**ZenYukti**
*Knowledge. Simplified.*
*Study Smarter, Not Harder*

# Unit II — Combinational Logic: MSI Devices

- Combinational Logic: output depends ONLY on current inputs — no memory or feedback
- MSI (Medium Scale Integration): ICs containing 12–100 gates on one chip
- Key devices covered: Comparator, MUX, DEMUX, Decoder, Encoder, Adders, Subtractors

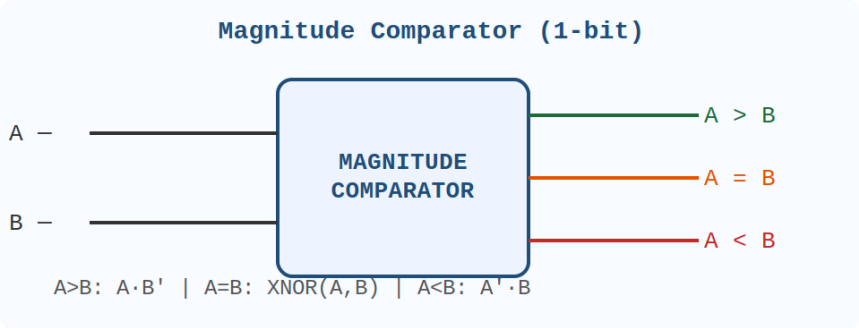
> **"In combinational circuits, the same input always produces the same output — instantly, with no stored state."**

## 1. Magnitude Comparator

- Compares two binary numbers A and B → gives three outputs: A > B, A = B, A < B
- Works bit-by-bit from MSB to LSB; first differing bit decides the result

### 1-bit Comparator — Boolean Equations

- A > B → $A \cdot B'$
- A = B → A XNOR B = $A'B' + AB$
- A < B → $A' \cdot B$




### 4-bit Comparator — IC 7485

- Compares two 4-bit numbers; evaluates pairs from bit 3 (MSB) down to bit 0 (LSB)
- Has cascading inputs (IA>B, IA=B, IA<B) — connect outputs of lower chip to inputs of higher chip
- Chain two IC 7485 chips → 8-bit comparison; four chips → 16-bit comparison

📌 Example: Compare A = 1011 (11) and B = 1001 (9) → A > B output = HIGH

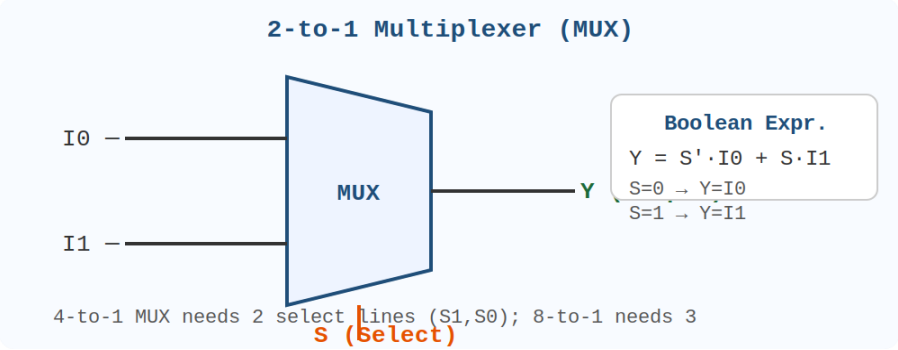
| A | B | A > B | A = B | A < B |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 |

## 2. Multiplexer (MUX)

- MUX = Many-to-One selector: picks ONE of N inputs and routes it to a single output
- A $2^n$-to-1 MUX requires exactly n select lines

### Boolean Expressions

- 2-to-1 MUX : Y = S'·I0 + S·I1
- 4-to-1 MUX : Y = S1'S0'·I0 + S1'S0·I1 + S1S0'·I2 + S1S0·I3




### Common ICs

- IC 74151 → 8-to-1 MUX (3 select lines)
- IC 74153 → Dual 4-to-1 MUX (2 select lines each)

### Applications

- Data routing between multiple sources and one destination
- Parallel-to-serial data conversion
- Implementing any Boolean function (universal logic element)

