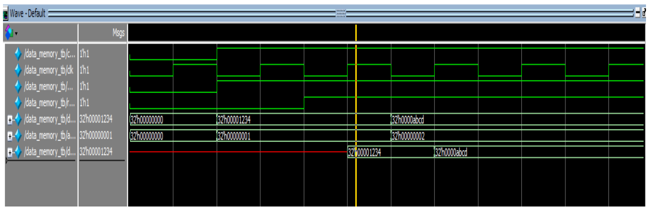
M=Cd mod n

In above calculation "d" can be a very large number. In this project, "d" is a sixteen bit number. If the cd is calculated first them for a very small value of "d" then resultant number can be very large and cannot be stored in the memory decryption.

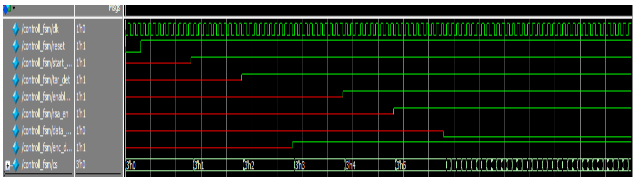
The encrypted data "C" will be sent multiplier, at first multiplicand and multiplier will be same as encrypted data "C" then the product is divided by value of "N" then the modulus of it will be sent back to multiplier if the counter value is less than the "D" value or else the modulus is considered as decrypted data. Once counter value reaches to the value of "D" it also toggles a "done" signal to indicate that the decryption process is over.

IV. SIMULATION RESULT FOR THE MODULES

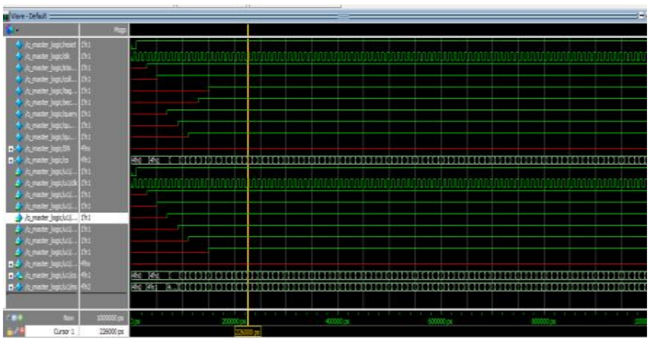
Simulation results of (a) Control FSM (b) Data memory (c) q_algorithm (d) RNG (e) RSA (f) Top module



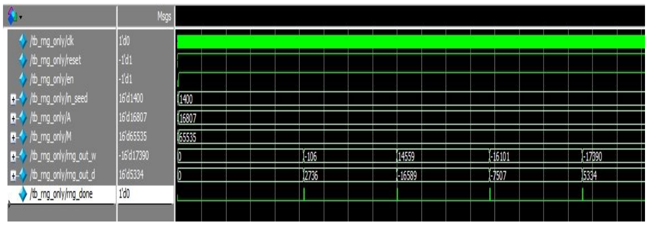
(a) It has 7 states where idle, send_querry, Rx_sn, send_querry_rep,querry_adj, sec_det, tag found are inputs and cs(current state),ns(next states) are outputs. Here clk and reset are global signals which are high for all states.



(b) Here clk and reset are the global signals which will be high always. It has 6 states where idle, start, tag_det, ENC_det_rx, RSA_en, data_rxd is inputs and cs (current state) is the output. All states will be enabled when it is high. If it is not high it goes back to the idle state and waits for the present state to become high.

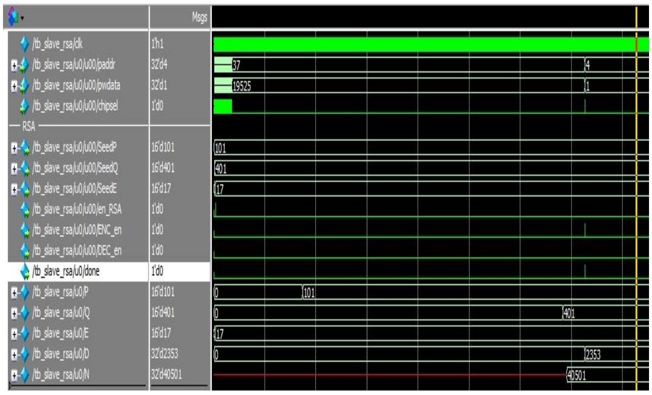


(c) Data memory is used to store the data which is designed under the TAG module. It stores the data up to 1024 bits of data which supports the Spartan 6 fpga kit. Here clk, data_in_bus, data_out_bus, address_bus, chip_select, read_enable and write_enable are the inputs and 32 bit memory is the output. Here clk is high for the entire process



