Labo 1 Digitale Elektronica: Modeling Concepts

Probleemanalyse

De VHDL modellering taal ondersteund drie soorten modellering stijlen: dataflow, structural en behavioral. In dit labo wordt aangeleerd om met deze stijlen te werken.

2-1. Create a 2-to-1 multiplexer using dataflow modeling.

Dataflow modeling kan gebruikt worden om combinatorische logica te beschrijven. In deze opgave is het de bedoeling dat deze modellering stijl gebruikt wordt om een 2-to-1 multiplexer te maken. Dit gebeurt als volgt:

```
entity Lab1_2_1 is
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           s : in STD_LOGIC;
           m : out STD_LOGIC);
end Lab1_2_1;

architecture Behavioral of Lab1_2_1 is

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

end Behavioral;
```

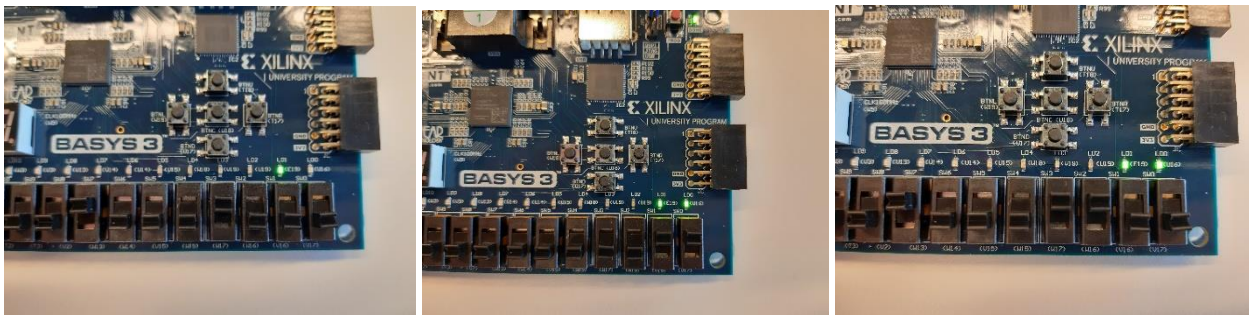
In de entity worden de verschillende gebruikte poorten gedefinieerd. Vervolgens wordt in de architecture beschreven wat onze hardware juist gaat doen. Dit gebeurt aan de hand van simpele comando's zoals *and* voor een and-gate, *or* voor een or-gate en *not* voor een inverter.

2-2. Create a two-bit wide 2-to-1 multiplexer using dataflow modeling.

Hier wordt dataflow modeling gebruikt om dezelfde 2-to-1 multiplexer te maken, maar dan met een twee bit input. Deze code wordt nogal omslachtig, maar is nog steeds bruikbaar.

```
Entity mux_2bit_2_to_1_gate 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_to_1_gate;

Architecture behavior of mux_2bit_2_to_1_gate Is

Signal s_bar : STD_LOGIC;
Signal x_int : STD_LOGIC_VECTOR(1 downto 0);
Signal y_int : STD_LOGIC_VECTOR(1 downto 0);

begin

        s_bar <= not s;
        x_int(1) <= x(1) and s_bar;
        y_int(1) <= y(1) and s;
        m(1) <= x_int(1) or y_int(1);
        x_int(0) <= x(0) and s_bar;
        y_int(0) <= y(0) and s;
        m(0) <= x_int(0) or y_int(0);

end behavior;
```

De verschillende poorten waarvoor meerdere bits nodig zijn, worden nu gedefinieerd met *STD_LOGIC_VECTOR* in plaats van *STD_LOGIC.*



2-3. Model a two-bit wide 2-to-1 multiplexer using dataflow modeling with net delays of 3 ns.

Het toevoegen van delay's is dan weer eenvoudig. Deze worden achteraan toegevoegd en werken enkel in op de lijn code waar ze bij staan.

