

Three Mode Design Flow Lab

Three Mode Design Flow Lab

Introduction

This lab uses the WATCHNEW project to demonstrate how a project can be designed using multiple design entry techniques of schematic, state diagram editor, and FPGA Express VHDL synthesis. Foundation Series Express uses **XNF** (Xilinx Netlist Format) and **EDIF** files as the default netlist formats. These netlists are optimized and implemented in the M1 software.

Objective

After completing this lab, students will be able to:

- Implement and optimize a design with the Foundation Series Express software
- Use the State Diagram Editor outputs and synthesize the resulting VHDL using Express.
- Use LogiBLOX to create a custom module.
- Combine the design entry components into one schematic top level design.

About the WATCH project

The WATCH project is designed to be a track coach's stopwatch. There are two inputs to the system (**RESET** and **STRTSTOP**). The configuration clock on the FPGA is used as a ten-hertz clock signal. However, since the internal XC4000 oscillator port, F15, is usually closer to 15 Hz rather than 10 Hz, the actual stopwatch may run slightly fast. Three seven-bit outputs are generated by this system for output to three seven-segment displays.

Procedure

Opening a partially created project in Foundation Series Express

- 1) Start Foundation Series Express by clicking on **Start → Programs → Xilinx Foundation Series → Xilinx Foundation Project Manager**
- 2) Select File → Open Project..., browse to **c:\F15_labs**, select **watchnew** and click **Open**.
- 3) Click once to open the **Schematic Editor**.
You will notice a partially completed schematic. We are going to develop each of the blocks so that we finish with a design that looks like **Figure 1**, below:

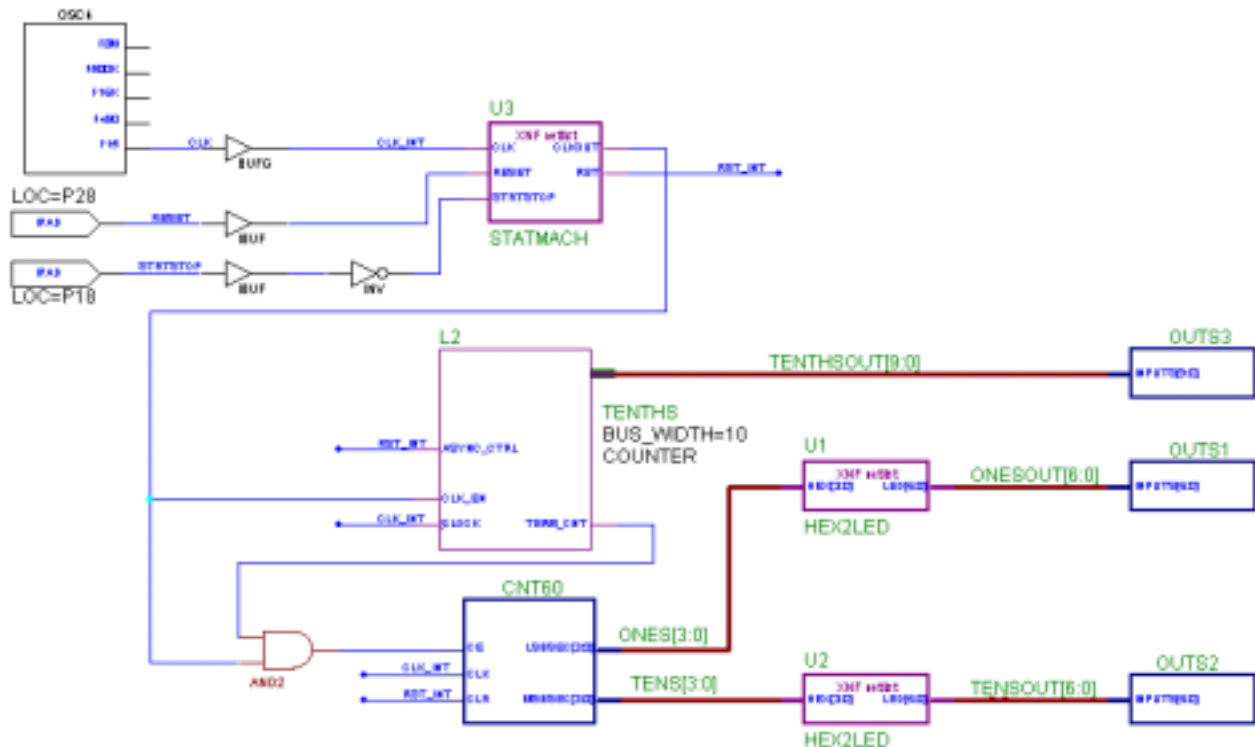




Figure 1

Creating a custom schematic macro using Express and VHDL synthesis

First, we will convert the 8-bit binary outputs from the CNT60 macro into two 7-segment LED signals using VHDL code which we will instantiate into the schematic. This VHDL code has already been written.

- 4) Open FPGA Express by clicking on **Start → Programs → Xilinx Foundation Series → Accessories → Foundation Express**. 
- 5) Once inside Foundation Express, click on **File → New...**
- 6) Go to the **c:\F15_labs\watchnew** directory. Enter the name of the project as **WATCHHEX** and click on the **Create** button. When asked to identify sources, click on the **Cancel** button.
- 7) Once the **Foundation Express Project Window** opens, click on the **WATCHHEX** project name in the **Design Sources** window. Then click on the command: **Synthesis → Identify Sources...** Go to the **c:\F15_labs\watchnew** directory, select the **HEX2LED.VHD** file, and click on the **Open** button. Express will check the file for errors, and should not find any.
- 8) To view this code, you can highlight **hex2led.vhd** under the **Design Sources** window, right-click on it, and select **Edit File**. When done viewing, close the VHDL text editor.

- 9) In the Design Sources window, double-click on **hex21ed.vhd**, and the entity **hex21ed** will be expanded below. Highlight **hex21ed**. Select **Synthesis → Create Implementation**. (Alternatively, you can click the Create Implementation icon .)
- 10) Since we are not creating a top-level VHDL design, eg. this will be a **module** within our schematic, check this box  Do not insert I/O pads so that module ports, and not chip-level I/O pads, are created. Also, make sure that the target device, **XC4005XLPC84C -3** is selected for proper optimization. Click <OK>.
- 11) Highlight **hex2led** under the Chips window, and select **Synthesis → Optimize**.
- 12) Highlight **hex2led-Optimized** and select **Synthesis → Export Netlist**. Accept the directory to Save in as **C:\F15_labs\watchnew**, and click <Save>. This creates two files.
- **hex21ed.xnf**, the Xilinx Netlist Format file containing module netlists
 - **hex21ed.xsf**, the Xilinx Symbol File used next to create a graphical symbol.
- 13) Return to the **Schematic Editor** (aka. Schematic Capture) application.
- 14) Now we will create a macro symbol from the VHDL code that we just synthesized. Select **Hierarchy → Create Macro Symbol from Netlist**. The directory location, **c:\F15_labs\watchnew**, should be correct.
- 15) Since we are looking for an XNF file, select **Files of type** as **Xilinx (*.XN*, *.BAX)**, and then select **hex21ed.xnf**. Click <Open>. Schematic and black box simulation files are created, and HEX2LED is added to the project library.
- 16) To prove this, from **Schematic Editor**, select **File → Project Libraries**, and click **Lib Manager**. Look for the current project name, **WATCHNEW**, on the left. This is shown as a **User**, rather than a **System** library. Highlight **WATCHNEW** (making sure you have the right one according to the path shown) and then click on the **Objects** tab. There you should notice that **HEX2LED** is listed as an object within this project's custom macro library.
- 17) Exit out of the **Library Manager**, exit from the **Project Libraries** box, and return to the **Schematic Editor**.
- 18) Open the **Symbols toolbox**, find the **HEX2LED** macro, and add two instances of the macro to the schematic as shown in the completed schematic diagram in **Figure 1**. Connect the busses directly using the **Draw busses** icon.

Creating a custom schematic macro using State Editor and Express

Next, we will create the state machine block using the **State Diagram Editor** in **Foundation Series**. The **State Diagram Editor** allows you to create a graphical state diagram, then automatically generate the corresponding VHDL or ABEL code, which can then be synthesized and used as the contents of a macro.

