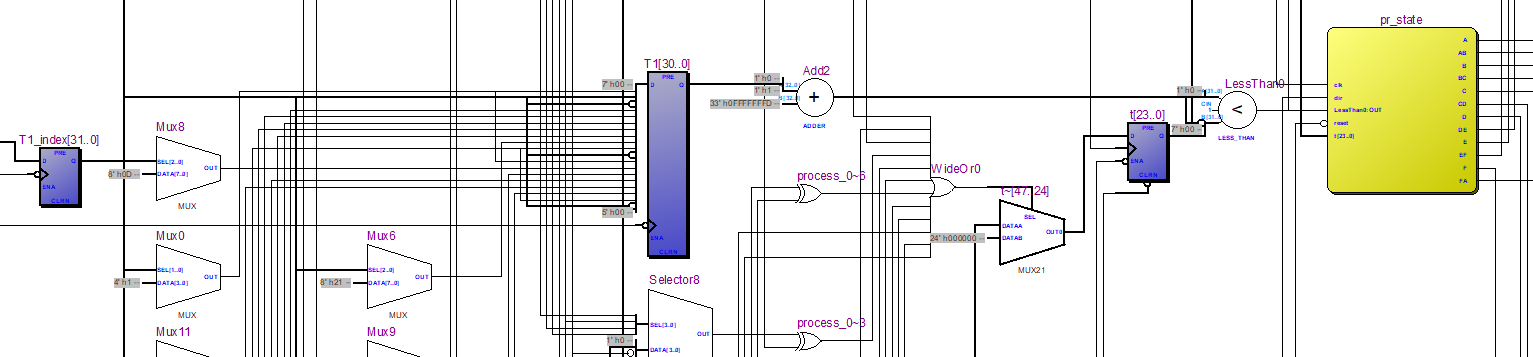
David Elliott

Evan Yeh

EE125 Homework #5

Exercise 9.6:

1. Block Diagram:
2. State transition diagrams:
3. T1 is registered since there are different speeds for the light rotator. With the largest value of T1 as 12,500,000, T1 should be ceil(log(12500000)) = 24 bits long, inferring 24 dffs. t, which counts up to T1 will also have to hold a value as large as 12,500,000 (minus 1), requiring 24 bits, 24 dffs. T1\_index is essentially a sequentially encoded state for the FSM of controlling the speed. There are 6 speeds to loop through, so T1\_index goes from 0 to 5, requiring ceil(log(6)) = 3 bits, 3 dffs. Then assuming sequential encoding for the main FSM of light rotator states, there are 12 states, so ceil(log2(12)) = 4 bits are needed, 4 dffs. In total there should be 24 + 24 + 3 + 4 = 55 dffs.
4. Looking at the RTL view, T1 ended up inferring 31 dffs, as did T1\_index. The encoding for the FSM states turned out to be one hot, so 12 bits, 12 dffs, were inferred. The rest of the predictions match.
5. Code:

001 -----------------------------------------------------------

002 **library** ieee**;**

003 **use** ieee**.**std\_logic\_1164**.all;**

004 -----------------------------------------------------------

005 **entity** light\_rotator **is**

006 **port** **(**

007 stp**,** clk**,** rst**,** dir**,** spd**:** **in** std\_logic**;**

008 ssd**:** **out** std\_logic\_vector**(**6 **downto** 0**));**

009 **end** **entity;**

010 -----------------------------------------------------------

011 **architecture** moore\_fsm **of** light\_rotator **is**

012

013 --FSM-related declarations:

014 **type** state **is** **(**A**,** AB**,** B**,** BC**,** C**,** CD**,** D**,** DE**,** E**,** EF**,** F**,** FA**);**

015 **signal** pr\_state**,** nx\_state**:** state**;**

016

017 --Timer-related declarations:

018 **type** int\_array **is** **array** **(**integer **range** **<>)** **of** integer**;**

019 **constant** T1\_TABLE**:** int\_array**(**0 **to** 5**)** **:=** **(**12\_500\_000**,** -- 250ms @fclk=50Mhz

