Indonesian Journal of Electrical Engineering and Computer Science
Vol. 24, No. 3, December 2021, pp. 1354~1366
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v24.i3.pp1354-1366

Power optimization of binary division based on FPGA

Fadi T. Nasser, Ivan A. Hashim
Department of Electrical Engineering, University of Technology, Baghdad, Iraq

Article Info

Article history:

Received Jun 3, 2021
Revised Oct 23, 2021
Accepted Oct 27, 2021

Keywords:

Dynamic power optimization
Dynamic power reduction techniques
Low power division algorithm
Non-restoring division algorithm
Restoring division algorithm
VHDL approach

ABSTRACT

In modern very large scale integrated (VLSI) digital systems, power consumption has become a critical concern of VLSI designers. As size shrinks and density increases in chips, it will be a challenge to design high-performance and low-power digital systems. Therefore, VLSI designers are trying to reduce power dissipation in these systems by using power-optimization techniques. Different mathematical operations can be found in the architectures of most digital systems. The focus of this paper is division. In comparison to other basic computational operations, division requires more iterations, takes a long time, covers a large area, and consumes more power from the digital system. As a result, the system's design requires high speed and a low-power divider in order to improve its overall performance. This paper focuses on dynamic power dissipation. In order to determine which design consumes the lowest dynamic power, different system designs of digit-recurrence division algorithms, such as restoring division and non-restoring division are suggested. An innovative power-optimization technique, the very hardware descriptions language (VHDL) technique, is utilized to the suggested system designs. The VHDL technique achieved the higher optimization in dynamic power, at 93.66% for non-restoring division with internal-loop iteration, than traditional approaches.

Corresponding Author:

Fadi T. Nasser
Department of Electrical Engineering
University of Technology
Baghdad, Iraq
Email: eee.19.23@grad.uotechnology.edu.iq

1. INTRODUCTION

In recent decades, the modern digital system design with low power has been the primary concern of very large scale integrated (VLSI) designers. Since the VLSI technology was introduced, designers have paid great attention to cost, area, and speed. Later, the continuous requirements for high performance and low power digital systems led to an increase in density and a decrease in size of the integrated circuits (IC) under Moor's law. Primarily, this law is a global predictor for the growth of the entire semiconductor industry [1], [2].

It can be understood from Moore's law that the number of transistors in a chip will double every eighteen months [3]. The increases in the density of IC's produce a significant increase in power dissipation. Moreover, the advent of portable systems in recent years, which operate on batteries, has led to longer battery life. Therefore, in the contemporary era, the VLSI designers aimed to reduce the power dissipation by creating and using new techniques to reduce the power these systems consumed. Generally, the significant advantages of power optimization are increased system reliability, battery life efficiency, noise immunity, lower system cooling and packaging cost, and demand for portable systems [4]. The total power dissipated in a VLSI circuit is the sum of dynamic power and leakage or static power [5].

The dominant power in digital circuits and systems is dynamic power consumption. The reason to reduce the dynamic power is due to the ability to apply the dynamic power reduction techniques, and it is easy to handle the structure of logic elements for the digital circuits and systems. Additionally, it is not dependent on technology. At the same time, static power is dependent on technology and concerns about the manufacturing design's intellectual property (IP), such as the size of the transistor, channel length, and width of the gate oxide [6]. Therefore, this work focuses on reducing the dynamic power dissipation rather than static power, which is outside the scope of this paper.

The design of complex digital systems comprises various data processing units and mathematical operations, such as addition, subtraction, multiplication, and division. This paper is concerned with division, which is considered the heart of most computational digital systems such as digital signal processing (DSP), image processing, communication systems, artificial intelligence, quantum computing, and the internet of things (IoT) [7], [8]. Based on the conversion method, the division algorithms can be classified as follows: digit recurrence, functional iteration, very high radix, a lookup table (LUT), and variable latency.

Division is the most challenging and complicated operation among all the mathematical operations because of its sequential operation [9]. Therefore, it is more costly in propagation delay, area, and power consumption than other mathematical operations. Thus, many studies of division techniques have been implemented to reduce the power dissipation in divider circuitry at structural and algorithmic levels. In [10], the authors suggest a 32-bit unsigned integer divider based on a recursive non-restoring division algorithm. The proposed design achieves an optimization of 82.9% in dynamic-power dissipation relative to the sequential divider. Hashemi, Bahar and Reda proposes a dynamic, low-power, low-error divider. The design in the standalone case can save the total power dissipation up to 70% with an average error of 3.08 % [11].

The researchers in [12] propose a hybrid division called Pre-scaling, Series expansion, and Tylor expansion (PST) division, in which the design consists of three algorithms. The PST design optimises the dynamic power and total-power dissipation by 67% and 9.7%, respectively, when compared to IP core division. The implementation of an 8-bit dividend by a 4-bit divisor of Vedic division is proposed by Kishor and Bhaaskaran in [13].

This design shows a reduction of about 52% in total power dissipation in comparison with conventional dividers (division). The researchers in [14], propose an 8-bit dividend by a 4-bit divisor of binary dividers, which saves dynamic power by about 27% in comparison with conventional dividers that use the repetitive subtraction method. In this design, the Vedic division algorithm is used to eliminate the recursion, which significantly reduces dynamic power and area overhead. BhanuTej implements a 32-bit dividend by a 16-bit divisor of binary Vedic division.

This proposed design is applied on 180 nm and 32 nm field-programmable gate array (FPGA) platforms. The obtained dynamic power and the total power saved are 90% and 86%, respectively, with respect to traditional division [15]. The above divider designs were verified and implemented using different platforms, such as FPGA, application-specific integrated circuits (ASIC) and general-purpose processors (GPP).

In this paper, FPGA is introduced as a platform for implementing the proposed division-algorithm designs. The following characteristics are the reasons why FPGA was chosen over the other platforms: it is high density and high performance and does not have costly multicore processors. Moreover, FPGA can operate in parallelism and obtain orders-of-magnitude speedup, unlike GPP, in which they operate sequentially [16]-[18]. FPGA also has the advantages of software flexibility, hardware speed, re-programmability, minimal time to market and is ideal for prototyping designs in which the hardware testing and verification are performed quickly on the chip. Also, errors in design can be fixed without any extra hardware costs [19].

The main goal of this work is to decrease the dynamic-power consumption in digit-recurrence division, mainly for restoring division and non-restoring division algorithms. Consequently, many suggested systems of dividers are presented, and new optimization techniques are utilized to these suggested systems to decrease the dynamic power consumption and increase the execution time. The very hardware descriptions language (VHDL) technique is one of these techniques, this technique reduces the dynamic power by reducing switching activities. This technique also transforms the design into its basic elements to consume less power. The number of reduced cycles achieves high-speed performance.

This paper is organized as follows: the second section introduces the sources of power dissipation, while the third section describes techniques for dynamic power optimization. As a result of this, the fourth section explains the types of division algorithms used in this work, while section five describes the suggested system designs in detail. The simulation results are presented and discussed in the sixth section, and the conclusion is given in the seventh section.

