بسمه تعالی

طراحی کامپیوتری سیستمهای دیجیتال

Computer-Aided Digital System Design

مدرس
فاطمه دارائی

f_daraei@semnan.ac.ir
https://fdaraei.profile.semnan.ac.ir

دانشگاه سمنان
Semnan University
پردیس فرزانگان

چشم‌انداز

- فلسفه زبان توصیف سخت‌افزار VHDL
- بلوک پایه یک طرح یا مجموعه entity/architecture
- ضوابط و نکات عمومی در VHDL
- روش‌های توصیف یا مدل‌سازی مدارهای دیجیتال

اسلایدها برگرفته از جزوه دکتر محمد علی شفیعیان
https://shafieian-education.ir/computer-aided-digital-system-design-3/

VHDL چیست؟

زبان توصیف سخت‌افزار VHDL برای توصیف و شبیه‌سازی مدارهای دیجیتال از ساده‌ترین نوع یعنی گیت‌ها تا سیستم‌های پیچیده‌تری مانند پروسسودرها و ... یا طراحی مدارهای ASIC به‌کار برده می‌شوند.

VHDL = VHSIC Hardware Description Language

VHSIC = Very High Speed Integrated Circuits

پروژه VHSIC توسط وزارت دفاع امریکا (DOD) با هدف توسعه نسل جدیدی از مدارهای مجتمع با سرعت بالا حمایت می‌شد.

VHDL چیست؟

در گذشته زبان‌های توصیف سخت‌افزار مانند ABEL، PALASM و ... توسط شرکت‌های مختلف برای برنامه‌ریزی مدارهای PLA، PAL و PLD به بازار عرضه شده بود.

۱۹۸۰

وزارت دفاع آمریکا طراحی یک زبان جدید استاندارد برای توصیف مدارهای سخت‌افزاری و همچنین انتقال داده سیستم دیجیتال از شرکتی به شرکت و یا به کشور دیگر را به سه شرکت IBM، Texas Instruments و Intermetrics سفارش داد.

VHDL چیست؟

۱۹۸۵ اولین نسخه زبان توصیف سخت‌افزار VHDL عرضه شود.

در مراحل بعدی توسعه، استانداردسازی زبان با همکاری نمایندگانی از دولت و دانشگاه به IEEE واگذار شد.

۱۹۸۶ مؤسسه IEEE آن را تصویب و استاندارد نمود و آن را VHDL86 نامید.

۱۹۸۷ این زبان تأیید شد و به‌صورت استاندارد IEEE 1076_1987 ملاک قرار گرفت.

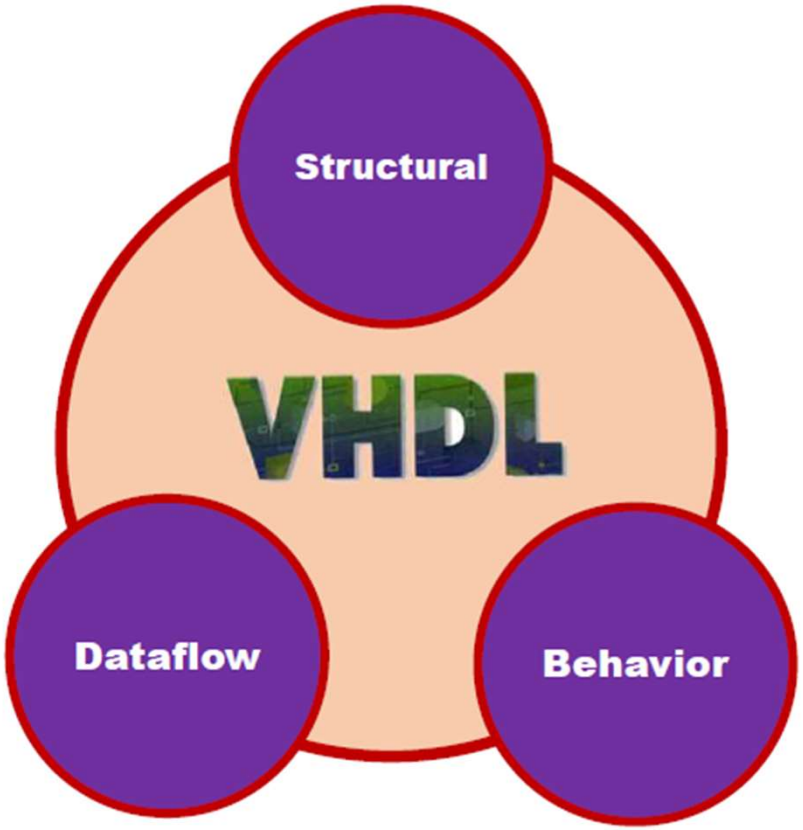
VHDL چیست؟

۱۹۹۳ طی نظرخواهی مجدد ویژگی‌های جدیدی به آن اضافه شد و به‌صورت نسخه 1076_1993 ارائه گردید.

۱۹۹۴ توسط IEEE به تصویب نهایی رسید و به‌صورت استاندارد IEEE به دنیا معرفی شد.

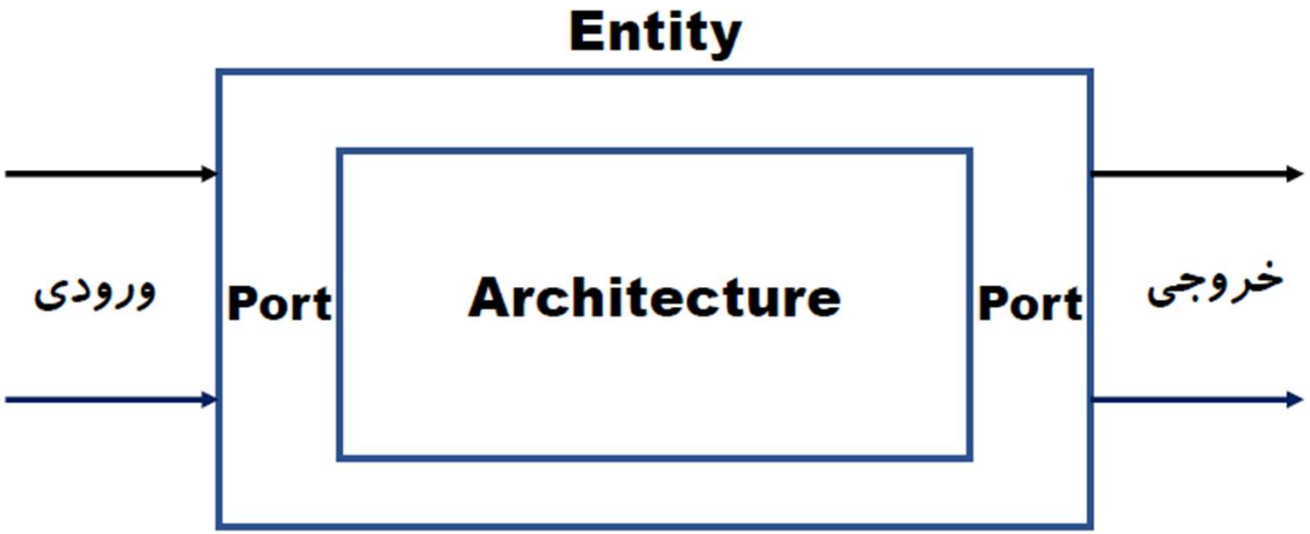
هر پنج سال یک بار تغییرات جزئی برای بالا بردن کارآیی آن انجام می‌شود.

VHDL چیست؟



ابزارهای برنامه‌ریزی FPGA

بلوک پایه یک طرح دیجیتال



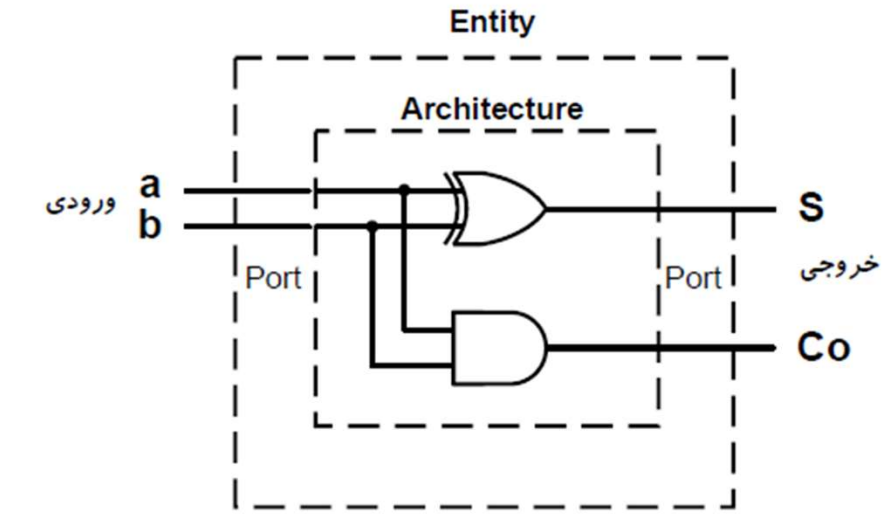
- یک توصیف VHDL شامل :
  - Entity declaration
  - Architecture body
- تعریف entity در حقیقت معرفی سیگنالهای ورودی و خروجی است.
- architecture رابطه بین سیگنالهای ورودی و خروجی است. (عملکرد/ساختار)

ساختار کلي يك فايل VHDL

```
ENTITY NAME_OF_ENTITY is
  input and output ports.
END [entity] [NAME_OF_ENTITY] ;
```

```
ARCHITECTURE architecture_name OF NAME_OF_ENTITY IS
  declarations.
BEGIN
  -- Statements
       :
END architecture_name;
```

بلوک پایه یک طرح دیجیتال



طرح (design) یک نیم‌جمع کننده

برنامه VHDL برای توصیف مدار نیم‌جمع کننده

```
library ieee;
use ieee.std_logic_1164.all;

entity ha is
port (a,b :in bit;
      s,co :out bit);
end ha;

architecture dataflow of ha is
begin
      s <= a xor b;
      co <= a and b;
end dataflow;
```

توصیف Port ورودی
خروجی یک
نیم‌جمع کننده

معماری (Architect)
مدار یک نیم‌جمع کننده

برنامه VHDL برای توصیف مدار نیم‌جمع کننده


```
library ieee;
use ieee.std_logic_1164.all;

architecture concurrent of example is
begin
        A <= 3;
        Sum <= A+B;
        B <= 7;
end;
```

همروندی

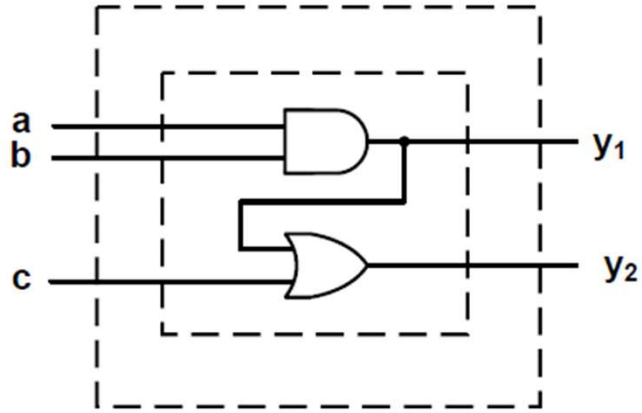
concurrency

برنامه VHDL برای توصیف مدار نیم‌جمع کننده

```
library ieee;
use ieee.std_logic_1164.all;

entity نام مدار is
        port (معرفی ورودی خروجی ها و نوع آنها);
end [entity نام مدار];

architecture نام آرشیتکت of نام مدار is
begin
        VHDL عبارات
End [architecture نام آرشیتکت];
```

بافر (Buffer)

```
library ieee;
use ieee.std_logic_1164.all;

entity cir is
port (a,b,c :in bit;
        y1,y2 :buffer bit);
end cir;

architecture dataflow of cir is
begin
        y1 <= a and b;
        y2 <= y1 or c;
end dataflow;
```



ورودی – خروجی

- در برخی از طرح‌ها، سیگنال‌های port باید هم ورودی و هم خروجی باشند.
- در این حالت، سیگنال‌های port دوطرفه است که به شکل inout باید تعریف شوند.
- وقتی port به‌صورت inout تعریف می‌شود، یعنی هم اطلاعات خروجی است و هم می‌توان به‌عنوان ورودی، اطلاعات را از آن خواند.

خلاصه

در entity و در port:

- توصیف ورودی ← in
- توصیف خروجی ← out یا buffer
- توصیف ورودی – خروجی ← inout

سیگنال (Signal)

- برای اتصال قسمت‌های مختلف مدار دیجیتال از سیگنال (signal) استفاده می‌شود.
- از نظر سخت‌افزاری، سیگنال مانند سیم اتصال است که قسمت‌های مختلف یک طرح دیجیتال را به هم متصل و داده‌ها را منتقل می‌کند.
- در واقع port نیز سیگنال (signal) است.
- سیگنال از دو قسمت تشکیل می‌شود:
  - قسمت اول: نام سیگنال
  - قسمت دوم: نوع (Type) سیگنال که با دونقطه (:) مجزا می‌شود.

```
signal a:bit;
```

سیگنال (Signal)

چندین سیگنال یا ورودی خروجی port با یک نوع (Type) را می‌توان در یک سطر نوشت و آنها را با کاما از هم جدا نمود

```
signal a,b,c:bit
```

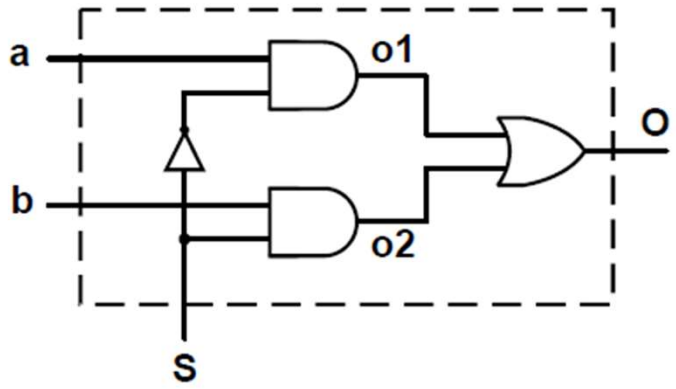
```
port(a,b:in bit
     s,co:out bit)
```

اعلانات (Declaration)

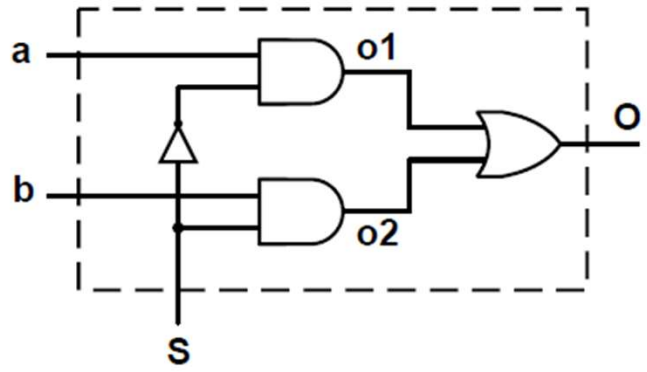
```
architecture archi of my-design is
      signal a:bit;
begin
        VHDL Statements;
end;

architecture archi2 of my-design is
      signal Bus1,Bus2:bit_vector(7 downto 0);
begin
        VHDL Statements;
end;
```

مثال

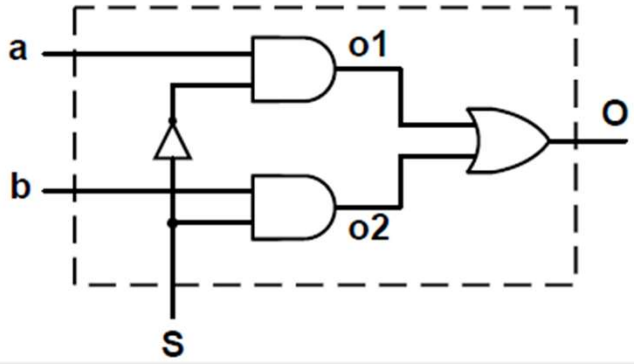
مداری را در نظر بگیرید که در آن اگر ورودی S برابر با صفر باشد، خروجی O برابر با a و اگر ورودی S برابر با 1 باشد، خروجی O مساوی b شود.

