An FPGA-Based Divider Circuit Using Simulated Annealing Algorithm

Zarrin Tasnim Sworna*, Mubin Ul Haque† and Shahedur Rahman‡
*Department of Computer Science and Engineering, University of Dhaka, Dhaka-1000, Bangladesh.
‡Computer Science Department, School of Science and Technology, Middlesex University, London, UK.
Email: *zarrin@cse.du.ac.bd, †mubin@cse.du.ac.bd, ‡S.Rahman@mdx.ac.uk

***Abstract*—Division is considered as the slowest and most difficult operation among four basic operations in microprocessors. This paper proposes a unique division algorithm using a new approach of simulated annealing algorithm. A heuristic function is proposed to determine the global and local optimum value, whereas the conventional approaches use random values to reach the target value. In addition, a new temperature schedule is introduced for faster computation of global maxima/minima. The proposed simulated annealing performs better than the best known existing method of simulated annealing algorithm for smooth energy landscape due to the introduction of a new goal-based temperature. Thus, the proposed division algorithm computes the current partial remainder and quotient bits simultaneously per iteration which reduces the delay of the proposed divider circuit significantly. Moreover, the proposed divider circuit requires only two operations per iteration, whereas the exiting best one requires three operations per iteration. The proposed divider circuit is coded in VHDL and implemented in a Virtex-6 platform targeting XC6VLX75T Xilinx FPGA with a -3 speed grade by using ISE 13.1. The proposed divider circuit achieves an improvement of 36.17% and 44.67% respectively in terms of LUTs and delay factor for a 256 by 128 bit division over the best known contemporary FPGA-based divider circuit. It can be used into the designs of arithmetic logic unit, image processing and robotics system. The experimental result indicates that the divider takes fewer resources, and its performance is steady and reliable.**

Keywords: Division, Modified Simulated Annealing Algorithm, Look-Up Table, Field Programmable Gate Arrays

I. Introduction

Among the basic operations division being the slowest operation on a modern microprocessor, is the prerequisite for faster mathematical and computational operation in processor [1]. In this paper, a divergent approach for division algorithm is proposed using modified simulated annealing. As simulated annealing guarantees a statistically optimal solution for arbitrary problem, proposed algorithm provides solution in optimum time.

The advancement in Field Programmable Gate Array (FPGA) technology has emerged a new horizon of technical progress due to its long time availability, rapid prototyping capability, reliability and hardware parallelism [2] [12]. FPGA-based technology being a part of today's advancement, an FPGA-based divider circuit is proposed with reduced number of Look-Up Tables (LUTs) and slices.

Three main contributions are addressed in this paper:

1) *A new approach of simulated annealing algorithm has been introduced which can be substantiated in any scientific and arithmetic computation.*
2) *A new division algorithm using the proposed simulated annealing technique has been presented with less number of iterations as well as number of operations per iterations than the best known exiting approaches.*
3) *Finally, an FPGA-Based Divider Circuit has been elucidated requiring optimum number of LUTs, Slices and delay.*

The organization of this paper is as follows: In the next section, basic definitions and properties of simulated annealing and the earlier approaches of FPGA-based dividers along with their limitations are described. Section III introduces the proposed division algorithm along with the divider circuit. In Section IV, analyses of the performance of the proposed algorithm and circuits are demonstrated. Lastly, the paper is concluded in Section V.

II. Preliminaries and Prior Works

This section outlines the properties and shortcomings of the simulated annealing algorithm. Besides, analysis of the existing FPGA-based dividers are also discussed later.

