دانشکده مهندسی برق و کامپیوتر
گروه مهندسی کامپیوتر

مقدمه‌ای بر زبان توصیف سخت افزار VHDL

توصیف مدارهای ترکیبی ساده

زمستان ۱۴۰۳

مقدمه (تاریخچه)

- نام VHDL شامل دو بخش V و HDL به معنی:

VHSIC : Very **H**igh Speed **I**ntegrated **C**ircuits
HDL : Hardware **D**escription **L**anguage

- استاندارد IEEE 1076-1987
- استاندارد IEEE 1076-1993
- **ABEL** (Advanced Boolean Equation Language)-یک زبان منسوخ شده - برای برای‌های پیاده سازی مدارات کوچک در PLD ها
- Verilog مانند VHDL مورد توجه است
- AHDL زبان اختصاصی شرکت Altera
- SystemVerilog مبتنی بر Verilog برای درستی سنجی در سطح RTL
- SystemC مبتنی بر C++ برای طراحی در سطح سیستم

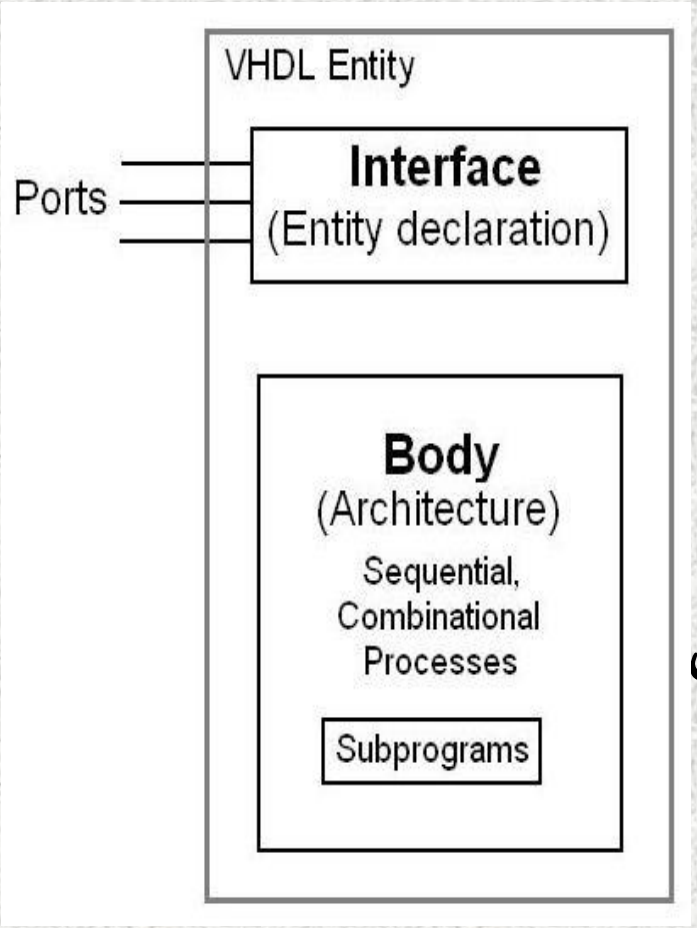
مقدمه (اهداف و نیازمندی ها)

- اهداف اساسی
  - مستند سازی: نگهداری، ارائه، تبادل، استفاده مجدد
  - شبیه سازی: بررسی نتایج و ارزیابی
  - سنتز: با هدف پیاده سازی در FPGA یا بصورت ASIC
- اجرا= شبیه سازی
- حداقل نیازمندی‌ها برای یادگیری زبان VHDL
  - دانستن جبر بول و آشنایی با مدارات منطقی
  - داشتن ابزار CAD مناسب

مقدمه (ویژگی ها)

- همروندی
  - ترتیب دستورات مهم نیست
  - مبتنی بر رخداد
- امکان استفاده از دستورات ترتیبی را نیز دارد
- امکان توصیف طرح بصورت
  - رفتاری (جریان داده- الگوریتمی)
  - ساختاری (با قابلیت سلسله مراتبی)
- امکان مدل کردن تاخیر دروازه‌ها را دارد
- به حروف کوچک و بزرگ حساس نیست
- بشدت نوع گرا است