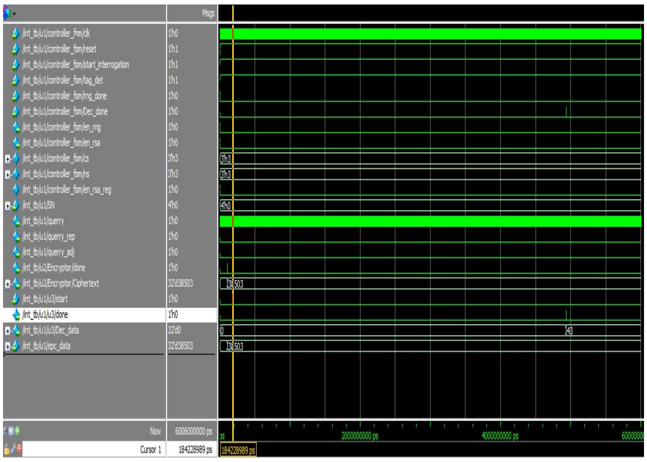
```
/tb_rng_only/clk          1'd0
/tb_rng_only/reset        -1'd1
/tb_rng_only/en           -1'd1
/tb_rng_only/in_seed      16'd1400
/tb_rng_only/A            16'd16807
/tb_rng_only/M            16'd65535
/tb_rng_only/rng_out_w    -16'd17390
/tb_rng_only/rng_out_d    16'd5334
/tb_rng_only/rng_done     1'd0
```

(d) Random number is generated using a park miller algorithm which i s based on the linear form. In the above simulation result shown clk, reset and en(enable) signals are high throughout the design. In_seeed is the input given of 16 bit, A is a constant which is the input signal and M is the message signal is always a 16 bit random number



```
/tb_slave_rsa/clk                  1'h1
/tb_slave_rsa/u0/u00/paddr         32'd4
/tb_slave_rsa/u0/u00/pwdata        32'd1
/tb_slave_rsa/u0/u00/chipsel       1'd0
RSA
/tb_slave_rsa/u0/u00/SeedP         16'd101
/tb_slave_rsa/u0/u00/SeedQ         16'd401
/tb_slave_rsa/u0/u00/SeedE         16'd17
/tb_slave_rsa/u0/u00/en_RSA        1'd0
/tb_slave_rsa/u0/u00/ENC_en        1'd0
/tb_slave_rsa/u0/u00/DEC_en        1'd0
/tb_slave_rsa/u0/done              1'd0
/tb_slave_rsa/u0/P                 16'd101
/tb_slave_rsa/u0/Q                 16'd401
/tb_slave_rsa/u0/E                 16'd17
/tb_slave_rsa/u0/D                 32'd2353
/tb_slave_rsa/u0/N                 32'd40501
```