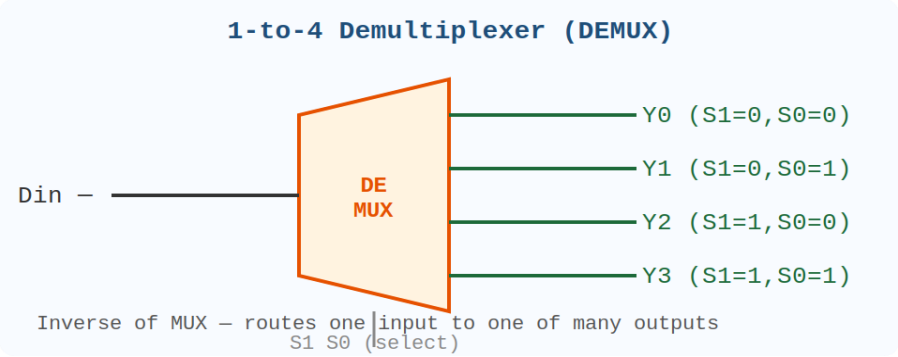
> 📌 Example: 4-to-1 MUX with S1=0, S0=1 → selects I1 as output, regardless of I0, I2, I3
>
> 💡 A $2^n$-to-1 MUX can implement ANY Boolean function of n variables — just connect minterms to inputs.

# 3. Demultiplexer (DEMUX)

- DEMUX = One-to-Many distributor: routes ONE input signal to ONE of N output lines
- Exact inverse operation of a MUX
- 1-to-4 DEMUX → 2 select lines;  1-to-8 DEMUX → 3 select lines

## Output Equation

- $Y_i$ = Din · (select combination for output i)
- Only the selected output follows Din; all others remain 0




## Common ICs

- IC 74139 → Dual 1-to-4 DEMUX
- IC 74138 → 1-to-8 DEMUX (also functions as a 3-to-8 decoder)

## Applications

- Data distribution from one source to multiple destinations
- Serial-to-parallel data conversion
- Memory address decoding alongside decoders

> 📌 Example: 1-to-4 DEMUX, Din = 1, S1 = 0, S0 = 1 → Y1 = 1, Y0 = Y2 = Y3 = 0

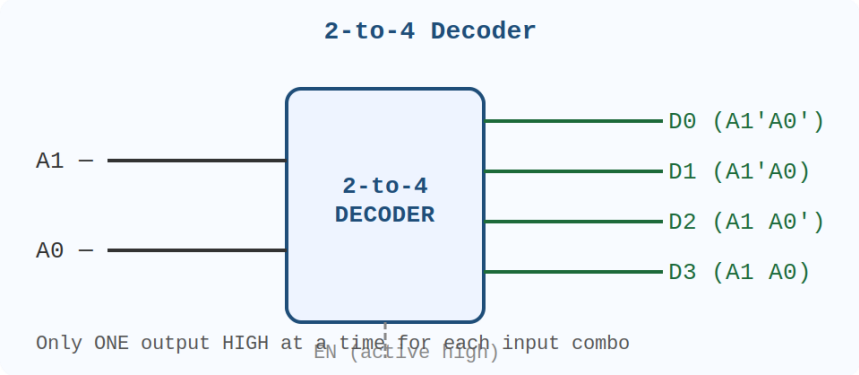
# 4. Decoder

- Converts an n-bit binary input into one of $2^n$ unique output lines (n-to-$2^n$)
- Exactly ONE output is HIGH at any time (active-high) — or LOW (active-low decoder)
- Optional Enable (EN) pin: when EN = 0, all outputs are disabled (= 0)

## 2-to-4 Decoder — Output Equations

- D0 = A1' · A0'  (input = 00)
- D1 = A1' · A0  (input = 01)

- D2 = A1 · A0' (input = 10)
- D3 = A1 · A0 (input = 11)




## Common ICs

- IC 74138 → 3-to-8 decoder (most exam-relevant; has 3 enable pins)
- IC 7442 → BCD-to-Decimal (4-to-10) decoder
- IC 7447 → BCD-to-7-segment decoder (drives 7-segment displays directly)

## Applications

- Memory chip-select logic — activate one memory bank at a time
- CPU instruction decoding — identify which operation to execute
- Seven-segment display driving via BCD decoder (IC 7447)

> 📌 Example: 3-to-8 decoder, input A2A1A0 = 101 → only D5 = HIGH, D0–D4 and D6–D7 = LOW
> 💡 Decoder can implement any combinational function — OR the required minterm outputs together.