ساختار کلی یک فایل VHDL



- 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 (عملکردی/ساختاری)

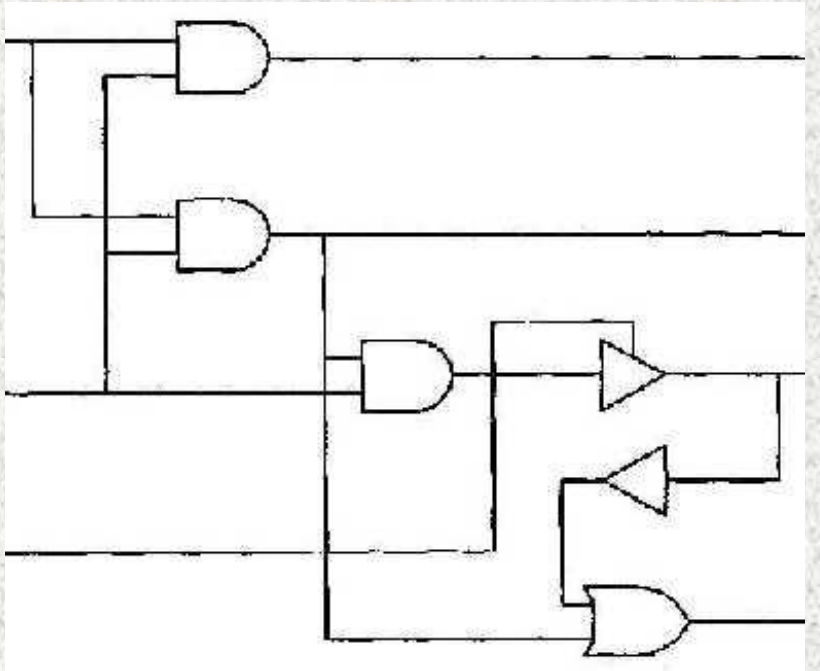
بخش Entity

```
entity NAME_OF_ENTITY is
        port (signal_names: mode type;
              signal_names: mode type;
                    :
              signal_names: mode type);
end [entity] [NAME_OF_ENTITY] ;
```

- **NAME_OF_ENTITY** : یک شناسه اختیاری
- **signal_names** : نام سیگنال‌های ورودی یا خروجی
- **mode** :جهت سیگنال (یکی از موارد in، out، buffer، Inout)
- **Type**: نوع سیگنال (یکی از انواع استاندارد یا تعریف شده توسط کاربر)

mode

- In :سیگنال ورودی
- out : سیگنال خروجی (فقط توسط یک entity دیگر قابل استفاده)
- buffer : خروجی که می تواند در داخل entity نیز استفاده شود
- Inout : سیگنا ل دوطرفه (ورودی / خروجی)



انواع استاندارد

- **Bit : می‌تواند مقدار صفر یا یک داشته باشد.**
- **Bit_vector: برداری از مقادیر بیتی است**

بجای Bit می‌توان از std_logic یا std_ulogic استفاده کرد. بجای Bit_vector نیز می‌توان از std_logic_vector یا std_ulogic_vector استفاده کرد. در این انواع برای هر بیت ۹ مقدار متفاوت تعریف شده است که در جای خود شرح داده خواهد شد.

- **Boolean: می‌تواند مقدار true یا false بگیرد.**
- **Integer: یک عدد در محدوده‌ای از اعداد صحیح است.**
- **real: می‌تواند یک عدد در محدوده‌ای از اعداد حقیقی را نگهداری کند.**
- **Character: هر کاراکتر قابل چاپ**
- **Time: برای نمایش زمان بکار می‌رود.**

مثال (۳–۱) تعریف entity برای یک نیم جمع کننده

```
entity HA is
port ( x, y: in bit;
       s, c: out bit );
end halfadder;
```

- port برای مشخص کردن ارتباطات بین entity و دنیای خارج
  - سیگنال‌های x و y از نوع bit و بعنوان ورودی‌های مدار
  - سیگنال‌های s و c از نوع bit و بعنوان خروجی‌های مدار

