M2 Master E3A Integration Circuits Systems

TD7-8: Design of a frequency divider based on a gray code counter

1. IFTAKHER Mohammed Akib – No. 22211204

2. KIBIWOTT Albert Kiplagat - No. 22211612

Instructor: Professor Hervé MATHIAS

TD7-8: Design of a frequency divider based on a gray code counter

In this practical work, a synchronous counter of a first clock has been designed in order to make a clock divider. Additionally, the output of the counter has been sampled by another asynchronous clock of the first clock.

The counter of interest here is the gray code counter. A 4-bit binary counter is considered. The table below shows the chronogram of the state of its outputs at the time of transition.

Table 1: binary to gray code conversion table with an additional virtual bit.

D	**b_3**	**b_2**	**b_1**	**b_0**	**y_3**	**y_2**	**y_1**	**y_0**	**x**
0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	0	1	0
2	0	0	1	0	0	0	1	1	1
3	0	0	1	1	0	0	1	0	0
4	0	1	0	0	0	1	1	0	1
5	0	1	0	1	0	1	1	1	0
6	0	1	1	0	0	1	0	1	1
7	0	1	1	1	0	1	0	0	0
8	1	0	0	0	1	1	0	0	1
9	1	0	0	1	1	1	0	1	0
10	1	0	1	0	1	1	1	1	1
11	1	0	1	1	1	1	1	0	0
12	1	1	0	0	1	0	1	0	1
13	1	1	0	1	1	0	1	1	0
14	1	1	1	0	1	0	0	1	1
15	1	1	1	1	1	0	0	0	0

Its output is sampled at the time of transition.
Multiple bits are often changed on a single count transition in binary-coded counter sequences. This may (will) result in decoding errors, particularly when counter values are checked for identity.

Gray-coded count sequences always change one bit at a time, and only one bit at a time. Comparing two such counts for identity will never result in a decoding error. When compared to binary counters, gray-coded counters need half the number of transitions. That is an advantage in terms of reduced dynamic power consumption.

A virtual bit, X, is added to the gray code counter to the right of the LSB to simplify its coding mechanism. This bit equals 1 when the counter is at 0 and changes the value at each clock cycle. Table 1 above shows the values taken by the counter over time with the addition of the virtual bit.

