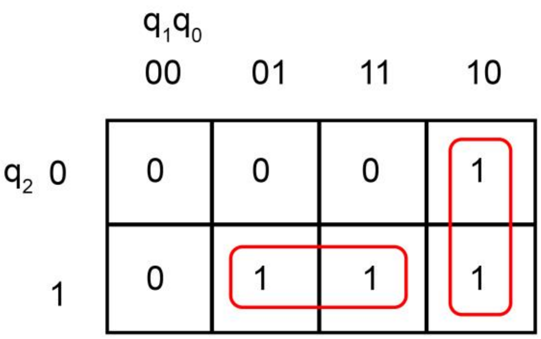
Counter in Gray code, variable number of bits, up/down, reset

Gray code 3 bit up counter

	Present state			Next state		
Dec	q_2	q_1	q_0	d_2	d_1	d_0
0	0	0	0	0	0	1
1	0	0	1	0	1	1
2	0	1	1	0	1	0
3	0	1	0	1	1	0
4	1	1	0	1	1	1
5	1	1	1	1	0	1
6	1	0	1	1	0	0
7	1	0	0	0	0	0

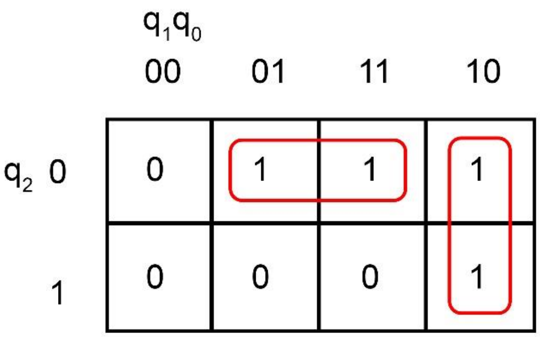
d_2:



q_2	q_1q_0 00	01	11	10
0	0	0	0	1
1	0	1	1	1

$d_2 = q_2q_0 + q_1/q_0$

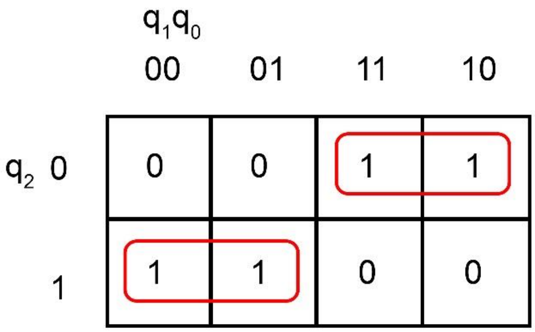
d_1 :



q_2	q_1q_0 00	01	11	10
0	0	1	1	1
1	0	0	0	1

$d_1 = /q_2q_0 + q_1/q_0$

d_0:



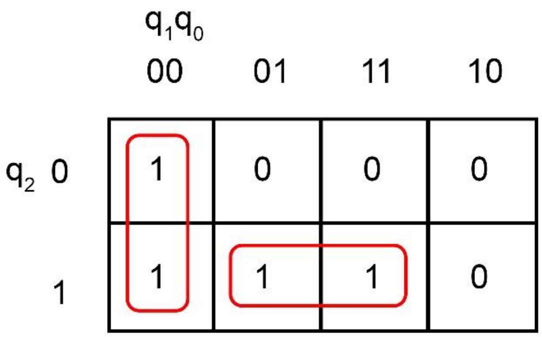
	q_1q_0 00	01	11	10
q_2 0	0	0	1	1
1	1	1	0	0

<u>Gray code 3 bit down counter</u>

	Present state			Next state		
Dec	q_2	q_1	q_0	d_2	d_1	d_0
0	0	0	0	1	0	0
7	1	0	0	1	0	1
6	1	0	1	1	1	1
5	1	1	1	1	1	0
4	1	1	0	0	1	0
3	0	1	0	0	1	1
2	0	1	1	0	0	1
1	0	0	1	0	0	0