2. SOURCES OF POWER DISSIPATION

Before understanding and studying the power reduction techniques in VLSI circuits, it is an urgent necessity to know the significant power dissipation sources in these digital system or circuits. Power dissipation in VLSI circuits can be categorized into two primary sources: dynamic power and static power. Static power is also known as leakage or quiescent power, which occurs due to the presence of different leakage currents, as demonstrated in Figure 1. Alternatively, it is the power in effect when a device is turned ON but there is no task to perform. This means it is idle (in standby mode), and there is no signal transition. Seven leakage currents participate in causing static-power dissipation.

These currents are: the reverse-biased PN junction diode leakage current(I_1), the reverse-biased PN junction current due to tunnelling of electrons (I_2), subthreshold leakage current (I_3), oxide-tunnelling leakage current (I_4), hot-carrier injection leakage current (I_5), GIDL (gate-induced drain leakage) current (I_6), and punch-through leakage current (I_7).

The equation of static power dissipation can be given as (1) [20],

$$P_{static} = I_{static} \times V_{DD} \quad (1)$$

Where P_{satitc} is the static power dissipation, I_{static} is the summation of all seven leakage current mechanisms, and V_{DD} is the power supply voltage.

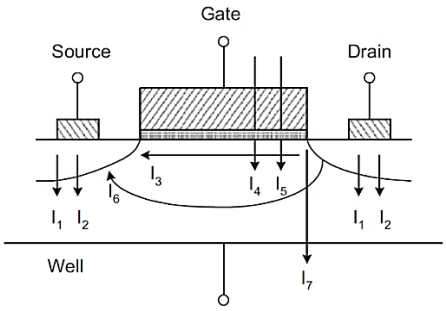

Figure 1. Sources of static power dissipation [21]

The other type of power dissipation is dynamic. Dynamic-power dissipation is the power consumed during a device's operation. In other words, it is the power consumed when a complementary metal oxide semiconductor (CMOS) device is in an active mode. This power has three major types. The first is dissipated power due to charging and discharging the capacitance known as switching power. The second is short-circuit power, which is the result of a crowbar flowing through a lapse of time when both $P\text{-}MOS$ (P-channel MOS) and $N\text{-}MOS$ (N-channel MOS) transistors are simultaneously turned ON.

The last one is glitching power, which occurs when input signals arrive at different times to a single logic block causing racing. Therefore, several intermediate transitions occur before the logic block output stabilizes. The calculation of three types of dynamic power dissipation can be given in the following [18]-[23],

$$P_{switch} = \alpha\, C_L V_{DD}^2 f_p \quad (2)$$

$$P_{s.c} = \frac{\beta}{12} V_{DD}^3 \left(1 - 2\frac{V_t}{V_{DD}}\right)^3 t_{rf}\ f_p \quad (3)$$

$$\mathrm{P_{gltich}} = \frac{1}{2}\mathrm{C_L V_{DD}(V_{DD} - V_t)} \quad (4)$$

The total dynamic power dissipation is given by the (5) [23], [24],

$$\mathrm{P_{dyn} = P_{switch} + P_{S.C} + P_{glitch}} \quad (5)$$

The average power dissipation can be given in the (6) [23], [24],

$$\mathrm{P_{total} = P_{static} + P_{dyn}} \quad (6)$$

By substituting ((1), (2)-(3)) in (4), the total power dissipation is summarized in the (7) [23], [24],

$$P_{total} = (I_{leakage} \times V_{DD}) + (\alpha C_L V_{DD}^2 f_p) + \left(\frac{\beta}{12} V_{DD}^3 \left(1 - 2\frac{V_t}{V_{DD}}\right)^3 t_{rf}\ f_p\right) + \left(\frac{1}{2} C_L V_{DD}(V_{DD} - V_t)\right) \quad (7)$$

where P_{switch} is the switching power, α known as switching activity, C_L is the total load capacitances of all transistors, f_p known as the frequency of input signal, $P_{S.C}$ is the short circuit current, β is the current gain of the MOS transistor, and t_{rf} is the rising and falling time, P_{gltich} is the glitching power and V_{th} is the threshold voltage.

3. TECHNIQUES FOR DYNAMIC-POWER OPTIMIZATION

There are two ways to categorize power-optimization techniques: abstraction levels and power dissipation. The digital circuit goes through different design stages for different abstraction levels (levels). There are several levels of abstraction, including system, algorithmic, register-transfer logic (RTL), logic or gate, and transistor. At higher levels, such as the system, algorithmic and RTL levels (where the optimized power (either dynamic or static) is about 10-100%). According to the type of power dissipation, there are other classifications. Dynamic power is the goal of this paper. As a result, the spotlight will only be on techniques that use dynamic-power. In this section, a number of different techniques for reducing the dynamic power will be briefly described [24].

These techniques are: Operand isolator technique is a technique based on operating transformations in equivalent computational implementations at the algorithmic level. This technique is a way of saving power for data-path operators or combinational circuits that are not completely used in each clock cycle by design. These operators execute inefficient and redundant operations, which waste power. Eliminating the unwanted operations done by an isolated operator is the fundamental concept of isolating an operator. In other words, when no proper computation is performed, the logic blocks are shut off. Shutting off is achieved when the block output is not used by not allowing the inputs to toggle in clock cycles [25].

Pre-computation technique is a logic-optimization method at logic-level design [26] which tends to minimise logic transitions in combinational digital circuits by selectively pre-evaluating the output values of a combinational logic function only one clock cycle before they are required and then using pre-evaluated values to decrease internal switching activities in the next clock cycle [27]. Guarded-evaluation design techniques is a technique of gate-level abstraction-power optimization based on disabling the inputs of complex combinational circuits or data-path systems to reduce the transition when these inputs do not relate to the generation of output for a given input vector. In other words, if an outcome is not detected under such situations, i.e. if it has observable don't care (ODC) situations, then it is possible to insert transparent latches or floating gates at the required input [28]-[31].

4. DIVISION ALGORITHM

Arithmetic operations, such as addition, subtraction, multiplication and division, are fundamental building blocks in digital systems [7]. This paper is focused on the division operation. Division is considered an essential operation in many digital systems, such as signal processing, rendering systems, artificial intelligence, graphics compression [32]. Division is the most complicated and the highest-cost arithmetic operation. Unlike addition and multiplication, division does not possess associative or commutative properties [7], [33]. This makes it very difficult to implement division in digital systems.

Division algorithms can be classified into five classes: digit recurrence, functional iteration, very high radix, lookup table and variable latency [32]. This work studies digit-recurrence division. Despite being the first and oldest division class, digit-recurrence division is characterised by its high accuracy in comparison to the other classes. It calculates the quotient by iteratively subtracting the divisor from the dividend until the resulting quantity of the subtraction is less than the divisor.

