Journal of Innovations in Computer Science and Engineering
Shahid Beheshti Unversity

May 2023, Volume 1, Issue 1

Efficient NTT Multiplier of NTRU Prime PQC Algorithm

Reza Rashidian a Code Orcid: 0009-0008-5880-6492,
Raziyeh Salarifard a ✉ Code Orcid: 0000-0003-1323-6680,
Reyhaneh Kharazmi a Code Orcid: 0009-0008-1082-8748,
Ali Jahanian a Code Orcid: 0000-0002-0925-6120
a Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran
Emails: r.rashidian@mail.sbu.ac.ie, r_salarifard@sbu.ac.ir, re.kharazmi@mail.sbu.ac.ir, jahanian@sbu.ac.ir

ABSTRACT

As quantum computers become increasingly powerful, the threat of attacking classical algorithms will become more significant. Hence, post-quantum cryptography algorithms are effective alternatives to previous asymmetric algorithms. NTRU Prime is one of the KEM algorithms based on the attention grid in the NIST competition. Implementing such algorithms involves heavy polynomial multiplications over a ring. Number theoretic transformations allow the multiplication of polynomials to be performed in quasi-linear time O(nlog(n)). Hardware implementations of NTT multipliers are typically implemented using a butterfly structure to increase efficiency. We have proposed an efficient architecture for the NTT multiplier. We have redesigned and modified the method for using and storing the pre-processed data, this idea results in a 7% increase in frequency and a reduction of over 14% in the use of LUTs, compared to the best previous work. As a result of the reduction in delay, as well as the reduction in resources consumed, the efficiency of the process has been increased.

KEYWORDS

Post-quantum cryptography, Number Theoretic Transform, Polynomial multiplication over a ring, NTRU Prime.

1. INTRODUCTION

Due to the advancement of quantum computers and the development of algorithms such as Shore's algorithm and Grover's algorithm, many popular cryptography algorithms, including public key cryptography, symmetric, and hashing algorithms, will either be broken or require larger outputs and keys in the future [1]. This will result in the potential threat to various aspects of life, including the security of bank transactions, the privacy of online social interactions, and even the secrecy of military communications soon [2]. There will be two general approaches to be taken to combat this inevitable invasion. As a first point, cryptography relies on the principles of quantum mechanics, which cannot be broken, but requires new hardware and communications infrastructure, which are expensive. A second approach involves the use of post-quantum cryptography. Despite its insecurity, this type of cryptography can only be implemented by making software changes to existing platforms. A variety of organizations and institutes are working toward the advancement of post-quantum cryptography. Through a procedure similar to a public competition [3], the National Institute of Technology and Standards (NIST) is currently conducting a procedure to select public key cryptography algorithms. NTRU (Number Theory Research Unit) Prime Algorithm in the lattice-based algorithm category reached the final rounds of this competition. Complex and heavy calculations are often required by cryptographic algorithms. In this algorithm, a significant amount of work is related to multiplying polynomials on the ring. Hence, enhancing multipliers is critical to improving cryptographic algorithms' efficiency. In this article, Streamlined NTRU Prime is discussed, one of the two implementations of NTRU Prime.

2. NTRU PRIME ALGORITHM

As a prerequisite to designing a key-encapsulation mechanism (KEM), key generation, encapsulation, and decapsulation algorithms are required. This algorithm applies the Ringed Space calculation rules to multiplying polynomials. Recent attacks have broken several lattice-based cryptographic algorithms, and the reason for this can be attributed to the ring structure used in these algorithms. This type of threat cannot be met by the NTRU Prime cryptography algorithm because such structures are removed [4]. According to Streamlined NTRU Prime [5], polynomials over a ring is defined as follows:

$$R/q = \mathbb{Z}/q\,[x]\,/\,(x^{p} - x - 1) \qquad (1)$$

Moreover, the parameters p, q, ω should be applied in the following conditions:

$$\omega > 0\ ;\ \omega \in \mathbb{Z}\ ;\ p, q \in \mathbb{P}$$
$$2p \geq 3\omega\ ;\ q \geq 16\omega + 1$$

