Group 9 Digital Integrated Circuits Design Level 3

Dmitriy Fedorov[1], Yerassyl Olzhabay [2], Aisultan Alimkhan[3]
[1]School of Electrical and
Electronic Engineering
Nazarbayev University
010000 Astana
Email: dmitriy.fedorov@nu.edu.kz
[2]School of Electrical and
Electronic Engineering
Nazarbayev University
010000 Astana
Email: yerassyl.olzhabay@nu.edu.kz
[3]School of Electrical and
Electronic Engineering
Nazarbayev University
010000 Astana
Email: aisultan.alimkhan@nu.edu.kz

I. Introduction

NOWADAYS , in the current century of electronics, social networks and internet, all information about personal life is increasingly stored in virtuality. Proof of your authorship, authenticity and verification is serious concern. Therefore, very strong and protected algorithms are needed. This report presents an implementation of such algorithms by using a level by level representation.

Hence, Elliptic Curve Digital Signature Algorithm, ECDSA, was obtained through those level steps

Level 1: Problem Definition: Design a digital block to create a uniform pseudo-random number sequence with a bit length of 256.

Tasks:

1. Test the digital block by generating it through HDL code.
2. Design the layout in ElectricVLSI and compute the gate level.
3. Simulate it using LTSpice and confirm design through IRSIM.

Level 2: Problem Definition: Construct a cryptographically secure pseudo-random number generator that can be utilized in the Elliptic Curve Digital Signature Algorithm using the digital block designed in level 1.

Tasks:

1. Test the digital block by generating it through HDL code.
2. Design the layout in ElectricVLSI and compute the gate level.
3. Simulate it using LTSpice and confirm design through IRSIM.

Level 3 : Problem Definition: Develop a chip that implements Elliptic curve algorithm. Level 2 block is used for initialization.

Tasks:

1. Test the digital block by generating it through HDL code.
2. Compute the gate level with FPGA simulation tool.
3. Design the full-custom chip layout in ElectricVLSI
4. Simulate it using LTSpice and confirm design through IRSIM.

Elliptic Curve : The connection of elliptic integrals in mathematics which can be used to define the length of arc of an ellipse is Elliptic curves or cubic curves.

Elliptic Curve Cryptography: The term elliptic curve in cryptography is known as the best method of current knowledge to public key cryptographically. Elliptic curve cryptography (ECC) can be used as a basis of the algebraic structure of elliptic curves over finite fields.

Elliptic curve cryptography is a corresponding discrete logarithm cryptography [1]. An equation for elliptic curve E over Zp can be defined in the Cartesian coordinate system by the following form:

$$y^2 \equiv x^3+ax+b \bmod(p)$$

Fig. 1. An Elliptic curve.

The values of a and b can be different for different curves. The point on the curve is a public key and the random number is a private key, where multiplying the generator point G in the curve with the private key gives the public key [2].

Private Parameters	Public Parameters
private key 'd'	Public key Q
Random integer 'k'	Generating point G
Elliptic curve parameter 'a,b'	Order of field 'n'
Field characteristics 'q'	Signature (r,s)
	Message hash 'z'

TABLE I
PUBLIC AND PRIVATE PARAMETERS OF ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM.

Elliptic Curve Digital Signature Algorithm : The Elliptic Curve Digital Signature Algorithm (ECDSA) has same elliptic curve analog as the Digital Signature Algorithm (DSA). It was officially recognized in1999 as an American National Standards Institute (ANSI) standard and as a National Institute of Standards and Technology (NIST) and in Institute of Electrical and Electronics Engineers (IEEE) standards in 2000 [3]. Similar pattern can be observed in elliptic-curve cryptography for bit size of the public key with the Elliptic Curve Digital Signature Algorithm that requires about twice bits of security level. In other words, an attacker will need about maximum of operations to find the private key of an ECDSA. To compare, the DSA public key at least 1024 bits are needed, whereas the ECDSA public key will require 160 bits for security level of 80 bits [4]. Conversely, the signature size for both ECDSA and DSA is same.

II. Proposed Algorithms for Each Level and Their Simulation Steps

A. For Level 1

Xorshift algoritm.: The design work flow:

Step 1: Behavioral model of xorshift64 was developed. Then four instances of xorshift64 was concatinated in order to construct 256bit version of xorshift. SplitMix64 generator was used to initialise xorshift. It works well because they have different architecture.

