International Journal of Latest Engineering Research and Applications (IJLERA) ISSN: 2455-7137

Volume – 02, Issue – 12, December – 2017, PP – 66-72

Implementation of Advanced encryption Standard using Reversible Logic Gate

D Srinivas[1], Boda Aruna[2], Ravi Boda[3]

[1] *Department of ECE, VJIT, Hyderabad, India*

[2] *Department of ECE, MIETW, Hyderabad, India*

[2]*Department ECE, University College of Engineering, Osmania University, India*

Abstract: The Rijndael cryptosystem is a new Advanced Encryption Standard (AES) till now many hardware implementations of AES have been reported. Some of these implementations are optimized for speed, some for area, some for reconfigurability, and some for low-power applications. Again, reversible logic synthesis methodologies have drawn the attention of researchers in recent times, mainly with the prospect of quantum computing becoming a reality, and the potential of reversible logic circuits for providing ultra low-power implementations. The implementation strategy of Encryption and Decryption includes the Verilog HDL coding for each block in a synthesizable way. Test benches are to be written for each module and simulated using Xilinx simulation environment. After satisfactory functioning of each block, these blocks are combined to form the top level module. The top level modules are then tested by generating appropriate test vectors through test bench. The Verilog HDL code is compiled using Xilinx ISE 13.2. The synthesis of the architecture is carried out on VIRTEX-V devices.

Keywords: Advanced Encryption Standard, Reversible Logic Synthesis, Top level Module,Virtex-V.

I. INTRODUCTION

In today's digital world, encryption is emerging as a disintegrable part of all communication networks and information processing systems, for protecting both stored and in transit data. Encryption is the transformation of plain data (known as plaintext) into unintelligible data (known as cipher text) through an algorithm referred to as cipher. There are numerous encryption algorithms that are now commonly used in computation, but the U.S. government has adopted the Advanced Encryption Standard (AES) [1] to be used by Federal departments and agencies for protecting sensitive information. The National Institute of Standards and Technology (NIST) have published the specifications of this encryption standard in the Federal Information Processing Standards (FIPS) Publication 197. Any conventional symmetric cipher, such as AES, requires a single key for both encryption and decryption, which is independent of the plaintext and the cipher itself. It should be impractical to retrieve the plaintext solely based on the cipher text and the encryption algorithm, without knowing the encryption key. Thus, the secrecy of the encryption key is of high importance in symmetric ciphers such as AES. Software implementation of encryption algorithms does not provide ultimate secrecy of the key since the operating system, on which the encryption software runs, is always vulnerable to attacks.

There are other important drawbacks in software implementation of any encryption algorithm, including lack of CPU instructions operating on very large operands, word size mismatch on different operating systems and less parallelism in software. In addition, software implementation does not fulfill the required speed for time critical encryption applications. Thus, hardware implementation of encryption algorithms is an important alternative, since it provides ultimate secrecy of the encryption key, faster speed and more efficiency through higher levels of parallelism.

Different versions of AES algorithm exist today (AES128, AES196, and AES256) depending on the size of the encryption key. This thesis presents a hardware model for implementing the AES128 algorithm [2] using the Verilog hardware description language. A unique feature of the design proposed in this thesis is that the round keys, which are consumed during different iterations of encryption, are generated in parallel with the encryption process.

Rijndael has proposed an algorithm for network security which is clean and fast with good security margin. The rijndael algorithm was selected as the Advanced Encryption Standard (AES) in October 2000. This algorithm is tested and is vulnerable to known attacks. This is also issued as the FIPS publication in November 2001 by NIST. Since the encryption and decryption of AES algorithm are reversible [3] in nature a little effort is made to implement the AES algorithm using the reversible logic. The reversible logic gates [4] are very speed in their operation and consumes low power. They are less in cost and enables higher density. This nature of the reversible logic gates has motivated to implement the AES algorithm using the reversible logic gates.

