|  |
| --- |
| **AN INTERNSHIP REPORT**  **submitted in partial fulfillment of the requirements for the Award of Degree of**  **BACHELOR OF TECHNOLOGY**  **in**  **ELECTRONICS AND COMMUNICATION ENGINEERING**  **on**  **VLSI SoC Design using Verilog HDL**  **in**  **MAVEN SILICON**  SUBMITTED BY: AMISH VERMA  (ROLL NO.- 2000910310033)  MENTOR: MR. SAMPATH KUMAR V.    **DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**  JSS ACADEMY OF TECHNICAL EDUCATION  C-20/1 SECTOR-62, NOIDA  2022- 2023 |

**CERTIFICATE**

****

**TABLE OF CONTENTS**

Page

INTERNSHIP CERTIFICATE 1

TABLE OF CONTENTS 2

ACKNOWLEDGEMENT 3

DECLARATION 4

ABSTRACT 5

LIST OF FIGURES 6

LIST OF TABLES 6

INTRODUCTION 7

ASIC vs FPGA 8

VLSI Design Flow 9

Verilog HDL 12

Verilog Basics 14

4x1 MUX Implementation using Verilog HDL 20

REFERENCES 24

**ACKNOWLEDGEMENT**

First of all, I would like to express my gratitude to Mr. Sivakumar PR, trainer of MAVEN SILICON for giving me the opportunity to do an internship within the organization and enabling me to complete this report. I am highly indebted to Principal Dr. Amarjeet Singh Sir, for the facilities provided to accomplish this internship, Head of the Department Dr. Arun Kumar G. Sir and Mr. Sampath Kumar V. Sir, internship guide Department of ECE for his kind and proper guidance in every phase of the project his supervision and guidance shaped this report to be completed perfectly.

Last but not the least my sincere gratitude to all people who knowingly or unknowingly supported me for completion of this internship.

Amish Verma

(2000910310033)

**DECLARATION**

I hereby declare that this internship report entitled “VLSI SoC Design using Verilog HDL” is submitted to JSS Academy of Technical Education, Noida in the Department of Electronics and Communication Engineering, is a record of my original work under the guidance Mr. Sampath Kumar V., my mentor, is a true replica of the training done under MAVEN SILICON in VLSI SoC Design. I also declare that no chapter of this manuscript is either in whole or in part is copied from any other document.

Amish Verma

Date: 06/12/2022

Student of B.Tech, Electronics and Communication Engineering

JSS Academy of technical Education, Noida

**ABSTRACT**

Very Large Scale Integration (VLSI) is the process of creating an integrated circuit (IC) by combining millions or billions of MOS transistors onto a single chip. Before the introduction of VLSI technology, most ICs had a limited set of functions they could perform. An electronic circuit might consist of a CPU, ROM, RAM and other glue logic. VLSI enables IC designers to add all of these into one chip.

“VLSI SoC Design Using Verilog HDL” is an online training program by Maven Silicon – a VLSI training company based in Bangalore, India. This training program is beginner-friendly and introduces students to VLSI design and usage of Hardware Description Languages (HDL), particularly Verilog HDL. The program is hands-on and provides complete tutorials of writing, compiling and simulating VLSI designs with Verilog in Intel’s Quartus Prime and ModelSim.

**LIST OF FIGURES**

| **FIGURE** | **PAGE** |
| --- | --- |
| 1.VLSI Design Flow | **9** |
| 2. (a) 4x1 MUX  (b) Truth Table | **20** |

**LIST OF TABLES**

| **TABLE** | **PAGE** |
| --- | --- |
| 1.ASIC vs FPGA | **8** |

**INTRODUCTION**

**VLSI** (Very Large Scale Integration)

Very-large-scale integration (VLSI) is the process of creating an integrated circuit (IC) by combining a very large number of transistors into a single chip. VLSI began in the 1970s when complex semiconductor and communication technologies were being developed. The microprocessor is a VLSI device.

Integrated Circuits are tiny electronic circuits that are used to perform a specific electronic function, such as amplification. The first Integrated Circuit (IC) was invented by Jack Kilby in 1958. As suggested by Moore, the capacity doubled roughly every 18 months. Today, a large single VLSI chip can contain over one billion transistors. These days, VLSI chiefly comprises Front End Design and Back End Design. Front End Design includes digital design using HDL and design verification through simulation and other techniques. The backend design comprises CMOS library design and its characterization. It also covers the physical design and fault simulation.

