Verilog interview Questions & answers for FPGA & ASIC.

1) Write a Verilog code to swap contents of two registers with and without a temporary register?

With temp reg ;

```
always @ (posedge clock)
begin
temp=b;
b=a;
a=temp;
end
```

Without temp reg;

```
always @ (posedge clock)
begin
a <= b;
b <= a;
end
```

2) Difference between blocking and non-blocking? (Verilog interview questions that is most commonly asked)

The Verilog language has two forms of the procedural assignment statement: blocking and non-blocking. The two are distinguished by the = and <= assignment operators. The blocking assignment statement (= operator) acts much like in traditional programming languages. The whole statement is done before control passes on to the next statement. The non-blocking (<= operator) evaluates all the right-hand sides for the current time unit and assigns the left-hand sides at the end of the time unit. For example, the following Verilog program

```
// testing blocking and non-blocking assignment

module blocking;
reg [0:7] A, B;
initial begin: init1
A = 3;
#1 A = A + 1; // blocking procedural assignment
B = A + 1;

$display("Blocking: A= %b B= %b", A, B ); A = 3;
#1 A <= A + 1; // non-blocking procedural assignment
B <= A + 1;
#1 $display("Non-blocking: A= %b B= %b", A, B );
```

```
end
endmodule
```

produces the following output:

```
Blocking: A= 00000100 B= 00000101
Non-blocking: A= 00000100 B= 00000100
```

The effect is for all the non-blocking assignments to use the old values of the variables at the beginning of the current time unit and to assign the registers new values at the end of the current time unit. This reflects how register transfers occur in some hardware systems.
blocking procedural assignment is used for combinational logic and non-blocking procedural assignment for sequential.

2.1). Tell me about verilog file I/O?

OPEN A FILE

```
integer file;
file = $fopenr("filename");
file = $fopenw("filename");
file = $fopena("filename");
```

The function $fopenr opens an existing file for reading. $fopenw opens a new file for writing, and $fopena opens a new file for writing where any data will be appended to the end of the file. The file name can be either a quoted string or a reg holding the file name. If the file was successfully opened, it returns an integer containing the file number (1..MAX_FILES) or NULL (0) if there was an error. Note that these functions are not the same as the built-in system function $fopen which opens a file for writing by $fdisplay. The files are opened in C with 'rb', 'wb', and 'ab' which allows reading and writing binary data on the PC. The 'b' is ignored on Unix.

CLOSE A FILE

```
integer file, r;
r = $fcloser(file);
r = $fclosew(file);
```

The function $fcloser closes a file for input. $fclosew closes a file for output. It returns EOF if there was an error, otherwise 0. Note that these are not the same as $fclose which closes files for writing.

3) Difference between task and function?

Function:
A function is unable to enable a task however functions can enable other functions.
A function will carry out its required duty in zero simulation time. (The program time will not be

incremented during the function routine)
Within a function, no event, delay or timing control statements are permitted
In the invocation of a function their must be at least one argument to be passed.
Functions will only return a single value and can not use either output or inout statements.

Tasks:
Tasks are capable of enabling a function as well as enabling other versions of a Task
Tasks also run with a zero simulation however they can if required be executed in a non zero simulation time.
Tasks are allowed to contain any of these statements.
A task is allowed to use zero or more arguments which are of type output, input or inout.
A Task is unable to return a value but has the facility to pass multiple values via the output and inout statements .

4) Difference between inter statement and intra statement delay?

```
//define register variables
reg a, b, c;

//intra assignment delays
initial
begin
a = 0; c = 0;
b = #5 a + c; //Take value of a and c at the time=0, evaluate
//a + c and then wait 5 time units to assign value
//to b.
end

//Equivalent method with temporary variables and regular delay control
initial
begin
a = 0; c = 0;
temp_ac = a + c;
#5 b = temp_ac; //Take value of a + c at the current time and
//store it in a temporary variable. Even though a and c
//might change between 0 and 5,
//the value assigned to b at time 5 is unaffected.
end
```

5) What is delta simulation time?

6) Difference between $monitor,$display & $strobe?

These commands have the same syntax, and display text on the screen during simulation. They are much less convenient than waveform display tools like cwaves?. $display and $strobe display once every time they are executed, whereas $monitor displays every time one of its parameters changes.
The difference between $display and $strobe is that $strobe displays the parameters at the very end of the current simulation time unit rather than exactly where it is executed. The format string is like that in C/C++, and may contain format characters. Format characters include %d (decimal), %h (hexadecimal), %b (binary), %c (character), %s (string) and %t (time), %m (hierarchy level). %5d, %5b etc. would give exactly 5 spaces for the number instead of the space needed. Append b, h, o to the task name to change default format to binary, octal or hexadecimal.
Syntax:

```
$display (“format_string”, par_1, par_2, ... );
$strobe (“format_string”, par_1, par_2, ... );
$monitor (“format_string”, par_1, par_2, ... );
```

7) What is difference between Verilog full case and parallel case?

A "full" case statement is a case statement in which all possible case-expression binary patterns can be matched to a case item or to a case default. If a case statement does not include a case default and if it is possible to find a binary case expression that does not match any of the defined case items, the case statement is not "full."
A "parallel" case statement is a case statement in which it is only possible to match a case expression to one and only one case item. If it is possible to find a case expression that would match more than one case item, the matching case items are called "overlapping" case items and the case statement is not "parallel."

8) What is meant by inferring latches,how to avoid it?

Consider the following :

```
always @(s1 or s0 or i0 or i1 or i2 or i3)
case ({s1, s0})
2'd0 : out = i0;
2'd1 : out = i1;
2'd2 : out = i2;
endcase
```

in a case statement if all the possible combinations are not compared and default is also not specified like in example above a latch will be inferred ,a latch is inferred because to reproduce the previous value when unknown branch is specified.
For example in above case if {s1,s0}=3 , the previous stored value is reproduced for this storing a latch is inferred.
The same may be observed in IF statement in case an ELSE IF is not specified.
To avoid inferring latches make sure that all the cases are mentioned if not default condition is provided.

