Jesse Bishop

PID: A10226163

Professor Dr. Kim

CSE 140L

20 November 2013

ReportL3C010

CID 010

Part A.) Work Description:

Part 1 Coin Input Operation:

For the coin input, I created a case/switch statement that represented each state from my attached FSM diagram. Each state was represented by the amount that has been entered and can transition to another state depending on the sw[] inputs. I also utilized two tasks, one for handling the HEX display and one for blinking 10 red LED's; they were named display and dispense respectively. To transition from one state to the next I used an if statement for each switch and increment or decrement accordingly to get to the next state.

Part 2 sw[9] = 1:

For this part, I created a global counter that incremented by 1 whenever a dispensing state was reached. Inside each state I have an if statement that checks to see if switch 9 is high, and if so, it displays the running total for the number dispensed.

Part 3 Credit Card and Reset Options:

For the credit card option I added another state that dispensed, regardless of what the previous deposits were, and gave change of zero as specified by the lab. For the reset option, I simply subtracted from the deposit amount accordingly to bring the deposit amount back down to the initial zero state.

Part 3: The On Dollar Bill Operation:

For this operation, I again created a series of other states in the switch/case for dollar input and dollar plus previous deposit states. The change for these operations are given accordingly.

Part 4: Err display Operation:

Inside each state, I have an if statement that checks to ensure that no multiple switches are in the high position during input, otherwise they go to an Err state. Once at the Err state, if key[3] is pressed, it is returned back to the 0 state or if reset is initiated. Also, consecutive credit card or dollar inputs are transitioned to the error state as well as credit card inputs entered at the 3500 state.

```
1   module L3C010(
2                 input [9:0] sw,
3                 input [3:0] key,
4                 input clock,
5                 output [9:0] ledr,
6                 output [7:0] ledg,
7                 output reg [6:0] hex0,hex1,hex2,hex3
8                 );
9
10
11    reg[24:0] delay;
12    reg[24:0] delay2;
13    reg blink = 1'b0;
14
15    //various toggle and increament
16    //variables
17    reg [6:0]state;
18    reg [6:0]totalDispense;
19    reg toggle = 1'b0;
20    reg updateOnce = 1'b1;
21
22    //reset parameters
23    reg firstTime = 1'b1;
24    reg mylatch = 1'b0;
25
26    //registers needed for the red LED's
27    reg [9:0]LED_reg;
28    assign ledr[0] = LED_reg[0];
29    assign ledr[1] = LED_reg[1];
30    assign ledr[2] = LED_reg[2];
31    assign ledr[3] = LED_reg[3];
32    assign ledr[4] = LED_reg[4];
33    assign ledr[5] = LED_reg[5];
34    assign ledr[6] = LED_reg[6];
35    assign ledr[7] = LED_reg[7];
36    assign ledr[8] = LED_reg[8];
37    assign ledr[9] = LED_reg[9];
38    parameter COUNT_LEDBLIP = 6000000;
39    reg[24:0] counter_LEDBLIP;
40
41    //Note: state 61 is the error state
42
43  always @ (posedge clock) begin
44
45      //Set equal to CID for the first time
46      if(firstTime == 1'b1) begin
47        LED_reg[9:0] = 10'b0000000000;
48        state = 6'b100100;
49        totalDispense = 6'b000000;
50        firstTime = 1'b0;
51      end
52
53    case(state)
54    0: begin
55        blink = 1'b0;
```

```
56          updateOnce = 1'b1;
57          LED_reg[9:0] = 10'b0000000000;
58          if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
59             sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
60             sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
61             sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
62             sw[4]&&sw[8])
63            increment(61);
64          if(sw[0])
65            increment(5);
66          if(sw[1])
67            increment(10);
68          if(sw[2])
69            increment(25);
70          if(sw[3])
71            increment(100);
72          if(sw[4])
73            increment(35);
74          if(sw[9]) begin
75            display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
76            display(19, hex2[6:0]); display(19, hex3[6:0]);
77          end
78          else begin
79            display(0, hex0[6:0]); display(0, hex1[6:0]);
80            display(0, hex2[6:0]); display(0, hex3[6:0]);
81          end
82
83        end
84       5: begin
85          blink = 1'b0;
86          updateOnce = 1'b1;
87          LED_reg[9:0] = 10'b0000000000;
88          if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
89             sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
90             sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
91             sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
92             sw[4]&&sw[8])
93            increment(56);
94          if(sw[8])
95            increment(-5);
96          if(sw[0])
97            increment(5);
98          if(sw[1])
99            increment(10);
100         if(sw[2])
101           increment(25);
102         if(sw[3])
103           increment(100);
104         if(sw[4])
105           increment(30);
106         if(sw[9]) begin
```

```
107           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