# 5. Encoder

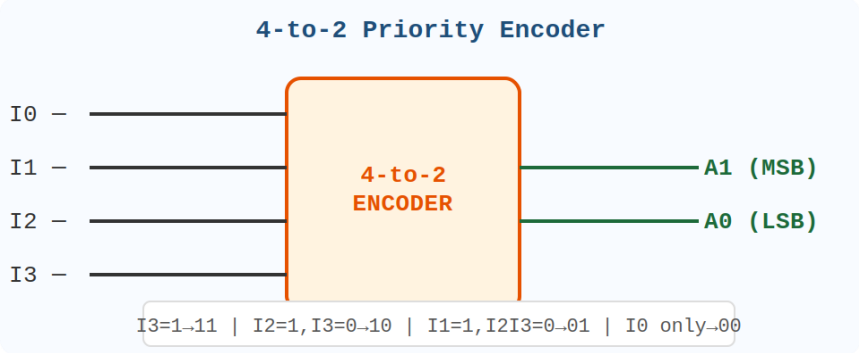
- Inverse of a decoder: converts $2^n$ one-hot inputs into an n-bit binary output code
- Standard encoder limitation: only ONE input must be active at a time

## 4-to-2 Encoder — Output Equations

- A1 = I2 + I3
- A0 = I1 + I3

## Priority Encoder

- Solves the multiple-active-input problem — the highest-indexed active input wins
  - 4-to-2 Priority Encoder: A1 = I3 + I2 A0 = I3 + I1·I2'
  - Has a Valid Output (V) flag: V = 1 when at least one input is active




### Common ICs

- IC 74147 → 10-to-4 BCD priority encoder (decimal keypad → BCD)
- IC 74148 → 8-to-3 octal priority encoder

### Applications

- Keyboard encoding — each key press produces a binary code
- Interrupt priority controllers — highest-priority interrupt gets CPU
- Data compression — encode active lines to fewer bits

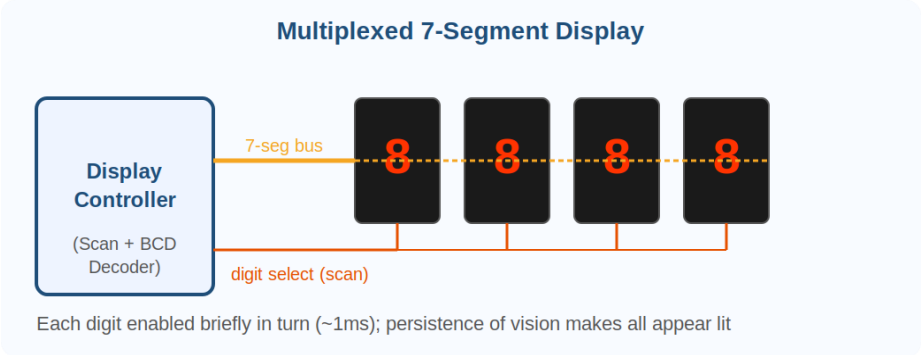
> 📌 Example: 4-to-2 priority encoder: I3 = 1 (highest priority active) → output A1A0 = 11 (regardless of I0, I1, I2)
> 📌 Example: Keyboard encoder: pressing '7' on a decimal keypad → IC 74147 outputs BCD 0111

## 6. Multiplexed Display

- Problem: driving N seven-segment displays statically needs N × 7 segment wires — too many
- Solution: share one set of 7 segment lines; rapidly scan which digit is active

### How It Works — Step by Step

- Step 1: Controller outputs BCD data for Digit 0 on the shared 7-segment bus
- Step 2: Digit 0 common pin is enabled (anode pulled HIGH / cathode pulled LOW)
- Step 3: After ~1 ms, Digit 0 is disabled; data for Digit 1 is placed on the bus
- Step 4: Digit 1 is enabled for ~1 ms; then cycle moves to Digit 2, Digit 3, …
- Step 5: The full scan repeats at ≥ 1 kHz — persistence of vision makes all digits appear lit




