# Performance Evaluation of Division Algorithms in FPGA

K. S. Mannatunga[1], and M. D. R. Perera[2]

[1] *Department of Physics, University of Sri Jayewardenepura, Gangodawila, Nugegoda, Sri Lanka*
[2] *Department of Computer Science, University of Sri Jayewardenepura, Gangodawila, Nugegoda, Sri Lanka*
#K. S. Mannatunga; < ksm@sjp.ac.lk >

***Abstract***— *One of the main reasons that researchers interact with the Field Programmable Gate Arrays (FPGAs) is the parallel processing feature which can be used to make high speed designs. However, arithmetic operations such as division and multiplication in FPGA limit this feature considerably. Although, hardware multipliers are included to reduce the effect, there are no built-in hardware division elements in any FPGA, where it is the most complicated and expensive operation among the others. This paper presents a comparative study of performance for several distributed division solutions for FPGAs. Restoring, Non-restoring, Radix-2 SRT (Sweeney, Robertson and Tocher), Radix-2 SRT with CSA (Carry Save Adder) and the Goldschmidt's division algorithms were selected for the study. In addition, Xilinx's LogiCORE Divider Generator core v3.0 and Matlab Simulink Divider Generator 3.0 were also evaluated. The comparison was done by means of resource utilization (RU), delay in critical path (DT) and area×time (RU×DT) parameter for Xilinx Spartan-3E XC3S100E and Spartan-6 XC6LX16 devices. The lowest logic consumption and RU×DT were seen in the non-restoring algorithmic divider in both Spartan-3E and Spartan-6. The lowest DT for Spartan-3E and Spartan-6 were reported by the Simulink Divider Generator 3.0, which is 3.692 ns and the Xilinx's LogiCORE Divider Generator core v3.0, which is 2.626 ns respectively. However, the non-restoring divider is identified as the best balanced division solution by concerting the RU×DT parameter.*

***Keywords***— **FPGA, Goldschmidt, Non-restoring, Restoring, Xilinx**

## I. INTRODUCTION

In the development of modem science and technology, complexities of the application are in increased. In most cases conventional microcontroller chips are not much fitted to fulfill the requirement. Than the microcontrollers, FPGAs provide more flexibility for the designer to implement a complex algorithm with gaining more performance. At present FPGA chips are popular in wide range of applications such as military, environmental monitoring(Dinesh and Saravanan, 2011; Mathurkar and Chaudhari, 2013), wireless sensor networks(Garcia et al., 2009; Perera et al., 2014; Portilla et al., 2007), robotic systems(Li et al., 2003; Piltan et al., 2011a, 2011b), etc. due to its constructive properties such as flexibility, re-configurability, low power consumption, parallel processing, and low latency.

In microcontrollers, mathematical operations such as addition, subtraction, multiplication and division are performed using the Arithmetical Logical Unit (ALU) thought FPGAs does not offer such a unit. In FPGAs, arithmetic operations are performed using the shift registers, lookup tables, hardware multipliers, and logic gates. Among the basic arithmetic operations (addition, subtractions, multiplications and divisions) divisions is the most complex and expensive among the four arithmetic operations in hardware. Since there are no built-in dividers inside the FPGA, algorithmic procedures are developed often; hence more resources are consumed by these algorithms. In addition to resources, division operation requires more than one clock cycle, and thus it is slower than the other operations. Unlike the other arithmetic operations, accuracy of the division is lesser. Therefore, it is advisable to use a minimal number of divisions in any FPGA design. However, knowing advantages and disadvantages of available dividers will be helpful for FPGA designers to implement systems, while achieve optimum performance.

In the robotic systems researchers use FPGAs for their implementations because FPGAs require less space and it provides more performances while consuming low power. In robotics, required data's are gathered through the different sensors such as cameras, magnetometers, gyroscopes, accelerometers, etc. and it is necessary to design real-time operations algorithms to process data. In those cases division operation are highly used and it is required to implement high performance division algorithm to achieve high throughput.

