به نام خدا

آموزش VHDL

(مقدمه)

Very High Speed Integrated Circuit Hardware Description Language

مقدمه ای بر VHDL

زبان توصیف سخت افزار می تواند فعالیت عمده خود را روی وسایل منطقی قابل برنامه ریزی (Programming Logic Devices: PLDs) با پیچیده تر (CPLDs) و (Field Programmable Gate Arrays: FPGAs) داشته باشد.
چند گونه از این زبان وجود دارد مانند VHDL,Verilog,Abel که ما در این بخش آموزشی به VHDL خواهیم پرداخت.
نرم افزارهای بسیاری برای طراحی VHDL ساخته شده که یکی از آنها Active-VHDL از سوی شرکت ALDEC می باشد که یادگیری آن بسیار ساده است.
گر چه VHDL توسط وزارت دفاع آمریکا به منظور اهداف نظامی تولید شد ولی به سرعت گسترش یافت و با یک استاندارد خاص (the Institute of Electrical and Electronic Engineers: IEEE)مورد استفاده عموم قرار گرفت که IEEE به مرور از سال ۱۹۸۷ تا ۱۹۹۸ کامل شد.

زبان VHDL ابتدا به منظور شبیه سازی و مدل سازی و درک بیشتر مدارهای منطقی بوده است که توسط محققان عمل Synthesis یا سنتز به عنوان اتوماتیک کردن فرایند طراحی به آن اضافه شده است.

فاکتورهای قابل ملاحظه در ارزش VHDL برای یک طراح :

۱- محبوبیت جهانی زبان HDL (زبان توصیف سخت افزار)
زبان VHDL در شمال امریکا و همچنین اروپا توسط ۸۰ درصد مهندسین سیستم استفاده میشود که این رقم همچنان در حال رشد است.

۲- وجود انواع مختلف روشهای توصیف در این زبان
VHDL برای کاربر از نظر یکتا بودن نوع توصیف هیچ گونه محدودیتی ندارد در واقع یک برنامه را می توان هم به صورت رفتاری و هم به صورت متنی یعنی ذکر تمام گیتهای موجود به کار برد همچنین از VHDL می توان در سطوح مختلفی از پیچیدگی استفاده کرد از یک ترانزیستور کوچک گرفته تا یک سیستم کامل را می توان با آن طراحی کرد.

۳-وجود نرم افزارهای شبیه ساز VHDL
یکی از دلایل افزایش محبوبیت این زبان وجود تعدادی از نرم افزارهای شبیه ساز با قیمتی مناسب مانند Active VHDL است که می توانید با مراجعه به سایت www.aldec.com در مورد آن بیشتر اطلاعات کسب کنید.

۴- در اختیار داشتن سنتز کننده های VHDL

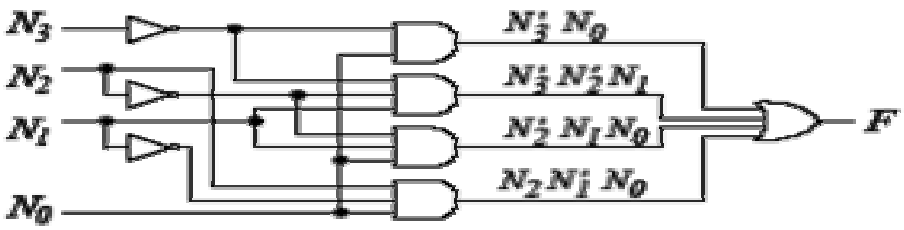
امروزه بسیاری از فروشندگان وسایل قابل برنامه ریزی (Programmable Device) و شرکاتهای تولید کننده نرم افزارهای EDA سنتز کننده های گوناگونی را ارائه داده اندمانند Active-CAD که یک سنتز کننده ی منطقی برای تمام مدارات مجتمع FPGA و CPLD می باشد.

۵-VHDL یک زبان مدل سازی جهانی

گر چه VHDL بیشتر ما را متوجه الکترونیک می کند در واقع به یک زبان مدل سازی جهانی شهرت یافته است و برای مدل سازی و شبیه سازی سیستمهای الکترو مکانیکی و هیدرولیکی و شیمیایی و ... مورد استفاده قرار می گیرد.