108           display(19, hex2[6:0]); display(19, hex3[6:0]);
109         end
110         else begin
111              display(0, hex0[6:0]); display(0, hex1[6:0]);
112              display(5, hex2[6:0]); display(0, hex3[6:0]);
113         end
114       end
115     10: begin
116         blink = 1'b0;
117         updateOnce = 1'b1;
118         LED_reg[9:0] = 10'b0000000000;
119
120         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
121            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
122            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
123            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
124            sw[4]&&sw[8])
125           increment(51);
126         if(sw[8])
127           increment(-10);
128         if(sw[0])
129           increment(5);
130         if(sw[1])
131           increment(10);
132         if(sw[2])
133           increment(25);
134         if(sw[3])
135           increment(100);
136         if(sw[4])
137           increment(25);
138         if(sw[9]) begin
139           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
140           display(19, hex2[6:0]); display(19, hex3[6:0]);
141         end
142         else begin
143           display(0, hex0[6:0]); display(0, hex1[6:0]);
144           display(0, hex2[6:0]); display(1, hex3[6:0]);
145         end
146       end
147     15: begin
148         blink = 1'b0;
149         updateOnce = 1'b1;
150         LED_reg[9:0] = 10'b0000000000;
151         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
152            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
153            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
154            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
155            sw[4]&&sw[8])
156           increment(46);
157         if(sw[8])
```

```
158           increment(-15);
159         if(sw[0])
160           increment(5);
161         if(sw[1])
162           increment(10);
163         if(sw[2])
164           increment(25);
165         if(sw[3])
166           increment(100);
167         if(sw[4])
168           increment(20);
169         if(sw[9]) begin
170           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
171           display(19, hex2[6:0]); display(19, hex3[6:0]);
172         end
173         else begin
174           display(0, hex0[6:0]); display(0, hex1[6:0]);
175           display(5, hex2[6:0]); display(1, hex3[6:0]);
176         end
177
178       end
179
180     20: begin
181         blink = 1'b0;
182         updateOnce = 1'b1;
183         LED_reg[9:0] = 10'b0000000000;
184         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
185            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
186            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
187            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
188            sw[4]&&sw[8])
189           increment(41);
190         if(sw[8])
191           increment(-20);
192         if(sw[0])
193           increment(5);
194         if(sw[1])
195           increment(10);
196         if(sw[2])
197           increment(25);
198         if(sw[3])
199           increment(100);
200         if(sw[4])
201           increment(15);
202         if(sw[9]) begin
203           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
204           display(19, hex2[6:0]); display(19, hex3[6:0]);
205         end
206         else begin
207           display(0, hex0[6:0]); display(0, hex1[6:0]);
208           display(0, hex2[6:0]); display(2, hex3[6:0]);
209         end
210     end
```

```
211
212     25: begin
213         blink = 1'b0;
214         updateOnce = 1'b1;
215         LED_reg[9:0] = 10'b0000000000;
216         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
217            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
218            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
219            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
220            sw[4]&&sw[8])
221           increment(36);
222         if(sw[8])
223           increment(-25);
224         if(sw[0])
225           increment(5);
226         if(sw[1])
227           increment(10);
228         if(sw[2])
229           increment(25);
230         if(sw[3])
231           increment(100);
232         if(sw[4])
233           increment(10);
234         if(sw[9]) begin
235           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
236           display(19, hex2[6:0]); display(19, hex3[6:0]);
237         end
238         else begin
239           display(0, hex0[6:0]); display(0, hex1[6:0]);
240           display(5, hex2[6:0]); display(2, hex3[6:0]);
241         end
242     end
243
244     30: begin
245         blink = 1'b0;
246         updateOnce = 1'b1;
247         LED_reg[9:0] = 10'b0000000000;
248
249         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
250            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
251            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
252            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
253            sw[4]&&sw[8])
254           increment(31);
255         if(sw[8])
256           increment(-30);
257         if(sw[0])
258           increment(5);
259         if(sw[1])
260           increment(10);
261         if(sw[2])
```

```
262           increment(25);
