گزارشکار آزمایش ۷ و ۸ درس آزمایشگاه معماری

اعضای گروه:

عرشیا آرین‌نژاد، حوریه سبزواری، الناز رضایی

هدف آزمایش:

واحد محاسبه و منطق در پردازنده ها بخشی جداگانه و حساس است که در تمامی پردازنده های قدیمی و جدید قرار دارد. این واحد که برای محاسبه منطق و عملیات حساب استفاده می شود یکی از بخش های حساس و مهم در پردازنده ها را تشکیل می دهد. فرقی نمی‌کند که پردازنده به عنوان یک CPU رایانه مورد استفاده قرار گیرد و یا یک میکروچیپ ساده باشد.

واحد ALU به عنوان یک واحد اساسی و مهم در پردازنده ها شناخته می شود. حتی ساده ترین پردازنده ها و میکروچیپ ها نیز این واحد را در اختیار دارند و ساده ترین بخش های محاسباتی را نیز در این واحد انجام می دهند. یکی از مهمترین کارها در محاسبه و منطق محاسبه زمان و نگهداری آن است.

واحد ALU مخفف کلمه Arithmetic Logic Unit است. این واحد کار انجام عملیات های محاسباتی را در پردازنده ها انجام می دهند. منظور از عملیات های محاسباتی منطقی عملیات های جمع و تفریق و ضرب است. همچنین عملیات های منطقی مانند عملیات های AND – OR – XOR و Not نیز در این واحد انجام می شود. شرایط انجام عملیات های منطقی در واحد محاسبه و منطق با عملیات های محاسباتی کمی متفاوت است و مقایسه صحنم و اشتباه بودن این عملیات ها در این واحد انجام می شود.

مهمترین خروجی های واحد ALU عبارت‌اند از:

۱. رقم نقلی
این نوع رقم ها معمولاً در عملیات های جمع رخ می دهد.

۲. رقم قرضی:
این نوع رقم ها معمولاً در عمل تفریق به وجود می آیند.

۳. بیت سرریز:
بیت سرریز یا همان Overflow که معمولاً در اثر عمل شیفت باینری رخ می دهد.

در این آزمایش با ALU و نحوه‌ی پیاده‌سازی آن در زبان vhdl آشنا شدیم و عملیات مختلف را تست کردیم. از جمله عملیات قابل انجام تقسیم است که کد آن به شرح زیر است:

Code:

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use ieee.std_logic_unsigned.all;
4
5  entity div_8_4 is
6      Port ( a : in  STD_LOGIC_VECTOR(7 downto 0);
7             b : in  STD_LOGIC_VECTOR(3 downto 0);
8             q : out  STD_LOGIC_VECTOR(7 downto 0);
9             r : out  STD_LOGIC_VECTOR(3 downto 0));
10 end div_8_4;
11
12 architecture behavior of div_8_4 is
13 procedure div4(
14       num1 : in  STD_LOGIC_VECTOR(7 downto 0);
15       num2 : in  STD_LOGIC_VECTOR(3 downto 0);
16       q : out  STD_LOGIC_VECTOR(3 downto 0);
17       r  : out  STD_LOGIC_VECTOR(3 downto 0)) is
18
19       variable d,n1 : STD_LOGIC_VECTOR(4 downto 0);
20       variable n2 : STD_LOGIC_VECTOR(3 downto 0);
21   begin
22   d := '0' & num2;
23   n2:= num1(3 downto 0);
24   n1:= '0' & num1(7 downto 4);
25   for i in 0 to 3 loop
26     n1 := n1( 3 downto 0) & n2(3);
27     n2 := n2(2 downto 0) & '0';
28     if n1 >= d then
29       n1 := n1 - d;
30       n2(0) := '1';
31     end if;
32   end loop;
33   q := n2;
34   r := n1(3 downto 0);
35   end div4;
36 begin
37   process(a,b)
38   variable r_H , r_L, q_H , q_L :STD_LOGIC_VECTOR(3 downto 0);
39   begin
40   div4("0000" & a( 7 downto 4) ,b ,q_H ,r_H);
41   div4(r_H & a(3 downto 0) ,b , q_L, r_L);
42   q( 7 downto 4) <= q_H;
43   q(3 downto 0) <= q_L;
44   r <= r_L;
45   end process;
46   end behavior;
```

TestBench Code:

```
65      -- Instantiate the Unit Under Test (UUT)
66      uut: div_8_4 PORT MAP (
67             a => a,
68             b => b,
69             q => q,
70             r => r
71           );
72
73      -- Clock process definitions
74
75
76      -- Stimulus process
77      stim_proc: process
78      begin
79         -- hold reset state for 100 ns.
80         wait for 100 ns;
81
82            a <= "01110100";
83            b <= "1100";
84         -- insert stimulus here
85
86         wait;
87      end process;
88
89  END;
```

TestBench Output: