VLSI -Curriculum

1.Introduction to VLSI

1.1 Overview

- VLSI (Very-Large-Scale Integration) is the process of creating integrated circuits by combining thousands to millions of transistors on a single chip.
- This allows for complex and powerful electronic circuits to be miniaturized.
- VLSI is widely used in processors, memory chips, and advanced digital devices.

1.2 Evolution of VLSI

- First Functioning Point Contact Transistor was built in the year 1947
- Metal Oxide Semiconductor Field Effective Transistor (MOSFET) was built in the year 1959
- SSI: Small-Scale Integration with a few transistors per chip.
- MSI: Medium-Scale Integration, increasing to hundreds of transistors
- LSI: Large-Scale Integration, reaching thousands of transistors per chip.
- VLSI: Very-Large-Scale Integration, with millions of transistors on one chip.

1.3 VLSI Design Flow

- System Specification
- Architectural Design
- Functional Design
- Logic Design
- Ciruit Design
- Physical Design
- Fabrication
- Packaging & Testing

2.Digital world

2.1 Digital Design

- Digital design involves creating circuits that process digital signals.
- It uses binary logic (0s and 1s) for computations and data handling.
- Common elements include logic gates, flip-flops, and registers.
- Digital design is foundational for computers, processors, and digital devices.

2.2 Digital System

- A digital system processes and stores information in binary format.
- It consists of components like logic gates, memory units, and processors.
- Digital systems are reliable, precise, and programmable.
- Examples include computers, calculators, and digital communication devices.

2.3 Number System

- Binary Number System
- Decimal Number System
- Octal Number Syste
- Hexadecimal Number System

VLSI -Curriculum

2.4 Number Base Conversion

- Any to Binay (Positional Power)
- Decmal to Any (Using LCM)
- Grouping (Binary to Octal or Hexadecimal)

2.5 Binary Addition & Subtraction

- Binary addition involves adding bits and carrying over when the sum exceeds 1 (1+1 = 10).
- Binary subtraction requires borrowing when subtracting 1 from 0 (1-0 = 1, 0-1 = borrow).
- 1's complement is used by inverting bits to perform subtraction.
- 2's complement involves inverting bits and adding 1 for subtracting signed numbers.

2.6 Further Reading - Number Systems

- BCD Code
- Excess-3 Code
- Gray Code
- Weighted & Non Weighted

3.Logic Design

3.1 Overview

Digital Signals

- High Level = +5v
- Low Level - 0v

Positive Logic

- High = 1/True
- Low = 0/False

Negetive Logic

- High = 0/False
- Low = 1/True

3.2 Boolean Algebra

Identities

- Identity Law
- Null Law
- Complement Law
- Idempotent Law
- Domination Law

Properties

- Commutative Property
- Associative Property
- Distributive Property
- Complementary Property
- Idempotency Property
- Absorption Property

VLSI -Curriculum

3.3 Logic Gates

- Logic gates perform basic operations on binary inputs (0s and 1s).
- AND: Outputs 1 when all inputs are 1.
- OR: Outputs 1 when at least one input is 1.
- NOT: Inverts the input (0 to 1, 1 to 0).
- NAND: Outputs 0 only when all inputs are 1.
- NOR: Outputs 0 when at least one input is 1.
- XOR: Outputs 1 when inputs are different.
- XNOR: Outputs 1 when inputs are the same.

Universal Gates

- NAND Gate: Can create any other gate by combining NANDs.
- NOR Gate: Can form all basic gates through NOR combinations.
- Both NAND and NOR are called universal gates for their versatility in digital circuits.

4.Combinational Circuits

4.1 Adder & Subtractor Circuits

Basic Function

- Adder and subtractor circuits are fundamental arithmetic circuits in digital electronics, used to perform addition and subtraction operations on binary numbers.

Half adder

- A half adder is a simple circuit that adds two single-bit binary numbers and outputs a sum and a carry, though it does not handle carry-in from previous stages.

