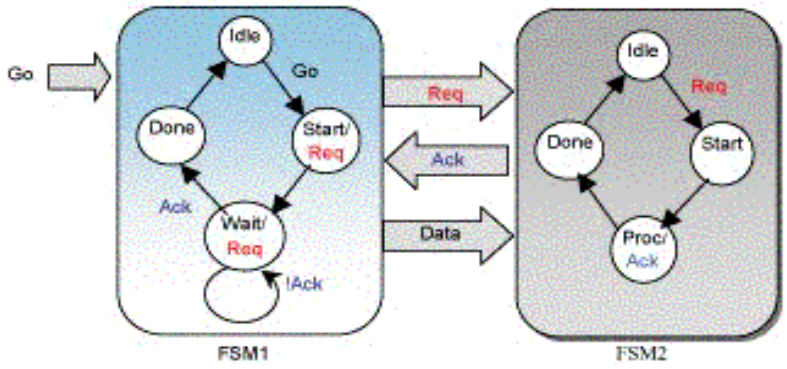
11) What are different ways to synchronize between two clock domains?

The following section explains clock domain interfacing

One of the biggest challenges of system-on-chip (SOC) designs is that different blocks operate on independent clocks. Integrating these blocks via the processor bus, memory ports, peripheral busses, and other interfaces can be troublesome because unpredictable behavior can result when the asynchronous interfaces are not properly synchronized

A very common and robust method for synchronizing multiple data signals is a handshake technique as shown in diagram below This is popular because the handshake technique can easily manage changes in clock frequencies, while minimizing latency at the crossing. However, handshake logic is significantly more complex than standard synchronization structures.



FSM1(Transmitter) asserts the req (request) signal, asking the receiver to accept the data on the data bus. FSM2(Receiver) generally a slow module asserts the ack (acknowledge) signal, signifying that it has accepted the data.

it has loop holes: when system Receiver samples the systems Transmitter req line and Transmitter samples system Receiver ack line, they have done it with respect to their internal clock, so there will be setup and hold time violation. To avoid this we go for double or triple stage synchronizers, which increase the MTBF and thus are immune to metastability to a good extent. The figure below shows how this is done.

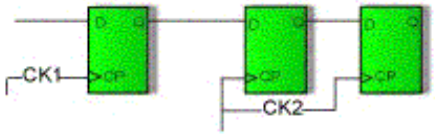


Figure 1a — Single-bit metastability sync

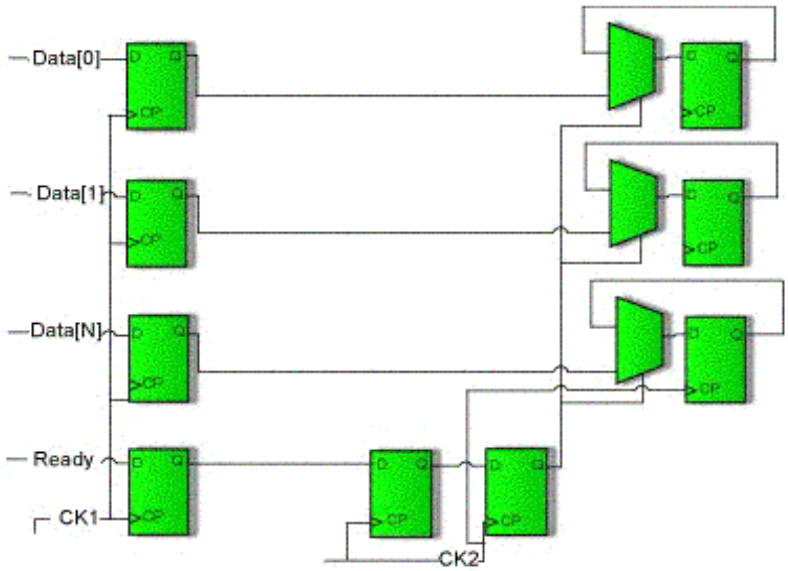


Figure 1b — Multi-bit sync

Blocking vs Non-Blocking. . .

self triggering blocks -

```
module osc2 (clk);
output clk;
reg clk;
initial #10 clk = 0;
always @(clk) #10 clk <= ~clk;
endmodule
```

After the first @(clk) trigger, the RHS expression of the nonblocking assignment is evaluated and the LHS value scheduled into the nonblocking assign updates event queue.
Before the nonblocking assign updates event queue is "activated," the @(clk) trigger statement is encountered and the always block again becomes sensitive to changes on the clk signal. When the nonblocking LHS value

is updated later in the same time step, the @(clk) is again triggered.

```
module osc1 (clk);
output clk;
reg clk;
initial #10 clk = 0;
always @(clk) #10 clk = ~clk;
endmodule
```

Blocking assignments evaluate their RHS expression and update their LHS value without interruption. The blocking assignment must complete before the @(clk) edge-trigger event can be scheduled. By the time the trigger event has been scheduled, the blocking clk assignment has completed; therefore, there is no trigger event from within the always block to trigger the @(clk) trigger.

Bad modeling: - (using blocking for seq. logic)

```
always @(posedge clk) begin
q1 = d;
q2 = q1;
q3 = q2;
end
```

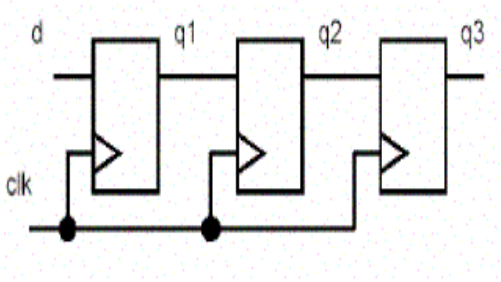
Race Condition

```
always @(posedge clk) q1=d;
always @(posedge clk) q2=q1;
always @(posedge clk) q3=q2;

always @(posedge clk) q2=q1;
always @(posedge clk) q3=q2;
always @(posedge clk) q1=d;

always @(posedge clk) begin
q3 = q2;
q2 = q1;
q1 = d;
end
```

