```
1   Library ieee;
2
3   use ieee.std_logic_1164.all;
4
5   -- This program is written for the 2017 Spring DLD class at Grove City College. The goal is
    to create a system which polls classrooms around campus to clooect information from them.
6   -- Information collected is ClassroomInUse, LightsAreOn, and ProjectorIsOn.
7   -- ClassroomInUse being a 1 represents that the classroom is being used, all of these
    signals come from sensors in the classroom that are explained in the writup attached with
    this code.
8
9   -- This VHDL program is written by Theo Stangebye, stangebyeTO1@gcc.edu, April 2017.
10
11  -- The classroomController Entity is the component which is being polled. Each classroom
    will have a classroomController which communicates with at campusController when it's id is
    selected by the campus controller for polling.  The classroomController will then connect
    to the communication line and transmit its data in serial to the campus controller.
12  entity ClassroomController is -- we'll ue classroomControllerHardware as our actual
    classroomController Simulator
13  port( gpio : inout std_logic_vector(39 downto 0);
14        ledr : out std_logic_vector(17 downto 0);
15        ledg : out std_logic_vector(8 downto 0);
16        sw : in std_logic_vector(17 downto 0);
17        key : in std_logic_vector (3 downto 0)
18     );
19  end ClassroomController;
20
21  -- Our architecture instantiates 4 classroomController components, simulating 4 classrooms
    which will be polled by a campusController
22  architecture a of ClassroomController is
23
24     -- Declare the components that we created below.
25     Component ClassroomControllerHardware is -- ClassroomControllerHardware is the model
    which simulates the hardware to be placed in each classroom.
26        port (   ClassroomInUse, LightsAreOn, ProjectorIsOn, RX : in std_logic; --
    ClassroomInUse, LightsAreOn, and ProjectorIsOn are boolean signals from sensors in the room
    which give infromation about that classroom's current condition.
27                 RoomID, OurID : in std_logic_vector(1 downto 0); -- here we are simulating 4
    classrooms, so the classroom IDs, only need to be 2 bits wides.  -- RoomID is broadcasted
    by the CampusController, and OurID is set to the the unique ID of a classroomController in
    a specific classroom.
28                 Clk_In : in std_logic; -- the clock signal.
29                 projectorEnable, LightsEnable, TX, transmitting : out std_logic -- each
    classroomController has outputs which can disable the lights or projector in a classroom.
30                 -- The TX bit is used to communicate with the campusController, it is
    directly connected to the connection line, and is internally set to highZ when it is not a
    classroomController's turn to communicate with the campusController, (a classroomController
    communicates when it is polled by the campusController or when it's OURID = RoomId.)
31              );
32     end Component;
33
34     signal master_clock : std_logic; -- the master signal of the alera board - this is read
    from another altera board in our case through gpio pin. (see GPIO below)
35     signal c0len, c0pen, c1len, c1pen, c2len, c2pen, c3len, c3pen : std_logic; -- c0len is
    Classroom0, lignts, enable. c2pen is Classroom2, Projector, Enable.  these signals are used
    internally to interact with the altera board's physical hardware for IO purposes.
36     signal sen0u, sen0l, sen0p, sen1u, sen1l, sen1p, sen2u, sen2l, sen2p, sen3u, sen3l, sen3p
     : std_logic; -- sensor associated with classroom #, sensing use, lights, projector - these
    repsresent the input sensors to a classroom.
37     signal net1RoomID : std_logic_vector(1 downto 0); -- the RoomId on network 1, (in this
    case, this is the default network since there is only one network.)
38     signal net1tx : std_logic; -- the communication line for network 1.
39     signal trans0, trans1, trans2, trans3 : std_logic; -- we'll use to display LEDs on the
    baord when a specific classroomControllerHardware is transmitting, (when it is being polled
    by the campusController)
40
41  begin
42     -- GPIO
43     -- Read clock from external source.
44     master_clock <= gpio(8);
45     -- Read roomIds from other altera board using GPIO ports.
46     net1RoomID <= gpio(1 downto 0);
47     -- TX bit for communicating with CampusController
```

```
48      gpio(6) <= net1tx;
49
50      -- flash ledg8 with clock signal.
51      ledg(8) <= master_clock;
52
53      -- input switches - used to simulate classrom sensor values.
