DIGITAL VLSI DESIGN

23EC2222F ACADEMIC YEAR: 2024-25 Even Semester

STUDENT ID:

STUDENT NAME:

Table of Contents

A.Y. 2024-25 LAB/SKILL CONTINUOUS EVALUATION

S.No	Date	Experiment Name	Pre-Lab (10M)	In-Lab (25M)			Post-Lab (10M)	Viva Voce (5M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10M)	Analysis & Inference (10M)				
1.		Introductory Session	-NA-							
2.		Realization of Boolean functions using basic and universal gates (SOP and POS forms) using Verilog HDL.								
3.		Design and Realization of following using basic gates and universal gates using Verilog HDL. a. Half Adder and Full adder b. Half subtractor and Full subtractor								
4.		Design and realize the 4-Bit Ripple Carry Adder (RCA) Full Adder functional module using Verilog HDL.								
5.		Design and realize the Multiplexers and De-multiplexers using Verilog HDL.								
6.		Design and realize the Encoders and Decoders using Verilog HDL.								
7.		Design and realize the magnitude comparator using Verilog HDL.								
8.		Design and realize the Flip Flops using NAND with Verilog HDL a. S-R Flip-Flop b. D Flip-Flop c. T Flip-Flop d. JK Flip-Flop e. Master Slave JK FF								

S.No	Date	Experiment Name	Pre-Lab (10M)	In-Lab (25M)			Post-Lab (10M)	Viva Voce (5M)	Total (50M)	Faculty Signature
				Program/ Procedure (5M)	Data and Results (10M)	Analysis & Inference (10M)				
9.		Design and realize the following shift registers using Verilog HDL a. SISO b. SIPO c. PISO d. PIPO.								
10.		Design and realize the Johnson Counter Verilog HDL.								
11.		Design and realize the Ring Counter Verilog HDL.								
12		Design and realize the sequence detector circuit using Verilog HDL.								
13.		Design and realize the FSM using Mealy and Moore Model using Verilog HDL								

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Realization of Boolean functions using basic and universal gates (SOP and POS forms) using Verilog HDL

Aim/Objective:

The aim of this experiment is to verify the functionality of basic logic gates and universal gates using a Verilog HDL.

Description:

In this experiment, we will focus on verifying the functionality of basic logic gates, including AND, OR, NOT, NAND, and NOR gates, using a Verilog HDL. The Test Bench allows us to simulate and test the behaviour of the logic gates by applying different input combinations and observing the corresponding output responses.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What are the basic logic gates and their corresponding truth tables?

2. What is the purpose of a Test Bench in digital circuit verification?

3. What is Verilog, and how is it used in circuit simulation and verification?

Insert text here

4. How do you represent a Boolean function in POS form?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design the basic logic gates using Verilog code.

Create a Test Bench module that generates input patterns and applies them to the logic gates.

Simulate the logic gates and Test Bench using the Verilog simulation environment.

Observe and record the output responses of the logic gates for different input combinations.

Sample VIVA-VOCE Questions (In-Lab):

1. What are the basic logic gates and their corresponding truth tables?
2. How does a Test Bench aid in the verification of digital circuits?
3. What is the role of Verilog in circuit simulation and verification?
4. How do you generate different input patterns in a Test Bench for logic gates?
5. What are the advantages of using a Test Bench for circuit verification?

Post-Lab:

1. Analysing the output responses of the logic gates helps verify their functionality.
2. Comparing the observed output responses with the expected behaviour ensures the correctness of the implemented logic gates.
3. Assessing any discrepancies or errors in the output responses helps identify and investigate potential issues in the logic gate implementations.
4. Making necessary adjustments or corrections to the logic gate implementations ensures their accuracy and adherence to the desired specifications.
5. Documenting the findings, observations, and modifications made during the post-lab analysis provides a comprehensive record of the verification process.

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment Title: Design and Realization of following using basic gates and universal gates using Verilog HDL.

a. Half Adder and Full adder

b. Half subtractor and Full subtractor.

Aim/Objective:

The aim of this experiment is to verify the functionality of a Half, Full Adder, half subtractor and Full Subtractor circuit using a Verilog HDL.

Description:

In this experiment, we will focus on verifying the functionality of the functionality of a Half, Full Adder, half subtractor and Full Subtractor circuit using a Verilog HDL. The functionality of a Half, Full Adder, half subtractor and Full Subtractor is a fundamental digital circuit used for addition in binary arithmetic. By designing the functionality of a Half, Full Adder, half subtractor and Full Subtractor circuit and simulating it using a Verilog HDL code, we can validate its behaviour and ensure it performs addition accurately.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the purpose of a Half/Full Adder circuit?