Division algorithms can be classified into two categories: Digit-Recurrence Algorithms and Convergence Algorithms, as shown in Fig. 1. Restoring, non- restoring(Ercegovac and Lang, 2004; Parhami, 2009), SRT (developed by Sweeney(Cocke and Sweeney, 1957), Robertson(Robertson, 1958) and Tocher(Tocher, 1958)), SRT with carrier save adders(SRT with CSA) algorithms are few examples for digit-Recurrence and most common convergence

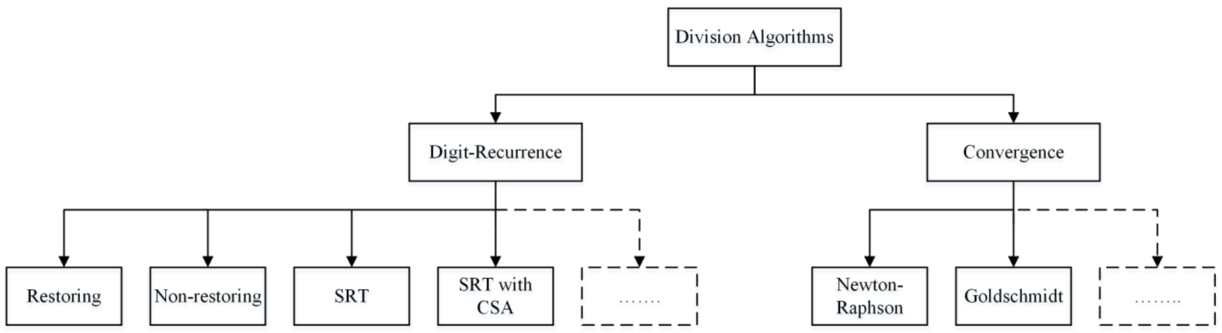

Figure 1. The classification of division algorithms.

algorithms are the Newton-Raphson and the Goldschmidt algorithms(Goldschmidt, 1964).

As IP core solutions for division, Xilinx provides LogiCORE Divider Generator cores, which can be configured to either radix-2 non-restoring or high-radix division algorithms. Xilinx also provides Divider Generator block for Mathworks's Simulink environment to work with Xilinx's system generator tool(Ownby and Mahmoud, 2003), which can also be configured to either radix-2 non-restoring or high-radix division algorithms. However, high-radix division cannot be used with low end FPGAs. Because it does not have XtremeDSP slices.

In this study, performance comparisons of restoring, non-restoring, radix-2 SRT, radix-2 SRT with CSA, the Goldschmidt division algorithms, LogiCORE Divider Generator core v3.0 and Simulink Divider Generator 3.0 in terms of logic utilization and critical path delay were carried out for Xilinx FPGA devices. document is a template and it adopts standard practices used by researchers in both hard and soft sciences.

## II. METHODOLOGY

Restoring, non-restoring, radix-2 SRT, radix-2 SRT with CSA and the Goldschmidt division algorithms were implemented with the VHDL hardware description language. Since almost all of FPGA manufacturers provide tools that support for VHDL, implemented algorithms can be easily ported to any FPGA type without making any changes. However, LogiCORE divider core and Simulink divider cannot be used with FPGAs other than the Xilinx FPGAs since they use Xilinx compiling tools, where other vendor's FPGAs do not support.

Design flow of restoring, non-restoring, radix-2 SRT and the Goldschmidt algorithms are shown in Fig. 2 to Fig. 5 respectively. Only shift, add and subtract operations were employed for developing the restoring, the non-restoring and the SRT algorithms while the Goldschmidt method involved multiplication also. In radix-2 SRT with CSA divider, carrier save adders were used in addition to shift registers.

As shown in the Fig. 2, In the restoring method, a quotient digit Q(i), which can be either 0 or 1, is decided by the Z such a way that 1 is selected when Z is positive other vice 0. The reason for called restoring algorithm is that this method re-adjust the Z by adding the divisor D when Q(i) is 0. Thus after every iteration, $0 \leqslant R \leqslant D$. This added back part in the restoring algorithm is avoided in the non-restoring algorithm as shown in the Fig. 3. In the non-restoring method, addition or subtraction, operation is chosen depending on the previous iterated value of R to calculate the new value for R. This method, after every iteration, R is kept in the range $-D \leqslant R \leqslant D$. Restoring and non-restoring are the most well-known digit-recurrence algorithms in hardware implementation of division.