5. Make a simple diagram based on logic gates and flip-flops of the 3 LSBs of the counter

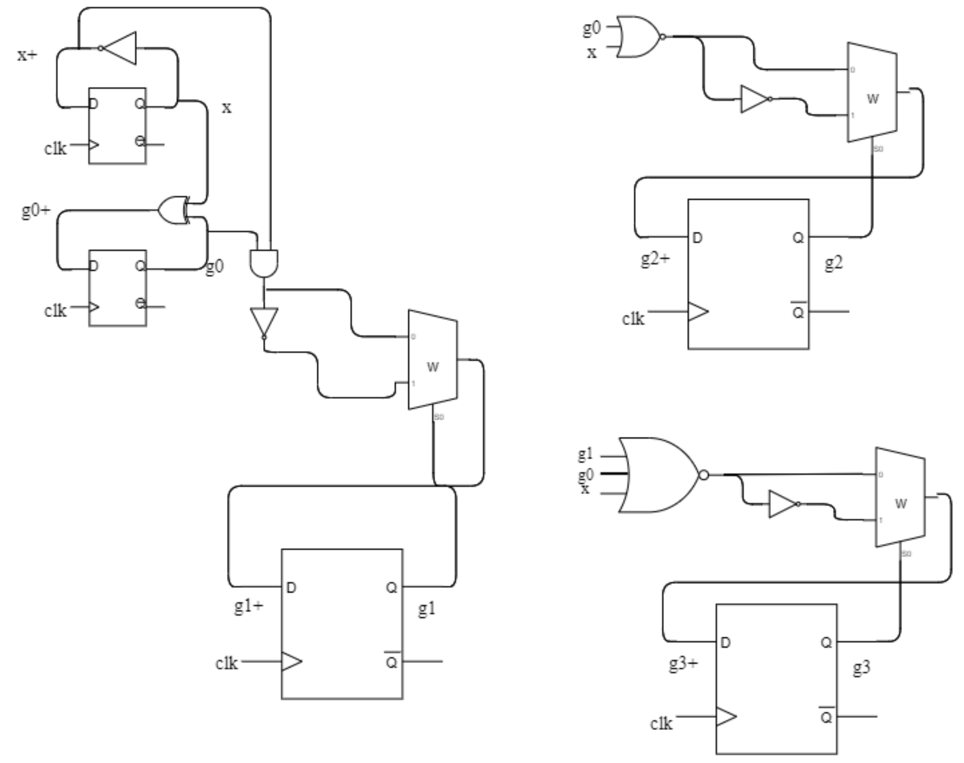


Figure 1: Diagram based on logic gates and flip-flops of the 3 LSBs of the counter.

Entity code

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
use work.myfuncs.all;

entity compteur_gray is
generic ( div2 : integer range 0 to 1 := 1          -- indique si on place d'abord un diviseur par 2
        ) ;
port ( clk,reset,clk2 : in std_logic;     -- horloge d'entre, reset et horloge d'échantillonnage
    enable     : in std_logic;          -- autorisation de fonctionnement
    factor    : in unsigned(6 downto 0);  -- facteur de division
    count_out : out unsigned(6 downto 0); -- compteur de division
    clock_out : out std_logic);         -- horloge divisee generee
end compteur_gray ;

```

6. Architecture a1 code

From the given entity, the counter has been programmed in the architecture a1 below. The enable signal allows the counter to change state when it is equal to 1.

```vhdl
architecture a1 of compteur_gray is
signal count : unsigned (6 downto 0) := (others => '0');

begin

P1 : process (clk, reset)
begin

   if(reset='1' ) then
   count <= (0 => '1', others => '0');

   elsif(clk'event and clk='1') then

      if (enable='1') then
         count(0) <= not count(0);
         count(1) <= count(1) xor count(0);
         for i in 2 to count'left-1 Loop
            if (count(i-1)='1' ) and (count(i-2 downto 0)=to_unsigned(0, i-1) ) then
               count(i) <= not count (i);
            end if;
         end loop;
         if (count(count'left-2 downto 0)= to_unsigned (0, count'left-1) ) then
            count(count'left) <= not count (count'left);
         end if;
      end if ;
end if;
end process;
count_out <= count(4 downto 1) ;
end a1;
```

Test bench Code

```vhdl
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.numeric_std.ALL;


ENTITY test_gray IS
END test_gray;

```

```vhdl
architecture a1 of test_gray is

----------------------------------------------------
-- tous les composants  connecter
----------------------------------------------------

component compteur_gray is
generic ( div2 : integer   -- indique si on place d'abord un diviseur par 2
         ) ;
port ( clk,reset,clk2 : in std_logic;     -- horloge d'entre et reset
    enable     : in std_logic;
    factor    : in unsigned(6 downto 0);  -- facteur de division
    count_out : out unsigned(3 downto 0); -- compteur de division
    clock_out : out std_logic);     -- horloge divise
end component ;

 component digital_gene IS
    PORT(
        -- reset      : OUT std_logic;
        -- enable     : OUT std_logic;
   -- factor     : OUT unsigned(6 downto 0); -- facteur de division
        -- clk,clk2   : OUT std_logic
    -- );
 end component;


---------------------------------
-- tous les signaux locaux
---------------------------------

signal reset : std_logic := '0';  -- reset et autorisation de la logique
signal enable : std_logic := '1';    -- reset et autorisation de la logique
signal clk,clk2 : std_logic := '1';  -- horloge de rfrence  10 Mhz du GSM
signal clock_out : std_logic ;      -- horloge de sortie du DCO
signal factor : unsigned(6 downto 0) ; -- facteur de division
signal count_out : unsigned(3 downto 0) := (others => '0') ;   -- phase attendue
begin

