CSCI 2020 Computer Architecture

Basic binary math and logic

Binary numbers operate in base 2, where the only digits are `0` and `1`. They are fundamental in digital systems, logic design, and computing.

1. **Binary Arithmetic: Addition:** Binary addition is similar to decimal addition, with the rules:

2. **Binary Subtraction**: **two's complement(invert bits add 1)** of what is subtracting, add what is being taken from and discard carry

Logical Operations:

- **AND (`A * B`)**: Outputs `1` only if both inputs are `1`
- **OR (`A + B`)**: Outputs `1` if at least one input is `1`.
- **XOR (`A ⊕ B`)**: Outputs `1` if the inputs are different.
- **NOT (`¬A`)**: Inverts the input.
- **NAND (`¬(A * B)`)**: Outputs `1` unless **both inputs** are `1`.
- **NOR (`¬(A + B)`)**: Outputs `1` only if **both inputs** are `0`.
- **XNOR (`¬(A ⊕ B)`)**: Outputs `1` if the **inputs are the same**.
- **Implication (`A → B`)**: Outputs `1` unless `A = 1` and `B = 0`.

CSCI 2020 Computer Architecture (cont)

- **Equivalence (`A ↔ B`)**: Outputs `1` if **A and B are equal**. Equivalent to XNOR.

Total bit Combinations = 2^n

Dec → Hex | Divide by 16, use remainders as hex digits. | 156 (dec) | 9C (hex)

Hex → Dec | Multiply digits by 16^n, sum the results. | 1A3 (hex) | 419 (dec)

Bin → Dec | Multiply bits by 2^n, sum the results. | 1101 (bin) | 13 (dec)

Dec → Oct | Divide by 8, use remainders as octal digits. | 125 (dec) | 175 (oct)

Bin → Oct | Group binary into 3 bits, convert each group to octal. | 110101 (bin) | 65 (oct)

Sign Representation MSB(the front of bit): `0` (positive), `1` (negative) Negative: Invert and add 1 | | **Arithmetic** | Complex | Simplified (no separate subtraction) | | **Range (4-bit)** | (signed Magnitude(\(-7 \) to \(+7 \) | (two's complement)\(-8 \) to \(+7 \) |

Combinational Logic

Combinational logic

Definition

- **Combinational Circuit**: Output depends only on current inputs; no memory.
- **Sequential Circuit**: Output depends on current inputs and past states (memory)

Key Differences

Feature	Combinational Circuit	Sequential Circuit
Output Dependency	Current inputs only	Current inputs + past states
Memory	No memory	Has memory (e.g., flip-flops)
Feedback	No feedback loop	Includes feedback loop
Clock Signal	Not required	Requires a clock signal
Time Dependency	Outputs appear immediately	Outputs depend on clock cycles

Examples

- **Combinational Circuit**:

Combinational Logic (cont)

- Half Adder: Adds two inputs, outputs Sum = \(A XOR B), Carry = \(A AND B).
- **Sequential Circuit**:
- D Flip-Flop: Stores 1 bit of data; updates on clock edge.

Applications

- **Combinational Circuits**:
- Adders, multiplexers, encoders, decoders.
- **Sequential Circuits**:
- Counters, shift registers, memory units.

Summary

- **Combinational Circuits**:
- Simple, stateless, used for logical operations.
- **Sequential Circuits**:
- Complex, state-based, used for time-sensitive or memory-based tasks.

For AND(also minterm) gates

- **Identity**: AND-ing anything with `1` keeps it the same.
- **Null**: AND-ing anything with `0` makes it `0`.
- **Commutative**: Changing the order of inputs doesn't matter.
- **Associative**: Grouping inputs in any way doesn't matter.

-**Minterm** = $A \cdot \overline{B} \cdot C \cdot \overline{D}$

Combinational Logic (cont)

- **sum of products**: Y=(A·B¯·C·D)+(A¯·B·C·D¯)+(A·B·C¯·D¯)

For OR(maxterm) gate

- **Identity**: \(A + 0 = A \)
- **Null**: \(A + 1 = 1 \)
- **Commutative**: \(A + B = B + A \)
- **Associative**: \(A + (B + C) = (A + B) + C \)
- **Idempotent**: \(A + A = A \)
- **Distributive**:
- Over AND: \(A + (B AND C) = (A + B) AND (A + C) \)
- Over OR: (A + (B + C) = (A + B) + C)

-- **Maxterm** = (A¯+B+C¯+D)

- **Product of sums**: Y=(A¯+B+C¯+D)·(A+B¯+C+D¯)·(A¯+B¯+C¯+D)

Prority Encoder

Definition:

A Priority Encoder is a combinational circuit that encodes the position of the highest-priority active input into a binary output.