263         if(sw[3])
264           increment(100);
265         if(sw[4])
266           increment(5);
267         if(sw[9]) begin
268           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
269           display(19, hex2[6:0]); display(19, hex3[6:0]);
270         end
271         else begin
272           display(0, hex0[6:0]); display(0, hex1[6:0]);
273           display(0, hex2[6:0]); display(3, hex3[6:0]);
274         end
275     end
276     35: begin
277       blink = 1'b1;
278       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
279         counter_LEDBLIP <= 0;
280         toggle <= !toggle;
281       end
282       else begin
283         counter_LEDBLIP <= counter_LEDBLIP + 1;
284       end
285       dispense(toggle, LED_reg[9:0]);
286       if(updateOnce) begin
287         totalDispense = totalDispense + 1;
288         updateOnce = 1'b0;
289       end
290         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
291            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
292            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
293            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
294            sw[4]&&sw[8])
295           increment(26);
296         if(sw[8])
297           increment(-35);
298         if(sw[0])
299           increment(-30);
300         if(sw[1])
301           increment(-25);
302         if(sw[2])
303           increment(-10);
304         if(sw[3])begin
305           increment(65);
306         end
307         if(sw[4])
308           increment(26);
309         if(totalDispense > 15)
310           totalDispense = 0;
311
312         if(sw[9]) begin
313           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
314           display(19, hex2[6:0]); display(19, hex3[6:0]);
```

```
315         end
316         else begin
317           display(0, hex0[6:0]); display(0, hex1[6:0]);
318           display(5, hex2[6:0]); display(3, hex3[6:0]);
319         end
320     end
321
322     40: begin
323       blink = 1'b1;
324       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
325         counter_LEDBLIP <= 0;
326         toggle <= !toggle;
327       end
328       else begin
329         counter_LEDBLIP <= counter_LEDBLIP + 1;
330       end
331       dispense(toggle, LED_reg[9:0]);
332       if(updateOnce) begin
333       if(updateOnce) begin
334         totalDispense = totalDispense + 1;
335         updateOnce = 1'b0;
336       end
337         updateOnce = 1'b0;
338       end
339         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
340            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
341            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
342            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
343            sw[4]&&sw[8])
344           increment(21);
345         if(sw[8])
346           increment(-40);
347         if(sw[0])
348           increment(-35);
349         if(sw[1])
350           increment(-30);
351         if(sw[2])
352           increment(-15);
353         if(sw[4])
354           increment(-5);
355         if(totalDispense > 15)
356           totalDispense = 0;
357         if(sw[9]) begin
358           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
359           display(19, hex2[6:0]); display(19, hex3[6:0]);
360         end
361         else begin
362           display(5, hex0[6:0]); display(0, hex1[6:0]);
363           display(5, hex2[6:0]); display(3, hex3[6:0]);
364         end
365     end
366
367     45: begin
```

```
368       blink = 1'b1;
369       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
370         counter_LEDBLIP <= 0;
371         toggle <= !toggle;
372       end
373       else begin
374         counter_LEDBLIP <= counter_LEDBLIP + 1;
375       end
376       dispense(toggle, LED_reg[9:0]);
377       if(updateOnce) begin
378         totalDispense = totalDispense + 1;
379         updateOnce = 1'b0;
380       end
381         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
382            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
383            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
384            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
385            sw[4]&&sw[8])
386           increment(16);
387         if(sw[8])
388           increment(-45);
389         if(sw[0])
390           increment(-40);
391         if(sw[1])
392           increment(-35);
393         if(sw[2])
394           increment(-20);
395         if(sw[4])
396           increment(-10);
397         if(totalDispense > 15)
398           totalDispense = 0;
399         if(sw[9]) begin
400           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
401           display(19, hex2[6:0]); display(19, hex3[6:0]);
402         end
403         else begin
404           display(0, hex0[6:0]); display(1, hex1[6:0]);
405           display(5, hex2[6:0]); display(3, hex3[6:0]);
406         end
407     end
408
409     50: begin
410       blink = 1'b1;
411       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
412         counter_LEDBLIP <= 0;
413         toggle <= !toggle;
414       end
415       else begin
416         counter_LEDBLIP <= counter_LEDBLIP + 1;
417       end
418       dispense(toggle, LED_reg[9:0]);
419       if(updateOnce) begin
420         totalDispense = totalDispense + 1;
```

```
