NEC Registration Examination

Computer Engineering (ACtE)

Chapter 4: Computer Organization and Embedded System (ACtE04)

Computer Engineering (ACtE)
Electronics and Communication
Telecommunication
Software Engineering
Information Technology

Booth's Algorithm with Worked Trace • All Cache Mappings • Replacement Policies • Scheduling Algorithms • Tables & Diagrams • Practice MCQs

COMPLETE NOTES WITH SOLVED MCQ AND PYQS

EASY EXPLANATION



EASY EXPLANATION

4.1 Control and Central Processing Units

Control Unit Design — Hardwired vs Microprogrammed

Feature	Hardwired Control Unit	Microprogrammed Control Unit
Implementation	Fixed logic gates, flip-flops, decoders	Control memory (ROM) storing microinstructions
Speed	FASTER (direct hardware)	SLOWER (extra control-memory access)
Flexibility	Difficult to modify — requires rewiring	Easy to modify — just rewrite the microcode
Design cost	High for complex instruction sets	Lower, more systematic
Instruction set suited	Small, simple, fixed-length	Large, complex, variable-length
Used in	RISC processors	CISC processors

- **Control memory:** ROM holding **microinstructions**; each microinstruction issues the control signals for one micro-operation.
- **Microinstruction format fields:** (1) **Control field** — which signals to activate; (2) **Condition field** — branch condition to test; (3) **Branch/Next-address field** — where to go next.
- **Address sequencing:** determines the next microinstruction address via `incrementing`, `opcode mapping`, `conditional branching`, or `subroutine return`.

CPU Registers — Complete Reference

Register	Full name	Function
PC	Program Counter	Holds address of the NEXT instruction to fetch
IR	Instruction Register	Holds the instruction CURRENTLY being executed
MAR	Memory Address Register	Holds the address of the memory location to be accessed
MBR / MDR	Memory Buffer / Data Register	Holds the data being transferred to/from memory
AC	Accumulator	Holds intermediate arithmetic/logic results
SP	Stack Pointer	Points to the top of the stack
Flags / PSW	Status Register	Holds condition flags: Zero, Carry, Sign, Overflow, Parity

Instruction Formats

Format	Example instruction	Meaning	Machine type
Zero-address	PUSH / POP / ADD	Operands implicit on the stack	Stack machine
One-address	ADD X	AC ← AC + X	Accumulator machine
Two-address	ADD A, B	A ← A + B (destination overwritten)	Most real CPUs
Three-address	ADD A, B, C	A ← B + C (sources preserved)	RISC machines

Addressing Modes — Complete Table

Mode	Where the operand is	Example	Memory accesses for operand
Immediate	Inside the instruction itself	MOV A, #5	0 — FASTEST
Direct (Absolute)	At the address given in the instruction	MOV A, 2000	1
Indirect	At the address stored AT the given address	MOV A, (2000)	2 — SLOWEST
Register	In a CPU register	MOV A, B	0
Register Indirect	At the address held in a register	MOV A, (R1)	1
Indexed	Base address + index register value	MOV A, 2000(R1)	1
Relative	PC + offset given in instruction	JMP +10	1
Implied	Implicitly known (no operand field)	CMA (complement AC)	0

Easy Explanation

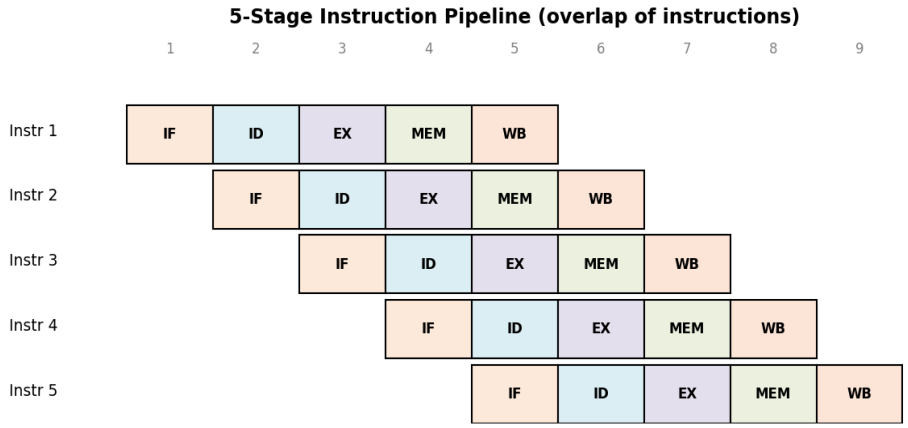
EASY EXPLANATION

Speed ranking for operand access: Immediate & Register (no memory access) → Direct & Register-Indirect & Indexed (one access) → Indirect (two accesses, slowest). Indexed addressing is the one used for ARRAY element access.

RISC vs CISC

Feature	RISC	CISC
Instruction set size	Small (~50–100), simple	Large (~200–300), complex
Instruction length	Fixed	Variable
Execution time	Mostly 1 clock cycle	Multiple cycles per instruction
Addressing modes	Few (3–5)	Many (12–24)
Registers	Many general-purpose registers	Fewer registers
Control unit	Hardwired	Microprogrammed
Memory access	Only LOAD / STORE touch memory	Many instructions can access memory
Pipelining	Easy and highly effective	Difficult due to variable lengths
Examples	ARM, MIPS, SPARC, RISC-V	Intel x86, Motorola 68000

Pipelining



Time (non-pipelined) = n × k × t

Time (pipelined) = (k + n − 1) × t

Speedup S = (n × k) / (k + n − 1) → approaches k for large n

(n = number of instructions, k = number of stages, t = time per stage)

Hazard type	Cause	Common remedy
Structural hazard	Two stages need the SAME hardware resource at once	Duplicate resources; separate instruction/data caches
Data hazard	Instruction needs a result not yet produced	Operand forwarding (bypassing); stalling; reordering
Control hazard	Branch changes the	Branch prediction; delayed branch;

	instruction flow	flushing the pipeline

Flynn's Classification of Parallel Processing

Class	Full form	Description	Example
SISD	Single Instruction Single Data	One instruction stream on one data stream	Traditional uniprocessor
SIMD	Single Instruction Multiple Data	Same instruction applied to many data items in parallel	Vector processors, GPUs
MISD	Multiple Instruction Single Data	Many instructions on the same data (rare/theoretical)	Fault-tolerant systems
MIMD	Multiple Instruction Multiple Data	Independent instructions on independent data	Multiprocessors, multicore CPUs

⍰ Probable MCQs: Control Unit and CPU

1. A microprogrammed control unit generates control signals from:

a) Cache memory
b) Control memory (ROM)
c) The ALU
d) The accumulator
✓Answer: b) Control memory (ROM)
Why: Microinstructions stored in control memory generate the control signals.

