SHRI B.V.VS SANGHAS
BASAVESHVAR ENGINEERING COLLEGE (AUTONOMOUS)

BAGALKOT. KARNATAKA

Academic year 2022.2023

Department of Electronics and communication engineering

Report on the project
8 bit Multiplication using counter

Project Guide:

Prof M.C.Aralimarad

Project Associates :

Name.	USN
1, Farheen M Matte.	2BA20EC026
2.Pooja M Magadum.	2BA20EC048

Eight (8) bit Multiplier Using Counter

As the process of multiplication indicates, the control has to performs two functions—generating add or shift signals as needed and counting the number of shifts. If the number of bits is large, it is convenient to divide the control circuit into a counter and an add-shift control, as shown in Figure 2(a). First, we will derive a state graph for the add-shift control that tests Start and M and outputs the proper sequence of add and shift signals (Figure 2-(b)). Then we will add a completion signal (K) from the counter that stops the multiplier after the proper number of shifts have been completed. Starting in State 0 in Figure 2(b), when a start signal Start= 1 is received, a load signal is generated and the circuit goes to state 1 (1). Then if M =1, an add signal is generated and the circuit goes to state 2(2), if M = 0, a shift signal is generated and the circuit stays in State1(1) . In State 2(2) ,a shift signal is generated since a shift always follows an add. The graph of Figure 2(b) will generate the proper sequence of add and shift signals, but it has no provision for stopping the multiplier.

To determine when the multiplication is completed, the counter is incremented each time a shift signal is generated. If the multiplier is n bits, n shifts are required. We will design the counter so that a completion signal (K) is generated after n -1 shifts have occurred. When K = 1, the circuit should perform one more addition, if necessary, and then do the final shift. The control operation in Figure 2(c) is the same as Figure 2(b) as long as K = 0. In state (1) , if K =1, we test M as usual. If M = 0, we output the final shift signal and go to the done state 3(3); however, if M =1, we add before shifting and go to state 2(2) . In state 2(2) , if K=1 we output one more shift signal and then go to State 3(3) . The last shift signal will increment the counter to 0 at the same time the add-shift control goes to the done state.

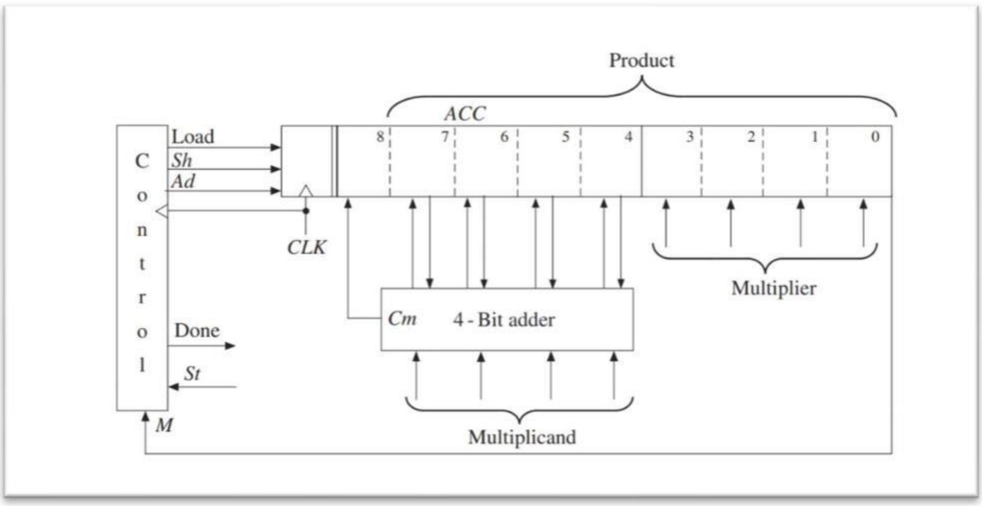


Figure -(1)

As an example, consider the multiplier of above figure (1) but replace the control circuit with Figure 2-(a). Since n =4, a 2-bit counter is needed to count the four shifts, and K =1 when the counter is in state 3 (111). Table ‘ shows the operation of the multiplier when 1101 is multiplied by 1011. State 0 ,state1 , State2 , and State3 represent states of the control circuit (Figure 2-(c)).

