MODULE 1

1. Combinational Logic

A combinational circuit is a digital circuit whose output at any instant depends only on the present combination of inputs, and not on any past input or output. Combinational circuits are constructed using interconnected logic gates and contain no memory elements or feedback paths. This module covers the systematic analysis and design of combinational circuits and studies several important standard building blocks used throughout digital system design.

1.1 Combinational Circuits

A combinational circuit consists of logic gates whose outputs, at any given time, are determined purely by the present combination of input values. Formally, for n binary input variables there are 2^n possible input combinations, and for each combination the circuit produces a unique, well-defined set of m binary outputs.

A combinational circuit can be described by:

- A truth table, listing the output values for every possible input combination.
- A set of Boolean functions, one for each output, expressed in terms of the input variables.
- A logic diagram, showing the interconnection of gates that implement the circuit.

Note: A combinational circuit has no storage/memory element and no feedback from output to input — this is what distinguishes it from a sequential circuit.

1.2 Analysis Procedure

Analysis means determining the function performed by a given logic diagram — i.e., deriving the truth table or Boolean expressions from a circuit.

Typical analysis steps:

- 1. Verify that the circuit is combinational (no feedback paths, no memory elements).
- 2. Label all gate outputs that are functions of the input variables with intermediate symbols.
- 3. Derive the Boolean expression for each intermediate output, proceeding gate-by-gate from inputs toward the final outputs.
- 4. Substitute intermediate expressions until each final output is expressed solely in terms of input variables.
- 5. Construct the truth table by evaluating the output expressions for all input combinations, if required.

1.3 Design Procedure

Design is the reverse process of analysis: starting from a verbal specification or a required truth table, a logic diagram is produced that implements the desired function.

- 1. State the problem and identify the required inputs and outputs.
- 2. Assign letter symbols to input and output variables.
- 3. Derive the truth table that defines the relationship between inputs and outputs.
- 4. Obtain simplified Boolean expressions for each output, typically using Karnaugh maps or Boolean algebra.
- 5. Draw the logic diagram and verify correctness against the truth table.

Note: A good design minimizes gate count, gate inputs, and propagation delay, while remaining easy to verify and implement.

1.4 Binary Adder–Subtractor

1.4.1 Half Adder

A half adder adds two single bits (A and B) and produces a Sum and a Carry, but cannot accept a carry-in from a previous stage.

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Boolean expressions: Sum = $A \oplus B$, Carry = $A \cdot B$

1.4.2 Full Adder

A full adder adds three bits — two significant bits (A, B) and a carry-in (Cin) — producing a Sum and a Carry-out (Cout). Multiple full adders can be cascaded to add multi-bit binary numbers.

A	B	Cin	Sum	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

Boolean expressions: Sum = A ⊕B ⊕Cin, Cout = AB + Cin(A ⊕B)

A full adder can also be built from two half adders and an OR gate.

1.4.3 Binary Adder (Ripple-Carry Adder)

An n-bit binary adder is constructed by cascading n full adders, with the carry-out of each stage connected to the carry-in of the next. This is known as a ripple-carry adder because the carry propagates (ripples) from the least significant to the most significant stage.

1.4.4 Binary Subtractor and the Adder–Subtractor Circuit

Subtraction (A − B) can be performed using the same adder hardware by adding A to the 2's complement of B. A combined adder–subtractor circuit is built by XOR-ing each bit of B with a mode-control input M before feeding it into the adder, and connecting M to the carry-in of the least significant stage.

- When M = 0: each B bit passes through unchanged and Cin = 0 → the circuit performs A + B (addition).
- When M = 1: each B bit is complemented and Cin = 1 → the circuit performs A + B' + 1 = A − B (subtraction).

Note: This technique allows one circuit to perform both addition and subtraction, saving hardware.

1.5 Decimal Adder (BCD Adder)

A decimal (BCD) adder adds two BCD digits and produces a sum in valid BCD form. Because a standard 4-bit binary adder produces results from 0 to 19 when adding two BCD digits plus carry, a correction step is required whenever the binary sum exceeds 9 (1001) or generates a carry.