2. Which addressing mode requires TWO memory accesses to fetch the operand?

a) Immediate
b) Direct
c) Indirect
d) Register
✓Answer: c) Indirect
Why: Indirect addressing first fetches the address, then fetches the operand — two accesses.

3. RISC processors typically use which type of control unit?

a) Microprogrammed
b) Hardwired
c) Software-based
d) None
✓Answer: b) Hardwired
Why: RISC's simple fixed instruction set suits fast hardwired control.

4. For a k-stage pipeline with large n, the ideal speedup approaches:

a) n
b) k
c) n×k
d) k/n
✓Answer: b) k
Why: Speedup = nk/(k+n−1), which tends to k as n becomes large.

5. A hazard caused by a branch instruction is classified as a:

a) Structural hazard
b) Data hazard
c) Control hazard
d) Memory hazard
✓Answer: c) Control hazard

EASY EXPLANATION

Why: Branches alter the instruction flow, causing control (branch) hazards.

6. GPUs and vector processors are examples of which Flynn class?

a) SISD

b) SIMD

c) MISD

d) MIMD

✓Answer: b) SIMD

Why: SIMD applies one instruction across many data elements simultaneously.

7. Which register holds the address of the memory location being accessed?

a) PC

b) IR

c) MAR

d) AC

✓Answer: c) MAR

Why: The Memory Address Register holds the address placed on the address bus.

8. Indexed addressing mode is most commonly used for accessing:

a) Constants

b) Array elements

c) Stack frames only

d) I/O ports

✓Answer: b) Array elements

Why: Base address + index register naturally maps to array element access.

EASY EXPLANATION

4.2 Computer Arithmetic and Memory System

Signed Number Representation

Representation	+5 (4-bit)	−5 (4-bit)	Range (n bits)	Zero representations
Sign-magnitude	0101	1101	$-(2^{n-1}-1)$ to $+(2^{n-1}-1)$	Two (+0 and −0)
1's complement	0101	1010	$-(2^{n-1}-1)$ to $+(2^{n-1}-1)$	Two (+0 and −0)
2's complement	0101	1011	-2^{n-1} to $+(2^{n-1}-1)$	ONE only

Why 2's complement is used in all modern computers: it has only **one representation of zero**, it allows **subtraction to be performed by addition** (no separate subtractor circuit), and the **sign bit participates normally** in arithmetic.

2's complement = 1's complement (invert all bits) + 1

Overflow detected when: Carry INTO sign bit ≠ Carry OUT of sign bit

Booth's Multiplication Algorithm — Detailed

Purpose: multiplies two **signed (2's complement)** binary numbers efficiently by treating strings of consecutive 1s as a single subtract-then-add operation, reducing the number of add/subtract steps.

Registers used

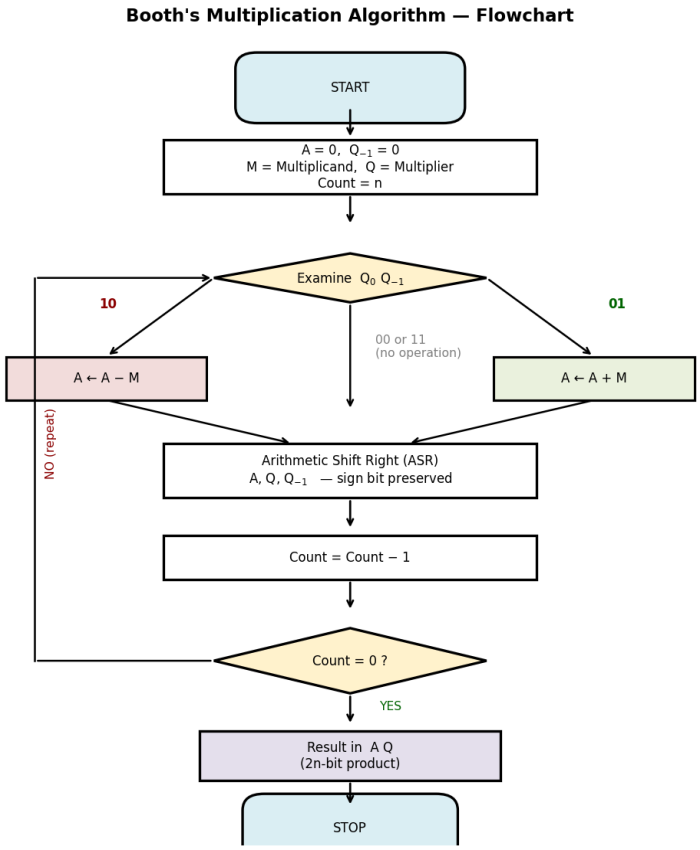
Register	Contents	Size
M	Multiplicand	n bits
Q	Multiplier	n bits
A	Accumulator, initialised to 0	n bits
Q_{-1}	Extra 1-bit flip-flop appended to the right of Q, initialised to 0	1 bit
Count	Sequence counter, initialised to n	—

The decision rule — examine the bit pair Q_0Q_{-1}

Q_0 Q_{-1}	Meaning	Operation performed
0 0	Middle of a string of 0s	No arithmetic — just shift
0 1	END of a string of 1s	A ← A + M (ADD), then shift
1 0	BEGINNING of a string of 1s	A ← A − M (SUBTRACT), then shift
1 1	Middle of a string of 1s	No arithmetic — just shift

After every case, perform an **Arithmetic Shift Right (ASR)** on the combined A−Q−Q_{-1} register and decrement Count. Repeat until Count = 0. The **2n-bit product** ends up in AQ.

EASY EXPLANATION



Worked example: multiply −5 × 2 using 4-bit Booth's algorithm

M = −5 = `1011`, Q = 2 = `0010`, A = `0000`, Q_{-1} = `0`, Count = 4. (Note `−M = +5 = 0101`.)

