Chapter 29: Appendix

29.1 PSpice – Brief User's Guide

29.1.1 General

PSpice from *OrCAD* (previously *MicroSim*) is a circuit simulator from the *Spice* family (S*imulation* P*rogram with* I*ntegrated* C*ircuit* E*mphasis*) for the simulation of analog, digital, and mixed analog/digital circuits. In 1970 *Spice* was developed at Berkely University and is available today in the version 3F5 for use without licence. On this basis, commercial offshoots evolved which contain specific expansions and additional modules for entering circuits graphically, presenting results and controlling processes. The most common ones are *PSpice* and *HSpice*. While *HSpice* from *Synopsys* (previously *Meta Software*) was designed for developing integrated circuits comprising several thousand transistors and is used in many IC design systems as the simulator, *PSpice* is a particularly well-priced and easy to operate simulation environment for developing small and medium-sized circuits on PCs with a Microsoft Windows operating system.

The following short instructions apply to the demo version of *PSpice 8* (*PSpice Eval 8*) for *Microsoft Windows 98/ME/2000/XP*.

29.1.2 Programs and Files

Spice

Every simulator in the *Spice* family uses *netlists*. A netlist is a description of a circuit prepared by an editor which contains component lists and circuit topology data augmented by simulation instructions and references to model libraries. Fig. 29.1.1 shows the programs and files involved in the circuit simulation process.

- The netlist of the circuit to be simulated is prepared by an editor and stored in the circuit file <*name*>*.CIR* (*CIRcuit*).
- The circuit file is read in by the simulator (*PSpice or Spice 3F5*), which then performs the simulations according to the simulation instructions; this may include the use of models from component libraries <*xxx*>*.LIB* (*LIBrary*).
- The simulation results and (error) messages are stored in the output file <*name*>*.OUT* (*OUTput*) and can be displayed and printed with the aid of an editor.

PSpice

In addition to the simulator *PSpice,* the *PSpice* software package also contains a program for the graphic input of circuit diagrams (*Schematics*) and a program for the graphic representation of the simulation results (*Probe*). Figure 29.1.2 shows the process with the programs and files involved:

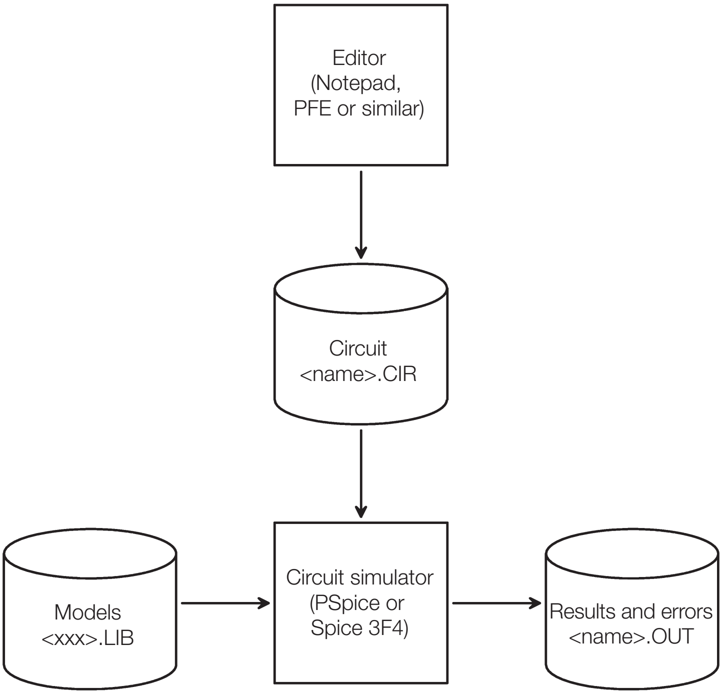


Fig. 29.1.1b. Programs and files used in *Spice*

- The *Schematics* program allows you to enter the circuit diagram for the circuit to be simulated and store it in the schematics file *<name>.SCH* (*SCHematic*); this process uses schematic symbols from symbol libraries *<xxx>.SLB* (*Schematic LiBrary*).
- By starting the simulation (*Analysis/Simulate*) or creating the netlist (*Analysis/Create Netlist*), the Schematics program generates the circuit file *<name>.CIR*; at the same time, the netlist is stored in the file *<name>.NET* and incorporated by an *Include* instruction. Another file *<name>.ALS* is generated which contains a list of alias names, but is of no relevance for the user.
- The *PSpice* simulator is activated in the Schematics program by starting the simulation (*Analysis/Simulate*); as an alternative *PSpice* can be started manually and the circuit file can be selected with *File/Open*. The simulation uses models from the component libraries *<xxx>.LIB*.
- The simulation results that may be presented graphically are stored in the data file *<name>.DAT*; nongraphic results and messages are stored in the output file *<name>.OUT* and can be displayed by selecting *Analysis/Examine Output* in the Schematics program or using an external editor.
- The simulation results can be presented graphically using the *Probe* program; this allows you to directly display individual signals or to perform calculations with one or more signals. The commands required to create a graph may be stored in the display file *<name>.PRB* using the *Options/Display Control* function and are available for later retrieval. If the simulation has been started via *Analysis/Simulate* in the *Schematics* program, the *Probe* program will start automatically upon completion of the simulation;

