Fixed-Point Divider Using Newton Raphson Division Algorithm

Pawan Kumar Pandey, Dilip Singh, and Rajeevan Chandel

Abstract A fixed point divider is needed for determining the result of division up to a fixed number of points in its fractional part. The divider does so with a good accuracy so that the result can be used for further applications, where the accuracy as well as speed of different modules should be high. This paper presents a 32-bit fixed point divider design based on Newton Raphson division algorithm. The design comprises of two units viz., one is reciprocal unit and the other is the multiplication unit. For the reciprocal unit the divider uses Newton Raphson method and for multiplication unit it uses the multiplication operator. The Verilog HDL coding of the proposed divider design is simulated using Xilinx Vivado tool and synthesized using Cadence EDA tool.

Keywords Verilog · Newton–Raphson method · Slack · FPGA · Divider

1 Introduction

Division algorithms mainly fall into two categories namely, slow division and fast division methods. The slow division methods produce one digit of the final quotient per iteration, which includes algorithms like restoring and non-restoring algorithms [1, 2]. These are based on add and shift operations, consequently, these are time consuming methods. Fast division method starts with a close approximation of final result, so the computation time of final result is much less. Division using Newton Raphson technique comes in the category of fast division algorithms as it uses an initial approximation [3, 4]. Fixed point arithmetic [2, 5, 6] is commonly used for computation because floating point calculation increases the size and complexity of modules [2].

Dividers play an important role in digital signal processing algorithms [1, 7, 8], like the algorithms used to compute the linear prediction coefficients in linear prediction coder. Hence, the study on various types of dividers is essential. Reference [9] gives a

P. K. Pandey (✉) · D. Singh · R. Chandel
Electronic and Communication Engineering Department, National Institute of Technology Hamirpur, Hamirpur, Himachal Pradesh, India

V. Nath and J. K. Mandal (eds.), *Proceeding of Fifth International Conference on Microelectronics, Computing and Communication Systems*, Lecture Notes in Electrical Engineering 748,
https://doi.org/10.1007/978-981-16-0275-7_19

review on various dividers of which the Vedic divider [10] comes out to be faster and area centric. In [5] an array structure for high speed division algorithm is reported in which pipelining method is used to increase the speed of divider. But the pipelining along with increasing the speed, also increase area and cost of the divider. Reference [5] also includes an Iterative Restoring divider which comes out to acquire small area and cost, but the clock cycles used for producing the results are high. Reference [11] shows floating point division using Taylor series expansion algorithm which is basically a third order Newton Raphson method with a reduction in lookup tables causing a reduction is area. In [6] a 32-bit 231 MHz arithmetic unit is given in which division is carried using logarithmic manipulation. It includes 8- region piecewise linear approximation for logarithmic at arithmetic for reducing the number of clock cycles needed for obtaining the final results. But the ASIC and FPGA implementation are not carried out. Reference [12] shows introduction and FPGA implementation of division using Goldschmidt algorithm which also comes under the fast division algorithms. The FPGA implementation of VLSI divider circuits is also important [2, 5, 7, 12] as it helps in the practical verification of the design. Other types of divider reported in the literature are high-redux; serial; integer and floating dividers [13–15]. The division using Newton Raphson method is also represented in [3, 4] using different techniques but the simulation and FPGA implementation are not carried out. The present paper is an attempt in this direction.

The present paper presents a 32-bit fixed point divider using Newton Raphson division algorithm in which the first bit from MSB is the sign bit, the next three bits are for integer part and the remaining 28 bits are for the fractional part. The large number of bits for the fractional part results in a good accuracy. This also includes multipliers in its operation for that the data representation in Q-format [16] is used.

A 32-bit fixed-point divider is also implemented in the present work using shift and subtract method. Shift and subtract method use the conventional method of division. In this for the integer part of the result it uses simple subtraction method where the divisor is repeatedly subtracted from dividend until the result is less then divisor. The number of times divisor is subtracted then become the integer part. For the fractional part it uses shifting and subtraction method, but the drawback of this method is that it produces one bit of result per clock cycle for fractional places. Hence, it takes large number of clock cycles for producing the result and is not suitable for delay centric designs.

This paper is organised as follows. The present section introduces the topic and provide a brief background on different division algorithms. In Sect. 2 Newton Raphson method, its use as division algorithm along with the flow chart are presented. Section 3 provides the simulation and synthesis results obtained using various EDA tools, their discussion and conclusion is drawn in Sect. 4, followed by references.

