INSTITUTE OF AERONAUTICAL ENGINEERING

Department of Electronics and Communication Engineering

Name: S. VENKATA SRI NAIMISHA

Roll Number:23951A04P4

Course: Digital System Design

Course Code: AECD03

1. Explain the Binary codes with examples?

Binary codes are very important in digital system design since they are the basis of representing and manipulating data in digital systems. Binary codes encode information such as numbers, characters, and instructions into formats that digital circuits can process.

Explaining binary codes in the context of digital system design with examples follows:

1. Number Systems in Digital Design
Digital systems represent numbers and process data using the binary number system because digital circuits intrinsically work with two states: high (1) and low (0) voltage levels.

Example: Representing Numbers
Decimal 10 in binary: 1010
Decimal 25 in binary: 11001
Digital circuits use adders, multiplexers, and logic gates to process these binary representations.

2. Common Binary Codes in Digital Systems
a. Binary Coded Decimal (BCD)
BCD represents decimal numbers in the binary form where each decimal digit has 4 binary bits representation.

Example: Decimal 93 in BCD:
9 → 1001
3 → 0011
BCD Code: 1001 0011
BCD is used in various devices such as calculators, digital clocks, and display systems.

2.What is unit-distance code? State where they are used.

A unit-distance code is a type of binary code system that differs by only one bit between successive code values.
This makes it less prone to errors when transitioning between

states, particularly in noisy environments
or where the possibility of simultaneous changes to bits may lead to ambiguity.

Applications of Unit-Distance Codes:

Rotary Encoders: Gray code is applied in rotary encoders for measuring the angular position of a rotating shaft.
It provides error-free readings since only one bit changes at a time during transitions.
Finite State Machines (FSMs), Counters,
Digital Signal Processing

Used in signal quantization where smooth transition between states is required.
Less Glitches: A glitch in digital
circuits occur when more than one bits change simultaneously. Unit-distance code avoids this by changing one bit at a time.
Improved robustness: Especially useful where the environment is noisy and also in critical systems for which accurate state transitions are very much required.
Simplified hardware design: Decoding and state
detection is done easily because of the single bit difference

3.Define data selector? How many selection lines are used for n bit data selector?

A data selector, also called a multiplexer (MUX), is a combinational
circuit which selects one of many input data lines and feeds them on to a single output line. A set of control (selection) lines controls the input line selected.

Selection Lines Formula
The number of selection lines m needed to choose one input from n inputs
is given by:
$n = 2m$
Here,
n = Number of input lines
m = Number of selection lines
Explanation: Each combination of

m selection lines can uniquely identify one of the n input lines.

4. Compare serial adder and parallel adder

Serial adders and parallel adders are two types of circuits used for binary addition, each with distinct characteristics tailored to specific applications.

1. Operation Mechanism

- **Serial Adder**: Operates bit by bit. It processes one pair of corresponding bits from the input numbers at a time, starting from the least significant bit (LSB). The carry from each addition is stored in a flip-flop and used in the next step. This process continues until all bits are added, requiring multiple clock cycles.
- **Parallel Adder**: Processes all bits of the input numbers simultaneously. Each pair of bits, along with the carry from the previous stage, is fed into a separate full adder circuit. The addition is completed in a single clock cycle.

2. Speed

- **Serial Adder**: Slower because it requires multiple clock cycles, one for each bit of the operands.
- **Parallel Adder**: Faster as all bits are added simultaneously in a single clock cycle.

3. Hardware Complexity

- **Serial Adder**: Simpler in design and requires fewer components. It uses a single full adder and a flip-flop to store the carry between steps.
- **Parallel Adder**: More complex, needing a full adder circuit for each bit of the input numbers. For n-bit addition, nnn full adders are required.

4. Carry Propagation

- **Serial Adder**: Propagates the carry bit serially from one stage to the next, which contributes to its slower speed.
- **Parallel Adder**: Propagates the carry across all bits in parallel. In designs like ripple-carry adders, the carry moves sequentially through stages, but in more advanced designs like carry-lookahead adders, this propagation is significantly sped up.