Bad style but still works



Good modeling: -

```
always @(posedge clk) begin
q1 <= d;
q2 <= q1;
q3 <= q2;
end

always @(posedge clk) begin
q3 <= q2;
q2 <= q1;
q1 <= d;
end
```

No matter of sequence for Nonblocking

```
always @(posedge clk) q1<=d;
always @(posedge clk) q2<=q1;
always @(posedge clk) q3<=q2;

always @(posedge clk) q2<=q1;
always @(posedge clk) q3<=q2;
always @(posedge clk) q1<=d;
```

Good Combinational logic :- (Blocking)

```
always @(a or b or c or d) begin
tmp1 = a & b;
tmp2 = c & d;
y = tmp1 | tmp2;
end
```

Bad Combinational logic :- (Nonblocking)

always @(a or b or c or d) begin will simulate incorrectly?br> tmp1 <= a & b; need tmp1, tmp2 insensitivity

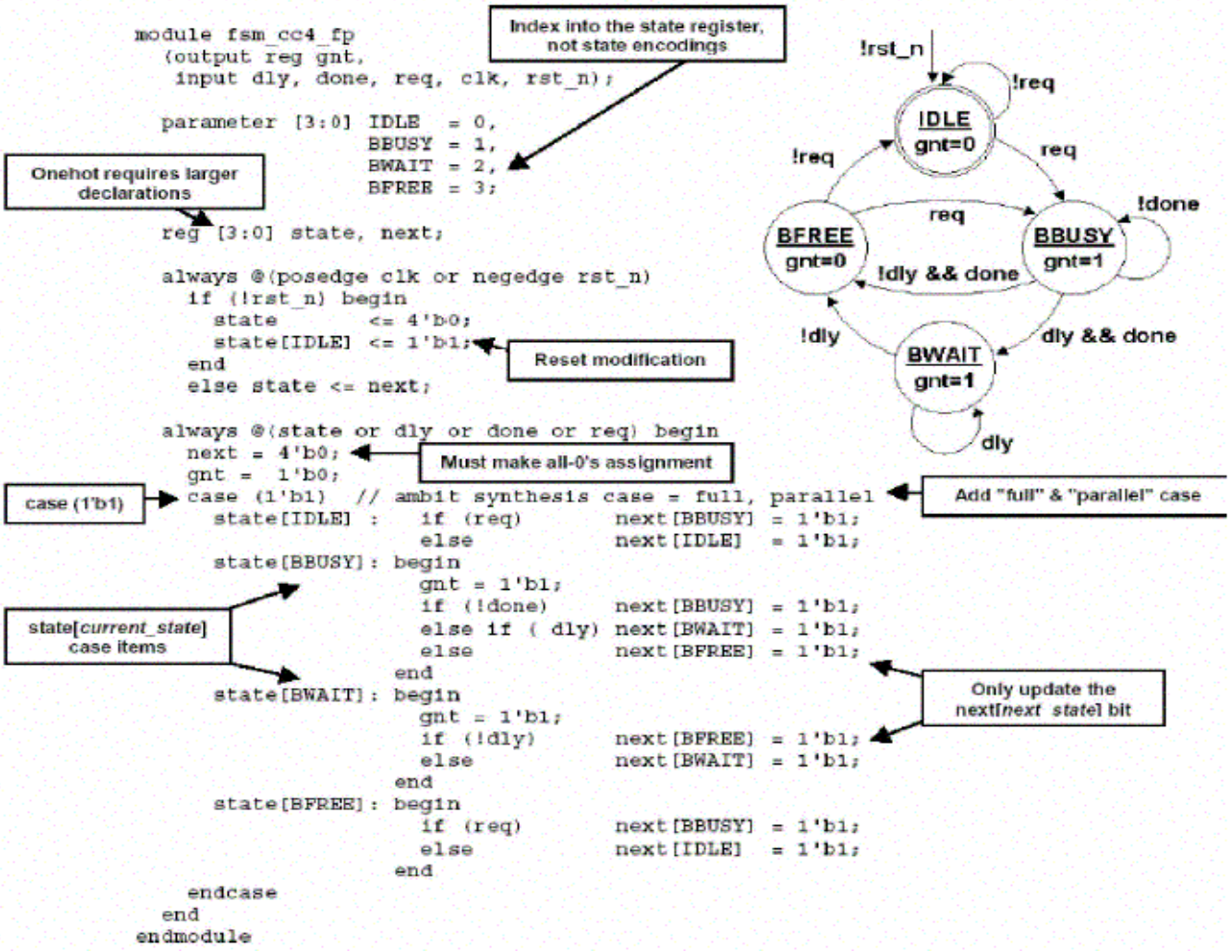
```
tmp2 <= c & d;
```

```
y <= tmp1 | tmp2;
end
```

Mixed design: -

Use Nonblocking assignment.
In case on multiple non-blocking assignments last one will win.

Verilog FSM



1) Explain about setup time and hold time, what will happen if there is setup time and hold tine violation, how to overcome this?

Set up time is the amount of time before the clock edge that the input signal needs to be stable to guarantee it is accepted properly on the clock edge.
Hold time is the amount of time after the clock edge that same input signal has to be held before changing it to make sure it is sensed properly at the clock edge.
Whenever there are setup and hold time violations in any flip-flop, it enters a state where its output is unpredictable: this state is known as metastable state (quasi stable state); at the end of metastable state, the flip-flop settles down to either '1' or '0'. This whole process is known as metastability

2) What is skew, what are problems associated with it and how to minimize it?

