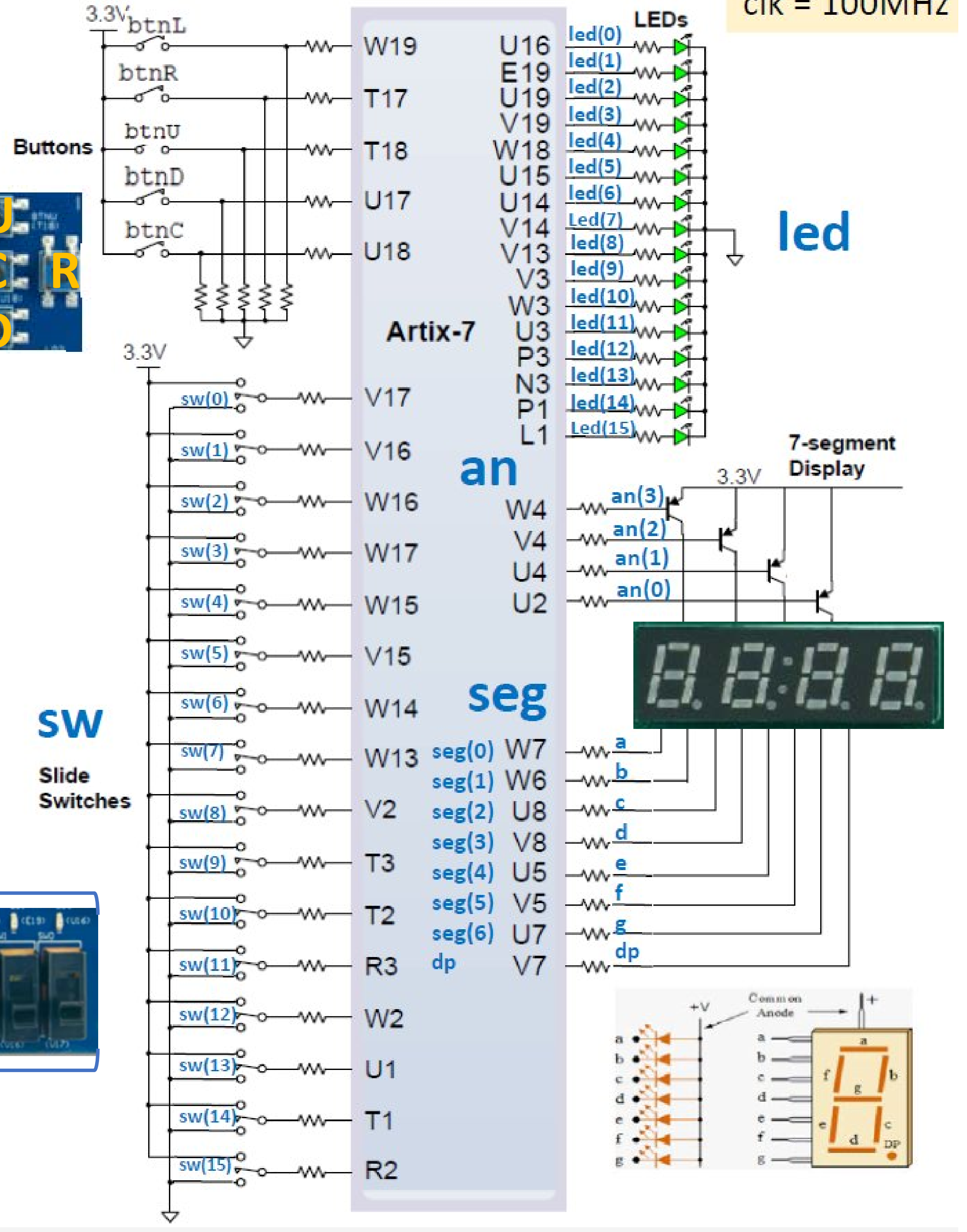
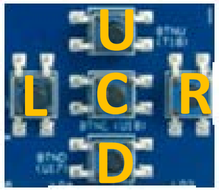
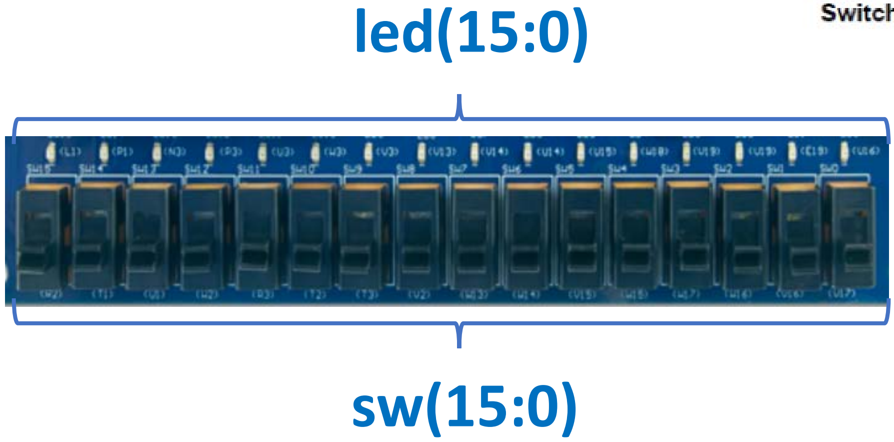
clk = 100MHz


3.3V
btnL
btnR
btnU
btnD
btnC
Buttons
W19
T17
T18
U17
U18
Artix-7
LEDs
led(0) U16
led(1) E19
led(2) U19
led(3) V19
led(4) W18
led(5) U15
led(6) U14
Led(7) V14
led(8) V13
led(9) V3
led(10) W3
led(11) U3
led(12) P3
led(13) N3
led(14) P1
Led(15) L1
led
sw
Slide Switches
sw(0) V17
sw(1) V16
sw(2) W16
sw(3) W17
sw(4) W15
sw(5) V15
sw(6) W14
sw(7) W13
sw(8) V2
sw(9) T3
sw(10) T2
sw(11) R3
sw(12) W2
sw(13) U1
sw(14) T1
sw(15) R2
an
an(3) W4
an(2) V4
an(1) U4
an(0) U2
7-segment Display
seg
seg(0) W7 a
seg(1) W6 b
seg(2) U8 c
seg(3) V8 d
seg(4) U5 e
seg(5) V5 f
seg(6) U7 g
dp V7 dp
+V
Common Anode
a b c d e f g
DP


U
L
C
R
D


led(15:0)
sw(15:0)

8.1 Seven-Segment Display

Uddrag fra Basys 3 Reference manual

The Basys3 board contains one four-digit common anode seven-segment LED display. Each of the four digits is composed of seven segments arranged in a “figure 8” pattern, with an LED embedded in each segment. Segment LEDs can be individually illuminated, so any one of 128 patterns can be displayed on a digit by illuminating certain LED segments and leaving the others dark, as shown in Fig 17. Of these 128 possible patterns, the ten corresponding to the decimal digits are the most useful.

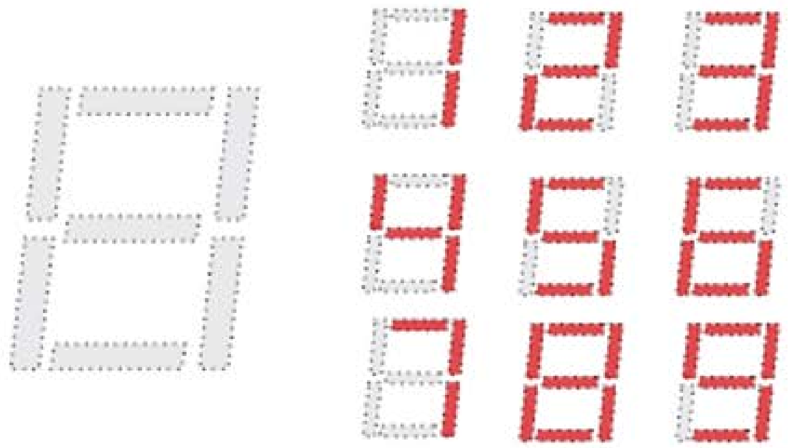


Figure 17. An un-illuminated seven-segment display, and nine illumination patterns corresponding to decimal digits

The anodes of the seven LEDs forming each digit are tied together into one “common anode” circuit node, but the LED cathodes remain separate, as shown in Fig 18. The common anode signals are available as four “digit enable” input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four “D” cathodes from the four digits are grouped together into a single circuit node called “CD”). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted.

The anodes of the seven LEDs forming each digit are tied together into one "common anode" circuit node, but the LED cathodes remain separate, as shown in Fig 18. The common anode signals are available as four "digit enable" input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four "D" cathodes from the four digits are grouped together into a single circuit node called "CD"). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted.

To illuminate a segment, the anode should be driven high while the cathode is driven low. However, since the Basys3 uses transistors to drive enough current into the common anode point, the anode enables are inverted. Therefore, both the AN0..3 and the CA..G/DP signals are driven low when active.

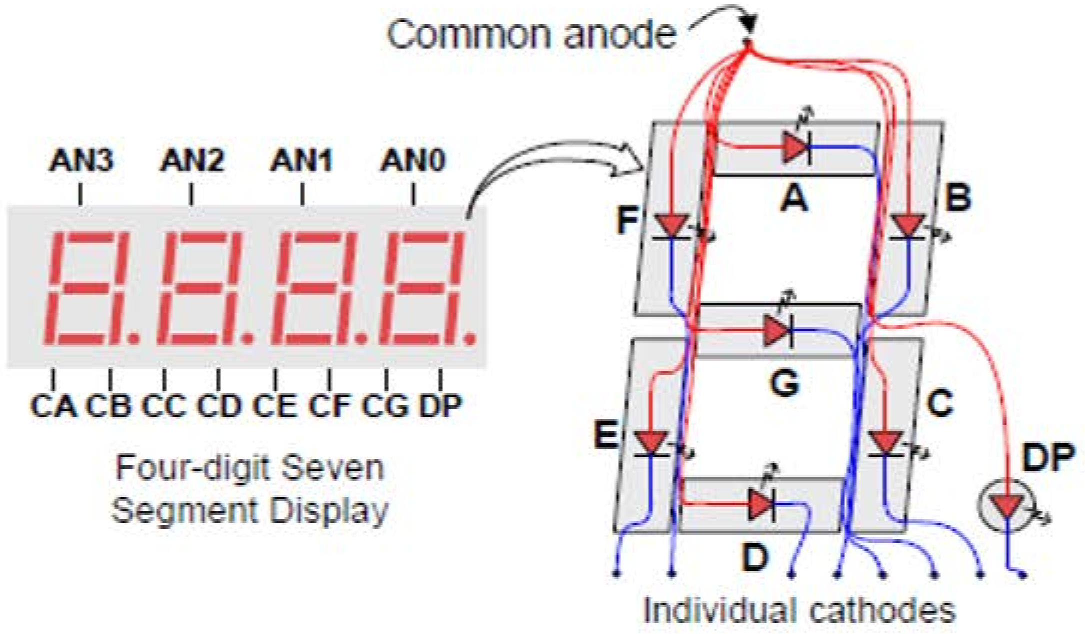


Figure 18. Common anode circuit node

A scanning display controller circuit can be used to show a four-digit number on this display. This circuit drives the anode signals and corresponding cathode patterns of each digit in a repeating, continuous succession at an update rate that is faster than the human eye can detect. Each digit is illuminated just one-fourth of the time, but because the eye cannot perceive the darkening of a digit before it is illuminated again, the digit appears continuously illuminated. If the update, or "refresh", rate is slowed to around 45 hertz, a flicker can be noticed in the display.

For each of the four digits to appear bright and continuously illuminated, all four digits should be driven once every 1 to 16ms, for a refresh frequency of about 1KHz to 60Hz. For example, in a 62.5Hz refresh scheme, the entire display would be refreshed once every 16ms, and each digit would be illuminated for 1/4 of the refresh cycle, or 4ms. The controller must drive low the cathodes with the correct pattern when the corresponding anode signal is driven high. To illustrate the process, if AN0 is asserted while CB and CC are asserted, then a "1" will be displayed in digit position 1. Then, if AN1 is asserted while CA, CB, and CC are asserted, a "7" will be displayed in digit position 2. If AN0, CB, and CC are driven for 4ms, and then AN1, CA, CB, and CC are driven for 4ms in an endless succession, the display will show "71" in the first two digits. An example timing diagram for a four-digit controller is shown in Fig 19.

