First Assignment

By

Calvin Boostman
Gabriel Cobos Tello
Rick Kloosterman
VHDL
G. Nanninga

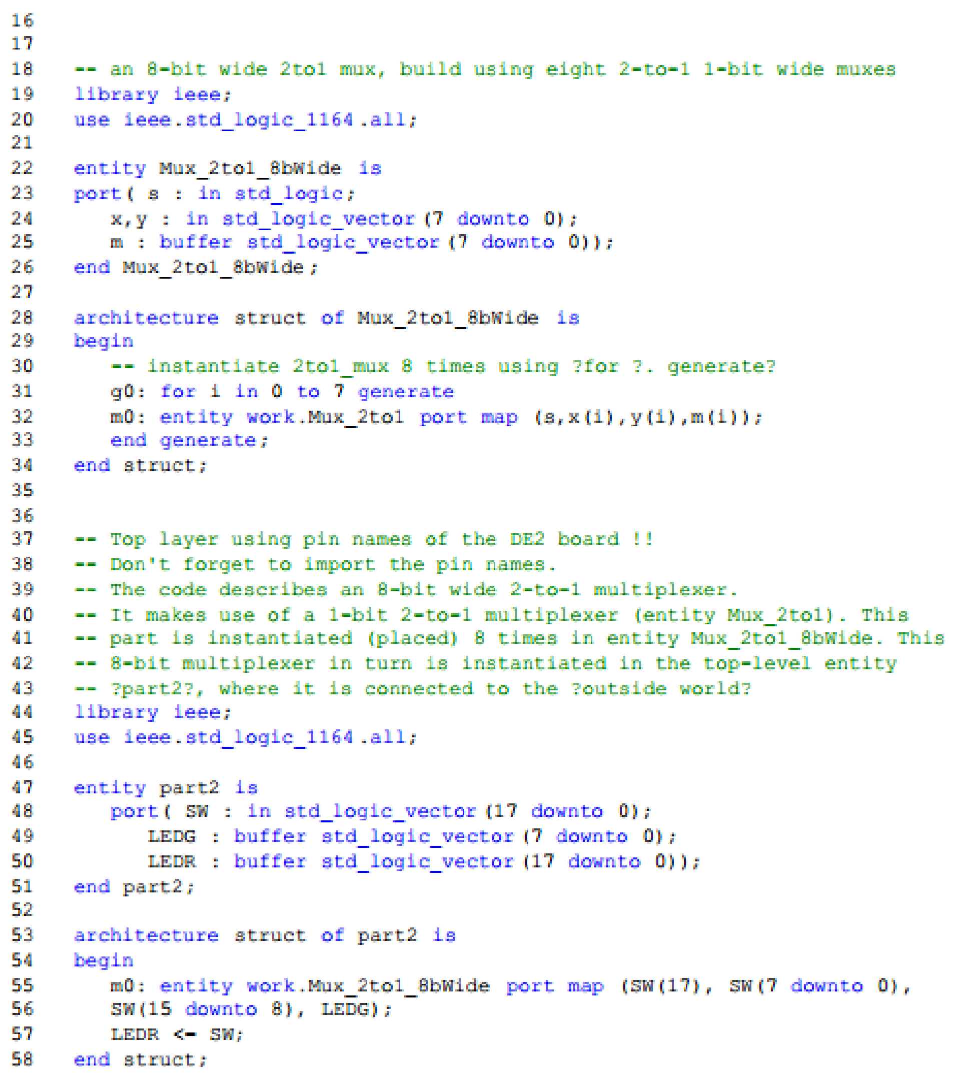
INTRODUTION

The idea behind this first assignment is for us to get a good look at the basics of VHDL, managing to become comfortable with its structure and methodology so that we can move on to more complex tasks and finally achieve a moderate mastery for this programing language.

PART 2

9.

```
library ieee;
use ieee.std_logic_1164.all;

entity Mux_2to1 is
   port( s,x,y : in std_logic;
      m : buffer std_logic);
end Mux_2to1;

architecture behavior of Mux_2to1 is
begin
-- Code not shown, enter here your own code !!


   m<= x when s='0' else y;
end behavior;
```



```
-- an 8-bit wide 2to1 mux, build using eight 2-to-1 1-bit wide muxes
library ieee;
use ieee.std_logic_1164.all;

entity Mux_2to1_8bWide is
port( s : in std_logic;
   x,y : in std_logic_vector(7 downto 0);
   m : buffer std_logic_vector(7 downto 0));
end Mux_2to1_8bWide;

architecture struct of Mux_2to1_8bWide is
begin
   -- instantiate 2to1_mux 8 times using ?for ?. generate?
   g0: for i in 0 to 7 generate
   m0: entity work.Mux_2to1 port map (s,x(i),y(i),m(i));
   end generate;
end struct;


-- Top layer using pin names of the DE2 board !!
-- Don't forget to import the pin names.
-- The code describes an 8-bit wide 2-to-1 multiplexer.
-- It makes use of a 1-bit 2-to-1 multiplexer (entity Mux_2to1). This
-- part is instantiated (placed) 8 times in entity Mux_2to1_8bWide. This
-- 8-bit multiplexer in turn is instantiated in the top-level entity
-- ?part2?, where it is connected to the ?outside world?
library ieee;
use ieee.std_logic_1164.all;

entity part2 is
   port( SW : in std_logic_vector(17 downto 0);
      LEDG : buffer std_logic_vector(7 downto 0);
      LEDR : buffer std_logic_vector(17 downto 0));
end part2;

architecture struct of part2 is
begin
   m0: entity work.Mux_2to1_8bWide port map (SW(17), SW(7 downto 0),
   SW(15 downto 8), LEDG);
   LEDR <= SW;
end struct;
```

13.

a) The pin assignments. When using the signal names in the file they are connected to the right pin numbers of the FPGA.

b) A sum of Products is the standard expression that defines the addition of binary numbers, were 1 is always dominant over 0**.** ***F = A'B + AB' + AB***, this would express the addition of 2 binary digits but can be extrapolated to any number of them, a truth table can be used for this.

c) Instantiation is the term that defines the introduction of entities as components in a VHDL design.

```
instance_label: component_name
   generic map (generic_association_list)
   port map (port_association_list);
```

```
u1: component port map (X,Y,S,C);
```

d) G0 : for i in 0 to 7 generate

```
    M0 :  Component port map (X,Y(i),S(i),C(i));
End generate;
```

e) A top-level entity is the linking file that unites the functions of all components and makes them relate to the world.

f) In the work library.

PART 3

TRUTH TABLE

C3	C2	C1	C0	H0	H1	H2	H3	H4	H5	H6
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	0	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

QUESTIONS

7.

a) 30,216
b) 4 are used as output
c) 4 are used as input
d) Check for whenever the output is supposed to be high, the input. Check from that input which gates are ‘1’. Those gates that are ‘1’are the input for an AND gate. The output of that AND gate is the input of an OR gate.

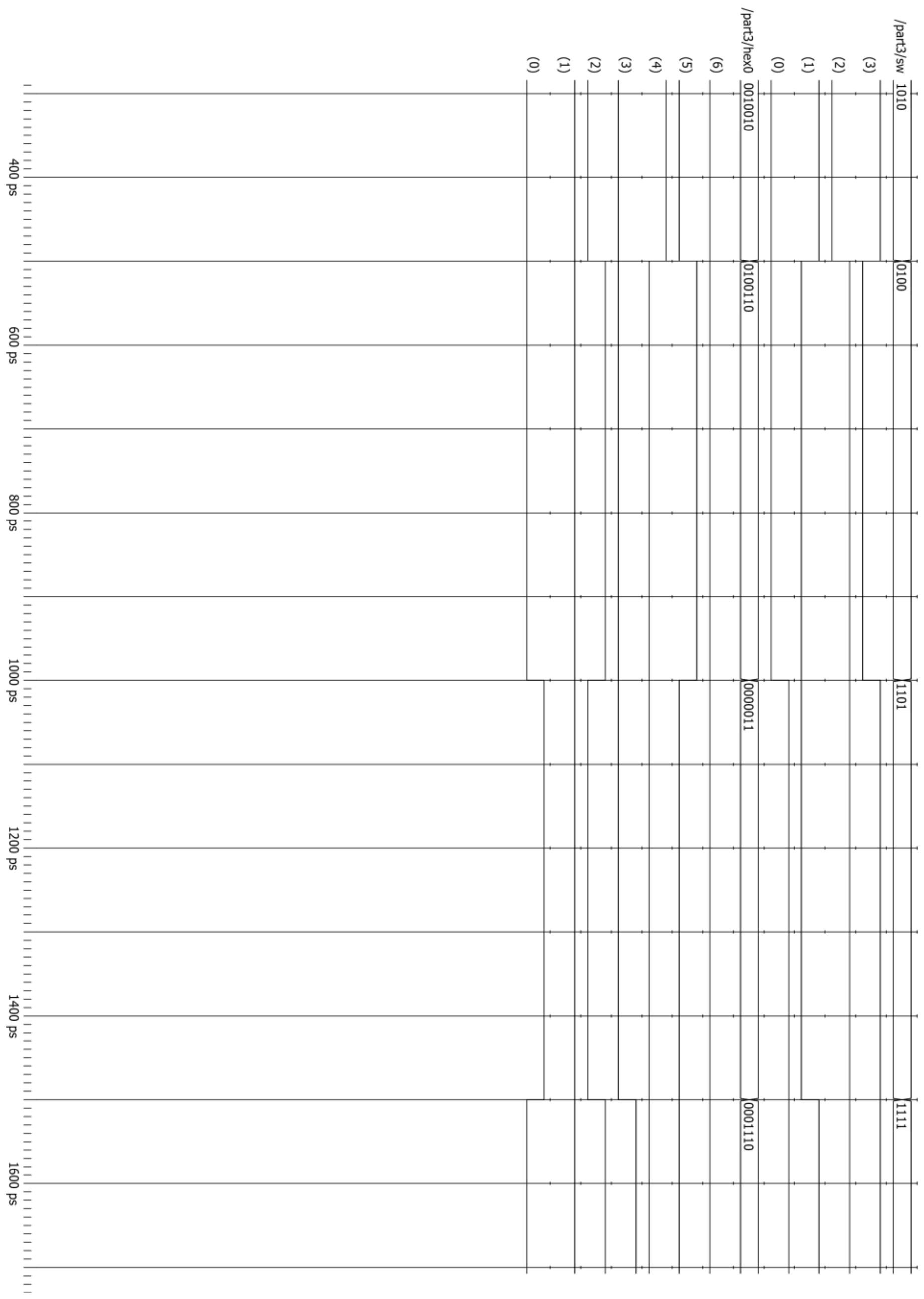
e) 7, because there are always 7 for the 7 segment display