Two algorithms come under this class: restoring and non-restoring division algorithms. These iterative algorithms are implemented sequentially and have a long latency, a significant area overhead and consume a large amount of power compared to the other mathematical operations [34]. Therefore, many reduction approaches can reduce the number of division cycles, thus reducing power dissipation [35].

4.1. Restoring division algorithm

A series of shifts and subtraction operations are performed by using the traditional restoring-division algorithm. In this algorithm, the partial remainder should always be positive. If the partial remainder is negative, the divisor is added back, which gives the operation the name 'restoring'[7], [32].

The restoring division consists of P, A, B and Q registers, which represent the n-bit accumulator register, the n-bit dividend register, the n-bit divisor register and the quotient register, respectively. The P and A registers will be concatenated to be the PA register (in which P is the high content and A is the low content), taking into account an additional bit that should be concatenated to the most significant bit (MSB) of the PA register for the shifting operation. The steps of restoring the division algorithm are described as follows [7], [36]:

1) Initialize the $P\ register$ with zeros, the $A\ register$ with the value of the dividend, the $B\ register$ with the value of divisor, and the counter is the number of bits in the dividend.
2) Shift the concatenated $PA\ register$ one-bit position to the left.
3) Perform the subtraction for the content of the$B\ register$ from the high content of the $PA\ register$.
4) Two cases are obtained from the subtraction :
   Case 1: If the result is positive (i.e., the MSB of $PA\ register\ = 0$), then the Q_0 is set to (1).
   Case 2: if the result is negative (i.e., the MSB of $PA\ register\ = 1$), then the Q_0 is set to (0).
5) Decrement the counter by one.
6) Repeat the steps from 2 to 5 until the counter become 0.
7) Finally, the quotient will be in the $Q\ register$, and the remainder will be in the $P\ register$.

4.2. Non-restoring division algorithm

Non-restoring division is an improved division in which the restoration of the partial remainder is eliminated. Therefore, after shifting, the arithmetic operation is either addition or subtraction, depending on the sign of the partial remainder. This algorithm is better and faster than restoring division, and this is due to only one decision per quotient and subtraction or addition per quotient bit. The main difference between the algorithms is how negative partial remainders are adjusted to give positive values. T Therefore, the sign of the partial remainder in restoring division can be either positive or negative. In contrast, the partial remainder in non-restoring division should always be positive [6], [34]. The procedure of non-restoring division algorithm has the same steps as in restoring division but with modification. The procedure of the non-restoring division algorithm has the same steps as restoring division but with a modification. This modification is applied in Step 4/Case 2. The non-restoring division algorithm is described as follows [7], [36]:

1) Initialize the $P\ register$ with zeros, the $A\ register$ with the value of the dividend, the $B\ register$ with the value of divisor, and the counter is the number of bits in the dividend.
2) Shift the concatenated $PA\ register$ one-bit position to the left.
3) Perform the subtraction for the content of the$B\ register$ from the high content of the $PA\ register$.
4) This step is modified from the restoring division algorithm. Once the sign of the $P\ register$ is checked, there can be two cases:
   Case 1: : If the result of the $P\ register$ is positive (i.e., the MSB of the $PA\ register\ = 0$), then shift the concatenated $PA\ register$ one bit to the left. The content of the $B\ register$ is subtracted from the high content of the $PA\ register$.
   Case 2: If the result of the$P\ register$ is negative (i.e., the MSB of the $PA\ register\ = 1$), then shift the concatenated $PA\ register$ one bit to the left. The content of the $B\ register$ is added to the high content of the $PA\ register$.
5) Check the sign of the $PA\ register$; there can be two cases:
   Case 1: If the $PA\ register$ is positive, then $Q_0 = 1$
   Case 2: If the $PA\ register$ is negative, then $Q_0 = 0$
6) Decrement the counter by one.
7) Repeat steps 2 to 6 until the counter becomes 0.
8) Check the sign of the $PA\ register$. If negative, add the divisor to the high content of the $PA\ register$.
9) Finally, the quotient will be in the $Q\ register$, and the remainder will be in the $P\ register$.

The essential differences between the restoring and the non-restoring division algorithms are the characteristics of the restoring division algorithm: it is similar to the long-division method, which resembles a standard pencil and paper algorithm. It restores a partial remainder while being worked on; it can require up to (2n+1) adders when performing division on 2n-bit numbers; it does not allow negative values of the partial remainder between two consecutives, and no error can be seen between successive iterations. While non-restoring division is similar to that of the restoring algorithm except for the restoring partial remainder, as it eliminates the restoring cycle, it requires only (n) adders to perform division on the 2n-bit number. This allows for positive and negative values of the partial remainder between two consecutive iterations; a small amount of error may occur during subsequent iterations [7].

5. PROPOSED DESIGNS OF DIVISION ALGORITHMS

Division algorithms are considered the core of many digital systems. In this research, restoring and non-restoring algorithms of the digit-recurrence class are targeted. These algorithms are sequentially implemented, and many reduction approaches can be used to reduce the number of division cycles. Because of the large area, long delay, and high power dissipation of these algorithms, the designers attempt to utilize the power reduction techniques to minimize the dissipated power. Six suggested system designs are realized in several algorithms in this section to determine which one has the lowest dynamic-power consumption.

Furthermore, new approaches are utilized to every algorithm in order to accomplish the best optimization of the dynamic power in the divider design. Two procedures are involved in reaching the optimal power divider: the first implements various structure designs, and the second applies the techniques of dynamic power reduction. A new VHDL approach is utilized to the suggested system designs to reduce the dynamic-power dissipation and reduce the execution time. This approach can be applied at the algorithmic and RTL levels. Non-restoring division with an external loop using VHDL is regarded as a reference system that has the maximum power consumption compared to the powers of the other suggested designs when power-optimization rates are evaluated.

All proposed division algorithms are implemented using Xilinx system generator (XSG) software, resulting in the configuration of MATLAB 2012a and the ISE 14.7 package. These algorithms have the same widths of dividend and divisor, which is 16 bits for each. The designs are verified and simulated by using the Xilinx Spartan 3A-3N/XC3S700a/-4/fg484 FPGA platform.

5.1. Restoring division algorithm based on hard FPGA blocks (proposed 1)

The proposed design of restoring division is implemented using Xilinx system generator blocks. These blocks can be extracted from Xilinx block-set libraries. The dvd (dividend) and dvr (divisor) are the inputs determined by the user. The dvd (A) and dvr (B) blocks represent GatewayIn inputs. Each block is adjusted with a fixed-point number with a width of 16-bit and 0-bit binary point (Fix_16_0), as shown in Figure 2.

Absolute and Absolute1 Xilinx blocks are used to take the absolute values for dividend and divisor, respectively. The size of (A) register block is (32-bit) and divided internally into two parts: the first one is (A1 (High)), representing that the high content should be initialized with zeros, and the other is (A2 (Low)), representing that the low content is initialized with the value of the dividend. The initialization of the A register takes place with the use of the bitbasher Xilinx block. The Mux block is used to choose between the dividend or shift operation through the Boolean output of the relational Xilinx block. This means when sel=0, the dividend will pass through, and when sel=1, the shifting process will occur.