Full adder

- A full adder adds three binary bits (including carry-in from previous addition) and outputs a sum and carry, making it suitable for multi-bit binary addition when chained together.

Subtractor circuit

- A binary subtractor circuit performs subtraction of two binary numbers, and often uses two's complement representation to handle both positive and negative numbers.

Combined adder-subtractor

- A single circuit can be designed to function as both an adder and subtractor by incorporating control signals to toggle between addition and subtraction modes.

Applications

- These circuits are essential in digital systems like calculators, computers, and microprocessors for performing arithmetic operations and are foundational for arithmetic logic units (ALUs).

VLSI -Curriculum

4.2 Multiplexer & De-Multiplexer Circuits

Multiplexer (MUX)

- A multiplexer is a digital circuit that selects one input from multiple inputs and forwards it to a single output based on control signals, enabling efficient data routing.

Control Signals

- Multiplexers use control signals to select which input to forward, with the number of control lines determined by the number of inputs (e.g., a 4-to-1 MUX requires 2 control lines).

Demultiplexer (DEMUX)

- A demultiplexer takes a single input and directs it to one of multiple outputs, based on control signals, distributing data to different channels.

Data management

- MUX and DEMUX circuits are crucial for data management in communication systems, allowing multiple signals to be transmitted over a single line and then distributed as needed.

Applications

- Used in digital electronics, MUX and DEMUX circuits are essential in systems like digital TVs, data routing, and information processing in computer systems.

Foundational Role

- Multiplexers and demultiplexers are foundational components in the design of complex digital circuits, aiding in efficient data flow and signal control within processors, memory units, and communication networks.

4.3 Encoder and Decoder Circuits

Encoder

- An encoder is a digital circuit that converts an active input signal into a coded output, typically reducing multiple input lines into fewer output lines, often in binary form.

Binary Encoding

- Encoders are commonly used to convert input signals into binary codes, providing a more compact representation of information for easier processing.

Priority Encoder

- A priority encoder assigns priority to inputs, encoding only the highest-priority active input, which is useful in interrupt handling in computer systems.

Decoder

- A decoder performs the reverse function of an encoder, converting coded binary inputs back into specific outputs, allowing binary data to be decoded for various applications.

Control Signals

- Decoders use control inputs to activate specific output lines, essential in memory addressing, display systems, and data routing in digital systems.

Applications

- Encoders and decoders are widely used in digital electronics, such as in communication systems, binary data conversion, memory address decoding, and interfacing with microcontrollers and processors.

VLSI -Curriculum

4.4 Comparator Circuits

Basic Function

- Comparator circuits compare two input voltages or currents and output a signal indicating which input is higher, essential for decision-making in digital electronics.

Types

- Comparators are commonly available as open-loop (non-feedback) circuits and include types such as inverting and non-inverting comparators.

Output Levels

- The output of a comparator is usually binary, providing a high (1) or low (0) signal based on the comparison, making it ideal for threshold detection.

Applications

- Comparators are used in applications like analog-to-digital converters (ADCs), level sensing, zero-crossing detection, and pulse-width modulation (PWM) circuits.

Precision & Speed

- Comparators are designed to provide quick response times and high accuracy, especially in applications requiring fast switching between states.

5.Sequential Circuits

5.1 Latches

- Latches are basic storage elements that hold a single bit of data.
- They are level-triggered and change output based on input when enabled.
- Common types include SR, D, and JK latches.
- Used in digital circuits to store and maintain data states.

5.2 Flip Flops

- SR Flip Flop
- JK Flip Flop
- D Flip Flop
- T Flip Flop

5.3 Shift Registers

- Shift registers store and shift data bits in a sequence.
- They move bits left or right with each clock pulse.
- Types include Serial-In Serial-Out (SISO) and Parallel-In Parallel-Out (PIPO).
- Commonly used in data storage, transfer, and manipulation.

