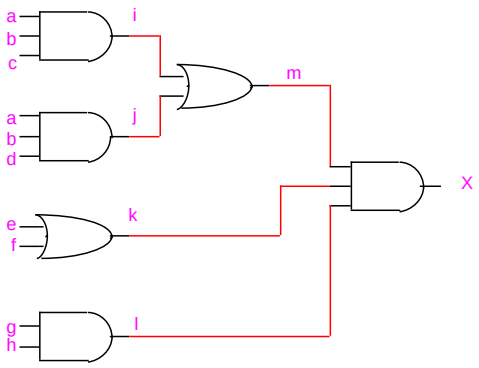
COE 405, Term 131

Design & Modeling of Digital Systems

HW# 5 Solution

Due date: Sunday, Dec. 8

Q.1. Consider the logic network below with inputs *{a, b, c, d, e, f, g, h}* and output *{X}*:



Assume that the delay of a gate is related to the number of its inputs i.e. the delay of a 2-input AND gate is 2. Also, assume that the input data-ready times are zero for all inputs.

(i) Compute the data ready times and slacks for all vertices in the network.

(i)

Data Ready Time	Required Time	Slack
$t_a = 0$	$\bar{t}_a = \min(0,0) = 0$	$S_a = 0$
$t_b = 0$	$\bar{t}_b = \min(0,0) = 0$	$S_b = 0$
$t_c = 0$	$\bar{t}_c = 0$	$S_c = 0$
$t_d = 0$	$\bar{t}_d = 0$	$S_d = 0$
$t_e = 0$	$\bar{t}_e = 3$	$S_e = 3-0 = 3$
$t_f = 0$	$\bar{t}_f = 3$	$S_f = 3-0 = 3$
$t_g = 0$	$\bar{t}_g = 3$	$S_g = 3-0 = 3$
$t_h = 0$	$\bar{t}_h = 3$	$S_h = 3-0 = 3$
$t_i = 3$	$\bar{t}_i = 3$	$S_i = 3-3 = 0$
$t_j = 3$	$\bar{t}_j = 3$	$S_j = 3-3 = 0$
$t_k = 2$	$\bar{t}_k = 5$	$S_k = 5-2 = 3$
$t_l = 2$	$\bar{t}_l = 5$	$S_l = 5-2 = 3$
$t_m = 5$	$\bar{t}_m = 5$	$S_m = 5-5 = 0$
$t_x = 8$	$\bar{t}_x = 8$	$S_x = 8-8 = 0$

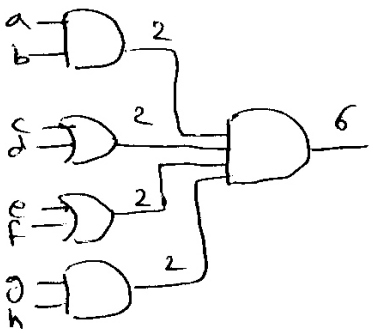
(ii) Determine the topological critical path(s).

(ii) The topological critical paths are:

{a, i, m, x}, {b, i, m, x}, {c, i, m, x},
{a, j, m, x}, {b, j, m, x}, {d, j, m, x}.

(iii) Suggest an implementation of the function *X* to reduce the delay of the circuit to the minimum possible and determine the maximum propagation delay in the optimized circuit. Has the area been affected?

(iii) To optimize the delay of the network, we need to improve the delay of nodes in the critical paths.
We rewrite m as ab(c+d) and then we have $x = m \times l = ab(c+d)(e+f)gh$ and we combine them into a 4-input AND gate to optimize both delay and area.
The resulting network is:

