**What is the basic difference between Analog and Digital Design?**  
In analog circuits we deal with physical signals which are continuous in amplitude and time. Ex: biological data, seismic signals, sensor output, audio, video etc. Digital design on the other hand is easier to process and has great immunity to noise. No room for automation in analog design as every application requires a different design. Whereas digital design can be automated. Analog circuits generally deal with instantaneous value of voltage and current (real time). Can take any value within the domain of specifications for the device consists of passive elements which contribute to the noise (thermal) of the circuit. They are usually more sensitive to external noise more so because for a particular function an analog design uses lot less transistors providing design challenges over process corners and temperature ranges. deals with a lot of device level physics and the state of the transistor plays a very important role Digital Circuits on the other hand deal with only two logic levels 0 and 1(Is it true that according to quantum mechanics there is a third logic level?) deal with lot more transistors for a particular logic, easier to design complex designs, flexible logic synthesis and greater speed although at the cost of greater power. Less sensitive to noise. Design and analysis of such circuits is dependent on the clock. Challenge lies in negating the timing and load delays and ensuring there is no set up or hold violation.

**What is signal integrity? How it affects Timing?**

Signal integrity or SI is a measure of the quality of an electrical signal. IR drop, Electro Migration (EM), Crosstalk, Ground bounce are signal integrity issues. If IR drop is more ==> delay increases. Crosstalk ==> there can be setup as well as hold violation.

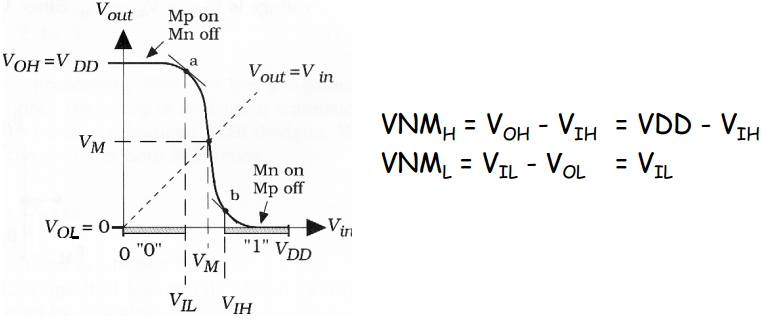
**What is Different Logic family?**   
Listed here in rough chronological order of introduction along with their usual abbreviations of Logic family  
\* Diode logic (DL)  
\* Direct-coupled transistor logic (DCTL)  
\* Complementary transistor logic (CTL)  
\* Resistor-transistor logic (RTL)  
\* Resistor-capacitor transistor logic (RCTL)  
\* Diode-transistor logic (DTL)  
\* Emitter coupled logic (ECL) also known as Current-mode logic (CML)  
\* Transistor-transistor logic (TTL) and variants  
\* P-type Metal Oxide Semiconductor logic (PMOS)  
\* N-type Metal Oxide Semiconductor logic (NMOS)  
\* Complementary Metal-Oxide Semiconductor logic (CMOS)  
\* Bipolar Complementary Metal-Oxide Semiconductor logic (BiCMOS)  
\* Integrated Injection Logic (I2L)

**What is Stuck-at fault?**   
A Stuck-at fault is a particular fault model used by fault simulators and Automatic test pattern generation (ATPG) tools to mimic a manufacturing defect within an integrated circuit. Individual signals and pins are assumed to be stuck at Logical '1', '0' and 'X'. For example, an output is tied to a logical 1 state **during test generation** to assure that a manufacturing defect with that type of behavior can be found with a specific test pattern. Likewise the output could be tied to a logical 0 to model the behavior of a defective circuit that cannot switch its output pin.

**Why is NAND gate preferred over NOR gate for fabrication?**

Draw the CMOS logic for both NAND and NOR gates, calculate the W/L Ratios and compare   
W/L ratio of NAND is lower than NOR, so Area of NAND is less i.e. power dissipation also less so we prefer NAND only.

**What is Noise Margin? Explain the procedure to determine Noise Margin**  
How much noise can a gate input see before it does not recognize the input?



**Explain sizing of the inverter?**  
In order to drive the desired load capacitance we have to increase the size (width) of the inverters to get an optimized performance. 

**What happens to delay if you increase load capacitance?**Delay increases.   
  
**What happens to delay if we include a resistance at the output of a CMOS circuit?**Increases (RC delay).   
  
**What are the limitations in increasing the power supply to reduce delay?**The delay can be reduced by increasing the power supply but if we do so the heating effect comes because of excessive power, to compensate this we have to increase the die size which is not practical.

**What is Charge Sharing? Explain the Charge Sharing problem while sampling data from a Bus?**In the serially connected NMOS logic the input capacitance of each gate shares the charge with the load capacitance by which the logical levels drastically mismatched than that of the desired once. To eliminate this, load capacitance must be very high compared to the input capacitance of the gates (approximately 10 times).   
In digital electronics, charge sharing is an undesirable [signal integrity](http://en.wikipedia.org/wiki/Signal_integrity) phenomenon observed most commonly in the [Domino logic](http://en.wikipedia.org/wiki/Domino_logic) family of [digital circuits](http://en.wikipedia.org/wiki/Digital_circuits). The charge sharing problem occurs when the charge which is stored at the output node in the phase is shared among the output or junction capacitances of transistors which are in the evaluation phase. Charge sharing may degrade the output voltage level or even cause erroneous output value.

**What are pros/cons of latch/FF (Flip Flop)?**

always @(d) begin

If (clk == 1’b1) begin q <= d;

end

end

D-Flip-flop:

always @(posedge clk) begin

q <= d;

end

Pros and cons of latch and Flip-Flop:

Latch takes less area, consume less power, and facilitate time borrowing or cycle stealing, not friendly with DFT tools

Latches are used in asynchronous systems.

Flip-flop takes more area, consumes more power, allow synchronous logic, friendly with DFT tools

**How to determine the depth of FIFO/ size of the FIFO?**

The logic is very simple. Suppose there is a person A and B. Now A can throw 10 mangoes at an instant of time. But B can catch only 6 mangoes at the same amount of time. So he cannot catch remaining 4 mangoes from A. So he requires a bag of size 4. This is the fundamental behind the FIFO size/depth calculation.

In technical terms I will explain:

Assume sender as Person A

Assume Receiver as Person B.

Assume FIFO as a bag.

Sender is writing data to FIFO and receiver is reading some data from FIFO. If sender writes 10 data in t time and receiver reads 6 data in the same t time then FIFO size is 10-6=4.

FIFO has two frequencies one is sender frequency (fs) and the receiver frequency (fr).

fs =1/Ts;

fr =1/Tr;

Now sender wants to transmit M words of data but FIFO can take only n words of data in Ts time. So the time taken to transmit M words is (M/n)\*Ts. That is sender wants M/n\*Ts time to write M words of data into FIFO. But our receiver can read 'p' words in Tr time interval. In Tr times receiver can read p words. So in M/n\*Ts time how many words can receiver read.

It is (M/n\*Ts\*p)/Tr words the receiver can read.

Till now what did we do?

We calculated how much time we want to write M words of data into FIFO. In the same amount of time how much data can our FIFO read. Simply subtract the data read from FIFO from the data written into the FIFO.

Here the data written into the FIFO is M words

Data read from the FIFO is (M/n\*Ts\*p)/Tr words.

So FIFO size is = M-(M/n\*Ts\*p)/Tr

This is equivalent to our example 10 - 6.

Deduce this u will get

Fifo size= M (n \* fs - &nbspp \*fr) / (n \*fs)

Where

M= Max number of bytes that the sender can send

n= Number of bytes that the sender sends per clock

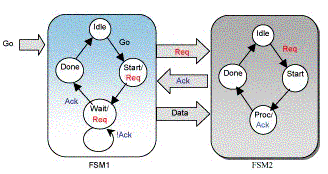
fs= sender clock frequency

p= Number of bytes that receiver can receive per clock

fr = receiver clock frequency

**What are different ways to synchronize between two clock domains?**

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.

The second method is based on using a FIFO between two systems.

**What is glitch? What causes it (explain with waveform)? How to overcome it?**   
A glitch is an undesired narrow pulse. Glitches are more likely to occur in multi-level logic networks because the signals arrive at gates at different times.

**Difference between mealy and moore state machine?**

A **Mealy** machine has outputs that depend on the state and input (thus, the FSM has the output written on edges)

A **Moore** machine has outputs that depend on state only (thus, the FSM has the output written in the state itself)

**What is the difference between one-hot and binary encoding?**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 one-hot 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 one-hot FSM requires a flip-flop for each state in the design  
FPGA vendors frequently recommend using a one-hot state encoding style because flip-flops are plentiful in an FPGA and the combinational logic required to implement a one-hot 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, one-hot FSMs typically run faster than a binary encoded FSM with larger combinational logic blocks

**What will happen if contents of register are shifted left, right?**

What is expected is in a left shift value gets Multiplied by 2 e.g. 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.

