ECEN302 : Embedded Systems
Lab 2 Submission

Daniel Eisen : 300447549

August 14, 2020

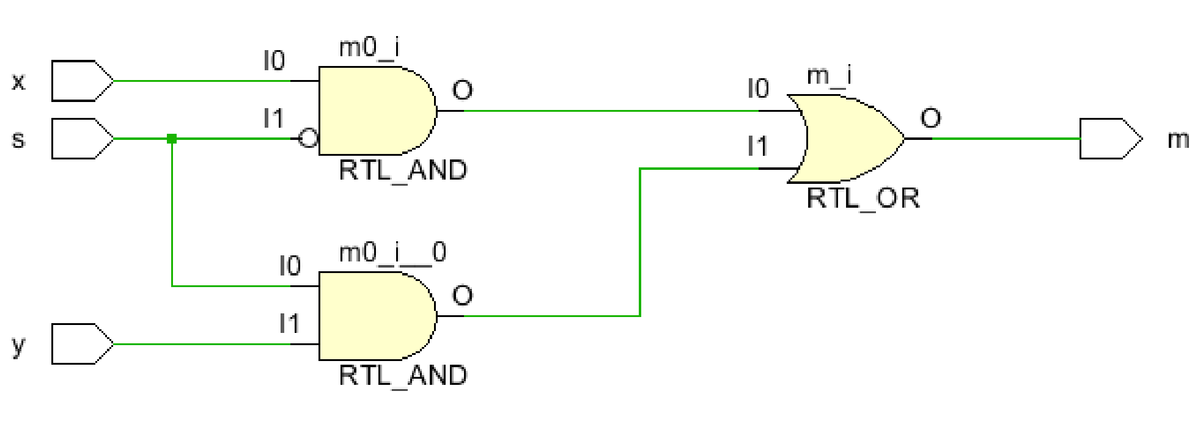
1 Objectives

When designing in VHDL (or Verilog etc) the are various methods/approaches in obtaining the same results. In this lab we explore multiple ways of structuring a VHDL module, using the example of a MUX in multiple configurations. This gave a good idea of how to use/implement standard module structures and the pros/cons of their use cases.