چند نکته مهم در مورد کد نویسی در VHDL

۱-زبان VHDL يك زبان حساس به حروف نيست يعني حروف بزرگ و كوچك را بدون تفاوت در نظر ميگيرد مثلا كلمات زير هيچ فرقي با هم ندارند:Architecture , ARCHITECTURE , architecture
۲-نقطه شروع ترجمه ي كدهاي شما در اول هر خط از دستورات مي باشد يا به عبارتي ديگر اجراي دستورات در VHDL مانند CPP ترتيبي نيست يعني دستورات يكباره و با هم اجرا مي شوند.
اکنون به برنامه زیر که به زبان VHDL نوشته شده و دیاگرام مقابل را توصیف می کند توجه کنید شما باید دیاگرام و کدهای VHDL را دقیقا دنبال کنید تا یک دید کلی در مورد روش و نمای آن بدست آورید:



```
Library IEEE;
Use IEEE.std_logic_1164.all;
Entity prime is
Port (N0, N1, N2, N3: in BIT ;
F: out BIT);
End prime;
Architecture prime1_arch of prime is
Signal and1, and2, and3, and4: bit;
Begin
and1<= (not n3) and n0;
and2<= (not n3) and (not n2) and n1;
and3<= (not n2) and n1 and n0;
and4<=  n2 and (not n1) and n0;
f   <= and1 or and2 or and3 or and4;
End prime1_arch;
```

(قسمت اول)

(Part 1)

نحوه تعریف ورودی و خروجی

با یاد و نام خدا اولین درس از VHDL را آغاز می کنیم قبل از شروع درس توجه به نکات زیر الزامی است:

شما چه هدفی را از آموختن این زبان دنبال می کنید؟ آیا تفریحی آن را یاد میگیرید؟ یا نه شاید اهداف مهمتری را دنبال می کنید؟

به نظر من این زبان ساده ترین زبان برای برنامه نویسی سخت افزاری می باشد.با وجود تکنولوژی جدید FPGA شما بیشتر به این نوع زبان احتیاج خواهید داشت . FPGA چیست؟ تراشه هایی هستند شبیه به IC با پایه هایی که چهار طرف آن را گرفته است.شما می توانید برنامه های پیچیده ای را روی FPGA قرار داده و حد اکثر استفاده را با هزینه ای کم و صرف وقت کم بدست آورید.ابتدا باید زبان VHDL را خوب بیاموزیم تا بتوانیم وارد وادی FPGA شویم تا به نحو احسن از آن سود ببریم.(ناگفته نماند که زبانهای دیگری هم برای Program کردن تراشه های FPGA وجود دارد که من VHDL را بهتر می پسندم!)

در آینده ای نزدیک قصد دارم مقاله ای توصیفی در مورد تراشه های قابل برنامه ریزی ارائه دهم.

به امید خدا.

توجه:این بخش آموزشي بر اساس کامپایلر و محیط نرم افزار Active-VHDL ارائه شده است.

Entity-۱

اکنون وقت آن رسیده که با اولین دستور آشنا شوید : **Entity**

همیشه هر سیستمی در طراحی VHDL با Entity آغاز خواهد شد.

مرکز اصلی برنامه شما یا به عبارت دیگر همان Main شما بین دو عبارت Entity و end Entity; تعریف می شود.نام سیستمی که شما قصد طراحی آن را دارید مقابل کلمه ی Entity قرار می گیرد مثلا اگر می خواهید یک Processor طراحی کنید باید بنویسید:

```
Entity Processor is
--این قسمت محل تعریف ورودی ها و خروجی های برنامه شماست
End entity;
```

حالا باید ببینیم منظور از ورودی و خروجی که بعد از Entity تعریف می شود چیست؟

۱-پارامتر ها : مانند پهنای یک گذر گاه یا Bus برای یک پردازنده یا ماکزمیم مقدار فرکانس برای clock سیستم.

۲-راه های ارتباطی از خارج به سیستم و از داخل سیستم به خارج که وظیفه انتقال داده ها را بر عهده دارند.

