Radix-4 Booth Multiplier

By

Puni Subramanya Sai Raghava - 2021H1230210H

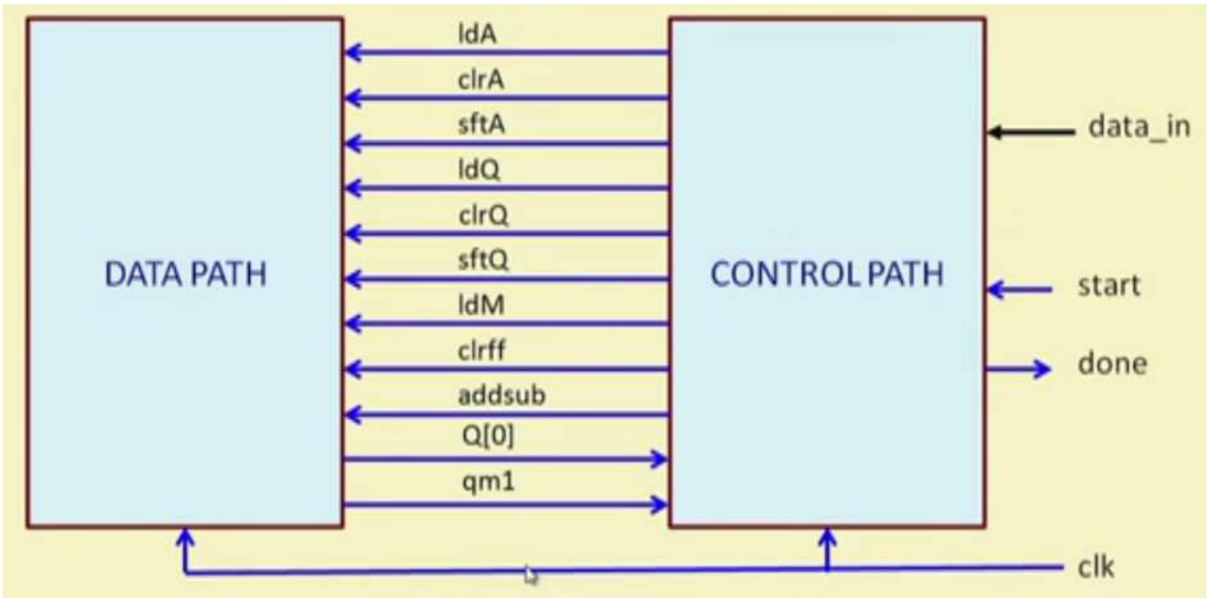
Shashank Nath – 2021H1230227H

Sravanth Reddy M - 2021H1230236H

Title: Implementation of 8-bit Radix-4 Booth Multiplier using Verilog HDL.