5. Area and Power Consumption

- **Serial Adder**: Occupies less area on the chip and consumes less power due to its minimal hardware requirements.

- **Parallel Adder**: Occupies more area and consumes more power because it requires multiple full adders and complex wiring for simultaneous operations.

6. Applications

- **Serial Adder**: Suitable for low-speed, resource-constrained applications where simplicity and cost are priorities, such as in small embedded systems.
- **Parallel Adder**: Used in high-speed systems like processors, arithmetic logic units (ALUs), and digital signal processing (DSP) units, where performance is critical.

7. Example

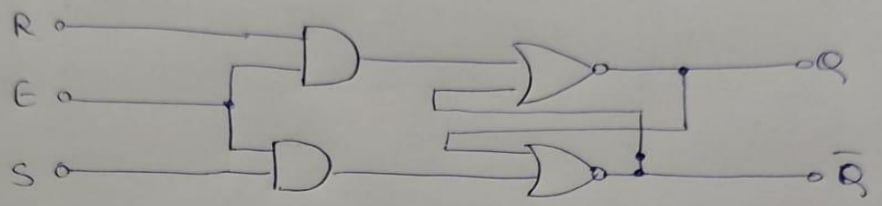
To add two 4-bit binary numbers $A=1011A = 1011A=1011$ and $B=0110B = 0110B=0110$:

- **Serial Adder**: Adds one pair of bits per clock cycle, taking 4 cycles to complete.
- **Parallel Adder**: Adds all bits simultaneously and completes the addition in 1 clock cycle.

5. Draw the characteristic table of gated SR-Latch?

Gated SR latch (or) Clocked SR Flip flop.

Circuit diagram:



Truth table:

E	S	R	Q	$\bar{Q}$
0	0	0	latch	latch
0	0	1	latch	latch
0	1	0	Latch	catch
0	1	1	catch	catch
1	0	0	catch	catch
1	0	1	0	1
1	1	0	1	0
1	1	1	0	0

