UNIT – II

Combinational Logic: Combinational Circuits, Analysis procedure Design procedure, Binary Adder- Subtractor Decimal Adder, Binary multiplier, magnitude comparator, Decoders, Encoders, Multiplexers, HDL for combinational circuits.
Sequential Logic: Sequential circuits, latches, Flip-Flops Analysis of clocked sequential circuits, state Reduction and Assignment, Design Procedure. Registers, shift Registers, Ripple counters, synchronous counters, other counters.

Combinational Circuits

Output depends only on the present input values

A combinational circuit is a digital circuit whose output is determined only by its current inputs.

It has no memory element and normally has no feedback path.

General structure: Inputs → Logic Network → Outputs.

Examples include adders, subtractors, comparators, decoders, encoders, and multiplexers.

Combinational circuits are fundamental building blocks of arithmetic and data-processing units.

Inputs

Present input variables

Logic Network

AND, OR, NOT and other gates

Outputs

Function of current inputs

Analysis and Design Procedure

Analysis Procedure

1. Identify inputs and outputs
2. Derive Boolean expressions
3. Construct truth table
4. Simplify expressions
5. Verify circuit operation

Design Procedure

1. Specify required function
2. Determine inputs/outputs
3. Develop truth table
4. Obtain simplified expressions
5. Draw and test logic circuit

Typical Flow

Problem specification
↓
Truth table
↓
Boolean expressions
↓
Simplification
↓
Logic diagram

A systematic procedure reduces design errors and produces an efficient circuit with fewer gates and connections.

Binary Adder–Subtractor

Half Adder

Adds two 1-bit numbers.
Sum: $S = A \oplus B$
Carry: $C = A \cdot B$

Full Adder

Adds A, B and Cin.
Sum = $A \oplus B \oplus Cin$
Carry generated for overflow.

Half Subtractor

Difference: $D = A \oplus B$
Borrow: $B = \overline{A} \cdot B$

Full Subtractor

Inputs A, B, Bin.
Outputs D and Bout.

An adder-subtractor combines arithmetic operations using controlled complementing of the second operand.

Subtraction can be performed using 2's complement: A − B = A + (2's complement of B).

Decimal Adder and Binary Multiplier

Decimal / BCD Adder

Adds two BCD digits and a carry.

If the binary sum exceeds 1001 (9), or a carry is generated, 0110
is added to correct the result to valid BCD.

Example concept: BCD addition → binary addition → correction.

Binary Multiplier

Binary multiplication follows the same principle as decimal
multiplication.

- Generate partial products using AND gates
- Shift partial products
- Add partial products using adders

Used in arithmetic processing units.

Hardware multipliers can be designed as combinational networks or optimized structures for speed and area.

Magnitude Comparator

A magnitude comparator compares two binary numbers and determines their relative magnitude.

For two numbers A and B, the possible outputs are A > B, A = B, or A < B.

A 1-bit comparator can be constructed using AND, OR and NOT gates.

Multi-bit comparators compare the most significant bits first; lower bits are considered when higher bits are equal.

Comparators are used in processors, sorting circuits, control logic, and address checking.

A > B

Output = 1 when A is greater than B.

A = B

Output = 1 when both numbers are equal.

A < B

Output = 1 when A is less than B.

Decoders and Encoders

Decoder

Converts n input lines into up to 2^n output lines.

Example: 2-to-4 decoder
00 → Y0
01 → Y1
10 → Y2
11 → Y3

Applications: memory selection, instruction decoding, Display control.

Encoder

Performs the reverse operation: converts one active input among 2^n lines into an n-bit code.

Example: 8-to-3 encoder

Applications: keyboard encoding, interrupt systems, data compression.

Priority encoders resolve situations in which more than one input may be active by assigning priority to one input.

Multiplexers

2:1 MUX

Two data inputs: I0, I1
One select line: S

$Y = \overline{S}I0 + SI1$

4:1 MUX

Four data inputs
Two select lines

Select lines choose exactly one input for the output.

