پروژه ضرب کننده 16 بیتی

استاد درس: دکتر شریعتمدار مدیر پروژه: مهندس گوهری دستیار پروژه: مهندس غازاریان – مهندس شفیع زاده

دانشجو: محمدحسین کاشانکی – علی امیری

```
library IEEE; -- استفاده از کتابخانه IEEE
use IEEE.STD_LOGIC_1164.ALL; -- 1164 استفاده از تمامی اجزای استاندارد منطق IEEE
use IEEE.STD_LOGIC_unsigned.ALL; -- استفاده از تمامی اجزای استاندارد منطق بدون علامت IEEE
```

این بخش کتابخانه‌های استاندارد را وارد می‌کند که انواع داده‌ها و توابع منطقی را فراهم می‌کند

```
entity multiplier_phase1 is -- تعریف موجودیت (entity) فاز 1   ضربکننده
    Port ( a : in STD_LOGIC_vector(31 downto 0); -- ورودی a بیت 32 با عرض
           b : in STD_LOGIC_vector(31 downto 0); -- ورودی b بیت 32 با عرض
           y : out STD_LOGIC_vector(31 downto 0); -- خروجی y بیت 32 با عرض
           start : in std_logic; -- سیگنال ورودی شروع
           ready : out std_logic; -- سیگنال خروجی آماده
           clk : in STD_LOGIC); -- سیگنال ورودی ساعت
end multiplier_phase1;
```

این بخش تعریف موجودیت (entity) :
است که پورت‌های ورودی و خروجی را مشخص می‌کند

- **a** و **b**: ورودی‌های 32 بیتی
- **y**: خروجی 32 بیتی
- **start**: سیگنال شروع
- **ready**: سیگنال آماده بودن نتیجه
- **clk**: سیگنال خود کلاک

```
architecture Behavioral of multiplier_phase1 is -- تعریف معماری رفتاری
signal product : std_logic_vector(31 downto 0):= (others =>'0'); -- 0 سیگنال محصول با مقدار اولیه
signal multiplicand : std_logic_vector(31 downto 0) :=(others =>'0'); -- 0 سیگنال مضروب با مقدار اولیه
signal multiplier : std_logic_vector(15 downto 0) := (others =>'0'); -- 0 سیگنال ضربکننده با مقدار اولیه
signal in_product : std_logic_vector(31 downto 0) := (others =>'0'); -- 0 سیگنال مقدار میانی محصول با مقدار اولیه
signal counter : integer := 1; -- 1 شمارنده با مقدار اولیه
type state1 is (idle, multiply); --  برای حالات ماشین حالت state1 تعریف نوع داده ای
signal state : state1 := idle; -- سیگنال حالت با مقدار اولیه idle
```

این بخش تعریف معماری رفتاری (Behavioral)
سیگنال‌ها و انواع داخلی به شرح زیر تعریف می‌شوند:

- **product**: حاصل ضرب موقت
- **multiplicand**: مضروب
- **multiplier**: ضرب‌کننده
- **in_product**: مقدار میانی حاصل ضرب
- **counter**: شمارنده برای کنترل مراحل ضرب
- **state1**: نوع داده‌ای برای حالات ماشین حالت
- **state**: سیگنال حالت که مقدار اولیه آن **idle** است.

```
begin
    in_product <= multiplicand + product ; -- تعیین مقدار in_product به عنوان جمع مضروب و محصول
    y <= product; -- برابر با مقدار سیگنال محصول y خروجی
```

این بخش مقادیر سیگنال‌ها را به‌روزرسانی می‌کند:

- **in_product**: جمع مضروب و حاصل ضرب موقت
- **y**: مقدار نهایی حاصل ضرب

```
process (clk) -- آغاز فرایند حساس به سیگنال ساعت
begin
 if rising_edge(clk) then -- اگر لبه بالارونده ساعت تشخیص داده شد
     case state is -- بررسی حالت ماشین حالت
         when idle => -- اگر حالت idle است
                 ready <= '0'; -- غیرفعال کردن سیگنال آماده
                 if start = '1' then -- اگر سیگنال شروع فعال است
                     multiplier <= b(15 downto 0); -- اختصاص دادن b مقدار 16 بیت پایین   را بضربکننده
                     multiplicand <= ("0000000000000000") & a(15 downto 0); -- به مضروب اختصاص دادن و 16 بیت بالا را 0 کردن a مقدار 16 بیت پایین
                     product <= (others =>'0'); -- تنظیم محصول به مقدار صفر
                     state <= multiply; -- تغییر حالت به multiply
                 else
                     state <= idle; -- در غیر این صورت، حالت به idle   باقیبماند
                 end if;
```

با هر لبه بالارونده سیگنال اجرا می‌شود:

- اگر حالت **idle** باشد:
  - **ready** غیرفعال است (**'0'**).
  - اگر **start** فعال باشد (**'1'**):
    - **multiplier** مقدار 16 بیت پایین **b** را دریافت می‌کند.
    - **multiplicand** مقدار 16 بیت پایین **a** را دریافت می‌کند و به 16 بیت بالا 0 افزوده می‌شود.
    - **product** به صفر تنظیم می‌شود.
    - حالت به **multiply** تغییر می‌کند.
  - در غیر این صورت، حالت به **idle** باقی می‌ماند.

```
        end if;
when multiply => -- اگر حالت multiply است
         if counter <17 then -- اگر شمارنده کمتر از 17 است
             multiplier <= '0' & multiplier(15 downto 1); --   شیفت راست‌بکننده
             multiplicand <= multiplicand(30 downto 0) & '0'; -- شیفت چپ مضروب
             counter <= counter +1; -- افزایش شمارنده
             if multiplier(0) = '1' then -- اگر بیت پایین‌ضربکننده 1 است
                product <= in_product; -- تنظیم محصول به مقدار in_product
             end if;
         elsif counter = 17 then -- اگر شمارنده برابر 17 است
            ready <= '1'; -- فعال کردن سیگنال آماده
            counter <= 1; -- تنظیم شمارنده به 1
            state <= idle; -- تغییر حالت به idle
         end if;
case;
```

- اگر حالت **multiply** باشد:
  - اگر شمارنده کمتر از 17 باشد:
    - بیت‌های **multiplier** به یک بیت به راست شیفت می‌شوند.
    - **multiplicand** به یک بیت به چپ شیفت می‌شود.
    - شمارنده افزایش می‌یابد.
    - اگر بیت پایین **multiplier** یک باشد، **product** به **in_product** تنظیم می‌شود.
  - اگر شمارنده برابر 17 باشد:
    - **ready** فعال می‌شود (**'1'**).
    - شمارنده به 1 تنظیم می‌شود.
    - حالت به **idle** تغییر می‌کند.