In Foundation Series Express 1.5, the flow is completely integrated when using the ABEL synthesis option. For VHDL and Verilog synthesis we will use the Synopsys-based **FPGA Express** tool to complete the flow.

- 19) Go to the **Foundation Project Manager** window.
- 20) Click on the **State Editor** icon.
- 21) To save time in this lab, we will use a state diagram that has already been created. Select **Open Existing document** and **<OK>**.
- 22) Select **Statmach.asf** (under **C:\F15_labs\watchnew** directory) and click **<Open>**. You will then see the state machine diagram. Notice that the state actions include VHDL style assignments, such as `clkout <= '0';`
- 23) Select **Synthesis → Configuration...** and notice that the language option is VHDL, and the tools is XVHDL. Select **<Cancel>**.
- 24) Select **Synthesis → Options** and check to see that this is setup to create a Macro rather than a Chip level output. It should be correct, so select **<Cancel>**.
- 25) Select **Synthesis → HDL Code Generation**.
- 26) That was quick! Select **<Yes>** to view the generated code.
- 27) Close the **HDL Editor** and move to the already opened **Foundation Express Project Manager**.
- 28) Create a new project to synthesize the recently generated VHDL code: **File → New...**
- 29) Select directory of **c:\F15_labs\watchnew** and type project of **watchst**, click **<Create>**.
- 30) Under the **Identify Sources** window, select **Statmach.vhd** **<Open>**.
- 31) Under the Design Sources window, highlight **Statmach.vhd**. You will see several errors that FPGA Express found in its initial VHDL code analysis. You will find errors in the State Diagram code to be consistent, and therefore easy to fix.

32) Let's fix the errors manually now. The errors relate to problems in the library declarations. We will also improve synthesis performance by making some other optional code modifications.

With **Statmach.vhd** highlighted, right-click, and select Edit File.

There are two ways to correct the errors. The easy way is to simply copy over the pre-corrected file from **C:\F15_labs\shortcut\Statmach.vhd** to

C:\F15_labs\watchnew\Statmach.vhd.

The harder way, which will teach you something, is to correct your existing code and re-save the resulting file. Let's try the harder option, but if you prefer, you may resort to the easier option.

27) Change this code

```
-- SYNOPSIS library declaration
library SYNOPSIS;
use SYNOPSIS.std_logic_arith.all;
use SYNOPSIS.std_logic_unsigned.all;

library METAMOR;
use METAMOR.ATTRIBUTES.all;
```

to this:


```
-- SYN_OPSYS library declaration
-- library SYN_OPSYS;
-- use SYNOPSIS.std_logic_arith.all;
-- use SYNOPSIS.std_logic_unsigned.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

-- library METAMOR;
-- use METAMOR.ATTRIBUTES.all;
```

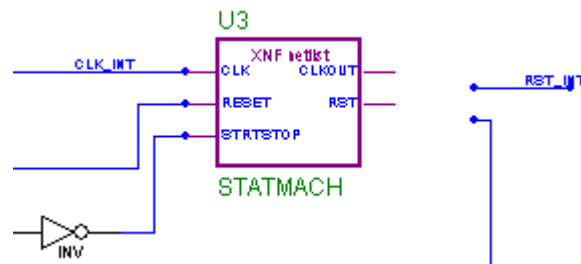
All we really did was remove the references to Metamor, and set libraries to IEEE libraries rather than Synopsys standard libraries.


Since Synopsys is a recognized keyword called a VHDL "pragma", you can delete the first two lines, or just modify the word to SYN_OPSYS, as in the example text above.

28) Exit the **HDL Editor** and save your changes: **File → Exit, <Yes>**.

29) Return to **FPGA Express**, highlight **Statmach.vhd**, right-click, and select Force Update File. You will see a green  checkmark, meaning that the file checks out OK.

