What do you mean by HDLs ?

Hardware description language or **HDL** is any language from a class of computer languages and/or programming languages for formal description of electronic circuits, and more specifically, digital logic. It can describe the circuit's operation, its design and organization, and tests to verify its operation by means of simulation. HDLs are standard text-based expressions of the spatial and temporal structure and behaviour of electronic systems. Like concurrent programming languages, HDL syntax and semantics includes explicit notations for expressing concurrency. However, in contrast to most software programming languages,HDLs also include an explicit notion of time, which is a primary attribute of hardware Languages whose only characteristic is to express circuit connectivity between a hierarchy of blocks are properly classified as netlist languages used on electric computer-aided design .VHDL and VERILOG are the two most widely used Hardware description languages .

What is VLSI Design ?
VLSI Design stands for Very Large scale Integrated circuit design.
Very-large-scale integration (**VLSI**) is the process of creating integrated circuits by combining thousands of transistor-based circuits into a single chip VLSI which involves the packing of more and more logic devices into smaller and smaller areas.

What is VHDL ? What are capabilities of VHDL ?
VHDL is a programming language, much like C++, it has its own syntax and semantics. The big difference from traditional programing languages is that instead of describing instructions which a processor will execute, it describes how circuits should be organized.VHDL is basically a programming language used to model digital Systems. As it is emulating real hardware it is inherently parallel and also treats timing as important. This language is a commonly used in the design of field-programmable gate arrays(FPGA)and application specific integrated circuits(ASIC). VHDL stands for Very High Speed Integrated Circuit Hardware Description language.VHDL is composed of language building blocks that consist of more than **75 *reserved words*** and about **200 *descriptive words*** or ***word combinations***.

What can be the various uses of VHDL ?

The VHDL language can be used for several goals like -
i) To synthesize digital circuits

ii) To verify and validate digital designs
iii) To generate test vectors to test circuits
iv) To simulate circuits

What are the Various levels of abstractions in VLSI design?
Abstraction is defined as the hiding of information that is too detailed. It is therefore necessary to diffrentiate between essential and non-essential information. Information that is not important for the current view of the problem will be left out from the description
The Various levels of abstraction in VLSI design are :-
1. Behaviour level
2. RTL (Register Transfer level)
3. Logic level
4. Layout (Transistor level)

In the behaviour level, complete systems can be modelled. Bus systems or complex algorithms are described without considering synthesizability. The stimuli for simulation of RTL models are described in the behaviour level, for example. Stimuli are signal values of the input ports of the model and are described in the testbench, sometimes called validation bench.

The designer has to take great care to find a consistent set of input stimuli that do not contradict the specification. The responses of the model have to be compared with the expected values which, in the simplest case, can be done with the help of a waveform diagram that shows the simulated signal values.

On the RT level, the system is described in terms of registers and logic that calculates the next value of the storage elements. It is possible to split the code into two blocks (cf. process statement) that contain either purely combinational logic or registers. The registers are connected to the clock signal and provide for synchronous behaviour. In practice, the strict separation of Flip Flops from combinational logic is often annulated and clocked processes describe the registers and the corresponding update functions.

The gate netlist is generated from the RT description with the help of a synthesis tool. For this task, a cell library for the target technology which holds the information about all available gates and their parameters (fan-in, fan-out, delay) is needed.

Based upon this gate netlist the circuit layout is generated. The resulting wire lengths can be converted into propagation delays which can be fed back into the gate level model (back annotation). This allows for thorough timing simulations without the need for additional simulator software.

What is Synthesis?

Synthesis represents the transformation of an abstract description into a more detailed descrition. In general, the term "synthesis" is used for the automated transformation of RT level descriptions into gate level representations.
This transformation is mainly influenced by the set of basic cells that is available in the target technology. While simple operations like comparisons and either/or decisions are easily mapped to boolean functions, more complex constructs like mathematical operators are mapped to a tool specific macro cell library first.
This means that a number of adder, multiplier, etc. architectures are known to the synthesis tool and these designs are treated as if they were designed by the user.

What is the difference between Entity and Architecture ?
Entity
The interface between a module and its environment is described within the entity declaration which is initiated by the keyword ' entity '. It is followed by a user-defined descriptive name. The interface description is placed between the keyword ' is ' and the termination of the entity statement which consists of the keyword ' end ' and the name of the entity. The input, output and bi-directional ports are defined in the entity.
In the new VHDL'93 standard the keyword ' entity ' may be repeated after the keyword ' end ' for consistency reasons.

Architecture
The architecture contains the implementation for an entity which may be either a behavioural description (behavioural level or, if synthesizable, RT level) or a structural netlist or a mixture of those alternatives..

An architecture is strictly linked to a certain entity. An entity, however, may have several architectures underneath, e.g. different implementations of the same algorithm or different abstraction levels. Architectures of the same entity have to be named differently in order to be distinguishable. The name is placed after the keyword ' architecture ' which initiates an architecture statement. 'RTL' was chosen in this case.

It is followed by the keyword ' of ' and the name of entity that is used as interface ('HALFADDER'). The architecture header is terminated by the keyword ' is ', like in entity statements. In this case, however, the keyword ' begin ' must be placed somewhere before the statement is terminated. This is done the same way as in entity statements: The keyword ' end ', followed by the architecture name. Once again, the keyword ' architecture ' may be repeated after the keyword ' end ' in VHDL'93.

Explain Various types of Modelling styles ?
The Various modelling styles are :-
Structural, behavioural, dataflow and mixed style.
*Structural Description Method**:*** It expresses the design as an arrangement of interconnected components. It is basically the representation of the schematic in VHDL Language form.

Behavioral Description Method: describes the functional behavior of a hardware design in terms of circuits and signal responses to various stimuli. A Behavioral Description uses a small number of processes where each process performs a number of sequential signal assignments to multiple signals.
The hardware behavior is described algorithmically and this modelling style is the most frequently used and the best way to model any algorithm.
The advantage of models at this level is that models for simulation can be built quickly.

Data-Flow Description Method: is similar to a register-transfer language
This method describes the function of a design by defining the flow of information from one input or register to another register or output.
Data-Flow Description uses a large number of concurrent signal assignment statements. A concurrent statement executes asynchronously with respect to other concurrent statements.

The concurrent statements used in data flow description include:-

- block statement (used to group one or more concurrent statements)
- concurrent procedure call- concurrent assertion statement
- concurrent signal assignment statement

Explain various types of delays in VHDL ?

The Various types of delays in VHDL are :-

1. *Delta delay* - In VHDL simulations, all signal assignments occur with some infinitesimal delay, known as **delta delay**. VHDL uses the concept of delta delay to keep track of processes that should occur at a given time step, but are actually evaluated in different machine cycles .A delta delay is a unit of time as far as the simulator hardware is concerned, but in the simulation itself time has no advance. Technically, delta delay is of no measurable unit, but from a hardware design perspective one should think of delta delay as being the smallest time unit one could measure, such as a femtosecond(fs).

2. *Inertial delay* - The inertial delay causes the pulses less than specified delay to get suppressed & will not propogate these pulses to change the output.
The inertial delay model is specified by adding an after clause to the signal assignment statement. Inertial delay is basically a default delay, i.e it's a component delay.

3. *Transport delay -* Tranport delay adds the propogation delay to the signal. The transport delay model just delays the change in the output by the time specified in the after clause.
Transport delay basically represents a wire delay.

```
e.g. q <=transport a nor b after 1ns ;
```

What are Generics ?

Generics are a way to provide static information to the VHDL program. Immediately after writing entity name, we will mention the generics, this generics will provide the data for entire program.
Generics basically allow a design entity to be described so that,for each use of that component,its structure and behavior can be changed by generic values.In general they are used to construct parameterized hardware components.Generics can be of any type**.**but mostly we will give the timing details there.

E.g. :- generic (width : integer := 7);

Generic is a great asset when you use your design at many places with slight change in the register sizes,input sizes etc. But if the design is very unique then,you need not have generic parameters. Also, Generic's are synthesizable***.***

What is the difference between STD_LOGIC and BIT types?

BIT has 2 values: '0' and '1'.

STD_LOGIC is defined in the library std_logic_1164.This is a nine valued logic system.
It has 9 values: 'U', 'X', '0', '1', 'Z', 'W', 'L' ,'H' and '-'.
The meaning of each of these characters are:
U = uninitialized
X = unknown - a multisource line is driven '0' and '1' simultaneously (*)
0 = logic 0
1 = logic 1
Z = high impedance (tri state)
W = weak unknown
L = weak "0"
H = weak "1"
- = dont care

Type std_logic is unresolved type because of 'U','Z' etc. It is illegal to have a multi-source signal in VHDL. So use 'bit' logic only when the signals in the design doesn't have multi sources. If you are unsure about this then declare the signals as std_logic or std_logic_vector,because then you will be able to get errors in the compilation stage itself. But many of the operators such as shift operators cannot be used on 'std_logic_vector' type.So you may need to convert them to bit_vector before using shift operations.
One example is given below:

example of how to shift a std_logic signal : right shifting logically by 2 bits.
Here, count is std_logic_vector.
output <= To_StdLogicVector(to_bitvector(count) srl 2); to_bitvector converts Std_Logic_Vector to bit_vector. To_StdLogicVector converts bit_vector to Std_Logic_Vec

What is the difference between Concurrent & Sequential Statements ?

Concurrent statements define interconnected processes and blocks that together describe a design's overall behavior or structure. They can be grouped using block statement. Groups of blocks can also be partitioned into other blocks.
At the same level, a VHDL component can be connected to define signals within the blocks It is a reference to an entity A process can be a single signal assignment statement or a series of sequential statements (SS)
Within a process, procedures and functions can partition the sequential statements

Discuss process and wait statements? Can they be used simultaneously
in the program ?
-- INCOMPLETE ANSWER
The Various features of the process statement are :-

- It contains sequentially executed statements
- It can exist within an architecture only
- Several processes run concurrently
- Execution is controlled either via
    - sensitivity list (contains trigger signals), or
    - wait-statements
- The process label is optional

Because the statements within an architecture operate concurrently, therefore another VHDL construct is necessary to achieve sequential behaviour. A process, as a whole, is treated concurrently like any other statement in an architecture and contains statements that are executed one after another like in conventional programming languages. In fact
it is possible to use the process statement as the only concurrent VHDL statement.
The execution of a process is triggered by events. Either the possible event sources are listed in the sensitivity list or explicit wait statements are used to control the flow of execution.

These two options are mutually exclusive, i.e. no wait statements are allowed in a process with sensitivity list. While the sensitivity list is usually ignored by synthesis tools, a VHDL simulator will invoke the process code whenever the value of at least one of the listed signals changes. Consequently, all signals that are read in a purely combinational process, i.e. that influence the behaviour, have to be mentioned in the sensitivity list if the simulation is to produce the same results as the synthesized hardware. Of course the same is true for clocked processes, yet new register values are to be calculated with every active clock edge only. Therefore the sensitivity list contains the clock signal and asynchronous control signals (e.g. reset).

A process statement starts with an optional label and a ':' symbol, followed by the ' **process** ' keyword. The sensitivity list is also optional and is enclosed in a '(' ')' pair. Similar to the architecture statement, a declarative part exists between the header code and the keyword ' **begin** '. The sequential statements are enclosed between ' **begin** ' and ' **end process** '. The keyword ' **process** ' has to be repeated! If a label was chosen for the process, it has to be repeated in the end statement, as well.

What is the difference between Signal and the Variable ?
Signals are interpreted as wires or wires with memory (i.e., FFs, latches etc.)

Signal are declared as :-
signal signal_name, signal_name, … : data_type

Signal assignment :-
signal_name <= projected_waveform;

The Concept of variables is found in traditional programming languages, in which a name represents a symbolic memory location where a value can be stored and modified. There is NO direct mapping between a variable and a hardware component. Variables can be declared and used only inside a process.

Variable declaration:
variable variable_name, … : data_type

Variable assignment:
variable_name := value_expression;
Variables contains no timing information (immediate assignment) i.e. no waveform is possible for variables.
Both signals and variables can be assigned initial values.
Although useful in simulations, synthesis canNOT deal with them

What are VHDL Subtypes ?
VHDL subtypes are used to constrain defined types. Constraints take the form of range constraints or index constraints. However, a subtype may include the entire range of the base type. Assignments made to objects that are out of the subtype range generate an error at run time. The syntax and an example of a subtype declaration is shown below :-

```
SUBTYPE First_ten IS INTEGER RANGE 0 to 9;
```

Explain Resolution Function ?
A resolution function defines how values from multiple sources, multiple drivers are resolved into a single value. The signals in VHDL can have multiple drivers.
The value of the signal is a function of all the drivers of that signal. The following figure shows an example of a bus signal which is driven by four independent signals. The value bus is computed by a bus resolution function (brf in this example).

A resolution function must be a pure function that has a single input parameter of class **constant** that is a one dimensional unconstrained array of the type of the resolved signal.

The drivers are labeled **s1,s2,s3,** and **s4**. The value of the signal **dbus** is computed by a bus resolution function (**brf** in our example). Bus resolution functions are user-defined and are evaluated when one of the drivers of the signal receives a new value (event). The 'resolved' value is then generated by the bus resolution function

Write short notes on Case statements ?
All branches are equal in priority when using a CASE statement. Therefore it is obvious that there must not be any overlaps among cases or choices and all possible values of the CASE EXPRESSION must be covered. For covering all remaining, i.e. not yet covered, cases, the keyword ' others ' may be used.

The type of the EXPRESSION in the head of the CASE statement has to match the type of the query values. Single values of EXPRESSION can be grouped together with the '|' symbol, if the consecutive action is the same. Value ranges allow to cover even more choice options with relatively simple VHDL code.

```
case EXPRESSION is
when VALUE_1 =>
```

```
-- sequential statements
when VALUE_2 | VALUE_3 =>
-- sequential statements
when VALUE_4 to VALUE_N =>
-- sequential statements
when others =>
-- sequential statements
end case ;
```

Write short note on Loop statement and Next statement ?

Three kinds of iteration statements.

```
[ label: ] loop
sequence-of-statements -- use exit statement to get out
end loop [ label ] ;

[ label: ] for variable in range loop
sequence-of-statements
end loop [ label ] ;

[ label: ] while condition loop
sequence-of-statements
end loop [ label ] ;

loop
input_something;
exit when end_file;
end loop;

for I in 1 to 10 loop
AA(I) := 0;
end loop;

while not end_file loop
input_something;
end loop;
```

all kinds of the loops may contain the 'next' and 'exit' statements.

--

A statement that may be used in a loop to cause the next iteration.

```
[ label: ] next [ label2 ] [ when condition ] ;
```

```
next;
next outer_loop;
next when A>B;
next this_loop when C=D or done; -- done is a Boolean variable
```

Discuss the Difference between array and records types ?

ARRAYS :-VHDL composite types consists of arrays and records. Each object of
this data type can hold more than one value***.*** Arrays consist of many similar elements of any data type, including arrays. The array is declared in a TYPE statement. There are numerous items in an array declaration. The first item is the name of the array. Second, the range of the array is declared. The keywords TO and DOWNTO designate ascending or descending indices, respectively, within the specified range. The third item in the array declaration is the specification of the data type in each element of the array.

E.g :- TYPE data_bus IS ARRAY (0 to 31) OF BIT ;
TYPE reg_type IS ARRAY (15 downto 0) of BIT ;

RECORDS :-

The second VHDL composite type is the record. Records are used to group elements of different types into a single VHDL object. An object of type record may contain elements of different types. Again, a record element may be of any data type, including another record. A TYPE declaration is used to define a record.
Note that the types of a record's elements must be defined before the record is defined. Also notice that there is no semi-colon after the word RECORD. The RECORD and END RECORD keywords bracket the field names. After the RECORD keyword, the record's field names are assigned and their data types are specified.

E.g :- TYPE Switch_info IS
Record
status : binary ;

```
Idnumber : integer ;
END Record;

switch.status = ON;
switch.IDnumber = 30;
```

In the above example, a record type, switch_info, is declared. This example makes use of the binary enumerated type declared previously. Note that values are assigned to record elements by use of the field name.