TIME OPTIMIZED FIXED POINT ALGORITHM FOR UNSIGNED DIVISION OPERATION IN PROGRAMMABLE LOGICAL DEVICES

Boyko Baev Petrov

Department of Electronics, Technical University – Sofia, branch Plovdiv, Bulgaria
E-mail: abpetrov@persecteam.com

Keywords: division, verilog, divider, FPGA, CPLD

The division operation is one of the basic arithmetical operations. Unfortunately, the known hardware description languages HDL such as ABEL, Cupl, VHDL and Verilog don't support this operation in their common libraries. For many applications such as linear interpolation and linear approximation there is a necessary to use division of two signed or unsigned integer operands. The known method "pipeline divider" is designed by pipeline architecture and it has a latency time depend of the length of input operands. In many applications this latency time is unacceptable.

To make division between x (dividend) and y (divisor) is equal to find a new value z, that $z.y = x$. To find a value of z we suggest using $z = \frac{x}{y} = x.\,1/y = x.f(y)$. After numerical experiment we found that number of different values of $f(y)$ is not so match - match less then different values of y (in given number of bits per word), so it is not need to calculate $f(y)$, but it is useful to choose the value of $f(y)$ from before-calculated table of values.

After these considerations, we suggest the main structure of implementation of this method is to be a true-table with combinatorial multiplier. The true-table values are before-calculated values of $f(y)$.

The method is implemented in HDL Verilog for 8-bit width of y. The number of values in true-table is 57. These values are calculated by Windows based Borland Builder C compiler. The implementation is made when x is 8-bit word and z is 16-bit word – 8-bits for integer part of the quotient and 8-bits of fractional part of the quotient. The implemented module is verified by ModelSim verification tool and fitted in CPLD and FPGA Xilinx devices by Xilinx ISE tool. The post-fit asynchronous delays after fitting shows that clock frequency for Xilinx XC95288XL device is up to 20MHz and clock frequency for Xilinx Spartan 2 or Xilinx Virtex devices is up to 125MHz.

1. INTRODUCTION

The division operation is one of the basic arithmetical operations. Unfortunately, the known hardware description languages HDL such as ABEL, Cupl, VHDL and Verilog do not support this operation in their common libraries. For many applications such as linear interpolation and linear approximation there is a necessary to use division of two signed or unsigned integer operands.

At now, a several division methods and algorithms are known:

- digit-recurrence method [1];
- multiplicative method [1];
- division algorithm Liddicoat and Flynn [3];
- special methods such as Cordic and continued product methods [1].
- subtractive (restoring, non restoring and SRT) [1], [4];
- multiplicative (Newton-Raphson, Binomial) [1], [2];
- expansion in Taylor and Maclaurin series and other various approximation methods [1], [7];
- table look up method (bipartite tables) [1];
- division by reciprocation $z = \frac{x}{y} = x.\,1/y = x.f(y)$ [7].

The implementations of these methods are based on one of the next main architectures:

- sequential
- combinatorial, where the realization may be done by:
  - pipelined organization or
  - nonpipelined organization.
- combinatorial/ sequential

For many applications, especially for real-time applications with limited error factor, it is need to have a method and device for integer signed/unsigned division that can produce the result of the next active edge of given global synchronous clock signal.

The variant of look-up table and method by reciprocation and its implementation, time-optimized for programmable logical device is described in this paper.

2. PROBLEM STATEMENT

To make division between x (dividend) and y (divisor) is equal to find a new value z, that $z.y = x$. To find a value of z we suggest using $z = \frac{x}{y} = x.\,1/y = x.f(y)$. After numerical experiment we found that number of different values of $f(y)$ is not so match - match less then different values of y (if different values of y are 256 - different values of $f(y)$ are 47 only). For realization of this approach it is not need to calculate values of $f(y)$, but it is useful to choose the value of $f(y)$ from before-calculated table of values (look-up table). In programmable logical applications, this table is minimized to simple combinatorial equations for logical realization in single logical cell for each equation.