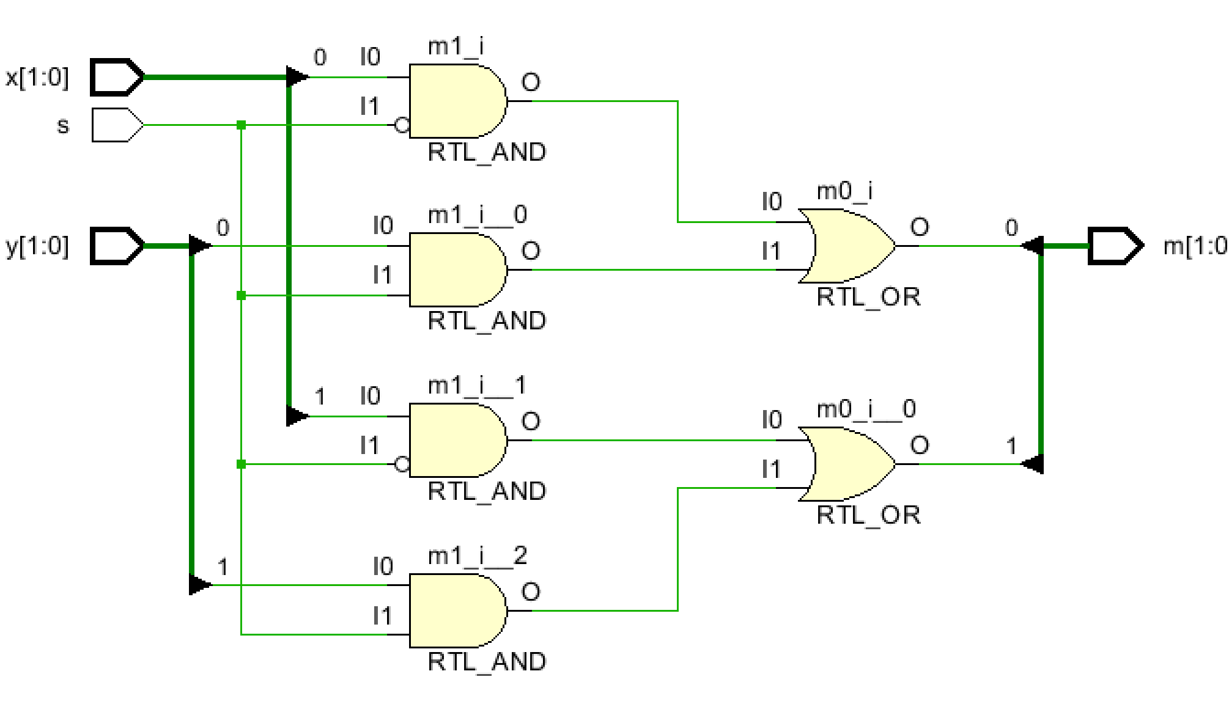
2 Methodology

2.1 Dataflow Modelling

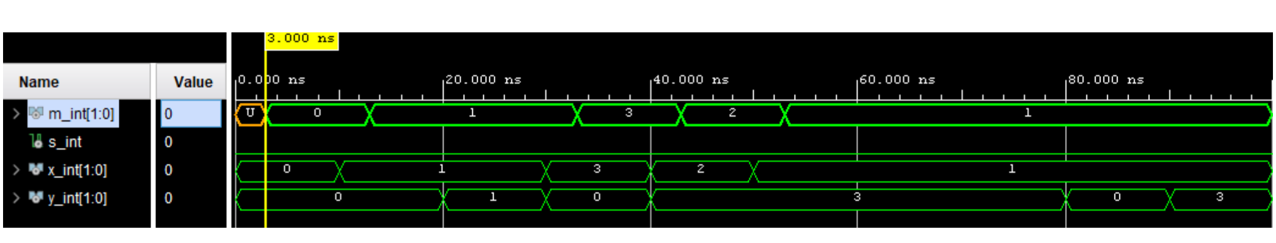
Firstly we used dataflow modelling to implement a 1bit and a 2bit 2-to-1 multiplexer. This method is based mainly around concurrent assignment to either the ports directly, or an intermediate signal. I found that this method of structuring a module is particularly good operations are around getting an input signal(s) and a known output is wanted, ie the data is known and the "component" makeup doesn't matter. Ie this is good for a simple module that isn't combining multiple pre-existing modules together.



Implementing a 1-bit wide 2 to 1 mux with dataflow is quite simple, with the output port being determined with a single line of logic.