Applications

- Data routing
- Bus selection
- Function generation
- Communication systems
- CPU datapaths

A multiplexer is a data selector that transfers one selected input to a single output.

Boolean functions can be implemented using multiplexers by connecting appropriate constants or variables to the data inputs.

HDL for Combinational Circuits

HDLs describe digital hardware using a textual representation that can be simulated and synthesized.

Common HDLs include Verilog and VHDL.

HDL supports modular design, simulation, verification, and automatic hardware synthesis.

A combinational circuit is modeled so that its output depends only on current input signals.

Example concept for a 2:1 MUX: assign Y = (~S & I0) | (S & I1);

Design

Describe circuit behavior or structure

Simulation

Verify outputs for test inputs

Synthesis

Convert HDL into hardware logic

Sequential Logic

Circuits with memory

A sequential circuit's output depends on present inputs as well as the previous state.

Memory is provided by latches or flip-flops.

Feedback paths allow the circuit to retain information.

Clocked sequential circuits change state in response to clock events.

Examples include registers, counters, sequence detectors, and control units.

Present Inputs

External signals applied now

Memory / State

Stores previous information

Next State & Output

Determined by inputs and state

Latches and Flip-Flops

SR Latch

Set and Reset inputs.
Basic storage element.

S=1 → Set
R=1 → Reset
Avoid invalid input combination depending on implementation.

D Flip-Flop

Stores one data bit.

At active clock edge:
Q(next) = D

Widely used in registers.

JK Flip-Flop

Improves SR behavior.

J=1,K=0 → Set
J=0,K=1 → Reset
J=K=1 → Toggle

T Flip-Flop

T=0 → Hold
T=1 → Toggle

Useful in counter design.

Latches are level-sensitive; flip-flops are generally edge-triggered storage elements.

Analysis of Clocked Sequential Circuits

Sequential circuit analysis determines how a circuit moves from one state to another for each clock event.

Identify flip-flop inputs and derive the next-state equations.

Construct the state table showing present state, inputs, next state, and outputs.

Represent state transitions using a state diagram.

Timing diagrams help visualize clock, input, state, and output relationships.

Present State

Current stored values of flip-flops

Next State

Values after the active clock edge

Output

Moore: state-based
Mealy: state + input

State Reduction, Assignment and Design Procedure

State Reduction

Remove equivalent states that produce the same behavior for all relevant inputs.

Benefits: fewer flip-flops and simpler logic.

State Assignment

Assign binary codes to symbolic states.

Example:
S0 = 00
S1 = 01
S2 = 10
S3 = 11

Design Procedure

1. Specification
2. State diagram
3. State table
4. State reduction
5. State assignment
6. Choose FFs
7. Derive logic

Implementation

Logic equations
↓
Circuit
↓
Verification

Good state design minimizes hardware while preserving the required sequence and output behavior.

Registers and Shift Registers

Register

A group of flip-flops used to store a multi-bit binary word.

Parallel loading transfers several bits simultaneously.

SISO

Serial-In Serial-Out
Data enters one bit at a time and leaves serially.

SIPO

Serial-In Parallel-Out
Useful for serial-to-parallel conversion.

PISO / PIPO

PISO: parallel-in serial-out.
PIPO: parallel-in parallel-out.

Shift registers move stored bits left or right on clock pulses.

Applications include data transfer, delay lines, serial communication, temporary storage, and sequence generation.

Counters and Unit–II Summary

Ripple / Asynchronous Counter

Flip-flops do not receive the clock simultaneously.

Simple hardware, but propagation delay accumulates through stages.

Synchronous Counter

All flip-flops receive the same clock.

Faster and easier to control for high-speed systems.

Other Counters

Up/down counters
Decade counters
Ring counters
Johnson counters
Frequency dividers

Unit summary: Combinational circuits → Adders/Subtractors → Comparators → Decoders/Encoders → MUX → HDL → Sequential circuits → Flip-flops → State design → Registers → Counters.

Key distinction: combinational circuits have no stored state; sequential circuits use memory and state transitions.