مقدمه

نرم افزار Max+Plus II ابزار سنتزی است که توسط شرکت Altera برای طراحی مدارهای دیجیتال به بازار عرضه شده است. با استفاده از این نرم‌افزار، کاربر می‌تواند مدار مورد نظر خود را بصورت شماتیک، شکل موج، زبانهای توصیف سخت افزاری AHDL ، VHDL و Verilog و یا ترکیبی از آنها توصیف کند. سپس این مدار بعنوان ورودی ابزار سنتز دریافت شده و پس از اینکه مدار کامپایل شد، بر روی یک تراشه‌ی خاص (از تراشه های Altera) نگاشت می‌شود. در این حالت کاربر می‌تواند با استفاده از قابلیتهایی که در این ابزار سنتز تعبیه شده است، شکل موج ورودیهای مورد نظر خود را به مدار اعمال کند و خروجیهای مدار را مشاهده کند.

در این گزارش تلاش شده است که چگونگی طراحی مدارهای دیجیتال با استفاده از Max+Plus II 8.3 بصورت ساده مورد بررسی قرار بگیرد. این گزارش در پنج بخش تنظیم شده است. در بخش نخست با طراحی مدار با استفاده از زبان توصیف سخت‌افزاری VHDL آشنا می‌شویم. در بخش دوم طراحی مدار به روش شماتیک مورد بررسی قرار می‌گیرد. در بخش سوم طراحی مدار توسط زبان توصیف سخت افزاری AHDL را مورد بررسی قرار می‌دهیم. در بخش چهارم مروری مختصر بر زبان توصیف‌سخت‌افزاری VHDL خواهیم داشت. در بخش پنجم زبان توصیف‌سخت‌افزاری AHDL را مورد بررسی قرار می‌دهیم.

در این گزارش برای سادگی از نمادهای زیر استفاده می‌کنیم :

<DL>	Double click the left button of mouse
<LB>	Click the left button of mouse
<RB>	Click the right button of mouse
<M> Menu1: Option1	Select Option1 from Menu1

۱ طراحی مدار با استفاده از VHDL

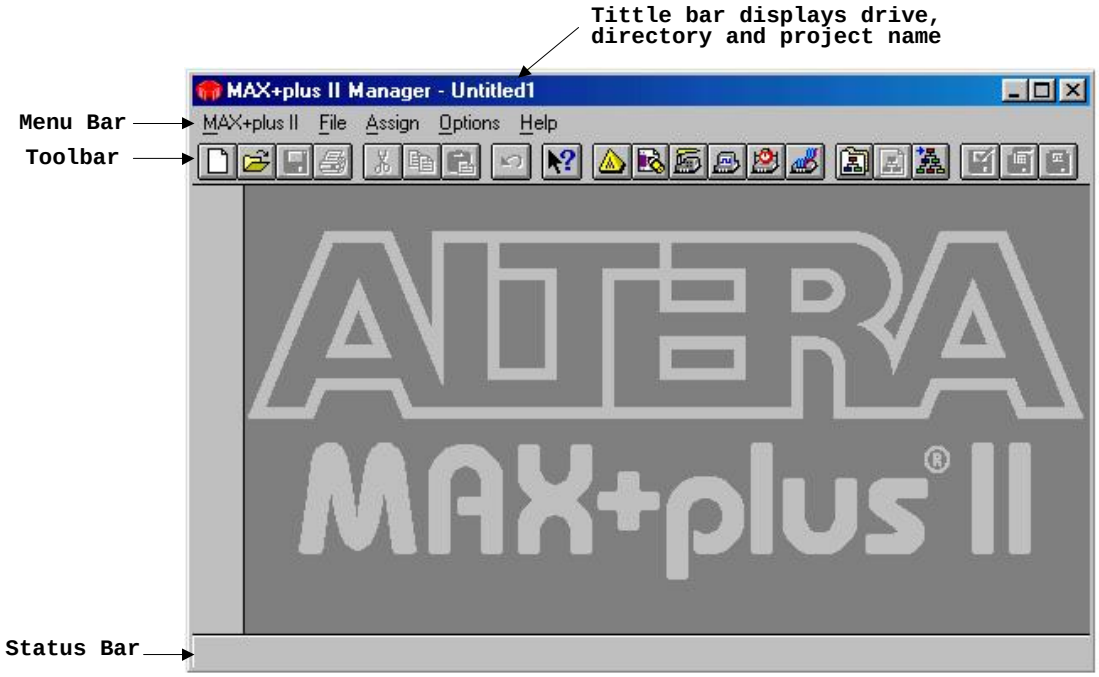
برای توضیح روش طراحی مدار با استفاده از زبان توصیف سخت‌افزاری VHDL ابتدا یک سلول Half Adder طراحی کرده و سپس با اتصال دو عدد از این سلولها به یکدیگر یک سلول جمع‌کننده‌ی Full Adder می‌سازیم و در پایان با اتصال این سلولها به یکدیگر یک جمع‌کننده‌ی چهار بیتی می‌سازیم. برای این منظور بصورت زیر عمل می‌کنیم.

۱-۱ اجرای نرم‌افزار Max+Plus II

برای اجرای برنامه‌ی Max+Plus II کافیست بصورت زیر عمل کنیم :

<DL> on Max+Plus II icon on desktop

در اینحالت پنجره‌ای مطابق شکل ۱-۱ روی صفحه ظاهر می‌شود. بخشهای مختلف این پنجره روی شکل مشخص شده است.



شکل ۱-۱- پنجره‌ی نرم‌افزار Max+Plus II

به صورت زیر می‌توان هر یک از موارد Tool bar و Status bar را فعال و یا غیر فعال نمود.

<M> Options: Prefrence

برای خروج از برنامه بصورت زیر عمل می‌کنیم :

<M> File: Exit Max+plus II

۱-۲ ایجاد پروژه‌ی جدید

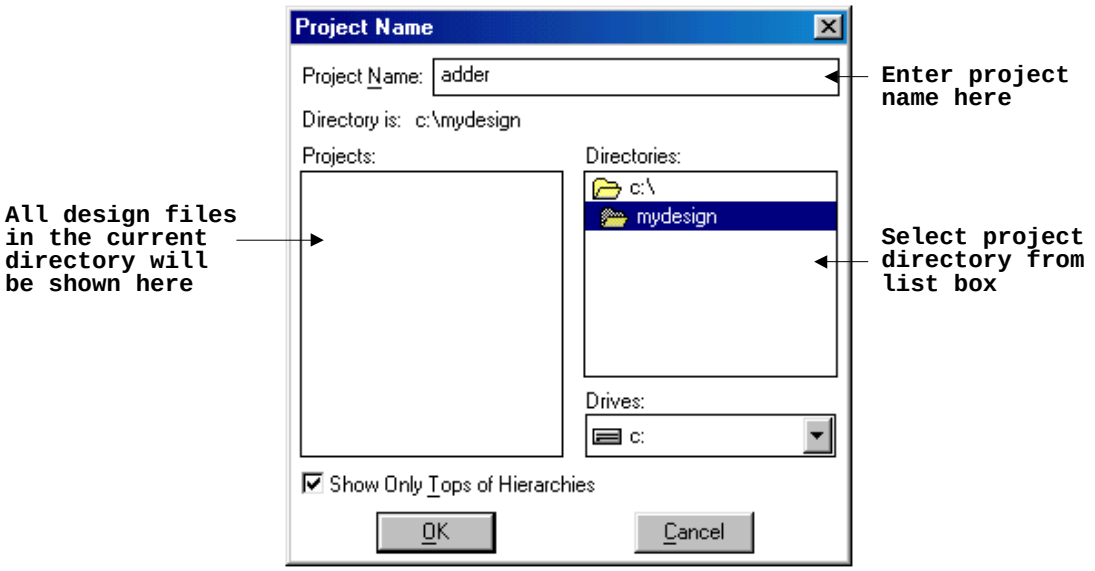
در این نرم افزار هر طرح جدید (که ممکن است از چندین زیر طرح تشکیل شده باشد) توسط یک پروژه مدیریت می‌شود. از اینرو برای ایجاد یک طرح جدید نخست باید یک پروژه‌ی جدید ایجاد کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> File: Project: Name

در اینحالت یک پنجره مشابه شکل ۱-۲ ظاهر می‌شود. در اینجا مطابق شکل دایرکتوری mydesign و نام پروژه را adder انتخاب می‌کنیم.

<LB> on OK

در اینحالت نام و دایرکتوری پروژه بر روی Tittle bar پنجره‌ی Max+PlusII نمایش داده می‌شود.



شکل ۱-۲- پنجره‌ی تعیین نام پروژه

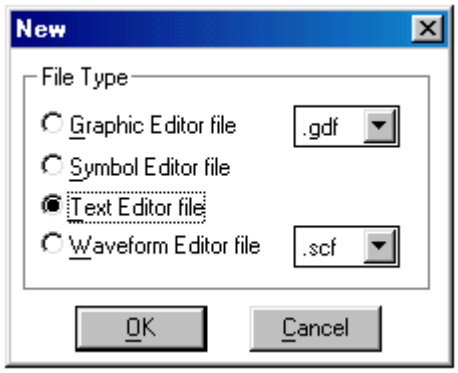
۱-۳ ایجاد فایل جدید

پس از اینکه نام پروژه را مشخص کردیم، باید طرحهایی را که در این پروژه مورد استفاده قرار می‌گیرد ، به این پروژه اضافه کنیم. این امر بصورت ایجاد فایلهای جدید صورت می‌گیرد. در اینجا ابتدا یک سلول Half Adder طراحی کرده و سپس با اتصال دو عدد از این سلولها به یکدیگر یک سلول جمع‌کننده‌ی Full Adder می‌سازیم و در پایان با اتصال این سلولها به یکدیگر یک جمع‌کننده‌ی چهار بیتی می‌سازیم. باید دقت داشت که همیشه باید نام پروژه با نام مدار در بالاترین سطح یکسان باشد.

<M> File: New

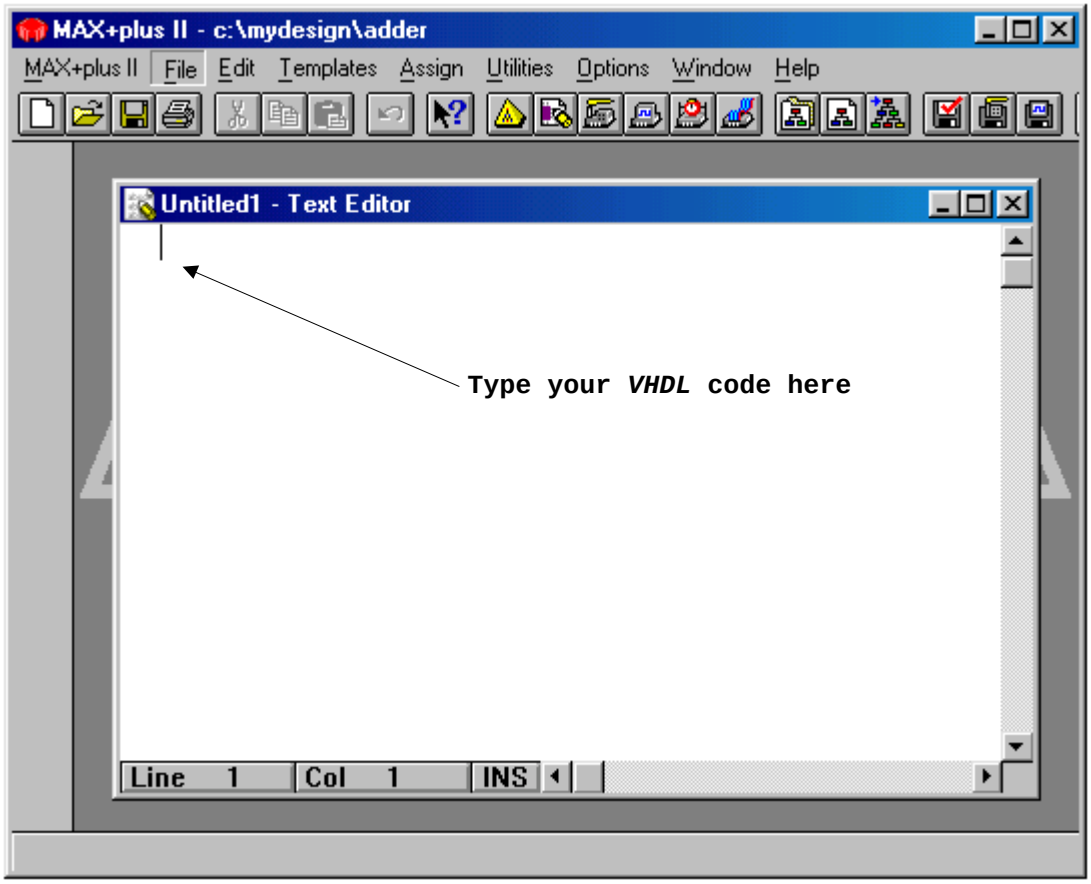
پس از این امر پنجره‌ی ایجاد فایل جدید مطابق شکل ۱-۳ بر روی صفحه نمایش داده می‌شود. حال بصورت زیر عمل می‌کنیم :

Select Text Editor file
<LB> on OK



شکل ۱-۳- پنجره‌ی ایجاد فایل جدید

پس از انجام این عملیات، پنجره‌ی ویرایشگر متنی مطابق شکل ۱-٤ بر روی صفحه ظاهر می‌شود. این ویرایشگر قابلیت ویرایش فایلهای زبانهای توصیف سخت‌افزاری VHDL، Verilog و AHDL را داراست. بنابراین پیش از آنکه توصیف مدار را وارد کنیم، بهتر است پسوند فایل متنی مورد نظر خود را مشخص کنیم. با این امر ویرایشگر تشخیص می‌دهد برای نمایش رنگی کلمات کلیدی از قواعد نحوی چه زبانی استفاده کند. بنابراین پیش از ورود اطلاعات فایل جدید را ذخیره می‌کنیم.



شکل ۱-٤- پنجره‌ی ویرایشگر متنی

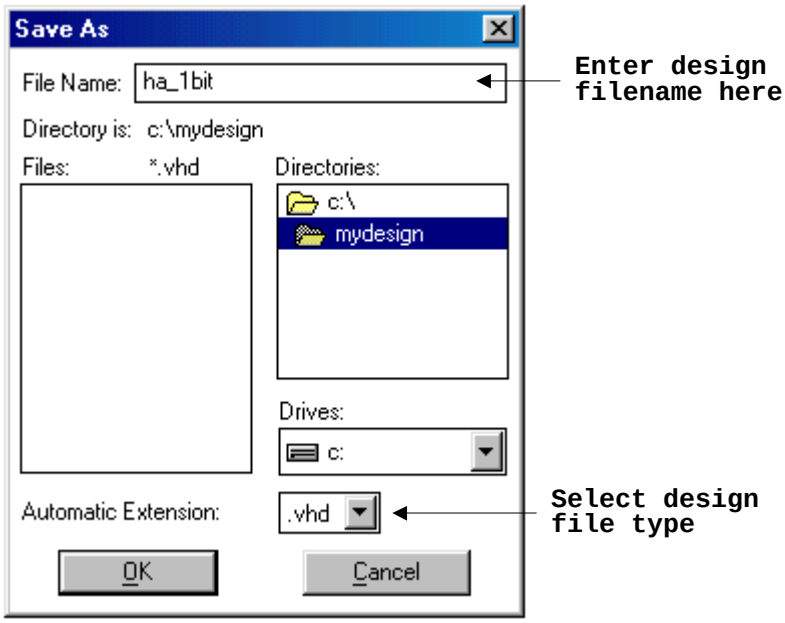
برای ذخیره کردن فایل جدید بصورت زیر عمل می‌کنیم.

