

**Scheme Of Evaluation**

Internal Assessment Test - I

Sub:	DSDV						Code:	15EC663	
Date:	/ 03 / 2019	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	Open Elective
Answer all Questions									

	Marks	OBE	
		CO	RBT
1 Explain the systematic process of digital system from design to Embedded system design.	[10]	CO1	L1
2 Develop a sequential circuit that has a single data input signal 'S' and produces an output 'Y'. The output is '1' whenever 'S' has the same value over three successive clock cycles, and '0' otherwise. Assume any value of 'S' for a given clock cycle.	[10]	CO1	L3
3 Develop a verilog code to find a given year is a leap year or not and generate a leap year flag.	[10]	CO1	L3
4 Develop a verification module for testing a design which down counts from '99' – '10' and repeats till input ready signal is reset.	[10]	CO1	L3
5 For complex multiplier unit having 8 pre binary and 8 post binary input's, design the data path which performs multiplication and stores the result in 64k x 8 bit memory (use 16k x 8 bit as component memory)	[10]	CO2	L3

Solution

i) Explain the systematic process of digital system from design to embedded system design. [10M]

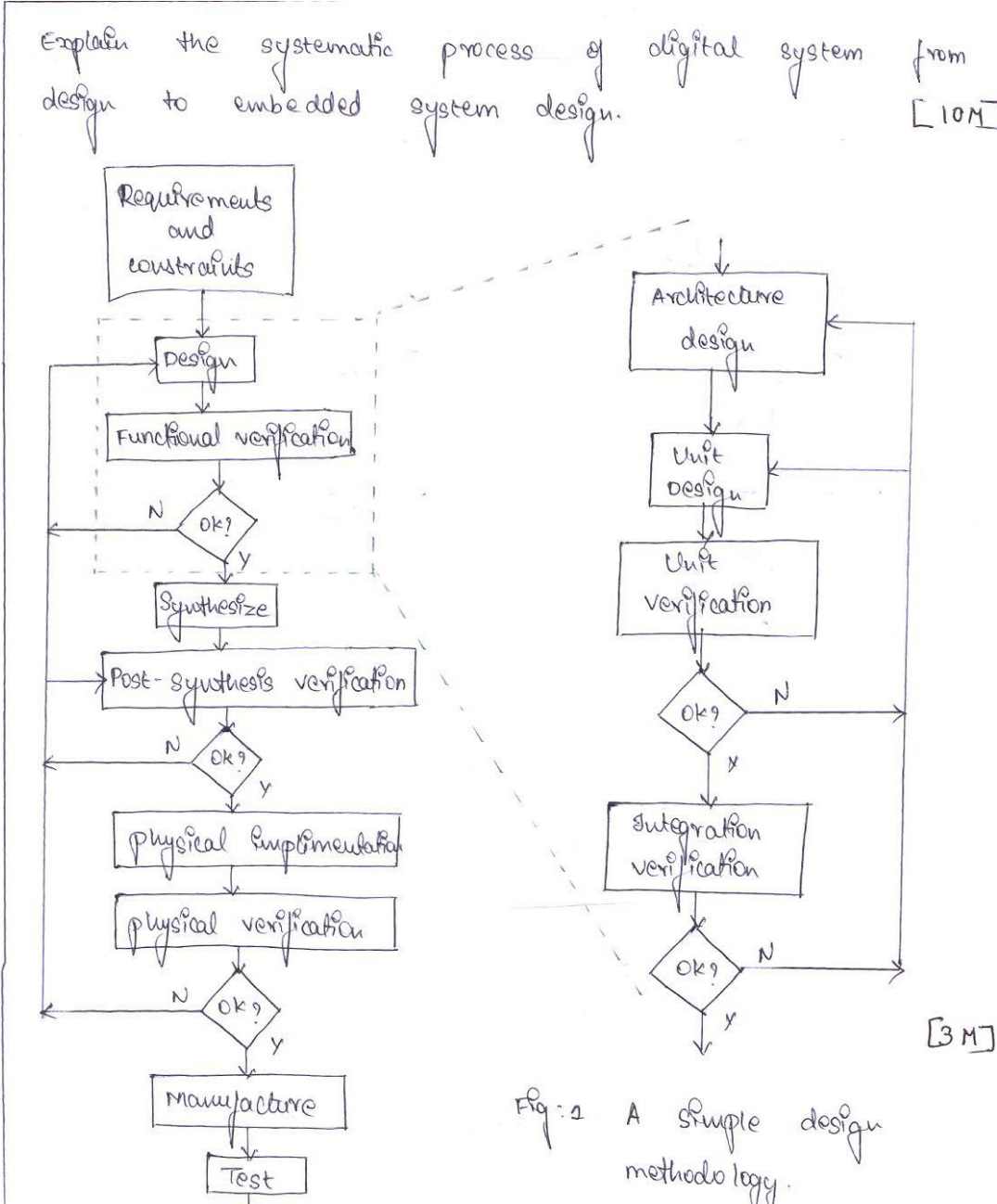
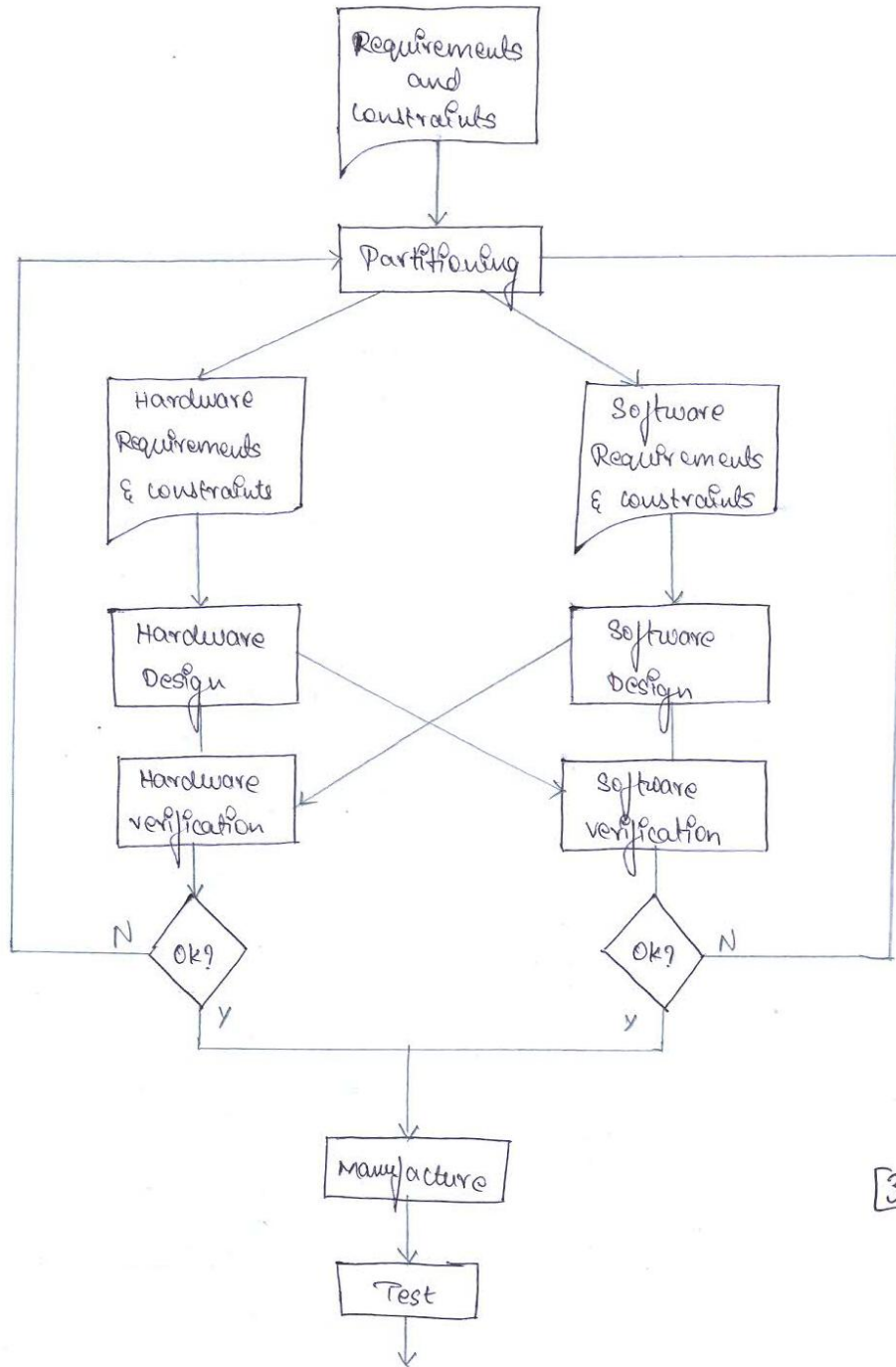


Fig: 1 A simple design methodology.