After these considerations we suggest the main structure of implementation of this method is to be a true-table with combinatorial multiplier. The true-table values are before-calculated values of $f(y)$. Fig.1 shows the main approach that is realized in this paper. In Special Cases Recognition module are recognised every one of the next special cases:

- $y = x$ that is always mean that $z = 1$
- $y = 1$ that is always mean that $z = x$
- in all other cases $z = x.f(y)$

Note, if $y = 0$ the value of z is postulated to be equal of 0 by look-up table

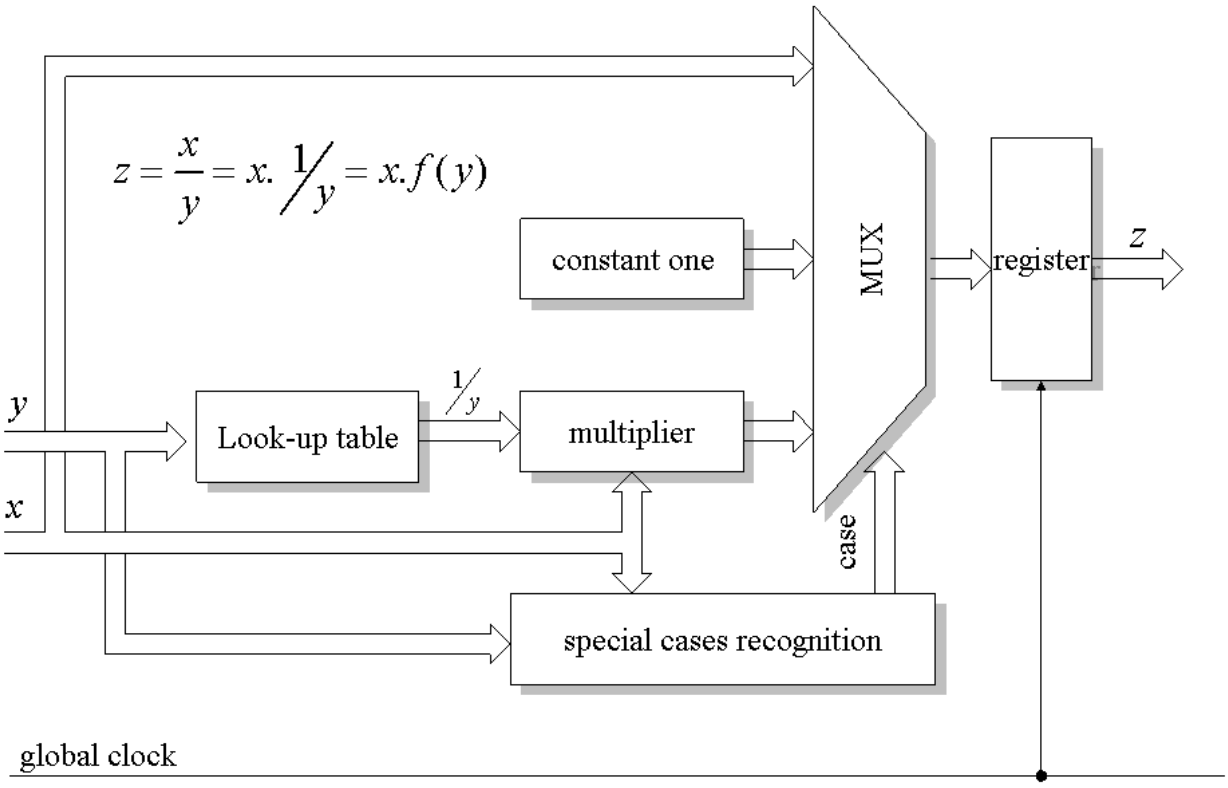

Fig.1. Main approach for divisor realization

As it is can see from fig.1, for gitter minimization after combinatorial realization, the output value of z is formed from output register (gitter-free device), synchronized by global clock signal.

