Digital Systems II - DGD 3

We have a circuit with 3 global inputs, however each entry level component only has 2 inputs. This is a signal vs an input.

You can use mapping to specify which port will be connected to which input

I.E. with inputs A_top, B_top, C_top

You can do

- map(A1 => A_top, B1 => B_top)

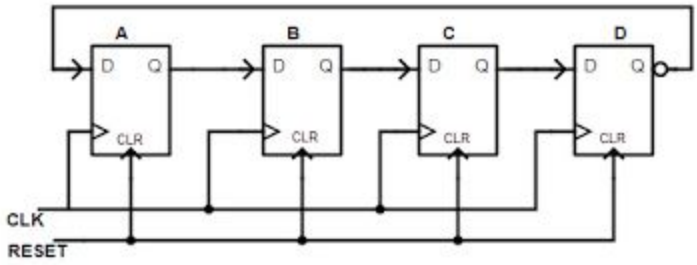
Or you can do

- map(A_top, B_top)

EN	A1	A0	D0	D1	D2	D3
0	DNC	DNC	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

Q2

Start with circuit, initialized with 1000



Following the right most flipflop, it will assign its value to the first and all the others assign their value to the flipfop to its right.

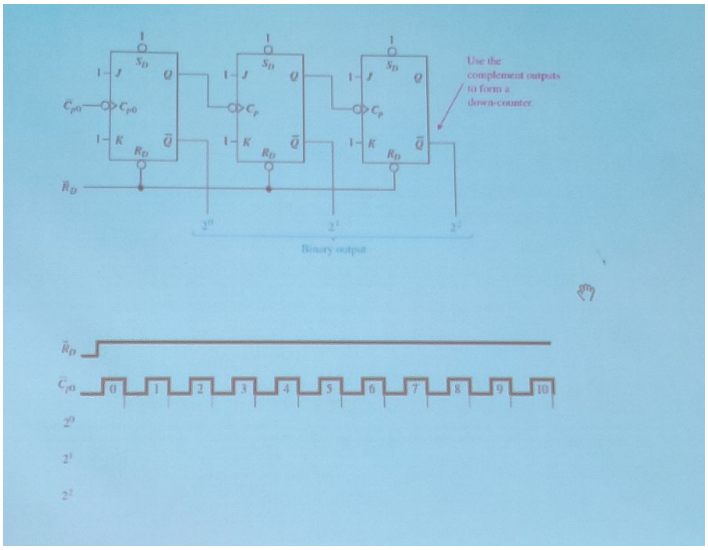
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
1 0 0 0

We have 4 states.

Assume we change the assignment of the first to $D_Q \oplus C_Q$

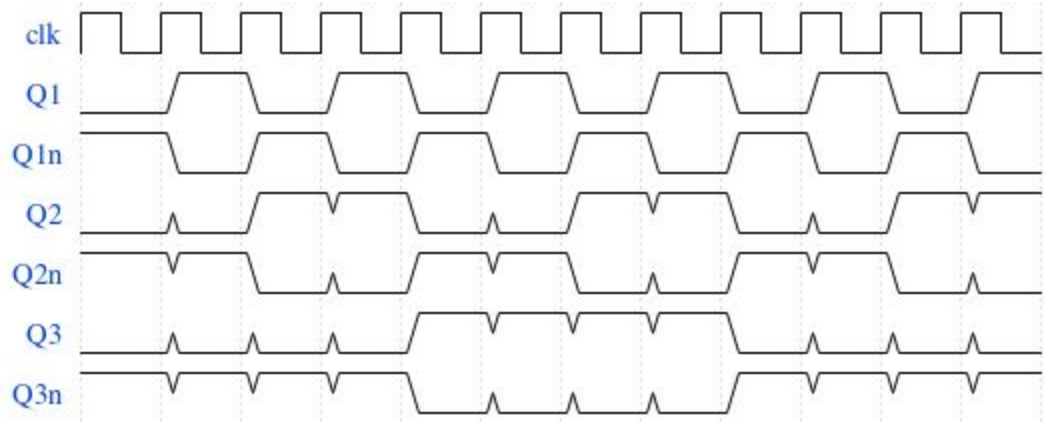
Starting with the same value, 15 states.

1 0 0 0
0 1 0 0
0 0 1 0
1 0 0 1
1 1 0 0
0 1 1 0
1 0 1 1
0 1 0 1
1 0 1 0
1 1 0 1
1 1 1 0
1 1 1 1
0 1 1 1
0 0 1 1
0 0 0 1
1 0 0 0



j	K	Q
0	0	No change
0	1	0
1	0	1
1	1	Toggle Q`

Left to right, the Qs are Q_0, Q_1, Q_2



Question 5:

We wish to design a shift-left circuit manually. The inputs include a, which is an 8-bit signal to be shifted, and ctrl, which is a 3 bit signal specifying the amount to be bitshifted. Both are with the std_logic_vector data type. The output Y is an 8-bit signal with the std_logic_vector data type. Use concurrent signal assignment statements to derive the circuit and draw the conceptual diagram.

```
entity {
    (
        a: in std_logic_vector(7 downto 0)
        ctl: in std_logic_vector(2 downto 0)
        y: out std_logic_vector(7 downto 0)
    )
}

begin
    with ctrl sel
    y <= a                          when "000",
         a(6 downto 0) & "0"        when "001",
         a(5 downto 0) & "00"       when "010",
         a(4 downto 0) & "000"      when "011",
         a(3 downto 0) & "0000"     when "100",
         a(2 downto 0) & "00000"    when "101",
         a(1 downto 0) & "000000"   when "110",
         a(0)          & "0000000"  when others;
end;
```