In circuit design, clock skew is a phenomenon in synchronous circuits in which the clock signal (sent from the clock circuit) arrives at different components at different times.
This is typically due to two causes. The first is a material flaw, which causes a signal to travel faster or slower than expected. The second is distance: if the signal has to travel the entire length of a circuit, it will likely (depending on the circuit's size) arrive at different parts of the circuit at different times. Clock skew can cause harm in two ways. Suppose that a logic path travels through combinational logic from a source flip-flop to a destination flip-flop. If the destination flip-flop receives the clock tick later than the source flip-flop, and if the logic path delay is short enough, then the data signal might arrive at the destination flip-flop before the clock tick, destroying there the previous data that should have been clocked through. This is called a hold violation because the previous data is not held long enough at the destination flip-flop to be properly clocked through. If the destination flip-flop receives the clock tick earlier than the source flip-flop, then the data signal has that much less time to reach the destination flip-flop before the next clock tick. If it fails to do so, a setup violation occurs, so-called because the new data was not set up and stable before the next clock tick arrived. A hold violation is more serious than a setup violation because it cannot be fixed by increasing the clock period.
Clock skew, if done right, can also benefit a circuit. It can be intentionally introduced to decrease the clock period at which the circuit will operate correctly, and/or to increase the setup or hold safety margins. The optimal set of clock delays is determined by a linear program,

in which a setup and a hold constraint appears for each logic path. In this linear program, zero clock skew is merely a feasible point.
Clock skew can be minimized by proper routing of clock signal (clock distribution tree) or putting variable delay buffer so that all clock inputs arrive at the same time

3) What is slack?

'Slack' is the amount of time you have that is measured from when an event 'actually happens' and when it 'must happen?. The term 'actually happens' can also be taken as being a predicted time for when the event will 'actually happen'.
When something 'must happen' can also be called a 'deadline' so another definition of slack would be the time from when something 'actually happens' (call this Tact) until the deadline (call this Tdead).
Slack = Tdead - Tact.
Negative slack implies that the 'actually happen' time is later than the 'deadline' time...in other words it's too late and a timing violation....you have a timing problem that needs some attention.

4) What is glitch? What causes it (explain with waveform)? How to overcome it?

The following figure shows a gated clock. The gated clock's corresponding timing diagram shows that this implementation can lead to clock glitches, which can cause the flip-flop to clock at the wrong time.

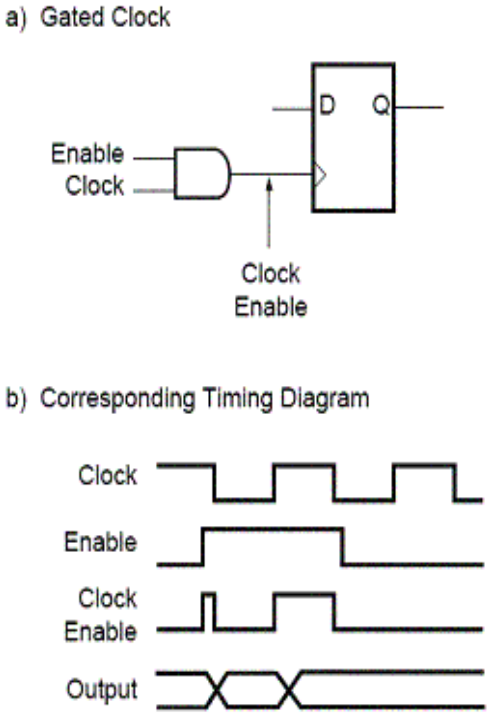
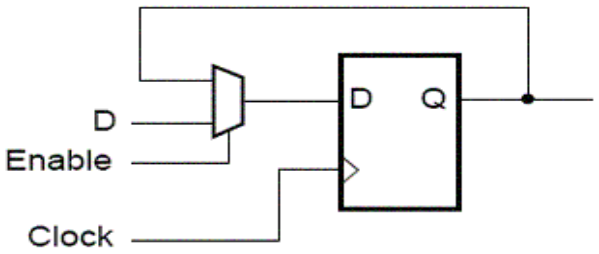


Figure 2-12: **Gated Clock**

The following figure shows a synchronous alternative to the gated clock using a data path. The flip-flop is clocked at every clock cycle and the data path is controlled by an enable. When the enable is Low, the multiplexer feeds the output of the register back on itself. When the enable is High, new data is fed to the flip-flop and the register changes its state

a) Using a Feedback Path



b) Corresponding Timing Diagram

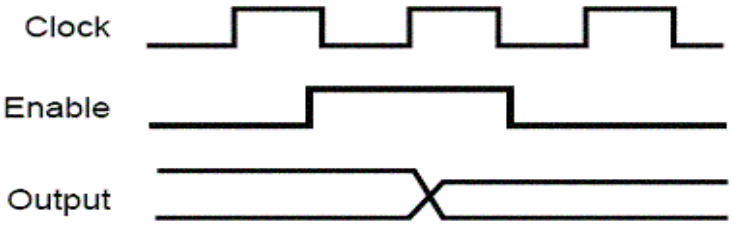


Figure 2-13: **Synchronous Design Using Data Feedback**

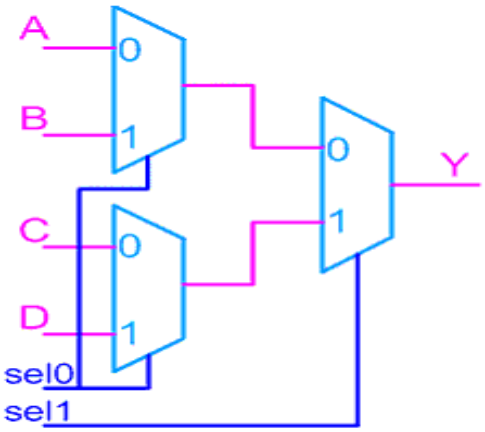
5) Given only two xor gates one must function as buffer and another as inverter?

Tie one of xor gates input to 1 it will act as inverter.
Tie one of xor gates input to 0 it will act as buffer.

6) What is difference between latch and flipflop?

The main difference between latch and FF is that latches are level sensitive while FF are edge sensitive. They both require the use of clock signal and are used in sequential logic. For a latch, the output tracks the input when the clock signal is high, so as long as the clock is logic 1, the output can change if the input also changes. FF on the other hand, will store the input only when there is a rising/falling edge of the clock.

7) Build a 4:1 mux using only 2:1 mux?



8) Difference between heap and stack?