The SRT algorithm was originally developed to reduce the number of additions and subtractions in division operation. The radix-2 SRT divider was implemented according to the procedure shown in Fig. 4. Final values of quotient and

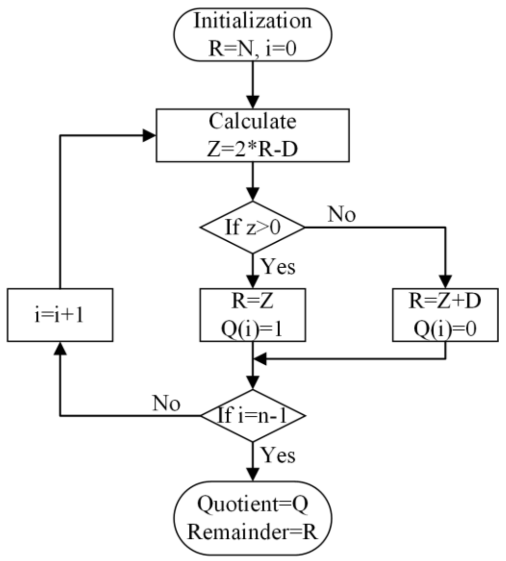

Figure 2. Flow chart of the restoring division algorithm

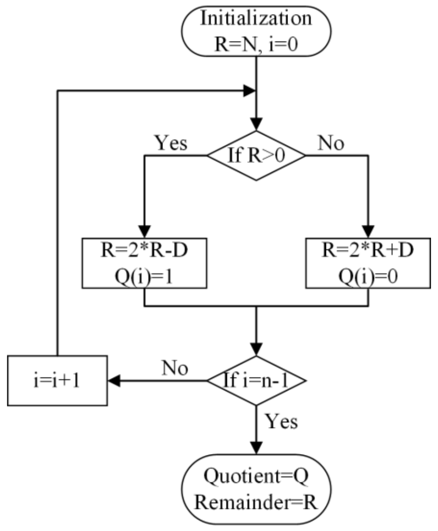

Figure 3. Flow chart of the non-restoring division algorithm.

reminder of the SRT divider are depended on the values of QN and QP. Radix-2 SRT with CSA is used carrier save adders which is the only difference to the SRT methods. In both SRT algorithms, range of R is [-D, D].

The MacLaurin series is the base of the Goldschmidt's algorithm of division, which consists of series multiplications. The Goldschmidt's divider was implemented according to the procedure shown in Fig. 5.

For this investigation, unsigned dividers were implemented

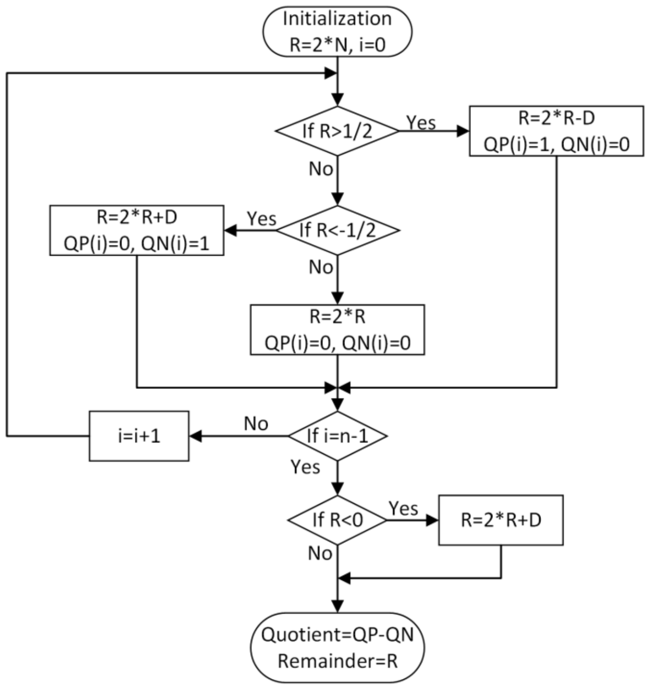

Figure 4. Flow chart of the SRT division algorithm.

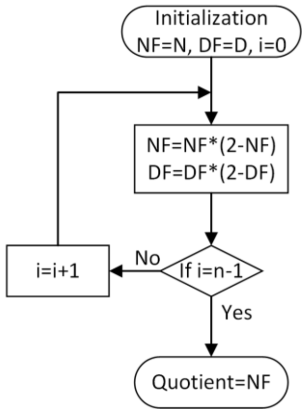

Figure 5. Flow chart of the Goldschmidt's division algorithm.

with equal width of 8-bit for divided (N), divisor (D), quotient (Q) and reminder (r). These dividers were tested on Spartan-3E XC3S100E and Spartan-6 XC6LX16 FPGAs, available in the Basys 2 and the Nexys 3 development boards respectively, and simulated with the Xilinx ISim simulator, which bundled with the Xilinx ISE webpack version 14.6.