Sntrup761 is a variant of Streamlined NTRU Prime which is proposed to have the parameters p = 761, ω = 286, and q = 4591. In KEM, significant multiplications R/3.R/q and R/q.R/q are performed. Various methods can be used to implement this type of

Submit Date: 2023-04-18
Revise Date: 2023-06-14
Accept Date: 2023-07-02
✉ Corresponding author

multiplication, including Karatsuba, NTT (Number Theoretic Transform), and classic multiplication. A(x) and B(x) can be multiplied over this ring in the following manner:

$$A(x) \boxtimes B(x) \triangleq (A(x)B(x) \bmod q) \bmod x^{p} - x - 1 \qquad (2)$$

In terms of calculations, the greatest cost is associated with multiplying polynomials with two measures of q and x^p - x -1. A binomial multiplication requires reducing the coefficients of x to the measure of q, and placing the coefficients of k resultant from the multiplication in a range q - 1 ≥ k ≥ 0. The reduction to the q scale in NTRU Prime results in the k coefficients being divided into two parts and placed in the $\lfloor q/2 \rfloor \geq k \geq -\lfloor q/2 \rfloor$ range. At the end of the process, the resulting polynomial is also reduced by the measure of x^p - x - 1.

3. NUMBER THEORETIC TRANSFORM MULTIPLICATION

The number theoretical transform (NTT) is a generalization of the discrete Fourier transform (DFT). It uses the primitive root of unity and a ring instead of a field of complex numbers.

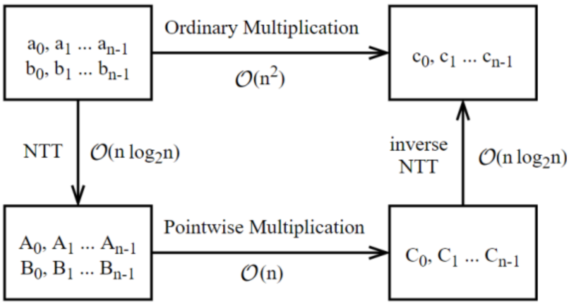

$a_0, a_1 \ldots a_{n-1}$
$b_0, b_1 \ldots b_{n-1}$
Ordinary Multiplication
$\mathcal{O}(n^2)$
$c_0, c_1 \ldots c_{n-1}$
NTT
$\mathcal{O}(n \log_2 n)$
inverse NTT
$\mathcal{O}(n \log_2 n)$
$A_0, A_1 \ldots A_{n-1}$
$B_0, B_1 \ldots B_{n-1}$
Pointwise Multiplication
$\mathcal{O}(n)$
$C_0, C_1 \ldots C_{n-1}$

Figure 1. Normal multiplication compared to NTT multiplication

According to Figure No. 1, in NTT multiplication, the NTT conversion of the polynomial coefficients must first be calculated. Afterward, multiply polynomial coefficients pointwise, and finally calculate the inverse of the NTT to obtain the desired product. As compared to other fast multiplication methods such as Karatsuba, this method is asymptotically faster and has a quasi-linear complexity equal to O (nlog(n)). The P'th root of the primary unit is referred to as ω. To calculate NTT(A[i]):

$$i = [0, p - 1] \; ; \; NTT(A[i]) = \sum^{p-1} A_j \omega^{ij} \; mod \; q \qquad (3)$$

4. BUTTERFLY STRUCTURE

For computing entity multiplication, there is a fast Fourier transform architecture called the Butterfly structure. Fast Fourier transform algorithms provide higher speed than classical matrix calculations by reusing the results of smaller and medium calculations. This structure will have a log(n) layers if the input length is n and a power of two, consisting of Cooley-Tukey base two units for entity calculation and Gentleman-Sande units for entity inverse calculation [6]. Alternatively, if necessary, other bases can also be used [7]. As shown in Figure 2, each butterfly unit has three operations: addition, multiplication, and subtraction. There is a network of these butterfly units, which play an influential role in their calculations, based on their location and layer.

