

High-speed dividing device with the formation of quotient and remainder

Yevgeniya Aitkhozhayeva, Khalicha Yubuzova

Department of Cybersecurity, Information Processing and Storage, Faculty of Automation and Information Technologies, Satbayev University, Almaty, Kazakhstan

Article Info

Article history:

Received Apr 24, 2024

Revised Apr 1, 2025

Accepted May 10, 2025

Keywords:

Hardware division

Integers

Performance

Quotient

Remainder

ABSTRACT

Considered the possibility of accelerating the time-critical operation of division for multi-bit integers. This problem is significant since, so multi-bit integers are widely used in specialized devices, including cryptographic transformations. A method for high-speed quotient and remainder determination with optimal hardware costs is proposed. A preliminary increase in the divisor and its subsequent decrease by shifting it to the right are used. A structural diagram and functional diagram of the hardware implementation have been developed using high-speed combinational logic circuits. The device's principle of operation, its step-by-step process, and specific examples illustrating its correct operation and resource efficiency are addressed. On average, it takes $(k/2+1)$ clock cycles to obtain the result, where $(k+1)$ bit capacity of the quotient. In most division schemes with optimal hardware costs, the number of clock cycles required to obtain the quotient (without remainder) is $(k+1)$. High-speed division with simultaneous determination of several quotient bits requires (m/p) clock cycles for the division operation, where p - the number of simultaneously determined quotient bits, m - bit capacity of dividend. However, this approach will require additional hardware. The research will continue by modeling the device in Vivado Design Suite computer aided design (CAD) based on Artix-7 field programmable gate array (FPGA) from Xilinx.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Khalicha Yubuzova

Department of Cybersecurity, Information Processing and Storage

Faculty of Automation and Information Technologies, Satbayev University

22a Satpaev street, Almaty 050013, Kazakhstan

Email: k.yubuzova@satbayev.university

1. INTRODUCTION

Issues of increasing the speed up of digital devices have always been relevant and attracted the attention of both theorists and practitioners - designers of digital computing devices [1]. To increase the speed up of digital devices, various methods have been developed and are being developed to speed up the execution of operations. This is especially true for complex operations of multiplication, division, and their modifications. Moreover, integer arithmetic is of particular importance, since integers are widely used in specialized devices, including for performing cryptographic transformations. The most complex and resource-intensive operation is division and its modification - reduction modulo. The division operation is crucial for both software processors and digital signal processing, machine learning, cryptographic or arithmetic logic devices [2]–[7]. Division and reduction modulo are also using in arithmetic in the Galois field, that is, in devices for forming elements of finite fields, along with multiplication, addition, and squaring. A large number of research-scientific works are devoted to the hardware implementation of high-

speed division devices and devices of modulo-reducing integers, the results of which have been reflected in patents and articles [8]–[16].

Various methods of accelerating the division of integers, implemented hardware, allow to obtaining of high-speed dividing devices with different structures of division devices and modulo-reduction devices. Matrix and pipeline devices for subtracting the divisor (modulus) B from the dividend (reducible number) A , Vedic algorithms, are high-speed [17]–[21]. Often, to speed up the division, the problem has been reduced to the use of multiplication operations, which can then be performed by fast multiplication algorithms. There are Barrett algorithm and its modifications, Montgomery algorithm and its modifications, Goldschmidt algorithm, Wallace algorithm, and others, used to speed up the division [22]–[30]. Most of these methods cannot be used when working with multi-bit binary integers, since the costs, which directly depend on the bit capacity of the numbers used, for hardware implementation are very high. In specialized computing device of a certain type, large integers with thousands or millions of decimal digits can be used (for example, those with an order of 10^{309} when implementing the Rivest–Shamir–Adleman (RSA) algorithm).

The purpose of the work is to develop a high-speed integer division device for obtaining the quotient and remainder, based on the well-known modulo reduction device with proven optimal hardware costs and high performance [31]. The proposed modification of the known device expands its functionality due to additional minimal hardware costs without performance loss. The global goal of developing this resource efficient device is to achieve sustainable development goal 12: responsible consumption and production. Using the proposed device in digital infrastructure processors operating with large integers will improve the quality of infrastructure, infrastructure reliability, and resiliency of infrastructure, which will contribute to achieving sustainable development goal 9: industry, innovation and infrastructure.

The paper is organized into the following sections. Section 2 describes the division method for obtaining the quotient and remainder using an enlarged divisor and provides a detailed description of the proposed equipment. Section 3 describes the step-by-step operation of the device, using specific examples to illustrate its correct operation. Section 4 contains a concluding remark. There is an acknowledgements section. The funding information section indicates that there is no funding source, the author contribution statement section provides information about the contribution of each author, and the conflict of interest section states that there is no conflict of interest. The data availability section provides the initials of an author who can provide data that supports the conclusions of this study.

2. METHOD

As a basis for obtaining an optimal dividing device, a modulo reduction device was chosen, which implemented a method for accelerating the obtaining of a remainder using the increased modulus [31]. The device has been modified to obtain quotient. In this case, the device can be used both to obtain quotient (division operation) and the remainder (modulo reduction operation).

The method used is iterative and is based on subtracting the increased divisor B (modulus) from the dividend A (reducible number). The divisor B is shifted towards the higher digits (to the left) so that its bit capacity becomes equal to the bit capacity of number A . In this case, the number B is increased by 2^k times, where $k = m - j$ (m is the number of binary digits of A , j is the number of binary digits of B). Then the increased divisor B is subtracted from A to obtain the remainder R_1 . There are no restrictions on the bit capacity of the number A and B , except that the bit capacity of the number A must be greater than or equal to the bit capacity of the number B . At the beginning of the operation, the dividend A is considered as a zero remainder R_0 .

On each iteration, after obtaining the next partial remainder R_i , the increased divisor B is shifted right by one bit (it is reduced by half). If the remainder R_i is positive, then the formed $i - th$ digit of the quotient is set to 1, and the reduced divisor is subtracted from the received remainder R_i . If the remainder R_i is negative, the formed $i - th$ digit of the quotient is “0” and the reduced divisor is subtracted from the previous remainder R_{i-1} . The received remainder R_i is analyzed at each iteration, making it possible to terminate the operation at any iteration if the remainder is less than the divisor ($R_i < B$). Most other division acceleration methods require all division steps to be performed.