2. How does a Half subtractor differ from a Full subtractor?

3. What is the role of a Verilog code in digital circuit verification?

4. How can Verilog be used to simulate and verify circuit behaviour?

In-Lab:

Set up the required software and tools, including a computer with a Verilog simulation environment.

Design the subtractor circuit using Verilog code, considering the input and output requirements.

Create a Test Bench module that generates input patterns for the adder circuit.

Simulate the Half, Full Adder, half subtractor and Full Subtractor circuit and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the adder circuit for different input combinations.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the purpose of a Half/Full subtractor circuit?
2. How does a Half Adder differ from a Full Adder?
3. Explain the significance of carry propagation in addition circuits.
4. How does a Test Bench aid in the verification of digital circuits?
5. How can Verilog be used to simulate and verify circuit behaviour?

Post-Lab:

1. Analysing the output results of the adder circuit helps verify its functionality and accuracy in performing binary addition.
2. Comparing the obtained output results with the expected behaviour ensures the correctness of the Half, Full Adder, half subtractor and Full Subtractor circuit implementation.
3. Evaluating the ability of the circuit to propagate carry and generate accurate sum outputs confirms its adherence to the desired specifications.
4. Identifying any discrepancies or errors in the output results allows for investigation and potential corrections to improve the circuit's performance.
5. Documenting the findings, observations, and modifications made during the post-lab analysis provides a comprehensive record of the verification process.

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the 4-Bit Ripple Carry Adder (RCA) Full Adder functional module using Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of a 4-bit Ripple Carry Adder (RCA) Full Adder circuit using a Verilog HDL.

Description:

In this experiment, we will focus on verifying the functionality of the functionality of a 4-bit Ripple Carry Adder circuit using a Verilog HDL. The functionality of a 4-bit Ripple Carry Adder is a fundamental digital circuit used for addition in binary arithmetic. By designing the functionality of a 4-bit Ripple Carry Adder circuit and simulating it using a Verilog HDL code, we can validate its behaviour and ensure it performs addition accurately.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the significance of the term "ripple carry" in a Ripple Carry Adder?

2. What is the purpose of a carry bit in binary addition?

3. What are some common issues that may arise during Verilog design and simulation?

4. How can Verilog be used to simulate and verify circuit behaviour?

In-Lab:

Set up the required software and tools, including a computer with a Verilog simulation environment.

Design the RCA circuit using Verilog code, considering the input and output requirements.

Create a Test Bench module that generates input patterns for the RCA adder circuit.

Simulate the 4-bit ripple carry adder circuit and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the adder circuit for different input combinations.

Sample VIVA-VOCE Questions (In-Lab):

1. What is a Ripple Carry Adder, and why is it called "ripple carry"?
2. What is the purpose of writing a Verilog module for a digital circuit?
3. How do you represent the sum and carry-out in the Verilog module?
4. How do you connect four full adders to create a 4-Bit Ripple Carry Adder?
5. How can Verilog be used to simulate and verify circuit behaviour?

Post-Lab:

1. Analysing the output results of the adder circuit helps verify its functionality and accuracy in performing binary addition.
2. Comparing the obtained output results with the expected behaviour ensures the correctness of the 4-bit RCA circuit implementation.
3. Evaluating the ability of the circuit to propagate carry and generate accurate sum outputs confirms its adherence to the desired specifications.
4. Identifying any discrepancies or errors in the output results allows for investigation and potential corrections to improve the circuit's performance.
5. Documenting the findings, observations, and modifications made during the post-lab analysis provides a comprehensive record of the verification process.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the Multiplexers and De-multiplexers using Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of Multiplexers and De-multiplexers circuit using a Verilog HDL.

Description:

In this experiment, we will focus on verifying the functionality of the functionality of a Multiplexers and De-multiplexers circuit using a Verilog HDL. The functionality of a Multiplexers and De-multiplexers is a fundamental digital circuit used for addition in binary arithmetic. By designing the functionality of a Multiplexers and De-multiplexers circuit and simulating it using a Verilog HDL code, we can validate its behaviour and ensure it performs addition accurately.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the primary purpose of a MUX?

