حافظه‌های ROM و BRAM

فاطمه علی‌ملکی
مهدی حق‌وردی

۲۲ آذر ۱۴۰۲

فهرست مطالب

۱ توضیحات فایل‌های تمرین

۱.۱ فایل rom8x8

این فایل، ساده‌ترین فایل این تمرین است که شامل یک موجودیت به نام rom8x8 و یک معماری به نام behave است.

۱.۱.۱ توضیحات موجودیت rom8x8

این موجودیت شامل دو پورت است:

- addr

این پورت یک ورودی std_logic_vector ۳ بیتی است که در واقع مقدار خروجی را تعیین می‌کند.

- dout

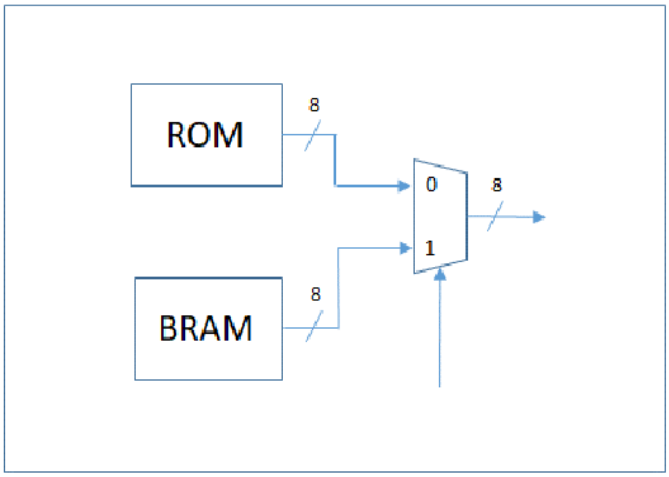
این پورت هم یک پورت خروجی از نوع std_logic_vector ۸ بیتی است که مقدار خروجی روی آن قرار می‌گیرد.

۲.۱.۱ توضیحات معماری behave

این تنها معماری حاضر برای موجودیت rom8x8 است که در واقع یک decoder ۳ به ۸ است.
مقادیری که برای خروجی انتخاب شده‌اند کاملا hard code شده‌اند و این کاملا با نام موجودیت (که rom است) هماهنگی دارد.
در معماری از language feature عی به نام select ... with استفاده شده است که با توجه به مقدار addr خروجی را روی dout قرار می‌دهد.

۲.۱ فایل romram

این فایل در واقع کل پروژه و کد این شکل است:



چرا کد این شکل است؟
چون در کد مشاهده می‌شود که مشخصا موجودیت‌های rom8x8 ، clock_divide و bram8x8 در آن به عنوان یک component استفاده شده‌اند (که موجودیت‌هایی هستند که در شکل وجود دارند، البته مالتی پلکسری که در شکل است در قسمت معماری همین فایل تعریف شده است).
کد شامل یک موجودیت به نام romram و یک معماری به نام structural است.

۱.۲.۱ موجودیت romram

موجودیت دارای دو ورودی و یک خروجی‌ست:

- ورودی‌ها

  - `clk_100MHZ`
    از نوع `std_logic` و تبعا یک بیتی‌ست که کلاک سیستم است.
  - `switch`
    از نوع `std_logic` است که همان پایه‌ی انتخاب داخل شکل پروژه است.

- خروجی

  - `leds`
    یک پورت ۸ بیتی از نوع `std_logic_vector` است مقداری انتخاب شده روی آن قرار می‌گیرد.