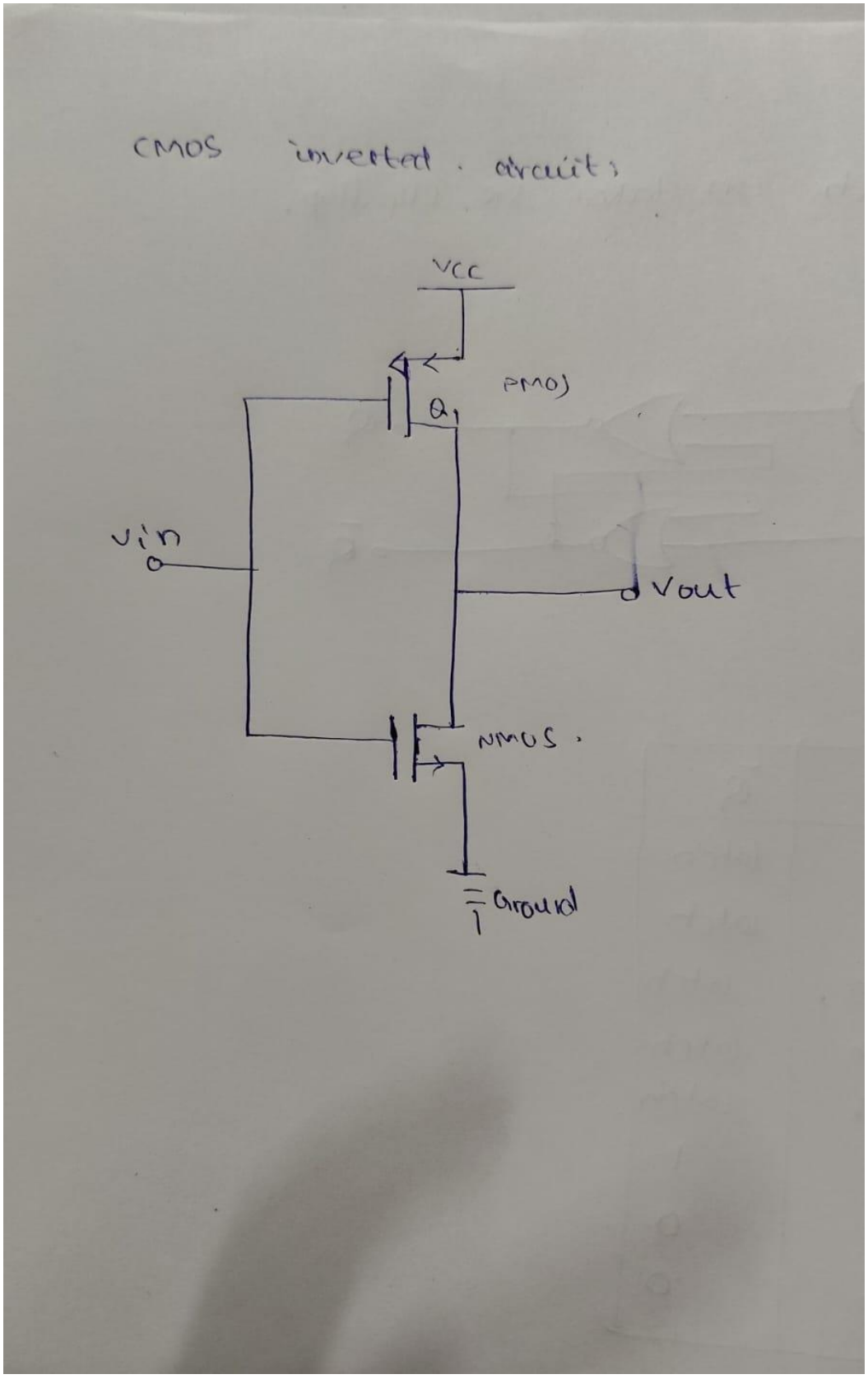
6. Define Stable State.

In the context of digital systems and circuits, a **stable state** refers to a condition in which a circuit or system remains in a specific state until an external input or event causes it to change. Once the system reaches a stable state, it does not exhibit any unintended oscillations or transitions.

Importance of Stable States

1. **Reliability**: Ensures digital systems function predictably.
2. **Data Storage**: Used in memory cells, registers, and storage devices to hold data.
3. **Synchronization**: Critical for timing and coordination in sequential circuits.

7. Draw the CMOS inverter circuit.



8. What would happen if three-state outputs turned on faster than they turned off?

Three-state outputs can exist in one of three states:

1. **Logic High (1)**: Actively drives the output high.
2. **Logic Low (0)**: Actively drives the output low.
3. **High Impedance (Z)**: Disconnects from the circuit, effectively "off," allowing other devices to use the bus.

They are commonly used in shared bus systems to ensure only one device drives the bus at a time.

Effects of Turning On Faster than Turning Off

1. **Bus Contention**:
    - If one output turns on before another output has fully turned off (entered the high-impedance state), both outputs may simultaneously drive the bus to conflicting logic levels (e.g., one driving high and the other driving low).
    - This causes **current to flow directly between the conflicting outputs**, leading to:
        - Increased power dissipation.
        - Potential overheating and damage to components.
2. **Glitches or Erroneous Signals**:
    - During bus contention, the voltage on the bus may fluctuate unpredictably between high and low levels.
    - These glitches can cause incorrect data to be interpreted by other devices connected to the bus.
3. **Reduced Circuit Reliability**:
    - Prolonged bus contention stresses components and can degrade their performance over time.
    - It can also lead to transient errors in critical applications.

4. **Power Supply Issues**:
    - The high current flow due to contention may cause power supply instability, leading to further operational issues across the system.

