توضیحات آزمایش هفتم

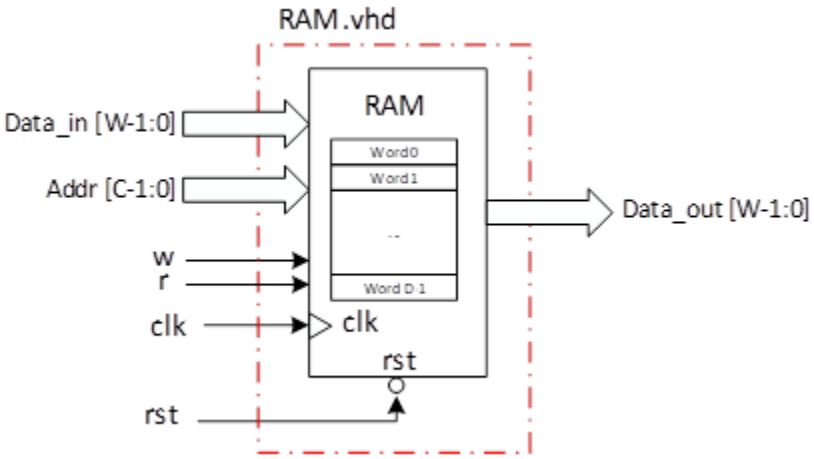
هدف از این این آزمایش آشنایی با واحد حافظه و نحوه طراحی و مدل کردن چند نوع واحد حافظه است.برای طراحی این حافظه‌ها می‌توانید از توصیف رفتاری استفاده کنید.

۱) طراحی حافظه

یک حافظه RAM مطابق با شکل زیر طراحی کنید. پارامترهای این بلوک به شرح زیر می‌باشد :

- W: عرض حافظه و برابر با ۸ در نظر گرفته شود.
- D: تعداد خانه‌های حافظه و برابر با ۱۶ در نظر گرفته شود.
- C عرض درگاه آدرس است که برابر با $\log_2(D)$ است.

در شکل ۱ پورت‌های ورودی و خروجی واحد حافظه RAM تک پورتی نشان داده شده است.



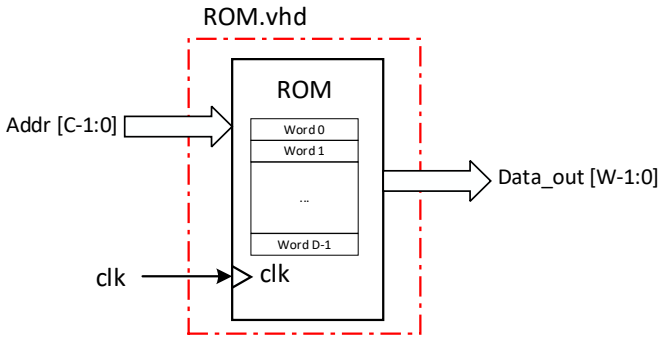
شکل ۱) واحد حافظه RAM و تعریف پورت‌های ورودی و خروجی

- data_in داده ورودی به حافظه است که باید در آدرس مورد نظر ریخته شود.
- Addr ورودی حافظه و آدرس داده ای که باید در حافظه ریخته شود و یا آز حافظه خوانده شود را مشخص می‌کند.
- data_Out داده خروجی است و داده ای که از حافظه خوانده می‌شود را نشان می‌دهد.
- w و r سیگنالهای کنترلی برای عملیات نوشتن و خواندن هستند. این سیگنال‌ها حساس به لبه بالارونده و همگام با پالس ساعت هستند. یعنی زمانی که مقدار سیگنال w برابر با ۱ باشد، داده ای که در data_in قرار دارد در آدرسی که در درگاه addr قرار دارد ریخته می‌شود. همینطور زمانی که سیگنال r در برابر با ۱ باشد ، عملیات خواندن از آدرس مربوطه انجام می‌شود. باید توجه داشت که در یک زمان تنها یک عملیات خواندن و یا نوشتن می‌تواند انجام شود.
- clk ورودی پالس ساعت است.
- سیگنال کنترلی rst غیرهمگام با پالس ساعت و Low-active باید باشد. rst تک بیتی و از نوع std_logic و است و زمانی که مقدار سیگنال rst= ‘0’ باشد، مقدار داده هر خانه برابر با آدرس آن می‌شود. به طور مثال مقدار

داده خانه ۰ برابر با "۰" می‌شود و مقدار خانه آدرس ۱ برابر با "۱" و به همین ترتیب تمام خانه‌های حافظه مقداردهی می‌شوند.

۲) طراحی حافظه ROM:

حافظه ROM با ۱۶ خط حافظه و کلمه‌های ۸ بیتی را مانند شکل ۲ طراحی کنید.

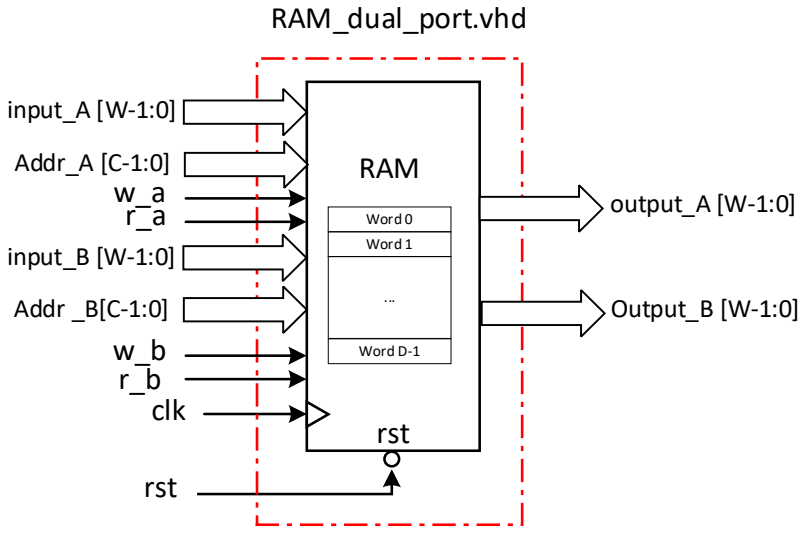


شکل ۲) واحد حافظه ROM و تعریف پورت‌های ورودی و خروجی

- Addr ورودی حافظه و آدرس داده‌ای که باید از حافظه خوانده شود را مشخص می‌کند.
- data_Out داده خروجی است و داده‌ای که از حافظه خوانده می‌شود را نشان می‌دهد.
- clk ورودی پالس ساعت است.

طراحی حافظه RAM دو درگاهه (dual-port)

این حافظه مشابه حافظه RAM طراحی شده در بخش ۱ است، با این تفاوت که دو درگاه کاملاً مستقل برای خواندن/نوشتن وجود دارد.