A. Simulated Annealing

Simulated Annealing (SA) is a technique to find a good solution to an optimization problem by trying random variations of the current solution [3] [4]. Let us consider a positive real-valued temperature T which is used to control how many worsening steps are accepted. Suppose A is the current assignment of a value to each variable. Suppose that $h(A)$ is the evaluation of assignment A to be minimized. For solving constraints, h is typically the number of conflicts. Simulated annealing selects a neighbor at random, which gives a new assignment A'. If $h(A') \leq h(A)$, it accepts the assignment and A' becomes the new assignment. Otherwise, the assignment is only accepted randomly with probability following Equation 1:

$$e^{(h(A)-h(A'))/T} \tag{1}$$

Thus, if $h(A')$ is close to $h(A)$, the assignment is more likely to be accepted. If the temperature is high, the exponent

TABLE I: Probability of simulated annealing accepting worsening steps

Temperature	Probability of acceptance		
	1-worse	2-worse	3-worse
10	0.9	0.82	0.74
1	0.37	0.14	0.05
0.25	0.018	0.0003	0.000006
0.1	0.00005	2×10^{-9}	9×10^{-14}

will be close to zero, and so the probability will be close to 1. As the temperature approaches zero, the exponent approaches infinity, and the probability approaches zero.

Fig. 1 shows the probability of accepting worsening steps at different temperatures. In this figure, k-worse means that $h(A') - h(A)$=k. If the temperature is high, as in the T=10, the algorithm tends to accept steps that only worsen a small amount; it does not tend to accept very large worsening steps. There is a slight preference for improving steps. As the temperature is reduced (e.g., when T=1), worsening steps, although still possible, become much less likely. When the temperature is low (e.g., T=0.1), it is very rare that it chooses a worsening step as shown in Table I.

B. Shortcomings of Existing Simulated Annealing [3] [4]

The disadvantages of existing simulated annealing is listed below:

- Heuristic methods, which are problem-specific or take advantage of extra information about the system, will often be better than general methods.
- The method itself cannot determine whether it has found an optimal solution. Some other method (e.g. branch and bound) is required to do this.
- Repeatedly annealing with a $1/\log k$ schedule is very slow, especially if the cost function is expensive to compute.
- For problems where the energy landscape is smooth, or there are few local minima/maxima, SA is overkill and consumes more computation time.

To solve the noted existing problems of simulated annealing algorithm, modification of the algorithm is proposed in Section III-A.

C. Analysis of Existing FPGA-based Divider Circuits

Division algorithms can be categorized into two parts: 1) Digit recurrence and 2) Digit convergence. Digit recurrence algorithms are considered suitable than digit convergence method for hardware implementation [1]. Researchers also showed that non restoring algorithms are simpler and takes small area than restoring and Sweeny, Robertson, Tocher (SRT) method [1]. Authors in [1] proposed a modified non restoring algorithm by reducing one unit delay of the n-bit multiplexer per iteration, where n=number of bits in dividend and removes the error of quotient bits in conventional non restoring algorithm. Their proposed design reduced the delay by 21% but area had been increased by 70% which is a major drawback. However, the most recent improvement over the non-restoring division algorithm has been demonstrated in [5]. Their proposed modified and improved non-restoring algorithm reduced the number of bits in shift operation during non restoring operations. The upper half of the dividend is shifted instead of whole dividend. The least significant bit of the shifted number is set to the value in lower half of the dividend depending on the iteration. Though the number of shifted operations are reduced to half, it requires three operations of addition, subtraction and multiplexing and suffer a significant amount of delay. A new division approach based on Vedic mathematics has been proposed for ultra-high-speed by the authors in [10]. Their proposed approach was applied in (32 $\div$ 16) division and it was found that it involves minimum memory space of the processor as compared with the conventional [6] [7] [11] methods. Authors in paper [9] and [8] had introduced the power model for the FPGA-based dividers. Their performance showed a significant improvement in terms of estimated power consumption over the Xillinx Core Divider 3.0. However, their design require a significant amount of delay, hardware complexities and more operations in trade of low power consumption.

III. Proposed Method

In this section, firstly, a new approach of simulated annealing is presented. Then, the advantages of the proposed simulated annealing algorithm is described. The proposed algorithm for simulated annealing is presented later on. Finally, a division algorithm using the proposed simulated annealing along with the FPGA-based divider circuit are presented.