[3M]





[3M]

Fig: 2 Hardware and software Codesign

Fig 1 illustrates a simple design methodology.

Requirements and constraints are generated externally by marketing group of a company or by customer. They include functional requirements, performance requirements, constraints on power consumption, cost and packaging.

- 3 Tasks:
- 1) design
  - 2) synthesis
  - 3) physical packing

[3M]

Each of these are followed by a verification task.

If verification fails at any stage, we must revisit a previous task to correct the error.

Fig 2 illustrates hardware and software codesign.

Designing the hardware and software for a system together is called hardware/software codesign. Deciding which parts to put in hardware and which in the software is called partitioning.

Once the functionality has been partitioned b/w hardware and software, development of the two can proceed concurrently.

For those aspects of the embedded system software that depends on hardware, the abstract behavioral models from the hardware design task can be used to verify software design. A similar approach can be used to

verify parts of hardware that interface directly with a processor core.

- 2) Develop a sequential circuit that has a single data i/p signal 's' and produces an o/p 'y'. The o/p is '1' whenever i/p s has same value for three successive cycles and '0' otherwise. Assume the value of 's' for the given cycle.

To compare the value of i/p to 3 successive clock cycles, the value of i/p should be same for previous two cycles

To store previous values we use a pair of D-FF

The o/p y is 1 if and only if 3 successive values of i/p are all 1 or all 0.

$g_1, g_2$  jointly determine if 3 values are 1

$g_3, g_4, g_5$  negate 3 values.

$g_6, g_7$  determine 3 values are 0.

$g_8$  combines both to yield final o/p.

[5M]

The o/p of 2 FF's follow value of s over 2 cycles (delayed)  $\therefore$  When either  $y_1$  or  $y_0$  is 1, the o/p y changes to 1.