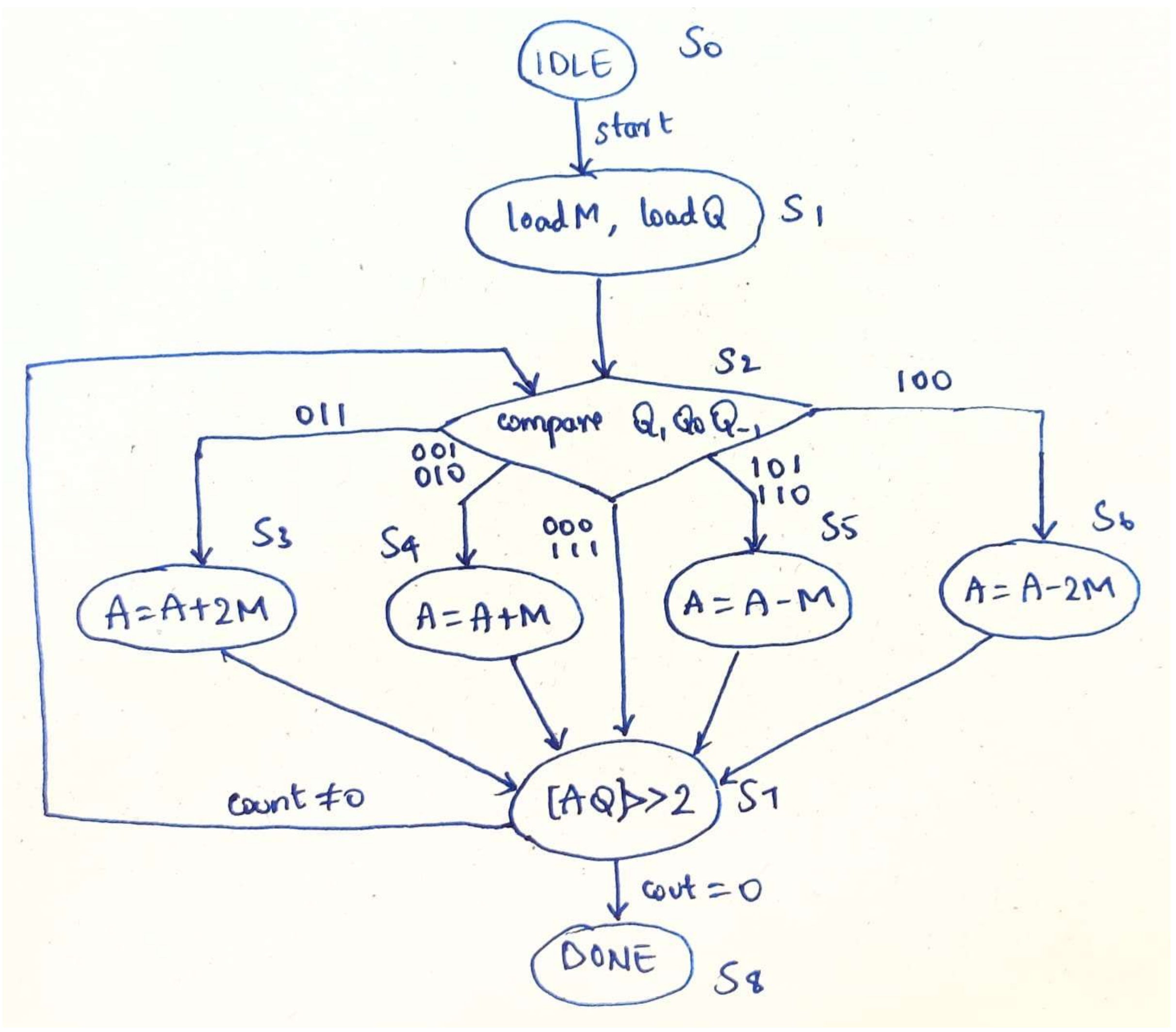
Agenda: Multiplication of given two 8-bit numbers using Radix-4 Booth algorithm needs to be performed. Implementation is done using data path and control path with Moore FSM.

The following diagram shows the overview of signal flow between Data Path module and Control Path module:



In Data Path all modules which work on the main data are instantiated. Using Moore FSM of 9 states the Control Path is designed which give output control signals that control the flow of data in Data Path.

Below mentioned figure shows the FSM implementation of control path:



Data Path contains the following mentioned modules:

Shift Register: A sequential module which performs a 2 bit right shift with new data appended in the MSB.

```
module ShiftRegisterA(data_out,data_in,ld,clr,shift_in,shift,clk);
output reg [7:0] data_out;
input [9:0] data_in;
input [1:0] shift_in;
input ld,clr,shift,clk;
reg [9:0] temp;

always @(posedge clk) begin
    if(clr) data_out<=0;
    else if(ld) data_out<=data_in[7:0];
    else if(shift) begin
                        if(ld) data_out<={shift_in,data_out[7:2]};
                        else data_out<={data_out[7],data_out[7],data_out[7:2]};
                   end
end


endmodule
```

PIPO: Parallel-in-Parallel-out register for storing the value of Multiplier(M).

```
module PIPO1(OUT,IN,clk,ld,clr);

output reg [7:0]OUT;
input [7:0] IN;
input clk,ld,clr;

always @(posedge clk) begin
    if(clr) OUT<=0;
    else if(ld) OUT<=IN;
end

endmodule
```

ALU: Arithmetic Logic Unit to perform logical operations on the given two inputs. Operations performed over here are A+M, A-M, A+2M, A-2M.

```
module ALU(OUT,IN1,IN2,ALU_op);

output reg [9:0] OUT;
reg [9:0] TEMP1,TEMP2;
input [7:0]IN1,IN2;
input [1:0]ALU_op;

always @(*) begin
    TEMP1={IN1[7],IN1[7],IN1};
    TEMP2={IN2[7],IN2[7],IN2};
    case(ALU_op)
        2'b00: OUT=TEMP1+TEMP2;
        2'b01: OUT=TEMP1-TEMP2;
        2'b10: OUT=TEMP1+(TEMP2<<1);
        2'b11: OUT=TEMP1-(TEMP2<<1);
    endcase
end

endmodule
```

Counter: Counter loaded with an initial value of 4 and decremented till zero.

```
module Counter(count,ld_count,clk,decr);

output reg [2:0]count;
input clk,decr,ld_count;

always @(posedge clk) begin
    if(ld_count) count<=3'b100;
    else if(decr) count<=count-1;
    else count<=count;
end

endmodule
```