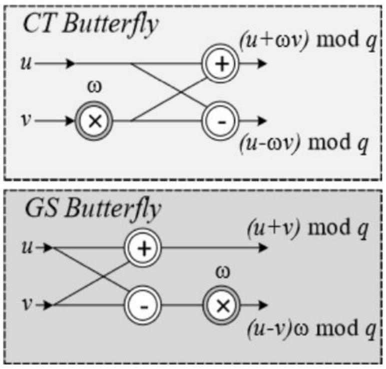


Figure 2. The structure of the Cooley-Tukey calculation units (above). The Gentleman-Sande structure (bottom)

The NTT is calculated using permutation, as well as its reverse. Essentially, this means that the order of coefficients of polynomials changes according to a pattern. The pattern must be followed when combining the results to perform point-wise multiplication. This displacement pattern can be seen in Figure 3 for a sequence of 16 coefficients. According to the figure, the left sub-branch of the tree contains odd values, while the right sub-branch contains even values from the parent node. As a final step, indices are arranged in a new manner at the lowest level of the tree.

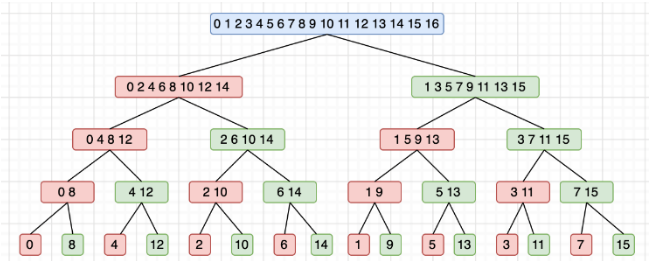


Figure 3. Permutation of indices in butterfly structure with n = 16

5. PREVIOUS WORKS

Yang, Marotske, et al. have implemented the sntrup761 algorithm using NTT multiplication, Good's technique [8], Chinese remainder theorem, butterfly architecture, and preprocessed values in their article [9].

Because NTT is usually conducted on polynomials with powers of 2, by using Good's technique and changing the variable x = y.z, equation 2 is transformed into equation 4 as follows:

$$A(x) \times B(x) = A(y.z)B(y.z) \bmod (y^3 - 1)(z^{512}-1) \qquad (4)$$

To arrive at three different butterfly units, q1, q2, and q3 are calculated using the Chinese remainder theorem at values of 15361, 12289, and 7681, respectively. Article [10] explains how to calculate $f y^j$ for each polynomial coefficient of A and B. A diagram of the multiplier's overall architecture is shown in Figure 4.

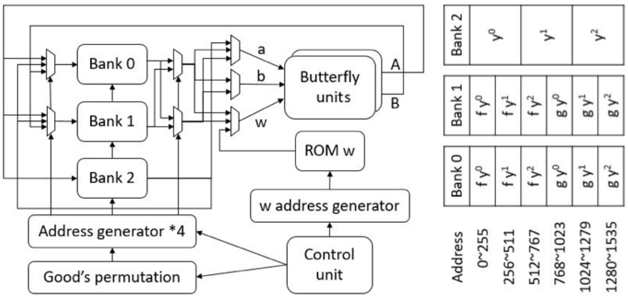


Figure 4. NTT multiplication diagram using Good's technique [6]

There are several general steps accomplished by this structure; each step is described in more detail below. Loading, NTT calculation, pointwise multiplication, reloading, Reverse NTT, application of the Chinese remainder theorem, and the reduction step. Several read-only and random-access memory structures are used in this architecture. Entity operations are accelerated by three butterfly units. In the calculation process, BRAMs are used in three data banks and other memories store powers of ω in a pre-processed form. Xilinx Zynq Ultrascale+ family FPGAs were used for this implementation, and the results will be presented in the following section.