The Stack is more or less responsible for keeping track of what's executing in our code (or what's been "called"). The Heap is more or less responsible for keeping track of our objects (our data, well... most of it - we'll get to that later.).
Think of the Stack as a series of boxes stacked one on top of the next. We keep track of what's going on in our application by stacking another box on top every time we call a method (called a Frame). We can only use what's in the top box on the stack. When we're done with the top box (the method is done executing) we throw it away and proceed to use the stuff in the previous box on the top of the stack. The Heap is similar except that its purpose is to hold information (not keep track of execution most of the time) so anything in our Heap can be accessed at any time. With the Heap, there are no constraints as to what can be accessed like in the stack. The Heap is like the heap of clean laundry on our bed that we have not taken the time to put away yet - we can grab what we need quickly. The Stack is like the stack of shoe boxes in the closet where we have to take off the top one to get to the one underneath it.

9) Difference between mealy and moore state machine?

A) Mealy and Moore models are the basic models of state machines. A state machine which uses only Entry Actions, so that its output depends on the state, is called a Moore model. A state machine which uses only Input Actions, so that the output depends on the state and also on inputs, is called a Mealy model. The models selected will influence a design but there

are no general indications as to which model is better. Choice of a model depends on the application, execution means (for instance, hardware systems are usually best realized as Moore models) and personal preferences of a designer or programmer

B) Mealy machine has outputs that depend on the state and input (thus, the FSM has the output written on edges)
Moore machine has outputs that depend on state only (thus, the FSM has the output written in the state itself.

Adv and Disadv
In Mealy as the output variable is a function both input and state, changes of state of the state variables will be delayed with respect to changes of signal level in the input variables, there are possibilities of glitches appearing in the output variables. Moore overcomes glitches as output dependent on only states and not the input signal level.
All of the concepts can be applied to Moore-model state machines because any Moore state machine can be implemented as a Mealy state machine, although the converse is not true.
Moore machine: the outputs are properties of states themselves... which means that you get the output after the machine reaches a particular state, or to get some output your machine has to be taken to a state which provides you the output. The outputs are held until you go to some other state Mealy machine:
Mealy machines give you outputs instantly, that is immediately upon receiving input, but the output is not held after that clock cycle.

10) Difference between onehot and binary encoding?

Common classifications used to describe the state encoding of an FSM are Binary (or highly encoded) and One hot.
A binary-encoded FSM design only requires as many flip-flops as are needed to uniquely encode the number of states in the state machine. The actual number of flip-flops required is equal to the ceiling of the log-base-2 of the number of states in the FSM.
A onehot FSM design requires a flip-flop for each state in the design and only one flip-flop (the flip-flop representing the current or "hot" state) is set at a time in a one hot FSM design. For a state machine with 9- 16 states, a binary FSM only requires 4 flip-flops while a onehot FSM requires a flip-flop for each state in the design
FPGA vendors frequently recommend using a onehot state encoding style because flip-flops are plentiful in an FPGA and the combinational logic required to implement a onehot FSM design is typically smaller than most binary encoding styles. Since FPGA performance is typically related to

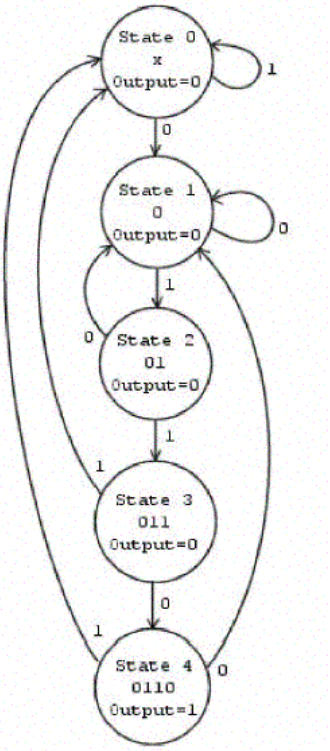
the combinational logic size of the FPGA design, onehot FSMs typically run faster than a binary encoded FSM with larger combinational logic blocks

12) How to calculate maximum operating frequency?

13) How to find out longest path?

You can find answer to this in timing.ppt of presentations section on this site

14) Draw the state diagram to output a "1" for one cycle if the sequence "0110" shows up (the leading 0s cannot be used in more than one sequence)?



15) How to achieve 180 degree exact phase shift?

Never tell using inverter
a) dcm an inbuilt resource in most of fpga can be configured to get 180 degree phase shift.
b) Bufgds that is differential signaling buffers which are also inbuilt resource of most of FPGA can be used.

16) What is significance of ras and cas in SDRAM?

SDRAM receives its address command in two address words.
It uses a multiplex scheme to save input pins. The first address word is latched into the DRAM chip with the row address strobe (RAS).
Following the RAS command is the column address strobe (CAS) for latching the second address word.
Shortly after the RAS and CAS strobes, the stored data is valid for reading.

17) Tell some of applications of buffer?

a)They are used to introduce small delays
b)They are used to eliminate cross talk caused due to inter electrode capacitance due to close routing.
c)They are used to support high fanout,eg:bufg

18) Implement an AND gate using mux?

This is the basic question that many interviewers ask. for and gate, give one input as select line,incase if u r giving b as select line, connect one input to logic '0' and other input to a.

19) What will happen if contents of register are shifter left, right?

It is well known that in left shift all bits will be shifted left and LSB will be appended with 0 and in right shift all bits will be shifted right and MSB will be appended with 0 this is a straightforward answer

What is expected is in a left shift value gets Multiplied by 2 eg:consider 0000_1110=14 a left shift will make it 0001_110=28, it the same fashion right shift will Divide the value by 2.

20)Given the following FIFO and rules, how deep does the FIFO need to be to prevent underflow or overflow?

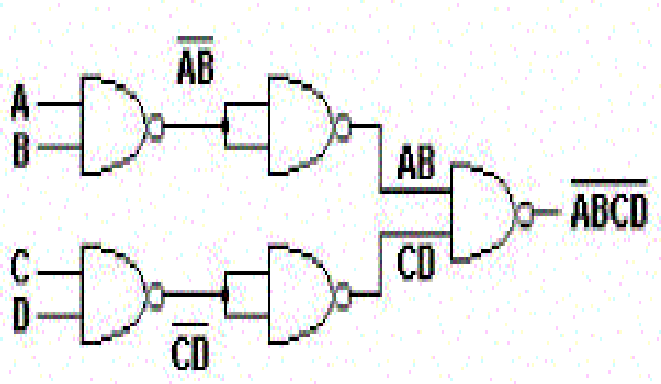
RULES:
1) frequency(clk_A) = frequency(clk_B) / 4
2) period(en_B) = period(clk_A) * 100
3) duty_cycle(en_B) = 25%

Assume clk_B = 100MHz (10ns)
From (1), clk_A = 25MHz (40ns)
From (2), period(en_B) = 40ns * 400 = 4000ns, but we only output for 1000ns,due to (3), so 3000ns of the enable we are doing no output work.