### Key Facts

- Refresh rate must exceed 50 Hz per digit to avoid visible flicker
- N digits require only $7 + N$ control lines (vs $7 \times N$ for static driving)
- IC 7447 (BCD-to-7-segment decoder) is used together with digit-select transistors

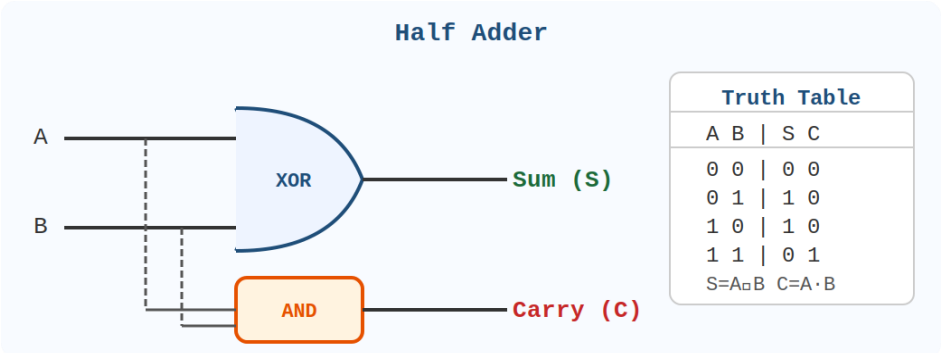
> 📌 Example: 4-digit clock display: scan cycle = 4 ms total; each digit ON for ~1 ms per cycle
> 💡 Multiplexed display is Time-Division Multiplexing (TDM) applied to LEDs — same principle as TDM in communications.

## 7. Half Adder

- Adds two 1-bit numbers A and B; produces Sum (S) and Carry-out (C)
- Cannot accept a carry-in from a previous stage — limited to first bit position

### Boolean Equations

- $S = A \oplus B$ (implemented with XOR gate)
- $C = A \cdot B$ (implemented with AND gate)



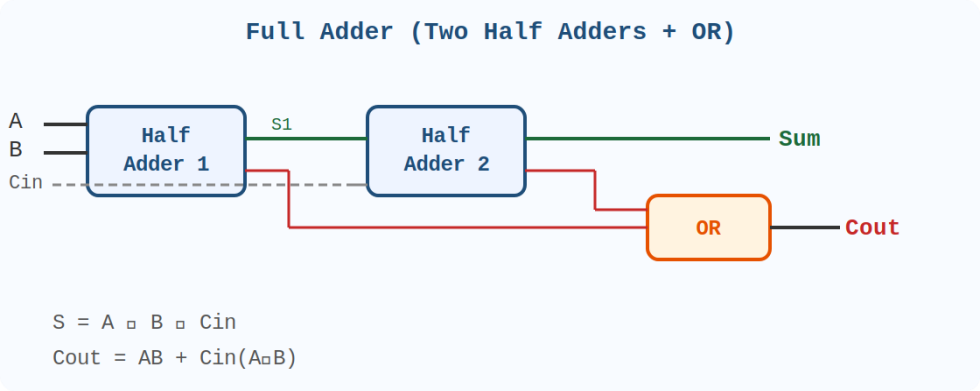

> 📌 Example: A = 1, B = 1 → S = 0, C = 1 (binary: 1 + 1 = 10)

## 8. Full Adder

- Adds three 1-bit inputs: A, B, and Carry-in (Cin) from the previous stage
- Built using two Half Adders and one OR gate

## Boolean Equations

- Sum S = A ⊕ B ⊕ Cin
- Carry Cout = A·B + Cin·(A ⊕ B) [equivalently: AB + BCin + ACin]




| A | B | Cin | Sum (S) | Cout |
|---|---|---|---|---|
| 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 |

# 9. Half Subtractor

- Subtracts B from A (both 1-bit); produces Difference (D) and Borrow-out (Bout)

## Boolean Equations

- D = A ⊕ B
- Bout = A' · B (borrow occurs when A = 0 and B = 1)

> 📌 Example: A = 0, B = 1 → D = 1, Bout = 1 (0 − 1 requires borrowing)

