Advancements in Electrical and Electronics Technology
Received: Nov 23, 2015, Accepted: Dec 17, 2015, Published: Dec 19, 2015
Adv Electr Electron Technol, Volume 1, Issue 1
http://crescopublications.org/pdf/AEET/AEET-1-R005.pdf
Article Number: AEET-1-R005

Short Communication Open Access

Introduction to VLSI

M Anusha and Sai Priya M

ECE Department, RVR Institute of Engineering and Technology, Hyderabad, Telangana, India

Citation: M Anusha and Sai Priya M (2015) Introduction to VLSI. Adv Electr Electron Technol 1: R005.

The first digital circuit was designed by using electronic components like vacuum tubes and transistors. Later Integrated Circuits (ICs) were invented, where a designer can be able to place digital circuits on a chip consists of less than 10 gates for an IC called SSI (Small Scale Integration) scale. With the advent of new fabrication techniques designer can place more than 100 gates on an IC called MSI (Medium Scale Integration). Using design at this level, one can create digital sub blocks (adders, multiplexes, counters, registers, and etc.) on an IC. This level is LSI (Large Scale Integration), using this scale of integration people succeeded to make digital subsystems (Microprocessor, I/O peripheral devices and etc.) on a chip.

At this point design process started getting very complicated. i.e., manually conversion from schematic level to gate level or gate level to layout level was becoming somewhat lengthy process and verifying the functionality of digital circuits at various levels became critical. This created new challenges to digital designers as well as circuit designers. Designers felt need to automate these processes. In this process, rapid advance in Software Technology and development of new higher level programming languages takes place. People could able to develop CAD/CAE (Computer Aided Design/Computer Aided Engineering) tools, for design electronics circuits with assistance of software programs. Functional verification and Logic verification of design can be done using CAD simulation tools with greater efficiency. It became very easy to a designer to verify functionality of design at various levels.

With advent of new technology, i.e., CMOS (Complementary Metal Oxide Semiconductor) process technology. One can fabricate a chip contains more than Million of gates. At this point design process still became critical, because of manual converting the design from one level to other. Using latest CAD tools could solve the problem. Existence of logic synthesis tools design engineer can easily translate to higher-level design description to lower levels. This way of designing (using CAD tools) is certainly a revolution in electronic industry. This may be leading to development of sophisticated electronic products for both consumer as well as business.

Designing Systems using Hardware always gives best results when compared to software (like Speed Reliability, performance and etc.,) Using CMOS VLSI Design methodology designer could design and fabricate ICs without spending much time when compared to traditional way of designing.