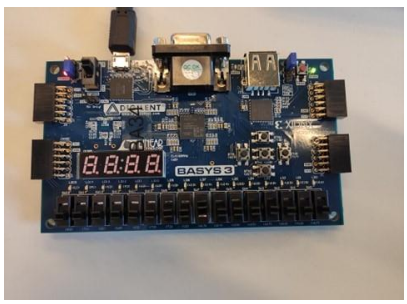
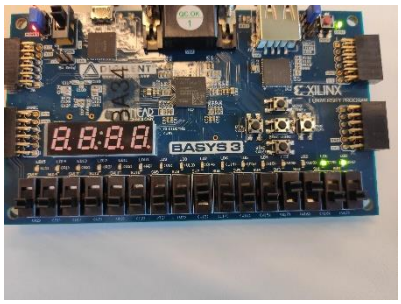
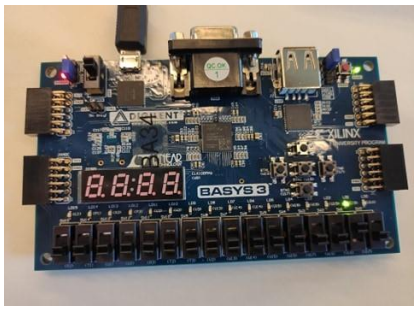
```
Entity mux_2bit_2_to_1_gate Is
port (
        x : in STD_LOGIC_VECTOR(1 downto 0);
        y : in STD_LOGIC_VECTOR(1 downto 0);
        s : in STD_LOGIC;
        m : out STD_LOGIC_VECTOR(1 downto 0)
    );
end mux_2bit_2_to_1_gate;

Architecture behavior of mux_2bit_2_to_1_gate Is
begin

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

end behavior;
```

3-1. Re-create the earlier lab 2-2 using structural modeling.

Bij structural modeling worden eerst aparte componenten gedefinieerd. Daarna wordt de hardware beschreven aan de hand van deze componenten. Dit vereenvoudigt het beschrijven van complexe systemen.

```
Entity mux_2bit_2_to_1_structural 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_to_1_structural;

architecture behavior of mux_2bit_2_to_1_structural is
    component and2 port
    ( i0, i1: in std_logic;
      o : out std_logic);
    end component;

    component or2 port
    ( i0, i1: in std_logic;
    o : out std_logic);
    end component;

    component inv port
    ( i: in std_logic;
    o : out std_logic);
    end component;

    signal s_bar : std_logic;
    signal x_int : std_logic_vector(1 downto 0);
    signal y_int : std_logic_vector(1 downto 0);
begin
    n1 : inv port map (i => s, o => s_bar);
    a1 : and2 port map (i0 => x(1), i1 => s_bar, o => x_int(1));
    a2 : and2 port map (i0 => y(1), i1 => s, o => y_int(1));
    o1 : or2 port map (i0 => x_int(1), i1 => y_int(1), o => m(1));
    a3 : and2 port map (i0 => x(0), i1 => s_bar, o => x_int(0));
    a4 : and2 port map (i0 => y(0), i1 => s, o => y_int(0));
    o2 : or2 port map (i0 => x_int(0), i1 => y_int(0), o => m(0));
end behavior;
```

In bovenstaande code worden eerst de componenten gedefinieerd en hun pinnen benoemd. Vervolgens worden deze pinnen en de poorten gedefinieerd in de entity gekoppeld.