5.4 Counter

- Counters are sequential circuits that count pulses.
- They can be binary, counting in binary sequence, or decimal.
- Types include synchronous and asynchronous counters.
- Used in timing, event counting, and frequency division.

VLSI -Curriculum

5.5 Finite State Machines (FSMs)

- FSMs are models of systems with a finite number of states.
- They transition between states based on inputs and conditions.
- Types include Mealy and Moore machines.
- Used in control systems, protocols, and digital logic design.

5.6 Switch Debounce

- Switch debounce eliminates false signals from mechanical switch bouncing.
- It can be implemented in hardware (e.g., with capacitors) or software (e.g., delays).
- Debouncing ensures a single, clean signal for each button press.
- Common in keyboards, buttons, and input devices to improve reliability.

5.6 Timing Analysis

- Timing analysis ensures that signals in a digital circuit meet required timing constraints.
- It involves checking setup, hold, and propagation delays in sequential circuits.
- Ensures that data is stable before and after clock transitions.
- Used to verify circuit performance and avoid timing violations.

6.Semiconductors

6.1 Overview

- Semiconductors are materials with conductivity between conductors and insulators.
- They are key to electronic components like diodes, transistors, and integrated circuits.
- Silicon is the most common semiconductor material.
- Semiconductors enable control of electrical current in modern electronic devices.

VLSI -Curriculum

6.2 N-Type & P-Type Semiconductors

N-Type Semiconductor

- An N-type semiconductor is doped with elements that have more valence electrons than the semiconductor (usually phosphorus in silicon), creating extra free electrons that serve as charge carriers.

Electron Majority

- In N-type semiconductors, the majority charge carriers are free electrons, which move towards the positive terminal under an electric field, contributing to electrical conductivity.

P-Type Semiconductor

- A P-type semiconductor is doped with elements that have fewer valence electrons (such as boron in silicon), creating "holes" (missing electrons) that act as positive charge carriers.

Hole Majority

- In P-type semiconductors, the majority charge carriers are holes, which are essentially the absence of an electron, and they move towards the negative terminal under an electric field.

Formation of P-N Junction

- When N-type and P-type materials are joined, a P-N junction is formed, enabling the creation of diodes, transistors, and other semiconductor devices essential for electronic circuits.

Applications

- N-type and P-type semiconductors are the building blocks for devices like diodes, transistors, and integrated circuits, used extensively in computers, solar cells, and other electronic components.

6.3 Transistors

Basic Function

- A transistor is a semiconductor device used to amplify or switch electronic signals, acting as a key building block in modern electronic circuits.

Types

- There are two main types of transistors—bipolar junction transistors (BJT) and field-effect transistors (FET)—each with unique operating principles and applications.

Switching & Amplification

- Transistors can function as switches, controlling the flow of current between two terminals, or as amplifiers, boosting the strength of weak signals.

Applications

- Transistors are essential in applications like signal processing, amplification in audio systems, logic gates in digital circuits, and power regulation in power supplies.

Integration in ICs

- Transistors are the fundamental components in integrated circuits (ICs), enabling the development of complex microprocessors, memory chips, and other electronic devices.

VLSI -Curriculum

7.Field Programmable Gate Array (FPGA)

7.1 Overview

- FPGA (Field-Programmable Gate Array) is a flexible, reconfigurable hardware device.
- It consists of an array of programmable logic blocks and interconnections.
- FPGAs are used for custom hardware design and parallel processing tasks.
- They are commonly used in digital circuits, signal processing, and embedded systems.

7.2 Basic Architecture

- FPGAs have programmable logic blocks (CLBs).
- Configurable interconnects link the blocks.
- Input/output blocks (IOBs) handle external connections.
- Programmable routing matrix enables flexible wiring.
- May include memory and DSP blocks.
- Configured via a bitstream.

7.3 Comparison of FPGAs with CPLDs, Microcontrollers and ASCIs

Flexibility

- FPGAs offer high flexibility with programmable logic; CPLDs are less flexible; Microcontrollers and ASCIs are typically fixed.