Therefore, FIFO size = 3000ns/40ns = 75 entries.

21) Design a four-input NAND gate using only two-input NAND gates ?

A:Basically, you can tie the inputs of a NAND gate together to get an inverter, so...



22)Difference between Synchronous and Asynchronous reset.?

1）Synchronous reset logic will synthesize to smaller flip-flops, particularly if the reset is gated with the logic generating the d-input. But in such a case, the combinational logic gate count grows, so the overall gate count savings may not be that significant.
2）The clock works as a filter for small reset glitches; however, if these glitches occur near the active clock edge, the Flip-flop could go metastable. In some designs, the reset must be generated by a set of internal conditions. A synchronous reset is recommended for these types of designs because it will filter the logic equation glitches between clock.

Disadvantages of synchronous reset:
1）Problem with synchronous resets is that the synthesis tool cannot easily distinguish the reset signal from any other data signal.
2）Synchronous resets may need a pulse stretcher to guarantee a reset pulse width wide enough to ensure reset is present during an active edge of the clock[if you have a gated clock to save power, the clock may be disabled coincident with the assertion of reset. Only an asynchronous reset will work in this situation, as the reset might be removed prior to the resumption of the clock.
3）Designs that are pushing the limit for data path timing, can not afford to have added gates and additional net delays in the data path due to logic inserted to handle synchronous resets.
Asynchronous reset :
The biggest problem with asynchronous resets is the reset release, also called reset removal. Using an asynchronous reset, the designer is guaranteed not to have the reset added to the data path. Another advantage favoring asynchronous resets is that the circuit can be reset with or

without a clock present.
Disadvantages of asynchronous reset: ensure that the release of the reset can occur within one clock period. if the release of the reset occurred on or near a clock edge such that the flip-flops went metastable.

23) Why are most interrupts active low?

This answers why most signals are active low
If you consider the transistor level of a module, active low means the capacitor in the output terminal gets charged or discharged based on low to high and high to low transition respectively. when it goes from high to low it depends on the pull down resistor that pulls it down and it is relatively easy for the output capacitance to discharge rather than charging. hence people prefer using active low signals.

24)Give two ways of converting a two input NAND gate to an inverter?

(a) short the 2 inputs of the nand gate and apply the single input to it.
(b) Connect the output to one of the input and the other to the input signal.

25) What are setup time & hold time constraints? What do they signify? Which one is critical for estimating maximum clock frequency of a circuit?

set up time: - the amount of time the data should be stable before the application of the clock signal, where as the hold time is the amount of time the data should be stable after the application of the clock. Setup time signifies maximum delay constraints; hold time is for minimum delay constraints. Setup time is critical for establishing the maximum clock frequency.

26) Differences between D-Latch and D flip-flop?

D-latch is level sensitive where as flip-flop is edge sensitive. Flip-flops are made up of latches.

27) What is a multiplexer?

Is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. (2n =>n).

28)How can you convert an SR Flip-flop to a JK Flip-flop?

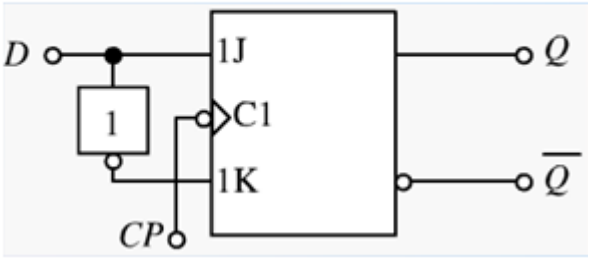
By giving the feed back we can convert, i.e !Q=>S and Q=>R.Hence the S and R inputs will act as J and K respectively.

29)How can you convert the JK Flip-flop to a D Flip-flop?

By connecting the J input to the K through the inverter.

$$Q^{n+1} = D = D(\overline{Q}^n + Q^n) = D\overline{Q}^n + DQ^n$$

$$\begin{cases} J = D \\ K = \overline{D} \end{cases}$$



30)What is Race-around problem?How can you rectify it?

The clock pulse that remains in the 1 state while both J and K are equal to 1 will cause the output to complement again and repeat complementing until the pulse goes back to 0, this is called the race around problem.To avoid this undesirable operation, the clock pulse must have a time duration that is shorter than the propagation delay time of the F-F, this is restrictive so the alternative is master-slave or edge-triggered construction.

31)How do you detect if two 8-bit signals are same?

XOR each bits of A with B (for e.g. A[0] xor B[0]) and so on.the o/p of 8 xor gates are then given as i/p to an 8-i/p nor gate. if o/p is 1 then A=B.

32)7 bit ring counter's initial state is 0100010. After how many clock cycles will it return to the initial state?

6 cycles

33) Convert D-FF into divide by 2. (not latch) What is the max clock frequency the circuit can handle, given the following information?

T_setup= 6nS T_hold = 2nS T_propagation = 10nS

Circuit: Connect Qbar to D and apply the clk at clk of DFF and take the O/P at Q. It gives freq/2.
Max. Freq of operation: 1/ (propagation delay+setup time) = 1/16ns = 62.5 MHz

34)Guys this is the basic question asked most frequently. Design all the basic gates(NOT,AND,OR,NAND,NOR,XOR,XNOR) using 2:1 Multiplexer?

