# <u>Tutorial 2: Sequential Logic Circuits using Aldec Active-HDL and</u> <u>Xilinx Vivado</u>

This tutorial will guide you through specifying a design for a 3-bit up/down counter using Aldec Active-HDL. The software includes a State Diagram wizard that allows you to draw a state diagram, specify paths and conditions, etc. This tutorial assumes that you have a basic familiarization with Aldec Active-HDL. If not, you may want to review "<u>Tutorial 1 - Combinational Logic Circuits using Aldec Active-HDL and Xilinx</u> <u>Vivado</u>".

This tutorial will guide you through:

- Setting up a new workspace
- Using the State Diagram wizard to create a state diagram, including the definition of states, conditional transitions, and outputs.
- Simulating the design using stimulators (clock waveforms in this example) and verifying that the output waveforms are correct.
- Using the created sequential circuit as a component. This component, along with a component to create a 1 Hz clock and a component for a BCD to 7-segment decoder, will be incorporated into an overall VHDL file to be synthesized and implemented on the BASYS3 FPGA board.

Refer to the tutorial **"Tutorial 1: Combinational Logic Circuits Using Aldec Active-HDL and Xilinx Vivado**" for synthesizing and implementing a design on the BASYS3 FPGA board.

### 1. Creating a Project with Aldec Active-HDL

- Launch Aldec Active-HDL
- If licensing windows appear, select <u>Next</u> until the Getting Started window appears as shown below.
- Select <u>Create new workspace</u> as shown below and then select <u>OK</u>.

Getting Started	×
C Open existing workspace	More
Create new workspace	
Always open last workspace	Cancel

• Enter a name for the workspace (**Counter3Bit** was selected below), change the location of the workspace folder (or use the default as below), and select <u>OK</u>. Note that the name for the project workspace, VHDL entity (to be entered later), and architecture (to be entered later) should be the same.

New Workspace		×
	Specify basic information about the new workspace.	
	Type the workspace name:	
Comin	Counter3Bit	
A marine we	Select the location of the workspace folder:	
A	c:\my_designs\	
A AL	<u>B</u> rowse	
	✓ Add New Design to Workspace	
T Jammes J.	OK Ca	incel

• Select Create an Empty Design with Design Flow as shown below and select Next.

New Design Wizard	×
How would you like to create Design Resources?	
C Create an Empty Design	
Create an Empty Design with Design Flow	
C Add existing Resource Files	
C Import a Design from Active-CAD	
This option creates an empty design and enables Design Flow Manager. You can select a vendor of your synthesis or implementation tool, technology, libraries, and specify the default HDL language of your new design entry sources.	
C Create New Workspace	
Add Design to Current Workspace	
< <u>B</u> ack <u>N</u> ext > Cancel	

• The next window that appears shows information about synthesis tools and implementation tools that might be configured to launch automatically from Aldec Active-HDL. Synthesis tools can also be launched separately rather than integrating them into Aldec. We will launch Xilinx Vivado separately, so you can ignore most settings shown. Select <u>VHDL</u> for the Default HDL Language and then select <u>Next</u>.

New Design Wizard	×
Specify additional information about the new design.	_
C-Synthesis tool:	
Synthesis tool:	
<none></none>	
Physical Synthesis tool: <none></none>	
Implementation tool:	
<none></none>	
Default Family:   Flow Settings	
Block Diagram Configuration: Default HDL Language	
< <u>B</u> ack <u>N</u> ext > Can	cel

• Enter the design name. Note that it should match the workspace name used earlier (Ex1 for this example). Enter the name and select <u>Next</u>. Select <u>Finish</u> on the final screen shown below.

New Design Wizard	< New Desig	gn Wizard		×
Specify basic information about the new design.	The net	w design will have the following specific	ations:	
Type the design name:	Design	n name: Counter3Bit	^	-
Select the location of the design folder:       c:\My_Designs \Counter3Bit				
Do not create design directory The name of the default working library of the design: Counter3Bit	6		v	
The name specified here will be used as the file name for the library files and as the logical name of the library. You can change the logical name later on.	1.5			
Design file path: c:\My_Designs\Counter3Bit\Counter3Bit\Counter3Bit.adf	Design	file path: Designs\Counter3Bit\Counter3Bit\Cour	iter3Bit.adf	-
	Cor	mpile source files after creation		
< <u>B</u> ack <u>N</u> ext > Cance	]	< <u>B</u> a	ck Finish Cancel	

#### 2. Specifying your sequential circuit design using a state diagram

- The Design Flow Manager screen should have appeared after the last step.
- The Design Browser should appear on the left of the screen. If it does not appear, it can be toggled on and off using Alt + 1.
- Double-click on <u>Add New File</u> under the Design Browser and the Add New File window should appear.
- Note that there are several types of files that can be specified and that a new file wizard can be used to assist you in specifying inputs and outputs. Select <u>Wizards</u>, select <u>State Diagram</u>, enter a File Name (<u>Counter3Bit</u> in this case) and then select <u>OK</u>.



