Intro to Digital Logic

Combinational Logic

By Apowware

Ref. Digital Design Principles & Practices (4th Ed.) John F. Wakerly

Introduction

In digital electronics, circuits are broadly categorized into two types, combinational and sequential.

Combinational logic circuits are digital circuits whose output depends **only** on the current state of their inputs. They do not have memory elements, meaning their past inputs or outputs do not affect their current output. Think of them as instantaneous function generators: as soon as the inputs change, the outputs change accordingly, after a very small propagation delay.

Key characteristics:

- Outputs are a direct function of current inputs.
- No memory elements (e.g., flip-flops).
- No feedback loops.
- Can be described by Boolean expressions, truth tables, or logic gates.

Encoders

An encoder is a combinational circuit that converts an active input signal into a coded output, typically a binary or BCD (Binary Coded Decimal) code.

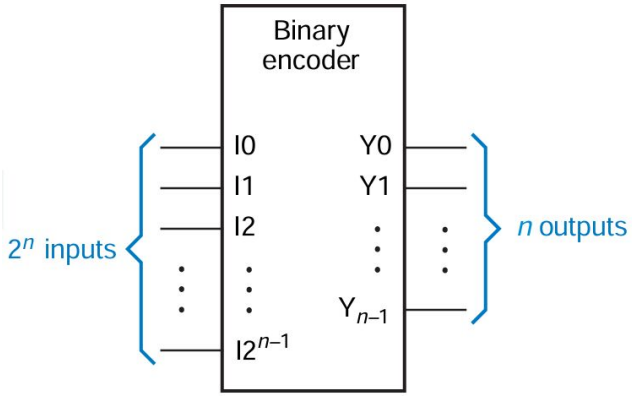
Types:

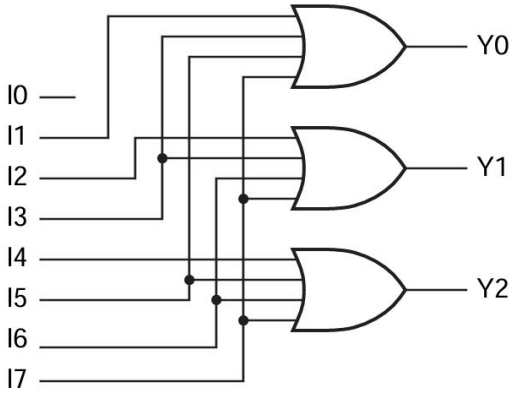
- **Decimal-to-BCD Encoder:** Converts a decimal input (0-9) to a 4-bit BCD output.
- **Priority Encoder:** This is a more practical type. If multiple inputs are active simultaneously, it prioritizes the highest-numbered active input and produces its corresponding code. This is crucial to avoid ambiguous outputs.

$Y0 = I1 + I3 + I5 + I7$

$Y1 = I2 + I3 + I6 + I7$

$Y2 = I4 + I5 + I6 + I7$





Example: Simple Decimal-to-Binary Encoder

Consider an encoder with 4 inputs (I_0, I_1, I_2, and I_3) and 2 outputs (O_1,O_0). If only one input is active at a time:

Here, $O_0=I_1+I_3$ and $O_1= I_2+I_3$

I_3	I_2	I_1	I_0	O_1	O_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

Example Decimal-to-BCD Encoder

This encoder has 10 inputs (to D9), one for each decimal digit, and 4 outputs (B3,B2,B1,B0) representing the 8421 BCD code. Only one input is assumed to be active (HIGH) at a time.

The Boolean expressions for the outputs would be:

- B0=D1+D3+D5+D7+D9
- B1=D2+D3+D6+D7
- B2=D4+D5+D6+D7
- B3=D8+D9

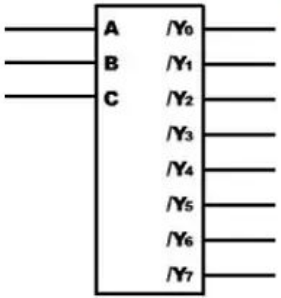
D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	B3	B2	B1	B0
0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	1
0	0	0	0	0	0	0	1	0	0	0	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0	0	0	0	1	0	1
0	0	0	1	0	0	0	0	0	0	0	1	1	0
0	0	1	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	0	0	0	0	0	0	1	0	0	1

Decoder

A decoder is a combinational circuit that converts a binary code input into a unique output line. For an n-bit input, a decoder can have up to 2^n output lines. Only one output line is active (high or low, depending on the design) for each unique input combination.

