

Lab Practical Objectives

Overview

ECPE 174 focuses on the practical engineering aspects of advanced digital design, preparing students to work on large-scale digital projects in industry and academic settings. Given this focus, the lab practical provides an ideal metric of a student's ability to use the design tools focused on in the class. This report provides an outline of the format of the practical, the topics covered by the practical, grading, and a summary of the specific objectives the exam covers.

Practical Format

The lab practical occurs on Tuesday, October 11 in Chambers 115 from 2-5pm. Students have the entire lab period to work on the practical although the design of it will ensure that those students who know the material finish in 1.5 hours.

Each student works independently on a computer in the lab with a Cyclone 2 board. Students can utilize any books, notes, or prior work during the exam. As everyone will have their own Internet-accessible computer, students can also access online resources and their *u* drive. Students cannot talk to other students, use their mobile phone, or chat online with anyone. Additionally, students cannot view or listen to any material requiring headphones. Anyone incorrectly utilizing materials or providing any distractions will receive one warning. Any further violations will result in the end of the exam for that student and dismissal from the lab.

Practical Topics and Coverage

The lab practical emphasizes those skills utilized in lab. However, in testing those skills, the exam can require design topics discussed in class such as finite state machine design (Mealy and Moore) and synchronizing logic.

All labs to date have required the use of VHDL coding; the practical may do so as well. This could include utilizing clock divider logic as well as testbenches. The exam will provide any extra code although students should know how to include such code in their design.

Labs have focused on Quartus and Cyclone 2 boards. Understanding Quartus consists of four areas: simulation, implementation verification, analysis, and programming. For simulation, students should know how to create a waveform simulation file, how to test a reasonable number of input cases, how to examine internal signals for debugging purposes, how to perform both functional and timing simulation, and how to determine the simulation coverage according to Quartus.

The diagram illustrates the state transitions of a 4-bit shift register. The states are labeled as **reset**, **s1**, **s2**, **s3**, and **s4**. The transitions are triggered by the **data** and **clock** signals.