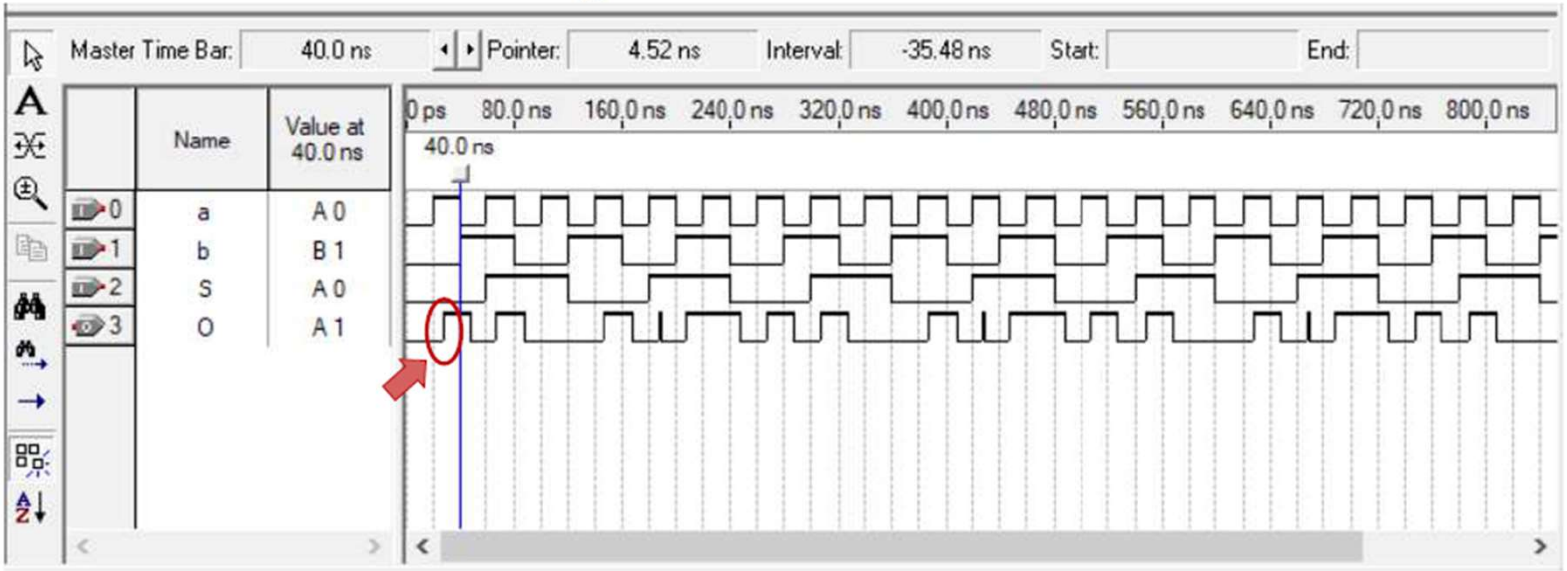


```
library ieee;
use ieee.std_logic_1164.all;
entity multi is
port (a,b : in bit;
      S : in bit;
      O : out bit);
end;

architecture mux of multi is
      signal o1,o2 : bit;
      begin
            o1 <= not(S) and a;
            o2 <= S and b;
            O <= o1 or o2;
      end mux;
```




```
O <= (not(S) and a) or (S and b);
```


a
b
S
o1
o2
O


Master Time Bar: 40.0 ns
Pointer: 4.52 ns
Interval: -35.48 ns
Start:
End:
Name
Value at 40.0 ns
0 a A 0
1 b B 1
2 S A 0
3 O A 1
0 ps 80.0 ns 160.0 ns 240.0 ns 320.0 ns 400.0 ns 480.0 ns 560.0 ns 640.0 ns 720.0 ns 800.0 ns
40.0 ns

ضوابط و نکات عمومی در VHDL

- **توضیح (Comment)**
  - در ادامه هر عبارت **VHDL** با قرار دادن دو خط تیره (--) می‌توان برای آن عبارت توضیح (**Comment**) نوشت که کار آن عبارت را روشن‌تر نماید.

- **نام‌گذاری یا شناسه (Identifier)**
  - نام‌گذاری در **VHDL** شامل حروف، رقم و خط تیره پایین (**Underline**) است.
  - نام باید با حروف شروع شود و بهتر است کمتر از ۱۵ حرف باشد.
  - برای نام‌گذاری نباید از کلمات کلیدی یا رزروشده در **VHDL** استفاده شود.

DATA_Bus
Delay1
RISE_Time
x10
X_10
Z_Bus
B2

ضوابط و نکات عمومی در VHDL

- **حروف کوچک و بزرگ در VHDL**
  - زبان VHDL به حروف بزرگ و کوچک حساس نیست یعنی هر دو را به یک معنی می‌پذیرید.
  - معمولاً کلمات کلیدی یا رزروشده در VHDL را با حروف کوچک و کلماتی که کاربر تعریف می‌کند با حروف بزرگ نمایش داده می‌شود.

- **نقطه ویرگول (;)**
  - هر عبارت VHDL با نقطه ویرگول پایان می‌یابد.

ضوابط و نکات عمومی در VHDL

- کاما (,)
  - لیست سیگنال‌ها معمولاً با کاما از هم جدا می‌شوند

```
Port (a,b,c:in bit;
      Sum,Cout:out bit;)
```

```
signal im1,im2,im3 : bit;
```

- نمایش 0 و 1 منطقی در VHDL
  - صفر و یک منطقی در VHDL در علامت کوتیشن (‘) به صورت ’0‘ و ’1‘ نمایش داده می‌شود.
  - تعدادی صفر و یک در دو کوتیشن (“) نشان داده می‌شود مانند: ”1011“

ضوابط و نکات عمومی در VHDL

- **نمایش اعداد در VHDL**

- در VHDL اعداد صحیح (Integer) به‌صورت پیش فرض اعداد دهدهی معمولی هستند.
- اعداد را می‌توان به‌صورت نمایی نیز نمایش داد مانند 256E3 که برابر 256×10^3 یا 256000 است.
- برای نمایش اعداد باینری از علامت b یا B جلوی بیت‌های باینری استفاده می‌شود.

نمایش عدد ۱۸ در مبنای ۲ ⟵ b"10010" یا B"10010"

- برای نمایش اعداد در پایه ۱۶ یا هگزادسیمال از علامت X یا x استفاده می‌شود

نمایش عدد هگزادسیکال C9 ⟵ X"C9"

نمایش عدد هگزادسیکال 12 ⟵ x"12"

ضوابط و نکات عمومی در VHDL

- نمایش اعداد در VHDL
  - برای نمایش اعداد در پایه ۸ یا اُکتال از علامت O یا o استفاده می‌شود

نمایش عدد اُکتال 311 ⟵ O"311"

نمایش عدد اُکتال 113 ⟵ O"113"

نمایش عدد اُکتال 35 ⟵ O"35"

مقدار اولیه دادن به سیگنال

- هنگام شبیه‌سازی یک مدار، تمام سیگنال‌ها یک مقدار اولیه پیش‌فرض می‌گیرند
- درصورتی‌که در قسمت اعلانات معماری، مقداری به سیگنال‌ها تخصیص داده شود، آن مقدار را در ابتدای شبیه‌سازی می‌گیرد:

```
signal a:bit:='1';
```

- سیگنال a در ابتدای شبیه‌سازی مقدار اولیه '1' را می‌گیرد.
- اگر به سیگنال a مقدار اولیه داده نشود، در این صورت چون نوع آن بیت است در ابتدای شبیه‌سازی سیگنال a مقدار پیش‌فرض 0 را به عنوان مقدار اولیه می‌گیرد

در معماری طرح نیز می‌توان به سیگنال‌ها مقدار مناسب تخصیص داد:

```
a<='1';
b<="10101011";
```

انواع (Types) نمایش اطلاعات سیگنال‌ها

- نوع Bit Type
- نوع Integer Type
- نوع bit_vector Type
- نوع Time Type
- نوع Boolean Type
- نوع Std_Logic Type و Std_Ulogic Type
- نوع Std_Logic_Vector Type و Std_Ulogic_Vector Type
- عبارات کلی others

Integer Type نوع

- سیگنال می‌تواند عددی Integer که اعدادی مثبت و منفی می‌باشند را دارا باشد.
- سیگنال Integer برای انجام محاسبات جمع (+)، تفریق (-)، ضرب (*)، تقسیم (/)، قدر مطلق (abs)، باقیمانده (rem) و mod به‌کار می‌رود.

```
c<=a+b
```

a و b با هم جمع شوند و حاصل به c داده شود

```
c<=a-b
```

```
c<=a*b
```

```
c<=a/b
```

اگر a و b و c از نوع Integer باشند و مقادیر a و b به‌ترتیب ۵ و ۳ باشند،
خارج قسمت ۱ که Integer است به c داده می‌شود.

```
c<=abs(a)
```

Integer Type نوع

`c<=a mod b` ⇒ a بر b تقسیم می‌شود و با قیمانده با علامت b به c داده می‌شود

`c <= 5 mod 3` ⇒ باقیمانده تقسیم ۵ بر ۳ یعنی ۲ به c داده می‌شود

`c <= (-5) mod 3` ⇒ `(-5) = ((-2)*3 + 1)` ⇒ باقیمانده ۱ به c داده می‌شود

`c <= (-5) mod (-3)` ⇒ `(-5) = (1*(-3) + (-2))` ⇒ باقیمانده ۲- به c داده می‌شود

`c <= 5 mod (-3)` ⇒ `5 = ((-2)*(-3) + (-1))` ⇒ باقیمانده ۱- به c داده می‌شود

Integer Type نوع

```
c<=a rem b
```
⇒ a بر b تقسیم می‌شود و با قیمانده با علامت a به c داده می‌شود

```
c <= 5 rem 3
```
⇒ `5 = (1*3 + 2)` ⇒ باقیمانده ۲به c داده می‌شود

```
c <= (-5) rem 3
```
⇒ `-5 = (-1)*3 + (-2)` ⇒ باقیمانده ۲- به c داده می‌شود

```
c <= (-5) rem (-3)
```
⇒ `-5 = (1*(-3) + (-2))` ⇒ باقیمانده ۲- به c داده می‌شود

```
c <= 5 rem (-3)
```
⇒ `5 = ((-1)*(-3) + (2))` ⇒ باقیمانده ۲+ به c داده می‌شود

Integer Type نوع

- برای اعداد بدون علامت rem و mod به‌طرز یکسان عمل می‌کنند.
- عملیات مقایسه‌ای مانند <، =<، >، =>، = و =/ (عدم تساوی) را برای دو سیگنال از نوع Integer نیز می‌توان انجام داد که نتیجه آن به‌صورت True یا False می‌باشد.
- در برخی از کاربردها، سیگنال‌ها و Variableها از نوع Integer هستند.

مثال

برنامه‌ای با VHDL بنویسید که دو ورودی a و b که به‌صورت Integer هستند را با هم جمع و به خروجی c بدهد.

```
entity adder is
port (a,b : in integer range 0 to 7;
        c : out integer range 0 to 15);
end adder;

architecture behav of adder is
begin
    c <= a + b;
end behav;
```

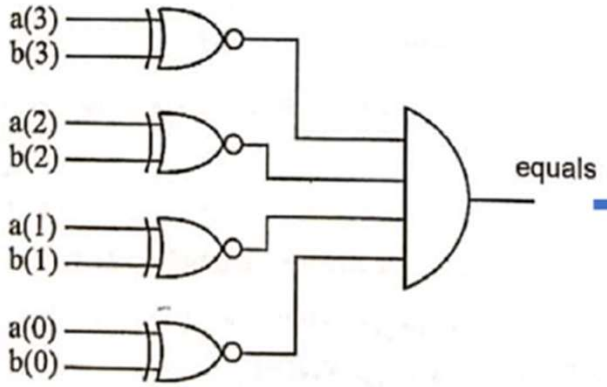
زمانی که سیگنال را Integer معرفی می‌کنیم، پیش‌فرض سیگنال یک عدد ۳۲ بیتی است:

[-2,147,483,648 to +2,147,483,647]

چون در عمل با اعدد کوچکتری کار می‌کنیم بهتر است محدوده Integer را کوچکتر کنیم.

نوع bit_vector Type

- در VHDL مجموعه چندین عنصر که از یک نوع باشند با آرایه توصیف می‌شود.
- مجموعه‌ای از چندین بیت، عنصرهای یک آرایه به نام bit_vector را تشکیل می‌دهند.
- مثلاً اگر سیگنال‌های a و b چهار بیتی باشند، نوع این سیگنال‌ها آرایه‌ای از چهار بیت می‌باشد که با bit_vector(3 downto 0) بیان می‌شود.

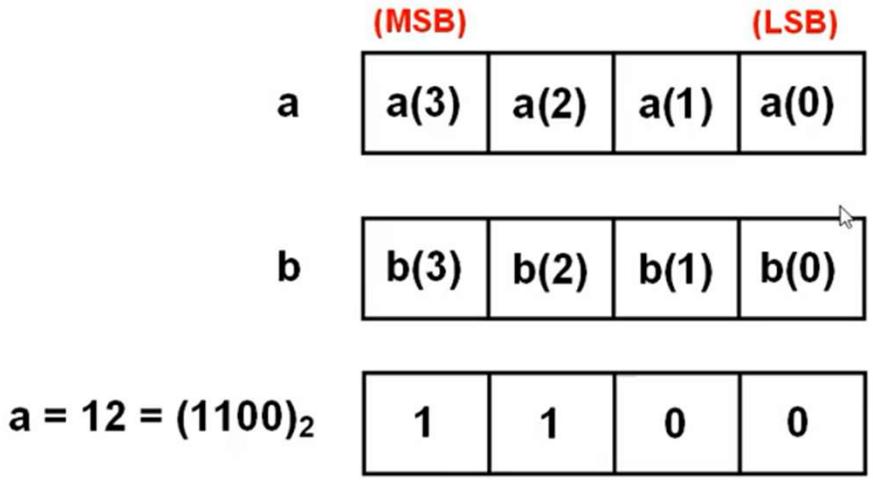



```
a,b:in bit_vector(3 downto 0)
```

نوع bit_vector Type

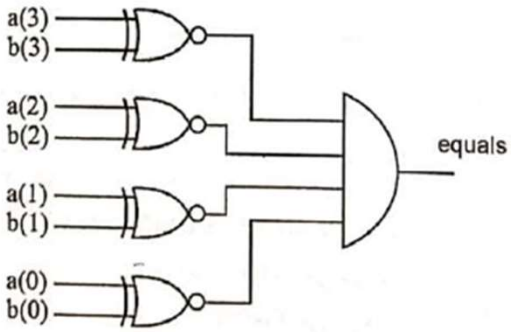
دانشگاه سمنان
Semnan University
پردیس فرزانگان

```
a,b:in bit_vector(3 downto 0)
```



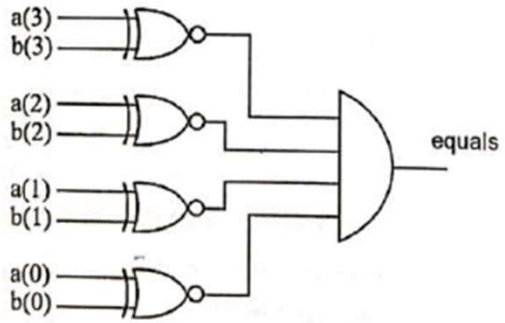
برای نمایش **اعداد باینری** باید از نوع **bit_vector** به‌صورت **downto** استفاده نمود.