II. LITERATURE REVIEW

An early and highly influential block cipher design is the Data Encryption Standard (DES). The DES is a cipher (a method for encrypting information) selected as an official Federal Information Processing Standard (FIPS) for the United States in 1976, and which has subsequently enjoyed widespread use internationally. The algorithm was initially controversial, with classified design elements, a relatively short key length, and suspicions about a National Security Agency (NSA) backdoor. DES [5] consequently came under intense academic scrutiny, and motivated the modern understanding of block ciphers and their cryptanalysis. DES is now considered to be insecure for many applications. This is chiefly due to the 56-bit key size being too small; DES keys have been broken in less than 24 hours. There are also some analytical results which demonstrate theoretical weaknesses in the cipher, although they are infeasible to mount in practice. The algorithm is believed to be practically secure in the form of Triple DES, although there are theoretical attacks. In recent years, the cipher has been superseded by the Advanced Encryption Standard (AES).

56-bit key is used in DES and 16 cycle of each 48-bit sub keys are formed by permuting 56-bit key. Order of sub keys is reversed when decrypting and the identical algorithm is used. Block size of 64-bit is made from L and R blocks of 32-bit.Triple DES [6] simply extends the key size of DES by applying the algorithm three times in succession with three different keys. The combined key size is thus 168 bits (3 times 56), beyond the reach of brute-force.

AES was designed after DES and 3DES. Most of the known attacks on DES were already tested on AES. It is definitely more secure than DES due to the larger-size key. There are no differential and linear attacks on AES as yet.

III. PROPOSED METHOD

The An overall goal of the project is to design and implement the AES algorithm using reversible logic gates. The detail research objectives are: First to design the AES algorithm using the reversible logic gates. The goal is to explore the speed of operation of the reversible logic AES. These timing reports are compared to the timing reports of a basic AES algorithm. Second simulate the designs in the register-transfer level (RTL-level) simulation by using the Xilinx Verilog compiler. The top modules of encryption and decryption are synthesized and the timing reports are generated. The timing report of Reversible logic AES is compared with the basic AES algorithm.

The AES algorithm is a symmetric cipher. In symmetric ciphers, a single secret key is used for both the encryption and decryption, whereas in asymmetric ciphers, there are two sets of keys known as private and public keys. The plaintext is encrypted using the public key and can only be decrypted using the private key. In addition, the AES algorithm is a block cipher as it operates on fixed-length groups of bits (blocks), whereas in stream ciphers, the plaintext bits are encrypted one at a time, and the set of transformations applied to successive bits may vary during the encryption process.

The AES algorithm operates on blocks of 128 bits, by using cipher keys with lengths of 128, 192 or 256 bits for the encryption process. Although the original Rijndael encryption algorithm [7] was capable of processing different blocks sizes as well as using several other cipher key lengths, but the NIST did not adopt these additional features in the AES. For 128 bits key, there will be 10 rounds of operation, for 192 bits key there will be 12 rounds of operation and for 256 bits key there will be 14 rounds of operation. The AES module with 128 bits plain text and 128 bits key is shown in figure 3.1 below, the output is a 128 bits cipher text

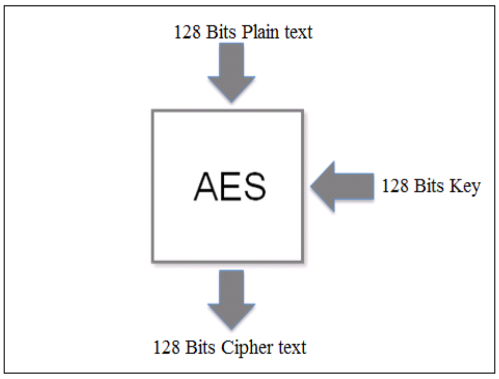


Figure 3.1: AES module

- **Inputs, Outputs and the State**

The plaintext input and cipher text output for the AES algorithms are blocks of 128 bits. The cipher key input is a sequence of 128, 192 or 256 bits. In other words the length of the cipher key, N_k, is either 4, 6 or 8 words which represent the number of columns in the cipher key. The AES algorithm is categorized into three versions based on the cipher key length. The number of rounds of encryption for each AES version depends on the cipher key size.

In the AES algorithm, the number of rounds is represented by N_r, where $N_r = 10$ when $N_k = 4$, $N_r = 12$ when $N_k = 6$, and $N_r = 14$ when $N_k = 8$. The following table illustrated the variations of the AES algorithm. For the AES algorithm the block size (N_b) [8], which represents the number of columns comprising the State is $N_b = 4$.