-- instantiation de tous les composants
----------------------------------------

-- c1 : digital_gene  port map(reset,enable,factor,clk,clk2) ;
c2 : compteur_gray generic map(div2 => 1)
   port map(clk,reset,clk2,enable,factor,count_out,clock_out) ;

reset <= '0';
enable <= '1';

clk_process: process --to form the clock
begin
      clk <= '0';
      wait for 50 ns;

      clk <= '1';
      wait for 50 ns;

end process;



end architecture a1 ;
```

Simulation

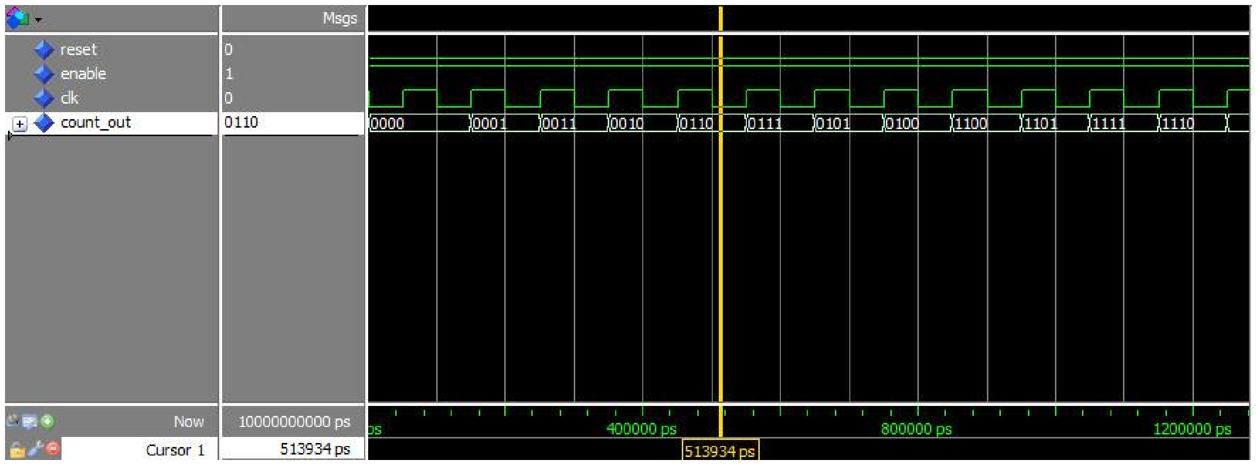


Figure 2: Simulation of the gray code counter.

7. Architecture a2 code

Here, an independent and asynchronous clock clk2 has been created to be used in sampling the output of the counter. At the rising edge of this clock, the count_out output must contain the value taken by the counter, recorded in binary. Using the myfuncs function, the library provided, the previous code has been modified to a new a2 architecture in order to carry out this sampling.

```
architecture a2 of compteur_gray is
signal count : unsigned (7 downto 0) := (others => '0');
signal countc : unsigned(6 downto 0) := (others => '0');

begin

P1 : process (clk, reset)
begin

   if(reset='0' ) then
   count <= (0 => '1', others => '0');

   elsif(clk'event and clk='1') then

      if (enable='1') then
         count(0) <= not count(0);
         count(1) <= count(1) xor count(0);
         for i in 2 to count'left-1 Loop
            if (count(i-1)='1' ) and (count(i-2 downto 0)=to_unsigned(0, i-1) ) then
               count(i) <= not count (i);
            end if;
         end loop;
         if (count(count'left-2 downto 0)= to_unsigned (0, count'left-1) ) then
            count(count'left) <= not count (count'left);
         end if;
      end if ;
end if;
end process;

P2 : process(clk2)
begin
  if(clk2'event and clk2='1') then
   countc <= count(count'left downto 1) ;
  end if ;
end process ;

P3 : count_out <= gray2bin(countc) ;

end a2;
```

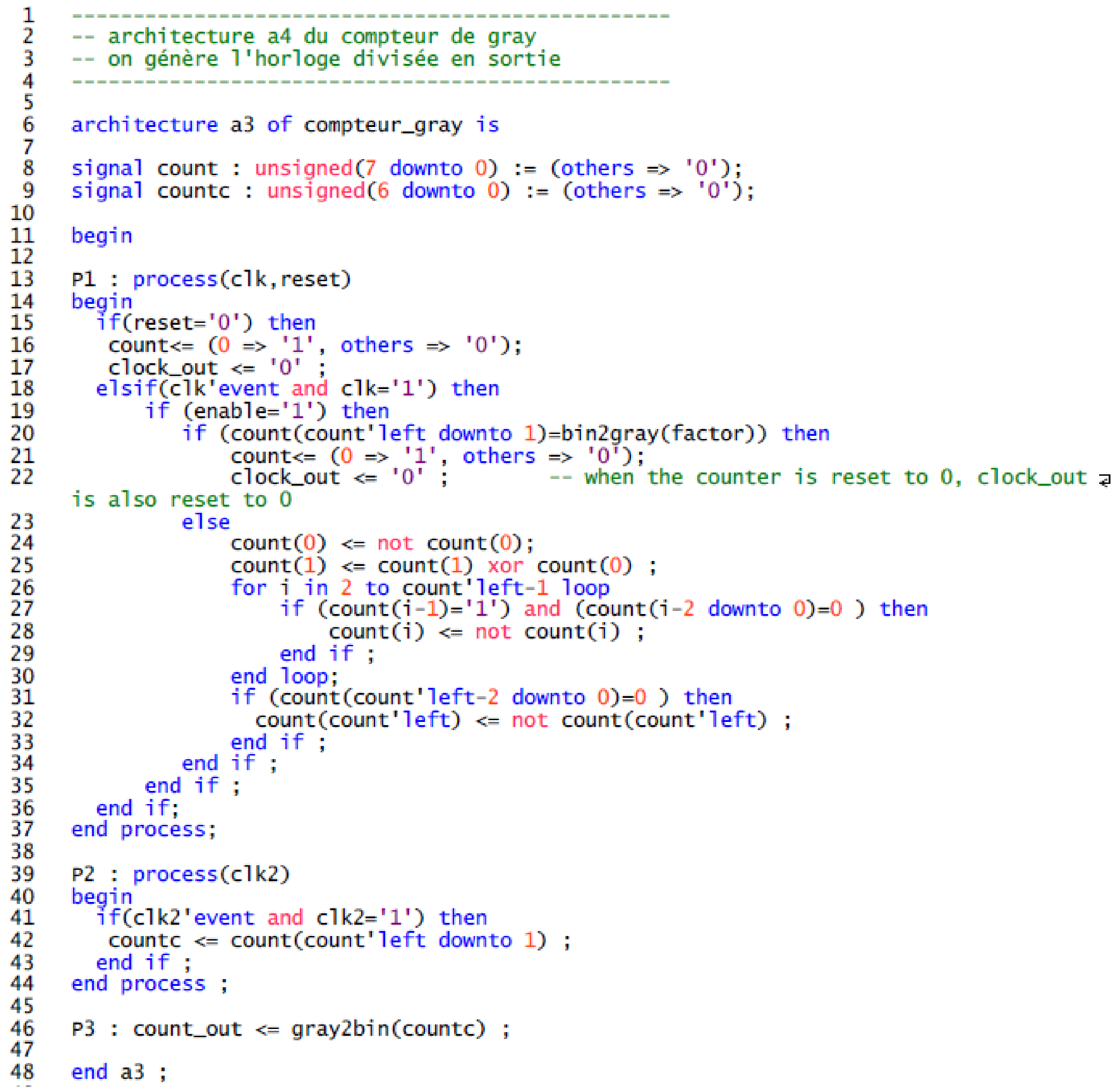
My funcs function code

```vhdl
-- Created by @(#)$CDS: vhdlin version 6.1.8-64b 07/16/2019 20:11 (cpgbld02.cadence.com) $
-- on Wed Apr  1 13:51:39 2020


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;

package myfuncs is
  function bin2gray (bin : unsigned)
                   return unsigned;
  function gray2bin (gray : unsigned)
                   return unsigned;
end myfuncs;

```