421         updateOnce = 1'b0;
422       end
423         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
424            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
425            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
426            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
427            sw[4]&&sw[8])
428           increment(11);
429         if(sw[8])
430           increment(-50);
431         if(sw[0])
432           increment(-45);
433         if(sw[1])
434           increment(-40);
435         if(sw[2])
436           increment(-25);
437         if(sw[4])
438           increment(-15);
439         if(totalDispense > 15)
440           totalDispense = 0;
441         if(sw[9]) begin
442           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
443           display(19, hex2[6:0]); display(19, hex3[6:0]);
444         end
445         else begin
446           display(5, hex0[6:0]); display(1, hex1[6:0]);
447           display(5, hex2[6:0]); display(3, hex3[6:0]);
448         end
449     end
450
451     55: begin
452       blink = 1'b1;
453       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
454         counter_LEDBLIP <= 0;
455         toggle <= !toggle;
456       end
457       else begin
458         counter_LEDBLIP <= counter_LEDBLIP + 1;
459       end
460       dispense(toggle, LED_reg[9:0]);
461       if(updateOnce) begin
462         totalDispense = totalDispense + 1;
463         updateOnce = 1'b0;
464       end
465         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
466            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
467            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
468            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
469            sw[4]&&sw[8])
470           increment(6);
471         if(sw[8])
```

```
472           increment(-55);
473         if(sw[0])
474           increment(-45);
475         if(sw[1])
476           increment(-40);
477         if(sw[2])
478           increment(-25);
479         if(sw[4])
480           increment(-20);
481         if(totalDispense > 15)
482           totalDispense = 0;
483         if(sw[9]) begin
484           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
485           display(19, hex2[6:0]); display(19, hex3[6:0]);
486         end
487         else begin
488           display(0, hex0[6:0]); display(2, hex1[6:0]);
489           display(5, hex2[6:0]); display(3, hex3[6:0]);
490         end
491     end
492
493     60: begin
494       blink = 1'b1;
495       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
496         counter_LEDBLIP <= 0;
497         toggle <= !toggle;
498       end
499       else begin
500         counter_LEDBLIP <= counter_LEDBLIP + 1;
501       end
502       dispense(toggle, LED_reg[9:0]);
503       if(updateOnce) begin
504         totalDispense = totalDispense + 1;
505         updateOnce = 1'b0;
506       end
507         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
508            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
509            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
510            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
511            sw[4]&&sw[8])
512           increment(1);
513         if(sw[8])
514           increment(-60);
515         if(sw[0])
516           increment(-55);
517         if(sw[1])
518           increment(-50);
519         if(sw[2])
520           increment(-45);
521         if(sw[4])
522           increment(-25);
523         if(totalDispense > 15)
524           totalDispense = 0;
```

```
525         if(sw[9]) begin
526           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
527           display(19, hex2[6:0]); display(19, hex3[6:0]);
528         end
529         else begin
530           display(5, hex0[6:0]); display(2, hex1[6:0]);
531           display(5, hex2[6:0]); display(3, hex3[6:0]);
532         end
533     end
534
535     61: begin display(19, hex0[6:0]); display(18, hex1[6:0]);
536               display(18, hex2[6:0]); display(17, hex3[6:0]);
537         updateOnce = 1'b1;
538
539         if(blink == 1'b1)begin
540           if(counter_LEDBLIP == COUNT_LEDBLIP) begin
541             counter_LEDBLIP <= 0;
542             toggle <= !toggle;
543           end
544           else begin
545             counter_LEDBLIP <= counter_LEDBLIP + 1;
546           end
547             dispense(toggle, LED_reg[9:0]);
548         end
549
550         if(sw[8])
551           increment(-61);
552         if(sw[0])
553           increment(-61);
554         if(sw[1])
555           increment(-61);
556         if(sw[2])
557           increment(-61);
558         if(sw[3])
559           increment(-61);
560         if(sw[4])
561           increment(-26);
562         if(!sw[8] || !sw[0] || !sw[1] || !sw[2] || !sw[3] || !sw[4])
563           increment(-61);
564         end
565
566     100: begin
567       blink = 1'b1;
568       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
569         counter_LEDBLIP <= 0;
570         toggle <= !toggle;
571       end
572       else begin
573         counter_LEDBLIP <= counter_LEDBLIP + 1;
574       end
575       dispense(toggle, LED_reg[9:0]);
576       if(updateOnce) begin
577         totalDispense = totalDispense + 1;
578         updateOnce = 1'b0;
579       end
```

