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 => Xregn);
63      U2: dec3to8 port map (input => IR(2 downto 0) , enable => '1', output => Yregn);
64
65      -- Control of the cycles of the excution 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 <= Yregn; Rin <= Xregn; Done <='1';
97                      when mvi    => Dinout <='1'; Rin <= Xregn; Done <='1';
98                      when add    => Rout <= Xregn; Ain <= '1';
99                      when others => Rout <= Xregn; Ain <= '1';
100                 end case;
101
102             when T2 =>
103                 case(I) is
104                     when add    => Rout <= Yregn; Gin <='1'; AddSub <='1';
105                     when others => Rout <= Yregn; Gin <='1';
106                 end case;
107
108             when T3 =>
109                 case(I) is
110                     when add    => Gout <='1'; Rin <= Xregn; Done <='1';
111                     when others => Gout <='1'; Rin <= Xregn; 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        => Bus_Wires <= DataIn;
151         end case;
152     end process;
153
154     -- Register instantiations
155     Reg0: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(0), D =>
    Bus_Wires, Q => R0);
156     Reg1: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(1), D =>
    Bus_Wires, Q => R1);
157     Reg2: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(2), D =>
    Bus_Wires, Q => R2);
158     Reg3: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(3), D =>
    Bus_Wires, Q => R3);
159     Reg4: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(4), D =>
    Bus_Wires, Q => R4);
160     Reg5: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(5), D =>
    Bus_Wires, Q => R5);
161     Reg6: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(6), D =>
    Bus_Wires, Q => R6);
162     Reg7: reg_n generic map(N => 9) port map (clock => clock, enable => Rin(7), D =>
    Bus_Wires, Q => R7);
163     Reg8: reg_n generic map(N => 9) port map (clock => clock, enable => Ain, D =>
    Bus_Wires, Q => A);
164     Reg9: reg_n generic map(N => 9) port map (clock => clock, enable => IRin, D => DataIn,
    Q => IR);
165     Reg10: reg_n generic map(N => 9) port map(clock => clock, enable => Gin, D =>
    std_logic_vector(sum), Q => G);
166
```

```
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;
```