The output of the A register block will be driven to the bitbasher1 block. The bitbasher1 block is used to extract the high content of the A register (i.e., A1 (High)) and then subtracted from the absolute value of the dividend, and this step is represented as (T=A1-B) in the algorithm shown in Figure 2. The output of the addsub block has two paths: one goes to the bitbasher2 block, and the other goes to the bitbasher3 block. bitbasher3 is used to concatenate the low content of the A register (i.e., A2 (Low)) with the output (T) of the addsub block. This step represents the restoring operation; thus, this algorithm is called restoring division. Q[0] is extracted by taking the complement of the MSB of output (T), in which Q[0] is used to select between passing the content of register A when sel = 0 or passing the restored A register when sel=1.

After that, the output of Mux1 will be shifted one-bit position to the left, and the counter is decremented by one. By initializing the Q register block with zeros, the BitBasher4 block concatenates the Q[0] and the rest of the content of the Q register. The output of this register will store the value of the remainder and the quotient. In the end, the result will be shown on the Display block. This system performs 34 cycles to obtain the result of the division.

5.2. Restoring division algorithm with external loop based on the VHDL approach (proposed 2)

The proposed design of the restoring division algorithm is implemented using the VHDL code. The VHDL code is written and verified by the ISE14.7 package and imported to XSG using the Xilinx black box. In this design, the pulse generator simulink block is used to generate the external loop.

The external loop is considered the external clock used to operate the division operation correctly, as shown in Figure 3. The flowchart of the proposed design that illustrated Figure 4 shows the procedure of the restoring division algorithm. The first step of this algorithm is initializing the high content of the A register, and the content of the Q register with zeros (i.e., A1[31:16]=0 and Q[31:0]=0), and the low content of the A register and B register with the absolute values of the dividend and the divisor, respectively (i.e., A[15:0]=|dvd| and B[15:0]=|dvr|). After the initialization mentioned above, the CLK (clock edge) is checked. If the CLK - positive edge, then the A register and the Q register are shifted left one bit, the divisor subtracted from the high content of the A register (i.e., T=A1[31:16]-B[15:0]), and Q[0] is extracted by taking the complement of the MSB of the T register.

There are two possibilities for Q[0], if Q[0]=1, then restore the content of the T register to the high content of the A register (i.e., A[31:16]=T), else do nothing and proceed. After the counter is incremented by one, check the counter if $C < 64$, then repeat the steps of left shifting. Else, proceed to the next step, which is checking the sign bit. If S=1, then take 2's complement of the Q register; otherwise, end the algorithm. Due to the external loop, each step of this algorithm is performed in one cycle. Therefore, the result is obtained after 64 cycles. The proposed restoring division algorithm flowchart is illustrated in Figure 4.

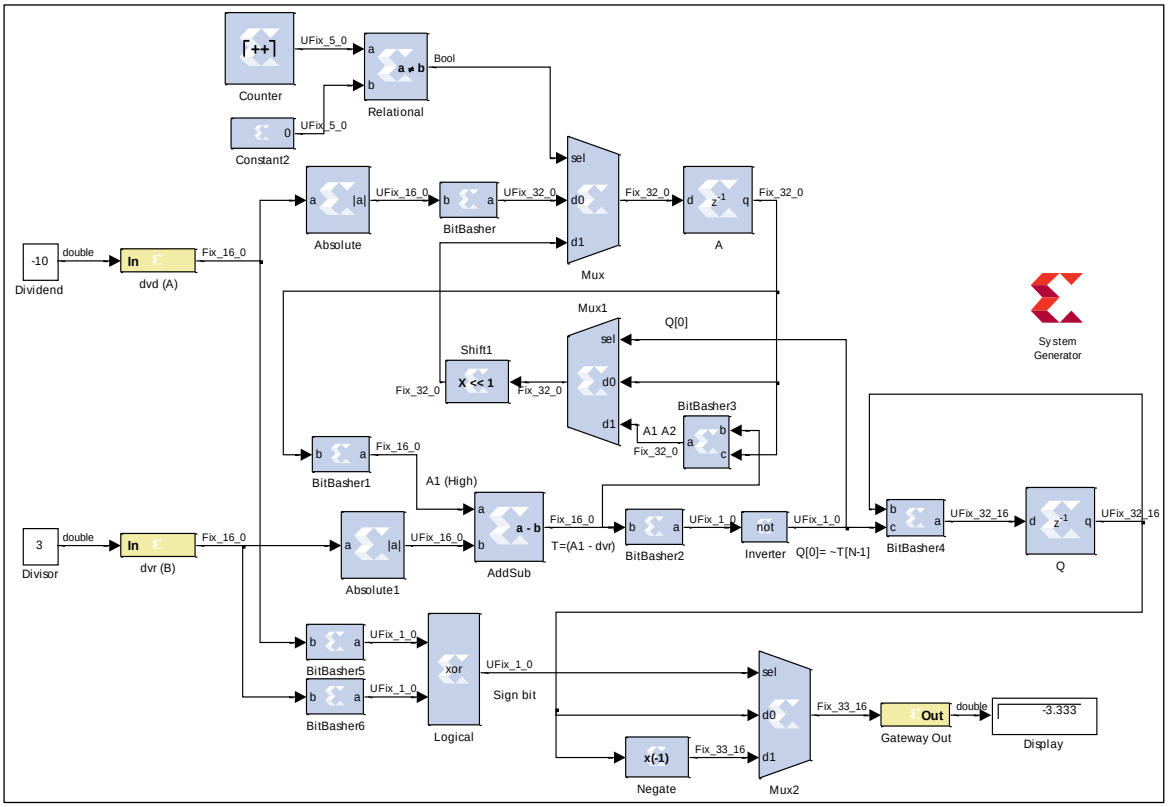

Figure 2. Proposed design of restoring division based on hard FPGA blocks

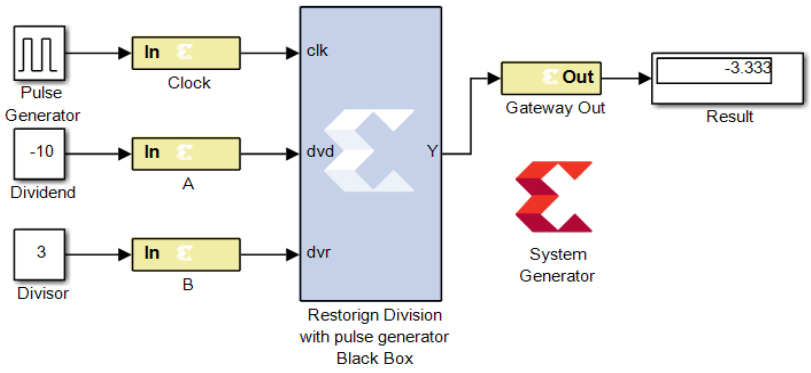

Figure 3. Restoring division algorithm with external loop using VHDL approach

5.3. Restoring division algorithm with internal loop based on the VHDL approach (proposed 3)

In this section, the proposed design is based on the same algorithm that used in the earlier design. The primer difference of this design is the use of the internal loop instead of using the external loop, which means the pulse generator block is eliminated, and the design will depend only on the internal clock. The advantage of this internal clock it can perform all the steps of the restoring division algorithm in one cycle instead of performing each step in one cycle as in the previous design.