Select <u>Next</u> in the screen below on the left.
 Select <u>VHDL</u> and then <u>Next</u> in the screen below on the right.

New Source File Wizard		×	New Source File Wizard - Language	×
	This wizard will create a source file with an initial block diagram using the design specification you will enter in the following wizard dialogs.           Image: Add the generated file to the design           Clear this check box if you do not want to add the file generated by the wizard to the current design.		Choose the language that will be generated from the block diagram. This can be changed from the Block Diagram Editor if required. C EDIF C VHDU C Verilog	
	< <u>B</u> ack <u>N</u> ext > Cancel		< Back Next > Cancel	

• Enter the source file name (Counter3Bit) in the window shown below and select <u>Next</u>.

New Source File Wizard - Name	×
New Source File Wizard - Name         Type the name of the source file to create:         Counter3Bit       Browse         You can use the Browse button to specify the file.         Type the name of the entity (optional):         By default, the entity name is the same as the file name.         Type the name of the architecture body (optional):         By default, the architecture name is the same as the entity name.         By default, the architecture name is the same as the entity name.	×
< Back Next > Cancel	-

- Before we proceed, it is a good idea to be clear on exactly what inputs and outputs are needed for a 3-bit up/down counter. The counter will require a clock input, a count direction control (x), and a 3-bit output (ABC where A is the MSB).
- Select <u>New</u> in the window below on the left to add a new port. Name it  $\underline{X}$  with direction <u>in</u>.
- Also add input port <u>CLK</u> as well as output ports A, B, and C as shown below and then select <u>Next</u>.

New Source File Wizard -	Ports X	New Source File Wizard - Ports	×
X Counter3Bit	To add a new port, click New. To edit a port, select it on the list. Then you can change its name, direction and type. To quickly change the index constraint of a port of a one-dimensional array type, use the Array Indexes box. To remove a port, select it on the list, and then click Delete. Name: Array Indexes: Port direction © in © inout © out © buffer New Delete Type	To add a new port, click New. To edit a port, select it on the list. Then you can constraint of a port of a one-dimensional array type the Array Indexes box. To remove a port, select it on the list, and then cliv Delete. A B CLK CLK COUNTER3BIT New Delete New Delete New Delete Type To add a new port, click New. To remove a port, select it on the list, and then cliv Delete. Name: Array I CLK CLK CLK New Delete Type To add a new port, click New. To remove a port, select it on the list, and then cliv Delete. Name: Array I CLK CLK To inout Cout C buffer Type To add a new port, click New. To remove a port, select it on the list, and then cliv Delete. New Delete Type To add a new port, click New. To remove a port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv Delete. Type To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it on the list, and then cliv To add a new port, select it o	hange e index , use k hdexes:
	< <u>B</u> ack Finish Cancel	< <u>B</u> ack Finish C	incel

• Note that after selecting <u>Next</u> in the screen above, a message appears (shown below): *"You have not entered any clock port. Do you want port CLK to be a clock?"* Select <u>Yes</u>.

New Source File Wizard -	Ports X	
	To add a new port, click New. To edit a port, select it on the list. Then you can change its name, direction and type. To quickly change the index constraint of a port of a one-dimensional array type, use the Array Indexes box. To remove a port, select it on the list, and then click Delete.	
Counter3Bit	A     Name:     Array Indexes:       B     CLK     •     •       CLK     •     •     •       X     •     •     •       © in     © inout     © out     © buffer	New Source File Wizard       ×         You have not entered any clock port.       >         Do you want port 'CLK' to be a clock?       Yes         Yes       No
	New     Delete     Type       < Back	

• Before proceeding the state diagram to be implemented must be known. In this example a 3-bit (mod-8) up/down counter is to be implemented. In input switch X will be used where the counter will count UP when X=1 and the counter will count DOWN when X=0. So the desired state diagram is:

# Add your state diagram here (by hand)

• Individual states and state transitions can be added in Aldec, but the State Machine wizard will do a lot of the work for you. In the example below



• The state diagram is now displayed as shown below.



