

Processor_9_bits.vhd

```

1
2 library ieee;
3 use ieee.std_logic_1164.all;
4 use ieee.numeric_std.all;
5 --entity declaration
6 entity processor is
7     --Instructions declaration
8     generic(
9         mv : std_logic_vector:= "000";
10        mvi: std_logic_vector:= "001";
11        add: std_logic_vector:= "010";
12        sub: std_logic_vector:= "011"
13    );
14    --Inputs Outputs declaration
15    port(
16        clock, aResetn, Run: in std_logic;
17        DataIn: in std_logic_vector(8 downto 0);
18        Bus_Wires: buffer std_logic_vector(8 downto 0);
19        Done: buffer std_logic
20    );
21
22 end processor;
23
24 --architecture declaration
25 architecture arch of processor is
26     --Decoder and Register declaration
27     component dec3to8 is
28         port(
29             input : in std_logic_vector(2 downto 0);
30             enable: in std_logic;
31             output: out std_logic_vector(0 to 7)
32         );
33     end component;
34
35     component reg_n is
36         generic( N: integer:=9);
37         port(
38             clock    : in std_logic;
39             enable   : in std_logic;
40             D        : in std_logic_vector(N-1 downto 0);
41             Q        : out std_logic_vector(N-1 downto 0)
42         );
43     end component;
44     --Signals declaration
45     type state_type is (T0, T1, T2, T3);
46     signal Tcycle_D, Tcycle_Q : state_type;
47     signal Rin, Rout, Xregn, Yregn : std_logic_vector(0 to 7);
48     signal DinOut, Gout, Ain, Gin, IRin, AddSub: std_logic;
49     signal R0, R1, R2, R3, R4, R5, R6, R7, IR, G, A: std_logic_vector(8 downto 0);
50     signal I: std_logic_vector(2 downto 0);
51     signal sel : std_logic_vector(9 downto 0);
52     signal sum: signed(8 downto 0);
53
54
55
56

```

```

57 begin
58     -- Extracting bits from IR
59     I <= IR(8 downto 6);
60
61     -- Instantiating dec3to8 components
62     U1: dec3to8 port map (input => IR(5 downto 3) , enable => '1', output => Xreg);
63     U2: dec3to8 port map (input => IR(2 downto 0) , enable => '1', output => Yreg);
64
65     -- Control of the cycles of the execution of the instructions
66     process(Tcycle_Q, Run, Done)
67     begin
68         case(Tcycle_Q) is
69             -- The instruction mv and mvi take 2 clock cycles to execute
70             -- The instruction add and sub take 4 clock cycles to execute
71
72             when T0 => if run='0' then Tcycle_D <= T0;          -- idle state
73                        else Tcycle_D <= T1; end if;          -- instruction execution
state
74
75             when T1 => if Done='1' then Tcycle_D <= T0;      -- In case of mv and mvi
instructions
76                        else Tcycle_D <= T2; end if;          -- In case of add and sub
instructions
77
78             when T2 => Tcycle_D <= T3;
79
80             when T3 => Tcycle_D <= T0;
81         end case;
82     end process;
83
84     -- Control of the signals of control of dataflow
85     process(Tcycle_Q, I)
86     begin
87         -- Resetting control signals
88         DinOut <= '0'; Gout <= '0'; Ain <= '0'; Gin <= '0'; IRin<='0';
89         AddSub <= '0'; Rin <= "00000000"; Rout <= "00000000"; Done <= '0';
90
91         -- Changin the control signals based on the current state and the instruction
92         case(Tcycle_Q) is
93             when T0 => IRin <= '1';
94             when T1 =>
95                 case(I) is
96                     when mv      => Rout <= Yreg; Rin <= Xreg; Done <= '1';
97                     when mvi     => Dinout <= '1'; Rin <= Xreg; Done <= '1';
98                     when add     => Rout <= Xreg; Ain <= '1';
99                     when others => Rout <= Xreg; Ain <= '1';
100                 end case;
101
102             when T2 =>
103                 case(I) is
104                     when add     => Rout <= Yreg; Gin <= '1'; AddSub <= '1';
105                     when others => Rout <= Yreg; Gin <= '1';
106                 end case;
107
108             when T3 =>
109                 case(I) is
110                     when add     => Gout <= '1'; Rin <= Xreg; Done <= '1';
111                     when others => Gout <= '1'; Rin <= Xreg; Done <= '1';
112                 end case;
113             end case;
114         end process;