K maps

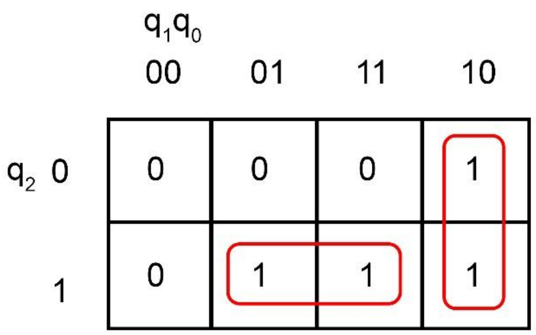
d_2:



	q_1q_0 00	01	11	10
q_2 0	1	0	0	0
1	1	1	1	0

$d_2 = /q_1/q_0 + q_2q_0$

d_1:

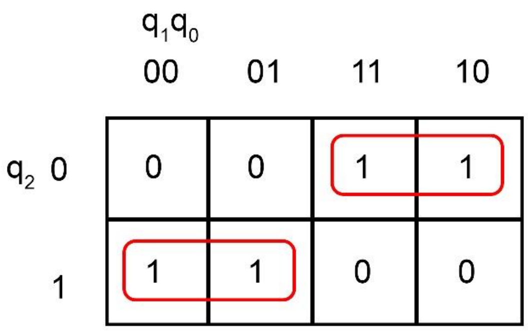


q_1q_0

	00	01	11	10
q_2 0	0	0	0	1
1	0	1	1	1

$d_1 = q_2q_0 + q_1/q_0$

d_0:



q_1q_0

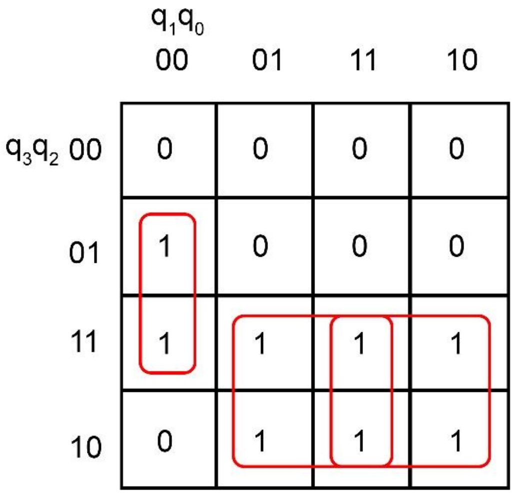
	00	01	11	10
q_2 0	0	0	1	1
1	1	1	0	0

$d_0 = q_2/q_1 + /q_2q_1$

Gray code 4 bit up counter

	Present state				Next state			
Dec	q_3	q_2	q_1	q_0	d_3	d_2	d_1	d_0
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	1
2	0	0	1	1	0	0	1	0
3	0	0	1	0	0	1	1	0
4	0	1	1	0	0	1	1	1
5	0	1	1	1	0	1	0	1
6	0	1	0	1	0	1	0	0
7	0	1	0	0	1	1	0	0
8	1	1	0	0	1	1	0	1
9	1	1	0	1	1	1	1	1
10	1	1	1	1	1	1	1	0
11	1	1	1	0	1	0	1	0
12	1	0	1	0	1	0	1	1
13	1	0	1	1	1	0	0	1
14	1	0	0	1	1	0	0	0
15	1	0	0	0	0	0	0	0

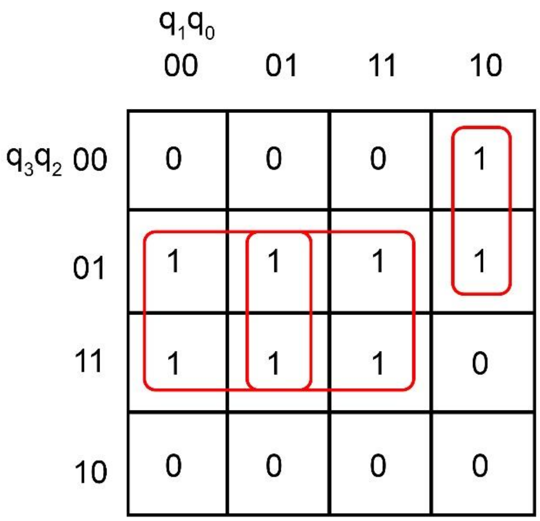
d_3:



q_3q_2 \ q_1q_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	1	1
10	0	1	1	1

