Part 1:

a) The shortest path is a(flip-flop)->f(comb. block)-> j(flip-flop)->m(comb. block)->n(flip-flop)

   (setup time of a) + (setup time of j) + (propagation delay for the two comb. blocks)

   + (hold time of n)

   = 2ns + 2ns + (4+3)ns + 1ns

   =12ns

   Max possible frequency : 83.33MHz

b) Since the default situation is not involved in the case statement, a latch is inferred. Adding a default case for the 4 outputs will avoid the latch and the code is then synthesizable.

```
module mod1(sel, g0, g1, g2, g3, a);
input [1:0] sel;
input a;
output logic g0, g1, g2, g3;
always_comb begin
case(sel) 2'b00: g0 = a;
          2'b01: g1 = a;
          2'b10: g2 = a;
          2'b11: g3 = a;

Default:

Begin
g0 = 0; g1 = 0; g2 = 0; g3 = 0;
end
endcase end endmodule
```

c) In this code, the always_comb blocks start to execute at the same time. The output ‘b’ is assigned a value both the times which means the value for ‘b’ is unclear.

   So ‘b’ should be assigned only in one of the always_comb blocks.

Part 2:

a) The inputs are considered valid only when the **Valid_in** signal goes high. A **Reset** signal has to be asserted initially to flush all the flip-flops and set their values to 0. Since this is a synchronous circuit, a **clock** signal has to be generated to trigger flip-flops.
   Since there are two 8-bit inputs there are $2^8 * 2^8$ possible input values. The test bench has to check the design for all the 256*256 values.
   Better design:
   The same can be done by incrementing the input values by a small constant number instead of 1 so that the output can be check with a fewer inputs, still checking for the accuracy of the output.
   Dividing the input values into multiple groups (using multiple for loops) so that Valid_in can be set/reset to check how the circuit reacts.

b) Critical path: "a_wire_reg" to "f_reg"

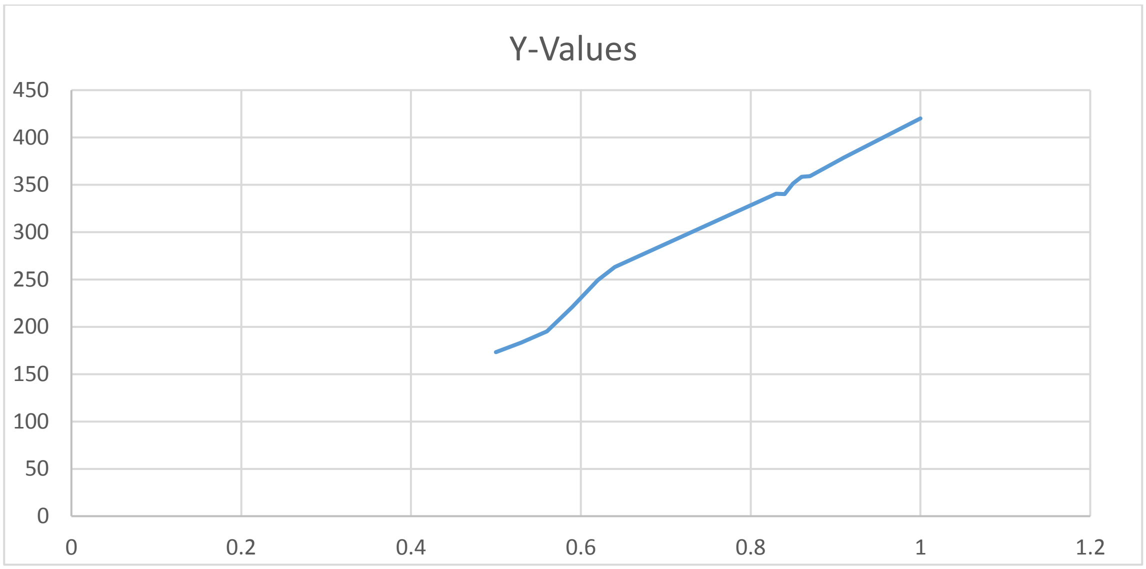
S.No.	Frequency(GHz)	Power(uW)	Area(um^2)	result
1	1	420.1442	704.633994	SLACK(violated)
2	0.91	378.9010	693.727993	SLACK(violated)
3	0.87	359.2533	688.939994	SLACK(violated)
4	0.86	358.3560	691.067994	SLACK(violated)
5	0.85	351.3051	680.427994	SLACK(Met)
6	0.84	340.3361	658.083994	SLACK(Met)
7	0.83	340.4174	660.211994	SLACK(Met)
8	0.64	263.1489	669.787993	SLACK(Met)
9	0.62	249.1668	639.729993	SLACK(Met)
10	0.59	220.9964	594.775995	SLACK(Met)
11	0.56	195.0706	551.949997	SLACK(Met)
12	0.53	183.3084	546.629998	SLACK(Met)
13	0.5	173.2555	539.713998	SLACK(Met)