9) Tell me how blocking and non blocking statements get executed?

Execution of blocking assignments can be viewed as a one-step process:
1. Evaluate the RHS (right-hand side equation) and update the LHS (left-hand side expression) of the blocking assignment without interruption from any other Verilog statement. A blocking assignment "blocks" trailing assignments in the same always block from occurring until after the current assignment has been completed

Execution of nonblocking assignments can be viewed as a two-step process:
1. Evaluate the RHS of nonblocking statements at the beginning of the time step. 2. Update the LHS of nonblocking statements at the end of the time step.

10) Variable and signal which will be Updated first?

Signals

11) What is sensitivity list?

The sensitivity list indicates that when a change occurs to any one of elements in the list change, begin…end statement inside that always block will get executed.

12) In a pure combinational circuit is it necessary to mention all the inputs in sensitivity disk? if yes, why?

Yes in a pure combinational circuit is it necessary to mention all the inputs in sensitivity disk other wise it will result in pre and post synthesis mismatch.

13) Tell me structure of Verilog code you follow?

A good template for your Verilog file is shown below.

```
// timescale directive tells the simulator the base units and precision of the simulation
`timescale 1 ns / 10 ps
module name (input and outputs);
// parameter declarations
parameter parameter_name = parameter value;
// Input output declarations
input in1;
input in2; // single bit inputs
output [msb:lsb] out; // a bus output
// internal signal register type declaration - register types (only assigned within always statements). reg
register variable 1;
reg [msb:lsb] register variable 2;
```

```
// internal signal. net type declaration - (only assigned outside always statements) wire net variable 1;
// hierarchy - instantiating another module
reference name instance name (
.pin1 (net1),
.pin2 (net2),
.
.pinn (netn)
);
// synchronous procedures
always @ (posedge clock)
begin
.
end
// combinatinal procedures
always @ (signal1 or signal2 or signal3)
begin
.
end
assign net variable = combinational logic;
endmodule
```

14) Difference between Verilog and vhdl?

Compilation
VHDL. Multiple design-units (entity/architecture pairs), that reside in the same system file, may be separately compiled if so desired. However, it is good design practice to keep each design unit in it's own system file in which case separate compilation should not be an issue.

Verilog. The Verilog language is still rooted in it's native interpretative mode. Compilation is a means of speeding up simulation, but has not changed the original nature of the language. As a result care must be taken with both the compilation order of code written in a single file and the compilation order of multiple files. Simulation results can change by simply changing the order of compilation.

Data types
VHDL. A multitude of language or user defined data types can be used. This may mean dedicated conversion functions are needed to convert objects from one type to another. The choice of which data types to use should be considered wisely, especially enumerated (abstract) data types. This will make models easier to write, clearer to read and avoid unnecessary conversion functions that can clutter the code. VHDL may be preferred because it allows a multitude of language or user defined data types to be used.

Verilog. Compared to VHDL, Verilog data types a re very simple, easy to use and very much geared towards modeling hardware structure as opposed to abstract hardware modeling. Unlike VHDL, all data types used in a Verilog model are defined by the Verilog language and not by the user. There are net data types, for example wire, and a register data type called reg. A model with a signal whose type is one of the net data types has a corresponding electrical wire in the implied modeled circuit. Objects, that is signals, of type reg hold their value over simulation delta cycles and should not be confused with the modeling of a hardware register. Verilog may be preferred because of it's simplicity.

Design reusability
VHDL. Procedures and functions may be placed in a package so that they are avail able to any design-unit that wishes to use them.

Verilog. There is no concept of packages in Verilog. Functions and procedures used within a model must be defined in the module. To make functions and procedures generally accessible from different module statements the functions and procedures must be placed in a separate system file and included using the `include compiler directive.

15) What are different styles of Verilog coding I mean gate-level,continuous level and others explain in detail?

16) Can you tell me some of system tasks and their purpose?

$display, $displayb, $displayh, $displayo, $write, $writeb, $writeh, $writeo.
The most useful of these is $display.This can be used for displaying strings, expression or values of variables.
Here are some examples of usage.
```
$display("Hello oni");
--- output: Hello oni
$display($time) // current simulation time.
--- output: 460
counter = 4'b10;
$display(" The count is %b", counter);
--- output: The count is 0010
```
$reset resets the simulation back to time 0; $stop halts the simulator and puts it in interactive mode where the
user can enter commands; $finish exits the simulator back to the operating system

17) Can you list out some of enhancements in Verilog 2001?

In earlier version of Verilog ,we use 'or' to specify more than one element in sensitivity list . In Verilog 2001, we can use comma as shown in the example below.
```
// Verilog 2k example for usage of comma
always @ (i1,i2,i3,i4)
```

Verilog 2001 allows us to use star in sensitive list instead of listing all the variables in RHS of combo logics . This removes typo mistakes and thus avoids simulation and synthesis mismatches,
Verilog 2001 allows port direction and data type in the port list of modules as shown in the example below

```
module memory (
input r,
input wr,
input [7:0] data_in,
input [3:0] addr,
output [7:0] data_out
);
```

18)Write a Verilog code for synchronous and asynchronous reset?

Synchronous reset, synchronous means clock dependent so reset must not be present in sensitivity disk eg:

```
always @ (posedge clk )

begin if (reset)
. . . end
```

Asynchronous means clock independent so reset must be present in sensitivity list.
Eg

```
Always @(posedge clock or posedge reset)
begin
if (reset)
. . . end
```

19) What is pli?why is it used?

Programming Language Interface (PLI) of Verilog HDL is a mechanism to interface Verilog programs with programs written in C language. It also provides mechanism to access internal databases of the simulator from the C program.
PLI is used for implementing system calls which would have been hard to do otherwise (or impossible) using Verilog syntax. Or, in other words, you can take advantage of both the paradigms - parallel and hardware related features of Verilog and sequential flow of C - using PLI.

20) There is a triangle and on it there are 3 ants one on each corner and are free to move along sides of triangle what is probability that they will collide?