- Add the two 4-bit BCD digits using a standard 4-bit binary adder.
- If the result is greater than 9, or a carry-out is generated, add binary 0110 (6) to correct the result back into valid BCD form and generate the correct decimal carry.
- Otherwise, the binary sum is already the correct BCD result.

1.6 Binary Multiplier

Binary multiplication of two unsigned numbers is performed the same way as decimal long multiplication: the multiplicand is multiplied by each bit of the multiplier to form partial products, which are then shifted according to bit position and summed.

- Each partial product bit is generated using an AND gate (bit of multiplicand AND bit of multiplier).
- Partial products are added together using an array of binary adders — this structure is called an array multiplier.

For two n-bit operands, the product requires up to 2n bits, and the circuit requires n × n AND gates along with (n − 1) n-bit adders in a typical array multiplier design.

1.7 Magnitude Comparator

A magnitude comparator is a combinational circuit that compares two binary numbers, A and B, and determines their relative magnitude, producing three possible outputs: A > B, A = B, and A < B.

- Equality is checked bit-by-bit using XNOR gates: two numbers are equal only if all corresponding bit pairs are equal.
- The relative magnitude (greater-than / less-than) is determined by comparing bits starting from the most significant bit and examining the first position where the bits differ.

A common example is the 4-bit magnitude comparator (e.g., IC 7485), which can also be cascaded to compare numbers of greater width.

1.8 Decoders

A decoder is a combinational circuit that converts binary information from n input lines into a maximum of 2^n unique output lines, activating exactly one output for each valid input combination (in an active-high decoder).

- An n-to-2^n line decoder has n inputs and up to 2^n outputs.
- Common examples include the 2-to-4, 3-to-8, and 4-to-16 line decoders.
- Decoders are often equipped with one or more enable inputs to control circuit operation and allow cascading of smaller decoders into larger ones.
- Applications: memory address decoding, binary-to-decimal conversion, and implementing arbitrary combinational functions (since a decoder generates all minterms of its input variables).

1.9 Encoders

An encoder performs the inverse operation of a decoder: it converts 2^n (or fewer) input lines, of which only one is active at a time, into an n-bit binary code representing that active input.

- A basic encoder assumes only one input is active at any given time; if none or more than one input is active, the output is undefined.
- A priority encoder resolves this limitation: if two or more inputs are active simultaneously, it outputs the code corresponding to the input with the highest priority.

Example: an octal-to-binary priority encoder has 8 inputs and produces a 3-bit binary output identifying the highest-priority active input, along with a valid-output indicator.

1.10 Multiplexers

A multiplexer (MUX) is a combinational circuit that selects one of several input lines and routes it to a single output line, based on the values of a set of select (control) lines. A multiplexer with 2^n input lines requires n select lines.

- A 2-to-1 MUX selects between two inputs using one select line.
- A 4-to-1 MUX selects among four inputs using two select lines; larger multiplexers (8-to-1, 16-to-1) follow the same pattern.
- Multiplexers are often called data selectors, and can also be used to implement arbitrary Boolean functions by connecting input lines to 0, 1, or literals.

The complementary circuit, a demultiplexer, routes a single input to one of several outputs based on select lines; a decoder with an enable input effectively functions as a demultiplexer.

1.11 HDL for Combinational Circuits

Hardware Description Languages (HDLs), such as Verilog and VHDL, are used to describe the structure and behavior of digital circuits in textual form so that they can be simulated and synthesized into hardware. Combinational circuits can be described in HDL using several modeling styles:

- Dataflow (behavioral) modeling: describing the circuit using continuous Boolean assignment statements, e.g. assign Sum = A ^ B ^ Cin;
- Structural modeling: describing the circuit as an interconnection of gate-level or module-level components, mirroring the logic diagram.
- Behavioral modeling: using procedural constructs (e.g., always/process blocks with case statements) to describe the input–output relationship, especially useful for multiplexers, decoders, and comparators.