با مطا لعه ی مثال زیر که یک ثبات ۸ بیتی را توصیف می کند دو مطلب بالا را بهتر درک خواهید کرد:


```
Entity eight_bit_register is
پارامترها:
طول = ۸
ماکسیمم فرکانس=۵۰ مگاهرتز
راه های ارتباطی با ثبات:
D_in ورودی ۸بیت
D_out خروجی ۸بیت
CLK یک بیت ورودی
End entity eight_bit_register;
```

نمونه دیگر :

توجه:پارامترها و را هها‌ی ارتباطی با ثبات گفته شده در این مثال فقط جنبه ی سمبولیک دارند و صرفا برای درک بیشتر آورده شده اند و نمونه کد VHDL نیستند.

```
Entity eight_bit_register is
Generic ( length=8
          Fmax =50 MHz
        );
Port     ( D_IN    eight-bit input
           D_OUT eight-bit output
           CLK      one-bit input
         );
End entity eight_bit_register;
```

پارامتر ها در زبان VHDL بعد از کلمه ی Generic می آیند و ورودی خروجی ها بعد از کلمه ی Port خواهند آمد که در ادامه بیشتر در مورد این دو قسمت بحث می کنیم قبلا کمی در مورد Architecture صحبت می کنیم:

Architecture-۲

مثال:وسایل پیشرفته الکترونیکی مانند تلویزیون گاهی کمتر استفاده میشوند زیرا اگر وسیله ای برای کنترل کردن و ارتباط برقرار کردن با آنها در اختیار نداشته باشیم.این وسیله قسمت مهمی در یک طراحی VHDL می باشد مثلا در TV همان Remote Control را فرض می کنیم Architecture هم دقیقا کار آن را انجام می دهد که به صورت زیر تعریف می شود:

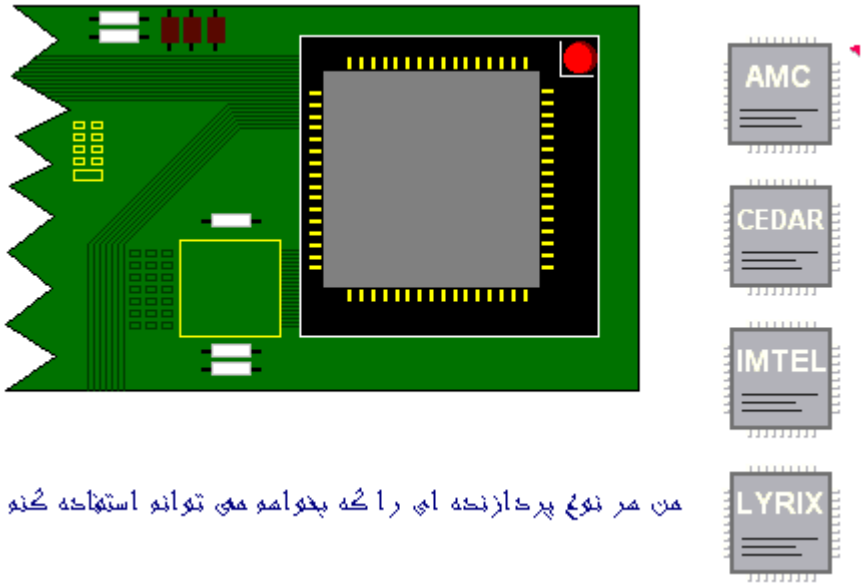
```
Entity Tvset is

...

End entity Tvset;

Architecture TV2000 of Tvset is

...

End Architecture TV2000;
```

دستوراتی که ما می توانیم در قسمت Architecture قرار دهیم ۲ نوع هستند:

1-**Functionality (Behavioral)** رفتاری
2-**Structural** متنی و ساختمانی

منظور از رفتاری این است که دستورات ما فقط رفتار سیستم را توصیف کنند و به انواع گیتهای به کار گرفته شده و ریزه کاري ها در سیستم کاری نداریم. سنتز کردن اینگونه برنامه ها توسط نرم افزارهای سنتز کننده دشوار خواهد بود ولی درک و فهم برنامه ای که ما به این روش نوشته ایم بسیار آسان است.
منظور از ساختمانی یعنی دستورات ما از نظر مدار های منطقی کامل باشند و تمام گیتهای موجود در سیستم را توصیف می کنیم که به این یک برنامه نویسی کامل گفته می شود.سنتز کردن اینگونه برنامه ها آسان می باشد ولی درک انها توسط ما کاری دشوار خواهد بود.