Step	Q_0Q_{-1}	Operation	A	Q	Q_{-1}	Count
Initial	—	—	0000	0010	0	4
1	0 0	No-op, then ASR	0000	0001	0	3
2	1 0	A ← A − M (A=0000+0101=0101)	0101	0001	0	3
		then ASR	0010	1000	1	2
3	0 1	A ← A + M (A=0010+1011=1101)	1101	1000	1	2
		then ASR	1110	1100	0	1
4	0 0	No-op, then ASR	1111	0110	0	0

Result: AQ = `1111 0110` which in 8-bit 2's complement = **−10**. Correct, since −5 × 2 = −10. ✓

💡 Easy Explanation

Memorize Booth's decision rule as a 'transition detector': 01 means you just LEFT a run of 1s → ADD; 10 means you just ENTERED a run of 1s → SUBTRACT; 00 and 11 mean no transition → shift only. The number of add/subtract operations equals the number of 0↔1 transitions in the multiplier, which is why long runs of 1s are handled so efficiently.

Other Arithmetic Algorithms

Algorithm	Purpose	Key idea
Booth's algorithm	Signed multiplication	Shift and add/subtract based on bit-pair transitions
Array multiplier	Unsigned multiplication	Parallel grid of AND gates + adders — fast, more hardware
Restoring division	Division	Subtract divisor; if result negative, RESTORE by adding it back
Non-restoring division	Division	Avoids the restore step — faster, fewer operations

Memory Types — Classification

Type	Volatile?	Technology	Refresh needed	Typical use
SRAM	Yes	Flip-flops (6 transistors)	NO	Cache memory
DRAM	Yes	Capacitor + 1 transistor	YES (periodic)	Main memory (RAM)
ROM	No	Mask-programmed at manufacture	—	Firmware/BIOS
PROM	No	Programmed ONCE by the user	—	One-time firmware
EPROM	No	Erasable by ULTRAVIOLET light	—	Reprogrammable firmware
EEPROM / Flash	No	Erasable ELECTRICALLY	—	BIOS, SSD, USB drives

Feature	SRAM	DRAM
Speed	Faster	Slower
Cost per bit	Higher	Lower
Density	Lower	Higher
Power consumption	Higher (static)	Lower, but refresh circuitry needed

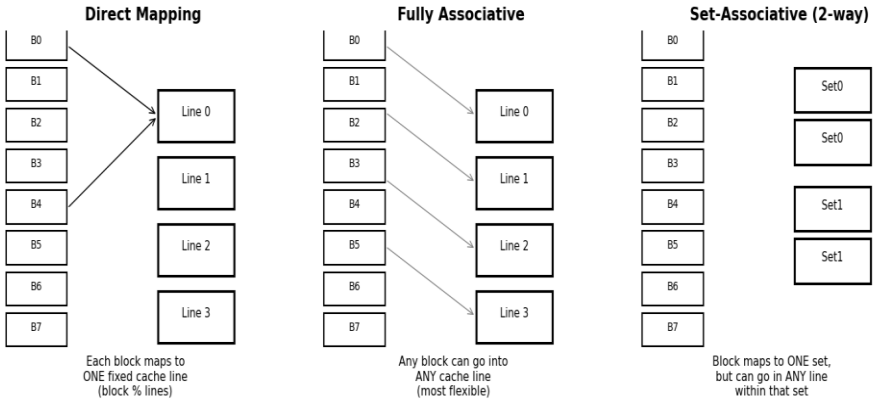
Cache Memory — Principle of Locality

- **Temporal locality:** a location recently accessed is **likely to be accessed again soon** (e.g. a loop counter).
- **Spatial locality:** locations **near** a recently accessed one are likely to be accessed soon (e.g. sequential array elements).

Hit Ratio H = (number of cache hits) / (total memory accesses)

Average Access Time Tavg = H × Tcache + (1 − H) × Tmemory

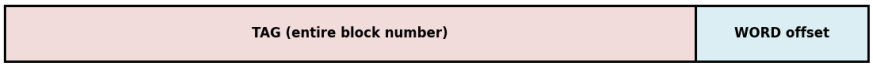
Cache Mapping — All Three Techniques in Detail



Direct Mapping



Fully Associative Mapping



Set-Associative Mapping



Feature	Direct Mapping	Fully Associative	Set-Associative
Block placement	Exactly ONE fixed line	ANY line in the cache	Any line WITHIN one fixed set
Mapping formula	Line = (Block no.) mod (No. of lines)	No fixed mapping	Set = (Block no.) mod (No. of sets)
Address fields	TAG + LINE + WORD	TAG + WORD	TAG + SET + WORD
Tag size	Smallest	LARGEST	Medium
Comparators needed	1 (simplest hardware)	One per line (most expensive)	One per line in a set (k for k-way)
Search speed	Fastest	Slowest (must search all lines)	Medium
Conflict misses	HIGH (two hot blocks may collide)	NONE	Low
Replacement policy	NOT needed (only one choice)	Needed	Needed (within the set)
Hardware cost	Lowest	Highest	Moderate — best compromise

Direct: Line number = Block number MOD Number of lines

Set-Associative: Set number = Block number MOD Number of sets

k-way set associative: each set holds k lines

Worked example: cache has 128 lines, main memory block number 1000.

- **Direct mapping:** line = 1000 mod 128 = 104 — the block can only go in line 104.
- **4-way set associative:** number of sets = 128/4 = 32; set = 1000 mod 32 = 8 — the block goes anywhere within set 8 (4 choices).
- **Fully associative:** the block may occupy **any** of the 128 lines.

Easy Explanation

Remember the three mappings as a spectrum of flexibility: Direct = 'one assigned seat' (fast, but collisions), Fully Associative = 'sit anywhere' (no collisions, expensive to search), Set-Associative = 'assigned row, any seat in that row' (the practical compromise used in real CPUs). Note that Direct mapping is the ONLY one that needs no replacement policy.

Cache Replacement Policies — Detailed

Policy	Full form	How it chooses the victim	Advantage	Disadvantage
LRU	Least Recently Used	Evicts the block UNUSED for the longest time	Exploits temporal locality — best practical hit ratio	Costly to track usage history
FIFO	First In First Out	Evicts the OLDEST loaded block regardless of use	Very simple to implement (queue)	May evict a heavily used block; suffers Belady's anomaly
LFU	Least Frequently Used	Evicts the block with the LOWEST access count	Good for stable access patterns	Old popular blocks linger; needs counters

Policy	Full form	How it chooses the victim	Advantage	Disadvantage
`Random`	Random replacement	Picks a victim at random	Cheapest hardware, no bookkeeping	Unpredictable performance
`Optimal (OPT)`	Belady's optimal	Evicts the block not needed for the LONGEST time in future	Theoretical BEST possible	Not implementable — needs future knowledge

Worked LRU example: 3 frames, reference string `1, 2, 3, 1, 4`.