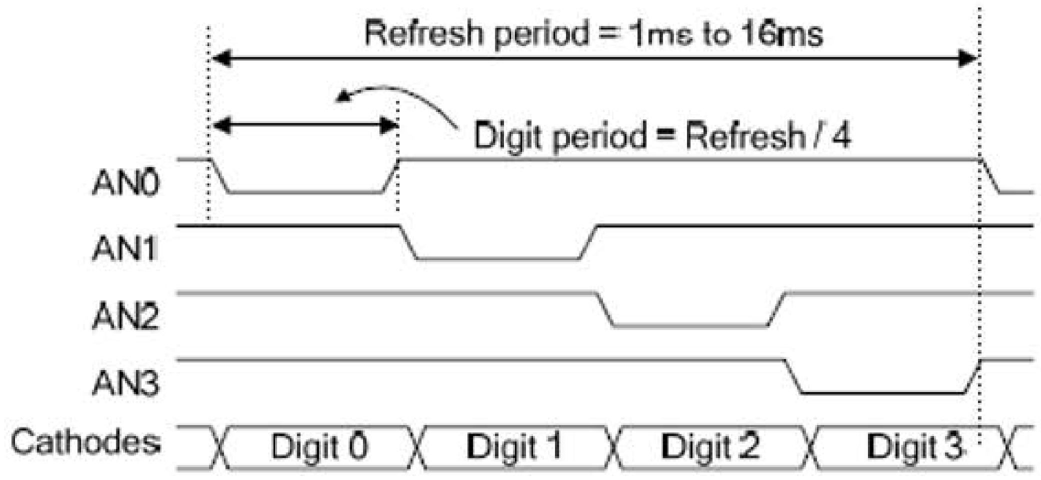
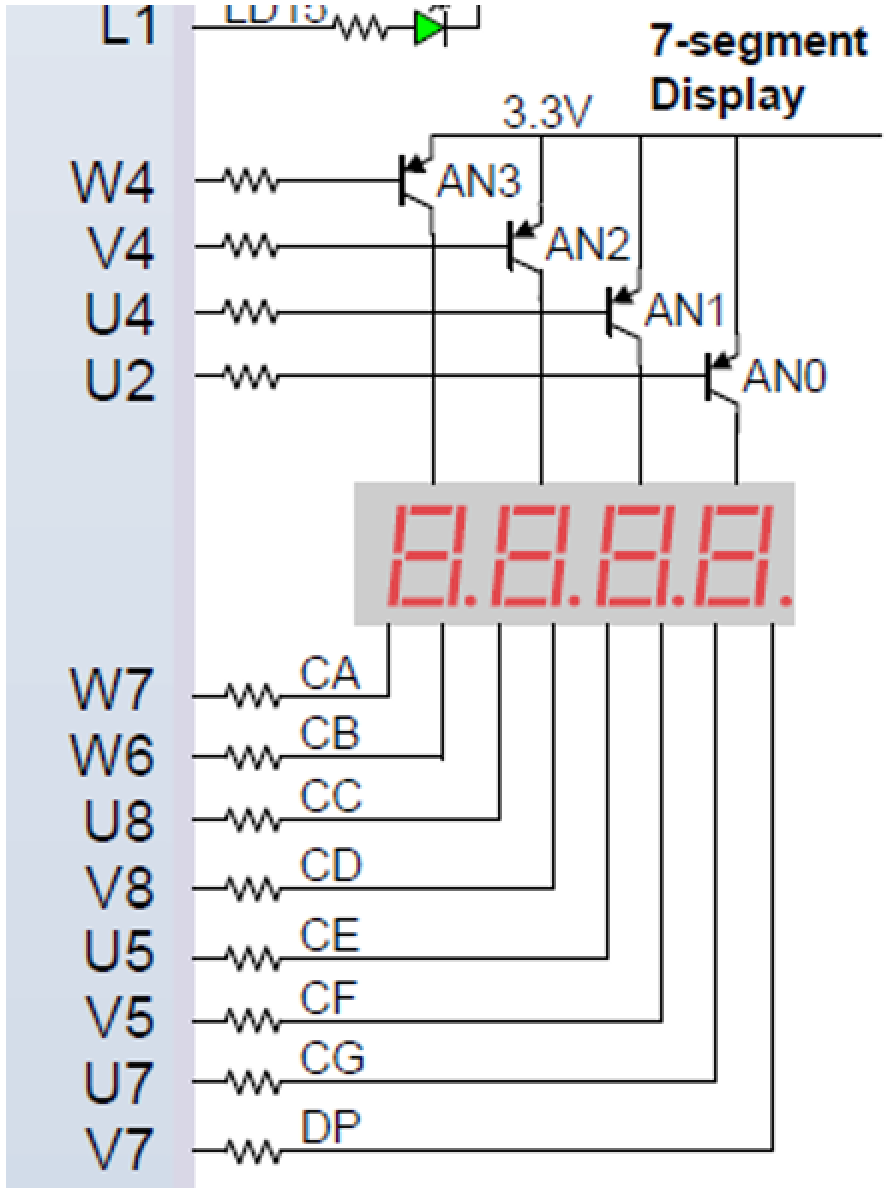
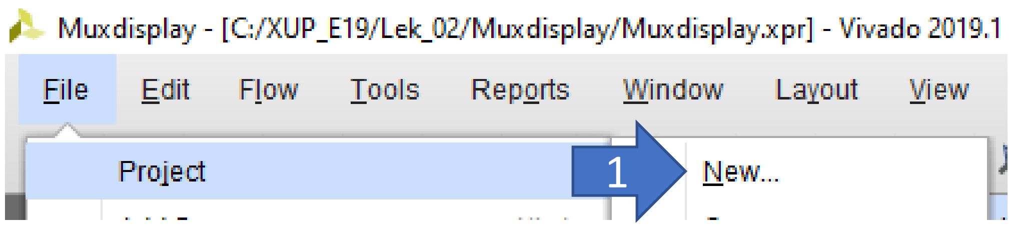


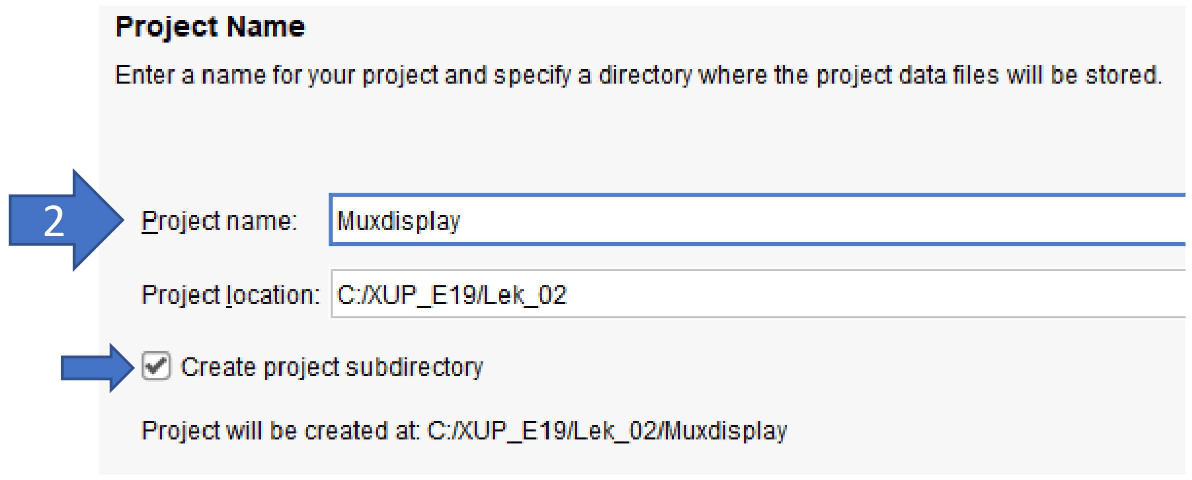
Figure 19. Four digit scanning display controller timing diagram

Der er med andre ord behov for at lave et kredsløb med Multipleksere, Decodere og senere tællere.



Muxdisplay - Opgave 1 – Lav et projekt og få ”hul igennem”


Muxdisplay - [C:/XUP_E19/Lek_02/Muxdisplay/Muxdisplay.xpr] - Vivado 2019.1


Project Name

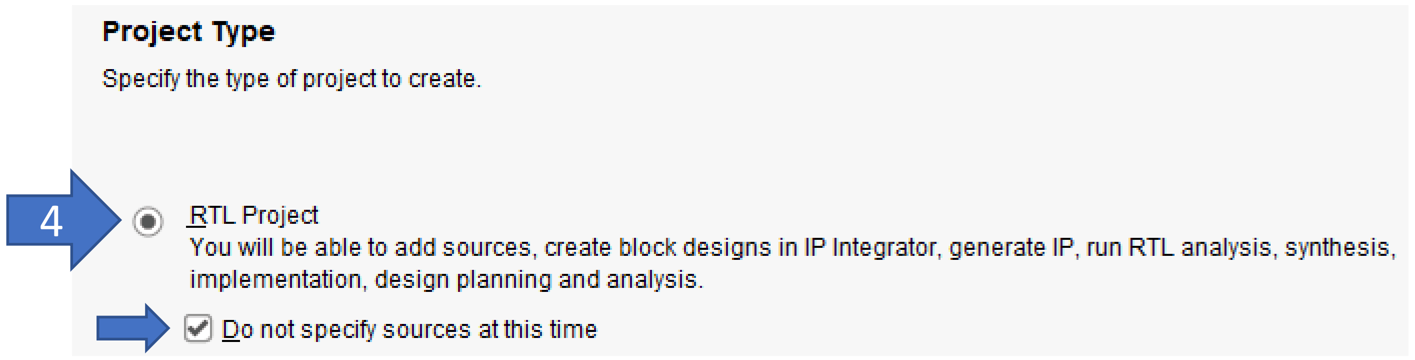
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: Muxdisplay

Project location: C:/XUP_E19/Lek_02

[✓] Create project subdirectory

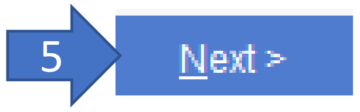
Project will be created at: C:/XUP_E19/Lek_02/Muxdisplay

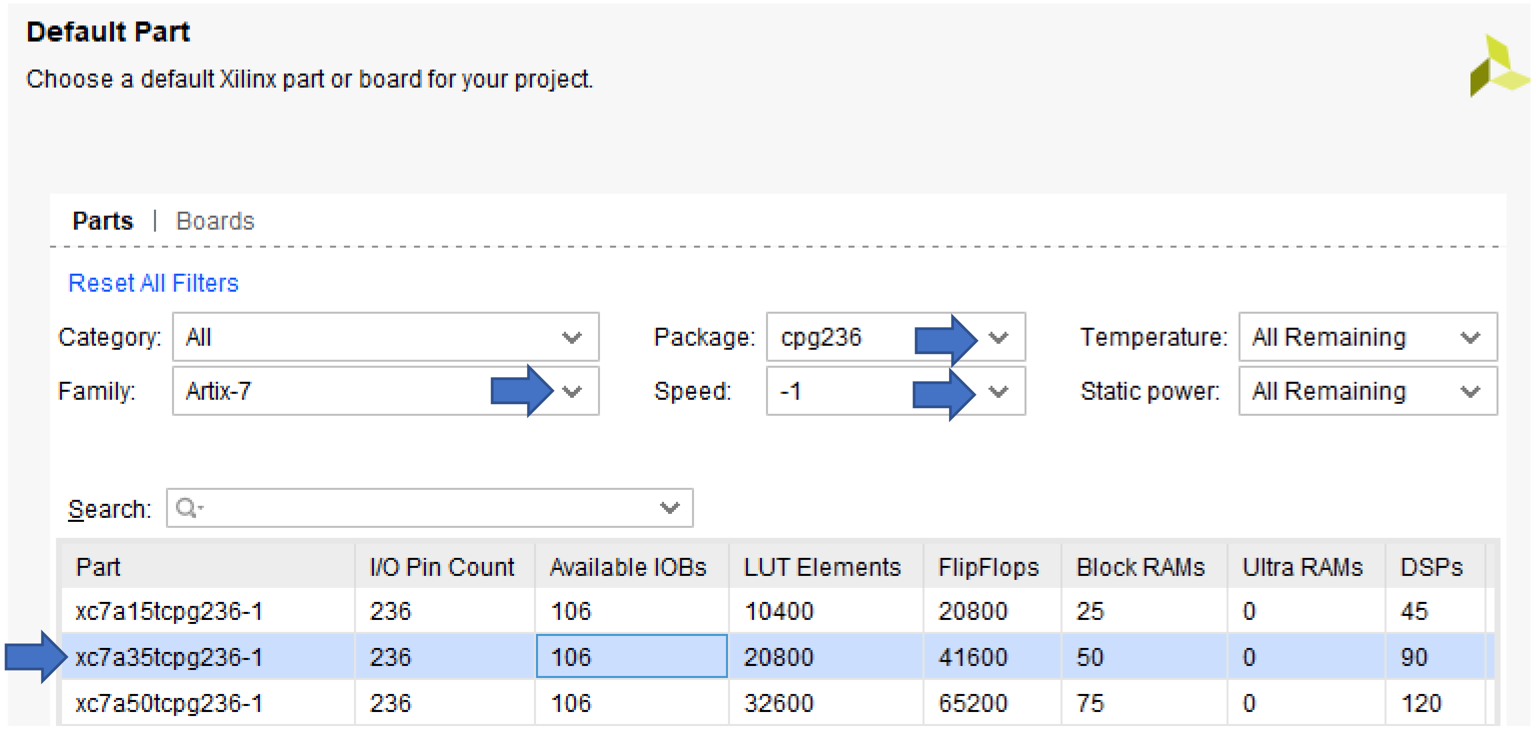

Project Type

Specify the type of project to create.

● RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.

