In this paper, we consider a schematic solution of the pipeline multiplier modulo, where multiplication begins with the analysis of the lowest order of the polynomial multiplier, which can serve as an operating unit for high-speed encryption and decryption of data by hardware implementation of cryptosystems based on a non-positional polynomial notation. The functional diagram of the pipeline and the structure of its logical blocks, as well as an example of performing the operation of multiplying polynomials modulo, are given. The correct functioning of the developed circuit was checked by modeling in the Vivado Design Suite computer-aided design for the implementation of the multiplication device on the development/evaluation kit Artix-7 based on the Spartan 6 field-programmable gate array series by Xilinx. The effectiveness of the proposed hardware pipeline multiplier in modulo is confirmed by the Verilog Testbench timing diagram implemented for the evaluation kit Artix-7 field-programmable gate array. In addition, the developed pipelined modulo multiplier takes no more than 0.02 % of the resources of the used field-programmable gate array for a given length of input data. Compared to the matrix multiplication method, a pipelined modulo multiplier can handle a large data stream without waiting for the result of the previous multiplication step. The modulo pipelined multiplier depth depends on the bit width of the input data. The developed pipeline device can be used in digital computing devices operating in a polynomial system of residue classes, as well as for high-speed data encryption in blocks of cipher processors operating on the basis of a non-positional polynomial number system

Keywords: cryptography, polynomial system of residue classes, pipeline multiplier modulo, FPGA

UDC 519.6:004.056.55
DOI: 10.15587/1729-4061.2022.251913

DEVELOPMENT OF PIPELINED POLYNOMIAL MULTIPLIER MODULO IRREDUCIBLE POLYNOMIALS FOR CRYPTOSYSTEMS

Sakhybay Tynymbayev
PhD, Professor
Department of Information Security Systems
Almaty University of Power Engineering and Telecommunication
Baytursynuli str., 126/1,
Almaty, Republic of Kazakhstan, 050013
Margulan Ibraimov
Corresponding author
Doctor PhD, Associate Professor*
E-mail: Margulan.Ibraimov@kaznu.edu.kz
Timur Namazbayev
Master, Senior Lecturer, Engineer*
Research Center «KazAlfaTech LTD»
Karasu str., 41a, Almaty, Republic of Kazakhstan, 050009
Sergiy Gnatyuk
Doctor of Technical Sciences, Professor
Department of Computer Science
Yessenov University
Microdistrict 32, Aktau, Republic of Kazakhstan, 130000
*Department of Solid State Physics and Nonlinear Physics
Al-Farabi Kazakh National University
Al-Farabi ave., 71, Almaty, Republic of Kazakhstan, 050040

Received date 23.11.2021
Accepted date 25.01.2022
Published date 25.02.2022

How to Cite: *Tynymbayev, S., Ibraimov, M., Namazbayev, T., Gnatyuk, S. (2022). Development of pipelined polynomial multiplier modulo irreducible polynomials for cryptosystems. Eastern-European Journal of Enterprise Technologies, 1 (4 (115)), 37–43. doi: https://doi.org/10.15587/1729-4061.2022.251913*

1. Introduction

Modern cryptography requires a search for more efficient calculation methods to improve the performance of the final encryption device [1–5]. A high-speed device for encrypting and decrypting data can be built on the basis of non-positional polynomial number systems (NPNS).

This is due to the fact that the NPNS belongs to the system of residue classes [6, 7], where the number A is represented as consecutive residues or deductions obtained by dividing the number A by the given positive integers $p1$, $p2$,..., pn, which are called the bases of the system. The main advantage of such a number system is the absence of transfers between residues when performing various arithmetic operations on the same-named residues on their base. This will make it possible to process data for each of the bases in parallel, which significantly speeds up the calculation process.

Therefore, research devoted to the development of pipelined polynomial multiplier modulo irreducible polynomials is relevant for improving the performance of cryptosystems in comparison with the existing above computation methods.

