

حافظه‌های ROM و BRAM

فاطمه علی‌ملکی

مهدی حق‌وردی

۲۲ آذر ۱۴۰۲

فهرست مطالب

۱	توضیحات فایل‌های تمرین	۱
۱	فایل rom8x8	۱.۱
۱	توضیحات موجودیت rom8x8	۱.۱.۱
۲	توضیحات معماری behave	۲.۱.۱
۲	فایل romram	۲.۱
۲	موجودیت romram	۱.۲.۱
۳	معماری structural	۲.۲.۱
۳	فایل clockDivider	۳.۱
۳	موجودیت clock_divide	۱.۳.۱
۳	معماری behave	۲.۳.۱
۴	تغییرات کد برای بلوک دیاگرام	۲
۴	ورودی‌ها	۱.۲
۴	تغییرات در سیگنال‌های میانی	۲.۲
۴	تغییرات در کامپوننت bram8x8	۳.۲
۴	تغییرات در فرآیند موجود	۴.۲
۴	فرآیند جدید	۵.۲

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.

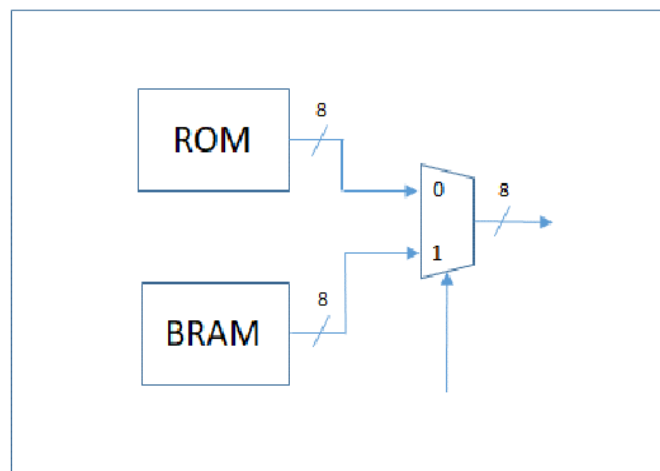
۲.۱.۱ توضیحات معماری behave

این تنها معماری حاضر برای موجودیت `rom8x8` است که در واقع یک decoder ۳ به ۸ است. مقادیری که برای خروجی انتخاب شده‌اند کاملاً `hard code` شده‌اند و این کاملاً 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 مشخصاً 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 خروجی است:

- 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 component ی 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 سینگال 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.

- سینگال‌های 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 مورد 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 دقیقاً 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

۱	تغییرات در ورودی، سیگنال‌های میانی، bram8x8 و فرآیند اول	۵
۲	فرآیند جدید	۶

```

55 process(clk_1Hz) is
56 begin
57   if(clk_1Hz'event and clk_1Hz = '1') then
58     case switch0 is
59       when '0' =>
60         before_leds <= dout_rom;
61       when '1' =>
62         before_leds <= dout_bram;
63     end case;
64     when others => null;
65     addr_counter <= std_logic_vector(unsigned(addr_cou
66   end if;
67 end process;

```

(د) تغییرات فرآیند اول

```

44 begin
45   clock_divider: clock_divide port map(clk_100MHz, clk_1Hz);
46   rom: rom8x8 port map(addr_counter, dout_rom);
47   bram: bram8x8 port map(
48     clka => clk_1Hz,
49     wea => wea_null,
50     addra => addr_counter,
51     dina => dina_null,
52     douta => dout_bram
53   );

```

(ج) bram8x8

```

37 signal clk_1Hz: std_logic;
38 signal addr_counter: std_logic_vector(2 downto 0) := "000";
39 signal dout_rom,dout_bram: std_logic_vector(7 downto 0);
40 signal dina_null: std_logic_vector(7 downto 0) := "000000000";
41 signal wea_null: std_logic_vector(0 downto 0) := "0";
42 signal before_leds: std_logic_vector(7 downto 0);
43 signal all_null: std_logic_vector(7 downto 0) := "000000000";
44
45 entity romram is
46   port(
47     clk_100MHz: in std_logic;
48     switch0, switch1: in std_logic;
49     leds: out std_logic_vector(7 downto 0)
50   );
51 end romram;

```

(ب) سیگنال‌ها

```

5 entity romram is
6   port(
7     clk_100MHz: in std_logic;
8     switch0, switch1: in std_logic;
9     leds: out std_logic_vector(7 downto 0)
10  );
11 end romram;

```

(آ) ورودی‌ها

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

```

63      when others => null;
64      end case;
65      addr_counter <= std_logic_vector(unsigned(addr_cou
66      end if;
67      end process;
68      end structural;
69
70      process(before_leds)
71      begin
72          case switch1 is
73              when '0' =>
74                  leds <= before_leds;
75              when '1' =>
76                  leds <= null;
77              when others => null;
78          end case;
79      end process;
80      end structural;

```

شکل ۲: فرآیند جدید