Ants can move only along edges of triangle in either of direction, let’s say one is represented by 1 and another by 0, since there are 3 sides eight combinations are possible, when all ants are going in same direction they won’t collide that is 111 or 000 so probability of not collision is 2/8=1/4 or collision probability is 6/8=3/4.

FPGA interview questions & answers.

What is FPGA ?

A field-programmable gate array is a semiconductor device containing programmable logic components called "logic blocks", and programmable interconnects. Logic blocks can be programmed to perform the function of basic logic gates such as AND, and XOR, or more complex combinational functions such as decoders or mathematical functions. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. A hierarchy of programmable interconnects allows logic blocks to be interconnected as needed by the system designer, somewhat like a one-chip programmable breadboard. Logic blocks and interconnects can be programmed by the customer or designer, after the FPGA is manufactured, to implement any logical function—hence the name "field-programmable". FPGAs are usually slower than their application-specific integrated circuit (ASIC) counterparts, cannot handle as complex a design, and draw more power (for any given semiconductor process). But their advantages include a shorter time to market, ability to re-program in the field to fix bugs, and lower non-recurring engineering costs. Vendors can sell cheaper, less flexible versions of their FPGAs which cannot be modified after the design is committed. The designs are developed on regular FPGAs and then migrated into a fixed version that more resembles an ASIC.

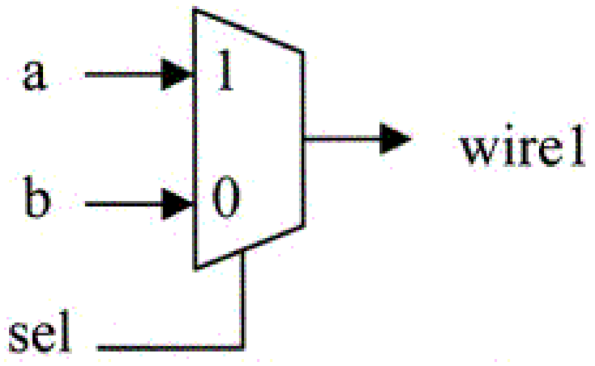
What logic is inferred when there are multiple assign statements targeting the same wire?

It is illegal to specify multiple assign statements to the same wire in a synthesizable code that will become an output port of the module. The synthesis tools give a syntax error that a net is being driven by more than one source.
However, it is legal to drive a three-state wire by multiple assign statements.

What do conditional assignments get inferred into?

Conditionals in a continuous assignment are specified through the “?:” operator. Conditionals get inferred into a multiplexor. For example, the following is the code for a simple multiplexor

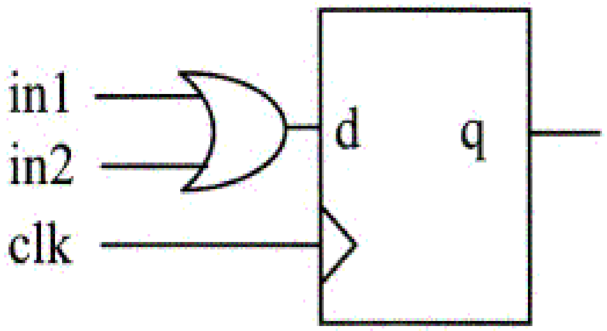
```
assign wire1 = (sel==1'b1) ? a : b;
```



What value is inferred when multiple procedural assignments made to the same reg variable in an always block?

When there are multiple nonblocking assignments made to the same reg variable in a sequential always block, then the last assignment is picked up for logic synthesis. For example

```
always @ (posedge clk) begin
out <= in1^in2;
out <= in1 &in2;
out <= in1|in2;
```



In the example just shown, it is the OR logic that is the last assignment. Hence, the logic synthesized was indeed the OR gate. Had the last assignment been the "&" operator, it would have synthesized an AND gate.

1) What is minimum and maximum frequency of dcm in spartan-3 series fpga?

Spartan series dcm's have a minimum frequency of 24 MHZ and a maximum of 248

2)Tell me some of constraints you used and their purpose during your design?

There are lot of constraints and will vary for tool to tool ,I am listing some of Xilinx constraints
a) Translate on and Translate off: the Verilog code between Translate on and Translate off is ignored for synthesis.
b) CLOCK_SIGNAL: is a synthesis constraint. In the case where a clock signal goes through combinatorial logic before being connected to the clock input of a flip-flop, XST cannot identify what input pin or internal net is the real clock signal. This constraint allows you to define the clock net.
c) XOR_COLLAPSE: is synthesis constraint. It controls whether cascaded XORs should be collapsed into a single XOR.
For more constraints detailed description refer to constraint guide.

3) Suppose for a piece of code equivalent gate count is 600 and for another code equivalent gate

count is 50,000 will the size of bitmap change?in other words will size of bitmap change it gate count change?

The size of bitmap is irrespective of resource utilization, it is always the same,for Spartan xc3s5000 it is 1.56MB and will never change.

4) What are different types of FPGA programming modes?what are you currently using ?how to change from one to another?

Before powering on the FPGA, configuration data is stored externally in a PROM or some other nonvolatile medium either on or off the board. After applying power, the configuration data is written to the FPGA using any of five different modes: Master Parallel, Slave Parallel, Master Serial, Slave Serial, and Boundary Scan (JTAG). The Master and Slave Parallel modes
Mode selecting pins can be set to select the mode, refer data sheet for further details.

5) Tell me some of features of FPGA you are currently using?

I am taking example of xc3s5000 to answering the question .

Very low cost, high-performance logic solution for
high-volume, consumer-oriented applications
- Densities as high as 74,880 logic cells
- Up to 784 I/O pins
- 622 Mb/s data transfer rate per I/O
- 18 single-ended signal standards
- 6 differential I/O standards including LVDS, RSDS
- Termination by Digitally Controlled Impedance
- Signal swing ranging from 1.14V to 3.45V
- Double Data Rate (DDR) support
• Logic resources
- Abundant logic cells with shift register capability
- Wide multiplexers
- Fast look-ahead carry logic
- Dedicated 18 x 18 multipliers
- Up to 1,872 Kbits of total block RAM
- Up to 520 Kbits of total distributed RAM
• Digital Clock Manager (up to four DCMs)
- Clock skew elimination
• Eight global clock lines and abundant routing

6) What is gate count of your project?

