3.5 Finite state machines

Name: _____ Class: _____ Date: _____ **Total: 21 marks**

KEY WORDS finite state machine • state diagram • Moore machine • Mealy machine

Objective

Design finite state machines from a plain-English behaviour: state diagram → state table → encodings → excitation equations, choosing Moore or Mealy deliberately, and merge redundant states.

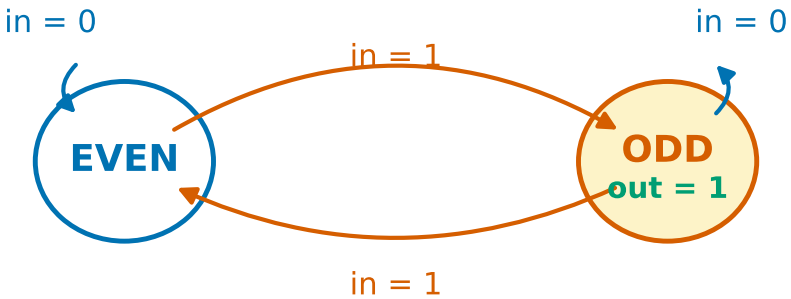
1 Key knowledge and worked examples

■ States, transitions, outputs

A **finite state machine** (FSM) = states + transitions + outputs. A **state diagram** draws states as bubbles, transitions as labelled arrows (input/output or input per arrow). **Moore** outputs depend on state only; **Mealy** outputs depend on state AND inputs – faster (same-cycle) but glitch-prone.

Design procedure: (1) state diagram from the behaviour; (2) **state table** (next state + output per state/input); (3) binary encoding; (4) excitation table per flip-flop; (5) minimise next-state logic (Karnaugh maps).

The parity detector: two states, one flip-flop



next-state logic: $Q^+ = Q \oplus in$ — the machine is a parity bit

■ **Worked example: the parity detector**

Detect odd number of 1s. Two states: EVEN, ODD.

state	in=0 →	in=1 →	output
EVEN	EVEN	ODD	0
ODD	ODD	EVEN	1

Trace on `1, 0, 1, 1`: EVEN → ODD → ODD → EVEN → ODD. Output after the fourth bit = 1: three 1s seen (odd ✓). Moore encoding: 1 flip-flop (Q = 1 ⇔ ODD); next-state equation: $Q^+ = Q \oplus in$ – the machine is one XOR and a flip-flop, which the trace confirms: $0 \oplus 1 = 1, 1 \oplus 0 = 1, 1 \oplus 1 = 0, 0 \oplus 1 = 1$ ✓.

■ **Minimisation: merge equivalent states**

Two states are equivalent if their outputs match AND their next states (per input) are equivalent. **Worked example**: a machine with states S2, S3 that both output 1 and both transition to S1 on 0, S0 on 1 – S2 and S3 are equivalent; merging them costs nothing in behaviour and saves a state (possibly a flip-flop).

2 Practice

2.1 For the parity FSM: which state follows ODD on input 1, and what is the output? [2]

2.2 Name the one structural difference between Moore and Mealy outputs, and one consequence of each. [3]

2.3 A two-state FSM has states A, B: both output 0, and both go to B on input 0 and to A on input 1. Are A and B equivalent? Why? [3]

3 Exam-style questions

3.1 Design a Moore FSM that raises its output for exactly one clock after every $1 \to 0$ input fall (a fall detector; output otherwise 0).

Draw (describe) the state diagram with states named for their meaning, write the state table, encode with one flip-flop, and derive the output equation. Trace the input `1, 1, 0, 0, 1, 0` giving the output after each bit. [7]

3.2 A vending machine's coin slot accepts only 10p and 20p coins; a 30p item vends and resets to zero credit; overpayment is held (credit 30 or 40 stays "paid").

Model the credit states and transitions (coin inputs only), give the state table, and state how many flip-flops the encoded machine needs and why. Then explain what changes if a "coin return" input must refund exact credit. [6]

Solutions

2.1

- ODD on input 1 → EVEN (a 1 flips the parity).
- Output = 1 (state ODD).

2.2

- Moore: outputs from state only – outputs change after the clock edge; Mealy: outputs from state and inputs – outputs can change mid-cycle.
- Moore consequence: glitch-free, aligned outputs (a cycle later at the earliest).
- Mealy consequence: faster response, but outputs can glitch with input transitions.

2.3

- Yes – equivalent.
- Both output 0; on input 0 both go to B, on input 1 both go to A: same outputs, same next states.
- Merging them leaves the external behaviour unchanged (one state suffices).

3.1

- States: S1 ("last input was 1"), S0 ("last input was 0, not just after a fall"), F ("fall just happened"). Transitions: S1 on 0 → F; S1 on 1 → S1; F on 0 → S0; F on 1 → S1; S0 on 0 → S0; S0 on 1 → S1.
- Outputs: F = 1, S1 = S0 = 0.
- State table:

state	in=0 →	in=1 →	out
S0	S0	S1	0
S1	F	S1	0
F	S0	S1	1

- Encoding: two flip-flops (three states need $\lceil \log_2 3 \rceil = 2$).
- Output equation: $OUT = state_i s_F$ – a single product term of the encoding.
- Trace `1,1,0,0,1,0`: S0→S1→S1→F→S0→S1→F; outputs after each input `0, 0, 1, 0, 0, 1`.

3.2

- Credit states: C0, C10, C20 (C30+ = PAID, output vend).
- State table: C0 +10→C10, +20→C20; C10 +10→C20, +20→PAID; C20 +10→PAID, +20→PAID (overpayment held); PAID → (vend, reset) C0.
- Four states → 2 flip-flops ($\lceil \log_2 4 \rceil = 2$).
- Coin return: every state needs a transition to C0 with output = current credit (refund) – the outputs become state-dependent amounts, adding output logic (or a refund output pulse plus a counter).