<M> File: Save As

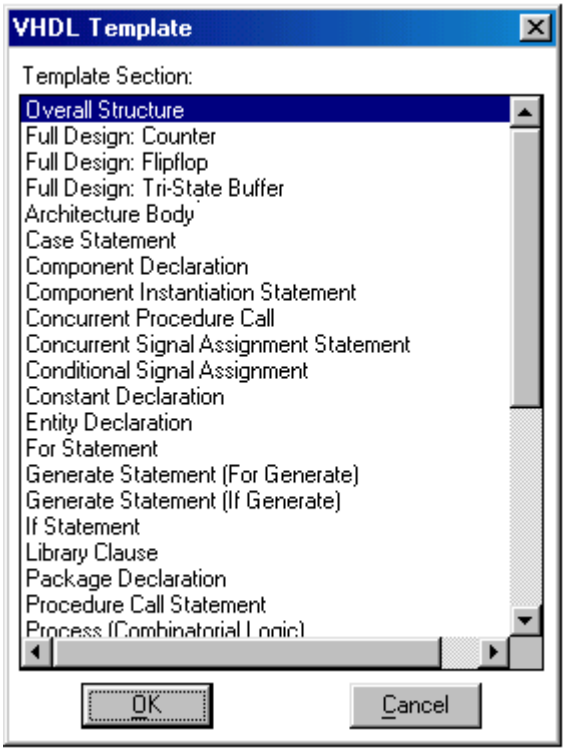
در پنجره‌ی ظاهر شده نام ha_1bit را بعنوان نام فایل وارد می‌کنیم (شکل ۱-۵ را ببینید). توجه کنید که در بخش مربوط به نوع فایل حتما پسوند vhd را انتخاب کنید (پسوند فایلهای VHDL بصورت vhd می‌باشد).

<LB> on OK

برای وارد کردن کد، ویرایشگر متنی دارای یک قابلیت بسیار مناسب بنام Template است که الگویی از بیشتر ساختارهای زبان را در اختیار شما قرار می‌دهد. برای دسترسی به این قابلیت کافیست در حالیکه ماوس روی صفحه متنی در محل مورد نظر است، با کلید سمت راست ماوس کلیک کنیم و سپس در منوی Case Sensitive ظاهر شده گزینه‌ی VHDL را انتخاب کنیم. در اینحالت پنجره‌ای مطابق شکل ۱-٦ روی صفحه ظاهر می‌شود که می‌توان از آن الگوی مورد نظر را نتخاب نمود و کلید Ok را زد. در اینصورت یک کپی از الگوی این ساختار در محل مورد نظر ظاهر می‌شود و حالا می‌توان این الگو را بر اساس نیاز تغییر داد. این قابلیت کاربر را از به خاطر سپردن الگوی ساختارهای زبان بی‌نیاز می‌سازد.



شکل ۱-۵- پنجره‌ی ذخیره‌ی فایل جدید



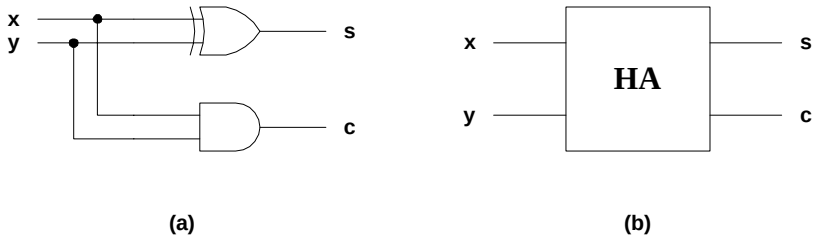
شکل ۱-۶- پنجره‌ی الگوی VHDL

برای مثال اگر از پنجره‌ی الگوی VHDL گزینه‌ی Entity Declaration را انتخاب کنیم و کلید OK را برنیم، کدی مشابه لیست ۱-۱ در محل مورد نظر ظاهر می‌شود، که کاربر می‌تواند با تغییر نام Entity و نام پورتهای ورودی، خروجی و پورتهای دوسویه توصیف مورد نظر خود را بیان کند.

```
ENTITY __entity_name IS
  GENERIC(__parameter_name : string :=  __default_value;
          __parameter_name : integer:=  __default_value);
  PORT(
       __input_name, __input_name : IN STD_LOGIC;
       __input_vector_name : IN STD_LOGIC_VECTOR(__high DOWNTO __low);
       __bidir_name, __bidir_name : INOUT STD_LOGIC;
       __output_name, __output_name : OUT STD_LOGIC);
END __entity_name;
```

لیست ۱-۱- الگوی Entity

در شکل ۱-۷ (a) نمودار منطقی سلول Half Adder آمده است. در شکل ۱-۷ (b) ورودی و خروجیهای این سلول نمایش داده شده‌اند. اکنون توصیف سلول Half Adder را با استفاده از زبان VHDL در پنجره‌ی ویرایشگر متنی وارد می‌کنیم. این توصیف در لیست ۱-۲ آمده است. نکته‌ای که باید به آن توجه داشت این است که همیشه نام entity باید با نام فایل حاوی آن یکسان باشد. توجه کنید که برای قابل سنتز بودن طرح باید بجای نوع داده‌ای bit از std_logic استفاده شود. تعریف این نوع داده‌ای در کتابخانه‌ی IEEE و در بسته‌ی std_logic_1164 وجود دارد.

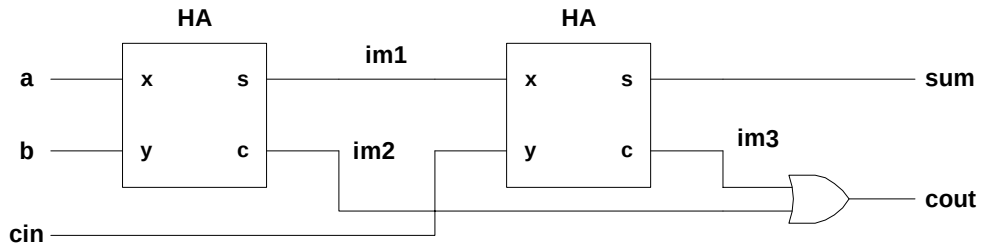


شکل ۱-۷- (a) نمودار منطقی سلول Half Adder (b) سلول Half Adder

```
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;
```

لیست ۱-۲- توصیف Half Adder

حال با اتصال دو سلول Half Adder به یکدیگر یک سلول Full Adder می‌سازیم. در شکل ۱-۸ چگونگی اتصال این سلولها به یکدیگر نمایش داده شده است. اکنون یک فایل متنی جدید دیگر ایجاد می‌کنیم و آنرا با نام fa_1bit ذخیره می‌کنیم. این فایل حاوی توصیف یک سلول جمع‌کننده‌ی یک بیتی است. این توصیف در لیست ۱-۳ آمده است.



شکل ۱-۸- نمودار منطقی سلول Full Adder

همین عملیات را برای فایل اصلی پروژه یعنی adder انجام می‌دهیم. توصیف جمع‌کننده‌ی چهار بیتی در لیست ۱-٤ آمده است.

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY fa_1bit IS
     PORT(
            a, b, cin    : IN  STD_LOGIC;
            sum, cout    : OUT STD_LOGIC);
END fa_1bit;
```

لیست ۱-۳- توصیف Full Adder یک بیتی

```
ARCHITECTURE structural 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 ( a, b, im1, im2 );
     ha2 : ha_1bit PORT MAP ( im1, cin, sum, im3 );
     cout <= im2 OR im3;
END structural;
```

لیست ۱-۳- توصیف Full Adder یک بیتی (ادامه)

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY adder 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 adder;

ARCHITECTURE structural OF adder IS
     COMPONENT fa_1bit PORT (
            a, b, cin : IN STD_LOGIC;
            sum, cout : OUT STD_LOGIC );
     END COMPONENT;
     SIGNAL im1, im2, im3 : STD_LOGIC;
BEGIN
     fa0 : fa_1bit PORT MAP ( a(0), b(0), cin, sum(0), im1 );
     fa1 : fa_1bit PORT MAP ( a(1), b(1), im1, sum(1), im2 );
     fa2 : fa_1bit PORT MAP ( a(2), b(2), im2, sum(2), im3 );
     fa3 : fa_1bit PORT MAP ( a(3), b(3), im3, sum(3), cout );
END structural;
```

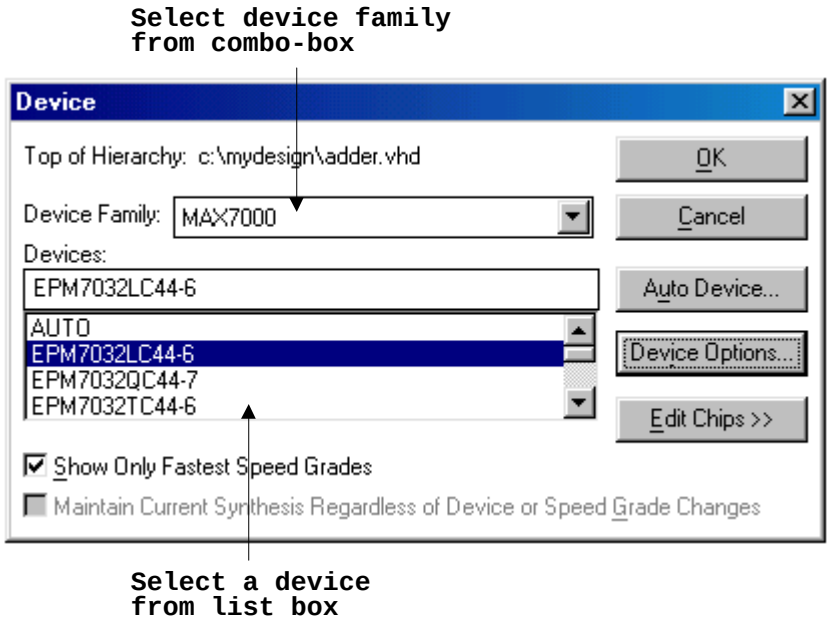
لیست ۱-٤- توصیف جمع‌کننده‌ی چهاربیتی

۱-٤ انتخاب تراشه

پس از اینکه طرح آماده شد، باید آن را کامپایل کنیم. برای این منظور نخست باید نوع تراشه‌ای را که می‌خواهیم طرح ما بر روی آن نگاشت شود، مشخص کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> Assign: Device, to show Device window

همانطور که در شکل ۱-۹ دیده می‌شود، در پنجره‌ی انتخاب تراشه، نخست باید از لیست Device Family خانواده‌ی تراشه را انتخاب کنیم. در اینصورت در لیست Devices نام تراشه‌های موجود در این خانواده به نمایش در می‌آید. در صورتیکه بخواهیم ابزار سنتز مناسبترین تراشه را برای طرح ما انتخاب کند، باید در لیست Devices گزینه‌ی AUTO را انتخاب کنیم.



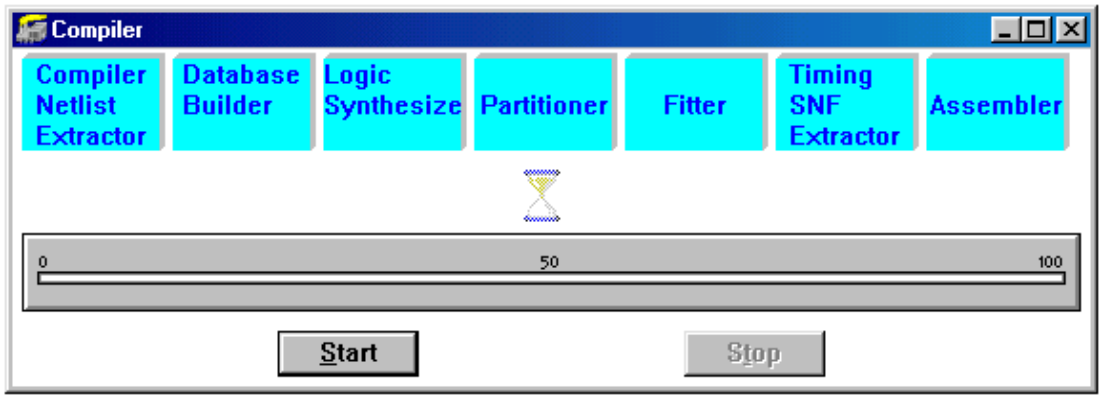
شکل ۱-۹- پنجره‌ی انتخاب تراشه

۱-۵ کامپایل طرح

پس از انتخاب نوع تراشه باید طرح را کامپایل نمود. برای این منظور باید بصورت زیر عمل کنیم:

<M> Max+Plus II: Compiler

در اینحالت پنجره‌ی کامپایلر مطابق شکل ۱-۱۰ بر روی صفحه به نمایش در می‌آید. برای شروع عملیات کامپایل توسط ماوس باید روی کلید Start کلیک کنیم.



شکل ۱-۱۰- پنجره‌ی کامپایلر

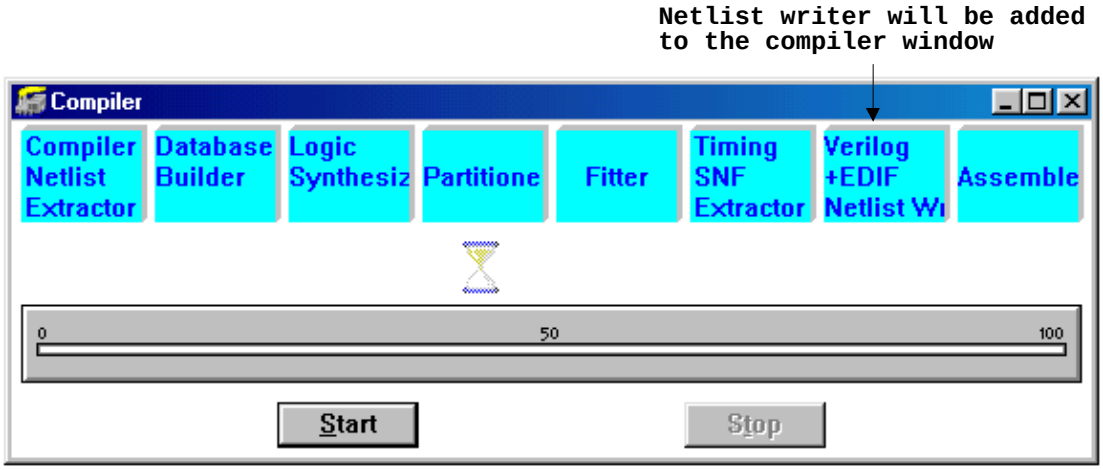
برای اینکه یک سیستم سنتز کامل (از توصیف رفتاری مدار تا لی‌اوت VLSI) داشته باشیم، باید توسط ابزار سنتز یک فایل خروجی Netlist تولید کنیم. این فایل خروجی، مدار مورد نظر را بصورت مجموعه‌ای از عناصر کتابخانه بیان می‌کند که به یکدیگر متصل هستند. قالب این فایل خروجی می‌تواند

بصورت Verilog یا EDIF باشد. از این فایل خروجی برای تولید لی‌اوت VLSI استفاده می‌کنیم.

برای اینکه نرم‌افزار Max+PlusII خروجی Verilog و یا EDIF را تولید کند، بصورت زیر عمل می‌کنیم:

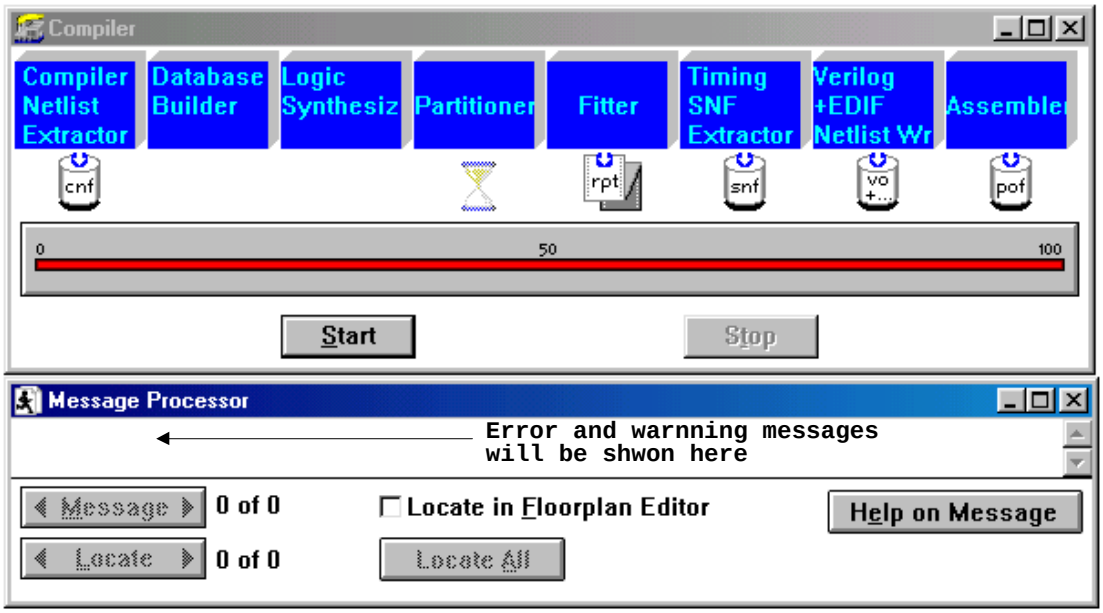
<M> Interface: EDIF Netlist Writer
<M> Interface: Verilog Netlist Writer

پس از انجام این عمل، پنجره‌ی کامپایلر بصورت شکل ۱-۱۱ در می‌آید.