Key Features:

- **Inputs**: \(2^n \) input lines; only one active at a time is expected.
- **Outputs**: \(n \) output lines representing the binary code of the highest-priority input.

Combinational Logic (cont)

- **Priority**: Higher-order inputs have precedence over lower-order inputs.
- **Enable Output**: Indicates if any input is active (optional).

Logic Expressions (4-to-2 Priority Encoder):

- Y_1 = D_3 + D_2
- Y_0 = D_3 +{D_2}¯ * D_1
- Enable Output (E = D_3 + D_2 + D_1 + D_0

Applications:

1. **Interrupt Handling**: Assigns priority to multiple interrupt signals in processors.
2. **Data Compression**: Encodes multiple inputs into fewer bits.
3. **Memory Decoding**: Selects the highest-priority address or resource.

Advantages:

- Handles multiple inputs with priority logic.
- Compresses input size efficiently.

Limitations:

- Requires additional handling if multiple inputs have the same priority.
- Extra circuitry needed for "no active input" conditions.

Causes of Delay in Circuit State Change (Low to High)

Definition:

Combinational Logic (cont)

Delays occur when a circuit element transitions from `0` (low) to `1` (high) due to physical and electrical factors.

Key Causes of Delay

1. **Propagation Delay**:
- Time taken for a signal to propagate through a circuit element.
- Affects overall circuit speed.
2. **Gate Capacitance**:
- Time required to charge or discharge the transistor's gate capacitance.
3. **Load Capacitance**:
- Higher load capacitance slows the charging/discharging process.
4. **Resistance of Interconnects**:
- Higher resistance in wires increases \(RC \) delay.
5. **Signal Rise Time**:
- Time taken for the signal to rise from 10% to 90% of its final value.
6. **Threshold Voltage**:
- Higher voltage thresholds cause slower transitions.
7. **Noise and Signal Integrity**:
- Crosstalk, interference, or power fluctuations can distort signals.
8. **Power Supply Voltage**:
- Lower voltages reduce drive strength, increasing delay.
9. **Temperature Effects**:

Combinational Logic (cont)

- High temperatures slow down transistor switching.

10. **Manufacturing Variations**:
- Fabrication inconsistencies can result in slower components.
11. **Clock Synchronization**:
- Skew or jitter in clock signals causes timing delays in sequential circuits.
12. **Parasitic Elements**:
- Unintended resistance, capacitance, or inductance contributes to delay.

Summary:

Delays result from intrinsic (e.g., propagation, capacitance) and external factors (e.g., noise, temperature). Optimizing design and materials can mitigate these delays.

Combinational Logic

Combinational logic

Definition

- **Combinational Circuit**: Output depends only on current inputs; no memory.
- **Sequential Circuit**: Output depends on current inputs and past states (memory)

Key Differences

| Feature | Combinational Circuit | Sequential Circuit |

Combinational Logic (cont)

|----------------------|-------------------------------|------------------------------|

| **Output Dependency** | Current inputs only | Current inputs + past states |

| **Memory** | No memory | Has memory (e.g., flip-flops) |

| **Feedback** | No feedback loop | Includes feedback loop |

| **Clock Signal** | Not required | Requires a clock signal |

| **Time Dependency** | Outputs appear immediately | Outputs depend on clock cycles |

Examples

- **Combinational Circuit**:

- Half Adder: Adds two inputs, outputs Sum = \(A XOR B), Carry = \(A AND B).

- **Sequential Circuit**:

- D Flip-Flop: Stores 1 bit of data; updates on clock edge.

Applications

- **Combinational Circuits**:

- Adders, multiplexers, encoders, decoders.

- **Sequential Circuits**:

- Counters, shift registers, memory units.

Summary

Combinational Logic (cont)

- **Combinational Circuits**:

- Simple, stateless, used for logical operations.

- **Sequential Circuits**:

- Complex, state-based, used for time-sensitive or memory-based tasks.

For AND(also minterm) gates

- **Identity**: AND-ing anything with `1` keeps it the same.

- **Null**: AND-ing anything with `0` makes it `0`.

- **Commutative**: Changing the order of inputs doesn't matter.

- **Associative**: Grouping inputs in any way doesn't matter.

-**Minterm** = A·B¯·C·D¯

- **sum of products**: Y=(A·B¯·C·D)+(A¯·B·C·D¯)+(A·B·C¯·D¯)

For OR(maxterm) gate

- **Identity**: \(A + 0 = A \)

- **Null**: \(A + 1 = 1 \)

- **Commutative**: \(A + B = B + A \)

- **Associative**: \(A + (B + C) = (A + B) + C \)

- **Idempotent**: \(A + A = A \)

- **Distributive**:

- Over AND: \(A + (B AND C) = (A + B) AND (A + C) \)

- Over OR: (A + (B + C) = (A + B) + C)

Combinational Logic (cont)

-- **Maxterm** = (A¯+B+C¯+D)

- **Product of sums**: Y=(A¯+B+C¯+D)·(A+B¯+C+D¯)·(A¯+B¯+C¯+D)

Prority Encoder

Definition:

A Priority Encoder is a combinational circuit that encodes the position of the highest-priority active input into a binary output.

Key Features:

- **Inputs**: \(2^n \) input lines; only one active at a time is expected.

- **Outputs**: \(n \) output lines representing the binary code of the highest-priority input.

- **Priority**: Higher-order inputs have precedence over lower-order inputs.

- **Enable Output**: Indicates if any input is active (optional).

Logic Expressions (4-to-2 Priority Encoder):

- Y_1 = D_3 + D_2

- Y_0 = D_3 +{D_2}¯ * D_1

- Enable Output (E = D_3 + D_2 + D_1 + D_0

Applications:

1. **Interrupt Handling**: Assigns priority to multiple interrupt signals in processors.

2. **Data Compression**: Encodes multiple inputs into fewer bits.

Combinational Logic (cont)

3. **Memory Decoding**: Selects the highest-priority address or resource.

Advantages:

- Handles multiple inputs with priority logic.

- Compresses input size efficiently.

Limitations:

- Requires additional handling if multiple inputs have the same priority.

- Extra circuitry needed for "no active input" conditions.

Causes of Delay in Circuit State Change (Low to High)

Definition:

Delays occur when a circuit element transitions from `0` (low) to `1` (high) due to physical and electrical factors.

Key Causes of Delay

1. **Propagation Delay**:

- Time taken for a signal to propagate through a circuit element.

- Affects overall circuit speed.

2. **Gate Capacitance**:

- Time required to charge or discharge the transistor's gate capacitance.

3. **Load Capacitance**:

- Higher load capacitance slows the charging/discharging process.

4. **Resistance of Interconnects**:

Combinational Logic (cont)

- Higher resistance in wires increases \(RC \) delay.

5. **Signal Rise Time**:
- Time taken for the signal to rise from 10% to 90% of its final value.

6. **Threshold Voltage**:
- Higher voltage thresholds cause slower transitions.

7. **Noise and Signal Integrity**:
- Crosstalk, interference, or power fluctuations can distort signals.

8. **Power Supply Voltage**:
- Lower voltages reduce drive strength, increasing delay.

9. **Temperature Effects**:
- High temperatures slow down transistor switching.

10. **Manufacturing Variations**:
- Fabrication inconsistencies can result in slower components.

11. **Clock Synchronization**:
- Skew or jitter in clock signals causes timing delays in sequential circuits.

12. **Parasitic Elements**:
- Unintended resistance, capacitance, or inductance contributes to delay.

Summary:

Combinational Logic (cont)

Delays result from intrinsic (e.g., propagation, capacitance) and external factors (e.g., noise, temperature). Optimizing design and materials can mitigate these delays.

Sequential Logic

State Table for A, B, Clock and XOR gate what is Q?

To determine the contents of the output register Q, we'll use a **state table** that describes the relationship between the inputs A, B, the clock, and the XNOR gate, along with the resulting Q.

XNOR Gate Logic

- The output of an XNOR gate is A↔B (logical equivalence):
- $A \oplus B = 0$: $\overline{(A \oplus B)} = 1$(when A = B)
- $A \oplus B=1$ $\overline{(A \oplus B)} = 0$: (when $A \neq B$)

The output Q of the register will update on every clock edge based on the XNOR output.

State Table

Assume:
- Inputs A and B are parallel inputs (change at each clock cycle).
- Register Q is updated at every **positive clock edge** with the XNOR output.