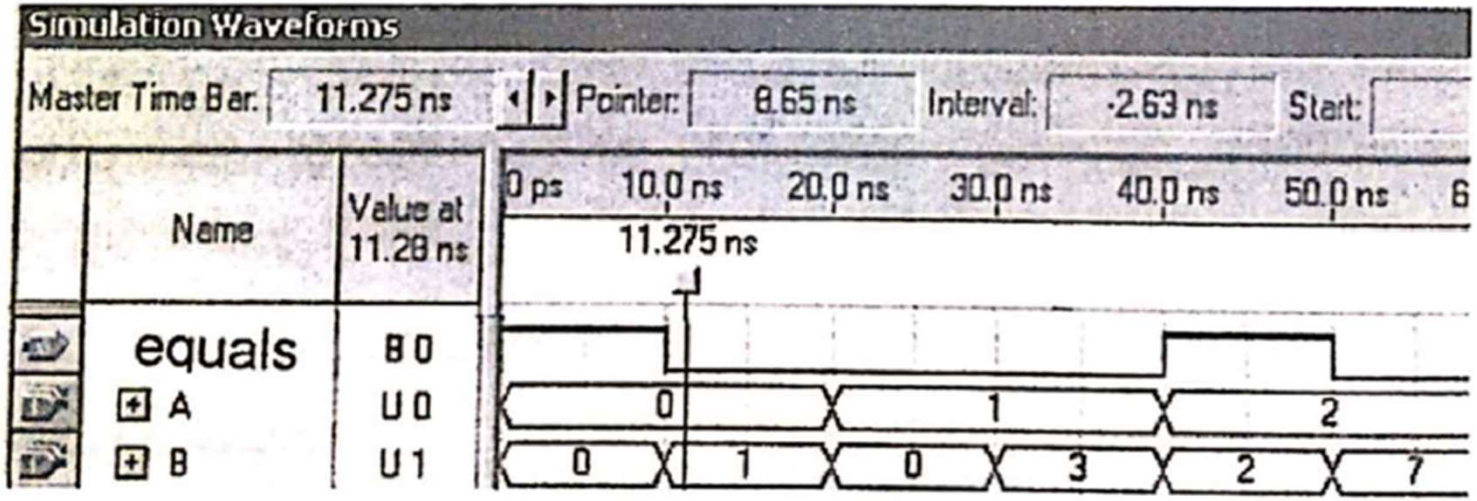
bit_vector Type نوع




```
entity equabo is
port (
        a : in bit_vector(3 downto 0);
        b : in bit_vector(3 downto 0);
        equals : out bit);
end equabo;

architecture equabo_arch of equabo is
begin
     equals <= (a(3) xnor b(3))
               and (a(2) xnor b(2))
               and (a(1) xnor b(1))
               and (a(0) xnor b(0));
end equabo_arch;
```

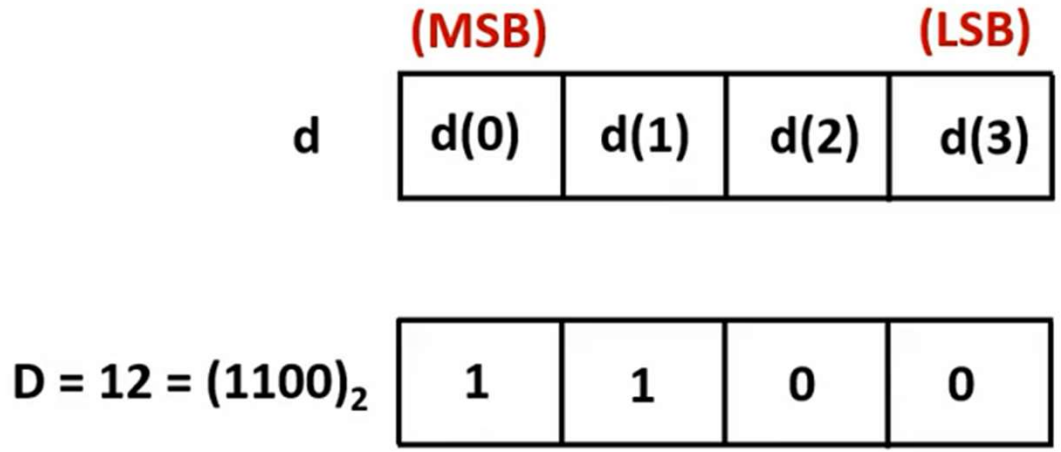
bit_vector Type نوع





bit_vector Type نوع


```
Signal d:bit_vector(0 to 3)
```



نوع bit_vector Type

- زمانی که عملیات بین دو آرایه انجام می‌شود، عملیات بین عناصر از چپ به راست انجام می‌شود:

```
signal v:in bit_vector(1 downto 0)
```


```
signal w:in bit_vector(0 to 1)
```


```
w <= v
```

معادل است با

```
w(1) <= v(0)
w(0) <= v(1)
```

نوع bit_vector Type

- اگر عملیات بر روی سیگنال‌های bit_vector انجام شود، باید سیگنال‌ها دارای تعداد عناصر مساوی باشند:

```
00101001 AND 11100101  ⟶  00100001
```

- می‌توان به قسمتی از عناصر آرایه نیز دسترسی داشت:

```
b(1 downto 0) <= a(3 downto 2)  ⟶  b(1) <= a(3)
                                   b(0) <= a(2)
```

باید نوع عناصر دو طرف یکسان باشد.

bit_vector Type نوع

- مقادیر ’0‘ و ’1‘ مقادیر bit هستند.
- برای دادن مقدار به سیگنال bit_vector باید مقدار 0 و 1 در علامت " قرار گیرند:

```
d : out bit_vector(3 downto 0)

d <= B"1100"   ⟵   d(0) <= '0'   ⟵   d <= X"C"
                   d(1) <= '0'
                   d(2) <= '1'
                   d(3) <= '1'
```

نوع bit_vector Type

```
d : out bit_vector(0 to 3)

d <= B"1100"   ⟶   d(0) <= '1'
                   d(1) <= '1'
                   d(2) <= '0'
                   d(3) <= '0'
```

- برای سیگنال‌های از نوع bit_vector می‌توان عملیات AND، OR، NOT، NAND، NOR، XOR و XNOR را انجام داد.
- عملیات مقایسه‌ای مانند <، =<، >، =>، = و =/ را که نتیجه آن به‌صورت Boolean یعنی True یا False است را می‌توان انجام داد.

نوع Time Type

- نوع Time را نوع Physical نیز می‌نامند.
- این نوع نمایش برای اندازه‌گیری زمان است.
- این Type شامل دو قسمت می‌باشد: یکی عدد و دیگری واحد زمان.

```
q <= r NOR nq after 5 ns;
```

- انواع واحدهای زمان:

```
1 sec = 1000 ms
1 ms = 1000 us
1 us = 1000 ns
1 ns = 1000 ps
1 ps = 1000 fs
```

Time Type نوع

مثال :

```
constant delay: Time:=5 ns;
.
.
.
a <= b after delay
```

Boolean Type نوع

- نوع boolean دارای دو مقدار True و False می‌باشد.
- نوع boolean نتیجه تست منطقی مانند مقایسه دو سیگنال حاصل می‌شود.
- عملیات مقایسه‌ای از می‌توان بین دو اپراند با عملگرهای <، =>، >، =<، = و =/ انجام داد.

Boolean Type نوع

مثال: برنامه‌ای با VHDL بنویسید که دو عدد a و b چهاربیتی را با هم مقایسه کند؛ اگر دو عدد با هم برابر باشند، 1 را به equal بدهد و در غیر این صورت 0 را به equal بدهد.

```
entity equalcom is
port (
      a : in bit_vector(3 downto 0);
      b : in bit_vector(3 downto 0);
      equals : out bit);
end equalcom;

architecture equalcom_arch of equalcom is
begin
    equal <= '1' when (a=b) else '0';
end equalcom_arch;
```

Boolean Type نوع

در عبارات شرطی if نیز، نتیجه تست به‌صورت True و False می‌باشد.

```
if r = '1' then
     q <= '0';
end if;

VHDL statements
```

```
if a>b then
     MAX <= a;
else
     MAX <= b;
end if;

VHDL statements
```

```
if a/=b then
     MIN <= a;
end if;
```

Boolean Type نوع

نوع boolean در نتیجه عملگرهای منطقی AND، OR، NOT، NAND، NOR، XOR و XNOR نیز حاصل می‌شود

```
if (clock'event AND clock='1') then
     q <= d;
end if;

VHDL statements
```

نوع Std_Logic Type و Std_Ulogic Type

- در شبیه‌سازی مدارهای دیجیتال، علاوه بر دو مقدار '0' و '1' که با نوع bit تعریف می‌شوند، مقادیر دیگری نیز می‌باشند.

مقدار	توضیح
'U'	Uninitialized
'X'	Forcing Unknown
'0'	Forcing '0' Strong '0'
'1'	Forcing '1' Strong '1'
'Z'	High Impedance
'W'	Weak Unknown weak '0' or '1'
'L'	Weak '0' (read)
'H'	Weak '1' (read)
'-'	Don't care

→

```
library ieee
ieee.std_logic_1164
```

نوع Std_Logic Type و Std_Ulogic Type

	U	X	0	1	Z	W	L	H	-
U	U	U	U	U	U	U	U	U	U
X	X	X	X	X	X	X	X	X	X
0	U	X	0	1	0	0	0	0	X
1	U	X	X	X	1	1	1	1	X
Z	U	X	0	1	Z	W	L	H	X
W	U	X	0	1	W	W	W	W	X
L	U	X	0	1	L	W	L	W	X
H	U	X	0	1	H	W	W	H	X
-	U	X	X	X	X	X	X	X	X

نوع Std_Logic Type و Std_Ulogic Type

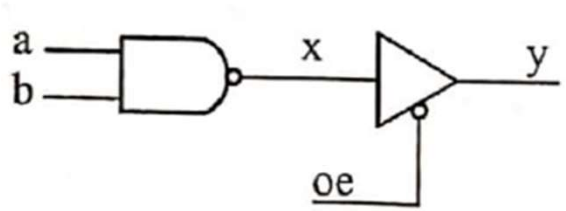
نوع std_logic این مزیت را دارد که دارای کتابخانه‌های استاندارد است به‌طوری که در تمام ابزارهای FPGA شناخته شده و قابل استفاده است، در صورتی که نوع bit یا ... دارای چنین ویژگی نیست.

نوع Std_Logic Type و Std_Ulogic Type

مثال

مدار زیر را با استفاده از std_logic توصیف نمایید.

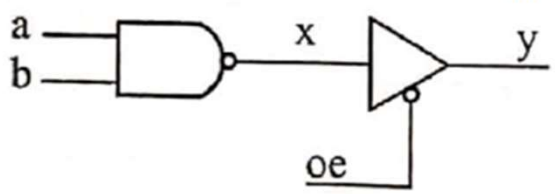
```
library ieee;
use ieee.std_logic_1164.all;
entity buf is
     port(a,b : in std_logic;
          oe : in std_logic;
          y : out std_logic;
end;
architecture bu of buf is
     signal x:std_logic;
     begin
          x <= a nand b;
          y <= x when oe='0' else 'Z';
     end;
```

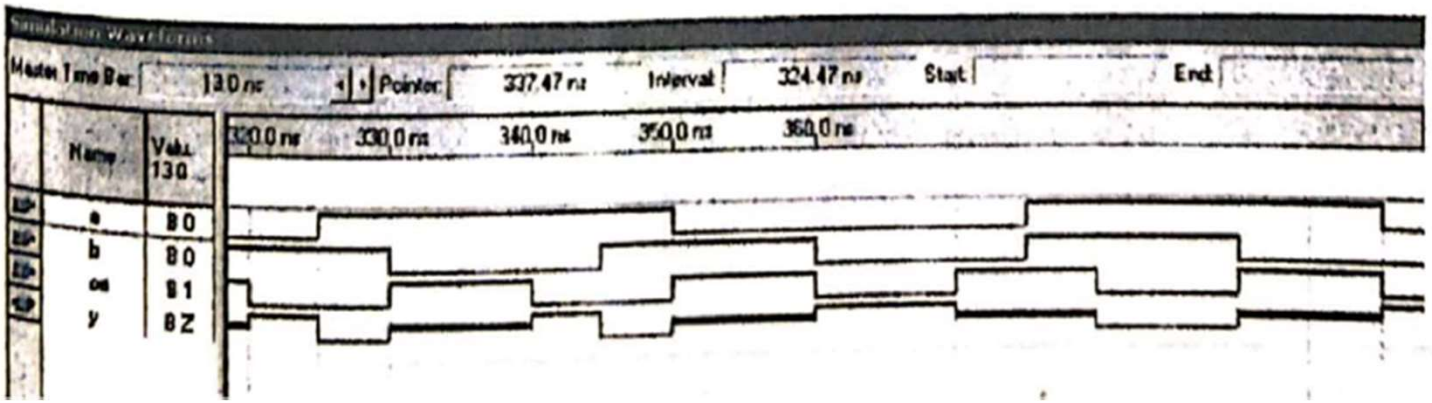


نوع Std_Logic Type و Std_Ulogic Type

مثال

مدار زیر را با استفاده از std_logic توصیف نمایید.





نوع Std_Logic_Vector Type و Std_Ulogic_Vector Type

برای std_logic و std_ulogic آرایه‌ای از بیت‌ها از نوع std_logic_vector و std_ulogic_vector نیز در package کتابخانه std_logic_1164 داریم.

تمام عملگرهایی که برای bit و bit_vector قابل اجرا می‌باشند، برای نوع‌های بالا نیز قابل اجرا هستند.

```
architecture RTL of exm is
     signal a,b:std_logic_vector(3 downto 0);
     signal c:std_logic_vector(3 downto 0);
     begin
          a<="0101";
          a<="1101";
          c<= a AND b;
     end;
```

باید دقت نمود که تعداد بیت‌های a و b یکسان باشد.

نوع Std_Logic_Vector Type و Std_Ulogic_Vector Type

یعنی برخی از بیت‌های f برابر با 0، 1 یا امپدانس بالا Z است.  `f = "111Z00Z1"`

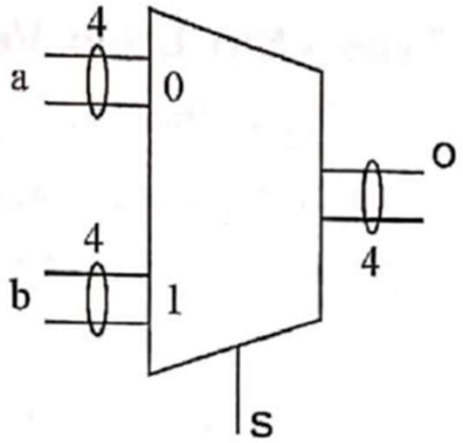
به این ترتیب، سیگنال نوع std_logic_vector می‌تواند مقادیری به‌غیر از 0 و 1 مانند Z و ... نیز داشته باشد.

در عبارات VHDL نوع (Type) سیگنال‌های دو طرف عبارت باید یکسان باشد.

نوع Std_Logic_Vector Type و Std_Ulogic_Vector Type

مثال

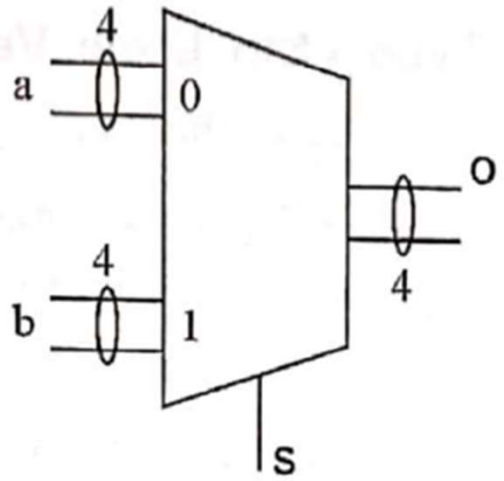
برنامه‌ای برای یک مولتی‌پلکسر ۲ به ۱ با ورودی‌های a و b و خروجی o چهاربیتی و با کنترل s یک بیتی بنویسید که اگر s=0 باشد، خروجی o برابر a و در غیر این‌صورت خروجی o مساوی b شود.



Std_Ulogic_Vector Type و Std_Logic_Vector Type نوع

دانشگاه سمنان
Semnan University
پردیس فرزانگان

```
library ieee;
use ieee.std_logic_1164.all;
----------
entity mux is
port (a,b: in std_logic_vector(3 downto 0);
      s: in std_logic;
      o: out std_logic_vector(3 downto 0));
end;
architecture data of mux is
begin
o <= a when s='0' else b ;
end;
```



تخصیص سیگنال با عبارت کلی Others

اگر در یک طرح تعداد بیت‌های سیگنال زیاد باشد و بخواهیم به تمام یا تعدادی از بیت‌ها، یک مقدار تخصیص دهیم در این صورت از عبارت others استفاده می‌کنیم

```
architecture RTL of example is
     signal a:std_logic_vector(7 downto 0);
     begin
          a<=others=>'0';
     end;
```

Aggregative Assignment

```
a<=others=>'z';
```

به تمام بیت‌های a که هشت بیت است، مقدار '0' را تخصیص می‌دهد که معادل است با

```
a<="00000000";
```

به تمام بیت‌های a، مقدار 'z' داده می‌شود

```
a<="zzzzzzzz";
```

تخصیص سیگنال با عبارت کلی Others

می‌توان به تعدادی از بیت‌ها مقداری تخصیص داد و برای بقیه بیت‌ها از others استفاده نمود

```
a<=(1=>'1' , 3=>'1' , others=>'0');
```

عبارت others باید آخرین انتخاب باشد.