شکل ۱-۱۱- پنجره‌ی کامپایلر برای تولید خروجی Verilog و EDIF

اکنون با کلیک کردن روی کلید Start عملیات کامپایل را آغاز می‌کنیم. در طی انجام عملیات کامپایل گزارشی از خطاها و warning های احتمالی در یک پنجره بنام Message به نمایش در می‌آید. چنانچه بخواهیم محل بروز خطا را مشاهده کنیم، کافیست توسط ماوس روی خطای مورد نظر کلیک کرده و سپس روی کلید Locate کلیک کنیم (شکل ۱-۱۲ را ببینید). در صورتیکه طرح شما درون یک تراشه جای نگیرد، ابزار سنتز به شما پیام می‌دهد که آیا می‌خواهید این طرح را درون دو (یا چند) تراشه جای دهید یا خیر. در صورتیکه شما ادامه‌ی عمل کامپایل را تایید کنید، ابزار سنتز طرح را به چند بخش تقسیم می‌کند و هر بخش را درون یک تراشه جای می‌دهد و لیستی از پایه‌های تراشه‌ها که باید به یکدیگر مرتبط شوند را ارائه می‌کند.



شکل ۱-۱۲- پنجره‌ی پیامها

۱-۶ شبیه‌سازی طرح

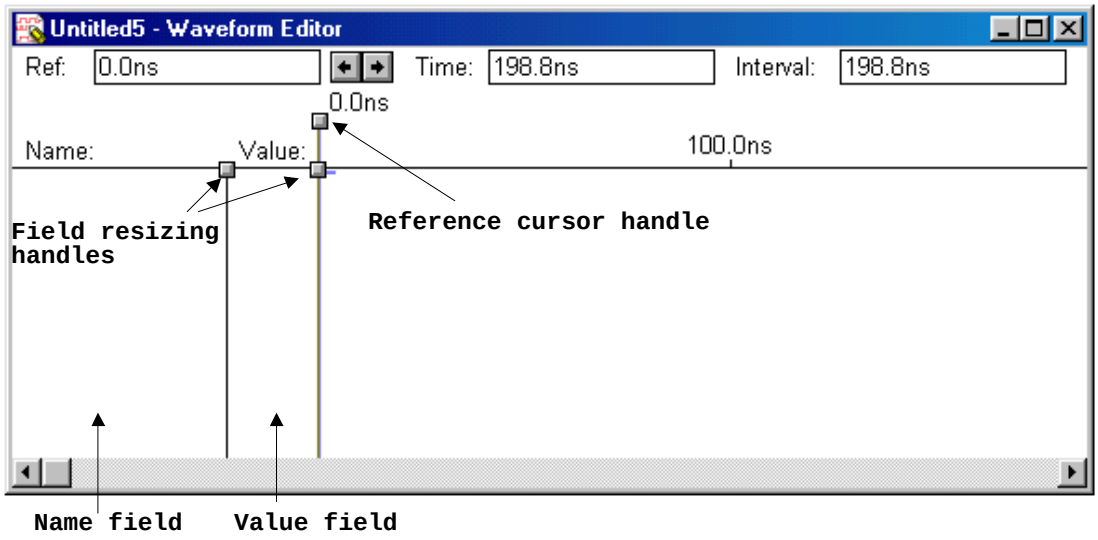
پس از اینکه طرح مورد نظر بر روی تراشه نگاشت شد، باید آن را شبیه سازی نمود تا از صحت طرح مطمئن شویم. برای این منظور نخست باید شکل موج ورودی مدار را طراحی کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> Max+Plus II: Waveform Editor

در اینحالت پنجره‌ی ویرایش شکل موج روی صفحه ظاهر می‌شود. این پنجره در شکل ۱-۱۳ نمایش داده شده است و قسمتهای مختلف این پنجره در شکل مشخص شده است.

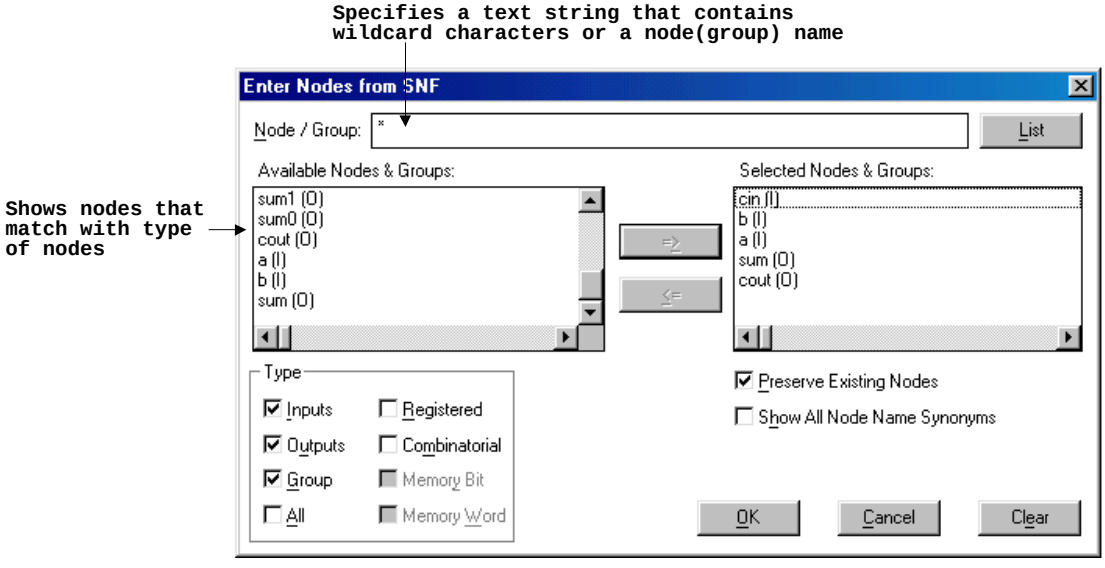
در مرحله‌ی کامپایل طرح، یک فایل با پسوند SNF ایجاد می‌شود که مشخصات تمام گره‌های مدار را در خود ذخیره می‌کند. از اینرو بهتر است که نام گره های ورودی و خروجی مدار را از این فایل استخراج کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> Node: Enter Node from SNF



شکل ۱-۱۳- پنجره‌ی ویرایش شکل موجها

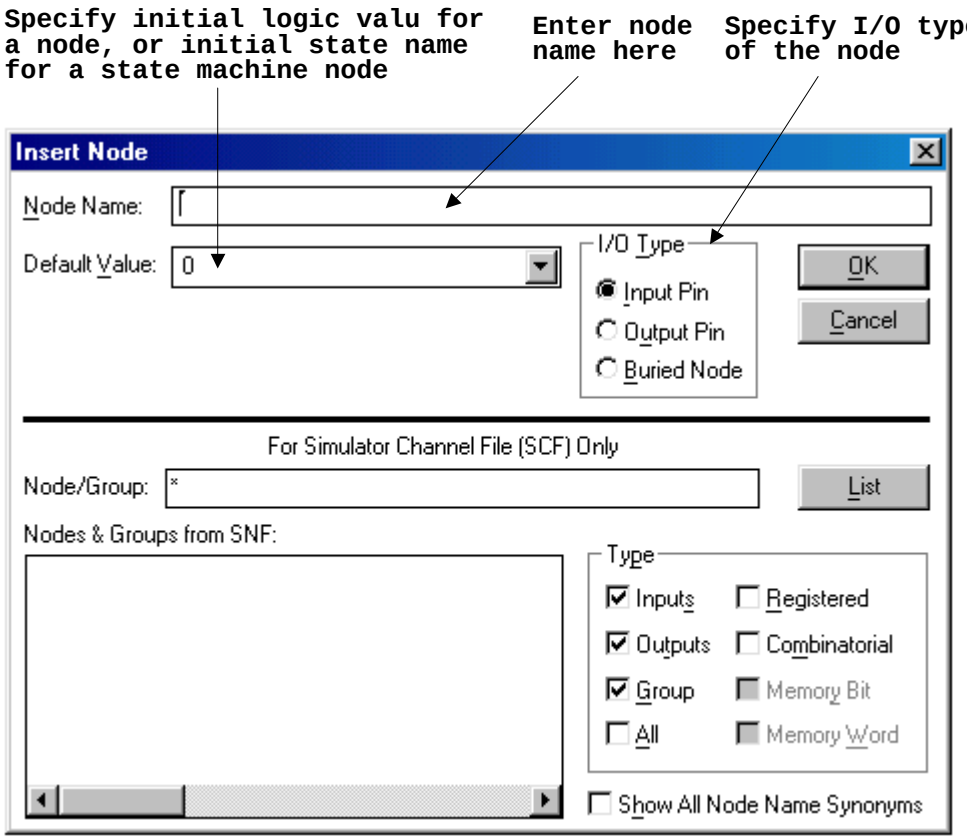
پنجره‌ای مطابق شکل ۱-۱٤ روی صفحه ظاهر می‌شود. در این پنجره می‌توان نوع گره‌های مورد نظر را انتخاب نمود، این امر در بخش Type انجام می‌شود. پس از انتخاب نوع گره ها کلید List را می‌زنیم تا لیست گره های موجود در مدار، در بخش سمت چپ پنجره ظاهر شود. سپس هر یک از ورودیها و یا خروجیها را که بخواهیم انتخاب می‌کنیم و آنها را توسط کلید => به لیست شکل موجها اضافه می‌کنیم و در پایان کلید Ok را می‌زنیم.



شکل ۱-۱٤- پنجره‌ی انتخاب گره‌ها

چنانچه بخواهیم بعدها یک گره را به این لیست اضافه کنیم در پنجره‌ی ویرایشگر شکل موج در محل فیلد نام <DL> می‌کنیم، در اینصورت پنجره ای مطابق شکل ۱-۱۵ روی صفحه ظاهر می‌شود، سپس نام گره مورد نظر و نوع گره را مشخص می‌کنیم و در صورت لزوم مقدار پیش فرض گره را وارد

می‌کنیم و کلید Ok را می‌زنیم تا گره مورد نظر به لیست گره های ویرایشگر شکل موج افزوده شود. به هنگام افزودن گره با این روش باید دقت کرد که گره حتما در طرح موجود باشد.



شکل ۱-۱۵- پنجره‌ی ورود گره

پس از اینکه گره‌های مورد نظر به ویرایشگر شکل موج افزوده شد، باید شکل موج مورد نظر را به ورودیها اعمال نمود. در اینحالت پنجره‌ی ویرایشگر شکل موج مطابق شکل ۱-۱۶ خواهد بود. در این شکل بخشهای مختلف این پنجره نمایش داده شده است.

در حالت عادی طول زمان شبیه سازی برابر ۱ میکروثانیه است. برای تغییر این مقدار باید بصورت زیر عمل نموده و سپس مقدار زمان شبیه سازی را در پنجره‌ی ظاهر شده وارد کنیم.

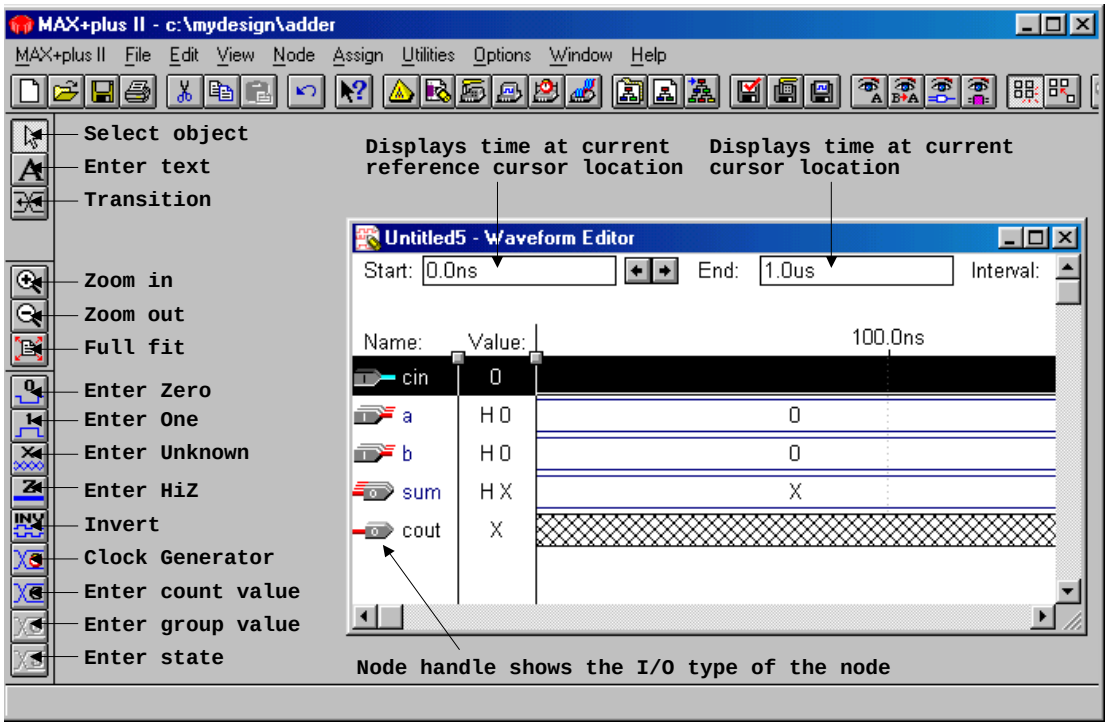
<M> File: End Time

در حالت عادی مقدار Grid زمان برابر ۱۰۰ نانوثانیه است. برای اینکه مقدار Grid زمان را تغییر دهیم بصورت زیر عمل نموده و سپس مقدار Grid زمان را در پنجره‌ی ظاهر شده وارد می‌کنیم.

<M> Options: Grid Size

برای جابجا کردن یک سیگنال در لیست نمایش می‌توان توسط ماوس Handle آن سیگنال را گرفته و آن را در لیست سیگنالها به محل مورد نظر Drag نمود.

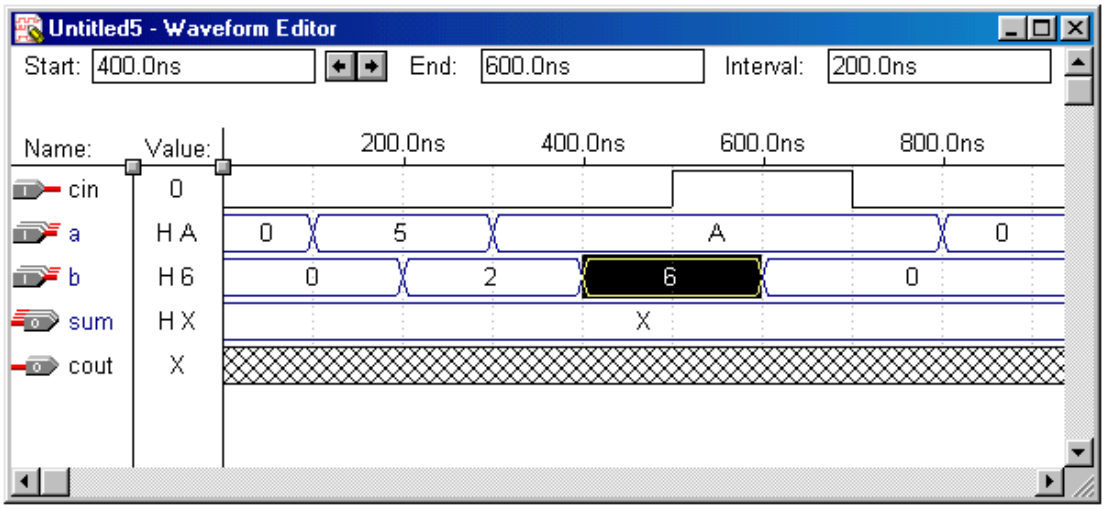
برای اینکه مقدار یک سیگنال ورودی را تغییر دهیم، ابتدا باید آن قسمتی از سیگنال را که می‌خواهیم تغییر دهیم، انتخاب کنیم. برای انتخاب قسمتی از یک سیگنال ابتدا توسط ماوس آیکون Select Object را انتخاب می‌کنیم، سپس قسمت مورد نظر از سیگنال را Drag می کنیم. به این ترتیب رنگ این قسمت تیره می‌شود، حال می‌توانیم توسط آیکونهای سمت چپ صفحه مقدار مورد نظر را وارد کنیم.