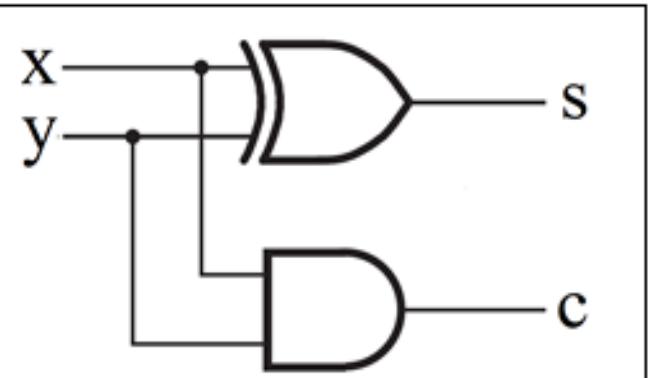
بخش Architecture

- مشخص می‌کند:
  - که عملکرد مدار چیست؟ (توصیف رفتاری)
  - یا چگونه ساخته می‌شود؟ (توصیف ساختاری)

```
architecture architecture_name of NAME_OF_ENTITY is
   -- Declarations
      -- components declarations
      -- signal declarations
      -- constant declarations
      -- function declarations
      -- procedure declarations
      -- type declarations
         …
begin
      -- Statements
         …
end architecture_name;
```

مثال (۳–۲) تعریف architecture نیم‌جمع کننده

```
architecture behavior of HA is
begin
    s <= x xor y;
    c <= x and y;
end behavior;
```

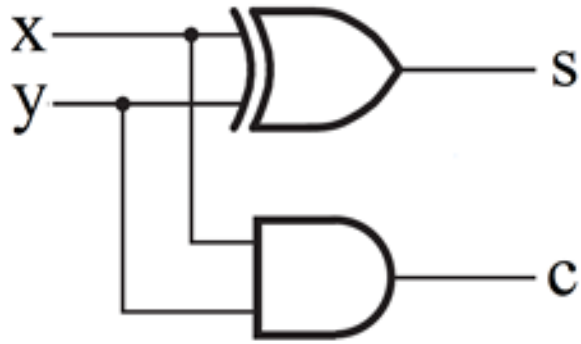


- شامل دو دستور انتساب هم‌روند
  - دستور انتساب اول معادل دروازه xor
  - دستور انتساب دوم معادل دروازه and
- xor و and دو عملگر منطقی هستند

تعریف کامل نیم‌جمع کننده

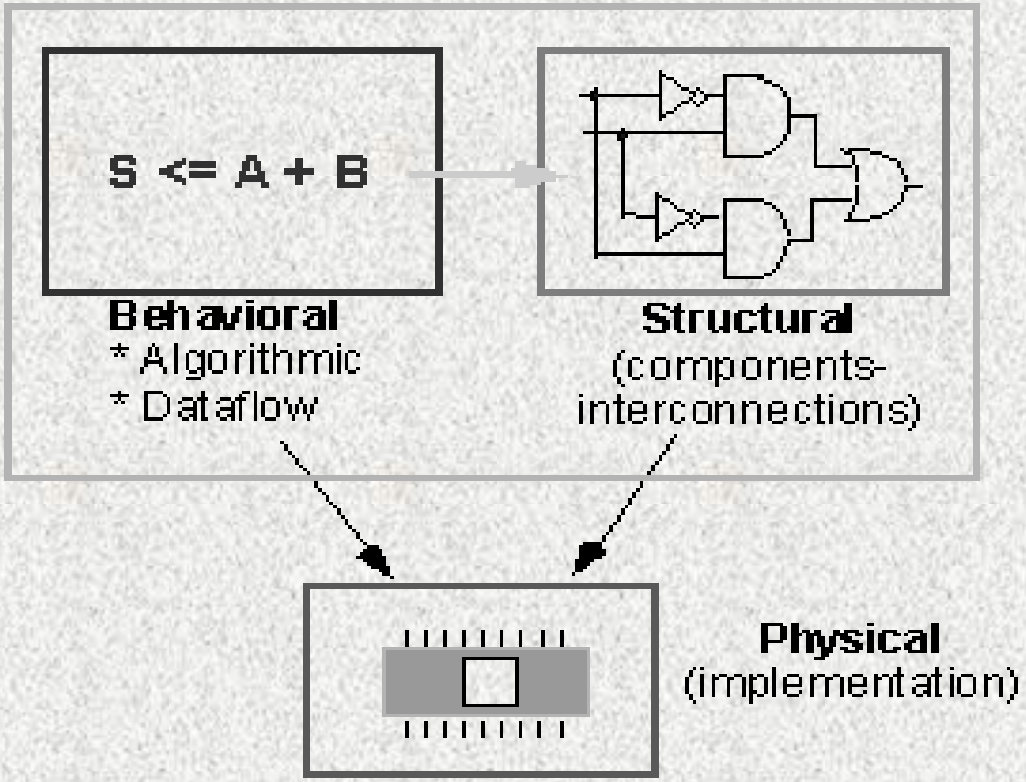
```
entity HA is
port ( x, y: in bit;
       s, c: out bit );
end HA;
```


```
architecture behavior of HA is
begin
    s <= x xor y;
    c <= x and y;
end behavior;
```