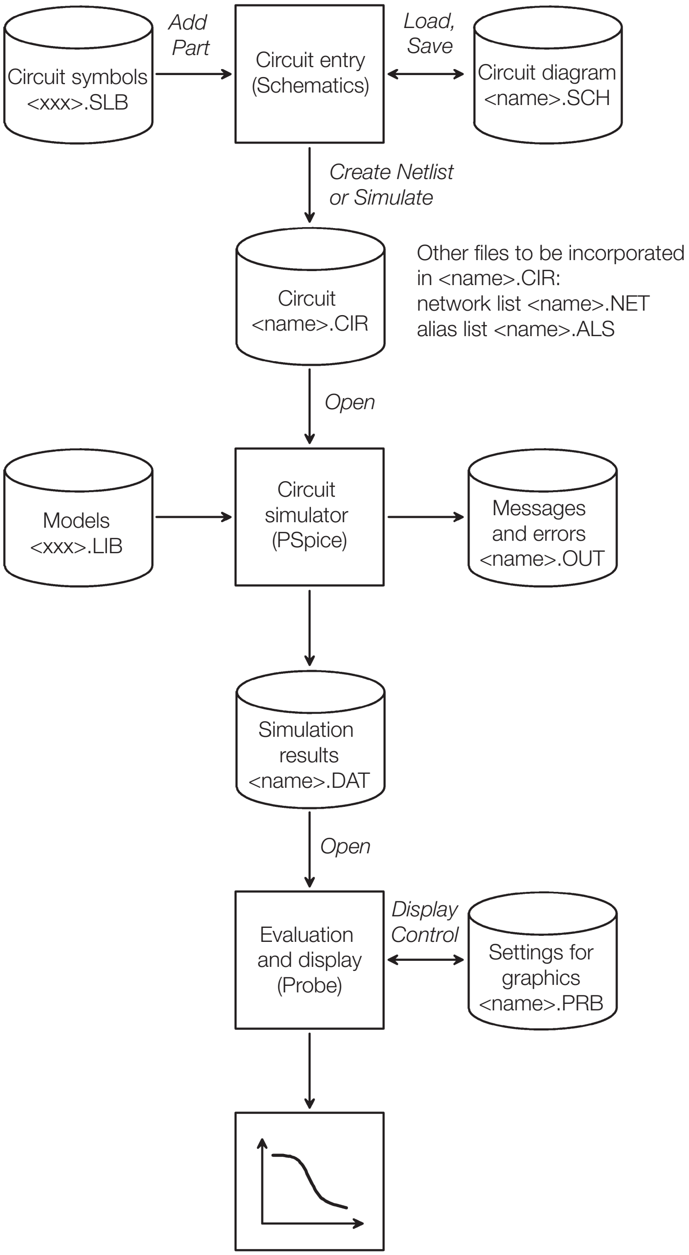


Fig. 29.1.2b. Programs and files used in *PSpice*

in this case the data file *<name>.DAT* is loaded automatically. If it has been started manually, the data file must be selected via *File/Open*.

With *PSpice* you can also work with netlists directly. To do so the circuit diagram should not be entered graphically; instead, the circuit file *<name>.CIR* should be created with the help of an editor. Unlike *Spice,* this offers the advantage of being able to represent the simulation results graphically using *Probe*. This procedure is often used when creating new models since the experienced user can eliminate any errors that may occur during model testing much faster in the circuit file than via the graphic diagram entry.

29.1.3 A Simple Example

A small-signal amplifier with AC coupling is taken as an example to illustrate how a circuit diagram is entered and circuit simulation is performed; Fig. 29.1.3 shows the corresponding circuit diagram.

Entering the Circuit Diagram

The *Schematics* program is activated in order for the circuit diagram to be entered; Fig. 29.1.4 shows the program window. The tool bar contains (from left to right) the *File* operations *New*, *Open*, *Save* and *Print*, the *Edit* operations *Cut*, *Copy*, *Paste*, *Undo* and *Redo*, and the *Draw* operations *Redraw*, *Zoom In*, *Zoom Out*, *Zoom Area* and *Zoom to Fit Page*. All are used in the usual way.

The diagram is entered by performing the following steps:

