

1-1:4 bit 加減法器(符號)

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity adder_4_plus is
7  Port(A:in STD_LOGIC_VECTOR(3 downto 0);
8       B:in STD_LOGIC_VECTOR(3 downto 0);
9       C:in std_LOGIC;
10      M:in std_LOGIC;
11      Co:out STD_LOGIC_VECTOR(3 downto 0);
12      S:out STD_LOGIC
13  );
14  end adder_4_plus;
15
16  architecture adder_4_plus of adder_4_plus is
17  signal temp:std_LOGIC_VECTOR(4 downto 0);
18  begin
19      temp <= ('0' & A) + ('0' & B) when (M='0') else
20      ('0' & A) - ('0' & B);
21      Co <= temp(3 downto 0);
22      S <= temp(4);
23  end adder_4_plus;
24

```

The timing diagram illustrates the operation of the 4-bit adder-subtractor. The signals are as follows:

- A**: 4-bit vector, values range from 0000 to 000F.
- B**: 4-bit vector, values range from 0000 to 000F.
- C**: Single bit, values are 0 or 1.
- M**: Single bit, values are 0 or 1.
- Co**: 4-bit vector, values range from 0000 to 000F.
- S**: Single bit, values are 0 or 1.

The diagram shows that when M=0, the circuit performs addition (A+B), and when M=1, it performs subtraction (A-B). The carry-out (Co) and sum/difference (S) are correctly calculated based on the input values and the mode (M).

1-2:加減法器(邏輯閘)

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity 4_b is
    Port (A:in STD_LOGIC_VECTOR(3 downto 0);
          B:in STD_LOGIC_VECTOR(3 downto 0);
          C:in std_LOGIC;
          M:in std_LOGIC;
          Co:out STD_LOGIC_VECTOR(3 downto 0);
          S:out STD_LOGIC
        );
end 4_b;

architecture 4_b of 4_b is
    signal temp:std_LOGIC_VECTOR(4 downto 0);
begin
    if (M=='0') then
        temp(0)=A(0)AND B(0);
        temp(1)=(A(1)AND B(1)) XOR temp(0);
        temp(2)=(A(2)AND B(2)) XOR temp(1);
        temp(3)=(A(3)AND B(3)) XOR temp(2);
        Co=temp;
        S= (A(3)AND B(3)) AND temp(2);
    else
        temp(3)=A(3) XOR B(3);
        temp(2)=(A(2) XOR B(2))XOR temp(3);
        temp(1)=(A(1) XOR B(1))XOR temp(2);
        temp(0)=(A(0) XOR B(0))XOR temp(1);
        Co=temp;
        S='0';
    end 4_b;
```

1-3 8 bit multiplexer

```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity mux_8 is
7  Port (DIN:in STD_LOGIC;
8        S:in STD_LOGIC_VECTOR(2 downto 0);
9        Y:out STD_LOGIC_VECTOR(7 downto 0)
10       );
11 end mux_8;
12
13 architecture mux_8 of mux_8 is
14   signal x:std_LOGIC_VECTOR(1 downto 0);
15 begin
16   Y<="1111111" &DIN when S="000" else
17     "111111" &DIN & "1" when S="001" else
18     "11111" &DIN & "11" when S="010" else
19     "1111" &DIN & "111" when S="011" else
20     "111" &DIN & "1111" when S="100" else
21     "11" &DIN & "11111" when S="101" else
22     "1" &DIN & "111111" when S="110" else
23     DIN & "1111111" when S="111";
24 end mux_8;

```