[✓] Do not specify sources at this time




Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

Reset All Filters

Category: All

Family: Artix-7

Package: cpg236

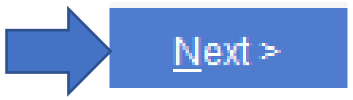
Speed: -1

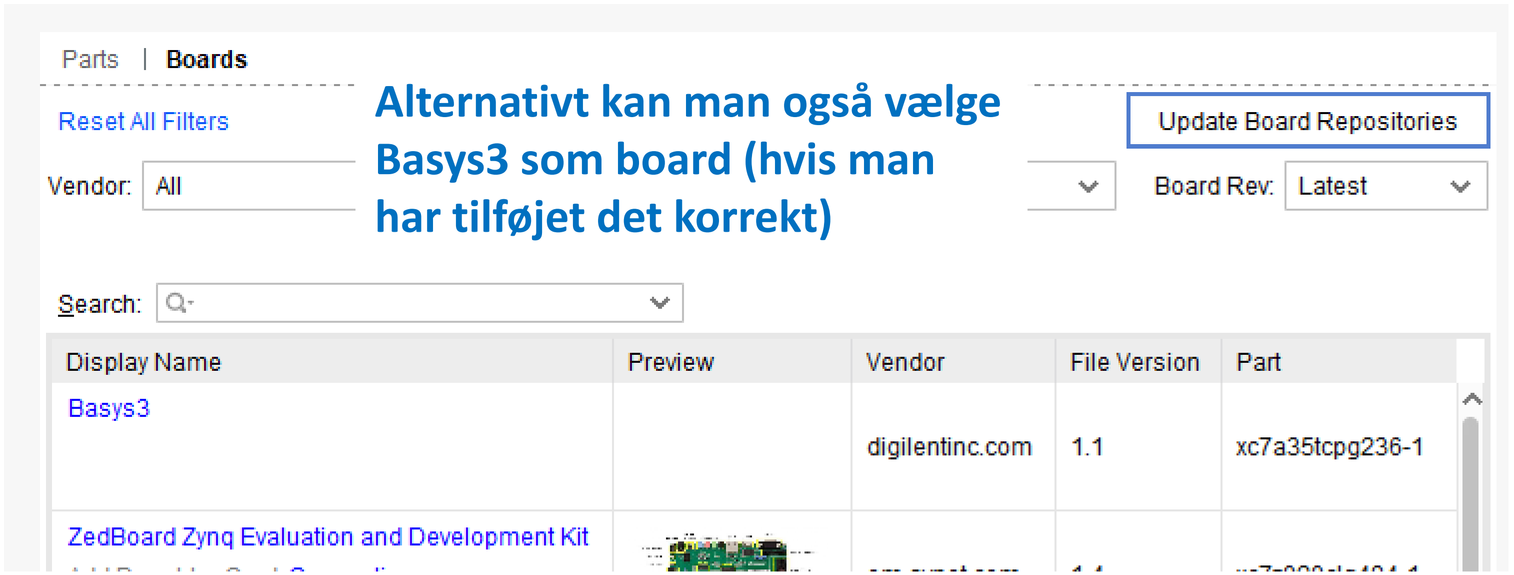
Temperature: All Remaining

Static power: All Remaining

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs
xc7a15tcpg236-1	236	106	10400	20800	25	0	45
xc7a35tcpg236-1	236	106	20800	41600	50	0	90
xc7a50tcpg236-1	236	106	32600	65200	75	0	120




Parts | Boards

Reset All Filters

Vendor: All

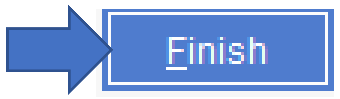
Board Rev: Latest

Update Board Repositories

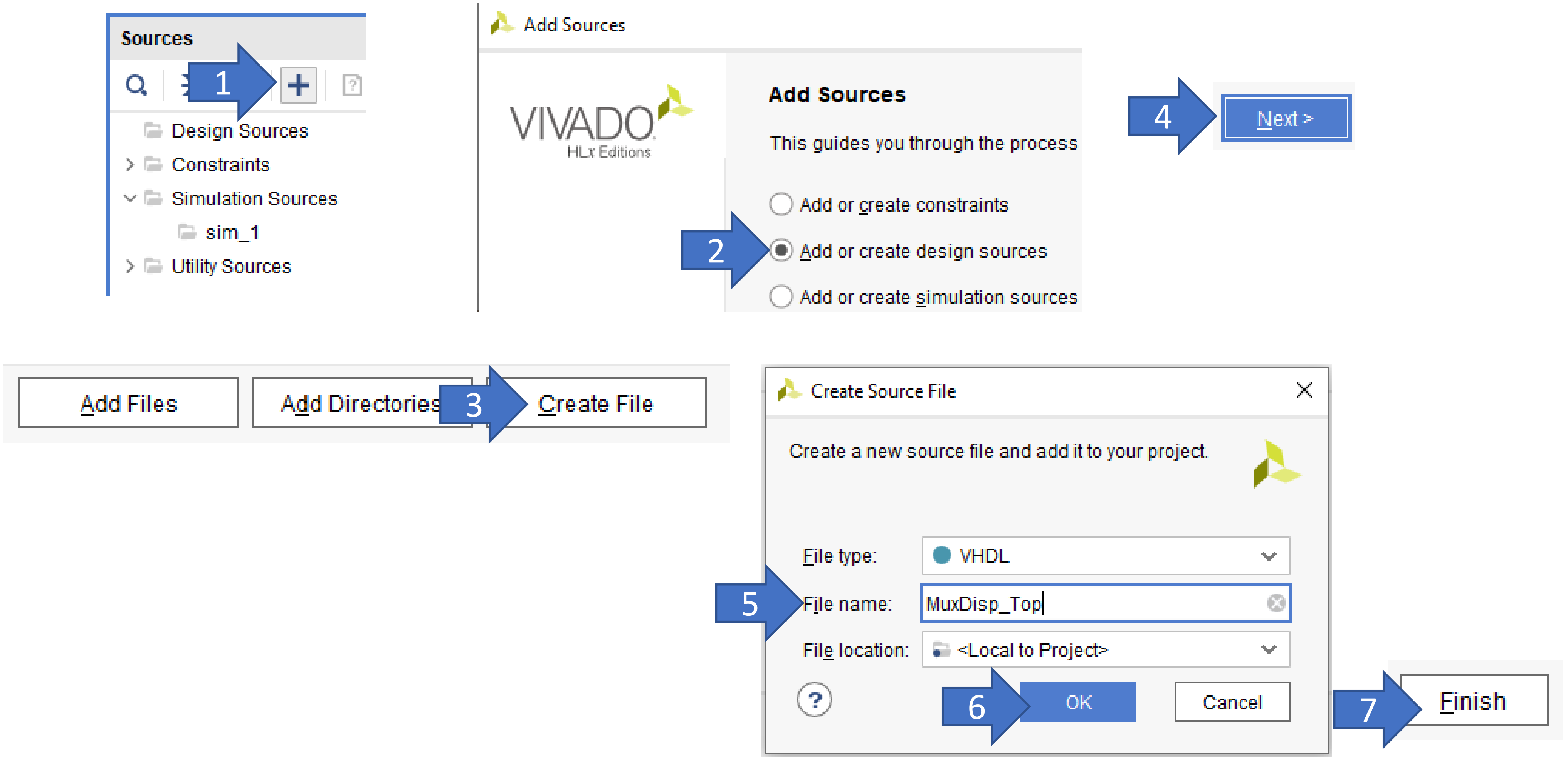
Search:

Display Name	Preview	Vendor	File Version	Part
Basys3		digilentinc.com	1.1	xc7a35tcpg236-1
ZedBoard Zynq Evaluation and Development Kit				

Alternativt kan man også vælge Basys3 som board (hvis man har tilføjet det korrekt)



Add Source – nyt design



Add Sources

This guides you through the process

- Add or create constraints
- [✓ Add or create design sources]
- Add or create simulation sources

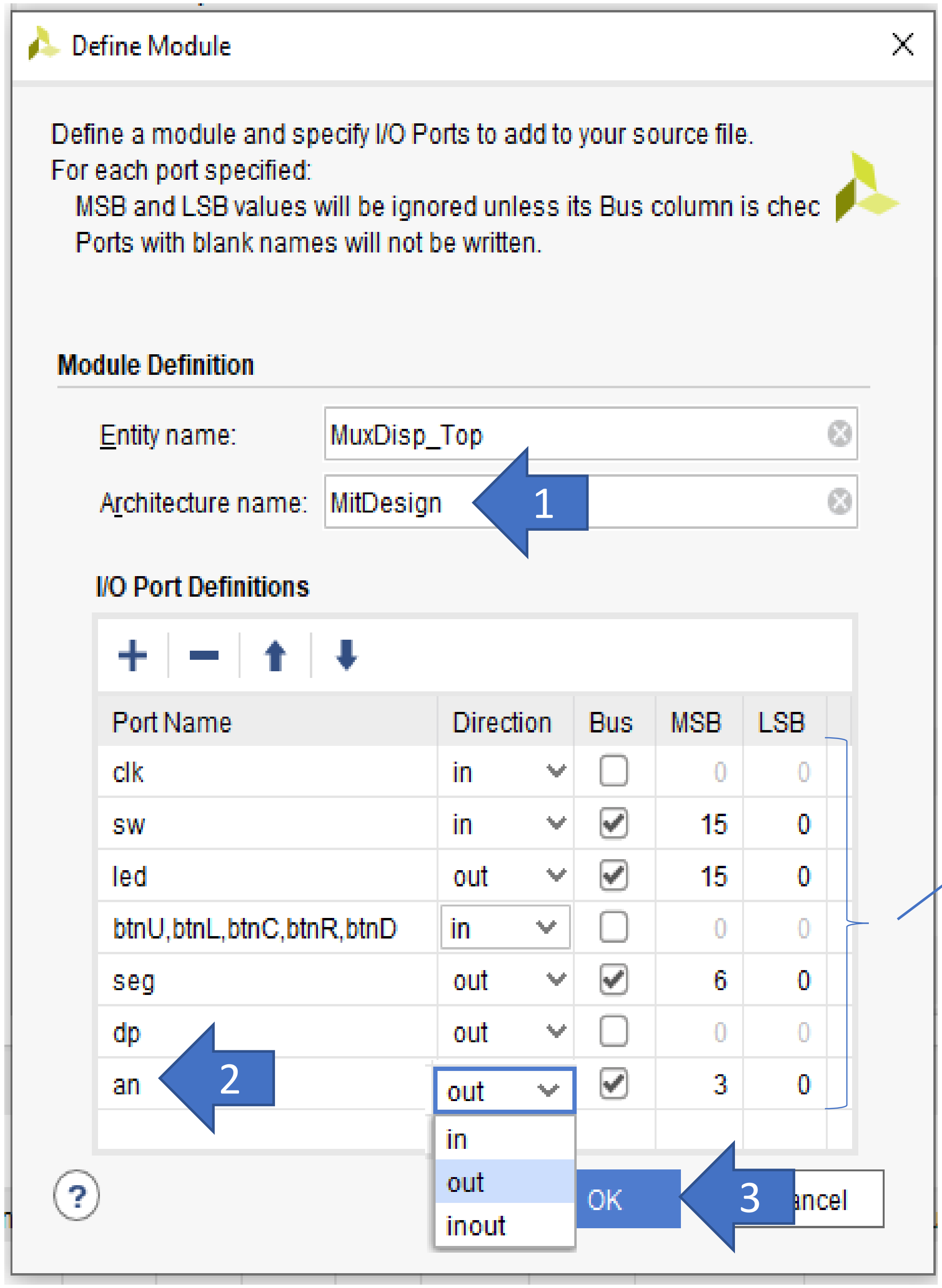
Create Source File

Create a new source file and add it to your project.

File type: VHDL

File name: MuxDisp_Top

File location: <Local to Project>


Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is chec
Ports with blank names will not be written.

