International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-3 Issue-1, June 2013

Pipelining Based Floating Point Division: Architecture and Modeling

Santanu Halder, Abul Hasnat, Azizul Hoque, Debotosh Bhattacharjee, Mita Nasipuri

Abstract— In this paper, an efficient FPGA based architecture for a fractional division based on Newton-Raphson method for IEEE single-precision floating point number is presented. With advent of more graphic, scientific and medical applications, floating point dividers have become indispensable and increasingly important. However, most of these modern applications need higher frequency or low latency of operations with minimal area occupancy. In this work, highly optimized pipelined architecture of an IEEE-754 single precision floating point divider is designed to achieve high frequency on FPGA. The division is performed by multiplying the numerator by the reciprocal value of the denominator and the initial approximation of the denominator is obtained from a Look-up Table.

Index Terms—FPGA, Newton-Raphson Method, IEEE 754 Single precision format, VHDL

I. INTRODUCTION

Nowadays, division is very important for several applications in digital signal and image processing, computer graphics and scientific computing [1]. But division is the most time consuming operation among four arithmetic operations. Designing a high-speed reciprocal unit is very useful for division operation because the division can be replaced as the following method: the reciprocal of divisor is computed at first, and then it is used as the multiplier in a subsequent multiplication with the dividend. If several divisions by the same divisor need to be performed, this method is particularly efficient, since once the reciprocal of divisor is found for the first division, each subsequent division involves just one additional multiplication.

During the recent years field programmable gate arrays (FPGA's) have become the dominant form of programmable logic [2-5]. In comparison to previous programmable devices like programmable array logic (PAL) and complex programmable logic devices (CPLD's), FPGA's can implement far larger logic functions. FPGA's supports sufficient logic to implement complete systems and sub-systems. FPGA exploit the increasing capacity of integrated circuits to provide designers with reconfigurable logic that can be programmed on application-specific basis.

This drastically increases flexibility in both the design process and the final artifact by permitting one board-level design to perform many functions, or to be upgraded in the field.

This paper is organized as follows: The section II describes the brief description of creating the look up table. Section III discusses the Newton-Raphson iteration briefly. Section IV depicts the top level design of the proposed system architecture. Section V shows system architecture of the proposed system, section VI the pipeline architecture, section VII describes RTL simulation, section VIII describes the experimental results and finally section IX concludes about some of the aspects analyzed in this paper.

Manuscript published on 30 June 2013.
***Correspondence Author(s)**

Dr. Santanu Halder, Assistant Professor, Dept. of Computer Science and Engineering, Government College of Engineering Textile Technology, Berhampore, West Bengal, India.
Mr. Abul Hasnat Assistant Professor, Dept. of Computer Science and Engineering, Government College of Engineering Textile Technology, Berhampore, West Bengal, India.
Azizul Hoque, Research Scholar, Kalyani University, West Bengal, India.
Dr. Debotosh Bhattacharjee, Associate Professor, Dept. of Computer Science and Engineering, Jadavpur University, Kolkata, India.
Dr. M. Nasipuri, Professor, Dept. of Computer Science and Engineering, Jadavpur University, Kolkata, India.

II. CREATING LOOKUP TABLE

In the present work, IEEE-754 single precision format has been used to represent a number. According to IEEE standard 754 [6], the operand Y, is a 32-bit normalized single precision floating-point number in the range of $1 \le Y < 2$. An implicit leading one and the 23 fractional bits constitute the mantissa
The 23-bit mantissa is expressed as:

$$Y_{mantissa} = 1.y_1 y_2 y_3 y_4 \ldots y_{23}$$

The Lookup table contains the first 8 bits of the reciprocal of Y and hence the size of the Lookup table is $2^8 \times 8$ bits. **Table 1** shows the values in the Lookup table.