**Why do we have to identify the type of circuit? Does it really matter? (Combinational or sequential)**

It is important to identify the type of circuit because our timing calculation approach differs accordingly. Combinational circuits timing analysis deals primarily with propagation delay issues. Sequential circuits have additional specific timing characteristics that must be satisfied in order to prevent meta-stability, including setup time, hold time, and minimum clock period. Designers of sequential devices must specify these important timing characteristics in order to allow the device to be used without error.

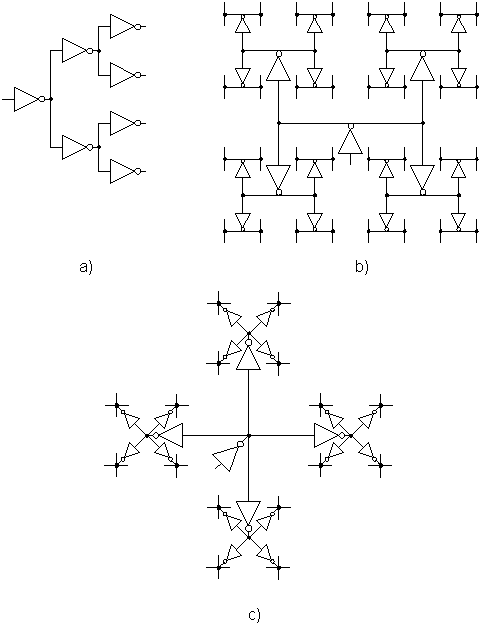
**What is the first thing to do if given a sequential circuit and asked to analyze its timing?**

Given a sequential circuit it is often advisable to first divide the circuit into three distinct parts i.e. Input Logic, State Memory and the Output Logic. Such division will also help with identifying whether the given circuit is Mealy or Moore.  The input logic (Next State Logic) and the output logic blocks constitute of only combinational logic components like gates, muxes etc. The state memory block is made of only sequential components like Flip Flops etc.

Clock distribution

**What are clock tree types?**

H tree, Balanced tree, X tree, Clustering tree, Fish bone

  
Examples of clock distribution network: a) binary tree, b) H-tree, c) X-tree

**In which layer do you prefer for clock routing and why?**

Next lower layer to the top two metal layers (global routing layers). Because it has less resistance hence less RC delay.

[**What is difference between normal buffer and clock buffer?**](http://asic-soc.blogspot.com/2007/10/what-is-difference-between-normal.html)

Clock net is one of the High Fan out Nets (HFN). The clock buffers are designed with some special property like high drive strength and less delay. Clock buffers have equal rise and fall time. This prevents duty cycle of clock signal from changing when it passes through a chain of clock buffers. Normal buffers are designed with W/L ratio such that sum of rise time and fall time is minimum. They too are designed for higher drive strength.

**Why for clock no synthesis is carried out in front end?**

Because no placement information of flip-flops is available. So synthesis won't meet true skew targets. In backend clock tree synthesis tries to meet "skew" targets. It inserts clock buffers (which have equal rise and fall time, unlike normal buffers). There is no skew information for any HFNs.

[**Is it possible to have a zero skew in the design?**](http://asic-soc.blogspot.com/2007/10/is-it-possible-to-have-zero-skew-in.html)

Theoretically it is possible! Practically it is impossible! Practically we can’t reduce any delay to zero. Delay will exist. Hence we try to make skew "equal" (or same) rather than "zero" now with this optimization all flops get the clock edge with same delay relative to each other.... so virtually we can say they are having "zero skew " or skew is "balanced".

**What is a Clock distribution network?**   
The clock distribution network distributes the clock signal(s) from a common point to all the elements that need it.   
Since the data signals are provided with a temporal reference by the clock signals, the clock waveforms must be particularly clean and sharp. Furthermore, these clock signals are particularly affected by technology scaling (see Moore's law), in that long global interconnect lines become significantly more resistive as line dimensions are decreased. This increased line resistance is one of the primary reasons for the increasing significance of clock distribution on synchronous performance. Finally, the control of any differences and uncertainty in the arrival times of the clock signals can severely limit the maximum performance of the entire system and create catastrophic race conditions in which an incorrect data signal may latch within a register. The clock distribution network often takes a significant fraction of the power consumed by a chip. Furthermore, significant power can be wasted in transitions within blocks, even when their output is not needed.