$d_3 = q_2/q_1/q_0 + q_3q_0 + q_3q_1$

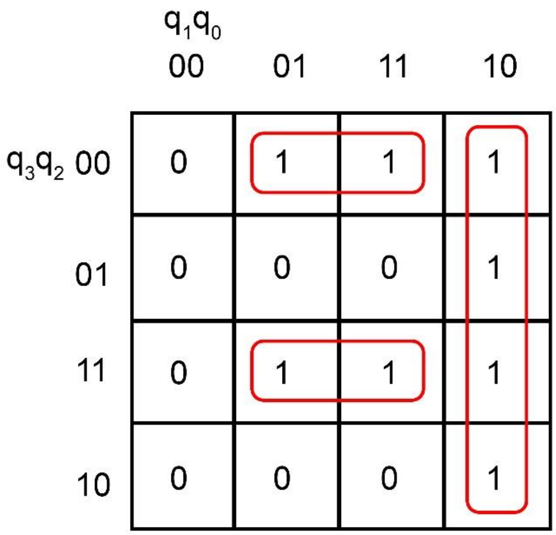
d_2:



	q_1q_0 00	01	11	10
q_3q_2 00	0	0	0	1
01	1	1	1	1
11	1	1	1	0
10	0	0	0	0

$d_2 = q_2/q_1 + q_2q_0 + /q_3q_1/q_0$

d_1:



	q_1q_0 00	01	11	10
q_3q_2 00	0	1	1	1
01	0	0	0	1
11	0	1	1	1
10	0	0	0	1

$d_1 = /\ q_3/q_2q_0 + q_3q_2q_0 + q_1/q_0$

d_0:

q_1q_0

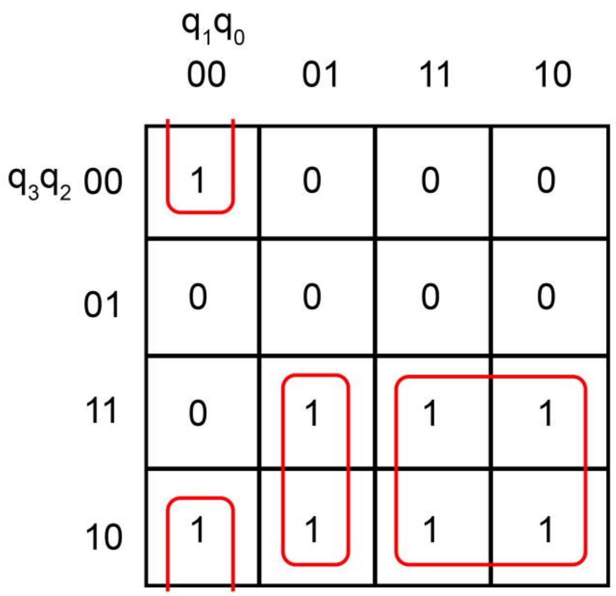
	00	01	11	10
q_3q_2 00	1	1	0	0
01	0	0	1	1
11	1	1	0	0
10	0	0	1	1

$d_0 = /q_3/q_2/q_1 + /q_3q_2q_1 + q_3q_2q_1 + q_3/q_2q_1$

Gray code 4 bit down counter

	Present state				Next state			
Dec	q_3	q_2	q_1	q_0	d_3	d_2	d_1	d_0
15	1	0	0	0	1	0	0	1
14	1	0	0	1	1	0	1	1
13	1	0	1	1	1	0	1	0
12	1	0	1	0	1	1	1	0
11	1	1	1	0	1	1	1	1
10	1	1	1	1	1	1	0	1
9	1	1	0	1	1	1	0	0
8	1	1	0	0	0	1	0	0
7	0	1	0	0	0	1	0	1
6	0	1	0	1	0	1	1	1
5	0	1	1	1	0	1	1	0
4	0	1	1	0	0	0	1	0
3	0	0	1	0	0	0	1	1
2	0	0	1	1	0	0	0	1
1	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	0	0

