
# HARDWARE IMPLEMENTATION OF METHODOLOGIES OF FIXED POINT DIVISION ALGORITHMS

D. Kumar, P. Saha and A. Dandapat
National Institute of Technology Meghalaya
Shillong, India 793003
Emails: deepak.enc@gmail.com, sahaprabir1@gmail.com, anup.dandapat@nitm.ac.in

***Submitted: May 28, 2017*** ***Accepted: July 24, 2017*** ***Published: Sep. 1, 2017***

***Abstract- This paper describes the hardware implementation methodologies of fixed point binary division algorithms. The implementations have been extended for the execution of the reciprocal of the binary numbers. Radix-2 (binary) implementations of digit recurrence and multiplicative based methods have been considered for comparison. Functionality of the algorithms have been verified in Verilog hardware description language (HDL) and synthesized in Xilinx ISE 8.2i targeting the device xc4vlx15-12sf363 of Virtex4 family. Implementation was done for both signed and unsigned number systems, having bit width of operands vary as an exponential function of*** $2^n$***, where*** $n$***=2 to 5. Performance parameters have been calculated in terms of clock frequency, FPGA slice utilization, latency and power consumption. Implementation results indicate that multiplicative based algorithm is superior in terms of latency, while digit recurrence algorithms are consuming low power along-with less area overhead.***

**Index terms*: Digit recurrence, Division algorithm, Fixed point, FPGA, Newton-Raphson, Verilog.***

# I. INTRODUCTION

Binary division is the most complicated computation technique among other arithmetic operations [1]. Substantial algorithms and the implementation methods so far have been proposed to execute this operation like digit recurrence method [2-5], the multiplicative method [1, 6-10], approximation methods [6], and special methods such as the COordinate Rotation DIgital Computer (CORDIC) [6].

Digit-recurrence algorithm produces a single quotient bit in each iteration [3], through add/subtract operation of multiple of divisor from the dividend /partial remainder [6]. Simplicity of the implementation and availability of the exact remainder are the main advantages of digit recurrence algorithm whereas convergence rate is linear with operand size. However, the same method can be adopted to get the fractional quotient in fixed point format by omitting the remainder.

Newton-Raphson (N-R) and Goldschmidt (GdS) methods are common examples for multiplicative division [8]. Divisions based on multiplicative implementation has a common factor of quadratic convergence (i.e. each iteration successively produces double of accurate bits) [8]. Self-correcting nature (i.e. generated error in each iteration does not reflect to the next) is the key strength of N-R method although two dependent multiplications limit the performance of such algorithm. Due to the dependency, multiplication cannot be computed in parallel. However, in GdS algorithm multiplication can be implemented in a parallel manner but lacks self-correcting nature. Both the algorithms (N-R and GdS) require efficient fixed point multiplication technique to achieve the high speed operation, and also the initial approximation to reduce the number of iterations [10]. Reciprocal computation can be treated as a special type of division where dividend has a fixed value of 1. This reciprocal unit plays a pivotal role for the implementation of N-R algorithm.

Though division is an infrequent operation but it has lots of application in common fields like digital signal processing, computer graphics and image processing [11] and in some specific fields like in robotics to calculate position and orientation values [12-13], in sensor networks to

find the different functions such as node's wakening probability, morphactin concentration and probability density[14-15], and in image compression to estimate the value of transfer function of wiener filter[16]. Now a days, high speed along with high clock frequency with less chip area is desirable for most of the system application. Although, some digital instrumentation like tachometer requires low cost, medium speed with limited accuracy [17]. Thereby, the implementation comparisons among the algorithms are required for further utilizations in application specific integrated circuit (ASIC).

In this paper, fixed point signed and unsigned number division has been implemented based on digit recurrence and multiplicative division algorithms. Also, the work has been extended for the implementation of reciprocal of a number using the same methodology. Restoring and non-restoring division algorithms have been adopted from digit recurrence group. Moreover, N-R and GdS algorithms have been taken from multiplicative division group. All the mentioned algorithms have been analyzed for different operands length. Functionality of each algorithm have been verified in Verilog HDL and synthesized in Xilinx ISE 8.2i for the device xc4vlx15-12sf363 of Virtex4 family. Performance parameters have been calculated in terms of clock frequency, FPGA slice utilization, latency and power consumption. Implemented results show that Goldschmidt algorithm has the lowest latency among others, while digit recurrence (restoring and non-restoring) algorithms are consuming low power along-with less area overhead for both division and reciprocal operations.