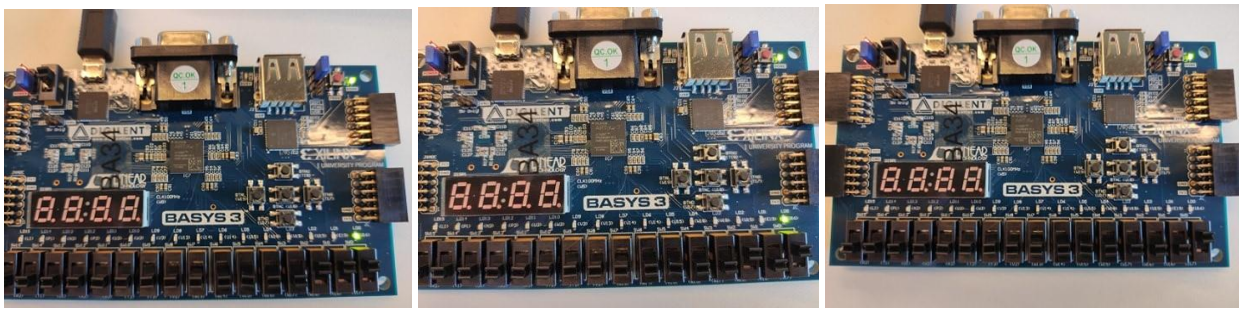




4-1. Create a 2-to-1 multiplexer using behavioral modeling.

Hier wordt dezelfde 2-to-1 multiplexer gemaakt met behavioral modeling in plaats van structural modeling. Deze modellering stijl wordt gebruikt voor het beschrijven van complexe circuits.

```
entity labl_4_1 is
    Port ( x : in STD_LOGIC;
           y : in STD_LOGIC;
           s : in STD_LOGIC;
           m : out STD_LOGIC);
end labl_4_1;

architecture Behavioral of labl_4_1 is

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

In de architecture block worden processen gedefinieerd om sequentiële logica te modelleren.

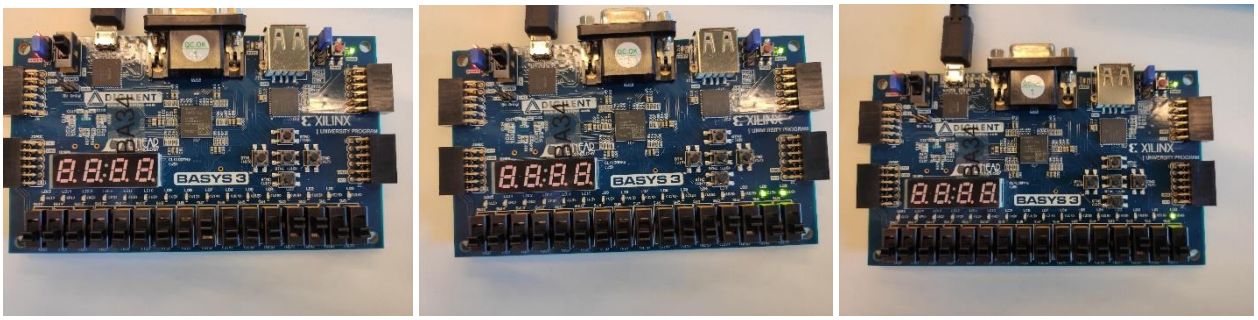


4-2. Create a two-bit wide 2-to-1 multiplexer using behavioral modeling.

Hier wordt dezelfde twee bit input multiplexer gemaakt, maar dan met behavioral modeling.

```
entity Labl_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 Labl_4_2;

architecture Behavioral of Labl_4_2 is
Signal m_int : STD_LOGIC_VECTOR (1 downto 0);

begin
m <= m_int;
process (x(1), x(0), y(1), y(0), s)
begin
      if(s='0') then
            m_int(0) <= y(0);
            m_int(1) <= y(1);
      else
            m_int(0) <= x(0);
            m_int(1) <= x(1);
end if;
end process;