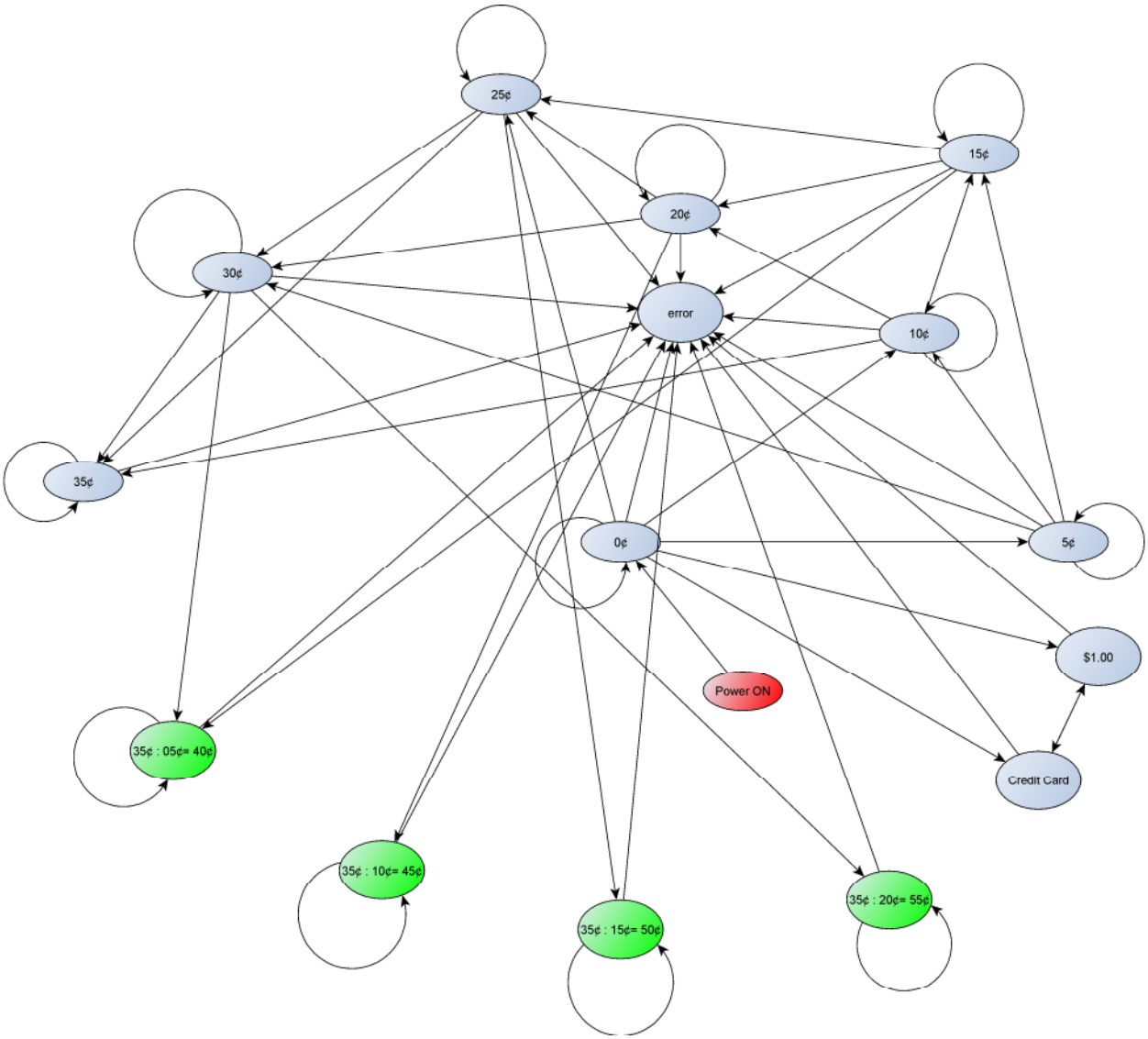
```
580         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
581            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
582            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
583            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
584            sw[4]&&sw[8])
585           increment(1);
586         if(sw[8])
587           increment(-100);
588         if(sw[0])
589           increment(-95);
590         if(sw[1])
591           increment(-90);
592         if(sw[2])
593           increment(-75);
594         if(sw[3])
595           increment(-39);
596         if(sw[4])
597           increment(-65);
598         if(totalDispense > 15)
599           totalDispense = 0;
600         if(sw[9]) begin
601           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
602           display(19, hex2[6:0]); display(19, hex3[6:0]);
603         end
604         else begin
605           display(5, hex0[6:0]); display(6, hex1[6:0]);
606           display(5, hex2[6:0]); display(3, hex3[6:0]);
607         end
608
609     end
610
611     105: begin
612       blink = 1'b1;
613       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
614         counter_LEDBLIP <= 0;
615         toggle <= !toggle;
616       end
617       else begin
618         counter_LEDBLIP <= counter_LEDBLIP + 1;
619       end
620       dispense(toggle, LED_reg[9:0]);
621       if(updateOnce) begin
622         totalDispense = totalDispense + 1;
623         updateOnce = 1'b0;
624       end
625         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
626            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
627            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
628            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
629            sw[4]&&sw[8])
630           increment(1);
```

```
631         if(sw[8])
632           increment(-105);
633         if(sw[0])
634           increment(-100);
635         if(sw[1])
636           increment(-95);
637         if(sw[2])
638           increment(-80);
639         if(sw[4])
640           increment(-70);
641         if(totalDispense > 15)
642           totalDispense = 0;
643         if(sw[9]) begin
644           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
645           display(19, hex2[6:0]); display(19, hex3[6:0]);
646         end
647         else begin
648           display(0, hex0[6:0]); display(7, hex1[6:0]);
649           display(5, hex2[6:0]); display(3, hex3[6:0]);
650         end
651     end
652
653     110: begin
654       blink = 1'b1;
655       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
656         counter_LEDBLIP <= 0;
657         toggle <= !toggle;
658       end
659       else begin
660         counter_LEDBLIP <= counter_LEDBLIP + 1;
661       end
662       dispense(toggle, LED_reg[9:0]);