2 Newton Raphson Method

Newton Raphson method is used to find the root of an algebraic function with the help of some initial approximation. Let the function be,

$$y = f(X) \tag{1}$$

Let the initial approximation of the root of this function is X_0. Then according to the Newton Raphson method, the next approximation of the root say X_1 is given as,

$$X_1 = X_0 - (f(X_0)/f'(X_0) \tag{2}$$

where, $f'(X)$ is the first derivative of the function $f(X)$.

With the help of the second approximation the next approximation is computed for the same i.e.,

$$X_2 = X_1 - (f(X_1)/f'(X_1)) \tag{3}$$

This computes the next approximation of the root. Thus, a generic relationship is given for calculating the next approximations as,

$$X_{n+1} = X_n - \left(f(X_n)/f'(X_n)\right) \tag{4}$$

where, X_{n+1} is the next approximation while X_n is just the previous approximation.

The approximations are computed till the next approximation is nearly the same as the previous approximation.

2.1 Newton Raphson Division Algorithm

Suppose two numbers say a and b are to be divided and let Y be the result after the division, which can be written as,

$$Y = a/b \tag{5}$$

Alternately Y can be written as,

$$Y = a \times (1/b) \tag{6}$$

Now the reciprocal of b i.e. $1/b$ is computed with the help of Newton Raphson method. Then it is multiplied by a to get the output Y.

The function $f(X)$ for calculating the reciprocal of b is,

$$f(X) = (1/X) - b \tag{7}$$

Its root is $1/b$.

So,

$$f'(X) = -(1/X^2) \tag{8}$$

Let the initial approximation for the root be X_0, then we can calculate the next approximation of the function using Newton Raphson method, which comes out to be,

$$X_1 = X_0(2 - b \times X_0) \tag{9}$$

2.2 *Computation of the Initial Approximation for Newton Raphson Division Algorithm*

Newton Raphson method needs an initial approximation, based on that the subsequent approximations are calculated. For division algorithm the initial approximation is calculated by making a linear approximation for the reciprocal function.

In Fig. 1 the black line shows the actual reciprocal function in which x-axis ranges from 0.5 to 1, and y-axis is the reciprocal of x-axis points, thus y-axis ranges from 1 to 2.

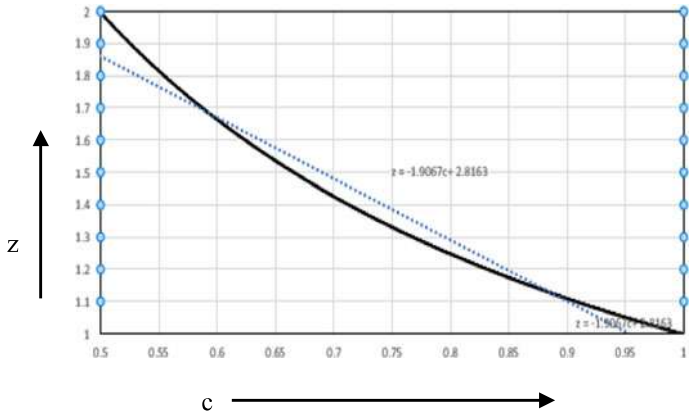

Fig. 1 Linear approximation for the reciprocal function

The dotted blue line shows the best linear approximation using MS office Excel, for the reciprocal function which comes out to be,

$$z = -1.9067c + 2.8163 \tag{10}$$

2.3 *Flow-Chart for Newton Raphson Division Algorithm*

Figure 2 illustrates the flow-chart for the Newton Raphson division algorithm.

As seen from the flow-chart in Fig. 2, firstly the inputs a and b are read. One of the inputs let b is normalized in the range of 0.5 to 1. In order to normalise it we have to divide or multiply b by suitable powers of 2. Divide or multiply by 2 in Verilog HDL [16] is equivalent to left or right shift by 1 respectively. After normalising the input, the Newton Raphson division algorithm is applied to find the best approximate result for $1/b$. Again, we must de-normalise the answer by dividing or multiplying the answer by the same powers of 2 by which it is normalised previously. To get the final answer we multiply a to de-normalised value i.e. $1/b$.