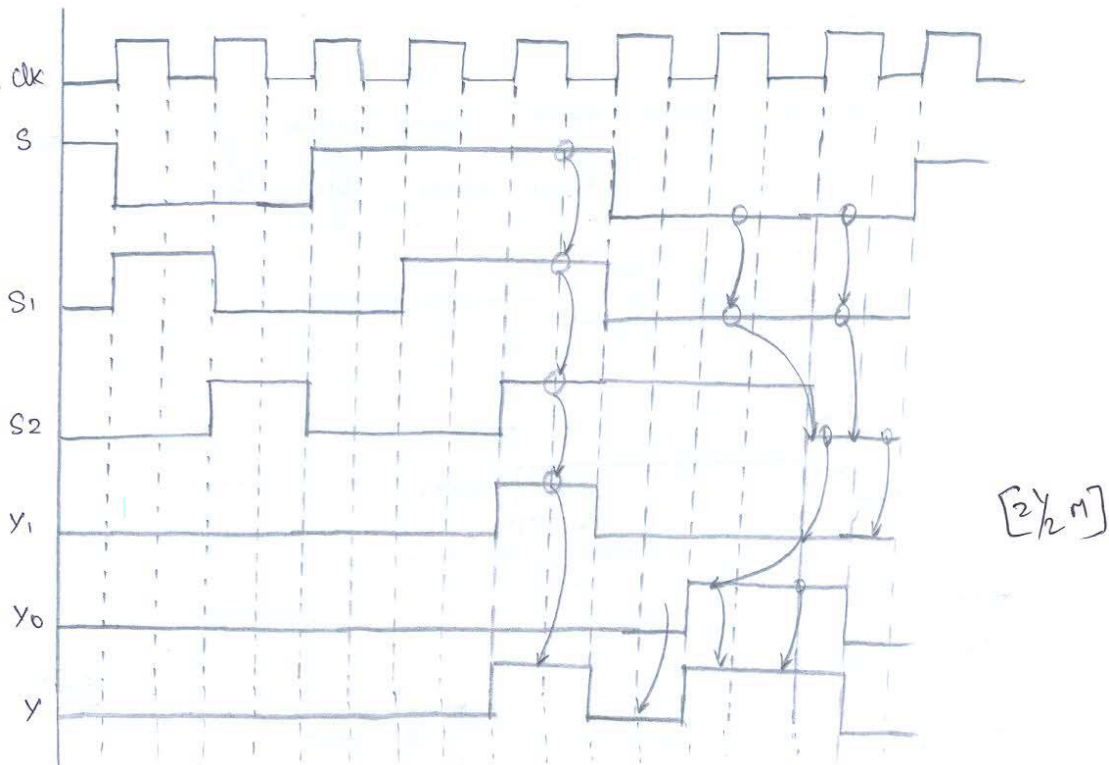
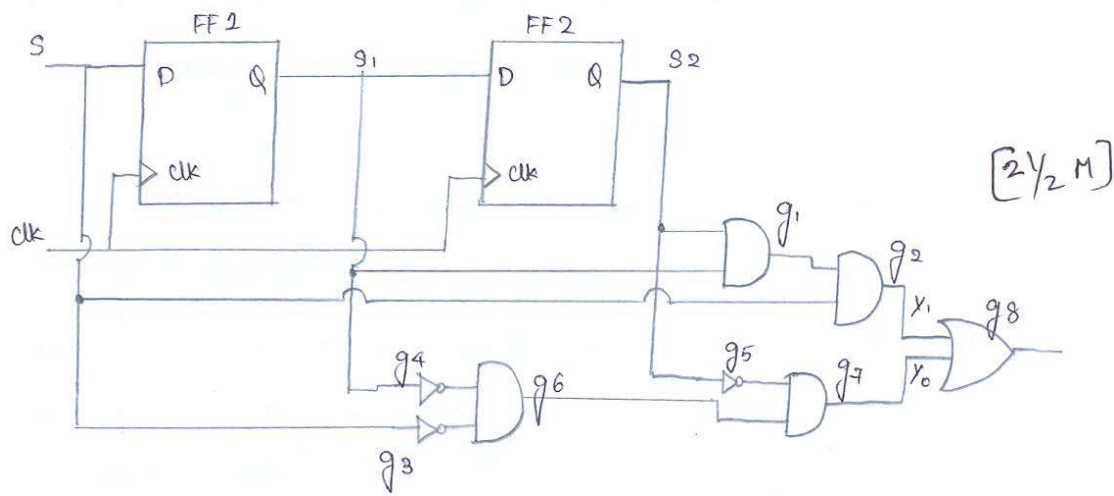


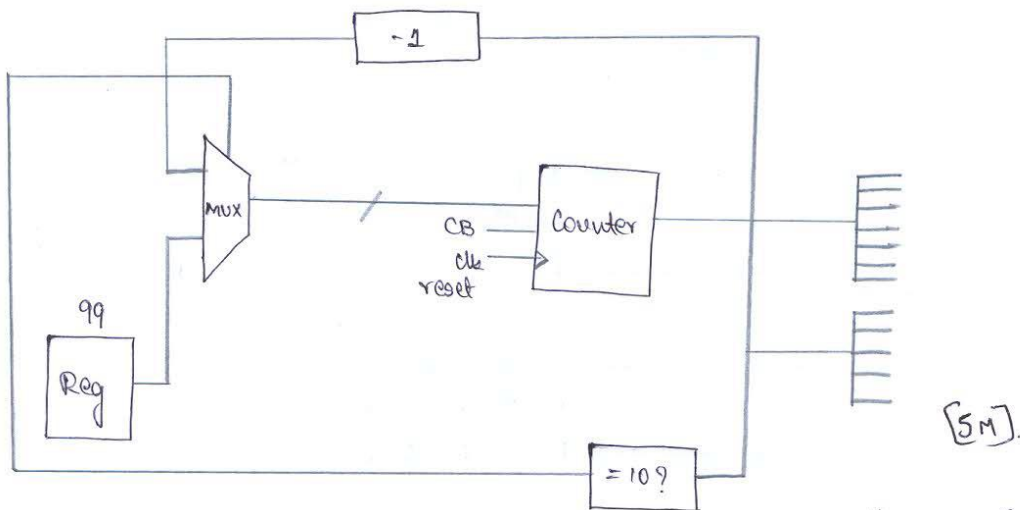
Fig: Timing diagram for the sequential comparison circuit

The circles and arrows indicate which signals are used to determine the values of other signals, leading to a 1 at the output.