Types:

- **Binary Decoder:** Converts an n-bit binary input to 2n unique output lines.
- **BCD-to-Decimal Decoder:** Converts a 4-bit BCD input to one of ten decimal output lines.



Input			Output							
A	B	C	/Y0	/Y1	/Y2	/Y3	/Y4	/Y5	/Y6	/Y7
L	L	L	L	H	H	H	H	H	H	H
H	L	L	H	L	H	H	H	H	H	H
L	H	L	H	H	L	H	H	H	H	H
H	H	L	H	H	H	L	H	H	H	H
L	L	H	H	H	H	H	L	H	H	H
H	L	H	H	H	H	H	H	L	H	H
L	H	H	H	H	H	H	H	H	L	H
H	H	H	H	H	H	H	H	H	H	L

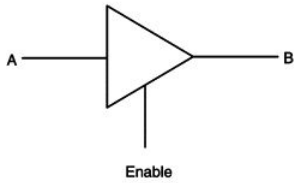
Three-State (Tri-State) Devices

A three-state device (often a buffer or inverter) has a third output state in addition to the standard logic HIGH (1) and logic LOW (0): the **high-impedance (Hi-Z)** state.

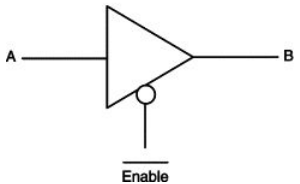
In the Hi-Z state, the output effectively disconnects from the circuit, acting like an open circuit. This means it neither drives the line high nor low, and it draws very little current.

Symbol: A standard buffer or inverter symbol with an additional enable input.

- When **Enable is active** (e.g., HIGH for active-high enable), the device behaves like a normal buffer/inverter (output = input or output = NOT input).
- When **Enable is inactive** (e.g., LOW for active-high enable), the output goes to the Hi-Z state.



Enable	A	B
0	0	Z
0	1	Z
1	0	0
1	1	1



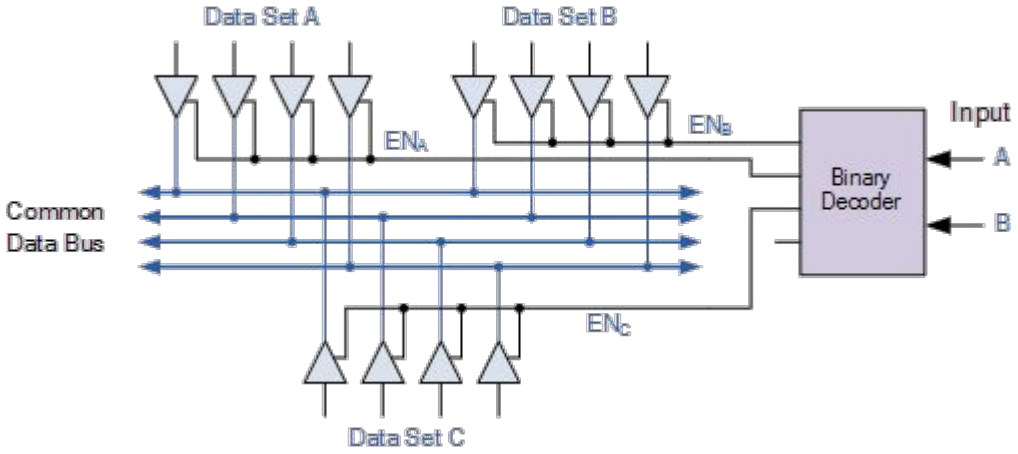
$\overline{\text{Enable}}$	A	B
0	0	0
0	1	1
1	0	Z
1	1	Z

Bus Driving with Tri-State Buffers

Imagine a shared data bus in a computer system where multiple components (e.g., CPU, Memory, I/O device) need to send data. If all components tried to output data simultaneously, it would lead to contention and damage. Tri-state buffers solve this:

- Each component connects to the bus via a tri-state buffer.
- A **bus controller** (another logic circuit) ensures that only **one** component's tri-state buffer is enabled at any given time.
- All other components' tri-state buffers are in the Hi-Z state, effectively disconnecting them from the bus, allowing the enabled component to drive data without interference.

This allows multiple devices to share the same physical wires without short-circuiting each other.