Size and Complexity

- FPGAs handle large, complex designs; CPLDs are simpler; Microcontrollers are compact with limited complexity; ASCIs are custom-designed for specific tasks.

Processing Power

- FPGAs excel in parallel processing; Microcontrollers focus on sequential tasks; CPLDs offer moderate processing; ASCIs provide tailored power.

Reconfigurability

- FPGAs are fully reconfigurable; CPLDs are limited; Microcontrollers and ASCIs are generally fixed.

Speed

- FPGAs are the fastest; Microcontrollers are slower; CPLDs sit between them; ASCIs are optimized for specific tasks.

Power Consumption

- Microcontrollers consume the least power; CPLDs are more efficient than FPGAs, which consume more power.

Cost

- Microcontrollers and CPLDs are more cost-effective; FPGAs are more expensive; ASCIs vary in cost.

Use Cases

- FPGAs are used for high-performance tasks; Microcontrollers in embedded systems; CPLDs for simpler logic; ASCIs for custom applications.

VLSI -Curriculum

8.Hardware Description LanguageS (HDL)

8.1 Verilog HDL Overview

Hardware Description Language

- Verilog HDL is used to model digital systems, describing the behavior, structure, and timing of circuits in a textual form.

Simulation and Testing

- It allows for simulation and verification of digital designs before hardware implementation, ensuring functionality and performance.

Design and Synthesis

- Verilog is widely used for both designing and synthesizing circuits, particularly in FPGA and ASIC development.

C-like Syntax

- Verilog has a syntax similar to the C programming language, making it easier to learn for those familiar with programming.

8.2 Typical Design Flow

- Write Verilog code to define the design.
- Behavioural Description
- RTL Description (HDL)
- Functional Verification & Testing
- Logic Synthesis/Timing Verification
- Gatelevel Netlist
- Logic Verification & Testing
- Floor Planning Automatic Place & Route
- Physical Layout
- Layout Verification
- Implementation

8.3 Lexical Convention

- Lexical conventions define the basic syntax rules of a language.
- They specify valid identifiers, keywords, and literals.
- Whitespace is ignored except in string literals or comments.
- Comments are used for code annotations, usually starting with // or /*.
- Symbols like =, +, and - represent operators in the language.

VLSI -Curriculum

8.4 Data Types

Value Set

- It is used to represent signals that can hold a specific set of values.
- The most common value sets are used for signals that have multiple states, and Verilog provides several data types to represent these different sets

Nets

- It represent continuous connections and signal pathways between circuit elements
- They do not store values but instead carry signals based on input values from connected drivers, like combinational logic.
- Net datatypes are essential for modeling connections that respond continuously to input changes in combinational circuits.

Registers

- Registers (reg) store values in sequential logic, retaining state until updated.
- They work by holding data on a clock edge, useful for flip-flops and memory.
- Assigned in always and initial blocks, updating based on triggers.

Vector

- A multi-bit data type, specified by a range (e.g., reg [7:0]), used to represent buses and multi-bit signals.
- Declared with a range, like [7:0], to specify bit-width and direction.
- Useful for handling binary data, signals, and registers of varying sizes.

Integer

- A 32-bit signed data type used for variables, counters, and loops in procedural blocks.

Real

- Represents floating-point numbers for simulation purposes, but not synthesizable to hardware.

Time

- A special datatype used to represent simulation time, typically in units of time like nanoseconds or picoseconds.

Arrays

- Arrays in Verilog store multiple values of the same data type, accessed by index.
- Declared with dimensions, like reg [7:0] my_array [0:3], to specify size.
- Commonly used for memory storage, lookup tables, or grouped data signals.

Memories

- Used to represent arrays of data elements, allowing for storage and retrieval of multiple values in a structured way.
- These are especially useful for modeling RAM, ROM, and other forms of storage within digital systems

Parameter

- Defines constants that are set at compile-time, commonly used for configuration in modules.