54      -- Class 0:
55      sen0u <= sw(0); -- classroom0 in Use
56      sen0l <= sw(1); -- classroom0 lightsAreOn
57      sen0p <= sw(2); -- classroom0 projectorIsOn
58      -- Class 1:
59      sen1u <= sw(4);
60      sen1l <= sw(5);
61      sen1p <= sw(6);
62      -- Class 2:
63      sen2u <= sw(8);
64      sen2l <= sw(9);
65      sen2p <= sw(10);
66      -- Class 3:
67      sen3u <= sw(12);
68      sen3l <= sw(13);
69      sen3p <= sw(14);
70
71      -- Outpus LEDs:
72      -- Class 0:
73      ledr(0) <= c0len; -- classroom0 lightsEnabled
74      ledr(1) <= c0pen; -- classroom0 projectorEnabled
75      ledr(2) <= trans0; -- classroom0 is being polled.
76      -- Class 1:
77      ledr(4) <= c1len;
78      ledr(5) <= c1pen;
79      ledr(6) <= trans1;
80      -- Class 2:
81      ledr(8) <= c2len;
82      ledr(9) <= c2pen;
83      ledr(10) <= trans2;
84      -- Class 3:
85      ledr(12) <= c3len;
86      ledr(13) <= c3pen;
87      ledr(14) <= trans3;
88
89      -- we will plot network 1 room id and network 1 tx on ledg
90      ledg(1 downto 0) <= net1RoomID;
91      ledg(7) <= net1tx;
92
93      -- Instantiate our classrooms.
94      Classroom0 : classroomControllerHardware port map(
95         ClassroomInUse => sen0u, -- sensor representing whether or not someone is in the room.
96         LightsAreOn => sen0l, -- sensor representing whether or not the lights are on the room
97         ProjectorIsOn => sen0p, -- sensor representing whether or not the projector is on in
    the room.
98         RX => '0', -- for now, we are not recieving serial from the campusController.
99         RoomID => net1RoomID, -- set the roomID in this classroomControllerHardware to be the
    ID recieved on the GPIO pins.
100        OurID => "00", -- set the unique ID of this specific classroomControllerHardware
101        Clk_In => master_clock, -- pass our clock signal
102        ProjectorEnable => c0pen, -- output enabling the projector
103        LightsEnable => c0len, -- ouput enabling the lights.
104        TX => net1tx, -- classroom0's communication line.
105        transmitting => trans0 -- this signal is true if classroom0 is online on the
    communication line and is transmitting it's data to the campusController.
106     );
107
108        Classroom1 : classroomControllerHardware port map(
109        ClassroomInUse => sen1u,
110        LightsAreOn => sen1l,
111        ProjectorIsOn => sen1p,
112        RX => '0',
113        RoomID => net1RoomID,
114        OurID => "01",
115        Clk_In => master_clock,
116        ProjectorEnable => c1pen,
117        LightsEnable => c1len,
```

```
118         TX => net1tx,
119         transmitting => trans1
120      );
121
122         Classroom2 : classroomControllerHardware port map(
123         ClassroomInUse => sen2u,
124         LightsAreOn => sen2l,
125         ProjectorIsOn => sen2p,
126         RX => '0',
127         RoomID => net1RoomID,
128         OurID => "10",
129         Clk_In => master_clock,
130         ProjectorEnable => c2pen,
131         LightsEnable => c2len,
132         TX => net1tx,
133         transmitting => trans2
134      );
135
136         Classroom3 : classroomControllerHardware port map(
137         ClassroomInUse => sen3u,
138         LightsAreOn => sen3l,
139         ProjectorIsOn => sen3p,
140         RX => '0',
141         RoomID => net1RoomID,
142         OurID => "11",
143         Clk_In => master_clock,
144         ProjectorEnable => c3pen,
145         LightsEnable => c3len,
146         TX => net1tx,
147         transmitting => trans3
148      );
149
150   end a;
151
152   -- Begin Component Declarations
153
154   -- here we declare the classroomControllerHardware which represents the hardware placed in
      each classroom.
155   Library ieee;
156   use ieee.std_logic_1164.all;
157   Entity ClassroomControllerHardware is -- these signals are explained at line 25.
158      port (   ClassroomInUse, LightsAreOn, ProjectorIsOn, RX : in std_logic;
159               RoomID, OurID : in std_logic_vector(1 downto 0);
160               Clk_In : in std_logic;
161               projectorEnable, LightsEnable, TX, transmitting : out std_logic
162            );
163   end ClassroomControllerHardware;
164
165   Architecture a of ClassroomControllerHardware is
166
167      -- declare signals and hardware components.