Using 2:1 Mux, (2 inputs, 1 output and a select line)
(a) NOT
Give the input at the select line and connect I0 to 1 & I1 to 0. So if A is 1, we will get I1 that is 0 at the O/P.
(b) AND

Give input A at the select line and 0 to I0 and B to I1. O/p is A & B
(c) OR
Give input A at the select line and 1 to I1 and B to I0. O/p will be A | B
(d) NAND
AND + NOT implementations together
(e) NOR
OR + NOT implementations together
(f) XOR
A at the select line B at I0 and ~B at I1. ~B can be obtained from (a) (g) XNOR
A at the select line B at I1 and ~B at I0

35)N number of XNOR gates are connected in series such that the N inputs (A0,A1,A2.....) are given in the following way: A0 & A1 to first XNOR gate and A2 & O/P of First XNOR to second XNOR gate and so on..... Nth XNOR gates output is final output. How does this circuit work? Explain in detail?

If N=Odd, the circuit acts as even parity detector, ie the output will 1 if there are even number of 1's in the N input...This could also be called as odd parity generator since with this additional 1 as output the total number of 1's will be ODD.
If N=Even, just the opposite, it will be Odd parity detector or Even Parity Generator.

36)An assembly line has 3 fail safe sensors and one emergency shutdown switch.The line should keep moving unless any of the following conditions arise:
(i) If the emergency switch is pressed
(ii) If the senor1 and sensor2 are activated at the same time.
(iii) If sensor 2 and sensor3 are activated at the same time.
(iv) If all the sensors are activated at the same time
Suppose a combinational circuit for above case is to be implemented only with NAND Gates. How many minimum number of 2 input NAND gates are required?

No of 2-input NAND Gates required = 6 You can try the whole implementation.

37)Design a circuit that calculates the square of a number? It should not use any multiplier circuits. It should use Multiplexers and other logic?

This is interesting....
1^2=0+1=1
2^2=1+3=4
3^2=4+5=9
4^2=9+7=16
5^2=16+9=25
and so on
See a pattern yet?To get the next square, all you have to do is add the next odd number to the previous square that you found.See how 1,3,5,7 and finally 9 are added.Wouldn't this be a

possible solution to your question since it only will use a counter,multiplexer and a couple of adders?It seems it would take n clock cycles to calculate square of n.

38) How will you implement a Full subtractor from a Full adder?

all the bits of subtrahend should be connected to the xor gate. Other input to the xor being one.The input carry bit to the full adder should be made 1. Then the full adder works like a full subtractor

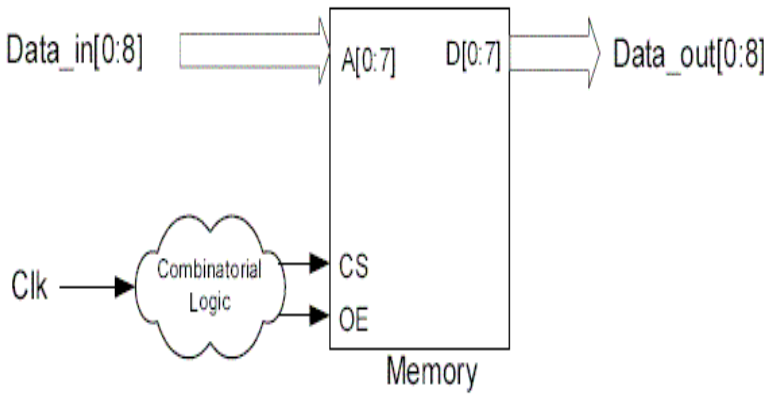
39)A very good interview question... What is difference between setup and hold time. The interviewer was looking for one specific reason , and its really a good answer too..The hint is hold time doesn't depend on clock, why is it so...?

Setup violations are related to two edges of clock, i mean you can vary the clock frequency to correct setup violation. But for hold time, you are only concerned with one edge and does not basically depend on clock frequency.

40)In a 3-bit Johnson's counter what are the unused states?

2(power n)-2n is the one used to find the unused states in johnson counter.
So for a 3-bit counter it is 8-6=2.Unused states=2. the two unused states are 010 and 101

41)The question is to design minimal hardware system, which encrypts 8-bit parallel data. A synchronized clock is provided to this system as well. The output encrypted data should be at the same rate as the input data but no necessarily with the same phase?



The encryption system is centered around a memory device that perform a LUT (Look-Up Table) conversion. This memory functionality can be achieved by using a PROM, EPROM, FLASH and etc. The device contains an encryption code, which may be burned into the device with an external programmer.

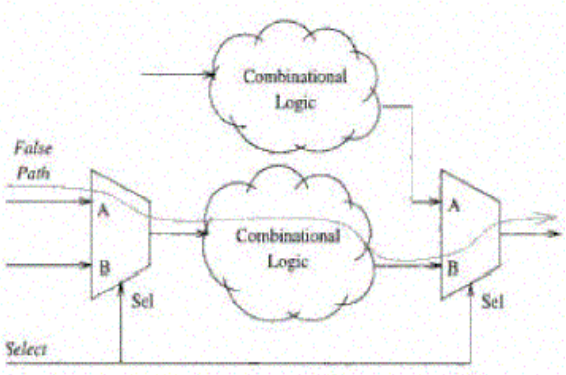
In encryption operation, the data_in is an address pointer into a memory cell and the combinatorial logic generates the control signals. This creates a read access from the memory. Then the memory device goes to the appropriate address and outputs the associate data. This data represent the data_in after encryption.

41) What is an LFSR .List a few of its industry applications.?

LFSR is a linear feedback shift register where the input bit is driven by a linear function of the overall shift register value. coming to industrial applications, as far as I know, it is used for encryption and decryption and in BIST(built-in-self-test) based applications..

42)what is false path?how it determine in ckt? what the effect of false path in ckt?