2.1 IC Design Flow

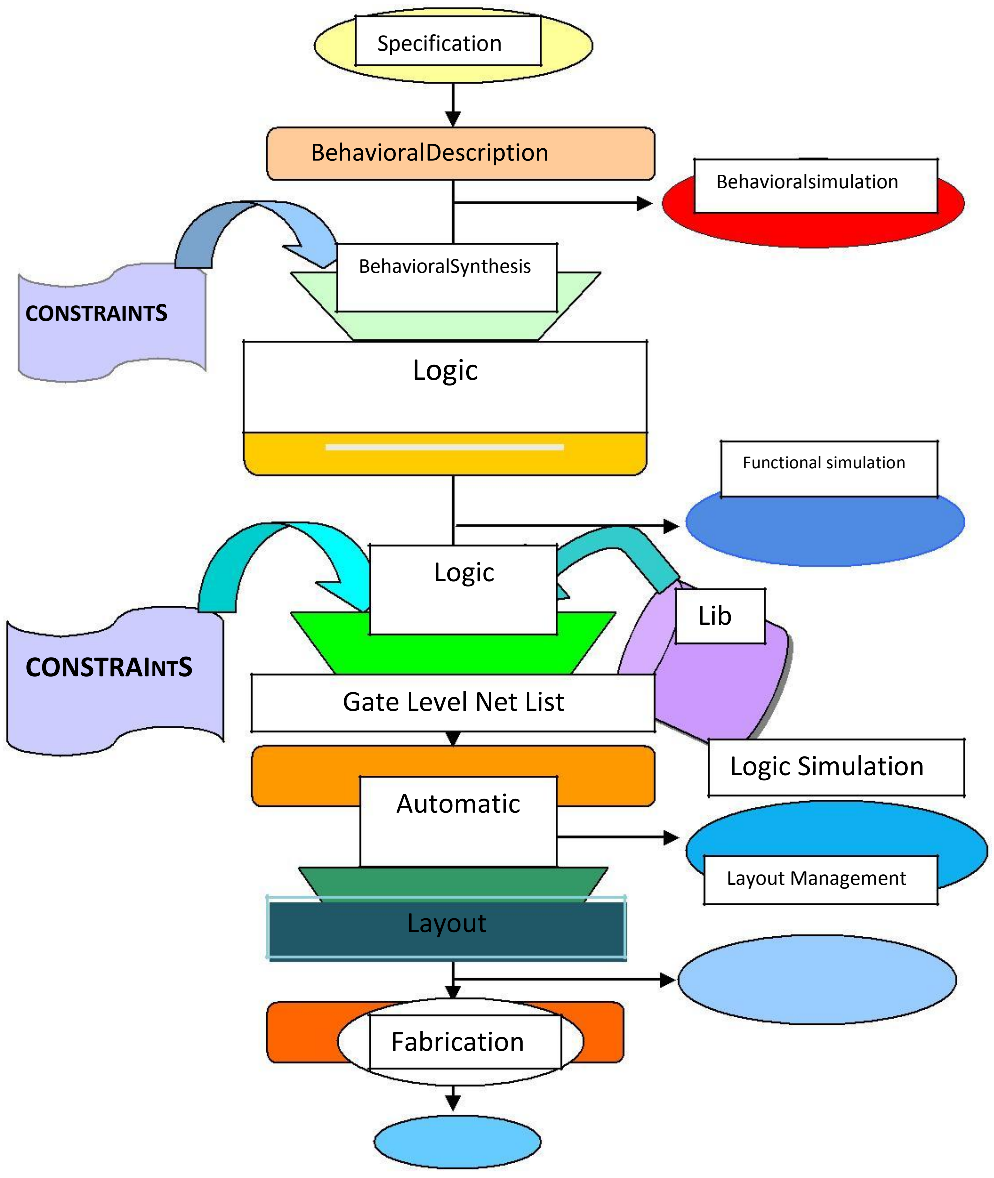

Figure 1: VLSI design flow

Micron Technology

The micron technology can be classified into 4 categories, Evolving from micron technology and extending up to VDSM.

- ❖ Micron Technology: The technology up to $10^{-6}\mu$m is the micron Technology.
- ❖ Submicron Technology: The technology below 1um is known as the Submicron technology. It generally ranges up to 0.36μm.
- ❖ DSM (Deep Sub Micron technology): The technology extending up to 0.18μm is DSM.
- ❖ VDSM (Very Deep Sub Micron technology): The presently used technology is **V**DSM. It ranges up to 0.09um.

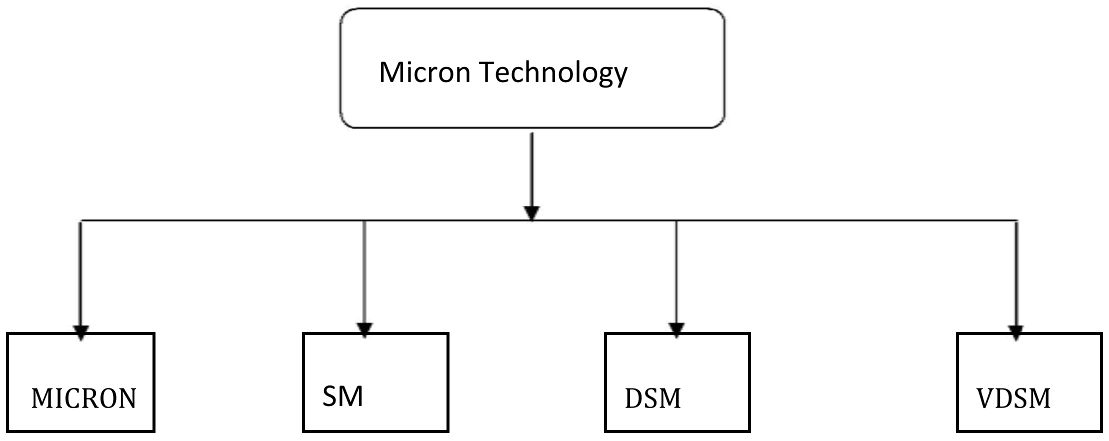

Figure 2: Micron technology

2.2 Introduction to VHDL

VHDL is acronym for VHSIC hardware Description language. VHSIC is acronym for very high speed Integrated Circuits. It is a hardware description language that can be used to model a digital system at many levels of abstraction, ranging from the algorithmic level to the gate level.

