Lab 6: Design of Digital Stop Watch (*counter0to59*, *decoder0to59* and *lpm_counter*)

Objective:

* To design a Digital Stop Watch using concepts of digital system partitioning

Digital Stop Watch - counter0to59, decoder0to59 and lpm_counter

Open *stopwatch.bdf* and connect the *counter0to5* to *counter0to9* as shown in Figure 6.1 to obtain 0 to 59 counting and decoding



Figure 6.1 stopwatch.bdf with counter0to5 connected to counter0to9

2. To add Symbol: Altera Megafunction lpm_counter

Double-click on the blank space in the Graphic Editor window, or Click on the Symbol Tool icon \overrightarrow{D} in the toolbar. A pop-up box in Figure 6.2 will appear. Type lpm_counter in the search bar to locate the symbol and Click *OK*.



Figure 6.2 Choose *lpm_counter* from the library



Figure 6.3 Type of output file: AHDL

MegaWizard Plug-In Manager - LPM	_COUNTER [page 3 of 7]		
LPM_COU	NTER		
- Constant		<u>A</u> bout	<u>D</u> ocumentation
1 Parameter 2 EDA 3 Sumr Settings	nary		
General > General2 > Optional Inpu	nts >		
lpm_counter0 ↓ clock q[26.0]	Currently selected <u>d</u> evice famil	y: Cyclo Ma	one II 🛛 🔻
	How wide should the 'q' output t	ous be? 🛛	💌 bits
-	-What should the counter direc	tion be? —	
	Up only		
	 Down only 		
e d	 Create an 'updown' input p Coupts up: 0 coupts down 	ort to allow i	me to do both
n .	(1 Counts up, 0 Counts dov	wij	
-			
K Resource Usage			
	Cancel	< <u>B</u> ack	<u>N</u> ext > <u>F</u> inish

Figure 6.4 2^26 = 67,108,864 => Wide of 'q' output bus: 27 bits

MegaWizard Plug-In Manager - L	PM_COUNTER [page 4 of]	7]	- 🗆 X
🎝 LPM_CO	UNTER	About	
1 Parameter 2 EDA 3 S Settings	ummary		
Clockmodulus 5000000	Vhich type of counter d Plain binary Modulus, with a cour Do you want any optiona Clock Enable Count Enable	o you want? nt modulus of 500000 al additional ports? Carry-in Carry-out	
Resource Usage 27 lut + 27 reg	Cance	al < Back Vext >	Ejnish

Figure 6.5 *Modulus* with count 50,000,000 and with *Carry-out* port

MegaWizard Plug-In Manager - LPM_	COUNTER [page 5 of 7]	- 🗆 X
	TER	About
1 Parameter 2 EDA 3 Summa Settings General Ceneral Optional Inputs		
Ipm_counter0	- Do you want any optional in	nputs?
Clockmodulus 50000000 q[260]	Synchronous inputs	Asynchronous inputs -
	 Set Set to all 1's Set to 0 	 Set to all 1's Set to D
e		
Resource Usage 27 lut + 27 reg	Cancel	Rack Nexts Finish
Resource Usage	Cancel	< <u>B</u> ack <u>N</u> ext > <u>F</u> inish

Figure 6.6 Click Next

MegaWizard Plug-In Manager - LP	PM_COUNTER [page 6 of 7] EDA
LPM_COL	JNTER
1 Parameter 2 EDA 3 Sur Settings	mmary
Ipm_counterD up counter clockmodulus 5000000 q[260] cout	Simulation Libraries To properly simulate the generated design files, the following simulation model file(s) are needed F Description Ipm LPM megafunction simulation library
	Timing and resource estimation Generates a netlist for timing and resource estimation for this megafunction. If you are synthesizing your design with a third party synthesis tool using a
	Not all big and using your design with a minute stript synthesis 100, during a timing and resource estimation netlist can allow for better design optimization. Not all third-party synthesis tools support this feature - check with the tool vendor for complete support information. Note: Netlist generation can be a time-intensive process. The size of the design and the speed of your system affect the time it takes for netlist generation to complete.
2 Resource Usage 27 lut + 27 reg	Generate netlist Cancel < Back Next > Einish

Figure 6.7 Click Next

j Mega₩izard Plug-In Manager - LF	PM_COUNTER [page 7 of 7]	Summary 📃 🗙
	JNTER	
1 Parameter 2 EDA 3 Su Settings	mmary	
Ipm_counter0 up counter clockmodulus 50000000 q[260] x cout	Turn on the files you wish to genera automatically generated, and a red Finish to generate the selected files subsequent MegaWizard Plug-In Ma The MegaWizard Plug-In Manager of directory: DIPLOMA/Year2/EG2163/g8_1/digit	ate. A gray checkmark indicates a file that is checkmark indicates an optional file. Click . The state of each checkbox is maintained in inager sessions. reates the selected files in the following al/stopwatch/
	File	Description
	Vilom counter() tdf	Variation file
	V lpm_counter0.inc	AHDL Include file
-	D lpm_counter0.cmp	VHDL component declaration file
	☑ lpm_counter0.bsf	Quartus II symbol file
	Ipm_counter0_inst.tdf	Instantiation template file
-	☑ Ipm_counter0_waveforms.html	Sample waveforms in summary
-	Ipm_counter0_wave*.jpg	Sample waveform file(s)
-		
1		
-		
n		
Resource Usage		
27 lut + 27 reg		Cancel < <u>B</u> ack <u>N</u> ext > <u>F</u> inish

Figure 6.8 Click Finish

ede ac	Quartus II IP Files	
x fi n e	When you create an Altera IP variation, a Quartus II IP File is generated. Quartus II IP Files are used to represent the Altera IP in your design. Do you want to add the Quartus II IP File to the project?	
	(Note: Turning on this option permanently suppresses this dialog box. You can change this setting in the Options dialog box)	
	Yes No	

Figure 6.9 Click Yes

2. Complete the circuit with *lpm_counter* as shown in Figure 6.10. Click *File > Save*.



Figure 6.10 Circuit with *lpm_counter*

3. Pin Assignments

During previous compilations, the Quartus II Compiler was free to choose any pins on the selected FPGA to serve as inputs and outputs. However, the training board (DE1) has hardwired connections between the FPGA pins and the other components on the board. The DE1 board has fixed pin assignments. A useful Quartus II feature allows the user to import the pin assignments from a special file format (*.csv*). You can import pin assignments by choosing *Assignments* > *Import Assignments*. This opens the dialogue in Figure 6.11 to select the file to import.



Figure 6.11 Importing the pin assignments

Browse to select the desired file *stopwatch.csv*. Click **Open**.

cout clk q3	Select File
Import Assignments	Look in: 🔁 stopwatch 🔽 🖻 📓 📸 📰 📰
Specify the source and categories of assignments to import.	db incremental_db
Eile name: Categories	stopwatch.csv
Copy existing assignments into stopwatch.qsf.bak before importing	stopwatch.gsr
OK. Cancel	
	File <u>n</u> ame: stopwatch
<pre>pwatch -c stopwatchtiming_analysis_only</pre>	Files of type: Import Files (".qsf;".esf;".acf;".csv;".txt;".sdc) Cancel

Figure 6.12 Select stopwatch.csv

Click **OK** to confirm pins assignments file *stopwatch.csv*.

nport Assig	nments		2
Specify the s	ource and categories of assignm	ents to import.	
<u>F</u> ile name:	/DIPLOMA/Year2/EG2163/	g8_1/digital/stopwatc	Categories
🔽 Copy exis	ting assignments into stopwatch	.qsf.bak before importing	Advanced
		04	

Figure 6.13 Confirm pin assignments file stopwatch.csv

4. Compile the circuit with pin assignments as shown in Figure 6.14 and ensure no error (Ignore warning/s).



Figure 6.14 Compile the circuit with pin assignments

5. Programming and Configuring the FPGA Device

The FPGA device must be programmed and configured to implement the designed circuit. The required configuration file is generated by the Quartus II Compiler's Assembler module. Altera's DE1 board allows the configuration to be done in two different ways, known as **JTAG** (Joint Test Action Group) and **AS** (Active Serial) modes. The configuration data is transferred from the host computer (which runs the Quartus II software) to the board by means of a cable that connects a USB port on the host computer to the leftmost USB connector on the board. Before using the board, make sure that the USB cable is properly connected and turn on the power supply switch on the board.

The choice between the two modes is made by the **RUN/PROG** switch on the DE1 board. The **RUN** position selects the **JTAG** mode, while the **PROG** position selects the **AS** mode. In the **JTAG** mode, the configuration data is loaded directly into the FPGA device. JTAG defined a simple way for testing digital circuits and loading data into them, which became an IEEE standard. FPGA configured by JTAG will retain its configuration as long as the power remains turned on. The configuration information is lost when the power is turned off.

Here we set JTAG programming. Select *Tools > Programmer* as shown in Figure 6.15.

8 🏽 🖉 🗑 🛜		
≪ Quartus II - /DIPLOMA/Year2/EG2163/g8_1/dig	gital/stopwatch/stopwatch - stopwatch - [st Tools Window Help	ot
□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □	Run EDA Simulation Tool 3 Run EDA Timing Analysis Tool Launch EDA Simulation Library Compiler Launch Design Space Explorer 3 Image: Image of the system of the s	
▲ Hierarchy È Files ௴ Design Units Tasks Flow: Early Timing Estimate with Synthesis	 SignalT ap II Logic Analyzer In-System Memory Content Editor Logic Analyzer Interface Editor In-System Sources and Probes Editor SignalProbe Pins 	
Task I ✓ ► Analysis & Synthesis Image Image		· <mark> </mark> · · · · · · · · · · · ·
	Customize Block Editor Options for Block Editor	

Figure 6.15 Programmer tools

It is necessary to specify the programming hardware (**USB Blaster**) and the mode (**JTAG**) that should be used. If it is not automatically detected, Click the *Hardware Setup* button and select the *USB-Blaster* in the pop-up window and select **JTAG** in the Mode box as shown in Figure 6.16 and Figure 6.17.

0000	\$						(🌖 🧠 A	pplicatio	ns Pla	ces S	ystem
💾 Quartus II - 7DI	PLOMA/Year2/EG21	63/g8_1/digital/stopw	atch/stopwate	ch - stopwatch	- [stopwatch	.cdf]						
<u>File E</u> dit P <u>r</u> ocessir	ng <u>T</u> ools <u>W</u> indow											
🔔 Hardware Setup	No Hardware										N	fode: JTAG 🗾 💌
Enable real-time IS	δP to allow background p	rogramming (for MAX II dev	vices)									
M Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP	
Stop	stopwatch.sof	EP2C20F484	00186696	FFFFFFF								•
Auto Detect												_
X Delete			Hardwar	e Setup							×	1
Add Ella			Hardwa	are Settings JTA	.G Settings							
Add file			Selec	t a programming h	ardware setup t	o use whe	n program	ming device	es. This prog	ramming		
Change File			naruw	are serup applies	Uniy to the care	ent program	nmer wind	0₩.				
Save File			Currer	ntly selected hards	ware: USB	-Blaster [U	SB 5-1.2]					
😂 Add Device			Ava	ilable hardware ite	ems:				-			
ф Up				ardware B-Blaster	L	erver ocal	Port USB	5-1.2	Add H	lardware	<u> </u>	
Down									Remov	ve Hardwa	re	
											ose	
											000	

Figure 6.16 If Hardware Setup not detected automatically

8 \$\$	}						!	A 🧑 🧑	pplicatio	ns Pla	ces Sy	stem	
💾 Quartus II - 7DI	PLOMA/Year2/EG21	63/g8_1/digital/stop	watch/stopwate	ch - stopwatcl	h - [stopwatch	.cdf]							
<u>File E</u> dit P <u>r</u> ocessir	ng <u>T</u> ools <u>W</u> indow												
🔔 Hardware Setup	USB-Blaster [USB 5-	1.2]									Mo	de: JTAG	•
Enable real-time IS	SP to allow background p	rogramming (for MAX II de	evices)										
🏴 Start	File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP		
Stop	stopwatch.sof	EP2C20F484	00186696	FFFFFFF	✓								
🙀 Auto Detect													
X Delete													
🍰 Add File													
🕒 Change File													
💾 Save File													
😂 Add Device													
ф Up													
Down													

Figure 6.17 Correct Hardware Setup window

Observe that the configuration file *stopwatch.sof* is listed in Figure 6.17. This is a binary file produced by the Compiler's Assembler module, which contains the data needed to configure the FPGA device. The extension.*sof* stands for **SRAM Object File**. Note also that the device selected is **EP2C20F484C7**, which is the FPGA device used on the DE1 board. Make sure the Program/Configure check box is ticked, Click *Start*. Having downloaded the configuration data into the FPGA device, you can now test the implemented circuit. If you need to make change/s to the design, first close the Programmer window. Then make the desired change/s, compile and reprogram the board. Repeat the process until you obtain the desired results.