پروژه alu

استاد درس: دکتر شریعتمدار مدیر پروژه: مهندس گوهری دستیار پروژه: مهندس رزاقی

دانشجو: محمدحسین کاشانکی – علی امیری

```vhdl
entity ALU is
    Port (
          clk           : in std_logic; -- سیگنال ساعت
          newvalue      : in STD_LOGIC; -- سیگنال ورودی برای تعیین مقدار جدید
          a             : in STD_LOGIC_VECTOR (32-1 downto 0); -- با عرض 31 بیت a ورودی
          alucontroler  : in STD_LOGIC_VECTOR (3-1 downto 0); -- با عرض 2 بیت ALU ورودی کنترل
          b             : in STD_LOGIC_VECTOR (32-1 downto 0); -- با عرض 31 بیت b ورودی

          z             : out STD_LOGIC; -- خروجی زرو
          o             : out STD_LOGIC; -- خروجی اورفلو
          n             : out STD_LOGIC; -- خروجی نگاتیو
          rdy           : out STD_LOGIC; -- خروجی آماده

          c             : out STD_LOGIC; -- خروجی کاری
          y             : out STD_LOGIC_VECTOR (32-1 downto 0)); -- با عرض 31 بیت y خروجی
end ALU;
```

پورت‌های ورودی و خروجی آن را مشخص می‌کند

- **clk**: سیگنال ساعت
- **newvalue**: سیگنال ورودی برای تعیین مقدار جدید
- **a**, **b**: ورودی‌های 31 بیتی
- **alucontroler**: سیگنال کنترل ALU با عرض 2 بیت
- **z**, **o**, **n**, **rdy**, **c**: سیگنال‌های خروجی مختلف
- **y**: خروجی 31 بیتی

تعریف معماری (Architecture)

```
architecture Behavioral of ALU is
```

```
component multi_16 is
    Port (
            clk   : in STD_LOGIC; -- سیگنال ساعت
            a     : in std_logic_vector(32-1 downto 0); -- با عرض 31 بیت a ورودی
            b     : in std_logic_vector(32-1 downto 0); -- با عرض 31 بیت b ورودی
            start : in std_logic; -- سیگنال شروع
            rdy   : out std_logic; -- سیگنال آماده
            o     : out std_logic_vector(32-1 downto 0) -- با عرض 31 بیت o خروجی
            );
end component;
```

این بخش کامپوننت **multi_16** را تعریف می‌کند که یک ضرب‌کننده است و پورت‌های ورودی و خروجی آن را مشخص می‌کند:

- **clk**: سیگنال ساعت
- **a**, **b**: ورودی‌های 31 بیتی
- **start**: سیگنال شروع
- **rdy**: سیگنال آماده
- **o**: خروجی 31 بیتی

تعریف سیگنال‌ها

```
signal add_op : std_logic_vector(33-1 downto 0) :=( others=>'0'); -- سیگنال جمع با عرض 32 بیت و مقدار اولیه صفر
signal sub_op : std_logic_vector(33-1 downto 0) :=( others=>'0'); -- سیگنال تفریق با عرض 32 بیت و مقدار اولیه صفر
signal mult_op : std_logic_vector(32-1 downto 0) :=( others=>'0'); -- سیگنال ضرب با عرض 31 بیت و مقدار اولیه صفر
signal slt_op : std_logic_vector(32-1 downto 0) :=( others=>'0'); -- با عرض 31 بیت و مقدار اولیه صفر سیگنال SLT
signal And_Op : std_logic_vector(32-1 downto 0) :=( others=>'0'); -- با عرض 31 بیت و مقدار اولیه صفر سیگنال AND
signal or_op : std_logic_vector(32-1 downto 0) :=( others=>'0'); -- با عرض 31 بیت و مقدار اولیه صفر سیگنال OR
signal xor_op : std_logic_vector(32-1 downto 0) :=( others=>'0'); -- با عرض 31 بیت و مقدار اولیه صفر سیگنال XOR
signal out_put : std_logic_vector(32-1 downto 0) :=( others=>'0'); -- سیگنال خروجی با عرض 31 بیت و مقدار اولیه صفر
signal start : std_logic := '0'; -- سیگنال شروع با مقدار اولیه صفر
signal rdy_m : std_logic := '0'; -- سیگنال آماده داخلی با مقدار اولیه صفر
signal i : integer := 0; -- سیگنال شمارنده با مقدار اولیه صفر
signal a2 : signed (32-1 downto 0); -- و عرض 31 بیت signed سیگنال a با نوع
signal b2 : signed (32-1 downto 0); -- و عرض 31 بیت signed سیگنال b با نوع
```