2. Literature review and problem statement

The paper [8] presents the results of coprocessor architecture for cryptography, where they use parallel multiplication to optimize and reduce the overall cycle count. The Saber engine was chosen to encapsulate lattice-based keys as an example. It is shown that multiplication of polynomials is an integral

part in Saber. However, because Saber uses power-of-two modules, it cannot simply use the Fast Asymptotic Number Theoretic Transform (NTT) based on polynomial multiplication. This leads to some changes when it is implemented for a lattice-based system. The authors in [9] show options for reducing the number of multiplication operations using the arithmetic of finite fields. The disadvantage of this method is its great complexity due to the large number of commands required to calculate the product of polynomials. Accordingly, in [8, 9] it is necessary to increase productivity; for this, when multiplying big data, it is necessary to take into account the number of operations performed per cycle. Additionally, if the residues and bases of the system are represented in a positional number system, then when performing arithmetic operations on the same-named residues, internal inter-digit transfers will be generated, which slows down the calculation process. In particular, in the paper [10] a digital signature scheme with error detection and correction was proposed, as well as the paper [11] considers the use of non-positional polynomial number systems (NPNS) to create a digital signature algorithm (EDS) and cryptographic methods. The works [10, 11] contain algorithms for the operation of polynomial multiplication, which are used in this paper. But the options to accelerate polynomial multiplication due to the organization of the pipelined or matrix method are not presented. In addition, modification of the encryption algorithm is required in order to be used in modern hardware encryption systems.

In [12], EDS module algorithms for block data encryption (based on exponentiation transform) were considered, as well as in [13], the development using NPNS was considered. In [12, 13], only software implementations of encryption and its testing with various test data are presented. As known, cryptosystems based on NPNS can be implemented in software, hardware-software and hardware. The main advantage of hardware and software implementation is high speed. Accordingly, to increase the speed of the encryption algorithm and the multiplication operations used in it, it needs to be implemented in hardware.

The central block of the cryptosystem in hardware implementations is the multipliers of polynomials modulo irreducible polynomials. The paper [14] presents a hardware implementation of the block for multiplying polynomials modulo irreducible polynomials in which the time of multiplying polynomials is directly proportional to the length of the multiplier. Nevertheless, the multiplication of polynomials occurs sequentially, which leads to delays while waiting for a response from the previous cycle. The paper [15] also presents a hardware implementation of the block for multiplying polynomials modulo irreducible polynomials. However, in this work, the multiplication of polynomials is accelerated relative to the work [14] due to the organization of the multiplying blocks in a matrix way. However, even here delays appear due to the fact; that the block has to wait for a response from the previous multiplication block.

All of the above papers do not consider the acceleration of multiplication of polynomials with a pipelined structure for processing an encrypted data stream.

3. The aim and objectives of the study

The aim of the study is to develop software and hardware that perform pipeline multiplier modulo in a non-positional notation. This will increase the efficiency of the implementation of algorithms for cryptographic protection of information.

To achieve the aim, the following objectives were set:

– to develop multipliers modulo with a pipelined organization, on the basis of which encryption and decryption of data, as well as comparison with sequential, parallel actions and analysis of advantages, are carried out;

– to develop and test an encryption device using FPGA based on non-positional polynomial notation.

4. Materials and methods

In the pipeline organization, the entire process of multiplying polynomials modulo is divided into a sequence of completed steps [16]. Each of the steps of the polynomial multiplication procedure is performed on its own stage of the pipeline and these stages work simultaneously. The results obtained at the i-th stage are transmitted to the $(i+1)$-th stage for further processing. The transfer of information from stage to stage occurs through buffer memory, which is placed between them.

The stage that has completed its operation stores the result in buffer memory and can proceed to perform the next portion of the operation data, while the next stage of the pipeline uses the data stored in buffer memory at its input as the initial data. The synchronicity of the pipeline stages is provided by clock signals, the period of which is determined by the slowest stage of the pipeline and the delay in the buffer memory element.

In a pipelined multiplier, N-stage multipliable data can be fed to the input with an interval N times smaller than in the case without pipelined multiplication. At the same pace, the results appear in the output.

The correct functioning of the developed circuit was checked by modeling in the Vivado Design Suite CAD for the implementation of the device on the Artix-7 FPGA series from Xilinx [17]. At the same time, the Verilog hardware description language was chosen for the device description [18].

5. Results of research on pipelined polynomial multiplier modulo irreducible polynomials

5. 1. Results of the development of a multiplier scheme modulo with a pipelined organization

In this paper, an N-stage pipeline scheme has been developed for multiplying the polynomials $A(x)$ and $B(x)$ modulo the irreducible polynomial $P(x)$, where the multiplication starts with a low-order analysis (Fig. 1). In the multiplication process, in each multiplication cycle, the coefficients of the polynomials $B^1(x)$, $B^2(x)$,..., $B^k(x)$ are taken into the N-bit input register Rg$B(x)$. The binary coefficients of irreducible polynomial modules $P^1(x)$, $P^2(x)$,..., $P^k(x)$ are taken into the input register Rg$P(x)$. The binary coefficients of the polynomials $A^1(x)$, $A^2(x)$,..., $A^k(x)$ in each clock cycle are taken into the multiplicative register Rg$A(x)$.

The first stage of the pipeline consists of a circuit block AND1, a partial remainder former PRF.1, registers Rg(x).1, Rgr_0 and RgR_0.1, which are buffer registers of the I-stage of the pipeline. The control input of the circuit block AND1 is connected to the low-order bit Rg$B(x)$, where the low-order bit of the binary coefficients of the multiplier polynomials $B(x)$ is stored. The information inputs of the circuit block AND1 are connected to the outputs of the register Rg$A(x)$ from where the binary coefficients of the polynomials $A^1(x)$, $A^2(x)$, ..., $A^k(x)$ are received as clock signals. The doubled

values of the binary coefficients of the polynomials $2A^1(x)$, $2A^2(x)$,..., $2A^k(x)$ are associated with the inputs of the partial remainder former PRF.1. The other inputs to the PRF.1 are connected to the outputs of the register Rg$P(x)$. The outputs of the Rg$B(x)$ register are also connected to the Rg$B(x)$.1 register. The outputs of PRF.1 are associated with the Rgr_1 register. The outputs of the Rg$P(x)$ register are also connected to the inputs of the Rg$P(x)$.1 register. The outputs of the AND1 circuit block are connected to the RgR_0 register of the first stage.

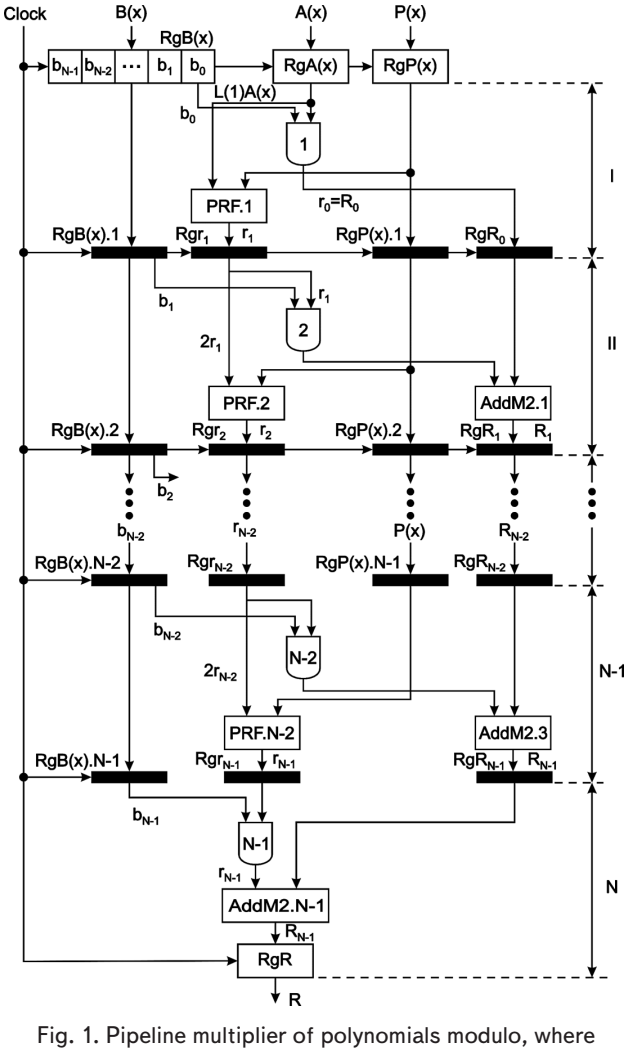

Fig. 1. Pipeline multiplier of polynomials modulo, where multiplication begins with the analysis of the lowest bits of the multiplier polynomial

The second stage of the pipeline contains blocks AND2, PRF.2 and the adder modulo two (AddM2.1), the buffer registers of the II-nd stage are Rg$B(x)$.2, Rgr_2, Rg$P(x)$.2, RgR_1.

The control inputs of the circuit block AND2 are connected to the lowest bit of the register Rg$B(x)$.1, and the information inputs of the circuit block AND2 are connected to the output Rgr1.

The first inputs of PRF.2 are also connected to the outputs of the Rgr_1 register. The second inputs of PRF.2 are associated with the register Rg$P(x)$.1. The outputs of the AND2 circuit block are connected to the first inputs of AddM2.1. The second inputs of AddM2.1 are connected to the outputs of the RgR_0 register. The outputs of the register Rg$B(x)$.1 are associated with the inputs of the buffer register Rg$B(x)$.2, and the outputs of PRF.2 are connected to the buffer register Rgr_2, the outputs of the buffer register of the I-stage Rg$P(x)$.1 are connected to the buffer register Rg$P(x)$.2 of the second stage. The outputs of the adder modulo two AddM2.1 are connected to the outputs of the RgR_1 register of the second stage.

Similar blocks AND, PRF, AddM2 and links are available in other stages of the pipeline. The exception is the N-stage, where there is a block of logic circuits AND.N-1 and AddM2.N-1 and the buffer register of the N-stage RgR.

The control input of the circuit block AND_{N-1} is supplied from the register Rg$B(x).N$–1, the highest digit of the binary coefficient of the multiplier $B(x) - b_{N-1}$, a partial remainder r_{N-1} is fed to the information input from the output of the register RgR_{N-1}.

The AND_{N-2} outputs are connected to the first inputs of AddM2.N–1. The second inputs are connected to the outputs of the buffer register RgR_{N-2} N–1 stage. The AddM2.N–1 outputs are linked to the N-stage buffer register RgR.

The logical blocks of the pipeline stages are partial remainder formers PRF and adders modulo two AddM2. The functional diagram of the PRF is shown in Fig. 2, which serves to form a partial remainder r_i-th doubled value from the previous partial remainder r_{i-1} modulo $P(x)$. The double value of the previous remainder $2r_{i-1}$ is obtained by shifting r_{i-1} towards the highest by one digit. The PRF includes an N-bit adder modulo two AddM2 and an MS multiplexer.

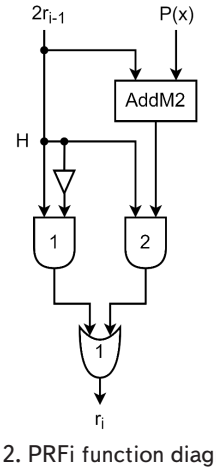

Fig. 2. PRFi function diagram

In this circuit, if $2r_{i-1i} \geq P(x)$, then the highest digit (H) of the 2_{r-1} code takes the value «1», which allows the AND2 circuit block to transmit the result of performing operations $r_i = 2r_{i-1}$ to the PRF outputs.

If the highest bit of the code $2r_{i-1}$ takes the value «0», i.e. $H=0$, then this indicates that $2r_{i-1} < P(x)$. In this case, the AND1 circuit block outputs the values $2r_{i-1}$. In this case, $2r_{i-1} = r_i \oplus P(x)$.

The adder modulo two AddM2 consists of an N-bit adder modulo two. At the inputs of which the values of partial residues r_i and the values of intermediate residues R_{i-1} are fed and the value is formed at the outputs:

$$R_i = r_i \oplus R_{i-1}.$$

Consider the operation of the pipeline. Parallel transfer of the first three polynomials $B^1(x)$, $A^1(x)$ and $P^1(x)$ to the inputs of the buffer registers of the first stage is carried out by the first clock pulse (Clock.1). During the action of this pulse, the control input of the circuit block AND1 is fed with the value of the lower bit b_0 of the polynomial $B^1(x)$, and the

information inputs AND1 are fed with the binary coefficients of the polynomial $A^1(x)$ from the register $A(x)$ and at $b_0=1$, an intermediate residue $R_0=r_0$ is formed at its outputs, which is written to the register RgR_0. By the same pulse, the binary coefficients of the $A^1(x)$ polynomial with a shift of one digit towards the higher digits are transmitted to the first inputs PRF.1, and the binary coefficients of the module $P^1(x)$) are transmitted to its second inputs. At the output of PRF.1, partial residue r_1 is formed, which is stored in the register Rgr_1. Simultaneously, by the Clock.1 pulse, the binary coefficients of the polynomials $B^1(x)$ without bit b_0 and the binary bits of the polynomial $P^1(x)$ are transferred to the buffer register Rg$B(x)$.1, and the contents of the register Rg$P(x)$ will change to Rg$P(x)$.1 of the first stage.

On the trailing edge of the Clock.1 pulse, the second triple of polynomials $B^2(x)$, $A^2(x)$ and $P^2(x)$ is taken into the input registers of the pipeline.

After the second Clock.2 pulse on the leading edge, the contents of the buffer registers of the first stage are transferred to the buffer registers of the second stage. In this case, the unit circuits AND2 b_1r_1 operation is performed and the result is transmitted to the first inputs of the adder modulo two AddM2.1, and the second inputs receive the contents of RgR_0–R_0 and at the output AddM2.1, the intermediate value of the residue R_1 is formed, which will be written to the buffer register RgR_1 of the second stage. The second clock pulse is Clock.2 from the output of the register Rgr_1 with a shift of one digit in the direction of the highest, the value $2r_i$ is transmitted to the first inputs PRF.2, and on the second inputs PRF.2, the value of the module $P(x)$ is also transmitted, at the outputs PRF.2 a partial remainder r_2 is formed, which is stored in the buffer register of the 2-stage pipeline Rgr_2. To the buffer register Rg$B(x)$.2, the contents from Rg$B(x)$.1 without bit b_1 are transferred, and to the register Rg$P(x)$.2, the contents of Rg$P(x)$.1 are transferred. On the trailing edge of the Clock.2 pulse, the third triple of polynomials $B^3(x)$, $A^3(x)$, and $P^3(x)$ is taken to the input registers of the pipeline. Polynomials $B^2(x)$, $A^2(x)$ and $P^2(x)$ along the leading edge of Clock.2 are processed on the logic blocks of the first stage and as a result, partial residues r_1 and R_0 are formed according to the binary coefficients of the polynomials of the second triple, which are written to the buffer registers of the first stage.

After the third Clock.3 pulse is applied in the buffer registers Rgr_3 and RgR_2 of the fifth stage, the residues r_3 and R_2 of the first three polynomials are stored. The buffer registers of the second stage, Rgr_2 and RgR_1, store the residues r_2 and R_1 of the second triple of polynomials, and the buffer register of the first stage stores r_1 and R_0 of the third triple of polynomials and on the trailing edge of Clock.3, the fourth triple of polynomials $B^4(x)$, $A^4(x)$ and $P^4(x)$ is taken to the input registers of the device.

After the Clock.4 pulse is applied fed to the pipeline, the fourth triple $B^4(x)$, $A^4(x)$ and $P^4(x)$ is processed by the logical blocks of the first stage, the third three polynomials $B^3(x)$, $A^3(x)$, $P^3(x)$ are processed in the logical block II stage, the second three polynomials $B^2(x)$, $A^2(x)$ and $P^2(x)$ are processed in the logical block III-stage pipeline, the first three polynomials $B^1(x)$ $A^1(x)$ and $P^1(x)$ are processed in logical units IV-stage pipeline.

After the Clock.N pulse is applied, from the register Rg$B(x).N$–1, the value of the b_{N-1} bit is fed to the control input of the AND.N–1 circuit block, and the r_{N-1} value is fed to its control inputs from the buffer register Rgr_{N-1} of the N–1 stage. The result from the outputs of the circuit block N_{N-1} is fed to the inputs of the adder modulo two AddM2.N–1, and at its second input, the value R_{N-1} is fed from the register RgR_{N-1} and at the output AddM2.N–1, the result $R=[A^1(x)\cdot B^1(x)]\cdot \mathrm{mod}P^1(x)=R_{N-1}$ is formed, which is recorded in the register RgR.

After applying the N+1 clock signal at the outputs of the RgR register, the result of multiplication modulo polynomials is formed:

$$[A^1(x)\cdot B^1(x)]\cdot \mathrm{mod}P^1(x)=R_N.$$

Further, as the next clock signals are applied, the output of the multiplier circuit will have the value of the residues R_{N+1}, R_{N+2}, ..., R_{N+K}.

Consider an example of multiplying polynomials modulo on a four-stage pipeline (N=4) of the following polynomials, which are shown in Table 1.

Table 2 shows the results of step-by-step calculation of the parameters of triples of polynomials TP_1÷TP_3 modulo on a four-stage pipeline.

Table 1

Input data

TP_i	Polynomials	Binary representation
TP_1	$A(x)_1=x^2+1$	0101_2
	$B(x)_1=x^3+x+1$	1011_2
	$P(x)_1=x^4+x+1$	10011_2
TP_2	$A(x)_2=x^3+x+1$	0101_2
	$B(x)_2=x^3+x^2+1$	1101_2
	$P(x)_2=x^4+x^3+1$	11001_2
TP_3	$A(x)_3=x^3+x^2$	1100_2
	$B(x)_3=x^3+x$	1010_2
	$P(x)_3=x^4+x^3+x^2+x+1$	11111_2

Table 2

Results of step-by-step calculation of parameters of polynomials TP_1, TP_2 and TP_3 on a four-stage pipeline