توصیف رفتاری و ساختاری

- سه حوزه رفتاری، ساختاری و هندسی (فیزیکی) داریم
- VHDL : امکان توصیف طرح بصورت رفتاری و ساختاری



توصیف رفتاری

- توصیف رفتاری:
  - رفتار سیستم چیست؟
  - رابطه بین سیگنال‌های ورودی و خروجی
- انواع توصیف رفتاری:
  - جریان داده (Data Flow) : براساس دستورات انتساب همروند
  - الگوریتمی (Algorithmic): با استفاده پردازه (process) و از دستورات ترتیبی
- سطح توصیف:
  - سطح دروازه
  - سطح انتقال ثبات

مدل رفتاری (Behavioral model)

- دستورات انتساب همروند
  - دستور انتساب همروند ساده
  - دستور انتساب همروند شرطی (when)
  - دستور انتساب همروند انتخابی (with)
- فرمت کلی دستور انتساب سیگنال هم‌روند ساده:
  - عبارت expression به سیگنال target_signal منتقل می‌شود
  - target_signal باید هم نوع عبارت expression باشد.

```
Target_signal <= expression;
```

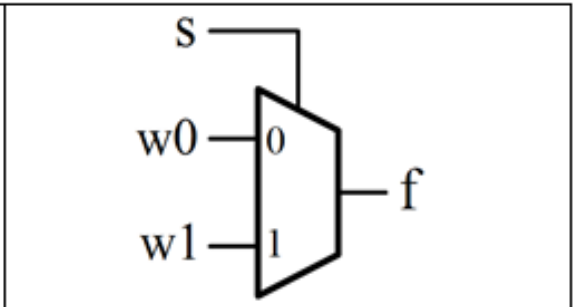
توصیف VHDL مدارهای ترکیبی ساده

- منظور مداراتی است که
  - شامل دروازه‌های منطقی
  - سیگنال‌های ورودی و خروجی آنها از نوع **bit** (یا انواع مشابه)
  - توصیف عملکرد با دستورات انتساب شامل عملگرهای منطقی
  - عملگرها شامل **AND**، **OR**، **NOT**، **NAND**، **NOR**، **XOR**، **XNOR**

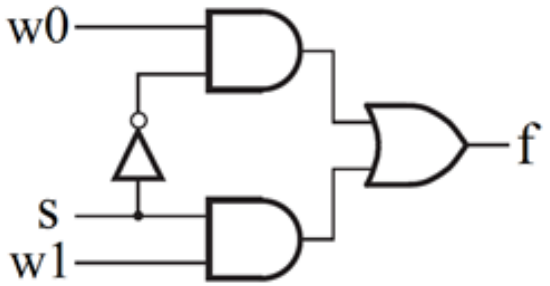
مثال ۳-۳) توصیف یک انتخاب‌کننده ۲ به ۱

- عملگر NOT اولویت بالاتری دارد
- بقیه عملگرهای منطقی در یک سطح هستند
- به نحوه استفاده از پرانتز توجه شود

```
ENTITY Mux21 IS
PORT (s, w0, w1 : IN BIT ;
                  f : OUT BIT );
END Mux21;
```



