

FPGA BASED RANDOM NUMBER GENERATION ACCESSED THROUGH ARDUINO

V Raghu Ram¹, T Naveen Kumar², G Kishore Naidu³, K Divya Kanti⁴

^{1,2,3,4}*ECE Department, Lendi Institute of Engineering and Technology, (India)*

ABSTRACT

In this paper we are using the concepts of random number generation for high security random digital lockers. As the present digital lockers are accessed with single password this may lead to security problems. To provide high security to these locker by using the concepts of random number generator. By using VHDL programming random numbers are generated and simulated in XILINX ISIM simulator and implemented on SPARTAN-3 FPGA kit. And further processed on ARDUINO MEGA 2560 board. As the world is adopted to mobile technologies ,we are interfacing the mobile technologies with microcontroller. By using Arduino ATMEGA 2560 microcontroller we are generating random password. These passwords are accessed through android mobile application.

Keywords: *Arduino, FPGA, Random number generators, Pseudo-random number, cryptography.*

I. INTRODUCTION

A vast development in technological aspects becomes one of the major problem for high security communication. These problems can be solved by applying crypto graphical techniques. The basic concept of the cryptography depends on random numbers. Random number generators are of two types, they are true random number generator (TRNGs) and pseudo random number generators (PRNGs). The prediction output of the true random number is not possible whereas in pseudo random number generator is based on the algorithm it may possible to predict the output. The best real time examples for true random number generator is rolling of dies, tossing a coin and picking a card from playing cards.

Our idea is to implement a random password lock system using the basic concepts of random number generator. As the technological aspects changes in the world a lot of implementation in the digital lock systems. But , they use the unique concept of the password systems. These systems depends on the password . If password is correct it will open unless it will not open. Hacking of the password is very simple job due to the development in the technology. To reduce the drawbacks of these digital lock systems we are using the technique and the concept of random number generator in digital lock systems.

As the world is being adopted to mobile technologies like Android, Apple, Windows. We are going to interface the micro controller with our mobile networks. This application in our mobile will be communicate with the micro controller . So, that the micro controller will send the random password to the mobile application . The received random password will be displayed on the mobile application. The new pass word will be generated for every regular time interval . The newly generated random password will be displayed on the mobile application every time. There is a password box in the mobile application to write the password . There is send button to

send the password to the micro controller. In the micro controller the send password is going to compare with the password from the mobile application. If both are same then it will open the lock. Otherwise it will not accept to open.

II. IMPLEMENTATION OF RANDOM NUMBER GENERATORS

By using the basic concepts of VHDL language the random numbers are generated. The VHDL code is simulated in XILINX ISIM simulator. The random numbers are generated using two algorithms, they are XOR GATE algorithm and LFSR algorithm.

2.1 XOR GATE algorithm

- Start a new project in isim simulator in VHDL module. Include the IEEE library files.
- Start a new entity for the declaration of ports clk, random_num(wd-1 downto 0) where wd is an integer 32.
- After declaration of ports end the entity with entity name.
- Now start the architecture of the entity with entity name.
- Begin the conditional statements that is processing the clock signal and taking the variable rand_temp, temp.
- Then open the if conditional statement that, if clock is at rising edge this statements will processed.

```
temp := rand_temp(wd-1) xor rand_temp(wd-2);
```

```
rand_temp(wd-1 downto 1) := rand_temp(wd-2 downto 0);
```

```
rand_temp(0) := temp;
```

- Then end the if condition. Now copy the 32 bit number in random_temp to ramdo_num.
- End the process and end the entity.
- Now write the program for processing the clk signal.
- The clock period is one nano second and for half clock period the clock is '1' and another half clock period it is '0'.
- End the process.

2.2 LFSR algorithm:

- A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state.
- The only linear function of single bits is xor, thus it is a shift register whose input bit is driven by the exclusive-or (xor) of some bits of the overall shift register value.
- The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state.
- Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle.
- However, an LFSR with a well-chosen feedback function can produce a sequence of bits which appears random and which has a very long cycle.

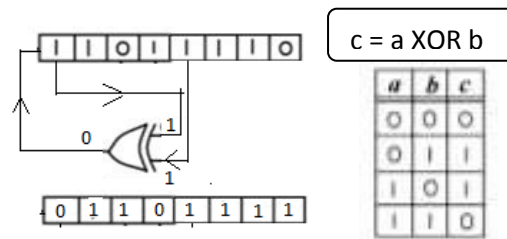


Fig 2.2(a)

Fig 2.2(b)

III. IMPLEMENTATION OF RANDOM NUMBER GENERATOR USING FPGAS

A field programmable gate array is an integrated circuit created to be configured by the user after manufacturing hence it is called field programmable. FPGAs contain programmable logic components called logic blocks. It also contains reconfigurable interconnects that allow the blocks to be connected together like a one chip programmable breadboard. These logic blocks can perform combinational functions like OR, XOR, XNOR, NAND, AND, NOR and they can function like memory blocks like flip-flops. Upload XOR GATE algorithm based random number generator VHDL code and LFSR algorithm based random number generator VHDL code written in the XILINK ISIM simulator to Spartan 3 kit.