Example (Verilog, dataflow style, full adder):

```
module full_adder(input A, B, Cin, output Sum, Cout);   assign Sum  = A ^ B ^ Cin;
assign Cout = (A & B) | (Cin & (A ^ B)); endmodule
```

Note: HDL descriptions allow a design to be simulated for correctness before being synthesized into an actual gate-level circuit or programmed onto an FPGA.

MODULE 2

2. Sequential Logic

A sequential circuit is a digital circuit whose outputs depend not only on the present input values but also on the past sequence of inputs, i.e., on the current state of the circuit. Sequential circuits contain memory (storage) elements and, in clocked sequential circuits, a clock signal that synchronizes state changes.

2.1 Sequential Circuits

A sequential circuit is generally defined as a combination of a combinational circuit and a set of memory elements connected in a feedback loop: the memory elements store the present state, and the combinational logic determines both the outputs and the next state based on the present state and current inputs.

- Synchronous sequential circuits: state changes occur only at discrete instants controlled by a clock signal (using clocked flip-flops).
- Asynchronous sequential circuits: outputs can change at any time in response to input changes, with no clock signal controlling transitions.

This module focuses primarily on clocked (synchronous) sequential circuits, which are used almost universally in digital system design because of their predictable, controlled timing behavior.

2.2 Latches

A latch is a basic memory device that can store one bit of information; unlike a flip-flop, a latch is level-sensitive rather than edge-triggered — its output can change whenever the enable/control input is active.

2.2.1 SR Latch

The SR (Set-Reset) latch is built from two cross-coupled NOR gates (or NAND gates). It has Set (S) and Reset (R) inputs, and Q, Q' outputs.

S	R	Q (next state)	Description
0	0	Q (no change)	Hold state
0	1	0	Reset
1	0	1	Set
1	1	Undefined	Invalid (avoided)

2.2.2 D Latch

The D (Data) latch eliminates the invalid state of the SR latch by ensuring S and R are never simultaneously 1. It has a single Data input (D) and an Enable (E) input: while enabled, the output Q follows D directly (Q = D); when disabled, Q retains its last value.

2.3 Flip-Flops

A flip-flop is an edge-triggered memory element whose output changes state only at a clock edge (rising or falling), rather than continuously following the input as a latch does. Flip-flops are the fundamental building blocks of clocked sequential circuits.

2.3.1 SR Flip-Flop

An edge-triggered version of the SR latch; changes state only on a clock edge, subject to the same S = R = 1 restriction.

2.3.2 D Flip-Flop

Captures the value of D at the triggering clock edge and holds it until the next edge: Q(next) = D. It is the most widely used flip-flop because it has no undefined state.

2.3.3 JK Flip-Flop

An enhancement of the SR flip-flop that resolves the invalid input combination: when J = K = 1, the output toggles (complements) at each clock edge instead of being undefined.

J	K	Q (next state)
0	0	Q (no change)
0	1	0 (reset)
1	0	1 (set)
1	1	Q' (toggle)

2.3.4 T Flip-Flop

The T (Toggle) flip-flop has a single input T: when T = 0 the state is held, and when T = 1 the output toggles on every clock edge. It is commonly derived from a JK flip-flop by tying J and K together (J = K = T), and is especially useful for building counters.

2.3.5 Characteristic and Excitation Tables

The characteristic table of a flip-flop specifies its next state as a function of its present state and inputs, and the characteristic equation expresses this relationship algebraically (for example, the D flip-flop's characteristic equation is Q(next) = D). The excitation table lists the required input values needed to produce a specified state transition, and is essential for the design of sequential circuits using a given flip-flop type.

2.4 Analysis of Clocked Sequential Circuits

Analysis of a clocked sequential circuit means obtaining a description of its time-sequence behavior — inputs, outputs, and internal states — from the circuit's logic diagram.

- Derive the next-state and output Boolean equations from the logic diagram, in terms of present-state variables and inputs.
- Construct the state table, listing present state, input, next state, and output for every valid combination.

- Draw the state diagram, a graphical representation with a circle for each state and directed arcs (labeled input/output) showing transitions between states.

Note: The state table and state diagram contain identical information; the state diagram is simply a more visual representation used for design intuition and verification.