**ASIC vs FPGA**

|  |  |
| --- | --- |
| **ASIC** (Application Specific Integrated Circuit) | **FPGA** (Field Programmable Gate Arrays) |
| It is a chip that is designed for a particular purpose and effectively performs a repeated function. | These are integrated circuits that are manufactured for general use and can include thousands of configurable logic blocks (CLBs) and programmable interconnects. |
| Fixed circuitry for product's life span. | Reconfigurable circuitry after manufacturing. |
| Analog/mixed-signal circuits can be fully implemented. | Suitable for digital designs only. |
| Are smaller than FPGA circuits. | Are typically larger than ASICs. |
| Operate at higher frequency rates. | Difficult to attain high-frequency rates. |
| Low power consumption, high-performance efficiencies. | Low performance efficiencies, higher power consumption. |
| Prototypes must be accurately validated to avoid design iterations. | Prototyping and validating with FPGA is easier |

**VLSI Design Flow**

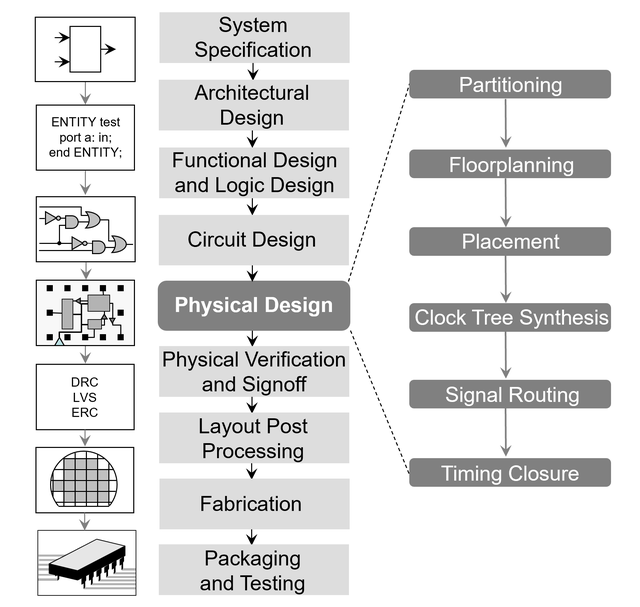
****

Figure 1: VLSI Design Flow

* **SYSTEM SPECIFICATION AND ARCHITECTURAL DESIGN**

The VLSI design cycle starts with a formal specification of a VLSI chip. At this stage, the system specifications are laid out. This involves a high level representation of the system.

* **FUNCTIONAL DESIGN**

In this stage, main functional units of the system and the interconnect requirements between the units are identified. The main purpose of this stage is to specify system behavior, in terms Input, Output and Timing of each unit. The outcome of functional design is usually a diagram showing the relationship of time and other aspects between units.

* **LOGIC DESIGN**

In this stage, the logic for the VLSI system is designed. This includes Boolean expressions, control flow, word width, register allocation, etc. The outcome of this stage is the Register Transfer Level (RTL) description. RTL is expressed in a Hardware Description Language (HDL) like VHDL and Verilog.

* **CIRCUIT DESIGN**

The purpose of the circuit design is to develop a circuit representation based on the logic design. The outcome of this stage is a netlist. Netlist is an electronic circuit system consisting of all the circuit element names/reference designators, listed in a format with their input and output signal names.

* **PHYSICAL DESIGN**

Physical design is the process of transforming netlist into layout. Main steps in physical design are floorplanning, placement of all logical cells, clock tree synthesis and routing. During the process of the physical design area, timing, power, design and technology constraints have to be met.

* **PHYSICAL VERIFICATION**

It will verify that the post-layout netlist and the layout are equivalent, i.e. all connections specified in the netlist is present in the layout.

* **FABRICATION**

Fabrication process includes lithography, polishing, deposition, diffusion, etc. This process consists of several steps and requires various masks. Before the chip is mass produced, a prototype is made and tested.

* **PACKAGING**

Packaging involves putting together the chips on a Printed Circuit Board (PCB) or a Multi-Chip Module (MCM).

