

```

1 -----
2 -- Engineer: Zachary Montoya
3 -- Submitted Date: 12-15-22
4 -- Module Name:    Top - Behavioral
5 -- Project Name:   Asteroids
6 -- Target Devices: Zybo Z7-10
7 -- Tool versions:  Vivado 2020.2
8 --
9 -- Comment:
10 --      This file was entirely created the engineer listed above.
11 --
12 -----
13
14 --
=====
=====
15 --
=====
=====
16 library ieee;
17 use ieee.std_logic_1164.all;
18 use ieee.numeric_std.all;
19
20 entity asteroid is
21     generic(
22         asteroid_velocity: integer := 5
23     );
24
25     port(
26         clk, reset: in std_logic;
27         -- btn: in std_logic_vector(3 downto 0); --R2 increasing the BTN array to 3
28         video_on: in std_logic;
29         pixel_x, pixel_y: in std_logic_vector(9 downto 0);
30         asteroid_on_out: out std_logic;
31         asteroid_rgb_out: out std_logic_vector(2 downto 0)
32     );
33 end asteroid;
34
35 --
=====
=====
36 --
=====
=====
37
38 architecture rtl of asteroid is
39
40 -- Signal used to control speed of ball and how often pushbuttons are checked for
41 -- paddle movement.
42     signal refr_tick: std_logic;
43
44 -- x, y coordinates (0,0 to (639, 479)
45     signal pix_x, pix_y: unsigned(9 downto 0);
46
47 -- Screen dimensions
48     constant MAX_X: integer := 640;
49     constant MAX_Y: integer := 480;
50
51 -- WALL1 - LEFT
52     constant WALL1_X_L: integer := 0;
53     constant WALL1_X_R: integer := 20;
54
55 -- WALL2 - RIGHT
56     constant WALL2_X_L: integer := 619;
57     constant WALL2_X_R: integer := 639;
58
59 -- WALL3 - BOTTOM
60     constant WALL3_X_T: integer := 409;
61     constant WALL3_X_B: integer := 479;
62
63 -- WALL4 - TOP
64     constant WALL4_X_T: integer := 0;
65     constant WALL4_X_B: integer := 20;
66
67 -- Paddle left, right, top, bottom and height -- left & right are constant. Top &
68 -- bottom are signals to allow movement. bar_y_t driven by register below.
69     --constant BAR_X_L: integer := 600;--R2
70     --constant BAR_X_R: integer := 603;--R2

```

```

69 -- signal bar_x_L,bar_x_R: unsigned(9 downto 0); --R2
70 -- signal bar_y_t, bar_y_b: unsigned(9 downto 0);
71 -- constant BAR_Y_SIZE: integer := 72;
72 -- constant BAR_X_SIZE: integer := 8; --R2
73
74 -- -- Reg to track top boundary (x position is fixed)
75 -- signal bar_y_reg, bar_y_next: unsigned( 9 downto 0);
76
77 -- -- Reg to track right boundary --R2
78 -- signal bar_x_reg, bar_x_next: unsigned( 9 downto 0); --R2
79
80 -- Bar moving velocity when a button is pressed -- the amount the bar is moved.
81 -- constant BAR_V: integer:= 4;
82
83 -- Square ball -- ball left, right, top and bottom all vary. Left and top driven by
84 registers below.
85 constant BALL_SIZE: integer := 16; -----CHANGED Increased to 16
86 signal ball_x_l, ball_x_r: unsigned(9 downto 0);
87 signal ball_y_t, ball_y_b: unsigned(9 downto 0);
88
89 -- Reg to track left and top boundary
90 signal ball_x_reg, ball_x_next: unsigned(9 downto 0);
91 signal ball_y_reg, ball_y_next: unsigned(9 downto 0);
92
93 -- reg to track ball speed
94 signal x_delta_reg, x_delta_next: unsigned(9 downto 0);
95 signal y_delta_reg, y_delta_next: unsigned(9 downto 0);
96
97 -- ball movement can be pos or neg
98 constant BALL_V_P: unsigned(9 downto 0):= to_unsigned(asteroid_velocity,10);
99 constant BALL_V_N: unsigned(9 downto 0):= unsigned(to_signed(-
100 asteroid_velocity,10));
101
102 -- round ball image
103 type rom_type is array(0 to 15) of std_logic_vector(0 to 15); -----Changed
104 from array(0 to 7)
105 constant BALL_ROM: rom_type:= (
106 "0001111111110000",
107 "0010110011011000",
108 "0111101101110100",
109 "1111011110111110",
110 "1011011011011111",
111 "1111011110110111",
112 "1110110110111111",
113 "1101111001111111",
114 "1101111110111111",
115 "1111011111111111",
116 "1011101111111111",
117 "1110110011111111",
118 "0111111111111110",
119 "0010111101101100",
120 "0001101111111000",
121 "0000111111110000");
122
123 -- Testing
124 -- 0 is to paint the background and 1 will be to use the ball
125 -- Can make more than one bit to make characters and other sprites.