9.

Number of LEs: 7 for both, because you only need 7. It will always be optimized to the least LEs necessary.

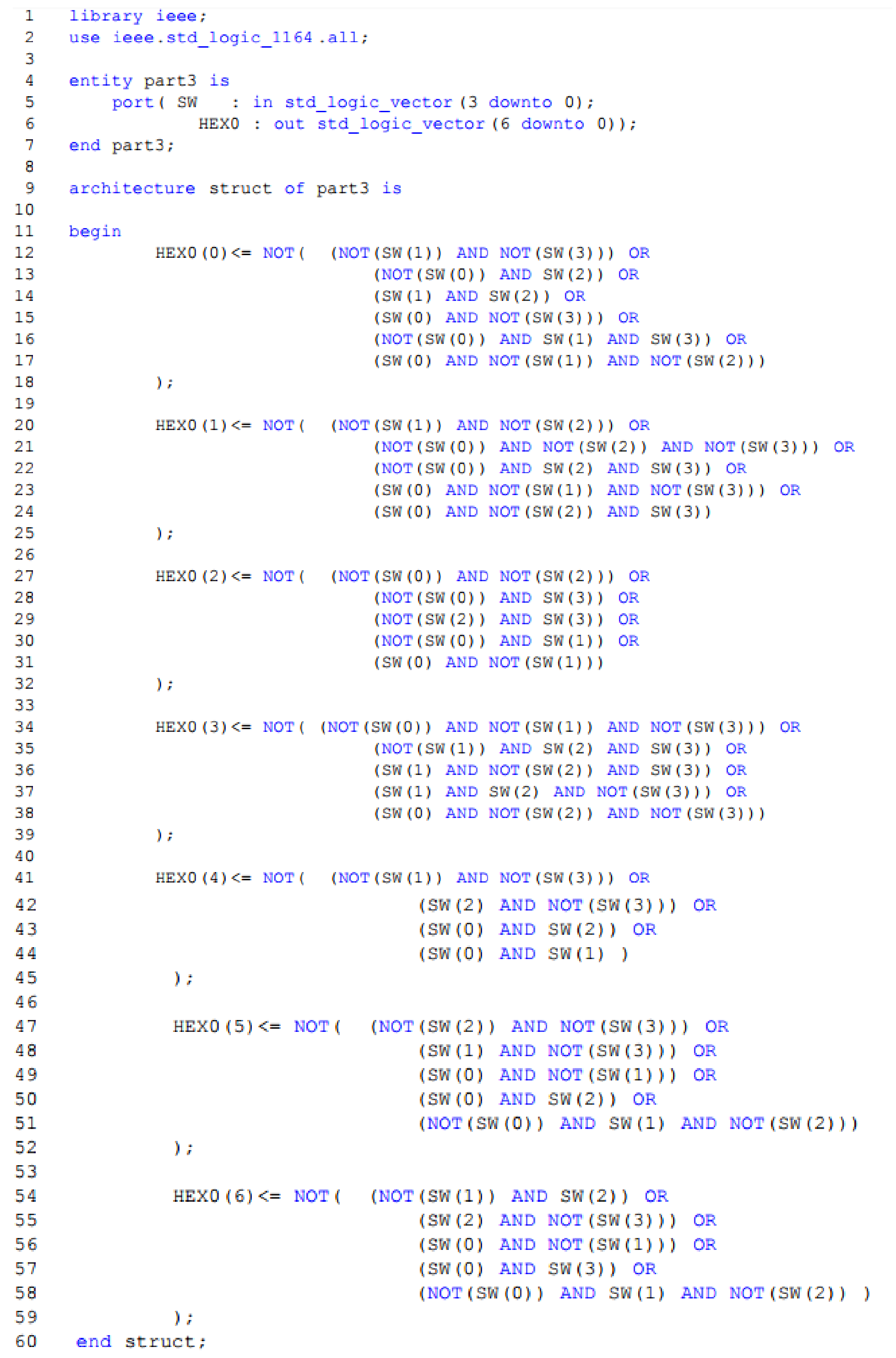
TCL:

```
force -freeze sim:/part3/sw
        1010 0 run

force -freeze sim:/part3/sw
        0100 0 run

force -freeze sim:/part3/sw
        1101 0 run

force -freeze sim:/part3/sw
        1111 0 run
```