- Zoom in on the state diagram shows it more clearly as seen below.
- Note that 8 states were created: S1 S8.

©∭No clock enable	53
Counto Rol	State Properties       ×         General Graphics Actions Links Comment View Texts         Name:       Code:         Count0       000         Default       Enter state code value using binary format (e.g. 0101)         Trap       binary format (e.g. 0101)         Hierarchical       If State Codes Visible
	OK Cancel Apply

- <u>Naming States</u>: Double-click on each state and change the state properties (see above and below).
   O Under the General tab:
  - Enter a name for the state (VHDL style) and a binary code.
  - The first state, S1, was renamed as Count0 with the code 000 (see below).
  - The second state, S2, was renamed as Count1 with the code 001.
  - Continue for the remaining states
  - Under the <u>Actions tab</u>: Enter the values to be assigned to the outputs (and copy this information so that you can paste it and edit it in the next state).
    - Count0 was given the following actions:  $A \le 0'; B \le 0'; C \le 0'; (see below)$
    - Count1 was given the following actions:  $A \le 0^{\circ}$ ;  $B \le 0^{\circ}$ ;  $C \le 1^{\circ}$ ;
    - Continue for the remaining states.

State Properties	$\times$	State Properties X
General       Graphics       Actions       Links       Comment       View Texts         Name:       Code:       000       000         Count0       Default       Enter state code value using binary format (e.g. 0101)         Hierarchical       Image: State Codes Visible		General         Graphics         Actions         Links         Comment         View Texts           Entry:
OK Cancel Apply		OK Cancel Apply

- The state diagram for the 3-bit up/down counter is shown below after naming all states and specifying actions for each state. You might also need to drag states or boxes to new locations to make room.
- <u>Adding Text</u>: Note that text was added to the top of the page using the text tool on the toolbar.



- <u>Adding Conditions to Transitions</u>: Double-click on the transition line and add the condition under the <u>HDL tab</u>
- In the example below, double-clicking on the transition from Count2 to Count3 opened the Transition Properties window. The HDL tab was selected and the condition X='1' was added (no semicolon).
- Recall that X=1 to count UP and X=0 to count DOWN.

Count2	:= '0'; B <= '1'; C <= '0';
Transition Properties X	
General Graphics HDL Comment View Texts	Count3 /011/
OK Cancel Apply	

- Continue adding all required transitions.
- Drag the straight transition lines to change them to arcs.
- Carefully drag the transition line conditions and actions as needed so that the state diagram is neat and easy to read.



• Note that a VHDL file has been generated with complete entity and architecture sections. Expand the folder Counter3Bit.asf in the browser (or a similar name for your design) to see the VHDL file listed (.vhd). You might want to look at the architecture section to appreciate the work that the Block Diagram tool has done for you!

Active-HDL 10.2 (Counter3Bit ,Counter3Bit) - C:/N	My_Designs/Counter3Bit/Counter3Bit/compile/Counter3Bit.vhd — 🛛 🗙				
<u>F</u> ile <u>E</u> dit Sea <u>r</u> ch <u>V</u> iew W <u>o</u> rkspace <u>D</u> esign <u>S</u>	<u>S</u> imulation <u>T</u> ools <u>W</u> indow <u>H</u> elp ↔ ×				
🗗 - 🖻 🖬 🐹 📖 🔳 🚳 💱 💻 🔎 🔛 2	🍃 🔍 🎁 🏆 🖓 🛅 🙀 🕺 🖧 🗇 🆃 🏷 🕨 🕨 🕨 👌 100 ns 🗄 🕂 🔳 🔺 🕨				
Design Browser 🔷 🗙	⊈ ⊈				
Top-Level selection					
OUnsorted	61				
→ Workspace 'Counter3Bit': 1 design(s)	62				
⊡ i Counter3Bit	63 Next State Logic (combinatorial)				
Add New File	65 Sreg0 NextState: process (X. Sreg0)				
1 □ 👻 🗸 Counter3Bit.asf	66 begin				
t a / Counter3Bit.vhd	<pre>67 NextState_Sreg0 &lt;= Sreg0;</pre>				
	68 Set default values for outputs and signals				
Add New Library	69				
	70 case Sregu 1s				
the countersbit library	72 $A \leq 10^{\circ}$ :				
	73 B <= '0';				
	74 C <= '0';				
	75 if X = '0' then				
	<pre>76 NextState_Sreg0 &lt;= Count7;</pre>				
	77 elsif X = '1' then				
	<pre>78 NextState_Sreg0 &lt;= Count1;</pre>				
	79 end if;				
	80 When Countries				
	82 B <= '0';				
	83 C <= '1';				
	84 if X = '1' then				
	<pre>85 NextState_Sreg0 &lt;= Count2;</pre>				
	86 elsif X = '0' then				
	<pre>87 NextState_Sreg0 &lt;= Count0;</pre>				
	88 end if;				
	89 when Count2 =>				
	$A = 10^{-1}$				
92 C <= '0':					
	93 if X = '0' then				
94 NextState Sreg0 <= Count1;					
	95 elsif X = '1' then				
	<pre>96 NextState_Sreg0 &lt;= Count3;</pre>				
	97 end if;				

