

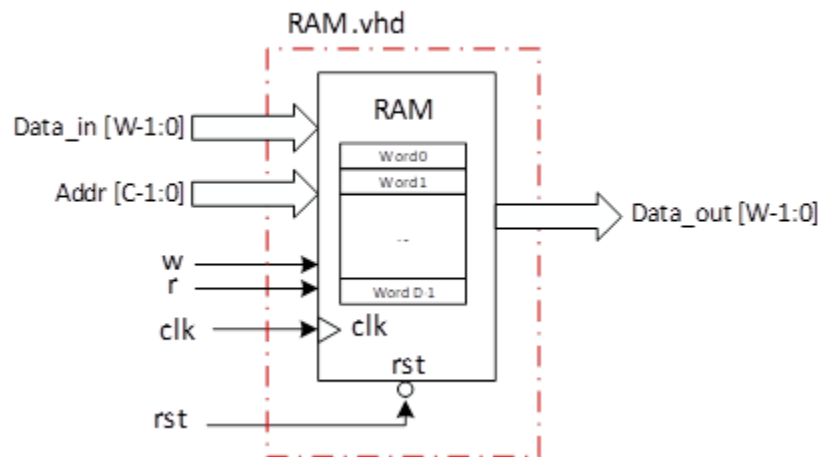
توضیحات آزمایش هفتم

هدف از 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 پورته نشان داده شده است.



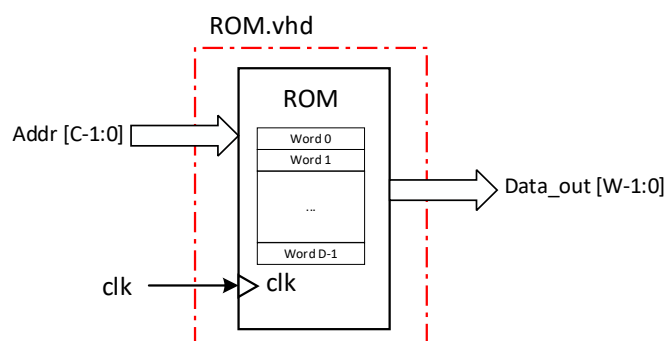
شکل (۱) واحد حافظه RAM و تعریف پورت‌های ورودی و خروجی

- $data_in$ داده ورودی به حافظه است که باید در آدرس مورد نظر ریخته شود.
- $Addr$ ورودی حافظه و آدرس داده‌ای که باید در حافظه ریخته شود و یا از حافظه خوانده شود را مشخص می‌کند.
- $data_Out$ داده خروجی است و داده‌ای 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، داده‌ای که در $data_in$ قرار دارد در آدرسی که در درگاه $addr$ قرار دارد ریخته می‌شود. همین‌طور زمانی که سیگنال r 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.

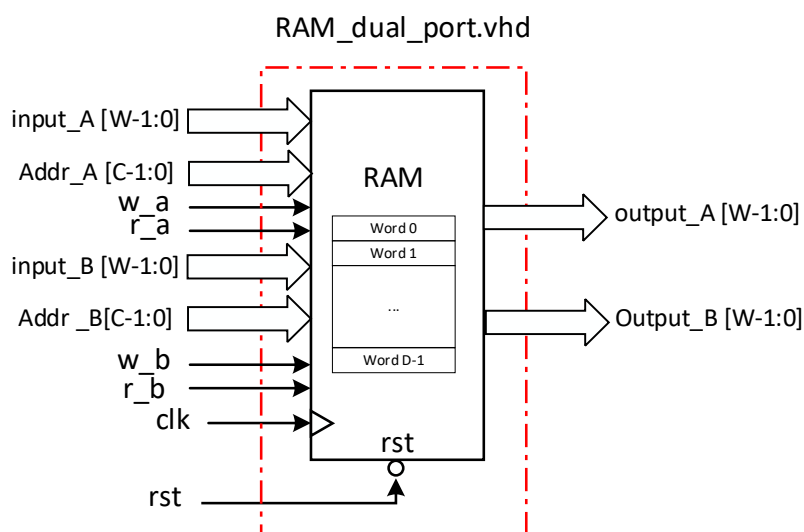


شکل ۲) واحد حافظه ROM و تعریف پورت‌های ورودی و خروجی

- 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 درگاه (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.