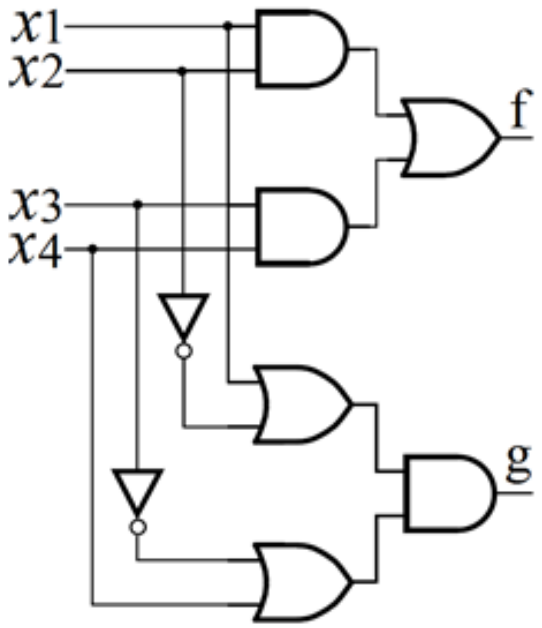

```
ARCHITECTURE behavior OF Mux21 IS
BEGIN
    f <= (NOT s AND w0) OR (s AND w1);
END behavior;
```



مثال ۳–۴) توصیف یک مدار شامل دو تابع

- توصیف تابع f بصورت جمع جملات ضربی(SOP)
- توصیف تابع g بصورت ضرب جملات جمعی (POS)
- به نحوه استفاده از پرانتز توجه شود

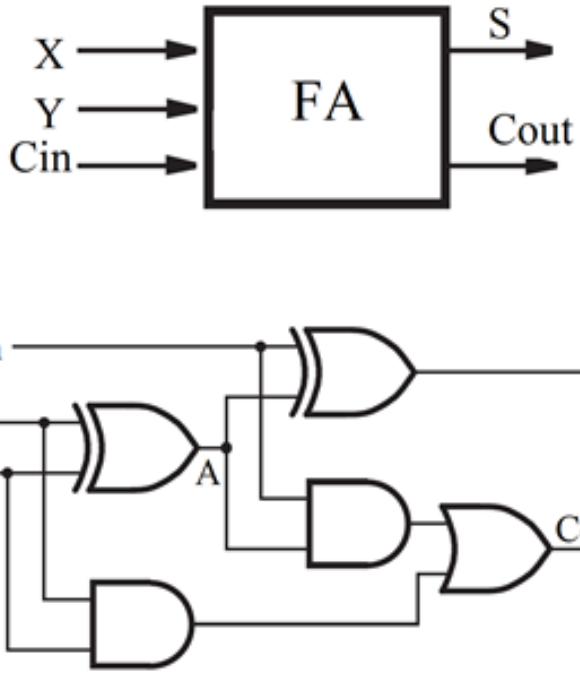
```
ENTITY Combin IS
PORT ( x1, x2, x3, x4: IN BIT;
                  f, g : OUT BIT );
END Combin;
-------------------------------------
ARCHITECTURE LogicFunc OF Combin IS
BEGIN
   f <= (x1 AND x3) OR (x2 AND x4);
   g <= (x1 OR NOT x3) AND (NOT x2 OR x4);
END LogicFunc;
```



مثال ۳–۵) تعریف سیگنال داخلی

- تعریف بعضی نقاط میانی مدار بعنوان سیگنال داخلی
- به نحوه تعریف سیگنال A توجه شود
- بدون سیگنال داخلی مجبوریم عبارت X xor Y را دو بار تکرار کنیم

```
ENTITY FA IS
PORT ( X, Y, Cin: IN BIT;
       S, Cout : OUT BIT );
END FA;
--------------------------------
ARCHITECTURE behav OF FA IS
  Signal A: BIT;
BEGIN
  A <= X xor Y;
  S <= Cin xor A;
  Cout <= (Cin AND A) OR (X AND Y);
END behav;
```