6. IMPROVEMENT IDEAS AND CAPABILITIES

Powers of ω play a key role in the butterfly structures used to speed up NTT calculations compared to the matrix and classical calculations. This power-raised ωs have been preprocessed and stored in three separate memories. By carefully examining the architecture, two basic points were identified. First, in some separate memories, the same memory addresses are referred to in a single clock. A view of the address signals and values of the implemented memories is provided in Figure 5. It can be seen that the three memories output values with the same addresses at a specific time and clock.

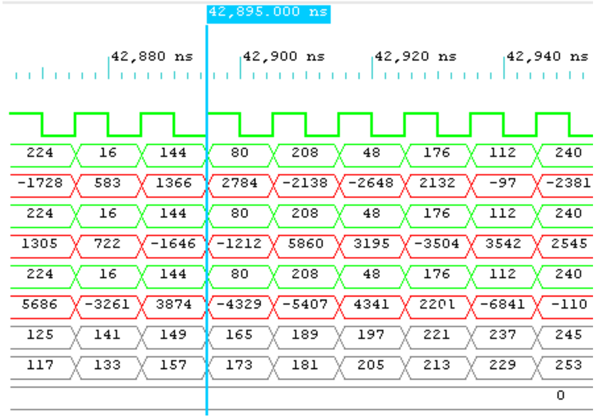


Figure 5. Memory address and output signals

Second, each memory has two types of symmetry with the default values of ω^n with n from zero to 511, which have been calculated in advance, and we have ROM[i] = - ROM[i + 256], which means the values of the i'th memory location are equal to the values of the (i+256)'th memory location with the opposite sign.

7. IMPLEMENTATION AND MODIFICATION

According to section 4, Yang et al.'s [9] implementation in Verilog language using Vivado can improve the performance of the multiplier. The pre-processed values of powers of ω used in three butterfly calculation units are stored in three separate memories. There are 512 signed 14-bit values in each memory. As the butterfly units are implemented in parallel, each of them reads and processes values from the same memory address independently. However, they all have the same clock, it is possible to combine these three separate memories into one ROM memory with a length of 42 bits which will have a common address signal.

Signed numbers are the values of ω when raised to a power. Concatenating the string of their bits, which are two's complement representations, with the input address signal, actually produces three different values corresponding to three butterfly cores. The second idea is to use the symmetry of the data pre-processed and stored in three memory modules W12289, W7681, and W15361. We found that half of the three memories had the same values as the second half. For example, in the implementation of the module W12289, which holds the pre-processed values of powers of ω equal to 3 to the measure of 12289, we have that: in the zero index row of memory, the value of ω to the power of zero to the measure of 12289 is equal to one. Also, on the 256th line of memory, the value of ω to the power of 256 to the measure of 12289 is equivalent to 12288, or -1 from the point of view of the ring. This is the reason for the symmetry observed in all three modules. As a result of eliminating symmetry to reduce occupied hardware resources, it is necessary to compensate for the removed half of the memory by making use of several control conditions and instructions. This allows the 256 last lines and deleted lines of memory to be generated and restored by using the values in the first 256 lines. Specifically, we have provided the RTL circuit of a hypothetical reduced ROM memory that has 8 lines of 6-bit memory and is capable of generating 16 lines of the appropriate value in the attached section. Also shown in Figure 6 is the simulation showing that the address bus control conditions could produce a symmetrical half-removed if appropriate control conditions were applied. CLK, Address, and output are signals.

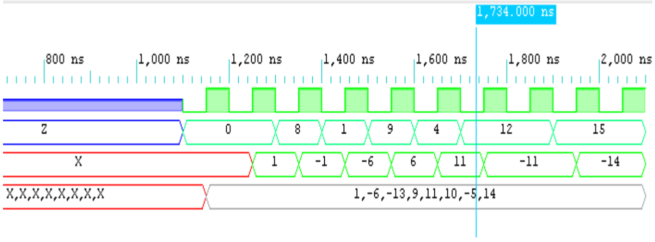


Figure 6. Simulation of the proposed ROM storing symmetric values of ω^n

RAMB36E2 block memories are available in the Zynq UlstaScale+ FPGA series, which includes various configurations with varying bit widths. Other available sources include URAM, DSP, and LUT. As a result of applying the relevant commands and commands on different platforms, different memories can be formed, and this will determine whether the memory is distributed or integrated. It is also critical to determine whether their output is stable or non-stable, since this affects the memory's timing and whether it is single-cycle or double-cycle memory. In the next section, the results of our work are compared to the most successful previous implementation.

8. COMPARISON OF IMPLEMENTATION RESULTS

As mentioned in the previous sections, we proposed two general ideas. The first idea is removing the symmetry of internal values within Rom memories, and the second idea is modifying and merging memories that are used for the same application, but separated from one another. We have found that by removing symmetry from the preprocessed values stored in the memory of the FPGA and ZUS+ chip, we have reduced the LUT resources by 8% and the LUT RAM by 19% in comparison to the previous successful implementation. The number of other hardware resources used such as DSPs remained almost unchanged. On one hand, the frequency is equal to 5.288 MHz, which represents a 53.2% drop in frequency and an increase in circuits. We can infer that this increase was caused by both the control condition and the execution instructions applied simultaneously to the memory at the same time as the rising edge of the clock.

Additionally, by modifying the structure and merging the identical memories into a single clock in Yang et al.'s work after synthesis and implementation, our results seem to be superior to those obtained in previous implementations [9]. This represents an incredibly significant improvement in terms of time and area, as well as overall efficiency. Earlier a Karatsuba method implementation has also been performed. The results are presented in Table 1. These are the results of a Verilog implementation report using Vivado 2019 software.

TABLE 1. IMPLEMENTATION RESULTS AND COMPARISON WITH PREVIOUS WORKS

	Our best work	Our other work	[9]	[3]
Frequency(MHz)	317.965	288.5	296.29	279
LUT	1616	1743	1883	6240
LUT Ram	34	34	42	-
FF	1347	1380	1341	6223
BRAM	7.5	7.5	7.5	9
DSP	3	3	3	3

Based on the results obtained, our work reduces the circuit's delay. It results in an increase in the frequency of 7% and a decrease in the area and hardware resources used in FPGAs. This means that over 14% of LUTs are no longer occupied. As a result of the reduction in delay, as well as the reduction in resources consumed, the efficiency of the process has increased. Power consumption has decreased despite the increase in frequency and efficiency. Based on our knowledge, our implementation is the most efficient implementation up to this paper.

9. SUGGESTIONS AND FUTURE WORK

As described in earlier sections, half of the data in ROM memories are comprised of repeated values, with the opposite sign. Thus, if a control structure is created properly, and by checking whether the input address is from the first half of memory or the last half, the values of the memory lines from 256th to 511th are derived from the values of the 0th to 255th lines, then memory can be generated and sent to the output. In general, we should see how this idea can be applied in a way that doesn't greatly delay the process. The two ideas of reducing memory space by removing symmetry and consolidating separate memories were implemented and compared separately. Considering optimal resource consumption, combining and applying both ideas seem to be an acceptable implementation.

As another suggestion, we propose using an optimal structure in terms of the limitations and resources in the propeller units. In our implementation, there is the multiplication of 19 x 33-bit signed numbers in each butterfly unit, which may be an example of an optimization. A second input to the butterfly unit is provided in this section, which is handled by the DSP by default. On the other hand, Karatsuba's multiplication method does not provide optimal efficiency since it requires the measurement of two numbers using a smaller expansion procedure.

In addition, this cryptographic algorithm will be implemented with our well-known and implemented multiplier on the RISC processor platform with ARM architecture such as ARM Cortex-M.

Other suggestions for future works include making reducers more efficient; research is being conducted on finding the most suitable parameters to enhance efficiency and meet security requirements, and investigating the type and structure of resources available in FPGA for more optimal storage of pre-processing data.

10. CONCLUSION

An overview of the concepts and methods used in implementing the efficient multiplication used in the NTRU Prime post-quantum cryptography algorithm, which is a lattice-based algorithm, is presented in this article. As well, we developed our suggestions to increase the efficiency of the best implementation before this article, and by comparing the results, it was determined that there had been a 7% improvement in frequency and an overall reduction of 14% in LUT resources. There was also a proposal for future work presented in section VIII.

11. REFERENCES

[1] L. e. a. Chen, "Report on post-quantum cryptography," US Department of Commerce, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2016.

[2] N. E. a. F. M. Dosti Mutlaq, "Strategic planning of post quantum cryptography to encounter quantum computing effects on cryptography," *Command and Control Quarterly,* 4 4 1400.

[3] G. e. a. Alagic, "Status report on the second round of the NIST post-quantum cryptography standardization process.," US Department of Commerce, NIST.

[4] C. C. L. T. V.-. V. C. Bernstein D J, "NTRU prime: reducing attack surface at low cost," in *Proceedings of the 24th International Conference on Selected Areas in Cryptography*, Ottawa, Canada, 2017.

[5] A. MAROTZKE, "A constant time full hardware implementation of streamlined NTRU prime.," in *International Conference on Smart Card Research and Advanced Applications*, 2020.

[6] K. Keersmaekers, "A Compact and Flexible Hardware Accelerator for NTRU Polynomial Multiplication using NTT," 2021.

[7] E. e. a. Alkim, "Polynomial multiplication in NTRU Prime: Comparison of optimization strategies on Cortex- M4," *Cryptology ePrint Archive,* 2020.

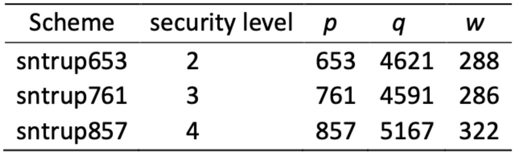
[8] I. Good, "Random motion on a finite abelian group," *Mathematical Proceedings of the Cambridge Philosophical Society,* vol. 47, no. pp. 756–762 Cambridge University Press, 1951.

[9] B.-Y. e. a. Peng, "Streamlined NTRU Prime on FPGA," *Cryptology ePrint Archive,* 2021.

[10] G. e. a. Alagic, "Status report on the second round of the NIST post-quantum cryptography standardization process," *US Department of Commerce, NIST,* 2020.

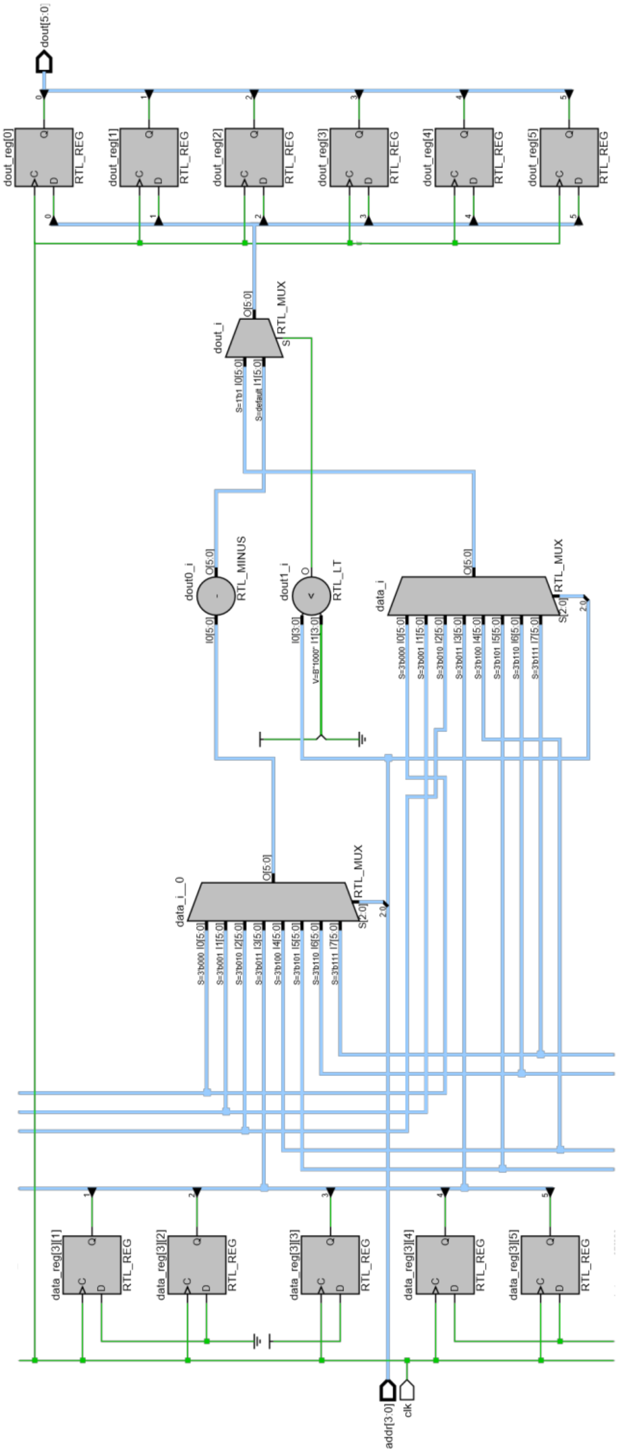
12. APPENDICES

- Sntrup set of parameters and security levels



Scheme	security level	*p*	*q*	*w*
sntrup653	2	653	4621	288
sntrup761	3	761	4591	286
sntrup857	4	857	5167	322

- The source code of our implementation is published in the GitHub repository https://github.com/hamidrezarashidian/Polynomial-Multiplication-Used-in-NTRU-Prime
- The RTL circuit of a reduced ROM using the data symmetry elimination method



Reza Rashidian
Master of Computer Architecture Engineering; Graduated from Shahid Beheshti University. Interested in hardware implementation of encryption algorithms on FPGA platform; Designing and programming web and Android applications; Senior in Django web framework; Data analysis and modeling in machine learning. Expert in building intelligentation with PLC and industrial automation.
ORCID: 0009-0008-5880-6492

Raziyeh Salarifard received the B.Sc. degree from the Sharif University of Technology, Tehran, Iran, in 2012, the M.Sc. and Ph.D. degrees from the same university, in 2014 and 2019 respectively, all in computer engineering (hardware). She is now working as an Assistant Professor at Shahid Beheshti University. Her research interests include hardware security, cryptographic engineering, and secure, efficient computing and architectures
ORCID: 0000-0003-1323-6680

Reihaneh Kharazmi
Student of Master degree of Data Science and Engineering at Politecnico di Torino, Italy. Bachelor of Computer Engineering; Graduated from Shahid Beheshti University, Iran. Interested in Data analysis, Machine Learning and Cryptography.
ORCID: 0009-0008-1082-8748

Ali Jahanian received the B.Sc. degree in Computer Engineering from
University of Tehran, Tehran, Iran in 1996 and the M.Sc. and Ph.D. degrees in Computer Engineering at Amirkabir University of Technology, Tehran, Iran in 1998 and 2008, respectively. He joined Shahid Beheshti University, Tehran, Iran in 2008. His current research interests are focused on Hardware security and Biochip design.
ORCID: 0000-0003-2292-4135