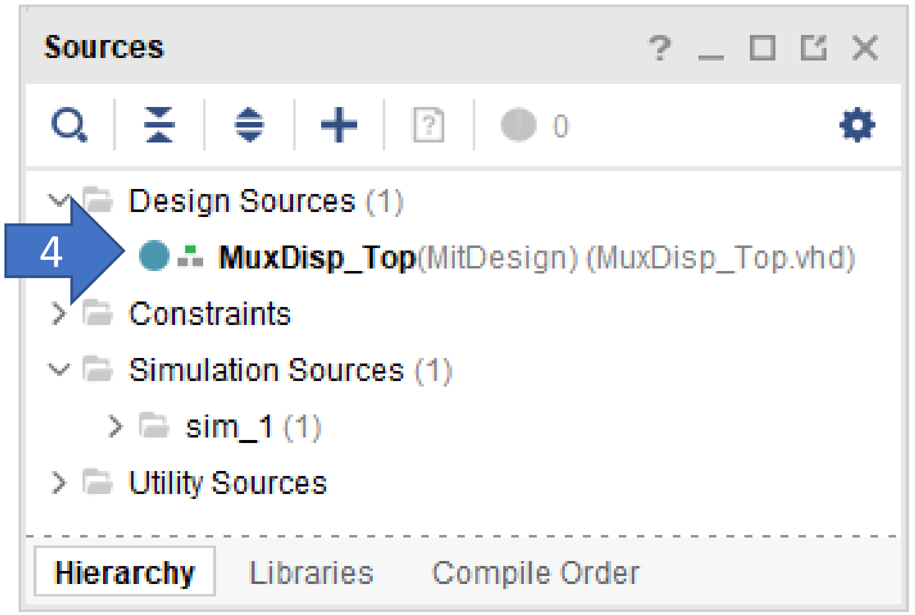
Module Definition

Entity name: MuxDisp_Top

Architecture name: MitDesign

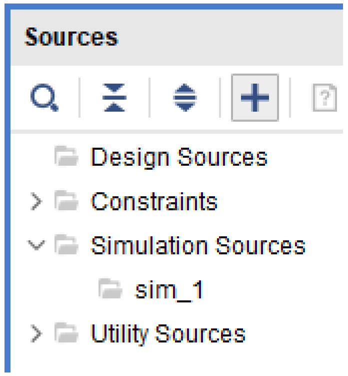
I/O Port Definitions

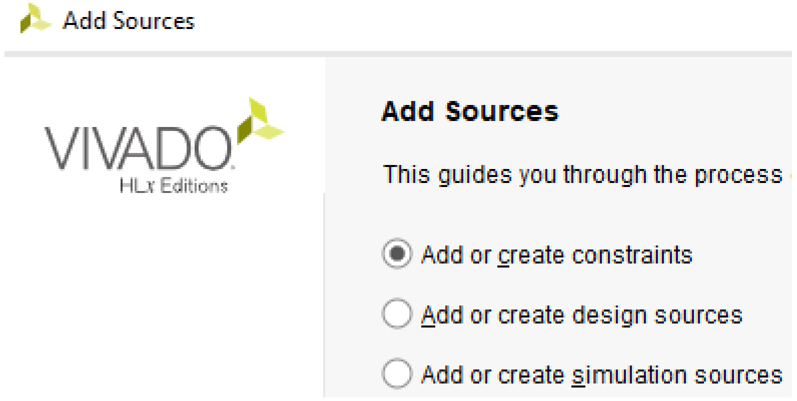
Port Name	Direction	Bus	MSB	LSB
clk	in	☐	0	0
sw	in	✓	15	0
led	out	✓	15	0
btnU,btnL,btnC,btnR,btnD	in	☐	0	0
seg	out	✓	6	0
dp	out	☐	0	0
an	out	✓	3	0



```
entity MuxDisp_Top is
    Port ( clk : in STD_LOGIC;
           sw : in STD_LOGIC_VECTOR (15 downto 0);
           led : out STD_LOGIC_VECTOR (15 downto 0);
           btnU,btnL,btnC,btnR,btnD : in STD_LOGIC;
           seg : out STD_LOGIC_VECTOR (6 downto 0);
           dp : out STD_LOGIC;
           an : out STD_LOGIC_VECTOR (3 downto 0));
end MuxDisp_Top;

architecture MitDesign of MuxDisp_Top is

begin

end MitDesign;
```

Add Sources – Constraints .. Hent Basys3_Master.XDC fra Blackboard

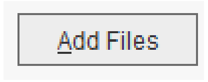


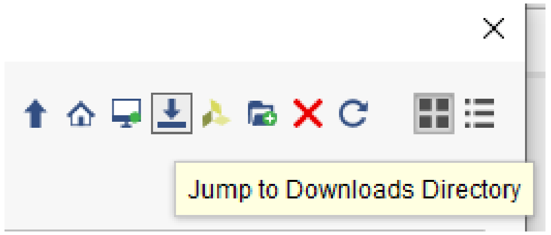

Add Sources

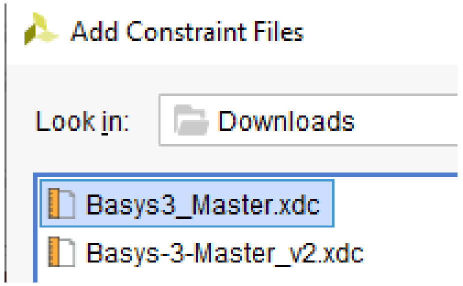
Add Sources

This guides you through the process

Add or create constraints [✓]
Add or create design sources
Add or create simulation sources

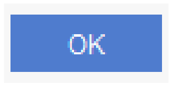


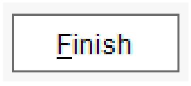


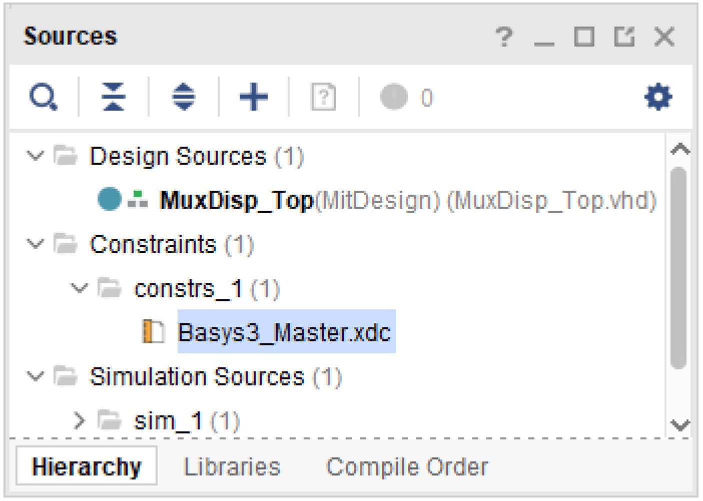

Add Constraint Files

Look in: Downloads

Basys3_Master.xdc
Basys-3-Master_v2.xdc

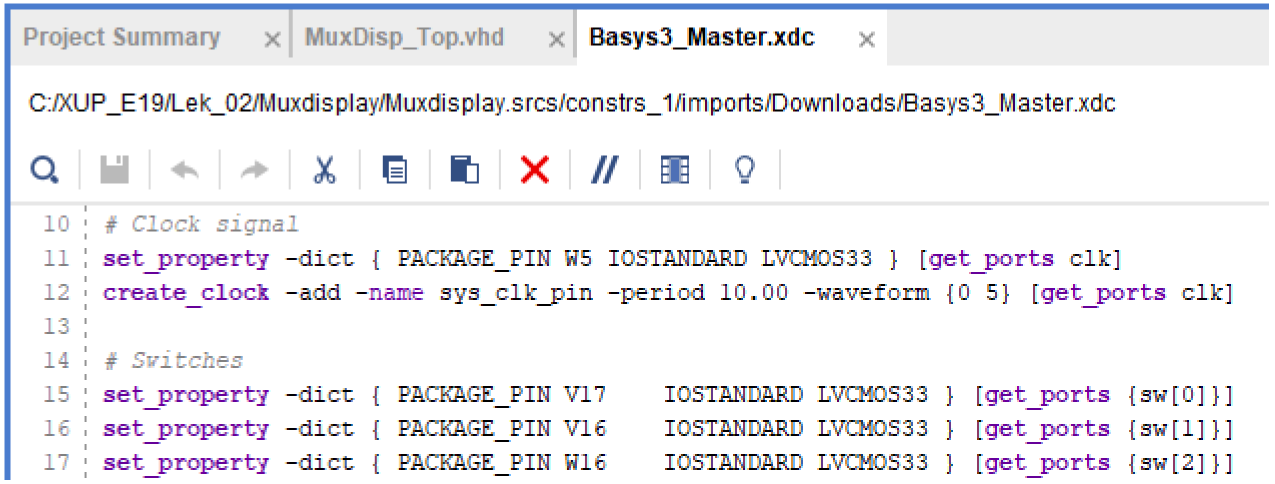





Sources

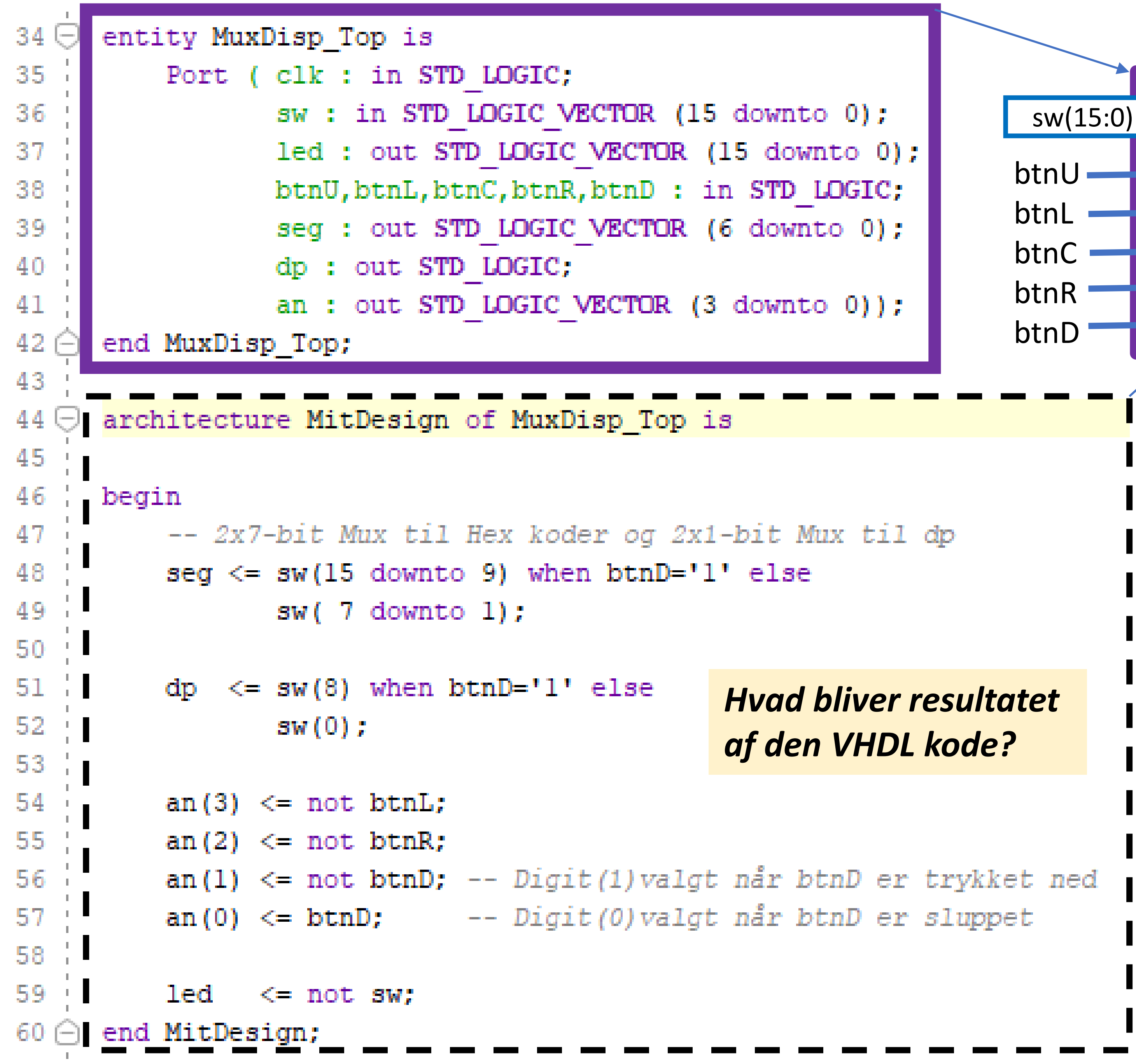
Design Sources (1)
MuxDisp_Top(MitDesign) (MuxDisp_Top.vhd)
Constraints (1)
constrs_1 (1)
Basys3_Master.xdc
Simulation Sources (1)
sim_1 (1)

Hierarchy Libraries Compile Order


Project Summary | MuxDisp_Top.vhd | Basys3_Master.xdc

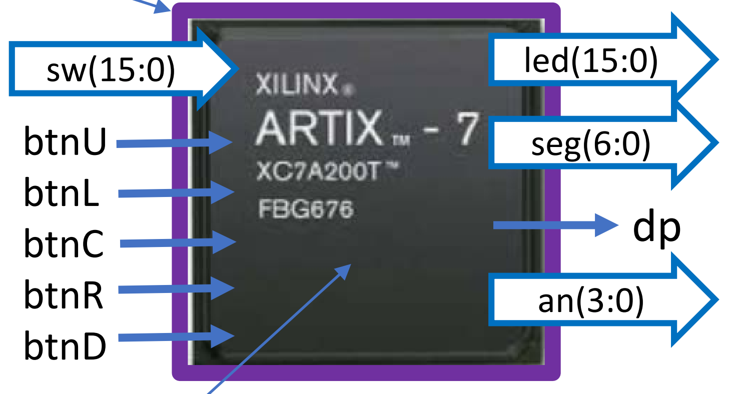
C:/XUP_E19/Lek_02/Muxdisplay/Muxdisplay.srcs/constrs_1/imports/Downloads/Basys3_Master.xdc

```
# Clock signal
set_property -dict { PACKAGE_PIN W5 IOSTANDARD LVCMOS33 } [get_ports clk]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

# Switches
set_property -dict { PACKAGE_PIN V17    IOSTANDARD LVCMOS33 } [get_ports {sw[0]}]
set_property -dict { PACKAGE_PIN V16    IOSTANDARD LVCMOS33 } [get_ports {sw[1]}]
set_property -dict { PACKAGE_PIN W16    IOSTANDARD LVCMOS33 } [get_ports {sw[2]}]
```



```vhdl
entity MuxDisp_Top is
    Port ( clk : in STD_LOGIC;
           sw : in STD_LOGIC_VECTOR (15 downto 0);
           led : out STD_LOGIC_VECTOR (15 downto 0);
           btnU,btnL,btnC,btnR,btnD : in STD_LOGIC;
           seg : out STD_LOGIC_VECTOR (6 downto 0);
           dp : out STD_LOGIC;
           an : out STD_LOGIC_VECTOR (3 downto 0));
end MuxDisp_Top;

architecture MitDesign of MuxDisp_Top is

begin
    -- 2x7-bit Mux til Hex koder og 2x1-bit Mux til dp
    seg <= sw(15 downto 9) when btnD='1' else
           sw( 7 downto 1);

    dp  <= sw(8) when btnD='1' else
           sw(0);

    an(3) <= not btnL;
    an(2) <= not btnR;
    an(1) <= not btnD; -- Digit(1)valgt når btnD er trykket ned
    an(0) <= btnD;     -- Digit(0)valgt når btnD er sluppet

    led   <= not sw;
end MitDesign;
```

Hvad bliver resultatet af den VHDL kode?