Figure 5 shows the flowchart of the proposed design. As demonstrated from the figure, the same procedure of the previous algorithm is applied to this design. But with one difference, which is the decision box for checking the clock edge, is eliminated. This decision box represents the behavioural operation of the pulse generator block (the external loop). Instead of that, the left-shifting operation for the registers is directly performed.

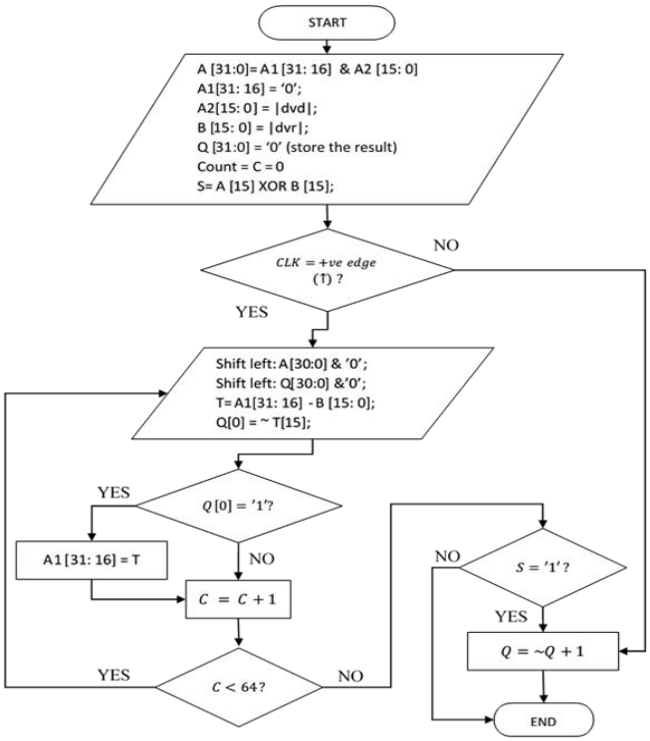

Figure 4. Flowchart of the proposed restoring algorithm with external loop based in the VHDL approach

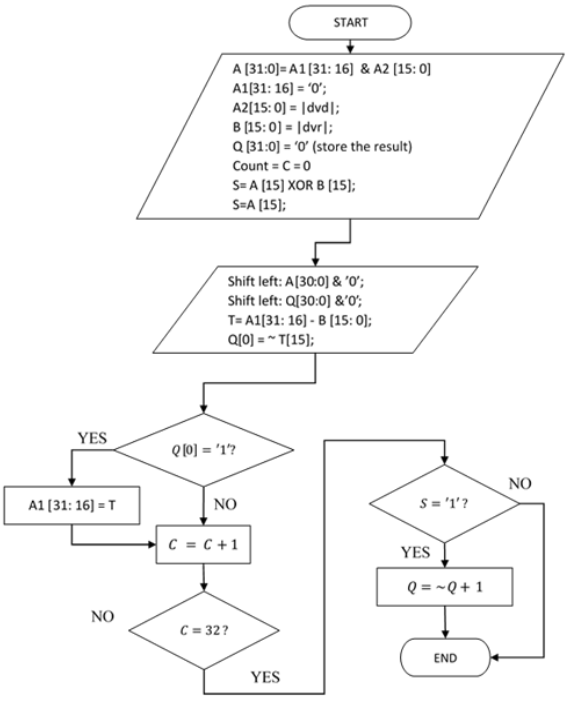

Figure 5. Flowchart of the proposed restoring division algorithm with internal loop based on the VHDL approach

5.4. Non-restoring division algorithm based on hard FPGA blocks (proposed 4)

The proposed non-restoring division algorithm is also implemented using XSG blocks, as in the proposed design in section 0, but in this design, there are some modifications to restoring the algorithm. These modifications start after the Xilinx bitbasher1 and end before the Xilinx bitbasher5 block, as shown in Figure 6.

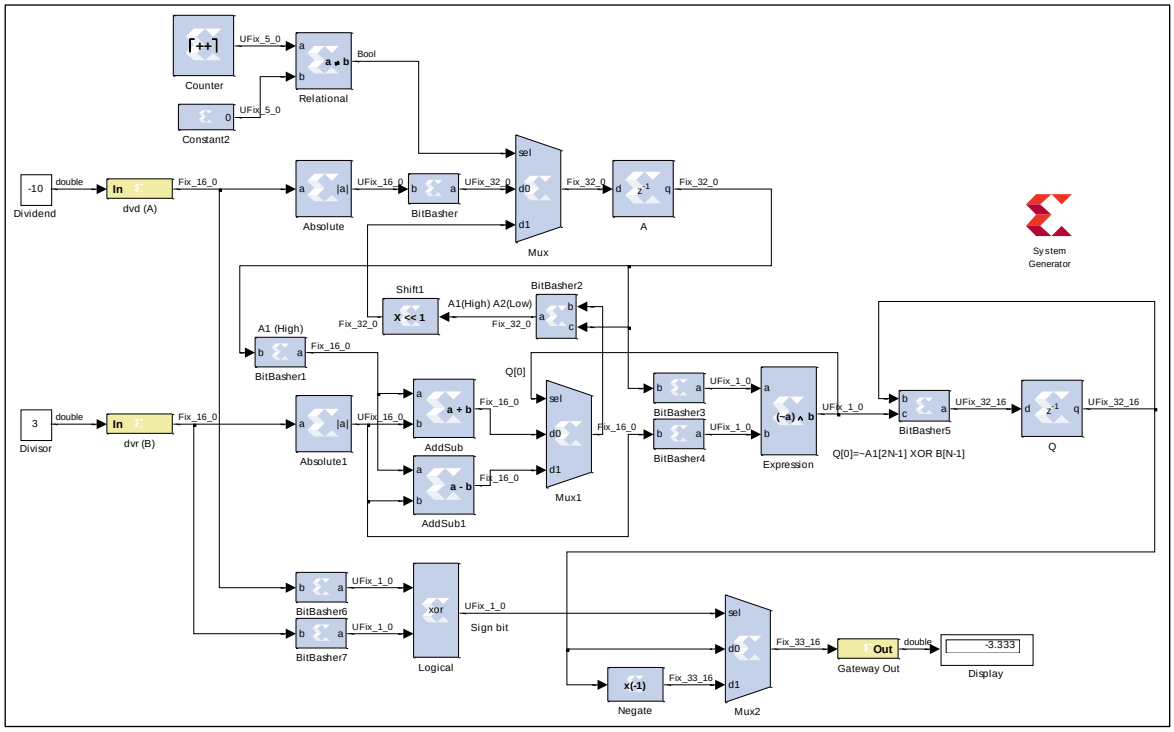

Figure 6. Non-restoring division algorithm based on hard FPGA blocks

After the register (A) block has been initialised with a dividend value for the content of A2 (low) and zeros for the content of A1 (high) content, A1 (high) is subtracted from or add to the absolute value of the dividend by the addsub and the Addsub1 blocks, respectively. The Xilinx Mux1 block is used to select between the addition or subtraction operation depending on Q[0]. When Q[0] = 0, then the A1 (high) is added to the divisor. And when Q[0] = 1, then A1 is subtracted.