- Insert components
- Configure components
- Draw connecting wires

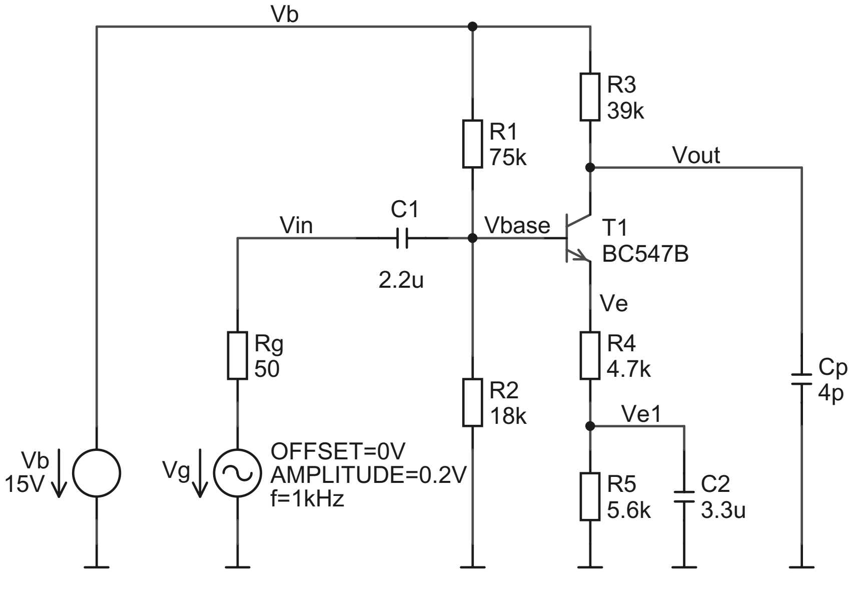


Fig. 29.1.3b. Circuit diagram for our example

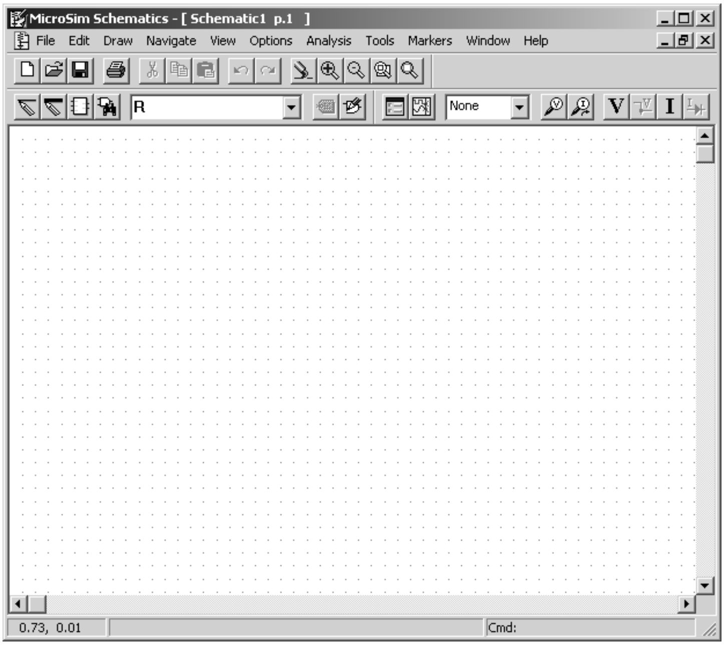
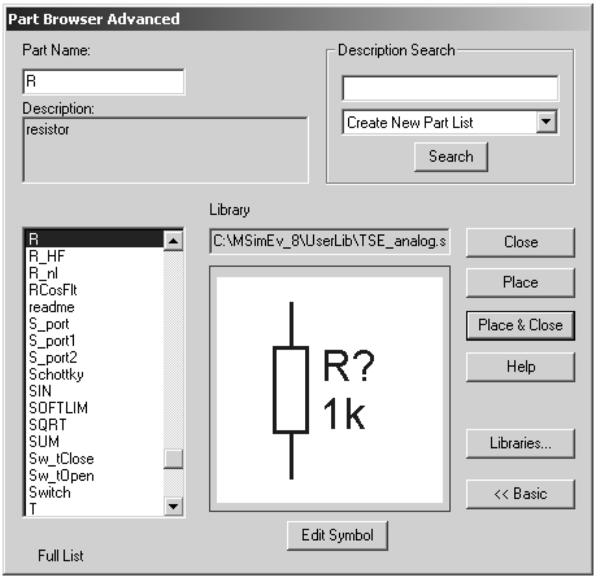


Fig. 29.1.4b. Program window of the *Schematics* program

This requires the following tools:

Step	Tool	Action
1	*Get New Part*	Inserts components
2	*Edit Attributes*	Configures the components
3	*Draw Wire*	Adds connecting wires
4	*Setup Analysis*	Enters the simulation instructions
5	*Simulate*	Starts the simulation



Part Browser Advanced

Part Name: R

Description: resistor

Library: C:\MSimEv_8\UserLib\TSE_analog.s

Fig. 29.1.5b. *Get New Part* dialog

Inserting the components: Use the *Get New Part* tool to open the dialog window *Part Browser Basic*; use *Advanced* to open the *Part Browser Advance* dialog window shown in Fig. 29.1.5. If the component name is known, it can be entered in the *Part Name* field; the component appears in the preview window and can be accepted by clicking on the *Place* or *Place & Close* button. If the name is not known, the component list must be searched to find the desired component. The *Libraries* button opens a dialog window containing the component list arranged according to libraries; here, however, a preview is not shown until a component has been selected and confirmed with *Ok*.

After accepting the component with the *Place* or *Place & Close* button the part is inserted into the diagram by clicking the left mouse button. Prior to insertion the symbol of the component can be rotated using *Ctrl-R* and/or mirrored using *Ctrl-F*. The insert mode continues until you have clicked the right mouse button or pressed the Esc key.

The names of some important passive and active components are listed in the table on page 1437

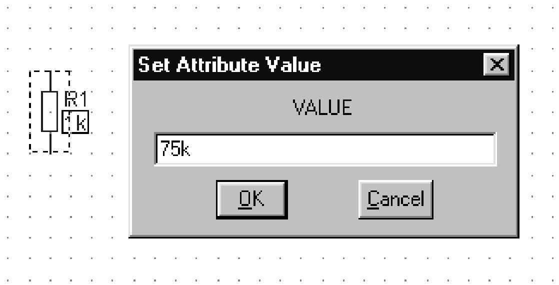
Configuring the components: Most components have to be configured following insertion. In the case of passive components such as resistors, capacitors and inductors this means entering the value; in the case of voltage and current sources this involves entering the parameters of the signal shape (amplitude, frequency, etc.) and for controlled sources the control ratio must be entered. For integrated transistors the size of the transistor and the name of the substrate node must be entered while for operational amplifiers parameters like transit frequency and slew-rate are needed. Some components such as standard transistors (e.g. BC547B) need not be configured since they are assigned a reference to a model in a model library which contains all the data required.

The value of a passive component can be changed by double clicking on the indicated value; this opens the *Set Attribute Value* option for entering the value (see Fig. 29.1.6).

Clicking on *Edit Attributes* or double clicking the component symbol opens the *Part* dialog window shown in Fig. 29.1.7 which contains a list of all parameters. Parameters not

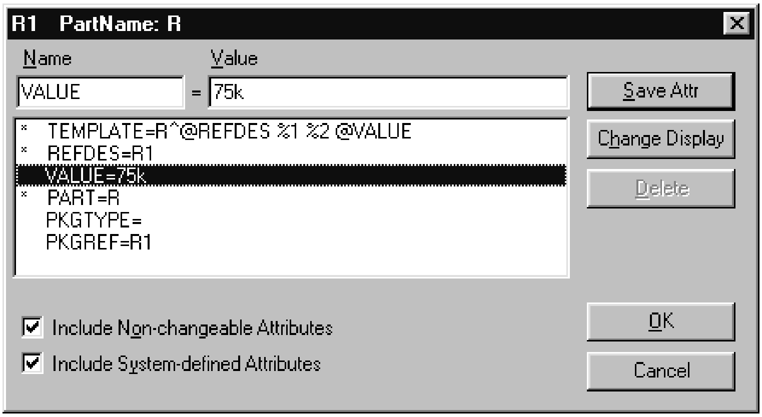
The names of some important passive and active components:

Name	Description	Library
R C L K E F G H Tr V V_AC V_bias V_pulse V_sine V_square V_triangle I I_bias GND	Resistor Capacitor Inductor Inductive coupling Voltage-controlled voltage source Current-controlled current source Voltage-controlled current source Current-controlled voltage source Ideal transformer General purpose voltage source Small-signal voltage source DC voltage source Large-signal pulse voltage source Large-signal sine-wave voltage source Large-signal square-wave voltage source Large-signal triangle voltage source General purpose current source DC current source Ground	TSE_ANALOG.SLB
1N4148 1N4001 BAS40 BC547B BC557B BD239 BD240	Small-signal diode 1N4148 (100mA) Rectifier diode 1N4001 (1A) Small-signal Schottky diode BAS40 npn small-signal transistor BC547B pnp small-signal transistor BC557B npn power transistor BD239 pnp power transistor BD240	TSE_BIPOLAR.SLB
BF245B IRF142 IRF9142	n-channel JFET BF245B n-channel power MOSFET IRF142 p-channel power MOSFET IRF9142	TSE_FET.SLB
N1 P1 NMOS PMOS	Integrated npn transistor Integrated pnp transistor Integrated n-channel MOSFET Integrated p-channel MOSFET	TSE_INTEGRATED.SLB
VV VC CV	Operational amplifier Transconductance operational amplifier Current-feedback operational amplifier	TSE_MODEL.SLB



75k

Fig. 29.1.6b. Dialog *Set Attribute Value* dialog



R1 PartName: R

Name: VALUE = Value: 75k

```
* TEMPLATE=R^@REFDES %1 %2 @VALUE
* REFDES=R1
  VALUE=75k
* PART=R
  PKGTYPE=
  PKGREF=R1
```

☑ Include Non-changeable Attributes

☑ Include System-defined Attributes

Fig. 29.1.7b. *Part* dialog

marked by an asterisk can be selected, modified in the *Value* field and saved by pressing the *Save Attr* button. Pressing *Change Display* allows you to ascertain whether or not and how the selected parameters are presented in the circuit diagram; usually only the value, e.g. $1k$, or the parameter name and the value, e.g. $R = 1k$, is displayed.

Numeric values can be entered as an exponential expression (e.g. *1.5E-3*) or can be provided with the following suffixes:

Suffix	f	p	n	u	m	k	Mega	G	T
Name	Femto	Pico	Nano	Micro	Milli	Kilo	Mega	Giga	Terra
Value	10^{-15}	10^{-12}	10^{-9}	10^{-6}	10^{-3}	10^{3}	10^{6}	10^{9}	10^{12}

There is no distinction between upper and lower case letters. A common error is the use of *M* for *Mega* which is then interpreted by *PSpice* as *Milli*.

Adding the connecting wires: After all the components have been placed and configured, the connecting wires must be entered using the *Draw Wire* tool; here, the cursor takes the form of a pencil. First fix the starting point of the wire with a left mouse click. The wire is then shown as a broken line and can be confirmed from point to point by clicking the left mouse button (see Fig. 29.1.8). The most simple case is a straight line between starting point and end point; in this case the course is determined automatically. By fixing intermediate points the course can be influenced. Placing a point on a component terminal or any other wire causes the program to assume that the wire entry is complete and conclude the entry. As an alternative the entry can be interrupted at any given point by clicking the right mouse button or pressing *Esc*.

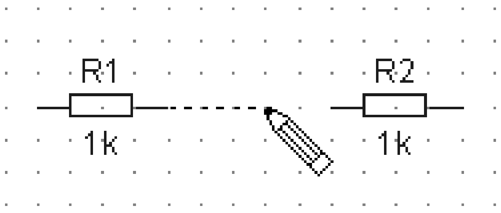


Fig. 29.1.8b. Drawing a connecting wire

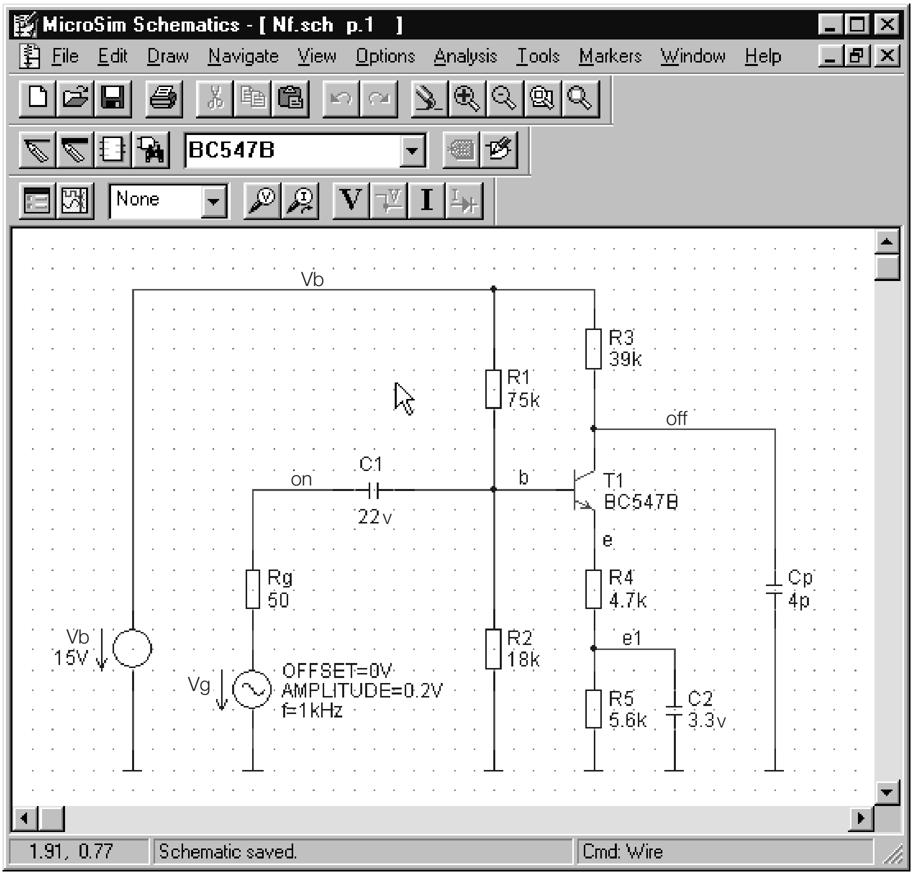


Fig. 29.1.9b. Completed circuit diagram for our example

Normally ground wires are not drawn. Each point connected to ground is marked with a ground symbol GND. In the netlist the nodal name 0 is assigned to any point that is part of the GND net.

