Implementation of 8-bit Booth's Restoring Division in Verilog

Chunduri Sai Abhishek[1]

[1]*Indian Institute of Space Science and Technology, Trivandrum*
[1]*chunduri.sc18b114@ug.iist.ac.in*

1 Algorithm and Flowcharts

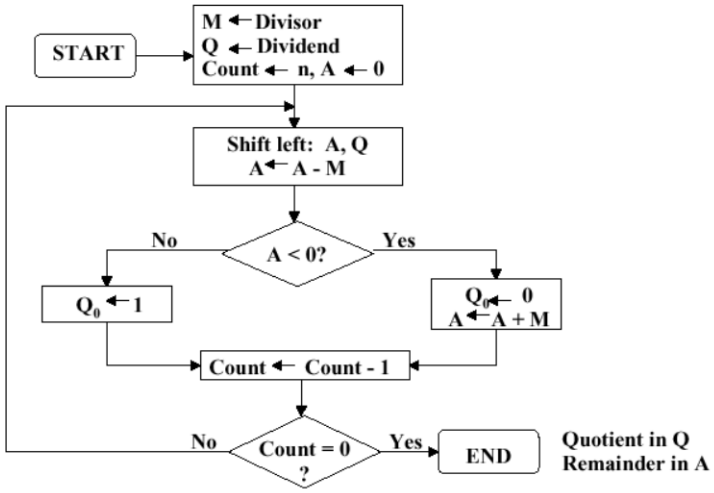

Figure 1: Algorithm gor unsigned division

A	Q	M = 0011	
0000	0111	Initial values	
0000	1110	Shift	1
1101		A = A - M	
0000	1110	A = A + M	
0001	1100	Shift	2
1110		A = A - M	
0001	1100	A = A + M	
0011	1000	Shift	3
0000		A = A - M	
0000	1001	$Q_0 = 1$	
0001	0010	Shift	4
1110		A = A - M	
0001	0010	A = A + M	