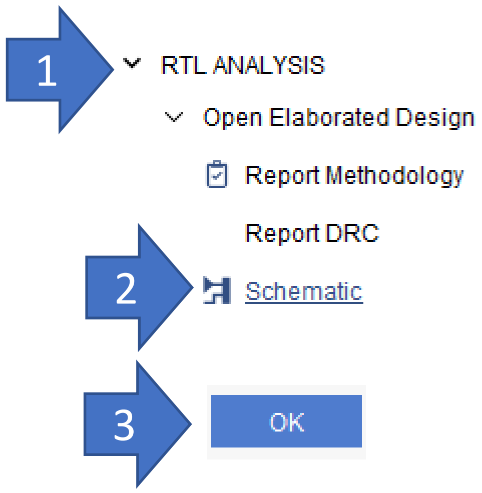
Vigtigt at forstå ang. VHDL

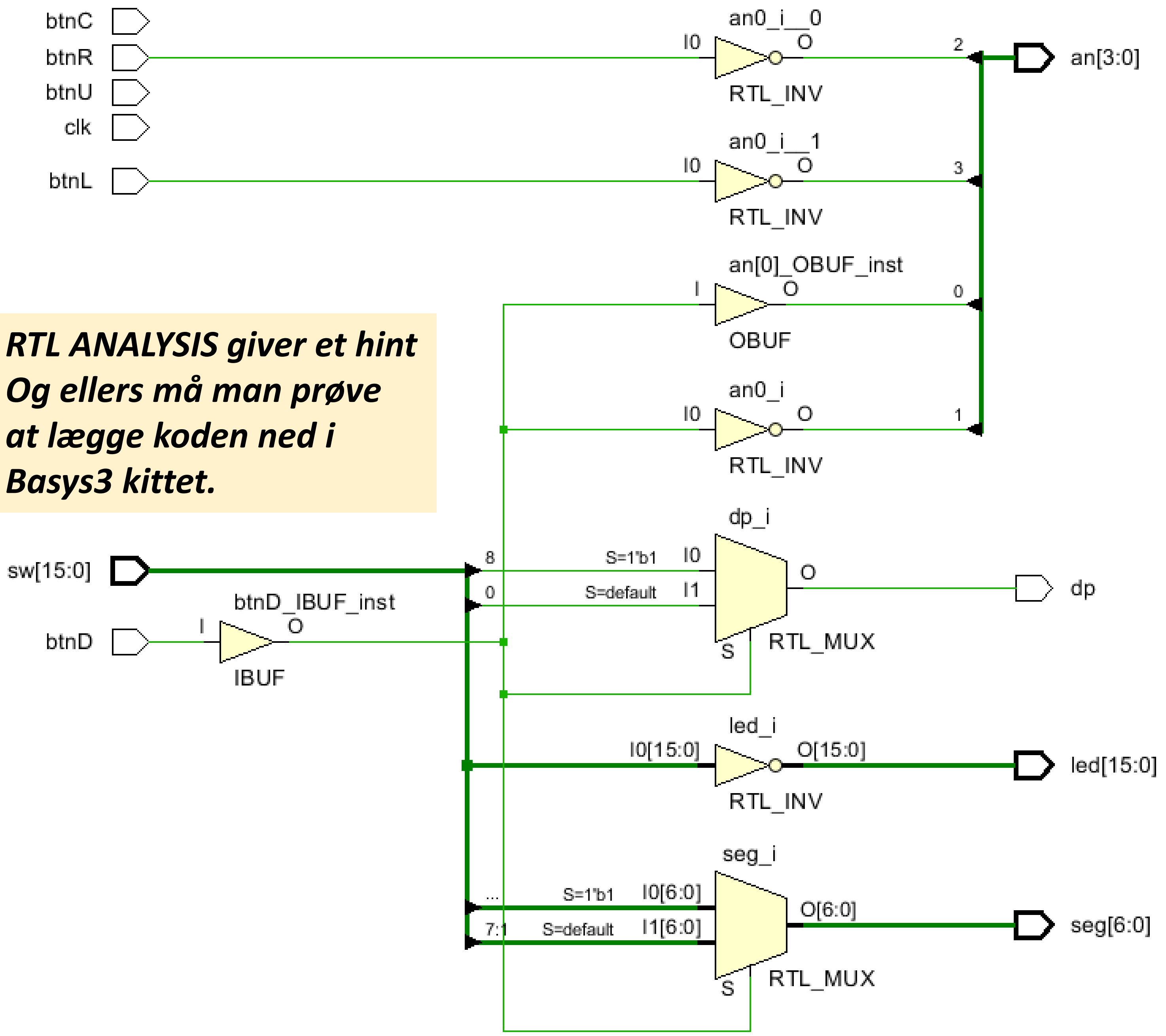
Entity beskriver grænsefladen til en komponent

Signaler kan enten være **in**, **out** eller **inout**

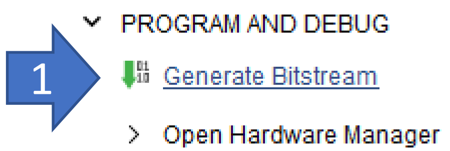
VECTOR angiver busser. **STD_LOGIC** bør anvendes i forbindelse med entity

Architecture beskriver funktionaliteten af komponenten.





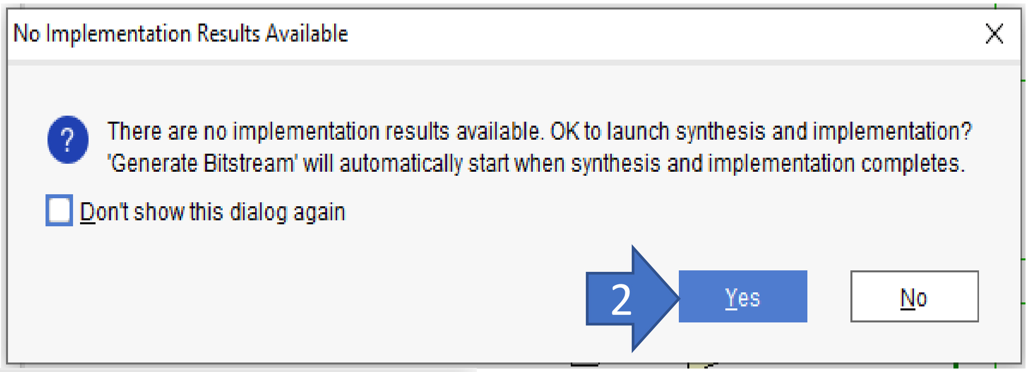
RTL ANALYSIS giver et hint
Og ellers må man prøve
at lægge koden ned i
Basys3 kittet.



PROGRAM AND DEBUG

Generate Bitstream

Open Hardware Manager

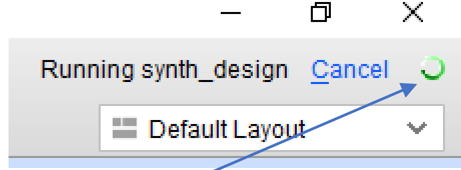


No Implementation Results Available

There are no implementation results available. OK to launch synthesis and implementation?
'Generate Bitstream' will automatically start when synthesis and implementation completes.

☐ Don't show this dialog again

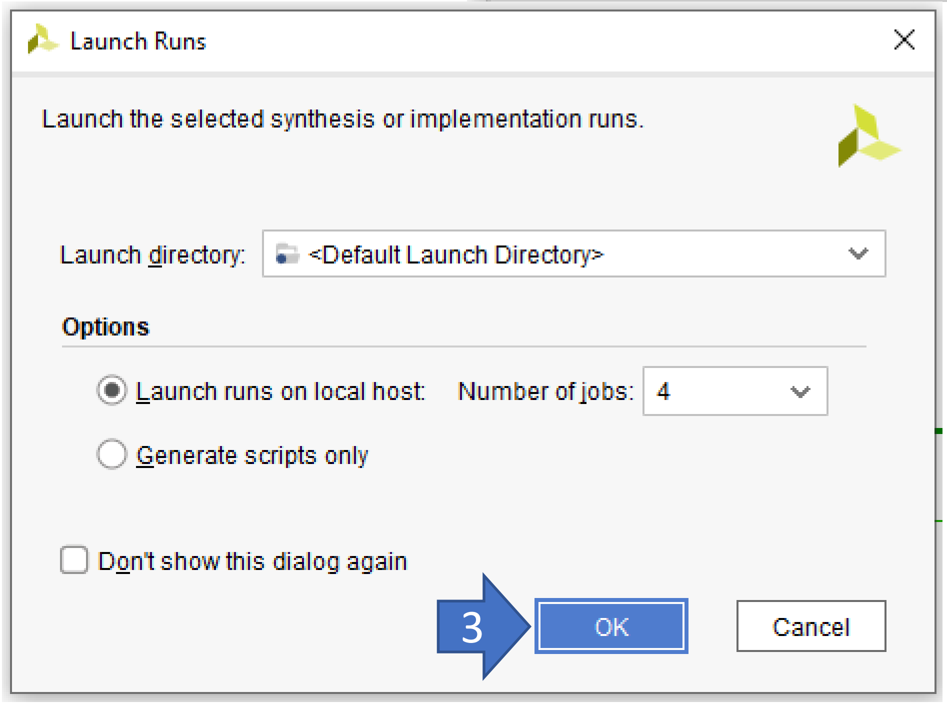
Yes   No



Running synth_design Cancel

Default Layout

Bemærk at det tager lidt tid …
Se øverst i hjørnet



Launch Runs

Launch the selected synthesis or implementation runs.

Launch directory: <Default Launch Directory>

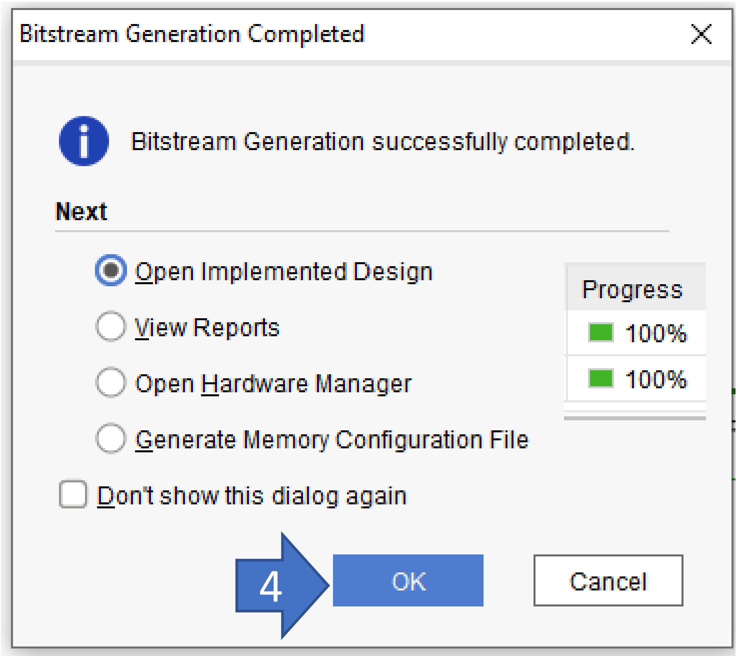
Options

● Launch runs on local host: Number of jobs: 4

○ Generate scripts only

☐ Don't show this dialog again

OK   Cancel



Bitstream Generation Completed

Bitstream Generation successfully completed.