Clock / Polynomials	Clock 1	Clock 2	Clock 3	Clock 4	Clock 5	Clock 6
$A(x)_1=0101$ $B(x)_1=1011$ $P(x)_1=10011$	$r_1=1010$ $R_0=A(x)_1=0101$	$r_2=0111$ $R_1=1111$	$r_3=1110$ $R_2=111$	$R_3=0001$	–	–
$A(x)_2=1011$ $B(x)_2=1101$ $P(x)_2=11001$	–	$r_1=1111$ $R_0=1011_2$	$r_2=0111$ $R_1=1011$	$r_3=1110$ $R_2=1100$	$R_3=0010$	–
$A(x)_3=1100$ $B(x)_3=1010$ $P(x)_3=11111$	–	–	$r_1=0111$ $R_0=0000$	$r_2=1110$ $R_1=0111$	$r_3=0011$ $R_2=0111$	$R_3=0100_2$

Checking:

$$R=[A(x)_1\cdot B(x)_1]\mathrm{mod}P(x)_1=$$

$$=(x^3+x+1)(x^2+1)\mathrm{mod}x^4+x+1=$$

$$=(x^5+x^2x+1)\mathrm{mod}x^4+x+1=1.$$

The polynomial $R=x^0=1$ corresponds to the binary code 0001_2:

$$R=[A(x)_2\cdot B(x)_2]\mathrm{mod}P(x)_2=$$

$$=(x^3+x+1)(x^3+x^2+1)\mathrm{mod}x^4+x^3+1=$$

$$=(x^6+x^5+x^4+x^3+x^2+x+1)\mathrm{mod}x^4+x^3+1=x.$$

The polynomial $R=x$ corresponds to the binary code 0010_2:

$$R=[A(x)_3\cdot B(x)_3]\mathrm{mod}P(x)_3=$$

$$=(x^3+x^2)(x^3+x)\mathrm{mod}x^4+x^3+x^2+x+1=$$

$$=(x^6+x^5+x^4+x^3)\mathrm{mod}x^4+x^3+x^2+x+1=x^2.$$

The polynomial $R=x^2$ corresponds to the binary code 0100_2.

5. 2. FPGA-based hardware implementation of pipeline multiplier circuit

The results of multiplication of polynomials TP_1-0001 are formed at the output of the pipeline after the CLK.4 clock pulse is applied. The results of processing triples of TP_2 and TP_3 polynomials accordingly, 0010 and 0100 at the pipeline outputs are formed after the CLK.5, CLK.6 clock signals are applied.

After applying clock signals CLK.1, CLK.2, CLK.3, triples of polynomials TP_1, TP_2 and TP_3 are fed from the input generators to the inputs of the first stage of the pipeline.

Thus, it took six cycles to multiply four-bit polynomials modulo.

Fig. 3 shows a time diagram of the modulo operation of the pipeline multiplier for triples of polynomials TP_1, TP_2 and TP_3.

Table 3 shows the number of LUTs of slices, the number of fully usable LUTRAM and FF pairs, the number of IOBs linked, the number of BUFG necessary in the design of the microcircuit.

The use of hardware resources, namely registers, increases as the length and depth of the encryption block increase. The algorithm is implemented for n-bit input data.

Table 3

Hardware parameters summary as FPGA synthesis report

Resource	Estimation	Available	Utilization, %
LUT	14	63,400	0.02
LUTRAM	1	19,000	0.01
FF	52	126,800	0.04
IO	29	210	13.81
BUFG	1	32	3.13

6. Discussion of experimental results of pipeline multiplier

The pipelined implementation of multiplication of polynomials shown in Fig. 1 is one of the methods for speeding up the operation of circuits, which represents a speed increase at a large data stream. Due to the pipeline structure of the circuit, all data are calculated without waiting for the end of the calculation of the previous data. In the sequential implementation of the polynomial multiplication circuit, the calculation of the next data has to wait for the end of calculation of the previous one, which leads to delays in obtaining the results of the multiplication. So, we can note the high performance of the pipeline calculation method relative to the sequential one. For clarity, below is a temporary calculation of the multiplication of polynomials using the pipeline method and without it.

As a result, the high performance of the system in time due to the pipelined method of calculation can be noted. For this purpose, the processing time of polynomials with and without a pipeline is given below.

The processing time for polynomials without the pipeline is defined by the formula $T_{w.p.}=NKT_{dcp}$, where K is the number of processed triples of polynomials, N is the number of stages of the pipeline, T_{dcp} is the duration of the clock period, which is determined by $T_{dcp}=T_{PRF}+T_{br}$, where T_{PRF} is the time of forming the partial residues, T_{br} is the recording time result format in the buffer registers.