# 10. Full Subtractor

- Computes A − B − Bin (includes Borrow-in from the previous stage)
- Built using two Half Subtractors and one OR gate — mirror of Full Adder structure

## Boolean Equations

- D = A ⊕ B ⊕ Bin
- Bout = A'·B + A'·Bin + B·Bin

> 📌 Example: A = 1, B = 1, Bin = 1 → D = 1, Bout = 1

| Feature | Half Subtractor | Full Subtractor |
|---|---|---|
| Inputs | 2 (A, B) | 3 (A, B, Borrow-in) |
| Outputs | Difference, Borrow-out | Difference, Borrow-out |
| Difference eqn | A ⊕ B | A ⊕ B ⊕ Bin |
| Chainable? | No (no Borrow-in) | Yes (cascades stages) |

# 11. Serial Adder

- Adds two n-bit numbers one bit per clock cycle, starting from the LSB

## Components

- 1 Full Adder (does the actual addition each cycle)
- 2 Shift Registers (hold operands A and B, shift right each cycle)
- 1 D Flip-Flop (stores Carry-out; becomes Carry-in for the next cycle)

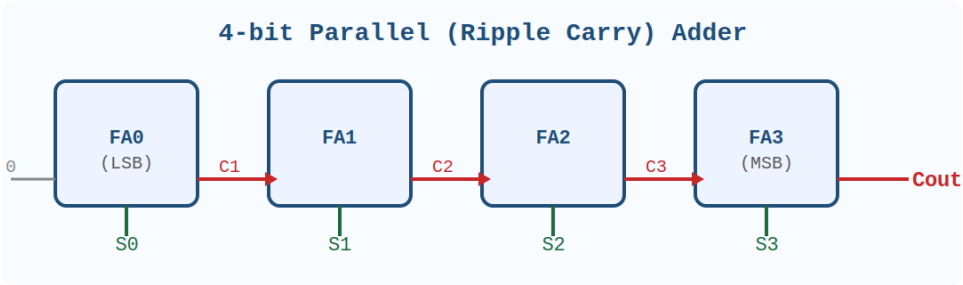
## Operation — Step by Step

- Step 1: Load n-bit numbers A and B into shift registers (LSB at output end)
- Step 2: Clock pulse → LSBs of A and B shift into Full Adder inputs
- Step 3: FA computes Sum bit; stored in result register, and Cout stored in D flip-flop
- Step 4: Next clock → next bits shift in; D flip-flop output is Cin for this cycle
- Step 5: Repeat for all n bits → n clock cycles total

- Advantage: only 1 Full Adder needed regardless of word length — minimal hardware
- Disadvantage: n clock cycles to complete — very slow for large operands

# 12. Parallel (Ripple Carry) Adder

- Adds all n bits of A and B simultaneously using n Full Adders connected in cascade
- Carry-out of each FA becomes the Carry-in of the next — carry 'ripples' left to right




- IC 7483: standard 4-bit parallel binary adder — most common in labs and exams
- Worst-case delay = n × (single FA carry propagation delay) — slow for large n
- Solution for speed: Carry Look-Ahead Adder (CLA) — computes all carries simultaneously

> 📌 Example: 4-bit adder: A = 1010 (10) + B = 0110 (6) → Sum = 10000, Cout = 1 (decimal 16 ✓)

| Feature | Serial Adder | Parallel (Ripple) Adder |
|---|---|---|
| Speed | Slow (n clock cycles) | Faster (1 cycle + ripple delay) |
| Hardware | 1 FA + 2 shift regs + 1 DFF | n Full Adders |
| Carry handling | D flip-flop (clocked) | Ripples from $FA_0$ → $FA_{n-1}$ |
| Cost | Low (minimal gates) | Higher (scales with n) |
| Use case | Area-constrained, low speed | General-purpose digital systems |

# 13. BCD Adder

- Adds two BCD (Binary Coded Decimal) digits, each representing 0–9
- Problem: raw binary addition can produce results 10–15 (binary 1010–1111) — invalid BCD codes

## Correction Rule

- If Sum > 9 OR Carry-out = 1 → add correction factor 0110 (6) to the result
- Correction detect: C4 + S3·S2 + S3·S1 (C4 = carry-out, S3–S0 = sum bits)
- After correction, a carry-out of 1 is generated and passed to the next BCD stage

