

Structural VHDL Design Top File

```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.ALL;
3  USE ieee.numeric_std.ALL;
4
5  ENTITY LogicalStep_Lab4_top IS
6  PORT(
7      clk_in_50      : in std_logic;
8      rst_n          : in std_logic;
9      pb             : in std_logic_vector(3 downto 0);
10     sw              : in std_logic_vector(7 downto 0); -- The switch inputs
11     leds            : out std_logic_vector(7 downto 0); -- for displaying the switch content
12     seg7_data       : out std_logic_vector(6 downto 0); -- 7-bit outputs to a 7-segment
13     seg7_char1      : out std_logic;                  -- seg7 digi selectors
14     seg7_char2      : out std_logic;                  -- seg7 digi selectors
15 );
16 END LogicalStep_Lab4_top;
17
18 ARCHITECTURE Simplecircuit OF LogicalStep_Lab4_top IS
19     component compx4 port (
20         a0, b0, a1, b1, a2, b2, a3, b3 : in std_logic;
21         a_less_b, a_equal_b, a_more_b : out std_logic
22     );
23     end component;
24
25     component Moore_SM port (
26         clk_input, rst_n : in std_logic;
27         MORE, EQUAL, LESS : in std_logic;
28         current_value : out std_logic_vector(3 downto 0)
29     );
30     end component;
31
32     component segment7_mux port (
33         clk : in std_logic := '0';
34         DIN2 : in std_logic_vector(6 downto 0);
35         DIN1 : in std_logic_vector(6 downto 0);
36         DOUT : out std_logic_vector(6 downto 0);
37         DIG2 : out std_logic;
38         DIG1 : out std_logic;
39     );
40     end component;
41
42     component SevenSegment port (
43         hex : in std_logic_vector(3 downto 0); -- digit 1 shows state number in hex of state machine; if they don't match; it changes
44         sevenseg : out std_logic_vector(6 downto 0) -- target is sw [3..0] and displayed on right digit
45     );
46     end component;
47
48     -----
49     CONSTANT SIM : boolean := FALSE; -- set to TRUE for simulation runs otherwise keep at 0.
50     CONSTANT CLK_DIV_SIZE : INTEGER := 24; -- size of vectors for the counters
51
52     signal Main_CLK : STD_LOGIC; -- main clock to drive sequencing of State Machine
53
54     signal bin_counter : UNSIGNED(CLK_DIV_SIZE-1 downto 0); -- := to_unsigned(0,CLK_DIV_SIZE); -- reset binary counter to zero
55
56     signal Simple_States : std_logic_vector(7 downto 0);
57     signal Left0_Right1 : std_logic;
58
59     signal more : std_logic; -- for 4-bit comparator
60     signal less : std_logic; -- for 4-bit comparator
61     signal equal : std_logic; -- for 4-bit comparator
62
63     signal current_value : std_logic_vector(3 downto 0);
64     signal target_value : std_logic_vector(3 downto 0);
65
66     signal seg7_A : std_logic_vector(6 downto 0); -- left display
67     signal seg7_B : std_logic_vector(6 downto 0); -- right display
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

```

MSM VHDL file

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.numeric_std.all;
4
5  Entity Moore_SM IS Port
6  (
7      clk_input, rst_n          : IN std_logic;
8      MORE, EQUAL, LESS        : IN std_logic;
9      current_value             : OUT std_logic_vector(3 downto 0)
10 );
11 END ENTITY;
12
13 Architecture MSM of Moore_SM is
14     TYPE STATE_NAMES IS (S0, S1, S2, S3, S4, S5, S6, S7,
15                          S8, S9, S10, S11, S12, S13, S14, S15); -- list all the STATES
16
17     signal current_state,next_state : STATE_NAMES; -- signals of type STATE_NAMES
18
19 BEGIN
20     -----
21     --State Machine:
22     -----
23
24     -- REGISTER LOGIC PROCESS
25     -- add clock and any related inputs for state machine register section into Sensitivity List
26     Register_Section: PROCESS (clk_input, rst_n,next_state) -- this process synchronizes the activity to a clock
27     BEGIN
28         IF (rst_n = '0') THEN
29             current_state <= S0;
30         ELSIF(rising_edge(clk_input)) THEN
31             current_state <=next_state;
32         ELSE
33             current_state <= current_state;
34         END IF;
35     END PROCESS;
36
37     -- TRANSITION LOGIC PROCESS (to be combinational only)
38     -- add all transition inputs for state machine into Transition section Sensitivity List
39     -- make sure that all conditional statement options are complete otherwise VHDL will infer LATCHES.
40     Transition_Section: PROCESS (MORE, LESS, EQUAL, current_state)
41     BEGIN
42         CASE current_state IS
43             WHEN S0 =>
44                 IF (MORE ='1') THEN
45                     next_state <= S1;
46                 ELSE
47                     next_state <= S0;
48                 END IF;
49             WHEN S1 =>
50                 IF (MORE ='1') THEN
51                     next_state <= S2;
52                 ELSIF (LESS ='1') THEN
53                     next_state <= S0;
54                 ELSE
55                     next_state <= S1;
56                 END IF;
57             WHEN S2 =>
58                 IF (MORE ='1') THEN
59                     next_state <= S3;
60                 ELSIF (LESS ='1') THEN
61                     next_state <= S1;
62                 ELSE
63                     next_state <= S1;
64                 END IF;
65             WHEN S3 =>
66                 IF (MORE ='1') THEN
67                     next_state <= S4;
68                 ELSIF (LESS ='1') THEN
69                     next_state <= S1;
70                 ELSE
71                     next_state <= S1;
72                 END IF;
```