A. A New Approach of Simulated Annealing

In this paper, a new approach of simulated annealing is used for division operation. The distinguishes between the conventional simulated annealing [3] [4] and the proposed simulated annealing are as follows:

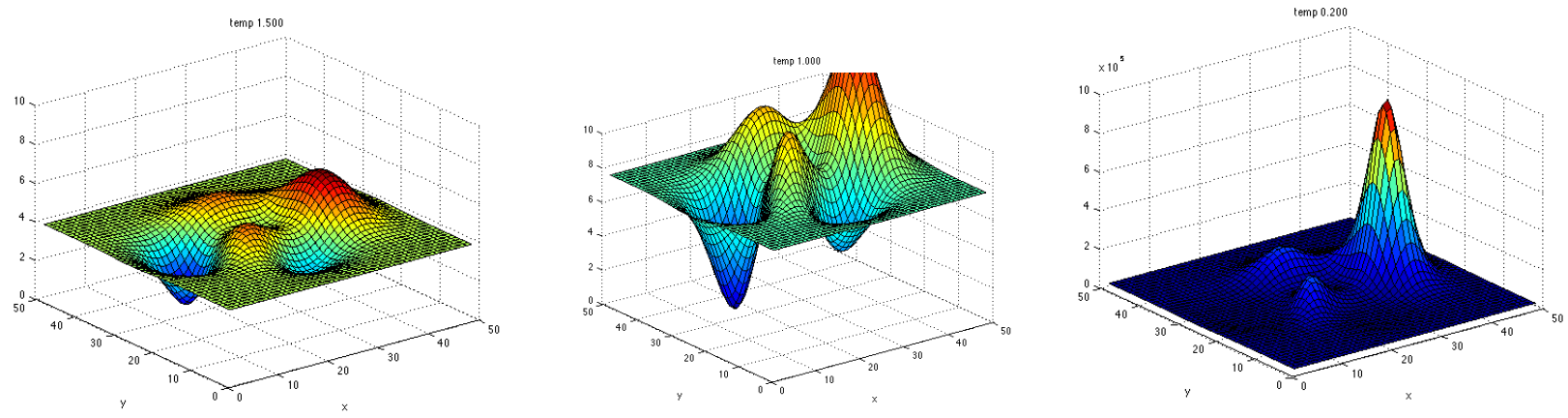
- *The proposed method of simulated annealing models the physical process of heating a material and then slowly lowering the temperature to decrease the defects by filling the remaining pits.*
- *The goal is to reach a target firstly by choosing a global optimal value and with the reduction of temperature, the local optimal values are chosen to fill the gap between the target and the current value.*
- *The temperature schedule might be non-increasing while in conventional approach, it is always decreasing.*
- *A heuristic function is used to find the global and local optimal, whereas the conventional approach chooses random value.*

The details of the proposed simulated annealing is exhibited in Section III-C.

B. Advantages of the Proposed Simulated Annealing

The advantages of the new approach of simulated annealing over the existing [3] [4] are specified below with corresponding reasoning.

- Heuristic methods, which are problem-specific or take advantage of extra information about the system, will often be better than general methods [3]. So in new simulated



(a) At Temperature 1.5 (b) At Temperature 1.00 (c) At Temperature 0.20

Fig. 1: The Probabilistic Acceptance of Optimal of Simulated Annealing at Different Temperatures.

annealing, a heuristic function is incorporated where the global and local optimal values will be determined by using the problem specific heuristic function.

- The method can guarantee to reach the optimal solution as the choices of global optimal values are selected at the first step and then, to fill the defected emptiness the local optimal values are chosen to fill the target.
- Temperature being unaware of the problem knowledge is tough to determine in simulated annealing on which the efficiency of the algorithm depends, whereas in the proposed modified simulated annealing, temperature is problem goal-specific. As in simulated annealing, it may be the case that, in spite of the optimal values are found (since the temperature is not diminished), it keeps moving back and forth unnecessarily or may be, though the optimal result is not found yet (since the temperature is cooled to zero), the simulation is to be terminated without finding the optimal result. So, the problem goal-based temperature would solve the noted two deficits of simulated annealing.
- For problems where the energy landscape is smooth, or there are few local minima/maxima, proposed simulated annealing will provide faster results as there will be few defects to fill with local minima/maxima.