The Verilog implementation of this approach is given here:

```
module div(in_x,in_y,clk,enb,out_x_y);
input [7:0] in_x;
input [7:0] in_y;
input clk,enb;
output [15:0] out_x_y;
reg [15:0] out_x_y;
reg [7:0] out;

always @ (in_y)
 begin
```

```
  casex  (in_y)
8'b00000010 : out = 8'h80;
8'b00000011 : out = 8'h55;
8'b00000100 : out = 8'h40;
8'b00000101 : out = 8'h33;
8'b00000110 : out = 8'h2B;
8'b00000111 : out = 8'h25;
.
.
.
8'b10101010 : out = 8'h02;
8'b10101011 : out = 8'h01;
8'b101011?? : out = 8'h01;
8'b1011???? : out = 8'h01;
8'b11?????? : out = 8'h01;
default :        out = 8'h00;
 endcase
end

always @ (posedge clk)
 begin
  if (enb)
   begin
    if (in_y == in_x) out_x_y <=16'h0100;
            else  if (in_y == 8'h01) out_x_y <={in_x,8'h00};
                    else  out_x_y <= out * in_x;
   end
 end
endmodule
```

3. RESULTS

For behavior simulation of Verilog implementation the next test bench module is written:

```
`timescale 1ns/100ps
module testbench();
   reg [7:0] in_x;
   reg [7:0] in_y;
   reg clk;
   reg enb;
   wire [15:0] out_x_y;
top_level uut (
       .clk(clk),
       .enb(enb),
       .in_x(in_x),
       .in_y(in_y),
       .out_x_y(out_x_y)
       );
       initial begin
// start condition
         in_x = 0;          in_y = 1;                clk = 0;          enb = 0;
```

```
// set the input operands and control signals
#4    in_x = 8'h3c;    in_y = 8'h01;    clk = 0;    enb = 1;    #4 clk=1;
#4    in_x = 8'h3c;    in_y = 8'h02;    clk = 0;    enb = 1;    #4 clk=1;
#4    in_x = 8'h3c;    in_y = 8'h03;    clk = 0;    enb = 1;    #4 clk=1;
#4    in_x = 8'h3c;    in_y = 8'h04;    clk = 0;    enb = 1;    #4 clk=1;
#4    in_x = 8'hff;    in_y = 8'h0d;    clk = 0;    enb = 1;    #4 clk=1;
#4    in_x = 8'h2f;    in_y = 8'h2f;    clk = 0;    enb = 1;    #4 clk=1;
#4    in_x = 8'ha4;    in_y = 8'h07;    clk = 0;    enb = 1;    #4 clk=1;
      end
endmodule
```

This module are implemented by Xilinx ISE [8] synthesis tool and tested by ModelSim [9] verification tool. The test conditions are shown in Tabl.1. and the waveform of simulation are shown in Fig.2

Tabl.1

Test number	Dividend	Divisor	Right result	Obtain result
1	*0x3C*	*0x01*	*0x3C00*	*0x3C00*
2	*0x3C*	*0x02*	*0x1E00*	*0x1E00*
3	*0x3C*	*0x03*	*0x1400*	*0x1400*
4	*0x3C*	*0x04*	*0x0F00*	*0x0F00*
5	*0xFF*	*0x0D*	*0x139D*	*0x13EC*
6	*0x2F*	*0x2F*	*0x0100*	*0x0100*
7	*0xA4*	*0x07*	*0x176D*	*0x17B4*

This module has fitted in some of FPGA and CPLD Xilinx devices and the fitting results are shown in Tabl.2.

Tabl.2

parameter	Cells-CPLD Slices-FPGA	Percent of usege	Max delay of global clock	Max global clock freguency
XC2C256	107	42%	105.7ns	9.5MHz
XC2S150-5	77	4%	7.748 ns	129MHz
XC2V250-6	35	2%	9.535 ns	105MHz