/part3/sw
1010
0100
1101
1111
(3)
(2)
(1)
(0)
/part3/hex0
0010010
0100110
0000011
0001110
(6)
(5)
(4)
(3)
(2)
(1)
(0)
400 ps
600 ps
800 ps
1000 ps
1200 ps
1400 ps
1600 ps

CODE



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity part3 is
    port( SW   : in std_logic_vector(3 downto 0);
            HEX0 : out std_logic_vector(6 downto 0));
end part3;

architecture struct of part3 is

begin
        HEX0(0)<= NOT(  (NOT(SW(1)) AND NOT(SW(3))) OR
                        (NOT(SW(0)) AND SW(2)) OR
                        (SW(1) AND SW(2)) OR
                        (SW(0) AND NOT(SW(3))) OR
                        (NOT(SW(0)) AND SW(1) AND SW(3)) OR
                        (SW(0) AND NOT(SW(1)) AND NOT(SW(2)))
        );

        HEX0(1)<= NOT(  (NOT(SW(1)) AND NOT(SW(2))) OR
                        (NOT(SW(0)) AND NOT(SW(2)) AND NOT(SW(3))) OR
                        (NOT(SW(0)) AND SW(2) AND SW(3)) OR
                        (SW(0) AND NOT(SW(1)) AND NOT(SW(3))) OR
                        (SW(0) AND NOT(SW(2)) AND SW(3))
        );

        HEX0(2)<= NOT(  (NOT(SW(0)) AND NOT(SW(2))) OR
                        (NOT(SW(0)) AND SW(3)) OR
                        (NOT(SW(2)) AND SW(3)) OR
                        (NOT(SW(0)) AND SW(1)) OR
                        (SW(0) AND NOT(SW(1)))
        );

        HEX0(3)<= NOT( (NOT(SW(0)) AND NOT(SW(1)) AND NOT(SW(3))) OR
                        (NOT(SW(1)) AND SW(2) AND SW(3)) OR
                        (SW(1) AND NOT(SW(2)) AND SW(3)) OR
                        (SW(1) AND SW(2) AND NOT(SW(3))) OR
                        (SW(0) AND NOT(SW(2)) AND NOT(SW(3)))
        );

        HEX0(4)<= NOT(  (NOT(SW(1)) AND NOT(SW(3))) OR
                            (SW(2) AND NOT(SW(3))) OR
                            (SW(0) AND SW(2)) OR
                            (SW(0) AND SW(1) )
         );

         HEX0(5)<= NOT(  (NOT(SW(2)) AND NOT(SW(3))) OR
                            (SW(1) AND NOT(SW(3))) OR
                            (SW(0) AND NOT(SW(1))) OR
                            (SW(0) AND SW(2)) OR
                            (NOT(SW(0)) AND SW(1) AND NOT(SW(2)))
         );