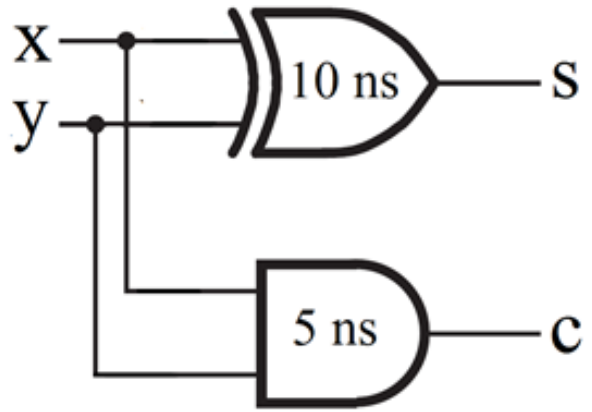
هم روندی

- VHDL یک زبان توصیف هم‌روند است
  - ترتیب نوشتن جملات همروند مهم نیست
  - چون اجرا بر اساس ترتیب نوشتن نیست
  - می‌توان ترتیب جملات انتساب همروند را جابجا کرد
  - همه دستورات انتساب همروند سیگنال از عملگر <=
- VHDL یک زبان مبتنی بر رخداد
  - در صورت بروز رخداد در سمت راست جمله اجرا می‌شود
  - رخداد به تغییر در مقدار یک سیگنال گفته می‌شود

زمان‌بندی رخدادها با استفاده از تأخیر

- با استفاده از کلمه کلیدی after
- مدل کردن تأخیرهای یک مدار واقعی
- شبیه‌سازی مدار را به واقعیت نزدیک‌تر می‌کند
- در سنتز مدار نادیده گرفته می‌شود

```
Entity halfadder is
port ( x, y  : in bit;
       s, c : out bit );
end halfadder;
------------------------------
architecture behav of halfadder is
begin
   s <= x xor y after 10 ns;
   c <= x and y after 5 ns;
end behav;
```



استفاده از نوع std_logic در تعریف سیگنال

- برای نوع BITفقط مقدار '0' و '1' تعریف شده است
- نوع STD_LOGIC می‌توانند ۹ مقدار متفاوت
- برای عملیات سنتز تنها چهار مقدار '0' ، '1' ، 'z' و '-'لازم است

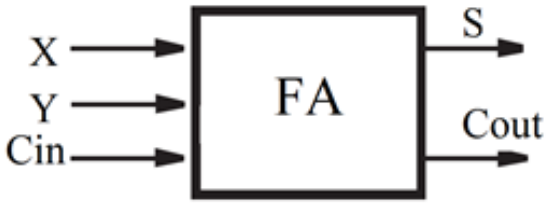
جدول ۳-۱) مقادیر مختلف نوع std_logic

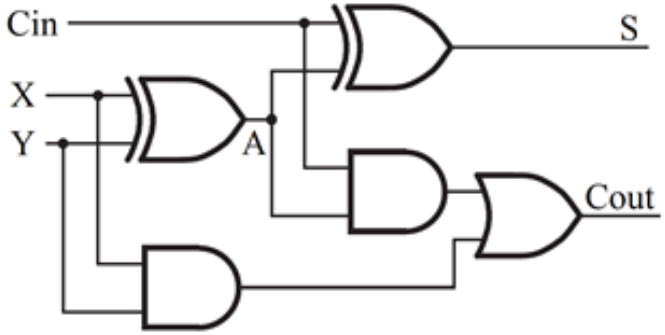
'U'	مقدار اولیه داده نشده	uninitialized
'X'	مقدار ناشناخته	forcing unknown
'0'	صفر	forcing 0
'1'	یک	forcing 1
'Z'	امپدانس بالا	high impedance
'W'	مقدار ناشناخته ضعیف	weak unknown
'L'	صفر ضعیف	weak 0
'H'	یک ضعیف	weak 1
'-'	مقدار داده نشده	don’t care

مثال ۳–۷) استفاده از std_logic در تمام جمع کننده

- معرفی std_logic_1164 از کتابخانه IEEE قبل از Entity ضروری است

```
library ieee;
use IEEE.std_logic_1164.all;
---------------------------------
ENTITY FA IS
PORT ( X, Y, Cin: IN STD_LOGIC;
       S, Cout : OUT STD_LOGIC);
END FA;
---------------------------------
ARCHITECTURE behav OF FA IS
  Signal A: STD_LOGIC;
BEGIN
  A <= X xor Y;
  S <= Cin xor A;
  Cout <= (Cin AND A) OR (X AND Y);
END behav;
```