168      component ls74 is
169      port( d, clr, pre, clk : IN std_logic;
170         -- d is the data input
171         -- clr: ACTIVE LOW: clears the output, q, asynchrnously.
172         -- Pre: ACTIVE LOW: sets the output q to 1 asynchronously,
173         -- clk is a clock signal (q is typically representitive of what d was 1 clock cycle
      ago)
174            q : out std_logic -- single bit output which is d delayed by 1 clock cycle.
175      );
176      end component;
177
178      component piso48b is -- this is a parallel input serial output shift register which is
      48b wide.
179      port (   parallel_In : in std_logic_vector(47 downto 0); -- the 16 bits of input for
      parallel loading
180               SorL : in std_logic; -- the Shift/Load signal. 1 = shift, 0 = load
181               clk : in std_logic; -- the clock signal for the DFFs contained in the shift reg.
182               q : out std_logic -- we shift out through this bit.
183            );
184      end component;
185
186      component comparator6b is -- a six bit comparator which only determines whether or not
```

```
    two six bit integers are equal.
187     port (   op1, op2 : in std_logic_vector(5 downto 0); -- our two 6b inputs.
188              equal : out std_logic -- our 1 bit equal signal. 1 if op1 = op2, else 0.
189           );
190     end component;
191
192     component tri_state_buffer_top is
193     Port (   A  : in  STD_LOGIC;    -- single buffer input
194              EN : in  STD_LOGIC;    -- single buffer enable
195              Y  : out STD_LOGIC    -- single buffer output
196           );
197     end component;
198
199     signal Equal, LoadShiftReg, txToBus, lastEqual, projectorIsEnabledAndOn,
    lightsEnabledAndOn: std_logic;
200     Signal toLoad : std_logic_vector(47 downto 0);
201     -- Equal is an internal signal which is true if the RoomID input matches OURID.
202     -- LoadShiftReg tells the classroomControllerHardware when to load its PISO register.
203     -- TX to bus stores our tx value for the bus before sending it through a tri_state_buffer
204     -- lastEqual is the equal signal 1 clock cycle ago.
205     -- projectorIsEnabledAndOn and lightsEnabledAndOn are signals which are used to simulate
    the closed loop system created by this hardware's ability to disable the lights and
    projector.
206
207   begin
208
209     -- for indication purposes, show when this classroom is transmitting
210     transmitting <= Equal;
211
212     -- Circuitry to determine when we are selected by the BuildingController to Transmit
213     classroomComparator : comparator6b port map(
214        op1 => "0000" & OurID,
215        op2 => "0000" & RoomID,  -- the comparator expects 6b of input but roomId and ourID
    are only 2 bits now.
216        equal => Equal
217     );
218
219     -- Circuitry to determine when we should load our shift registers with new data to shift
    out over serial.
220     -- we want to load the first clock cycle after being selected by the Building Controller
    (first equal cycle)
221     RisingEqualDFF : ls74 port map(
222        d => equal,
223        clr => '1',
224        pre => '1',
225        clk => Clk_In,
226        q => lastEqual
227     );
228
229     LoadShiftReg <= Not(lastEqual) and equal;
230
231     -- Finally implement shift out register for parallel in and serial out - used to
    communicate with campusControllers in serial.
232     serialOutReg : piso48b port map(
233        parallel_In => toLoad,
234        SorL => Not(LoadShiftReg), -- Load is low state.
235        clk => Clk_In,
236        q => txToBus
237     );
238
239     -- specify toLoad
240     toLoad <= "1010101010101010" & "00000000" & projectorIsEnabledAndOn & lightsEnabledAndOn
    & ClassroomInUse & "00000" & "1111111111111111";
241     -- these bitstrings come from predetermined serial communication flags.
242
243     -- wire txto bus to tx bus with a tristate buffer
244     busBuffer : tri_state_buffer_top Port map (
245        A => txToBus,
246        En => Equal,
247        Y => TX
248     );
249
250     -- disable lights and projector when classroom is not in use.