## II. BACKGROUND THEORY

Dividend (dvd) and divisor (dvr) are the operands of a division operation and results are quotient (q) and remainder (rem). The rem is an optional term of the division operation based on design aspect. All the operands and results have been considered as $N-\text{bit}$ binary number and have the mathematical representation as [18] :

$$\text{dvd} = \sum_{i=0}^{N-1} \text{dvd}_i 2^i$$

$$\mathrm{dvr} = \sum_{i=0}^{N-1} \mathrm{dvr}_i 2^i$$

$$q = \sum_{i=0}^{N-1} q_i 2^i$$

$$\mathrm{rem} = \sum_{i=0}^{N-1} \mathrm{rem}_i 2^i$$

a. Unsigned Division

Division operation for unsigned can be written as [6]:

$$\mathrm{dvd} = q \times \mathrm{dvr} + \mathrm{rem} \quad (1)$$

and

$$\mathrm{rem} < dvr \quad (2)$$

b. Signed Division

In case of unsigned number system the equation (1, 2) is adequate, where as for signed number system the equation (2) can be written as

$$|\mathrm{rem}| < |\mathrm{dvr}| \text{ and } \mathrm{sign}(\mathrm{rem}) = \mathrm{sign}(\mathrm{dvd}) \quad (3)$$

c. Fixed-point Division

To implement the fixed point division operation, equation (1) can be re-written as

$$\mathrm{dvd} = q \times \mathrm{dvr} \quad (4)$$

Where dvd and dvr are same as above mentioned $N-\mathrm{bit}$ array. However, the quotient can be redefined as

$$q = \sum_{i=-N}^{N-1} q_i 2^i$$

Assuming $q = q_{N-1} q_{N-2} \ldots q_1 q_0 \cdot q_{-1} q_{-2} \ldots q_{-N}$

i.e. q is a 2N-bit binary number having N-bit whole (integer) part and N-bit fractional part implied dot (.) at position N.

## III. IMPLEMENTATION

The In this section, implementation of the above mentioned division methodology has been described. Some of the symbolic notation used in following flow charts are as follows: '0'/ '1' represents a single binary bit, ~ stands for bit-wise negation operation, an N- bit register R is represented as R[N-1:0] and XOR for Exclusive-OR operation.

a. Restoring Division Algorithm

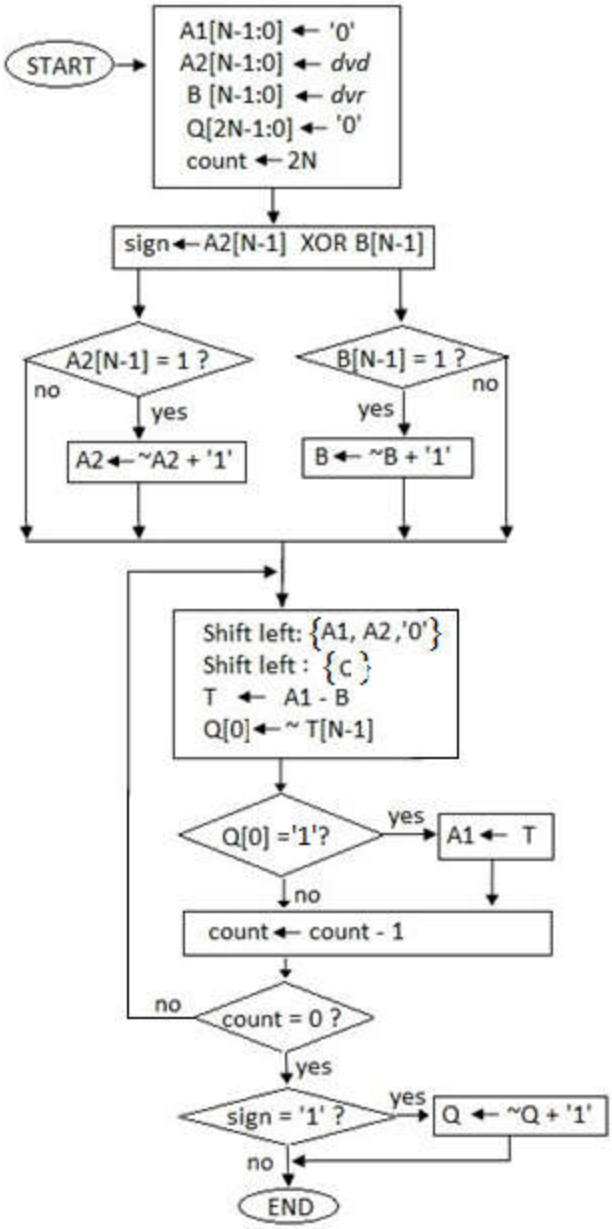


Figure.1. Restoring division for signed binary number (RST_S)