         HEX0(6)<= NOT(  (NOT(SW(1)) AND SW(2)) OR
                            (SW(2) AND NOT(SW(3))) OR
                            (SW(0) AND NOT(SW(1))) OR
                            (SW(0) AND SW(3)) OR
                            (NOT(SW(0)) AND SW(1) AND NOT(SW(2)) )
         );
 end struct;
```

```vhdl
library ieee;
```

PART 4

TRUTH TABLE

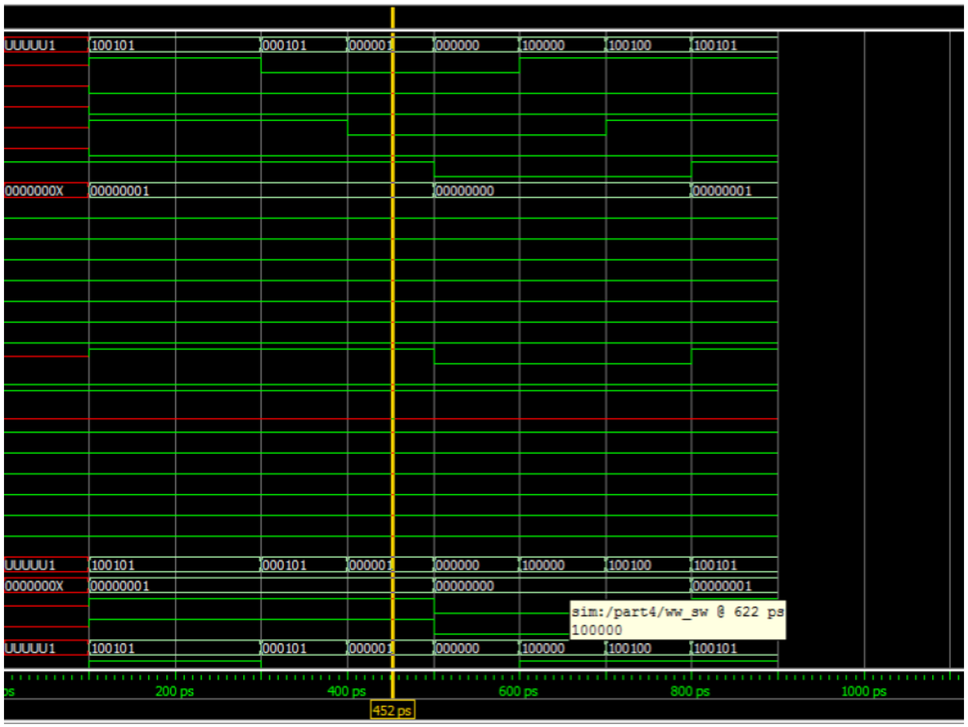
A2	A1	A0	B2	B1	B0	Output
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	1
0	1	1	0	0	0	1
1	0	0	0	0	0	1
1	0	1	0	0	0	1
1	1	0	0	0	0	1
1	1	1	0	0	0	1
0	0	0	0	0	1	0
0	0	1	0	0	1	0
0	1	0	0	0	1	1
0	1	1	0	0	1	1
1	0	0	0	0	1	1
1	0	1	0	0	1	1
1	1	0	0	0	1	1
1	1	1	0	0	1	1
0	0	0	0	1	0	0
0	0	1	0	1	0	0
0	1	0	0	1	0	0
0	1	1	0	1	0	1
1	0	0	0	1	0	1
1	0	1	0	1	0	1
1	1	0	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	0
0	0	1	0	1	1	0
0	1	0	0	1	1	0
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	0	1	1	1
1	1	1	0	1	1	1
0	0	0	1	0	0	0

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

CODE USING BOOLEAN EXPRESSIONS

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


                    entity part4 is
                    Port (
                    SW : in std_logic_vector(5 downto 0);
                    LEDG: out std_logic_vector(7 downto 0));
                    end part4;

                                        architecture behavioral of part4 is
                                        begin
                                        LEDG(0)<=(  (SW(2) and not(SW(3)) and
                                        not(SW(4)) and not(SW(5))) or
                                        (SW(0) and SW(2) and not(SW(4)) and
                                        not(SW(5))) or
                                        (SW(1) and SW(2) and not(SW(3)) and
                                        not(SW(5))) or
                                        (SW(0) and SW(1) and SW(2) and not(SW(5))) or
                                        (SW(1) and not(SW(3)) and not(SW(4))) or
                                        (SW(0) and SW(1) and not(SW(4))) or
                                        (SW(0) and not(SW(3))));
                                        end behavioral;
```