C. Proposed Algorithm for the New Approach of Simulated Annealing

The distinguishes of the proposed simulated annealing with conventional simulated annealing is noted in Section III-A. The following pseudocode of Algorithm 1 presents the proposed simulated annealing. It starts from a state s with a goal, G where s is obtained using a heuristic function to find global optimal value, $heuGlobalOpt()$. The algorithm continues until a state with an energy of e_{min} or less is found. In the process, the call $heuLocalOpt()$ will generate a local maximal number using a heuristic function to fill the distance from the goal, G. The probability of acceptance is 1 if current condition satisfies the requirement else it is zero.

ALGORITHM 1: Proposed Algorithm for Simulated Annealing

Input: goal, G;
Output: The final state s;
$s \leftarrow heuGlobalOpt()$;
repeat
  $T \leftarrow temperature(r)$;
  $s_{new} \leftarrow heuLocalOpt(G, s)$;
  if $P(E(s), E(s_{new}), G, T) > 0$ **then**
    Accept the new state:
    $s+ = s_{new}$;
  end
until $T > e_{min}$;

The annealing schedule is defined by the call $temperature(r)$, which would yield the temperature to use, given the fraction r of the distance to goal traversed so far. To apply this algorithm the following parameters are to be defined: the state, the energy (goal) function $E()$, the candidate generator heuristic function of finding global and local optimal values, the acceptance probability function $P()$, and the annealing schedule temperature() and initial temperature $init\ temp$. The following section performs division operation using this algorithm.

D. Proposed Division Algorithm Using New Approach of Simulated Annealing

Suppose, X is the m bit divisor and Y is the n bit dividend. The targeted quotient is Q and remainder is R. To perform the modified simulated annealing firstly a heuristic function is required to find the global and local optimal values. In division operation, the global optimal value is considered as possible safe nearer value of Y and local optimal values depicts the possible nearer value of updated Y, depicted as Y' value. Hence, a single heuristic function can be used for both global and local optimal value.

If a heuristic function is based on $A = n-m$, then multiplying $n-m$ bits with m bits will create a maximum value of n bits that is $n - m + m = n$ bits. The maximum binary value of $n - m$ bits is a sequence of $n - m$ number of 1 such as $1_1 1_2 1_3 ... 1_{n-m-1} 1_{n-m}$. After multiplying the maximum value