2. How does a multiplexer select one of its input lines based on the select lines?

3. What role do the select lines play in a de-multiplexer?

4. How do you declare input and output ports in a Verilog MUX module?

In-Lab:

Set up the required software and tools, including a computer with a Verilog simulation environment.

Design the Multiplexers and De-multiplexers circuit using Verilog code, considering the input and output requirements.

Create a Test Bench module that generates input patterns for the Multiplexers and De-multiplexers circuit.

Simulate the Multiplexers and De-multiplexers circuit and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the Multiplexers and De-multiplexers circuit for different input combinations.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the fundamental purpose of a multiplexer in digital circuits?
2. Can you briefly describe the structure of a Verilog module for a multiplexer?
3. How do you declare input and output ports for a multiplexer in Verilog?
4. How do you simulate the behaviour of a De-multiplexer using Verilog?
5. How can Verilog be used to simulate and verify circuit behaviour?

Post-Lab:

1. Analysing the output results of the adder circuit helps verify its functionality and accuracy in performing binary addition.
2. Comparing the obtained output results with the expected behaviour ensures the correctness of the 4-bit RCA circuit implementation.
3. Evaluating the ability of the circuit to propagate carry and generate accurate sum outputs confirms its adherence to the desired specifications.
4. Identifying any discrepancies or errors in the output results allows for investigation and potential corrections to improve the circuit's performance.
5. Documenting the findings, observations, and modifications made during the post-lab analysis provides a comprehensive record of the verification process.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the Encoders and Decoders using Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of Encoders and Decoders circuit using a Verilog HDL.

Description:

In this experiment, we will focus on verifying the functionality of the functionality of an Encoders and Decoders circuit using a Verilog HDL. The functionality of an Encoders and Decoders is a fundamental digital circuit used for addition in binary arithmetic. By designing the functionality of an Encoders and Decoders circuit and simulating it using a Verilog HDL code, we can validate its behaviour and ensure it performs addition accurately.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. Explain the concept of an Encoder.

2. How do you simulate the behaviour of an Encoder using Verilog?

3. Explain the concept of a Decoder.

4. What are enable inputs in a Decoder, and how can they be utilized in Verilog designs?

In-Lab:

Set up the required software and tools, including a computer with a Verilog simulation environment.

Design the subtractor circuit using Verilog code, considering the input and output requirements.

Create a Test Bench module that generates input patterns for the Encoders and Decoders circuit.

Simulate the Encoders and Decoders circuit and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the Encoders and Decoders circuit for different input combinations.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the fundamental purpose of a multiplexer in digital circuits?
2. Can you briefly describe the structure of a Verilog module for a multiplexer?
3. How do you declare input and output ports for a multiplexer in Verilog?
4. How do you simulate the behaviour of a De-multiplexer using Verilog?
5. How can Verilog be used to simulate and verify circuit behaviour?

Post-Lab:

1. Analysing the output results of the adder circuit helps verify its functionality and accuracy in performing Encoders and Decoders.
2. Comparing the obtained output results with the expected behaviour ensures the correctness of the Encoders and Decoders circuit implementation.
3. Evaluating the ability of the circuit to Encoders and Decoders outputs confirms its adherence to the desired specifications.
4. Identifying any discrepancies or errors in the output results allows for investigation and potential corrections to improve the circuit's performance.
5. Documenting the findings, observations, and modifications made during the post-lab analysis provides a comprehensive record of the verification process.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment Title: Design and realize the magnitude comparator using Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of a Magnitude Comparator using a Verilog HDL.

Description:

In this experiment, we will focus on verifying the functionality of a Magnitude Comparator using a Verilog Test Bench. A Magnitude Comparator is a digital circuit used to compare the magnitudes of two binary numbers. By designing a Magnitude Comparator and simulating it using a Test Bench, we can validate its behaviour and ensure it performs accurate magnitude comparison.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the purpose of a Magnitude Comparator circuit?

2. How does a 2-bit Magnitude Comparator differ from other types of comparators?

3. What is the significance of magnitude comparison in digital circuits?

4. What is the role of a Test Bench in circuit verification?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design the 2-bit Magnitude Comparator circuit using Verilog code, considering the input and output requirements.

Create a Test Bench module that generates input patterns for the comparator circuit.

Simulate the 2-bit Magnitude Comparator circuit and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the comparator circuit for different input combinations.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the purpose of a Magnitude Comparator circuit?
2. How does a 2-bit Magnitude Comparator differ from other types of comparators?
3. Explain the significance of magnitude comparison in digital circuits.
4. How does a Test Bench aid in the verification of digital circuits?
5. How can Verilog be used to simulate and verify circuit behaviour?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the comparator circuit with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the comparator circuit implementation, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the Flip Flops using NAND with Verilog HDL

a. **S-R Flip-Flop**
b. **D Flip-Flop**
c. **T Flip-Flop**
d. **JK Flip-Flop**
e. **Master Slave JK FF**

Aim/Objective:

The aim of this experiment is to verify the functionality of different types of flip-flops using a Verilog HDL code.

Description:

In this experiment, we will focus on verifying the functionality of various flip-flops using a Verilog HDL code. Flip-flops are fundamental building blocks in digital circuits used for storing and synchronizing data. By designing and simulating different types of flip-flops using a Verilog HDL code, we can validate their behaviour and ensure they function correctly.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the purpose of a flip-flop in digital circuits?

2. What are the different types of flip-flops available?

3. How does a D flip-flop differ from a JK flip-flop?

4. What are the input and output characteristics of a flip-flop?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design and implement different types of flip-flops (such as D flip-flop, JK flip-flop, etc.) using Verilog code.

Create a Test Bench module that generates input patterns for the flip-flops.

Simulate the flip-flops and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the flip-flops for different input sequences.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the purpose of a flip-flop in digital circuits?
2. How do different types of flip-flops differ from each other?
3. Explain the significance of clock signals in flip-flop operation.
4. How can Verilog be used to simulate and verify the behaviour of flip-flops?
5. What are the key characteristics to consider when evaluating the performance of a flip-flop?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the flip-flops with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the flip-flop implementations, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the following shift registers using Verilog HDL

a. **SISO**
b. **SIPO**
c. **PISO**
d. **PIPO.**

Aim/Objective:

The aim of this experiment is to verify the functionality of different types of shift registers using a Verilog HDL code.

Description:

In this experiment, we will focus on verifying the functionality of various shift registers using a Verilog HDL code. A shift register is a type of digital circuit using a cascade of flip-flops where the output of one flip-flop is connected to the input of the next. They share a single clock signal, which causes the data stored in the system to shift from one location to the next. By designing and simulating different types of shift register using a Verilog HDL code, we can validate their behaviour and ensure they function correctly.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the purpose of a shift registers in digital circuits?

2. What are the different types of shifts registers available?

3. How does a SISO differ from a SIPO?

4. What are the input and output characteristics of a shift registers?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design and implement different types of shift register (such as SISO, SIPO, PISO and PIPO) using Verilog code.

Create a Test Bench module that generates input patterns for the shift register.

Simulate the shift register and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the shift register for different input sequences.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the purpose of a shift registers in digital circuits?
2. How do different types of shift registers differ from each other?
3. What is the purpose of the clock signal in a SISO shift register?
4. How can Verilog be used to simulate and verify the behaviour of shift registers?
5. What are the key characteristics to consider when evaluating the performance of a shift registers?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the shift register with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the shift register implementations, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment Title: Design and realize the Johnson Counter Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of Johnson Counter using a Verilog HDL code.

Description:

In this experiment, we will focus on verifying the functionality of Johanson counter using a Verilog HDL code. The Johnson counter circuit is the cascaded arrangement of 'n' flip-flops. In such design, the output of the proceeding flip-flop is fed back as input to the next flip-flop. By designing and simulating Johnson counter circuit using a Verilog HDL code, we can validate their behaviour and ensure they function correctly.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is an asynchronous counter?

2. What is the difference between the ring counter and Johnson counter?

3. What is D flip flop?

4. What are the input and output characteristics of a Counters?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design and implement different types of Johnson counter using Verilog code.

Create a Test Bench module that generates input patterns for the Johnson counter.

Simulate the Johnson counter and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the Johnson counter for different input sequences.

Sample VIVA-VOCE Questions (In-Lab):

1. What are the key characteristics and applications of Johnson Counters?
2. What are the key components of a Verilog module?
3. How is a Johnson Counter different from other types of counters?
4. How do you handle clocking in your Verilog code?
5. How does their operation make them suitable for these applications?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the Johnson counter with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the Johnson counter implementations, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the Ring Counter Verilog HDL..

Aim/Objective:

The aim of this experiment is to verify the functionality of Ring Counter using a Verilog HDL code.

Description:

In this experiment, we will focus on verifying the functionality of Ring counter using a Verilog HDL code. The Johnson counter circuit is the cascaded arrangement of 'n' flip-flops. In such design, the output of the proceeding flip-flop is fed back as input to the next flip-flop. By designing and simulating Ring counter circuit using a Verilog HDL code, we can validate their behaviour and ensure they function correctly.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is a synchronous counter?

2. What is the difference between the ring counter and Johnson counter?

3. What is ring counter?

4. What are the input and output characteristics of a Ring Counters?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design and implement different types of Johnson counter using Verilog code.

Create a Test Bench module that generates input patterns for the Ring counter.

Simulate the Ring counter and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the Ring counter for different input sequences.

Sample VIVA-VOCE Questions (In-Lab):

1. What are the key characteristics and applications of Ring Counters?
2. What are the key components of a Verilog module?
3. How is a Ring Counter different from other types of counters?
4. How do you handle clocking in your Verilog code?
5. How does their operation make them suitable for these applications?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the Ring counter with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the Ring counter implementations, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the sequence detector circuit using Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of different types of sequence detector using a Verilog HDL code.

Description:

In this experiment, we will focus on verifying the functionality of various sequence detector using a Verilog HDL code. A sequence detector is a sequential circuit that outputs 1 when a particular pattern of bits sequentially arrives at its data input by designing and simulating different types of sequence detector using a Verilog HDL code, we can validate their behaviour and ensure they function correctly.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is the purpose of a sequence detector in digital circuits?

2. What are the typical applications of sequence detectors?

3. Why is it important to clearly define the sequence?

4. How does the state table guide the behaviour of the circuit?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design and implement sequence detector using Verilog code.

Create a Test Bench module that generates input patterns for the sequence detector.

Simulate the sequence detector and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the shift register for different input sequences.

Sample VIVA-VOCE Questions (In-Lab):

1. What are the essential input and output ports for this module?
2. How are state registers implemented in Verilog for the sequence detector?
3. How does the current state and input determine the next state?
4. How does your Verilog code handle clocking in the context of a sequence detector?
5. What is the purpose of a sequence detector circuit?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the sequence detector with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the sequence detector implementations, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Experiment Title: Design and realize the FSM using Mealy and Moore Model using Verilog HDL.

Aim/Objective:

The aim of this experiment is to verify the functionality of different types of Finite State Machine using a Verilog HDL code.

Description:

In this experiment, we will focus on verifying the functionality of various Finite State Machine using a Verilog HDL code. Designing and realizing a Finite State Machine (FSM) using both Mealy and Moore models involves specifying state transitions and outputs based on the current state and inputs. Designing and simulating different types of Finite State Machine using a Verilog HDL code, we can validate their behaviour and ensure they function correctly.

Pre-Requisites:

Verilog, Xilinx Vivado

Pre-Lab:

1. What is a Finite State Machine (FSM)?

2. What is its significance in digital design?

3. What is the difference between Mealy and Moore models?

4. How do you implement a Mealy FSM in Verilog HDL?

In-Lab:

Set up the necessary software and tools, including a computer with Verilog simulation environment.

Design and implement Finite State Machine using Verilog code.

Create a Test Bench module that generates input patterns for the Finite State Machine.

Simulate the Finite State Machine and Test Bench using the Verilog simulation environment.

Monitor and record the output results of the shift register for different input sequences.

Sample VIVA-VOCE Questions (In-Lab):

1. What is the fundamental difference between a Mealy and Moore model.?
2. Draw a state diagram for your FSM.
3. How do you implement a Mealy FSM in Verilog HDL?
4. How are state transitions and outputs determined in a Moore machine?
5. How do you implement a Moore FSM in Verilog HDL?

Post-Lab:

1. Analyze the collected data and results from the simulation.
2. Compare the obtained output results of the Finite State Machine with the expected behaviour.
3. Identify any discrepancies or errors in the output results and investigate their causes.
4. Make necessary adjustments or corrections to the Finite State Machine implementations, if required.
5. Document the findings, observations, and any modifications made during the post-lab analysis.

Experiment #	<TO BE FILLED BY STUDENT>	Student ID	<TO BE FILLED BY STUDENT>
Date	<TO BE FILLED BY STUDENT>	Student Name	<TO BE FILLED BY STUDENT>

Inferences:

Evaluator Remark (if Any):	**Marks Secured: _____out of 50**
	Signature of the Evaluator with Date

The evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.