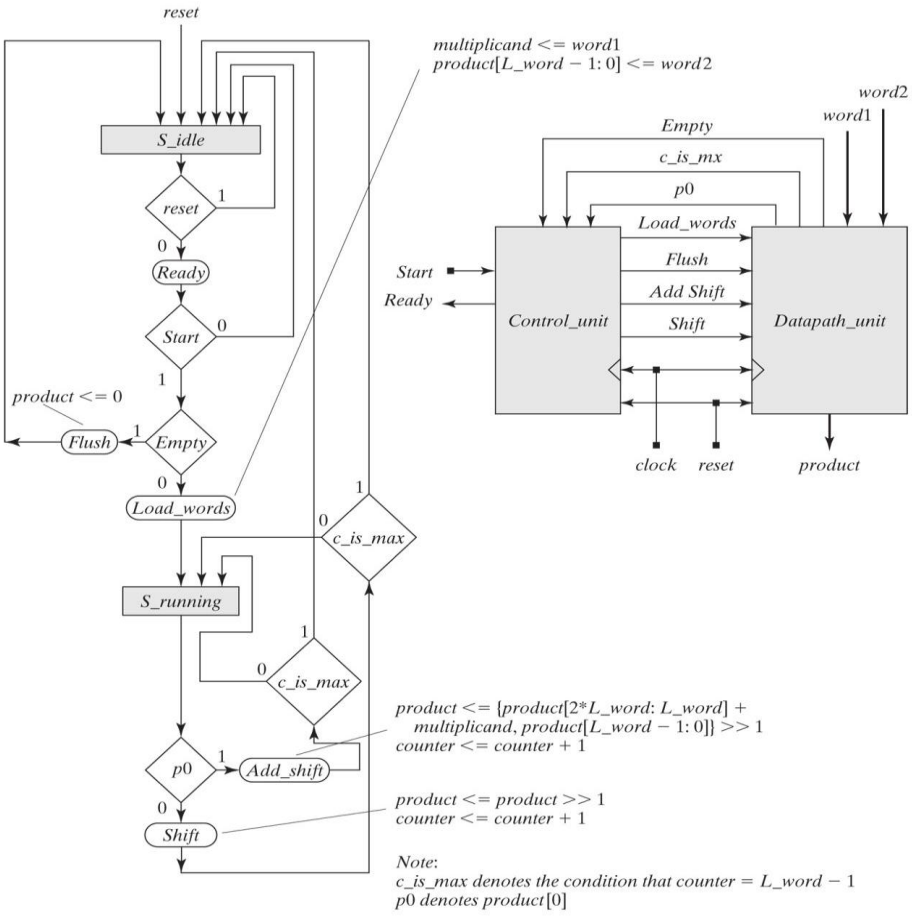
COE 405, Term 122

Design & Modeling of Digital Systems

HW# 5 Solution

Due date: Monday, April 15

Q.1. It is required to design an unsigned 4-bit sequential multiplier. The multiplier is assumed to have an 8-bit register to hold the result, a 4-bit register to hold the multiplicand and a 3-bit counter. When reset is 1, the multiplicand, the product and counter registers are reset. When operation is started, the multiplicand register is loaded with word1 while the least significant 4-bits of the product are loaded with word2. The block diagram and the ASMD chart of the sequential multiplier are given below. *Ready* signals that the unit is ready to accept a command to multiply. If word1 or word2 are 0, *Empty* is set to 1. *Flush* loads the product with 0. *C_is_mx* is set when the counter is equal to 3. *P0* is the least significant bit of the product i.e. product[0].



(i) Write a Verilog model to model the data-path.

```
module Multiplier_DPU #(parameter word_size=4)(
output [2*word_size-1: 0] product,
output Empty, c_is_mx, p0,
input [word_size-1:0] word1, word2,
input Load_words,Flush, Add_shift, Shift, Ready, clock, reset
);

reg [word_size-1:0] multiplicand;
reg [2*word_size:0] tproduct;
reg [1:0] counter;

assign product = tproduct[2*word_size-1:0];
assign p0 = tproduct[0];
assign w1z = ~| word1;
assign w2z = ~| word2;
assign Empty = w1z | w2z;
assign c_is_mx = & counter;

always @ (posedge clock)
  if (reset == 1'b1) counter <= 0;
  else if (Shift || Add_shift) counter <= counter + 1;

always @ (posedge clock)
  if (reset || Flush) tproduct <= 0;
  else if (Load_words) begin
    tproduct[word_size-1:0] <= word2;
  end
  else if (Shift) tproduct <= tproduct >> 1;
  else if (Add_shift)
    tproduct <= {tproduct[2*word_size:word_size]+{1'b0,multiplicand}, product[word_size-1:0]}
>>1;

always @ (posedge clock)
  if (Load_words) multiplicand <= word1;


endmodule
```

(ii) Write a Verilog model to model the control unit based on the ASMD chart i.e. not based on equations.