تخصیص سیگنال با عبارت کلی Others

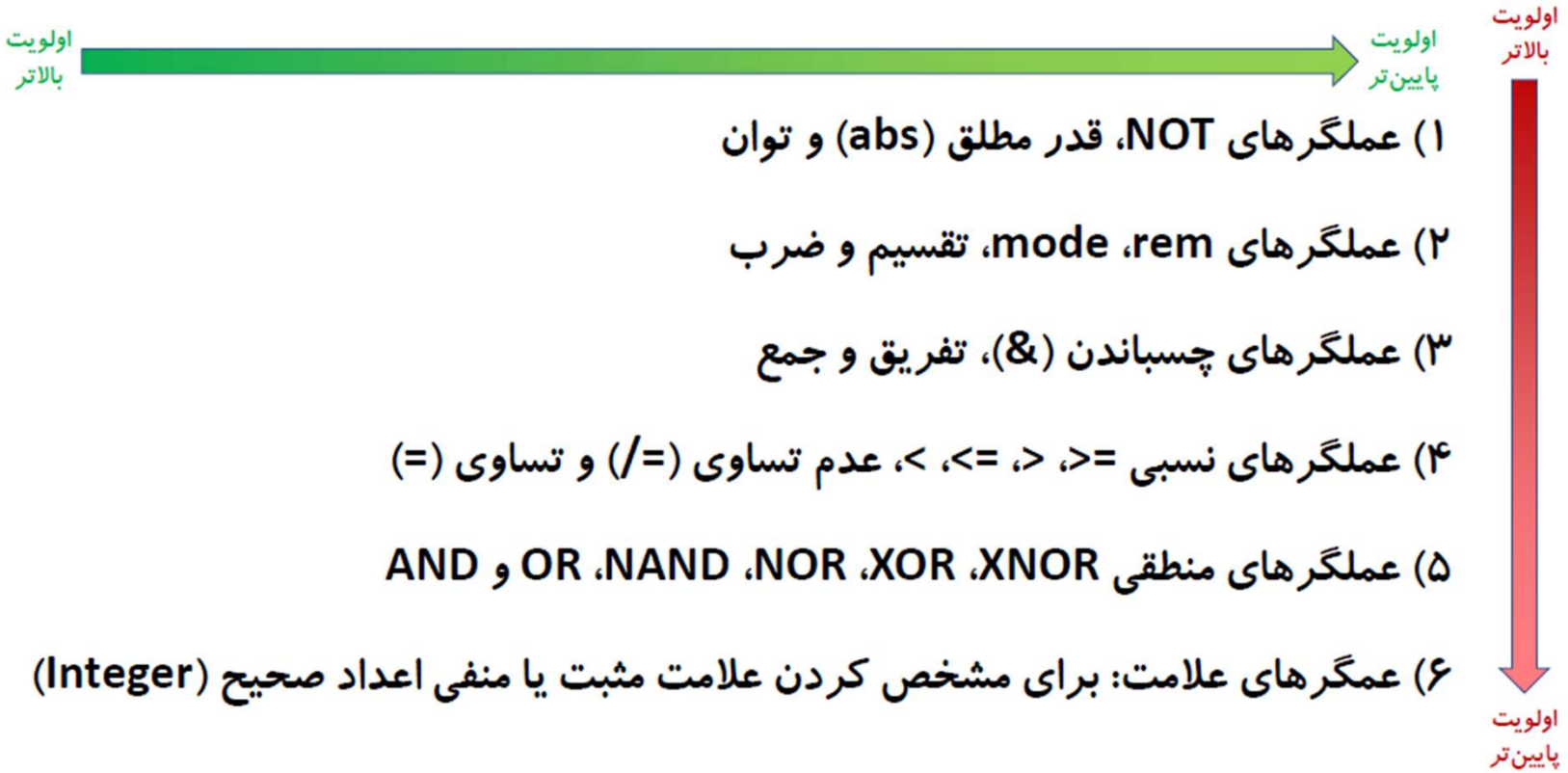
عبارت کلی others در تخصیص مقداری به سگنال به‌صورت شرطی

“Conditional Signal Assignment”

برای case و when ... else نیز نوشته می‌شود

```
begin
      f<=a when sel="00" else
           c when others;
end
```

عملگرها در VHDL



۱) عملگرهای NOT، قدر مطلق (abs) و توان

۲) عملگرهای rem، mode، تقسیم و ضرب

۳) عملگرهای چسباندن (&)، تفریق و جمع

۴) عملگرهای نسبی =<، <، =>، >، عدم تساوی (=/) و تساوی (=)

۵) عملگرهای منطقی XNOR، XOR، NOR، NAND، OR و AND

۶) عمگرهای علامت: برای مشخص کردن علامت مثبت یا منفی اعداد صحیح (Integer)

عملگرها در VHDL

```
res <= a AND NOT(b)    OR    NOT(b) AND b;

res <= (a AND NOT(b))    OR    (NOT(b) AND b);
```

عملگرهای منطقی

عملگرهای منطقی شامل XNOR، XOR، NOR، NAND، OR و AND هستند که برای عملیات روی سیگنال‌های نوع bit و bit_vector یا std_logic، std_logic_vector و Boolean استفاده می‌شوند.

در سیگنال‌های bit_vector و std_logic_vector عملیات منطقی بر روی هر یک از بیت‌ها انجام می‌شود و نتیجه به‌صورت بیت با بیت حاصل می‌شود.

عملگرهای ریاضی

این عملگرها عبارتند از +، -، *، / (تقسیم)، توان، abs، mod و rem هستند که برای عملیات روی سیگنال‌های نوع integer و با استفاده از کتابخانه‌ها روی سیگنال‌های نوع std_logic_vector نیز انجام می‌شوند.

مثال

برنامه‌ای بنویسید که دو عدد چهار بیتی a و b که به‌صورت std_logic_vector توصیف شده‌اند را با هم جمع کند و نتیجه را به sum بدهد.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity sample is
     port(a,b:in std_logic_vector(3 downto 0);
          sum:out std_logic_vector(3 downto 0));
end;
architecture a1 of sample is
begin
     sum <= a+b;
end;
```

عملگرهای مقایسه‌ای یا نسبی

- این عملگرها دو اپراند را با هم مقایسه می‌کنند و اعلام می‌کنند که آیا نتیجه مقایسه «درست True» یا «نادرست False» است.
- این عملگرها شامل =>، <، =<، >، عدم تساوی (=/) و تساوی (=) هستند که نتیجه تمام این عملیات، Boolean یعنی True یا False می‌باشد.
- اپراندهایی که با هم مقایسه می‌شوند برای عملگرهای = و =/ از هر نوع می‌توانند باشند و برای عملگرهای =>، <، =<، > ممکن است از نوع Integer یا bit_vector باشد.

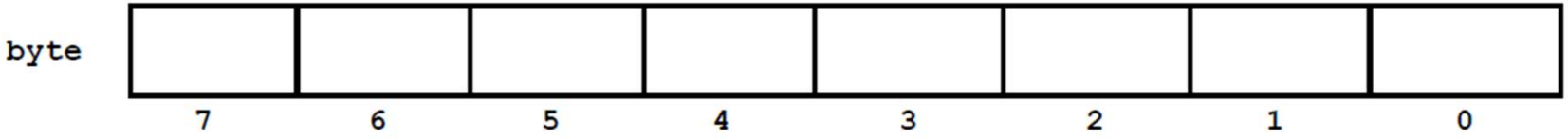
```
if ( (A=B) AND ((C>D) OR (E/=F)) ) then
      Y <= '1';
end if;
```

عملگر چسباندن &

در VHDL گاهی لازم است یک یا گروهی از سیگنال‌های از یک نوع به هم چسبیده شوند. در این مواقع از عملگر & در طرف راست عبارت استفاده می شود.

```
architecture example1 of concatenation is
     signal byte:bit_vector(7 downto 0);
     signal A_BUS , B_BUS : bit_vector(7 downto 0);
     begin
          byte <= A_BUS & B_BUS;
end example1
```

عملگر چسباندن &



عملگر چسباندن &

```
architecture example2 of concatenation is
      signal Z_BUS:bit_vector(7 downto 0);
      signal A,B,C,D : bit;
      begin
            Z_BUS <= A & B & C & D;
end example2
```

عملگر چسباندن (&) حتماً در طرف راست عبارت قرار می‌گیرد و سیگنال‌های از یک نوع را به هم می‌چسباند.

دانشگاه سمنان
Semnan University
پردیس فرزانگان

طراحی کامپیوتری سیستم‌های دیجیتال

Computer-Aided Digital System Design

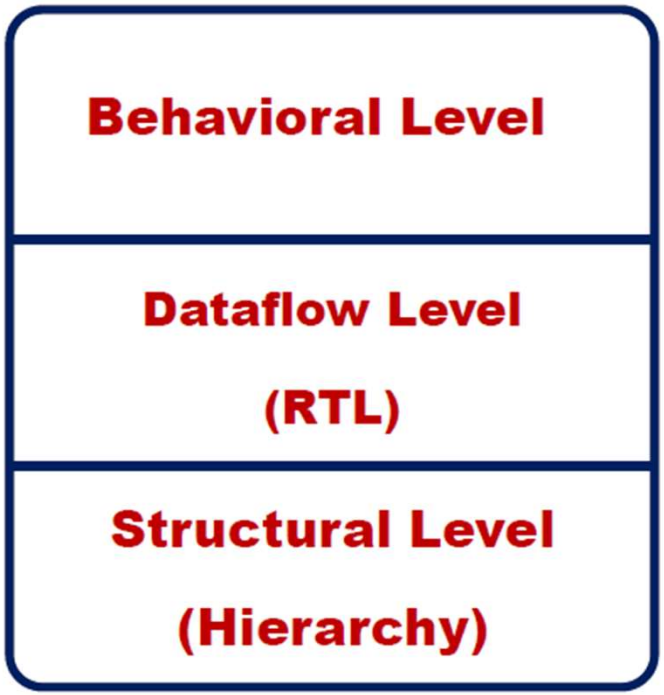
زبان توصیف سخت‌افزار VHDL

«بخش دوم»

چشم‌انداز

- روش‌های توصیف یا مدل‌سازی مدارهای دیجیتال
- توصیف یا مدل‌سازی Dataflow یا VHDL هم‌روند
- مدل کردن تأخیر در VHDL
- تخصیص یا انتساب مقداری به سیگنال به‌صورت شرطی و انتخابی

روش‌های توصیف یا مدل‌سازی مدارهای دیجیتال در VHDL



طراحی مدارهای دیجیتال با VHDL

با رویکرد بالا به پایین (Top-Down)

توصیف یا مدل‌سازی Behavioral

- طراحی در سطح سیستم یا در سطح Behavioral است که شامل یک سری عملیات منطقی و ریاضی روی اطلاعات ورودی مدار می‌باشد.
- در این مرحله، الگوریتم مدار بررسی و تست می‌شود.
- فقط مدل طرح، شبیه‌سازی می‌شود که مشخصات مدار مورد بررسی قرار گیرد.
- در این مرحله، سنتز مورد نظر نیست.

```
counter <= counter + 1
```

توصیف یا مدل‌سازی Dataflow یا RTL

- در این مرحله، مدار را به‌طریقی که داده‌ها داخل سیستم می‌شوند، از قسمت‌های مختلف می‌گذرند و به خروجی می‌روند، توصیف می‌کنند.

```
a <= a xor b
```

- اگر مدار دارای قسمت ترکیبی و همچنین فلیپ‌فلاپ یا ثبات باشد، در این صورت توصیف مدار به‌صورت RTL نامیده می‌شود.

توصیف یا مدل‌سازی Structural

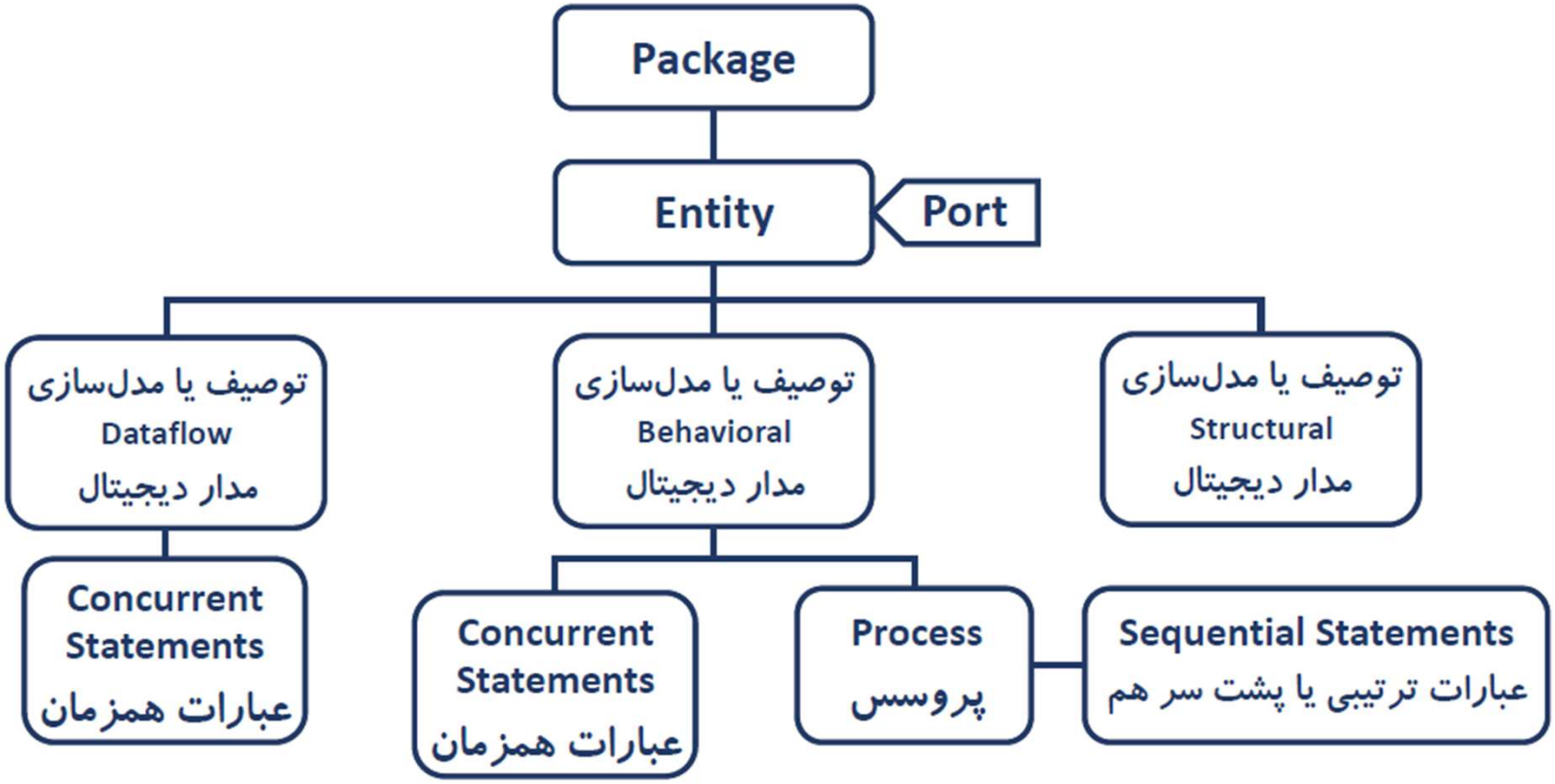
- در این روش، مدار به‌صورت مجموعه‌ای از قطعات و طرز اتصال آن‌ها توصیف می‌شود.
- در این روش مشخص می‌شود مدار از چه گیت‌ها یا قطعاتی تشکیل می‌شود و چگونه با هم ارتباط دارند.
- با این ساختار می‌توان مدار را می‌توان به‌صورت بلوک دیاگرام با قطعات و طرز اتصال آن‌ها به‌صورت سلسله مراتبی (Hierarchy) توصیف نمود.

توصیف یا مدل‌سازی Structural

- اگر مدل Behavioral را به ابزار سنتز بدهیم، یک طرح کلی تولید می‌کند که ممکن است این طرح، فرم بهینه نباشد و مقدار سخت‌افزار زیادی نیاز داشته باشد و یا سرعت کافی نداشته باشد.

- اما اگر توصیف Structural به ابزار سنتز داده شود، طرح مشخص‌تر و با حداقل سخت‌افزار و سرعت مناسب تولید خواهد شد.

توصیف یا مدل‌سازی مدارهای دیجیتال در VHDL

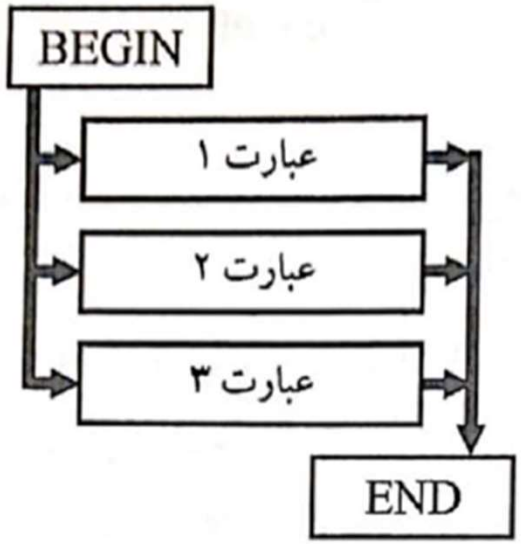


توصیف یا مدل‌سازی Dataflow یا VHDL هم‌روند (Concurrent VHDL) برای مدارهای ترکیبی

- در این روش چگونگی اتصال ورودی‌ها و خروجی‌های قطعات در یک سیستم دیجیتال توصیف می‌شود.
- در توصیف Dataflow توضیح داده می‌شود که چطور یک مدار کار می‌کند و چگونه داده‌ها از ورودی‌های مدار ترکیبی وارد و از قسمت‌های مختلف مدار می‌گذرد و در نهایت به خروجی می‌روند.
- در VHDL این عمل با «عبارت تخصیص سیگنال به‌طور همزمان با به‌طور ساده» یا به‌صورت ساره با عبارات «VHDL همزمان» انجام می‌شود.

Simple or Concurrent Signal Assignment Statements

توصیف یا مدل‌سازی Dataflow یا VHDL هم‌روند (Concurrent VHDL) برای مدارهای ترکیبی