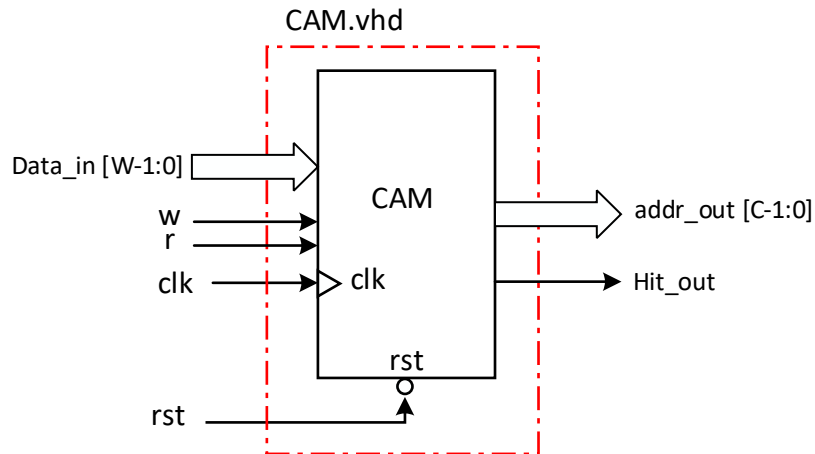


شکل ۲) واحد حافظه dual-port RAM و تعریف پورت‌های ورودی و خروجی

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

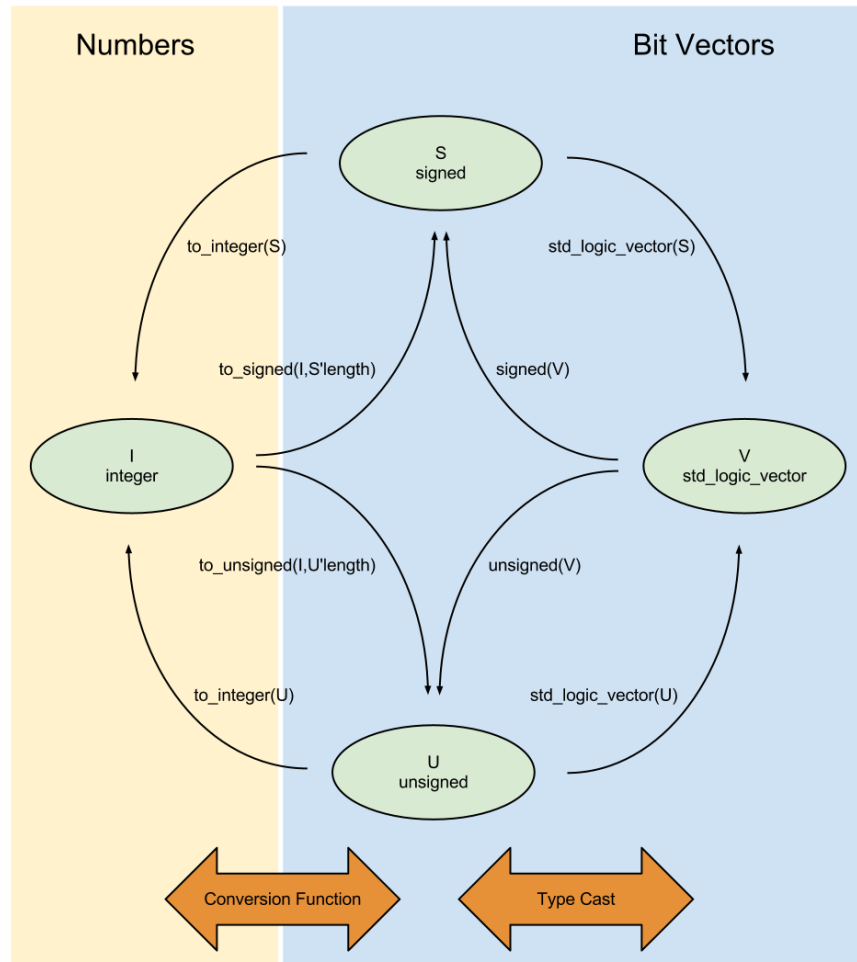
Type conversion

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

- **CONV_INTEGER**--Converts a parameter of type INTEGER, UNSIGNED, SIGNED, or STD_ULONGIC 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_ULONGIC to an UNSIGNED value with SIZE bits.
- **CONV_SIGNED**--Converts a parameter of type INTEGER, UNSIGNED, SIGNED, or STD_ULONGIC 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.

```
3  --signal definitions
4  signal i : integer;
5  signal sgn : signed(7 downto 0);
6  signal usgn : unsigned(7 downto 0);
7
8  --FROM integer TO signed/unsigned
9  sgn <= to_signed(i,8);
10 usgn <= to_unsigned(i,8);
11
12 --FROM signed/unsigned TO integer
13 i <= to_integer(sgn);
14 i <= to_integer(usgn);
```

توصیف کد VHDL حافظه ROM

```
1  LIBRARY IEEE;
2  USE IEEE.std_logic_1164.ALL;
3  USE IEEE.std_logic_unsigned.ALL;
4  USE IEEE.std_logic_arith.all;
5  --USE IEEE.numeric_std.ALL;
6  entity ROM2 is
7  generic (
8    W1 : integer := 8; -- number of word bit
9    D : integer := 4; -- address bit
10   C : integer := 16 -- number of word
11  );
12  port (
13    clk : in std_logic;
14    addr : in std_logic_vector(D-1 downto 0);
15    data_out : out std_logic_vector(W1-1 downto 0));
16  end ROM2;
17  architecture ROM_arch of ROM2 is
18  type mem_type is array (C-1 downto 0) of std_logic_vector (W1-1 downto 0);
19  constant ROM_block : mem_type := ( "00111000",
20    "00000001",
21    "00000010",
22    "00000011",
23    "00000100",
24    "00000101",
25    "00000110",
26    "00000111",
27    "00001000",
28    "00001001",
29    "00001010",
30    "00001011",
31    "00001100",
32    "00001101",
33    "00001110",
34    "00001111");
35  begin
36  process (clk)
37  begin
38    if (rising_edge(clk)) then
39      data_out <= ROM_block(conv_integer(addr));
40    end if;
41  end process;
42  end architecture ROM_arch;
43
```

SIGNAL IN VHDL VERSUS VARIABLE IN VHDL

SIGNAL IN VHDL

A primary object describing a hardware system and are equivalent to "wires"

An object with a past history of values

```
variable variable_name :  
    type; and variable  
variable_name : type :=  
    initial_value;
```

VARIABLE IN VHDL

A variable is an object which store information local to processes and subprograms (procedures and functions) in which they are defined

An object with a single current value

```
signal signal_name :  
    type; AND signal  
signal_name : type :=  
    initial_value;
```

Visit www.PEDIAA.com