Flowchart diagram for signed restoring division (RST_S) is shown in Fig.1. Although original unsigned (RST_U) restoring division method does not support signed operands directly, it can be modified for the signed number. The modified implementation procedure is shown in Fig.1.

b. Non-restoring Division Algorithm

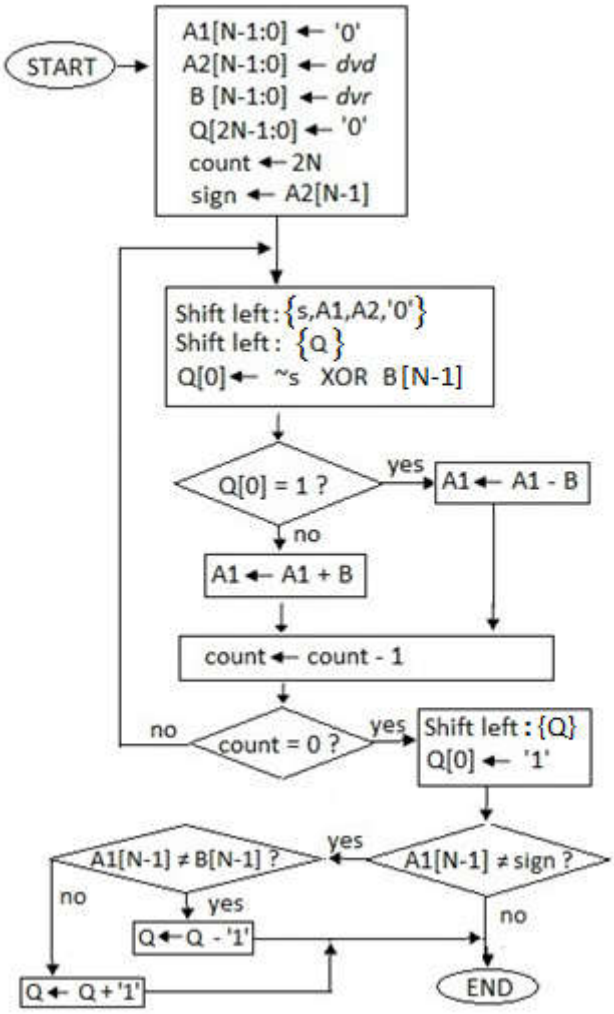


Figure 2. Non-restoring division for signed binary number (NRT_S)

Non-restoring division method has inherent support for signed number system if signed binary {-1, 1} is used to represent the quotient bits. To implement the signed number division (NRT_S), the operands have to be presented in 2's complement format. The implementation procedure is shown in Fig.2. During this implementation generated quotient bit {-1, 1} is encoded as {0, 1} respectively. Further signed binary result i.e. Q converted to 2' complement binary by using

standard method given in [19-23]. Non-restoring division method for unsigned numbers (NRT_U) can be carried out by the same flow chart (Fig.2), where the chosen quotient bits should be taken as conventional binary number system i.e. {0, 1}.