**Verilog HDL** (Hardware Description Language)

Verilog is a Hardware Description Language (HDL). It is a language used for describing a digital system such as a network switch, a microprocessor, a memory, or a flip-flop. We can describe any digital hardware by using HDL at any level. Designs described in HDL are independent of technology, very easy for designing and debugging, and are normally more useful than schematics, particularly for large circuits.

Verilog is like any other hardware description language. It permits the designers to design the designs in either Bottom-up or Top-down methodology.

* **Bottom-Up Design:** The traditional method of electronic design is bottom-up. Each design is performed at the gate-level using the standard gates. This design gives a way to design new structural, hierarchical design methods.
* **Top-Down Design:** It allows early testing, easy change of different technologies, and structured system design and offers many other benefits.

### **Verilog Abstraction Levels**

Verilog supports a design at many levels of abstraction, such as:

* Behavioral level
* Register-transfer level
* Gate level

### **Behavioral level**

The behavioral level describes a system by concurrent behavioral algorithms. Every algorithm is sequential, which means it consists of a set of executed instructions one by one. Functions, tasks, and blocks are the main elements. There is no regard for the structural realization of the design.

### **Register-Transfer Level**

Designs using the Register-Transfer Level specify a circuit's characteristics using operations and the transfer of data between the registers.

The modern definition of an RTL code is "Any code that is synthesizable is called RTL code".

### **Gate Level**

The characteristics of a system are described by logical links and their timing properties within the logical level. All signals are discrete signals. They can only have definite logical values (`0', `1', `X', `Z`).

The usable operations are predefined logic primitives (basic gates). Gate level modeling may not be the right idea for logic design. Gate level code is generated using tools such as synthesis tools, and his netlist is used for gate-level simulation and backend.

# **Verilog Basics**

# **Lexical Tokens**

A lexical token may consist of one or more characters, and every single character is in exactly one token. The tokens can be keywords, comments, numbers, white space, or strings. All lines should be terminated by a semi-colon (;).

* Verilog HDL is a case-sensitive language.
* And all keywords are in lowercase.

### **Comments**

There are two types to represent the comments, such as:

1. Single line comments begin with the token // and end with a carriage return.  
   For example, //this is the single-line syntax.
2. Multi-Line comments begin with the token /\* and end with the token \*/  
   For example, /\* this is multiline syntax\*/

**Number Representation in Verilog**

**<size>’<radix><value>**

Ex1: 4’b 1010

Ex2: 8’h ax = 1010xxxx

Ex3: 12’o 3zx7 = 011zzzxxx111

### **Identifiers**

The identifier is the name used to define the object, such as a function, module, or register. Identifiers should begin with alphabetical characters or underscore characters.

For example, A\_Z and a\_z.

### **Operators**

Operators are special characters used to put conditions or to operate the variables. There are one, two, and sometimes three characters used to perform operations on variables.

**1. Arithmetic Operators**

These operators perform arithmetic operations. The + and - are used as either unary (x) or binary (z-y) operators.

The operators included in arithmetic operation are addition, subtraction, multiplication, division, and modulus.

**2. Relational Operators**

These operators compare two operands and return the result in a single bit, 1 or 0. The Operators included in relational operation are:

* == (equal to)
* != (not equal to)
* (greater than)
* >= (greater than or equal to)
* < (less than)
* <= (less than or equal to)

**3. Bitwise Operators**

Bitwise operators do a bit-by-bit comparison between two operands. The Operators included in Bit-wise operation are:

* & (Bitwise AND)
* | (Bit-wiseOR)
* ~ (Bitwise NOT)
* ^ (Bitwise XOR)
* ~^ or ^~(Bitwise XNOR)

**4. Logical Operators**

Logical operators are bitwise operators and are used only for single-bit operands. They return a single bit value, 0 or 1. They can work on integers or groups of bits, expressions and treat all non-zero values as 1.