هر سیستمی که ما قصد طراحی آن را داریم فقط یک Entity دارد با چند Architecture .
عکس این مطلب درست نیست یعنی برای یک Architecture نمی توانیم تعدادی Entity داشته باشیم.
به مثال زیر توجه کنید:
اگر مینبوردی(MainBoard) که با پردازنده های مختلف کار میکند را به عنوان یک برنامه VHDL در نظر بگیرید خواهید دید که مینبورد ما همان قسمت Entity برنامه است چون فقط یک مینبورد داریم پس فقط یک Entity هم داریم و پردازنده های مختلف همان Architecture های برنامه ما می باشند مثلا برای CPU های AMD,Intel,Cyrix خوهیم داشت:




```
Entity pentium is
...
End entity pentium;
Architecture Amd of pentium is
..
End Amd;
Architecture Intel of pentium is
..
End Intel;
Architecture Cyrix of pentium is
..
End Cyrix;
```

با داشتن یک چنین برنامه ای هر زمان به هر نوع پردازنده ای که احتیاج داشتیم می توانیم آن را روی مینبورد وصل کنیم.
همه دستوراتی که می توان در Architecture استفاده کرد در کتابخانه ای به نام IEEE وجود دارد با مراجعه به آن مي توان ابهامات احتمالی را رفع کرد.اگر به لفظی غیر استاندارد و نا مفهوم برخوردید ابتدا Library را صدا زده سپس در کتابهای موجود در آن به دنبال کلمه مورد نظر بگردید.بسته بندی ها یا Packages که در Library وجود دارند به سه دسته تقسیم می شوند:

۱-Standard
این کتابخانه به صورت Default در نظر گرفته می شود و اگر در ابتدای برنامه نام کتابخانه مورد نظر ذکر نشد کامپایلر به طور پیش فرض آن را Standard در نظر می گیرد که شامل تمام انواع استاندارد عملگرها و اشیا می باشد.

۲-Textio
این Library فقط به منظور شبیه سازی و مدل سازی با VHDL به کار می رود و به عنوان برنامه ای که بتواند روی مدارات مجتمع پیاده سازی شود نیست و بدین صورت در ابتدای برنامه نوشته می شود:

```
Library Std;
Use Std.TextIO.all;
```

۳-STD_LOGIC_1164
این مهمترین و پرکاربرد ترین کتابخانه است و همیشه به عنوان مرجع استفاده می شود نمایش آن بدین صورت است:

```
Library IEEE;
Use IEEE.Std_Logic_1164.all;
```

اگر دقیقا پس از خواندن مطالب گفته شده در بالا ، درس بعدی را بخوانید خیلی بهتر است...

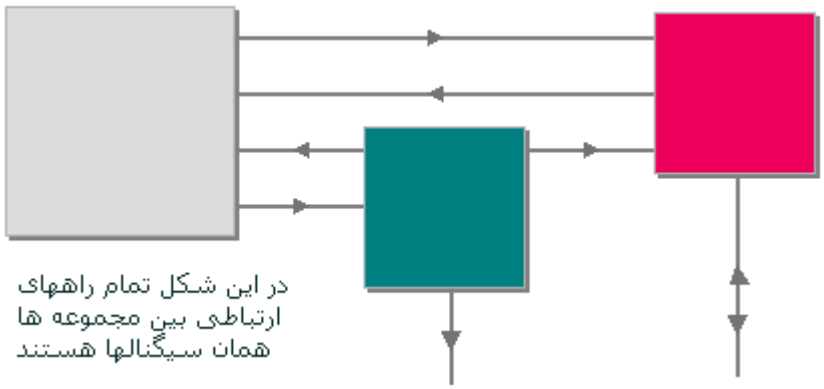
درس بعد راجع به طرز تعریف سیگنالها و پورتها خواهد بود.