```

69         next_state <= S2;
70     END IF;
71
72
73 WHEN S3 =>
74     IF (MORE = '1') THEN
75         next_state <= S4;
76     ELSIF (LESS = '1') THEN
77         next_state <= S2;
78     ELSE
79         next_state <= S3;
80     END IF;
81
82 WHEN S4 =>
83     IF (MORE = '1') THEN
84         next_state <= S5;
85     ELSIF (LESS = '1') THEN
86         next_state <= S3;
87     ELSE
88         next_state <= S4;
89     END IF;
90
91 WHEN S5 =>
92     IF (MORE = '1') THEN
93         next_state <= S6;
94     ELSIF (LESS = '1') THEN
95         next_state <= S4;
96     ELSE
97         next_state <= S5;
98     END IF;
99
100 WHEN S6 =>
101     IF (MORE = '1') THEN
102         next_state <= S7;
103     ELSIF (LESS = '1') THEN
104         next_state <= S5;
105     ELSE
106         next_state <= S6;
107     END IF;
108
109 WHEN S7 =>
110     IF (MORE = '1') THEN
111         next_state <= S8;
112     ELSIF (LESS = '1') THEN
113         next_state <= S6;
114     ELSE
115         next_state <= S7;
116     END IF;
117
118 WHEN S8 =>
119     IF (MORE = '1') THEN
120         next_state <= S9;
121     ELSIF (LESS = '1') THEN
122         next_state <= S7;
123     ELSE
124         next_state <= S8;
125     END IF;
126
127 WHEN S9 =>
128     IF (MORE = '1') THEN
129         next_state <= S10;
130     ELSIF (LESS = '1') THEN
131         next_state <= S8;
132     ELSE
133         next_state <= S9;
134     END IF;

```

```

135 WHEN S10 =>
136     IF (MORE = '1') THEN
137         next_state <= S11;
138     ELSIF (LESS = '1') THEN
139         next_state <= S9;
140     ELSE
141         next_state <= S10;
142     END IF;
143
144 WHEN S11 =>
145     IF (MORE = '1') THEN
146         next_state <= S12;
147     ELSIF (LESS = '1') THEN
148         next_state <= S10;
149     ELSE
150         next_state <= S11;
151     END IF;
152
153 WHEN S12 =>
154     IF (MORE = '1') THEN
155         next_state <= S13;
156     ELSIF (LESS = '1') THEN
157         next_state <= S11;
158     ELSE
159         next_state <= S12;
160     END IF;
161
162 WHEN S13 =>
163     IF (MORE = '1') THEN
164         next_state <= S14;
165     ELSIF (LESS = '1') THEN
166         next_state <= S12;
167     ELSE
168         next_state <= S13;
169     END IF;
170
171 WHEN S14 =>
172     IF (MORE = '1') THEN
173         next_state <= S15;
174     ELSIF (LESS = '1') THEN
175         next_state <= S13;
176     ELSE
177         next_state <= S14;
178     END IF;
179
180 WHEN S15 =>
181     IF (LESS = '1') THEN
182         next_state <= S14;
183     ELSE
184         next_state <= S15;
185     END IF;
186
187 WHEN others =>
188     next_state <= S0;
189 END CASE;
190
191 END PROCESS;
192
193 Decoder_Section: PROCESS(current_state)
194
195 BEGIN
196     CASE current_state IS
197     WHEN S0 =>
198         current_value <= "0000";
199     WHEN S1 =>
200         current_value <= "0001";