A nodal point 0 must exist in every diagram; every circuit diagram must therefore contain at least one ground symbol GND.

A name is assigned automatically to every node. These names appear in the netlist and are required in the *Probe* program to select the signals to be displayed. Since the names assigned automatically do not appear in the circuit diagram and are thus not known without referencing the netlist, an appropriate name should be given to any nodal point in the diagram that is of particular interest; this is done by double clicking on one of the wires that is connected to the node and entering an appropriate name.

After all the components have been entered, connecting wires inserted and nodal names assigned, the completed circuit diagram is presented as shown in Fig. 29.1.9; if you have not already done so, store the diagram by clicking on the *File/Save* button.

Entering Simulation Instructions

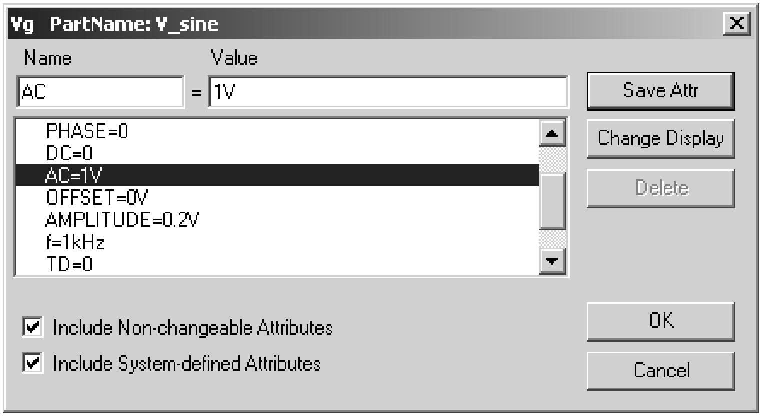
This step describes the simulations to be performed and the parameters for controlling the voltage and current sources used. There are three simulation methods that use different sources:

- *DC Sweep (DC voltage analysis):* This analysis examines the DC response of a circuit; in this process one or two of the sources are varied. The result is a characteristic curve or a family of characteristics. In this analysis only DC voltage sources and DC components of other sources (parameter *DC=*) are taken into consideration.
- *AC Sweep (Small-signal analysis):* This analysis examines the small-signal response. First the operating point of the circuit is determined with the help of the DC sources and the circuit is linearized at this operating point. Then the frequency response is determined for the given frequency range using complex-valued linear AC circuit analysis. In this second step, only the small-signal components of the sources are taken into consideration (parameter *AC=*). As the small-signal analysis is linear there is a linear relation between the result and the given amplitudes of the sources; therefore a normalized amplitude of $1V$ or $1A$, i.e. *AC=1* is commonly used.
- Transient (Large-signal analysis): This analysis covers the large-signal response; it determines the temporal characteristic of all voltages and currents by numeric integration. This analysis only takes into consideration large-signal sources and the large-signal components of other sources.

In our example a small-signal analysis is to be performed to determine the small-signal frequency response, as well as a large-signal analysis with a sine-wave signal with an amplitude of 0.2 V and a frequency of 1 kHz. In this case a large-signal voltage source *V_sine* with the additional *AC* parameter is used at the input (see circuit diagram in Fig.29.1.9). Figure 29.1.10 shows the source parameters that result from the values specified.

Simulation instructions are required in addition to the source settings; these instructions determine the analyses to be performed and the parameters used in each analysis:

- *DC Sweep:* name and range of the source(s) to be varied
- *AC Sweep:* frequency range



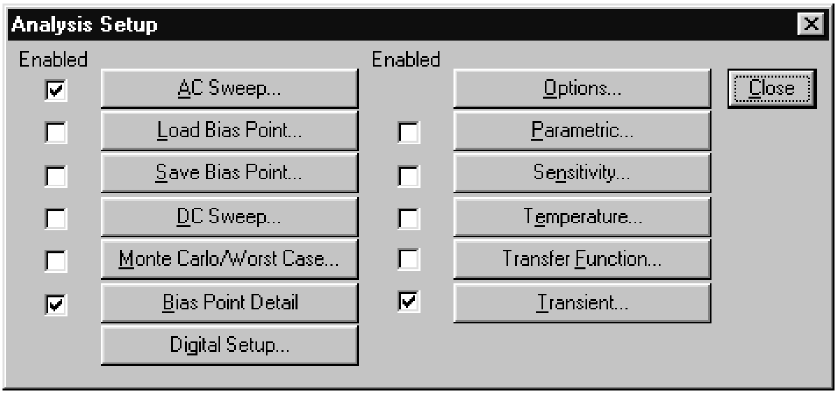
Vg PartName: V_sine

Name: AC = Value: 1V

PHASE=0
DC=0
AC=1V
OFFSET=0V
AMPLITUDE=0.2V
f=1kHz
TD=0

☑ Include Non-changeable Attributes
☑ Include System-defined Attributes

Fig. 29.1.10b. Parameters of the driving voltage source



Analysis Setup

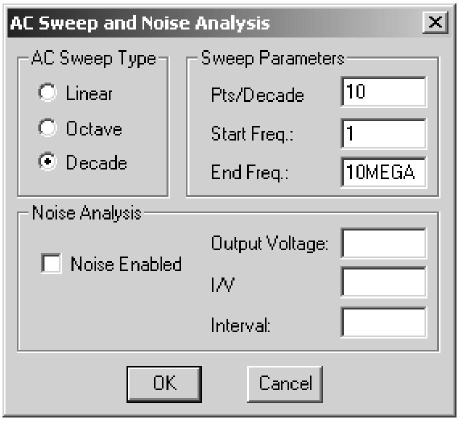
Enabled		Enabled	
☑	AC Sweep...		Options...
□	Load Bias Point...	□	Parametric...
□	Save Bias Point...	□	Sensitivity...
□	DC Sweep...	□	Temperature...
□	Monte Carlo/Worst Case...	□	Transfer Function...
☑	Bias Point Detail	☑	Transient...
	Digital Setup...		