TABLE1: VALUES IN THE LOOKUP TABLE

$y_1 y_2 y_3 y_4 y_5 y_6 y_7$	Initial Approximation
0000000	11111111
0000000	11111110
0000000	11111101
0000000	11111100
0000000	11111011
0000000	11111010
0000000	11111001
0000000	11111000
0000000	11110111
0000000	11110110
------------	-------------
------------	-------------
1111111	10000000

III. NEWTON-RAPHSON ITERATION

A general division operation is expressed by the **Eq. 1**.

$$Z = \frac{X}{Y} \qquad (1)$$

Alternatively **Eq.-1** can be rewritten as:

$$Z = X \times reciprocal(Y) \qquad (2)$$

Now, the Newton-Raphson iteration method is used to find the reciprocal value of Y. Let,

$$D = \frac{1}{Y} \qquad (3)$$

Therefore,

$$f(Y) = \frac{1}{Y} - D \qquad (4)$$

$$f'(Y) = -\frac{1}{Y^2} \qquad (5)$$

According to the Newton-Raphson approximation method, if Y_i is the current root, then the next estimate Y_{i+1} is given by **Eq. – 6**.

$$Y_{i+1} = Y_i - \frac{f(Y_i)}{f'(Y_i)} \qquad (6)$$

Substituting the value of $f(Y)$ and $f'(Y)$ in **Eq. 6**, we get,

$$Y_{i+1} = Y_i - \frac{\frac{1}{Y_i} - D}{-\frac{1}{Y_i^2}}$$

$$= Y_i + Y_i - DY_i^2$$

$$= Y_i(2 - DY_i) \qquad (7)$$

IV. TOP LEVEL DESIGN

Fig. 1 shows the top level design of the hardware implementation of fractional division. The FPGA module takes two inputs D1 and D2 in IEEE-754 single precision format and divides D1 by D2. The output E contains the exponent part of the result, Q contains the mantissa part of the result and S contains the sign of the result.

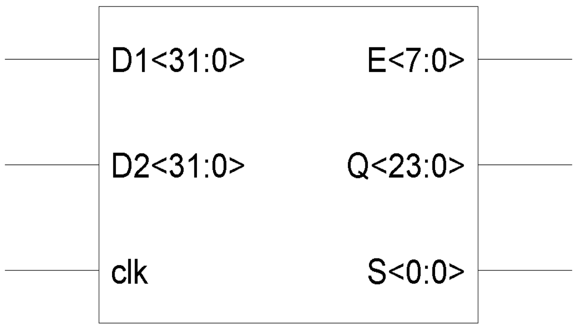
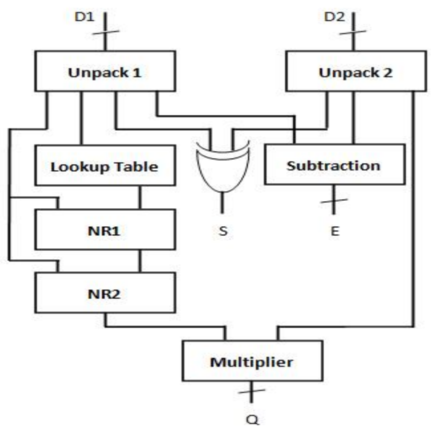


Fig 1: Top Level design of fractional division architecture

V. SYSTEM ARCHITECTURE

The proposed architecture of fractional division operation is shown in **Fig. 2**. The architecture contains two Unpack modules, one Lookup table, two Newton-Raphson iteration modules, one multiplier module, one Subtraction module and one X-OR gate. The modeling of the internal architecture of each block has been designed using Very high-speed integrated circuit Hardware Description Language (VHDL). Each block is controlled by a global clock.



NR: Newton-Raphson Iteration module
Fig. 2: System Architecture of Proposed methodology

A. *Unpack Module*

There are two unpack modules in the design. Module Unpack1 takes one 32-bit value Y and decodes it into four values, the sign of the number (S), the exponent of the number (E) and the Mantissa part (M). This module produces another output (L) which is fed as the input to the Lookup table module. Another unpack module Unpack2 takes the 32-bit value of X and extracts the Sign, Exponent and Mantissa part of X. **Eq. 8** to **Eq. 11** shows the decoded values.

$$S = Y(31) \qquad (8)$$

$$E = Y(30:23) \qquad (9)$$

$$M = Y(22:0) \qquad (10)$$

$$L = Y(22:15) \qquad (11)$$

These blocks offer a latency of one clock cycle each. The symbolic representations of these two blocks are shown in the **Fig. 3** and **Fig. 4**. **Algorithm 1 and Algorithm 2** describes the function of these modules.