سیگنال‌های داخلی مختلفی را تعریف میکنیم که در پیاده‌سازی مورد استفاده قرار می‌گیرند. این سیگنال‌ها شامل عملیات ALU جمع، تفریق، ضرب، SLT، AND، OR، XOR و دیگر سیگنال‌های کنترلی و میانی است.

```
begin
inst_mult: multi_16
    Port map (
            clk => clk, -- اتصال سیگنال ساعت به کامپونننت
            a  => a, -- به کامپونننت a اتصال ورودی
            b  => b, -- به کامپونننت b اتصال ورودی
            start => start, -- اتصال سیگنال شروع به کامپونننت
            rdy   => rdy_m, -- اتصال سیگنال آماده داخلی به خروجی آماده کامپونننت
            o  => mult_op -- اتصال خروجی ضرب به سیگنال داخلی mult_op
             );
```

اتصال ضرب کننده به alu

عملیات محاسباتی

```
add_op <= std_logic_vector(signed(a(31) & a) + signed(b(31) & b));
-- جمع دو عدد با توسعه‌یگنال‌ها  به 33 بیت برای پشتیبانی از بیت نشانه و تبدیل به نوع signed

sub_op <= std_logic_vector(signed(a(31) & a) - signed(b(31) & b));
-- تفریق دو عدد با توسعه‌یگنال‌ها  به 33 بیت برای پشتیبانی از بیت نشانه و تبدیل به نوع signed

a2 <= signed(a);
-- تبدیل سیگنال a به نوع signed و اختصاص به سیگنال a2

b2 <= signed(b);
-- تبدیل سیگنال b به نوع signed و اختصاص به سیگنال b2
```

عملیات مقایسه و منطقی

```
slt_op <= x"0000000" & "0001" when (signed(a) < signed(b)) else x"00000000";
-- تعیین نتیجه عملیات SLT (Set Less Than) با مقایسه دو عدد a و b

And_Op <= a and b;
-- عملیات منطقی AND بر روی ورودی‌های a و b

or_op <= a or b;
-- عملیات منطقی OR بر روی ورودی‌های a و b

xor_op <= a xor b;
-- عملیات منطقی XOR بر روی ورودی‌های a و b
```

```
    process (clk)
begin
    if (rising_edge(clk)) then
        z <= '0'; -- تنظیم سیگنال صفر
        o <= '0'; -- تنظیم سیگنال اورفلو
        n <= '0'; -- تنظیم سیگنال نگاتیو
        c <= '0'; -- تنظیم سیگنال کاری
        rdy <= '0'; -- تنظیم سیگنال آماده

        if (newvalue = '1') then
            i <= 1; -- به 1 شروع عملیات جدید با تنظیم شمارنده
        end if;

        if i = 1 then
            case (alucontroler) is
                when "000" =>  -- عملیات NOP (بدون عملیات)
                    y <= x"00000000"; -- به صفر y تنظیم خروجی
                    z <= '0'; -- تنظیم سیگنال صفر
                    o <= '0'; -- تنظیم سیگنال اورفلو
                    c <= '0'; -- تنظیم سیگنال کاری
                    n <= '0'; -- تنظیم سیگنال نگاتیو
                    rdy <= '1'; -- تنظیم سیگنال آماده
                    i <= 0; -- i بازنشانی شمارنده
```

clk: ورودی این ورودی سیگنال ساعت است که برای هماهنگ‌سازی عملیات ALU استفاده می‌شود.

z, o, n, c, rdy: سیگنال‌های این سیگنال‌ها به ترتیب نشان‌دهنده وضعیت‌های صفر، اورفلو، نگاتیو، کاری و آماده بودن نتیجه هستند.

if (newvalue = '1'): شرط اگر سیگنال newvalue فعال باشد، شمارنده i به 1 تنظیم می‌شود تا نشان دهد که عملیات جدیدی باید شروع شود.

if i = 1: شرط وقتی شمارنده i برابر با 1 باشد، عملیات مربوط به کنترلر ALU (بر اساس سیگنال alucontroler) اجرا می‌شود.

عملیاتAdd

```
        when "001" =>  -- Add
y <= add_op(32-1 downto 0); -- مقدار جمع شده به خروجی y  اختصاص دادهمیشود.
rdy <= '1'; --  سیگنال آماده به 1 تنظیمهمیشود.
i <= 0; -- شمارنده i  به 0 تنظیمهمیشود.
if (signed(add_op(32-1 downto 0)) /= signed(add_op)) then -- بررسی اورفلو
    o <= '1'; --  اگر اورفلو رخ داده باشد، سیگنال اورفلو به 1 تنظیمهمیشود.
end if;
if add_op(32) = '1' then -- بررسی بیت کاری
    c <= '1'; --  اگر بیت کاری فعال باشد، سیگنال کاری به 1 تنظیمهمیشود.
end if;
if add_op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
    z <= '0'; -- اگر نتیجه جمع صفر باشد، سیگنال z  به 0 تنظیمهمیشود.
end if;
if add_op(31) = '1' then -- بررسی بیت نشانه
    n <= '1'; --  اگر بیت نشانه فعال باشد، سیگنال نگاتیو به 1 تنظیمهمیشود.
end if;
```