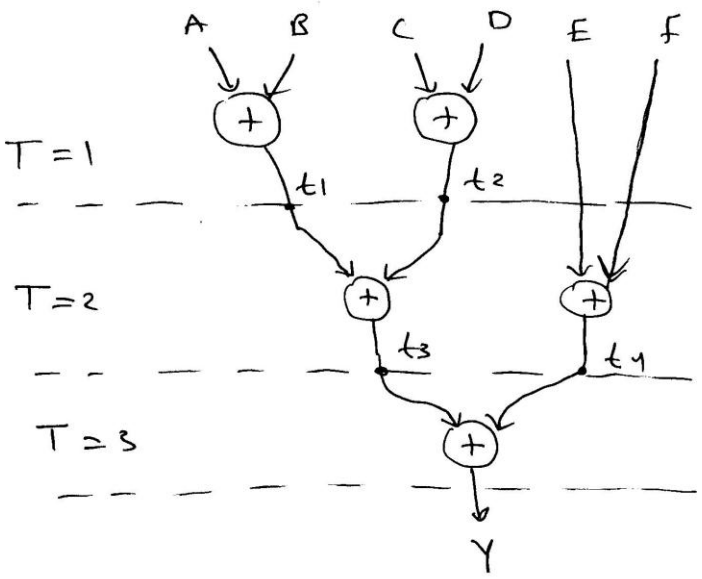


Resulting delay = 6
Number of literals = 12

Thus, we have improved both delay from 8 to 6 and area from 15 literals to 12.

Q.2. It is required to design a circuit to compute the equation Y=A+B+C+D+E+F using only two adders. Assume that the inputs will hold their values until the end of the operation. Also, assume that the inputs will be ready when a Start input is asserted and a Done signal will be set when the result is ready.

(i) Show a schedule with minimum latency (i.e., clock cycles) satisfying the area constraints.



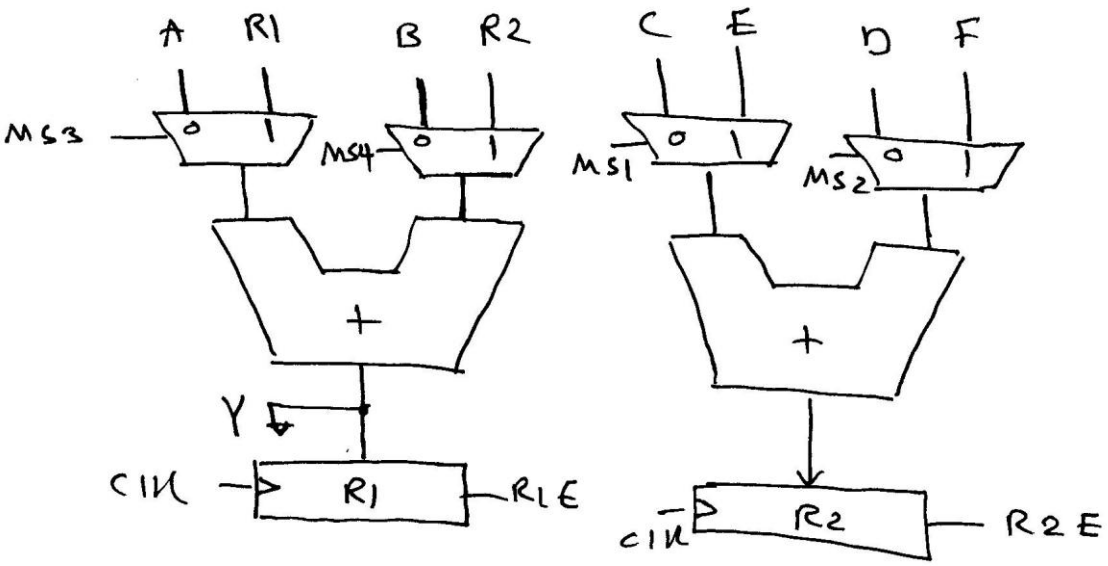
(ii) Show the DataPath design of your circuit.

we need two registers to store the temporary variables t_1, t_2, t_3 and t_4.