Fig. 29.1.11b. List of analysis options

- *Transient:* length of the time interval to be simulated and, if applicable, the increment for numeric integration

The simulation instructions are finalized with the *Setup Analysis* tool. Here, a list of analysis set-up options appears. This list is shown in Fig. 29.1.11. In addition to the *AC Sweep, DC Sweep* and *Transient* analyses already described, there are other analysis types and capabilities which will be explained later. The *Bias Point Detail* analysis calculates the operating point on the basis of the DC sources and saves the result in the output file <*name*>*.OUT*; this analysis is activated by default. For our example the *AC Sweep* and *Transient* analyses must be activated.

Selecting the field *AC Sweep* opens the *AC Sweep* dialog shown in Fig.29.1.12, in which the frequency range is entered. In our example the frequency range 1 Hz to 10 MHz with 10 points per decade is to be investigated.



AC Sweep and Noise Analysis

AC Sweep Type

○ Linear
○ Octave
● Decade

Sweep Parameters

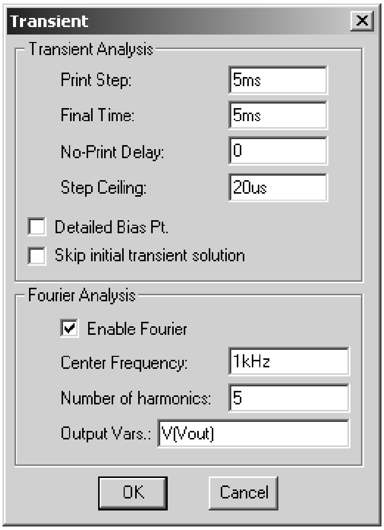
Pts/Decade: 10
Start Freq.: 1
End Freq.: 10MEGA

Noise Analysis

□ Noise Enabled

Output Voltage:
I/V
Interval:

Fig. 29.1.12b. Setting the frequency range for *AC Sweep*



Transient

Transient Analysis

Print Step: 5ms
Final Time: 5ms
No-Print Delay: 0
Step Ceiling: 20us

□ Detailed Bias Pt.
□ Skip initial transient solution

Fourier Analysis

☑ Enable Fourier
Center Frequency: 1kHz
Number of harmonics: 5
Output Vars.: V(Vout)

Fig. 29.1.13b. Setting the parameters for *Transient*

Selecting the *Transient* field opens the *Transient* dialog shown in Fig. 29.1.13. Here, the end point of the simulation is entered in the *Final Time* field and the maximum increment for numeric integration in the *Step Ceiling* field. The start time for recording the results is to be entered in the *No-Print Delay* field; normally this is 0 to allow every value calculated to be displayed graphically. If only the steady-state condition of circuits with long transient times is to be investigated, the estimated transient time can be entered in the *No-Print Delay* field so that recording does not begin until after the transient time. The *Print Step* parameter still exists for historical reasons and is not needed. Nevertheless, this parameter must not be set to 0 and must be less than or equal to the *Final Time*. In addition, the output signal V(Vout) undergoes a Fourier analysis at a fundamental frequency of 1 kHz, which corresponds to the frequency of the source; in doing so, five harmonics are determined, which, together with the resulting harmonic content, are saved in the output file *<name>.OUT*.

After the simulation instructions have been entered, the schematics file is complete and can be stored by pressing the *File/Save* button.

Starting the Simulation

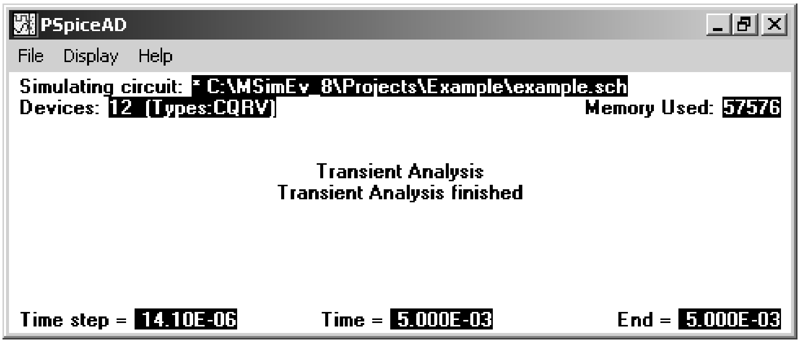
The *Simulate* tool starts the simulation; first the netlist is generated, after which the *PSpice* simulator is started. The *PSpice* window appears during the simulation process; the window shown in Fig. 29.1.14 appears when the simulation is complete.

Displaying the Results

If no errors have occurred during simulation, the Probe program will start automatically. If the simulation contains several analyses, the window shown in Fig. 29.1.15 for selecting an analysis will appear; selecting *AC* opens the *AC* window shown in Fig.29.1.16, which already contains a frequency scale corresponding to the frequency range simulated.

The *Add Trace* tool is used to select the signals to be displayed:

Add Trace	Add a signal



Simulating circuit: * C:\MSimEv_8\Projects\Example\example.sch
Devices: 12 (Types:CQRV) Memory Used: 57576

