

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY ALUMUX IS
5      PORT (
6          ALUOP_IN      : IN STD_LOGIC_VECTOR(2 DOWNTO 0);
7          AND_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
8              "00000000000000000000000000000000";
9          SRL_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
10             "00000000000000000000000000000000";
11         SRA_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
12             "00000000000000000000000000000000";
13         SLL_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
14             "00000000000000000000000000000000";
15         OOR_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
16             "00000000000000000000000000000000";
17         ADD_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
18             "00000000000000000000000000000000";
19         SUB_BUFFER_IN : IN STD_LOGIC_VECTOR(31 DOWNTO 0) :=
20             "00000000000000000000000000000000";
21         CARRY_ADD_IN  : IN STD_LOGIC;
22         CARRY_SUB_IN  : IN STD_LOGIC;
23         ZERO_ADD_IN   : IN STD_LOGIC;
24         ZERO_SUB_IN   : IN STD_LOGIC;
25         OVER_SUB_IN   : IN STD_LOGIC;
26         OVER_ADD_IN   : IN STD_LOGIC;
27         CLOCK_IN      : IN STD_LOGIC;
28         RESULT_MUX    : OUT STD_LOGIC_VECTOR(31 DOWNTO 0) :=
29             "00000000000000000000000000000000";
30         ZERO_MUX      : OUT STD_LOGIC := '0';
31         CARRY_MUX     : OUT STD_LOGIC := '0';
32         OVER_MUX      : OUT STD_LOGIC := '0');
33
34  END ALUMUX;
35
36  ARCHITECTURE STRUCTURAL OF ALUMUX IS
37
38  BEGIN
39      PROCESS(CLOCK_IN)
40      BEGIN
41          IF (CLOCK_IN = '1') THEN
42              CASE ALUOP_IN is
43                  WHEN "000" => -- ADDER
44                      RESULT_MUX <= ADD_BUFFER_IN;
45                      CARRY_MUX  <= CARRY_ADD_IN;
46                      ZERO_MUX   <= ZERO_ADD_IN;
47                      OVER_MUX   <= OVER_ADD_IN;
48                  WHEN "001" => -- SUBBER
49                      RESULT_MUX <= SUB_BUFFER_IN;
50                      CARRY_MUX  <= CARRY_SUB_IN;
51                      ZERO_MUX   <= ZERO_SUB_IN;
52                      OVER_MUX   <= OVER_SUB_IN;
53                  WHEN "010" => -- ANDDER
54                      RESULT_MUX <= AND_BUFFER_IN;
55                  WHEN "011" => -- ORRER
56                      RESULT_MUX <= OOR_BUFFER_IN;
57                  WHEN "100" => -- shift_left_logical
58                      RESULT_MUX <= SLL_BUFFER_IN;
59                  WHEN "101" => -- shift_right_logical
60                      RESULT_MUX <= SRL_BUFFER_IN;
61                  WHEN "110" => -- shift_LEFT_arithmetic
62                      RESULT_MUX <= SLL_BUFFER_IN;
63                  WHEN "111" => -- shift_right_arithmetic
64                      RESULT_MUX <= SRA_BUFFER_IN;
65                  WHEN OTHERS => -- SHOULD NOT HAPPEN
66
67              END CASE;
68          END IF;
69      END PROCESS;
70  END STRUCTURAL;

```