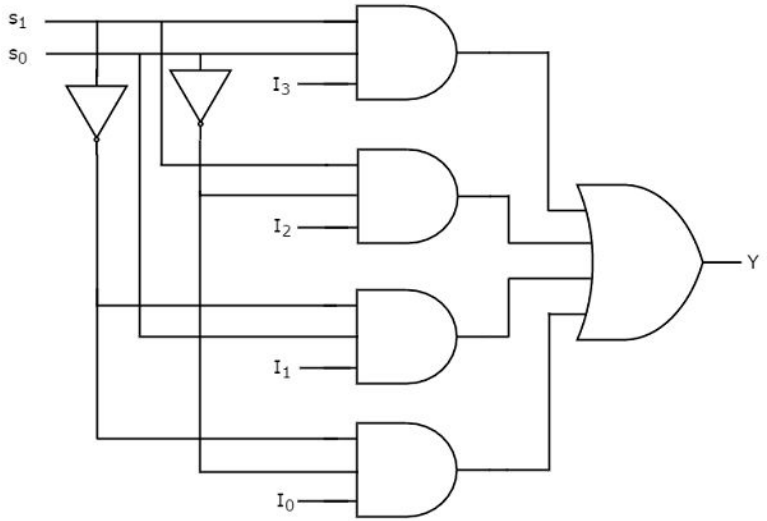


Multiplexers (Mux)

A multiplexer, or Mux, is a data selector. It takes multiple data inputs and selects one of them to be passed through to a single output line. The selection is controlled by a set of **select lines**.

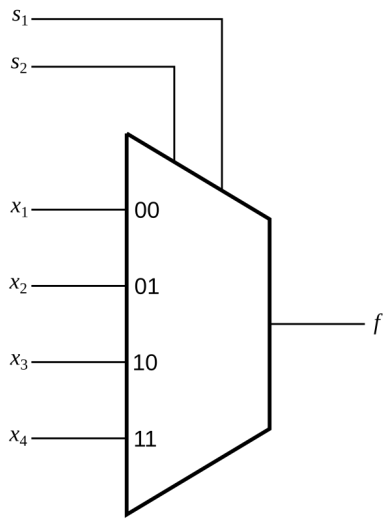
For N data inputs, a Mux requires $\log_2 N$ select lines.

- An 8-to-1 Mux has 8 data inputs and 3 select lines ($2^3=8$).
- A 4-to-1 Mux has 4 data inputs and 2 select lines ($2^2=4$).



Example 4 to 1 Mux

S_1	S_0	F
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3



Parity Circuits using XOR Gates

Parity is a simple error detection method used in digital communication and storage. A parity bit is an extra bit added to a binary word to make the total number of 1s in the word (including the parity bit) either even or odd.

- **Even Parity:** The parity bit is set such that the total number of 1s is even.
- **Odd Parity:** The parity bit is set such that the total number of 1s is odd.

XOR Gate for Parity Generation: The XOR gate is perfect for parity generation because it outputs a 1 if an odd number of its inputs are 1, and a 0 if an even number of its inputs are 1.

- For **Even Parity Generation**: XOR all data bits. The output of the XOR chain is the parity bit. If the data has an even number of 1s, the XOR output is 0 (making total 1s even). If odd, XOR output is 1 (making total 1s even).
  - $P_{even}=D_3 \oplus D_2 \oplus D_1 \oplus D_0$
- For **Odd Parity Generation**: XOR all data bits, then invert the result.
  - $P_{odd}=(D_3 \oplus D_2 \oplus D_1 \oplus D_0)'$

Example Parity Generation & Checking

Let's consider a 4-bit data word D=1011.

Even Parity Generation: $P_{even}=1\oplus0\oplus1\oplus1=(1\oplus0)\oplus(1\oplus1)=1\oplus0=1$. The transmitted data with even parity would be 10111 (Data + Parity). Let's check: The number of 1s in 10111 is four, which is an even number. Correct.

Even Parity Checking (Receiver Side): Suppose the received word is 10111. Check bit $C=1\oplus0\oplus1\oplus1\oplus1=(1\oplus0)\oplus(1\oplus1)\oplus1=1\oplus0\oplus1=1\oplus1=0$. Since C=0, it indicates no error in transmission (for single-bit errors).

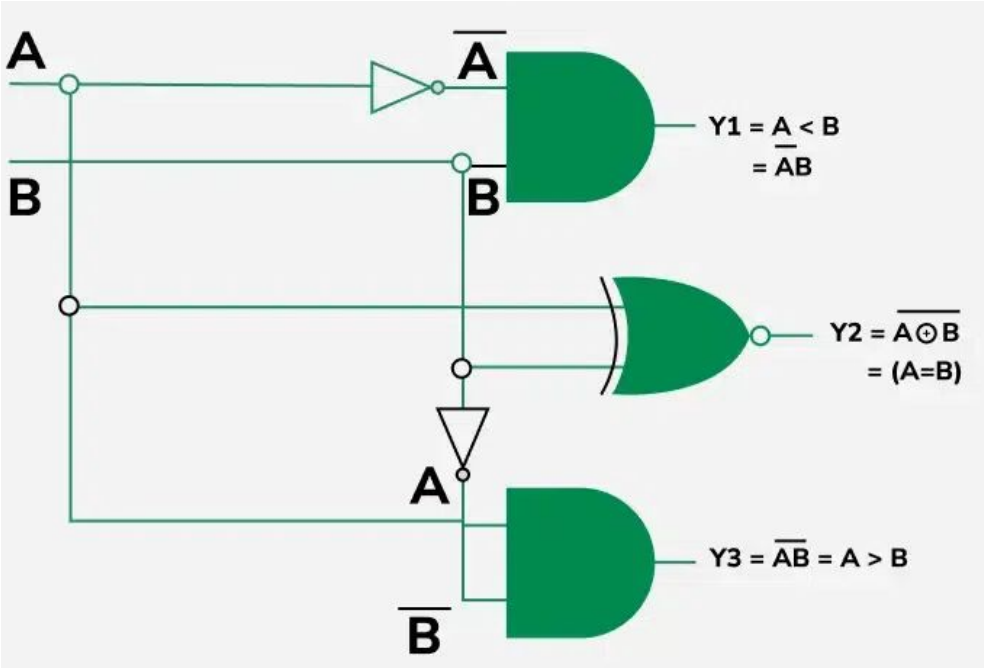
Now, suppose a single-bit error occurs and the received word is 10011 (the third bit changed from 1 to 0). Check bit $C=1\oplus0\oplus0\oplus1\oplus1=(1\oplus0)\oplus(0\oplus1)\oplus1=1\oplus1\oplus1=0\oplus1=1$. Since C=1, it indicates an error has occurred.

Comparators

A comparator is a combinational circuit that compares two binary numbers and determines their relationship: whether one is greater than, less than, or equal to the other.

Types:

- **Equality Comparator:** Outputs a HIGH if the two numbers are equal, LOW otherwise.
- **Magnitude Comparator:** Outputs HIGH for one of three conditions: A>B, A<B, or A=B.



Adders

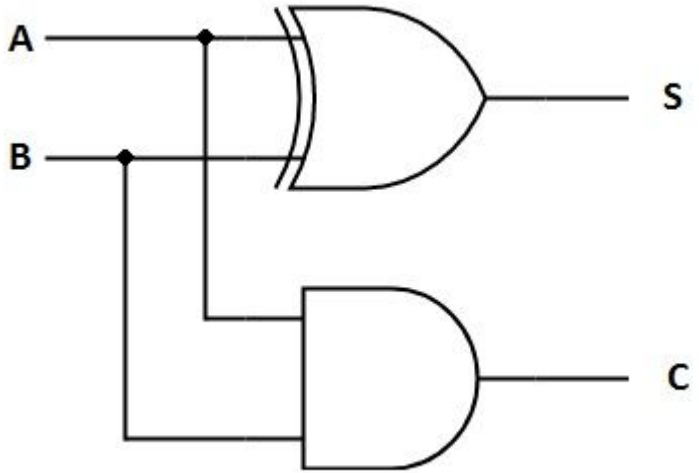
Adders are fundamental combinational circuits used to perform arithmetic addition of binary numbers.