Figure 1 shows the structural diagram of a dividing device with the formation of a quotient and a remainder. Block A is required to synchronize the operation of the device. It contains the clock pulse generator, the start trigger, the AND gate and the delay element. Block B is a block of memory containing a block of logic gates AND and registers for storing the divisor B and the increased divisor B (shifted towards the most significant bits). Block C includes the delay element, the register, blocks of logic gates OR , AND , and the combinational binary adder, performing on the first iteration of the operation of subtraction of the increased divisor B from the dividend A . On subsequent iterations the binary adder performs subtraction the

right-shifting divisor B from the resulting partial remainder R_i . The dividend A and remainders R_i are stored in the register of block C , the dividend A is considered as a zero remainder R_0 . Block D is required to form the result: the quotient (*Quotient* output) and the remainder (*Remainder* output) from the division. It contains the comparator circuit, blocks of logic gates *AND*, a clock pulse counter, an incomplete decoder, and the quotient register on triggers T . The initial state of the start trigger, counter, and registers into all blocks is zero. The forming of the quotient starts from the highest digit. Below is a description of the device operation at the structural diagram level.

The *Start* signal enables the writing of dividend A to block C , the writing divisor B , and increased divisor B to block B . The *Start* signal sets the start trigger in block A to the high state but with a delay. During the delay in block C on the combinational binary adder, subtraction the increased divisor B from the dividend A is performed, resulting in the remainder R_1 . During the same time, the dividend A (zero remainder R_0) and divisor B are compared in block D . If $A < B$, then the operation is over. In block D a signal *Cancel* is generated, which outputs a quotient equal to zero on the *Quotient* output, outputs the remainder (dividend A) on the *Remainder* output, and switches the start trigger in the synchronization block A to the inactive state. The remainder of R_1 is not used.

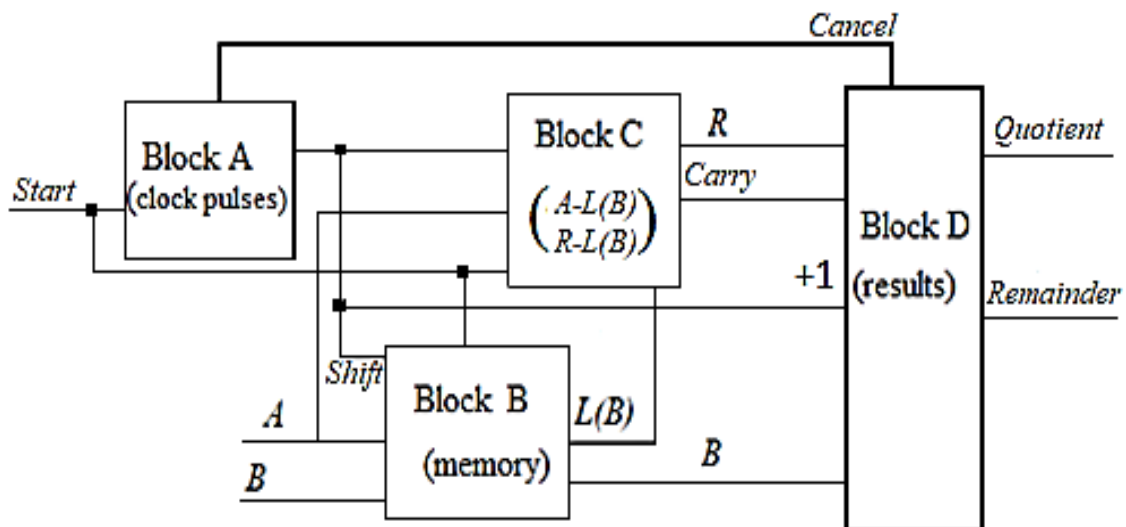


Figure 1. Structural diagram of a device

If $A \geq B$, then signal *Cancel*, is no generated and the operation continues. When calculating the remainder R_1 in block C , a carry signal is generated at the output of the combinational adder. If the remainder R_1 is negative, the carry signal is equal "0", and the remainder R_1 is not used. If the remainder R_1 is positive, the carry signal is equal to "1", and allows writing R_1 instead of number A with the arrival of the first clock pulse in the register of block C . The carry signal equal to "1" goes to block D , in which with the arrival of the next clock pulse is formed one in the high digit of the quotient register. Also, this clock pulse with a delay is received by block B . In block B , the increased divisor ($B \times 2^k$) is shifted one binary digit to the right and reduced by half, resulting in $(B \times 2^{(k-1)})$. The division process continues.

For each i -th step of the division, it is true: a) if at the i -th step of division $R_i < B$, R_i will be the remainder of the division, the quotient has already been formed, the process is completed; b) if at the i -th step of division $R_i \geq B$ and $R_i < 0$, then the previous remainder R_{i-1} remains in the register of block C without changes, in the i -th digit of the quotient register remains "0", and the process of dividing continues; and c) if at the i -th step of division $R_i \geq B$ and $R_i > 0$, the remainder R_i is written to the register of block C instead of the previous remainder R_{i-1} . In addition, "1" is written to the i -th bit of the quotient register of block D , and the operation continues.

The next bit of the quotient is formed in block D with the arrival of each clock pulse. The decoder, using the value of the clock pulse counter, determines the number of the quotient register digit into which the next quotient digit is written. The counter bit capacity and decoder input count are determined by $n = \log_2(k) + 1$ (rounded up to an integer). The number of decoder outputs is $(k + 2)$. The bit capacity of the quotient register is $(k + 1)$. The outputs of the decoder, starting from the second one, are connected to the inputs of the quotient register through block of logic gates *AND* (a total of $k + 1$ connections).