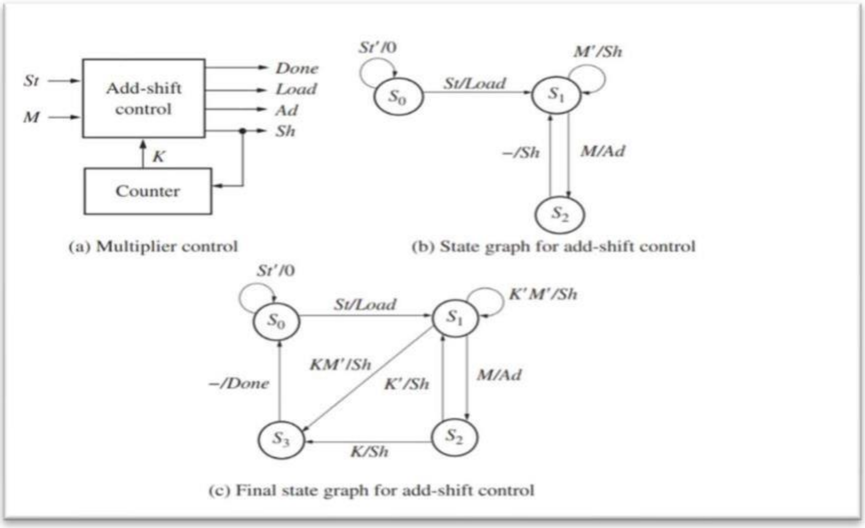


Figure 2 Multiplier control with counter

At time t 0, the control is reset and waits for a start signal. At time t 1 , the start signal Start is 1, and a Load signal is generated. At time t 2 , M = 1, so an Ad signal is generated. When the next clock occurs, the output of the adder is loaded into the accumulator and the control goes to State2 . At t 3 , an Sh signal is generated, so at the next clock shifting occurs and the counter is incremented. At t 4 , M =1 so Ad = 1, and the adder output is loaded into the accumulator at the next clock. At t 5 and t 6 , shifting and counting occur. At t 7 , three shifts have occurred and the counter state is 11, so K= 1. Since M addition occurs and control goes to State2 . At t 8 , Sh = K =1, so at the next clock the final shift occurs and the counter is incremented back to state 00. At t 9 , a Done signal is generated.

The multiplier design given here can easily be expanded to 8 bits simply by increasing the register size and the number of bits in the counter. The add-shift control would remain unchanged.so here we have written a code for 8-bit Multiplier using the counter and counting the number of shifts.

Time	State	Counter	Product Register	St	M	K	Load	Ad	Sh	Done
t_0	S_0	00	000000000	0	0	0	0	0	0	0
t_1	S_0	00	000000000	1	0	0	1	0	0	0
t_2	S_1	00	000001011	0	1	0	0	1	0	0
t_3	S_2	00	011011011	0	1	0	0	0	1	0
t_4	S_1	01	001101101	0	1	0	0	1	0	0
t_5	S_2	01	100111101	0	1	0	0	0	1	0
t_6	S_1	10	010011110	0	0	0	0	0	1	0
t_7	S_1	11	001001111	0	1	1	0	1	0	0
t_8	S_2	11	100011111	0	1	1	0	0	1	0
t_9	S_3	00	010001111	0	1	0	0	0	0	1