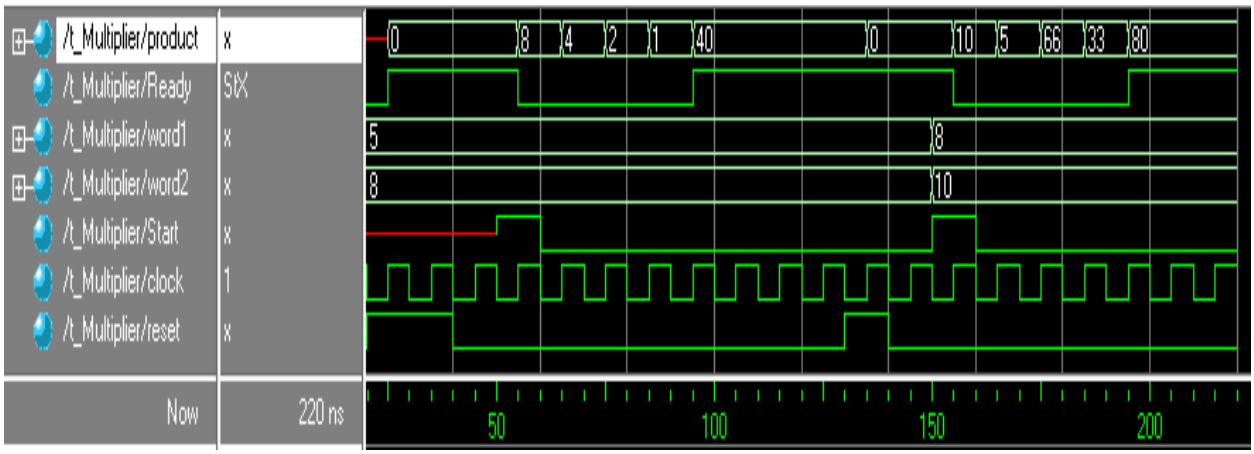
```
module Multiplier_CU  (output reg Load_words,
Flush, Add_shift, Shift, Ready,
input Start, p0, c_is_mx, Empty, clock, reset);

parameter S_idle = 1'b0, S_running=1'b1;
```

```
reg state, next_state;

 always @(posedge clock)
  if (reset) state <= S_idle;
  else state <= next_state;

 always @(state, Start, p0, c_is_mx, Empty) begin
  Load_words=0; Flush=0; Add_shift=0; Shift=0; Ready=0;
  case (state)
    S_idle: begin
       Ready=1;
       if (Start) begin
         if (Empty) begin
              next_state=S_idle; Flush=1; end
         else begin
              next_state=S_running; Load_words=1; end
         end
       else next_state=S_idle;
    end
    S_running:
          if (p0) begin
           Add_shift=1;
           if (c_is_mx) next_state=S_idle;
           else   next_state=S_running;
          end
          else begin
           Shift=1;
           if (c_is_mx) next_state=S_idle;
           else   next_state=S_running;
     end

   endcase
 end
endmodule
```

(iii) Write a Verilog test bench to test the correctness of a 4-bit sequential multiplier.