Verilog Code Snippets:

```

11 // 4-bit shift register
12
13 --reset
14 state = OFFOFF(01_1clocked);
15
16 --decrement the decrement
17 decrement = OFFOFF(01_0dec);
18
19 --decrement the decrement
20 decrement = OFFOFF(decrement);
21
22 --s1
23 --s1_1clocked is 1 shift 1 clocked
24 s1_1clocked = OFFOFF(s1_0clk, 01_0 . . . . .);
25
26 --s1_0dec is 1 shift 1 clocked (0 dec)
27 s1_0dec = OFFOFF(s1_0clk, 01_0 . . . . .);
28
29 --decrement the decrement
30 decrement = OFFOFF(s1_0clk, 01_0 . . . . .);
31
32 --s2
33 --s2_1clocked is 1 shift 1 clocked (0 dec)
34 s2_1clocked = OFFOFF(s2_0clk, 01_0 . . . . .);
35
36 --s2_0dec is 1 shift 1 clocked (0 dec)
37 s2_0dec = OFFOFF(s2_0clk, 01_0 . . . . .);
38
39 --decrement the decrement
40 decrement = OFFOFF(s2_0clk, 01_0 . . . . .);
41
42 --s3
43 --s3_1clocked is 1 shift 1 clocked (0 dec)
44 s3_1clocked = OFFOFF(s3_0clk, 01_0 . . . . .);
45
46 --s3_0dec is 1 shift 1 clocked (0 dec)
47 s3_0dec = OFFOFF(s3_0clk, 01_0 . . . . .);
48
49 --decrement the decrement
50 decrement = OFFOFF(s3_0clk, 01_0 . . . . .);
51
52 --s4
53 --s4_1clocked is 1 shift 1 clocked (0 dec)
54 s4_1clocked = OFFOFF(s4_0clk, 01_0 . . . . .);
55
56 --s4_0dec is 1 shift 1 clocked (0 dec)
57 s4_0dec = OFFOFF(s4_0clk, 01_0 . . . . .);
58
59 --decrement the decrement
60 decrement = OFFOFF(s4_0clk, 01_0 . . . . .);
61
62 --s5
63 --s5_1clocked is 1 shift 1 clocked (0 dec)
64 s5_1clocked = OFFOFF(s5_0clk, 01_0 . . . . .);
65
66 --s5_0dec is 1 shift 1 clocked (0 dec)
67 s5_0dec = OFFOFF(s5_0clk, 01_0 . . . . .);
68
69 --decrement the decrement
70 decrement = OFFOFF(s5_0clk, 01_0 . . . . .);
71
72 --s6
73 --s6_1clocked is 1 shift 1 clocked (0 dec)
74 s6_1clocked = OFFOFF(s6_0clk, 01_0 . . . . .);
75
76 --s6_0dec is 1 shift 1 clocked (0 dec)
77 s6_0dec = OFFOFF(s6_0clk, 01_0 . . . . .);
78
79 --decrement the decrement
80 decrement = OFFOFF(s6_0clk, 01_0 . . . . .);
81
82 --s7
83 --s7_1clocked is 1 shift 1 clocked (0 dec)
84 s7_1clocked = OFFOFF(s7_0clk, 01_0 . . . . .);
85
86 --s7_0dec is 1 shift 1 clocked (0 dec)
87 s7_0dec = OFFOFF(s7_0clk, 01_0 . . . . .);
88
89 --decrement the decrement
90 decrement = OFFOFF(s7_0clk, 01_0 . . . . .);
91
92 --s8
93 --s8_1clocked is 1 shift 1 clocked (0 dec)
94 s8_1clocked = OFFOFF(s8_0clk, 01_0 . . . . .);
95
96 --s8_0dec is 1 shift 1 clocked (0 dec)
97 s8_0dec = OFFOFF(s8_0clk, 01_0 . . . . .);
98
99 --decrement the decrement
100 decrement = OFFOFF(s8_0clk, 01_0 . . . . .);
101
102 --s9
103 --s9_1clocked is 1 shift 1 clocked (0 dec)
104 s9_1clocked = OFFOFF(s9_0clk, 01_0 . . . . .);
105
106 --s9_0dec is 1 shift 1 clocked (0 dec)
107 s9_0dec = OFFOFF(s9_0clk, 01_0 . . . . .);
108
109 --decrement the decrement
110 decrement = OFFOFF(s9_0clk, 01_0 . . . . .);
111
112 --s10
113 --s10_1clocked is 1 shift 1 clocked (0 dec)
114 s10_1clocked = OFFOFF(s10_0clk, 01_0 . . . . .);
115
116 --s10_0dec is 1 shift 1 clocked (0 dec)
117 s10_0dec = OFFOFF(s10_0clk, 01_0 . . . . .);
118
119 --decrement the decrement
120 decrement = OFFOFF(s10_0clk, 01_0 . . . . .);
121
122 --s11
123 --s11_1clocked is 1 shift 1 clocked (0 dec)
124 s11_1clocked = OFFOFF(s11_0clk, 01_0 . . . . .);
125
126 --s11_0dec is 1 shift 1 clocked (0 dec)
127 s11_0dec = OFFOFF(s11_0clk, 01_0 . . . . .);
128
129 --decrement the decrement
130 decrement = OFFOFF(s11_0clk, 01_0 . . . . .);
131
132 --s12
133 --s12_1clocked is 1 shift 1 clocked (0 dec)
134 s12_1clocked = OFFOFF(s12_0clk, 01_0 . . . . .);
135
136 --s12_0dec is 1 shift 1 clocked (0 dec)
137 s12_0dec = OFFOFF(s12_0clk, 01_0 . . . . .);
138
139 --decrement the decrement
140 decrement = OFFOFF(s12_0clk, 01_0 . . . . .);
141
142 --s13
143 --s13_1clocked is 1 shift 1 clocked (0 dec)
144 s13_1clocked = OFFOFF(s13_0clk, 01_0 . . . . .);
145
146 --s13_0dec is 1 shift 1 clocked (0 dec)
147 s13_0dec = OFFOFF(s13_0clk, 01_0 . . . . .);
148
149 --decrement the decrement
150 decrement = OFFOFF(s13_0clk, 01_0 . . . . .);
151
152 --s14
153 --s14_1clocked is 1 shift 1 clocked (0 dec)
154 s14_1clocked = OFFOFF(s14_0clk, 01_0 . . . . .);
155
156 --s14_0dec is 1 shift 1 clocked (0 dec)
157 s14_0dec = OFFOFF(s14_0clk, 01_0 . . . . .);
158
159 --decrement the decrement
160 decrement = OFFOFF(s14_0clk, 01_0 . . . . .);
161
162 --s15
163 --s15_1clocked is 1 shift 1 clocked (0 dec)
164 s15_1clocked = OFFOFF(s15_0clk, 01_0 . . . . .);
165
166 --s15_0dec is 1 shift 1 clocked (0 dec)
167 s15_0dec = OFFOFF(s15_0clk, 01_0 . . . . .);
168
169 --decrement the decrement
170 decrement = OFFOFF(s15_0clk, 01_0 . . . . .);
171
172 --s16
173 --s16_1clocked is 1 shift 1 clocked (0 dec)
174 s16_1clocked = OFFOFF(s16_0clk, 01_0 . . . . .);
175
176 --s16_0dec is 1 shift 1 clocked (0 dec)
177 s16_0dec = OFFOFF(s16_0clk, 01_0 . . . . .);
178
179 --decrement the decrement
180 decrement = OFFOFF(s16_0clk, 01_0 . . . . .);
181
182 --s17
183 --s17_1clocked is 1 shift 1 clocked (0 dec)
184 s17_1clocked = OFFOFF(s17_0clk, 01_0 . . . . .);
185
186 --s17_0dec is 1 shift 1 clocked (0 dec)
187 s17_0dec = OFFOFF(s17_0clk, 01_0 . . . . .);
188
189 --decrement the decrement
190 decrement = OFFOFF(s17_0clk, 01_0 . . . . .);
191
192 --s18
193 --s18_1clocked is 1 shift 1 clocked (0 dec)
194 s18_1clocked = OFFOFF(s18_0clk, 01_0 . . . . .);
195
196 --s18_0dec is 1 shift 1 clocked (0 dec)
197 s18_0dec = OFFOFF(s18_0clk, 01_0 . . . . .);
198
199 --decrement the decrement
200 decrement = OFFOFF(s18_0clk, 01_0 . . . . .);
201
202 --s19
203 --s19_1clocked is 1 shift 1 clocked (0 dec)
204 s19_1clocked = OFFOFF(s19_0clk, 01_0 . . . . .);
205
206 --s19_0dec is 1 shift 1 clocked (0 dec)
207 s19_0dec = OFFOFF(s19_0clk, 01_0 . . . . .);
208
209 --decrement the decrement
210 decrement = OFFOFF(s19_0clk, 01_0 . . . . .);
211
212 --s20
213 --s20_1clocked is 1 shift 1 clocked (0 dec)
214 s20_1clocked = OFFOFF(s20_0clk, 01_0 . . . . .);
215
216 --s20_0dec is 1 shift 1 clocked (0 dec)
217 s20_0dec = OFFOFF(s20_0clk, 01_0 . . . . .);
218
219 --decrement the decrement
220 decrement = OFFOFF(s20_0clk, 01_0 . . . . .);
221
222 --s21
223 --s21_1clocked is 1 shift 1 clocked (0 dec)
224 s21_1clocked = OFFOFF(s21_0clk, 01_0 . . . . .);
225
226 --s21_0dec is 1 shift 1 clocked (0 dec)
227 s21_0dec = OFFOFF(s21_0clk, 01_0 . . . . .);
228
229 --decrement the decrement
230 decrement = OFFOFF(s21_0clk, 01_0 . . . . .);
231
232 --s22
233 --s22_1clocked is 1 shift 1 clocked (0 dec)
234 s22_1clocked = OFFOFF(s22_0clk, 01_0 . . . . .);
235
236 --s22_0dec is 1 shift 1 clocked (0 dec)
237 s22_0dec = OFFOFF(s22_0clk, 01_0 . . . . .);
238
239 --decrement the decrement
240 decrement = OFFOFF(s22_0clk, 01_0 . . . . .);
241
242 --s23
243 --s23_1clocked is 1 shift 1 clocked (0 dec)
244 s23_1clocked = OFFOFF(s23_0clk, 01_0 . . . . .);
245
246 --s23_0dec is 1 shift 1 clocked (0 dec)
247 s23_0dec = OFFOFF(s23_0clk, 01_0 . . . . .);
248
249 --decrement the decrement
250 decrement = OFFOFF(s23_0clk, 01_0 . . . . .);
251
252 --s24
253 --s24_1clocked is 1 shift 1 clocked (0 dec)
254 s24_1clocked = OFFOFF(s24_0clk, 01_0 . . . . .);
255
256 --s24_0dec is 1 shift 1 clocked (0 dec)
257 s24_0dec = OFFOFF(s24_0clk, 01_0 . . . . .);
258
259 --decrement the decrement
260 decrement = OFFOFF(s24_0clk, 01_0 . . . . .);
261
262 --s25
263 --s25_
```