Reference	Frames after access	Hit / Miss	Comment
1	`[1]`	MISS	Frame empty
2	`[1, 2]`	MISS	
3	`[1, 2, 3]`	MISS	Cache now full
1	`[2, 3, 1]`	HIT	1 becomes most recently used
4	`[3, 1, 4]`	MISS	2 was least recently used → evicted

Cache Write Policies

Policy	When main memory is updated	Advantage	Disadvantage
Write-through	IMMEDIATELY, on every cache write	Memory always consistent; simple recovery	Slower — every write goes to memory
Write-back	ONLY when the modified line is replaced	Much faster; fewer memory writes	Memory temporarily inconsistent; needs a DIRTY bit

Design element	Options / notes
Cache size	Larger = higher hit ratio but slower and costlier
Block (line) size	Larger exploits spatial locality, but too large reduces number of lines
Number of caches	Split (separate I-cache and D-cache) vs Unified; Multi-level L1, L2, L3
Mapping function	Direct / Fully associative / Set-associative
Replacement algorithm	LRU / FIFO / LFU / Random
Write policy	Write-through / Write-back

⍰ Probable MCQs: Computer Arithmetic and Memory

1. In Booth's algorithm, the bit pair Q_0Q_{-1} = 10 causes which operation?

a) A ← A + M
b) A ← A − M
c) No operation
d) Reset A to 0
✓Answer: b) A ← A − M
Why: 10 marks the BEGINNING of a run of 1s → subtract the multiplicand.

2. In Booth's algorithm, the bit pair Q_0Q_{-1} = 01 causes:

a) A ← A − M
b) A ← A + M
c) No operation
d) Shift left
✓Answer: b) A ← A + M
Why: 01 marks the END of a run of 1s → add the multiplicand.

3. After every step of Booth's algorithm, which operation is always performed?

a) Logical shift left
b) Arithmetic Shift Right (ASR)
c) Rotate left

EASY EXPLANATION

d) Complement A

✓Answer: b) Arithmetic Shift Right (ASR)

Why: ASR preserves the sign bit, which is essential for signed multiplication.

4. Booth's algorithm is most efficient when the multiplier contains:

a) Alternating 0s and 1s

b) Long runs of consecutive 1s

c) Only zeros

d) Random bits

✓Answer: b) Long runs of consecutive 1s

Why: A whole run of 1s is handled by just one subtract and one add, regardless of its length.

5. Which cache mapping technique does NOT require a replacement policy?

a) Direct mapping

b) Fully associative

c) Set-associative

d) All require one

✓Answer: a) Direct mapping

Why: In direct mapping a block has only one possible line, so there is no choice to make.

6. Which cache mapping requires the LARGEST tag field?

a) Direct mapping

b) Fully associative mapping

c) Set-associative mapping

d) All are equal

✓Answer: b) Fully associative mapping

Why: With no line/set field, the entire block number must be stored as the tag.

7. A cache has 64 lines. Using direct mapping, memory block 500 maps to line:

a) 4

b) 20

c) 44

d) 500

✓Answer: b) 20

Why: 500 mod 64 = 20.

8. Which replacement policy can suffer from Belady's anomaly?

a) LRU

b) FIFO

c) Optimal

d) LFU

✓Answer: b) FIFO

Why: Only FIFO exhibits the anomaly where more frames can cause more misses.

9. Which replacement policy gives the theoretically best hit ratio but cannot be implemented?

a) LRU

b) FIFO

c) Optimal (Belady's)

d) Random

✓Answer: c) Optimal (Belady's)

Why: Optimal requires knowledge of future references, so it is only a benchmark.

EASY EXPLANATION

10. In write-back policy, main memory is updated:

a) On every cache write

b) Only when the modified block is replaced

c) Never

d) At fixed time intervals

✓Answer: b) Only when the modified block is replaced

Why: Write-back defers the memory write until eviction, using a dirty bit to track changes.

11. 2's complement is preferred over 1's complement mainly because it has:

a) Two representations of zero

b) Only ONE representation of zero

c) A larger tag field

d) Faster shifting

✓Answer: b) Only ONE representation of zero

Why: A single zero representation simplifies arithmetic and comparison hardware.

12. Which memory requires periodic refreshing to retain data?

a) SRAM

b) DRAM

c) ROM

d) EEPROM

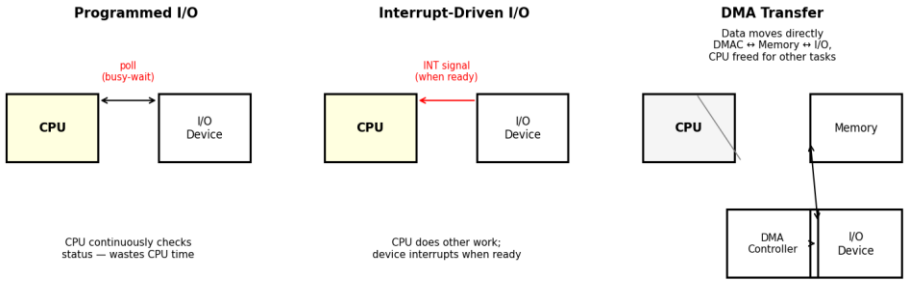
✓Answer: b) DRAM

Why: DRAM stores charge on capacitors, which leaks and must be refreshed.

EASY EXPLANATION

4.3 Input-Output Organization and Multiprocessor

Modes of I/O Data Transfer



Mode	How it works	CPU involvement	Efficiency	Best for
Programmed I/O	CPU repeatedly polls the device status flag (busy-wait)	Continuous — CPU fully occupied	LOWEST	Simple, slow devices
Interrupt-driven I/O	Device raises an interrupt when ready; CPU services via ISR	Only during actual transfer	Medium	Moderate-speed devices
DMA	DMA controller transfers data directly between device and memory	Only setup and completion	HIGHEST	Bulk transfer: disk, video, network

DMA Transfer Modes

DMA mode	Behaviour	Effect on CPU
Burst (Block) mode	Transfers an entire block while holding the bus	CPU blocked for the whole block — fastest transfer
Cycle stealing	Transfers ONE word at a time, releasing the bus in between	CPU slowed slightly; good balance
Transparent mode	Transfers only when CPU is not using the bus	No CPU slowdown at all, but slowest transfer

DMA handshake: DMAC asserts `HOLD / Bus Request` → CPU finishes current cycle and replies `HLDA / Bus Grant` → DMAC becomes bus master and transfers → DMAC releases the bus and may raise an interrupt to signal completion.