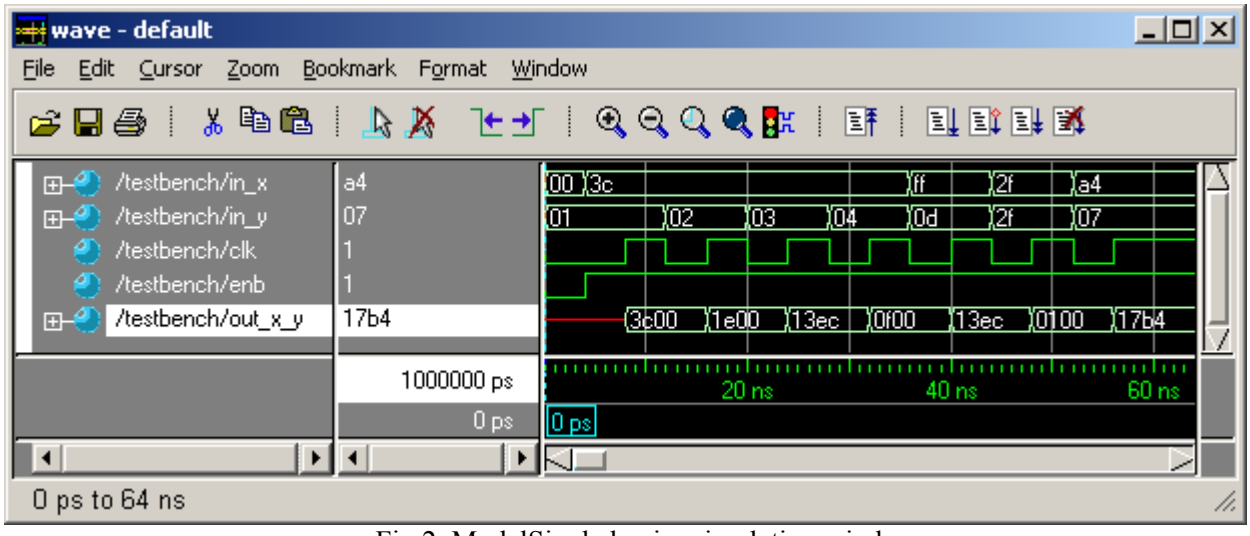

Fig.2. ModelSim behavior simulation window

4. CONCLUSIONS

According these (and others not shown here) results, it is can reach conclusion that now suggested method has the next advantages and faults:

- single cycle operation;
- can be fitted in many FPGA and CPLD logical devices;
- existing of error in remainder that make it unusable.

In conclusion this method can be use in real-time non critical error integer application such as signal processing, linear approximation, linear interpolation and many other tasks.

5. REFERENCES

[1] Milos D. Ercegovac, Tom_as Lang. *DIGITAL ARITHMETIC*, Computer Science Department University of California Los Angeles and Department of Electrical and Computer Engineering University of California at Irvine, Viewgraphs Copyright 2003, Elsevier Science 2004, Morgan-Kauffman Publishers 2004 http://www.cs.ucla.edu/digital_arithmetic/files/

[2] Taek-Jun Kwon, Joong-Seok Moon, Jeff Sondeen, Jeff Draper. *A 0.18μm IMPLEMENTATION OF A FLOATING-POINT UNIT FOR A PROCESSING-IN-MEMORY SYSTEM*, USC Information Sciences Institute 4676 Admiralty Way, Marina del Rey, CA 90292, U.S.A.

[3] Marc Joye and Karine Villegas. *A Protected Division Algorithm*, P. Honeyman, Ed., Fifth Smart Card Research and Advanced Application Conference (CARDIS '02), pp. 69–74, Usenix Association, 2002.], Gemplus Card International, Card Security Group La Vigie, Avenue des Jujubiers, ZI Athelia IV, 13705 La Ciotat Cedex, France http://www.geocities.com/MarcJoye/ http://www.gemplus.com/smart/

[4] David L. Harris, Stuart F. Oberman, and Mark A. Horowitz. *SRT Division Architectures and Implementations*, Computer Systems Laboratory, Stanford University, Stanford, CA 94305 fharrisd, oberman, horowitzg@leland.stanford.edu; dh_arith_97.pdf

[5] M. J. Flynn. *Multiply iteration and an introduction to Divide*, Computer Architecture & Arithmetic Group, Stanford University http://www.uspto.gov/patft/index.html

[6] Jen-Shiun Chiang, Hung-Da Chung and Min-Show Tsai. *Carry-Free Radix-2 Subtractive Division Algorithm and Implementation of the Divider,* Department of Electrical Engineering Tamkang University Tamsui, Taipei, Taiwan, Tamkang Journal of Science and Engineering, Vol. 3, No. 4, pp. 249-255 (2000), pp 249-255

[7] Mary Jane Irwin. *Computer Arithmetic*, CSE 575, MJIrwin, PSU, 2002 www.cse.psu.edu/~mji/cse575-convergencedivide.pdf

[8] www.xilinx.com

[9] www.modeltech.com