c. Newton-Raphson Division Algorithm

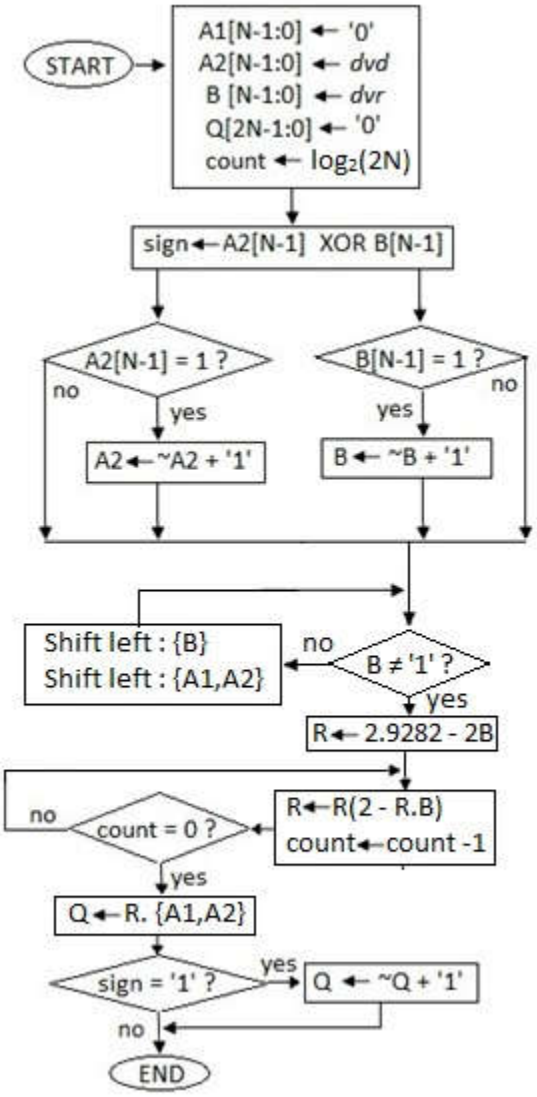


Figure 3. Newton-Raphson division for signed binary number (N-R_S)

To implement the division by Newton–Raphson method, reciprocal ($R \cong 1/dvr$) of divisor ($dvr$) is estimated using Newton's method. Finally quotient ($Q$) is obtained by the multiplication of estimated reciprocal with dividend($dvd$). To implement the reciprocal via

Newton's method, reciprocal function is approximated [6, 19] and shown in equation (5).

$$R_{i+1} = R_i \times (2 - R_i \times dvr) \tag{5}$$

The above recurrence can be initiated with possible initial approximation [6, 19]

$$R_0 = a - b \times dvr \tag{6}$$

Where $a$ and $b$ are constants. The chosen values are 2.9282 and 2 respectively assuming $\frac{1}{2} \leq dvr < 1$. To hold this condition, both $dvd$ and $dvr$ can be shifted left. The flowchart diagram for signed implementation (N-R_S) is shown in Fig.3. Unsigned number division can be implemented through similar approach ignoring the sign handling steps.

d. Goldschmidt Division Algorithm

Basic idea behind the Goldschmidt division algorithms is as follows: multiply both the $dvd$ and $dvr$ by carefully chosen common factor $F_i$ such that the dividend converges to quotient while divisor converges to 1 i.e.

$$Q = \frac{dvd}{dvr} = \frac{dvd}{dvr}\frac{F_0}{F_0}\frac{F_1}{F_1} \ldots \frac{F_i}{F_i} \ldots \tag{7}$$

Under the assumption that $\frac{1}{2} \leq dvr < 1$, common factor can be selected as $F_i = 2 - dvr_i$ with $dvr_0 = dvr$, and using the following recursive equation[19]

$$\frac{dvd_{i+1}}{dvr_{i+1}} = \frac{dvd_i}{dvr_i}\frac{F_i}{F_i} \tag{8}$$

Fig.4 shows the flow chart diagram for signed numbers division through Goldschmidt algorithm (GdS_S). Same diagram can be used for unsigned division (GdS_U) by ignoring the steps for sign handling.

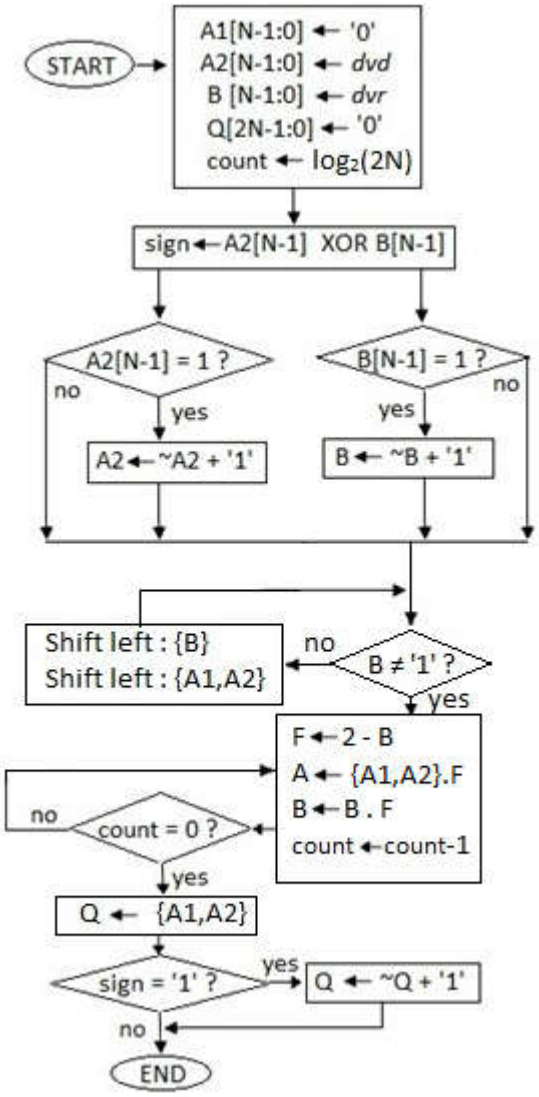


Figure 4. Goldschmidt division for signed binary number (GdS_S)