Memory-mapped vs I/O-mapped I/O

Feature	Memory-mapped I/O	I/O-mapped (Isolated) I/O
Address space	SHARED with memory	SEPARATE address space
Instructions used	Normal memory instructions (MOV, LOAD, STORE)	Special instructions (IN, OUT)
Available address space for memory	Reduced (I/O consumes some)	Full memory space available
Control signals	Same MEMR / MEMW	Separate IOR / IOW signals
Flexibility	Any memory instruction works on I/O	Limited to IN/OUT only

Multiprocessor Systems

Feature	Tightly Coupled	Loosely Coupled
Memory	Shared common memory	Each processor has its OWN local memory
Communication	Through shared memory (fast)	Message passing over a network (slower)
Also known as	Shared-memory multiprocessor / SMP	Distributed-memory multicomputer
Scalability	Limited (memory contention)	Highly scalable
Example	Multicore desktop CPU	Computing cluster / distributed system

Interconnection structure	Description	Cost	Performance
Time-shared common bus	All units share one bus	Lowest	Poor — bus contention is a bottleneck

EASY EXPLANATION

Interconnection structure	Description	Cost	Performance
Multiport memory	Memory modules have multiple ports, one per processor	High	Good, but complex memory units
Crossbar switch	A dedicated switch at every processor-memory crosspoint	HIGHEST (n^2 switches)	Best — simultaneous non-conflicting access
Multistage switching network	Several stages of small switches (e.g. omega network)	Moderate	Good compromise between cost and speed
Hypercube	Processors at the corners of an n-dimensional cube	Moderate	Good for massively parallel systems

- **Inter-processor synchronization:** uses `semaphores`, `mutex locks`, `test-and-set (atomic) instructions` and `barriers` to protect shared data.
- **Cache coherence problem:** in a multiprocessor, each CPU's private cache may hold a **different copy** of the same memory block. Solved by protocols such as **MESI** (Modified, Exclusive, Shared, Invalid) and bus **snooping**.

⍰ Probable MCQs: I/O Organization and Multiprocessor

1. Which I/O mode keeps the CPU busy-waiting on a status flag?

a) DMA
b) Interrupt-driven I/O
c) Programmed I/O
d) Cycle stealing
✓Answer: c) Programmed I/O
Why: Programmed I/O polls the device continuously, wasting CPU cycles.

2. In which DMA mode does the controller transfer one word at a time, releasing the bus between words?

a) Burst mode
b) Cycle stealing
c) Transparent mode
d) Block mode
✓Answer: b) Cycle stealing
Why: Cycle stealing 'steals' single bus cycles, slowing the CPU only slightly.

3. Memory-mapped I/O has the disadvantage of:

a) Requiring special IN/OUT instructions
b) Reducing the available memory address space
c) Being slower than isolated I/O always
d) Not supporting interrupts
✓Answer: b) Reducing the available memory address space
Why: Since I/O shares the memory address space, some addresses are consumed by devices.

4. Which interconnection structure provides a dedicated path for every processor-memory pair?

a) Common bus
b) Crossbar switch
c) Multistage network
d) Hypercube
✓Answer: b) Crossbar switch
Why: A crossbar places a switch at every crosspoint, allowing simultaneous access.

5. The cache coherence problem arises in:

a) Single-processor systems only

EASY EXPLANATION

b) Multiprocessor systems with private caches

c) Systems with no cache

d) ROM-only systems

✓Answer: b) Multiprocessor systems with private caches

Why: Multiple private caches can hold inconsistent copies of the same block.

6. MESI is a protocol used for:

a) Interrupt priority

b) Cache coherence

c) DMA arbitration

d) Pipeline hazard removal

✓Answer: b) Cache coherence

Why: MESI (Modified, Exclusive, Shared, Invalid) maintains coherence across caches.

7. Loosely coupled multiprocessors communicate primarily via:

a) Shared memory

b) Message passing over a network

c) Direct register transfer

d) The instruction cache

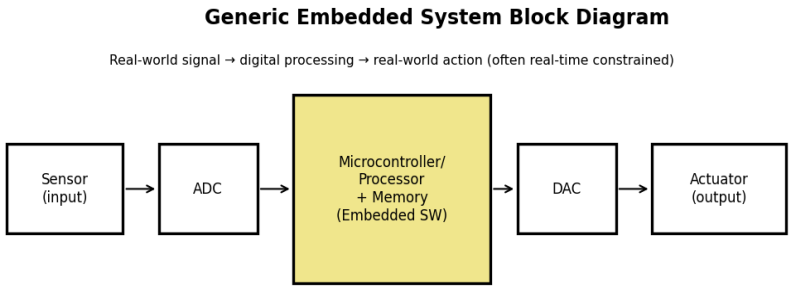
✓Answer: b) Message passing over a network

Why: Each node has private memory, so communication uses explicit messages.

EASY EXPLANATION

4.4 Hardware-Software Design Issues on Embedded Systems

Embedded System Overview



Definition: a combination of hardware and software designed to perform a **specific, dedicated function**, usually with real-time constraints and limited resources.

Characteristic	Embedded System	General-Purpose Computer
Purpose	Single, dedicated task	Many different applications
User interface	Minimal or none	Rich (keyboard, monitor, GUI)
Resources	Constrained (memory, power, size)	Abundant
Real-time constraints	Often critical	Usually not critical
Examples	Washing machine controller, ABS, pacemaker	Desktop PC, laptop, server

Classification of Embedded Systems

Basis	Types	Description
Performance/Function	Stand-alone	Works independently, no host needed (calculator, digital camera)
	Real-time	Must respond within strict deadlines
	Networked	Connected to a network (smart home devices, IoT)
	Mobile	Portable, battery-powered (smartphone, fitness band)
Complexity	Small scale	8-bit microcontroller, battery powered
	Medium scale	16/32-bit microcontroller, may run an RTOS
	Sophisticated	Multiple processors, ASIC/FPGA, complex software

Real-time type	Deadline miss consequence	Example
HARD real-time	Catastrophic — total system failure	Airbag deployment, pacemaker, flight control
FIRM real-time	Result becomes useless, but no catastrophe	Financial forecasting, some robotics
SOFT real-time	Quality degrades, system still usable	Video streaming, online gaming

Processor Choices — The Flexibility vs Efficiency Spectrum

Processor type	Flexibility	Performance / power efficiency	Unit cost (high volume)	NRE design cost
General-Purpose Processor	HIGHEST (fully programmable)	Lowest	Low	Lowest
ASIP (Application-Specific Instruction-set Processor)	Medium (domain-tuned ISA)	Medium-high	Medium	Medium
Custom Single-Purpose Processor / ASIC	LOWEST (fixed function)	HIGHEST	Lowest at very high volume	HIGHEST
FPGA	Reconfigurable (between the above)	Medium	Higher per unit	Low-medium

EASY EXPLANATION

Easy Explanation

Remember the spectrum left to right: General-Purpose Processor → ASIP → FPGA → ASIC. Moving RIGHT gives more speed, lower power and smaller size, but LESS flexibility and much higher design (NRE) cost. This trade-off ordering is the most frequently tested embedded-systems concept.