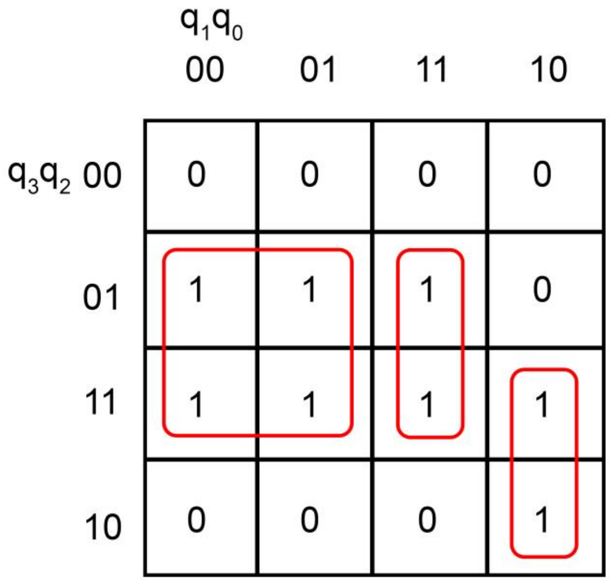
d_3:



	q_1q_0			
	00	01	11	10
q_3q_2 00	1	0	0	0
01	0	0	0	0
11	0	1	1	1
10	1	1	1	1

$d_3 = /q_2/q_1/q_0 + q_3/q_1q_0 + q_3q$

d_2:

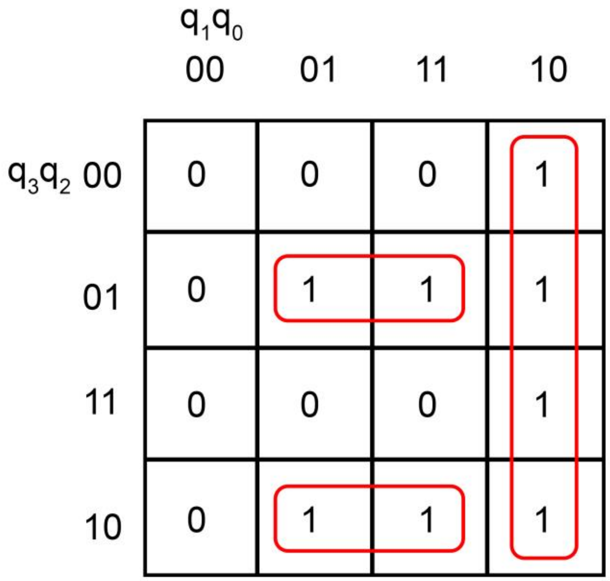


q_1q_0

	00	01	11	10
q_3q_2 00	0	0	0	0
01	1	1	1	0
11	1	1	1	1
10	0	0	0	1

$d_2 = q_2/q_1 + q_2q_1q_0 + q_3q_1/q_0$

d_1:



q_1q_0

	00	01	11	10
q_3q_2 00	0	0	0	1
01	0	1	1	1
11	0	0	0	1
10	0	1	1	1

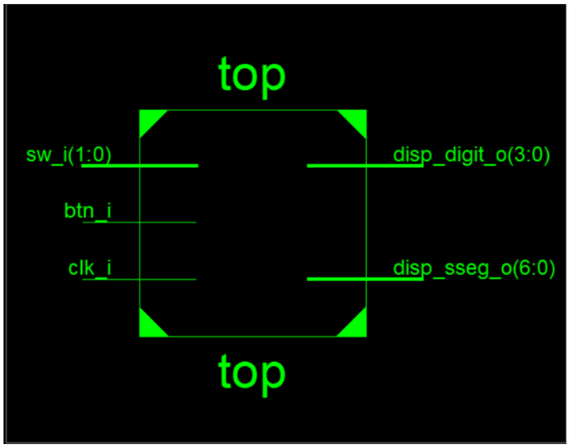
$d_1 = /q_3q_2q_0 + q_3/q_2q_0 + q_1/q_0$