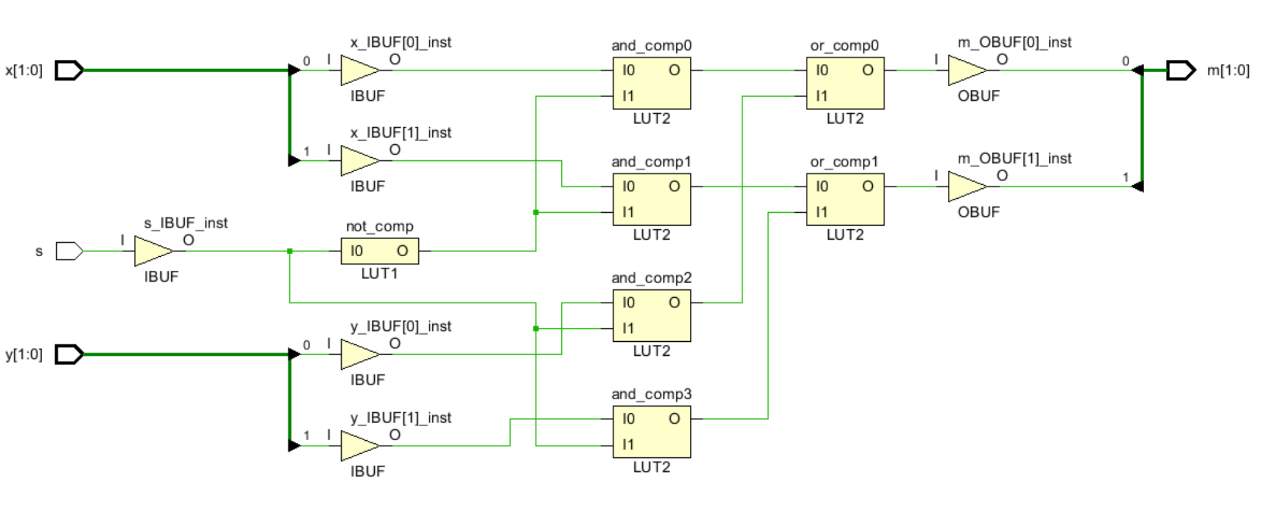
Extending this to a bus input only required a second line for the second bit.



When the source value changes it is evaluated, but the result can be delayed from being assigned to its destination. This is seen in the simulation above (3ns).

2.2 Structural Modelling

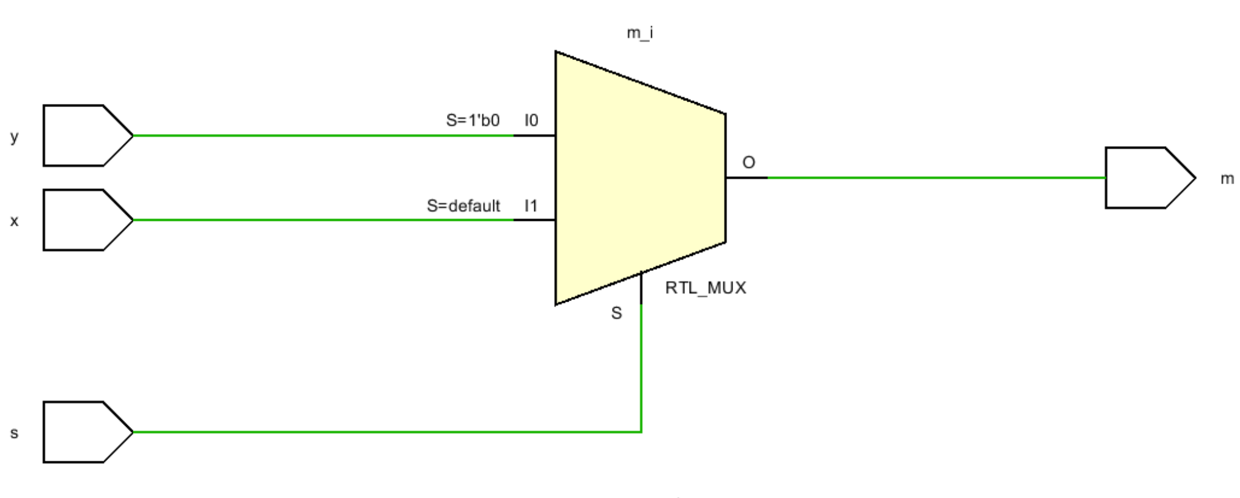
When the design more fits connecting multiple pre-existing components to defined the overall functionality of the circuit, structural modelling can be used to instantiate these components (whether they are other modules or just gate primitives).
In this section the previous design of the 2bit 2-to-1 with structural modelling. Building it out of individual logic gate primitives, this involves instantiating each element type (and, not, or), port mapping them, and creating each part and connecting them via internal signals. For this particular use case this is far too tedious, and probably the wrong application of the structural modelling approach, as for such a simple more data oriented design the code becomes non-representatively long and complex/unreadable.