The method of the performance evaluation of the developed dividers was based on the resource utilization by mean of the number of flip flops (FFs), look-up tables (LUTs), multiplexers (MUXs) and DSP slices, and the maximum delay in the critical path.

## III. RESULTS AND DISCUSSION

Table 1 and Table 2 show the logic utilization of implemented 8-bit divider architectures in Spartan-3E and Spartan-6 FPGA respectively.

By considering the resource, it can be seen that difference of the restoring and the non-restoring dividers is small, which the restoring divider consumed few more look-up

Table 1. Resource utilization of each division architecture in the Spartan-3E FPGA

| Division Architecture | Resources | | |
|---|---|---|---|
| | FFs | LUTs | MUXs |
| Restoring | 21 | 31 | 1 |
| Non-restoring divider | 22 | 25 | 1 |
| SRT | 31 | 59 | 1 |
| SRT with CSA | 40 | 100 | 1 |
| Goldschmidt's algorithm | 22 | 197 | 1 |
| Xilinx Divider core | 224 | 79 | 1 |
| Matlab system generator | 436 | 143 | 1 |

Table 2. Resource utilization of each division architecture in Spartan-6 FPGA

| Division Architecture | Resources | | | |
|---|---|---|---|---|
| | FFs | LUTs | MUXs | DSP Slices |
| Restoring | 21 | 31 | 12 | 0 |
| Non-restoring divider | 21 | 25 | 8 | 0 |
| SRT | 37 | 38 | 12 | 0 |
| SRT with CSA | 40 | 84 | 16 | 0 |
| Goldschmidt’s algorithm | 21 | 22 | 0 | 2 |
| Xilinx Divider core | 224 | 152 | 104 | 0 |
| Matlab system generator | 436 | 296 | 192 | 0 |

tables than the non-restoring, and the non-restoring divider consumed one more flip-flops than the restoring in both the Spartan-3E and the Spartan-6.

In the Spartan-3E, lowest resource utilization was found in the non-restoring divider. In the Spartan-6 device, the Goldschmidt’s algorithmic divider consumed the lowest number of resources among other dividers. However, it utilized two DSP slices additionally, which comprise of multiplexers, flip-flops, adders, subtracters, multipliers, etc. Thus, actual resources such as the number of flip-flops and multiplexers may be different than the tabulated values in the Table II. The second lowest resource count was found in the non-restoring divider. Since, the Spartan-3E does not have DSP resources, the Xilinx compiler implemented the Goldschmidt’s divider using other hardware resources.

Table III shows the critical path delay in each division architecture for both the Spartan-3E and the Spartan-6 FPGAs. According to the table, the lowest delay in the critical path was found in the divider that generated by the Matlab system generator for DSP tools for the Spartan-3E FPGA, and in the divider that produced by the Xilinx’s LogiCORE Divider Generator core v3.0 for the Spartan-6

Table 3. Delay in the Spartan-3E and Spartan-6 FPGAs

| Division Architecture | Device | |
|---|---|---|
| | Spartan-3E | Spartan-6 |
| Restoring | 4.992 ns | 2.906 ns |
| Non-restoring divider | 4.719 ns | 3.314 ns |
| SRT | 6.116 ns | 3.258 ns |
| SRT with CSA | 7.112 ns | 5.016 ns |
| Goldschmidt’s algorithm | 13.791 ns | 7.998 ns |
| Xilinx Divider core | 3.957 ns | 2.626 ns |
| Matlab system generator | 3.692 ns | 3.150 ns |

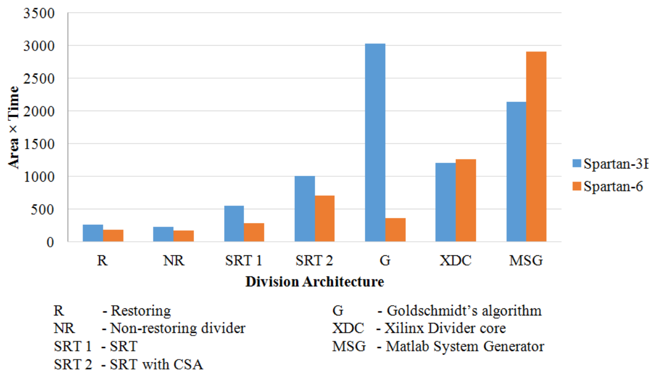

Figure 6. Product of area and time for different division architectures.

FPGA. However, these two architectures consumed lots of device resources compared to other architectures for both device families as indicated in Table II. Resource utilization can be improved by increasing the number of clocks per division parameter, which will reduce the through put also. Therefore, performances of dividers were evaluated by ranking them according to the value of RU×DT where the lower value provides the better performance.

The RU×DT value of each divider architecture is shown in Figure 6. The best result in RU×DT figure is provided by the non-restoring algorithmic divider for both the Spartan-3E and the Spartan-6 FPGAs.

## III. CONCLUSIONS

This paper, the restoring, the non-restoring, the radix-2 SRT, the radix-2 SRT with CSA, the Goldschmidt’s algorithms, the Xilinx’s LogiCORE Divider IP core and the Simulink Divider Generator were compared with based on the resource utilizations and the delays in the critical path values. All dividers were implemented on two families of Xilinx FPGAs: the Spartan-3E and the Spartan-6.

The results show the best balanced division solution in both FPGA devices is the non-restoring algorithmic divider by considering the RU×DT values. However, for optimum speed, the divider generator of the Matlab system generator for DSP tools and the Xilinx’s LogiCORE Divider IP core are the most suitable divider solutions in the Spartan-3E and the Spartan-6 FPGA devices respectively. Since other manufacturers have different architectures than the Xilinx has, these cores may not be ported to devices other than the Xilinx.

## References

Cocke, J., Sweeney, D.W., 1957. High speed arithmetic in a parallel device. IBM Rep. Feb.

Dinesh, M., Saravanan, P., 2011. FPGA based real time monitoring system for agricultural field. Int. J. Electron. Comput. Sci. Eng. 1, 1514–1519.

Ercegovac, M.D., Lang, T., 2004. Digital arithmetic. Elsevier.

Garcia, R., Gordon-Ross, A., George, A.D., 2009. Exploiting partially reconfigurable FPGAs for situation-based reconfiguration in wireless sensor networks, in: Field Programmable Custom Computing Machines, 2009. FCCM'09. 17th IEEE Symposium on. IEEE, pp. 243–246.

Goldschmidt, R.E., 1964. Applications of division by convergence. Massachusetts Institute of Technology.

Li, T.-H.S., Chang, S.-J., Chen, Y.-X., 2003. Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot. Ind. Electron. IEEE Trans. On 50, 867–880.

Mathurkar, S.S., Chaudhari, D.S., 2013. A review on smart sensors based monitoring system for agriculture. Int. J. Innov. Technol. Explor. Eng. IJITEE 2, 76–78.

Ownby, M., Mahmoud, W.H., 2003. A design methodology for implementing DSP with Xilinx System Generator for Matlab, in: SOUTHEASTERN SYMPOSIUM ON SYSTEM THEORY. pp. 404–408.

Parhami, B., 2009. Computer arithmetic: algorithms and hardware designs. Oxford University Press, Inc.

Perera, M.D.R., Meegama, R.G.N., Jayananda, M.K., 2014. FPGA Based Single Chip Solution with 1-Wire Protocol for the Design of Smart Sensor Nodes. J. Sens. 2014.

Piltan, F., Sulaiman, N., Jalali, A., Aslansefat, K., 2011a. Evolutionary Design of Mathematical tunable FPGA Based MIMO Fuzzy Estimator Sliding Mode Based Lyapunov Algorithm: Applied to Robot Manipulator. Int. J. Robot. Autom. 2, 317–343.

Piltan, F., Sulaiman, N., Marhaban, M.H., Nowzary, A., Tohidian, M., 2011b. Design of FPGA-based Sliding Mode Controller for Robot Manipulator. Int. J. Robot. Autom. IJRA 2, 173–194.

Portilla, J., Riesgo, T., De Castro, A., 2007. A reconfigurable fpga-based architecture for modular nodes in wireless sensor networks, in: Programmable Logic, 2007. SPL'07. 2007 3rd Southern Conference on. IEEE, pp. 203–206.

Robertson, J.E., 1958. A new class of digital division methods. Electron. Comput. IRE Trans. On 218–222.

Tocher, K.D., 1958. Techniques of multiplication and division for automatic binary computers. Q. J. Mech. Appl. Math. 11, 364–384.