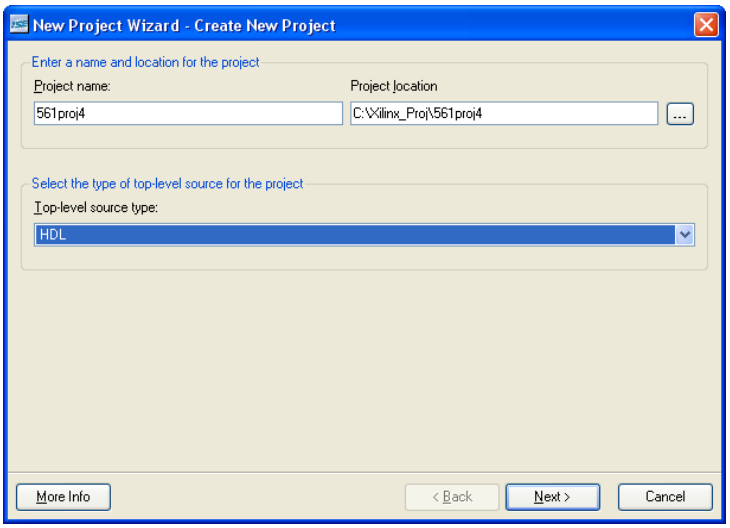
Using *Xilinx ISE* for VHDL Based Design

In this project you will learn to create a design module from VHDL code. With *Xilinx ISE*, you can easily create modules from VHDL code using the *ISE* Text Editor tool. This project consists of 2 parts.

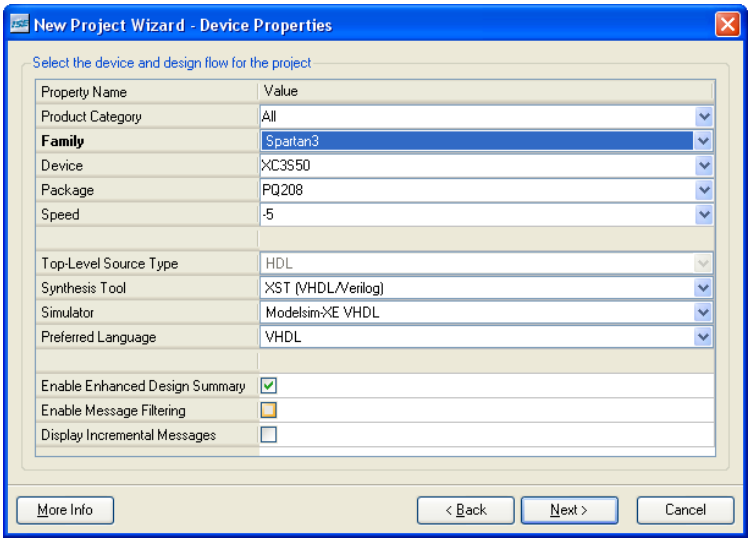
PART I : Using Xilinx with VHDL as design entry.

A. Create new project

1. Select File → New Project. A *New Project Wizard* window opens; you can name your project (i.e. 561proj4). Select *HDL* as the *Top-level source type*. Click *Next*.



2. Select *Spatan3* in the *Family* field; and *VHDL* in the *Preferred Language* field.



3. Click *Next, Next, Finish.* You do not need to create a new source at this time.

B. Start with you VHDL code

In order to create the module, in the New Source Wizard, create a file and specify the name. The resulting "skeleton" VHDL file is then modified further in the *ISE* Text Editor.

To create the source file:

1. Select Project → New Source. A dialog box opens in which you specify the type of source you want to create.

2. Select *VHDL Module*.

3. In the *File Name* field, type '***cnt1_fsm***'.

4. Click *Next*. You do not need to create ports of the component at this time.

5. Copy and pasted your VHDL code in to the schematic editor '***cnt1_fsm.vhd***'. Make sure you properly replaced *ENTITY* and *ARCHITECTURE* blocks.