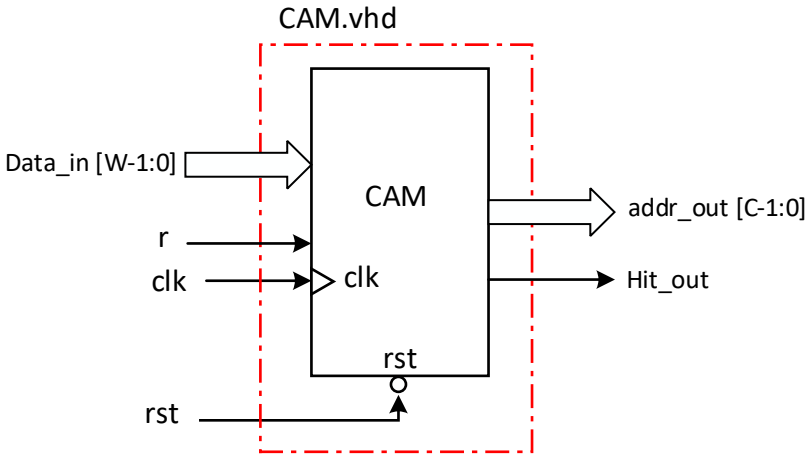


شکل ۲) واحد حافظه dual-port RAM و تعریف پورت‌های ورودی و خروجی

۳) طراحی حافظه CAM

در حافظه CAM با دادن محتوای داده، آدرس داده مشابه با داده ورودی را در خروجی نشان می‌دهد. این حافظه بدون درگاه آدرس بوده، و خواندن یا نوشتن در آن بر اساس محتوا انجام خواهد شد. در هنگام نوشتن داده یعنی w=’1’، اگر داده داخل حافظه نباشد، آن داده در اولین مکان خالی نوشته می‌شود. اما اگر داده در حافظه باشد آدرس مربوطه در addr_out قرار می‌گیرد و مقدار سیگنال خروجی hit_out برابر با ’1‘ می‌شود. در هنگام خواندن داده یعنی r=’1’، در صورتی که داده وارد شده در حافظه وجود داشته باشد، خروجی hit_out برابر با ’1‘ می‌گردد به معنای آنکه داده در حافظه یافت شده است و در غیر اینصورت صفر خواهد بود.

با مشخصات داده شده در بخش ۱) حافظه آدرس‌پذیر محتوا (Content Addressable Memory) طراحی و پیاده‌سازی کنید. پورت‌های ورودی و خروجی مانند شکل ۴ تعریف گردد.



شکل ۴) واحد حافظه CAM و تعریف پورت‌های ورودی و خروجی

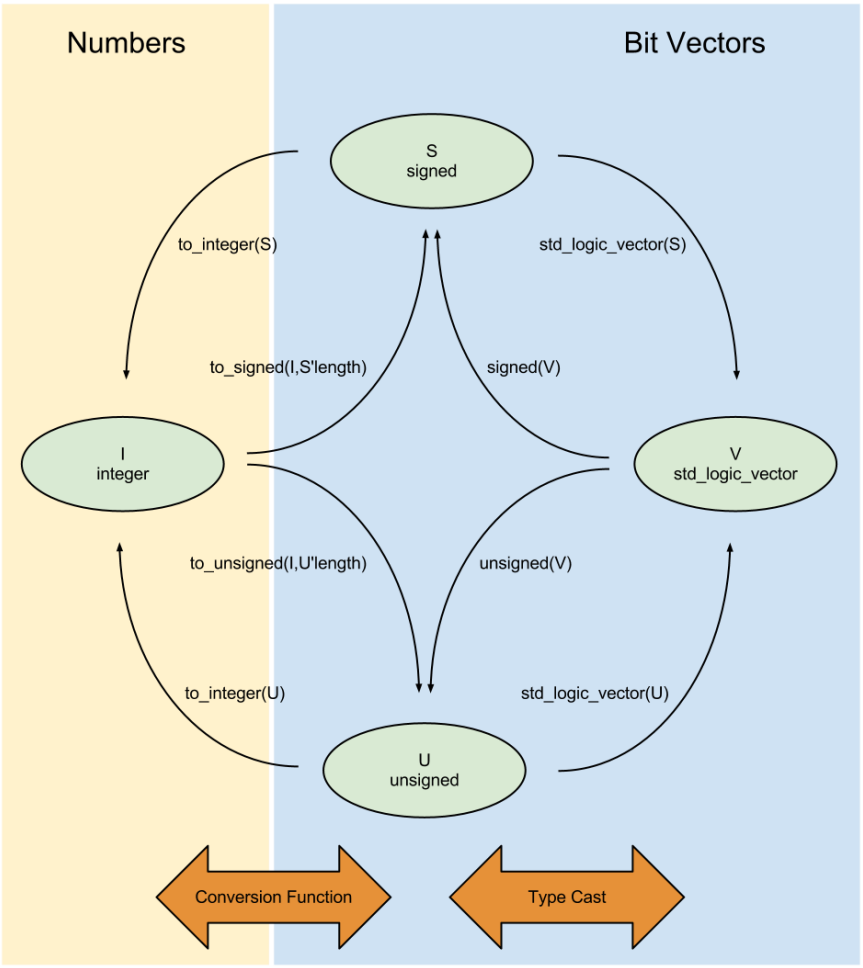
Type conversion

1) The **std_logic_arith** package in the **ieee** library:

- **CONV_INTEGER**--Converts a parameter of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to an INTEGER value. The size of operands in CONV_INTEGER functions are limited to the range -2147483647 to 2147483647, that is, to a 31-bit UNSIGNED value or a 32-bit SIGNED value.
- **CONV_UNSIGNED**--Converts a parameter of type INTEGER,UNSIGNED, SIGNED, or STD_ULOGIC to an UNSIGNED value with SIZE bits.
- **CONV_SIGNED**--Converts a parameter of type INTEGER, UNSIGNED, SIGNED, or STD_ULOGIC to a SIGNED value with SIZE bits.
- **CONV_STD_LOGIC_VECTOR**--Converts a parameter of type INTEGER, UNSIGNED, SIGNED, or STD_LOGIC to a STD_LOGIC_VECTOR value with SIZE bits

```
3   --signal definitions
4   signal index : integer: = 8;
5   signal size : integer: = 4;
6   signal int : integer;
7   signal V : std_logic_vector
8
9
10  --FROM integer TO std_logic_vector
11  v <= conv_std_logic_vector(index, size);
12
13  --FROM std_logic_vector TO  integer
14  int <= conv_integer(addr_in);
```

1. The **library:numeric_std** package in the **ieee** library



Type conversion in vhdl

توابعی که برای تبدیل مقادیر علامت دار/ بدون علامت به std_logic_vector و یا برعکس استفاده می‌شوند.

```
3   -- signal definitions
4   signal slv : std_logic_vector(7 downto 0);
5   signal s : signed(7 downto 0);
6   signal us : unsigned(7 downto 0);
7
8   -- FROM std_logic_vector TO signed/unsigned
9   sgn <= signed(slv);
10  usgn <= unsigned(slv);
11
12  -- FROM signed/unsigned TO std_logic_vector
13  svl <= std_logic_vector(sgn);
14  svl <= std_logic_vector(usgn);
```

توابعی که برای تبدیل مقادیر علامت دار/ بدون علامت به Integer و یا برعکس استفاده می‌شوند.

```
--signal definitions
signal i : integer;
signal sgn : signed(7 downto 0);
signal usgn : unsigned(7 downto 0);

--FROM integer TO signed/unsigned
sgn <= to_signed(i,8);
usgn <= to_unsigned(i,8);

--FROM signed/unsigned TO integer
i <= to_integer(sgn);
i <= to_integer(usgn);
```

توصیف کد VHDL حافظه ROM

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_unsigned.ALL;
USE IEEE.std_logic_arith.all;
--USE IEEE.numeric_std.ALL;
entity ROM2 is
generic (
W1 : integer := 8;  -- number of word bit
D: integer := 4;   -- address bit
C: integer := 16   -- number of word
);
port (
        clk : in std_logic;
        addr   : in std_logic_vector(D-1 downto 0);
        data_out  : out std_logic_vector(W1-1 downto 0));
end ROM2;
architecture ROM_arch of ROM2 is
type mem_type is array (C-1 downto 0) of std_logic_vector (W1-1 downto 0);
constant ROM_block : mem_type := ( "00111000",
                "00000001",
                "00000010",
                "00000011",
                "00000100",
                "00000101",
                "00000110",
                "00000111",
                "00001000",
                "00001001",
                "00001010",
                "00001011",
                "00001100",
                "00001101",
                "00001110",
                "00001111");
begin
process (clk)
begin
        if (rising_edge(clk)) then
                data_out <= ROM_block(conv_integer(addr));
end if;
end process;
end architecture ROM_arch;
```