Step 2: Yosys software was used in order to convert behavioral model that was designed in previous step into gate level model. It can be used by electric vlsi in order to develop layout.

Step 3: Compilation the rats-nest circuit. Insert the hardware description language code, Verilog, into a cell. Through testing it, convert “current cell to rats-nest structure” command. It will generate all of the standard cells by placing them randomly and connecting with "unrouted" arcs.

Step 4: Location of the following cells. After, applying the “floorplan and place current cell” command, the rats-nest cell will be taken and redo its placement to the right place. Some preferences were set, like "run routing after placement" and "padding" around subcells[5].

Step 5: Routing of the cells with applying the “sea of gates route this cell” command. To get better results, some routing parameters were adjusted via unrouting and re-routing individual networks [6]. Gate level design was generated.

Step 6: Layout design was created in ElectricVLSI only for 64 bit xorshift because internal structure of xorshift256 is complicated and time consuming (it is NP problem) to compile highly error pron.

B. For Level 2

Xoroshiro128 algorithm - modification of xorshift: The design work flow:

Step 1 : Behavioral model of xoroshiro128 [7] was developed based on original C++ implementation. Then four instances of xoroshiro128 was concatenated in order to construct 256bit version of xoroshiro. For initialization SplitMix64 generator was used, same as for xorshift

Steps 2-6: The same as in level 1. The following results were obtained (Figure 2 and 3)