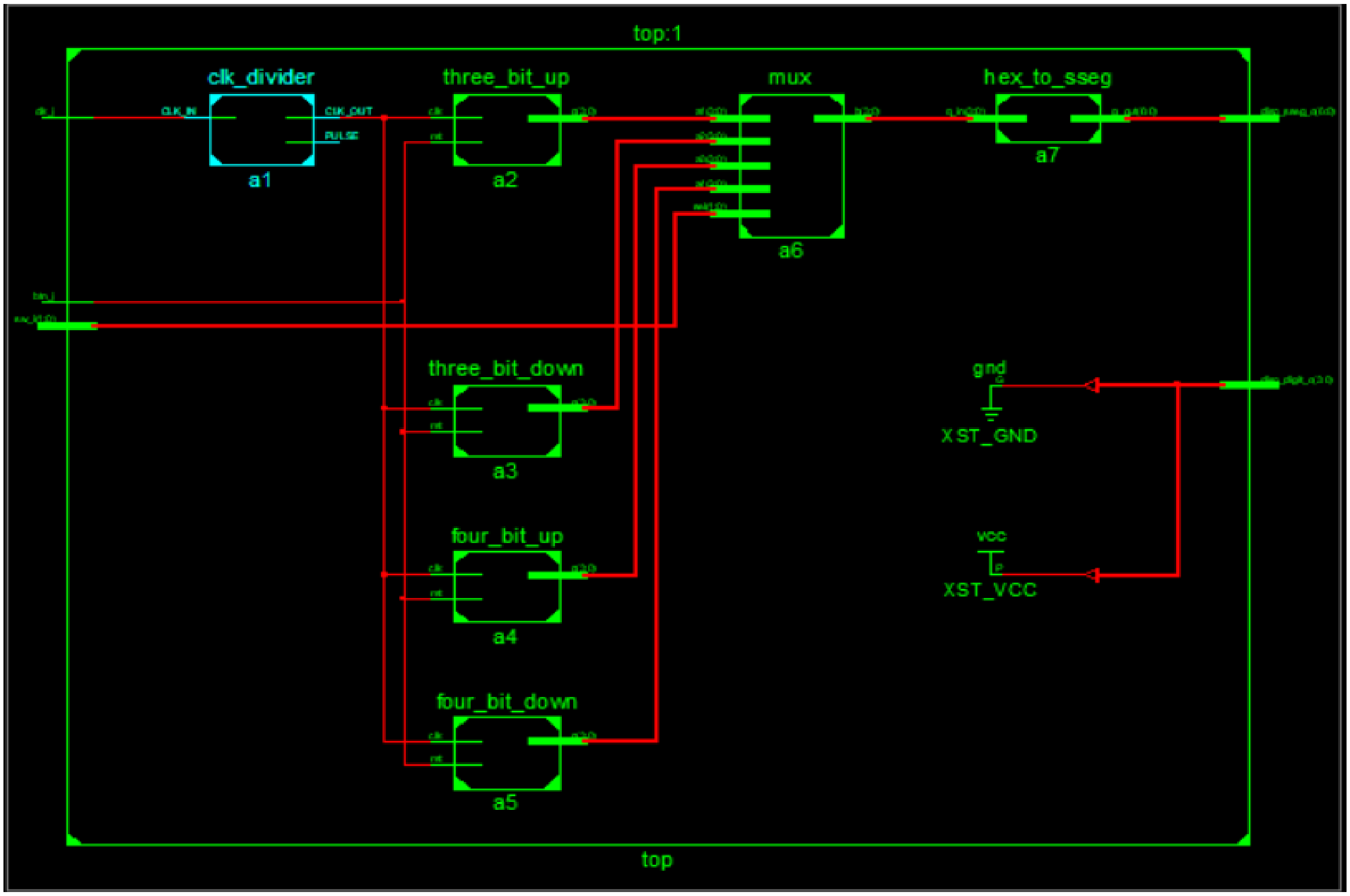
d_0

	q_1q_0 00	01	11	10
q_3q_2 00	0	0	1	1
01	1	1	0	0
11	0	0	1	1
10	1	1	0	0

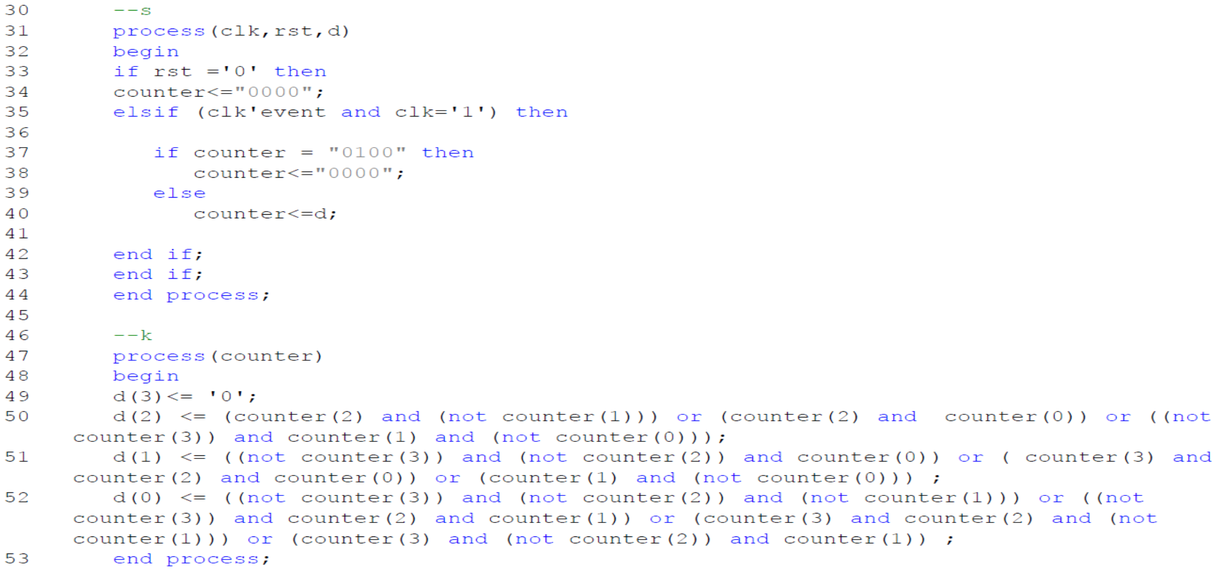
$$d_0 = /q_3/q_2q_1 + /q_3q_2/q_1 + q_3q_2q_1 + q_3/q_2/q_1$$

Schematic diagram