The processing time over K input streams of polynomials on an N-stage pipeline with a clock period T_{dcp} can be determined by the ratio [10]:

$$T_{NK}=(N+(K-1))T_{dcp}.$$

This formula reflects the fact that N cycles must pass before the result of calculating the first three polynomials appears at the output of the pipeline. The gain in time C during pipelining can be calculated by the formula:

$$C=(NK-(N+K-1)T_{dcp}.$$

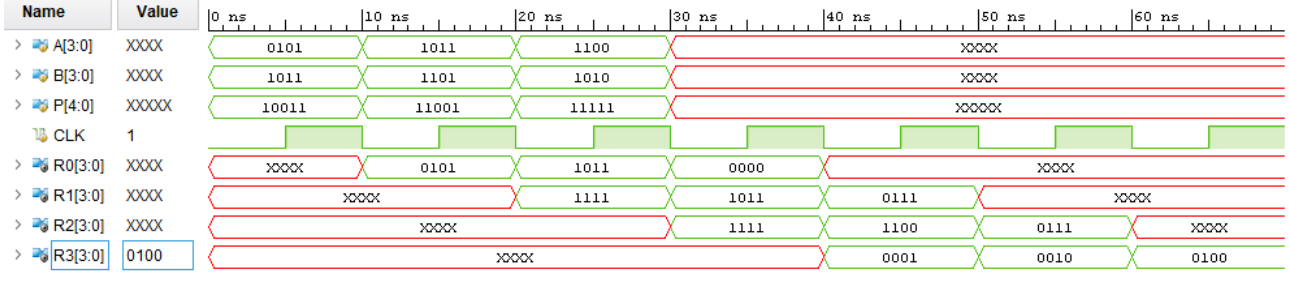

Fig. 3. Time diagram of the operation of a four-stage pipeline

For our example shown in Fig. 3, $K=3$ and $N=4$, then:

$$C=(3\cdot4-(4+2))T_{dcp}=(12-6)T_{dcp}=6T_{dcp}.$$

As the number of processed triples of polynomials K and the number of steps N increase, the value of C increases sharply. For example, when $K=50$ and $N=10$.

$$C=[(50\cdot10)-(10+49)\ T_{dcp}]=(500-59)=441\ T_{dcp}.$$

In accordance with the calculations presented above, the use of the pipeline method of multiplication of polynomials in this work shows an increase in the speed of work even when using polynomials of small bit. Nevertheless, as we know, cryptosystems use polynomials of large capacity and, accordingly, the difference in speed will grow with bit length.

Thus, the pipeline multiplier modulo, in comparison with the considered multiplication methods [14, 15, 19] (sequential, matrix), makes it possible to process the initial data in a streaming mode. This method improves the performance of the data encryption process. The developed pipeline multiplier can be used in hardware and software-hardware implementations of NPNS cryptosystems.

As shown in Fig. 1, the polynomial multiplication pipeline scheme consists of N multipliers, which in turn perform the polynomial multiplication operation. Accordingly, with an increase in the bit length of the polynomials, hardware resources that will be used in the construction of this circuit will also increase. To eliminate this shortcoming, it is planned to optimize the polynomial multiplication circuit used as part of the pipeline. One option is to consider several bits of the multiplier per clock signal, which leads to the use of a small number of steps in polynomial multiplications and a decrease in the hardware resources used.

7. Conclusions

1. A more efficient functional diagram with a detailed structure of logical blocks has been developed based on the pipeline multiplier modulo. In addition to the fact that this method can process a large data stream, the pipelined method has an internal buffer memory, which allows the calculation to be performed independently of previous calculations. For the hardware implementation of this algorithm, FPGA was chosen, where it is possible to implement all the features of the developed circuit. The time characteristics of performing calculations with sequential and matrix circuits were compared and the advantages of the new circuit were shown.

2. The functional circuit model was tested by using FPGA Artix 7, where the multiplication operation is performed without error for a minimum of 4 bits of data in 10 ns. In this case, the hardware resource used is no more than 0.02 %.

Acknowledgments

This research was funded by the Committee of Science of the Ministry of Education and Science of the Republic of Kazakhstan grant OR11465439.