Well mine was 3.2 million, I don’t know yours.!

7) Can you list out some of synthesizable and non synthesizable constructs?

not synthesizable->>>>
initial
ignored for synthesis.
delays
ignored for synthesis.
events
not supported.
real
Real data type not supported.
time
Time data type not supported.
force and release
Force and release of data types not supported.
fork join
Use nonblocking assignments to get same effect.
user defined primitives
Only gate level primitives are supported.

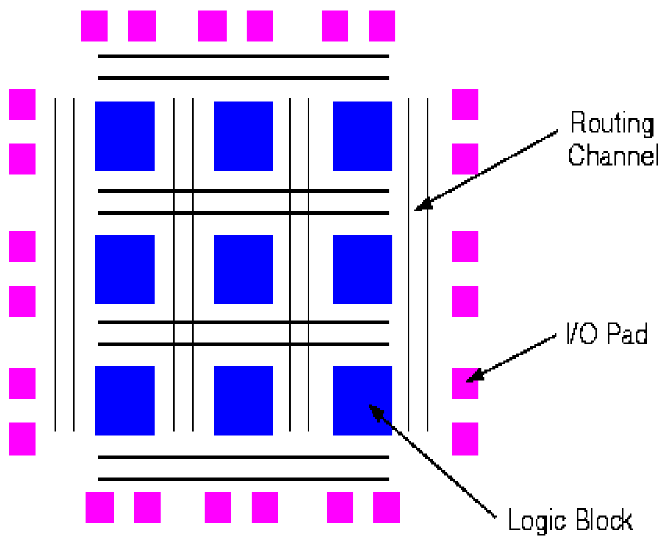
synthesizable constructs->>
assign,for loop,Gate Level Primitives,repeat with constant value...

8)Can you explain what struck at zero means?

These stuck-at problems will appear in ASIC. Some times, the nodes will permanently tie to 1 or 0 because of some fault. To avoid that, we need to provide testability in RTL. If it is permanently 1 it is called stuck-at-1 If it is permanently 0 it is called stuck-at-0.

9) Can you draw general structure of fpga?

FPGA structure



10) Difference between FPGA and CPLD?

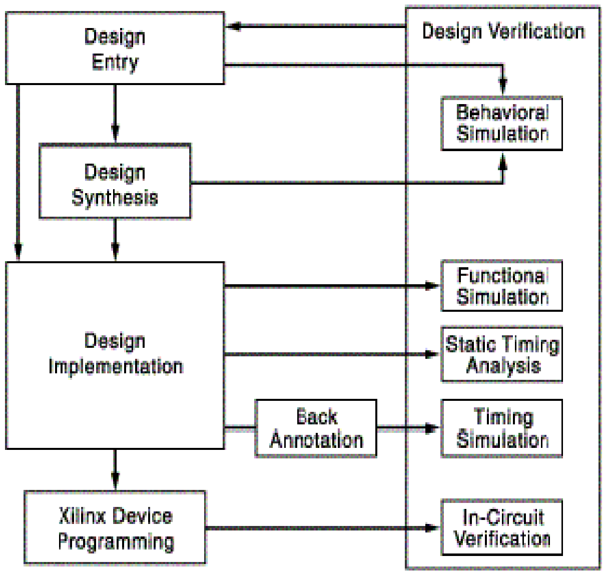
FPGA:
a)SRAM based technology.
b)Segmented connection between elements.
c)Usually used for complex logic circuits.
d)Must be reprogrammed once the power is off.
e)Costly

CPLD:
a)Flash or EPROM based technology.
b)Continuous connection between elements.
c)Usually used for simpler or moderately complex logic circuits.
d)Need not be reprogrammed once the power is off.
e)Cheaper

11) What are dcm's?why they are used?

Digital clock manager (DCM) is a fully digital control system that
uses feedback to maintain clock signal characteristics with a
high degree of precision despite normal variations in operating
temperature and voltage.
That is clock output of DCM is stable over wide range of temperature and voltage , and also skew associated with DCM is minimal and all phases of input clock can be obtained . The output of DCM coming form global buffer can handle more load.

12) FPGA design flow?



Also,Please refer to presentation section synthesis ppt on this site.

13)what is slice,clb,lut?

I am taking example of xc3s500 to answer this question

The Configurable Logic Blocks (CLBs) constitute the main logic resource for implementing synchronous as well as combinatorial circuits.
CLB are configurable logic blocks and can be configured to combo,ram or rom depending on coding style
CLB consist of 4 slices and each slice consist of two 4-input LUT (look up table) F-LUT and G-LUT.

14) Can a clb configured as ram?

YES.