Figure 2 shows the functional diagram of the device. The following describes how the device operates at the functional diagram level, including a detailed description of signals passing. The device contains registers *RGA* (*ReGister A*), *RGS* (*ReGister S*), *RGB* (*ReGister B*), *RGQ* (*ReGister Q*); combinational adder *Add*; two delay elements *DL1* and *DL2*; a comparator circuit *Comp*; clock generator *GCLC* (*Generator Clock LogiC*); trigger *T*; *AND1* gate; blocks of logic gates *AND2* ÷ *AND7*, *OR*; clock pulse counter *CT* (*CounTer*); incomplete decoder *DC* (*DeCoder*). *RGA* register of *m* bit capacity first for storing the dividend *A* and then the partial remainders R_i . *RGB* register of *j* bit capacity for storing the divisor *B*. Shift register *RGS* of *m* bit capacity for storing the increased divisor. Combinational adder *Add* has *m* bit capacity. *RGQ* register implementing on *T*-triggers, to store a quotient. The number of gates in blocks of logic gates *AND2*, *AND3*, *AND5*, *OR* is *m*. The number of gates in blocks of logic gates *AND6*, *AND7* is $(k + 1)$. The number of gates in block of logic gates *AND4* is *j*.

The input *Start* is used to start the quotient and remainder calculation. Input *A* is used to input the binary code of the dividend *A*, the input *B* is used to input the binary code of the divisor *B*. The *Remainder* output is used to output the result of the remainder; the *Quotient* output is used to output the result of the quotient. The start of device operation is determined by the moment a single *Start* signal is sent to the device input. It is necessary to consider actions upon receipt of a clock pulse (comparison operations are performed on the *Comp* circuit, changes in the state of the *CT* counter, formation of the next quotient digit in *RGQ*) and a delayed clock pulse (shift the increased divisor in *RGS* and writing the calculated remainder in *RGA*).

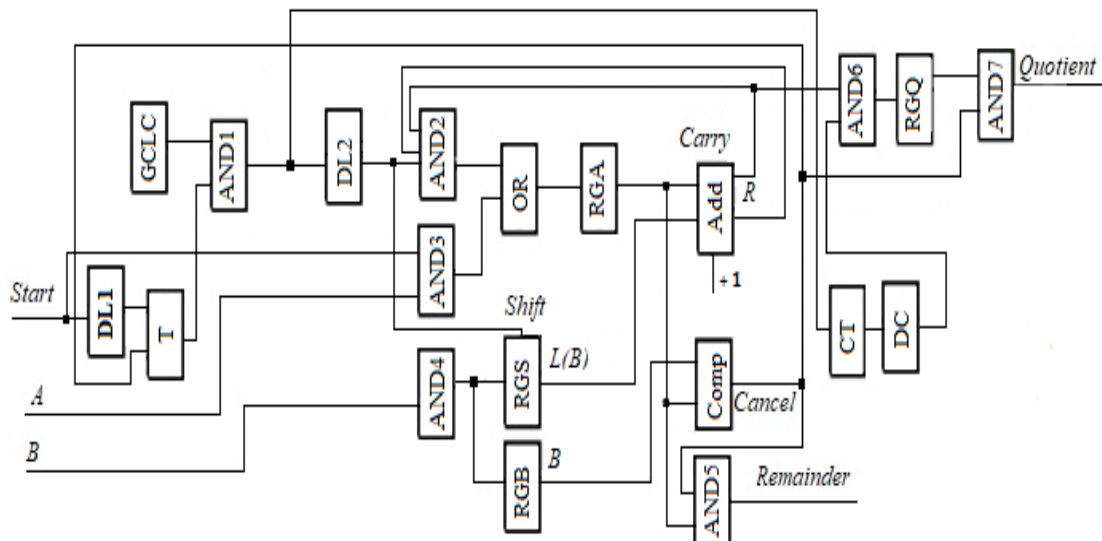


Figure 2. Functional diagram of a device

The *Start* signal permits the entry of dividend *A* through the *AND3* and the *OR* into the *RGA* register. In the *RGA* register, initially the number *A*, then the remainder R_i . At this moment also, the writing of the divisor *B* through the *AND4* into the register *RGB*, and to high digits of the shift register *RGS*. In the shift register *RGS* initially written $B \times 2^k$, then after the *i* – *th* shift stored $B \times 2^{k-i}$. This same *Start* signal, with a delay sets trigger *T* to the high state through delay element *DL1*. The high potential from the direct output of trigger *T* enabling the passage of clock pulses from the output of clock pulse generator *GCLC* through the *AND1* for clocking the device operation. The initial number *A* is considered as zero remainder R_0 . The code of the remainder R_0 from the register *RGA* and the divisor *B* from the register *RGB* are fed into the comparator circuit *Comp*. During comparing, alignment is based on the least significant digits. It is determined whether $A < B$ or not.

If $A < B$, then the number *A* is already the remainder. At the output of the comparator circuit *Comp*, a signal “1” (operation end signal *Cancel*) is generated, which is fed to the allowing inputs of logic gates *AND7* and *AND5*. So, allows the output of the quotient, equal to zero, from the output of the register *RGQ* through the *AND7* to the output *Quotient* and the remainder R_0 through the *AND5* from the output of the register *RGA* to the output *Remainder*. The same signal resets the trigger *T*, which disables the passage of clock signals through the *AND1*, the division operation stops.

Otherwise, if $A \geq B$, the division process continues. The dividend A from the register RGA is supplied to the first inputs of the combinational adder Add , and the inverse code of the increased divisor $B \times 2^k$ is supplied to the second inputs from the inverse outputs of the shift register RGS . "1" supplied at the input of the least significant bit of the combinational adder Add . Thus, subtraction is realized on the combinational adder Add using an additional code, i.e., the formation of the remainder $R_1 = R_0 - (B \times 2^k)$.

When subtracting, from the most significant bit of the combinational adder a carry signal is formed, equal to "0" or "1". The value of the carry signal depends on the relationship between R_0 (A) and $(B \times 2^k)$. Based on this carry signal, $A \geq (B \times 2^k)$ or $A < (B \times 2^k)$ is determined. If $A < (B \times 2^k)$, then the remainder R_1 is negative. In this case, the carry signal from the most significant bit of the combinational adder Add is equal to "0" (inhibiting signal to the logic gates $AND2$), and therefore the received remainder R_1 is not used.