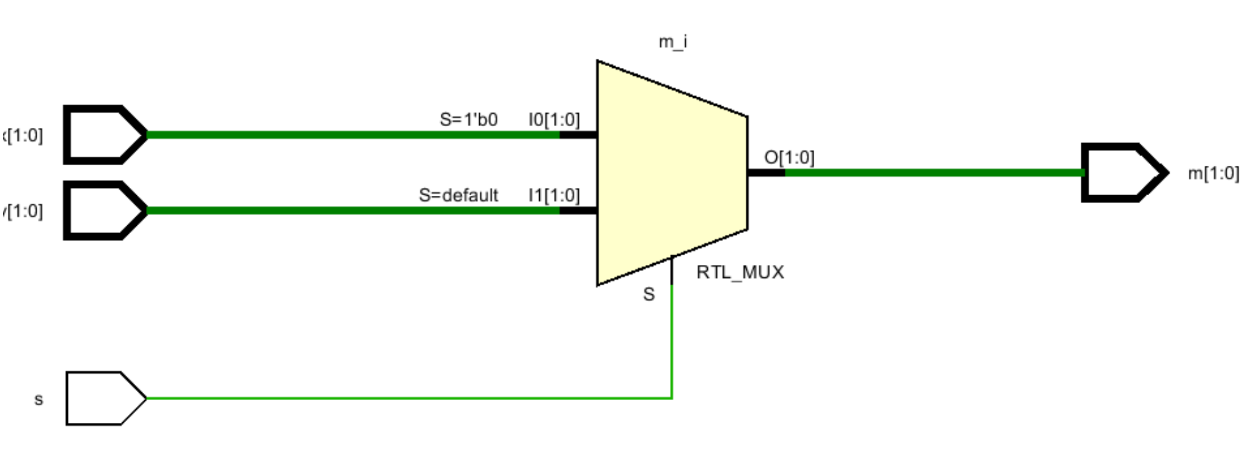


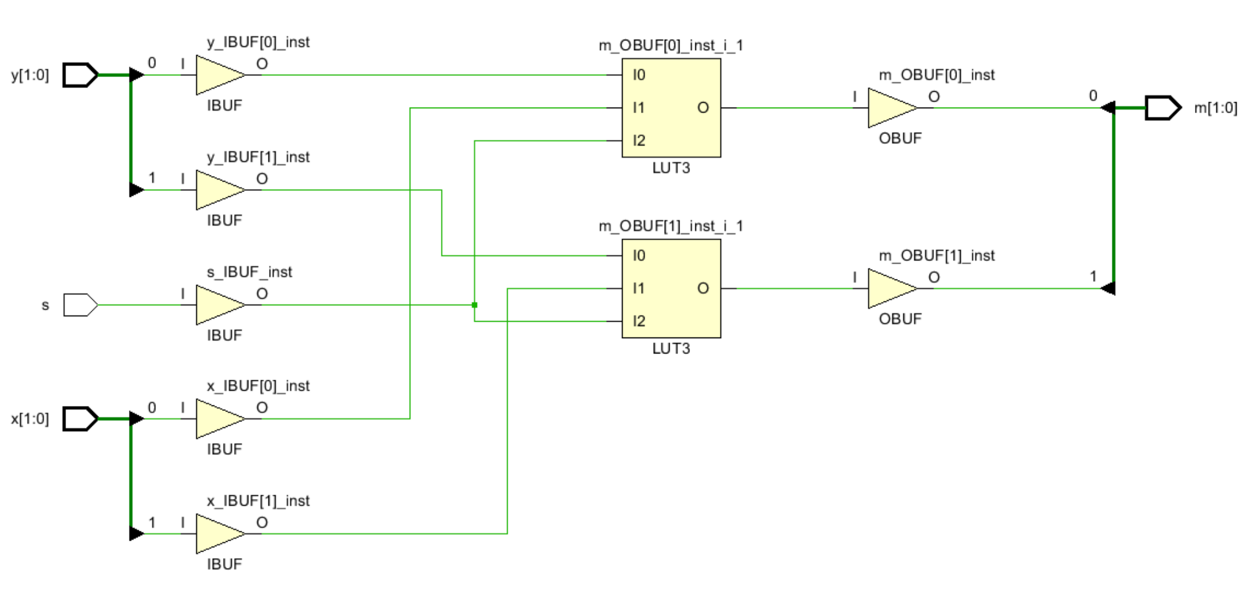
Above is the result of the structural modelling approach.

2.3 Behavioural Modelling

In the dataflow example multiple port assignment is down concurrently, in the Behavioural model a process is defined for a specific function and the statements are then executed sequentially. When doing this with the muxs it is displayed as a mux element before synthesis, but after synthesis can either be implemented in a concurrent or sequential circuit. If this allows for the use of `if,else,then` etc.

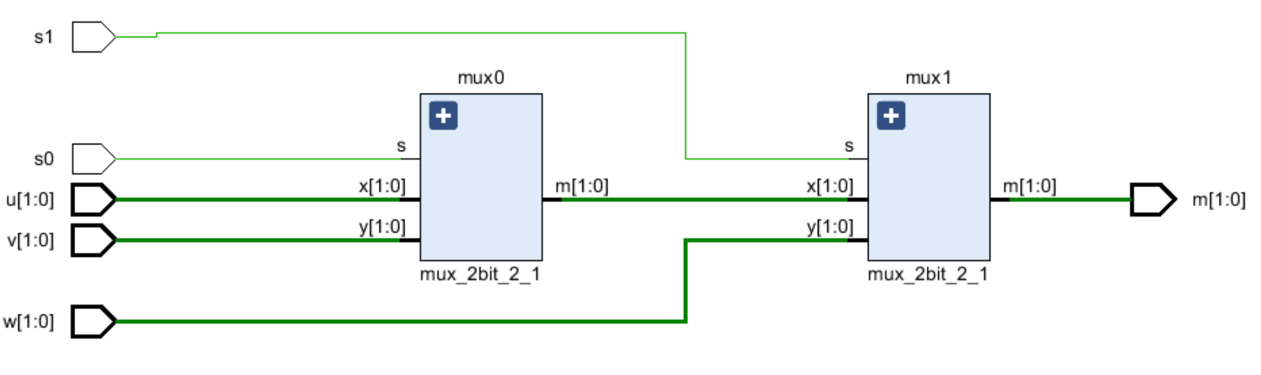


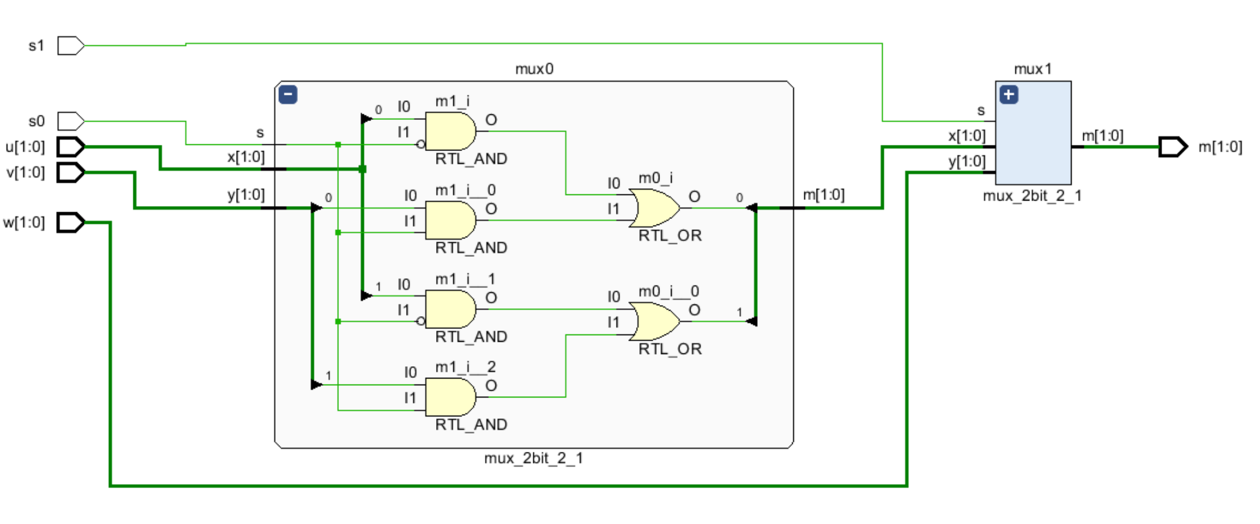




2.4 Mixed-Design Modelling

To bring it all together in a more useful application of the modelling methods, the dataflow modelled 2-to-1 mux was imported into the project and used to build a 3-to-1 mux. In this case, the mux_2bit_2_1 was imported and mapped, a intermediate output signal created to feed the first out into the second, and the circuit functionality/connections defined using structural modelling. This is an example of good application of the structural method of using smaller modules to construct a more complex function.





2.5 BCD to 7 segment

In a previous the 7 segment display was used. This time a config word was used to set the use a specific displayed digit by setting/clearing the anodes. So the display of the first digit, its anode is cleared and the rest set high. `an <= "11111110"`. To speed up the process the previously written BCB to 7 segment encoding lookup table (select statement) was used.