#### 3. <u>Simulating your state machine</u>

- Now we need to simulate the design to see if is correct. We could simulate it as we did with combinational logic circuits: Use the testbench wizard and add VHDL code to specify input waveforms. An alternate way to simulate the circuit is described below.
- First, what input waveforms should we specify? We should clock our 3-bit counter at least 8 times while X = 1 (count up) and at least 8 times while X = 0 (count down). Suppose that we clock it 10 times in each direction for a total of 20 clock pulses. If each clock pulse is 100ns in length then the analysis should last 2000ns and the clock has a frequency of 1/100ns = 10 MHz. Additionally, we could use another clock for X with a period of 2000ns (i.e., low for the first 10 counts or 1000ns and HIGH for the

next 10 counts), so the clock frequency for X would be 1/2000ns = 500 kHz. These waveforms are illustrated below.



(f = 500 kHz)

• Select <u>Counter3Bit (Counter3Bit Arch)</u> (or similar name for your design) in the Design Browser as shown below.



- Before we display waveforms, be sure to set the default waveform viewer as follows:
- Select <u>Tools Preferences</u> and the window below should appear. Select Waveform Viewer/Editor from the Categories listed in the left part of the window, change the default waveform viewer/editor to <u>Standard Waveform Viewer/Editor</u> and select <u>Apply</u> and then select <u>OK</u>.



- Select **Initialize Simulation** from the **Simulation** menu.
- Create a new waveform window by selecting the <u>New Waveform tool</u> on the main menu (shown below).
- Select the <u>Structure tab</u> in the design browser and select <u>counter3bit (counter3bit arch)</u> or something similar using your file name so that the available waveforms are listed as shown below.



• Select the desired waveforms and drag them to the Waveform Viewer window. (Select any waveform and then used Ctrl + A to select all waveforms.) Drag the waveforms to rearrange them in the desired order if necessary.



• Right click on waveform CLK and pick <u>Stimulators...</u> from the menu that appears.



• Click on <u>Clock</u>, set the frequency to 10 MHz, and select <u>Apply</u>.

<b>N</b> Stimulators		$\times$
Signals Hotkeys Predefined		
Set : ASDB Stimulators	New Remove	
Signals: Name Type CLK Clock	Type: Forces a clock pulse of a specific frequency and duty cycle f(t) Formula 010 Value	
Display paths Save	Apply Strength: Override	
	Close	

• Similarly, right-click on waveform X, select <u>Stimulators</u>, select <u>Clock</u>, set the frequency to 500 kHz, and select <u>Apply</u>.

N Stimulators	$\times$
Signals       Hotkeys       Predefined         Set :       ASDB Stimulators <ul> <li>New</li> <li>Remove</li> </ul> Signals:       Type:         V       Clock       Forces a clock pulse of a specific frequency and duty cycle         V       Clock <ul> <li>f(t)</li> <li>Formula</li> <li>0</li> <li>Frequency: 500kHz</li> </ul> 0 <ul> <li>Frequency: 500kHz</li> <li>State</li> <li>State</li> <li>State</li> <li>State</li> <li>State</li> <li>State</li> <li>State</li> <li>State</li> <li>State</li> </ul>	
Display paths Save Apply Strength: Override	
Close	

• Set the final time on the main menu to <u>2000ns</u> and select the <u>Run Until</u> button. The Run Until window will appear as shown below. Select <u>OK</u>.