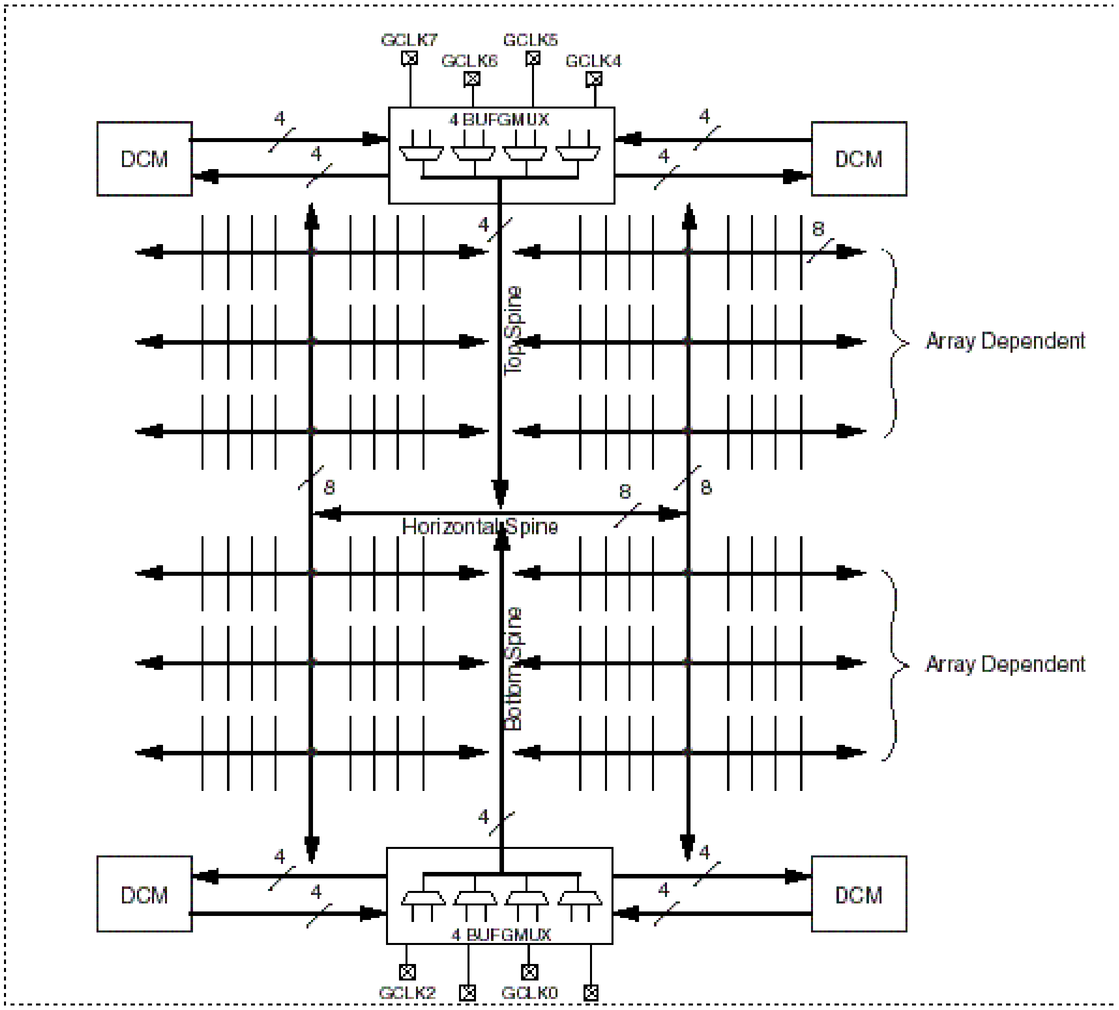
The memory assignment is a clocked behavioral assignment, Reads from the memory are asynchronous, And all the address lines are shared by the read and write statements.

15)What is purpose of a constraint file what is its extension?

The UCF file is an ASCII file specifying constraints on the logical design. You create this file and enter your constraints in the file with a text editor. You can also use the Xilinx Constraints Editor to create constraints within a UCF(extention) file. These constraints affect how the logical design is implemented in the target device. You can use the file to override constraints specified during design entry.

16) What is FPGA you are currently using and some of main reasons for choosing it?

17) Draw a rough diagram of how clock is routed through out FPGA?



18) How many global buffers are there in your current fpga,what is their significance?

There are 8 of them in xc3s5000
An external clock source enters the FPGA using a Global Clock Input Buffer (IBUFG), which directly accesses the global clock network or an Input Buffer (IBUF). Clock signals within the FPGA drive a global clock net using a Global Clock Multiplexer Buffer (BUFGMUX). The global clock net connects directly to the CLKIN input.

19) What is frequency of operation and equivalent gate count of u r project?

20)Tell me some of timing constraints you have used?

21)Why is map-timing option used?

Timing-driven packing and placement is recommended to improve design performance, timing, and packing for highly utilized designs.

22)What are different types of timing verifications?

Dynamic timing:
a. The design is simulated in full timing mode.
b. Not all possibilities tested as it is dependent on the input test vectors.
c. Simulations in full timing mode are slow and require a lot of memory.
d. Best method to check asynchronous interfaces or interfaces between different timing domains.
Static timing:
a. The delays over all paths are added up.
b. All possibilities, including false paths, verified without the need for test vectors.
c. Much faster than simulations, hours as opposed to days.
d. Not good with asynchronous interfaces or interfaces between different timing domains.

23) Compare PLL & DLL ?

PLL:
PLLs have disadvantages that make their use in high-speed designs problematic, particularly when both high performance and high reliability are required.
The PLL voltage-controlled oscillator (VCO) is the greatest source of problems. Variations in temperature, supply voltage, and manufacturing process affect the stability and operating performance of PLLs.

DLLs, however, are immune to these problems. A DLL in its simplest form inserts a variable delay line between the external clock and the internal clock. The clock tree distributes the clock to all registers and then back to the feedback pin of the DLL.
The control circuit of the DLL adjusts the delays so that the rising edges of the feedback clock align with the input clock. Once the edges of the clocks are aligned, the DLL is locked, and both the input buffer delay and the clock skew are reduced to zero.
Advantages:
· precision
· stability
· power management
· noise sensitivity
· jitter performance.

24) Given two ASICs. one has setup violation and the other has hold violation. how can they be made to work together without modifying the design?

Slow the clock down on the one with setup violations..
And add redundant logic in the path where you have hold violations.

25)Suggest some ways to increase clock frequency?

· Check critical path and optimize it.
· Add more timing constraints (over constrain).
· pipeline the architecture to the max possible extent keeping in mind latency req's.

26)What is the purpose of DRC?

DRC is used to check whether the particular schematic and corresponding layout(especially the mask sets involved) cater to a pre-defined rule set depending on the technology used to design. They are parameters set aside by the concerned semiconductor manufacturer with respect to how the masks should be placed , connected , routed keeping in mind that variations in the fab process does not effect normal functionality. It usually denotes the minimum allowable configuration.

27)What is LVs and why do we do that. What is the difference between LVS and DRC?