IV. IMPLEMENTATION OF RANDOM NUMBER GENERATOR WITH ARDUINO:

The Arduino IDE software allows us to write program and upload the program to Arduino. Open the Arduino IDE software. Write the program in the editing window. Verify and compile and upload the code to arduino. We can see the output in serial monitor.

4.1 Implementing True random number generator

Algorithm:

- Open Arduino IDE software click on files and click on new to get the new editing window to write the program.
- Take a variable randomNumber in long data type.
- Then open the void setup function for declaration of ports and begin the code with baud rate 9600 and include the function randomSeed(analogRead(A0)) function to get the true random number.
- Open the loop function where we can write the conditional statements port working.
- Random(min,max) is the function which generates the random number between the minimum and maximum limit, put the random number obtained in variable randomNumber.
- Now print the random number using Serial.println(randomNumber) statement.
- Put required time delay to get the random number in that interval.
- Verify the code and upload to the arduino board. Now observe the output in serial monitor

4.2 Implementation of pseudo random number generator:

Algorithm:

- Open Arduino IDE software click on files and click on new to get the new editing window to write the program.
- Take a variable randomNumber in long data type.
- Then open the void setup function for declaration of ports and begin the code with baud rate 9600 .
- Open the loop function where we can write the conditional statements port working.
- Random(min,max) is the function which generates the random number between the minimum and maximum limit, put the random number obtained in variable randomNumber.
- Now print the random number using Serial.println(randomNumber) statement.
- Put required time delay to get the random number in that interval.
 - Verify the code and upload to the arduino board. Now observe the output in serial monitor

V. IMPLEMENTATION OF RANDOM DIGITAL LOCKER USING ARDUINO:

5.1 Algorithm:

- Upload the embedded-c code written in arduino IDE software in the Arduino mega 2560 board.
- Make the connection as per the circuit diagram fig 5.1 (b).
- Now install the digital locker mobile application in the latest android mobile.
- Now Arduino mega 2560 board will generate a new password for every 10 seconds.
- Type the password in the mobile application.
- Password typed will come to arduino through Bluetooth communication channel.
- The Arduino will compare both generated password and password from mobile application.
- If the password is correct then it displays password accepted , blue LED will glow and locker will open.
- If the password is wrong then it displays password declined, red LED will glow and buzzer will on and locker will not open.
- If password is not accessed in ten seconds then the password will collapse and it will generate a new password for every ten seconds.

5.2 Flow Chart

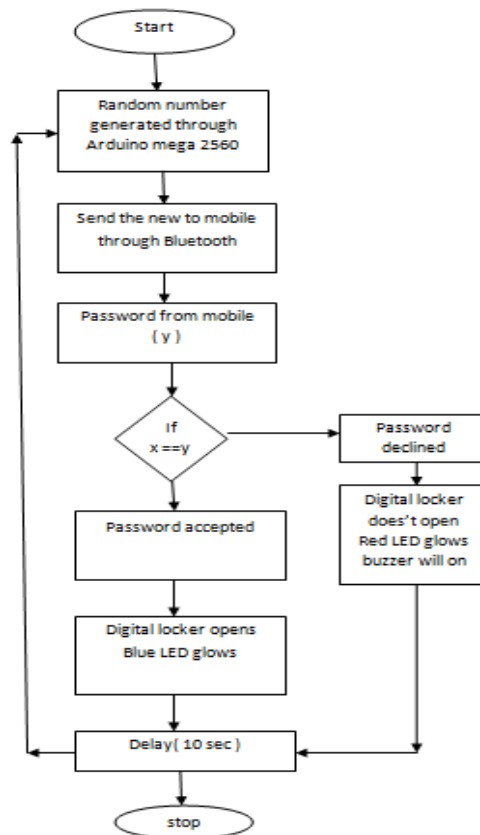


Fig 5.1(a)

5.3 Components:

1	Arduino mega 2560 board
2	HC-05 Bluetooth module
3	Digital locker
4	Buzzer
5	LEDs(Red, Blue)
6	Android mobile

5.4 Circuit

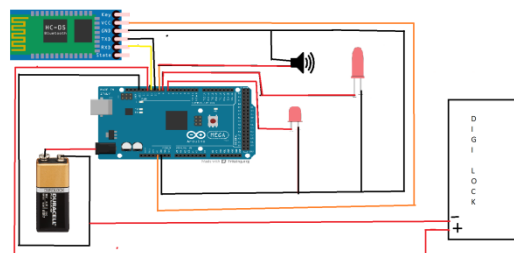


Fig 5.1(b)