**Why do we gradually increase the size of inverters in buffer design? Why not give the output of a circuit to one large inverter?** We cannot use a big inverter to drive a large output capacitance because, who will drive the big inverter? The signal that has to drive the output cap will now see a larger gate capacitance of the big inverter. So this results in slow rise or fall times. A unit inverter can drive approximately an inverter that’s 4 times bigger in size. So say we need to drive a cap of 64 unit inverter then we try to keep the sizing like say 1, 4, 16, 64 so that each inverter sees a same ratio of output to input cap. This is the prime reason behind going for progressive sizing.

**How will you synthesize clock tree?**

Single clock-normal synthesis and optimization

Multiple clocks with domain crossing-Synthesis each clock separately and balance the skew

**How many clocks were there in this project?**

It is specific to your project. More the clocks more challenging!

**How did you handle all those clocks?**

Multiple clocks--> synthesize separately--> balance the skew--> optimize the clock tree

**Do they come from separate external resources or PLL?**

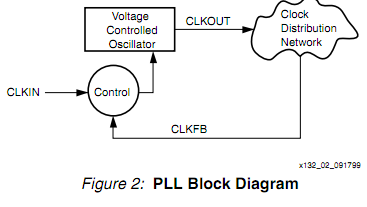
If it is from separate clock sources (i.e. asynchronous; from different pads or pins) then balancing skew between these clock sources becomes challenging.

If it is from PLL (i.e. synchronous) then skew balancing is comparatively easy.

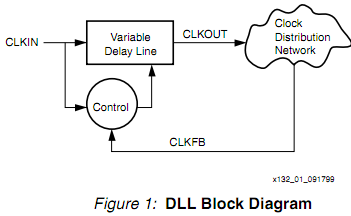
**Why buffers are used in clock tree?**

To balance skew (i.e. flop to flop delay)

**What is ring oscillator? And derive the frequency of operation?**  
Ring oscillator circuit is a coupled inverter chain with the output being connected to the input as feedback. The number of stages (inverters) is always odd to ensure that there is no single stable state (output value). Sometimes one of the stages consists of a logic gate which is used to initialize and control the circuit. The total time period of operation is the product of 2\*number of gates and gate (inverter) delay. And frequency of operation will be inverse of time period.  
  
**Compare PLL & DLL?**The PLL control logic compares the input clock to the feedback clock CLKFB and adjusts the oscillator clock until the rising edge of the input clock aligns with the feedback clock. The PLL then "locks."



A DLL works by inserting delay between the input clock and the feedback clock until the two rising edges align, putting the two clocks 360° out of phase (meaning they are in phase). After the edges from the input clock line up with the edges from the feedback clock, the DLL "locks." As long as the circuit is not evaluated until after the DLL locks, the two clocks have no discernible difference. Thus, the DLL output clock compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads.



When it comes to choosing between a PLL and a DLL for a particular application, the differences in the architectures must be understood. The oscillator used in the PLL inherently introduces instability and an accumulation of phase error. This in turn degrades the performance of the PLL when attempting to compensate for the delay of the clock distribution network. Conversely, the unconditionally stable DLL architecture does not accumulate phase error. For this reason, for delay compensation and clock conditioning, DLL architecture should be used. On the other hand, the PLL typically has an advantage when it comes to frequency synthesis.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Power

**How is clock gating done?**

Clock tree consume more than 50 % of dynamic power. The components of this power are:

1) Power consumed by combinatorial logic whose values are changing on each clock edge   
2) Power consumed by flip-flops  
3) The power consumed by the clock buffer tree in the design

Clock gating provides a way to selectively stop the clock, and thus, force the original circuit to make no transition, whenever the computation that is to be carried out at the next clock cycle is redundant. In other words, the clock signal is disabled according to the idle conditions of the logic network. It is good design idea to turn off the clock when it is not needed.

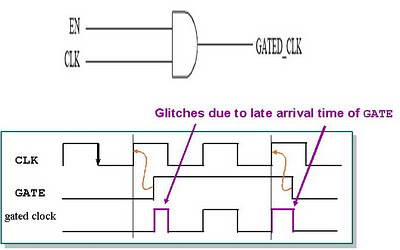
RTL clock gating uses an enable signal to control a clock gating circuit which is connected to the clock ports of all of the flip-flops with the common enable term. Therefore, if a bank of flip-flops which share a common enable term have RTL clock gating implemented, the flip-flops will consume zero dynamic power as long as this enable signal is false.

There are two types of clock gating styles available. They are:

1) Latch-based clock gating   
2) Latch-free clock gating.