## IV. RESULTS AND DISCUSSIONS

The functionality of above mentioned algorithms (for both signed and unsigned operands) have been implemented and verified in Verilog hardware description language (HDL). All the algorithms have been taken from different references and simulated in the same environment for comparison purpose. All the designs have been synthesized through Xilinx ISE 8.2i targeting the device xc4vlx15-12sf363. Performance parameters in terms of clock frequency, slice utilization

Table 1(a): Clock frequency (MHz) analysis of different implementation for division

| Bit-length / Algorithm | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| **RST_U** | 288.317 | 288.317 | 261.491 | 219.651 |
| **RST_S** | 288.317 | 269.452 | 257.170 | 215.545 |
| **NRT_U** | 288.317 | 288.317 | 286.676 | 217.042 |
| **NRT_S** | 261.209 | 253.949 | 254.729 | 221.205 |
| **N-R_U** | 202.286 | 187.949 | 49.117 | 38.261 |
| **N-R_S** | 196.07 | 179.01 | 49.098 | 38.242 |
| **GdS_U** | 132.371 | 117.835 | 67.932 | 56.718 |
| **GdS_S** | 132.954 | 115.030 | 67.897 | 56.721 |

Table 1(b): Slice utilization report of different implementation for division

| Bit-length / Algorithm | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| **RST_U** | 54 | 72 | 106 | 173 |
| **RST_S** | 60 | 92 | 159 | 293 |
| **NRT_U** | 54 | 65 | 91 | 140 |
| **NRT_S** | 56 | 74 | 109 | 126 |
| **N-R_U** | 78 | 83 | 182 | 1585 |
| **N-R_S** | 105 | 129 | 235 | 1660 |
| **GdS_U** | 87 | 122 | 199 | 471 |
| **GdS_S** | 96 | 140 | 253 | 582 |

(area), latency and power have been analyzed for different bit lengths. Analysis of clock frequency (MHz) and slice utilization are shown in Table 1(a) and Table 1(b) respectively while latency and power consumption analysis are shown in Fig.5(a) and Fig.5(b) respectively for division operation. From these analysis, it has been observed that, for 32-bits operands,

Goldschmidt algorithm is superior in terms of latency among others, while digit recurrence (restoring and non-restoring) algorithms are consuming low power along-with less area overhead. Beside division, the designs have been extended for the implementation of reciprocal operation. Similar performance parameters have been calculated and shown in Table 2(a), Table 2(b), Fig.6(a) and Fig.6(b).