```
a <= a XOR b;
c <= a AND b;
```

توصیف یا مدل‌سازی Dataflow یا VHDL هم‌روند (Concurrent VHDL) برای مدارهای ترکیبی

- این روش توصیف مدار را VHDL همزمان یا VHDL ترکیبی نیز می‌گویند.

- چون در طراحی برخی از سیستم‌های دیجیتال، ثبات‌ها وجود دارند، بنابراین در طراحی سخت‌افزارهایی که تعدادی ثبات یا فلیپ‌فلاپ دارند، توصیف Dataflow را توصیف RTL نیز می‌نامند.

- در این حالت مدارهای ترکیبی، ورودی فلیپ‌فلاپ‌ها را فراهم می‌کنند.

تخصیص یا انتساب مقداری به سیگنال در VHDL
Concurrent Signal Assignment Statement or Simple

```
Target Signal <= Expression;
```

و یا

سیگنال <= عبارت VHDL;

```
s <= a XOR b;
```

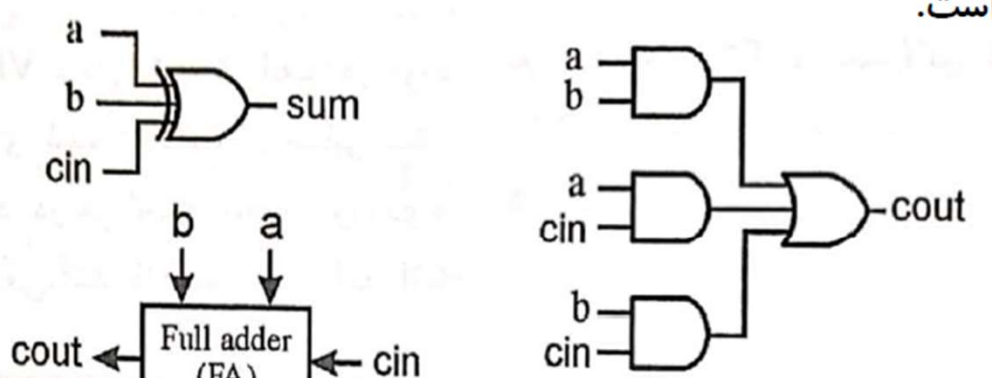
در تخصیص مقداری به سیگنال، دو طرف عبارت باید دارای نوع‌های (Types) یکسان باشند.

برای تخصیص مقداری به سیگنال، از علامت <= استفاده می‌شود.

تخصیص یا انتساب مقداری به سیگنال در VHDL
Concurrent Signal Assignment Statement or Simple

مثال

جمع‌کننده کامل (FA) با ورودی‌های a و b و cin خروجی‌های sum و cout به‌صورت شکل زیر است.



الف- مدار جمع‌کننده کامل (FA)

ب- نمایش سمبولیک

تخصیص یا انتساب مقداری به سیگنال در VHDL
Concurrent Signal Assignment Statement or Simple

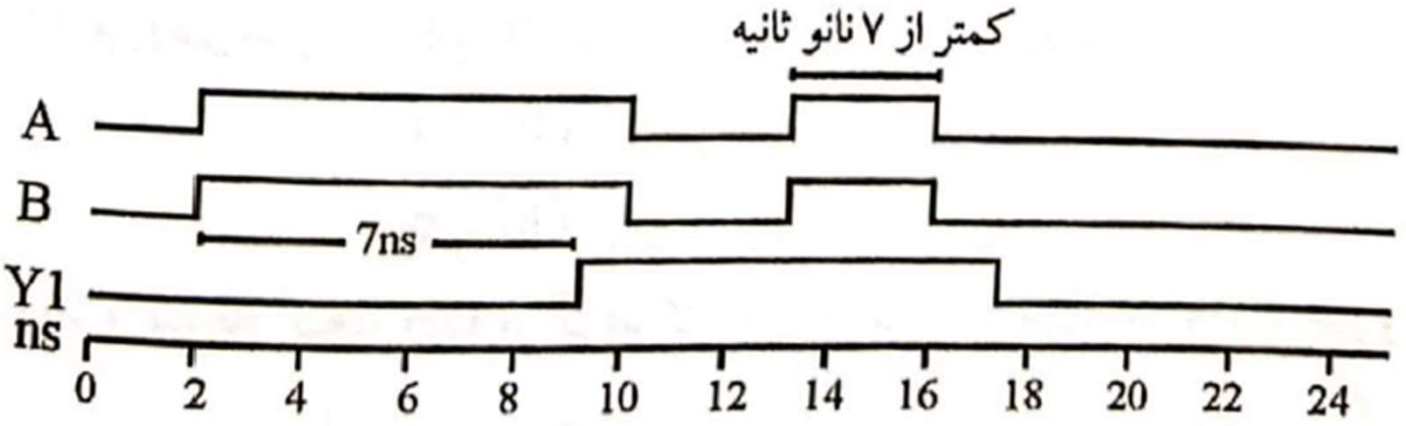

```
library ieee;
use ieee.std_logic_1164.all;
entity dataflow is
     port(a,b,cin : in std_logic;
          sum,cout : out std_logic);
end;
architecture df of dataflow is
Begin
     sum <= a xor b xor cin;
     cout <= (a and b) or (b and cin) or (a and cin);
end
```

مدل کردن تأخیر در VHDL

الف ـ تأخیر در گیت‌ها

به عنوان مثال برای مدل کردن تأخیر در گیت AND می‌توان نوشت:

```
y1 <= A and B after 7 ns;
```

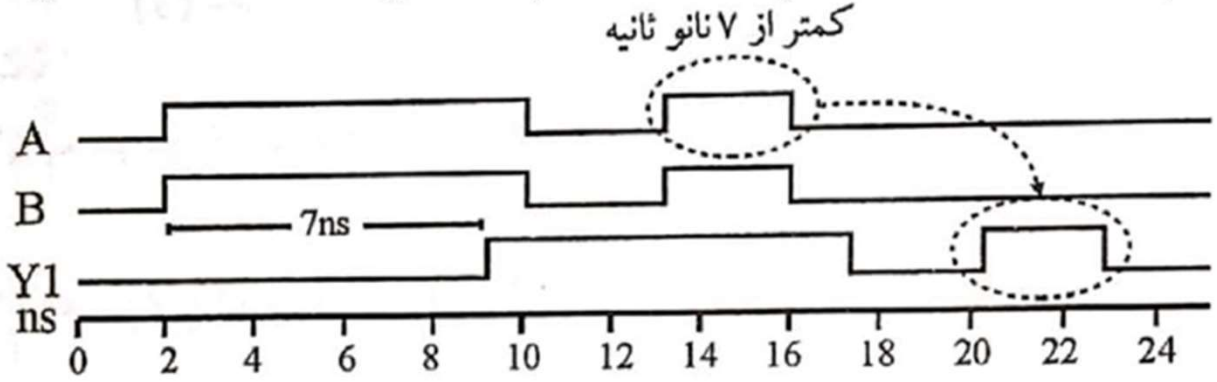


مدل کردن تأخیر در VHDL

الف ـ تأخیر انتشار در سیم Transport Delay

تأخیر Transport مقدار تأخیر در یک سیم را مدل می‌کند و عرض پالس ورودی در آن اثری ندارد.

```
y2 <= Transport(A and B) after 7 ns;
```



تخصیص یا انتساب مقداری به سیگنال به‌صورت شرطی
Conditional Signal Assignment

سیگنال <= عبارت ۱ when شرط else عبارت ۲;

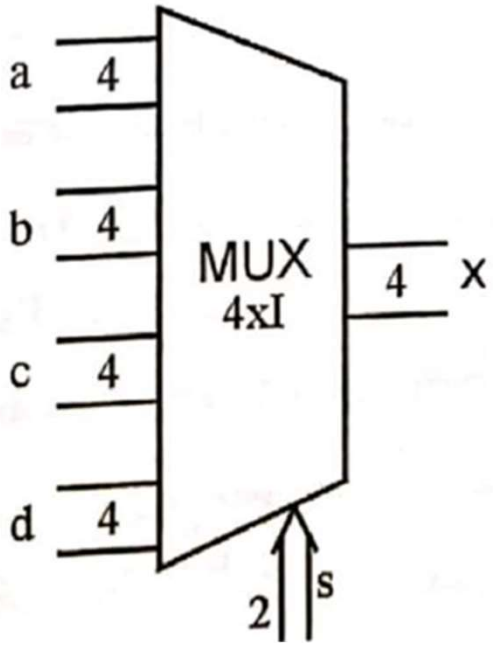
```
Y <= A when sel='1' else B;

sum <= (a xor b) when c='0' else NOT(a xor b);

Z <= a when sel1='0' else
     b when sel2='1' else
     c;
```

شرط‌ها به‌ترتیب از بالا به پایین تست می‌شوندتا نتیجه تست معلوم شود؛ یعنی عبارات از بالا به پایین اولویت دارند.

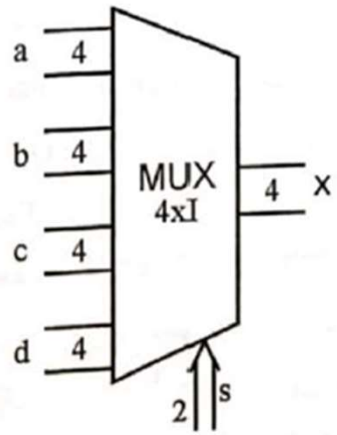
تخصیص یا انتساب مقداری به سیگنال به‌صورت شرطی
Conditional Signal Assignment

مثال

برنامه VHDL مولتی‌پلکسر زیر را بنویسید.



```
library ieee;
use ieee.std_logic_1164.all;
entity maxwhen is
     port(a : in std_logic_vector(3 downto 0);
          b : in std_logic_vector(3 downto 0);
          c : in std_logic_vector(3 downto 0);
          d : in std_logic_vector(3 downto 0);
          s : in std_logic_vector(1 downto 0);
          x : out std_logic_vector(3 downto 0));
End maxwhen;
architecture maxwhen_arch of maxwhen is
     begin
          x <= a when (s='00') else
               b when (s='01') else
               c when (s='10') else
               d;
     end maxwhen_arch ;
```



تخصیص یا انتساب مقداری به سیگنال به‌صورت انتخابی
Selected Signal Assignment

این روش، شبیه روش Conditional Signal Assignment است و بر حسب شرط، یکی از مقادیر به سیگنال سمت چپ تخصیص داده می‌شود.

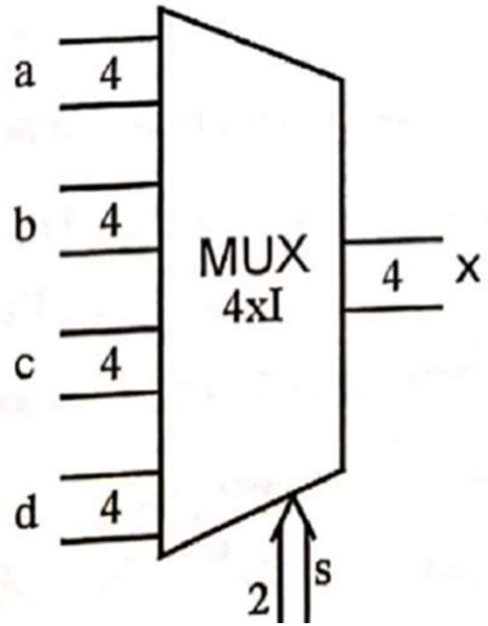
```
with s select
        Z <= a when '1',
             b when '0';
```

در روش تخصیص مقداری به سیگنال به‌صورت شرطی (when ... else) شرط‌ها به‌ترتیب از بالا به پایین تست می‌شوند. در صورتی‌که در روش تخصیص به‌صورت انتخابی (with ... select) تمام شرط‌ها در سطرهای مختلف، همزمان بررسی می‌شوند و اولولتی نسبت به هم ندارند.

تخصیص یا انتساب مقداری به سیگنال به‌صورت انتخابی
Selected Signal Assignment

مثال

برنامه VHDL مولتی‌پلکسر زیر را با روش تخصیص به‌صورت انتخابی بنویسید.



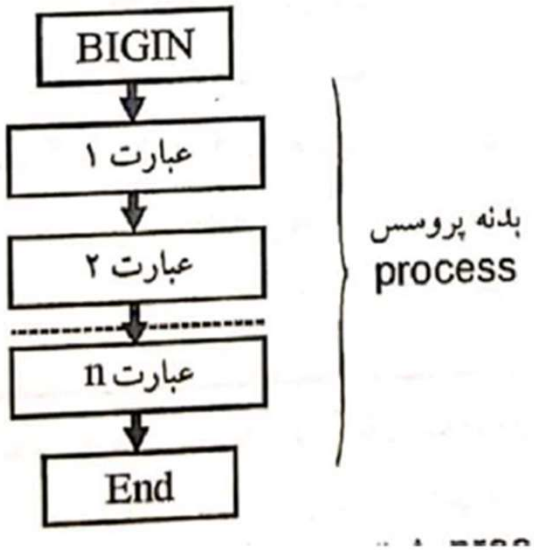
```
library ieee;
use ieee.std_logic_1164.all;
entity maxwith is
     port(a : in std_logic_vector(3 downto 0);
          b : in std_logic_vector(3 downto 0);
          c : in std_logic_vector(3 downto 0);
          d : in std_logic_vector(3 downto 0);
          s : in std_logic_vector(1 downto 0);
          x : out std_logic_vector(3 downto 0));
end maxwith;
architecture maxwith_arch of maxwith is
     begin
          with s select
               x <= a when "00",
                    b when "01",
                    c when "10",
                    d when others;
end maxwith_arch ;
```

توصیف یا مدل‌سازی رفتاری (Behavioral Modeling)

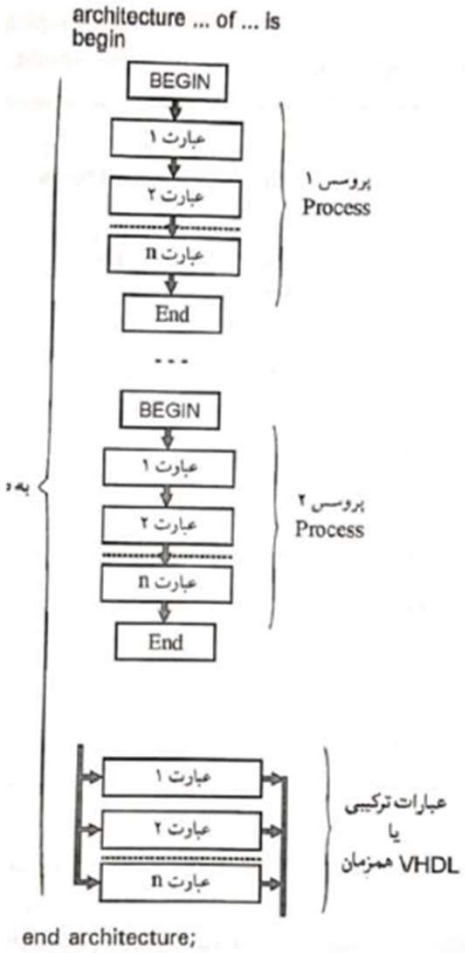
- توصیف یا مدلسازی رفتاری یک سیستم دیجیتال به‌صورت الگوریتمی است، یعنی تغییرات خروجی بر حسب ورودی با الگوریتمی مشخص می‌شود.
- توصیف رفتاری معمولاً برای طراحی و شبیه‌سازی مدارهای پیچیده استفاده می‌شود.
- در این حالت ساختار داخلی مدار برای ما مهم نیست، بلکه عملکرد و طرز کار آن مورد نظر می‌باشد.
- در مدلسازی رفتاری فقط الگوریتم مدار مورد توجه است که با شبیه‌سازی، عملکرد آن مورد تأیید قرار می‌گیرد.

پروسس (process) و VHDL ترتیبی (sequential VHDL)

پروسس (process) در **بدنه آرشیتکت** یک طرح قرار می‌گیرد و عبارات داخلی process مانند برنامه‌های معمولی نرم‌افزاری مثلاً زبان C به‌صورت **پشت سر هم**، یکی یکی اجرا می‌شوند که به **sequential VHDL** معروف است.



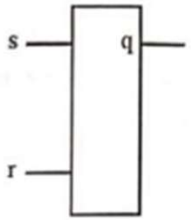
پروسس (process) و VHDL ترتیبی (sequential VHDL)



پروسس (process) و VHDL ترتیبی (sequential VHDL)

مثال

برنامه VHDL برای یک فلیپ‌فلاپ rs بنویسید، به‌طوری‌که اگر "s="0 شود، خروجی q برابر 1 گردد و اگر "r="0 شود، خروجی q مساوی 0 گردد. در غیر این‌صورت خروجی تغییری ننماید.

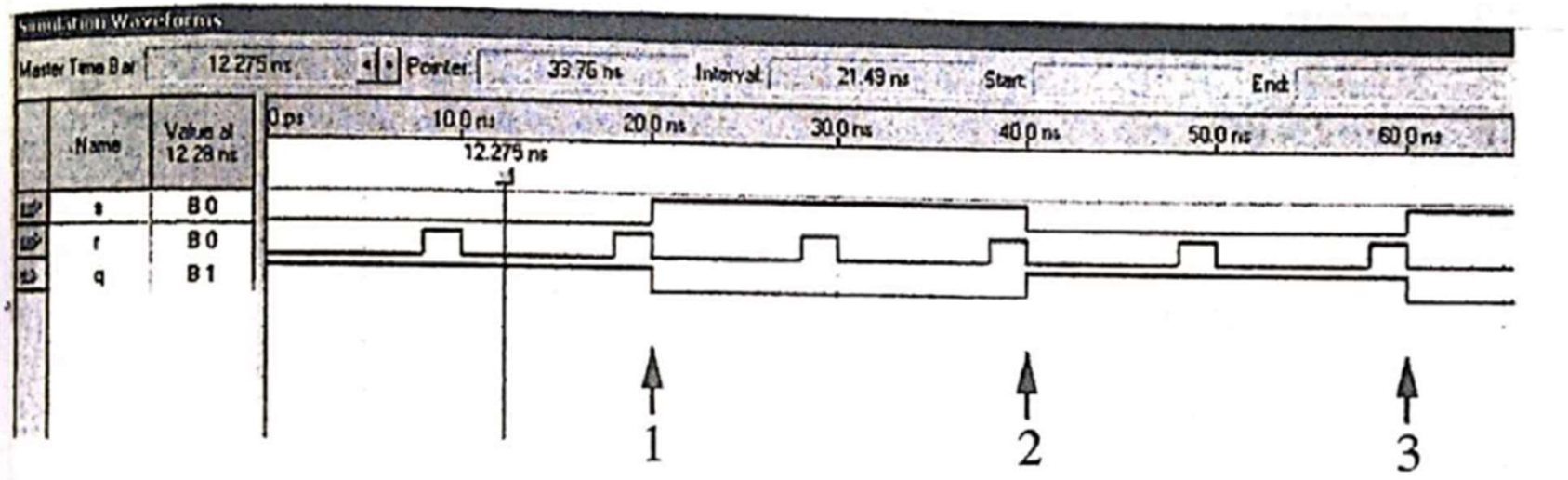