When all  $s$ ,  $s_1$  and  $s_2$  are 1,  $Y_1$  changes to 1, indicating that  $s$  has been 1 for 3 successive cycles. Similarly when all  $s$ ,  $s_1$  and  $s_2$  are 0,  $Y_0$  changes to 1, indicating that  $s$  has been 0 for 3 successive cycles. When either  $Y_1$  or  $Y_0$  is 1, the output  $Y$  changes to 1.

3) Design a digital system which down counts from 99 to 00 and suspends when input ready signal is off.

Sol:-



[5M].

verilog PGM [5M].

4) Develop a verilog code to find a given year is a leap year or not and generate a leap year flag.



301-

```
module leapyear (output leap, input year);
```

```
begin
```

```
if (year % 4 == 0 && year % 100 != 0 || year % 400 == 0)
```

```
    leap <= 1'b1;
```

```
else
```

```
    leap <= 1'b0;
```

[5M]

```
endmodule.
```

5) Develop a verification for testing code:

```
module div-by-4();
```

```
    reg [3:0] ym, yu, ye, yo;
```

```
    wire out;
```

```
);
```

```
div-by-4 four (out, ym, yu, ye, yo);
```

[2M]

initial

```
begin
```

```
    # 100 ym = 4'b0000;
```

```
    # 100 yu = 4'b0000;
```

```
    # 100 ye = 4'b0000;
```

```
    # 100 yo = 4'b0000;
```

```
        .  
        .  
        .
```

```
end
```

```
endmodule
```

[4M]

```
// divisible by zero
module iszero (a, b, c, d, ym, yu, yt, yo);
```

```
input [3:0] ym, yu, yt, yo;
```

```
output A, B, C, D;
```

```
assign # 10 A = ~yo[3] & ~yo[2] & ~yo[1] & ~yo[0];
```

```
assign # 10 B = ~yt[3] & ~yt[2] & ~yt[1] & ~yt[0];
```

```
assign # 10 C = ~yu[3] & ~yu[2] & ~yu[1] & ~yu[0];
```

```
assign # 10 D = ~ym[3] & ~ym[2] & ~ym[1] & ~ym[0];
```

```
endmodule
```

- [4m]

```
module iszerotu ();
```

```
reg [3:0] ym, yu, yt, yo;
```

```
wire leave;
```

```
);
```

```
div. by zero mer (leave, ym, yu, yt, yo);
```

```
initial
```

```
begin
```

```
# 100 ym = 4'b0000;
```

```
# 1000 yt = 4'b0000;
```

```
# 1000 yu = 4'b0000;
```