Architecture and Development

- **Datapath:** performs the computation — registers, ALU, multiplexers, buses.
- **Controller (FSM):** sequences the datapath operations, generating control signals each clock cycle.
- **Programmer's view:** the abstraction visible to software — register set, memory map, instruction set — hiding internal hardware detail.

Development tool	Purpose
Cross-compiler	Runs on the host PC but generates code for the TARGET processor
Simulator / Emulator	Runs target code on the host for testing without real hardware
Debugger	Step through code, set breakpoints, inspect registers/memory
JTAG	Standard hardware interface for IN-CIRCUIT debugging and testing
Programmer / Flasher	Burns the compiled binary into the target's flash memory

Architecture	Instruction & data memory	Characteristic	Typical use
Von Neumann	SHARED single memory and bus	Simpler, but bus bottleneck	General-purpose CPUs
Harvard	SEPARATE memories and buses	Instruction and data fetched simultaneously — faster	Microcontrollers, DSPs

Probable MCQs: Embedded System Design

1. A system where missing a deadline causes catastrophic failure is:

a) Soft real-time
b) Hard real-time
c) Firm real-time
d) Stand-alone
✓Answer: b) Hard real-time
Why: Hard real-time deadlines are absolute; missing one means total system failure.

2. Which processor type offers the HIGHEST performance and efficiency but lowest flexibility?

a) General-purpose processor
b) ASIP
c) FPGA
d) Custom ASIC
✓Answer: d) Custom ASIC
Why: An ASIC is fixed-function silicon — maximum efficiency, zero reprogrammability.

3. An ASIP is best described as a processor with:

a) A fully general instruction set
b) An instruction set customized for a specific application domain
c) No instruction set at all
d) Only hardwired logic
✓Answer: b) An instruction set customized for a specific application domain
Why: ASIPs tune the ISA to a domain (e.g. DSP), balancing flexibility and efficiency.

4. Harvard architecture differs from Von Neumann by having:

a) A single shared memory

EASY EXPLANATION

b) Separate instruction and data memories/buses

c) No memory at all

d) Only ROM

✓Answer: b) Separate instruction and data memories/buses

Why: Separate buses allow simultaneous instruction and data access, improving speed.

5. JTAG is primarily used for:

a) Compiling source code

b) In-circuit hardware debugging and testing

c) Storing program data

d) Regulating power supply

✓Answer: b) In-circuit hardware debugging and testing

Why: JTAG provides a standard boundary-scan/debug interface to the chip.

6. A cross-compiler is a compiler that:

a) Compiles on the target device itself

b) Runs on a host PC but generates code for a different target processor

c) Only compiles assembly

d) Cannot produce binaries

✓Answer: b) Runs on a host PC but generates code for a different target processor

Why: Embedded targets usually lack resources to compile, so compilation happens on the host.

7. The 'programmer's view' of a processor exposes:

a) Transistor layout

b) Registers, memory map and instruction set

c) Power supply design

d) PCB routing

✓Answer: b) Registers, memory map and instruction set

Why: It is the software-visible abstraction, hiding physical implementation details.

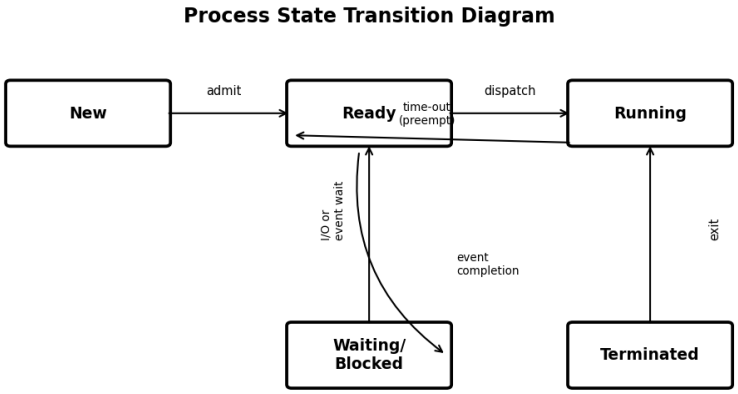
EASY EXPLANATION

4.5 Real-Time Operating and Control System

Process vs Thread

Feature	Process	Thread
Definition	A program in execution with its own address space	A lightweight unit of execution within a process
Memory	Separate, isolated address space	SHARES the parent process's memory and resources
Creation overhead	High	Low
Context switch cost	Expensive	Cheap
Communication	Needs IPC (pipes, shared memory, messages)	Direct — via shared variables
Fault isolation	One crash does not affect others	A crash can bring down the whole process
Own stack / PC?	Yes	Yes (but shares code, data and files)

Process States



State	Meaning	Transition out
New	Process is being created	Admitted → Ready
Ready	Waiting to be assigned to the CPU	Dispatched → Running
Running	Instructions are being executed	Time-out → Ready; I/O request → Waiting; Exit → Terminated
Waiting / Blocked	Waiting for an event or I/O completion	Event completes → Ready
Terminated	Execution finished	—

PCB (Process Control Block) stores everything needed to suspend and resume a process: process ID, state, **program counter**, **CPU registers**, scheduling priority, memory limits, open file list, and accounting information.