شکل ۱-۱۶- پنجره‌ی ویرایش شکل موجها پس از افزودن گره‌ها

در صورتیکه بخواهیم یک سیگنال را بصورت کامل انتخاب کنیم، باید روی مقدار آن سیگنال <LB> کنیم. برای تغییر دادن مقدار یک سیگنال به این نکته توجه کنید که سیگنال یک گره است یا یک گروه و برای مقداردهی به آن از آیکون مناسب استفاده کنید. برای تست این مدار فرض کنید که شکل موج ورودی را بصورت شکل ۱-۱۷ طراحی کرده‌ایم. اکنون بصورت زیر این شکل موج را با نام adder ذخیره می‌کنیم.

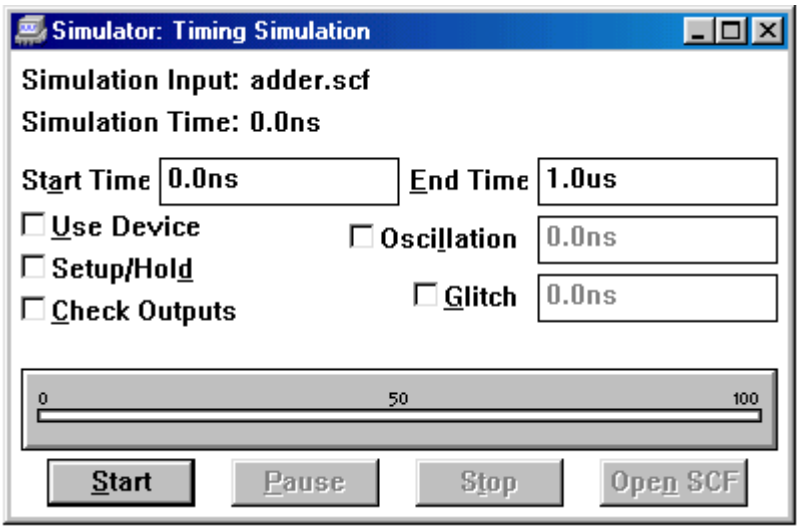
<M> File: Save



شکل ۱-۱۷- پنجره‌ی ویرایش شکل موجها پس از طراحی شکل موجها

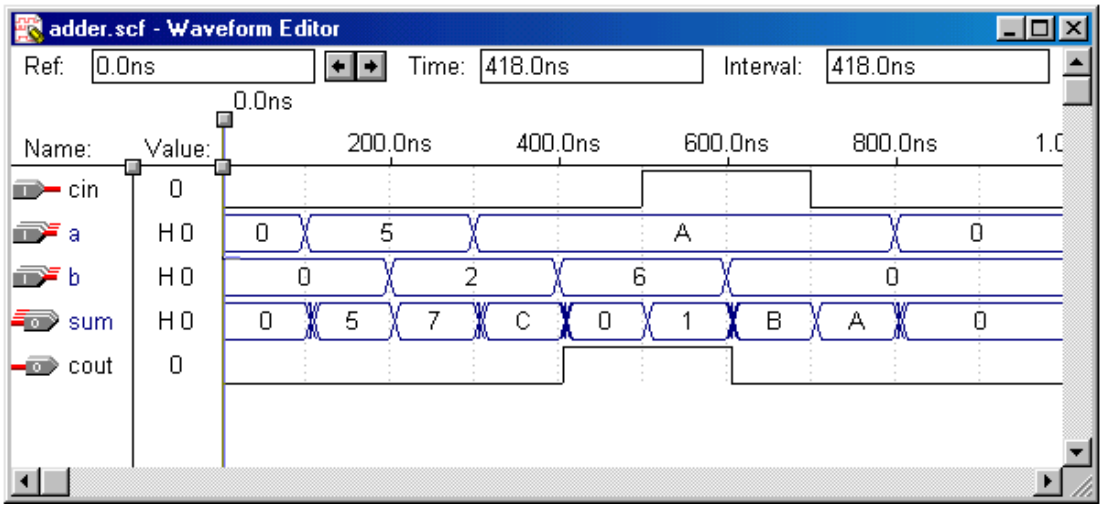
اکنون برای اینکه شبیه سازی این طرح را مشاهده کنیم، باید عمل شبیه سازی را انجام دهیم.
برای این منظور بصورت زیر عمل می‌کنیم :

<M> Max+PlusII: Simulator, to show the simulator window (fig 1-18)
<LB> on Start, to simulate the design
<LB> on Open SCF



شکل ۱-۱۸- پنجره‌ی شبیه‌سازی

با انجام این عمل مجددا پنجره‌ی شکل موجها باز می شود با این تفاوت که در این حالت مقادیر خروجیها نیز مشخص شده اند. حال می توان صحت عملکرد مدار را بررسی نمود (شکل ۱-۱۹ را ببینید).



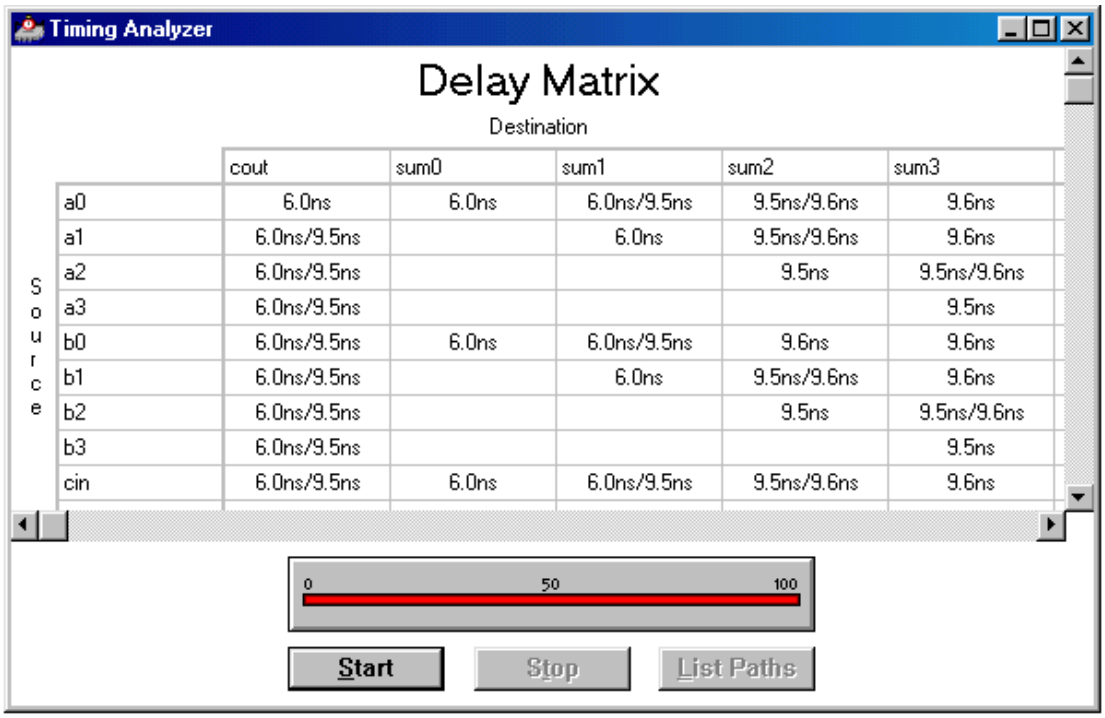
شکل ۱-۱۹- پنجره‌ی شکل موج

۱-۷ تحلیگر زمانی

برای بررسی تاخیر مدار بصورت زیر عمل می کنیم.

<M> Max+PlusII: Timing Analyzer, to show timing analyzer window
<LB> Start

در اینحالت در پنجره‌ی تحلیلگر زمانی، تاخیر ترکیبی ورودیها به خروجیها محاسبه شده و در ماتریس تاخیر به نمایش درمی‌آید (شکل ۱-۲۰ را ببینید).



شکل ۱-۲۰- پنجره‌ی تحلیلگر زمانی

۱-۸ ویرایشگر تراشه

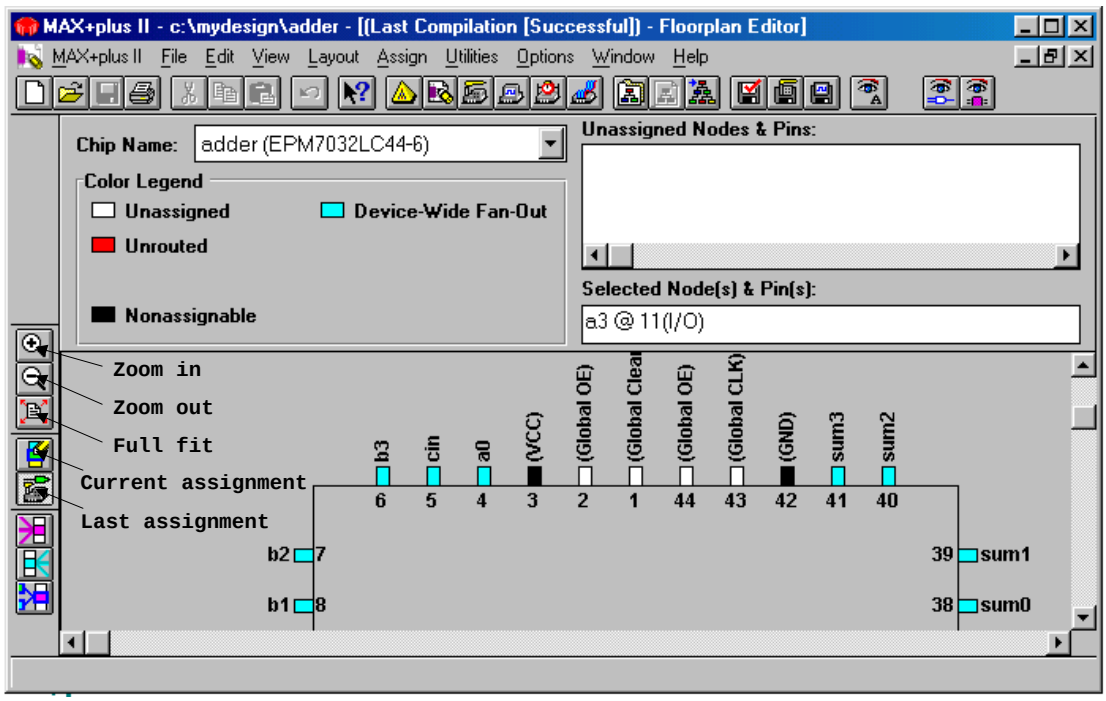
پس از اینکه از صحت عملکرد طرح مطمئن شدیم، باید نحوه‌ی اعمال ورودیها و خروجیها به تراشه را مشخص کنیم. این امر توسط ویرایشگر تراشه انجام می شود. برای اجرای ویرایشگر تراشه بصورت زیر عمل می کنیم.

<M> Max+PlusII: Floorplan Editor, to show floorplan editor window

در حالت عادی این پنجره تراشه را بصورت Logic Array Block (LAB) نمایش می دهد. برای اینکه تراشه بصورت عادی نمایش داده شود، باید بصورت زیر عمل کنیم.

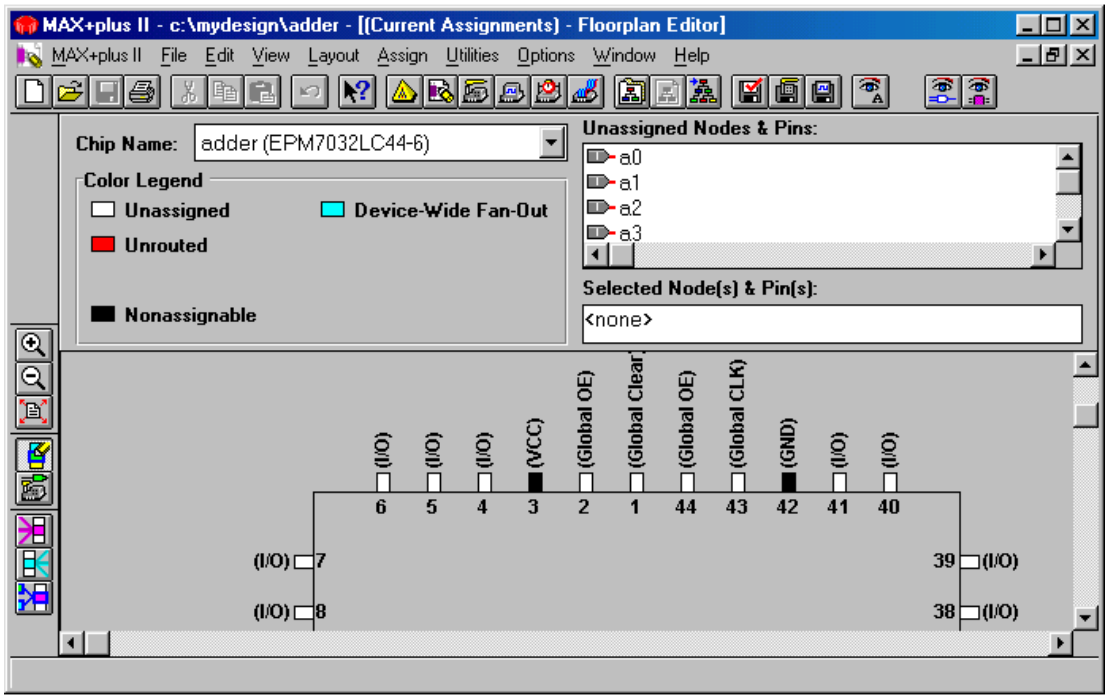
<M> Layout: Device View

در اینحالت پنجره‌ی ویرایشگر تراشه بصورت شکل ۱-۲۱ در می‌آید. بخشهای مختلف این پنجره در شکل نمایش داده شده است.



شکل ۱-۲۱- پنجره‌ی ویرایشگر تراشه

همانطور که در شکل دیده می شود، در قسمت پایین صفحه هر یک از ورودیها و خروجیهای مدار به یکی از پایه های تراشه اختصاص یافته است. چنانچه این تخصیص که مربوط به عملیات کامپایل قبلی است، برای شما مطلوب نباشد می توانید آن را تغییر دهید. برای این منظور باید توسط ماوس روی کلید Current Assignment کلیک کنیم، در اینحالت در قسمت پایین صفحه یک تراشه‌ی خالی نمایش داده می شود (توجه کنید این امر درصورتی امکان پذیر است که قبلا در مرحله‌ی کامپایل، نوع تراشه را انتخاب کرده باشید، بعبارت دیگر انتخاب نوع تراشه AUTO نباشد) و در لیست بالای صفحه لیست گره های مدار نمایش داده می شود (شکل ۱-۲۲ را ببینید). حال می توانید هر یک از گره‌ها را از لیست انتخاب نموده و آن را Drag نموده روی پایه‌ی مورد نظر قرار دهید. دقت کنید که گره‌های ورودی را فقط به پایه های ورودی یا دو سویه و گره های خروجی را فقط به پایه های خروجی یا دو سویه و گره های دو سویه را فقط به پایه های دو سویه تخصیص دهید. نکته‌ی دیگری که باید به آن توجه داشته باشید این است که ممکن است با تخصیصی که شما انتخاب نموده‌اید ابزار سنتز نتواند طرح را درون تراشه جای دهد، از اینرو پس از اینکه تخصیص خود را انتخاب نمودید باید مجددا طرح را کامپایل کنید. در صورتیکه عمل کامپایل موفقیت آمیز باشد، تخصیص شما امکانپذیر است (این امر به علت محدود بودن مسیرهای ارتباطی است که در تراشه موجود است) .