020 9\_000\_000**,** -- 180ms @fclk=50Mhz

021 6\_500\_000**,** -- 130ms @fclk=50Mhz

022 5\_000\_000**,** -- 100ms @fclk=50Mhz

023 3\_500\_000**,** -- 70ms @fclk=50Mhz

024 2\_000\_000**);** -- 40ms @fclk=50Mhz

025 **signal** T1**:** natural **:=** T1\_TABLE**(**0**);** --120ms @ fclk=50MHz

026 **constant** T2**:** natural **:=** 1\_000\_000**;** --20ms @ fclk=50MHz

027 **constant** tmax**:** natural **:=** T1**-**1**;** --tmax ≥ max(T1,T2)-1

028 **signal** t**:** natural **range** 0 **to** tmax**;**

029 **signal** debounced**:** std\_logic **:=** '0'**;** --signal for debounced spd switch press

030 **signal** T1\_index**:** integer **:=** 0**;** --index for table of T1

031

032 **begin**

033 debouncer**:** **entity** work**.**switch\_debouncer**(**switch\_debouncer**)** **port** **map(**spd**,** debounced**,** clk**);**

034

035 --Timer (using strategy #1):

036 **process** **(**clk**,** rst**,** stp**)**

037 **begin**

038 **if** rst**=**'0' **then**

039 t **<=** 0**;**

040 **elsif** **rising\_edge(**clk**)** and stp**=**'0' **then**

041 **if** pr\_state **/=** nx\_state **then**

042 t **<=** 0**;**

043 **elsif** t **/=** tmax **then**

044 t **<=** t **+** 1**;**

045 **end** **if;**

046 **end** **if;**

047 **end** **process;**

048

049 --Speed controller

050 **process** **(**debounced**,** spd**,** clk**)**

051 **begin**

052 **if** **falling\_edge(**debounced**)** **then**

053 T1\_index **<=** T1\_index **+** 1**;**

054 **if** T1\_index **=** 5 **then**

055 T1\_index **<=** 0**;**

056 **end** **if;**

057 T1 **<=**T1\_TABLE**(**T1\_index**);**

058 **end** **if;**

059 **end** **process;**

060

061 --FSM state register:

062 **process** **(**clk**,** rst**)**

063 **begin**

064 **if** rst**=**'0' **then**

065 pr\_state **<=** A**;**

066 **elsif** **rising\_edge(**clk**)** **then**

067 pr\_state **<=** nx\_state**;**

068 **end** **if;**

069 **end** **process;**

070

071 --FSM combinational logic:

072 **process** **(all)**

073 **begin**

074 **case** pr\_state **is**

075 **when** A **=>**

076 ssd **<=** "0111111"**;**

077 **if** dir **=** '1' **then**

078 **if** t **>=** T1**-**1 **then** -- or t=T1-1

079 nx\_state **<=** AB**;**

080 **else**

081 nx\_state **<=** A**;**

082 **end** **if;**

083 **else**

084 **if** t **>=** T1**-**1 **then** -- or t=T1-1

085 nx\_state **<=** FA**;**

086 **else**

087 nx\_state **<=** A**;**

088 **end** **if;**

089 **end** **if;**

090 **when** AB **=>**

091 ssd **<=** "0011111"**;**

092 **if** dir **=** '1' **then**

093 **if** t **>=** T2**-**1 **then** -- or t=T2-1

094 nx\_state **<=** B**;**

095 **else**

096 nx\_state **<=** AB**;**

097 **end** **if;**

098 **else**

099 **if** t **>=** T2**-**1 **then** -- or t=T2-1

100 nx\_state **<=** A**;**

101 **else**

102 nx\_state **<=** AB**;**

103 **end** **if;**

104 **end** **if;**

105 **when** B **=>**

106 ssd **<=** "1011111"**;**

107 **if** dir **=** '1' **then**

108 **if** t **>=** T1**-**1 **then** -- or t=T1-1

109 nx\_state **<=** BC**;**

110 **else**

111 nx\_state **<=** B**;**

112 **end** **if;**

113 **else**

114 **if** t **>=** T1**-**1 **then** -- or t=T1-1

115 nx\_state **<=** AB**;**

116 **else**

117 nx\_state **<=** B**;**

118 **end** **if;**

119 **end** **if;**

120 **when** BC **=>**

121 ssd **<=** "1001111"**;**

122 **if** dir **=** '1' **then**

123 **if** t **>=** T2**-**1 **then** -- or t=T2-1

124 nx\_state **<=** C**;**

125 **else**

126 nx\_state **<=** BC**;**

127 **end** **if;**

128 **else**

129 **if** t **>=** T2**-**1 **then** -- or t=T2-1

130 nx\_state **<=** B**;**

131 **else**

132 nx\_state **<=** BC**;**

133 **end** **if;**

134 **end** **if;**

135 **when** C **=>**

136 ssd **<=** "1101111"**;**

137 **if** dir **=** '1' **then**

138 **if** t **>=** T1**-**1 **then** -- or t=T1-1

139 nx\_state **<=** CD**;**

140 **else**

141 nx\_state **<=** C**;**

142 **end** **if;**

143 **else**

144 **if** t **>=** T1**-**1 **then** -- or t=T1-1

145 nx\_state **<=** BC**;**

146 **else**

147 nx\_state **<=** C**;**

148 **end** **if;**

149 **end** **if;**

150 **when** CD **=>**

151 ssd **<=** "1100111"**;**

152 **if** dir **=** '1' **then**

153 **if** t **>=** T2**-**1 **then** -- or t=T2-1

154 nx\_state **<=** D**;**

155 **else**

156 nx\_state **<=** CD**;**

157 **end** **if;**

158 **else**

159 **if** t **>=** T2**-**1 **then** -- or t=T2-1

160 nx\_state **<=** C**;**

161 **else**

162 nx\_state **<=** CD**;**

163 **end** **if;**

164 **end** **if;**

165 **when** D **=>**

166 ssd **<=** "1110111"**;**

167 **if** dir **=** '1' **then**

168 **if** t **>=** T1**-**1 **then** -- or t=T1-1

169 nx\_state **<=** DE**;**

170 **else**

171 nx\_state **<=** D**;**

172 **end** **if;**

173 **else**

174 **if** t **>=** T1**-**1 **then** -- or t=T1-1

175 nx\_state **<=** CD**;**

176 **else**

177 nx\_state **<=** D**;**

178 **end** **if;**

179 **end** **if;**

180 **when** DE **=>**

181 ssd **<=** "1110011"**;**

182 **if** dir **=** '1' **then**

183 **if** t **>=** T2**-**1 **then** -- or t=T2-1

184 nx\_state **<=** E**;**

185 **else**

186 nx\_state **<=** DE**;**

187 **end** **if;**

188 **else**

189 **if** t **>=** T2**-**1 **then** -- or t=T2-1

190 nx\_state **<=** D**;**

191 **else**

192 nx\_state **<=** DE**;**

193 **end** **if;**

194 **end** **if;**

195 **when** E **=>**

196 ssd **<=** "1111011"**;**

197 **if** dir **=** '1' **then**

198 **if** t **>=** T1**-**1 **then** -- or t=T1-1

199 nx\_state **<=** EF**;**

200 **else**

201 nx\_state **<=** E**;**

202 **end** **if;**

203 **else**

204 **if** t **>=** T1**-**1 **then** -- or t=T1-1

205 nx\_state **<=** DE**;**

206 **else**

207 nx\_state **<=** E**;**

208 **end** **if;**

209 **end** **if;**

210 **when** EF **=>**

211 ssd **<=** "1111001"**;**

212 **if** dir **=** '1' **then**

213 **if** t **>=** T2**-**1 **then** -- or t=T2-1

214 nx\_state **<=** F**;**

215 **else**

216 nx\_state **<=** EF**;**

217 **end** **if;**

218 **else**

219 **if** t **>=** T2**-**1 **then** -- or t=T2-1

220 nx\_state **<=** E**;**

221 **else**

222 nx\_state **<=** EF**;**

223 **end** **if;**

224 **end** **if;**

225 **when** F **=>**

226 ssd **<=** "1111101"**;**

227 **if** dir **=** '1' **then**

228 **if** t **>=** T1**-**1 **then** -- or t=T1-1

229 nx\_state **<=** FA**;**

230 **else**

231 nx\_state **<=** F**;**

232 **end** **if;**

233 **else**

234 **if** t **>=** T1**-**1 **then** -- or t=T1-1

235 nx\_state **<=** EF**;**

236 **else**

237 nx\_state **<=** F**;**

238 **end** **if;**

239 **end** **if;**

240 **when** FA **=>**

241 ssd **<=** "0111101"**;**

242 **if** dir **=** '1' **then**

243 **if** t **>=** T2**-**1 **then** -- or t=T2-1

244 nx\_state **<=** A**;**

245 **else**

246 nx\_state **<=** FA**;**

247 **end** **if;**

248 **else**

249 **if** t **>=** T2**-**1 **then** -- or t=T2-1

250 nx\_state **<=** F**;**

251 **else**

252 nx\_state **<=** FA**;**

253 **end** **if;**

254 **end** **if;**

255 **end** **case;**

256 **end** **process;**

257

258 **end** **architecture;**

259 -----------------------------------------------------------