of $n-m$ bits with X may create a large number greater than Y which may produce a negative value after subtraction. To avoid that complication this paper considers a possible optimal value of $n-m-1$ number of zeros following only a one at the MSB (Most Significant Bit) such as $1_0 0_1 0_2 ... 0_{n-m}$.
Considering $Y > X$, initially the minimum value of A can be zero that means the quotient would be 1 and remainder would be $Y-X$. Otherwise, the proposed algorithm considers a new value optimal value B which is as earlier mentioned is $1_0 0_1 0_2 ... 0_{n-m}$. Now, considering e_{min} as 1 initially, a loop is considered with terminating condition variable T as temperature, where temperature is calculated on condition when updated $Y(Y')$ is less than or equal to X, that is the remainder is less than the divisor. At that point, remainder and accumulated quotient Q is provided as output.
As a probabilistic condition of a step being accepted is whether B is less than Y' or not. Then, divisor, X is subtracted from $(m+1)$ number of MSB (Most Significant Bit) bits of Y, stored in diff variable, which is a noticeable improvement as previously in other division mechanisms n-bit subtraction was required at each step whereas, in proposed algorithm the subtraction of $(m+1)$ bits is sufficient reducing the mathematical complexity and delay as subtraction is a sequential procedure. Y' is then updated by appending $(m+2)^{th}$ to 0^{th} LSB bits of Y after diff making $Y' = diff||(m+2)^{th}$ to 0^{th} bits of Y. The value of n is updated with the length of updated Y'. This step is accepted with probability 1 and the value of the resultant quotient is updated by adding B to the current value of Q. This process continues until temperature T is less than e_{min}. Finally, the remainder R is updated with the value of Y' and the quotient and remainder is given as output. The proposed algorithm has been illustrated in Algorithm 2 and a flowchart of the proposed division approach has been shown in Fig. 2. Moreover, an illustrated example is demonstrated in Fig. 3 which accomplishes the division of a binary number $(101110)_2$ (dividend) by $(10111)_2$ (divisor) in 2 iterations (8 steps), whereas the existing [1] non-restoring division algorithm requires 7 iterations (20 steps) and paper [5] requires 5 iterations (17 steps) for the same dividend and divisor. Moreover, the proposed division method does not require any multiplexing for selection of quotient bit due to the application of the proposed simulated annealing algorithm which subsequently reduces the number of operations, requiring only two operations (addition and subtraction), whereas three operations (addition, subtraction and multiplexing) are required to perform the division operation in [1] and [5].
The proposed divider circuit has been exhibited in Fig. 4. Subtractor $S1$ computes the number of bit differences between divisor and dividend, whereas $S2$ finds the updated Y‘ by subtracting X from first $(m+1)^{th}$-bit of Y. Register A, B, R and Q stores the dividend Y, number of bit differences between divisor and dividend, partial remainder or updated Y' and final quotient, respectively. It is noted from the architecture that the quotient bits and partial remainders are generating simultaneously. The path constitutes of subtractor $S1$, subtractor $S2$, Register Q and Register R (shown as red path in Fig. 4) produces the partial remainder. On the other hand, subtractor $S1$, Register B, Adder and Register Q forms the path for calculating the quotient bits (marked as blue path in Fig. 4).

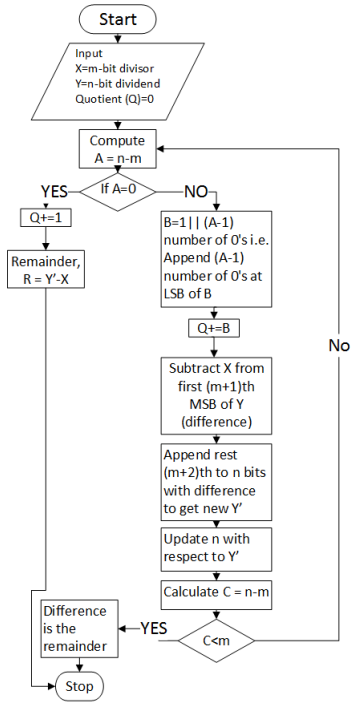


Fig. 2: Flowchart of the Proposed Division Algorithm.

Problem: Dividend (Y) = 101110, Divisor (X) = 10111

Initialization:

m = 5 bit (Number of bits in divisor X)
n = 6 bit (Number of bits in dividend Y)
Quotient is stored in a register, Q = 0
Compute A = (n-m)
= (6-5)
= 1

Iteration 1:

i. Since A ≠ 0, B = 1 || zero number of 0's
= 1 (i.e. , Appending Zero
number of 0's since (6-5-1)=0)

ii. Update Q = Q + B
= 0 + 1
= 1 (Binary Addition)

Q
1

iii. difference = 101110 (first 6th MSB bit of Y)
- 10111
10111

iv. Update Y which is Y' = 10111 ||
zero number of 0's = 10111

v. C = n-m = 5-5 = 0

Iteration 2:

i. Since A = n-m = 5-5 = 0

ii. Update Q = Q + 1
= 1 + 1
= 10 (Binary Addition)

Q
10

iii. Remainder = Y' - X
= 10111-10111
= 0

Remainder = 0 , Quotient = 10

Fig. 3: Example Simulation of Proposed Division Algorithm.

Lemma 1 presents the proof for the required number of iterations by the proposed division approach.

Lemma 1: *The proposed division algorithm requires at most* $(n-m+1)$ *number of iterations, where* n *is number of bits in dividend,* m *is number of bits in divisor and* $n \geq m$. ■

Proof: The above statement is proved by the mathematical induction.