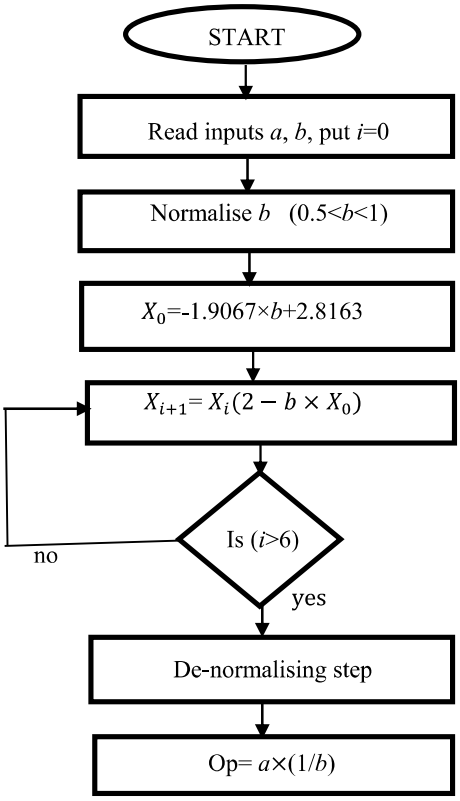

Fig. 2 Flow-chart for the Newton Raphson division algorithm

3 Results and Discussion

The simulation of Newton Raphson division algorithm is performed using Xilinx Vivado 2015.2 tool [17]. For comparative analysis of the designed Newton Raphson division algorithm, the shift and subtract division algorithm is also simulated in the present work.

From Fig. 3, it can be seen that the output of division using Newton Raphson method is attained after every 6 clock pulses. However, for Fig. 4 output is coming after every 27 clock pulses as it produces one bit at a time for its fractional part, thus it can be inferred that the Newton Raphson division algorithm has higher latency than that of shift and subtract method. The divider using Logarithmic arithmetic [6] can produce the result only in 2 clock cycles with an error of 0.2%. However, the percentage error in cases of both of the designed dividers is nearly 0. Consequently, for better latency and high accuracy the division using Newton Raphson method is the best.

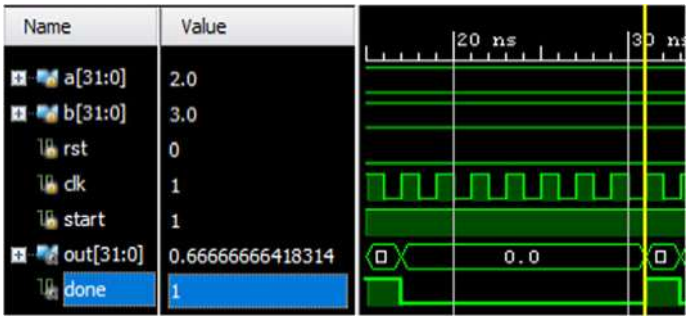

Fig. 3 Simulation result for Newton Raphson division algorithm

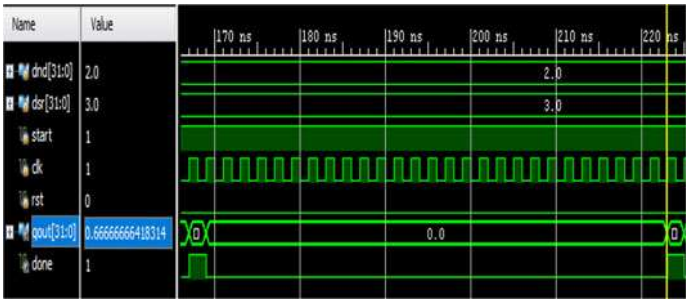

Fig. 4 Simulation result for division using shift and subtract method

3.1 Analysis of Synthesis Results

The synthesis of the two types of dividers is carried out using two different types of EDA tools. For FPGA synthesis Xilinx Vivado 2015.2 tool [17] is used by selecting Zybo board on it. For ASIC synthesis Genus tool from Cadence [18] is used. The coding is accomplished in Verilog HDL [19] (Table 1).

Figure 5 shows the total on-chip power consumed (P_t) by the Newton Raphson divider and is the sum of static power and dynamic power. Here it is,

$$P_t = 0.102W + 0.007W = 0.109W \tag{11}$$

The total on-chip power consumed by the shift and subtract divider is (Fig. 6).

$$P_t = 0.102W + 0.003W = 0.105W \tag{12}$$

The clock frequencies for these two types of dividers is calculated on the basis of slack and for both dividers the clock frequency comes out to be 25 MHz As seen from the power results the power consumed by the divider using Newton Raphson method is nearly 3.7% more than that of shift and subtract method. This is because the divider using Newton Raphson method uses 4 multipliers which consume more power than any other RTL component.