Table 3.1: AES Variations

AES Version	Key Length (N_k)	Block Size (N_b)	Number Of Rounds (N_r)
AES 128	4	4	10
AES 192	6	4	12
AES 256	8	4	14

The basic processing unit for the AES algorithm is a byte. As a result, the plaintext, cipher text and the cipher key are arranged and processed as arrays of bytes. For an input, an output or a cipher key denoted by a, the bytes in the resulting array are referenced as a_n, where n is in one of the following ranges:

Block length = 128 bits, 0 <= n < 16
Key length = 128 bits, 0 <= n < 16
Key length = 192 bits, 0 <= n < 24
Key length = 256 bits, 0 <= n < 24

All byte values in the AES algorithm are presented as the concatenation of their individual bit values between braces in the order {b7, b6, b5, b4, b3, b2, b1, b0}. These bytes are interpreted as finite field elements using a polynomial representation:

$$b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x+b_0x=\sum b_ix^i \qquad (3.1)$$

All the AES algorithm operations are performed on a two dimensional 4x4 array of bytes which is called the State, and any individual byte within the State is referred to as $s_{r,c}$ where letter ‘r’ represent the row and letter ‘c’ denotes the column. At the beginning of the encryption process, the State is populated with the plaintext. Then the cipher performs a set of substitutions and permutations on the State. After the cipher operations are conducted on the State, the final value of the state is copied to the cipher text output as is shown in the following figure 3.2.

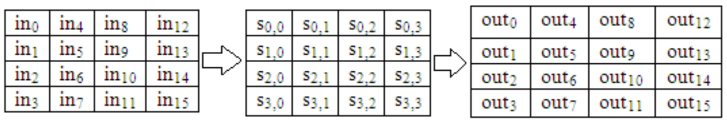

in_0 in_4 in_8 in_{12}
in_1 in_5 in_9 in_{13}
in_2 in_6 in_{10} in_{14}
in_3 in_7 in_{11} in_{15}
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
out_0 out_4 out_8 out_{12}
out_1 out_5 out_9 out_{13}
out_2 out_6 out_{10} out_{14}
out_3 out_7 out_{11} out_{15}

Figure 3.2: State Population and Results

At the beginning of the cipher, the input array is copied into the State according the following scheme:

S[r, c] = in [r + 4c] for $0 \le r < 4$ and $0 \le c < 4$

And at the end of the cipher the State is copied into the output array as shown below:

Out[r+4c] = s[r, c] for $0 \le r < 4$ and $0 \le c < 4$

- **Cipher Transformations**

The AES cipher either operates on individual bytes of the State or an entire row/column. At the start of the cipher, the input is copied into the State as described in Section 3.2. Then, an initial Round Key addition is performed on the State. Round keys are derived from the cipher key using the Key Expansion routine. The key expansion routine generates a series of round keys for each round of transformations that are performed on the State. The operation of AES top module is shown in the figure 3.3 below,

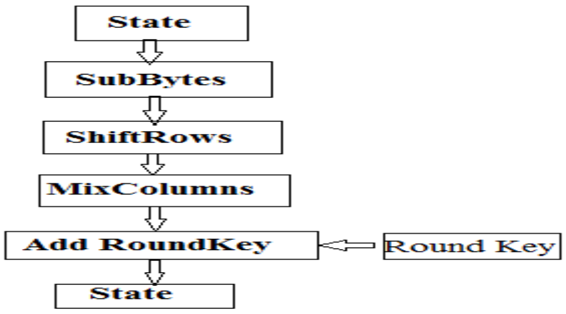


Figure 3.3: AES Block diagram

The transformations performed on the state are similar among all AES versions but the number of transformation rounds depends on the cipher key length. The final round in all AES versions differs slightly from the first $N_r - 1$ rounds as it has one less transformation performed on the State. Each round of AES cipher (except the last one) consists of all the following transformation:

- Sub Bytes ( )
- Shift Rows ( )
- Mix Columns ( )
- Add Round Key ( )