Strings

- Represents an array of characters, used in simulation for text handling and debugging, but not synthesizable.

VLSI -Curriculum

8.5 System Tasks & Compliers

System Tasks

- Displaying Information
- Monitoring Information
- Stopping and Finishing a Simulation

Compiler Directives

- Define
- Include

8.6 Modules & Ports

- Modules in Verilog define the design blocks or components of a circuit.
- Each module has ports (input, output, inout) for communication with other modules.
- Ports enable data flow into and out of the module, making designs modular.
- Modules can be instantiated within other modules, allowing hierarchical design.

8.7 Verilog Modeling

Behavioral Modeling

- Describes the functionality of a design at a high level using constructs like always blocks, if statements, and loops, focusing on what the design does rather than how it is implemented.

Dataflow Modeling

- Uses continuous assignments and operators (e.g., assign statements) to describe the flow of data between signals, emphasizing how data moves through the design.

Structural Modeling

- Builds a design by interconnecting lower-level modules, representing the hardware structure directly, similar to a schematic or physical connections.

Gate-Level Modeling

- Describes the circuit using basic logic gates (e.g., and, or, not) and flip-flops, providing a very detailed, low-level representation of the design.

Switch-Level Modeling

- Models circuits at the transistor level using switches (pmos, nmos), capturing the exact behavior of MOSFET-based designs and suitable for low-level simulations.

Mixed-Modeling

- Combines behavioral, dataflow, and structural modeling techniques within a single design to optimize performance and flexibility in different parts of a Verilog module.

8.8 Basic Circuits written in Verilog

- Verilog describes digital circuits using modules, defining structure and behavior.
- Circuits can be designed at various abstraction levels: behavioral, dataflow, or structural.
- Supports combinational (logic gates) and sequential circuits (flip-flops, registers).
- Simulations verify functionality, while synthesis converts code to hardware.

VLSI -Curriculum

8.9 Design Methodologies

- Top-Down Design
- Start with a high-level overview, then break down into detailed components.
- Bottom-Up Design
- Begin with basic components, combining them to build complex systems.
- HDL-Based Design
- Use Hardware Description Languages like Verilog or VHDL to model circuits.
- Simulation and Testing
- Verify functionality through simulations before hardware implementation.

VLSI -Curriculum

CAPSTONE PROJECTS

1 **Sequence Detector**

- **FSM Design Principles:** Hands-on experience will be gained in designing and implementing finite state machines, enhancing understanding of sequential logic.
- **Verilog Proficiency:** Skills in Verilog coding will be improved, including state transitions and output generation, which are crucial for digital design.
- **Simulation and Testing:** The importance of simulation for verifying functionality and identifying errors will be learned, ensuring reliable performance before deployment.

2 **Traffic Light Controller**

- **Asynchronous Counter Design:** Develop a solid understanding of asynchronous counter mechanisms, focusing on state transitions for traffic control.
- **Verilog Coding Techniques:** Enhance Verilog programming skills, particularly in managing timing and synchronization for effective traffic light sequencing.
- **System Simulation:** Emphasize the significance of thorough simulation to validate the traffic light controller's operation under different conditions.

3 **Vending Machine using Verilog HDL**

- **Digital Design Understanding:** Gain insight into designing a complex digital system, managing transactions and product selection efficiently.
- **Verilog HDL Proficiency**: Improve skills in Verilog HDL for creating modules that handle product selection, payment processing, and change return.
- **Testing and Simulation:** Learn the importance of thorough simulation and testbench creation to validate system functionality and handle edge cases.

LIVE PROJECT

1 **16-Bit RISC Processor**

- **Processor Architecture Insight:** Gain valuable knowledge of RISC-V architecture, focusing on instruction sets and their execution in hardware.
- **Verilog Implementation Skills:** Enhance Verilog skills through practical application in designing and integrating ALU, registers, and control units.
- **Debugging and Verification:** Develop techniques for debugging and verifying processor functionality, ensuring accurate instruction execution and data handling.