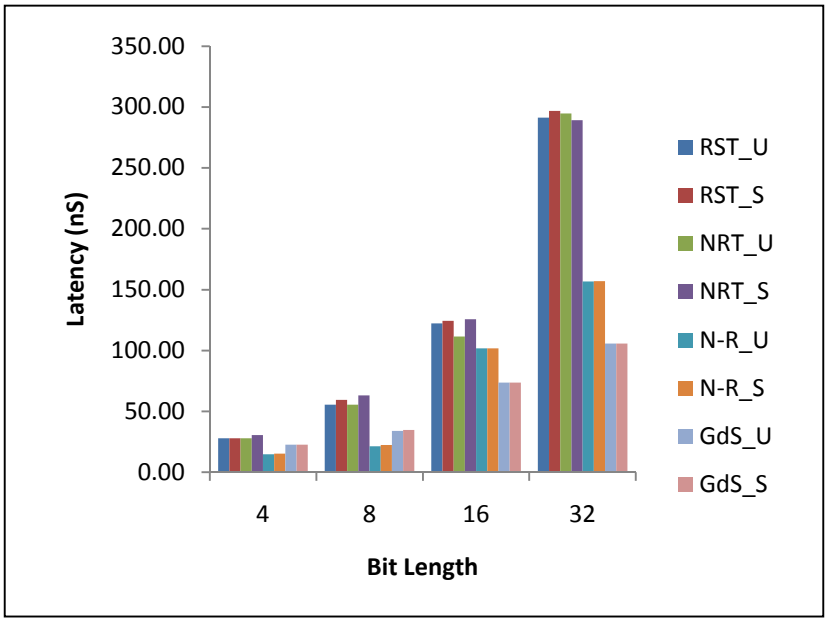


Figure 5(a). Latency (nS) of different implementation for division

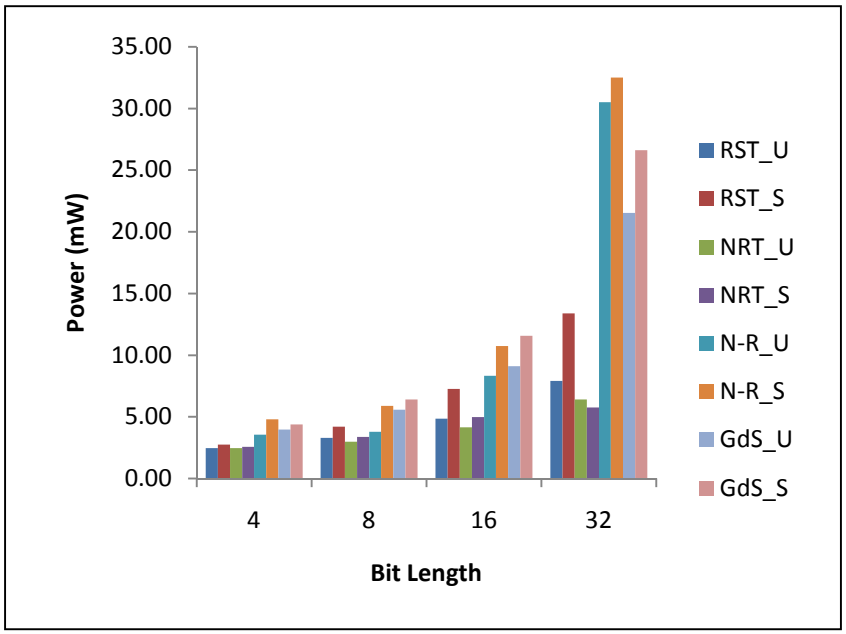


Figure 5(b). Power (mW) analysis of different implementation for division

Table 2(a): Clock frequency (MHz) analysis of different implementation for reciprocal

| Bit-length / Algorithm | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| **RST_U** | 288.317 | 288.317 | 261.491 | 223.968 |
| **RST_S** | 288.317 | 288.317 | 258.451 | 215.545 |
| **NRT_U** | 288.317 | 288.317 | 286.676 | 217.042 |
| **NRT_S** | 288.317 | 250.890 | 251.653 | 221.205 |
| **N-R_U** | 95.617 | 79.008 | 49.152 | 33.242 |
| **N-R_S** | 95.617 | 78.759 | 49.152 | 33.181 |
| **GdS_U** | 132.954 | 115.030 | 67.932 | 56.718 |
| **GdS_S** | 132.954 | 117.489 | 67.676 | 56.567 |

Table 2(b): Slice utilization report of different implementation for reciprocal

| Bit-length / Algorithm | 4 | 8 | 16 | 32 |
|---|---|---|---|---|
| **RST_U** | 53 | 69 | 99 | 160 |
| **RST_S** | 54 | 74 | 122 | 205 |
| **NRT_U** | 53 | 62 | 84 | 127 |
| **NRT_S** | 57 | 71 | 102 | 125 |
| **N-R_U** | 88 | 92 | 136 | 309 |
| **N-R_S** | 89 | 95 | 156 | 353 |
| **GdS_U** | 87 | 120 | 193 | 459 |
| **GdS_S** | 93 | 134 | 236 | 549 |

## V. CONCLUSIONS

In this paper digit recurrence and multiplicative based division methods have been analyzed for both signed and unsigned, fixed point operands. Implementation has been carried out and