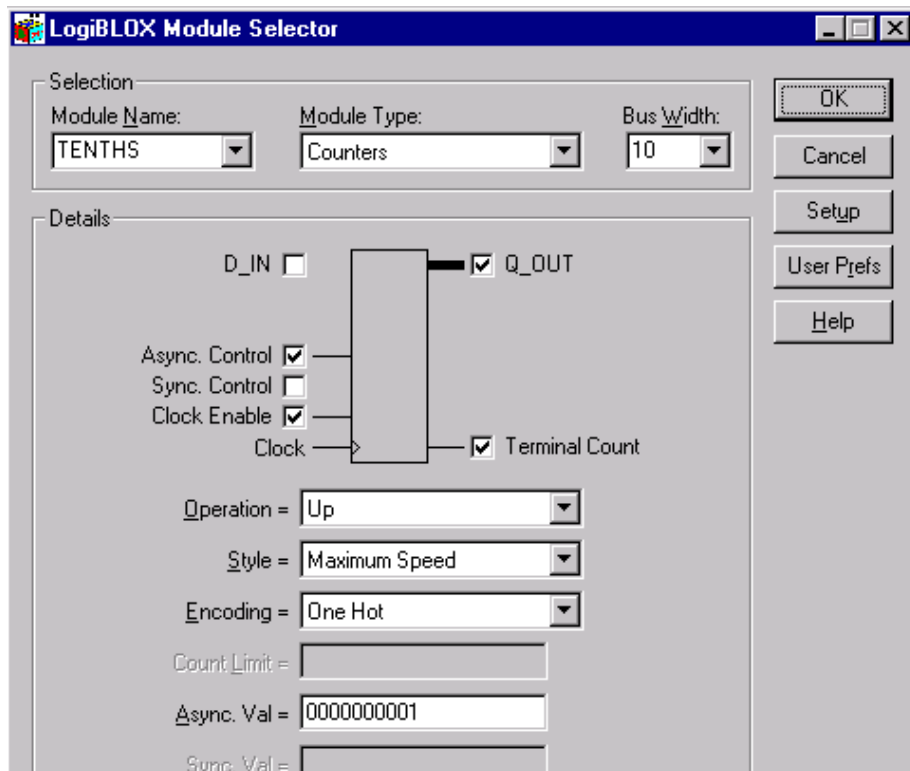
- 30) Double-click the **statmach.vdh** file (or click once on the “+” sign) to expand the source to its component, **statmach**.
- 31) As we did for the HEX2LED macro, select **Synthesis → Create Implementation**. Again, make sure that the Target device is **XC4005XL-3PC84**, and check the box for ‘Do not insert I/O pads’. Hit <OK>.
- 32) Under the **Chips** window, optimize your chip design by selecting **Synthesis → Optimize Chip**.
- 33) Select **Synthesis → Export Netlist...** The default directory of **C:\F15_labs\watchnew** is fine, so click <Save>.
- 34) You can exit **FPGA Express** now and return to the **Foundation Schematic Capture** window.
- 35) Now we will create a macro symbol from the VHDL code that we just synthesized. Select **Hierarchy → Create Macro Symbol from Netlist**. The directory location, **c:\F15_labs\watchnew**, should be correct.
- 36) Since we are looking for an XNF file, select **Files of type** as **Xilinx (*.XN*, *.BAX)**, and then select **statmach.xnf**. Click <Open>. Schematic and black box simulation files are created, and STATMACH is added to the project library.
- 37) Open the **Symbols toolbox**, find the **STATMACH** macro, and place one of these new macro symbols in the schematic. Place it near the top of the diagram so that the three input pins are just touching the hanging nets, as shown in this picture:



- 38) Then click once on **STATMACH** to select it. A red box surrounds STATMACH.
- 39) Connect the symbol using the Connect Symbol icon. 
- 40) You still need to connect the 2 output lines. **RST_INT** connects to the **RST** output and the unnamed net connects to the **CLKOUT** output, as shown in **Figure 1**.

Creating a custom schematic macro using LogiBLOX

- 41) We still need to create the counter in the center of **Figure 1**. Although we could use any of several existing counter macros, a custom counter will provide exactly what we need for this particular design. We want a counter with asynchronous control, terminal count, and 10 one-hot style outputs. We'll use **LogiBLOX** to create this.
- 42) In the **Schematic Editor**, select **Options → LogiBLOX...**
- 43) Under the **Module Name**, type **TENTHS**. Under **Module Type**, select **Counters**. Under **Bus Width**, type **10**.
- 44) Select the **<Setup>** button, choose the **Device Family** tab, and make sure that the xc4000xl has been selected. Click **<OK>**.
(Note: If we were planning to incorporate this LogiBLOX module into a VHDL design, we would also go into the Options tab and check the boxes to create a Behavioral VHDL Netlist and a VHDL Template.)
- 45) Set the other attributes as shown in the diagram below. Note that the Async. Value, which is the state value upon startup, is an explicit 10 digit entry, '0000000001'.
- 46) Make the LogiBLOX GUI Messages window visible so that you can watch the software complete its task. In the **LogiBLOX Module Selector** window, select **<OK>**, and let LogiBLOX create the custom macro.