عملیات Sub

```
        when "010" =>  -- Sub
y <= sub_op(32-1 downto 0); -- مقدار تفریق شده به خروجی y  اختصاص داده‌می‌شود.
rdy <= '1'; --  سیگنال آماده به 1 تنظیم‌می‌شود.
i <= 0; -- شمارنده i  به 0 تنظیم‌می‌شود.
if (signed(sub_op(32-1 downto 0)) /= signed(sub_op)) then -- بررسی اورفلو
    o <= '1'; --  اگر اورفلو رخ داده باشد، سیگنال اورفلو به 1 تنظیم‌می‌شود.
end if;
if sub_op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
    z <= '0'; -- اگر نتیجه تفریق صفر باشد، سیگنال z  به 0 تنظیم‌می‌شود.
end if;
if sub_op(31) = '1' then -- بررسی بیت نشانه
    n <= '1'; --  اگر بیت نشانه فعال باشد، سیگنال نگاتیو به 1 تنظیم‌می‌شود.
end if;
```

عملیات Mult

```
        when "011" =>  -- Mult
start <= '1'; --   سیگنال شروع به 1 تنظیم‌می‌شود.
if rdy_m = '1' then -- اگر سیگنال آماده داخلی فعال باشد
    y <= mult_op; -- نتیجه ضرب به خروجی y   اختصاص داده‌می‌شود.
    rdy <= '1'; --   سیگنال آماده به 1 تنظیم‌می‌شود.
    start <= '0'; --   سیگنال شروع به 0 تنظیم‌می‌شود.
    i <= 0; -- شمارنده i   به 0 تنظیم‌می‌شود.
    if mult_op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
        z <= '0'; -- اگر نتیجه ضرب صفر باشد، سیگنال z   به 0 تنظیم‌می‌شود.
    end if;
```

```
when "100" =>  -- Slt
 y <= slt_op; -- نتیجه عملیات SLT به خروجی y  اختصاص داده‌می‌شود.
 rdy <= '1'; --  سیگنال آماده به 1 تنظیم‌می‌شود.
 i <= 0; -- شمارنده i  به 0 تنظیم‌می‌شود.
 if slt_op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
     z <= '0'; -- اگر نتیجه SLT صفر باشد، سیگنال z  به 0 تنظیم‌می‌شود.
 end if;
 if slt_op(31) = '1' then -- بررسی بیت نشانه
     n <= '1'; --  اگر بیت نشانه فعال باشد، سیگنال نگاتیو به 1 تنظیم‌می‌شود.
 end if;
```

عملیات Slt: بررسی می‌کند که آیا `a` کمتر از `b` است و نتیجه را در `y` ذخیره می‌کند. همچنین نتیجه صفر و بیت نشانه را بررسی و تنظیم می‌کند.

```
        when "101" =>  -- And
y <= And_Op; -- نتیجه عملیات AND به خروجی y  اختصاص داده‌شود.
rdy <= '1'; --  سیگنال آماده به 1 تنظیم‌می‌شود.
i <= 0; -- شمارنده i  به 0 تنظیم‌می‌شود.

if And_Op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
    z <= '0'; -- اگر نتیجه AND صفر باشد، سیگنال z  به 0 تنظیم‌می‌شود.
end if;

                    --   if  And_Op(31) = '1' then
                      --     n <= '1' ;
                      -- end if;

       when "110" =>  -- Or
y <= or_op; -- نتیجه عملیات OR به خروجی y  اختصاص داده‌شود.
rdy <= '1'; --  سیگنال آماده به 1 تنظیم‌می‌شود.
i <= 0; -- شمارنده i  به 0 تنظیم‌می‌شود.

if or_op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
    z <= '0'; -- اگر نتیجه OR صفر باشد، سیگنال z  به 0 تنظیم‌می‌شود.
end if;

   when "111" =>  -- Xor
 y <= xor_op; -- نتیجه عملیات XOR به خروجی y  اختصاص داده‌شود.
 rdy <= '1'; --  سیگنال آماده به 1 تنظیم‌می‌شود.
 i <= 0; -- شمارنده i  به 0 تنظیم‌می‌شود.

 if xor_op(32-1 downto 0) = x"00000000" then -- بررسی نتیجه صفر
     z <= '0'; -- اگر نتیجه XOR صفر باشد، سیگنال z  به 0 تنظیم‌می‌شود.
 end if;

                    --    if  xor_op(31) = '1' then
                     --      n <= '1' ;
                     --   end if;
```

در آخر در صورت عدم تطابق با هیچ‌کدام از کدهای کنترلی، تمام سیگنال‌ها به حالت صفر بازنشانی می‌شوند.

به دلیل جلوگیری از وضعیت ناخواسته

سپاس از توجه شما