Active-HDL 10.2 (Counter3Bit , Co	ounter3Bit) - untitled.awc *			
<u>F</u> ile <u>E</u> dit Sea <u>r</u> ch <u>V</u> iew W <u>o</u> rks	pace <u>D</u> esign <u>S</u> imulation	Waveform Tools	s <u>W</u> indow <u>H</u> elp	
🗗 ▼ 📽 📕 🛣 🔛 🔳	ş 📃 🔎 🛗 💈 🔍 📶	🏆 🖓 🛅 🙀 🛛	😵 🕒 🕸 🍪 📙 🕨 🛃 2000 ns 🛨 📢	
Design Browser *	🗟 🕐 🐜 🗟 🕂	船 🏨 🏰 🔂 🗎	L 🔍 🖳 🖳 🔲 🕨 🔏 🏙 🕫 🦂 🎘 🖄	
🔄 counter3bit (counter3bit_ 💌	Signal name	Value	· · · 400 · · · 800	
Hierarchy	► CLK	U	0 fs	
+-	• X	U	Run Until	
-P std.standard	-• A	U	button Final time	
-P std.TEXTIO	-• B	U		
P ieee.std logic 1164	-• C	U Run	n Until X	
- P ieee.std_logic_arith	™ Sreg0	count0	Enter time you want to run simulation until.	
□ eee.STD_LOGIC_UN			Default time unit is picosecond.	
			200	
▼				
Name Value OK Cancel				
► CLK U				
►X U				

- The output waveforms A, B, and C should now be correct. Study the waveforms and note that initially X = 0 and the count is 0, 7, 6, 5, 4, 3, 2, 1, 0, 7, 6 and then when X = 1 the count is 6, 7, 0, 1, 2, 3, 4, 5, 6, 7, 0. Include the signal Sreg0 so that the name of each state will be displayed.
- Use *Zoom to Fit* to display all counts concisely.

Active-HDL 10.2 (Counter3Bit , Co	ounter3Bit) - u	ntitled.awc '		п×			
<u>F</u> ile <u>E</u> dit Sea <u>r</u> ch <u>V</u> iew W <u>o</u> rksp	ace <u>D</u> esign	Simulation	<u>W</u> aveform <u>T</u> ools <u>W</u> indow <u>H</u> elp	⇔ × ≪			
🛛 🗗 📽 🔚 👯 🗱 📕 🚳 💈	ş 📃 🔈 🔛	; 🌮 🔍 🏙	🛛 🏧 🖓 🔝 🙀 🕺 🌢 🍪 🤣 🕨 🕨 🕨 👌 2000 ns 🛨 📢 🔳 🚽 🕪 🖣 🗐 📮 💷 🛛 2us	+ 1			
Design Browser 🔷 🔺	🛛 🖬 🕐 ຈິນ	تے 🎜 🖉	• 船 漁 頌 🖥   风 있 🔍 * 🔹 🍓 ۴ 🥕 🌾 ル 🎘 🛝 🖬 🖬 🖬 👘 🖓 🖉 🦛 🖬 🖬 🖉 🛲 🖉 📰 🗣 🗋 🗁 🍴				
🔚 counter3bit (counter3bit_ 💌	Signal name	Value	······································	''' ns			
Hierarchy	<ul> <li>CLK</li> </ul>	1 to 0		2 us _ ^			
- The Countor 2 Pit (Count	⊳-X	1 to 0	X = 0, so count DOWN $X = 1$ , so count UP				
Countersbit (Count	-• A	0					
std.standard	-n B	0					
- P std.TEXTIO		0					
Pieee.std logic 1164	-• C	0					
P ieee std logic arith	J Sreg0	count0	🔜 X count? X counts X counts X counts X counts X count2 X count1 X count1 X count2 X count3 X count3 X count3 X count3 X count4 X count3 X count4 X count5	όχ count7χ			
leee.STD_LOGIC_ON	Cursor 1						
II				<u>20</u> , ↓			
wave.asdb SIM							
📑 Files / 🐺 Stru/ 📴 Res/	🚪 design f	flow 🛓 🧶 co	unter3bit),≘ counter3bit); <mark>i≩ untitled.awc *</mark> /				

#### **Using Components in VHDL Designs**

Now that we have simulated the 3-bit counter and are sure that it is working correctly, we would like to implement design into an FPGA. However, we also need additional items:

- Clock source: a 100MHz internal clock is available on the BASYS3 FPGA board
- Clock divider: we can divide the 100MHz clock by 100E6 to generate a 1 Hz clock
- BCD to 7\_segment decoder: the output of our 3-bit counter is in BCD form
- 7-segment display: available on the BASYS3

This is illustrated by the schematic on the following page.