By timing all the paths in the circuit the timing analyzer can determine all the critical paths in the circuit. However, the circuit may have false paths, which are the paths in the circuit which are never exercised during normal circuit operation for any set of inputs.
An example of a false path is shown in figure below. The path going from the input A of the first MUX through the combinational logic out through the B input of the second MUS is a false path. This path can never be activated since if the A input of the first MUX is activated, then Sel line will also select the A input of the second MUX.
STA (Static Timing Analysis) tools are able to identify simple false paths; however they are not able to identify all the false paths and sometimes report false paths as critical paths. Removal of false paths makes circuit testable and its timing performance predictable (sometimes faster)

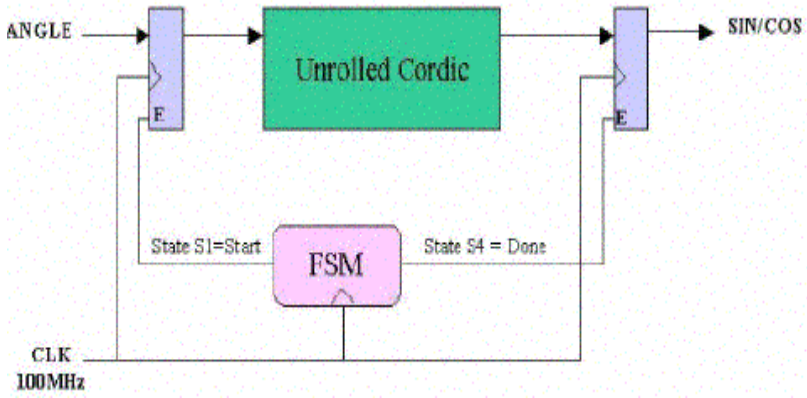


43)Consider two similar processors, one with a clock skew of 100ps and other with a clock skew of 50ps. Which one is likely to have more power? Why?

Clock skew of 50ps is more likely to have clock power. This is because it is likely that low-skew processor has better designed clock tree with more powerful and number of buffers and overheads to make skew better.

44)What are multi-cycle paths?

Multi-cycle paths are paths between registers that take more than one clock cycle to become stable.
For ex. Analyzing the design shown in fig below shows that the output SIN/COS requires 4 clock-cycles after the input ANGLE is latched in. This means that the combinatorial block (the Unrolled Cordic) can take up to 4 clock periods (25MHz) to propagate its result. Place and Route tools are capable of fixing multi-cycle paths problem.



45)You have two counters counting upto 16, built from negedge DFF , First circuit is synchronous and second is "ripple" (cascading), Which circuit has a less propagation delay? Why?

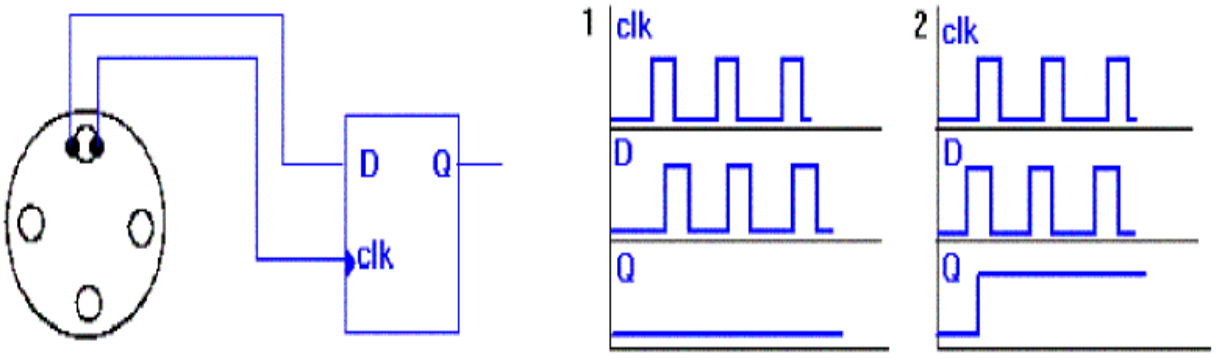
The synchronous counter will have lesser delay as the input to each flop is readily available before the clock edge. Whereas the cascade counter will take long time as the output of one flop is used as clock to the other. So the delay will be propagating. For Eg: 16 state counter = 4 bit counter = 4 Flip flops Let 10ns be the delay of each flop The worst case delay of ripple counter = 10 * 4 = 40ns The delay of synchronous counter = 10ns only.(Delay of 1 flop)

46) what is difference between RAM and FIFO?

FIFO does not have address lines
Ram is used for storage purpose where as fifo is used for synchronization purpose i.e. when two peripherals are working in different clock domains then we will go for fifo.

47)The circle can rotate clockwise and back. Use minimum hardware to build a circuit to indicate the direction of rotating.?

2 sensors are required to find out the direction of rotating. They are placed like at the drawing. One of them is connected to the data input of D flip-flop, and a second one - to the clock input. If the circle rotates the way clock sensor sees the light first while D input (second sensor) is zero - the output of the flip-flop equals zero, and if D input sensor "fires" first - the output of the flip-flop becomes high.



49)Implement the following circuits:
(a) 3 input NAND gate using min no of 2 input NAND Gates
(b) 3 input NOR gate using min no of 2 inpur NOR Gates
(c) 3 input XNOR gate using min no of 2 inpur XNOR Gates
Assuming 3 inputs A,B,C?

3 input NAND:
Connect :
a) A and B to the first NAND gate
b) Output of first Nand gate is given to the two inputs of the second NAND gate (this basically realizes the inverter functionality)
c) Output of second NAND gate is given to the input of the third NAND gate, whose other input is C
((A NAND B) NAND (A NAND B)) NAND C Thus, can be implemented using '3' 2-input NAND gates. I guess this is the minimum number of gates that need to be used.
3 input NOR:
Same as above just interchange NAND with NOR ((A NOR B) NOR (A NOR B)) NOR C
3 input XNOR:
Same as above except the inputs for the second XNOR gate, Output of the

first XNOR gate is one of the inputs and connect the second input to ground or logical '0'
((A XNOR B) XNOR 0)) XNOR C

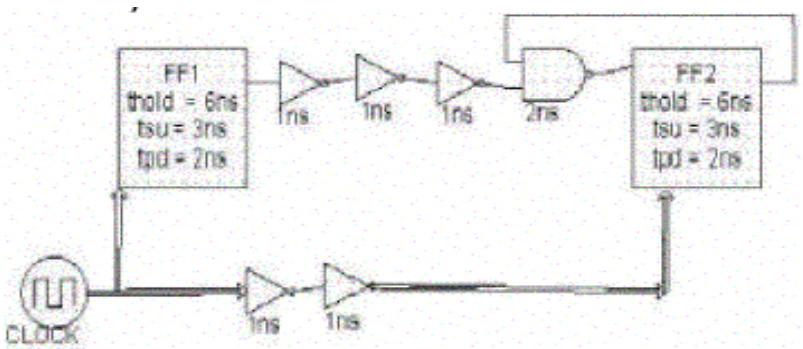
50) Is it possible to reduce clock skew to zero? Explain your answer ?