Types:

- **Half Adder (HA):** Adds two single binary digits (A and B). It produces two outputs:
  - **Sum (S):** $S=A \oplus B$
  - **Carry Out (C_{out}):** $C_{out}=A \cdot B$
- **Full Adder (FA)::** Adds three single binary digits (A, B, and a Carry In, C_{in}). It produces two outputs:
  - **Sum (S):** $S=A \oplus B \oplus C_{in}$
  - **Carry Out (Cout):** $C_{out}=(A \cdot B)+(C_{in} \cdot (A \oplus B))$ or $C_{out}=AB+AC_{in}+BC_{in}$

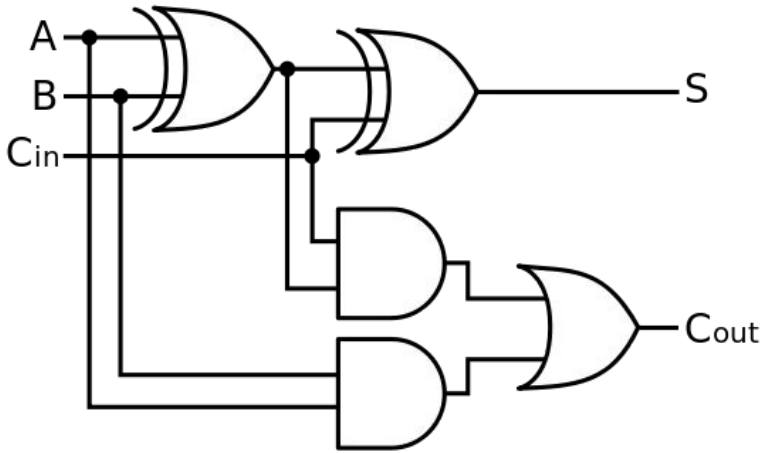
Half Adder

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



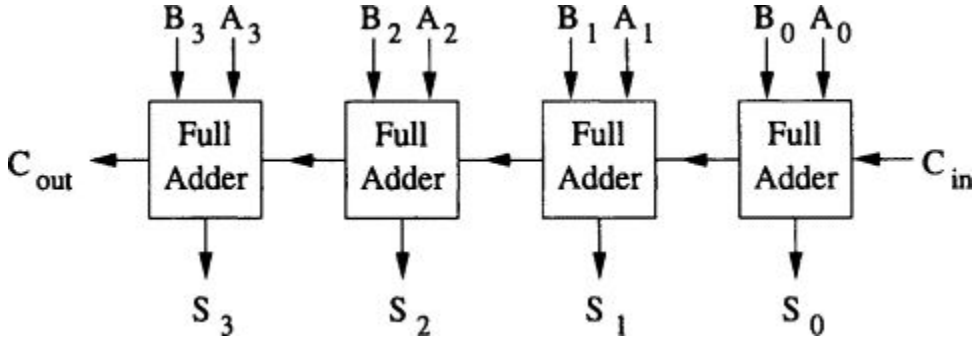
Full Adder

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Ripple-Carry Adder

Multiple full adders can be cascaded to add multi-bit numbers. For example, to add two 4-bit numbers ($A_3A_2A_1A_0$ and $B_3B_2B_1B_0$), you would use four full adders. The carry-out (C_{out0}) from the least significant full adder (adding A_0,B_0,C_{in0}) becomes the carry-in (C_{in1}) for the next full adder (adding A_1,B_1,C_{in1}), and so on. This is simple to implement but can be slow for large numbers due to the "ripple" effect of the carry propagating through each stage.



Arithmetic Logic Unit (ALU)

An ALU is a core component of a CPU (Central Processing Unit) that performs arithmetic and logical operations on binary data. It's a complex combinational circuit that typically includes:

- **Arithmetic operations:** Addition, subtraction, increment, decrement, etc.
- **Logical operations:** AND, OR, NOT, XOR, etc.
- **Shift operations:** Left shift, right shift.

An ALU takes two input operands (e.g., A and B), and a set of **function select lines** that determine which operation the ALU will perform. It also has status outputs like Carry, Zero, Negative, Overflow flags.

Simplified ALU Structure & Operation Example

Consider a simple 4-bit ALU that can perform ADD, SUBTRACT, AND, and OR operations. It would have:

- Two 4-bit data inputs: A[3:0] and B[3:0]
- Two function select lines: S_1,S_0
- A 4-bit output: F[3:0]
- Status flags: Carry Out (Cout), Zero (Z), etc.

The internal structure might involve:

1. A 4-bit adder/subtractor circuit.
2. A 4-bit bitwise AND gate array.
3. A 4-bit bitwise OR gate array.
4. A 4-to-1 multiplexer for each output bit. The select lines of these multiplexers would be controlled by S_1,S_0.

If S_1S_0=00, the multiplexers would select the output from the AND gate array to be the ALU's final output. If S_1S_0=10, the output from the adder would be selected.

S_1	S_0	OP
0	0	A AND B
0	1	A OR B
1	0	A + B
1	1	A - B