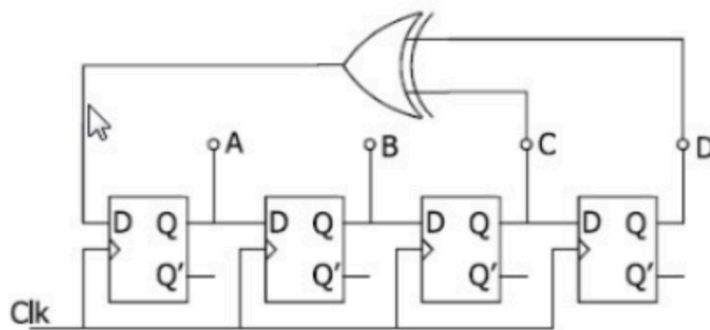


Exercise 1 (The figure below shows a 4-stage shift register made out of D-flip-flops.

The Q output from the last 2 stages are used as inputs for an XOR gate.

The output of the XOR gate is fed to the 1st flip-flop as an input.

As initial value the 1st stage holds an 1 (A=1). The subsequent stages hold zeros.):



a) Determine the content of register stages after each of the the sub-sequent clock pulses and put results in a table.

CS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0	1
B	0	1	0	0	1	1	0	1	0	1	1	1	1	0	0	0
C	0	0	1	0	0	1	1	0	1	0	1	1	1	1	0	0
D	0	0	0	1	0	0	1	1	0	1	0	1	1	1	1	0

Hver 16. kommer vi tilbage til start, altså b1000, det vil sige vi har 15 forskellige stadier der kører i cyklus. Kan blandt andet bruges til kryptografi

b) What happens if all stages are initial zero.

Så forbliver det 0, b0000

Exercise 2:

```

entity bitshifter is
    Port ( klok : in STD_LOGIC;
          switch : in STD_LOGIC;
          LED : out std_logic_vector(3 downto 0)
        );
end bitshifter;

architecture Behavioral of bitshifter is

begin
    process(klok)
        variable buff : std_logic_vector(3 downto 0) := (others => '0');

    begin

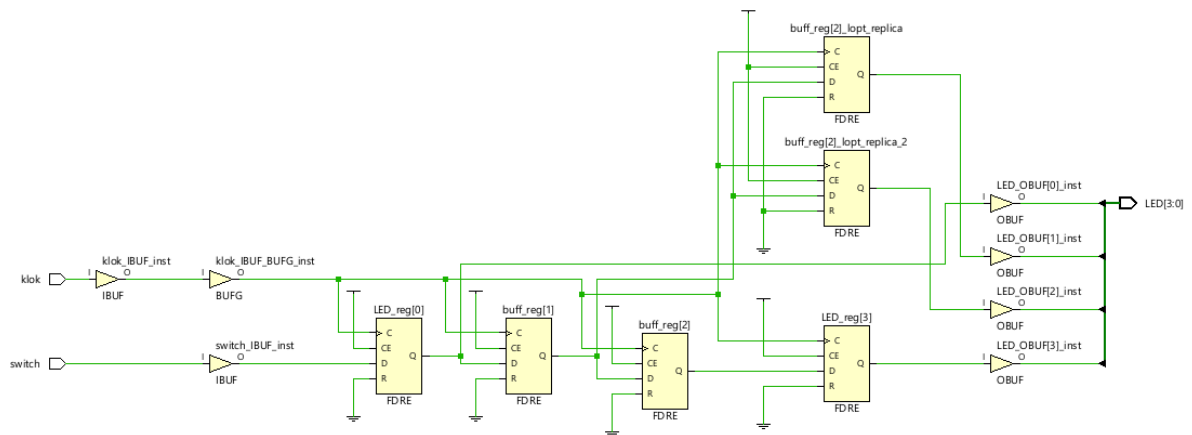
        if(klok'event and klok = '1') then

            buff(3) := buff(2);
            buff(2) := buff(1);
            buff(1) := buff(0);
            buff(0) := switch;

            LED(3) <= buff(3);
            LED(2) <= buff(2);
            LED(1) <= buff(1);
            LED(0) <= buff(0);

        end if;
    end process;
end Behavioral;

```



Exercise 3:

```

34 entity bitshifter is
35     Port ( klok : in STD_LOGIC;
36           switch : in STD_LOGIC;
37           LED : out std_logic_vector(3 downto 0)
38           );
39 end bitshifter;
40
41 architecture Behavioral of bitshifter is
42     signal buff : std_logic_vector(3 downto 0) := "0001";
43 begin
44
45     process(klok)
46
47     begin
48         if(klok'event and klok = '1') then
49
50
51
52             buff(3) <= buff(2);
53             buff(2) <= buff(1);
54             buff(1) <= buff(0);
55             buff(0) <= buff(3) XOR buff(2);
56
57             LED(3) <= buff(3);
58             LED(2) <= buff(2);
59             LED(1) <= buff(1);
60             LED(0) <= buff(0);
61         end if;
62     end process;
63 end Behavioral;

```

Vi bruger signal i stedet for variable, da variable virker anderledes

Exercise 4:

Det passer fint med skemaet nu!