CPU Scheduling Algorithms — Detailed Comparison

Algorithm	Preemptive?	Selection criterion	Advantages	Disadvantages
FCFS	No	Arrival order (first come first served)	Simplest; fair in arrival order; no starvation	CONVOY EFFECT — short jobs wait behind long ones; poor average waiting time
SJF	No	Shortest CPU burst time	OPTIMAL average waiting time (provably minimum)	Burst time must be estimated; STARVATION of long processes
SRTF	Yes	Shortest remaining time (preemptive SJF)	Even better average waiting time than SJF	High context-switch overhead; worse starvation
Priority	Either	Highest priority value first	Important jobs run first; flexible	STARVATION of low-priority jobs — fixed by AGING
Round Robin	Yes	Cyclic, each gets a fixed TIME QUANTUM	Fair; excellent response time; no starvation	Performance depends heavily on quantum size; more context switches

Algorithm	Preemptive?	Selection criterion	Advantages	Disadvantages
Multilevel Queue	Either	Separate queue per job class, each with its own policy	Suits mixed workloads (system/interactive/batch)	Inflexible — processes cannot change queue
Multilevel Feedback Queue	Yes	Processes MOVE between queues based on behaviour	Most flexible and adaptive; avoids starvation via aging	Most complex to configure and implement

Turnaround Time TAT = Completion Time − Arrival Time

Waiting Time WT = Turnaround Time − Burst Time

Response Time RT = First CPU allocation time − Arrival Time

Throughput = number of processes completed per unit time

Worked FCFS example: processes arriving at time 0 in the order P1, P2, P3 with burst times 24, 3, 3.

Process	Burst Time	Start	Completion	Turnaround	Waiting
P1	24	0	24	24	0
P2	3	24	27	27	24
P3	3	27	30	30	27

Average waiting time = (0 + 24 + 27) / 3 = 17 ms. If the order were P2, P3, P1 (i.e. SJF), average waiting time drops to **(0+3+6)/3 = 3 ms** — this is the **convoy effect** demonstrated.

Easy Explanation

Key MCQ facts: SJF gives the provably MINIMUM average waiting time but risks starvation; Round Robin gives the best RESPONSE time and never starves; FCFS is simplest but suffers the convoy effect. Aging (gradually raising the priority of waiting processes) is the standard cure for starvation in priority scheduling.

Real-Time Scheduling

Algorithm	Priority basis	Type	Note
Rate Monotonic (RMS)	Shorter period = higher priority	Static priority	Optimal among fixed-priority algorithms
Earliest Deadline First (EDF)	Nearest deadline = highest priority	Dynamic priority	Optimal for uniprocessor; can achieve 100% utilization

Task Synchronization

Mechanism	Type	Key property
Semaphore	Counting or binary integer with wait(P) / signal(V)	Controls access to a pool of resources
Mutex	Binary lock WITH ownership	Only the locking task may unlock it
Monitor	High-level construct: mutual exclusion + condition variables	Easier and safer than raw semaphores
Message passing	send() / receive()	Works across separate address spaces / machines

Priority Inversion: a high-priority task is blocked because a low-priority task holds a needed resource. Fixed by **priority inheritance** (the low-priority holder temporarily inherits the high priority) or **priority ceiling** protocols.

- **Device driver:** software module providing a **standard interface** between the OS/application and specific hardware, hiding device-specific details.

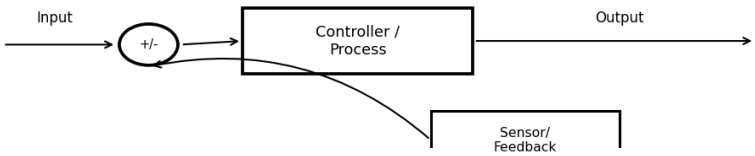
Open-Loop vs Closed-Loop Control Systems

Open-Loop Control System



No feedback — output does NOT affect input (e.g. washing machine timer, toaster)

Closed-Loop Control System



Feedback compares actual output with desired input & corrects error (e.g. thermostat, cruise control)

Feature	Open-Loop	Closed-Loop (Feedback)
Feedback path	Absent	Present — output is measured and compared to input
Accuracy	Lower — cannot self-correct	Higher — errors are automatically corrected
Effect of disturbances	Cannot compensate	Automatically compensated
Stability	Generally stable	Can become unstable if poorly tuned
Complexity & cost	Simple and cheap	Complex and costlier
Examples	Toaster, washing machine timer, traffic light on a timer	Thermostat, cruise control, servo motor, automatic voltage regulator

⍰ Probable MCQs: RTOS and Control Systems

1. Which scheduling algorithm provides the provably MINIMUM average waiting time?

a) FCFS
b) SJF
c) Round Robin
d) Priority
✓Answer: b) SJF
Why: Shortest Job First is optimal for average waiting time, though it may starve long jobs.

2. The convoy effect is a drawback of which scheduling algorithm?

a) FCFS
b) SJF
c) Round Robin
d) SRTF
✓Answer: a) FCFS
Why: In FCFS, short processes queue behind a long one, inflating average waiting time.

3. Starvation in priority scheduling is commonly solved by:

a) Increasing the time quantum
b) Aging
c) Disabling interrupts
d) Using FCFS instead
✓Answer: b) Aging
Why: Aging gradually raises the priority of long-waiting processes so they eventually run.

4. If the Round Robin time quantum becomes very large, RR behaves like:

EASY EXPLANATION

a) SJF
b) FCFS
c) Priority scheduling
d) SRTF
✓Answer: b) FCFS
Why: With a quantum longer than every burst, no preemption occurs, reducing RR to FCFS.

5. Turnaround Time is calculated as:
a) Completion − Arrival
b) Completion − Burst
c) Burst − Waiting
d) Arrival + Burst
✓Answer: a) Completion − Arrival
Why: TAT measures total time spent in the system: completion minus arrival.

6. Earliest Deadline First (EDF) scheduling assigns priority based on:
a) Shortest period
b) Nearest deadline
c) Arrival order
d) Process size
✓Answer: b) Nearest deadline
Why: EDF is a dynamic-priority algorithm favouring the task whose deadline is closest.

7. Priority inversion is resolved using:
a) Round Robin
b) Priority inheritance protocol
c) FCFS ordering
d) Disabling the scheduler
✓Answer: b) Priority inheritance protocol
Why: The low-priority resource holder temporarily inherits the higher priority to finish quickly.

8. Threads of the same process share:
a) Separate address spaces
b) The process's memory and open files
c) Nothing at all
d) Only the program counter
✓Answer: b) The process's memory and open files
Why: Threads share code, data and resources but keep their own stack and registers.

9. A closed-loop control system differs from open-loop by having:
a) No output
b) A feedback path that corrects errors
c) No input
d) Lower cost always
✓Answer: b) A feedback path that corrects errors
Why: Feedback compares actual output against the desired value and corrects the difference.

EASY EXPLANATION

4.6 Hardware Description Language and IC Technology

VHDL Overview

VHDL = **V**HSIC **H**ardware **D**escription **L**anguage (VHSIC = Very High Speed Integrated Circuit). Used to **model, simulate and synthesize** digital circuits.