end Behavioral;
```

Analoog aan 2-2 wordt hier *STD_LOGIC_VECTOR* gebruiktin plaats van *STD_LOGIC*.



5-1. Model a 3-to-1 multiplexer using 2-to-1 multiplexers.

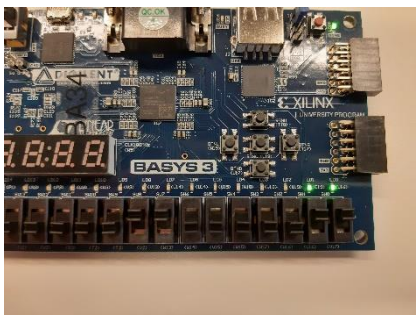
Hier wordt gebruik gemaakt van mixed-design style modeling. Deze stijl gebruik een hiërarchische beschrijving. Het design kan beschreven worden gebruikmakend van:

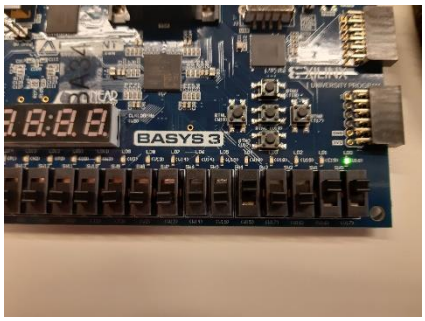
- Dataflow modeling
- Structural modeling
- Behavioral modeling
- En combinaties van bovenstaande

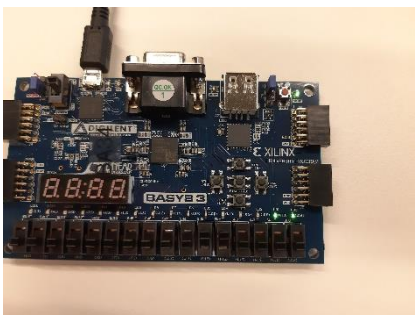
Hier wordt een 3-to-1 multiplexer gemaakt met de eerder gemaakte 2-to-1 multiplexer. Hiervan zullen er twee opgeroepen worden met structural modeling en aan elkaar gekoppeld worden.

```
entity lab1_5_1 is
    Port ( u : in STD_LOGIC_VECTOR (0 to 1);
           v : in STD_LOGIC_VECTOR (0 to 1);
           w : in STD_LOGIC_VECTOR (0 to 1);
           s0 : in STD_LOGIC;
           s1 : in STD_LOGIC;
           m : out STD_LOGIC_VECTOR (0 to 1));
end lab1_5_1;

architecture Behavioral of lab1_5_1 is

    component mux_2bit_2_to_1_structural
    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 component;

    Signal out1 : STD_LOGIC_VECTOR (0 to 1);
    Signal m_int : STD_LOGIC_VECTOR (0 to 1);

begin

    mp1 : mux_2bit_2_to_1_structural PORT MAP (
        x => u,
        y => v,
        s => s0,
        m => out1
    );
    mp2 : mux_2bit_2_to_1_structural PORT MAP (
        x => w,
        y => out1,
        s => s1,
        m => m_int
    );

    m <= m_int;

end Behavioral;
```

Hierin wordt de eerder ontwikkelde mux_2bit_2_to_1_structural gebruikt als component.

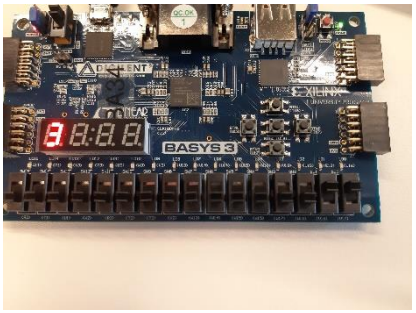
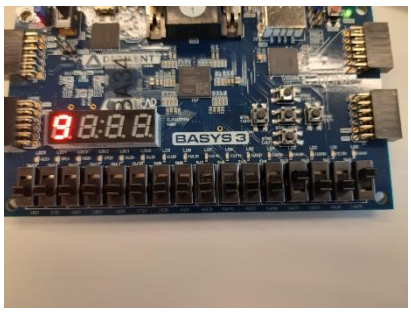






5-2. Model a BCD to 7-Segment Decoder.

Hier wordt met de verworven kennis een BCD-to-7-segment encoder gemaakt. Een 7-segment display wordt gebruikt om decimale getallen weer te geven. De code om ieder segment aan te sturen wordt geschreven met dataflow modeling.


Conclusie

De drie aangeleerde modellering stijlen hebben ieder hun sterktes en zwaktes. Door deze goed te begrijpen en ze aanvullend met elkaar te gebruiken kunnen de meeste complexe problemen opgelost worden.