2.5 State Reduction and Assignment

2.5.1 State Reduction

State reduction is the process of eliminating redundant (equivalent) states from a state table without changing the input–output behavior of the circuit. Two states are considered equivalent if, for every possible input sequence, they produce identical output sequences. Reducing the number of states can simplify the circuit and reduce the number of flip-flops required.

2.5.2 State Assignment

Since flip-flops store binary values, each named state in a state table must be assigned a unique binary code before the circuit can be implemented. For a design with m states, at least $\lceil \log_2(m) \rceil$ flip-flops are required. Different assignments of binary codes to states can result in circuits with different complexity, so the choice of assignment can affect the final gate count.

2.6 Design Procedure

The design of a clocked sequential circuit follows a systematic sequence of steps, translating a word specification into a working circuit:

- 1. State the design specification and derive the state diagram from the required behavior.
- 2. Reduce the number of states, if possible, by eliminating equivalent states.
- 3. Assign binary codes to the states (state assignment) and construct the state table.
- 4. Choose the type of flip-flop to be used and derive the excitation table using the flip-flop's excitation requirements.
- 5. Derive simplified Boolean expressions for the flip-flop inputs and the circuit outputs (e.g., using Karnaugh maps).
- 6. Draw the final logic diagram, showing flip-flops and combinational logic gates.

2.7 Registers

A register is a group of flip-flops, each capable of storing one bit, together with associated gates that control when and how new data is loaded. An n-bit register can store an n-bit binary word and is a fundamental building block for CPU data storage, temporary storage, and pipelining.

- A simple register uses common clock and clear lines connected to all flip-flops so that data is loaded into all bits simultaneously (parallel load).
- A load-control input, combined with multiplexers or gated clocking, determines whether the register accepts new data or retains its current value on each clock edge.

2.8 Shift Registers

A shift register is a register that, in addition to storing data, can shift its stored bits left or right by one position on each clock pulse. Shift registers are constructed by connecting flip-flops in cascade, with the output of one flip-flop feeding the input of the next.

Type	Description
SISO	Serial-In, Serial-Out — data enters and exits one bit at a time.
SIPO	Serial-In, Parallel-Out — data enters serially; all bits are available simultaneously at the output.
PISO	Parallel-In, Serial-Out — data is loaded into all stages at once; output is read out one bit at a time.
PIPO	Parallel-In, Parallel-Out — data is loaded and read out simultaneously, in parallel.
Bidirectional	Can shift data either left or right depending on a mode-control input.

2.9 Ripple Counters

A ripple counter (asynchronous counter) is built by cascading flip-flops (typically T or JK configured to toggle) such that the output of one flip-flop serves as the clock input for the next stage. Each flip-flop toggles only when the preceding stage transitions from 1 to 0 (or vice versa), so the count value 'ripples' through the chain rather than changing all bits simultaneously.

- Simple to build, using minimal additional logic.
- Suffers from cumulative propagation delay, since each stage must wait for the previous stage to toggle — this limits maximum operating speed and can cause transient (glitch) output values.

2.10 Synchronous Counters

In a synchronous counter, all flip-flops share a common clock signal, and all bits change state simultaneously on the same clock edge. Additional combinational logic (typically AND gates driving J-K or T inputs) determines which flip-flops should toggle on each clock pulse, based on the current count value.

- Eliminates the cumulative propagation delay of ripple counters, allowing much higher operating frequency.
- Requires more combinational logic than a ripple counter, since toggle conditions must be computed for every stage.

2.11 Other Counters

- Up-Down Counter: Can count in either the increasing or decreasing direction, controlled by a mode-select (up/down) input.
- Modulo-N (Mod-N) Counter: Counts through N distinct states before repeating (e.g., a decade/BCD counter is a mod-10 counter that counts 0 through 9 and then resets).

- Ring Counter: A shift register in which the output of the last flip-flop is fed back to the input of the first, circulating a single active bit ('walking 1') around the register.
- Johnson Counter (Twisted-Ring Counter): Similar to a ring counter, but the complemented output of the last stage is fed back to the first stage, producing a longer unique-state sequence (2n states for n flip-flops) with only one bit changing per clock pulse.