Basis: The basis case holds for the number of bits in divisor and dividend are equal that is $n = m$ and $(m - m+1) = 1$.

Hypothesis: Assume that the statement holds for $n = k$. So,

ALGORITHM 2: Proposed Algorithm for Division Operation Using Proposed Simulated Annealing

Input: m-bit divisor X and n-bit dividend Y;
Output: The quotient Q and Remainder R;
$Y' = Y$;
$e_{min} = 1$;
$Q = 0$;
$T = 0$;
$Q \leftarrow heuOpt()$;
repeat
  $T \leftarrow temperature()$;
  $Q \leftarrow heuOpt()$;
  if $Prob() > 0$ **then**
    Accept the new state:
    $Q+ = B$;

  end
until $T > e_min$;
heuOpt()
$A = n - m$; **if** A=*0* **then**
  Q+=1;
  R=$Y' - X$;
else
  $B = 1$ || $(A - 1)$ number of 0's at LSB of B;
end
return Q;
temperature()
if $Y' \leq X$ **then**
  T=1;
  $R \leftarrow Y'$;
else
  T=0;
end
return T;
prob()
if $B < Y'$ **then**
  diff=$(m+1)^{th}$ MSB of $Y' - X$;
  Y' = $diff$ || $(m+2)^{th}$ to n bits of Y';
  n=length of Y';
  p=1;
else
  p=0;
end
return p;

a k-bit dividend requires $(k - m + 1)$ number of iterations.
Induction: Now, considering $n = k+1$, a $(k+1)$-bit dividend requires $(k+1-m+1) = (k-m+2)$ number of operations. Now, reduce the number of bit in dividend by one to produce $n = k$. Then, a k-bit dividend requires $(k - m + 2 - 1) = (k - m + 1)$ number of iterations which holds the hypothesis. So, the statement holds for $n = k + 1$ □

Example 1: For $n = 6$ and $m = 5$, the proposed algorithm performs the division operation in (6-5+1) = 2 iterations which has been also illustrated in Fig. 3.

IV. Simulation Results & Performance Analysis

The proposed divider circuit is coded in VHDL and implemented in a Virtex-6 platform targeting XC6VLX75T Xilinx FPGA with a -3 speed grade by using ISE 13.1. The results are shown in Table II in terms of LUT, slices and delay. The proposed algorithm outperforms the best known existing [7] approach in terms of delay by an improvement of 40% for 16-bit by 16-bit, 41.2% for 32-bit by 32-bit,

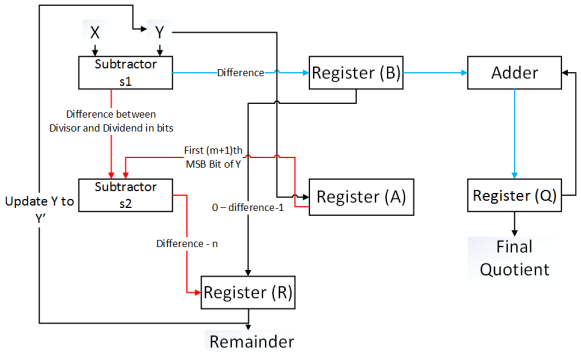


Fig. 4: Architecture of the Proposed Divider Circuit.