> 📌 Example: 7 (0111) + 6 (0110) = binary 1101 (13 — invalid BCD) → add 0110 → 0001 0011 = BCD 13 ✓
> 📌 Example: 5 (0101) + 4 (0100) = binary 1001 (9) — valid BCD, no correction needed ✓

| Binary Sum Result | Decimal Equivalent | Valid BCD? | Action Required |
|---|---|---|---|
| 0000 – 1001 | 0 – 9 | Yes ✓ | No correction — output directly |
| 1010 – 1111 | 10 – 15 | No ✗ | Add 0110 (6); generate carry-out = 1 |
| Carry-out = 1 | 16 + | No ✗ | Add 0110 (6); carry propagates to next stage |

## Quick Comparison — All MSI Devices

| Device | Function | I/O Summary | Key IC |
|---|---|---|---|
| Magnitude Comparator | Compare two binary numbers | 2 × n-bit → 3 flag outputs | IC 7485 |
| Multiplexer (MUX) | Select 1 of N inputs → 1 output | $2^n$ inputs + n sel → 1 | IC 74151, 74153 |
| Demultiplexer (DEMUX) | Route 1 input → 1 of N outputs | 1 input + n sel → $2^n$ | IC 74138, 74139 |
| Decoder | Binary → one-hot output | n inputs → $2^n$ outputs | IC 74138, 7442 |
| Encoder | One-hot → binary code | $2^n$ inputs → n outputs | IC 74147, 74148 |
| Half Adder | Add 2 bits | A, B → S, C | Discrete gates |
| Full Adder | Add 3 bits (with Cin) | A, B, Cin → S, Cout | IC 7483 |
| Half Subtractor | Subtract 2 bits | A, B → D, Bout | Discrete gates |
| Full Subtractor | Subtract with Borrow-in | A, B, Bin → D, Bout | Discrete gates |
| Serial Adder | Add n-bit, 1 bit/cycle | Shift regs + 1 FA | Custom |
| Parallel Adder | Add n-bit simultaneously | n FAs in cascade | IC 7483 |
| BCD Adder | Add BCD digits (0–9) | Parallel adder + correction | IC 7483 + logic |

## Glossary

| Term | Definition |
|---|---|
| Combinational Logic | Digital circuit where output depends only on present inputs — no memory |
| MSI | Medium Scale Integration — ICs with 12–100 logic gates on one chip |
| MUX | Multiplexer — selects one of several inputs and routes it to a single output |
| DEMUX | Demultiplexer — routes one input to one of several outputs via select lines |
| Decoder | Converts n-bit binary input to one-of-$2^n$ active output (only one HIGH at a time) |
| Encoder | Converts one-of-$2^n$ active input to n-bit binary output code |
| Priority Encoder | Encoder where the highest-indexed active input takes output priority |
| Magnitude Comparator | Compares two binary numbers; outputs A > B, A = B, or A < B |
| Half Adder | Adds two 1-bit numbers; outputs Sum and Carry (no carry-in accepted) |
| Full Adder | Adds three 1-bit values (A, B, Cin); outputs Sum and Carry-out |
| Ripple Carry Adder | Parallel adder where carry propagates (ripples) from LSB to MSB stage |
| Serial Adder | Adds n-bit operands one bit per clock using shift registers and one FA |
| BCD | Binary Coded Decimal — each decimal digit 0–9 encoded as 4 binary bits |
| BCD Adder | Adds BCD digits; adds correction factor 6 (0110) when result exceeds 9 |
| 7-Segment Display | Display with 7 LED segments (a–g) arranged to show decimal digits 0–9 |
| Multiplexed Display | Scanning display technique — digits enabled one at a time rapidly; uses fewer wires |
| Persistence of Vision | Human eye retains image ~20 ms; makes rapidly scanned digits appear continuously lit |

| Term | Definition |
| --- | --- |
| Carry Propagation Delay | Time for carry to ripple through all FA stages in a parallel adder |
| Enable (EN) Pin | Control pin on decoder/DEMUX; when inactive, all outputs are disabled |
| Cascading | Connecting multiple IC chips to extend bit-width (e.g., two 7485 → 8-bit compare) |

END OF NOTES