Appendix

Part 1 - Dataflow

Mux

```
library IEEE;
use IEEE.std_logic_1164.all;

entity lab_2_1_1 is
    Port ( x : in std_logic;
           y : in std_logic;
           s : in std_logic;
           m : out std_logic);
end lab_2_1_1;

architecture Behavioral of lab_2_1_1 is

begin
    m <= (x and (not s)) or (s and y);
end Behavioral;
```

Two Bit Mux

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

entity mux_2bit_2_1 is
    Port ( x : in std_logic_vector (1 downto 0);
           y : in std_logic_vector(1 downto 0);
           s : in std_logic;
           m : out std_logic_vector (1 downto 0));
end mux_2bit_2_1;

architecture Behavioral of mux_2bit_2_1 is

begin
    m(0) <= (x(0) and (not s)) or (y(0) and s);
    m(1) <= (x(1) and (not s)) or (y(1) and s);
end Behavioral;
```

Mux delay

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

entity mux_2bit_2_1 is
    Port ( x : in std_logic_vector (1 downto 0);
           y : in std_logic_vector(1 downto 0);
           s : in std_logic;
           m : out std_logic_vector (1 downto 0));
end mux_2bit_2_1;

architecture Behavioral of mux_2bit_2_1 is

begin
    m(0) <= (x(0) and (not s)) or (y(0) and s) after 3 ns;
    m(1) <= (x(1) and (not s)) or (y(1) and s) after 3 ns;
end Behavioral;
```