663       if(updateOnce) begin
664         totalDispense = totalDispense + 1;
665         updateOnce = 1'b0;
666       end
667         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
668            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
669            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
670            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
671            sw[4]&&sw[8])
672           increment(1);
673         if(sw[8])
674           increment(-110);
675         if(sw[0])
676           increment(-105);
677         if(sw[1])
678           increment(-10);
679         if(sw[2])
680           increment(-85);
681         if(sw[4])
682           increment(-75);
683         if(totalDispense > 15)
```

```
684           totalDispense = 0;
685         if(sw[9]) begin
686           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
687           display(19, hex2[6:0]); display(19, hex3[6:0]);
688         end
689         else begin
690           display(5, hex0[6:0]); display(7, hex1[6:0]);
691           display(5, hex2[6:0]); display(3, hex3[6:0]);
692         end
693     end
694
695     125: begin
696       blink = 1'b1;
697       if(counter_LEDBLIP == COUNT_LEDBLIP) begin
698         counter_LEDBLIP <= 0;
699         toggle <= !toggle;
700       end
701       else begin
702         counter_LEDBLIP <= counter_LEDBLIP + 1;
703       end
704       dispense(toggle, LED_reg[9:0]);
705       if(updateOnce) begin
706         totalDispense = totalDispense + 1;
707         updateOnce = 1'b0;
708       end
709         if(sw[0]&&sw[1] || sw[0]&&sw[2] || sw[1]&&sw[2] ||
710            sw[0]&&sw[8] || sw[1]&&sw[8] || sw[2]&&sw[8] ||
711            sw[3]&&sw[0] || sw[3]&&sw[1] || sw[3]&&sw[2] || sw[3]&&sw[8
    ] ||
712            sw[4]&&sw[0] || sw[4]&&sw[1] || sw[4]&&sw[2] || sw[4]&&sw[3
    ] ||
713            sw[4]&&sw[8])
714           increment(1);
715         if(sw[8])
716           increment(-125);
717         if(sw[0])
718           increment(-120);
719         if(sw[1])
720           increment(-115);
721         if(sw[2])
722           increment(-100);
723         if(sw[4])
724           increment(-90);
725         if(totalDispense > 15)
726           totalDispense = 0;
727         if(sw[9]) begin
728           display(totalDispense, hex0[6:0]); display(19, hex1[6:0]);
729           display(19, hex2[6:0]); display(19, hex3[6:0]);
730         end
731         else begin
732           display(0, hex0[6:0]); display(9, hex1[6:0]);
733           display(5, hex2[6:0]); display(3, hex3[6:0]);
734         end
735     end
736
```

```
737     default: begin display(0, hex0[6:0]); display(1, hex1[6:0]);
738                    display(0, hex2[6:0]); display(0, hex3[6:0]);
739                    increment(-36);
740              end
741     endcase
742   end
743
744
745     //create a task that simply increments