Data Path: Instantiates all the submodules and connects them for proper data flow.

```
module
DataPath(Q1,Q0,Qm1,eqz,ldA,shiftA,clrA,ldQ,shiftQ,clrQ,decr,ld_count,clrff,
ldM,clrM,ALU_op,Q_in,M_in,clk);

output Q1,Q0,Qm1,eqz;
input ldA,shiftA,clrA,ldQ,shiftQ,clrQ,decr,ld_count,clrff,ldM,clrM,clk;
input [7:0] Q_in,M_in;
input [1:0]ALU_op;

wire [7:0] A,Q,M;
wire [9:0] ALU_OUT;
wire [2:0] count;

assign eqz=~|count;
assign Q1=Q[1];
assign Q0=Q[0];

ShiftRegisterA Areg(A,ALU_OUT,ldA,clrA,ALU_OUT[9:8],shiftA,clk);
ShiftRegister Qreg(Q,Q_in,ldQ,clrQ,{A[1],A[0]},shiftQ,clk);
PIPO1 Mreg(M,M_in,clk,ldM,clrM);
ALU Booth4ALU(ALU_OUT,A,M,ALU_op);
Counter count8(count,ld_count,clk,decr);
Dff Qm1ff(Qm1,Q[1],clk,clrff);

endmodule
```

Control Path: Control signals for data path are generated using Moore FSM of 9 States.

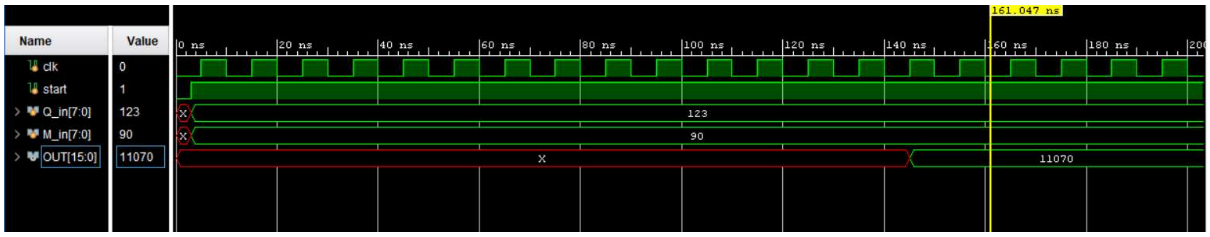
```
module
ControlPath(ldA,shiftA,clrA,ldQ,shiftQ,clrQ,decr,ld_count,clrff,ldM,clrM,AL
U_op,done,clk,start,Q1,Q0,Qm1,eqz);

output                                                                  reg
ldA,shiftA,clrA,ldQ,shiftQ,clrQ,decr,ld_count,clrff,ldM,clrM,done;
output reg [1:0] ALU_op;
input start,clk,Q1,Q0,Qm1,eqz;

parameter
IDLE=4'b0000,LOAD=4'b0001,COMP=4'b0010,A1=4'b0011,A2=4'b0100,A3=4'b0101,A4=
4'b0110,SHIFT=4'b0111,DONE=4'b1000;
reg [3:0]PS=0,NS;

always @(posedge clk) PS<=NS;

always @(PS) begin
    case(PS)
        IDLE:   begin  ldA=0;  shiftA=0;  clrA=1;  ldQ=0;  shiftQ=0;  clrQ=1;
decr=0; ld_count=0; clrff=1; ldM=0; ALU_op=0; clrM=0; done=0; end
        LOAD:  begin ldQ=1; ldM=1; clrA=0; clrQ=0; clrff=0; ld_count=1; end
        COMP:   begin  shiftA=0;  shiftQ=0;  decr=0;  ldQ=0;  ldM=0;  ld_count=0;
end
        A1:    begin ALU_op=0; ldA=1; decr=0; shiftA=0; shiftQ=0; end
        A2:    begin ALU_op=1; ldA=1; decr=0; shiftA=0; shiftQ=0; end
        A3:    begin ALU_op=2; ldA=1; decr=0; shiftA=0; shiftQ=0; end
        A4:    begin ALU_op=3; ldA=1; decr=0; shiftA=0; shiftQ=0; end
        SHIFT: begin shiftA=1; shiftQ=1; ldA=0; decr=1; end
        DONE:  begin done=1; shiftA=0; shiftQ=0;  end
        default: begin ldA=0; shiftA=0; clrA=0; ldQ=0; shiftQ=0; clrQ=0;
decr=0; ld_count=0; clrff=0; ldM=0; ALU_op=0; clrM=0; done=0;   end
    endcase
end

always @(PS or Q1 or Q0 or Qm1 or eqz) begin
    case(PS)
        IDLE:  begin
                  if(start) NS=LOAD;
                  else NS=IDLE;
               end
        LOAD:  begin
                  NS=COMP;
               end
        COMP:  begin
                  if(({Q1,Q0,Qm1}==3'b000 || {Q1,Q0,Qm1}==3'b111) && !eqz)
NS=SHIFT;
                  else if(({Q1,Q0,Qm1}==3'b001 || {Q1,Q0,Qm1}==3'b010) &&
!eqz) NS=A1;
                  else if(({Q1,Q0,Qm1}==3'b101 || {Q1,Q0,Qm1}==3'b110) &&
!eqz) NS=A2;
                  else if(({Q1,Q0,Qm1}==3'b011) && !eqz) NS=A3;
                  else if(({Q1,Q0,Qm1}==3'b100) && !eqz) NS=A4;
                  else if(eqz) NS=DONE;
               end
        A1:    begin NS=SHIFT; end
        A2:    begin NS=SHIFT; end
        A3:    begin NS=SHIFT; end
        A4:    begin NS=SHIFT; end
        SHIFT: begin
                  if (eqz) NS=DONE;
                  else NS=COMP;
               end
        DONE:  begin NS=DONE;  end
        default: begin NS=IDLE;   end
    endcase
end

endmodule
```

