**What are standard Cell's?**  
Any time you start with a HDL description of the logic and work your way down you are following this kind of approach. The advantages of this approach are that you can describe logic in words without committing to a specific circuit. For instance A <= B+C, would define and adder, but you have not committed to a ripple carry of carry look ahead architecture. You can optimize your adding at a higher level without having to do an adder circuit by hand. This approach along with the use of standard cell library which has an estimate of the delay and power for various circuits that are already laid out can be used to automatically generate layouts from a Verilog or VHDL net-list.

**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

**What is RTL?**

In RTL design a circuit is described as a set of registers and a set of transfer functions describing the flow of data between the registers, (i.e. FSM + Data path).

RTL is an acronym for *register transfer level*. This implies that your Verilog code describes how data is transformed as it is passed from register to register. The transforming of the data is performed by the combinational logic that exists between the registers. RTL code also applies to pure combinational logic - you don't have to use registers. To show you what we mean by RTL code, let's consider a simple example.

module AOI (input A, B, C, D, output F);

assign F = ~((A & B) | (C & D));

endmodule

Yes! The AOI gate that we have used as an example so far has actually been written in RTL form. This means that continuous assignments are a valid way of describing designs for input to RTL synthesis tools. What other code techniques can we use? How about:

module MUX2 (input SEL, A, B, output F);

input SEL, A, B;

output F;

INV G1 (SEL, SELB);

AOI G2 (SELB, A, SEL, B, FB);

INV G3 (.A(FB), .F(F));

endmodule

**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.

**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

Vitex 5: Maximum clock rate: 550 MHz.  
  
**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.

**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). Mode selecting pins can be set to select the mode, refer data sheet for further details.

**What is gate count of your project?**Well mine was 3.2 million, I don’t know yours.

**What is IP?**

Hard macro, firm macro and soft macro are all known as IP (Intellectual property). They are optimized for power, area and performance. They can be purchased and used in your ASIC or FPGA design implementation flow. Soft macro is flexible for all type of ASIC implementation. Hard macro can be used in pure ASIC design flow, not in FPGA flow. Before buying any IP it is very important to evaluate its advantages and disadvantages over each other, hardware compatibility such as I/O standards with your design blocks, reusability for other designs.

**Soft macros**

Soft macros are in synthesizable RTL. Soft macros are more flexible than firm or hard macros.

Soft macros are not specific to any manufacturing process. Soft macros have the disadvantage of being somewhat unpredictable in terms of performance, timing, area, or power. Soft macros carry greater IP protection risks because RTL source code is more portable and therefore, less easily protected than either a netlist or physical layout data.

From the physical design perspective, soft macro is any cell that has been placed and routed in a placement and routing tool such as Astro. (This is the definition given in Astro Rail user manual)

Soft macros are editable and can contain standard cells, hard macros, or other soft macros.

**Firm macros**

Firm macros are in netlist format. Firm macros are optimized for performance/area/power using a specific fabrication technology. Firm macros are more flexible and portable than hard macros.

Firm macros are predictive of performance and area than soft macros.

**Hard macro**

Hard macros are generally in the form of hardware IP. Hard macros are targeted for specific IC manufacturing technology. Hard macros are block level designs which are silicon tested and proved. Hard macros have been optimized for power or area or timing. In physical design you can only access pins of hard macros unlike soft macros which allow us to manipulate in different way. You have freedom to move, rotate, flip but you can't touch anything inside hard macros. Very common example of hard macro is memory. It can be any design which carries dedicated single functionality (in general). For example it can be an MP4 decoder. Be aware of features and characteristics of hard macro before you use it in your design... other than power, timing and area you also should know pin properties like sync pin, I/O standards etc.

LEF, GDS2 file format allows easy usage of macros in different tools.

From the physical design (backend) perspective:

Hard macro is a block that is generated in a methodology other than place and route (i.e. using full custom design methodology) and is brought into the physical design database (eg. Milky way in Synopsys; Volcano in Magma) as a GDS2 file.

**What does synthesis mean?**

Synthesis is the stage in the design flow which is concerned with translating your Verilog code into gates. 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 synthesized that represents the chip which can be fabricated through an ASIC or FPGA vendor.

**Difference between Block RAM and Distributed RAM in FPGA**

In Xilinx FPGAs, a Block RAM is a dedicated two-port memory containing several kilobits of RAM. The FPGA contains several (or many) of these blocks.  
Inside each of small logic block is a configurable lookup table. It is normally used for logic functions, but you can reconfigure it as a few bits of RAM. You can combine several (or many) of them into a larger RAM. This is distributed RAM.  
Both types of RAM can be initialized with data, or used as ROM.

CLB

CLB

CLB

CLB

CLB

CLB

CLB

CLB

CLB

I/O pads

FPGA structure

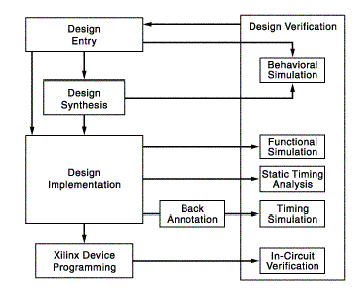
LUT

FF

Configurable Logic Block (CLB)

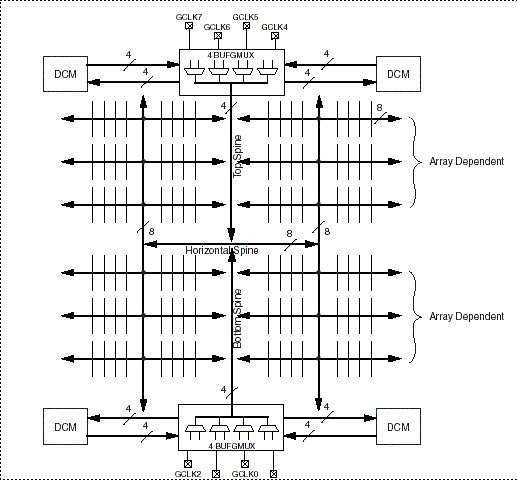
Programmable switches

**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 from global buffer can handle more loads.   
  
**FPGA design flow?**

**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 (extension) 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.

Quartus can use multiple constraint files (just like ISE). The main Quartus file is the ".QSF" file. This file contains project settings, pin assignments, etc. For timing analysis, Quartus projects using the TimeQuest timing analyzer use an ".SDC" (Synopsys design constraints) file.  
  
**Draw a rough diagram of how clock is routed through out FPGA?**

**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.

**What does synthesis mean?**

Synthesis is the stage in the design flow which is concerned with translating your Verilog code into gates. 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 synthesized that represents the chip which can be fabricated through an ASIC or FPGA vendor.

**How to achieve 180 degree exact phase shift?**

Never tell using inverter

a) dcm’s (digital clock manager) 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.

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.

There are 8 of them in xc3s5000