If the remainder R_1 is positive, so from the high digit of the combinational adder Add the carry signal "1" is formed. The first clock pulse through the $AND1$ goes to the clock pulse counter CT , whose binary code becomes 00...01. The binary code from the outputs of the counter CT is fed to the inputs of the decoder DC . A signal "1" is formed on the second output of the decoder DC , and "0" is present on the other outputs of DC . The first output of DC is not used. The second and the rest outputs of the DC have been connected to the inputs of the quotient register RGQ through the logic gates $AND6$, starting from the high-digit RGQ . The binary code from DC is fed into the quotient register RGQ , as there is a signal "1" from the carry output of the combinational adder Add on the allowing inputs $AND6$. "1" is written into the most significant bit of the quotient register RGQ . The content of the quotient register becomes 10...0.

The carry signal "1" from the most significant bit of the combinational adder Add also arrives at the allowing inputs $AND2$ and, with the arrival of the first clock pulse delayed by the delay element $DL2$, allows the transfer R_1 through the $AND2$ to the input of the logic gates OR . The previous remainder R_0 ($R_0 = A$) in RGA is replaced by the received remainder R_1 . The first clock pulse delayed by the delay element $DL2$ also arrives to the input shift of the register RGS , shifting the increased divisor $(B \times 2^k)$ to right by one bit, reducing it by half, is obtaining $(B \times 2^{k-1})$. Thus, the contents of the register RGS and the register RGA are prepared for the next division step.

Further, the operation of the device is carried out similarly. At each $i - th$ iteration, the divisor B from the register RGB is supplied to the first inputs of the comparator circuit $Comp$, the number from the register RGA (R_{i-1}) is received to the second inputs of the comparator circuit $Comp$. In this way, $R_{i-1} < B$ or $R_{i-1} \geq B$ is determined. If $R_{i-1} < B$, a signal equal to "1" (signal *Cancel*) is generated at the output of the comparator circuit $Comp$. This signal is sent to the allowing inputs $AND7$ and $AND5$, which allows to output the quotient through the $AND7$ from the quotient register RGQ to the *Quotient* output and output the remainder R_{i-1} through the $AND5$ from the register RGA to the *Remainder* output. This signal resets trigger T (passage of the clock signal is prohibited), the formation of a quotient, and a remainder is complete.

If $R_{i-1} \geq B$, the comparator circuit $Comp$ generates a signal equal to "0", and the process continues. Simultaneously, the combinational adder Add receives the number R_{i-1} from register RGA at its first inputs, and to its second inputs the inverted code $B \times 2^{k-i}$ from the inverted outputs of shift register RGS . "1" is fed to the third input of the least significant bit of the combinational adder Add . Subtraction is implemented using an additional code, formed the remainder $R_i = R_{i-1} - (B \times 2^{k-i})$. $R_{i-1} \geq (B \times 2^{k-i})$ or not, is determined by the value of the carry signal from the most significant bit of the combinational adder Add .

If R_{i-1} is less than $B \times 2^{k-i}$, the carry signal from the high bit of the combinational adder Add equals "0" and the received negative remainder R_i will not be used. Otherwise ($R_{i-1} \geq (B \times 2^{k-i})$) a carry signal equal to "1" is generated from the high bit of the combinational adder Add , and the received positive remainder R_i will be used.

Without delay, the $i - th$ clock pulse through the $AND1$ is fed to the CT clock pulse counter, increasing its value by "1". The binary number from the counter CT output is input to the decoder DC . At the $(i + 1) - th$ output of the decoder DC , which is connected to the $i - th$ left digit of the quotient register RGQ through $AND6$, generates a signal "1". If the remainder of R_i is negative, then the $i - th$ digit of the register RGQ remains equal to "0", since there is no signal "1" from the *Carry* output of the combinational adder Add at the enabling inputs of $AND6$.

If remainder R_i is positive, in the $i - th$ digit of the register RGQ is set to "1" because at the allowing inputs of $AND6$ has signal "1" from the output *Carry* of the combinational adder Add and arrives signal "1" from the $(i + 1)$ output of DC . This same $i - th$ clock pulse, passing through the delay element $DL2$, arrives at the allowing inputs $AND2$. Since the second allowing inputs $AND2$ receive a carry signal from the most significant bit of the combinational adder Add equal to "1", then the transfer of the remainder R_i from the outputs of the combinational adder Add through information inputs $AND2$ and OR to the inputs of the register RGA is allowed. The remainder R_i is written to the RGA register (instead of the previous

remainder R_{i-1}).

The same clock pulse from the output $DL2$ goes to the input of the shift of the shift register RGS , shifting the value $B \times 2^{k-i}$ in the RGS to the right by one digit, halving it. The contents of the register RGS and the register RGA are prepared for the $(i + 1) - th$ division step. The process continues until the next remainder is less than B .

The clock pulse delay time on the delay element $DL2$ must be equal to the total time of operation of the counter CT , decoder DC , and block of logic gates $AND6$ so that in the register RGA remainder R_i is written after the formation of the $i-th$ digit for quotient in RGQ . The quotient register RGQ is built on T -triggers, so the arrival of “0” signals from the decoder outputs does not reset to “0” the bits where a “1” was formed in previous division steps.

On each $i - th$ clock pulse, if there is a carry signal “1” from the high digit of the combinational adder Add , the next positive remainder R_i is written from the outputs of the combinational adder Add to the register RGA . R_i replaces the previous remainder R_{i-1} . Simultaneously, the value of $B \times 2^{k-i}$ in shift register RGS is shifted by one bit to the right. Therefore, the delay time of the $Start$ signal on the delay element $DL1$ to set the trigger T to the “1” state, allowing the passage of clock pulses, and the period of clock pulses, must exceed the sum of the time the signal passes through the block of logical gates $AND2$ ($AND3$), OR , $AND6$, write time in the register RGA and subtraction time on the combinational adder Add .

The time of writing divisor B in the registers RGB and RGS and the shift time of value $B \times 2^{k-i}$ in the shift register RGS in the next steps of the dividing, has not been considered. This is because the time writing and shifting coincide with the signal time passage through the blocks of logic gates $AND2$ ($AND3$) and OR and the writing of dividend A at the register RGA . The operation time of the comparator circuit $Comp$ is not considered since it performs the comparing simultaneously with the subtraction operation on the combinational adder Add .