(قسمت دوم)

نحوه تعریف انواع سیگنالها و پورتها

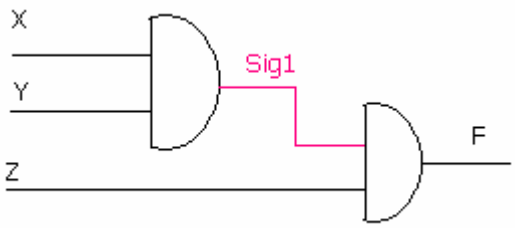
Signal-۲

شاید کسانی که درس مدار منطقی را گذراندند تعریف سیگنال را بدانند به هر حال در شکل زیر کاملا واضح است:



به عبارت ساده تر اگر ما دو گیت مثلا AND داشته باشیم که خروجی یکی ، ورودی دیگری باشد آنگاه تکلیف داده ای که از اولین AND خارج می شود چیست بله! این داده را باید به یک متغیر که به صورت سیگنال تعریف شده است نسبت دهیم.(در ضمن عملگر نسبت دادن این است :" a<=b " یعنی مقدار b را درون a قرار بده ، که در مثال زیر واضح تر است:

```
Entity test1 is
Port ( x , y , z :in bit ; f :out bit );
End entity;
Architecture t1 of test1 is
Signal sig1 : bit ;
Begin
Sig1<=x and y ;
F<=sig1 and z ;
End t1;
```

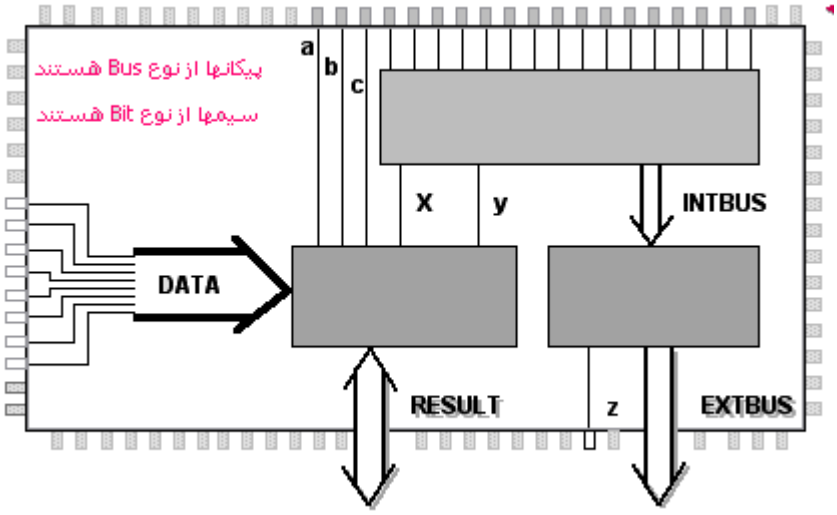


٤-انواع سیگنالها

به ورودی و خروجیها هم می توان سیگنال گفت اما انواع مختلفی هستند

١-سیم یا Wire
٢-گذرگاه یا Bus

که هر کدام از آنها در VHDL با نام مخصوصی عنوان می شود مثلا سیمها همیشه حامل یک بیت داده هستند و Bus ها حامل چند بیت.
که در VHDL هر متغیری را که می خواهیم از نوع سیم یا wire تعریف کنیم باید آن را از نوع Bit در نظر بگیریم همچنین برای متغیرهای از نوع گذرگاه یا Bus باید بنویسیم Bit_Vector مانند شکل زیر:



مثلا برای تعریف یک ورودی به نام Data از نوع Bus که حامل ٨ بیت داده باشد و یک ورودی به نام C که از نوع بیت است داریم:

```
Entity test2 is
Port(data : in bit_vector(7 downto 0) ; c : in bit);
End entity;
```