```

```

115
116 -- Clock and reset process
117 process(clock, aResetn)
118 begin
119     if (aResetn='0') then
120         Tcycle_Q <=T0;
121     elsif rising_edge(clock) then
122         Tcycle_Q <= Tcycle_D;
123     end if;
124 end process;
125
126 -- Dataflow process for addition/subtraction
127 process(A, Bus_Wires)
128 begin
129     if AddSub ='1' then
130         sum <= signed(A) + signed(Bus_Wires);
131     else
132         sum <= signed(A) - signed(Bus_Wires);
133     end if;
134 end process;
135
136 -- Process for selecting Bus_Wires value based on the selection signal
137 process(DataIn, R0, R1, R2, R3, R4, R5, R6, R7, G, sel)
138 begin
139     case(sel) is
140         when "1000000000" => Bus_Wires <= DataIn;
141         when "0100000000" => Bus_Wires <= R0;
142         when "0010000000" => Bus_Wires <= R1;
143         when "0001000000" => Bus_Wires <= R2;
144         when "0000100000" => Bus_Wires <= R3;
145         when "0000010000" => Bus_Wires <= R4;
146         when "0000001000" => Bus_Wires <= R5;
147         when "0000000100" => Bus_Wires <= R6;
148         when "0000000010" => Bus_Wires <= R7;
149         when "0000000001" => Bus_Wires <= G;
150         when others
151             => Bus_Wires <= DataIn;
152     end case;
153 end process;
154
155 -- Register instantiations
156 Reg0: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(0), D =>
157 Bus_Wires, Q => R0);
158 Reg1: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(1), D =>
159 Bus_Wires, Q => R1);
160 Reg2: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(2), D =>
161 Bus_Wires, Q => R2);
162 Reg3: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(3), D =>
163 Bus_Wires, Q => R3);
164 Reg4: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(4), D =>
165 Bus_Wires, Q => R4);
166 Reg5: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(5), D =>
167 Bus_Wires, Q => R5);
168 Reg6: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(6), D =>
169 Bus_Wires, Q => R6);
170 Reg7: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(7), D =>
171 Bus_Wires, Q => R7);
172 Reg8: reg_n generic map(N => 9) port map (clock => clock, enable => Ain, D =>
173 Bus_Wires, Q => A);
174 Reg9: reg_n generic map(N => 9) port map (clock => clock, enable => IRin, D => DataIn,
175 Q => IR);
176 Reg10: reg_n generic map(N => 9) port map(clock => clock, enable => Gin, D =>
177 std_logic_vector(sum), Q => G);
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

```

```

167     -- Assigning value to the selection signal
168     sel <= DinOut & Rout & Gout;
169
170 end arch;
171
172 -- 3 to 8 decoder
173 library ieee;
174 use ieee.std_logic_1164.all;
175 entity dec3to8 is
176     port(
177         input : in std_logic_vector(2 downto 0);
178         enable: in std_logic;
179         output: out std_logic_vector(0 to 7)
180     );
181 end dec3to8;
182 architecture arch of dec3to8 is
183 begin
184     process(input, enable)
185     begin
186         if (enable = '1') then
187             case(input) is
188                 when "000" => output <= "10000000";
189                 when "001" => output <= "01000000";
190                 when "010" => output <= "00100000";
191                 when "011" => output <= "00010000";
192                 when "100" => output <= "00001000";
193                 when "101" => output <= "00000100";
194                 when "110" => output <= "00000010";
195                 when others => output <= "00000001";
196             end case;
197             else output <= "00000000";
198             end if;
199         end process;
200 end arch;
201
202 -- 9-bit register
203 library ieee;
204 use ieee.std_logic_1164.all;
205 entity reg_n is
206     generic( N: integer:=9);
207     port(
208         clock   : in std_logic;
209         enable  : in std_logic;
210         D       : in std_logic_vector(N-1 downto 0);
211         Q       : out std_logic_vector(N-1 downto 0)
212     );
213 end reg_n;
214 architecture arch of reg_n is
215 begin
216     process(clock)
217     begin
218         if(rising_edge(clock)) then
219             if(enable='1') then
220                 Q<=D;
221             end if;
222         end if;
223     end process;
224 end arch;

```