```
library IEEE;
use IEEE.std_logic_1164.all;

entity flipsrbe is
   port (
      s: in STD_LOGIC;
      r: in STD_LOGIC;
      q: inout STD_LOGIC
   );
end flipsrbe;

architecture flipsrbe_arch of flipsrbe is
begin
process (s,r,q)                    --(1)
begin
        if s='0' then              --(2)
                q<='1';
        elsif r='0' then   --(3)
                q<='0';
        else q<=q;                 --(4)
        end if;
end process;
end flipsrbe_arch;
```

پروسس (process) و VHDL ترتیبی (sequential VHDL)

مثال

برنامه VHDL برای یک فلیپ‌فلاپ rs بنویسید، به‌طوری‌که اگر "s="0 شود، خروجی q برابر 1 گردد و اگر "r="0 شود، خروجی q مساوی 0 گردد. در غیر این‌صورت خروجی تغییری ننماید.



پروسس (process) و VHDL ترتیبی (sequential VHDL)

```
[process برچسب] : process (لیست سیگنال‌ها)
            اعلانات
begin
            عبارت ۱
            عبارت ۲
            عبارت ۳
            .
            .
            .
            عبارت n
end process [process برچسب]
```

پشت سر هم اجرا می‌شوند

عبارات شرطی در پروسس (process)

```
if condition then
      statement 1
else
      statement 2
end if;
```

```
if r = '1' then q<='0';
else q<=q;
end if;
```

```
if condition then
      statement
end if;
```

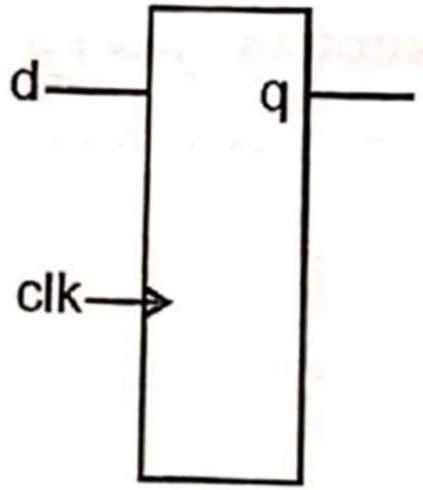
```
if a/=b then
      MIN <= a;
end if;
```

```
if condition1 then
      statement 1
elsif condition2 then
      statement 2
else
      statement 3
end if;
```

عبارات شرطی در پروسس (process)

مثال

برنامه VHDL به‌صورت behavioral برای فلیپ‌فلاپ D بنویسید به‌طوری‌که در لبه بالارونده پالس ساعت (clk)، اطلاعات ورودی D به خروجی Q منتقل شود.



عبارات شرطی در پروسس (process)


```
library ieee;
use ieee.std_logic_1164.all;
entity flip is
     port(d,clk : in std_logic;
          q : out std_logic);
end flip;
architecture flipclk of flip is
     begin
     process(d,clk)
     begin
          if (clk'event and clk='1') then q<=d;
          end if;
     end process;
end;
```

عبارات شرطی در پروسس (process)

مثال

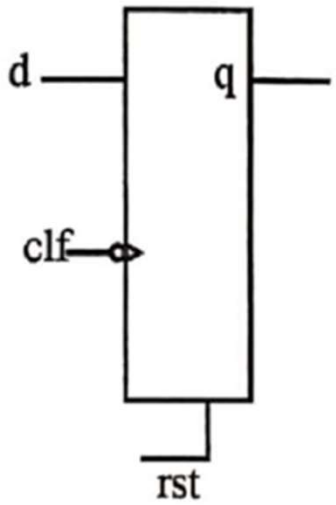
برنامه VHDL به‌صورت behavioral برای فلیپ‌فلاپ D بنویسید به‌طوری‌که در لبه پایین‌رونده پالس ساعت (clk)، اطلاعات ورودی D به خروجی Q منتقل شود.

```
library ieee;
use ieee.std_logic_1164.all;
entity flipclk is
     port(d,clk : in std_logic;
          q : out std_logic);
end flipclk;
architecture flipclk_arch of flipclk is
     begin
     process(d,clk)
     begin
          if (clk'event and clk='0') then q<=d;
          end if;
     end process;
End flipclk_arch;
```

عبارات شرطی در پروسس (process)

مثال

برنامه VHDL به‌صورت behavioral برای فلیپ‌فلاپ D بنویسید به‌طوری‌که اگر ورودی rst برابر 0 شود، خروجی Q مساوی 0 گردد، در لبه بالارونده پالس ساعت (clk)، اطلاعات ورودی D به خروجی Q منتقل شود.



عبارات شرطی در پروسس (process)

```
library ieee;
use ieee.std_logic_1164.all;
entity flipclk is
     port(d,rst : in std_logic;
          clk : in std_logic;
          q : out std_logic);
end flipclk;
architecture flipclk_arch of flipclk is
     begin
     process(clk,rst)
     begin
          if rst='0'then q<='0';
          elsif (clk'event and clk='1') then q<=d
          end if;
     end process;
end flipclk_arch;
```

عبارات شرطی در پروسس (process)

مثال

برنامه VHDL برای یک ثبات هشت بیتی با فلیپ‌فلاپ D بنویسید که با لبه بالارونده پالس ساعت (clk)، اطلاعات هشت بیتی ورودی a به خروجی Q منتقل شود.

عبارات شرطی در پروسس (process)

```
library ieee;
use ieee.std_logic_1164.all;
entity register1 is
     port(a: in bit_vector(7 downto 0);
          clk : in bit;
          d : out bit_vector(7 downto 0));
end entity register1;
architecture beh of register1 is
     begin
     process(clk)
     begin
          if (clk'event and clk='1') then
               d <= a;
          end if;
     end process;
end architecture;
```

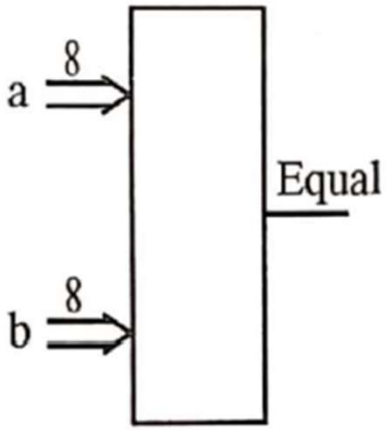
پروسس (process) برای مدارهای ترکیبی (Combinational Process)

- پروسس (process) می‌تواند برای مدارهای ترکیبی نیز به‌کار رود.
- چون در مدارهای ترکیبی، با هر تغییر ورودی، خروجی نیز تغییر می‌کند، لذا تمام ورودی‌های مدار باید در لیست سیگنال‌های process باشند تا با تغییر آن‌ها process اجرا شود و مقدارهای جدید خروجی‌ها را ارزیابی و محاسبه کند.

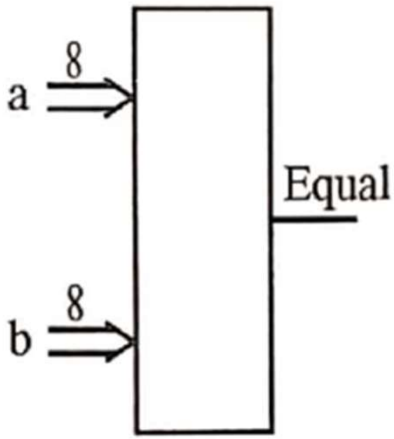
پروسس (process) برای مدارهای ترکیبی (Combinational Process)

مثال

مقایسه‌کننده‌ای دارای دو ورودی a و b هشت بیتی و خروجی equal یک بیتی است. اگر a=b باشد، باید equal برابر 1 شود، در غیر این‌صورت equal برابر صفر می‌گردد. برنامه VHDL آن را به‌صورت behavioral بنویسید.



```
library ieee;
use ieee.std_logic_1164.all;
entity proc_if2 is
     port(a: in std_logic_vector(7 downto 0);
          b: in std_logic_vector(7 downto 0);
          equal : out std_logic);
end proc_if2;
architecture proc_if2_arch of proc_if2 is
     begin
     process(a,b)
     begin
          if a=b then
               equal<='1'
          else
               equal<='0'
          end if;
     end process;
end proc_if2_arch;
```



عبارات case در پروسس (process)

```
case condition statement is
     when condition statement => statement 1
     when condition statement => statement 1
      .
      .
      .
     when others => statement n
end case;
```

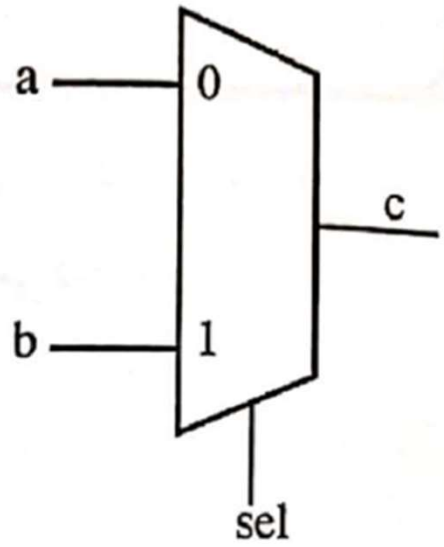
موقعی که case به‌کار برده می‌شود، تمام انشعاب‌ها دارای اولویت مساوی هستند.

پروسس (process) برای مدارهای ترکیبی (Combinational Process)

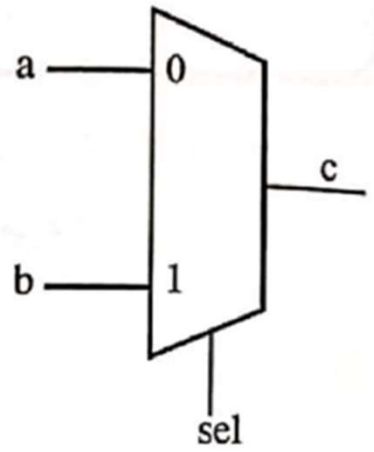
مثال

مولتی‌پلکسری با دو ورودی a و b و ورودی کنترل sel و خروجی c موجود است.

برنامه VHDL آن را با case بنویسید.



```
library ieee;
use ieee.std_logic_1164.all;
entity case1 is
     port(a: in std_logic;
          b: in std_logic;
          sel: in std_logic;
          c : out std_logic);
end case1;
architecture case1_arch of case1 is
     begin
     pro: process(sel,a,b)
     begin
          case sel is
               when '0' =>  c<=a;
               when others =>  c<=b;
          end case;
     end process;
end case1_qrch;
```

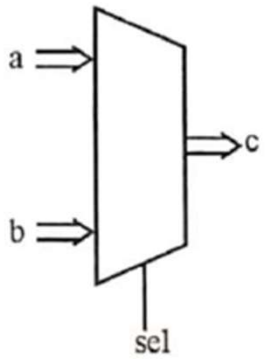


عبارت null

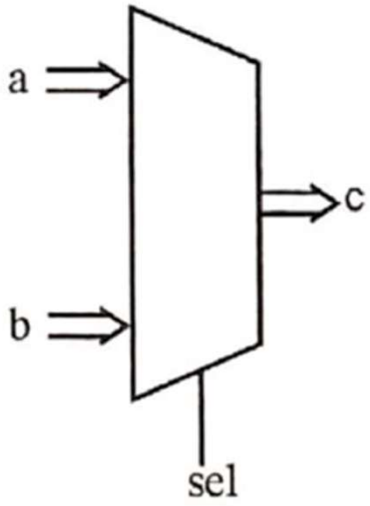
در VHDL عبارت null به معنای آن است که کاری انجام نشود و معمولاً در case استفاده می‌شود.

مثال

برنامه VHDL با case و null برای یک مولتی‌پلکسر دارای دو ورودی a و b چهاربیتی با کنترل sel و خروجی c چهاربیتی بنویسید.



```
library ieee;
use ieee.std_logic_1164.all;
entity case1 is
      port(a,b: in std_logic_vector(3 downto 0);
            sel: in std_logic;
            c : out std_logic_vector(3 downto 0));
end case1;
architecture case1_arch of case1 is
      begin
      pro: process(a,b,sel)
      begin
            case sel is
                  when '0' =>  c<=a;
                  when '1' =>  c<=b;
                  when others =>  null;
            end case;
      end process;
end;
```



کاربرد متغیر (variable) در process

در توصیف dataflow برای نمایش اطلاعات معمولاً از signal استفاده می‌شود.

در process علاوه بر سیگنال برای نگه‌داری و ذخیره اطلاعات، متغیر (variable) به‌کار برده می‌شود.

```
Count : process(clk)
        variable cnt:Integer;
        begin
        - - -
        - - -
        cnt := cnt + 1;
        - - -
        - - -
end process
```

```
process نام: process (لیست سیگنال‌ها)
        اعلانات { نوع : نام variable
                  ...
begin
    بدنه process شامل عبارات

end process process نام;
```

کاربرد متغیر (variable) در process


```
variable count_bit : bit:='0';
variable VAR1 : Boolean := False;
variable Sum : Integer range 0 to 255:=16;
variable sts_bit : bit_vector(7 downto 0);
variable count,overflow : Integer;
```

کاربرد متغیر (variable) در process

مثال

برنامه VHDL به‌صورت behavioral برای شمارنده هشت بیتی با ورودی clk و خروجی count با استفاده از process و variable بنویسید.

```
library ieee;
use ieee.std_logic_1164.all;
entity counter_int is
     port(clk: in std_logic;
          count : out Integer range 0 to 255);
end counter_int;
architecture behavioral of counter_int is
     begin
     process(clk)
     variable cnt : Integer range 0 to 255
     begin
          if (clk'event and clk='1') then
               cnt := cnt+1;
          end if;
          count <= cnt;
     end process;
end;
```

تفاوت variable و signal در process

- در process می‌توان signal و variable را استفاده نمود.
- اگر در process مقداری با علامت (:=) به variable داده شود، بلافاصله variable مقدار را می‌گیرد.
- اگر در process مقداری با علامت (<=) به signal داده شود، بعد از اینکه تمام عبارات process ارزیابی و تمام شد و process پایان یافت، مقدار مذکور به سیگنال داده می‌شود.

تفاوت variable و signal در process


```
begin
---
signal x,y,z : bit;
---
process(y)
    begin
        x <= y;
        z <= NOT x;
    end process;
end architecture;
```

```
begin
process(y)
variable x,z : bit;
    begin
        x := y;
        z := NOT x;
    end process;
end architecture;
```

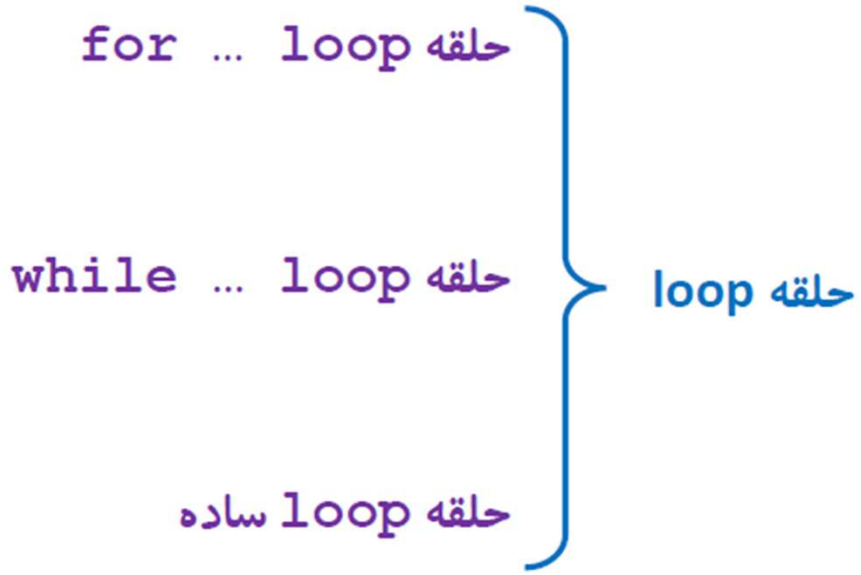
حلقه loop

- loop روشی برای تکرار اجرای عبارات به‌صورت پشت سر هم، یا انجام عملیاتی مشابه در برنامه VHDL است.
- loop داخل process قرار می‌گیرد و عبارات داخل آن به‌صورت پشت سر هم یا sequential اجرا می‌شوند.