نکته: ورودی Data عددی است ٨ بیتی که اولین رقم از سمت راست معادل باینری آن دارای اندیس صفر می باشد اگر بخواهیم اندیسها از چپ به راست شروع شوند باید اینگونه بنویسیم:

```
Entity test2 is
Port(data : in bit_vector(0 to 7) ; c : in bit);
End entity;
```

یادآوری مهم: همیشه مکان تعریف ورودی و خروجی ها در Entity و قسمت Port میباشد و Signal ها همیشه در Architecture و قبل از Begin تعریف می شوند.مثلا در شکل قبل فقط آن پیکانی که بین دو مستطیل قرار گرفته و مقابلش عبارت INTBUS نوشته شده از نوع Signal است این ادعا در مورد دو Wire به نامهای X و Y هم صادق است.

به مثال زیر که در مورد شکل قبل طرح شده توجه فرمایید:

```
Entity Test3 is
Port(a , b , c : in bit ; Data : in Bit_Vector(7 downto 0) ;
      Z : in bit ; EXTBUS : out Bit_Vector(4 downto 0) ;
      RESULT : inout Bit_Vector (0 to 7) );
End entity;
Architecture RTL of Test3 is
Signal : x , y : bit ;
Signal : INTBUS : bit_vector(4 downto 0);
Begin
End RTL;
```

IN : داده از نوع ورودی می باشد.
OUT : داده از نوع خروجی می باشد.
INOUT : داده می تواند هم خروجی و هم ورودی باشد.
درس اول را با ذکر دو نکته زیر به پایان می بریم:

۱-اسامی که شما برای Entity وَArchitecture هایتان انتخاب می کنید حتما باید با حروف الفبا آغاز شوند.

۲-اگر در میان خطوط برنامه خواستید توضیحات (Comment) اضافه کنید قبل از آن باید دو خط فاصله (همان علامت تفریق) قرار دهیم .

امید وارم این درس بدون هیچ ابهامی از نظر شما گذشته باشد.
تا درس آینده ...

(قسمت سوم)

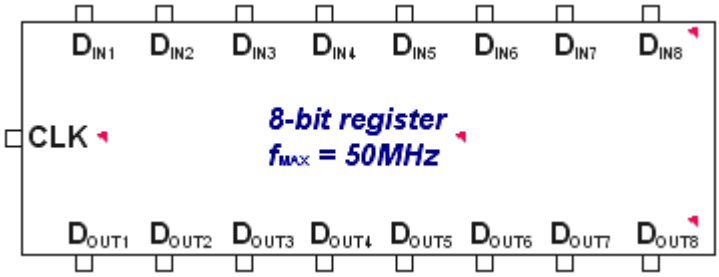
در این درس به نحوه تعریف متغیر و کلمه Generic و ارائه یک مثال حل شده و تمرین می پردازیم.

Generic-۵

این کلمه ای است رزرو شده و قبل از تعریف پورت یعنی در Entity استفاده می شود .از این کلمه به منظور مقدار دادن به یک متغیر استفاده می شود که هر تعریف به صورت Generic شامل دستورات زیر خواهد بود:

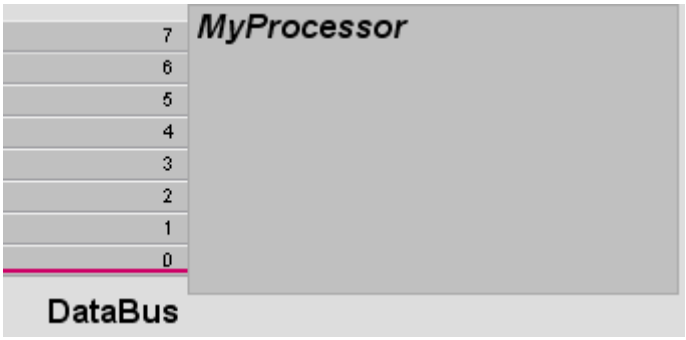
- نام Generic و پس از آن یک کولن ":" (دونقطه)
- نوع این Generic
- مقدار مورد انتساب که بوسیله ":=" انتساب داده می شود
- یک توضیح کوتاه در مورد کاربرد این متغیر در برنامه که بوسیله - - مشخص می شود