The VHDL language can be regarded as an integrated amalgamation of the following languages:

- Sequential language
- Concurrent language
- Net-list language
- Timing specifications

2.2.1 Waveform generation language ➔VHDL

This language not only defines the syntax but also defines very clear simulation semantics for each language construct. Therefore, models written in this language can be verified using a VHDL simulator. This subset is usually sufficient to model most applications .The complete language, however, has sufficient power to capture the descriptions of the most complex chips to a complete electronic system.

- **History**

The requirements for the language were first generated in 1988 under the VHSIC chips for the department of Defense (DOD). Reprocurement and reuse was also a big issue. Thus, a need for a standardized hardware description language for the design, documentation, and verification of the digital systems was generated. The IEEE in the December 1987 standardized VHDL language; this version of the language is known as the IEEE STD 1076-1987. The official language description appears in the IEEE standard VHDL language Reference manual, available from IEEE. The language has also been recognized as an American National Standards Institute (ANSI) standard.

According to IEEE rules, an IEEE standard has to be reballoted every 5 years so that it may remain a standard so that it may remain a standard. Consequently, the language was upgraded with new features, the syntax of many constructs was made more uniform, and many ambiguities present in the 1987 version of the language were resolved. This new version of the language is known as the IEEE STD 1076-1993.

- **Capabilities**

The following are the major capabilities that the language provides along with the features that the language provides along with the features that differentiate it from other hardware languages.

- ❖ The language can be used as exchange medium between chip vendors and CAD tool users. Different chip vendors can provide VHDL descriptions of their components to system designers.
- ❖ The language can be used as a communication medium between different CAD and CAE tools
- ❖ The language supports hierarchy; that is a digital can be modeled as asset of interconnected components; each component, in turn, can be modeled as a set of interconnected subcomponents.
- ❖ The language supports flexible design methodologies: top-down, bottom-up, or mixed. It supports both synchronous and asynchronous timing models.
- ❖ Various digital modeling techniques, such as finite –state machine descriptions, and Boolean equations, can be modeled using the language.
- ❖ The language is publicly available, human-readable, and machine-readable.
- ❖ The language supports three basic different styles: Structural, Dataflow, and behavioral.
- ❖ It supports a wide range of abstraction levels ranging from abstract behavioral descriptions to very precise gate-level descriptions.
- ❖ Arbitrarily large designs can be modeled using the language, and there are no limitations imposed by the language on the size of the design.

2.2.2 Hardware Abstraction

VHDL is used to describe a model for a digital hardware device. This model specifies the external view of the device and one or more internal views. The internal view of the device specifies functionality or structure, while the external view specifies the interface of the device through which it communicates with the other modules in the environment.

In VHDL each device model is treated as a distinct representation of a unique device, called an Entity. The Entity is thus a hardware abstraction of the actual hardware device. Each Entity is described using one model, which contains one external view and one or more internal views.

2.2.3 Basic terminology

VHDL is a hardware description language that can be used to model a digital system. A hardware abstraction of this digital system is called an entity. An entity X, when used in another entity Y, becomes a component for the entity Y.
To describe an entity, VHDL provides five different types of primary constructs, called design units. They are:

1. Entity declaration
2. Architecture body
3. Configuration declaration
4. Package declaration
5. Package body

1. An entity is modeled using an entity declaration and at least one architecture body the Entity declaration describes the external view of the entity;

   For example: the input and output signal names.
2. The architecture body contains the internal description of the entity; for example, as a set of interconnected components that represents the structure of the entity, or a set of concurrent or sequential statements that represents the behavior of the entity.
3. A configuration declaration is used to create a configuration for an entity. It specifies the binding of one architecture body from the many architecture bodies that may be associated with the entity .It may also specify the bindings of the architecture components used in the selected architecture body to other entities. An entity may have any number of configurations.
4. A package declaration encapsulates a set of related declarations, such type of declaration s, subtype declaration and subprogram declaration, which can be shared across two or more design units.