شکل ۱-۲۲- پنجره‌ی ویرایشگر تراشه جهت تخصیص گره‌ها به پایه‌های تراشه

۱-۹ سلسله مراتب پروژه

برای مشاهده‌ی سلسله مراتب پروژه بصورت زیر عمل می‌کنیم:

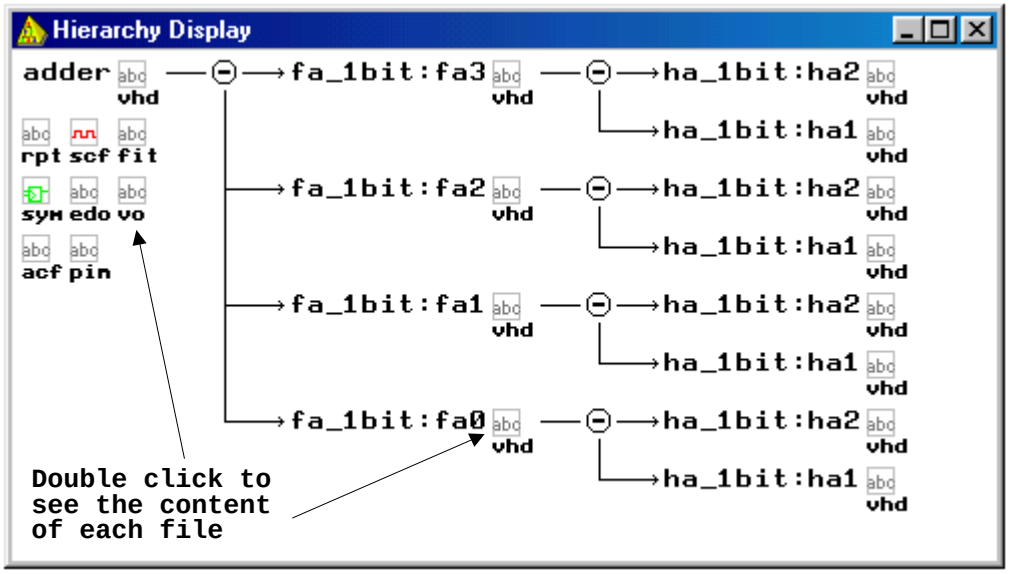
<M> Max+PlusII: Hierarchy Display

در اینحالت یک پنجره مانند شکل ۱-۲۳ روی صفحه ظاهر می‌شود. همانطور که در شکل مشاهده می‌شود سلسله‌مراتب کامل پروژه در این پنجره نمایش داده شده است. با <DB> کردن روی هر یک از آیکونها می‌توان فایل مربوطه را مشاهده نمود. با انجام این مراحل، عملیات طراحی به پایان می‌رسد. حال برای اینکه با مراحل طراحی توسط این نرم‌افزار بیشتر آشنا شویم، به ذکر دو مثال دیگر می‌پردازیم.

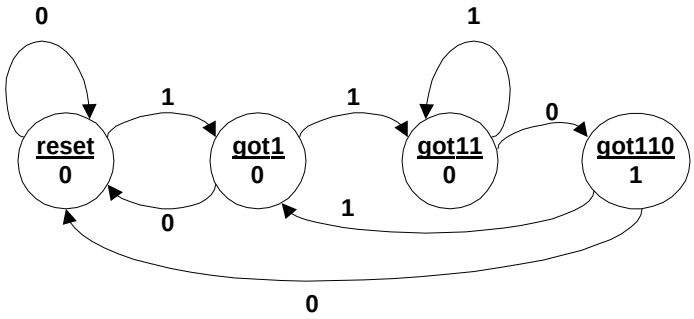
مثال ۱: طراحی تشخیص‌دهنده‌ی دنباله[1]

می‌خواهیم یک مدار طراحی کنیم که دنباله‌ی ۱۱۰ را روی ورودی خود تشخیص دهد و در صورت مشاهده‌ی این دنباله، خروجی خود را یک کند. نمودار حالت این مدار در شکل ۱-۲۴ نمایش داده شده است. کد VHDL مربوطه در لیست ۱-۵ آمده است.

[1] این مثال از کتاب Z. Navabi, “VHDL, Analysis and Modeling of Digital Systems”, McGraw-Hill, 1999 اقتباس شده است.



شکل ۱-۲۳- پنجره‌ی سلسله مراتب پروژه



شکل ۱-۲٤- نمودار حالت تشخیص دهنده‌ی دنباله

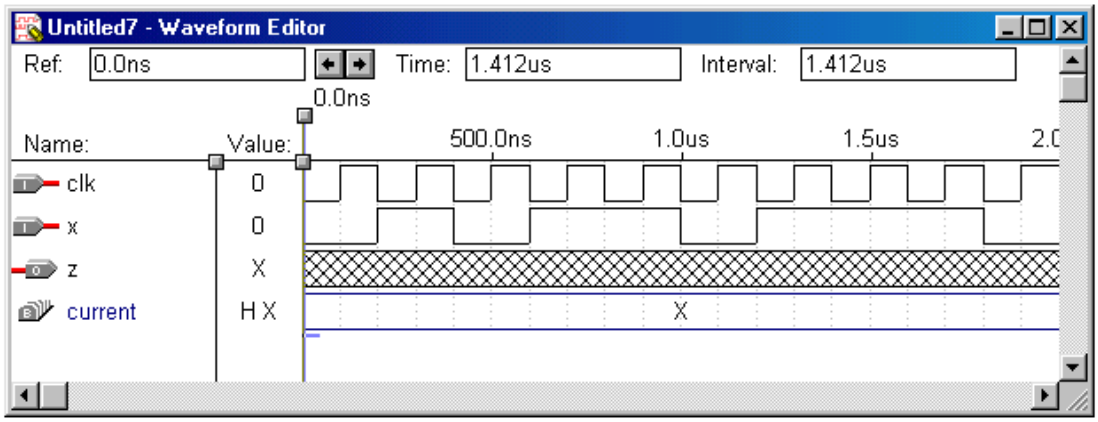
```
LIBRARY IEEE;
USE IEEE.std_logic_1164.all;
ENTITY detector IS
      PORT(
            x, clk : IN STD_LOGIC;
            z      : OUT STD_LOGIC);
END detector;

ARCHITECTURE behavioral OF detector IS
     TYPE state IS (reset, got1, got11, got110);
     SIGNAL current: state:=reset;
BEGIN
     PROCESS (clk)
     BEGIN
      IF clk='1' AND clk'EVENT THEN
            CASE current IS
                  WHEN reset =>
                      IF x='1' THEN
                        current <= got1;
                      ELSE
                        current <= reset;
                      END IF;
```

لیست ۱-۵- کد VHDL تشخیص دهنده‌ی دنباله

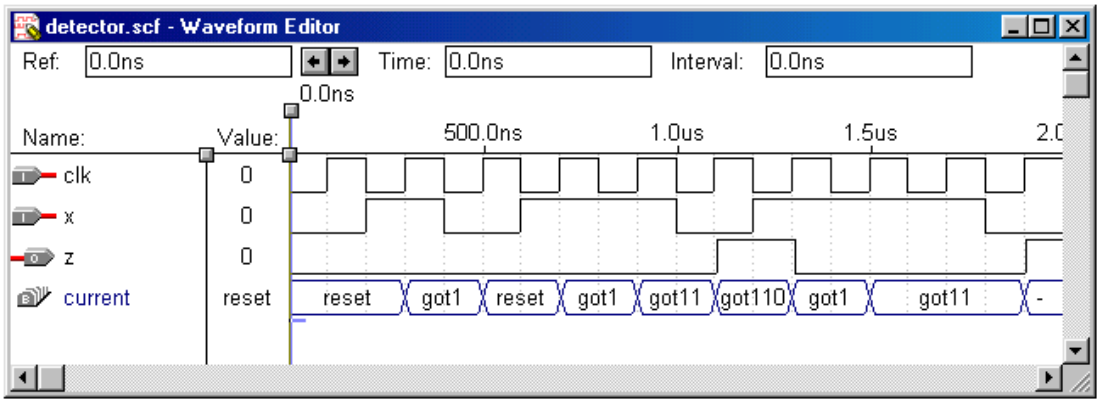
```
                    WHEN got1 =>
                        IF x='1' THEN
                          current <= got11;
                        ELSE
                          current <= reset;
                        END IF;
                    WHEN got11 =>
                        IF x='1' THEN
                          current <= got11;
                        ELSE
                          current <= got110;
                        END IF;
                    WHEN got110 =>
                        IF x='1' THEN
                          current <= got1;
                        ELSE
                          current <= reset;
                        END IF;
            END CASE;
      END IF;
     END PROCESS;
     z <= '1' WHEN current=got110 ELSE '0';
END behavioral;
```

لیست ۱-۵- کد VHDL تشخیص دهنده‌ی دنباله (ادامه)

مراحل ایجاد پروژه و فایل متنی را مانند قبل انجام می‌دهیم. برای اطمینان از صحت عمکرد این مدار، شکل موجی مطابق شکل ۱-۲۵ طراحی می‌کنیم. پس از انجام شبیه‌سازی خروجی مدار مطابق شکل ۱-۲۶ خواهد بود. همانطور که در شکل ۱-۲۶ مشاهده می‌شود، در یک State Machine می‌توان حالتی را که مدار در آن قرار دارد، مشاهده نمود. این امر در عیب‌یابی مدارها بسیار مفید است.



شکل ۱-۲۵- شکل موج ورودی تشخیص دهنده‌ی دنباله

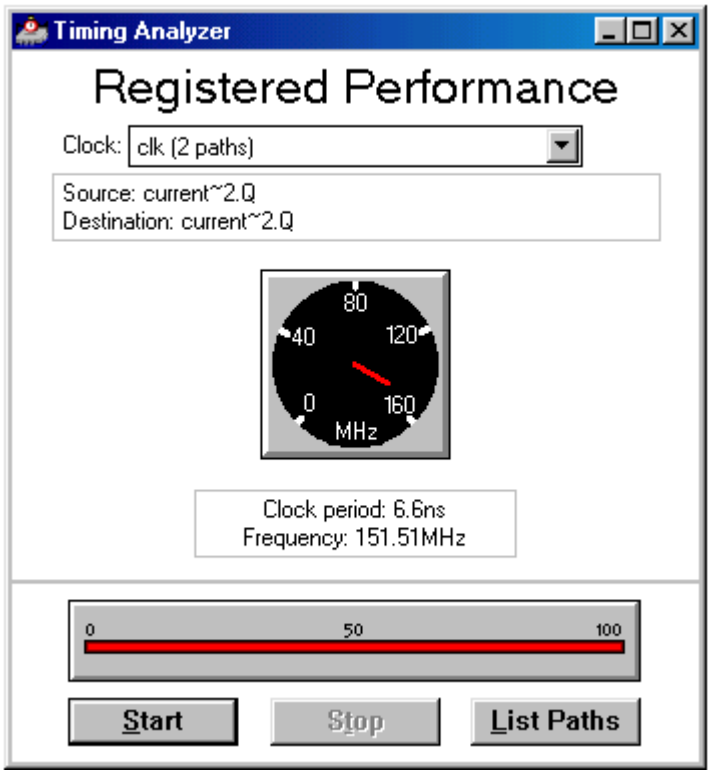


شکل ۱-۲۶- شکل موج خروجی تشخیص دهنده‌ی دنباله

برای اینکه ببینیم این مدار ترتیبی با چه فرکانسی کار می‌کند، به صورت زیر عمل می‌کنیم:

<M> Max+PlusII: Timing Analyzer, to show timing analyzer window
<M> Analysis: Registered Performance
<LB> Start

همانطور که در شکل ۱-۲۷ دیده می‌شود، در اینصورت حداکثر فرکانس کاری مدار بدست می‌آید.



شکل ۱ -۲۷- پنجره‌ی تحلیلگر زمانی (اندازه‌گیری فرکانس کاری مدار)

مثال۲: طراحی یک ALU هشت بیتی

یک ALU هشت بیتی طراحی کنید که دارای جدول صحت زیر باشد.

Opcode	Operation
00	Result <= a + b
01	Result <= a – b
10	If a=b then Result=1 Else E=0
11	If a<>b then Result=1 Else E=0

کد VHDL مربوطه در لیست ۱-٦ آمده است. کلیه‌ی مراحل طراحی مانند قبل انجام می‌شود.

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
ENTITY alu IS
      PORT (
            a, b  : IN std_logic_vector(7 DOWNTO 0);
            opcode: IN std_logic_vector(1 DOWNTO 0);
            result: OUT std_logic_vector(7 DOWNTO 0) );
END alu;

ARCHITECTURE behavioral OF alu IS
   CONSTANT plus  : std_logic_vector(1 DOWNTO 0) := "00";
   CONSTANT minus : std_logic_vector(1 DOWNTO 0) := "01";
   CONSTANT equal : std_logic_vector(1 DOWNTO 0) := "10";
   CONSTANT not_eq: std_logic_vector(1 DOWNTO 0) := "11";