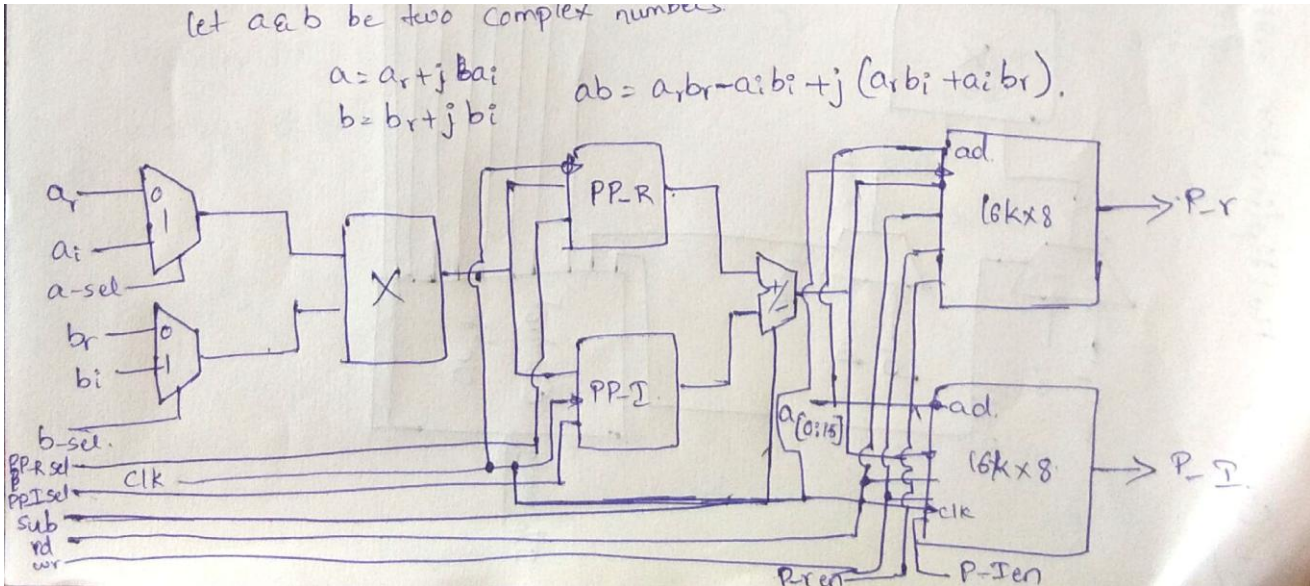
```
...
```

```
end
```

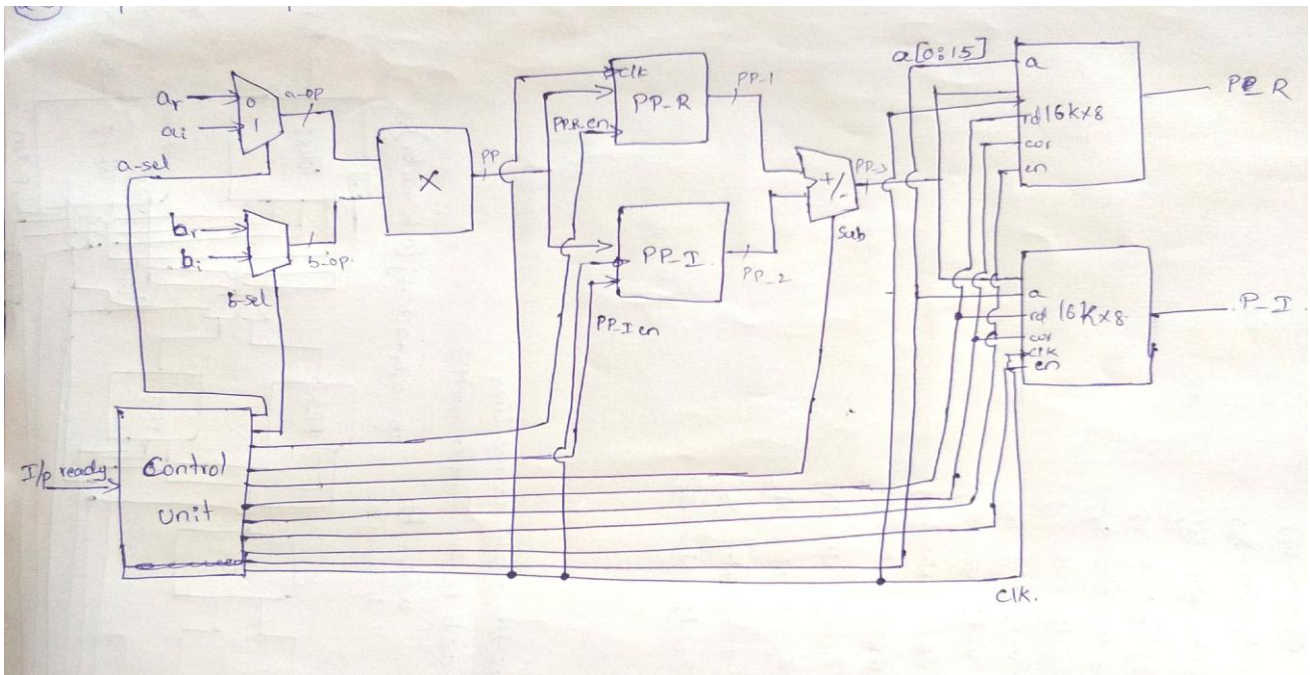
```
endmodule
```

- [6m]

5) For complex multiplier unit having 8 pre binary and 8 post binary input's, design the data path which performs multiplication and stores the result in 64k x 8 bit memory (use 16k x 8 bit as component memory)



5 Marks



5 Marks