```

```
251      ProjectorEnable <= ClassroomInUse;
252      LightsEnable <= ClassroomInUse;
253
254         -- We simulate that the projector is on or off by doing the following:
255      projectorIsEnabledAndOn <= ClassroomInUse and projectorIsOn;
256      lightsEnabledAndOn <= ClassroomInUse and lightsAreOn;
257
258   end a;
259
260   -- Create a 74x74 chip a DFF
261   -- This component intends to simulate the behaviors of a 74x74 chipset.
262   Library ieee;
263   use ieee.std_logic_1164.all;
264   Entity ls74 is
265      port( d, clr, pre, clk : IN std_logic;
266         -- d is the data input
267         -- clr: ACTIVE LOW: clears the output, q, asynchrnously.
268         -- Pre: ACTIVE LOW: sets the output q to 1 asynchronously,
269         -- clk is a clock signal (q is typically representitive of what d was 1 clock cycle
    ago)
270            q : out std_logic -- single bit output which is d delayed by 1 clock cycle.
271      );
272   end ls74;
273   Architecture a of ls74 is
274   begin
275      Process(clk, clr, pre) -- the DFF should update its output when any of these change.
276      begin
277         if clr = '0' then -- preset q to zero, reguardless of d.
278            q <= '0'; -- note that clr and pre are active low.
279         elsif pre = '0' then  -- preset q to zero, reguardless of d.
280            q <= '1';
281         elsif clk'EVENT and clk = '1' then -- mimic d 1 clock cycle ago on q.
282            if d = '1' then
283               q <= '1';
284            else
285               q <= '0';
286            end if;
287         end if;
288      end process;
289   End a;
290
291   -- We also need a 6 bit comparator.  In actual hardware, this would likely be two 74x85s in
    series but here we will make our own 6b comparator.
292   -- We only need to know if the two operands are equal, so we'll leave out greater than/
    less than capabilities.
293   Library ieee;
294   Use ieee.std_logic_1164.all;
295   Entity comparator6b is
296      port (   op1, op2 : in std_logic_vector(5 downto 0); -- our two 6b inputs.
297               equal : out std_logic -- our 1 bit equal signal. 1 if op1 = op2, else 0.
298            );
299   end comparator6b;
300   Architecture a of comparator6b is
301   begin
302      Process (op1, op2)
303      begin
304         if op1 = op2 then -- if they are equal, represent that on equal.
305            equal <= '1';
306         else
307            equal <= '0';
308         end if;
309      end process;
310   end a;
311
312   -- Finally, we will need a 16b PISO shift regerister.
313   --    in our circuit schematic, we wired two 8 bit shift regeristers together, but here, we
    can just create a 48b shift register.
314   Library ieee;
315   Use ieee.std_logic_1164.all;
316   Entity piso48b is
317      port (   parallel_In : in std_logic_vector(47 downto 0); -- the 16 bits of input for
    parallel loading
318               SorL : in std_logic; -- the Shift/Load signal. 1 = shift, 0 = load
```

```
319               clk : in std_logic; -- the clock signal for the DFFs contained in the shift reg.
320               q : out std_logic -- we shift out through this bit.
321           );
322   end piso48b;
323   Architecture a of piso48b is
324      signal temp : std_logic_vector(47 downto 0);
325   begin
326      -- Note: in this shift regerister, elements are shifted "up" meaning that an item which
      enters at temp(0) is consecutively shifted down the register to temp(47) at which point it
      shows up on the output q.
327      process(clk) -- Our register updates every clock cycle, nothing is asynchronous.
328      begin
329         if clk'EVENT and clk = '1' then
330            if SorL = '0' then -- we should load from our parallel input
331               temp <= parallel_In;
332            else -- otherwise we should shift down the register.
333               temp(47 downto 1) <= temp(46 downto 0);
334               temp(0) <= '1';  -- we never need to shift in serial for this project
      component, so we can simply simulate shifting in a zero.
335            end if;
336         end if;
337      end process;
338
339      q <= temp(47); -- connect our temp vector (the zero element) to our output.
340   end a;
341
342   -- Tristate Buffer
343   Library ieee;
344   use ieee.std_logic_1164.all;
345   entity tri_state_buffer_top is
346      Port( A : in std_logic;    -- single buffer input
347            EN : in std_logic;    -- single buffer enable
348            Y : out std_logic    -- single buffer output
349      );
350   end tri_state_buffer_top;
351   architecture Behavioral of tri_state_buffer_top is
352      begin
353       -- single active low enabled tri-state buffer
354      Y <= A when (EN = '1') else 'Z';
355   end Behavioral;
356
```