DAY 9 -111 DAYS VERIFICATION CHALLENGE

Topic: Sequential Circuits

Skill: Digital Electronics

DAY 9 CHALLENGE:

1. List the difference between Sequential & Combinational Circuits.

Combinational Circuit

1. In this output depends only upon present input.
2. Speed is fast.
3. It is designed easy.
4. There is no feedback between input and output.
5. This is time independent.
6. Elementary building blocks: Logic gates
7. Used for arithmetic as well as boolean operations.
8. Combinational circuits don’t have capability to store any state.
9. As combinational circuits don’t have clock, they don’t require triggering.
10. These circuits do not have any memory element.
11. It is easy to use and handle.

Sequential Circuit

1. In this output depends upon present as well as past input.
2. Speed is slow.
3. It is designed tough as compared to combinational circuits.
4. There exists a feedback path between input and output.
5. This is time dependent.
6. Elementary building blocks: Flip-flops
7. Mainly used for storing data.

8. Sequential circuits have capability to store any state or to retain earlier state.
9. As sequential circuits are clock dependent they need triggering.
10. These circuits have memory element.
11. It is not easy to use and handle.

2. List the applications of Sequential & Combinational Circuits.

Combinational Circuit

Arithmetic and Logic Units (ALUs):

Adders, Subtractors, Multipliers, Comparators

Data Processing:

Multiplexers, Demultiplexers, Encoders, Decoders

Memory Address Decoding:

Decoders, Encoders

Code Converters:

Binary to Gray Code Converters, Gray Code to Binary Converters

Sequential Circuit

Storage Devices:

Flip-Flops, Latches

Counters:

Binary Counters, BCD Counters

Registers:

Shift Registers, Parallel Registers

Control Systems:

Finite State Machines (FSMs), Sequence Generators

Communication Systems:

Serial to Parallel Converters, Parallel to Serial Converters

Digital Clocks and Timers:

Timing circuits

3. Why does sequential circuit contain memory element?

Sequential circuits have memory elements because their outputs depend on both current inputs and previous states. Memory elements, like flip-flops and latches, store these previous states, allowing the circuit to remember past inputs and make decisions based on a sequence of events, not just immediate inputs. This is essential for tasks such as counting, timing, and implementing state machines.

4. What is an asynchronous sequential circuit?

An asynchronous sequential circuit is a type of sequential circuit that does not use a clock signal to synchronize state changes. Instead, state changes occur immediately in response to input changes. This allows for faster operation since the circuit can respond directly to inputs without waiting for a clock pulse. However, designing these circuits can be more complex due to potential timing issues like race conditions and hazards. Asynchronous circuits are often used in applications where speed is critical and clock distribution is challenging, such as in certain communication systems and low-power devices.

5. What are the two types of asynchronous sequential circuits?

- **Fundamental Mode:** This is the most common approach for design. The circuit operates under the assumption that only **one input can change at a time** after the circuit reaches a stable state. This simplifies analysis and avoids some potential issues.
- **Pulse Mode:** This mode allows for **multiple inputs to change simultaneously** after a stable state. While it offers more flexibility, it also introduces challenges in analysis and design. Pulse mode circuits are more susceptible to hazards and race conditions, which need careful consideration.

6. What are the types of hazards in asynchronous sequential circuits?

- **Static 1-hazard (Essential Hazard)**: This occurs when a change in an input briefly causes the output to momentarily go high or low before stabilizing to the correct state. It's termed "static" because the output eventually settles correctly.

- **Static 0-hazard**: Similar to the static 1-hazard, but it involves a temporary glitch where the output briefly goes low before stabilizing to the correct logic level.
- **Dynamic Hazard**: This type of hazard results in the output oscillating between correct and incorrect states due to propagation delays and asynchronous paths within the circuit.

7. What is race condition in asynchronous sequential circuits? How to avoid this race conditions?

Race Condition Explanation:

1. **Timing Dependency**: Race conditions arise when the correct operation of the circuit depends on the exact timing or sequence of events, which can vary due to delays in signal propagation or asynchronous inputs.
2. **Undesirable States**: They can cause the circuit to temporarily enter unintended states or produce incorrect outputs due to the simultaneous or overlapping changes in signals.

Avoiding Race Conditions:

1. **Use of Timing Constraints**: Define and adhere to timing constraints to ensure that signals have sufficient time to stabilize before being used in subsequent operations.
2. **Synchronization Techniques**: Employ synchronization techniques such as edge-triggered flip-flops or synchronization signals to ensure that inputs are processed in a controlled and sequential manner.
3. **Critical Path Analysis**: Identify and analyze the critical paths in the circuit where race conditions are most likely to occur. Optimize these paths to minimize timing discrepancies.
4. **State Machine Design**: Design the state machine or logic circuits with clear and well-defined state transitions to minimize the possibility of unintended overlaps or conflicts.
5. **Asynchronous Protocol Design**: Use asynchronous protocols that incorporate handshaking mechanisms or delay elements to manage and synchronize data transfers and state changes.

8. What is the difference between race condition and deadlock

- **Race Condition**: Timing issue in concurrent systems where the outcome depends on unpredictable execution order of processes accessing shared resources.
- **Deadlock**: Circular dependency where processes are stuck waiting indefinitely for resources held by each other, preventing further progress.

9. How do you prevent deadlocks?

Prevent deadlocks by using strategies like resource allocation policies (e.g., banker's algorithm), ensuring a fixed order for resource requests, employing timeouts, avoiding hold and wait situations, and implementing deadlock detection with recovery mechanisms. These measures help maintain system stability and prevent processes from becoming stuck indefinitely.

10.What is a glitch? How to avoid & fix glitches?

Glitch:

- **Definition**: A temporary and unintended fluctuation or inconsistency in system behavior due to noise or timing issues.
- **Avoidance**:
  - Reduce noise through proper shielding and grounding.
  - Ensure signal integrity with good PCB layout and impedance matching.
  - Use synchronization techniques and proper timing analysis.
- **Fix**:
  - Implement filters or debounce circuits.
  - Adjust timing parameters and logic levels.
  - Update firmware or software to address underlying issues.