Here I used the VHDL of the input 1s counter mod 4 from the lectures as an example.
The VHDL was:

```
ENTITY cnt1s_fsm IS
   PORT (x,y : IN BIT;
         clk : IN BIT;
         fsm_out : OUT BIT);
END cnt1s_fsm;

ARCHITECTURE one OF cnt1s_fsm IS
   TYPE state_type IS (s0,s1,s2,s3);
   SIGNAL state, next_state : state_type;
BEGIN
   PROCESS
   BEGIN
      WAIT UNTIL clk = '1' AND clk'event;
      state <= next_state;
   END PROCESS;

   PROCESS (state,x,y)
   BEGIN
      CASE state IS
         WHEN s0 => IF (x='1' AND y='1') THEN next_state <= s2;
                    ELSIF (x/=y) THEN next_state <= s1;
                    END IF;
         WHEN s1 => IF (x='1' AND y='1') THEN next_state <= s3;
                    ELSIF (x/=y) THEN next_state <= s2;
                    END IF;
         WHEN s2 => IF (x='1' AND y='1') THEN next_state <= s0;
```

```
                ELSIF (x/=y) THEN next_state <= s3;
                END IF;
        WHEN s3 => IF (x='1' AND y='1') THEN next_state <= s1;
                ELSIF (x/=y) THEN next_state <= s0;
                END IF;
    END CASE;
  END PROCESS;

  PROCESS (state)
  BEGIN
    CASE state IS
      WHEN s0 => fsm_out <= '1';
      WHEN s1 => fsm_out <= '0';
      WHEN s2 => fsm_out <= '0';
      WHEN s3 => fsm_out <= '0';
    END CASE;
  END PROCESS;

END one;
```

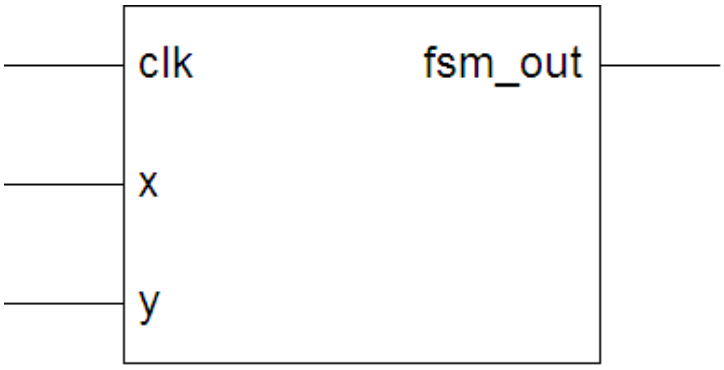
C. Synthesize your design

1. In the Sources window, select ***cnt1_fsm.vhd***. In the Processes window, you can then double click on the *Synthesize – XST* selection. This synthesizes the design from that file.

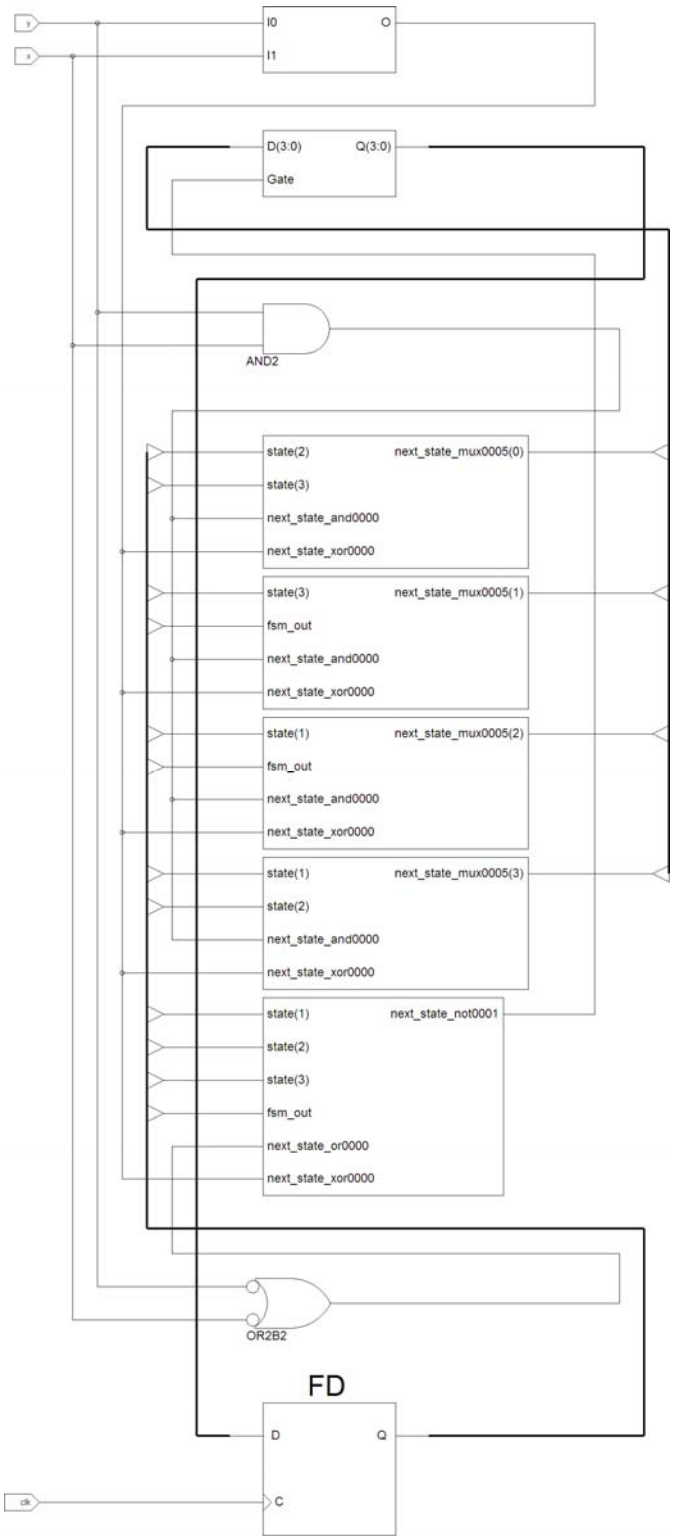
You may get some errors and if you do check the HDL code that it is compiling especially the entity and port modes.