126
127 -- ball is 8x8, the address only needs to be 3 bits then.
128 -- data will first be read as a row
129 -- rom_bit will go to the bit in the row
130 signal rom_addr, rom_col: unsigned(3 downto 0); -----Changed
131 to 4 bits from unsigned (2 downto 0)
132 signal rom_data: std_logic_vector(15 downto 0); -----Changed
133 from (7 downto 0)
134 signal rom_bit: std_logic;
135
136 -- object output signals -- new signal to indicate if scan coord is within ball
137 -- signal wall1_on, wall2_on, wall3_on, wall4_on, bar_on, sq_ball_on, rd_ball_on:
138 std_logic;
139 -- signal wall1_rgb, wall2_rgb, wall3_rgb, wall4_rgb, bar_rgb, ball_rgb:
140 std_logic_vector(2 downto 0);
141 signal wall1_on, wall2_on, wall3_on, wall4_on, sq_ball_on, rd_ball_on: std_logic;
142 signal wall1_rgb, wall2_rgb, wall3_rgb, wall4_rgb, ball_rgb: std_logic_vector(2
143 downto 0);
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

```

```

=====
139 begin
140
141 asteroid_on_out <= rd_ball_on;
142 asteroid_rgb_out <= ball_rgb;
143
144 process (clk, reset)
145 begin
146     if (reset = '1') then
147         -- bar_y_reg <= (others => '0');
148         -- bar_x_reg <= (others => '0');--R2
149         ball_x_reg <= (others => '0');
150         ball_y_reg <= (others => '0');
151         x_delta_reg <= ("0000000100");
152         y_delta_reg <= ("0000000100");
153     elsif (clk'event and clk = '1') then
154         -- bar_y_reg <= bar_y_next;
155         -- bar_x_reg <= bar_x_next;--R2
156         ball_x_reg <= ball_x_next;
157         ball_y_reg <= ball_y_next;
158         x_delta_reg <= x_delta_next;
159         y_delta_reg <= y_delta_next;
160     end if;
161 end process;
162
163 --
=====
164 pix_x <= unsigned(pixel_x);
165 pix_y <= unsigned(pixel_y);
166
167 -- Refr_tick: 1-clock tick asserted at start of v_sync, e.g., when the screen is
refreshed -- speed is 60 Hz
168 refr_tick <= '1' when (pix_y = 1) and (pix_x = 1) else '0';
169
170 --
=====
171 -- -- wall1 left vertical stripe
172 -- wall1_on <= '1' when (WALL1_X_L <= pix_x) and (pix_x <= WALL1_X_R) else '0'; --
convert pix_x to pix_y to make horizontal
173 -- wall1_rgb <= "011"; -- paddle colors blue
174 -- -- wall2 right vertical stripe
175 -- wall2_on <= '1' when (WALL2_X_L <= pix_x) and (pix_x <= WALL2_X_R) else '0';
176 -- wall2_rgb <= "011"; -- paddle colors blue
177 -- -- wall3 left vertical stripe
178 -- wall3_on <= '1' when (WALL3_X_T <= pix_y) and (pix_y <= WALL3_X_B) else '0'; --
convert pix_x to pix_y to make horizontal
179 -- wall3_rgb <= "011"; -- paddle colors blue
180 -- -- wall4 right vertical stripe
181 -- wall4_on <= '1' when (WALL4_X_T <= pix_y) and (pix_y <= WALL4_X_B) else '0';
182 -- wall4_rgb <= "011"; -- paddle colors blue
183
184 --
=====
185 -- -- pixel within paddle
186 -- bar_x_l <= bar_x_reg;--R2
187 -- bar_x_r <= bar_x_l + BAR_X_SIZE - 1;--R2
188
189 -- bar_y_t <= bar_y_reg;
190 -- bar_y_b <= bar_y_t + BAR_Y_SIZE - 1;
191 -- bar_on <= '1' when (BAR_X_L <= pix_x) and (pix_x <= BAR_X_R) and (bar_y_t <=
pix_y) and (pix_y <= bar_y_b) else '0';
192 -- bar_rgb <= "100"; -- Red color
193
194 --
=====
195 -- set coordinates of square ball.