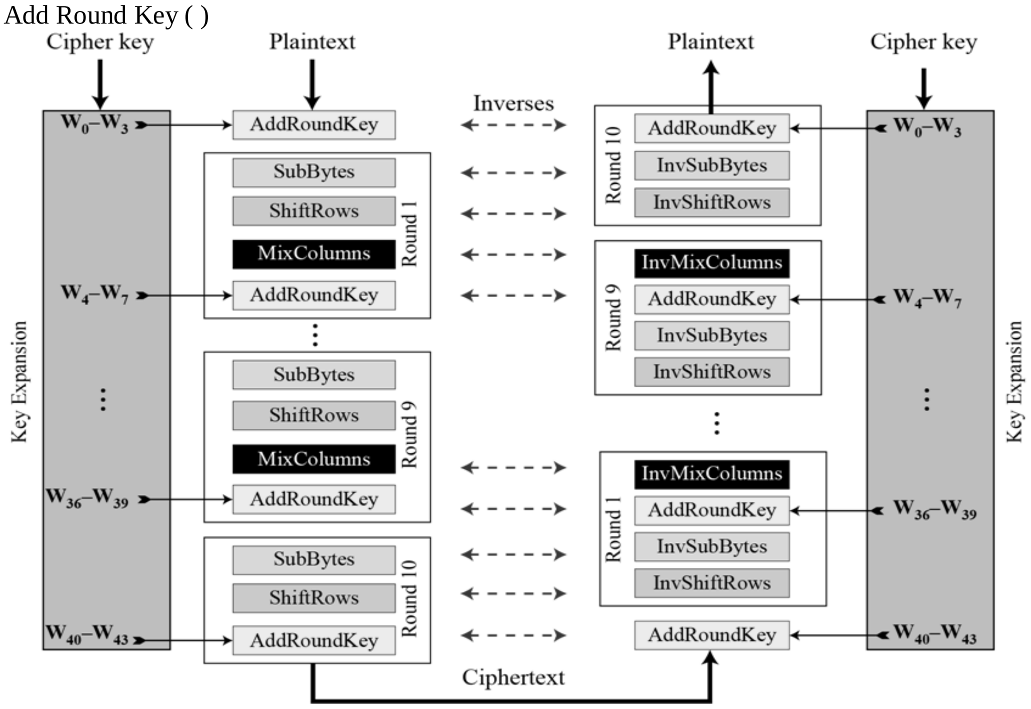


Figure 3.4: Schematic representation of AES Encryption and Decryption

The overall structure of the AES algorithm for encryption and decryption is shown in the following figure 3.8. The encryption and decryption are almost identical. Decryption follows the bottom to top approach of the encryption process with the inverted sub-blocks like Inverse Sub Bytes, Inverse Shift Rows, Inverse Mix Columns, Inverse key expansion. The tenth round key of the encryption process is taken as the cipher key for the decryption process.

- **Reversible logic implementation of AES algorithm**

In this section, a hardware model for implementing the AES128 algorithm is introduced. The model is implemented using the Reversible logic gates in System Verilog hardware description language. It gives the implementation issues of the AES128 algorithm using reversible logic. Use of reversible gates improves the speed of operation. The hardware model developed in this chapter is synthesizable. This means that the model provides a cycle-by-cycle RTL description of the circuit so that a logic synthesis tool can convert it into an optimized gate-level net list. The implementation of AES algorithm using the reversible logic [8] is almost identical to the operation of basic AES except that the outputs of every round are given to the registers. The pipelined implementation of 128 bit reversible logic AES is shown in the figure 4.3 below.

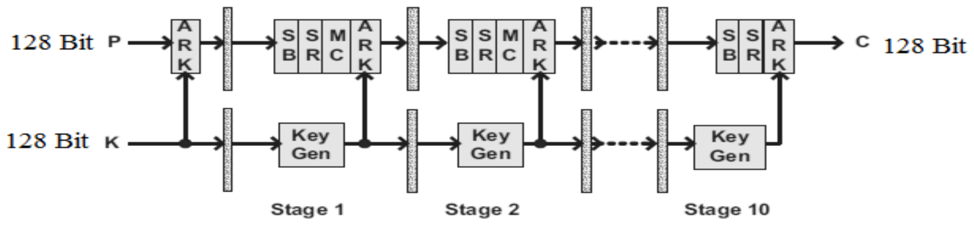


Figure 3.5: Reversible Pipelined Implementation of 128 Bit AES

The registers present in between the round operations consists of the reversible logic gates i.e.; Toffoli gates and Fredkin gates [10]. These registers process the data coming from every round and give as inputs to the successive rounds. The outputs of the Add round key and Key generators are also processed using this registers[11]. The following figure 4.4 shows the 128 bit Reversible register.

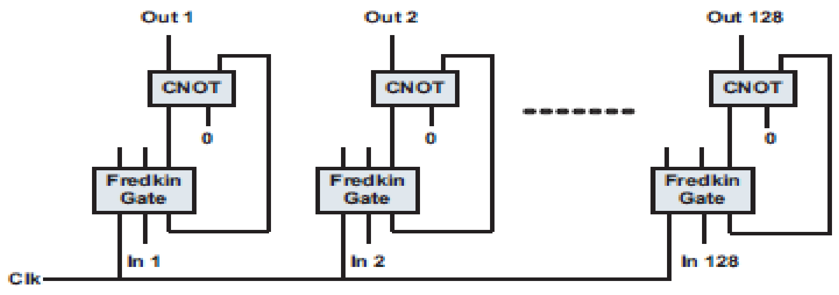


Figure 3.6: 128 Bit Reversible register

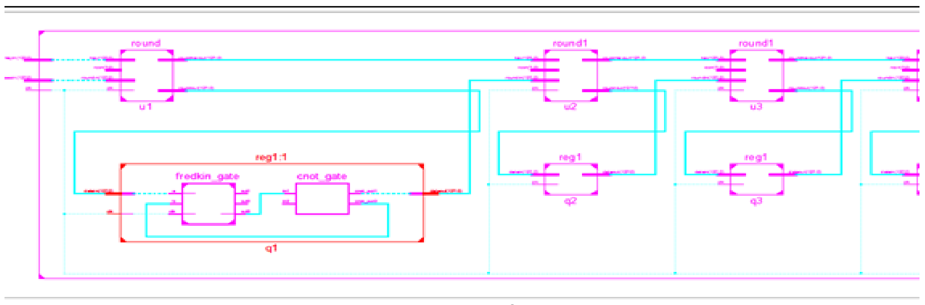


Figure 3.7: RTL schematic of encryption module

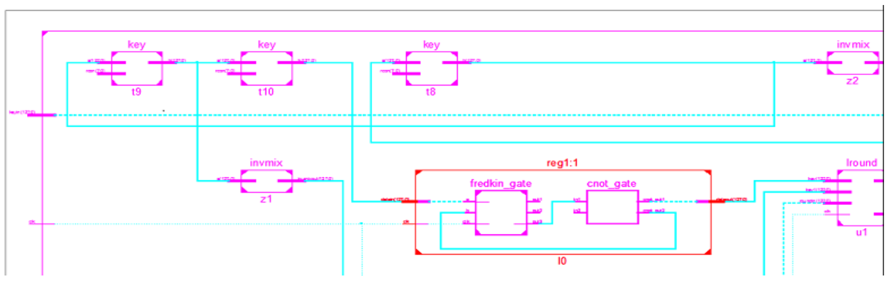
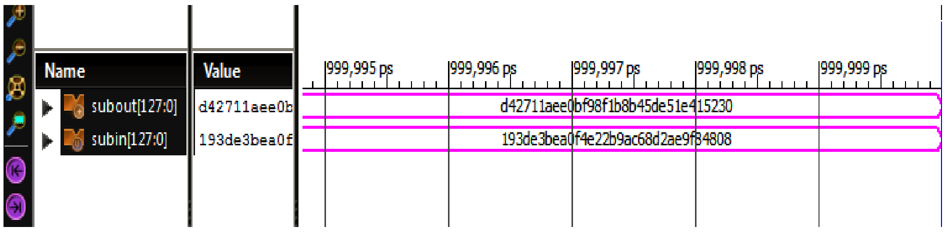


Figure 3.8: RTL schematic of decryption module

IV. RESULT AND DISCUSSION

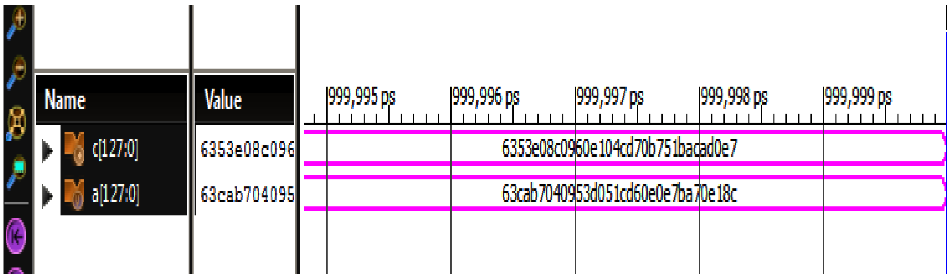
- **Simulation results of sub byte**

Sub Byte is a sub-block in the AES module which takes 128 bits as input and provides a 128 bit data output. The input state values are checked in to the S-Box [12] table and the corresponding byte states are substituted in the form of state matrix.



