آزمایشگاه سیستم های دیجیتال

گزارش آزمایش شماره ۳

سهیل مهریزی ، آرش جوادی

شرح:

در این آزمایش خواسته شده تا عملکرد رفتاری واحد منطق و محاسبه کامپیوتر مبنا را شبیه سازی کنیم .

واحد منطق و محاسبه :

در کامپیوتر مبنا، واحد منطق و محاسبه،وظیفه انتخاب عملیات های منطقی (OR, AND, XOR, NOT) و همچنین عملیات های محسباتی نظیر (Sum, Sub, transmission, ...) و اعمال آنها بر روی داده های ورودی را برعهده دارد .

واحد منطق و محاسبه خواسته شده در این کارگاه دارای ۲ ورودی ۱۶ بیت ، ۷ عملیات محاسباتی و ۴ عملیات منطقی است ، که باید با توجه به خط انتخاب ورودی ، عملیات مدنظر برروی ورودی ها اعمال شود و خروجی هارا آماده کند ،

برای عملیات های محسباتی ممکن است پس از محاسبه رقم نقلی ایجاد شود که در Carr_out ذخیره میشود .

برنامه رفتاری واحد منطق و محاسبه خواسته شده به شکل زیر است :

```vhdl
library ieee;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use ieee.numeric_std.all;

  entity ALU is
    port (
      s : in std_logic_vector(3 downto 0);
      A : in std_logic_vector(15 downto 0);
      B : in std_logic_vector(15 downto 0);
      out_put : out std_logic_vector(15 downto 0);
      carry_out : out std_logic
    );
  end ALU;

architecture Behavioral of ALU is
  signal ALU_result : std_logic_vector(15 downto 0);
  signal temp: std_logic_vector(16 downto 0);

  begin
  with s select
      -- Arithmatic Unit
      ALU_result <= A + B when "0000",
      A + B + '1' when "0001",
      A - B when "0010",
      A+ B + '1' when "0011",
      A when "0100",
      A + '1' when "0101",
      A - '1' when "0110",
      A when "0111",
      -- Logic Unit
      A and B when "1000",
      A or B when "1001",
      A xor B when "1010",
      not A when others;

  out_put <= ALU_result;
  temp <= ('0' & A) + ('0' & B);
  carry_out <= temp(16);

end Behavioral;
```

در این برنامه مشاهده میشود
که یک ورودی ۴ بیت برای
سلکتور ، و ۲ ورودی ۱۶ بیت
برای داده های ورودی
یک خروجی ۱۶ بیت
و یک بیت نقلی خروجی

در بخش Architecture
این برنامه ، ۲ سیگنال یکی ۱۵ بیت
برای گرفتن خروجی عملیات های
انجام شده برروی داده های ورودی
و دیگری با اندازه ۱۷ بیت
برای پیدا کردن بیت نقلی
همانطور که مشاهده میکنید از بعد
خط ۲۱ مقدار بردار سلکتور نوع
عملیات را انتخاب میکند
که اگر سلکتور در باز زیر باشد
(”0111”_”0000”)
انتخاب میان عملیات های
محاسباتی است و بعد از آن تا
وضعیت”1010” انتخاب میان عملیات های منطقی است .

در خط ۳۷ محتوای سیگنال ALU result به خروجی منتقل میشود

و با استفاده از رابطه خط ۳۸ بیت نقلی احتمالی عملیات های محاسباتی بدست می‌آید.

تست بنچ این برنامه شبیه ساز رفتاری واحد منطق و محاسبه به صورت زیر است :

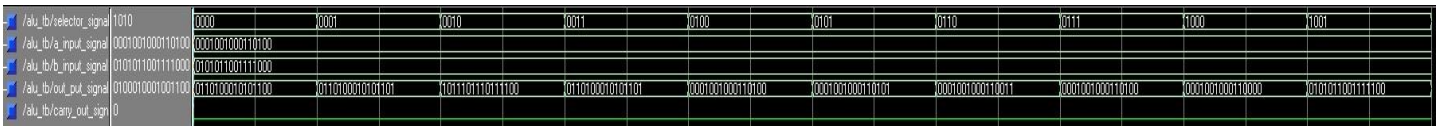
```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity ALU_tb is
end ALU_tb;
architecture test_bench of ALU_tb is
Component ALU
    port (
      s : in std_logic_vector(3 downto 0);
      A : in std_logic_vector(15 downto 0);
      B : in std_logic_vector(15 downto 0);
      out_put : out std_logic_vector(15 downto 0);
      carry_out : out std_logic
    );
end Component;

signal selector_signal : std_logic_vector(3 downto 0) := "0000";
signal A_input_signal : std_logic_vector(15 downto 0) := (others => '0');
signal B_input_signal : std_logic_vector(15 downto 0) := (others => '0');
signal out_put_signal : std_logic_vector(15 downto 0) := (others => '0');
signal carry_out_signal: std_logic := '0';

begin
p1: ALU port map (
    selector_signal, A_input_signal, B_input_signal, out_put_signal, carry_out_signal
);
process
begin
-- initializing input signals
A_input_signal <= "0001001000110100";
B_input_signal <= "0101011001111000";
-- slector changing states
selector_signal <= "0000";-- Sum(A, B)
wait for 10 ns;
selector_signal <= "0001";-- Sum(Sum(A, B), '1')
wait for 10 ns;
selector_signal <= "0010";-- Subtract(A, B)
wait for 10 ns;
 selector_signal <= "0011";-- Passing the A itself
 wait for 10 ns;
 selector_signal <= "0100";-- Sum(Sum(A, B), '1')
 wait for 10 ns;
 selector_signal <= "0101";-- Sum(A, '1')
 wait for 10 ns;
 selector_signal <= "0110";-- Subtract(A, '1')
 wait for 10 ns;
 selector_signal <= "0111";-- Passing the A itself
 wait for 10 ns;
 selector_signal <= "1000";-- and(A, B)
 wait for 10 ns;
 selector_signal <= "1001";-- or(A, B)
 wait for 10 ns;
 selector_signal <= "1010";-- xor(A, B)
 wait for 10 ns;
 selector_signal <= "1011";-- not(A)

 end process;

 end test_bench;
```

در تست بنچ این برنامه پس از تعریف وارد کردن کامپوننت برنامه ، سیگنال هایی را که میخواهیم با port map به کامپوننت وصل کنیم و مقدار دهی کنیم را مشخص میکنیم .

سپس مقدار اولیه ای برای سیگنال های ورودی در نظر میگیریم و هر ۱۰ نانوثانیه یکبار مقدار سلکتور را تغییر میدهیم که منجر به تغییر عملیات میشود .

همانطور که مشاهده میکنید انتظار داریم ، هر ۱۰ نانوثانیه اثر یکی از عملیات های محاسباتی و منطقی را مشاهده نماییم .

شکل موج های شبیه سازی شده این این تست بنچ بصورت زیر است که نشان میدهد کاملا این برنامه یک واحد منطق و محاسبه را مدل کرده است .


/alu_tb/selector_signal
1010
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
/alu_tb/a_input_signal
0001001000110100
0001001000110100
/alu_tb/b_input_signal
0101011001111000
0101011001111000
/alu_tb/out_put_signal
0100010001001100
0110100010101100
0110100010101101
1011101110111100
0110100010101101
0001001000110100
0001001000110101
0001001000110011
0001001000110100
0001001000110000
0101011001111100
/alu_tb/carry_out_sign
0