196 ball_x_l <= ball_x_reg;
197 ball_y_t <= ball_y_reg;
198 ball_x_r <= ball_x_l + BALL_SIZE - 1;
199 ball_y_b <= ball_y_t + BALL_SIZE - 1;
200
201 -- pixel within square ball
202 sq_ball_on <= '1' when (ball_x_l <= pix_x) and (pix_x <= ball_x_r) and (ball_y_t
<= pix_y) and (pix_y <= ball_y_b) else '0';
203

```

```

204 -- Map scan coord to ROM addr/col -- use low order three bits of pixel and ball
positions. ROM row
205 rom_addr <= pix_y(3 downto 0) - ball_y_t(3 downto 0); ----- CHANGED
TO 4 BITS
206
207 -- ROM column
208 rom_col <= pix_x(3 downto 0) - ball_x_l(3 downto 0);----- CHANGED
TO 4 BITS
209
210 -- Get row data
211 rom_data <= BALL_ROM(to_integer(rom_addr));
212
213 -- Get column bit
214 rom_bit <= rom_data(to_integer(rom_col));
215
216 -- Turn ball on only if within square and the ROM bit is 1.
217 rd_ball_on <= '1' when (sq_ball_on = '1') and (rom_bit = '1') else '0';
218 ball_rgb <= "111"; -- WHITE BALL COLOR
219
220 -- Update the ball position 60 times per second.
221 ball_x_next <= ball_x_reg + x_delta_reg when refr_tick = '1' else ball_x_reg;
222 ball_y_next <= ball_y_reg + y_delta_reg when refr_tick = '1' else ball_y_reg;
223
224 -- Set the value of the next ball position according to the boundaries.
225 -- process(x_delta_reg, y_delta_reg, ball_y_t, ball_x_l, ball_x_r, ball_y_t,
ball_y_b, bar_y_t, bar_y_b)
226 process(x_delta_reg, y_delta_reg, ball_y_t, ball_x_l, ball_x_r, ball_y_t,
ball_y_b)
227 begin
228 x_delta_next <= x_delta_reg;
229 y_delta_next <= y_delta_reg;
230
231 -- Ball reached top, make offset positive
232 if ( ball_y_t < WALL4_X_B ) then --MAKE WALL4
233 y_delta_next <= BALL_V_P;
234
235 -- Reached bottom, make negative
236 elsif (ball_y_b > (WALL3_X_T - 1)) then --MAKE WALL3
237 y_delta_next <= BALL_V_N;
238
239 -- Reach wall1, bounce back
240 elsif (ball_x_l <= WALL1_X_R ) then
241 x_delta_next <= BALL_V_P;
242 -- Reach wall2, bounce back
243 elsif (ball_x_r > WALL2_X_L) then
244 x_delta_next <= BALL_V_N;
245
246 -- -- Right corner of ball inside bar
247 -- elsif ((BAR_X_L <= ball_x_r) and (ball_x_r <= BAR_X_R)) then
248
249 -- -- Some portion of ball hitting paddle, reverse direction
250 -- if ((bar_y_t <= ball_y_b) and (ball_y_t <= bar_y_b)) then
251 -- x_delta_next <= BALL_V_N;
252 -- end if;
253 end if;
254 end process;
255
256 --
=====
=====
257
258
259 end rtl;
260
261

```