3. RESULTS AND DISCUSSION

Empirical validation of the proposed method and its hardware implementation was carried out using specific examples and below considered two examples of performing the division operation step-by-step.

3.1. Example 1

The largest possible number A has a bit capacity of $m = 7$, and the divisor B has a bit capacity of $j = 4$. In this case, the bit capacity of the registers RGA , RGS and the combinational adder Add equal $m = 7$. The bit capacity of the register RGB is $j = 4$, $k = 7 - 4 = 3$, the bit capacity of the counter CT equal $n = \log_2(3) + 1 = 3$. The incomplete decoder DC has three inputs, but the number of used outputs of the decoder DC is four (outputs from the second to the fifth). The quotient register RGQ has a bit capacity of $k + 1 = 4$.

$$A=110_{10}=1101110_2, B=9_{10}=1001_2, B \times 2^k=B \times 2^3=1001000_2.$$

At the $Start$ signal, the binary code A (number 1101110) is written to the register RGA , the binary code of the divisor B (number 1001) is written to the register RGB , and $B \times 2^3$ (number 1001000) is written to the shift register RGS . In the comparator circuit $Comp$, the numbers from the register RGA and the register RGB are compared bit by bit. When comparing, the alignment is based on the low-order bits, i.e., numbers 1101110 and 0001001 are compared. Since $1101110 > 0001001 (A > B)$, the process continues. At the same time, the combinational adder Add performs the subtraction $R_l = A - (B \times 2^3)$:

$$R_l = 1101110 - 1001000 = 1101110 + 0110111 + 1_{\text{low-order digit}} = (1) 0100110$$

(1) indicates the carry signal out from the most significant bit of the adder. The remainder is positive. The carry signal is permissive, as it is equal to (1). The resulting remainder R_l will be used. The same $Start$ signal with a delay, since it passes through the delay element $DL1$, will set the trigger T to the “1” state.

Since trigger T is set to “1”, the first clock pulse is sent to the input of the counter CT through $AND1$. The content of the counter CT becomes 001. Code 001 is input to the decoder DC , generating a “1” signal at the second output of the decoder. This “1” through $AND6$ is writing to the first high bit of the quotient register RGQ , since on the allowing inputs of $AND6$ is present a carry signal (1) from the high bit of the combinational adder Add . As a result, the content of the quotient RGQ register becomes 1000.

The first clock pulse enters the allowing inputs of $AND2$ through delay $DL2$. The second allowing inputs of $AND2$ receives a permissive carry signal (1) from the high digit of the combinational adder Add . So the calculated remainder R_l from the combinational adder Add through the $AND2$ and OR is writing to register RGA . This causes the RGA register value to change to 0100110. Simultaneously, the first delayed

clock pulse, as it passes through delay element $DL2$, shifts the contents of the RGS shift register to the right by one bit. In the register RGS , $B \times 2^3$ is halved and becomes equal to $B \times 2^2 = 0100100$.

The process is repeated. In the comparator circuit $Comp$ compares the numbers from the register RGA and the register RGB : 0100110 and 1001. Since 0100110 is greater than 0001001 ($R_1 > B$), the process continues. At the same time, subtraction is performed on the combinational adder Add : $R_2 = R_1 - (B \times 2^2)$.

The result is $R_2 = 0100110 - 0100100 = 0100110 + 1011011 + 1_{\text{low-order digit}} = (1) 0000010$.

The (1) indicates the carry signal out from the most significant bit of the adder. The remainder is positive. The carry signal is permissive because it is equal to (1). The resulting remainder of R_2 will be used.

The second clock pulse is supplied to the input of the CT counter through the $AND1$. Counter CT is set to 010. This value (010) is received at the input of the DC , resulting in a "1" signal at the third output of DC . Signal from output DC , by the carry enable signal (1) from the high digit of the combinational adder Add , is writing to the second bit of the register quotient RGQ . As a result, the content of the quotient RGQ register becomes 1100.

The second clock pulse that passes through the delay $DL2$ arrives at the allowing inputs of the $AND2$. The second allowing inputs of the $AND2$ receive a permissive carry signal (1) from the high digit of the combinational adder Add . The information inputs of the $AND2$ receive the calculated remainder R_2 from the output of the combinational adder Add . The remainder of R_2 passes through the blocks of logic gates $AND2$ and the OR and is write to the register RGA . The value of the register RGA changes to 0000010.

The process is repeated. The comparator circuit $Comp$ compares the numbers in the registers RGA and the RGB : 0000010 and 1001. Since $0000010 < 0001001$ ($R_2 < B$), the comparator circuit $Comp$ generates a process end signal. This signal resets the T trigger to the zero state (prohibiting the passage of clock pulses from the $GCLC$ (*Generator Clock LogiC*)) and goes to the enabling inputs of the $AND5$ and $AND7$. The division is completed, the remainder is $0000010_2 = 2_{10}$, and the quotient is $1100_2 = 12_{10}$. The calculation was correct because $110/9$ leaves a remainder of 2 and the quotient is 12. It took two clock pulses to complete the division.

3.2. Example 2

For greater clarity, consider the example of operation of the dividing device with the formation of the quotient and remainder with an illustration in both the decimal and binary number systems. $A = 916_{10} = 1110010100_2$, $m = 10$, $B = 45_{10} = 101101_2$, $j = 6$, $k = 10 - 6 = 4$. In this case, the register RGA and the shift register RGS have a bit capacity of 10, the register RGB has a bit capacity of 6. Therefore, $k = 10 - 6 = 4$. The quotient register RGQ has a bit capacity of $(k + 1) = 5$, and the CT clock pulse counter has a bit capacity of $\log_2(4) + 1 = 3$.

$$B \times 2^k = B \times 2^4 = 720_{10}, B \times 2^{k-1} = B \times 2^3 = 360_{10}, B \times 2^{k-2} = B \times 2^2 = 180_{10}, \\ B \times 2^{k-3} = B \times 2^1 = 90_{10}, B \times 2^{k-4} = B \times 2^0 = 45_{10}.$$

Table 1 shows what will happen in device when each i -th clock pulse and i -th delayed clock pulse arrives. For greater clarity, the values of the registers (RGA , RGB , RGS) and remainder R_i , actions on the $Comp$ comparator circuit, and the adder Add at each calculation step when performing a division operation are given in Table 1 in decimal number system.

For a better understanding of the process, the values of the quotient register RGQ , the counter CT at each step of division operation are given in Table 1 in decimal and binary number systems. The values the carry $Carry$ and the six outputs of the incomplete decoder DC are given in Table 1 in binary number system. The first output of the decoder DC is not used and is indicated in parentheses ().

Table 1. Procedure for calculating quotient Q and remainder R

Steps	RGA	RGB	RGS	$Comp$	Add	R_i	$Carry$	CT	DC	RGQ
Start	916	45	720	$916 > 45$	$916 - 720$	196	1	$0_{10} = 000_2$	(1)00000	$00000_2 = 0_{10}$
Clock pulse 1	916	45	720	$916 > 45$	$916 - 720$	196	1	$1_{10} = 001_2$	(0)10000	$10000_2 = 16_{10}$
Delayed clock pulse 1	196	45	360	$196 > 45$	$196 - 360$	-164	0	$1_{10} = 001_2$	(0)10000	$10000_2 = 16_{10}$
Clock pulse 2	196	45	360	$196 > 45$	$196 - 360$	-164	0	$2_{10} = 010_2$	(0)01000	$10000_2 = 16_{10}$
Delayed clock pulse 2	196	45	180	$196 > 45$	$196 - 180$	16	1	$2_{10} = 010_2$	(0)01000	$10000_2 = 16_{10}$
Clock pulse 3	196	45	180	$196 > 45$	$196 - 180$	16	1	$3_{10} = 011_2$	(0)00100	$10100_2 = 20_{10}$
Delayed clock pulse 3	16	45	90	$16 < 45$	$16 - 90$	-74	0	$3_{10} = 011_2$	(0)00100	$10100_2 = 20_{10}$

Table 1 shows that with the arrival of the delayed third clock pulse, the remainder $R_3 = 16$ from the output of the combinational adder Add is written to the RGA . The remainder $R_3 = 16$ and divisor $B = 45$ is compared on the comparator circuit $Comp$. Since $16 < 45$, the signal *Cancel* is generated. This signal resets the

T trigger to the zero state, prohibiting the passage of clock pulses from the $GCLC$ and the process is over. The calculated remainder $R_4 = (-74)$ will not be used because there is no clock pulse and signal $Carry = 0$ on the allowing inputs $AND2$. The signal $Cancel$ allows the remainder R_3 from register RGA to be output to the $Remainder$ output, and the quotient from register RGQ to the $Quotient$ output.

To check the accuracy of correct operation of the device, we refer to Table 1. The remainder R is equal to R_3 , which is $16_{10} = 10000_2$ in this case, and the quotient from the division is $Q = 20_{10} = 10100_2$. Using these values and the divisor B , the dividend A can be calculated:

$$A = (Q \times B) + R = (20_{10} \times 45_{10}) + 16_{10} = 900_{10} + 16_{10} = 916_{10}.$$

The division has been performed correctly. This example demonstrates that it took three clock pulses to form a quotient with a remainder. When dividing by the method without restoring the remainder, which requires minimal hardware costs, it would take five clock pulses since the bit capacity of the quotient is five.

3.3. Discussion

In the developed dividing device, the formation of the quotient is performed simultaneously with the formation of the remainder, and the division time depends on the ratio of the values of the dividend A and the divisor B . If the quotient is equal to 2^i , then to obtain the quotient with a remainder it will take the i -th number of clock pulses. If the quotient has a "1" in the highest significant digit and the remaining digits are "0" ($100...0$), then the minimum number of clock pulses is required to obtain the quotient with the remainder - one clock pulse. The maximum number of clock pulses required to obtain a quotient with a remainder is $(k+1)$, where $(k+1)$ is bit capacity of quotient. On average, it takes $(k+1+1)/2 = (k/2+1)$ clock pulses to get the result. It should be noted that in most division schemes with optimal hardware costs, the number of clock cycles required to obtain the quotient (without remainder) is $(k+1)$.

The proposed method and the device for its implementation can be compared with high-speed division with the simultaneous determination of several bits of quotient, which requires m/p clock cycles to perform the division operation, where p is the number of simultaneously determined bits of quotient, m - bit capacity of dividend. In this case, performance is achieved by increasing hardware costs. The larger p , the less clock pulses are required, but the more additional circuits to simultaneously generate and store multiples of the divisor are required and more additional comparator circuits. These methods cannot be used when working with multi-bit binary integers, since the costs, which directly depend on the bit capacity of the numbers used, for hardware implementation will be very high.

4. CONCLUSION

Reducing the number of clock pulses when performing an operation in the presented device is achieved by comparing the remainder and divisor in each clock cycle. In this case, the calculation time does not increase, since the comparison of the remainder and the divisor is performed simultaneously with the subtraction operation on the combinational adder. The empirical verification is presented by two examples of performing the division operation for specific numbers. The examples demonstrated the correctness and the resource efficiency of the proposed method. With the high-speed division method, without restoring the remainder and requires minimal hardware costs, $(k+1)$ clock pulses are required. The proposed division method requires $(k/2+1)$ clock pulses, which is almost 50% less.

Let's compare the proposed method and method of simultaneous determination of two digits of the quotient for a specific value of m , at which requires $m/2$ clock pulses. If $m = 128$, then division with the simultaneous determination of two digits of the quotient will always require 64 clock cycles, two registers (for storing $2B$ and $3B$) instead of one register RGS , and three comparison circuits instead of one. Additional AND and OR logic gates will be required to transfer either B , or $2B$, or $3B$ to the adder, depending on the comparison results. If $m = 1024$, then always 512 clock cycles and the same additional circuits will be required, but the bit capacity of these circuits will increase.