References

1. Li, L., Li, S. (2016). High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication Over GF(2m). IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 24 (4), 1223–1232. doi: https://doi.org/10.1109/tvlsi.2015.2453360
2. Mohaghegh, S., Yemiscioglu, G., Muhtaroglu, A. (2020). Low-Power and Area-Efficient Finite Field Multiplier Architecture Based on Irreducible All-One Polynomials. 2020 IEEE International Symposium on Circuits and Systems (ISCAS). doi: https://doi.org/10.1109/iscas45731.2020.9181179
3. Nejatollahi, H., Gupta, S., Imani, M., Rosing, T. S., Cammarota, R., Dutt, N. (2020). CryptoPIM: In-memory Acceleration for Lattice-based Cryptographic Hardware. 2020 57th ACM/IEEE Design Automation Conference (DAC). doi: https://doi.org/10.1109/dac18072.2020.9218730
4. Singh, J., Kumar, S. (2021). A new class of irreducible polynomials. Communications in Algebra, 49 (6), 2722–2727. doi: https://doi.org/10.1080/00927872.2021.1881789
5. Devi, S., Mahajan, R., Bagai, D. (2021). A low complexity bit parallel polynomial basis systolic multiplier for general irreducible polynomials and trinomials. Microelectronics Journal, 115, 105163. doi: https://doi.org/10.1016/j.mejo.2021.105163
6. Svoboda, A. Valach, M. (1955). Operatorove obvody. Stroje Na Zpracovani Informaci, 3, 247–296.
7. Akushskiy, I. Ya., Yuditskiy, D. I. (1968). Mashinnaya arifmetika v ostatochnykh klassakh. Moscow: Sovetskoe radio, 440.
8. Sinha Roy, S., Basso, A. (2020). High-speed Instruction-set Coprocessor for Lattice-based Key Encapsulation Mechanism: Saber in Hardware. IACR Transactions on Cryptographic Hardware and Embedded Systems, 443–466. doi: https://doi.org/10.46586/tches.v2020.i4.443-466
9. Cenk, M., Özbudak, F. (2011). Multiplication of polynomials modulo x_n. Theoretical Computer Science, 412 (29) 3451–3462. doi: https://doi.org/10.1016/j.tcs.2011.02.031
10. Biyashev, R. G., Nyssanbayeva, S. E. (2012). Algorithm for creating a digital signature with error detection and correction. Cybernetics and Systems Analysis, 48 (4), 489–497. doi: https://doi.org/10.1007/s10559-012-9428-5
11. Nysanbaev, R. K. (1999). Kriptograficheskiy metod na osnove polinomial'nykh bazisov. Vestnik Ministerstva nauki i vysshego obrazovaniya i Natsional'noy akademii nauk Respubliki Kazakhstan, 5, 63–65.
12. Yenlik, B., Olga, U., Rustem, B., Saule, N. (2020). Development of an automated system model of information protection in the cross-border exchange. Cogent Engineering, 7 (1), 1724597. doi: https://doi.org/10.1080/23311916.2020.1724597

13. Kapalova, N., Khompysh, A., Arici, M., Algazy, K. (2020). A block encryption algorithm based on exponentiation transform. Cogent Engineering, 7 (1), 1788292. doi: https://doi.org/10.1080/23311916.2020.1788292
14. Kalimoldayev, M., Tynymbayev, S., Magzom, M., Ibraimov, M., Khokhlov, S., Abisheva, A., Sydorenko, V. (2019). Polynomials multiplier under irreducible polynomial module for high-performance cryptographic hardware tools. CEUR Workshop Proceedings, 2393, 729–737. Available at: http://ceur-ws.org/Vol-2393/paper_363.pdf
15. Kalimoldayev, M., Tynymbayev, S., Gnatyuk, S., Khokhlov, S. et. al. (2019). Matrix multiplier of polynomials modulo analysis starting with the lower order digits of the multiplier. NEWS of National Academy of Sciences of the Republic of Kazakhstan, 4 (436), 181–187. doi: https://doi.org/10.32014/2019.2518-170x.113
16. Jankowski, K., Laurent, P., O'Mahony, A. (2012). Intel Polynomial Multiplication Instruction and its Usage for Elliptic Curve Cryptography. White Paper, 17. Available at: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/polynomial-multiplication-instructions-paper.pdf
17. Xilinx. Available at: https://www.xilinx.com/products/boards-and-kits.html
18. IEEE Standard 1364-2005. IEEE Standard for Verilog Hardware Description Language. doi: https://doi.org/10.1109/ieeestd.2006.99495
19. Kalimoldayev M., Tynymbayev S., Ibraimov M., Magzom M., Kozhagulov Y., Namazbayev T. (2020). Pipeline multiplier of polynomials modulo with analysis of high-order bits of the multiplier. Bulletin of National Academy of Sciences of the Republic of Kazakhstan, 4 (386), 13–20. doi: https://doi.org/10.32014/2020.2518-1467.98