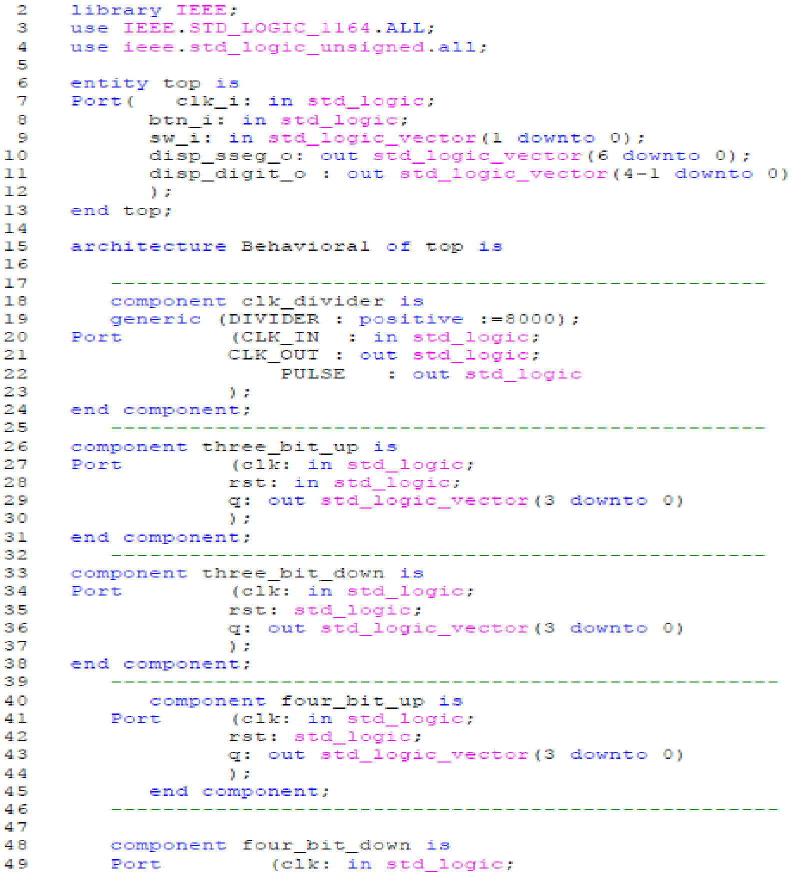

top:1
clk_divider
CLK_IN
CLK_OUT
PULSE
a1
three_bit_up
clk
rst
a2
mux
a6
hex_to_sseg
a7
clk_i
btn_i
three_bit_down
clk
rst
a3
gnd
XST_GND
four_bit_up
clk
rst
a4
vcc
XST_VCC
four_bit_down
clk
rst
a5
top