The screenshot displays the Cadence Virtuoso schematic editor interface. On the left, the 'Hierarchy List' pane shows the project structure, including 'testbench', 'Instances', 'Primitives', 'Pins', and 'Nets'. The main workspace shows a schematic diagram titled 'Page Title: testbench'. The circuit includes several components: a 'resyncDev' block, two 'process' blocks labeled 'process_0-3' and 'process_0-1', a 'devBTest' block, and a 'lab5 labConnect' block. Inputs on the left are 'send', 'clk', and 'testWrite'. The circuit uses various logic gates and multiplexers to connect these components. Outputs on the right are labeled 'devAView', 'rise Send', and 'devBView'.

Quartus also provides several analysis tools. Figures 3 and 4 show the two different tools we use regularly in lab: the Fitter Summary and Timing Summary, respectively. Students should know where to find this information.

Fitter Status	Successful - Sat Sep 18 01:10:29 2010
Quartus II Version	9.1 Build 304 01/25/2010 SP 1 SJ Web Edition
Revision Name	lab3
Top-level Entity Name	lab3
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	24 / 18,752 (< 1 %)
Total combinational functions	15 / 18,752 (< 1 %)
Dedicated logic registers	20 / 18,752 (< 1 %)
Total registers	20
Total pins	5 / 315 (2 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Type	Slack	Required Time	Actual Time	From	To	From Clock	To Clock	Failed Paths
Worst-case tsu	N/A	None	3.508 ns	send	devBtest	--	clk	0
Worst-case tco	N/A	None	7.408 ns	lab5.labConnectIriseSend	riseSend	clk	--	0
Worst-case th	N/A	None	-2.828 ns	testWrite	devBtest	--	clk	0
Clock Setup: 'clk'	N/A	None	Restricted to 380.08 MHz (period = 2.631 ns)	lab5.labConnectIdevBFF2	lab5.labConnectIdevA	clk	clk	0
Total number of failed paths								0

Figure 4: Timing Summary.

Finally, students should know how to implement a design on the Cyclone 2 board. This entails pin assignment (requiring determining the correct pins for different on-board components), programming, and testing. Students should know the functionality of the different components including whether the component uses active high or active low logic. Students will not need to connect their board to any external hardware.

Grading

The lab practical counts as 10% of the course grade and will consist of 100 points. Each item will indicate its point value. Process counts so students may need to explain steps taken to achieve a goal. If any problems (such as the design does not work) or anomalies arise (for example, Quartus does not provide a state diagram or uses a large number of registers to implement the design), students should explain their steps to solve and to understand them. In these cases, we will only give partial credit if students document their debugging and thought processes.

Summary

This report outlined the structure, topics, and grading of the lab practical. Given its importance, we conclude by summarizing the objectives:

- Design VHDL code to solve a given problem
- Include code in higher level testbenches or add lower level components
- Simulate design with functional and timing simulation, providing a specified coverage level and testing the majority of input combinations
- View the synthesized design through equations, state machines, and RTL
- Analyze the timing and space requirements of the design
- Program a Cyclone board with the design and correctly implement design on board, using proper debugging skills