Name	Value	
subout[127:0]	d42711aee0b	d42711aee0bf98f1b8b45de51e415230
subin[127:0]	193de3bea0f	193de3bea0f4e22b9ac68d2ae9f84808

Figure 4.1:Sub byte simulation results



Name	Value	
c[127:0]	6353e08c096	6353e08c0960e104cd70b751bacad0e7
a[127:0]	63cab704095	63cab7040953d051cd60e0e7ba70e18c

Figure 4.2: Shift rows simulation results

Table 5.1: Comparison of Reversible logic AES with basic AES

Constraints	Basic AES	Reversible logic AES
Minimum period	4.514ns	6.711ns
Maximum Frequency	221.516MHz	149.018MHz
Minimum input arrival time before clock	6.255ns	3.231ns
Maximum output required time after clock	6.407ns	5.824ns
Maximum combinational path delay	No path found	No path found

From the above comparison results it is clear that the minimum input arrival time before clock and the maximum output required time after the clock are less for Reversible logic AES when compared with the basic AES. Hence Reversible logic AES operation is faster than the basic AES.

V. CONCLUSION

The reversible logic implementation of the 128-bit AES block cipher has been implemented. The detailed synthesis results for the encryption module and decryption module has been presented. The synthesis results are compared with the basic AES algorithm and the comparison emphasized that the reversible logic AES speed of operation is greater than that of basic AES algorithm.

REFERENCES

[1] H. Bennett. Logical reversibility of computation. Journal of IBM Research and Development, 17:525–532, 1961.

[2] R. Drechsler, A. Finder, and R. Wille. Improving ESOP-based synthesis of reversible logic using evolutionary algorithms. In Proceedings of Intl. Conference on Applications of Evolutionary Computation (Part II), pages 151–161, 2011.

[3] Grosse, R. Wille, G. W. Dueck, and R. Drechsler. Exact multiple control Toffoli network synthesis with SAT techniques. IEEE Trans. on CAD of Integrated Circuits and Systems, 28(5):703–715, May 2009.

[4] W. N. N. Hung, X. Song, G. Yang, J. Yang, and M. Perkowski. Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis. IEEE Trans. on CAD of Integrated Circuits and Systems, 25(9):1652–1663, September 2006.

[5] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In Proceedings of Advances in Cryptology (CRYPTO '99), LNCS Vol. 1666, pages 388–397, 1999.

[6] R. Landauer. Irreversibility and heat generation in computing process. Journal of IBM Research and Development, 5:183–191, 1961.

[7] M. Miller, D. Maslov, and G. W. Dueck. A transformation based algorithm for reversible logic synthesis. In Proceedings of Design Automation Conference, pages 318–323, 2003.

[8] A. Mishchenko and M. Perkowski. Fast heuristic minimization of exclusive-sums-of-products. In Proceedings of 6th Reed-Muller Work-shop, pages 242–250, 2001.

[9] N. Nayeem, L. Jamal, and H. Babu. Efficient reversible Montgomery multiplier and its applications to hardware cryptography. Journal of Computer Science, 5(1):49–56, January 2009.

[10] N. Nayeem and J. E. Rice. A shared-cube approach to ESOP-based synthesis of reversible logic. Facta Universitatis of NiE, Elec. Energ., 24(3):385–402, 2011.

[11] Rodriguez-Henriquez, N. Saqib, A. Perez, and C. Koc. Cryptographic Algorithms on Reconfigurable Hardware. Springer: Series on Signals and Communication Technology, New York, 2006.

[12] H. Thapliyal and M. Zwolinski. Reversible logic to cryptographic hardware: a new paradigm. In Proceedings of 49th Midwest Symposium on Circuits and Systems (MWSCAS '06), pages 342–346, 2006.