Next

● Open Implemented Design

○ View Reports

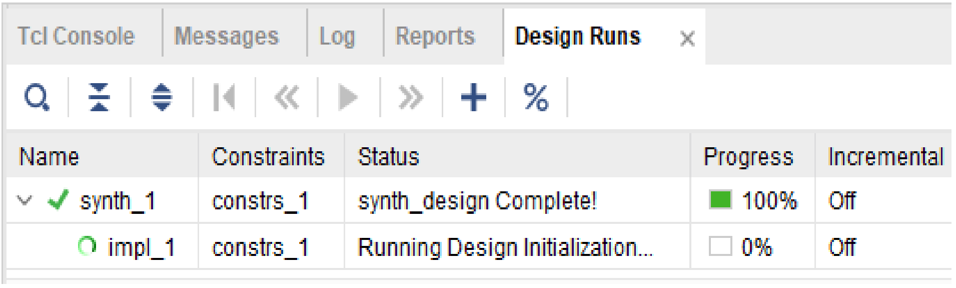
○ Open Hardware Manager

○ Generate Memory Configuration File

Progress
🟩 100%
🟩 100%

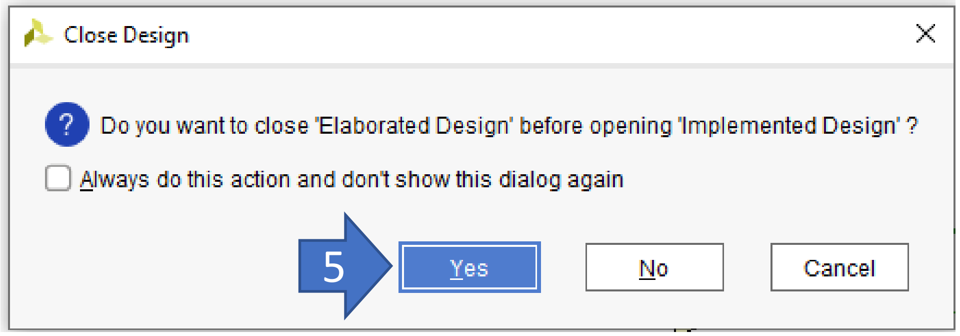
☐ Don't show this dialog again

OK   Cancel



Tcl Console | Messages | Log | Reports | Design Runs

Name	Constraints	Status	Progress	Incremental
✓ synth_1	constrs_1	synth_design Complete!	🟩 100%	Off
impl_1	constrs_1	Running Design Initialization...	☐ 0%	Off

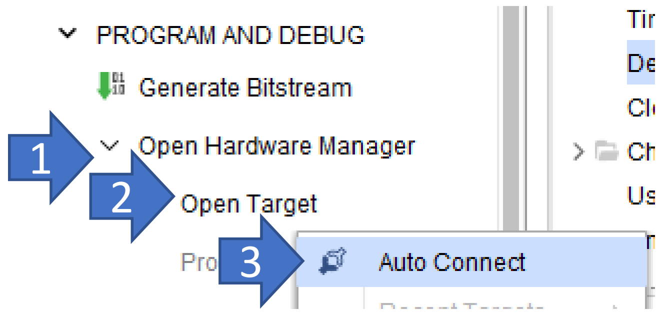


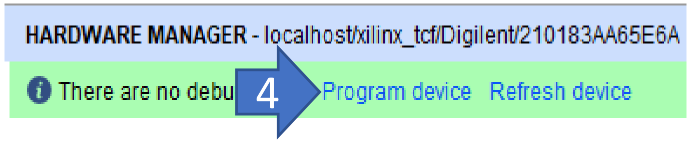
Close Design

Do you want to close 'Elaborated Design' before opening 'Implemented Design' ?

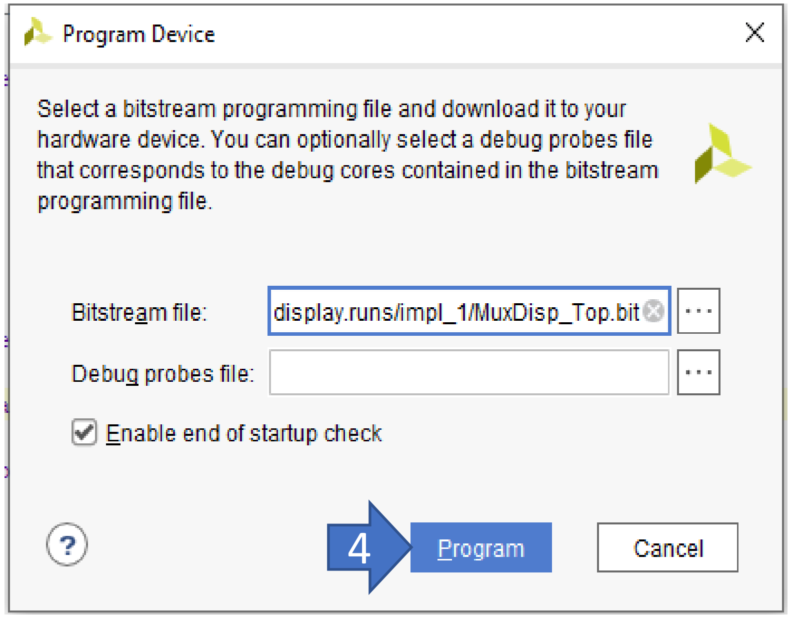
☐ Always do this action and don't show this dialog again

Yes   No   Cancel




HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AA65E6A

There are no debu


Program Device

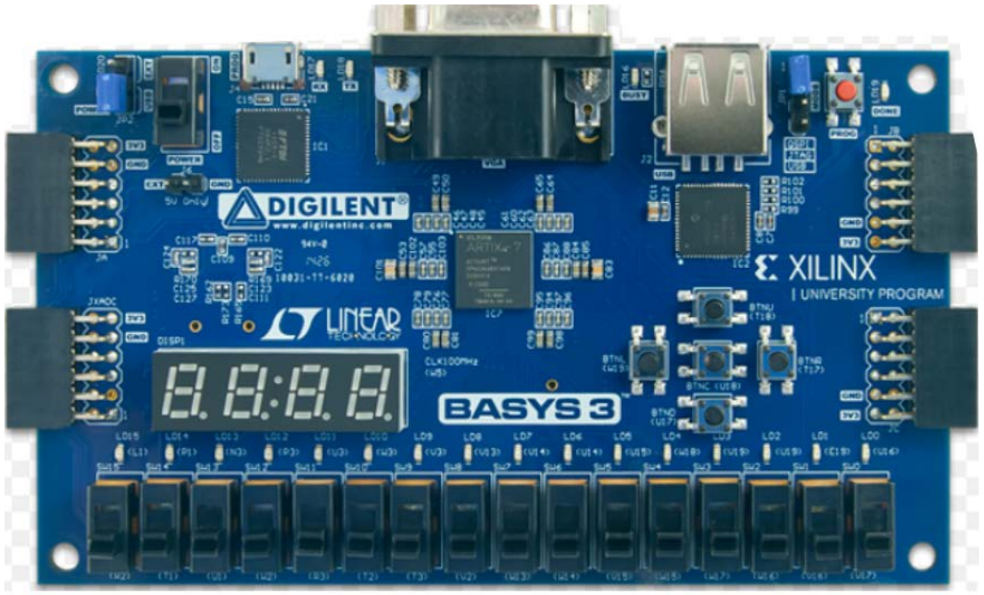
Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: display.runs/impl_1/MuxDisp_Top.bit

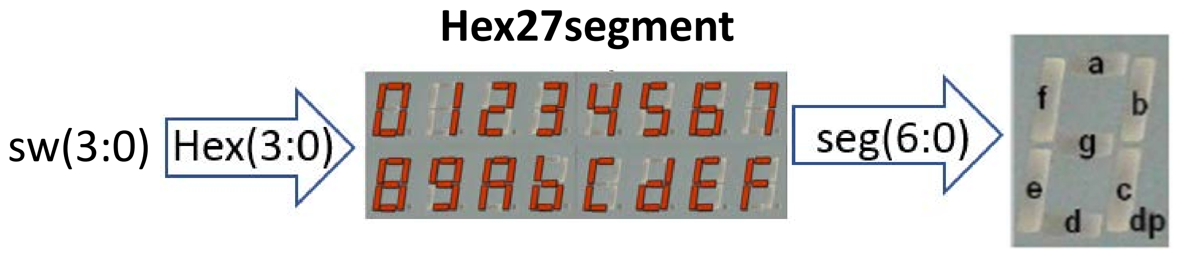
Debug probes file:

☑ Enable end of startup check

```vhdl
architecture MitDesign of MuxDisp_Top is

begin
    -- 2x7-bit Mux til Hex koder og 2x1-bit Mux til dp
    seg <= sw(15 downto 9) when btnD='1' else
           sw( 7 downto 1);

    dp  <= sw(8) when btnD='1' else
           sw(0);

    an(3) <= not btnL;
    an(2) <= not btnR;
    an(1) <= not btnD; -- Digit(1)valgt når btnD er trykket ned
    an(0) <= btnD;     -- Digit(0)valgt når btnD er sluppet

    led   <= not sw;
end MitDesign;
```



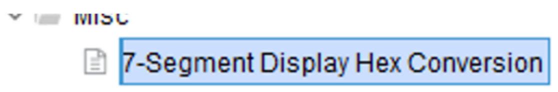
Muxdisplay - Opgave 2 – Lav en Hex27segment decocer komponent

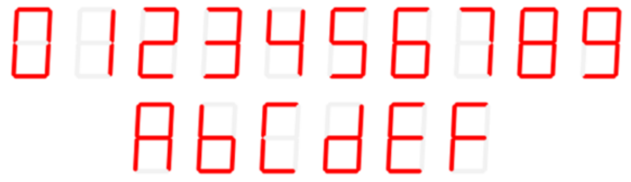


Hex27segment decoder

find inspiration under Misc eller i lærebogen

Listing 8.28 VHDL Description of the Seven-Segment Display Decoder Module



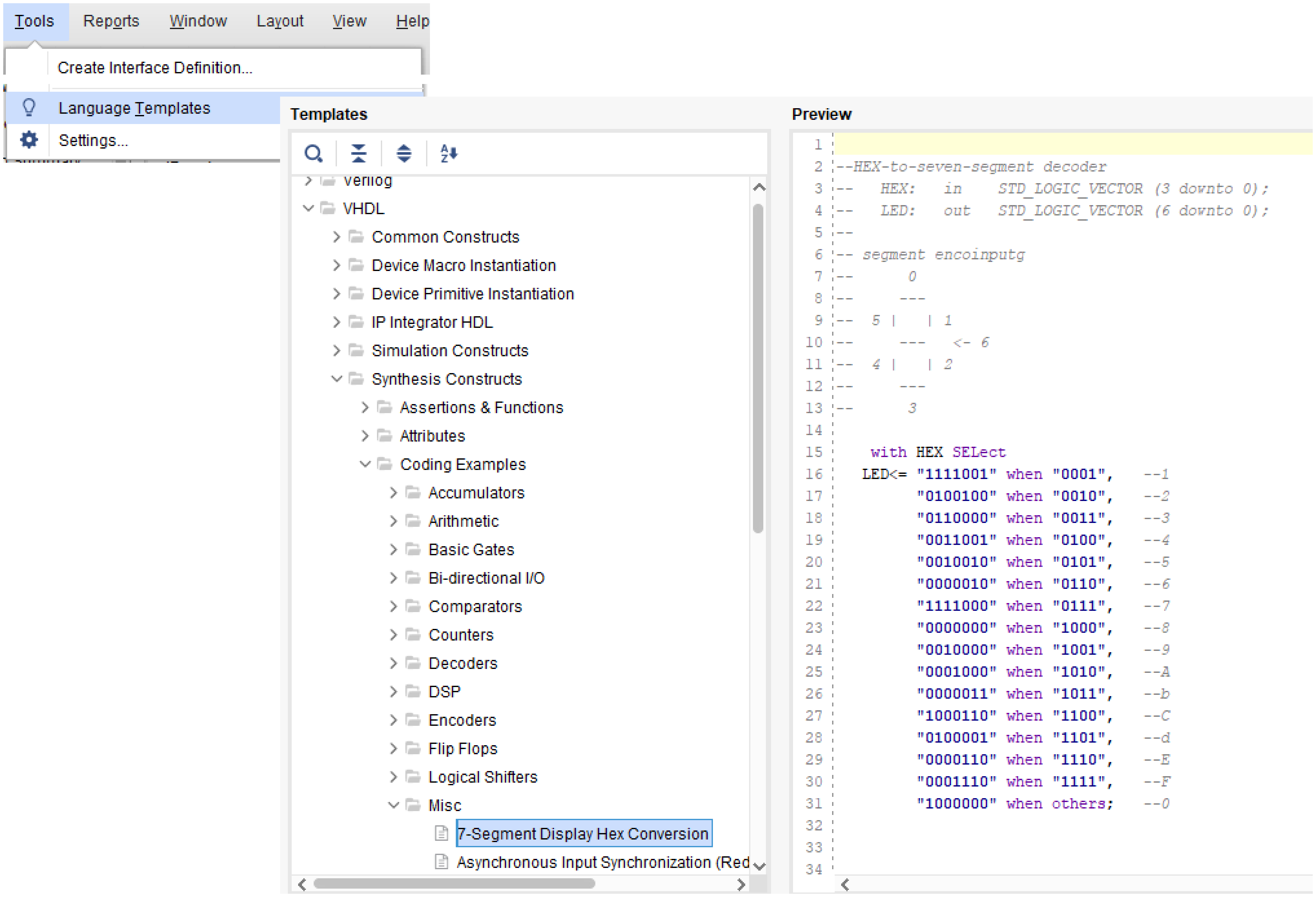


Begynd med en sandhedstabel og skriv VHDL kode

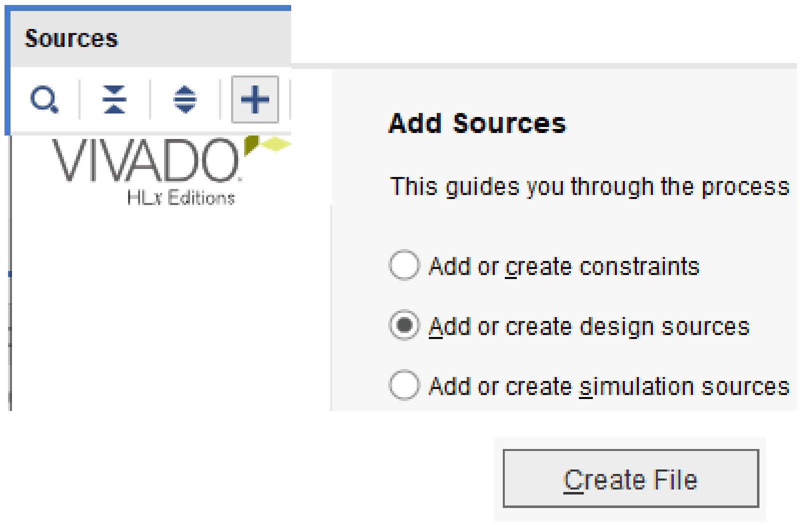
Eller find hjælp i language templates

seg(6:0)

