UNIVERSITY ELECTRICAL ENGINEERING • EXERCISE SHEET

3.6 Registers, counters and memory

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

KEY WORDS register • shift register • ripple counter • synchronous counter • memory

Objective

Build registers and counters (ripple and synchronous), trace shift-register data flow, and size memory from address/data line counts – every counter trace checked against the 2^n state-count rule.

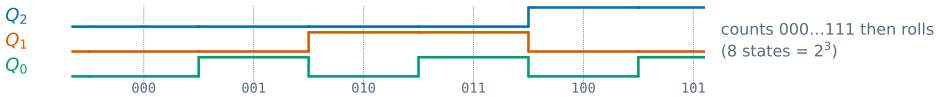
1 Key knowledge and worked examples

■ Registers: n bits per edge

A **register** is n flip-flops sharing a clock – n bits load at each edge. A **shift register** connects each Q to the next D: bits march along one place per clock. Serial-in serial-out delays a stream by n clocks; serial-in parallel-out is the serial-to-parallel converter.

Worked example. A 4-bit shift register (`q <= {q[2:0], sin}`: the new bit enters the rightmost stage, contents shift left), serial input `1, 0, 1, 1`: after four clocks the register holds `1011` – the last input sits at the far right, the first at the far left. Trace: 0001 → 0010 → 0101 → 1011 ✓.

One clock, computed toggles: the 3-bit up-counter



■ Counters: ripple vs synchronous

A **ripple counter** chains T-flip-flops, each clocked by the previous output – simple, but the delay accumulates stage to stage. A **synchronous counter** clocks every flip-flop from the same edge and computes each toggle with logic – fast, and the standard design.

Worked example. The 3-bit synchronous up-counter: bit 0 toggles every clock; bit 1 toggles when `Q0 = 1`; bit 2 toggles when $Q1 \cdot Q0 = 11$. From 000 the sequence runs 000, 001, 010, 011, 100 … 111, then wraps to 000 – $2^3 = 8$ states ✓ (the count-range check every counter answer must pass).

■ Memory: address bits select words

Memory is an array of m-bit words; k **address lines** select one of 2^k words. $2^k \times m$ is the organisation: 8K×8 has $k = 13$ (since $8K = 2^13$) and 8 data lines. The sizing rule both ways: `k = log2(number of words)`; words = `2^k`.

2 Practice

2.1 A 4-bit register loads `1011` at one rising edge. A 4-bit shift register with serial input 1, 0, 1, 1 (new bit entering the rightmost stage, contents shifting left, as in `q <= {q[2:0], sin}`) clocks four times: give the contents after each clock. [3]

2.2 How many states does a 4-bit counter have, and what is the largest count before wrap? [2]

2.3 A memory is 4K × 16. How many address lines and data lines? [3]

3 Exam-style questions

3.1 Design a 3-bit synchronous DOWN-counter (counts 7, 6, 5 … 0, 7 …) using T flip-flops.

Give the toggle condition for each bit, trace the first six counts from 111, and verify the wrap 000 → 111 with your conditions. Then state why this counter's outputs all change relative to one clock edge (and what a ripple version would cost). [7]

3.2 A 4-bit serial-in parallel-out shift register receives the bits `1, 1, 0, 1` (first bit first) and clocks four times.

Give the register contents after each clock. Then explain how this register plus a counter converts a 4-bit serial stream to a parallel word, and state the total conversion time in clock cycles. An 8-bit conversion of the same rate would take how many cycles, and how many flip-flops in total? [6]

Solutions

2.1

- Convention (as in the handout's `q <= {q[2:0], sin}`): each new bit enters the rightmost stage and the contents shift left. Trace: 0001 → 0010 → 0101 → `1011` after four clocks – the LAST input sits at the far right, the first at the far left.
- (Each clock shifts the previous contents one place; the fourth entry lands in the last stage ✓.)

2.2

- $2^4 = 16$ states.
- Largest count 15 (1111); wrap to 0 after.

2.3

- $4K = 2^{12} \rightarrow 12$ address lines.
- 16 data lines (16-bit words).

3.1

- Bit 0 toggles every clock.
- Bit 1 toggles when `Q0 = 0` (i.e. `Q0' = 1`).
- Bit 2 toggles when `Q1 = 0 AND Q0 = 0` ($Q1' \cdot Q0'$).
- Trace from 111 (toggle decisions use the CURRENT state): 111 → 110 (b0 1→0; b1 holds – Q0 was 1; b2 holds – Q1 was 1) → 101 → 100 → 011 (b0 0→1; b1 toggles – Q0 was 0; b2 toggles – Q1Q0 was 00) → 010 → 001.
- Wrap 000: b0 → 1, b1 → 1, b2 → 1 → 111 ✓ (all toggle on 000).
- All flip-flops share the one clock, so changes are aligned to a single edge; a ripple version would accumulate one flip-flop delay per stage, misaligning the wider bits' change times.

3.2

- New bit enters the rightmost stage, contents shift left (the handout's `q <= {q[2:0], sin}`): after clock 1 `0001`, clock 2 `0011`, clock 3 `0110`, clock 4 `1101` – the last input (1) at the far right, the first (1) at the far left.
- Conversion: the shift register collects bits while a 2-bit counter counts to four; at count 4 the parallel outputs latch – 4 clock cycles total.
- 8-bit conversion: 8 cycles; 8 shift-register flip-flops + 3 counter flip-flops = 11 flip-flops total.