Table 1 Description of RTL components used in dividers using Xilinx Vivado 2015.2

RTL components	Divider using Newton Raphson's method	Divider using shift and subtract method
Adders	4	5
XOR gates	1	1
Multipliers	4	0
Multiplexers	9	11
Registers	8	9

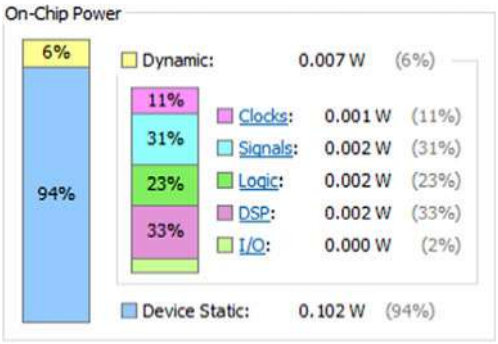

Fig. 5 Power results for divider designed using Newton Raphson method

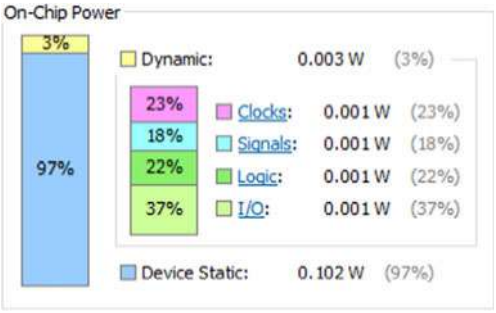

Fig. 6 Power results for divider designed using shift and subtract method

Table 2 FPGA results for the designed 32-bit fixed point dividers

Performance metrics	Goldschmidt algorithm [12]	Shift and subtract method (This work)	Newton Raphson method (This work)
Slice LUTs (look up tables)	9281	348	9281
Flip flops	3025	192	3025
IOBs (input output blocks)	161	100	161

Table 3 ASIC area requirement for various dividers

Performance metric	Taylor series method [11]	Shift and subtract method (This work)	Newton Raphson method (This work)
Area (mm^2)	0.037	0.003215	0.016113

The FPGA implementation of the designed 32-bit fixed point dividers is presented in Table 2. These are also compared with the divider designed using Goldschmidt Algorithm reported in [12].

From Table 2 it can be seen that the number of Look up tables (which is basically a block of SRAM that is indexed by LUT's inputs) used in Newton Raphson divider are 753 whereas the divider shift and subtract method uses 348 also the flip flops and IOBs(input output blocks) in both the dividers are comparable. However, in case of divider reported in [12] the number of slice LUTs, flip flops and IOBs are much higher than the dividers presented in this paper. Tables 3 and 4 show the ASIC results for dividers designed in the present work and for the comparative analysis the ASIC results for Vedic Divider [10] and divider using Taylor series expansion method [11] are also considered.

From Table 3, it can be seen that the area of the divider using Newton Raphson method is nearly 5 times more than that of add and subtract method. This is because it uses 4 number of multipliers which need a larger area. However, the area of divider

Table 4 ASIC dynamic power dissipation in different dividers

Performance metric	Vedic divider [10]	Taylor series method [11]	Shift and subtract method (This work)	Newton Raphson method (This work)
Power (mW)	0.024	>0.812	0.0142	0.812

using Taylor series expansion method is nearly twice than that of the Newton Raphson divider. In [10] the ASIC area of Vedic divider results are not given.

From Table 4 it is analysed that the power dissipation in case of shift and subtract divider is least of the four divider designs given in Table 4 because it does not require any multipliers for its operation and is therefore power centric design. The dynamic power results for division using Taylor series in not given in [11] but it also uses four number of multipliers for its operation [11], so its power consumption is assumed to be equal to or more than division using Newton Raphson method because its area (Table 4) is also more than that of Newton Raphson method. The Vedic divider [10] comes out be better in case of dynamic power consumption than that of division using Newton Raphson method.