```
[برچسب] : روش تکرار loop
          عبارت ۱
          عبارت ۲
          عبارت ۳
end loop [برچسب]
```

پشت سر هم اجرا می‌شوند

حلقه loop



حلقه for ... loop

• حلقه for ... loop روشی برای تکرار و اجرای قسمتی ار کد یا برنامه VHDL به تعداد معین داخل process است.

```
for i in 3 downto 0 loop;
for i in 0 to 3 loop;
```

• فرم کلی حلقه for ... loop

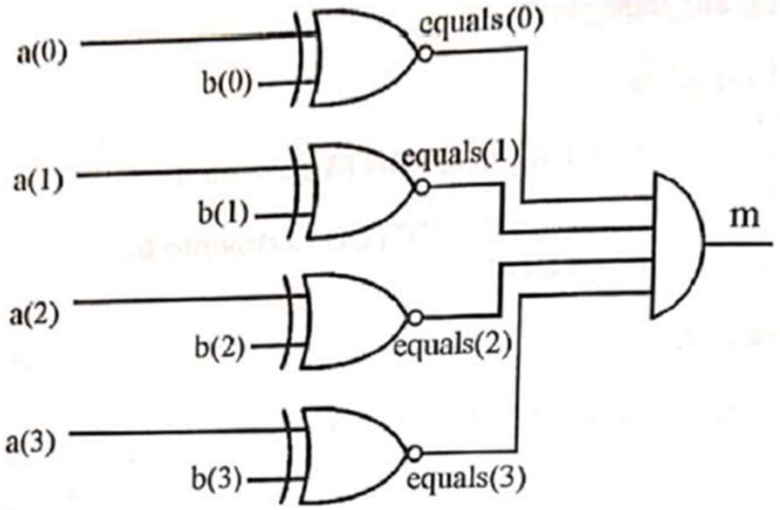
```
[برچسب] : for i محدوده عمل loop
          عبارت ۱
          عبارت ۲
          عبارت ۳
end loop [برچسب]
```

} پشت سر هم اجرا می‌شوند

for ... loop حلقه

مثال

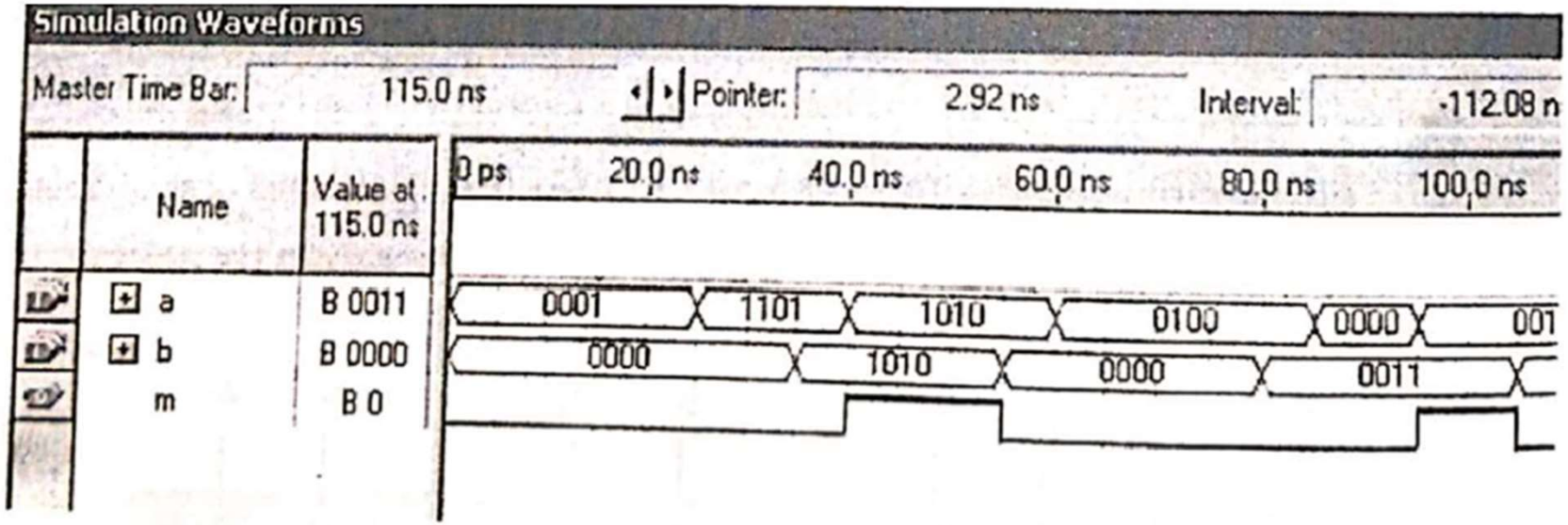
برنامه VHDL بنویسید که عناصر دو آرایه ۴ عنصری a و b را با هم مقایسه کند و در صورتی که عناصر دو آرایه با هم برابر بودند خروجی m را برابر با 1 کند.



for ... loop حلقه


```
library IEEE;
use IEEE.std_logic_1164.all;

entity forloop1 is
   port (
      a: in STD_LOGIC_VECTOR (3 downto 0);
      b: in STD_LOGIC_VECTOR (3 downto 0);
      m:out STD_LOGIC
         );
end forloop1;

architecture behaviour of forloop1 is
 signal  equals: STD_LOGIC_VECTOR (3 downto 0);                --(1)
  begin
   process (a,b)
     begin
      for i in 3 downto 0 loop
--sequence of statement
        equals(i)<= a(i) xnor b(i);                          --(2)
      end loop;
   endprocess;
   m <=equals(3) and equals(2) and equals(1) and equals(0);   --(3)
end behaviour;
```

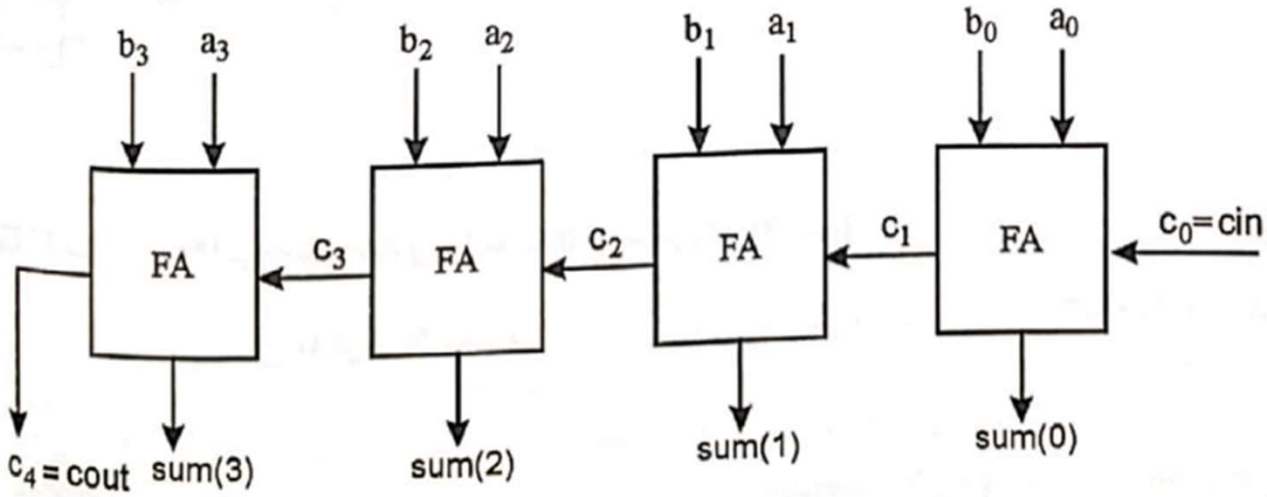
for ... loop حلقه



حلقه for ... loop

مثال

برنامه‌ای بنویسید که با استفاده از for ... loop یک جمع‌کننده ۴ بیتی را با چهار تمام‌جمع‌کننده (FA) توصیف نماید.

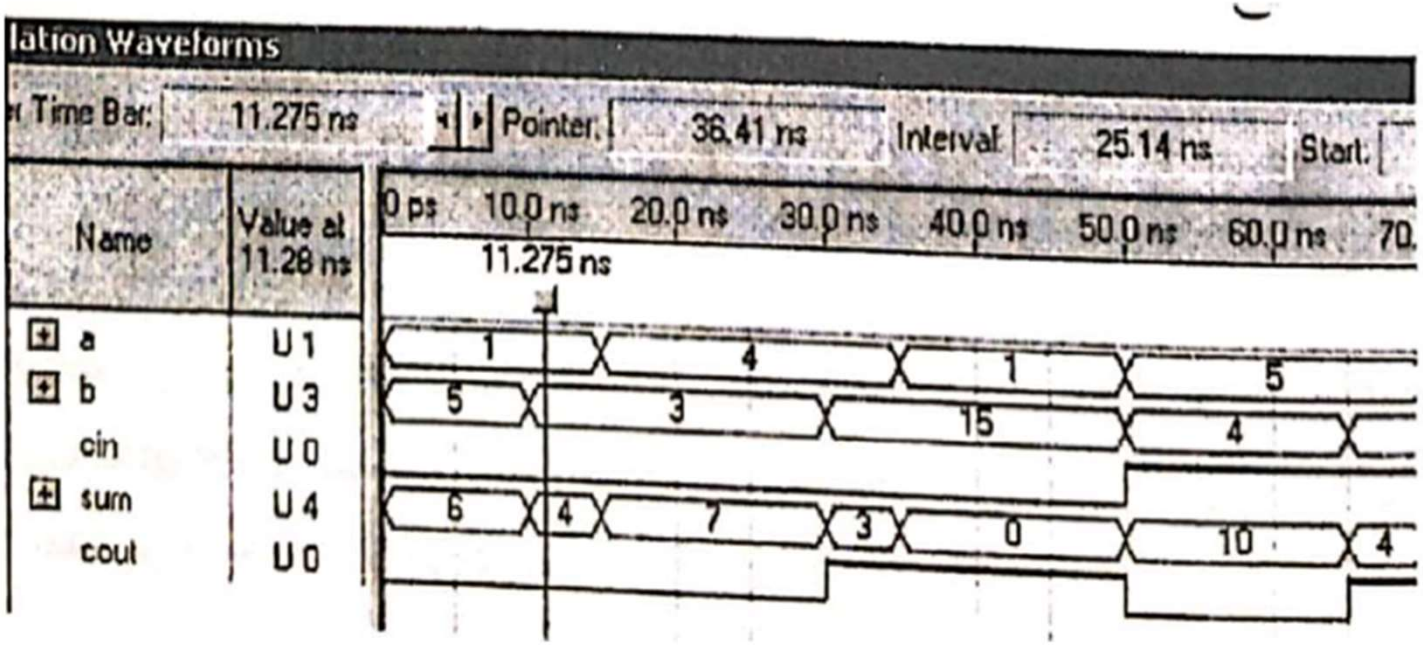


for ... loop حلقه

```
library IEEE;
use IEEE.std_logic_1164.all;

entity forloop2 is
  port (
    a,b: in STD_LOGIC_VECTOR (3 downto 0);
    cin: in STD_LOGIC;
    sum: out STD_LOGIC_VECTOR (3 downto 0);
    cout: out STD_LOGIC
  );
end forloop2;

architecture behaviour of forloop2 is
begin
  process(a,b)                                                    --(1)
    variable c:std_logic_vector(4 downto 0);                      --(2)
    begin
    c(0):=cin;
      for i in 0 to 3 loop                                        --(3)
        sum(i)<=a(i) xor b(i) xor c(i);                           --(4)
        c(i+1):=(a(i) and b(i)) or (b(i) and c(i)) or (a(i) and c(i));
      end loop;                                                   --(6)
     cout<=c(4);                                                  --(7)
  end process;
end behaviour;
```

for ... loop حلقه

دانشگاه سمنان
Semnan University
پردیس فرزانگان



عبارت exit در حلقه for ... loop

- در اجرای عبارات VHDL گاهی لازم است به خارج از حلقه loop رفته با وجود آنکه تعداد تکرارهای حلقه for ... loop ممکن است پایان نیافته باشد.

```
process(a)                                    --(1)
   variable int_a:integer;                    --(2)
  begin
     int_a:=a;                                --(3)
     for i in 0 to 50 loop                    --(4)
       if (int_a <=0) then                    --(5)
         exit;                                --(6)
      else
         int_a:=int_a -1;                     --(7)
       end if;                                --(8)
     end loop;                                --(9)
   Y<=q;                                      --(11)
end process;                                  --(12)
```

exit [برچسب] [when شرط]

عبارت Next در حلقه for ... loop

- عبارت Next در حلقه loop باعث می‌شودکه عبارت بعد از next صرفنظر شود و به ابتدای حلقه برگردد و حلقه‌های بعدی اجرا شود.

```
Process(a,b)                          --(1)
  begin
    for i in 0 to 50 loop             --(2)
      if (done(i)=True then           --(3)
        next;                         --(4)
      else
        done(i):=True;                --(5)
      end if;                         --(6)
    q(i)<=a(i) AND b(i);              --(7)
    end loop;                         --(8)
end process;
```

next [برچسب] [when شرط]

حلقه while ... loop

- در حلقه while ... loop ابتدا شرط تست می‌شود تا وقتی که شرط درست (True) باشد، اجرای loop ادامه می‌یابد و به محض اینکه شرط نادرست (False) شد loop پایان می‌یابد.

```
process
     begin
     while (شرط) loop
        عبارات پشت سر هم اجرا می‌شوند
     end loop;
end process;
```

while ... loop حلقه


```
process
  variable clock:bit:='0';
begin
  while error_flag='0' loop    --(1)
  clock:= NOT clock after 50 ns;--(2)
  end loop;                    --(3)
end process;                   --(4)
```

حلقه loop ساده

- حلقه loop ساده معمولاً با عبارت exit استفاده می‌شود

```
architecture infinit_loop of my_entity
     begin
     - - -
     process( )
     - - -
     begin
     - - -
          loop
               - - -
               exit;
          end loop;
     end process;
     - - -
end;
```

حلقه loop ساده

```
Process
   begin
    loop
      clock<=NOT clock;              --(1)
      wait for 50 ns;                --(2)
        if error_flag='1' then            --(3)
      exit;                    --(4)
      end if;                        --(5)
     end loop;                       --(6)
end process;                         --(7)
```

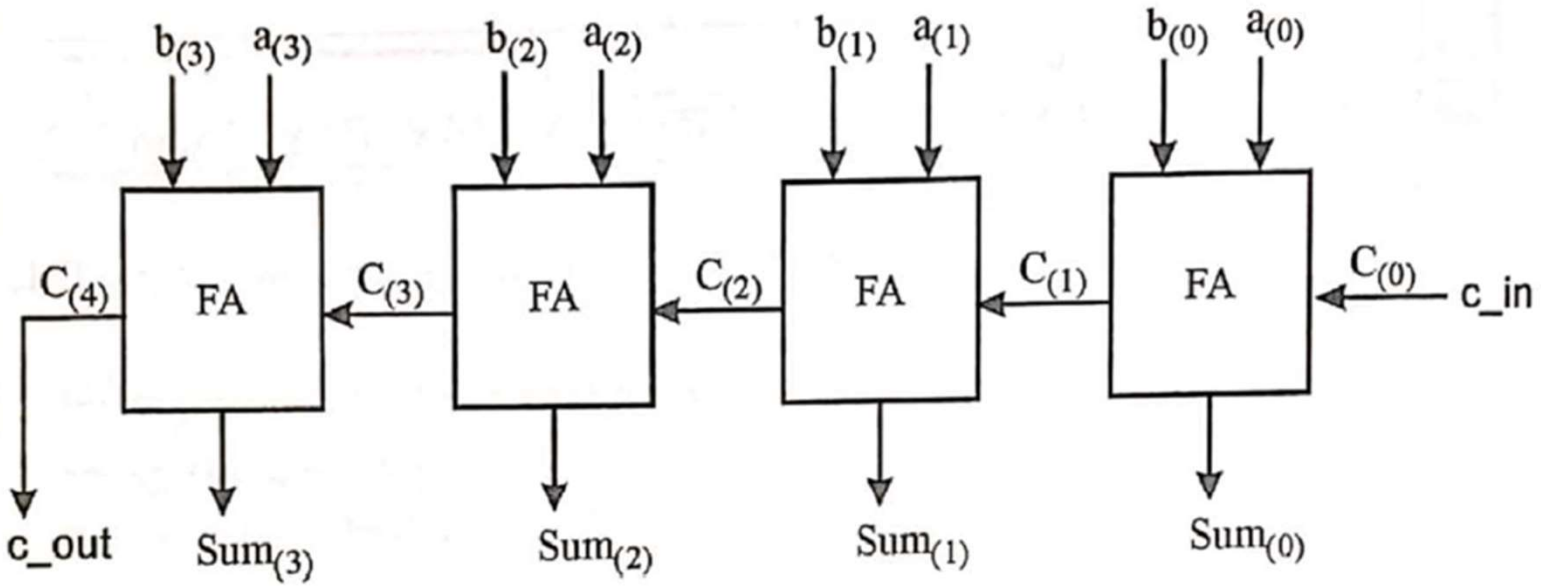
عبارت for ... generate

- عبارت for ... generate برای تکرار یا کپی قسمتی از سخت‌افزار است و به‌صورت همروند (concurrent) اجرا می‌شوند.