Part 2 - Structural

Two Bit Mux

```
library IEEE;
use IEEE.std_logic_1164.all;

entity lab_2_3_1 is
    Port ( x : in std_logic_vector (1 downto 0);
           y : in std_logic_vector (1 downto 0);
           s : in std_logic;
           m : out std_logic_vector (1 downto 0));
end lab_2_3_1;

architecture Behavioral of lab_2_3_1 is

    signal not_sig : std_logic;
    signal and_sig : std_logic_vector (3 downto 0);

    -- 2 input and gate
    component and2
        port(
            i0, i1 : in std_logic;
            o : out std_logic);
    end component;

    -- 2 input OR gate
    component or2
        port(
            i0, i1 : in std_logic;
            o : out std_logic);
    end component;

    -- 1 inpput NOT gate
    component inv
        port(
            i : in std_logic ;
            o : out std_logic );
    end component;

begin

    not_comp : inv
        port map(
            i => s,
            o => not_sig);

    and_comp0 : and2
        port map(
            i0 => x(0),
            i1 => not_sig,
            o => and_sig(0));

    and_comp1 : and2
        port map(
            i0 => x(1),
            i1 => not_sig,
            o => and_sig(1));

    and_comp2 : and2
        port map(
            i0 => y(0),
            i1 => s,
            o => and_sig(2));

    and_comp3 : and2
        port map(
            i0 => y(1),
            i1 => s,
            o => and_sig(3));

    or_comp0 : or2
        port map(
            i0 => and_sig(0),
            i1 => and_sig(2),
            o => m(0));

    or_comp1 : or2
        port map(
            i0 => and_sig(1),
            i1 => and_sig(3),
            o => m(1));

end Behavioral;
```

Part 3 - Behavioural

Mux

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

entity lab_2_4_1 is
    Port ( x : in std_logic;
           y : in std_logic;
           s : in std_logic;
           m : out std_logic);
end lab_2_4_1;

architecture Behavioral of lab_2_4_1 is
begin

mux_2_1 : process (x,y,s)
    begin
        if(s='0') then
            m <= y;
        else
            m <= x;
        end if;
    end process ;
end Behavioral;
```

Two Bit Mux

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

entity lab_2_4_2 is
    Port ( x : in std_logic_vector (1 downto 0);
           y : in std_logic_vector (1 downto 0);
           s : in std_logic;
           m : out std_logic_vector (1 downto 0));
end lab_2_4_2;

architecture Behavioral of lab_2_4_2 is

begin
    process (x,y,s)
        begin
            if (s='0') then
                m <= x;
            else
                m <= y;
            end if;
    end process;

end Behavioral;
```

Part 4 - Mixed

3 to 1 Mux

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

entity mux_3_to_1 is
    Port ( u : in std_logic_vector (1 downto 0);
           v : in std_logic_vector (1 downto 0);
           w : in std_logic_vector (1 downto 0);
           s0 : in std_logic;
           s1 : in std_logic;
           m : out std_logic_vector (1 downto 0));
end mux_3_to_1;

architecture Behavioral of mux_3_to_1 is

    signal m_int : std_logic_vector (1 downto 0);

    component mux_2bit_2_1
        port ( x, y: in std_logic_vector (1 downto 0);
           s : in std_logic;
           m : out std_logic_vector (1 downto 0));
    end component;

begin

    mux0 : mux_2bit_2_1
        port map(
            x => u,
            y => v,
            s => s0,
            m => m_int);

    mux1 : mux_2bit_2_1
        port map(
            x => m_int,
            y => w,
            s => s1,
            m => m);

end Behavioral;
```

DCD to 7 seg

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

entity dcd_7seg is
    Port ( x : in STD_LOGIC_VECTOR (3 downto 0);
           an : out STD_LOGIC_VECTOR (7 downto 0);
           seg : out STD_LOGIC_VECTOR (6 downto 0));
end dcd_7seg;

architecture Behavioral of dcd_7seg is

begin

    an <= "01111111";

    with x select
        seg <= "1111001" when "0001", --1
                   "0100100" when "0010", --2
                   "0110000" when "0011", --3
                   "0011001" when "0100", --4
                   "0010010" when "0101", --5
                   "0000010" when "0110", --6
                   "1111000" when "0111", --7
                   "0000000" when "1000", --8
                   "0010000" when "1001", --9
                   "0001000" when "1010", --a
                   "0000011" when "1011", --b
                   "1000110" when "1100", --c
                   "0100001" when "1101", --d
                   "0000110" when "1110", --e
                   "0001110" when "1111", --f
                   "1000000" when others; --0
end Behavioral;
```