**Latch free clock gating**

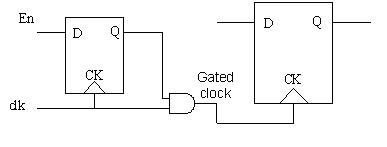
The latch-free clock gating style uses a simple AND or OR gate (depending on the edge on which flip-flops are triggered). Here if enable signal goes inactive in between the clock pulse or if it multiple times then gated clock output either can terminate prematurely or generate multiple clock pulses. This restriction makes the latch-free clock gating style inappropriate for our single-clock flip-flop based design.



**Latch free clock gating**

**Latch based clock gating**

The latch-based clock gating style adds a level-sensitive latch to the design to hold the enable signal from the active edge of the clock until the inactive edge of the clock. Since the latch captures the state of the enable signal and holds it until the complete clock pulse has been generated, the enable signal need only be stable around the rising edge of the clock, just as in the traditional un-gated design style.

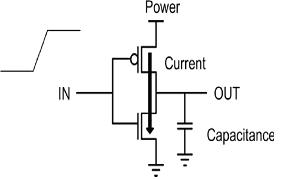


**Latch based clock gating**

Specific clock gating cells are required in library to be utilized by the synthesis tools. Availability of clock gating cells and automatic insertion by the EDA tools makes it simpler method of low power technique. Advantage of this method is that clock gating does not require modifications to RTL description.

**Explain short circuit current?**

When dynamic power is analyzed the switching component of power consumption, an instantaneous rise time was assumed, which insures that only one of the transistors is ON. In practice, finite rise and fall times results in a direct current path between the supply and ground, GND, this exists for a short period of time during switching.



Consider an example of inverter. During switching both NMOS and PMOS transistors in the circuit conduct simultaneously for a short amount of time. Therefore reducing the input transition times will decrease the short circuit current component. But propagation delay requirements have to be considered while doing so. To minimize the total average short-circuits current, it is desirable to have equal input and output edge times. In this case, the power consumed by the short-circuit currents is typically less than 10% of the total dynamic power. An important point to note is that if the supply is lowered to be belowthe sum of the thresholds of the transistors, Vdd (lesser than) Vthn + |Vthp|, the short-circuit currents can be eliminated because both devices will not be on at the same time for any value of input voltage.

**Leakage power:**

Leakage power consumption is the power consumed by the sub-threshold currents and by reverse biased diodes in a source and drain CMOS transistor, and also gate leakage.

**Sub-threshold conduction:**

It is the current between the source and drain of a MOSFET when the transistor is in sub-threshold region, or weak-inversion region (gate-to-source voltages below the threshold voltage). In digital circuits, sub-threshold conduction is generally viewed as a parasitic leakage in a state that would ideally have no current.

**Power components of CMOS technology:**

Ptotal = Pstatic + Pdynamic + Pshort-circuit + Pglitch

Static power: leakage

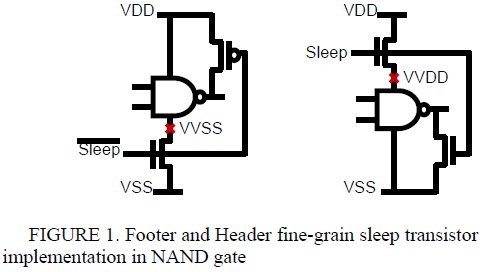
Dynamic power: Cload.Vdd^(2).f

**Power gating**

This technique uses high Vt sleep transistors which cut off VDD from a circuit block when the block is not switching. The sleep transistor sizing is an important design parameter.

**Sleep transistors**

In the power gating, sleep transistors are used as switches to shut off power supplies to parts of a design in standby mode.



VVSS, VVDD: Virtual power nets

Sizing sleep transistors is a challenge.

**For CMOS logic, give the various techniques you know to minimize power consumption?**Power dissipation = C.Vdd^(2).f , from this minimize the load capacitance, dc voltage and the operating frequency.

**How can you reduce dynamic power?**

Reduce switching activity by designing good RTL

Clock gating

Architectural improvements

Reduce supply voltage

Use multiple voltage domains-Multi Vdd

**During power analysis, if you are facing IR drop problem, then how did you avoid?**

Increase power metal layer width. Go for higher metal layer. Spread macros or standard cells.

Provide more straps.

**Do you know about input vector controlled method of leakage reduction?**

Leakage current of a gate is dependent on its inputs also. Hence find the set of inputs which gives least leakage. By applying this minimum leakage vector to a circuit it is possible to decrease the leakage current of the circuit when it is in the standby mode. This method is known as input vector controlled method of leakage reduction.

**What are the vectors of dynamic power?**

Voltage and Current