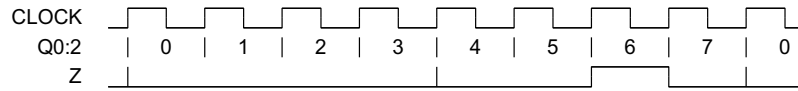


ELEC50001 EE2 Circuits and Systems

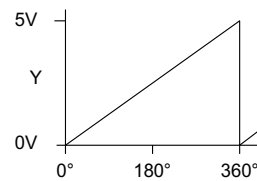
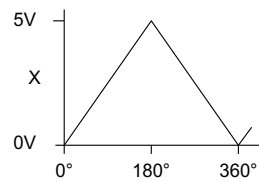
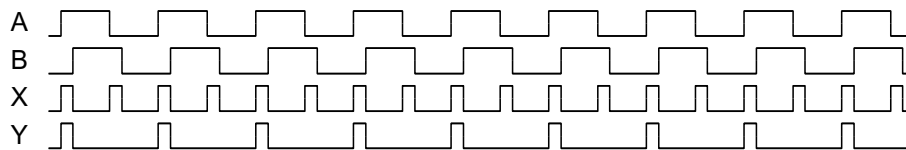
Problem Sheet 4 Solutions

(Counters and Shift Registers – Lecture 7)

- 1B. $Z = Q_2 \cdot Q_1 \cdot \sim Q_0$. Note that (a) Q_2 is always the MSB and (b) we must include the $\sim Q_0$ term. Glitches in Z are possible for the transitions $3 \rightarrow 4$ and $7 \rightarrow 0$.



- 2C. The XOR gate goes high twice per cycle whereas the more complicated circuit only goes high once per cycle. The advantage of the complicated circuit is that it covers a full 360° monotonically.

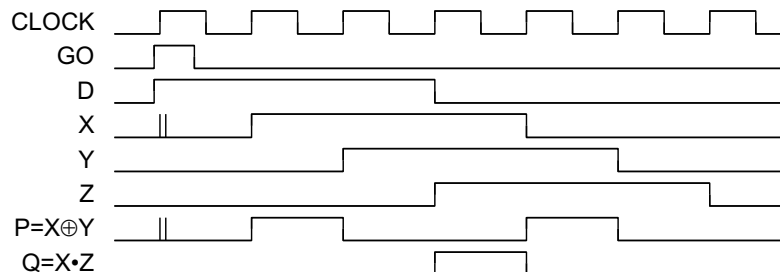


- 3B. $Z = B \oplus C + \sim D \cdot E$

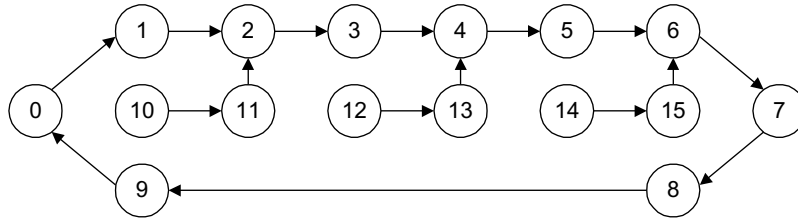
Note that since this expression does not involve A , it will be glitch-free/

- 4C. The output of the first shift-register stage can go metastable if $D \uparrow$ occurs just before the $CLOCK \uparrow$ edge. This will only affect the P output because Z will be low at the time which will force Q low regardless of X .

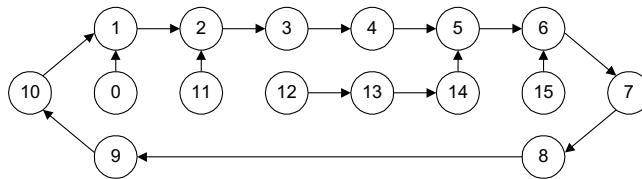
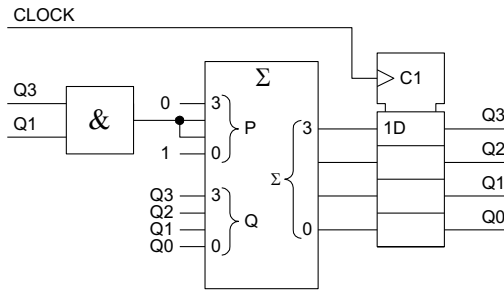
The average time delay between $GO \uparrow$ and $Q \uparrow$ will be $2\frac{1}{2}$ clock periods.



5C. The P input of the adder equals 7 when Q is 9, 11, 13 or 15. For all other values of Q it equals 1. Bearing in mind that the adder result is modulo 16 (i.e. $10+7=1$), this results in the following state diagram:



6C. We want to make 10 the maximum count rather than 9, so we need to detect when Q3 and Q1 are high. We will now add 7 onto Q in states 10, 11, 14 and 15.



7B.

$1+X^3+X^4$		X^3+X^4	$1+X+X^4$		$X+X^4$
binary	decimal	next LSB	binary	decimal	next LSB
0001	1	0	0001	1	1
0010	2	0	0011	3	1
0100	4	1	0111	7	1
1001	9	1	1111	15	0
0011	3	0	1110	14	1
0110	6	1	1101	13	0
1101	13	0	1010	10	1
1010	10	1	0101	5	1
0101	5	1	1011	11	0
1011	11	1	0110	6	0
0111	7	1	1100	12	1
1111	15	0	1001	9	0
1110	14	0	0010	2	0
1100	12	0	0100	4	0
1000	8	1	1000	8	1

8B. According to table in Lecture 5 slide 17, a 7-bit LFSR primitive polynomial is $1 + X^3 + X^7$.

```

module lfsr_7 (
    clk,
    enable,
    prbs
);

    parameter BIT_SZ = 7;
    input  clk, enable;
    output [BIT_SZ-1:0] prbs;

    reg [BIT_SZ:1] sreg;

    initial sreg = 7'd1;

    always @ (posedge clk)
        if (enable==1'b1) begin
            sreg[BIT_SZ:2] <= sreg[BIT_SZ-1:1];
            sreg[1] <= sreg[BIT_SZ] ^ sreg[3];
        end

    assign prbs = sreg;

endmodule

```

9C. Here is a 1kHz clock with a high pulse of 20ns every microsecond:

```

module clktick_1us (
    clk,          // Clock input to the design
    tick         // pulse_out goes high for one cycle (n+1) clock cycles
);             // End of port list

parameter N_BIT = 16; // 16-bit needed to div 20MHz by 50,000
parameter TC = 16'd49999; // Terminal count is one less
//-----Input Ports-----
input clk;
input [N_BIT-1:0] N;

//-----Output Ports-----
output tick;

//-----Output Ports Data Type-----
// Output port can be a storage element (reg) or a wire
reg [N_BIT-1:0] count;
reg tick;

initial tick = 1'b0;

//----- Main Body of the module -----

always @ (posedge clk)
    if (count == 0) begin
        tick <= 1'b1;
        count <= TC;
    end
    else begin
        tick <= 1'b0;
        count <= count - 1'b1;
    end

endmodule // End of Module clktick

```