746     task increment;
747       input integer incrementBy;
748
749       //The counting process
750       if(!key[3] && (mylatch == 1'b0)) begin
751         if(delay2 >= 20) begin
752           mylatch = 1'b1;
753           state = state + incrementBy;
754           delay2 <= 0;
755         end
756         else begin
757           delay2 <= delay2 +1;
758         end
759       end
760       else if(key[3] && (mylatch == 1'b1)) begin
761         if(delay >= 20)begin
762           mylatch = 1'b0;
763           delay <= 0;
764         end
765         else begin
766           delay <= delay + 1;
767         end
768       end
769     endtask
770
771     task display;
772       input integer num;
773       output reg[6:0]HEX;
774
775       case (num)
776         0:  begin HEX[6:0] = 7'b1000000; end
777         1:  begin HEX[6:0] = 7'b1111001; end
778         2:  begin HEX[6:0] = 7'b0100100; end
779         3:  begin HEX[6:0] = 7'b0110000; end
780         4:  begin HEX[6:0] = 7'b0011001; end
781         5:  begin HEX[6:0] = 7'b0010010; end
782         6:  begin HEX[6:0] = 7'b0000010; end
783         7:  begin HEX[6:0] = 7'b1111000; end
784         8:  begin HEX[6:0] = 7'b0000000; end
785         9:  begin HEX[6:0] = 7'b0010000; end
786         10: begin HEX[6:0] = 7'b0001000; end
787         11: begin HEX[6:0] = 7'b0000011; end
788         12: begin HEX[6:0] = 7'b1000110; end
789         13: begin HEX[6:0] = 7'b0100001; end
790         14: begin HEX[6:0] = 7'b0000110; end
791         15: begin HEX[6:0] = 7'b0001110; end
```

```
792         16: begin HEX[6:0] = 7'b1111111; end
793         17: begin HEX[6:0] = 7'b0000110; end//E
794         18: begin HEX[6:0] = 7'b0101111; end//r
795         19: begin HEX[6:0] = 7'b1111111; end//blank
796         default:                         ;
797       endcase
798     endtask
799
800     //Task that blinks the red LED's during
801     //dispensing...
802     task dispense;
803       input integer num;
804       output reg [9:0]LED_reg;
805
806       if(num == 1) begin
807         LED_reg[9:0] = 10'b1111111111;
808       end
809       else begin
810         LED_reg[9:0] = 10'b0000000000;
811       end
812     endtask
813   endmodule
```


25¢
15¢
20¢
30¢
error
10¢
35¢
0¢
5¢
$1.00
Power ON
35¢ : 05¢= 40¢
Credit Card
35¢ : 10¢= 45¢
35¢ : 20¢= 55¢
35¢ : 15¢= 50¢

Compilation Report - Flow Summary

Flow Status	Successful - Tue Nov 19 23:55:08 2013
Quartus II Version	9.0 Build 235 06/17/2009 SP 2 SJ Web Edition
Revision Name	L3C010
Top-level Entity Name	L3C010
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	1,696 / 18,752 (9 %)
Total combinational functions	1,690 / 18,752 (9 %)
Dedicated logic registers	121 / 18,752 (< 1 %)
Total registers	121
Total pins	61 / 315 (19 %)
Total virtual pins	0
Total memory bits	0 / 239,616 (0 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Total Logic Elements: 18,752 (9%)