Hex(3:0)	seg(0)	Segments					seg(6)
	a	b	c	d	e	f	g
0000	0	0	0	0	0	0	1
0001							
0010							
0011							
0100							
0101							
0110							
0111							
1000							
1001							
1010							
1011							
1100							
1101							
1110							
1111							



```

--HEX-to-seven-segment decoder
--   HEX:   in    STD_LOGIC_VECTOR (3 downto 0);
--   LED:   out   STD_LOGIC_VECTOR (6 downto 0);
--
-- segment encoinputg
--       0
--      ---
--  5 |     | 1
--      ---   <- 6
--  4 |     | 2
--      ---
--       3

   with HEX SELect
  LED<= "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



```



Sources

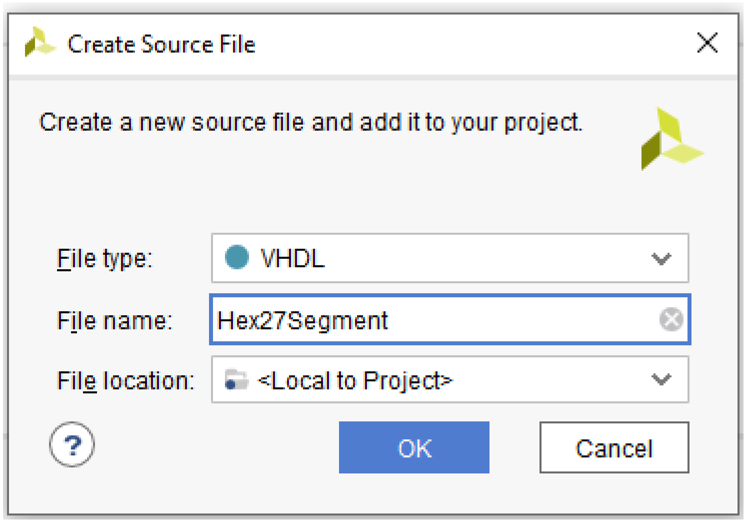
VIVADO HLx Editions

Add Sources

This guides you through the process

- ( ) Add or create constraints
- (●) Add or create design sources
- ( ) Add or create simulation sources

Create File

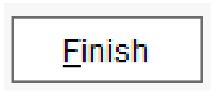


Create Source File

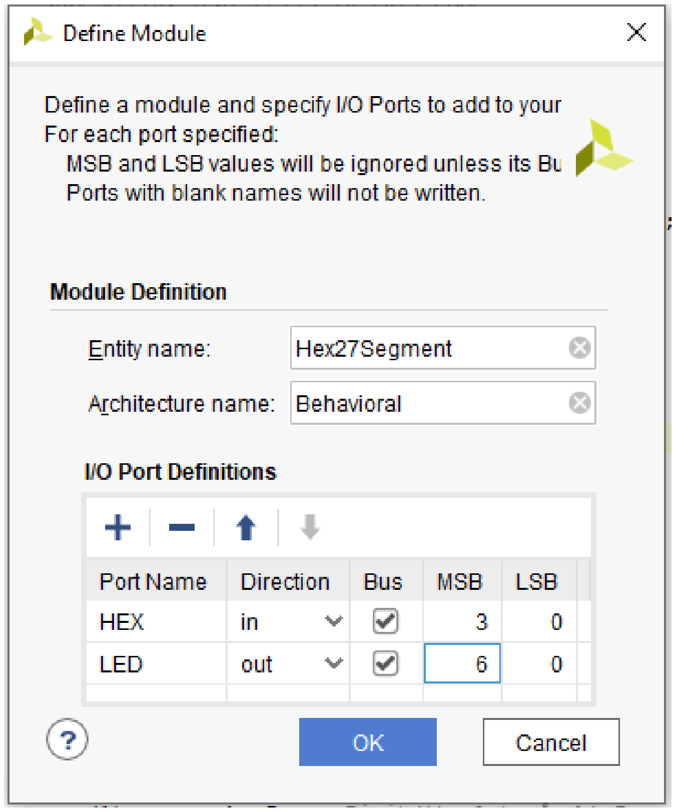
Create a new source file and add it to your project.

File type:	VHDL
File name:	Hex27Segment
File location:	<Local to Project>

OK Cancel



Finish



Define Module

Define a module and specify I/O Ports to add to your
For each port specified:
  MSB and LSB values will be ignored unless its Bu
  Ports with blank names will not be written.

Module Definition

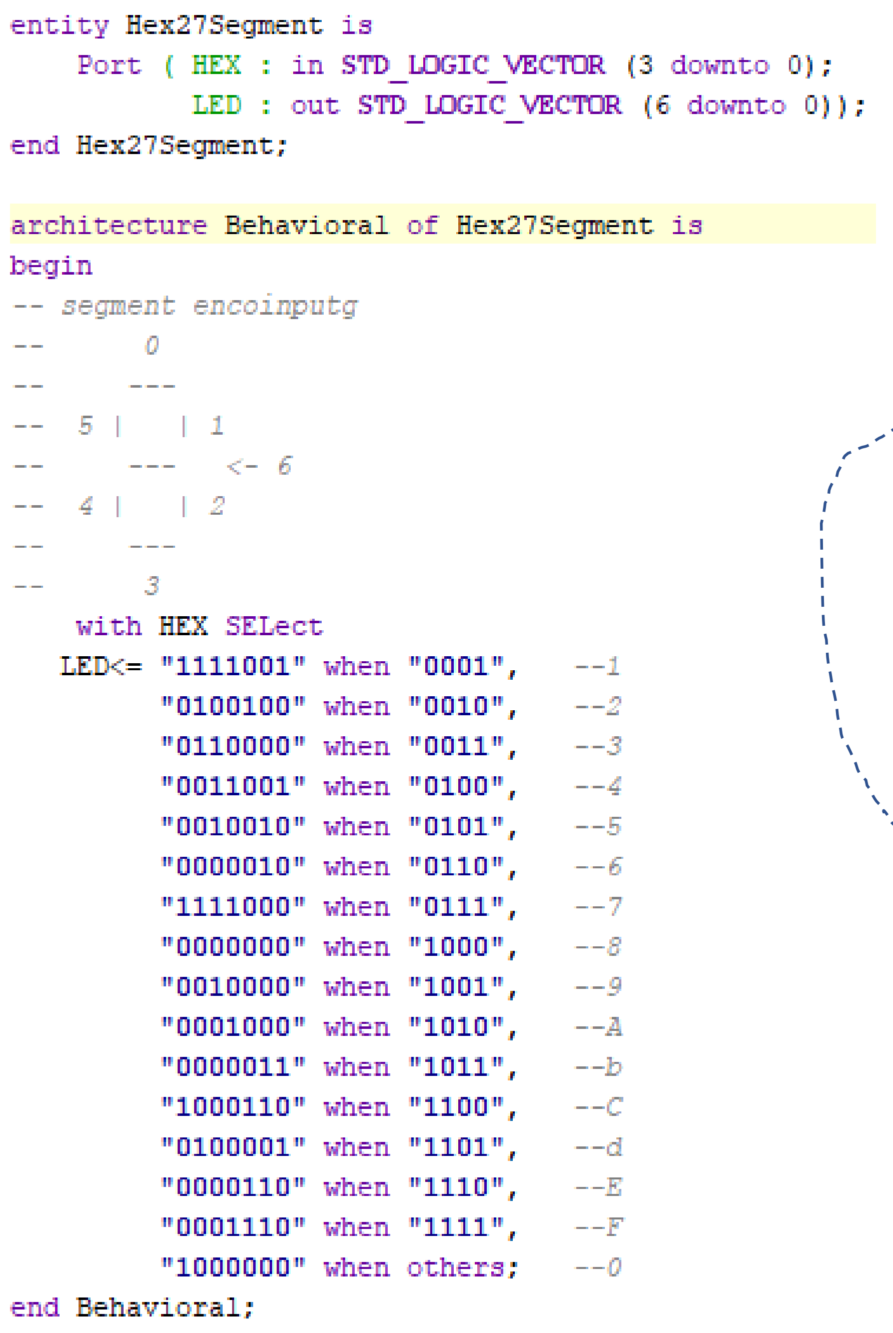
Entity name:	Hex27Segment
Architecture name:	Behavioral

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
HEX	in	✓	3	0
LED	out	✓	6	0

OK Cancel

```
entity Hex27Segment is
    Port ( HEX : in STD_LOGIC_VECTOR (3 downto 0);
           LED : out STD_LOGIC_VECTOR (6 downto 0));
end Hex27Segment;

architecture Behavioral of Hex27Segment is

begin


end Behavioral;
```

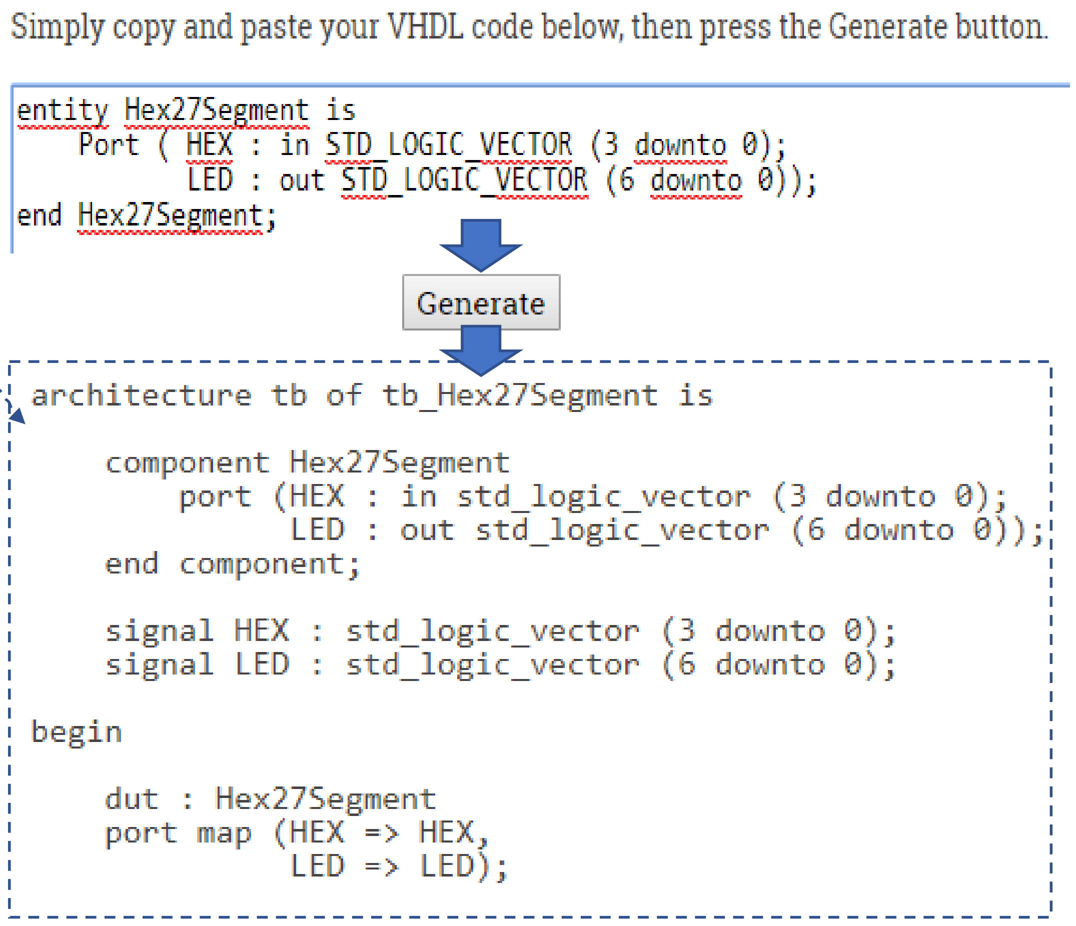


```
entity Hex27Segment is
    Port ( HEX : in STD_LOGIC_VECTOR (3 downto 0);
           LED : out STD_LOGIC_VECTOR (6 downto 0));
end Hex27Segment;

architecture Behavioral of Hex27Segment is
begin
-- segment encoinputg
--      0
--     ---
--  5 |   | 1
--     ---   <- 6
--  4 |   | 2
--     ---
--      3
    with HEX SELect
   LED<= "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;
```

https://vhdl.lapinoo.net/testbench/

Hvordan bruger man en VHDL-komponent i et nyt design?
Det er smart at bruge ovenstående link – selvom det egentligt er beregnet til at lave en TestBench template

Start med at kopiere entity Hex27Segment til hjemmesiden
Tryk => Generate
-På den måde får man en **component** og en **dut**:



Simply copy and paste your VHDL code below, then press the Generate button.

```
entity Hex27Segment is
    Port ( HEX : in STD_LOGIC_VECTOR (3 downto 0);
           LED : out STD_LOGIC_VECTOR (6 downto 0));
end Hex27Segment;
```

```
architecture tb of tb_Hex27Segment is

    component Hex27Segment
        port (HEX : in std_logic_vector (3 downto 0);
              LED : out std_logic_vector (6 downto 0));
    end component;

    signal HEX : std_logic_vector (3 downto 0);
    signal LED : std_logic_vector (6 downto 0);

begin

    dut : Hex27Segment
    port map (HEX => HEX,
              LED => LED);
    led   <= not sw;
end MitDesign;
```

kopi

```
--##################################################################
architecture MitDesign2 of MuxDisp_Top is
    component Hex27Segment
        port (HEX : in std_logic_vector (3 downto 0);
              LED : out std_logic_vector (6 downto 0));
    end component;

    signal xHEX : std_logic_vector (3 downto 0);
--    signal LED : std_logic_vector (6 downto 0);
begin
    dut : Hex27Segment
    port map (HEX => xHEX,
              LED => seg);

    xHEX <= sw( 7 downto 4) when btnD='1' else
            sw( 3 downto 0);
    dp  <= sw(9) when btnD='1' else
           sw(8);
    an(3) <= not btnL;
    an(2) <= not btnR;
    an(1) <= not btnD; -- Digit(1)valgt når btnD er trykket ned
    an(0) <= btnD;     -- Digit(0)valgt når btnD er sluppet

    led   <=  sw;
end MitDesign2;
```

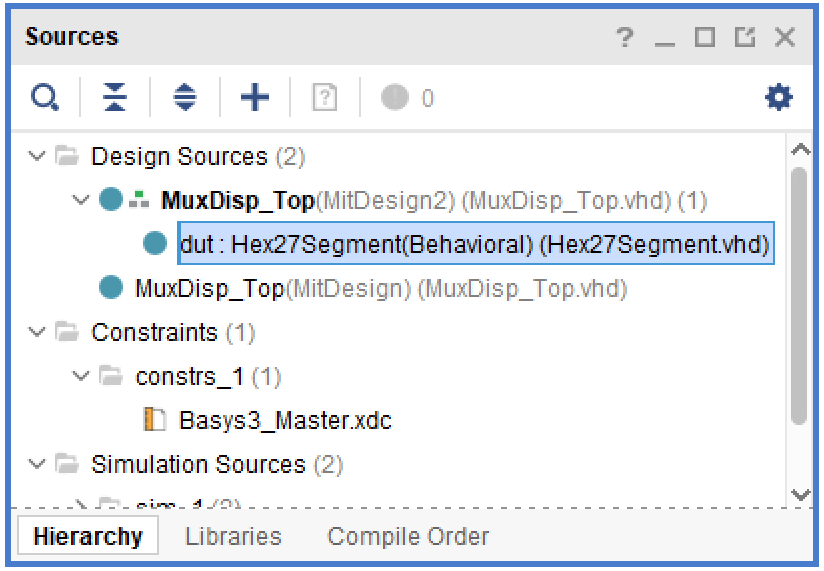
Bemærk! – man kan have flere architecture til samme entity – og det er den architecture som ligger nederst der bruges. Det er nødvendigt at lave lidt justeringer af koden

```
component Hex27Segment
    port (HEX : in std_logic_vector (3 downto 0);
          LED : out std_logic_vector (6 downto 0));
end component;
```

```
signal HEX : std_logic_vector (3 downto 0);
signal LED : std_logic_vector (6 downto 0);
```

```
begin
dut : Hex27Segment
port map (HEX => HEX,
          LED => LED);
```

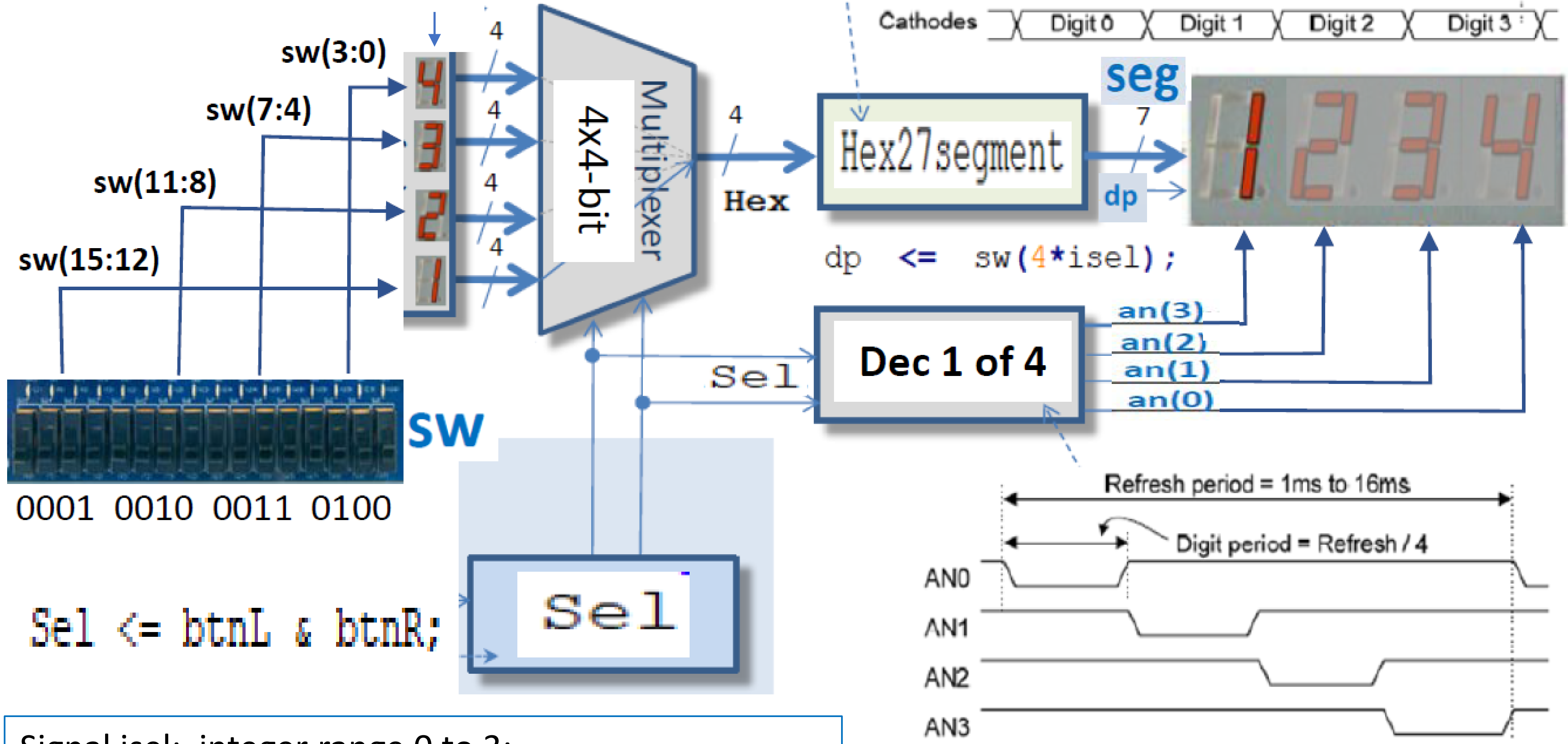
Bemærk! Nu er Hex27Segment en del af MitDesign2



Design Sources (2)
MuxDisp_Top(MitDesign2) (MuxDisp_Top.vhd) (1)
dut : Hex27Segment(Behavioral) (Hex27Segment.vhd)
MuxDisp_Top(MitDesign) (MuxDisp_Top.vhd)
Constraints (1)
constrs_1 (1)
Basys3_Master.xdc
Simulation Sources (2)

Muxdisplay - Opgave 3 – Mere advanceret display-styring

```
COMPONENT Hex27segment
   Port (Hex: in  STD_LOGIC_VECTOR (3 downto 0);
         Seg: out STD_LOGIC_VECTOR (6 downto 0));
end COMPONENT;
```



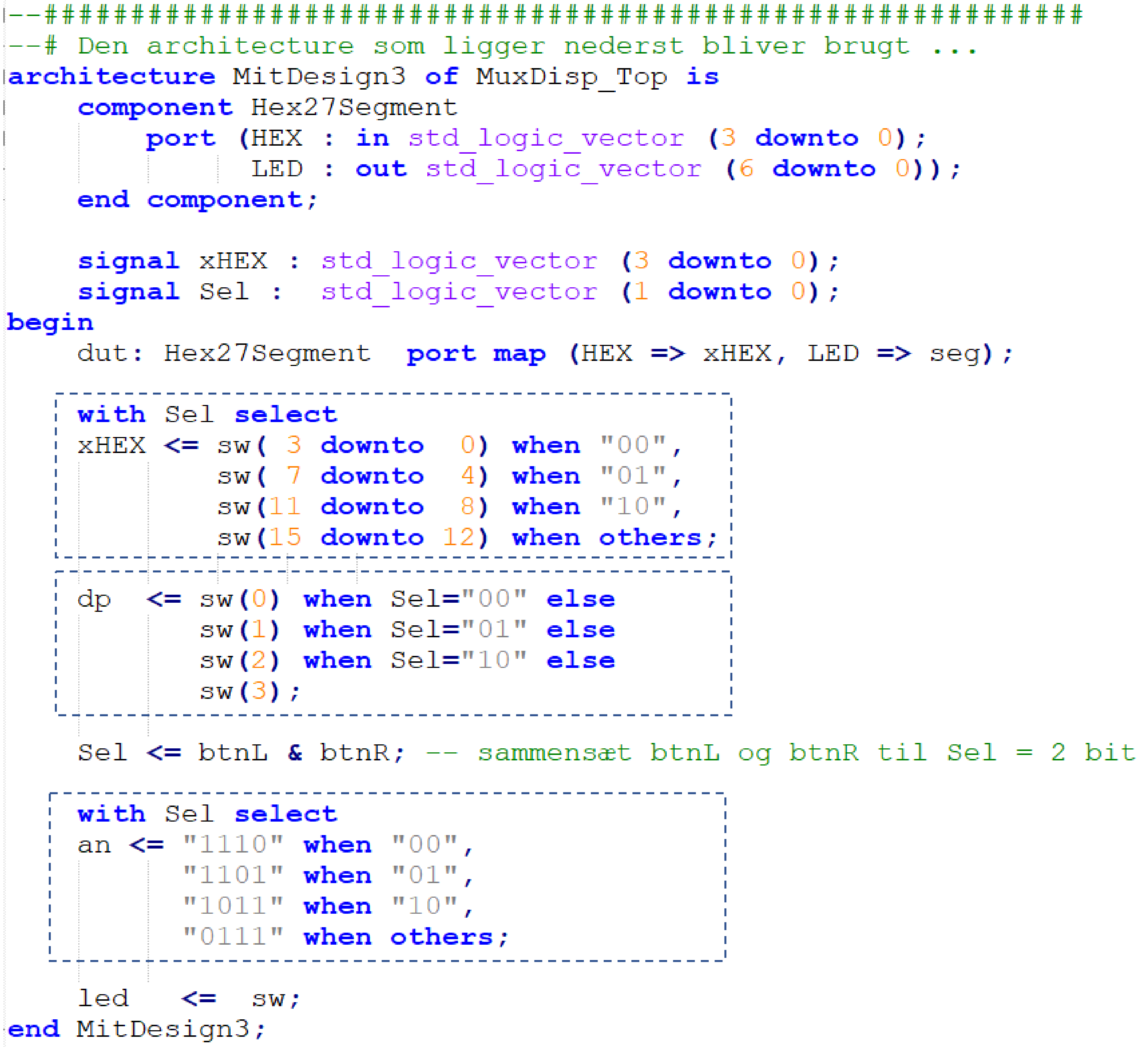
```
dp  <=  sw(4*isel);
```

```
Sel <= btnL & btnR;
```

Signal isel: integer range 0 to 3;
isel <= conv_integer(Sel);
dp <= sw(4*isel);   -- alterternativ mux til dp signal

Løsning på næste side

Løsningsforslag - men bare for eksemplet skal vi prøve at dele dette design op på flere komponenter.



```vhdl
--#############################################################
--# Den architecture som ligger nederst bliver brugt ...
architecture MitDesign3 of MuxDisp_Top is
    component Hex27Segment
        port (HEX : in std_logic_vector (3 downto 0);
              LED : out std_logic_vector (6 downto 0));
    end component;

    signal xHEX : std_logic_vector (3 downto 0);
    signal Sel :  std_logic_vector (1 downto 0);
begin
    dut: Hex27Segment  port map (HEX => xHEX, LED => seg);

    with Sel select
    xHEX <= sw( 3 downto  0) when "00",
            sw( 7 downto  4) when "01",
            sw(11 downto  8) when "10",
            sw(15 downto 12) when others;

    dp  <= sw(0) when Sel="00" else
           sw(1) when Sel="01" else
           sw(2) when Sel="10" else
           sw(3);

    Sel <= btnL & btnR; -- sammensæt btnL og btnR til Sel = 2 bit

    with Sel select
    an <= "1110" when "00",
          "1101" when "01",
          "1011" when "10",
          "0111" when others;

    led    <=  sw;
end MitDesign3;
```