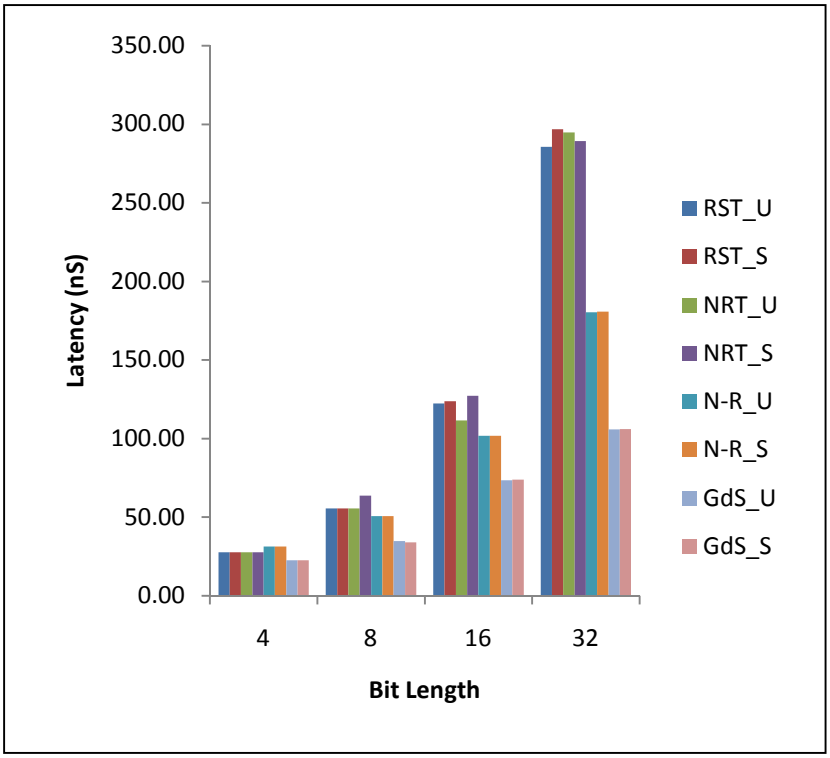


Figure 6(a). Latency (nS) of different implementation for reciprocal

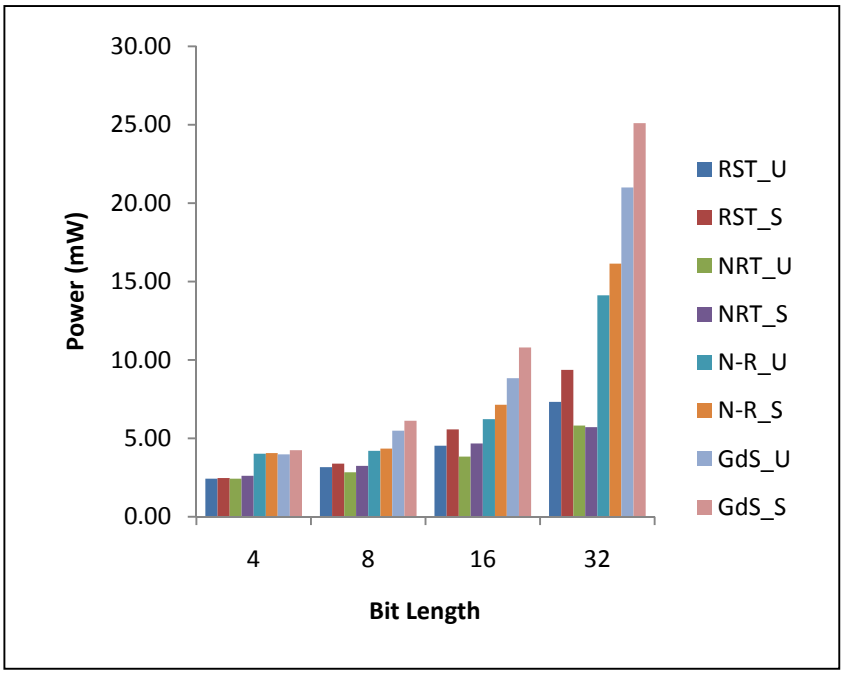


Figure 6(b). Power (mW) analysis of different implementation for reciprocal

performance parameters have been calculated for further utilization of division and reciprocal circuitry. Functionality of the algorithms has been verified in Verilog HDL and synthesized. Clock frequency, slice utilization, latency and power consumption have been measured for different bit lengths. Implementation results show that multiplicative based algorithm particularly Goldschmidt is superior in terms of latency, while digit recurrence (non-restoring) algorithm requires low power along-with less area overhead.

## REFERENCES

[1] D. W. Matula, M. T. Panu and J. Y. Zhang, "Multiplicative Division Employing Independent Factors," IEEE Transactions on Computers, vol. 64, no. 7, pp. 2012-2019, July 2015.