The layout must be drawn according to certain strict design rules. DRC helps in layout of the designs by checking if the layout is abide by those rules.
After the layout is complete we extract the netlist. LVS compares the netlist extracted from the layout with the schematic to ensure that the layout is an identical match to the cell schematic.

28)What is DFT ?

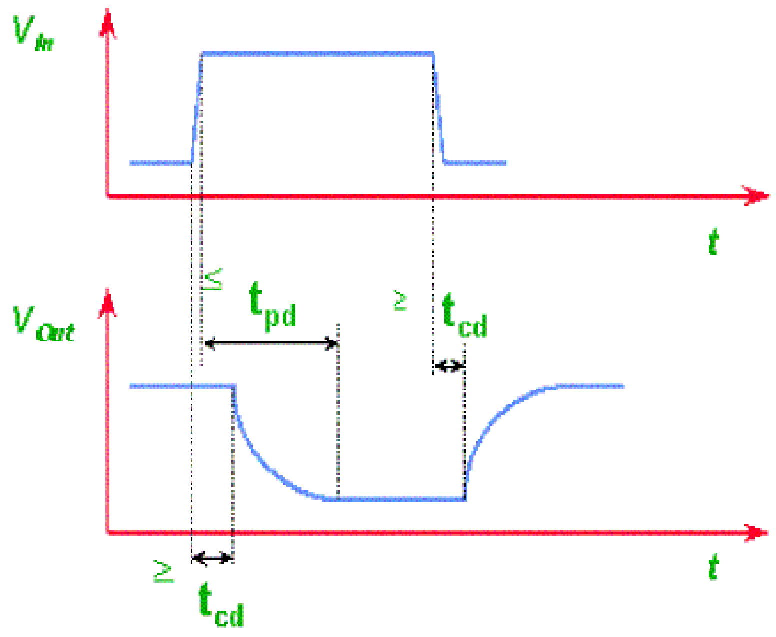
DFT means design for testability. 'Design for Test or Testability' - a methodology that ensures a design works properly after manufacturing, which later facilitates the failure analysis and false product/piece detection
Other than the functional logic,you need to add some DFT logic in your design.This will help you in testing the chip for manufacturing defects after it come from fab. Scan,MBIST,LBIST,IDDQ testing etc are all part of this. (this is a hot field and with lots of opportunities)

29) There are two major FPGA companies: Xilinx and Altera. Xilinx tends to promote its hard processor cores and Altera tends to promote its soft processor cores. What is the difference between a hard processor core and a soft processor core?

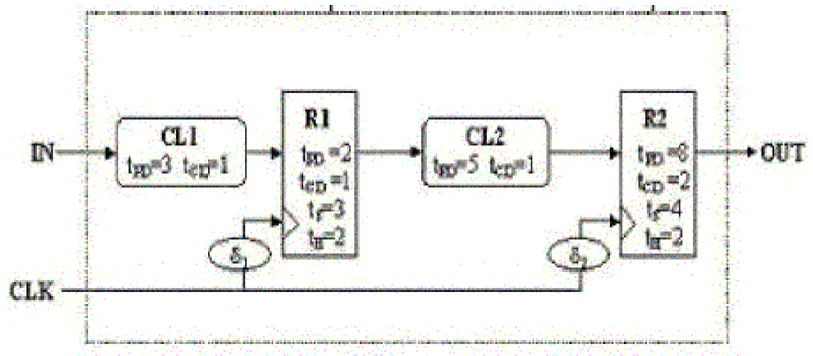
A hard processor core is a pre-designed block that is embedded onto the device. In the Xilinx Virtex II-Pro, some of the logic blocks have been removed, and the space that was used for these logic blocks is used to implement a processor. The Altera Nios, on the other hand, is a design that can be compiled to the normal FPGA logic.

30)What is the significance of contamination delay in sequential circuit timing?

Look at the figure below. tcd is the contamination delay.



Contamination delay tells you if you meet the hold time of a flip flop. To understand this better please look at the sequential circuit below.



The contamination delay of the data path in a sequential circuit is critical for the hold time at the flip flop where it is exiting, in this case R2.
mathematically, th(R2) <= tcd(R1) + tcd(CL2)
Contamination delay is also called tmin and Propagation delay is also called tmax in many data sheets.

31)When are DFT and Formal verification used?

DFT:
· manufacturing defects like stuck at "0" or "1".
· test for set of rules followed during the initial design stage.

Formal verification:
· Verification of the operation of the design, i.e, to see if the design follows spec.
· gate netlist == RTL ?
· using mathematics and statistical analysis to check for equivalence.

32)What is Synthesis?

Synthesis is the stage in the design flow which is concerned with translating your Verilog code into gates - and that's putting it very simply! First of all, the Verilog must be written in a particular way for the synthesis tool that you are using. Of course, a synthesis tool doesn't actually produce gates - it will output a netlist of the design that you have synthesised that represents the chip which can be fabricated through an ASIC or FPGA vendor.

33)We need to sample an input or output something at different rates, but I need to vary the rate? What's a clean way to do this?

Many, many problems have this sort of variable rate requirement, yet we are usually constrained with a constant clock frequency. One trick is to implement a digital NCO (Numerically Controlled Oscillator). An NCO is actually very simple and, while it is most naturally understood as hardware, it also can be constructed in software. The NCO, quite simply, is an accumulator where you keep adding a fixed value on every clock (e.g. at a constant clock frequency). When the NCO "wraps", you sample your input or do your action. By adjusting the value added to the accumulator each clock, you finely tune the AVERAGE frequency of that wrap event. Now - you may have realized that the wrapping event may have lots of jitter on it. True, but you may use the wrap to increment yet another counter where each additional Divide-by-2 bit reduces this jitter. The DDS is a related technique. I have two examples showing both an NCOs and a DDS in my File Archive. This is tricky to grasp at first, but tremendously powerful once you have it in your bag of tricks. NCOs also relate to digital PLLs, Timing Recovery, TDMA and other "variable rate" phenomena.

ASIC interview questions & answers

What is Body effect ?