The output of the Mux1 block, which represents the new content of A1 (high), will be concatenated with the low content of the register (A) block (i.e. A2 (low)) and then shift the contents of the (A) register one-bit position to the left. Likewise, the counter is decremented by one. Bitbasher3 and bitbasher4 blocks are used to slice the MSBs of the register (A) block and divisor, respectively. To achieve the conditions for the addition or subtraction operation, which will represent the least significant bit of the Q register, Q[0], the Expression Xilinx block is used for this step of the algorithm. In this step, the value of the register (A) block is not restored, as in the restoring algorithm, which is the significant difference between the two algorithms. The Xilinx bitbasher5 block is used to concatenate the Q[0] with the other initialised bits of the Q register since this operation is represented by shifting the Q register to the left. This algorithm takes 35 cycles to obtain the final result of the division.

5.5. Non-restoring algorithm with external loop based on the VHDL approach (proposed 5)

The suggested system design of the non-restoring division algorithm is implemented using VHDL code. The VHDL code is written and verified using the ISE 14.7 package and exported to system generator by using Xilinx black box block. In this design, a pulse generator is used as the main clock of the system to run and process the division operation correctly. The procedure of the suggested Non-restoring algorithm flowchart is shown in Figure 5. In this algorithm, the pulse generator is used in the suggested model of restoring division mentioned in section (0). The result of this design can be obtained after 32 cycles. This extensive execution time is due to the counter's count, which is equal to the sum of the divisor and dividend bits, as well as clock initialization. Alternatively, counter=2×bits of dividend. Where in this case the number of bits=2×16=32=the number of counts.

5.6. Non-restoring algorithm with internal loop based on the VHDL approach (proposed 6)

The VHDL approach is used to implement the non-restoring division in this part. The verification of the VHDL code is done using the ISE 14.7 package. This algorithm is identical to the previous design, except the pulse generator is eliminated. The internal clock or internal loop is used to run this division operation. The demonstration of the suggested non-restoring algorithm is given in Figure 8. As can be noticed from the flow char, after the step of determining the inputs and initializing the registers, the left shifting operation is directly performed. Alternatively, the decision box that detects the clock edge in the previous design is eliminated. Therefore, this step is considered significantly different from the earlier.

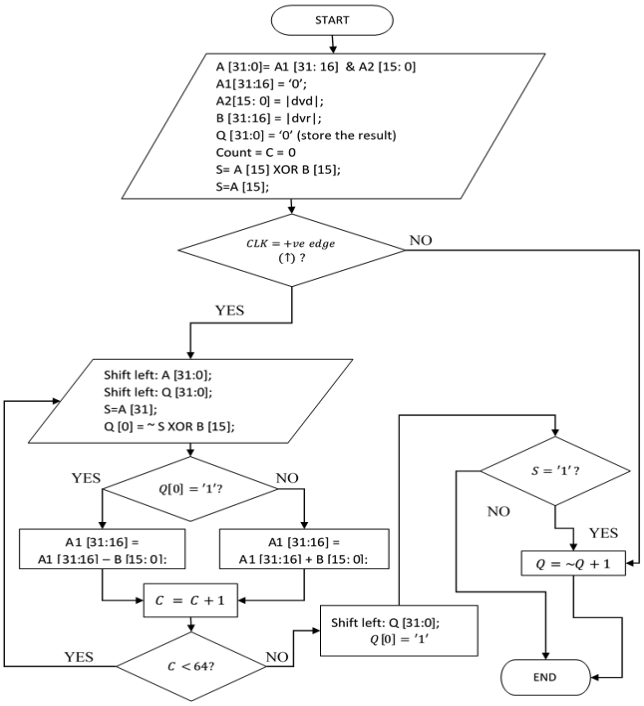

Figure 7. Flowchart of the proposed non-restoring algorithm with external loop

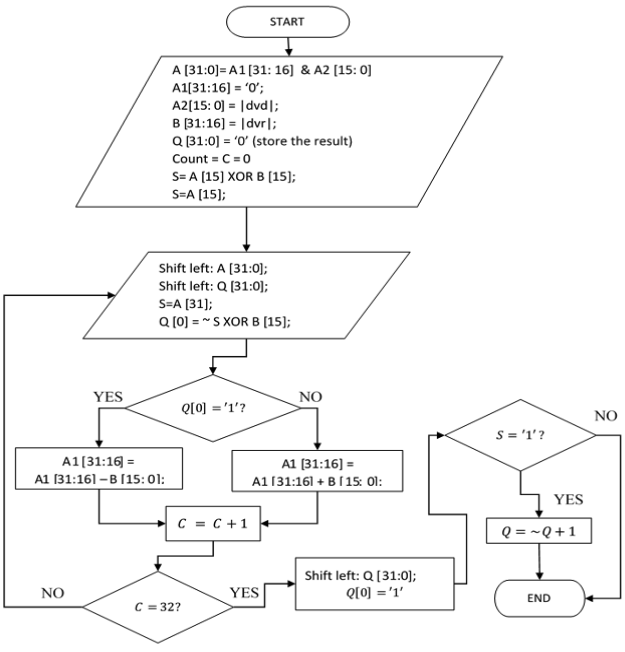

Figure 8. Flowchart of proposed non-restoring algorithm with internal loop

6. SIMULATION RESULTS AND DISCUSSION

Six suggested system designs of division algorithms are realized using various techniques, In this work. Each divider has dividend and divisor widths of 16 bits. All the suggested systems are verified using Spartan 3A of the Xilinx FPGA kit. Some of the suggested system designs are realized using XSG block sets obtained by the configuration of MATLAB R2012a and ISE 14.7 package, and the other systems are realized using the VHDL technique.

The VHDL technique can be regarded as a new approach for power optimization, mainly for dynamic-power optimization, which is explained in the discussion paragraph. Dynamic power is the scope of this research. Xilinx power analyser software is used to estimate the performances of the proposed designs in terms of dynamic power, speed and utilised area.

Two comparisons have been prepared regarding to the dynamic power analysis, in this work. The first comparision compares the highest power for the proposed non-restoring division with an external loop using VHDL with the other five proposed designs, as shown in Table 1. The second comparision is between the related works and the best optimal power divider, as listed in Table 2.