- 47) Place the **TENTHS** macro into the middle of your schematic. Make sure to place it so that the individual pins match the position of the existing nets. Select the new macro so

that a red box surrounds it, and then click the **Connect symbol** icon. (You may still have to manually connect the bus output to the TENTHSOUT[9:0] bus.)

- 48) If you wish, you can make your macro look ‘prettier’ by double-clicking the macro, changing attributes such that the Symbol Name is visible, by moving the Symbol Name and/or parameters, etc.

- 49) Save your new schematic and return to **Foundation Project Manager**.

Note: The schematic explicitly set certain pin locations for input and output pins, for example, the pad connected to net **RESET** is shown with **LOC=P28**. These pinouts apply to the UW-FPGABOARD. But since we are using the XS40 and XSTEND boards, different pad locations have been defined in the **WATCHNEW.UCF** file. Settings in the UCF file take precedence over attributes in the schematic design. To see the UCF file, double-click on **WATCHNEW.UCF** on the left side of the Foundation Project Manager.

- 50) Click the “**Implement M1**” button.

- 51) Implement the design. When finished, download to the demo board. Use the SPARE button for the STRTSTOP input and the SW8 switch Reset. Set SW8 high to reset the stop watch.

- 52) You will probably notice that the stopwatch mostly works. However, when you start and stop the stopwatch, it seems to reset. Why is this?
The reason is that there is a glitch in the asynchronous ‘rst’ circuit of the STATMACH state machine that was generated. This is a common danger with HDL design.

Note the code below:

```
rst <= '0' when (stopwtdch = counting) else
'0' when (stopwtdch = start) else
'0' when (stopwtdch = stop) else
'0' when (stopwtdch = stopped) else
'0' when (stopwtdch = zero) else
'1';
```

In this portion of the code, one can see that the **rst** net could go to ‘1’ if there is a glitch in the state machine control outputs. This bug was fixed in the **C:\F15_labs\shortcut\watchnew\Statmach.vhd** version of the design by replacing the code above with the following VHDL code:

```
-- Keep rst as 1 all the time except for beginning to prevent glitches.
rst <= '1' when (stopwtdch = clear) else '0';
```

Conclusion

In this lab, the three different design entry modes were used – schematic, state editor, and Express for VHDL synthesis. The top level design was a schematic. Various ways of creating custom design components were used and integrated into one design.

Optional – Using State Diagram Editor with the ABEL Option

Modify your existing design using **ABEL** instead of **VHDL** for the **STATMACH** state machine output. This flow is more integrated, and is similar to what the state editor & VHDL flow will be like in version F1.5.

- 1) Open the **State Editor** tool and select **Statmach.asf** as your design file.
- 2) Choose **Synthesis → HDL Configuration**. Set this to **ABEL** language. <OK>
- 3) Since the state transition and signal output constructs are defined in the state editor in VHDL syntax, you must manually change each text box in the diagram.

| <u>Syntax</u> | <u>VHDL</u> | <u>ABEL</u> |
|--------------------------|--------------------|--------------------|
| Assignment operator | <= | = |
| 'Is equal to' comparator | = | == |

For example, change **reset='1'** to **reset= = '1'** to match the ABEL syntax. Do the same for all other text in the diagram.

- 4) Save the revised **statmach.asf** file.
- 5) Select **Synthesis → HDL Code Generation**.
- 6) Select **Synthesis → Synthesize**. This will synthesize your generated ABEL code using the ABEL HDL synthesizer.
- 7) Select **Project → Create Macro**.
- 8) Go into the **Schematic Editor** and add this new macro to your schematic.
- 9) Continue with implementation and download.