The threshold voltage of a MOSFET is affected by the voltage which is applied to the back contact. The voltage difference between the source and the bulk, VBS changes the width of the depletion layer and therefore also the voltage across the oxide due to the change of the charge in the depletion region. This results in a difference in threshold voltage which equals the difference in charge in the depletion region divided by the oxide capacitance, yielding.

What are standard Cell's?
In semiconductor design, standard cell methodology is a method of designing Application Specific Integrated Circuits (ASICs) with mostly digital-logic features. Standard cell methodology is an example of design abstraction, whereby a low-level VLSI-layout is encapsulated into an abstract logic representation

(such as a NAND gate). Cell-based methodology (the general class that standard-cell belongs to) makes it possible for one designer to focus on the high-level (logical function) aspect of digital-design, while another designer focused on the implementation (physical) aspect. Along with semiconductor manufacturing advances, standard cell methodology was responsible for allowing designers to scale ASICs from comparatively simple single-function ICs (of several thousand gates), to complex multi-million gate devices (SoC).

What are Design Rule Check (DRC) and Layout Vs Schematic (LVS) ?
Design Rule Check (DRC) and Layout Vs Schematic (LVS) are verification processes. Reliable device fabrication at modern deep submicrometre (0.13 µm and below) requires strict observance of transistor spacing, metal layer thickness, and power density rules. DRC exhaustively compares the physical netlist against a set of "foundry design rules" (from the foundry operator), then flags any observed violations. LVS is a process that confirms that the layout has the same structure as the associated schematic; this is typically the final step in the layout process. The LVS tool takes as an input a schematic diagram and the extracted view from a layout. It then generates a netlist from each one and compares them. Nodes, ports, and device sizing are all compared. If they are the same, LVS passes and the designer can continue. Note: LVS tends to consider transistor fingers to be the same as an extra-wide transistor. For example, 4 transistors in parallel (each 1 um wide), a 4-finger 1 um transistor, and a 4 um transistor are all seen as the same by the LVS tool. Functionality of .lib files will be taken from spice models and added as an attribute to the .lib file.

What is Antenna effect ?
The antenna effect, more formally plasma induced gate oxide damage, is an efffect that can potentially cause yield and reliability problems during the manufacture of MOS integrated circuits. Fabs normally supply antenna rules, which are rules that must be obeyed to avoid this problem. A violation of such rules is called an antenna violation. The word antenna is somewhat of a misnomer in this context—the problem is really the collection of charge, not the normal meaning of antenna, which is a device for converting electromagnetic fields to/from electrical currents. Occasionally the phrase antenna effect is used this context[6] but this is less common since there are many effects[7] and the phrase does not make clear which is meant.

What are steps involved in Semiconductor device fabrication ?
This is a list of processing techniques that are employed numerous times in a modern electronic device and do not necessarily imply a specific order.

Wafer processing
Wet cleans
Photolithography
Ion implantation (in which dopants are embedded in the wafer creating regions of increased (or decreased) conductivity)
Dry etching
Wet etching
Plasma ashing

Thermal treatments
Rapid thermal anneal
Furnace anneals
Thermal oxidation
Chemical vapor deposition (CVD)
Physical vapor deposition (PVD)
Molecular beam epitaxy (MBE)
Electrochemical Deposition (ECD). See Electroplating
Chemical-mechanical planarization (CMP)
Wafer testing (where the electrical performance is verified)
Wafer backgrinding (to reduce the thickness of the wafer so the resulting chip can be put into a thin device like a smartcard or PCMCIA card.)
Die preparation
Wafer mounting
Die cutting
IC packaging
Die attachment
IC Bonding
Wire bonding
Flip chip
Tab bonding
IC encapsulation
Baking
Plating
Lasermarking
Trim and form
IC testing

What is Clock distribution network ?
In a synchronous digital system, the clock signal is used to define a time reference for the movement of data within that system. The clock distribution network distributes the clock signal(s) from a common point to all the elements that need it. Since this function is vital to the operation of a synchronous system, much attention has been given to the characteristics of these clock signals and the electrical networks used in their distribution. Clock signals are often regarded as simple control signals; however, these signals have some very special characteristics and attributes.
Clock signals are typically loaded with the greatest fanout, travel over the greatest distances, and operate at the highest speeds of any signal, either control or data, within the entire synchronous system. 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. These observations have lead to a power saving technique called clock gating, which involves adding logic gates to the clock distribution tree, so portions of the tree can be turned off when not needed.

What is Clock Gating ?
Clock gating is one of the power-saving techniques used on many synchronous circuits including the Pentium 4 processor. To save power, clock gating refers to adding additional logic to a circuit to prune the clock tree, thus disabling portions of the circuitry where flip flops do not change state. Although asynchronous circuits by definition do not have a "clock", the term "perfect clock gating" is used to illustrate how various clock gating techniques are simply approximations of the data-dependent behavior exhibited by asynchronous circuitry, and that as the granularity on which you gate the clock of a synchronous circuit approaches zero, the power consumption of that circuit approaches that of an asynchronous circuit.

What is Netlist ?
Netlists are connectivity information and provide nothing more than instances, nets, and perhaps some attributes. If they express much more than this, they are usually considered to be a hardware description language such as Verilog, VHDL, or any one of several specific languages designed for input to simulators.
Most netlists either contain or refer to descriptions of the parts or devices used. Each time a part is used in a netlist, this is called an "instance." Thus, each instance has a "master", or "definition". These definitions will usually list the connections that can be made to that kind of device, and some basic properties of that device. These connection points are called "ports" or "pins", among several other names.
An "instance" could be anything from a vacuum cleaner, microwave oven, or light bulb, to a resistor, capacitor, or integrated circuit chip.
Instances have "ports". In the case of a vacuum cleaner, these ports would be the three metal prongs in the plug. Each port has a name, and in continuing the vacuum cleaner example, they might be "Neutral", "Live" and "Ground". Usually, each instance will have a unique name, so that if you have two instances of vacuum cleaners, one might be "vac1" and the other "vac2". Besides their names, they might otherwise be identical.
Nets are the "wires" that connect things together in the circuit. There may or may not be any special attributes associated with the nets in a design, depending on the particular language the netlist is written in, and that language's features.
Instance based netlists usually provide a list of the instances used in a design. Along with each instance, either an ordered list of net names are provided, or a list of pairs provided, of an instance port name, along with the net name to which that port is connected. In this kind of description, the list of nets can be gathered from the connection lists, and there is no place to associate particular attributes with the nets themselves. SPICE is perhaps the most famous of instance-based netlists.
Net-based netlists usually describe all the instances and their attributes, then describe each net, and say which port they are connected on each instance. This allows for attributes to be associated with nets.