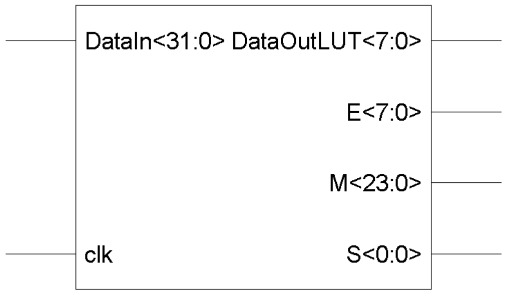


Fig. 3: Symbolic representation of Unpack1 module

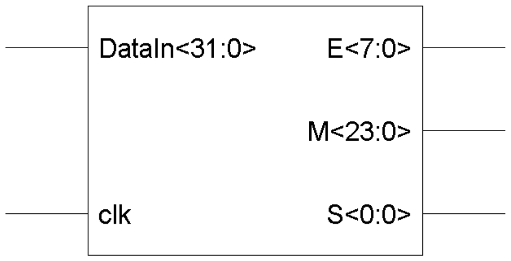


Fig. 4: Symbolic representation of Unpack2 module

Algorithm 1

```
Algorithm Unpack1
{Input: Y}
{Output: S, E, M, L}
Begin
    SY = Y(31);
    EY = Y(30 : 23);
    MY  = "1" & Y(22 : 0);
    LY = Y(22 : 15);
End {End of Algorithm}
```

Algorithm 2

```
Algorithm Unpack2
{Input: X }
{Output: S, E, M}
Begin
    SX = X(31);
    EX = X(30 : 23);
    MX  = "1" & X(22 : 0);
End {End of Algorithm}
```

B. Lookup Table

This module takes one 8-bit value and finds the initial approximation of its reciprocal. This block offers a latency of one clock cycle. The symbolic representation of this block is shown in the **Fig. 5**. **Algorithm 3** describes the function of this module.

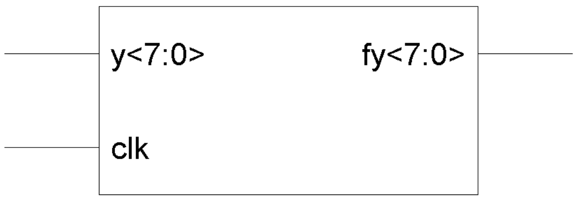


Fig. 5: Symbolic representation of Lookup table

Algorithm 3

```
Algorithm Lookup table
{Input: y }
{Output: R}
Begin
    R=lookuptable(y);
End {End of Algorithm}
```

C. Newton-Raphson Iteration Module

There are two Newton-Raphson Iteration modules NR1 and NR2. NR1 is responsible to find the first stage reciprocal value using Newton-Raphson iteration and NR2 find the second stage approximation. These blocks offer a latency of one clock cycle each. The symbolic representations of these two blocks are shown in the **Fig. 6** and **Fig. 7**. **Algorithm 4 and Algorithm 5** describes the function of these modules.

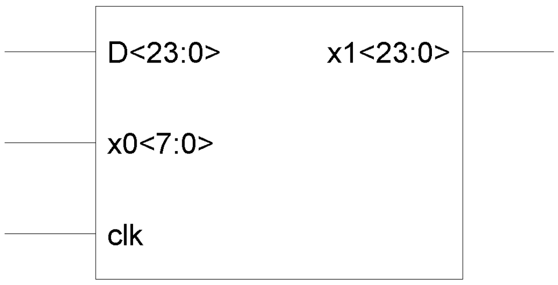


Fig. 6: Symbolic representation of NR1 module

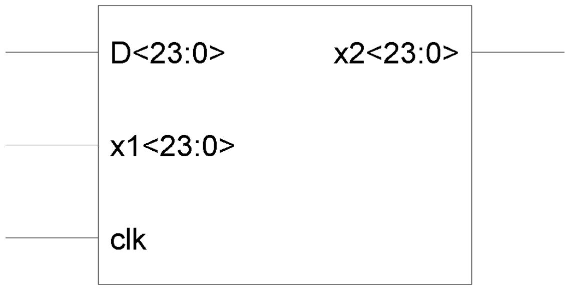


Fig. 7: Symbolic representation of NR2 module

Algorithm 4

```
Algorithm NR1
{Input: R, D}
{Output: Y1}
Begin
    Temp1 = 2 × R;
    Temp2 = R × R;
    Temp3 = Temp2 × D;
    Y1 = Temp1 – Temp3;
End {End of Algorithm}
```

Algorithm 5

```
Algorithm NR2
{Input: Y1, D}
{Output: Y2}
Begin
    Temp1 = 2 × Y1;
    Temp2 = Y1 × Y1;
    Temp3 = Temp2 × D;
    Y2 = Temp1 – Temp3;
End {End of Algorithm}
```

D. Multiplier Block

This block multiplies the normalized mantissa part of X with the reciprocal value of the normalized mantissa value of Y getting from the module NR2 and produces the mantissa part of the final result. This block offers the latency of one clock cycle. The symbolic representation of this block is shown in **Fig. 8**. **Algorithm 6** describes the function of this module.