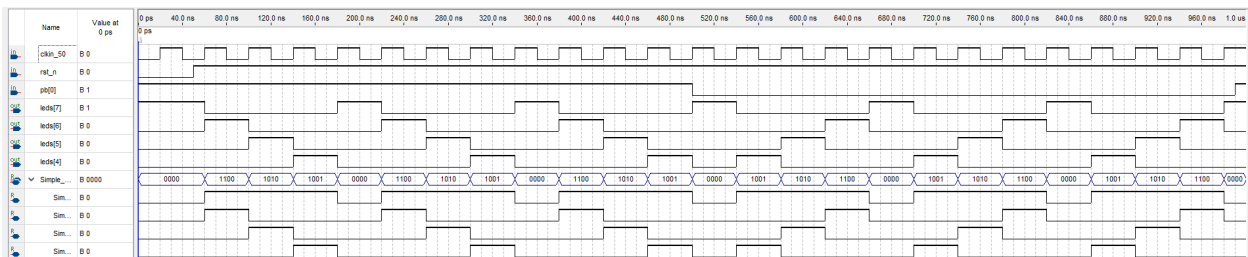
```

```
201 WHEN S2 =>
202     current_value <= "0010";
203 WHEN S3 =>
204     current_value <= "0011";
205 WHEN S4 =>
206     current_value <= "0100";
207 WHEN S5 =>
208     current_value <= "0101";
209 WHEN S6 =>
210     current_value <= "0110";
211 WHEN S7 =>
212     current_value <= "0111";
213 WHEN S8 =>
214     current_value <= "1000";
215 WHEN S9 =>
216     current_value <= "1001";
217 WHEN S10 =>
218     current_value <= "1010";
219 WHEN S11 =>
220     current_value <= "1011";
221 WHEN S12 =>
222     current_value <= "1100";
223 WHEN S13 =>
224     current_value <= "1101";
225 WHEN S14 =>
226     current_value <= "1110";
227 WHEN S15 =>
228     current_value <= "1111";
229 END CASE;
230 END PROCESS;
231
232 END ARCHITECTURE MSM;
```

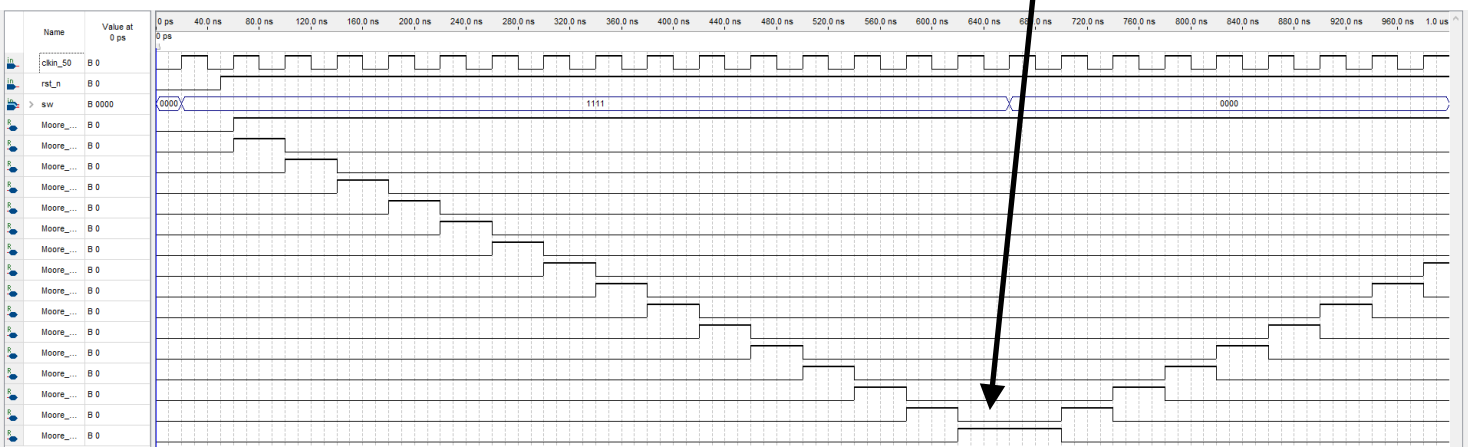
Resource Utilization

Flow Status	Successful - Thu Jun 29 23:53:04 2017
Quartus Prime Version	15.1.0 Build 185 10/21/2015 SJ Standard Edition
Revision Name	LogicalStep_Lab4_top
Top-level Entity Name	LogicalStep_Lab4_top
Family	MAX 10
Device	10M08SAE144C8G
Timing Models	Final
Total logic elements	103 / 8,064 (1 %)
Total combinational functions	103 / 8,064 (1 %)
Dedicated logic registers	44 / 8,064 (< 1 %)
Total registers	44
Total pins	31 / 101 (31 %)
Total virtual pins	0
Total memory bits	0 / 387,072 (0 %)
Embedded Multiplier 9-bit elements	0 / 48 (0 %)
Total PLLs	0 / 1 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 1 (0 %)

Simulation of 4 stage Shift Register operating in both directions



Simulation of Moore State Machine counting over entire range to Max target value



State Diagram