In terms of dynamic power, a comparison is made between the proposed non-restoring division design based on the VHDL approach, which is considered a reference design, and the other five proposed designs, as shown in Table 1. This comparison shows the dynamic-power optimization rates are as follows: 8.25% of restoring division with an external loop based on the VHDL approach; 50.58% of non-restoring division based on hard FPGA blocks; 58.42% of restoring division based on hard FPGA blocks. The restoring division and non-restoring division with an internal loop based on VHDL approaches have introduced the highest rates of 91.11% and 93.67%, respectively. The resource utilization of the six proposed designs is depicted in Figure 9. As demonstrated from this figure, LUTs and occupied slices represent the significant number of elements consumed by the restoring and non-restoring divisions with an internal loop based on VHDL approaches. In addition, it can be noted that there are zero flip-flops in either design. In contrast, the highest number of flip-flops is introduced in the other four proposed designs, while the number of LUTs and slices are considered small relative to the two proposed designs mentioned above.

Table 1. Power dissipation comparison of the proposed designs

Type of division	Dynamic power (mW)	Static power (mW)	Total power (mW)	Rate of dynamic power
proposed 1	143.23	32.31	175.54	-
proposed 2	131.42	32.24	163.66	8.25 %
proposed 3	70.78	31.90	102.68	50.58 %
proposed 4	59.56	31.84	91.40	58.42 %
proposed 5	12.74	31.59	44.32	91.11 %
proposed 6	9.07	31.57	40.64	93.67 %

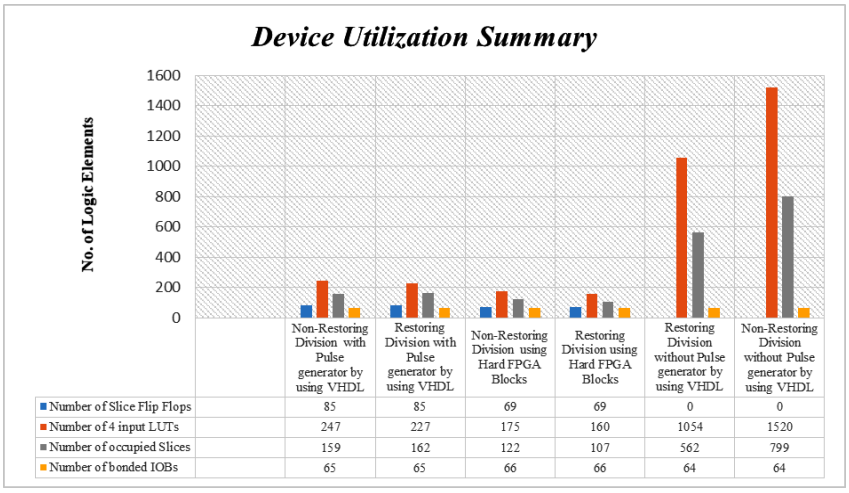

Figure 9. Device utilization summary of the proposed division algorithm designs

Despite the proposed design, non-restoring division with an external loop based on the VHDL approach has less resource utilization in terms of the number of LUTs (247), slices (159) and I/OB (input/output bounded) buffers blocks (65), as illustrated in Figure 9. Simultaneously, the design had the highest dynamic-power dissipation. The high consumption is due to two reasons: first, the design has the highest flip-flops (85), which is

the primary reason for the increase in switching activities, causing the dynamic power to dissipate according to the dynamic-power equation. Second, using an external loop increases the execution time to 64 cycles in this proposed design; this leads to an increase in critical-path delays, thus increasing the power dissipation.

While the proposed design of non-restoring division with an internal loop based on the VHDL approach utilises high-logic resources, it consumes less dynamic power than the others. This reduction in power is due to the VHDL approach, which transforms the suggested system design into basic element components. Furthermore, the execution time for this proposed division algorithm is less than one cycle. This means there is no latency (delay), which leads to minimization of the critical paths. Further, the placing and routing phases handle internal optimization. The switching activities are reduced or eliminated in this design because there are no flip-flops. This is considered an additional reason why the dynamic power is reduced.

The second comparison of this work is illustrated in Table 2. This table shows the comparisons between the proposed design, with the best dynamic-power optimization, and the previous works. Different techniques and approaches to reduce dynamic-power dissipation from the earlier works are presented in this table. Compared to the previous works, the best result of dynamic-power optimization (93.66%) is obtained when using an internal-loop division based on the VHDL approach.

Table 2. Comparison of the proposed design with the related works

Type of division	Bits of dividend	Bits of divisor	Rate of dynamic power
[8]	32	32	82.9 %
[9]	32	16	70 %
[10]	8	8	67 %
[11]	8	4	52 %
[12]	8	4	27 %
[13]	32	16	90 %
proposed 6	16	16	93.66 %

7. CONCLUSION

In this work, various division algorithms of 16 bit by 16 bit have been suggested, where two of them are realized using Hard FPGA blocks, and four designs were executed using the VHDL technique. The performance analysis regarding to the dynamic power dissipation shows that the suggested system designs that contain flip-flops have the highest dynamic power. At the same time, the proposed designs with zero flip-flops and a lot of LUTs and slices have the lowest dynamic power. This reduction is due to decreasing or eliminating the switching activities in these designs and low critical path delays. From the experimental results, the efficient technique to obtain the higher power savings is using the VHDL approach, where the power saving of (93.66%) for the dynamic power is found in the proposed design of non-restoring division with an internal loop. In which it is the highest dynamic power optimization relative to the previous works.

REFERENCES

[1] V. Natarajan, A. K. Nagarajan, N. Pandian and V. G. Savithri, "Low Power Design Methodology," *IntechOpen*, 2018, doi: 10.5772/intechopen.73729.

[2] A. P. Kumar, B. Aditya, G. Sony, C. Prasanna and A. Satish, "Estimation of power and delay in CMOS circuits using LCT," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 14, pp. 990-998, 2019, doi: 10.11591/ijeecs.v14.i2.pp990-998.

[3] G. E. Moore, "Cramming more components onto integrated circuits," *IEEE Solid-State Circuits Soc. Newsl.*, vol. 11, no. 3, pp. 33-35, 2009, doi: 10.1109/n-ssc.2006.4785860.

[4] S. P. Mohanty, N. Ranganathan, E. Kougianos and P. Patra, "Low-power high-level synthesis for nanoscale CMOS circuits," *Springer Science & Business Media*, 2008, doi: 10.1007/978-0-387-76474-0.

[5] A. K. Dwivedi and A. Islam, "Nonvolatile and robust design of content addressable memory cell using magnetic tunnel junction at nanoscale regime," *IEEE Trans. Magn.*, vol. 51, no. 12, pp. 1-13, 2015, doi: 10.1109/TMAG.2015.2454477.

[6] T. L. Floyd, "*Digital Fundamentals*," 11th edition, Pearson Education India, 2010.