VI. OUTPUTS

6.1 Simulated output of XOR algorithm:



Fig 6.1

6.2 True random generator using arduino:

6.2.1 Before reset:

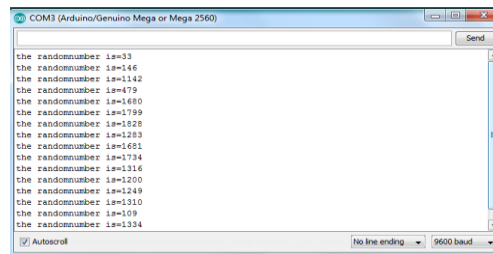


Fig 6.2.1

6.2.2 After reset:

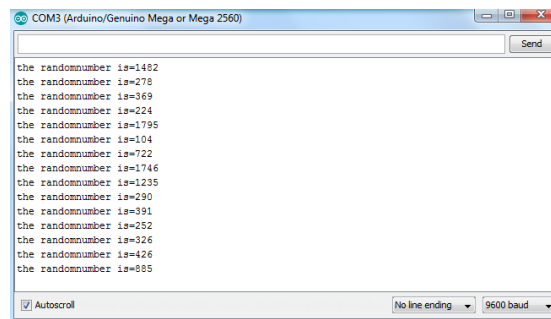


Fig 6.2.2

6.3 Pseudo random number generator:

6.3.1 Before reset:

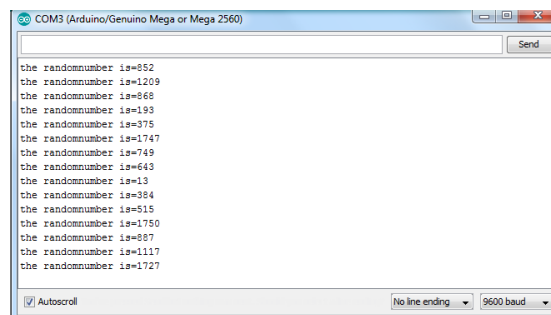
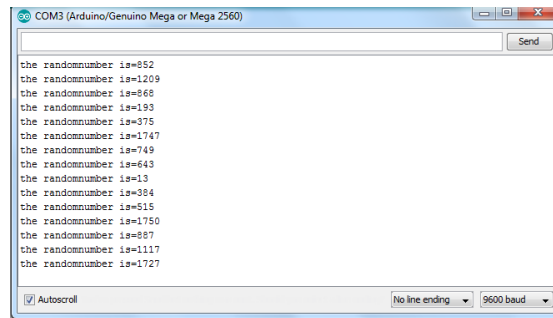


Fig 6.3.1

6.3.2 After reset:



```
COM3 (Arduino/Genuino Mega or Mega 2560)
the randomnumber is=852
the randomnumber is=1209
the randomnumber is=868
the randomnumber is=193
the randomnumber is=375
the randomnumber is=1747
the randomnumber is=749
the randomnumber is=643
the randomnumber is=13
the randomnumber is=384
the randomnumber is=515
the randomnumber is=1750
the randomnumber is=887
the randomnumber is=1117
the randomnumber is=1727
```

Fig 6.3.2

VII. CONCLUSION

In this paper we had generated random numbers using XOR , LFSR algorithms simulated in XLINX simulator and tested in FPGA Spartan-3 kit. We had also implemented both true and pseudo random number generators using Arduino. This concept is used for the implementation of high security digital lockers.

REFERENCE

- [1] Viktor Fischer and Milos Drutarovsky, "True random number generator embedded in reconfigurable hardware", Proc. of CHES 2002, LNCS 2523, pp.415 430,2003.
- [2] D. Davis, R. Ihaka, and P. Fenstermacher, "Cryptographic randomness from airturbulence in disk drivers ", Proc. ofAdvances in Cryptology Conf. (CRYPTO'94), pp. 114-120, 1994.
- [3] Thomas Jennuwein and Ulrich Achleitner, "A fast and compact quantum random number generator", American Institute of Physics, Vol. 71, No. 4, 2000.
- [4] K. H. Tsoi and K. H. Leung, "Compacted FPGAbased and Pseudo Random Generators", Proc. ofthe 1 i Annual IEEE Symposium on FieldProgrammable Custom computing Machines (FCCM'03), 2003.
- [5] Michael Epstein and Laszlo Hars, "Generator based on Digital Circuit Artifacts", Proc. ofCHES 2003, LNCS 2779, pp. 152-165, 2003. [6] PeterAlfke, "Metastable recovery in Virtex-II Pro FPGAs", Application Note: Virtex-II Pro Family, XAPP094 (v3.0) Feb. 10, 2005.
- [7] Eastlake, Corker & Schiller, "Randomness recommendations for Security", RFC 1750, Dec. 1994. [8] B. Schneier, Applied Cryptography, John Wiley & Sons, New York, 1994.
- [9] Gary Anthes. The Quest for Randomness. CACM, 54(4):13–15, 2011.
- [10] Arduino.cc. Arduino Reference Manual, 2011.
- [11] ATMEL. 8-bit Atmel microcontroller with 4/8/16/32K Bytes In-System Programmable Flash, Datasheet, 2011.
- [12] Don Davis, Ross Ihaka, Philip Fenstermacher. Cryptographic Randomness from Air Turbulence in Disk Drives. Advances in Cryptography, 839:114– 120, 1994.
- [13] Ian Goldberg and David Wagner. Randomness and the Netscape Browser. Dr. Dobbs Journal, 21:66–106, 1996