```
[lablel یا برچسب ] : for i  in … to … generate
          …
            عبارات همزمان
          …
end generate;
```

for ... generate عبارت

دانشگاه سمنان
Semnan University
پردیس فرزانگان



```
library ieee;
use ieee.std_logic_1164.all;
entity adder_4b is
     port(a,b: in std_logic_vector(3 downto 0);
          c_in : in std_logic;
          sum : out std_logic_vector(3 downto 0);
          c_out : out std_logic; );
end;
architecture beh of adder_4b is
     signal c : std_logic_vector(4 downto 0);
     begin
          c(0) <= c_in;
          gen : for i in 0 to 3 generate
               sum(i) <= a(i) xor b(i) xor c(i);
c(i+1) <= (a(i)and b(i)) or (a(i)and c(i)) or (b(i)and c(i));
          end generate;
          c_out <= c(4);
end;
```


```
architecture Data_flow of adder_4 is
signal c : std_logic_vector(4 downto 0);
begin
     c(0)<=c_in;
     sum(0)<=a(0) xor b(0) xor c(0);
     c(1)<=(a(0)and b(0)) or (a(0)and c(0)) or (b(0)and c(0));

     sum(1)<=a(1) xor b(1) xor c(1);
     c(2)<=(a(1)and b(1)) or (a(1)and c(1)) or (b(1)and c(1));

     sum(2)<=a(2) xor b(2) xor c(2);
     c(3)<=(a(2)and b(2)) or (a(2)and c(2)) or (b(2)and c(2));

     sum(3)<=a(3) xor b(3) xor c(3);
     c(4)<=(a(3)and b(3)) or (a(3)and c(3)) or (b(3)and c(3));

end;
```

یادآوری (عملگرهای VHDL)

Operator	Name	Explanation
=	equivalence	is some value equivalent to some other value?
/ =	non-equivalence	is some value not equivalent to some other value?
<	less than	is some value less than some other value?
<=	less than or equal	is some value less than or equal to some other value?
>	greater than	is some value greater than some other value?
>=	greater than or equal	is some value greater than or equal to some other value?

یادآوری (عملگرهای VHDL)

Operator type							
logical	and	or	nand	nor	xor	xnor	not
relational	=	/=	<	<=	>	>=	
shift	sll	srl	sla	sra	rol	ror	
addition	+	-					
unary	+	-					
multiplying	*	/	mod	rem			
others	**	abs	/				

یادآوری (عملگرهای VHDL)

Operator	Name		Example	Result
logical	sll	shift left logical	result <= "10010101" sll 2	"01010100"
	srl	shift right logical	result <= "10010101" srl 3	"00010010"
arithmetic	sla	shift left arithmetic	result <= "10010101" sla 3	"10101111"
	sra	shift right arithmetic	result <= "10010101" sra 2	"11100101"
rotate	rol	rotate left	result <= "101000" rol 2	"100010"
	ror	rotate right	result <= "101001" ror 2	"011010"

```
ieee.numeric_std
ieee.numeric_bit
```

توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی

- همان‌گونه که یک مدار دیجیتال از چند قطعه (component) یا IC تشکیل می‌شود، طراحی سخت‌افزار با روش ساختاری (structural) نیز از چند قطعه یا component تشکیل می‌شود.
- مدارهای دیجیتال پیچیده را می‌توان به مجموعه‌ای از قطعات یا مدول‌ها به نام مدارهای سطح پایین تقسیم نمود.

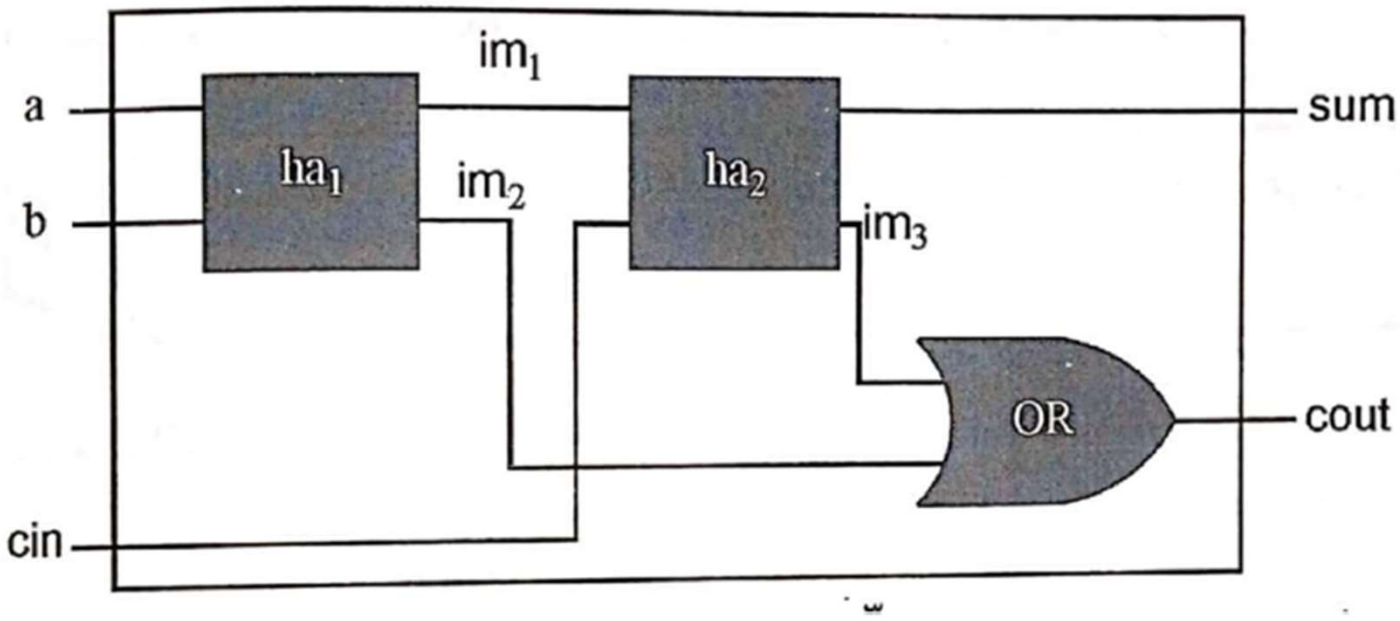
توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی

- طراحی structural را می‌توان به‌صورت تعدادی component که به هم متصل شده‌اند توصیف کرد.
- طراحی به‌صورت structural را در سطح خیلی بالا مانند بلوک دیاگرام می‌توان توصیف نمود و یا در سطح پایین‌تر به‌صورت قطعه‌های دیجیتال یا به‌صورت مجموعه‌ای از گیت‌ها و یا حتی به‌صورت مجموعه‌ای از ترانزیستورها نیز توصیف نمود.

توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی

- اگر توصیف structural یا طراحی سلسله مراتبی در سطح گیت انجام شود، اکثراً به آن netlist می‌گویند که در آن قطعات (components) و طرز اتصال آن‌ها مشخص می‌شود.

توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی



توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی


```
library ieee;
use ieee.std_logic_1164.all;
entity fa_1bit is
     port(
          a: in std_logic;
          b: in std_logic;
          cin : in std_logic;
          sum : out std_logic;
          cout : out std_logic;
          );
end fa_1bit;
```

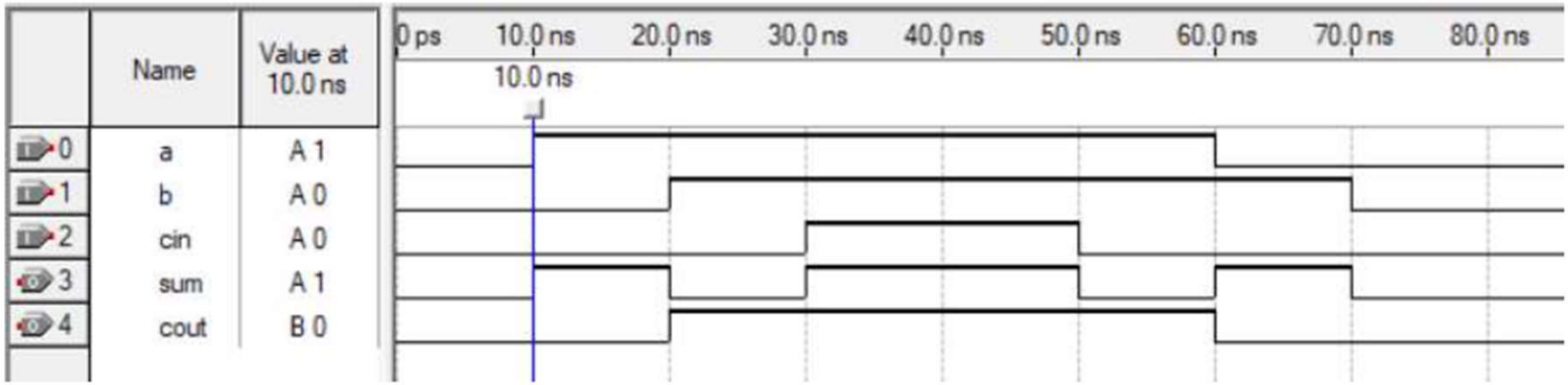
توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی


```
architecture fa_1bit_arch of fa_1bit is
     component ha_1bit port(x,y : in std_logic;
                            s,c : out std_logic);
     end component;
     signal im1,im2,im3 : std_logic;
     begin
          ha1:ha_1bit port map(x=>a,y=>b,s=>im1,c=>im2);
          ha2:ha_1bit port map(x=>im1,y=>cin,s=>sum,c=>im3);
          cout <= im2 or im3;
end fa_1bit_arch ;
```

توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی


```
library ieee;
use ieee.std_logic_1164.all;
entity ha_1bit is
     port(x,y : in std_logic;
          s,c: out std_logic);
end ha_1bit;
architecture gate_level of ha_1bit is
begin
     s <= x xor y;
     c <= x and y;
end gate_level;
```

توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی



توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی

برنامه VHDL یا entity/architecture هر یک از componentها باید در پوشه (folder) برنامه اصلی قرار داده شود و یا در یک package قرار گیرد که با دستور use در ابتدای برنامه از این package استفاده شود.

اعلام قطعه‌ها (Component Declaration)

```
component ha_1bit port(x,y : in std_logic;
          s,c : out std_logic);
end component;
```

نام‌های ورودی خروجی اعلام شده در componentها باید همان نام‌های entity قطعه باشد

port map

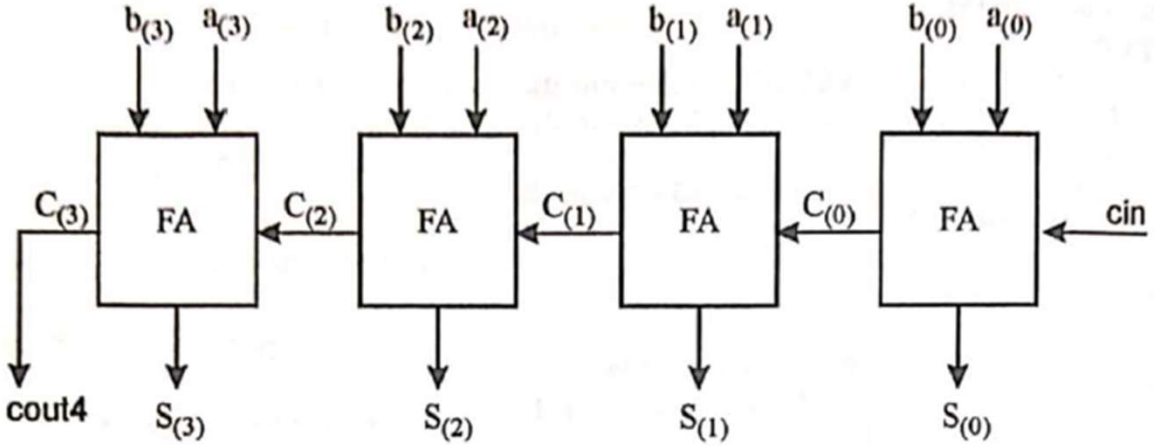
```
Label : component_name port map(port1=>signal1,
port2=>signal2, …);
```

در port map لیست عناصر با کاما (,) از هم جدا می‌شوند.

توصیف یا مدل‌سازی مدارهای دیجیتال به‌صورت ساختاری (Structural) یا طراحی به‌صورت سلسله مراتبی

مثال

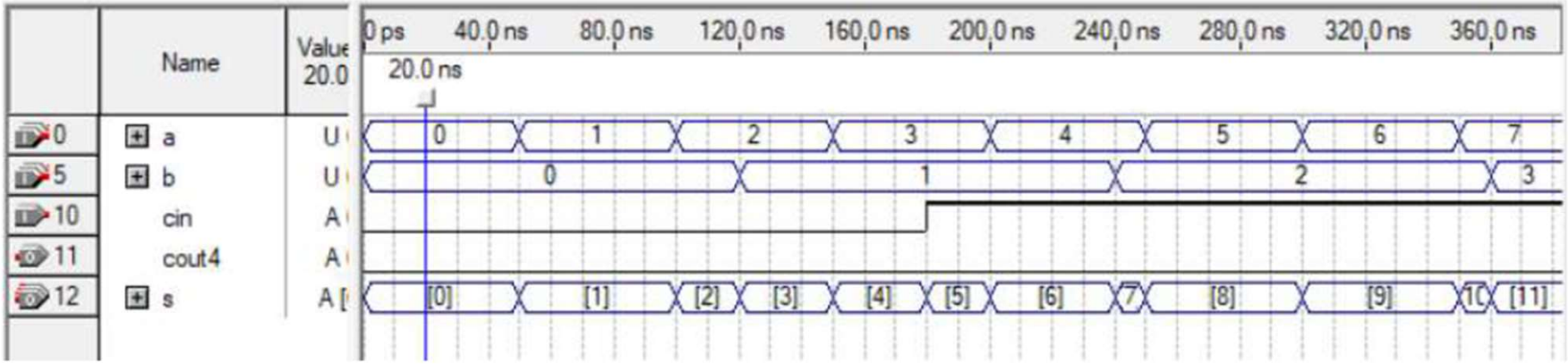
برای یک جمع‌کننده ۴ بیتی، برنامه VHDL به‌صورت ساختاری (structural) با استفاده از component جمع‌کننده کامل (FA) بنویسید.




```
library ieee;
use ieee.std_logic_1164.all;
entity add_4bit is
     port(
          a: in std_logic_vector(3 downto 0);
          b: in std_logic_vector(3 downto 0);
          cin : in std_logic;
          s : out std_logic_vector(3 downto 0);
          cout4 : out std_logic;
          );
end add_4bit;
```

```
architecture add_4bit_arch of add_4bit is
     signal c : std_logic_vector(3 downto 0);
     component full_adder port(a,b,cin : in std_logic;
                sum,cout : out std_logic);
     end component;
     signal im1,im2,im3 : std_logic;
     begin
          U3:full_adder port map(a(3),b(3),c(2),s(3),c(3));
          U2:full_adder port map(a(2),b(2),c(1),s(2),c(2));
          U1:full_adder port map(a(1),b(1),c(0),s(1),c(1));
          U0:full_adder port map(a(0),b(0),cin,s(0),c(0));
          cout4 <= c(3);
end add_4bit_arch;
```


```
library ieee;
use ieee.std_logic_1164.all;
entity full_adder is
     port (a,b,cin : in std_logic;
           sum,cout : out std_logic);
end full_adder;
architecture behavior of full_adder is
begin
     sum <= a xor b xor cin;
     cout <= (a and b)
             or (a and cin)
             or (b and cin);
end;
```


Name
Value
20.0
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
200.0 ns
240.0 ns
280.0 ns
320.0 ns
360.0 ns
20.0 ns
0
a
5
b
10
cin
11
cout4
12
s