5. A package body contains the definition of subprogram declared in a package declaration.

Once an entity has been modeled, it needs to be validated by a VHDL system. A typical VHDL system consists of an analyzer and a simulator. The analyzer reads in one or more design units contained in a single file and compiles them into a design library after validating the syntax and performing some static checks.

The language is case insensitive; that is lowercase and uppercase characters are treated alike the Language is also free format comments are specified in the language by preceding the text with two Consecutive dashes (- -).

2.2.4 Entity Declaration

The entity declaration specifies the name of entity being modeled and lists the set of inter face ports. Ports are signals through which entity communicates with other models in its external environment.

Example:

Entity declaration for the half adder circuit is

```
Entity half adder is Port (A, B: in Bit; sum, carry: out Bit);
End half adder;
```

The entity called half adder has two input ports, A and B and two output ports sum and carry Bit is predefined type of the language.

2.3 Architecture Body

An architecture body using any of the following modeling styles specifies the internal details of an entity.

1. As a set of interconnected components (to represent structure)
2. As a set of concurrent assignment statements (to represent data flow)
3. As a set of sequential assignment statements (to represent behavior)
4. As any combination of the above three.

2.3.1 Structural style of modeling

In this one an entity is described as a set of interconnected components. Such a model for the HALF_ADDER entity, is described in a n architecture body

```
Architecture ha of ha is
Component Xor2
Port (X, Y: in BIT; Z:out BIT);
End component;
Component And2
            Port (L, M: in BIT; N:outBIT);
End component;
                    Begin
X1: Xor2portmap (A, B, SUM)
                A1: AND2portmap (A, B, CARRY);
                    End ha;
```

The name of the architecture body is ha .the entity declaration for half adder specifies the interface ports for this architecture body. The architecture body is composed of two parts: the declaration part and the statement part. Two component declarations are present in the declarative part of the architecture body.

The declared components are instantiated in the statement part of the architecture body using component instantiation. The signals in the port map of a component instantiation and the port signals in the component declaration are associated by the position.

2.3.2 Dataflow Style of Modeling

In this modeling style, the flow of data through the entity is expressed primarily using concurrent signal assignment statements. The data flow model for the half adder is described using two concurrent signal assignment statements .In a signal assignment statement, the symbol <=implies an assignment of a value to a signal.

2.3.3 Behavioral Style of Modeling

The behavioral style of modeling specifies the behavior of an entity as a set of statements that are executed sequentially in the specific order. These sets of sequential statements, which are specified inside a process statement, do not explicitly specify the structure of the entity but merely its functionality. A process statement is a concurrent statement that can appear with in an architecture body.

2.3.4 Mixed Style of Modeling

It is possible to mix the three modeling styles in a single architecture body. That is, within an architecture body, we could use component instantiation statements, concurrent signal assignment statements and process statements.

2.3.5 Model Analysis

Once an entity is declared in VHDL, it can be validated using analyzer and a simulator that are a part of a VHDL system. The first step in the validation process is analysis. The analyzer takes a file that contains one or more design units and compile s them into an intermediate form. The generated intermediate form is stored in a specific design library that has been designated as the working library.

There is a design library with the logic name STD predefined by the VHDL language environment. This library contains two packages: STANDARD and TEXTIO. The STANDARD package contains declarations for all the predefined types of the language .The Text IO package contains procedures and functions that are necessary for supporting formatted text read and write operations. There also exists an IEEE standard package called STD_LOGIC_1164,and contains its associated sub types; overloaded operator functions,

2.4 Simulation

For a hierarchical entity to be simulated, all of its lowest –level components must be described at the behavioral level. A simulation can be performed on either one of the following:

1. An entity declaration and an architecture body pair.
2. A configuration

Preceding the actual simulation are two major steps:

1. Elaboration phase: IN this phase, the hierarchy of the entity is expanded and linked, components are bound to entities in a library, and the top-level entity is built as a network of behavioral models that is ready to be simulated.

2. Initialization phase: Driving and effective values for all explicitly declared signals are computed, implicit signals are assigned values, processes are executed once until they suspend, and simulation time is set to 0ns.

Simulation commences by advancing time to that of the next event. Values that are assigned to signals at this time are assigned. If the value of a signal changes, and if that signal is present in the sensitivity list of a process, the process is executed until it suspends. Simulation stops when an assertion occurs, depending on the implementation of the VHDL system or when the maximum time as defined by the language is reached.