Logical operators are generally used in conditional statements since they work with expressions. The operators included in Logical operation are:

* ! (logical NOT)
* && (logical AND)
* || (logical OR)

**5. Reduction Operators**

Reduction operators are the unary form of the bitwise operators and operate on all the bits of an operand vector. These also return a single-bit value. The operators included in Reduction operation are:

* & (reduction AND)
* | (reduction OR)
* ~& (reduction NAND)
* ~| (reduction NOR)
* ^ (reduction XOR)
* ~^ or ^~(reduction XNOR)

**6. Shift Operators**

Shift operators are shifting the first operand by the number of bits specified by the second operand in the syntax.

Vacant positions are filled with zeros for both directions, left and right shifts (There is no use sign extension). The Operators included in Shift operation are:

* << (shift left)
* >> (shift right)

**7. Concatenation Operator**

The concatenation operator combines two or more operands to form a larger vector. The operator included in Concatenation operation is:

* { }(concatenation)

**8. Replication Operator**

The replication operator is making multiple copies of an item. The operator used in Replication operation is:

* {n{item}} (n fold replication of an item)

**9. Conditional Operator**

Conditional operator synthesizes to a multiplexer. It is the same kind as is used in C/C++ and evaluates one of the two expressions based on the condition. The operator used in Conditional operation is:

* (Condition) ? (expression1) : (expression2)

### **Operands**

Operands are expressions or values on which an operator operates or works. All expressions have at least one operand.

**1. Literals**

Literals are constant-valued operands that are used in Verilog expressions. The two commonly used Verilog literals are:

* **String**: A literal string operand is a one-dimensional array of characters enclosed in double quotes (" ").
* **Numeric**: A constant number of the operand is specified in binary, octal, decimal, or hexadecimal number.

**2. Wires, Regs, and Parameters**

Wires, regs, and parameters are the data types used as operands in Verilog expressions. Bit-Selection "x[2]" and Part-Selection "x[4:2]"

Bit-selects and part-selects are used to select one bit and multiple bits, respectively, from a wire, regs or parameter vector using square brackets "[ ]".

**3. Function Calls**

In the Function calls, the return value of a function is used directly in an expression without first assigning it to a register or wire.

It just places the function call as one of the types of operands. It is useful to know the bit width of the return value of the function call.

**4x1 MUX Implementation using Verilog**

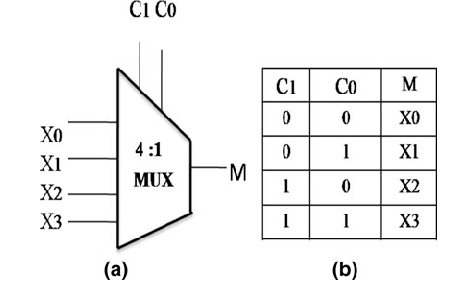


Figure: (a) 4x1 MUX (b) Truth Table

**Verilog Code:**

module mux4\_1(a, sel, y);

input [3:0] a;

input [1:0] sel;

output reg y;

always@(\*)

begin

case(sel)

2'd0:y=a[0];

2'd1:y=a[1];

2'd2:y=a[2];

2'd3:y=a[3];

endcase

end

endmodule

**CONCLUSION**

VLSI technology is a fascinating area in engineering. The technological advancements in this field have had major impacts in almost every other industry in the world and in our lives too. It is the backbone of our digital economy.

With the new technologies in software domain like Machine Learning, Web3 and Quantum Computing, VLSI is easily overlooked. But it must be remembered that it’s the VLSI technology that is powering all the software technologies. The extensive computing power that machine learning algorithms require or the chips in commercial personal computers that allow us to write these algorithms, all are powered by integrated circuits, which are in turn are created using VLSI technology.

In this one month online training by Maven Silicon, I learnt a lot about how chips are designed and how engineers use tools like Quartus Prime, ModelSim and EDA Playground to create, simulate and test chip designs. I learnt in detail about the hardware description language Verilog and then used it to create my very own circuit designs. I then wrote test-benches to test my designs by simulating them. It was a fun and rewarding experience and I cannot wait to put my skills to test by creating more complex circuit designs.

**References**

[1]-VLSI Introduction:-

<https://www.tutorialspoint.com/vlsi_design/vlsi_design_quick_guide.htm>

[2]-VLSI Design Flow:-

<https://www.geeksforgeeks.org/vlsi-design-cycle/>

[3]Verilog HDL:-

<https://www.javatpoint.com/verilog>

[4]-MAVEN Silicon provided resources.