BEGIN
      PROCESS (opcode)
      BEGIN
            CASE opcode IS
                  WHEN plus =>
                        result <= a + b;        -- add
                  WHEN minus =>
                        result <= a - b;        -- subtract
                  WHEN equal =>                 -- equal
                              IF (a = b) THEN
                                    esult <= "00000001";
                              ELSE
                                    result <= "00000000";
                              END IF;
                  WHEN not_eq =>                -- not equal
                              IF (a /= b) THEN
                                    result <= "00000001";
                              ELSE
                                    result <= "00000000";
                              END IF;
                  WHEN OTHERS =>
                              NULL;
            END CASE;
      END PROCESS;
END behavioral;
```

لیست ۱-٦- کد VHDL مربوط به ALU هشت بیتی

۲ طراحی مدار دیجیتال بصورت شماتیک

در این بخش ابتدا یک سلول Full Adder یک بیتی را طراحی می‌کنیم، سپس با استفاده از اتصال این سلولها به یکدیگر یک جمع کننده‌ی چهار بیتی را می‌سازیم. برای این منظور ابتدا برنامه‌ی Max+Plus II را اجرا می‌کنیم. سپس یک پروژه‌ی جدید ایجاد می‌کنیم. در اینجا دایرکتوری max2work و نام پروژه را adder انتخاب می‌کنیم.

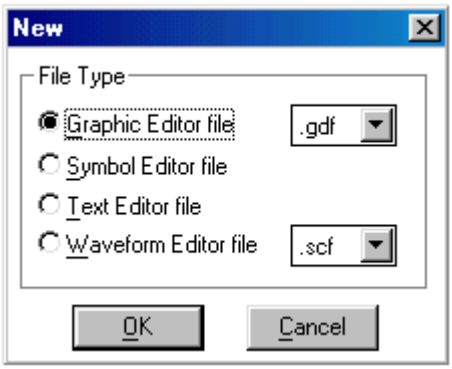
۲-۱ ایجاد فایل جدید

پس از اینکه نام پروژه را مشخص کردیم، باید طرحهایی را که در این پروژه مورد استفاده قرار می‌گیرد، به این پروژه اضافه کنیم. برای این امر فایلهای جدید را ایجاد می‌کنیم. در اینجا ابتدا یک سلول جمع کننده‌ی یک بیتی را ساخته و سپس با استفاده از آن یک جمع کننده‌ی چهار بیتی را می‌سازیم. باید دقت داشت که همیشه باید نام پروژه با نام مدار در بالاترین سطح یکسان باشد.

<M> File: New

پس از این امر پنجره‌ی ایجاد فایل جدید مطابق شکل ۲-۱ بر روی صفحه نمایش داده می‌شود. حال بصورت زیر عمل می‌کنیم :

Select Graphic Editor file
Select .gdf filename extension from combo box
<LB> on Ok



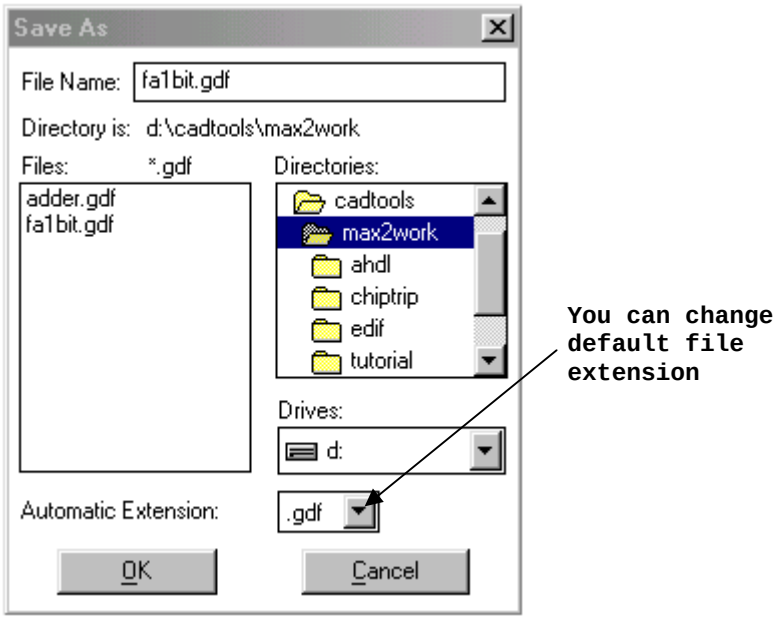
شکل ۲-۱- پنجره‌ی ایجاد فایل جدید

با انجام این عملیات پنجره‌ی ویرایشگر فایل شماتیک بر روی صفحه ظاهر می‌شود. حال باید این فایل جدید را ذخیره نمود. برای این منظور بصورت زیر عمل می‌کنیم :

<M> File: Save As

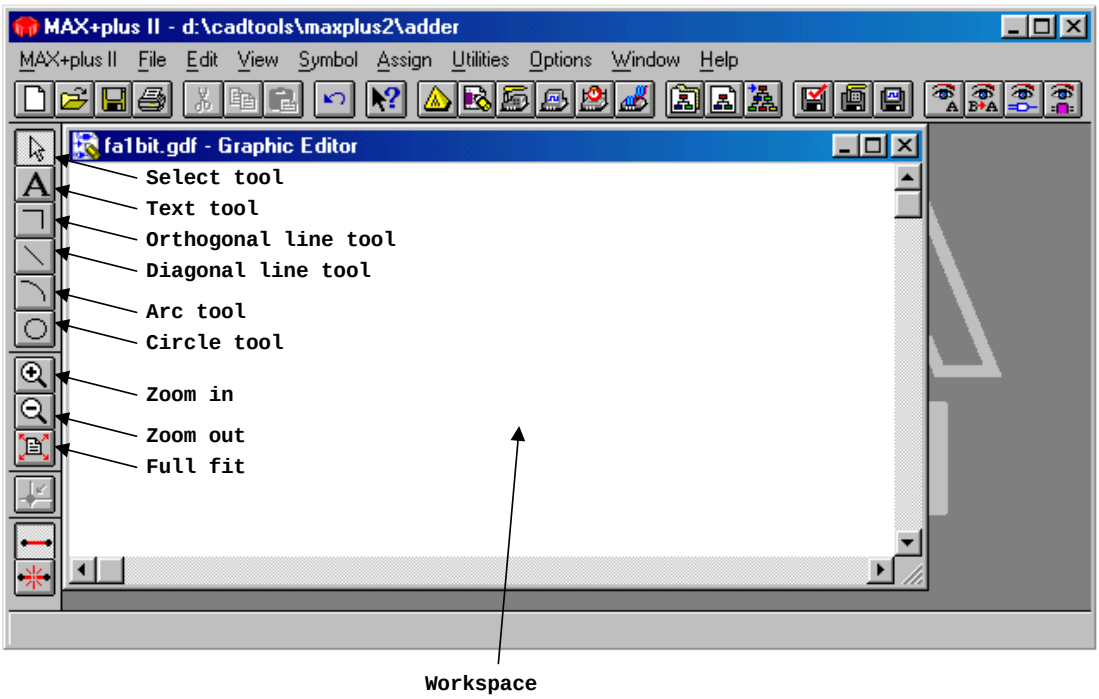
حال در پنجره‌ی ظاهر شده نام fa1bit را بعنوان نام فایل وارد می‌کنیم (شکل ۲-۲ را ببینید).

<LB> on Ok



شکل ۲-۲- پنجره‌ی ذخیره ی فایل جدید

در شکل ۲-۳ پنجره‌ی ویرایشگر فایل شماتیک و بخشهای مختلف آن نمایش داده شده است.



شکل ۲-۳- پنجره‌ی ویرایشگر فایل شماتیک

۲-۲ افزودن نمادها

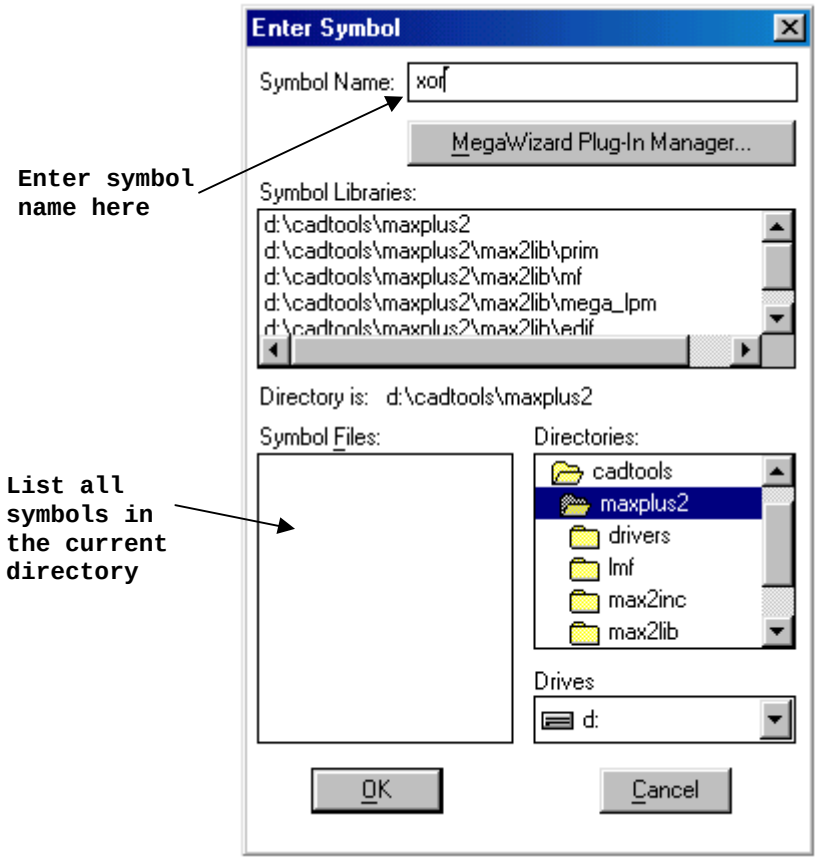
حال برای طراحی جمع کننده‌ی یک بیتی ابتدا باید دو گیت XOR و دو گیت AND و یک گیت OR را به ناحیه ی کاری اضافه کنیم. برای افزودن یک نماد (انواع گیتها، عناصر از پیش تعریف شده، عناصر تعریف شده توسط کاربر، پورتهای ورودی و خروجی و ..) بصورت زیر عمل می‌کنیم :

<DL> in an empty space in the Workspace

پنجره‌ی ورود نماد مطابق شکل ۲-٤ ظاهر می‌شود. برای ورود یک عنصر در محل Symbol Name نام نماد مورد نظر، در اینحالت XOR را وارد می‌کنیم.

<LB> on Ok

همین عملیات را برای چهار گیت دیگر (یک گیت XOR ، دو گیت AND2 و یک گیت OR2) تکرار می‌کنیم (دقت کنید برای گیتهای پایه بجز XOR تعداد ورودیهای گیت به دنبال نام گیت می‌آید برای مثال AND2 یک گیت AND دو ورودی است). در اینحالت مدار بصورت شکل ۲-۵ در می‌آید.



شکل ۲-٤- پنجره‌ی ورود نماد

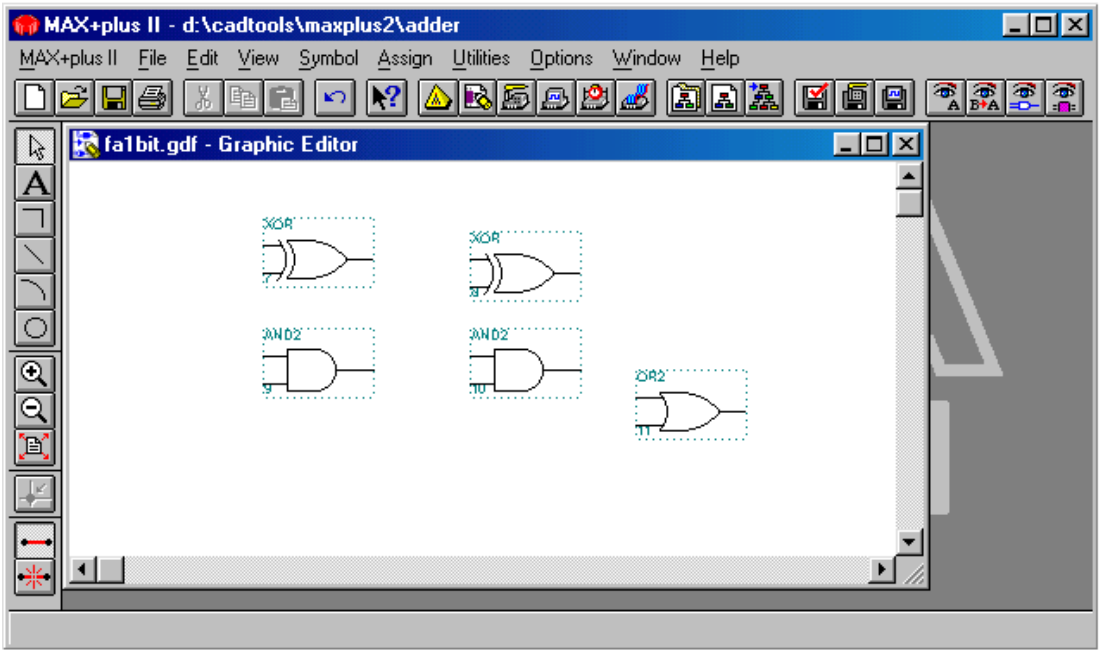
برای حرکت دادن یک نماد کافیست ابتدا در پنجره‌ی ویرایشگر فایل شماتیک روی آیکون Select با کلید سمت چپ ماوس کلیک کرده سپس در حالیکه کلید ماوس همچنان پایین است، آن را به محل مورد نظر برده و کلید ماوس را رها می‌کنیم. به این ترتیب نماد به محل مورد نظر انتقال داده می‌شود. برای اینکه یک کپی دیگر از یک نماد ایجاد کنیم، کافیست در حالیکه کلید Ctrl را زده ایم، عملیات فوق را انجام دهیم. در اینصورت یک کپی از نماد در محل جدید ایجاد می‌شود.

پس از افزودن گیتها نوبت به افزودن پورتهای ورودی و خروجی است. برای این منظور بصورت زیر عمل می‌کنیم :

<DL> in an empty space in the Workspace
In the Enter Symbol window, Enter input in the Symbol Name box

<LB> on Ok

همین عملیات را برای دو پورت ورودی دیگر و برای دو پورت خروجی انجام می‌دهیم (برای پورتهای خروجی باید نام نماد را بصورت output وارد کنیم).



شکل ۲-۵- پنجره‌ی ویرایشگر فایل شماتیک پس از افزودن نمادها

۲-۳ نامگذاری پورتهای ورودی و خروجی

پس از افزودن پورتهای ورودی و خروجی باید این پورتها را نامگذاری کنیم. در حالت عادی هر پورت بصورت <symbol name>:<symbol ID> مشخص می‌شود، برای مثال INPUT:2 نشان می‌دهد که این پورت، یک پورت ورودی با شماره شناسه ی ۲ است (شماره ی شناسه نشاندهنده ی این است که نماد وارد شده چندمین نمادی است که در ویرایشگر شماتیک افزوده شده است). برای اینکه به یک پورت ورودی/خروجی نامی‌را نسبت دهیم بصورت زیر عمل می‌کنیم :

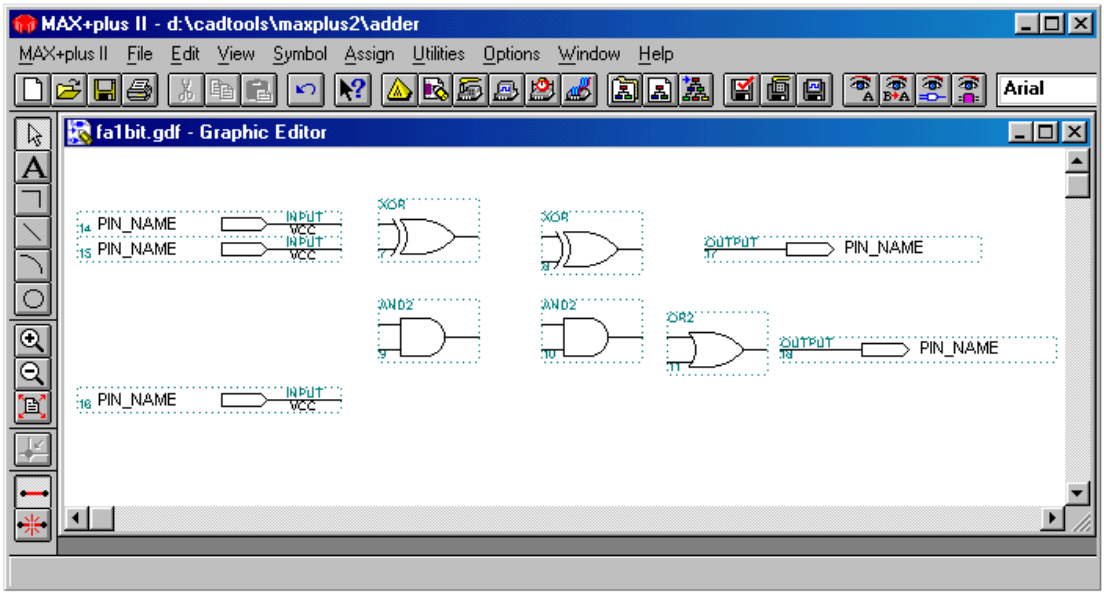
Point to the default pin name PIN_NAME of an input/output port
<DL> to select the entire name
Enter desired name (in this example x) to change the port name
Press Enter to select the next port name below it is automatically selected for editing

پس از انجام این عملیات شکل مدار بصورت شکل ۲-۶ در خواهد آمد.

۲-۴ اتصال نمادها به یکدیگر

پس از اینکه نمادها و پورتها را به فایل شماتیک افزودیم و پورتها را نامگذاری کردیم، باید نمادها را به نحو مناسب به یکدیگر متصل کنیم. برای این منظور از سیم و گذرگاه استفاده می‌کنیم. برای اتصال دو نماد از طریق سیم به یکدیگر بصورت زیر عمل می‌کنیم :

<LB> on Orthogonal line tool
<LB> on the output of the x input port and drag the wire to the input of the XOR gate and then release the left button of the mouse



شکل ۲-٦- پنجره‌ی ویرایشگر فایل شماتیک پس از نامگذاری پورتها

به همین ترتیب سایر سیمها را رسم می‌کنیم تا مدار بصورت شکل ۲-۷ در آید. سپس فایل fa1bit را ذخیره می‌کنیم.

<M> File: Save

نحوه ی رسم گذرگاه نیز دقیقا مانند سیم است، با این تفاوت که پس از اینکه سیم را رسم نمودیم، بصورت زیر عمل می‌کنیم :

<LB> on the wire to select it
<RB> on the selected wire
In the case sensitive menu, Select Line Style, and then select the second item (thick line)
The selected wire changes to the bus

۲-۵ ایجاد نماد از روی طرح

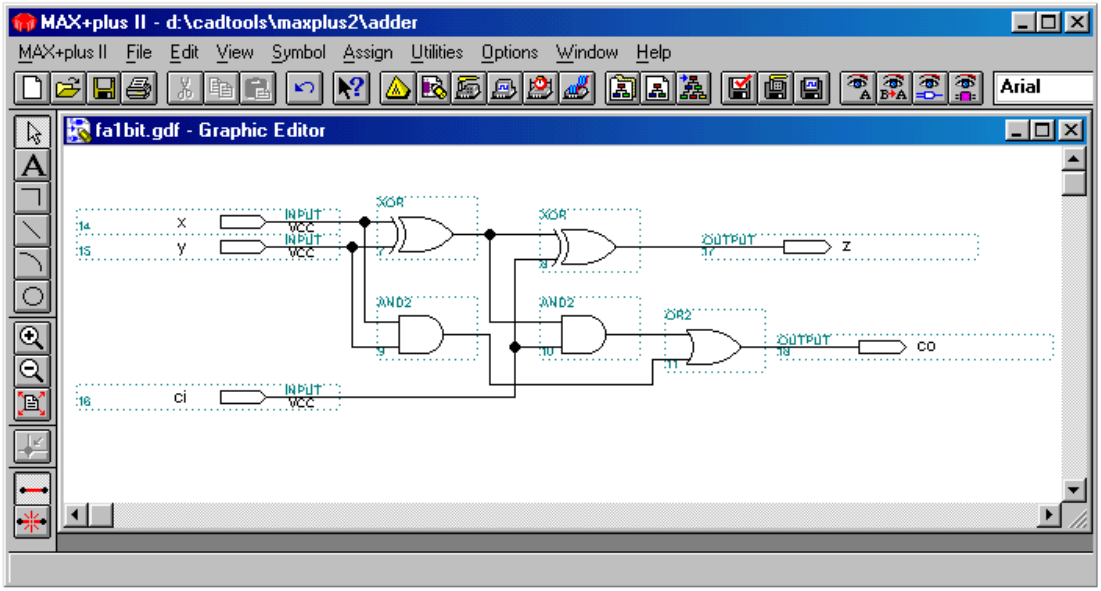
پس از اینکه جمع کننده ی یک بیتی را ذخیره کردیم باید از روی آن یک نماد ایجاد کنیم، تا در آینده بتوان بعنوان یک نماد از آن استفاده نمود. برای این منظور بصورت زیر عمل می‌کنیم :

<M> File: Create Default Symbol

به این صورت یک نماد به نام fa1bit ایجاد می‌شود که می‌توانیم از آن در محیط شماتیک استفاده کنیم (البته بهتر است پیش از ایجاد یک نماد از روی یک طرح، از صحت آن مطمئن شویم).

اگر بخواهیم این نماد در یک فایل متنی AHDL مورد استفاده قرار بگیرد باید به ترتیب زیر عمل کنیم:

<M> File: Create Default Include File



شکل ۲-۷- پنجره‌ی ویرایشگر فایل شماتیک پس از اتصال نمادها

۲-۶ ایجاد یک جمع کننده چهار بیتی

اکنون باید یک مدار دیگر ایجاد کنیم که با اتصال چهار نماد fa1bit به یکدیگر، یک جمع‌کننده‌ی چهار بیتی بدست بیاید. برای این منظور باید یک فایل شماتیک جدید ایجاد کنیم. ولی همانطورکه قبلا اشاره شد نام مدار در بالاترین سطح باید با نام پروژه یکسان باشد. از اینرو یک فایل شماتیک با نام adder ایجاد می‌کنیم.

<M> File: New, to show New File window
Select Graphic Editor file
Select .gdf filename extension from combo box
<LB> on Ok, to show Graphic Editor window
<M> File: Save As, to show Save As window
Enter the name of the top level design, (in this example adder)
<LB> on Ok

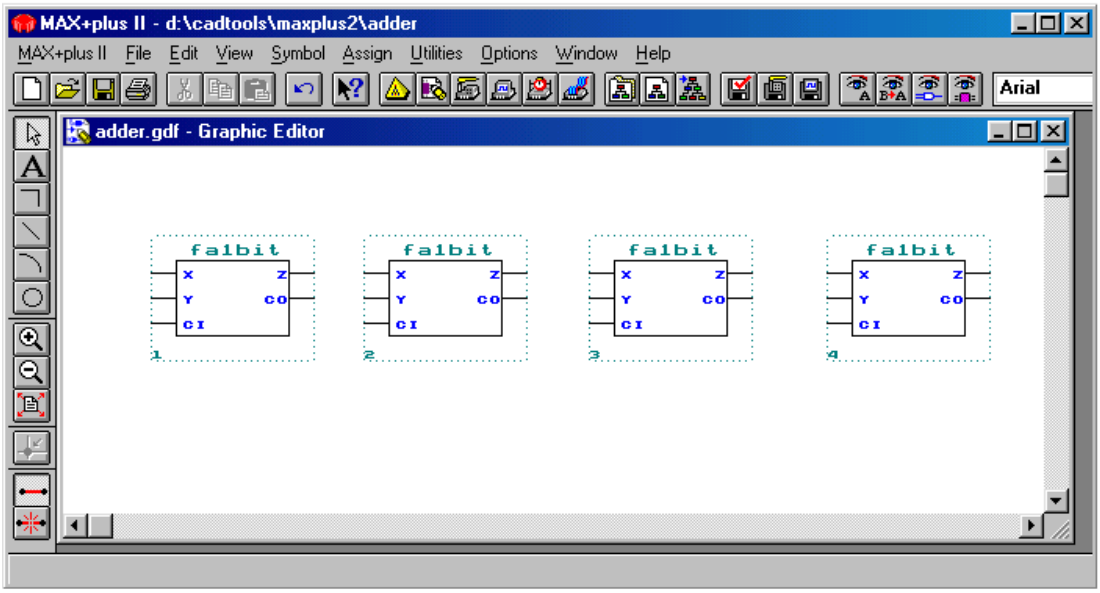
سپس در این فایل جدید چهار نماد fa1bit اضافه می‌کنیم (شکل ۲-۸ را ببینید).

۲-۷ اتصال نمادها به یکدیگر از طریق نام

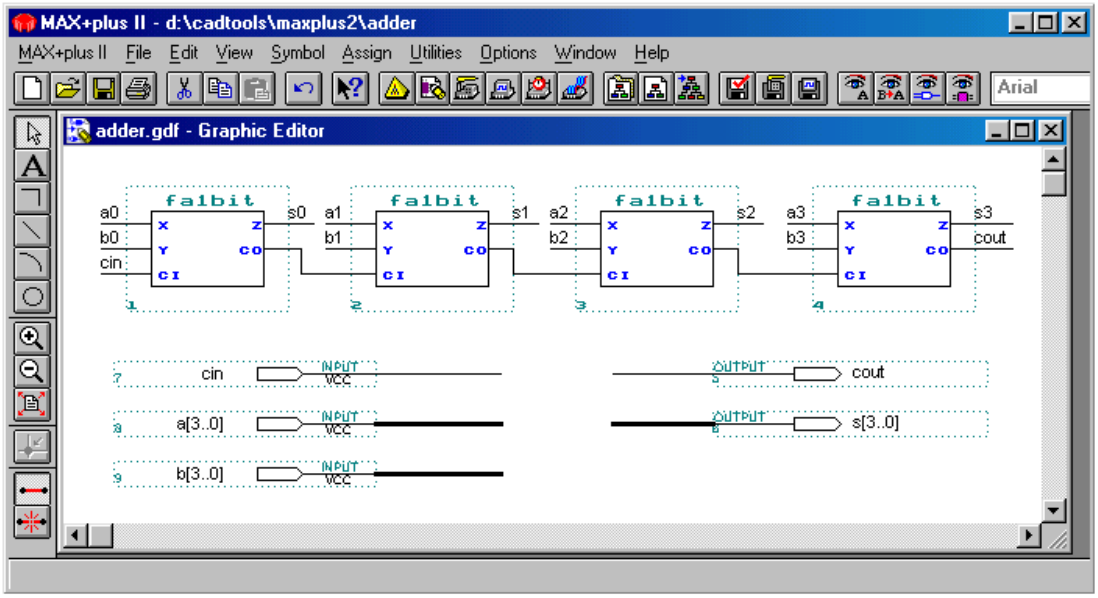
در بخش ۲-۶ نحوه ی استفاده ازسیم برای اتصال دو نماد به یکدیگر را دیدیم. در این بخش می‌خواهیم ببینیم چگونه می‌توان برای اتصال نماده به یکدیگر از نام استفاده نمود. در ویرایشگر شماتیک چنانچه دو سیم دارای نام یکسان باشند، به یکدیگر متصل هستند. دقت کنید این امر فقط در مورد سیم و گذرگاه صادق است، از اینرو باید در ورودی و خروجی نمادها یک قطعه سیم یا گذرگاه رسم نمود، سپس به آنها نام اختصاص داد. برای اختصاص دادن نام به یک قطعه سیم بصورت زیر عمل می‌کنیم :

<LB> on the wire to select it, a small blinking square appears below the line, to show the insertion point
Enter the name of the wire, the name appears above the line Press the Enter key, to assign the name to the wire

در شکل ۲-۹ همین امر در مورد مثال جمع کننده به نمایش در آمده است. دقت کنید که خطوط پررنگ گذرگاه هستند. به هنگام نامگذاری گذرگاه باید از روش نامگذاری گروهی استفاده نمود. پس از آن فایل را ذخیره می‌کنیم.



شکل ۲-۸- پنجره‌ی ویرایشگر فایل شماتیک جمع کننده ی چهار بیتی



شکل ۲-۹- پنجره‌ی ویرایشگر فایل شماتیک جمع کننده ی چهار بیتی پس از اتصال از طریق از نام

۲-۸ انتخاب تراشه

پس از اینکه طرح آماده شد، باید آن را کامپایل کنیم. برای این منظور نخست باید نوع تراشه ای که می‌خواهیم طرح ما روی آن نگاشت شود را مشخص کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> Assign: Device, to show Device window

در پنجره‌ی انتخاب تراشه، نخست باید از لیست Device Family خانواده‌ی تراشه را انتخاب نمود. در اینصورت در لیست Devices نام تراشه های موجود در این خانواده به نمایش در می‌آید. در صورتیکه بخواهیم ابزار سنتز بهترین تراشه را برای طرح ما انتخاب کند، باید در لیست Devices گزینه‌ی AUTO را انتخاب نمود.

۹-۲ کامپایل طرح

پس از انتخاب نوع تراشه باید طرح را کامپایل نمود. برای این منظور باید بصورت زیر عمل نمود :

<M> Max+PlusII: Compiler

برای شروع عملیات کامپایل توسط ماوس روی کلید Start کلیک می‌کنیم. در طی انجام عملیات کامپایل گزارشی از خطاها و warning های احتمالی در یک پنجره بنام Message به نمایش در می‌آید. چنانچه بخواهیم محل بروز خطا را مشاهده کنیم، کافیست توسط ماوس روی خطای مورد نظر کلیک کرده و سپس روی کلید Locate کلیک کنیم. در صورتیکه طرح شما درون یک تراشه جای نگیرد، ابزار سنتز به شما پیام می‌دهد که آیا می‌خواهید این طرح را درون دو (یا چند) تراشه جای دهید یا خیر. در صورتیکه شما ادامه عمل کامپایل را تایید کنید، ابزار سنتز طرح را به چند بخش تقسیم می‌کند و هر یک را درون یک تراشه جای می‌دهد و لیستی از پایه‌های تراشه‌ها را که باید به یکدیگر مرتبط شوند ارائه می‌کند.

۱۰-۲ شبیه سازی طرح

پس از اینکه طرح مورد نظر روی تراشه نگاشت شد باید آن را شبیه سازی نمود تا از صحت طرح مطمئن شویم. برای این منظور نخست باید شکل موج ورودی مدار را طراحی کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

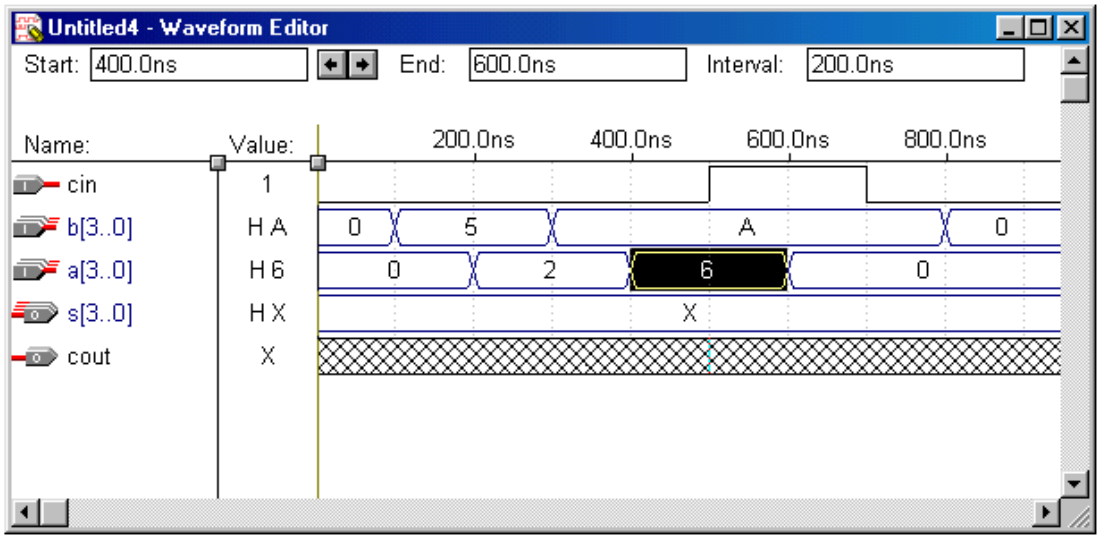
<M> Max+Plus II: Waveform Editor

در اینحالت پنجره‌ی ویرایش شکل موج روی صفحه ظاهر می‌شود. در مرحله ی کامپایل طرح، یک فایل با پسوند SNF ایجاد می‌شود که مشخصات تمام گره های مدار را نگاه می‌دارد. از اینرو بهتر است که نام گره های ورودی و خروجی مدار را از این فایل استخراج کنیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> Node: Enter Node from SNF

در این پنجره می‌توان نوع گره های مورد نظر را انتخاب نمود، این امر در بخش Type انجام می‌شود. پس از انتخاب نوع گره ها کلید List را می‌زنیم تا لیست گره های موجود در بخش سمت چپ پنجره ظاهر شود. سپس هر یک از ورودیها و یا خروجیها را که بخواهیم انتخاب می‌کنیم و آنها را توسط کلید => به لیست شکل موجها اضافه می‌کنیم. و در پایان کلید Ok را می‌زنیم.

پس از اینکه گره های مورد نظر به ویرایشگر شکل موج افزوده شد، باید شکل موج مورد نظر را به ورودیها اعمال نمود.



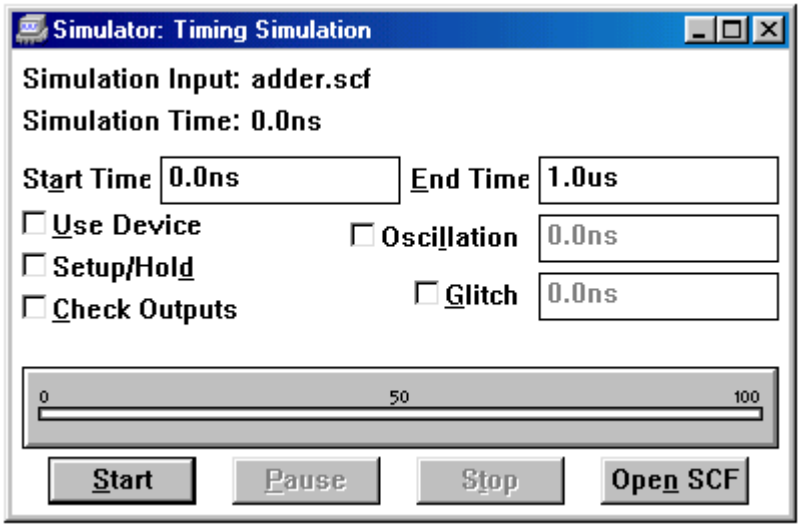
شکل ۲-۱۰- پنجره‌ی ویرایش شکل موجها پس از طراحی شکل موجها

در صورتیکه بخواهیم یک سیگنال را بصورت کامل انتخاب کنیم، باید روی مقدار آن سیگنال <LB> کنیم. برای تغییر دادن مقدار یک سیگنال به این نکته توجه کنید که سیگنال یک گره است یا یک گروه و برای مقداردهی به آن از آیکون مناسب استفاده کنید. برای تست این مدار فرض کنید شکل موج را بصورت شکل ۲-۱۰ طراحی کرده‌ایم. سپس بصورت زیر این شکل موج را با نام adder ذخیره می‌کنیم.

<M> File: Save

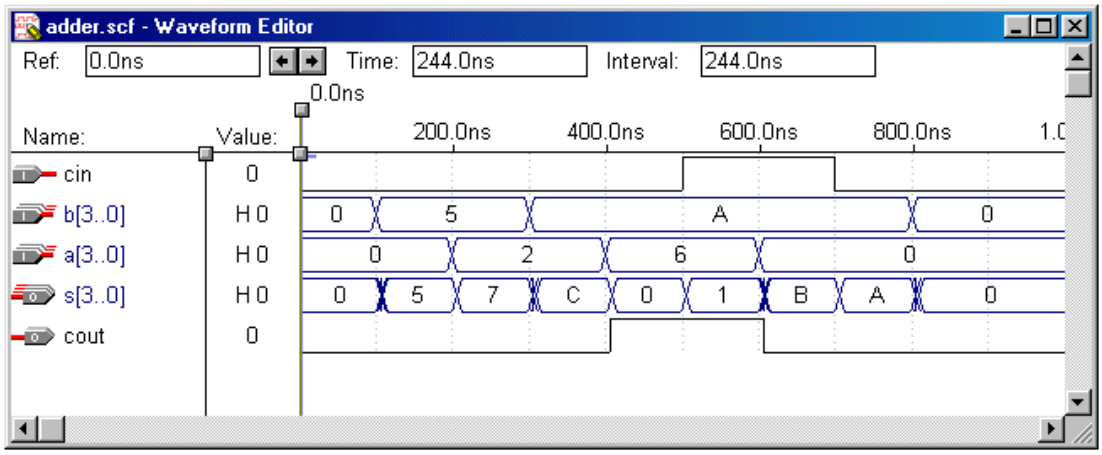
اکنون برای اینکه شبیه سازی این طرح را مشاهده کنیم، باید عمل شبیه سازی را انجام دهیم. برای این منظور بصورت زیر عمل می‌کنیم :

<M> Max+PlusII: Simulator, to show the simulator window (fig 2-11)
<LB> on Start, to simulate the design
<LB> on Open SCF



شکل ۲-۱۱- پنجره‌ی شبیه سازی

با انجام این عمل مجددا پنجره‌ی شکل موجها باز می‌شود با این تفاوت که در این حالت مقادیر خروجیها نیز مشخص شده اند. حال می‌توان صحت عملکرد مدار را بررسی نمود (شکل ۲-۱۲ را ببینید)



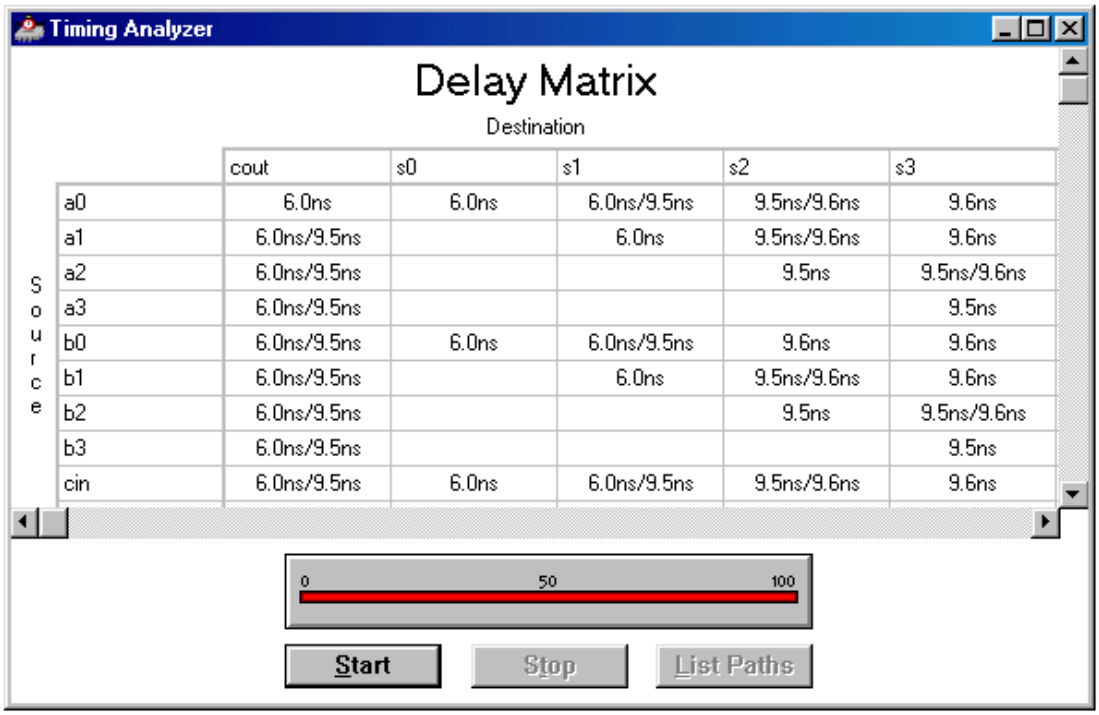
شکل ۲-۱۲- پنجره‌ی شکل موجها

۲-۱۱ تحلیگر زمانی

برای بررسی تاخیر مدار بصورت زیر عمل می‌کنیم.

<M> Max+PlusII: Timing Analyzer, to show timing analyzer window
<LB> Start

در اینحالت در پنجره‌ی تحلیگر زمانی تاخیر ترکیبی ورودیها به خروجیها را محاسبه کرده و در ماتریس تاخیر نمایش می‌دهد (شکل ۲-۱۳ را ببینید).



شکل ۲-۱۳- پنجره‌ی تحلیلگر زمانی

۱۲-۲ ویرایشگر تراشه

پس از اینکه از صحت عملکرد طرح مطمئن شدیم، باید نحوه‌ی اعمال ورودیها و خروجیها به تراشه را مشخص کنیم. این امر توسط ویرایشگر تراشه انجام می‌شود. برای اجرای ویرایشگر تراشه بصورت زیر عمل می‌کنیم.

<M> Max+PlusII: Floorplan Editor, to show floorplan editor window

در حالت عادی این پنجره تراشه را بصورت Logic Array Block (LAB) نمایش می‌دهد. برای اینکه تراشه را بصورت عادی نمایش دهد بصورت زیر عمل می‌کنیم.

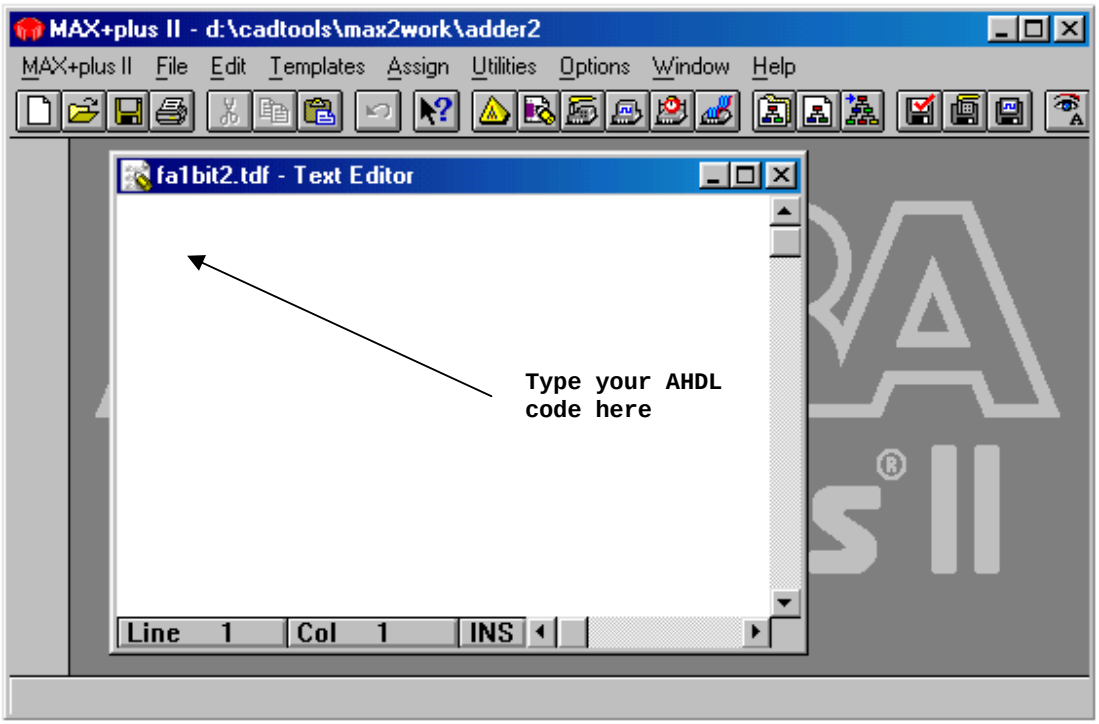
<M> Layout: Device View

در قسمت پایین صفحه هر یک از ورودیها و خروجیهای مدار به یکی از پایه های تراشه اختصاص یافته است. چنانچه این تخصیص که مربوط به کامپایل قبلی است، مورد نظر شما نباشد می‌توانید آن را تغییر دهید. برای این منظور باید توسط ماوس روی کلید Current Assignment کلیک کنیم، در اینحالت در قسمت پایین صفحه یک تراشه‌ی خالی نمایش داده می‌شود (توجه کنید این امر درصورتی امکان پذیر است که قبلا در مرحله ی کامپایل، نوع تراشه را انتخاب کرده باشید، بعبارت دیگر نوع تراشه AUTO نباشد) و در لیست بالای صفحه لیست گره های مدار نمایش داده می‌شود. حال می‌توانید هر یک از گره ها را از لیست انتخاب نموده و آن را Drag نموده روی پایه ی مورد نظر قرار دهید. دقت کنید که گره های ورودی را فقط به پایه های ورودی یا دو سویه و گره های خروجی را فقط به پایه های خروجی یا دو سویه و گره های دو سویه را فقط به پایه های دو سویه تخصیص دهید. نکته ی دیگری که باید به آن توجه داشته باشید این است که ممکن است با تخصیصی که شما انتخاب نموده اید ابزار سنتز نتواند طرح را درون تراشه قرار دهد، از اینرو پس از اینکه تخصیص خود را انتخاب نمودید باید مجددا طرح را کامپایل کنید. در صورتیکه عمل کامپایل موفقیت آمیز باشد، تخصیص شما امکانپذیر است (این امر به علت محدود بودن مسیرهای ارتباطی ایت که در تراشه موجود است) .

۳ طراحی مدار با استفاده از AHDL

برای یادگیری طراحی مدار با استفاده از زبان توصیف سخت افزاری AHDL همان مثال جمع کننده را مورد بررسی قرار می‌دهیم. در اینحالت نیز اگر پروژه ی ما دارای چند زیر طرح باشد، ابتدا هر یک از زیر طرحها را ایجاد می‌کنیم (با روش طراحی شماتیک یا با استفاده از زبان توصیف سخت افزاری) سپس از هر یک از زیر طرحها یک Default Include File ایجاد می‌کنیم. سپس در فایل اصلی که همنام پروژه است، تمام زیر طرحها را include می‌کنیم تا بتوانیم از آنها استفاده کنیم. در اینحالت نام پروژه را adder2 و نام جمع کننده ی تک بیتی را fa1bit2 و نام فایل جمع کننده ی چهار بیتی را adder2 قرار می‌دهیم.

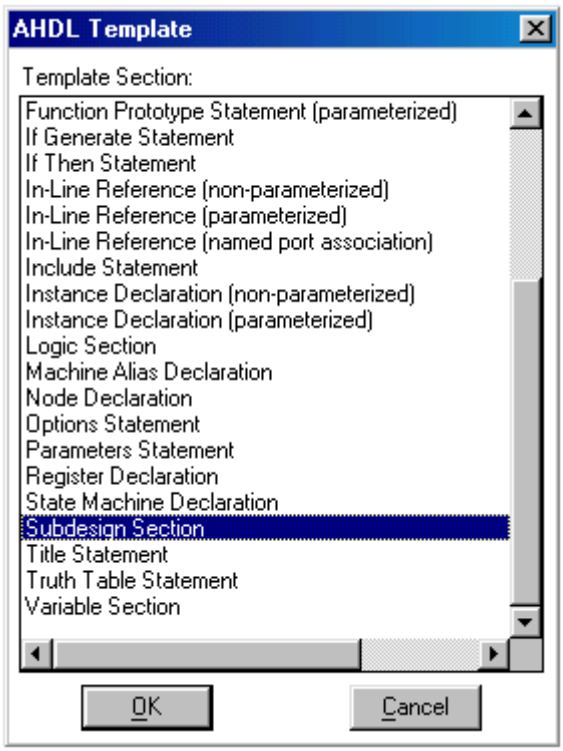
ابتدا یک پروژه ی جدید ایجاد می‌کنیم و آن را با نام adder2 ذخیره می‌کنیم. سپس یک فایل جدید متنی در این پروژه ایجاد می‌کنیم و آن را با نام fa1bit2 ذخیره می‌کنیم. دقت کنید که پسوند فایل متنی حتما بصورت tdf باشد (پسوند فایلهای AHDL بصورت tdf است). در اینحالت پنجره ویرایشگر متنی مطابق شکل ۳-۱ خواهد بود.



شکل ۳-۱- پنجره‌ی ویرایشگر متنی

برای وارد کردن کد، ابزار سنتز دارای یک قابلیت بسیار مناسب بنام Template است که الگویی از تمام ساختارهای زبان را در اختیار شما قرار می‌دهد. برای دسترسی به این قابلیت کافیست در حالیکه ماوس روی صفحه متنی در محل مورد نظر است، با کلید سمت راست ماوس کلیک کنیم و در منوی Case Sensitive ظاهر شده گزینه ی AHDL را انتخاب می‌کنیم. در اینحالت پنجره ای مطابق شکل ۳-۱ روی صفحه ظاهر می‌شود که می‌توان از آن الگوی مورد نظر را نتخاب نمود و کلید Ok را زد. در

اینصورت یک کپی از الگوی این ساختار در محل مورد ظاهر می‌شود و حالا می‌توان این الگو را بر اساس نیاز تغییر داد. این قابلیت کاربر را از به خاطر سپردن تمام ساختارهای زبان بی نیاز می‌سازد.



شکل ۳-۲- پنجره‌ی الگوی AHDL

حال در پنجره‌ی ویرایشگر متنی توصیف جمع‌کننده‌ی یک بیتی (لیست ۳-۱ را ببینید) را وارد می‌کنیم و آنرا با نام fa1bit2.tdf آن را ذخیره می‌کنیم.

```
SUBDESIGN fa1bit2
(
      _x, _y, _ci   : INPUT ;
      s, co         : OUTPUT;
)
BEGIN
   s  = ( _x XOR _y ) XOR _ci;
   co = ( _x AND _y ) OR ( _x AND _ci ) OR ( _y AND _ci );
END;
```

لیست ۳-۱- توصیف جمع‌کننده‌ی یک بیتی به زبان AHDL

سپس از روی این فایل یک Default Include File ایجاد می‌کنیم، برای ایجاد این فایل، ابزار سنتز نام پروژه را به fa1bit2 تغییر می‌دهد و سپس طرح را کامپایل می‌کند تا مطمئن شود که طرح دارای خطا نیست. حال مجددا پروژه adder2 را باز می‌کنیم. سپس یک فایل جدید متنی در این پروژه ایجاد می‌کنیم و آن را با نام adder2 ذخیره می‌کنیم. سپس در این فایل توصیف لیست ۳-۲ را وارد کرده و آن را ذخیره می‌کنیم. از این به بعد تمام مراحل مانند مراحل بخش اول است.

```
INCLUDE "fa1bit2";

SUBDESIGN adder2
(
      a[3..0], b[3..0], cin   : INPUT ;
      s[3..0], cout           : OUTPUT;
)
VARIABLE
   add1bit[3..0]        : fa1bit2;
   c[3..1]         : NODE;
BEGIN
   add1bit[0]._x  = a[0];
   add1bit[0]._y  = b[0];
   add1bit[0]._ci = cin;
   s[0] = add1bit[0].s;
   c[1] = add1bit[0].co;

   add1bit[1]._x  = a[1];
   add1bit[1]._y  = b[1];
   add1bit[1]._ci = c[1];
   s[1] = add1bit[1].s;
   c[2] = add1bit[1].co;

   add1bit[2]._x  = a[2];
   add1bit[2]._y  = b[2];
   add1bit[2]._ci = c[2];
   s[2] = add1bit[2].s;
   c[3] = add1bit[2].co;

   add1bit[3]._x  = a[3];
   add1bit[3]._y  = b[3];
   add1bit[3]._ci = c[3];
   s[3] = add1bit[3].s;
   cout = add1bit[3].co;

END;
```

لیست ۳-۱- توصیف جمع‌کننده‌ی یک بیتی به زبان AHDL