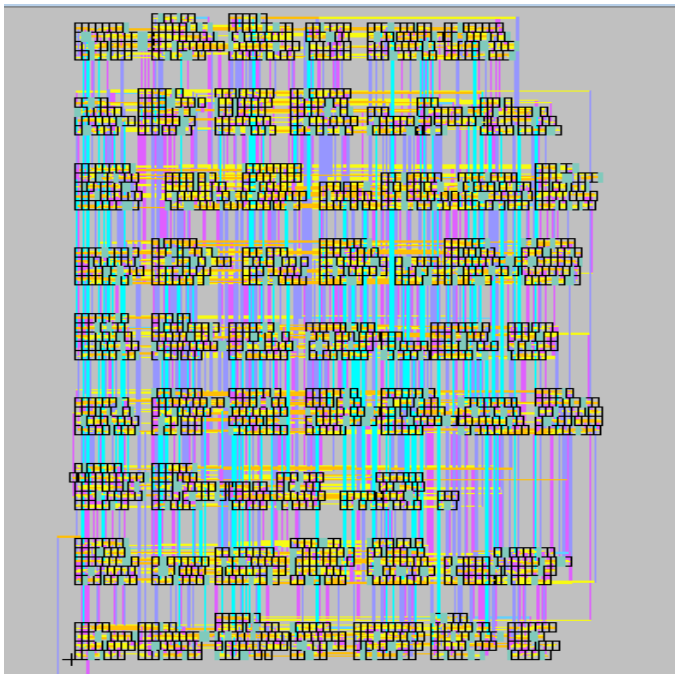

Fig. 2. Xoroshiro64 layout

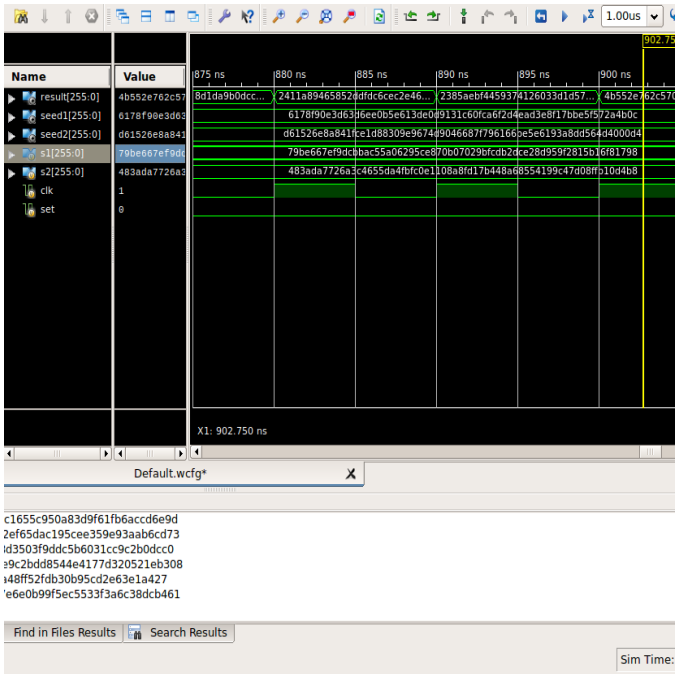

Fig. 3. Xoroshiro256 simulation

C. For Level 3

Step 1: New code was produced to implement Discrete Elliptic curve algorithm. The aim of this algorithm is to traverse through Elliptic modular field.

Step 2: Bottom up design approach was used and algorithm was divided into functional blocks, which implement some mathematical operations such as mod and inverse mod.

To compute inverse modulus Extended Euler Algorithm could be used, however this algorithm is iterative and not hardware friendly. Therefore, it was replaced with ROM module which stores precomputed values [8]. It reduced the complexity of the circuit, however some flexibility was compromised.

Step 3: The goal of top module is to implement high level math. Low level blocks with some additional code were used. As a result, the module taking clock cycles to compute next iteration was obtained.

Input: Xoroshiro.

Output: point on elliptical curve (described with 2 coordinates).

III. Discussion

Challenges faced :

- 3 layers tested first (not sufficient for avoiding wire crosses);
- Long compilation/routing time (40 min clustering and placement time, 16 hours routing algorithm).
- Inverse modulus operation requires the circuitry with high complexity (not hardware friendly)

Solutions::

- Additional layers and increased scales;
- Decreased to 64-bit xoroshiro128+ instead of 4 concatenated ones, taking only one of them for fast work;
- ROM module with pre-calculated values was added instead of inverse mod

IV. Advantages of ECDSA of ECC

At the end, some remarkable advantages of the ECC and ECDSA were observed by reading research papers, performing simulations and designing chips.

For ECC:

- Affords greater security for same number of bits ;
- Allows effective executions for cryptographic processes requiring small chips;
- Less power consumption and heat generation;
- Appropriate for low computing power, less memory and low bandwidth machines.

For ECDSA:

- Provides effective security against hackers by only sharing two points publically and generating point in a private sector.
- In signature verification method, reduces the number of point addition operation and multiplication.

V. Conclusion

In this project, various pseudo-random generators and were analyzed and Xorshift and Xoroshiro128 algorithms were constructed. Additionally, SplitMix64 algorithm was implemented in order to generate seeds. Xoroshiro128 algorithm has shown to be more robust in comparison with other algorithms. The gate level and layout of this algorithms were designed. At the end, proper working was confirmed by simulating it through ISIM. Layout for Level 3 was not completed, because of very high complexity(~800 thousand lines of code). However code itself works properly and it can be seen from simulation. If the our future work will be based on this kind of project, the obtained results will be useful to improve, design and implement a more robust, stable algorithm with less time consumption.

References

[1] J. Don, A. Menezes, and S. Vanstone, ""the elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001.

[2] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw*, vol. 33, no. 4, 2007, article 22.

[3] L. Shweta and M. Sharma, "An efficient elliptic curve digital signature algorithm (ecdsa)," *2013 International Conference on Machine Intelligence and Research Advancement*, 2013.

[4] K. Aqeel, "Implementation of elliptic curve digital signature algorithm," 2010. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.206.5193&rep=rep1&type=pdf

[5] S. M. Rubin, "Chapter 9: Tools. 9-12: Placement." Silicon Compiler. [Online]. Available: http://www.staticfreesoft.com/jmanual/mchap09-12.html

[6] S. Rubin, "Chapter 9: Tools. 9-13: Placement." Silicon Compiler. [Online]. Available: http://www.staticfreesoft.com/jmanual/mchap09-13.html#mchap09-13.html

[7] "Xoroshiro /xorshift*/xorshift generators and the prng shootout." [Online]. Available: http://vigna.di.unimi.it/xorshift/xoroshiro128plus.c

[8] GeeksforGeeks, "Modular multiplicative inverse." [Online]. Available: http://www.geeksforgeeks.org/multiplicative-inverse-under-modulo-m/