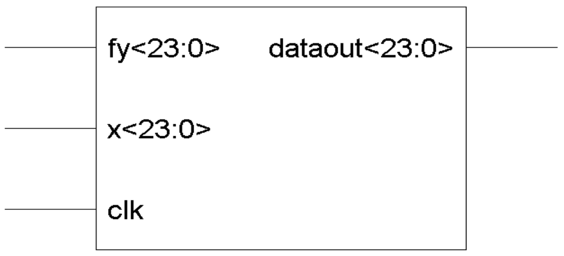


Fig. 8: Symbolic representation of Multiplier Block

Algorithm 6

```
Algorithm Multiplier
{Input: MX, Y2 }
{Output: Q}
Begin
    Q = MX × Y2;
End {End of Algorithm}
```

E. Subtraction Block

This block is used to find the exponent of the result. It subtracts the exponents of Y from the exponent of Y and generates the exponent value of the final result. This block also offers the latency of one clock cycle. The symbolic representation of this block is shown in **Fig. 9**. **Algorithm 7** describes the function of this module.

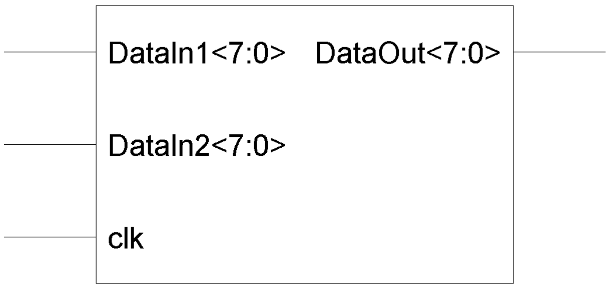


Fig. 9: Symbolic representation of Subtraction block

```
Algorithm 7
Algorithm Subtraction
{Input: EX, EY}
{Output: E}
Begin
    E = EX – EY;
End {End of Algorithm}
```

F. X-OR Block

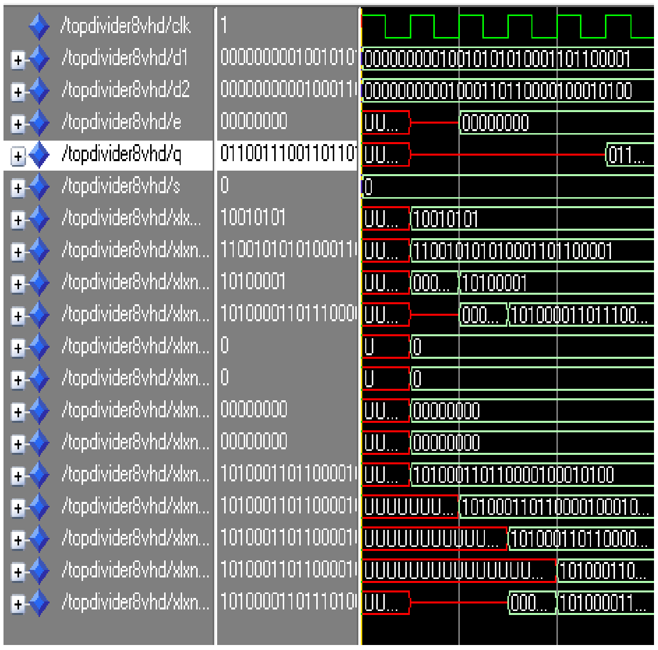
This block is used to find the sign of the result. It performs X-OR operation between the sign bit of X and sign bit of Y and sign of the result. This block also offers the latency of one clock cycle.

VI. PIPELINING ARCHITECTURE

Pipelining is a well known technique for achieving faster clock rates while sacrificing latency. Pipeling offers an economic way to realize temporal paralallism in digital systems. To achieve pipelining one must subdivide the input process into a sequence of subtasks each of which can be executed by specialized hardware stage that operates concurrently with other stages in the pipeline. In the present method, five stage pipelining is incorporated for faster performace.