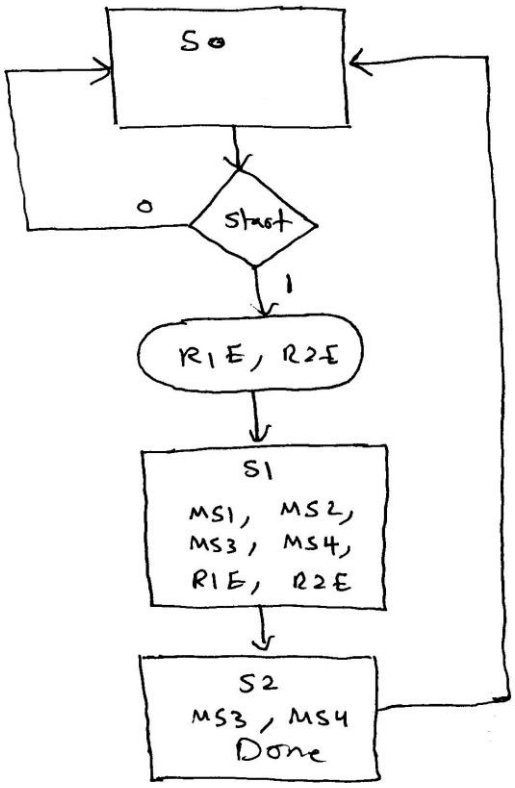
We will allocate t_1 and t_3 to R1 and t_2 and t_4 to R2.

We will also allocate operations t_1, t_3, and Y to the first adder while operations t_2 and t_4 to the second adder.

Data Path



(iii) Show the ASMD diagram of your control unit.



(iv) Model the DataPath and Control units in Verilog and verify the correctness of your model.

```
module ADD6 #(parameter n=4)(output done, output [n+1:0] Y,
input [n-1:0] A, B, C, D, E, F, input start, clk, reset);

DPADD6  M1 (Y, A, B, C, D, E, F, MS1, MS2, MS3, MS4, R1E, R2E, clk, reset);
CUADD6  M2 (done, MS1, MS2, MS3, MS4, R1E, R2E, start, clk, reset);

endmodule

module DPADD6 #(parameter n=4)
(output [n+1:0] Y, input [n-1:0] A, B, C, D, E, F,
input MS1, MS2, MS3, MS4, R1E, R2E, clk, reset);
reg [n+1:0] R1, R2;
wire [n+1:0] Mux1, Mux2, Mux3, Mux4, add1, add2;
assign Mux1 = MS1?E:C;
assign Mux2 = MS2?F:D;
assign Mux3 = MS3?R1:A;
assign Mux4 = MS4?R2:B;
assign add2 = Mux1 + Mux2;
assign add1 = Mux3 + Mux4;
assign Y = add1;
always @ (posedge clk, posedge reset) begin
  if (reset) begin
    R1 <= 0; R2 <= 0;
  end
  else begin
  if (R1E)
    R1 <= add1;
  if (R2E)
    R2 <= add2;
  end
end
endmodule



module CUADD6 (output reg done, MS1, MS2, MS3, MS4, R1E, R2E,
input start, clk, reset);

parameter S0 = 2'b00, S1=2'b01, S2=2'b10;
reg [1:0] state, next_state;

always @(posedge clk, posedge reset)
  if (reset) state <= S0;
  else state <= next_state;
```

```
always @(state, start) begin
  done=0; MS1=0; MS2=0; MS3=0; MS4=0; R1E=0; R2E=0;
  case (state)
    S0:
    if (start)  begin
      next_state=S1; R1E=1; R2E=1; end
    else next_state=S0;
    S1: begin
         MS1=1; MS2=1; MS3=1; MS4=1; R1E=1; R2E=1;
         next_state=S2;
        end
    S2: begin
         done=1; MS3=1; MS4=1;
         next_state=S0;
        end
    default: begin
               done='bx; MS1='bx; MS2='bx; MS3='bx; MS4='bx; R1E='bx; R2E='bx;
               next_state='bx;
             end
  endcase
end

endmodule
```

To verify the correctness of the design, we add the numbers 1, 2, 3, 4, 5, and 6 and it is evident from simulation results we get the correct output in the third clock cycle at Y=21.