A total of 4 VHDL files will be used:

- <u>Counter3Bit.vhd</u> We just created and tested this file
- ClockDivider.vhd
  - Available on the course Bb site
  - Shown on the following pages
  - Divides a 100MHz input clock to produce a 1 Hz output clock
- BCD-to-7Segment.vhd
  - Available on the course Bb site
  - Shown on the following pages
  - Converts a BCD value (output of our counter) to common-anode 7-segment display values

#### • <u>CounterWithClock.vhd</u>

- o Available on the course Bb site
- Shown on the following pages
- This is the overall VHDL file that uses the other three VHDL files as components. This structural file essentially makes the connections to implement the schematic above.

```
- Clock Divider
   -- Divides 100 MHz clock by 100E6 to produce a 1 Hz clock (with a 32.89% duty cycle)
   -- Reference: www.digilentinc.com - Real Digital, Module 10
   -- File: ClockDivider.vhd
   library IEEE;
                                                ClockDivider.vhd
   use IEEE.STD LOGIC 1164.all;
   use IEEE.STD_LOGIC_ARITH.all;
   use IEEE.STD_LOGIC_UNSIGNED.all;
9
   _____
   entity ClockDivider is
      Port ( clkin, reset : in STD LOGIC;
            clkout
                    : out STD LOGIC);
13
   end ClockDivider;
14
   _____
   architecture ClockDivider of ClockDivider is
   constant cntendval : STD LOGIC VECTOR(26 downto 0) := "10111110101111000010000000";
16
17
   -- Note: 100E6 in binary is 10111110101111000010000000
   signal cntval : STD LOGIC VECTOR (26 downto 0);
19
   begin
      process (clkin, reset) -- run process whenever clkin or reset change
21
          begin
             if reset = '1' then cntval <= "00000000000000000000000000";
23
             elsif (clkin'event and clkin = '1') then
                24
                else cntval <= cntval + 1;</pre>
26
                end if;
27
             end if;
      end process;
29
   clkout <= cntval(26);
                       -- Concurrent statement, so clkout changes when cntval changes
   31
   -- so clkout = 0 for 67108864 pulses and clkout = 1 for 32891136 pulses
33
   -- for a duty cycle of (32891136/10000000)*100 = 32.89%
34
   end ClockDivider;
```