VII. RTL SIMULATION

Simulation for the Newton-Raphson based fractional division architecture described in this paper is done with the Model SimSE 6.2c. For the testing of the system correctness a testbench file is written in VHDL. The testbench file reads the values of X and Y from a text file *Input.txt* and writes the result in a different text file *Output.txt*. The simulation result for the testbench is shown in the **Fig. 10**. From the simulation result it is clear that the first output is generated after the five clock cycles of the first input set given to the system.



```
/topdivider8vhd/clk      1
/topdivider8vhd/d1       000000000100101010
/topdivider8vhd/d2       000000000010001110
/topdivider8vhd/e        00000000
/topdivider8vhd/q        011001110011011010
/topdivider8vhd/s        0
/topdivider8vhd/xlx...   10010101
/topdivider8vhd/xlxn...  110010101010001110
/topdivider8vhd/xlxn...  10100001
/topdivider8vhd/xlxn...  101000011011100000
/topdivider8vhd/xlxn...  0
/topdivider8vhd/xlxn...  0
/topdivider8vhd/xlxn...  00000000
/topdivider8vhd/xlxn...  00000000
/topdivider8vhd/xlxn...  101000110110000110
/topdivider8vhd/xlxn...  101000110110000110
/topdivider8vhd/xlxn...  101000110110000110
/topdivider8vhd/xlxn...  101000110110000110
/topdivider8vhd/xlxn...  101000011011101000
```

Fig. 10: Simulation Result

VIII. EXPERIMENTAL RESULT

The proposed architecture was implemented using VHDL and synthesized on a Xilinx Vertex 2P XC2VP2fg256 -6 FPGA with simulation on the Model Sim 6.2c from Mentor Graphics Corporation. Some sample result is shown in Tab 2.

TABLE 2: SAMPLE RESULTS

X (Only Mantissa)	Y (Only Mantissa)	Actual Result (only Mantissa)	Our Method (only Mantissa)
01000110110000100010100	10010101010001101100001	01100111001101101100101	01100111001101101100101
00011000101010101110001	10010101010001101100001	01011000101001111011100	01011000101001111011100

The device utilization summary is given in **Table 3**. The architecture is capable of operating at a clock frequency of 89.690 MHz.

TABLE 3: DEVICE UTILIZATION SUMMARY

	Usage	Total	Percentage
Number of Slices	154	1408	10%
Number of Slice Flip Flops	214	2816	7%
Number of 4 input LUTs	276	2816	9%
Number of bonded IOBs	87	140	62%
Number of MULT18X18s:	8	12	66%
Number of GCLKs	1	4	25%

IX. CONCLUSION

The FPGA based architecture for Newton-Raphson based fractional division is useful in most of the modern applications. This architecture is capable of operating at a speed of 89.690 MHz on a Vertex 2P FPGA kit. The architecture has been implemented using VHDL on a Xilinx Vertex 2P XC2VP2fg256 -6 FPGA with simulation on the Model Sim 6.2c from Mentor Graphics Corporation.

ACKNOWLEDGMENT

Authors are thankful to the "Center for Microprocessor Application for Training Education and Research", "Project on Storage Retrieval and Understanding of Video for Multimedia" of Computer Science & Engineering Department, Jadavpur University, for providing infrastructural facilities during progress of the work. Two of the authors, Mr. Abul Hasnat and Dr. Santanu Halder are thankful to Government College of Engineering and Textile Technology, Berhampore for kindly permitting him to carry on the research work.

REFERENCES

1. M. J. Schulte, J. E. Stine and K. E. Wires, "High-Speed Reciprocal Approximations", Signals, Systems & Computers, 1997. Conference Record of the Thirty-First Asilomar Conference on Volume 2, 2-5 Nov. 1997 pp: 1183 - 1187 vol.2.
2. Michael John, and Sebastian Smith. (1997). Application Specific Integrated Circuits, Pearson Education.
3. Jenkins, Jesse H. (1994). Designing with FPGAs and CPLDs, Prentice-Hall Publications.
4. Weste Neil H. and Eshraghian, Kamran (2000). Principles of CMOS VLSI Design: A Systems Perspective, Pearson Education Asia.
5. Wakerly, John F. (2002). Digital Design: Principles and Practices, Pearson Education Asia.
6. Institute of Electrical and Electronics Engineers, New York, NY. ANSI/IEEE 754-1985 standard for Binary Floating-Point Arithmetic, 1985.