دوره مقدماتی آشنایی با FPGA و VHDL

مقدمه

محمدرضا عزیزی

امیرعلی ابراهیمی

بهار ۱۳۹۷

بخش‌های اصلی کد VHDL

- یک کد VHDL از حداقل سه بخش اصلی زیر تشکیل شده است:
  - فراخوانی LIBRARY ها: لیست تمام کتابخانه های استفاده شده.

    برای مثال: *ieee*، *std*، *work* و ...
  - ENTITY: پورت های ورودی/خروجی مدار
  - ARCHITECTURE: بخشی از کد VHDL که نحوه رفتار مدار را توصیف می‌کند.

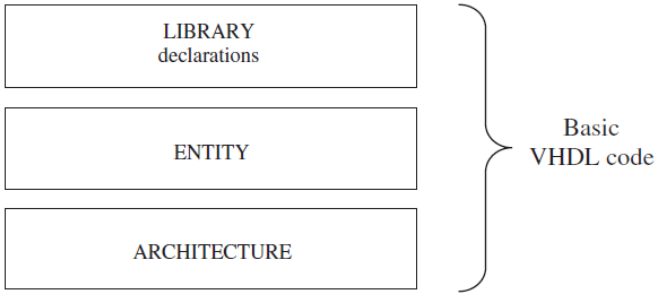

Figure 2.1
Fundamental sections of a basic VHDL code.

کتابخانه (LIBRARY)

- مجموعه ای از کدهایی که زیاد استفاده می‌شود.
- استفاده از کتابخانه‌ها ← افزایش resuability

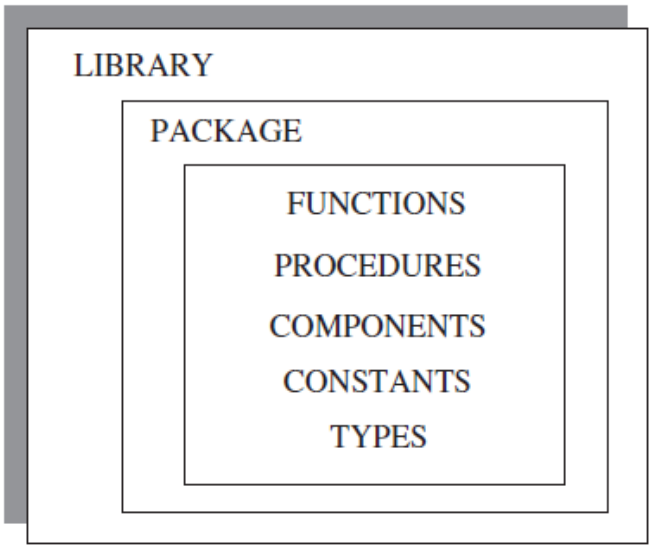

Figure 2.2
Fundamental parts of a LIBRARY.

فراخوانی کتابخانه (LIBRARY DECLARATION)

- آشکارکردن یک کتابخوانه برای یک طراحی
- ساختار:

```
LIBRARY library_name;
USE library_name.package_name.package_parts;
```

- حداقل وجود سه پکیج برای یک طراحی الزامی است.
  - *ieee.std_logic_1164* از کتابخانه *ieee*
  - *standard* از کتابخانه *std*
  - *work* از کتابخانه *work*

فراخوانی کتابخانه (LIBRARY DECLARATION)

- مثال: نحوه فراخوانی این سه کتابخانه:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY std;
USE std.standard.all;

LIBRARY work;
USE work.all;
```

- نکته: از ; برای پایان یک عبارت و از -- برای کامنت گذاری استفاده می‌شود.

فراخوانی کتابخانه (LIBRARY DECLARATION)

- نکته: کتابخانه‌های std و work به صورت پیش‌فرض، برای هر طرح آشکار هستند و نیازی به فراخوانی نیست.
- نکته: کتابخانه ieee فقط زمانی استفاده می‌شود که از نوع داده‌های STD_LOGIC یا STD_ULOGIC استفاده شود.
- اهداف هر یک از این سه کتابخانه:
  - پکیج ***std_logic_1164*** از کتابخانه ***ieee***: معرفی نوع داده هایی با چند سطح منطقی
  - ***std***: کتابخانه مرجع – انواع داده، ورودی و خروجی متنی و ...
  - ***work***: محل ذخیره شدن طراحی (فایل vhd. و فایل‌های ایجاد شده توسط کامپایلر، شبیه‌ساز و ...)

پکیج‌های کتابخانه IEEE

- *std_logic_1164*:
  - نوع داده‌های STD_LOGIC(۸ مقداره) و STD_ULOGIC (۹مقداره)
- *std_logic_arith*:
  - نوع داده‌های SIGNED و UNSIGNED
  - عملیات مقایسه و ریاضی برای این نوع داده‌ها
  - function های تبدیل نوع داده:
    - *conv_integer(p)*
    - *conv_unsigned(p, b)*
    - *conv_signed(p, b)*
    - *conv_std_logic_vector(p, b)*

پکیج‌های کتابخانه IEEE

- *std_logic_signed*:
  - کتابخانه‌هایی که عملیاتی بر روی نوع داده std_logic_vector تعریف می‌کند که گویی نوع‌داده signed هستند.
- *std_logic_unsigned*:
  - کتابخانه‌هایی که عملیاتی بر روی نوع داده std_logic_vector تعریف می‌کند که گویی نوع‌داده unsigned هستند.

← توضیح بیشتر در فصل ۳

(۱) ENTITY

- لیست پورت‌های ورودی و خروجی مدار
- ساختار:

```
ENTITY entity_name IS
   PORT (
      port_name : signal_mode signal_type;
      port_name : signal_mode signal_type;
      ...);
END entity_name;
```

- signal_mode:
  - زمانی از BUFFER استفاده می‌شود که یک خروجی در داخل کد بخواهی خوانده شود.

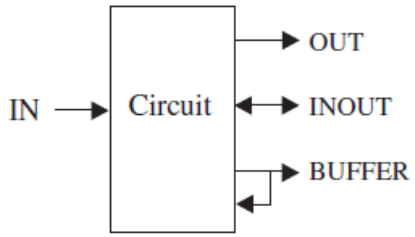

Figure 2.3
Signal *modes*.

(۲) ENTITY

➢ BUFFER:

```
if some_out = x then                          -- some_out is output
          -- do something --
end if ;
some_out <= intermediate_some_out;            -- some_out is output
if intermediate_some_out = x then
          -- do something --
end if ;
--------------------------
if some_out = x then                          -- some_out is buffer
          -- do something --
end if ;
```

(۳) ENTITY

- signal_type:
  - BIT، STD_LOGIC، INTEGER و ...
- نام سیگنال(signal_name):
  - هر نامی به جز نام‌های رزرو شده زبان VHDL

From VHDL 87:			
	ENTITY	OPEN	WAIT
	EXIT	OR	WHEN
ABS	FILE	OTHERS	WHILE
ACCESS	FOR	OUT	WITH
AFTER	FUNCTION	PACKAGE	XOR
ALIAS	GENERATE	PORT	
ALL	GENERIC	PROCEDURE	*From VHDL 93:*
AND	GUARDED	PROCESS	
ARCHITECTURE	IF	RANGE	GROUP
ARRAY	IN	RECORD	IMPURE
ASSERT	INOUT	REGISTER	INERTIAL
ATTRIBUTE	IS	REM	LITERAL
BEGIN	LABEL	REPORT	POSTPONED
BLOCK	LIBRARY	RETURN	PURE
BODY	LINKAGE	SELECT	REJECT
BUFFER	LOOP	SEVERITY	ROL
BUS	MAP	SIGNAL	ROR
CASE	MOD	SUBTYPE	SHARED
COMPONENT	NAND	THEN	SLA
CONFIGURATION	NEW	TO	SLL
CONSTANT	NEXT	TRANSPORT	SRA
DISCONNECT	NOR	TYPE	SRL
DOWNTO	NOT	UNITS	UNAFFECTED
ELSE	NULL	UNTIL	XNOR
ELSIF	OF	USE	
END	ON	VARIABLE	

(۴) ENTITY

- مثال: گیت nand:

Figure 2.4
NAND gate.

```
ENTITY nand_gate IS
   PORT (a, b : IN BIT;
         x : OUT BIT);
END nand_gate;
```

(۱) ARCHITECTURE

- توضیح این که یک مدار چگونه باید رفتار کند.
- ساختار:

```
ARCHITECTURE architecture_name OF entity_name IS
    [declarations]
BEGIN
    (code)
END architecture_name;
```

- هر architecture شامل دو بخش است:
  - **بخش declerative**: فراخوانی signal ها و constant ها. (اختیاری)
  - **بخش کد**: از BEGIN به پایین.
- نحوه نام‌گذاری مشابه با نحوه نام‌گذاری entity است. می‌تواند همنام با entity باشد.

(۲) ARCHITECTURE

- مثال: گیت nand:

Figure 2.4
NAND gate.

```
ARCHITECTURE myarch OF nand_gate IS
BEGIN
   x <= a NAND b;
END myarch;
```

- عملگر انتساب: **<=**

پایان بخش ساختار کد