SIMULATION



TCL

```
force -freeze sim:/part4/sw(0) 1 0
run
force -freeze sim:/part4/sw(1) 0 0
force -freeze sim:/part4/sw(2) 1 0
noforce sim:/part4/sw(3)
force -freeze sim:/part4/sw(3) 0 0
force -freeze sim:/part4/sw(4) 0 0
force -freeze sim:/part4/sw(5) 1 0
run
add wave \
{sim:/part4/sw(5) }
add wave \
{sim:/part4/sw }
```

```
add wave \
{sim:/part4/ledg(0) }
run
force -freeze sim:/part4/sw(5) 0 0
run
force -freeze sim:/part4/sw(2) 0 0
force -freeze sim:/part4/sw(2) 0 0
force -freeze sim:/part4/sw(2) 0 0
run
force -freeze sim:/part4/sw(0) 0 0
run
force -freeze sim:/part4/sw(5) 1 0
run
force -freeze sim:/part4/sw(2) 1 0
run
force -freeze sim:/part4/sw(0) 1 0
run
```

LES USED

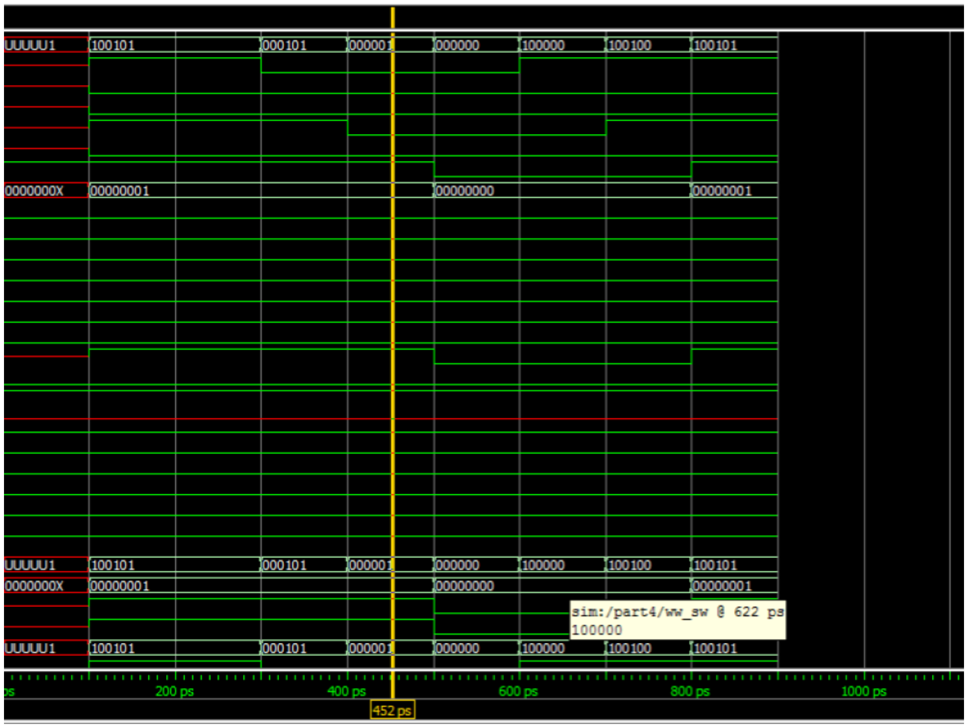
Logic elements used 2

We use a Greater Than operator that consumes 2, 3-bit inputs. The output is one bit.

CODE USING BEHAVIOURAL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

            entity part4 is
            Port (
            SW : in std_logic_vector(5 downto 0);
            LEDG: out std_logic_vector(7 downto 0));
            end part4;

    architecture struct of part4 is
    begin
    GT: entity work.comparator port map (SW(2 downto 0 ), SW(5 downto 3), LEDG(0));
    end struct;

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

    entity comparator is
    port (
    A, B : in std_logic_vector(2 downto 0);
    Y : out std_logic);
    end comparator;

architecture behavioral of comparator is
begin
     proc: process(A, B) is begin
        if A > B then
        Y <= '1'; else
        Y <= '0'; end if;
    end process proc;
end behavioral;
```

SIMULATION



LES USED

Logic elements used 2

QUESTIONS

- a We would get negative numbers. The code have a different range but will still work.
- b Yes, working it out on the board
- c Only if we can program it in behavioral programming.