3 bit up counter

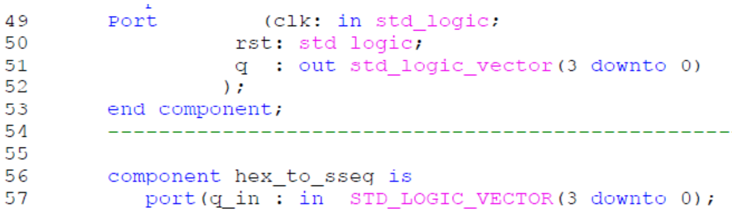


```
--s
process(clk,rst,d)
begin
if rst ='0' then
counter<="0000";
elsif (clk'event and clk='1') then

   if counter = "0100" then
      counter<="0000";
   else
      counter<=d;

end if;
end if;
end process;

--k
process(counter)
begin
d(3)<= '0';
d(2)  <= (counter(2) and (not counter(1))) or (counter(2) and  counter(0)) or ((not counter(3)) and counter(1) and (not counter(0)));
d(1)  <= ((not counter(3)) and (not counter(2)) and counter(0)) or ( counter(3) and counter(2) and counter(0)) or (counter(1) and (not counter(0))) ;
d(0)  <= ((not counter(3)) and (not counter(2)) and (not counter(1))) or ((not counter(3)) and counter(2) and counter(1)) or (counter(3) and counter(2) and (not counter(1))) or (counter(3) and (not counter(2)) and counter(1)) ;
end process;
```

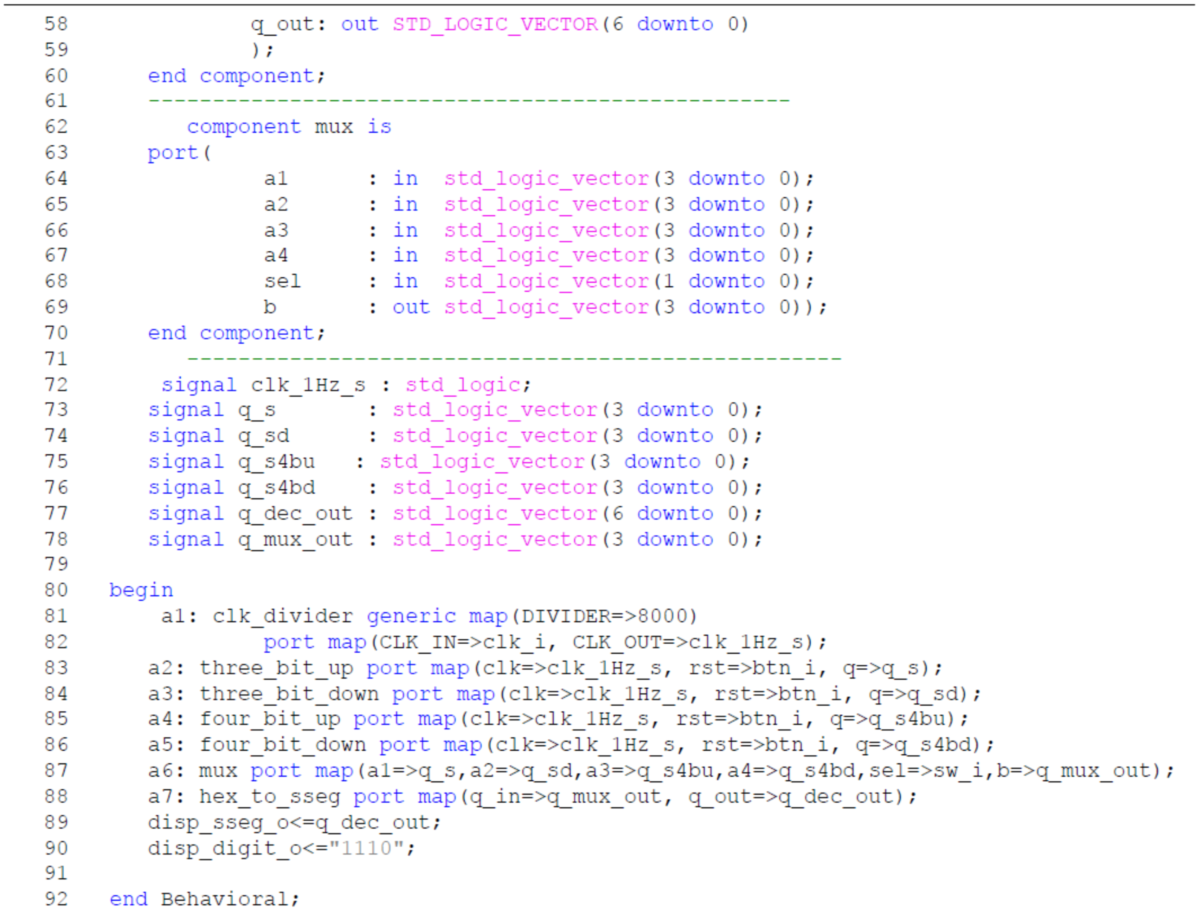
Main file code



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity top is
Port(   clk_i: in std_logic;
      btn_i: in std_logic;
      sw_i: in std_logic_vector(1 downto 0);
      disp_sseg_o: out std_logic_vector(6 downto 0);
      disp_digit_o : out std_logic_vector(4-1 downto 0)
      );
end top;

architecture Behavioral of top is

   --------------------------------------------------
   component clk_divider is
   generic (DIVIDER : positive :=8000);
Port         (CLK_IN  : in std_logic;
            CLK_OUT : out std_logic;
                PULSE   : out std_logic
            );
end component;
   --------------------------------------------------
component three_bit_up is
Port         (clk: in std_logic;
            rst: in std_logic;
            q: out std_logic_vector(3 downto 0)
            );
end component;
   --------------------------------------------------
component three_bit_down is
Port         (clk: in std_logic;
            rst: std_logic;
            q: out std_logic_vector(3 downto 0)
            );
end component;
   ---------------------------------------------------
      component four_bit_up is
   Port      (clk: in std_logic;
            rst: std_logic;
            q: out std_logic_vector(3 downto 0)
            );
      end component;
   ---------------------------------------------------

   component four_bit_down is
   Port         (clk: in std_logic;
```



```
Port        (clk: in std_logic;
          rst: std logic;
          q  : out std_logic_vector(3 downto 0)
         );
end component;
--------------------------------------------------

component hex_to_sseg is
   port(q_in : in  STD_LOGIC_VECTOR(3 downto 0);
```



```
           q_out: out STD_LOGIC_VECTOR(6 downto 0)
           );
   end component;
   --------------------------------------------------
      component mux is
   port(
            a1       : in  std_logic_vector(3 downto 0);
            a2       : in  std_logic_vector(3 downto 0);
            a3       : in  std_logic_vector(3 downto 0);
            a4       : in  std_logic_vector(3 downto 0);
            sel      : in  std_logic_vector(1 downto 0);
            b        : out std_logic_vector(3 downto 0));
   end component;
      --------------------------------------------------
    signal clk_1Hz_s : std_logic;
   signal q_s        : std_logic_vector(3 downto 0);
   signal q_sd       : std_logic_vector(3 downto 0);
   signal q_s4bu    : std_logic_vector(3 downto 0);
   signal q_s4bd     : std_logic_vector(3 downto 0);
   signal q_dec_out : std_logic_vector(6 downto 0);
   signal q_mux_out : std_logic_vector(3 downto 0);

begin
    a1: clk_divider generic map(DIVIDER=>8000)
            port map(CLK_IN=>clk_i, CLK_OUT=>clk_1Hz_s);
   a2: three_bit_up port map(clk=>clk_1Hz_s, rst=>btn_i, q=>q_s);
   a3: three_bit_down port map(clk=>clk_1Hz_s, rst=>btn_i, q=>q_sd);
   a4: four_bit_up port map(clk=>clk_1Hz_s, rst=>btn_i, q=>q_s4bu);
   a5: four_bit_down port map(clk=>clk_1Hz_s, rst=>btn_i, q=>q_s4bd);
   a6: mux port map(a1=>q_s,a2=>q_sd,a3=>q_s4bu,a4=>q_s4bd,sel=>sw_i,b=>q_mux_out);
   a7: hex_to_sseg port map(q_in=>q_mux_out, q_out=>q_dec_out);
   disp_sseg_o<=q_dec_out;
   disp_digit_o<="1110";

end Behavioral;
```