۲.۲.۱ معماری `structural`

در قسمت معماری این فایل، ابتدا ۳ componentی که در این پروژه استفاده شده‌اند، تعریف شده‌اند و پس از آن ۵ سینگال میانی تعریف و مقداردهی اولیه شده‌اند.
در قسمت `begin` این معماری، ابتدا ۳ component بالا اصطلاحا `port map` شده و به سینگال‌های ورودی، میانی و خروجی وصل شده‌اند.
معماری شامل یک `process` است که به سیگنال `clk_1Hz` حساس است. سپس این فرآیند برای هر یک کلاک با مقدار ۱، با توجه به ورودی `switch` که ۰ باشد یا ۱، خروجی را به ترتیب با `dout_rom` و یا `dout_bram` پر می‌کند. سپس فارغ از مقدار `switch` یک شمارنده یا یکی اضافه می‌کند.

۳.۱ فایل clockDivider

شامل یک موجودیت و یک معماری‌ست:

۱.۳.۱ موجودیت `clock_divide`

صرفا شامل یک ورودی و یک خروجی یک بیتی به نام‌های `clk_in` و `clk_out` است.

۲.۳.۱ معماری `behave`

این معماری شامل ۴ سینگال میانی و سه فرآیند است.

- سیگنال‌های میانی
  شامل دو سینگال از نوع `unsigned` به طول ۲۷ بیت، و دو سینگال یک بیتی از نوع `std_logic` است.

- فرآیندها
  قسمت معماری این فایل شامل ۳ فرایند و یک statement است که به صورت موازی اجرا می‌شوند.

کلاکی که بورد پازج در اختیار ما قرار می‌دهد، کلاک 100MHz است که چشم انسان قابلیت دیدن تغییرات آن را روی چیزی مانند LED ندارد.

برای تبدیل کردن این تعداد کلاک بسیار زیاد به یک کلاک در ثانیه می‌توان تا نصف این عدد را مقدار صفر داد و نصف دیگر را عدد یک. این فایل هم دقیقا همین کار را انجام می‌دهد. این فایل دونه دونه کلاک‌ها را می‌شمارد و یک متغیر را با هر کلاک یکی اضافه می‌کند، تا زمانی که به مقدار 50e6 که نصف 100MHz است برسد؛ و تا این زمان، سیگنالی که روی خروجی می‌افتد را (oeb_next) صفر می‌کند.

از بعد از عدد 50e6 به همین منوال، سیگنال oeb_next مقدار ۱ را داراست.

۲ تغییرات کد برای بلوک دیاگرام

بلوک دیاگرامی که در گروه فرستادید، تغییراتی را در کد romram.vhd می‌طلبید که تغییرات آن با استفاده از پلاگین git نرم‌افزار PyCharm Community Edition عکس گرفته شده‌اند.

۱.۲ ورودی‌ها

تغییرات کلی ورودی‌ها در تصویر ۱(آ) آورده شده‌اند.

۲.۲ تغییرات در سیگنال‌های میانی

تغییراتی که روی سیگنال‌های میانی انجام داده شده‌اند، در تصویر ۱(ب) آورده شده‌اند.

۳.۲ تغییرات در کامپوننت bram8x8

تغییرات لازم در تصویر ۱(ج) نشان داده شده‌اند.

۴.۲ تغییرات در فرآیند موجود

تغییرات لازم در فرآیندی که از قبل در فایل بود، در تصویر ۱(د) نشان داده شده‌اند.

۵.۲ فرآیند جدید

بخاطر آخرین مالتی‌پلکسر نیاز به نوشتن یک فرآیند جدید احساس می‌شد که کد این فرآیند در تصویر ۲ نوشته شده است.

فهرست تصاویر

(د) تغییرات فرآیند اول

(ج) bram8x8

(ب) سیگنال‌ها

(آ) ورودی‌ها

شکل ۱: تغییرات در ورودی، سیگنال‌های میانی، bram8x8 و فرآیند اول

```
    when others => null;
    end case;
    addr_counter <= std_logic_vector(unsigned(addr_cou
  end if;
end process;
end structural;
```

```
process(before_leds)
begin
  case switch1 is
    when '0' =>
      leds <= before_leds;
    when '1' =>
      leds <= null;
    when others => null;
  end case;
end process;
end structural;
```

شکل ۲: فرآیند جدید