When dividing integers, the bit capacity of the quotient $(k+1)$ is always less than or equal to the bit capacity of the dividend (m) . In the proposed method, the number of clock cycles depends only on the bit capacity of the quotient, and not on the bit capacity of the dividend. In the worst case $(k+1) = m$, for $m = 128$ and $m = 1024$, the number of clock cycles during division will be 65 and 513, respectively. The execution time of the operation will increase by only one clock cycle compared to the method of simultaneously determining two bits of the quotient, which is 1.6% (if $m = 128$) and 0.2% (if $m = 1024$). The larger the bit capacity of the integers, the lower the percentage of time loss, and the greater the gain in hardware cost.

The proposed method has a limitation: the bit capacity of divisor B should not exceed the bit capacity of the dividend A . This division device is not intended for use in systems without hardware cost

restrictions. The presented dividing device is designed to accelerate the division operation for large integers in computing systems that require optimal hardware costs.

The research will continue by modeling the device in Vivado Design Suite computer aided design (CAD) based on the Artix-7 FPGA from Xilinx. The expansion of the problem aims to develop high-speed methods for performing arithmetic operations with optimal hardware costs. This will help to improve the reliability of the infrastructure and the fault tolerance of the digital society infrastructure.

ACKNOWLEDGEMENTS

The authors would like to thank the Department of Cybersecurity, Information Processing and Storage at Satbayev University for approving and supporting this study. The work was conducted as an initiative project by the authors without a formal contract. The authors also extend their thanks to the editor and reviewers for their valuable comments.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Yevgeniya Aitkhozhayeva	✓	✓		✓	✓	✓		✓	✓	✓		✓	✓	
Khalicha Yubuzova	✓	✓			✓	✓		✓		✓	✓			

C : Conceptualization

M : Methodology

So : Software

Va : Validation

Fo : Formal analysis

I : Investigation

R : Resources

D : Data Curation

O : Writing - Original Draft

E : Writing - Review & Editing

Vi : Visualization

Su : Supervision

P : Project administration

Fu : Funding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author, [Yevgeniya Aitkhozhayeva], upon reasonable request.




REFERENCES

- [1] R. P. Brent and P. Zimmermann, *Modern Computer Arithmetic*. Cambridge University Press, 2010. doi: 10.1017/cbo9780511921698.
- [2] G. Han, W. Zhang, L. Niu, C. Zhang, Z. Wang, and Z. Wang, "Hardware Implementation of Approximate Fixed-point Divider for Machine Learning Optimization Algorithm," in *2022 IEEE Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia)*, IEEE, Nov. 2022, pp. 22–25. doi: 10.1109/PrimeAsia56064.2022.10104001.
- [3] A. A. Purohit, M. R. Ahmed, and R. V. S. Reddy, "Design of Area Optimized Arithmetic and Logical Unit for Microcontroller," in *2020 IEEE VLSI device circuit and system (VLSI DCS)*, IEEE, Jul. 2020, pp. 335–339. doi: 10.1109/VLSIDCS47293.2020.9179942.
- [4] U. S. Patankar, M. E. Flores, and A. Koel, "Novel data dependent divider circuit block implementation for complex division and area critical applications," *Scientific Reports*, vol. 13, no. 1, p. 3027, Feb. 2023, doi: 10.1038/s41598-023-28343-3.
- [5] Y. Z. Aitkhozhayeva, S. T. Tynymbayev, S. Adilbekkyzy, A. Skabylov and M. Ibraimov, "Design and research of the behavioral model for the modular reduction device," *Eurasian Physical Technical Journal*, vol.17, no.1, pp.151-156, Jun. 2020, doi: 10.31489/2020No1/151-156.
- [6] P. Choi, M.-K. Lee, J.-H. Kim, and D. K. Kim, "Low-Complexity Elliptic Curve Cryptography Processor Based on Configurable Partial Modular Reduction Over NIST Prime Fields," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 11, pp. 1703–1707, Nov. 2018, doi: 10.1109/TCSII.2017.2756680.
- [7] U. S. Patankar and A. Koel, "Review of Basic Classes of Dividers Based on Division Algorithm," *IEEE Access*, vol. 9, pp.




- 23035–23069, 2021, doi: 10.1109/ACCESS.2021.3055735.
- [8] S. S. Erdem and S. S. Erdem, "Efficient and Constant Time Modular Reduction With Generalized Mersenne Primes," in *IEEE Access*, vol. 12, pp. 189307–189316, Dec. 2024, doi: 10.1109/ACCESS.2024.3514918.
 - [9] M. Angioli *et al.*, "Design, Implementation and Evaluation of a New Variable Latency Integer Division Scheme," *IEEE Transactions on Computers*, vol. 73, no. 7, pp. 1767–1779, Jul. 2024, doi: 10.1109/TC.2024.3386060.
 - [10] A. R. Omondi, "Modular Reduction," *Advances in Information Security. Springer International Publishing*, pp. 105–141, 2020. doi: 10.1007/978-3-030-34142-8_4
 - [11] A. Greuet, S. Montoya and C. Vermeersch, "Quotient Approximation Modular Reduction," *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*, 2022, pp. 103–110, doi: 10.1109/ARITH54963.2022.00028.
 - [12] H. Yu, G. Bai, and H. Hao, "Efficient modular reduction algorithm without correction phase," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, pp. 304–313. doi: 10.1007/978-3-319-19647-3_28.
 - [13] F. Hassan, A. Ammar, and H. Drennen, "A 32-bit integer division algorithm based on priority encoder," in *ICECS 2020 - 27th IEEE International Conference on Electronics, Circuits and Systems, Proceedings*, IEEE, Nov. 2020, pp. 1–4. doi: 10.1109/ICECS49266.2020.9294888.
 - [14] S. Tynymbayev, R. Berdibayev, T. Omar, Y. Aitkhozhayeva, A. Shaikulova, and S. Adilbekkyzy, "High-speed devices for modular reduction with minimal hardware costs," *Cogent Engineering*, vol. 6, no. 1, Jan. 2019, doi: 10.1080/23311916.2019.1697555.
 - [15] A. Obshta, V. Khoma, and A. Prokopchuk, "Digital Division Algorithms for Efficient Execution on Integrated Circuits," *Advances in Cyber-Physical Systems*, vol. 9, no. 1, pp. 46–53, May 2024, doi: 10.23939/acps2024.01.046.
 - [16] A. Magyari and Y. Chen, "Hardware Optimized Modular Reduction," *Electronics*, vol. 14, no. 3, p. 550, Jan. 2025, doi: 10.3390/electronics14030550.
 - [17] V. M. Zakharov, V. A. Pesoshin, and S. V. Shalagin, "The Complexity of a Pipelined Algorithm for Remainder Computing in a Given Modulo," *Automatic Control and Computer Sciences*, vol. 52, no. 4, pp. 251–255, Jul. 2018, doi: 10.3103/S0146411618040120.
 - [18] M. K. S. Vikasini and B. J. Kailath, "16-bit Modified Vedic Paravartya Divider with quotient in fractions," in *TENSYMP 2021 - 2021 IEEE Region 10 Symposium*, IEEE, Aug. 2021, pp. 1–5. doi: 10.1109/TENSYMP52854.2021.9551013.
 - [19] J. Kuppli, M. S. D. Abhiram, and N. A. Manga, "Design of Vedic Mathematics based 16 bit MAC unit for Power and Delay Optimization," in *2021 International Conference on Nascent Technologies in Engineering, ICNET 2021 - Proceedings*, IEEE, Jan. 2021, pp. 1–4. doi: 10.1109/ICNET51185.2021.9487570.
 - [20] A. Eshack and S. Krishnakumar, "Pipelined Vedic multiplier with manifold adder complexity levels," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 10, no. 3, pp. 2951–2958, Jun. 2020, doi: 10.11591/ijece.v10i3.pp2951-2958.
 - [21] S. H. Alkurwy and I. S. Hameed, "A novel pipelined carry adder design based on half adder," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 25, no. 2, pp. 763–770, Feb. 2022, doi: 10.11591/ijeecs.v25.i2.pp763-770.
 - [22] P. L. Montgomery, "Modular Multiplication Without Trial Division," *Mathematics of Computation*, vol. 44, no. 170, p. 519, Apr. 1985, doi: 10.2307/2007970.
 - [23] J. C. Bajard, J. Eynard, and N. Merkiche, "Montgomery reduction within the context of residue number system arithmetic," *Journal of Cryptographic Engineering*, vol. 8, no. 3, pp. 189–200, Sep. 2018, doi: 10.1007/s13389-017-0154-9.
 - [24] G. Kim, E. Seo, Y. Lee, Y.-S. Kim and J.-S. No, "Lazy Modular Reduction for NTT," *Electronics*, vol. 13, no. 24, p. 550, p.4887, Dec. 2024, doi: 10.3390/electronics13244887.
 - [25] A. P. Renardy, N. Ahmadi, A. A. Fadila, N. Shidqi, and T. Adiono, "Hardware implementation of montgomery modular multiplication algorithm using iterative architecture," in *2015 International Seminar on Intelligent Technology and Its Applications, ISITIA 2015 - Proceeding*, IEEE, May 2015, pp. 99–102. doi: 10.1109/ISITIA.2015.7219961.
 - [26] W. Dai, D. Chen, R. C. C. Cheung, and C. K. Koc, "FFT-Based McLaughlin's Montgomery Exponentiation without Conditional Selections," *IEEE Transactions on Computers*, vol. 67, no. 9, pp. 1301–1314, 2018, doi: 10.1109/TC.2018.2811466.
 - [27] K. H. Kim, S. Mesnager and K. I. Pak, "Montgomery curve arithmetic revisited," *Journal of Cryptographic Engineering*, vol. 14, no. 2, pp. 343–362, May. 2024, doi: 10.1007/s13389-024-00353-5.
 - [28] C. S. Wallace, "A Suggestion for a Fast Multiplier," *IEEE Transactions on Electronic Computers*, vol. EC-13, no. 1, pp. 14–17, Feb. 1964, doi: 10.1109/PGEC.1964.263830.
 - [29] K. Juneja, A. Jangra, and D. Khurana, "Design of a Quaternary Component and Wallace Tree Integrated Baugh-Wooley Multiplier," *International Journal of Networked and Distributed Computing*, vol.13, no. 3, Dec. 2024, doi: 10.1007/s44227-024-00047-8.
 - [30] R. E. Goldschmidt, "Applications of division by convergence," Cambridge, 1964. [Online]. Available: <https://dspace.mit.edu/handle/1721.1/11113>.
 - [31] Y. Z. Aitkhozhayeva, S. T. Tynymbayev, N. A. Seilova, L. A. Tereikovska, and A. Zh. Imanbayev, "Method and device for modulus reduction," *Bulletin of National Academy of Sciences of the Republic of Kazakhstan*, vol. 2, no. 378, Apr. 2019, doi: 10.32014/2019.2518-1467.59.

BIOGRAPHIES OF AUTHORS



Yevgeniya Aitkhozhayeva    is Professor in Department of Cybersecurity, Information Processing and Storage Satbayev University in the Republic Kazakhstan, Candidate of Technical Sciences, academician of the International Academy of Informatization. One of the leading scientists of the Republic of Kazakhstan in the field of Computing and Information Security, a teacher with over 40 years of experience. She received a degree of Candidate of Technical Sciences at the Department of Computer Engineering St. Petersburg State Electro Technical Institute (Technical University “LETI”), Russia. She is interested in information security, databases systems, and hardware of cryptography, and has over 200 scientific and educational works, patents and certificates of authorship (including patents in the international database Derwent Innovations Index). She can be contacted at email: y.aitkhozhayeva@satbayev.university.



Khalicha Yubuzova    received the degree of Doctor of Philosophy (Ph.D.) in the specialty 6D070400 – “Computer Engineering and Software” from the Satbayev University, the Republic of Kazakhstan, Almaty, with the Dissertation “Methods of secure key distribution based on quantum cryptography protocols”. She is an associate professor in the Department of Cybersecurity, Information Processing and Storage, Satbayev University, Almaty. She has published over 100 scientific and educational works in journal papers, conference proceedings, including certificates of authorship. One of the leading scientists of the Republic of Kazakhstan in the field of computing and information security, a teacher with over 30 years of experience. The main fields of interest are information security, quantum cryptography, and cryptography of hardware. She can be contacted at email: k.yubuzova@satbayev.university.