Table: Operation of multiplier Using Counter

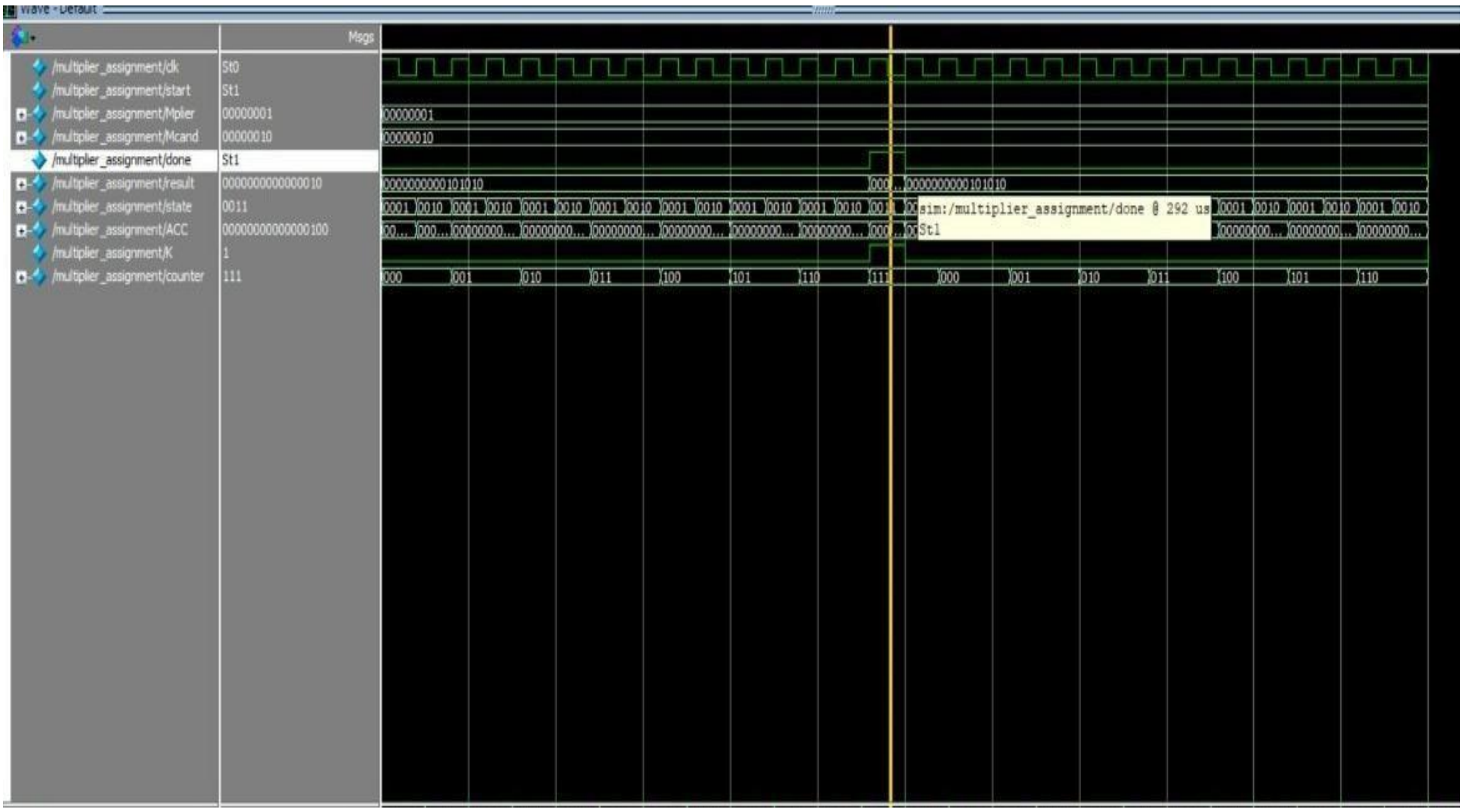
Verilog code

```
//multipluer using counter
`define M ACC[0]
module multiplier_assignment(clk,start,Mplier,Mcand,done,result);
 input clk;
 input start;
 input[7:0] Mplier;
input[7:0] Mcand;
output done;
output[15:0] result;
 reg [3:0] state;
 reg [16:0] ACC;
reg K;
reg [2:0]
counter;
 initial
begin
  state=0;
ACC=0;
end
always      @(posedge
clk)
begin
case(state)
```

```
0:      begin
      counter=0;
       K=0;
          if(start==1'b1)
                  begin
                        ACC[16:8] <= 9'b000000000;


                        ACC[7:0] <= Mplier;
                        state <= 1;
                      end
        end

1:          begin
        if(K==1'b1)
         state <= 3;
          else
               begin
                   if(`M==1'b1)
                        begin
                              ACC[16:8]<={1'b0,ACC[15:8]+Mcand;
                              State <= 2;
                         end
                    else
                      state <= 2;
               end
             end
 2:         begin
            ACC          <=       {1'b0,   ACC[16:1]};
             Counter=counter+1'b1;
```

```
            if(counter==3'b111)
                  begin
                        K=1'b1;
                       state <=3;
                  end
              else
                    state <= 1;


      end
  3:          begin
            ACC <= {1'b0, ACC [16:1]};
            K =1'b0;
            state <= 0;
              end
  endcase
end
  assign done = (state==3) ? 1'b1: 1'b0;
assign   result = (state==3) ? ACC [16:1] : 16’b0000000000101010;
endmodule
```

Output:



Wave - Default
Msgs
/multiplier_assignment/clk
St0
/multiplier_assignment/start
St1
/multiplier_assignment/Mplier
00000001
/multiplier_assignment/Mcand
00000010
/multiplier_assignment/done
St1
/multiplier_assignment/result
0000000000000010
/multiplier_assignment/state
0011
/multiplier_assignment/ACC
0000000000000100
/multiplier_assignment/K
1
/multiplier_assignment/counter
111
sim:/multiplier_assignment/done @ 292 us
St1