[2] M. D. Ercegovac and J. M. Muller, "Variable radix real and complex digit-recurrence division," IEEE International Conference on Application-Specific Systems, Architecture Processors (ASAP'05), 2005, pp. 316-321.

[3] E. Antelo, T. Lang, P. Montuschi and A. Nannarelli, "Digit-recurrence dividers with reduced logical depth," in IEEE Transactions on Computers, vol. 54, no. 7, pp. 837-851, July 2005. doi: 10.1109/TC.2005.115

[4] J. Ebergen, I. Sutherland and A. Chakraborty, "New division algorithms by digit recurrence," Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers, 2004, pp. 1849-1855 Vol.2.

[5] E. Antelo, T. Lang, P. Montuschi and A. Nannarelli, "Low latency digit-recurrence reciprocal and square-root reciprocal algorithm and architecture," 17th IEEE Symposium on Computer Arithmetic (ARITH'05), 2005, pp. 147-154.

[6] M. D. Ercegovac and T. Lang, “Digital Arithmetic”, Morgan Kaufmann publishers, New York, 2004.

[7] R. Goldberg, G. Even and P. M. Seidel, "An FPGA implementation of pipelined multiplicative division with IEEE Rounding," 15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007), Napa, CA, 2007, pp. 185-196.

[8] N. Kikkeri and P. M. Seidel, "Formal verification of parametric multiplicative division implementations," 2005 International Conference on Computer Design, 2005, pp. 599-602.

[9] G. Even and P. M. Seidel, "Pipelined multiplicative division with IEEE rounding," Proceedings 21st International Conference on Computer Design, 2003, pp. 240-245.

[10] M. Ito, N. Takagi and S. Yajima, "Efficient initial approximation for multiplicative division and square root by a multiplication with operand modification," in IEEE Transactions on Computers, vol. 46, no. 4, pp. 495-498, Apr 1997.

[11] B. Jovanovic, R. Jevtic and C. Carreras, "Binary Division Power Models for High-Level Power Estimation of FPGA-Based DSP Circuits," IEEE Transactions on Industrial Informatics, vol. 10, no. 1, pp. 393-398, Feb. 2014.

[12] Z. Liang and H. Gao, "Formation Algorithms for Multiple Mobile Robots Based on Vision Detection", International Journal on Smart sensing and Intelligent Systems, vol. 9, no. 4, pp. 1840-1858, December 2016.

[13] W. Xiong, "Structural Design and Motion Analysis of Universal Mobile Quadruped Robot", International Journal on Smart sensing and Intelligent Systems, vol. 9, no. 3, pp. 1305-1322, September 2016.

[14] Y. Qin and H. Ying, "PGSA-Based Localization Algorithm for Wireless Sensor Network", International Journal on Smart sensing and Intelligent Systems, vol. 9, no. 3, pp. 1287-1304, September 2016.

[15] Z. Ju-Wei, W. Yu and W. Ya-le, "A Deployment Algorithm of Heterogeneous Underwater Sensor Network Based on Acoustic and Magnetic Joint Sensing Model", International Journal on Smart sensing and Intelligent Systems, vol. 9, no. 4, pp. 2149-2166, December 2016.

[16] Q. Zhou and X. Liu, "A Blind Assessment Method of Image Compression Quality Based on Image Variance", International Journal on Smart sensing and Intelligent Systems, vol. 9, no. 4, pp. 2131-2148, December 2016.

[17] J. C. Majithia, T. J. Koehler and W. Banks, "A Low-Cost Binary Division Circuit for Digital Instrumentation," IEEE Transactions on Instrumentation and Measurement, vol. 23, no. 1, pp. 32-35, March 1974.

[18] M. D. Ercegovac and T. Lang, "Division with limited precision primitive operations," Proceedings of Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, 2001, pp. 841-845.

[19] B. Parhami, "Computer Arithmetic: Algorithms and hardware design", Oxford university press, New York, 2000.

[20] K. Jun and E. E. Swartzlander, "Improved non-restoring division algorithm with dual path calculation," IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS), Columbus, OH, 2013, pp. 1379-1382.

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

[22] M. E. Isenkul, "A comparative performance analysis for the computer arithmetic based fast division algorithms," 24th Signal Processing and Communication Application Conference (SIU), Zonguldak, 2016, pp. 629-632.

[23] P. Krishnamoorthy and R. Tekumalla, "Quotient prediction for low power division," IEEE International SOC Conference, Erlangen, 2013, pp. 273-277. doi: 10.1109/SOCC.2013.6749700