4 Conclusion

The paper presents the design and analysis of 32-bit divider using Newton Raphson algorithm. The comparative analysis of this divider is performed by designing and analysing another divider based on shift and subtract method along with other dividers given in references. From the results it can be seen that the Newton Raphson divider comes out to be a faster division algorithm. But since it uses 4 number of multipliers, this causes increment in the area and power. The Vedic divider is consuming less dynamic power than divider using Newton Raphson division algorithm bur its accuracy and latency is not proposed in [10]. For reducing the power and area of divider using Newton Raphson method the normal multipliers can be replaced by Booth multipliers [20] and by proper pipelining the area and speed requirement can be improved. However, it is analysed that for power and area centric designs, divider designed using shift and subtract method proves to be better candidate. Thus, the choice of divider circuits depends upon the applications and the usage.

Acknowledgements The technical support through SMDP-C2SD project awarded to NIT Hamirpur by MeitY (Ministry of Electronics and Information Technology), Government of India, New Delhi, is duly acknowledged.

References

1. Meyer-Baese U (2004) Digital signal processing with field programming gate arrays. Springer, Heidelberg
2. Sorokin N (2006) Implementation of high-speed fixed-point dividers on FPGA. J Comput Sci Technol 6(1):8–11
3. Rodriguez-Garcia A, Pizano-Escalante L, Parra-Michel R, Longoria-Gandara O, Cortez J (2013) Fast fixed-point divider based on Newton-Raphson method and piecewise polynomial approximation. In: International conference on reconfigurable computing and FPGAs, ReConFig 2013, vol 2, no 2
4. Louvet N, Muller JM, Panhaleux A (2010) Newton-Raphson algorithms for floating-point division using an FMA. In: International conference on application-specific systems, architectures and processors, pp 200–207
5. Narendra K, Shabirahmed BJ, Swaroop Kumar K, Asha GH (2015) FPGA implementation of fixed-point integer divider using iterative array structure. Int J Eng Tech Res (IJDTR) 3(4):170–179
6. Kim H, Nam BG, Sohn JH, Yoo HJ (2005) A 231MHz, 2.18mW 32-bit logarithmic arithmetic unit for fixed-point 3D graphics system. In: IEEE Asian solid-state circuits conference ASSCC 2005, pp 305–308
7. Malik P (2015) High throughput floating-point dividers implemented in FPGA. In: IEEE 18th international symposium on design and diagnostics of electronic circuits & systems (DDECS), pp 291–294
8. Rodellar V, Sacristan MA, Alvarez A, Diaz A, Peinado V, Gómez P (2001) A divider-multiplier high level synthesis library element for DSP applications. In: IEEE international conference on electronics, circuits and systems, vol 1, pp 545–548
9. Vemula R, Chari KM (2018) A review on various divider circuit designs in VLSI. In: Conference on signal processing and communication engineering systems SPACES 2018, January 2018, pp 206–209
10. Saha P, Banerjee A, Bhattacharyya P, Dandapat A (2011) Vedic divider: novel architecture (ASIC) for high speed VLSI applications. In: International symposium on electronic system design ISED 2011, pp 67–71
11. Kwon TJ, Draper J (2009) Floating-point division and square root using a Taylor-series expansion algorithm. Microelectron J 40(11):1601–1605
12. Singh N, Sasamal TN (2016) Design and synthesis of Goldschmidt algorithm based floating point divider on FPGA. In: International conference on communication and signal processing ICCSP 2016, pp 1286–1289
13. Aoki T, Nakazawa K, Higuchi T (2000) High-radix parallel VLSI dividers without using quotient digit selection tables. In: International symposium on multiple-valued logic, pp 345–352
14. Mishra R (2017) An efficient VLSI architecture for a serial divider. In: 2nd international conference on devices for integrated circuit, DevIC 2017, pp 482–486
15. Saadat H, Javaid H, Parameswaran S (2019) Approximate integer and floating-point dividers with near-zero error bias. In: Design automation conference, pp 1–6
16. Hussain MZ, Parvin KN (2016) Q-format data representation and its arithmetic. Int J Electron Commun Technol (IJECT) 7(2):57–62
17. Xilinx Inc. (2015) Vivado Design Suite User Guide. Ug903, vol 4, pp 1–173
18. Xcelium C (2019) Tempus, Voltus Cadence RTL-to-GDSII Flow Course Version 2.0 Developed By University Support Team Cadence Design Systems, Bangalore
19. Palnitkar S (2003) Verilog HDL, a guide to digital design and synthesis. IEEE 1364-2001 Compliant
20. Chen YH, Li CY, Chang TY (2011) Area-effective and power-efficient fixed-width Booth multipliers using generalized probabilistic estimation bias . IEEE J Emerg Sel Top Circuits Syst 1(3):277–288