[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] T. T. Hasan, M. H. Jasim and I. A. Hashim, "FPGA Design and Hardware Implementation of Heart Disease Diagnosis System Based on NVG-RAM Classifier," in *2018 Third Scientific Conference of Electrical Engineering (SCEE)*, 2018, pp. 33-38, doi: 10.1109/SCEE.2018.8684125.

[9] N. Aggarwal, K. Asooja, S. S. Verma and S. Negi, "An improvement in the restoring division algorithm (needy restoring division algorithm)," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 246-249, doi: 10.1109/ICCSIT.2009.5234956.

[10] K. Hill and J. E. Stine, "An Efficient Implementation of Radix-4 Integer Division Using Scaling," in *2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2020, pp. 1092-1095, doi: 10.1109/MWSCAS48704.2020.9184631.

[11] S. Hashemi, R. I. Bahar and S. Reda, "A low-power dynamic divider for approximate applications," in *Proceedings - Design Automation Conference*, Jun. 2016, no. 105, doi: 10.1145/2897937.2897965.
[12] J. Liu, M. Chang and C. K. Cheng, "An iterative division algorithm for FPGAs," in *ACM/SIGDA International Symposium on Field Programmable Gate Arrays - FPGA*, 2006, pp. 83-89, doi: 10.1145/1117201.1117213.
[13] D. Kishor and V. Bhaaskaran, "Low power divider using vedic mathematics," in *2014 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2014, pp. 575-580, doi: 10.1109/ICACCI.2014.6968436.
[14] R. Senapati, B. Bhoi and M. P.- On, "Novel binary divider architecture for high speed VLSI applications," in *2013 IEEE Conference on Information & Communication Technologies*, 2013, pp. 675-679, doi: 10.1109/CICT.2013.6558180.
[15] S. BhanuTej, "Vedic divider-A high performance computing algorithm for VLSI applications," in *2013 International conference on Circuits, Controls and Communications (CCUBE)*, 2013, pp. 1-5, doi: 10.1109/CCUBE.2013.6718577.
[16] K. Paulsson, M. Hübner and J. Becker, "Dynamic power optimization by exploiting self-reconfiguration in Xilinx Spartan 3-based systems," *Microprocess. Microsyst.*, vol. 33, no. 1, pp. 46-52, 2009, doi: 10.1016/j.micpro.2008.08.006.
[17] C. V. S. Chaitanya, C. Sundaresan, P. R. Venkateswaran and K. Prasad, "Asic design of low power-delay product carry pre-computation based multiplier," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 13, no. 2, pp. 845-852, 2019, doi: 10.11591/ijeecs.v13.i2.pp845-852.
[18] A. Muttaqin, Z. Abidin, R. A. Setyawan and I. A. Zahra, "Development of advanced automated test equipment for digital system by using FPGA," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 15, no. 2, pp. 661-670, 2019, doi: 10.11591/ijeecs.v15.i2.pp661-670.
[19] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput. Des. Integr. circuits Syst.*, vol. 26, no. 2, pp. 203-215, 2007, doi: 10.1109/TCAD.2006.884574.
[20] A. Pal, "Sources of Power Dissipation," in *Low-Power VLSI Circuits and Systems*, pp.141-173, Springer, New Delhi, 2015, doi: 10.1007/978-81-322-1937-8_6.
[21] N. P. Kumar, B. S. Charles and V. Sumalatha, "A Review on Leakage Power Reduction Techniques at 45nm Technology," *Mater. Today Proc.*, vol. 2, no. 9, pp. 4569-4574, 2015, doi: 10.1016/j.matpr.2015.10.074.
[22] R. Y. Reetu, "Dynamic power reduction of VLSI circuits: a review," in *Int. J. Adv. Res. Electron. Commun. Eng*, vol. 7, no. 3, pp. 245-259, 2018.
[23] S. Alluri, B. R. Naik, N. S. S. Reddy and M. V. Ramanaiah, "Performance Analysis of VLSI Circuits in 45 nm Technology," in *Advances in Decision Sciences, Image Processing, Security and Computer Vision*, Springer, 2020, pp. 281-289, doi: 10.1007/978-3-030-24318-0_34.
[24] M. Arora, "The art of hardware architecture: Design methods and techniques for digital circuits," *Springer Science & Business Media*, 2011, doi: 10.1007/978-1-4614-0397-5.
[25] A. Pal, "Low-Power VLSI circuits and systems," *Springer*, 2014, doi: 10.1007/978-81-322-1937-8.
[26] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," in *Proceedings of the 32nd annual ACM/IEEE Design Automation Conference*, 1995, pp. 242-247, doi: 10.1145/217474.217536.
[27] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh and M. Papaefthymiou, "Precomputation-based sequential logic optimization for low power," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 2, no. 4, pp. 426-436, 1994, doi: 10.1109/92.335011.
[28] S. Mitra and D. Das, "A Comprehensive Review of Applications of Don't Care Bit Filling Techniques for Test Power Reduction in Digital VLSI Systems," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 12, no. 3, pp. 941-949, 2018.
[29] E. Macii, M. Pedram, and F. Somenzi, "High-level power modeling, estimation, and optimization," *IEEE Trans. Comput. Des. Integr. circuits Syst.*, vol. 17, no. 11, pp. 1061-1079, 1998, doi: 10.1109/43.736181.
[30] S. Ravi, G. Lakshminarayana and N. K. Jha, "TAO: Regular expression based high-level testability analysis and optimization," in *Proceedings International Test Conference 1998 (IEEE Cat. No. 98CH36270)*, 1998, pp. 331-340, doi: 10.1109/TEST.1998.743171.
[31] V. Tiwari, S. Malik and P. Ashar, "Guarded evaluation: Pushing power management to logic synthesis/design," *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 17, no. 10, pp. 1051-1060, 1998, doi: 10.1109/43.728924.
[32] U. S. Patankar, M. E. Flores and A. Koel, "Division algorithms-From Past to Present Chance to Improve Area Time and Complexity for Digital Applications," in *2020 IEEE Latin America Electron Devices Conference (LAEDC)*, 2020, pp. 1-4, doi: 10.1109/LAEDC49063.2020.9073050.
[33] E. Matthews, A. Lu, Z. Fang and L. Shannon, "Rethinking integer divider design for FPGA-based soft-processors," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2019, pp. 289-297, doi: 10.1109/FCCM.2019.00046.
[34] K. Kataria and S. Patel, "Design Of High Performance Digital Divider," in *2020 IEEE VLSI DEVICE CIRCUIT AND SYSTEM (VLSI DCS)*, 2020, pp. 1-6, doi: 10.1109/VLSIDCS47293.2020.9179903.
[35] A. Parashar, G. Aggarwal, R. Dang, P. Dalmia and N. Pandey, "Fast Combinational Architecture for a Vedic Divider," in *2017 14th IEEE India Council International Conference (INDICON)*, 2017, pp. 1-5, doi: 10.1109/INDICON.2017.8487598.
[36] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, New York, 2010.