41.55% for 64-bit by 32-bit and 44.67% for 256-bit by 128-bit division, respectively. Moreover, the proposed design achieves an enhancement of 31.8% for 16-bit by 16-bit, 32.9% for 32-bit by 32-bit, 34.85% for 64-bit by 32-bit and 36.17% for 256-bit by 128-bit, respectively in terms of number of LUTs used for division operation. Fig. 5 and Fig. 6 demonstrate the efficiency of the proposed division algorithm using the new approach of simulated annealing for the calculation as well as the generation of partial remainders and the number of iterations required to obtain the final quotients, respectively. The proposed algorithm requires only 3 iterations producing the partial remainders with the value of $(10001)_2$, $(101)_2$ and $(10)_2$, respectively for dividend $(11101)_2$ and divisor $(11)_2$, whereas the best known existing [10] method requires 5 iterations producing the partial remainders with the value of $(11100)_2$, $(1011)_2$, $(111)_2$, $(101)_2$ and $(10)_2$, respectively for the same set of divisor and dividend as shown in Fig. 5. In addition, the proposed approach requires 50% less number of iterations over the best known existing approach [7] as depicted in Fig. 6. Table III refers to the comparison with respect to logic gates used in division operation along with the average improvements. It is shown that the improvements are increasing for higher bit division which is a clear indication of the prominent performance of the proposed division algorithm. The CPU execution time has been calculated in a dual core CPU, 4 GHz clock cycle, 4 Gigabytes of RAM and 64-bit operating system as illustrated in Fig. 7 which reflects the superiority of the proposed division algorithm over the contemporary approaches. The main reason behind the enhancement of the proposed division algorithm is the using of new simulated annealing technique. Firstly, it uses a heuristic function to compute the number of bit-differences between the dividend and divisor. Secondly, the proposed method defines the number of quotient bits (global optimal). Thirdly, the temperature schedule is calculated which might be non-increasing depending on how the local optimal fills the defects to reach global optimal. Fourthly, less number of partial remainders (local optimal) are generated on basis of temperature schedule. Finally, during subtraction operation, X is subtracted from the first $(m+1)^{th}$-bit of Y , where X is the divisor, Y is the dividend and m is the number of bits in dividend. Thus, the proposed method of division algorithm optimizes the required number of LUTs, slices, delay, logic

TABLE II: Analysis Between Existing and Proposed FPGA-based Dividers in Terms of LUTs, Slices and Delay

Divider	16-bit by 16-bit			
Method	Existing[6]	Existing[7]	Existing[11]	Proposed
LUT	1060	712	320	218
Slice	620	417	191	133
Delay(ns)	48.7	28.4	31	17.04
Method	32-bit by 32-bit			
LUT	4172	2704	1152	773
Slice	2275	1713	687	464
Delay(ns)	102.5	74.6	85	43.8
Method	64-bit by 32-bit			
LUT	665	591	482	314
Slice	503	371	291	197
Delay(ns)	274.4	204.8	243.2	119.7
Method	256-bit by 128-bit			
LUT	2247	1473	1435	916
Slice	1952	1045	1016	652
Delay(ns)	2066.8	1715.2	1894	949.02

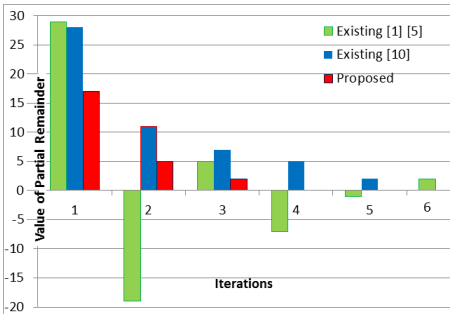


Fig. 5: Graphical Representation of Generation of Partial Remainders.

gates and number of iterations.

V. Concluding Remarks

This paper presents a novel divider circuit using a new approach of simulated annealing, an algorithm which overcomes the deficits of existing simulated annealing [3] [4] algorithm providing guaranteed optimal solution with intelligent temperature parameter using the problem specific heuristic functions to find optimal values. The proposed division algorithm outperforms the existing algorithms [1], [5] and [10] in terms of number of iterations required to obtain the final quotient and generation of partial remainders. Moreover, the proposed FPGA-based divider circuit shows better performances over the existing approaches [6], [7] and [11]. These improvements in FPGA-based divider circuit will consequently influence the advancement in many FPGA-based applications like cryptography, image processing, scientific computations, signal processing and many more [2].