2.4.1 Entity Declaration

An entity declaration describes the external interface of the entity. It specifies the name of the entity, the names of the interface ports, their mode and the type of ports .The syntax for entity declaration is:

```
    Entity entity _name is
      [generic (list of –generics and –their types);]
    [port (list of interface-port-names-and their types
);] [entity item declarations]
    [beginentity statements]
    end [entity][entity name];
```

The entity –name is the name of the entity, and the interface ports are the signals through which entity passes the information to and from its external environment. Each interface port can have one of the following modes:

1. **In**: The value of an input port can only read with in the entity model .
2. **Out**: The value of an output port can only be updated within the entity model.
3. **In out**: The value of a bi directional port can be read and updated within the entity model.
4. **Buffer**: The value of a buffer port can be read and updated within the entity model .It cannot have more than one source.

Declarations that are placed in the entity are common to all the design units that are associated with that entity declaration.

2.4.2 Architecture Body

An architecture body describes the internal view of an entity. It describes the functionality of the structure of the entity. Architecture <architecture name> of< entity name> is

```
        Begin
            Concurrent statements;
             Process statements;
             Block statements;
             Concurrent signal assignment-
             statement; Component –instantiation-
             statement; Generate statement;
       End [architecture] [architecture name];
```

The concurrent statements describe the internal composition of the entity. All concurrent statements are executed in parallel. The internal composition of an entity can be expressed in terms of structure, dataflow and sequential behavior.

Here we describe an entity by using the behavioral model. A process statement, which is a concurrent statement, is the primary mechanism used to describe the Functionality of an entity in this modeling style.

2.4.3 Process Statement

A process statement contains sequential statements that describe the functionality of a portion of an entity in sequential terms. The syntax for the process statement is:

```
[Process-label:] process [(sensitivity-list)] [is]
          begin
             sequential statements;
```

```
            variable-assignment-
            statement signal assignment-
            statement wait statement
            if-statement case-
           statement loop-
           statement null-
           statement exit-
           statement next-
           statement assertion-
           statement
            report-statement
           procedure-call-
           statement return
    end process [process label];
```

A set of signals to which the process is sensitive is defined by the sensitivity list. In other words, each time an event occurs on any of the signals in the sensitivity list, the sequential statements within the process are executed in a sequential order, that is in the order in which they appear. The process then suspends after executing the last sequential statement and waits for another event to occur on a signal in the sensitivity list.

2.4.4 Variable Assignment Statement

Variables can be declared and used inside a process statement. A variable is assigned a value using the variable assignment statement that typically has the form

```
Variable-object: = expression
```

The expression is evaluated when the statement is executed, and the computed value is assigned to the variable object instantaneously, that is, at the concurrent simulation time.

A variable can be declared outside of a process or subprogram. Such a variable can be read and updated by more than one process. These variables are called shared variables.

2.4.5 Signal Assignment Statement

Signals are assigned values using a signal assignment statement. The simplest form of a signal assignment statement is:

```
Signal-object <= expression [after a delay value];
```

A signal assignment statement can appear within a process or outside of a process. If it occurs outside of a process, it is considered to be a concurrent signal assignment statement.

When a signal assignment statement appears with in a process, it is considered to be a sequential signal assignment statement and is executed in sequences with respect to the other statements which appear with in the process.

2.4.6 Conditional Statements

IF Statement

An if statement selects a sequence of statements for execution of statements for execution based on the value of a condition .the condition .The condition can be any expression that evaluates to a Boolean value. The general form of an if statement is:

```
If Boolean expression then Sequential statements
    {else if Boolean-expression then Sequential-
    statements}
    [else sequential statements]
                end if;
```

If statement is executed by checking each condition sequentially until the first true condition is found; the set of sequential statements associated with this condition is executed. If statement is also called as sequential statement.

Case Statement

The format of a case statement is:

```
Case expression is
    When choices =>sequential statements
    When choices =>sequential statements
          End case;
```

The case statement selects one of the branches for the execution based on the value of the expression. The expression value must be of a discrete type or one-dimensional array type. Choices may be expressed as single values, as a range of **values by choosing "others". The other clause can be used as a choice to cover the "catch-all" values and, if present,** must be
the last branch in the case statement

Loop Statements

A loop statement is used to iterate through a set of sequential statements the syntax for loop statement is:

```
    [Loop-label:] iteration-scheme loop
    Sequential-statements
                     End loop [loop label];
```