2. Then after the synthesis is correct go to the Processes window (lower of the 2 windows on the left). Expand the *Synthesize – XST* selection by clicking on the + in the box before it. It will expand. Double click on the *View RTL Schematic.*

You should see comparable to the following

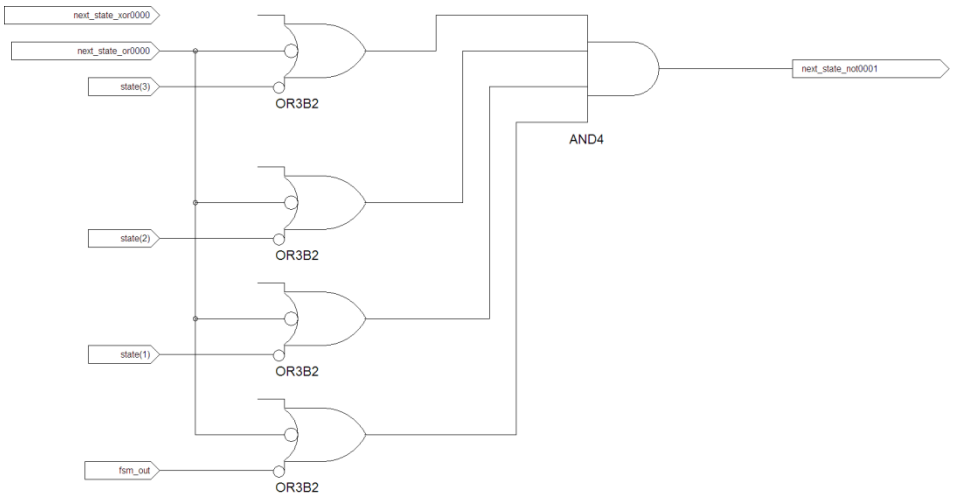


3. Now double click on this figure to expand what is in it and you should see the following:



You will notice there are 2 flip-flops, a couple of gates, and a few other units.

4. Double click on one of the larger boxes. You will see the following:



You can now implement the design by double clicking on that line. Further explore the remaining options.

5. The project that you write the VHDL for is

Write a VHDL description of a tail light controller. This will be very similar in operation to the T-Bird Taillight controller we discussed earlier. Now however, there are 4 segments in each taillight, LD,LC,LB,LA and RA,RB,RC,RD.

Write up the VHDL code for this. Enter it into XILINX. Synthesize it and capture the circuit generated after synthesis. You can select, then copy, and then paste into a word document for you report.

Turn in a report that has the VHDL code and the generated circuit.

Part II. Repeat of 7 Segment Decoder

Write a VHDL entity and architecture for a 7 segment decoder. As before, the unit will have a 4 bit input. For the style of code to be used, it is easiest if the data input is declared in the port list as: D : IN BIT_VECTOR(3 downto 0); which says that the input is named D with is a 4 bit vector of bits. You can use all 4 bits by just specifying D. Values of D would be “0000”, “0001”, …, “1110”,”1111”.

In the ARCHITECTURE you will have 1 process and in that process, with will be sensitive to D, a single CASE statement: CASE D IS. The choices will be WHEN “0000” => s0<=’1’;s1<=’1’; etc. for the 7 s outputs. Enter the design and then synthesize it. As before add to report your code and the results of the synthesis.

<u>Also, comment on how the unit was implemented in the FPGA.</u>