```vhdl
-- Created by @(#)$CDS: vhdlin version 6.1.8-64b 07/16/2019 20:11 (cpgbld02.cadence.com) $
-- on Wed Apr  1 13:51:39 2020


package body myfuncs is
  function bin2gray (bin : unsigned)
                   return unsigned is
  variable gray : unsigned(bin'range) ;
  begin
   gray(bin'left) := bin(bin'left) ;
   for J in 0 to bin'length-2  loop
     gray(J):=bin(J+1) xor bin(J) ;
   end loop ;
   return gray ;
  end bin2gray;

  function gray2bin (gray : unsigned)
                   return unsigned is
  variable bin : unsigned(gray'range) ;
  begin
   bin(gray'left) := gray(gray'left) ;
   for J in 0 to gray'left-1 loop
       bin(J) := gray(gray'left) ;
       for k in gray'left-1 downto J loop
         bin(J):=bin(J) xor gray(K) ;
       end loop ;
   end loop ;
   return bin ;
  end gray2bin;


end myfuncs;

```

8. Architecture a3 code

In this part, a gray code counter with a programmable modulo has been created. The value of the modulo is an entry of this counter called 'factor’ in the entity. In practice, the input factor is equal to the value of the modulo minus 1 because the counter will count from 0 to 'factor’. Using myfuncs function library provided, a new a3 architecture has been created in order to set the modulo of the counter.



```vhdl
-------------------------------------------------
-- architecture a4 du compteur de gray
-- on génère l'horloge divisée en sortie
-------------------------------------------------

architecture a3 of compteur_gray is

signal count : unsigned(7 downto 0) := (others => '0');
signal countc : unsigned(6 downto 0) := (others => '0');

begin

P1 : process(clk,reset)
begin
  if(reset='0') then
   count<= (0 => '1', others => '0');
   clock_out <= '0' ;
  elsif(clk'event and clk='1') then
      if (enable='1') then
         if (count(count'left downto 1)=bin2gray(factor)) then
             count<= (0 => '1', others => '0');
             clock_out <= '0' ;          -- when the counter is reset to 0, clock_out is also reset to 0
         else
             count(0) <= not count(0);
             count(1) <= count(1) xor count(0) ;
             for i in 2 to count'left-1 loop
                 if (count(i-1)='1') and (count(i-2 downto 0)=0 ) then
                     count(i) <= not count(i) ;
                 end if ;
             end loop;
             if (count(count'left-2 downto 0)=0 ) then
               count(count'left) <= not count(count'left) ;
             end if ;
         end if ;
      end if ;
  end if;
end process;

P2 : process(clk2)
begin
  if(clk2'event and clk2='1') then
   countc <= count(count'left downto 1) ;
  end if ;
end process ;

P3 : count_out <= gray2bin(countc) ;

end a3 ;
```

9. Architecture a4 code

Here, a clock signal whose frequency is the frequency of the initial clock divided by factor +1 has been generated as depicted in architecture a4 below:

```
-------------------------------------------------
-- architecture a4 du compteur de gray
-- on génère l'horloge divisée en sortie
-------------------------------------------------

architecture a4 of compteur_gray is

signal count : unsigned(7 downto 0) := (others => '0');
signal countc : unsigned(6 downto 0) := (others => '0');

begin

P1 : process(clk,reset)
begin
  if(reset='0') then
   count<= (0 => '1', others => '0');
   clock_out <= '0' ;
  elsif(clk'event and clk='1') then
      if (enable='1') then
   if (count(count'left downto 1)=bin2gray(factor)) then
       count<= (0 => '1', others => '0');
       clock_out <= '0' ;        -- quand le compteur est remis à 0, clock_out est aussi remis à 0
   else
       count(0) <= not count(0);
       count(1) <= count(1) xor count(0) ;
       for i in 2 to count'left-1 loop
           if (count(i-1)='1') and (count(i-2 downto 0)=0 ) then
          count(i) <= not count(i) ;
           end if ;
       end loop;
       if (count(count'left-2 downto 0)=0 ) then
         count(count'left) <= not count(count'left) ;
       end if ;
   end if ;
   if (count(count'left downto 1)=bin2gray(factor/2)) then  -- quand on atteint la moitié du modulo, clock_out est mis à 1
     clock_out <= '1';
   end if ;
      end if ;
  end if;
end process;

P2 : process(clk2)
begin
  if(clk2'event and clk2='1') then
   countc <= count(count'left downto 1) ;
  end if ;
end process ;

P3 : count_out <= gray2bin(countc) ;

end a4 ;
```

Conclusion

In this practical work, the operation and programming of gray code counter have been demonstrated. From the practical, a frequency divider has been designed by generating a clock signal that has half the frequency of the initial clock. The objective of this practical session is to exhibit the efficiency of the gray code when it comes to bit transition and power consumption.