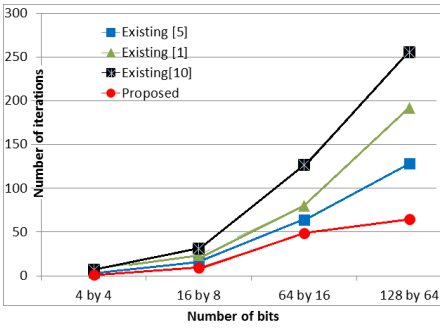


Fig. 6: Graphical Analysis of Number of Iterations.

TABLE III: Complexity Analysis Among the Proposed & Existing Methods in Terms of Number of Logic Gates

Method	8-bit	16-bit	32-bit	64-bit	128-bit
Existing[8]	90	177	348	680	1351
Existing[9]	76	130	267	520	1102
Proposed	45	78	157	306	628
Avg. Improvement	45.39%	47.96%	48.04%	48.07%	48.26%

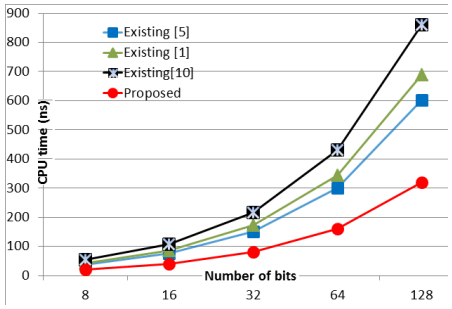


Fig. 7: CPU Time Requirement for Division Operation.

References

[1] Jun, K., & Swartzlander, E. E. (2012, November). "Modified non-restoring division algorithm with improved delay profile and error correction". In 2012 Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR) (pp. 1460-1464). IEEE.

[2] Sworna, Zarrin Tasnim, Mubin Ul Haque, Nazma Tara, Hafiz Md Hasan Babu, and Ashis Kumar Biswas. "Low-power and area efficient binary coded decimal adder design using a look up table-based field programmable gate array." IET Circuits, Devices & Systems 10, no. 3 (2016): 163-172.

[3] Aarts, Emile, Jan Korst, and Wil Michiels. "Simulated annealing." In Search methodologies, Springer US, 2014. (pp. 265-285).

[4] Dowsland, K. A., & Thompson, J. M. (2012). "Simulated annealing". In Handbook of Natural Computing (pp. 1623-1655). Springer Berlin Heidelberg.

[5] Subha, S. (2016). "An Improved Non-Restoring Algorithm". International Journal of Applied Engineering Research, 11(8), (pp. 5452-5454).

[6] Sorokin, N. (2006). "Implementation of high-speed fixed-point dividers on FPGA". Journal of Computer Science & Technology, 6.

[7] Sutter, G., & Deschamps, J. P. (2009, August). "High speed fixed point dividers for FPGAs". In 2009 International Conference on Field Programmable Logic and Applications (pp. 448-452). IEEE.

[8] Jevtic, R., Jovanovic, B., & Carreras, C. (2011, May). "Power estimation of dividers implemented in FPGAs". In Proceedings of the 21st edition of the great lakes symposium on Great lakes symposium on VLSI (pp. 313-318). ACM.

[9] Jovanovic, B., Jevtic, R., & Carreras, C. (2014). "Binary Division Power Models for High-Level Power Estimation of FPGA-Based DSP Circuits". IEEE Transactions on Industrial Informatics, 10(1), 393-398.

[10] Saha, P., Kumar, D., Bhattacharyya, P., & Dandapat, A. (2014). "Vedic division methodology for high-speed very large scale integration applications". The Journal of Engineering, IET, 1(1).

[11] Muoz, Daniel M., Diego F. Sanchez, Carlos H. Llanos, and Mauricio Ayala-Rincn. "Tradeoff of FPGA design of a floating-point library for arithmetic operators." Journal of Integrated Circuits and Systems 5, no. 1 (2010): 42-52.

[12] Mubin Ul Haque, Zarrin Tasnim Sworna, Hafiz Md. Hasan Babu and Asish Kumer Biswas. "A Fast FPGA-Based BCD Adder", in the Journal of Circuit, System and Signal Processing (CSSP), Springer. (February, 2018), DOI: 10.1007/s00034-018-0770-3.