Even though there are clock layout strategies (H-tree) that can in theory reduce clock skew to zero by having the same path length from each flip-flop from the pll, process variations in R and C across the chip will cause clock skew as well as a pure H-Tree scheme is not practical (consumes too much area).

52)Convert D-FF into divide by 2. (not latch)? What is the max clock frequency of the circuit , given the following information?
T_setup= 6nS
T_hold = 2nS
T_propagation = 10nS

Circuit:
Connect Qbar to D and apply the clk at clk of DFF and take the O/P at Q. It gives freq/2.
Max. Freq of operation:
1/ (propagation delay+setup time) = 1/16ns = 62.5 MHz

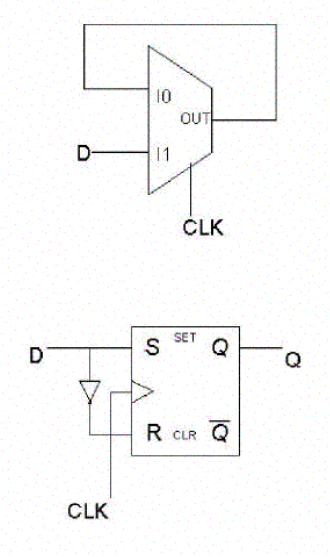
54) For the Circuit Shown below, What is the Maximum Frequency of Operation?Are there any hold time violations for FF2? If yes, how do you modify the circuit to avoid them?



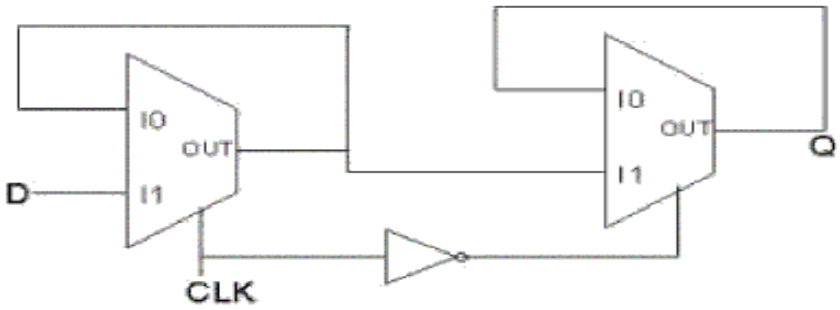
The minumum time period = 3+2+(1+1+1) = 8ns Maximum Frequency = 1/8n= 125MHz.
And there is a hold time violation in the circuit,because of feedback, if you observe, tcq2+AND gate delay is less than thold2,To avoid this we

need to use even number of inverters(buffers). Here we need to use 2 inverters each with a delay of 1ns. then the hold time value exactly meets.

55)Design a D-latch using (a) using 2:1 Mux (b) from S-R Latch ?



56)How to implement a Master Slave flip flop using a 2 to 1 mux?

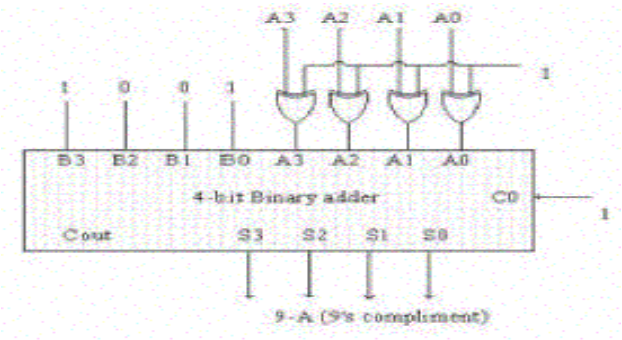


57)how many 2 input xor's are needed to inplement 16 input parity generator ?

It is always n-1 Where n is number of inputs.So 16 input parity generator will require 15 two input xor's .

58)Design a circuit for finding the 9's compliment of a BCD number using 4-bit binary adder and some external logic gates?

9's compliment is nothing but subracting the given no from 9.So using a 4 bit binary adder we can just subract the given binary no from 1001(i.e. 9).Here we can use the 2's compliment method addition.



59) what is Difference between writeback and write through cache?

A caching method in which modifications to data in the cache aren't copied to the cache source until absolutely necessary. Write-back caching is available on many microprocessors , including all Intel processors since the 80486. With these microprocessors, data modifications to data stored in the L1 cache aren't copied to main memory until absolutely necessary. In contrast, a write-through cache performs all write operations in parallel -- data is written to main memory and the L1 cache simultaneously. Write-back caching yields somewhat better performance than write-through caching because it reduces the number of write operations to main memory. With this performance improvement comes a slight risk that data may be lost if the system crashes.
A write-back cache is also called a copy-back cache.

60)Difference between Synchronous,Asynchronous & Isynchronous communication?

Sending data encoded into your signal requires that the sender and receiver are both using the same enconding/decoding method, and know where to look in the signal to find data. Asynchronous systems do not send separate information to indicate the encoding or clocking information. The receiver must decide the clocking of the signal on it's own. This means that the receiver must decide where to look in the signal stream to find ones and zeroes, and decide for itself where each individual bit stops and starts. This information is not in the data in the signal sent from transmitting unit.

Synchronous systems negotiate the connection at the data-link level before communication begins. Basic synchronous systems will synchronize two clocks before transmission, and reset their numeric counters for errors etc. More advanced systems may negotiate things like error correction and compression.

Time-dependent. it refers to processes where data must be delivered within certain time constraints. For example, Multimedia stream require an isochronous transport mechanism to ensure that data is delivered as fast as it is displayed and to ensure that the audio is synchronized with the video.

61) What are different ways Multiply & Divide?