تفاسیر گفته شده در بالا را با مثال زیر دقیقا مقایسه کنید تا به مرتبه درک بالاتری برسید:




```
Entity Eight_Bit_Register is
   Generic ( Bus_Width : integer := 8 ; -- Parameters
             MaxDelay : time :=20 ns); -- Parameters
-- Connections
-- Connections
End Entity Eight_Bit_Register;
```

به تصویر زیر خوب نگاه کنید مجموعه MyProcessor دارای یک ورودی به صورت Bus می باشد که دارای پهنای ۷ بیت است حالا فرض کنید در برنامه VHDL آن ، پهنای این Bus متغیر باشد یعنی می تواند اعداد دیگری برای پهنای Bus در نظر گرفته شود در اینجاست که می توان از دستور Generic بهره گرفت به Code های زیر که در مورد همین شکل نوشته شده است توجه کنید:




```
Entity MyProcessor is
  Generic ( BusWidth : integer := 8 );
Port ( . . .
       DataBus : inout bit_vector ( BusWidth-1  downto  0 );
       . . .
  );
End entity MyPeocessor;
```

حالا باید مفهوم و طرز استفاده از Generic را خوب فهمیده باشید اگر نفهمیدید دوباره از ابتدای درس با خونسردی کامل آن را مطالعه کنید.

چند نکته مهم برنامه نویسی:

۱-همیشه برنامه هایتان را با ارائه توضیحی مناسب برای هر خطش خواناتر و زیباتر کنید ، منظور ، آوردن Comment های مناسب در انتهای خطوط برنامه است.

۲-همیشه در انتخاب نام برای متغیرها و اسامی Entity و Architecture و ... وسواس به خرج دهید یعنی نامهای منتخب از سوی شما باید به تنهایی قادر به ارائه یک Comment به خواننده باشد.

٦-انواع داده ها Data Types

الف-Boolean

حتما با این نوع داده آشنا هستید : درست و غلط یا به عبارتی *True , False*

نحوه اعلان آن به صورت مقابل است:

```
Type Boolean is ( false , true );
```

ب-Character

نحوه اعلان :

```
Type character is ( null , soh , ... ,'a','b',... );
```

ج-Integer

نحوه اعلان :

```
Type integer is range -2147483647 to 2147483647;
```

د-Real

نحوه اعلان:

```
Type real is range -1.0E308 to 1.0E308;
```

ه-Bit

نحوه اعلان:

```
Type Bit is ( '0' , '1' );
```

تعریف فیزیکی یک نوع داده دلخواه:

کمیتهای فیزیکی مانند زمان (Time)، فاصله (Distance) ، جریان (Current) ، دما (Temperature) و غیره را می توان تعریف کرد به مثال زیر که در مورد تعریف Time می باشد توجه کنید:

```
Type time is range -2147483647 to 2147483647
   Units
       Fs:                 --Primary Units
       Ps = 1000 fs;       --Secondary Units
       Ns = 1000 ps;       --Secondary Units
       Us = 1000 ns;       --Secondary Units
       Ms = 1000 us;       --Secondary Units
       Sec = 1000 ms;      --Secondary Units
       Min = 60 sec;       --Secondary Units
       Hr   =60 min;       --Secondary Units
  End Units;
```

به مثال حل شده زیر که در مورد دروس قبل و این درس می باشد توجه کنید :

مثال : می خواهیم برنامه ای بنویسیم که سه بیت x,y,z را به عنوان ورودی دریافت کرده و حاصل جمع آنها را در خروجی S و Carry آن را در خروجی C قرار می دهد.

```
Entity Sum is
  Port ( x , y , z : in Bit ;
         S , C : out Bit );
End entity;

Architecture RTL of Sum is
   Signal Sig : Bit ;
    Begin
              Sig<=x xor y ;
              S   <=Sig xor z ;
              C   <=(x and y) or (y and z) or (x and z) ;
End RTL;
```

تمرین : برنامه یک Full Adder را بنویسید مثلا این برنامه ی شما باید دو ورودی از نوع Bus چهار بیتی داشته باشد که حاصل جمع آنها را در یک خروجی از نوع Bus چهار بیتی قرار می دهد.در ضمن مقدار Carry فراموش نشود.
جواب این سوال را در درس بعدی خواهم آورد.
اگر جواب سوال را نتوانستید بنویسید هیچ مهم نیست مهم این است که در مورد راه حل آن فکر کنید فقط همین ! تا درس آینده حق یارتان باد .