دستور انتساب سیگنال شرطی

- سیگنال Target_signal مقدار اولین expression که شرط آن درست است
- اگر هیچ شرطی درست نباشد مقدار آخرین expression که بدون شرط ذکر شده

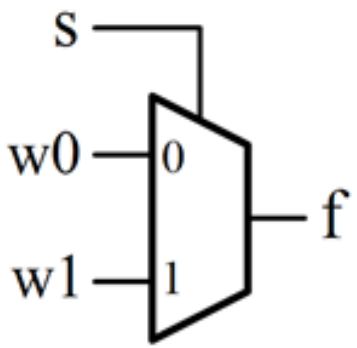
```
Target_signal <= expression WHEN Boolean_condition ELSE
                 expression WHEN Boolean_condition ELSE
                  …
                 expression;
```

- دو نکته مهم:
  - همروند بودن دستور انتساب شرطی
  - هم رخداد در expression و هم رخداد در شرط باعث ارزیابی مجدد می‌شود

مثال ۳–۸) مالتی پلکسر ۲ به ۱

- این مثال از نظر رفتاری معادل مثال ۳-۳ است.
- در مثال ۳-۳ از دستور انتساب ساده استفاده شده بود.
- در اینجا از دستور انتساب شرطی استفاده شده است

```
LIBRARY ieee;
USE IEEE.std_logic_1164.ALL;
-----------------------------------
ENTITY Mux21 IS
PORT ( s, w0, w1 : IN std_logic;
                f : OUT std_logic );
END Mux21;
-----------------------------------
ARCHITECTURE behav OF Mux21 IS
BEGIN
   f <= w0 WHEN s='0' ELSE w1;
END behav;
```



تابع رزولوشن نوع std_logic

- برای نوع std_ulogic هیچ تابع رزولوشنی تعریف نشده
- نوع std_logic دارای یک تابع رزولوشن به شکل زیر است:

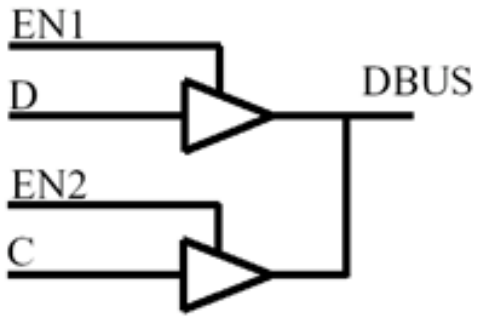
جدول ۳-۲) عملکرد تابع رزولوشن نوع std_logic

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

مثال ۳–۹) اتصال خروجی دو بافر سه حالته

- برای جلوگیری از تداخل باید در هر لحظه فقط فعال‌ساز خروجی یکی از آنها فعال باشد.

```
LIBRARY ieee;
USE IEEE.std_logic_1164.ALL;
-------------------------------------
ENTITY ResBus IS
PORT (D, C, EN1, EN2: IN std_logic;
                DBUS: OUT std_logic);
END ResBus;
-------------------------------------
ARCHITECTURE behav OF ResBus IS
BEGIN
   DBUS <= D WHEN EN1='1' ELSE 'Z';
   DBUS <= C WHEN EN2='1' ELSE 'Z';
END behav;
```



تمرین

تمرین ۳-۴) با استفاده از یک بافر سه حالته می‌توان درگاه‌های دوطرفه (ورودی خروجی) ایجاد کرد. در مدار زیر که کد Entityی آن نوشته شده یک خروجی دو طرفه و یک خروجی از نوع بافر وجود دارد. توصیف رفتاری آن را با استفاده از دستورات انتساب همروند بنویسید.

```
ENTITY Bidir IS
PORT ( A: IN STD_LOGIC;
       B: IN STD_LOGIC;
       C: IN STD_LOGIC;
       D: OUT STD_LOGIC
       E: BUFFER STD_LOGIC
       F: OUT STD_LOGIC
       G: INOUT STD_LOGIC );
END Bidir;
```