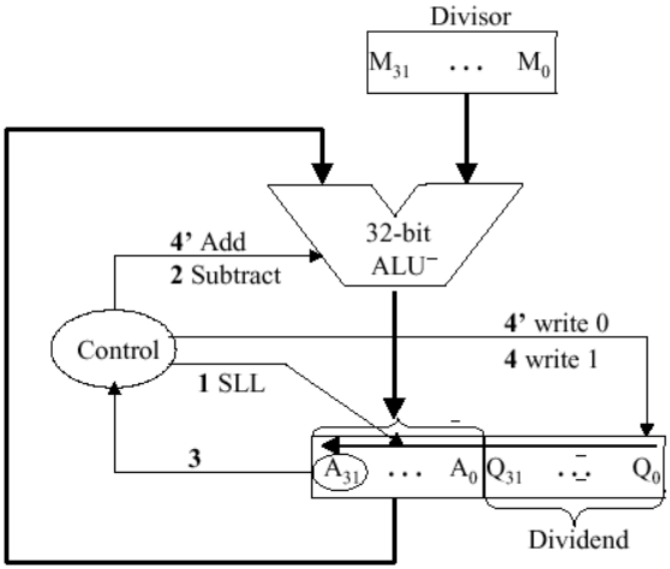

Figure 2: Circuit Implementation

(+7) / (+3): Q = 2; R = 1
(-7) / (+3): Q = -2; R = -1
(+7) / (-3): Q = -2; R = 1
(-7) / (-3): Q = 2; R = -1

Figure 3: Conditions used for evaluating signed division

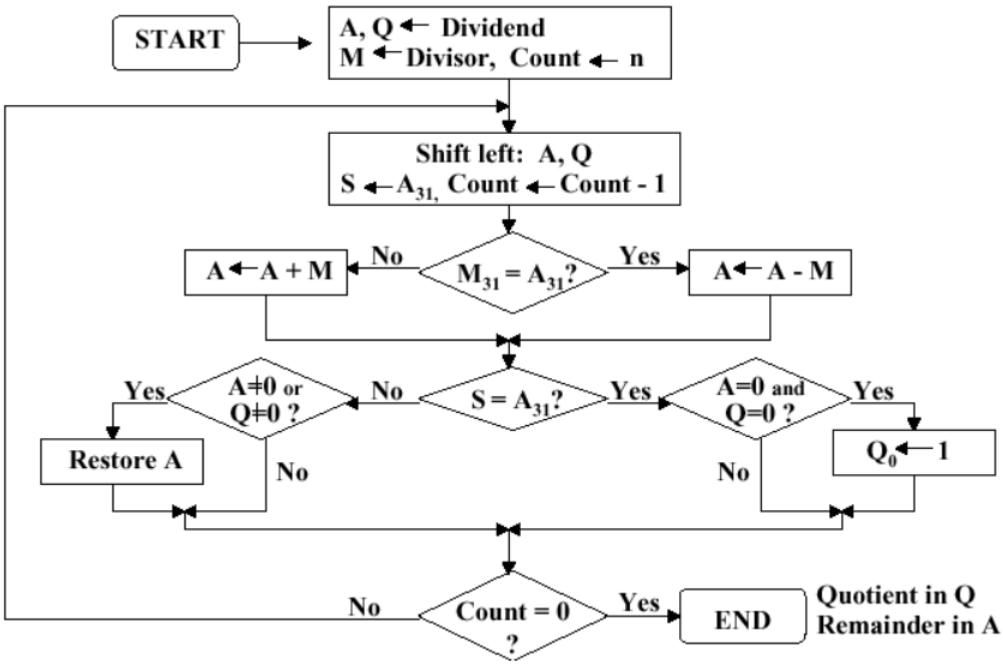

Figure 4: Final Algorithm Implemented

2 Verilog Codes

2.0.1 Restoring Division Algorithm

```
1  `timescale 1ns / 1ps
2
3  module restoring_divider(Q,M,Quo,Rem);
4
5  /* 8-bit Divider , both Q and M can be max 8-bits each */
6
7
8  input [7:0] Q;  //Dividend//
9  input [7:0] M; //Divisor//
10
11 output [7:0] Quo;
12 output [7:0] Rem;
13
14 // Internal Variables //
15
16 reg [7:0] Quo = 0;
17 reg [7:0] Rem = 0;
18 reg [7:0] Q_temp=0;
19 reg [7:0] M_temp = 0; // Loop Variables //
20 reg [7:0] Acc = 0;    // Accumilator 'A' //
21
22   integer i;      // Keeps track of how many times loop must run
       //
23
24   always@(*)
25     begin
```

```
26
27      //Initialize the Registers //
28
29      Q_temp=Q;
30      M_temp=M;
31      Acc=0;    //Initialize Accumilator with zero //
32
33
34 //*****************Converting Dividend and Divisors into
    positive numbers if they are negative ****************** //


35
36      if (Q_temp[7]==1) begin //i.e Q is negative //
37      Q_temp = 0-Q_temp;     // Now Q is positive //
38      end
39
40      if(M_temp[7] == 1) begin
41      M_temp = 0-M_temp;
42      end
43
44      if ((M_temp[7]==1) && (Q_temp[7]==1)) begin
45      Q_temp = 0-Q_temp;
46      M_temp = 0-M_temp;
47      end
48
49 //************************************** Algorithm Block
    ************************************************//
50          for(i=0;i<8 ;i=i+1) begin
51
52           Acc = {Acc[6:0],Q_temp[7]} ; //Left shift of A//
53           Q_temp[7:1] = Q_temp[6:0] ;     //Left Shift of Q //
54           Acc = Acc - M_temp ; // A = A-M //
55
56
57               if (Acc[7] == 1) begin // Checking A < 0 or not i.
    e MSB = 1 //
58
59                   Q_temp[0] = 0 ; // Making Q0 = 0 if step was
    unsuccessful //
60
61                   Acc = Acc + M_temp ;// A=A+M //
62
63                   end
64               else
65                   begin
66                   Q_temp[0] = 1 ;// Making Q0 = 1 is step was
    successful //
67               end
68          end
69 //********************** Adjust signs of Quotient and Reminder
    According to Q and M **********************//
70
71 /* LOGIC USED
72        Q>0 AND M>0 ---> THEN QUOTIENT >0 AND REMINDER >0
73        Q<0 AND M>0 ---> THEN QUOTIENT <0 AND REMINDER <0
```

```
        Q>0 AND M<0 ---> THEN QUOTIENT <0 AND REMINDER >0
        Q<0 AND M<0 ---> THEN QUOTIENT >0 AND REMINDER <0
*****************************************************************
   */


  if((Q[7]==1) && (M[7] ==0))
  begin
  assign Quo = 0-Q_temp;
  assign Rem = 0-Acc;
  end

  else if((Q[7]==0) && (M[7] ==1))
  begin
  assign Quo = 0-Q_temp;
  assign Rem = Acc;
  end

  else if((Q[7]==1) && (M[7] ==1))
  begin
  assign Quo = Q_temp;
  assign Rem = 0-Acc;
  end


  else
  begin
  assign Quo = Q_temp;
  assign Rem = Acc;
  end

end
endmodule
```

2.0.2 Testbench

```
`timescale 1ns / 1ps


module testbench;

  // Inputs
  reg [7:0] Q;
  reg [7:0] M;

  // Outputs
  wire [7:0] Quo;
  wire [7:0] Rem;

  // Instantiate the Unit Under Test (UUT)
  restoring_divider uut (
    .Q(Q),
    .M(M),
    .Quo(Quo),
    .Rem(Rem)
```

```
20    );
21
22    initial begin
23      // Initialize Inputs
24      Q = 9;
25      M = 4;
26      #100;
27
28      Q = -9;
29      M = 4;
30      #100;
31
32      Q = 9;
33      M = -4;
34      #100;
35
36      Q = -9;
37      M = -4;
38      #100;
39
40
41
42
43
44
45
46    end
47
48  endmodule
```

3 Simulation Results