(قسمت چهارم)

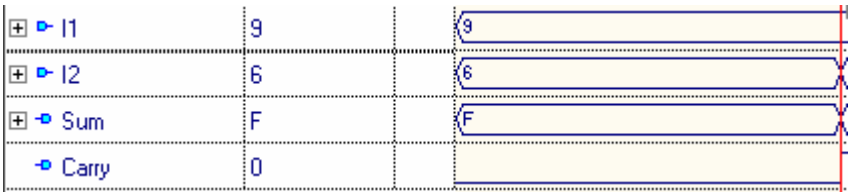
تحلیل یک برنامه VHDL که در درس قبلی به عنوان تمرین مطرح شد و آشنایی با چند دستور دیگر:

برنامه یک Full Adder چهار بیتی :

```
library IEEE ;
use IEEE.STD_LOGIC_1164.all ;
entity Adder is
port(
I1 : in STD_LOGIC_VECTOR(3 downto 0) ;
I2 : in STD_LOGIC_VECTOR(3 downto 0) ;
Carry : out STD_LOGIC;
Sum : out STD_LOGIC_VECTOR(3 downto 0)
) ;
end Adder;
architecture FullAdder of Adder is
signal c1,c2,c3:std_logic ;
begin
c1<=i1(0) and i2(0) ;
sum(0)<=i1(0) xor i2(0) ;
c2<=(i1(1)and i2(1)) or (i1(1)and c1) or (i2(1)and c1) ;
sum(1)<=i1(1) xor i2(1) xor c1 ;
c3<=(i1(2)and i2(2)) or (i1(2)and c2) or (i2(2)and c2) ;
sum(2)<=i1(2) xor i2(2) xor c2 ;
carry<=(i1(3)and i2(3)) or (i1(3)and c3) or (i2(3)and c3) ;
sum(3)<=i1(3) xor i2(3) xor c3 ;
end FullAdder ;
```

نکته ای درمورد نوع داده ی STD_LOGIC :

شما می توانید به جای نوع داده ی Bit از آن استفاده کنید زیرا مزایایی دارد که در طول دروس بعدی متوجه آنها خواهید شد پس از این به بعد همیشه به جای Bit از STD_LOGIC استفاده خواهیم کرد.
در برنامه فوق دو ورودی داریم از نوع Bus و یک خروجی از نوع Wire و یک خروجی از نوع Bus.
شکل زیر حاصل تست برنامه جمع کننده فوق در Wave Form نرم افزار Active VHDL است:



می دانید که حاصل جمع دو بیت را می توانیم توسط XOR بدست آوریم و مقدار Carry هم با And کردن دو بیت بدست می آید اگر دوست دارید با این مسئله در گیر شوید برنامه Full Adder هشت بیتی را بنویسید.فکر نمی کنم با خواندن برنامه بالا چیزی دستگیرتان شود فقط باید خودتان زحمت بکشید.البته برنامه فوق یک برنامه Structural است و به ساده ترین روش نوشته شده است.
این درس همین جا تمام شد و قسمت اعظم کار با شماست یعنی برنامه ی فوق را خط به خط تریس کنید تا کاملا متوجه شوید (البته دوستانی که ضعیفند)
می توانید برنامه بالا رو راحت کپی کنید در نرم افزار Active VHDL سپس کامپایل کرده و به شکل Wave که در بالا آوردم برسید .به متغیرها همونطور که من عدد دادم عدد بدین و اونو اجرا کنید.
اگر با این نرم افزار آشنا نیستید الان فرصت خوبیه برای آشنایی !
بدرود

نگارنده : فرشید سفیدگران
کارشناسی کامپیوتر سخت افزار
خرداد ۱۳۸۲
Sefidgaran@gmail.com
http://Sefidgaran.blogfa.com