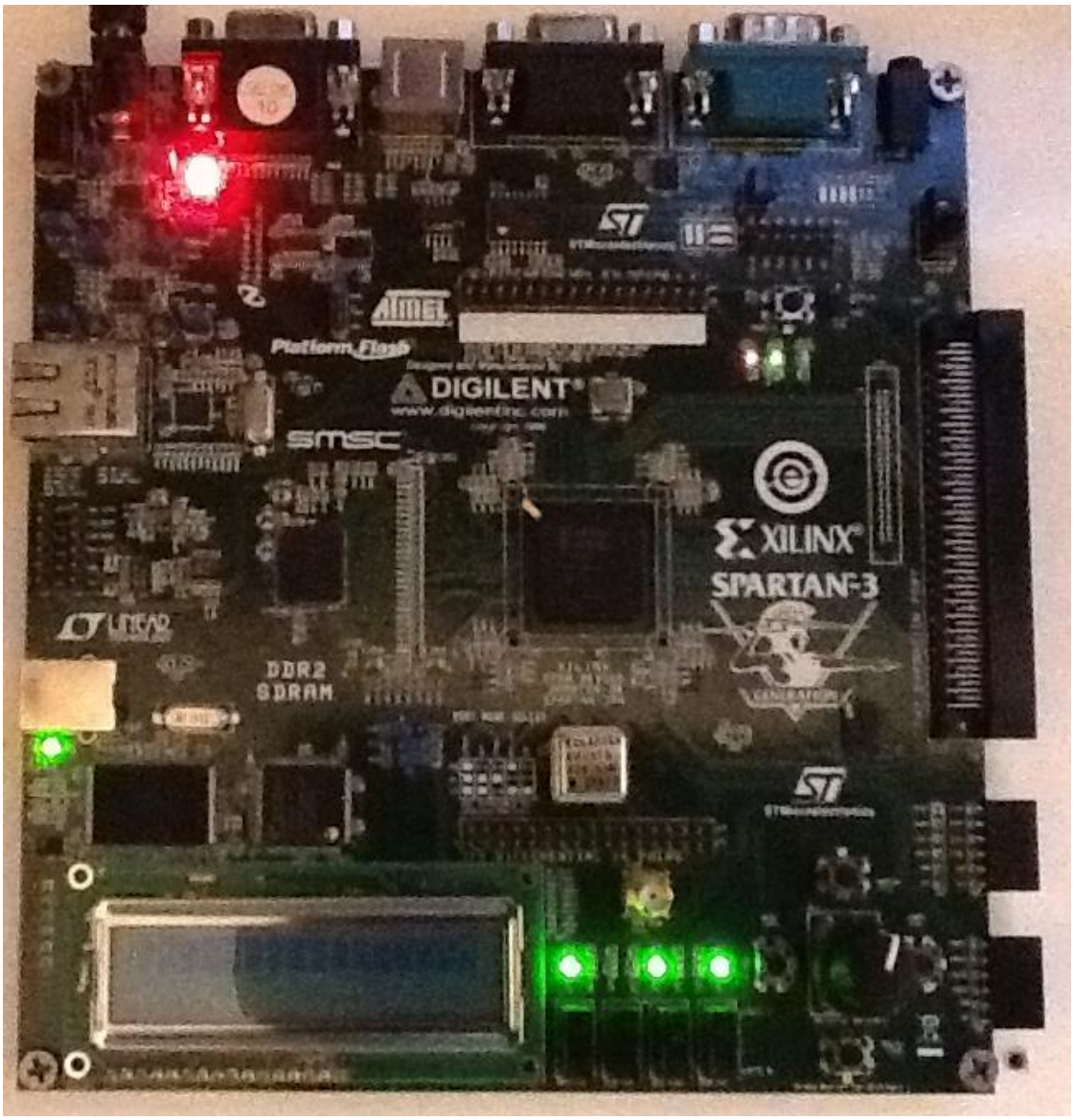
```
module Multiplier #(parameter word_size=4)(
output [2*word_size-1: 0] product,
output Ready,
input  [word_size-1:0] word1, word2,
input  Start, clock, reset
);


Multiplier_CU  M1 (Load_words,Flush, Add_shift, Shift, Ready,
Start, p0, c_is_mx, Empty, clock, reset);

Multiplier_DPU M2 ( product, Empty, c_is_mx, p0, word1, word2,
Load_words, Flush, Add_shift, Shift, Ready, clock, reset);

endmodule
```

```
module t_Multiplier ();

wire [7: 0] product;
wire Ready;
reg  [3:0] word1, word2;
reg  Start, clock, reset;

Multiplier M1 (product, Ready, word1, word2, Start, clock, reset);
initial #220 $finish;
initial begin clock = 0; forever #5 clock = ~clock; end
initial fork
#10 word1 = 'd5;
#10 word2 = 'd8;
#10 reset = 0; // Power-up reset
#20 reset = 1;
#40 reset = 0;
#50 Start = 1;
#60 Start = 0;
#130 reset = 1;
#140 reset = 0;
#150 word1 = 'd8;
#150 word2 = 'd10;
#150 Start = 1;
#160 Start = 0;
join
endmodule
```



(iv) Implement the 4-bit sequential multiplier using FPGA and demonstrate its correctness.

Input = 3 Output=9



Input = 5 Output=25