EDIF is probably the most famous of the net-based netlists.

What Physical timing closure ?
Physical timing closure is the process by which an FPGA or a VLSI design with a physical representation is modified to meet its timing requirements. Most of the modifications are handled by EDA tools based on directives given by a designer. The term is also sometimes used as a characteristic, which is ascribed to an EDA tool, when it provides most of the features required in this process. Physical timing closure became more important with submicrometre technologies, as more and more steps of the design flow had to be made timing-aware. Previously only logic synthesis had to satisfy timing requirements. With present deep submicrometre technologies it is unthinkable to perform any of the design steps of placement, clock-tree synthesis and routing without timing constraints. Logic synthesis with these technologies is becoming less important. It is still required, as it provides the initial netlist of gates for the placement step, but the timing requirements do not need to be strictly satisfied any more. When a physical representation of the circuit is available, the modifications required to achieve timing closure are carried out by using more accurate estimations of the delays.

What Physical verification ?
Physical verification of the design, involves DRC(Design rule check), LVS(Layout versus schematic) Check, XOR Checks, ERC (Electrical Rule Check) and Antenna Checks.
XOR Check

This step involves comparing two layout databases/GDS by XOR operation of the layout geometries. This check results a database which has all the mismatching geometries in both the layouts. This check is typically run after a metal spin, where in the re-spin database/GDS is compared with the previously taped out database/GDS.
Antenna Check

Antenna checks are used to limit the damage of the thin gate oxide during the manufacturing process due to charge accumulation on the interconnect layers (metal, polysilicon) during certain fabrication steps like Plasma etching, which creates highly ionized matter to etch. The antenna basically is a metal interconnect, i.e., a conductor like polysilicon or metal, that is not electrically connected to silicon or grounded, during the processing steps of the wafer. If the connection to silicon does not exist, charges may build up on the interconnect to the point at which rapid discharge does take place and permanent physical damage results to thin transistor gate oxide. This rapid and destructive phenomenon is known as the antenna effect. The Antenna ratio is defined as the ratio between the physical area of the conductors making up the antenna to the total gate oxide area to which the antenna is electrically connected.
ERC (Electrical rule check)

ERC (Electrical rule check) involves checking a design for all well and substrate areas for proper contacts and spacings thereby ensuring correct power and ground connections. ERC steps can also involve checks for unconnected inputs or shorted outputs.

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.

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 Different Types of IC packaging ?
IC are packaged in many types they are: * BGA1
* BGA2
* Ball grid array
* CPGA
* Ceramic ball grid array
* Cerquad
* DIP-8
* Die attachment
* Dual Flat No Lead
* Dual in-line package
* Flat pack
* Flip chip
* Flip-chip pin grid array
* HVQFN
* LQFP
* Land grid array
* Leadless chip carrier
* Low insertion force
* Micro FCBGA

* Micro Leadframe Package
* MicroLeadFrame
* Mini-Cartridge
* Multi-Chip Module
* OPGA
* PQFP
* Package on package
* Pin grid array
* Plastic leaded chip carrier
* QFN
* QFP
* Quadruple in-line package
* ROM cartridge
* Shrink Small-Outline Package
* Single in-line package
* Small-Outline Integrated Circuit
* Staggered Pin Grid Array
* Surface-mount technology
* TO220
* TO3
* TO92
* TQFP
* TSSOP
* Thin small-outline package
* Through-hole technology
* UICC
* Zig-zag in-line package

What is Substrate coupling ?
In an integrated circuit, a signal can couple from one node to another via the substrate. This phenomenon is referred to as substrate coupling or substrate noise coupling.
The push for reduced cost, more compact circuit boards, and added customer features has provided incentives for the inclusion of analog functions on primarily digital MOS integrated circuits (ICs) forming mixed-signal ICs. In these systems, the speed of digital circuits is constantly increasing, chips are becoming more densely packed, interconnect layers are added, and analog resolution is increased. In addition, recent increase in wireless applications and its growing market are introducing a new set of aggressive design goals for realizing mixed-signal systems. Here, the designer integrates radio frequency (RF) analog and base band digital circuitry on a single chip. The goal is to make single-chip radio frequency integrated circuits (RFICs) on silicon, where all the blocks are fabricated on the same chip. One of the advantages of this integration is low power dissipation for portability due to a reduction in the number of package pins and associated bond wire capacitance. Another reason that an integrated solution offers lower power consumption is that routing high-frequency signals off-chip often requires a 50O impedance match, which can result in higher power dissipation. Other advantages include improved high-frequency performance due to reduced package interconnect parasitics, higher system

reliability, smaller package count, smaller package interconnect parasitics, and higher integration of RF components with VLSI-compatible digital circuits. In fact, the single-chip transceiver is now a reality.

What is Latchup ?

A latchup is the inadvertent creation of a low-impedance path between the power supply rails of an electronic component, triggering a parasitic structure, which then acts as a short circuit, disrupting proper functioning of the part and possibly even leading to its destruction due to overcurrent. A power cycle is required to correct this situation. The parasitic structure is usually equivalent to a thyristor (or SCR), a PNPN structure which acts as a PNP and an NPN transistor stacked next to each other. During a latchup when one of the transistors is conducting, the other one begins conducting too. They both keep each other in saturation for as long as the structure is forward-biased and some current flows through it - which usually means until a power-down. The SCR parasitic structure is formed as a part of the totem-pole PMOS and NMOS transistor pair on the output drivers of the gates.