Design unit	Purpose	Analogy
ENTITY	Declares the INTERFACE — input/output ports	The 'black box' pin diagram
ARCHITECTURE	Describes the INTERNAL behaviour/structure	What is inside the box
CONFIGURATION	Binds a particular architecture to an entity	Choosing which internal version to use
PACKAGE	Reusable declarations (types, functions, constants)	A shared library

VHDL Modeling Styles

Style	Describes	Typical constructs	Example use
Behavioral	WHAT the circuit does (algorithm)	process, if-else, case, loops	High-level function description
Dataflow	How data FLOWS between signals	Concurrent signal assignment (<=), when-else	Boolean equations
Structural	HOW components are interconnected	component, port map	Netlist / schematic equivalent

VHDL Data Types and Operators

Data type	Description	Example
std_logic	9-valued single bit ('0','1','Z','X','U'...)	`signal a : std_logic;`
std_logic_vector	Array of std_logic	`signal b : std_logic_vector(7 downto 0);`
bit / bit_vector	Simple 2-valued ('0','1')	`signal c : bit;`
integer	Whole numbers	`signal d : integer range 0 to 255;`
boolean	true / false	`variable e : boolean;`

Overflow and data representation: VHDL requires explicit bit widths. Adding two 8-bit numbers can produce a 9-bit result, so the target signal must be widened (or a carry-out declared) — otherwise the MSB is lost, which is an **overflow**.

Combinational Logic in VHDL

Program 1: Half adder (dataflow style)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity half_adder is
    port ( a, b   : in  std_logic;
           sum, carry : out std_logic );
end half_adder;

architecture dataflow of half_adder is
begin
    sum   <= a xor b;      -- concurrent statements
    carry <= a and b;
end dataflow;
```

Program 2: Combinational logic using a process

```
-- Same combinational logic written as a PROCESS
process(a, b)              -- ALL inputs in the sensitivity list
begin
    sum   <= a xor b;
```

EXPLANATION

```
    carry <= a and b;
end process;
```

Sequential Logic in VHDL

Program 3: D flip-flop (sequential)

```
-- D flip-flop with asynchronous reset
process(clk, reset)        -- ONLY clock (and reset) in sensitivity list
begin
    if reset = '1' then
        q <= '0';
    elsif rising_edge(clk) then    -- edge-triggered
        q <= d;
    end if;
end process;
```

Aspect	Combinational process	Sequential process
Sensitivity list	ALL input signals used	Clock only (plus async reset)
Edge detection	None	rising_edge(clk) / falling_edge(clk)
Memory	No — output depends only on current inputs	Yes — output depends on past and present
Synthesizes to	Gates / multiplexers	Flip-flops / registers

Easy Explanation

The fastest way to classify a VHDL process in an exam: look ONLY at the sensitivity list. If it contains all the input signals and there is no rising_edge, it is COMBINATIONAL. If it contains just the clock and uses rising_edge/falling_edge, it is SEQUENTIAL and will synthesize to flip-flops.

Pipelining in VHDL

Program 4: Two-stage pipeline

```
-- Inserting a pipeline register between two combinational stages
process(clk)
begin
    if rising_edge(clk) then
        stage1_reg <= a + b;          -- stage 1 result latched
        stage2_reg <= stage1_reg * c; -- stage 2 works on previous cycle's result
    end if;
end process;
```

Each clocked register separates one combinational stage from the next, so all stages can operate **concurrently on different data** — increasing throughput at the cost of extra latency and registers.

IC Technology Classification

Technology	Configurability	When customized	Best suited for
Full-custom / VLSI (ASIC)	None — fixed at fabrication	At manufacture (all layers)	Very high volume, maximum performance
Semi-custom (Gate array / Standard cell)	Partial	Some layers customized	Medium volume
PLD / FPGA	Fully reconfigurable by the user	After manufacture, in the field	Prototyping, low-medium volume

Scale of integration	Gates per chip	Example
SSI (Small)	< 10	Basic logic gate ICs (7400 series)
MSI (Medium)	10 – 100	Counters, multiplexers, decoders
LSI (Large)	100 – 10,000	Simple microprocessors, memory
VLSI (Very Large)	10,000 – 1 million	Modern CPUs, GPUs
ULSI (Ultra Large)	> 1 million	Current multicore processors

? Probable MCQs: VHDL and IC Technology

1. In VHDL, which design unit declares only the input/output ports?

a) ARCHITECTURE

b) ENTITY

c) PACKAGE

d) PROCESS

✓Answer: b) ENTITY

Why: The ENTITY defines the external interface; ARCHITECTURE defines the internals.

2. Which VHDL modeling style uses component instantiation and port map?

a) Behavioral

b) Dataflow

c) Structural

d) Sequential

✓Answer: c) Structural

Why: Structural modeling wires together sub-components, like a netlist.

3. A VHDL process modeling COMBINATIONAL logic must include in its sensitivity list:

a) Only the clock

b) All input signals used in the process

c) No signals

d) Only the outputs

✓Answer: b) All input signals used in the process

Why: Combinational logic must react to any input change, so all inputs must be listed.

4. 'rising_edge(clk)' in a VHDL process indicates:

a) Combinational logic

b) Edge-triggered sequential logic

c) An analog circuit

d) A syntax error

✓Answer: b) Edge-triggered sequential logic

Why: It detects the clock transition, synthesizing to flip-flops.

5. Pipelining in VHDL is implemented by inserting:

a) More combinational gates only

b) Clocked registers between combinational stages

c) Extra entities

d) Additional packages

✓Answer: b) Clocked registers between combinational stages

Why: Pipeline registers latch intermediate results each clock, enabling stage overlap.

6. Which data type is most commonly used for single-bit signals in synthesizable VHDL?

a) integer

EASY EXPLANATION

b) boolean

c) std_logic

d) real

✓Answer: c) std_logic

Why: std_logic supports multiple drive strengths and high-impedance, essential for real hardware.

7. An FPGA differs from an ASIC because an FPGA is:

a) Fixed function after manufacture

b) Reconfigurable by the user after manufacture

c) Always faster than an ASIC

d) Cheaper at very high volumes

✓Answer: b) Reconfigurable by the user after manufacture

Why: FPGAs are field-programmable; ASICs are fixed at fabrication.

8. A chip containing more than one million gates is classified as:

a) MSI

b) LSI

c) VLSI

d) ULSI

✓Answer: d) ULSI

Why: ULSI (Ultra Large Scale Integration) exceeds one million gates per chip.

EASY EXPLANATION