1 2 3 4	BCD to 7-segment decoder File: BCD_to_7segment.vhd library IEEE; use IEEE.STD_LOGIC_1164.all;	BCD_to_7segment.vhd							
6 7 8 9	<pre>entity BCD_to_7segment is     port(BCD : in STD_LOGIC_VECTOR(3 downto 0); BCD(3) = MSB         Seg : out STD_LOGIC_VECTOR(6 downto 0)); Seg(6) = cathode a end BCD_to_7segment; Seg(0) = cathode g</pre>								
10 11 12 13	architecture BCD_to_7segment of BCD_to_7segment is begin DisplayEncodeProc : process (BCD) run proces	ss whenever input BCD changes							
	begin								
15	Case BCD 15								
17	Cathode Value (ad	CTIVE-LOW)							
10	abcderg	l diit 0							
10	when "0000" => Seg <= "0000001"; di	splay algit U							
19	when "0000" => Seg <= "1001111"; di	splay digit i							
20	when 0010 => Seg <= 0010010; di	splay digit 2							
21	when 0011 => Seg <= 0000110; di	splay digit 5							
22	when 0100 => Seg <= 1001100; di	splay digit 4							
23	when "0101" => Seg <= "0100100"; di	splay digit 5							
24	when "0110" => Seg <= "0100000"; d1	splay digit 6							
25	wnen "0111" => Seg <= "0001111"; d1	splay digit /							
26	when "1000" => Seg <= "0000000"; al.	splay digit 8							
27	when "1001" => Seg <= "0000100"; all	splay algit 9							
28	when "1010" => Seg <= "1110010"; all	splay unique pattern used by 7447 for input 10							
29	when "1011" => Seg <= "1100110"; all	splay unique pattern used by 7447 for input 11							
30	when "1100" => Seg <= "1011100"; di	splay unique pattern used by 7447 for input 12							
31	when "1101" => Seg <= "0110100"; d1	splay unique pattern used by 7447 for input 13							
32	when "1110" => Seg <= "1110000"; d1	splay unique pattern used by 7447 for input 14							
33	when others => Seg <= "IIIIIII"; al.	I algits turnea off for input 15 (or X's, U's)							
34	ena case;								
35	end process;								
36	end BCD to /segment;								

```
-- File: Counter with Clock.vhd
     -- Structural file for Lab 7 that uses 3 components
    library IEEE;
                                                     CounterWithClock.vhd
4
5
6
7
    use IEEE.std logic 1164.all;
    use IEEE.std logic arith.all;
    use IEEE.std_logic_unsigned.all;
    entity counter_with_Clock is
       port (
         CLK100M, UPDOWN, reset : in STD LOGIC; -- 100 MHz clock input from BASYS3 (pin W5)
         An : out STD LOGIC VECTOR (3 downto 0); -- Enable lines to four 7-segment displays on BASYS3
         Seg : out STD_LOGIC_VECTOR(6 downto 0)); -- Seg(6) = cathode a, ..., Seg(0) = cathode g
13
    end counter_with_Clock;
14
    _____
15
    architecture counter_with_Clock_arch of counter_with_Clock is
16
    -- component declarations
17
    component ClockDivider -- use component defined in file Clock1Hz.vhd to create 1Hz clock
18
       port (
19
                    : in STD_LOGIC; -- 100 MHz clock available on BASYS2 FPGA board
          clkin
                    : in STD_LOGIC; -- used to reset the clock divide counter to 0
          reset
21
                    : out STD LOGIC); -- 1 Hz clock output used to clock 7-seg displays
          clkout
    end component;
    component counter3bit -- use component defined in file counter3bit.vhd (or student file name)
23
24
      port (
25
                    : in STD LOGIC;
          CLK, X
                                    -- 1 Hz clock input -- X = 1 to count UP, X = 0 to count DOWN
26
                    : out STD_LOGIC); -- counter output (A = MSB)
          A, B, C
27
    end component;
28
    component BCD to 7segment -- use component defined in file BCD to 7segment
29
        port (
          BCD : in STD_LOGIC_VECTOR(3 downto 0); -- BCD(3) = MSB
31
          Seg : out STD_LOGIC_VECTOR(6 downto 0)); -- Seg(6) = cathode a, ..., Seg(0) = cathode g
32
    end component;
33
    _____
                                _____
34
     -- Define intermediate signals
    signal CLK1 : std logic; -- define intermediate signal between components
36
    signal B : std logic vector (3 downto 0); -- intermediate signals between counter and BCD to 7segment
37
    38
    begin
39
       B(3) <= '0';
                     -- Only 3 bits used for 3-bit counter, so set 4th bit (MSB) to 0
       AN <= "1110":
                    -- Turn OFF first three 7-segment displays and turn on the 4th display
       -- instantiate components
42
       Ul: ClockDivider -- define an instance of ClockDivider
43
       port map(clkin => CLK100M, reset => reset, clkout => CLK1);
44
       U2: counter3bit -- define an instance of counter3bit
       port map(CLK => CLK1, X => UPDOWN, A => B(2), B => B(1), C => B(0));
46
       U3: BCD to 7segment -- define an instance of BCD to 7segment
       port map(BCD => B, Seg => Seg);
48
    end counter with Clock arch;
49
    _____
```

- <u>Download the following files from the course Bb site.</u> They can be stored anywhere, but a good place is to store them in the source (src) folder of the project. (Note that the file Type may not have the correct description and the extension may not be shown, but they will appear correctly within Aldec.)
  - ClockDivider.vhd
  - BCD-to-7Segment.vhd
  - CounterWithClock.vhd

I I I I I I I I I I I I I I I I I I I								
File Home Share	view							
← → ヾ  📙 : Th	iis PC > OSDisk (C:) > My_Desig	ns > Counter3Bit > Counter3Bit >	src					
^	Name	Date modified	Туре	Size				
🖈 Quick access	wave.asdbw	11/5/2018 1:03 PM	File folder	1 1				
📃 Desktop 🖈	BCD_to_7segment	4/5/2010 10:50 PM	Hard Disk Image File	2 KB				
👆 Downloads 🖈	ClockDivider	10/28/2018 6:20 PM	Hard Disk Image File	2 KB				
🚆 Documents 🖈	Counter_with_Clock	10/29/2018 2:42 PM	Hard Disk Image File	3 KB				
📰 Pictures 🛛 🖈	💝 Counter3Bit	11/1/2018 1:08 PM	Active-HDL State Diagram	34 KB				
👌 Music	Counter3Bit.prc	11/1/2018 1:08 PM	PRC File	1 KB				
📑 Videos	🗱 wave	11/5/2018 1:01 PM	Active-HDL Accelerated	4 KB				