Max Frequency: 850MHz

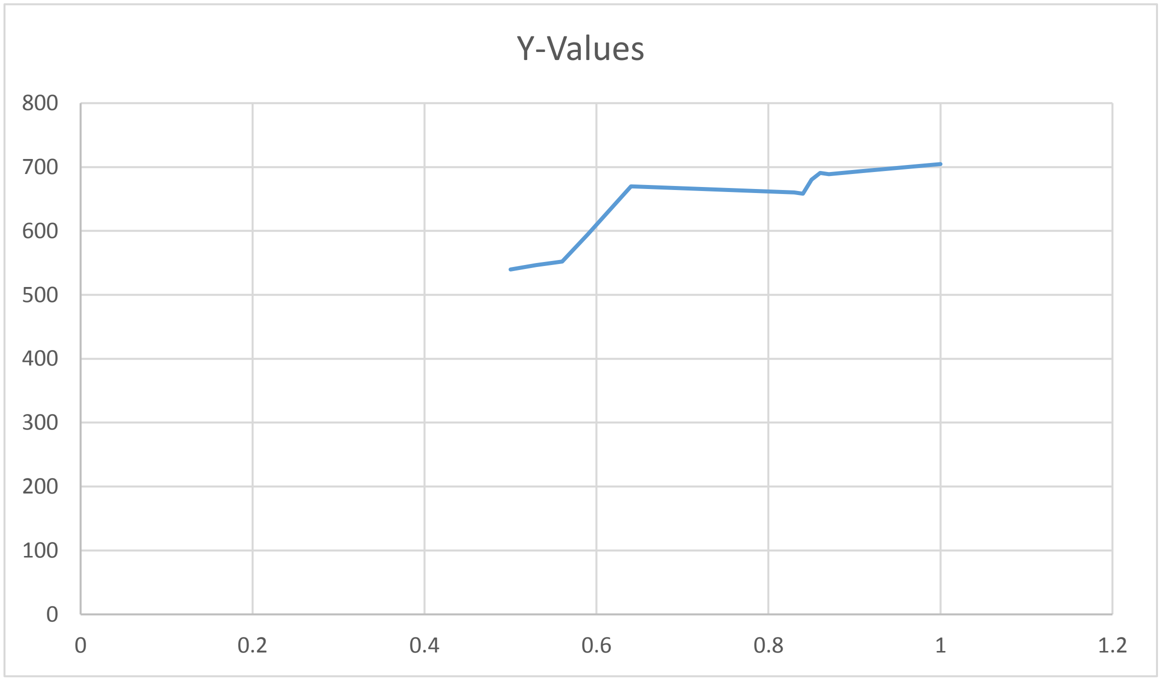
c)

Power vs Frequency graph:

Power varies linearly with frequency



Power vs Area graph:



d) From the above graph power changes linearly with frequency. Since power is directly proportional to energy, from the graph, energy changes linearly with frequency:

   E α f

   E = cf+t   // c is the slope, t is some constant; This is an approximation.

   // E is the Energy required per second; f is no. of operations per second

   In the best case, when there is data occurring continuously

   Per second – 850 million MAC operations occur.

   Per second – 350.3051 uW is the power consumed, which is the (energy consumed)/(second)

   Energy per operation will be (350.3051)/(850 x 1,000,000) = 0.4121 x 10^{-12} J/operation

e) No. Energy per operation doesn't change significantly. That is because whenever there is an increase in frequency, the energy also changes proportionally keeping the change in energy per operation insignificant.

f) Yes it is necessary to reset all the registers. Since the values on the registers are unknown when the system is turned on, any operation has to be carried out only after resetting all registers.

Part 3:

a) The extra register reduces the critical path by storing the value. This means neither the adder nor the multiplier will be empty in one clock cycle. They keep computing outputs simultaneously which increases the throughput of the MAC system.
An enable signal has to be added to the control logic so that the output of the additional register (the product) can be read. All the operations happen on a positive clock edge. A reset signal is also included so that the register remains in a known state.
The test bench need not be changed.
Critical Path: **a_wire_reg** to **prod_ff_reg**

S.No.	Frequency(GHz)	Power(uW)	Area(um^2)	Result
1	1.11	569.4183	791.881992	SLACK(violated)
2	1.08	553.9247	780.177992	SLACK(violated)
3	1.07	537.2275	776.719993	SLACK(Met)
4	1.06	551.2760	787.891992	SLACK(Met)
5	1.05	535.1020	785.763993	SLACK(violated)
6	1.01	508.7244	755.971992	SLACK(Met)

Max Frequency: 1.07GHz

b) The maximum frequency achieved is 1.78 GHz when the multiplier is pipelined in 6-stages.

c)

Design Spec.	Max. frequency(GHz)	Power (uW)	Area (um^2)	Energy per operation (10^{-12}J/operation)
MAC is pipelined using a register between multiplier and adder	1.07	537.2275	776.719993	0.50
MAC is pipelined and the multiplier is pipelined in 2 stages	1.17	740.3314	830.185989	0.63

MAC is pipelined and the multiplier is pipelined in 3 stages	1.56	1307.3	996.435983	0.83
MAC is pipelined and the multiplier is pipelined in 4 stages	1.78	1677.8	1079.693979	0.94
MAC is pipelined and the multiplier is pipelined in 5 stages	1.78	1895.6	1146.193978	1.06
MAC is pipelined and the multiplier is pipelined in 6 stages	1.78	1992.0	1197.531975	1.12

Energy per operation for non-pipelined design: 0.4121 x 10^{-12} J/operation
Energy per operation for best pipelined design: 0.94 x 10^{-12} J/operation

The energy required for 1 operation increased by a factor of around 2.1 and the frequency increased by almost the same factor. This is the tradeoff between the achieving better frequency and energy per operation.

d) The best design is the one which uses a pipelined MAC and 4-stage pipelined multiplier.
Max. Frequency: 1.78GHz
Area: 1079.6939um^2
Power: 1.6778e+03 uW

The maximum frequency is achieved when the multiplier is pipelined in 4, 5, 6 stages. The most efficient of those designs is however the one which uses 4 stage pipelined multiplier because of difference in requirements of area and power. 4 stage pipelined multiplier uses the least power and area compared to the other designs with higher levels of pipelining.

e) Yes the adder can be pipelined.
Since a 16-bit adder contains full adder modules, 15 registers can be used to pipeline the 16-bit adder. But when the delay that occurs due to the pipelined registers such as setup time or hold time is taken into account it might nullify the effect of pipelining and add extra delays to the circuit.