

```

1  -----
2  -- Company:
3  -- Engineer:
4  --
5  -- Create Date:    18:24:43 03/14/2022
6  -- Design Name:
7  -- Module Name:    ALU_8bit - Behavioral
8  --      GitHub:    https://github.com/MasoudHeidary/
9  --      License:    MIT
10 -----
11 library IEEE;
12 use IEEE.STD_LOGIC_1164.ALL;
13
14 entity ALU_8bit is
15     Port ( DR1 : in  STD_LOGIC_VECTOR (7 downto 0);
16           DR2 : in  STD_LOGIC_VECTOR (7 downto 0);
17           Cin : in  STD_LOGIC;
18           OPCODE : in  STD_LOGIC_VECTOR (3 downto 0);
19           AC : out STD_LOGIC_VECTOR (7 downto 0);
20           Cout : out STD_LOGIC);
21 end ALU_8bit;
22
23 architecture Behavioral of ALU_8bit is
24     signal OPCODE_A          : STD_LOGIC_VECTOR (7 downto 0);
25     signal OPCODE_AND_A_B    : STD_LOGIC_VECTOR (7 downto 0);
26     signal OPCODE_OR_A_B     : STD_LOGIC_VECTOR (7 downto 0);
27     signal OPCODE_XOR_A_B    : STD_LOGIC_VECTOR (7 downto 0);
28     signal OPCODE_NOT_A      : STD_LOGIC_VECTOR (7 downto 0);
29
30     signal A                  : STD_LOGIC_VECTOR (7 downto 0);
31     signal AND_A_B            : STD_LOGIC_VECTOR (7 downto 0);
32     signal OR_A_B             : STD_LOGIC_VECTOR (7 downto 0);
33     signal XOR_A_B            : STD_LOGIC_VECTOR (7 downto 0);
34     signal NOT_A              : STD_LOGIC_VECTOR (7 downto 0);
35 begin
36     -- instruction 1: A
37     OPCODE_A(0) <= (not OPCODE(0)) and (not OPCODE(1)) and (not OPCODE(2)) and (not
OPCODE(3) and (not Cin));
38     OPCODE_A(1) <= OPCODE_A(0);
39     OPCODE_A(2) <= OPCODE_A(0);
40     OPCODE_A(3) <= OPCODE_A(0);
41     OPCODE_A(4) <= OPCODE_A(0);
42     OPCODE_A(5) <= OPCODE_A(0);
43     OPCODE_A(6) <= OPCODE_A(0);
44     OPCODE_A(7) <= OPCODE_A(0);
45
46     A <= DR1 and OPCODE_A;
47
48
49     -- instruction 2: A and B
50     OPCODE_AND_A_B(0) <= (not OPCODE(0)) and (not OPCODE(1)) and OPCODE(2) and (not
OPCODE(3));
51     OPCODE_AND_A_B(1) <= OPCODE_AND_A_B(0);
52     OPCODE_AND_A_B(2) <= OPCODE_AND_A_B(0);
53     OPCODE_AND_A_B(3) <= OPCODE_AND_A_B(0);
54     OPCODE_AND_A_B(4) <= OPCODE_AND_A_B(0);
55     OPCODE_AND_A_B(5) <= OPCODE_AND_A_B(0);
56     OPCODE_AND_A_B(6) <= OPCODE_AND_A_B(0);
57     OPCODE_AND_A_B(7) <= OPCODE_AND_A_B(0);
58
59     AND_A_B <= DR1 and DR2 and OPCODE_AND_A_B;

```

```

60
61
62 -- instruction 3: A or B
63 OPCODE_OR_A_B(0) <= OPCODE(0) and (not OPCODE(1)) and OPCODE(2) and (not OPCODE(3) and
(not Cin));
64 OPCODE_OR_A_B(1) <= OPCODE_OR_A_B(0);
65 OPCODE_OR_A_B(2) <= OPCODE_OR_A_B(0);
66 OPCODE_OR_A_B(3) <= OPCODE_OR_A_B(0);
67 OPCODE_OR_A_B(4) <= OPCODE_OR_A_B(0);
68 OPCODE_OR_A_B(5) <= OPCODE_OR_A_B(0);
69 OPCODE_OR_A_B(6) <= OPCODE_OR_A_B(0);
70 OPCODE_OR_A_B(7) <= OPCODE_OR_A_B(0);
71
72 OR_A_B <= (DR1 or DR2) and OPCODE_OR_A_B;
73
74
75 -- instruction 3: A xor B
76 OPCODE_XOR_A_B(0) <= (not OPCODE(0)) and OPCODE(1) and OPCODE(2) and (not OPCODE(3));
77 OPCODE_XOR_A_B(1) <= OPCODE_XOR_A_B(0);
78 OPCODE_XOR_A_B(2) <= OPCODE_XOR_A_B(0);
79 OPCODE_XOR_A_B(3) <= OPCODE_XOR_A_B(0);
80 OPCODE_XOR_A_B(4) <= OPCODE_XOR_A_B(0);
81 OPCODE_XOR_A_B(5) <= OPCODE_XOR_A_B(0);
82 OPCODE_XOR_A_B(6) <= OPCODE_XOR_A_B(0);
83 OPCODE_XOR_A_B(7) <= OPCODE_XOR_A_B(0);
84
85 XOR_A_B <= (DR1 xor DR2) and OPCODE_XOR_A_B;
86
87
88 -- instruction 4: not A
89 OPCODE_NOT_A(0) <= OPCODE(0) and OPCODE(1) and OPCODE(2) and (not OPCODE(3));
90 OPCODE_NOT_A(1) <= OPCODE_NOT_A(0);
91 OPCODE_NOT_A(2) <= OPCODE_NOT_A(0);
92 OPCODE_NOT_A(3) <= OPCODE_NOT_A(0);
93 OPCODE_NOT_A(4) <= OPCODE_NOT_A(0);
94 OPCODE_NOT_A(5) <= OPCODE_NOT_A(0);
95 OPCODE_NOT_A(6) <= OPCODE_NOT_A(0);
96 OPCODE_NOT_A(7) <= OPCODE_NOT_A(0);
97
98 NOT_A <= (not DR1) and OPCODE_NOT_A;
99
100 -- output
101 AC <= A or AND_A_B or OR_A_B or XOR_A_B or NOT_A;
102
103 end Behavioral;

```