Testbench: Simulation of the main module performed with values of Multiplicand(Q) and Multiplier(M) provided in our testbench.

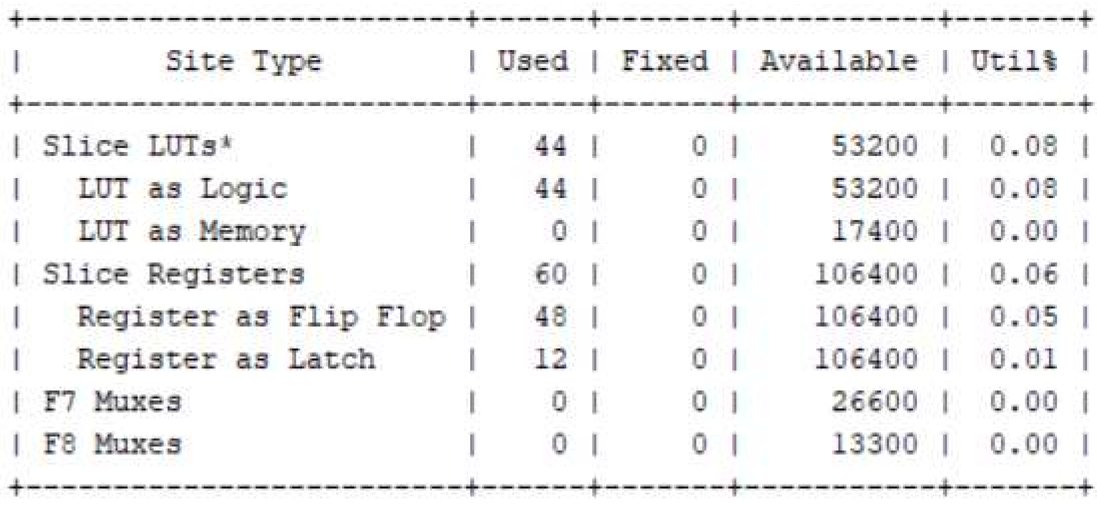
```
module Radix4Booth_tb();

reg clk,start;
reg [7:0] Q_in,M_in;
wire [15:0] OUT;
```

```
Radix4BoothMain dut(OUT,clk,start,Q_in,M_in);

initial begin
    clk=0; start=0;
    #3 start=1;
    Q_in=8'b01111011;
    M_in=8'b01011010;
    $write(" %c",Q_in[7]?45:32);
    $write("%d x ",Q_in[7]?(-Q_in):Q_in);
    $write("%c",M_in[7]?45:32);
    $write("%d",M_in[7]?(-M_in):M_in);
    #200 $finish;
end

always #5 clk=~clk;


endmodule
```

Simulation waveform: Q=123 M=90 Output=11070



Utilization Report:



```
+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |   44 |     0 |     53200 |  0.08 |
|   LUT as Logic          |   44 |     0 |     53200 |  0.08 |
|   LUT as Memory         |    0 |     0 |     17400 |  0.00 |
| Slice Registers         |   60 |     0 |    106400 |  0.06 |
|   Register as Flip Flop |   48 |     0 |    106400 |  0.05 |
|   Register as Latch     |   12 |     0 |    106400 |  0.01 |
| F7 Muxes                |    0 |     0 |     26600 |  0.00 |
| F8 Muxes                |    0 |     0 |     13300 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```