- <u>Add the files to the Aldec Project</u>.
  - Select Design Add Files to Design



• Locate the files in the project src directory (or wherever you stored them) and select **Open.** 

Add Files to Design X							
Loc	ok in: 🔄 src 💌	+ 🗈 💣 📰 - 🖷	j				
Quick acce Desktop	Name wave.asdbw BCD_to_7segment ClockDivider Counter_with_Clock Counter3Bit Counter3Bit.prc wave	Date modified 11/5/2018 1:03 PM 4/5/2010 10:50 PM 10/28/2018 6:20 PM 10/29/2018 2:42 PM 11/1/2018 1:08 PM 11/1/2018 1:08 PM 11/5/2018 1:01 PM	Type File folder Hard Disk Hard Disk Hard Disk Active-HE PRC File Active-HE				
This PC							
	<		· ·				
	File name:     ["Counter_with_Clock.vhd" "BCD_to       Files of type:     All Files (*.*)       Open as:     Auto       Image:     Auto	o_7segment ▼	<u>O</u> pen Cancel				
	✓ Always open in the design src for der						

- <u>Compile all of the VHDL files</u>. After adding the files to the project, you should see the files listed in the Design Browser as shown below (Pick the Files tab). Question marks (?) indicate that the files have not yet been compiled.
  - Select <u>Compile All</u> to compile the files. Each VHDL file should have a check mark next to it if it compiled successfully. If any files did not compile successfully, you must correct the errors before proceeding.





## Using Xilinx Vivado to synthesize the design into the Artix-7 FPGA

The steps involved here are nearly identical to those used in Lab 5, so refer to the tutorial:

• Tutorial 1 - Combinational Logic Circuits using Aldec Active-HDL and Xilinx Vivado

Some important differences:

- When we <u>add sources</u> to our design using the Xilinx software, we need to add all four VHDL files, as shown below.
- Ignore any warnings in Xilinx about excessive delay due to our ClockDivider circuit.
- The final bit file (or bin file) produced should have the name of the overall VHDL file, so when you generate the bitstream, look for the name (in this example): **CounterWithClock.bit** or **CounterWithClock.bin**

4	À New Project X						
Ad Sp file	Id Sources ecify HDL, ne on disk and :	i tlist, Block De add it to your p	sign, and IP files, or directori roject. You can also add an	ies containing th d create sources	iose files, to add to your pr s later.	oject	: Create a new source
	+  =	<b>1</b> = [ <b>1</b> ]					
		Index	Name	Library	HDL Source For		Location
	•	1	BCD_to_7segment.vhd	xil_defaultlib	Synthesis & Simulation	Ŧ	L:/BASYS3 - Vivado/Lab7/Cou
	٠	2	ClockDivider.vhd	xil_defaultlib	Synthesis & Simulation	~	L:/BASYS3 - Vivado/Lab7/Cou
	•	3	Counter_with_Clock.vhd	xil_defaultlib	Synthesis & Simulation		L:/BASYS3 - Vivado/Lab7/Cou
	•	4	Counter3Bit.vhd	xil_defaultlib	Synthesis & Simulation		L:/BASYS3 - Vivado/Lab7/Cou
	<		<u>A</u> dd Files	A <u>d</u> d Director	ies <u>C</u> reate File		>
(   	Scan and add RTL include files into project         ✓ Copy sources into project         ✓ Add sources from subdirectories         Target language:       VHDL         ✓ Simulator language:       Mixed						
	2			<	Back Next >		Einish Cancel

A	🝌 New Project					
A SI	dd Constraints (optiona becify or create constraint files	al) For physical and timing constra	aints.			4
	$ +_{\mathbf{z}}  =  +  +  $					
	Constraint File	Location				
	Basys3_Master_Lab7.xdc	L:\BASYS3 - Vivado\Lab7				
				7		
	✓ Copy constraints files into	Add Fil	les Create File			
(	?		< <u>B</u> ack	<u>N</u> ext >	<u>F</u> inish	Cancel