9. Write HDL behavioral description of JK flipflop and D flipflop using if- else statement based on value of present state

The JK Flip-Flop toggles its state based on the inputs JJJ and KKK, and the clock signal.

```
module JK_FlipFlop (
    input wire clk,
    input wire reset,
    input wire J,
    input wire K,
    output reg Q
);
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            Q <= 1'b0;
        end else begin
            if (J == 0 && K == 0) begin
                Q <= Q; // No change
            end else if (J == 0 && K == 1) begin
                Q <= 1'b0; // Reset
            end else if (J == 1 && K == 0) begin
                Q <= 1'b1; // Set
            end else if (J == 1 && K == 1) begin
                Q <= ~Q; // Toggle
            end
        end
```

```
    end
endmodule
```

The D Flip-Flop simply transfers the value of the DDD input to the output QQQ on the clock edge.

```
module D_FlipFlop (
    input wire clk,
    input wire reset,
    input wire D,
    output reg Q
);
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            Q <= 1'b0; // Reset the flip-flop
        end else begin
            Q <= D; // Transfer D to Q
        end
    end
endmodule
```

10. Give the comparison between PAL and PLA.

PAL (Programmable Array Logic) and PLA (Programmable Logic Array) are both types of programmable logic devices used in digital circuits. They allow designers to configure logic gates to perform custom logic functions. However, there are significant differences in terms of flexibility, complexity, and structure.

1. Structure

- **PAL (Programmable Array Logic)**:
    - **AND Array**: Programmable.

    - **OR Array**: Fixed (not programmable).
    - In a PAL device, the AND array can be programmed to create any combination of inputs, but the OR array is fixed. This means that the outputs of the AND gates are directly connected to the OR gates, and the OR array is static for all configurations.
- **PLA (Programmable Logic Array)**:
    - **AND Array**: Programmable.
    - **OR Array**: Programmable.
    - In a PLA, both the AND array and the OR array are fully programmable, allowing more flexibility in creating custom logic functions. This makes the PLA more versatile than the PAL.

---

2. Flexibility

- **PAL**:
    - **Less Flexible**: The fixed OR array limits the types of logic that can be implemented. PALs are typically used for simpler logic functions where only the AND array needs customization.
    - Limited to a predefined number of output terms.
- **PLA**:
    - **More Flexible**: The programmability of both the AND and OR arrays gives greater flexibility, making PLAs suitable for more complex logic functions. Designers can program both the input combinations and the way the outputs are generated.
    - Can support more complex logic expressions.

---

3. Complexity

- **PAL**:
    - **Lower Complexity**: PALs are simpler devices since the OR array is fixed, making them easier to use and configure. They are often used for applications where the logic requirements are not very complex.

    - PAL devices typically have fewer inputs and outputs compared to PLAs.

- **PLA**:
    - **Higher Complexity**: PLAs are more complex due to their fully programmable nature, which allows for more intricate logic functions. This added complexity can be advantageous for implementing more advanced and customized logic, but it also makes them harder to design and program.

---

4. Speed

- **PAL**:
    - **Faster**: PALs can be faster because of the simpler structure (fixed OR array). The fixed OR array reduces the complexity of the routing and speeds up the propagation of signals.
- **PLA**:
    - **Slower**: PLAs are generally slower due to the greater complexity of the programmable OR array, which adds additional delay in the circuit.

---

5. Cost

- **PAL**:
    - **Lower Cost**: Because of the fixed OR array and simpler structure, PALs are usually less expensive to manufacture. This makes them a cost-effective solution for simpler designs.
- **PLA**:
    - **Higher Cost**: Due to their flexibility and fully programmable nature, PLAs tend to be more expensive. The added complexity increases the manufacturing cost, but they are suitable for applications where high customization is needed.

---

6. Applications

- **PAL**:

    - **Used for Simpler Applications**: PALs are typically used in applications where logic functions are straightforward and do not require extensive customization. Examples include simple combinational logic circuits, address decoders, or small control units.
- **PLA**:
    - **Used for Complex Applications**: PLAs are used for more complex digital designs that require extensive customization. Examples include more complex state machines, custom logic circuits, and larger control units.