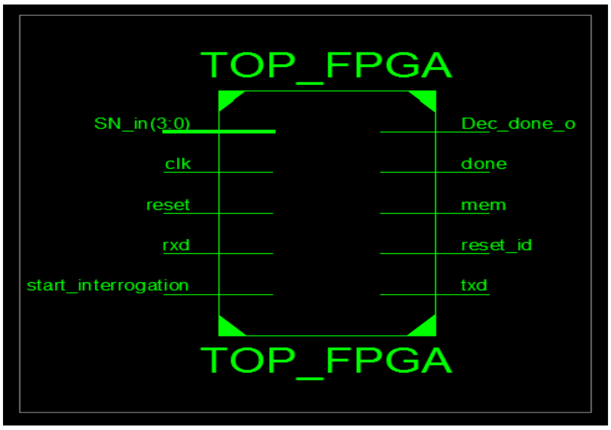
(e) Here clk is a global signal.paddr, pwdata and chipsel are the input programmable busses. Seed p, q,e are the inputs to the rsa module where p,q,e are the inputs to the seed p,q,e respectively. Dec_en and enc_en are the toggle signals to enable encryption and decryption process. D is the decrypted key and N is the encrypted key generated as outputs from the rsa module.



```
/int_tb/u1/controller_fsm/clk                  1'h0
/int_tb/u1/controller_fsm/reset                1'h1
/int_tb/u1/controller_fsm/start_interrogation  1'h1
/int_tb/u1/controller_fsm/tag_det              1'h1
/int_tb/u1/controller_fsm/rng_done             1'h0
/int_tb/u1/controller_fsm/Dec_done             1'h0
/int_tb/u1/controller_fsm/en_rng               1'h0
/int_tb/u1/controller_fsm/en_rsa               1'h0
/int_tb/u1/controller_fsm/cs                   3'h3
/int_tb/u1/controller_fsm/ns                   3'h3
/int_tb/u1/controller_fsm/en_rsa_reg           1'h0
/int_tb/u1/SN                                  4'h0
/int_tb/u1/querry                              1'h0
/int_tb/u1/querry_rep                          1'h0
/int_tb/u1/querry_adj                          1'h0
/int_tb/u2/Encryptor/done                      1'h0
/int_tb/u2/Encryptor/Ciphertext                32'd38503
/int_tb/u1/u3/start                            1'h0
/int_tb/u1/u3/done                             1'h0
/int_tb/u1/u3/Dec_data                         32'd0
/int_tb/u1/epc_data                            32'd38503

Now        6006000000 ps
Cursor 1   184228989 ps
```

(f) Clk and reset is high throughout the process. Once reset is high reader will start interrogation with the tag and when tag is detected tag_det will become high and send an acknowledgement rng_done back to reader when it is high. Then dec_done signal is enabled and the decrypted data is received at the output. The cipher text generated is the random number generated for the encrypted message output signal



(g)RTL schematic of top module EPC Gen2 system

Above figure shows the RTL of the top module of EPC gen2 system design. It consists of clk, reset, rxd, mem, start_interrogation as input signals and dec_done, sn_in and txd as output.

V. CONCLUSION

In this paper we have designed and implemented the EPC Gen-2 protocol in VERILOG code to get better performance, improved speed and efficiency and security. The design is targeting the Xilinx FPGA device, this leads to real time environment verification of the system.VLSI Design of protocol is performed to achieve high speed, minimum hardware and enhanced system efficiency with highly secured data transfer to authenticated reader. The RSA architecture generates all the possible random numbers for the given 16 bit input and stores it in a memory. Prime numbers from the generated random numbers were identified and stored in a memory. The first two prime numbers stored in the memory is selected for encryption. Also the generation stops as soon as the system detects two prime numbers. The efficiency will be increased by a considerable amount in terms of the above mentioned parameters.

REFERENCES

[1] A user guide EPC Gen-2, Thing Magic, version1.0, April2005, www.thingmagic.com.

[2] Yinghua Cui and Yuping Zhao, A Modified Q-parameter Anti- collision scheme for RFID systems, October 12-14, Beijing, China. J. H. Choi, D.Lee and H. Lee, Query Tree-based reservation for efficient RFID tag anti-collision, IEEE Communications Letters, vol. 11, pp.85-87 (2007).

[3] Ching-Nung Yang and Jyun-Yan He, An Effective 16-bit Random Number Aided Query Tree Algorithm for RFID tag Anti-collision, IEEE Communications Letters, vol.15, No.5 (2011).

[4] H. Yang, J. Huang, T. Chang, “A chaos-based fully digital 120 MHz pseudo random number generator,” Circuits and Systems, 2004, the 2004 IEEE Asia-Pacific Conference on, pp.357-360, 6-9 December 2004.

[5] T. Sato, K. Kikuchi, M. Fukase, “Chip Design of a Wave-Pipelined PRNG,” Communications and Information Technologies, 2006, ISCIT 2006, International Symposium on, pp. 978-983, 18-20 October 2006.

[6] S. K. Park and K. W. Miller, “Random number generators: Good ones are hard to and,” Communications of the ACM, vol. 32, Issue 10, pp. 1192-1201, October 1988.

[7] Chiranth E Chakravarthy H.Y.A, Nagamohanareddy P, Umesh T.H, Chethan Kumar M, " Implementation of RSA Cryptosystem Using Verilog', International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-201IISSN 2229-5518

[8] R.L.Ri vest, A.Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public- Key Cryptosystems", communicatios of the ACM21.

[9] Joyashree Bag, Rajanna K.M. Subir Kumar Sarkar, Jadavpur University, Jadavpur, Kolkata-700032.