Transient Analysis
Transient Analysis finished

Time step = 14.10E-06 Time = 5.000E-03 End = 5.000E-03

Fig. 29.1.14b. *PSpice* window following completion of the simulation

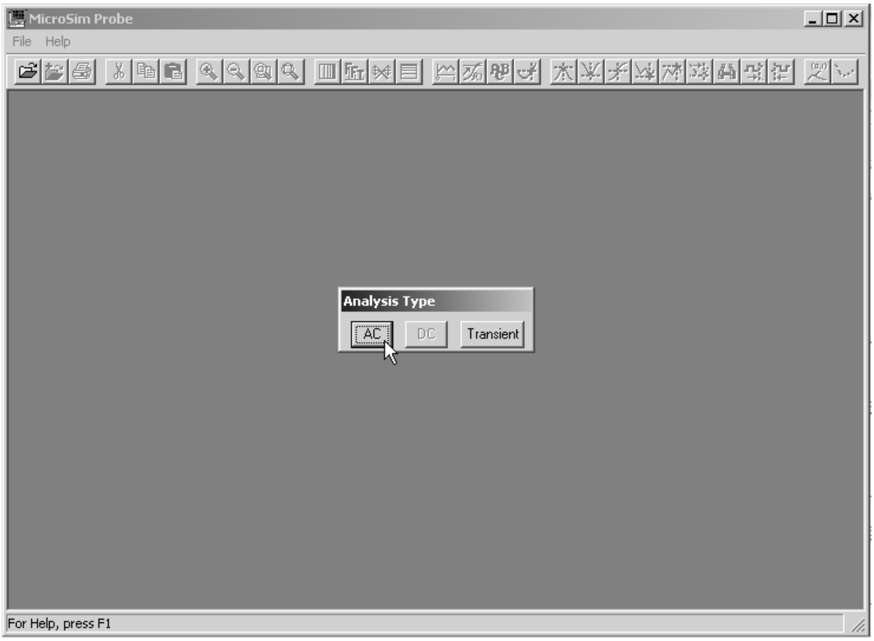


Fig. 29.1.15b. Selecting the analysis type in the *Probe* start-up window

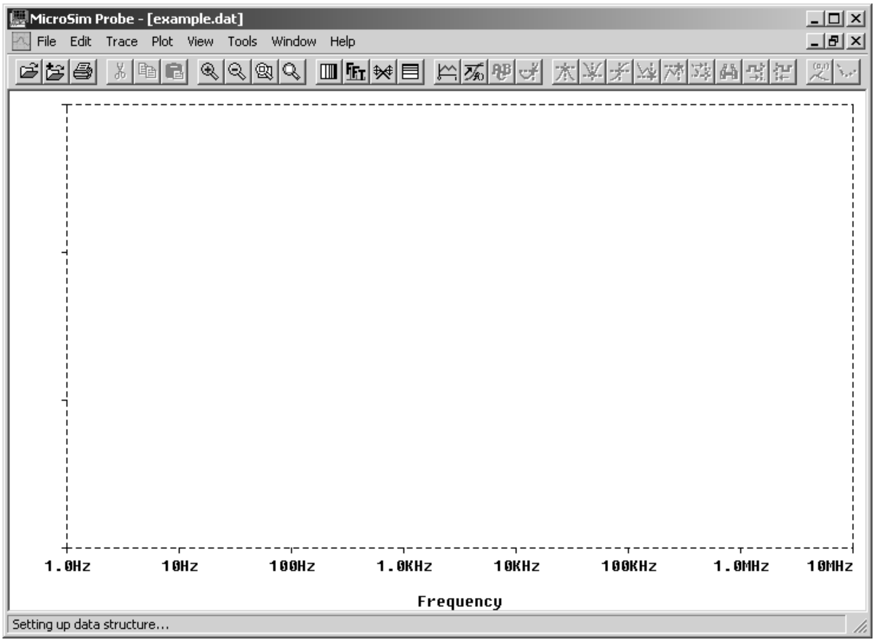
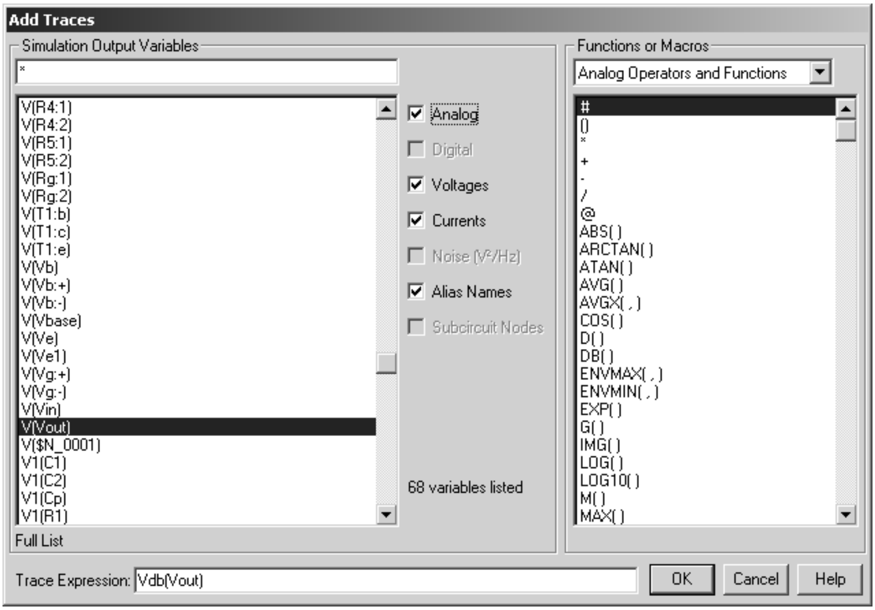

MicroSim Probe - [example.dat]
File Edit Trace Plot View Tools Window Help
1.0Hz
10Hz
100Hz
1.0KHz
10KHz
100KHz
1.0MHz
10MHz
Frequency
Setting up data structure...

Fig. 29.1.16b. *Probe* window after selecting *AC*


Add Traces

Simulation Output Variables: *

☑ Analog
□ Digital
☑ Voltages
☑ Currents
□ Noise (V²/Hz)
☑ Alias Names
□ Subcircuit Nodes

68 variables listed

Full List

Functions or Macros: Analog Operators and Functions

Trace Expression: Vdb(Vout)

Fig. 29.1.17b. Dialog *Add Traces* dialog window

Figure 29.1.17 shows the *Add Traces* dialog window displaying a list of signals on the left and a selection of mathematical functions on the right. The following designations are used:

Designation	Example	Meaning
I(<component>)	I(R1)	Current flowing through a component with two terminals, e.g. the current through resistor R1
I<terminal>(<component>)	IB(T1)	Current flowing through the terminal of a component, e.g. the base current of transistor T1
V(<node name>)	V(Vout)	Voltage between a node and ground, e.g. the voltage at the *Vout* node
V(<component:terminal>)	V(C1:1)	Voltage at the terminal of a component, e.g. the voltage at terminal 1 of capacitor C1
V<terminal>(<component>)	VB(T1)	Voltage at the terminal of a component, e.g. the voltage at the base terminal of transistor T1

Click on the desired signals or functions to select. These will then be shown in the *Trace Expression* field, where they can be edited if necessary. The following options are available for *AC* signals:

Display	Magnitude	Magnitude in dB	Phase

Example	M(V(Vout))
	VM(Vout)
	V(Vout)
		DB(V(Vout))
		VDB(Vout)
			P(V(Vout))
			VP(Vout)

In our example *Vdb(Vout)* represents the magnitude of the output voltage (see Fig. 29.1.18). Since the driving voltage source has an amplitude of 1 V $(AC = 1)$ this value represents the small-signal gain of the circuit. The scale factor of the x and y-axes can be altered via the menu options *Plot/X Axis Settings* and *Plot/Y Axis Settings.*

Other signals that use the same scale factors can be added to the display without further alteration. If signals with different scale factors, e.g. the phase *Vp(Vout),* are to be represented in a meaningful manner, it is first necessary to create another y-axis using the *Plot/Add Y Axis* menu option. The active Y-axis is marked with a » and can be selected with a mouse click; after using *Plot/Add Y Axis* the new y-axis is automatically active. Adding the phase *Vp(Vout)* displays the window shown in Fig. 29.1.19.

Finally we also wish to display the results of the large-signal transient analysis. To do so, we must switch over using the *Plot/Transient* menu option; an empty window appears which already has a time scale corresponding to the time interval simulated. If the voltages *V(Vin)*, $V(Vbase)$, $V(Ve)$ and *V(Vout)* are added via the *Add Traces* option, the display shown in Fig. 29.1.20 appears.

The settings for a particular display can be saved with the *Tools/Display Control* menu option and can be retrieved at a later date. The settings for the various analyses are saved separately so that only those settings are shown that belong to the analysis selected. The settings used last can be called up via the *Last Session* option.

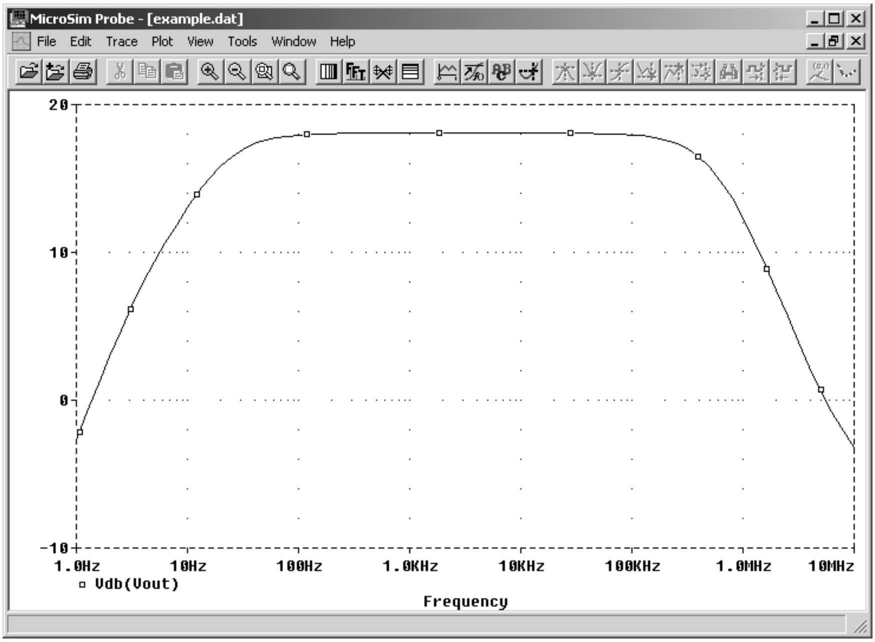

MicroSim Probe - [example.dat]
File Edit Trace Plot View Tools Window Help
20
10
0
-10
1.0Hz
10Hz
100Hz
1.0KHz
10KHz
100KHz
1.0MHz
10MHz
□ Vdb(Vout)
Frequency

Fig. 29.1.18b. Small-signal gain in dB

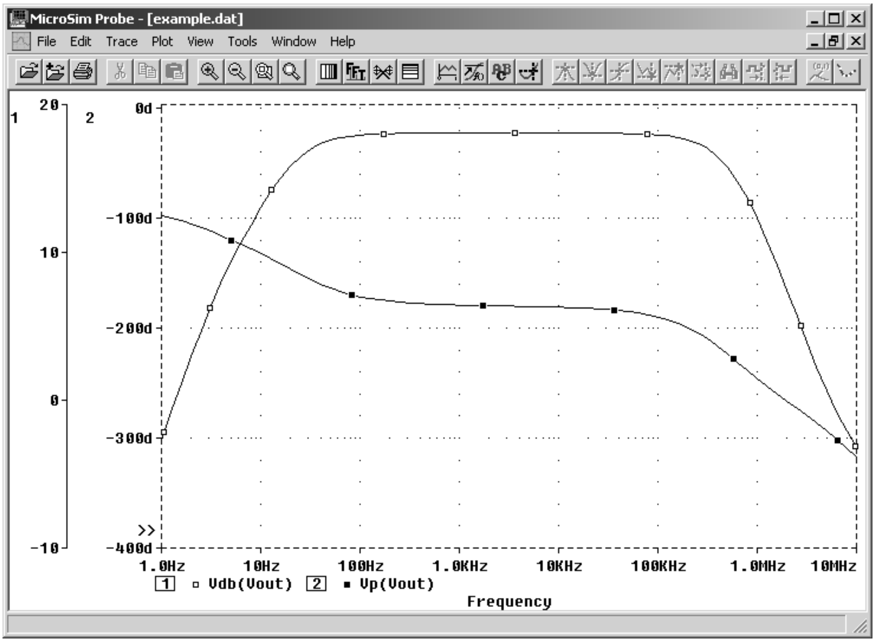

MicroSim Probe - [example.dat]
File Edit Trace Plot View Tools Window Help
1
20
10
0
-10
2
0d
-100d
-200d
-300d
-400d
>>
1.0Hz
10Hz
100Hz
1.0KHz
10KHz
100KHz
1.0MHz
10MHz
1 □ Vdb(Vout) 2 ■ Vp(Vout)
Frequency

Fig. 29.1.19b. Small-signal gain and phase

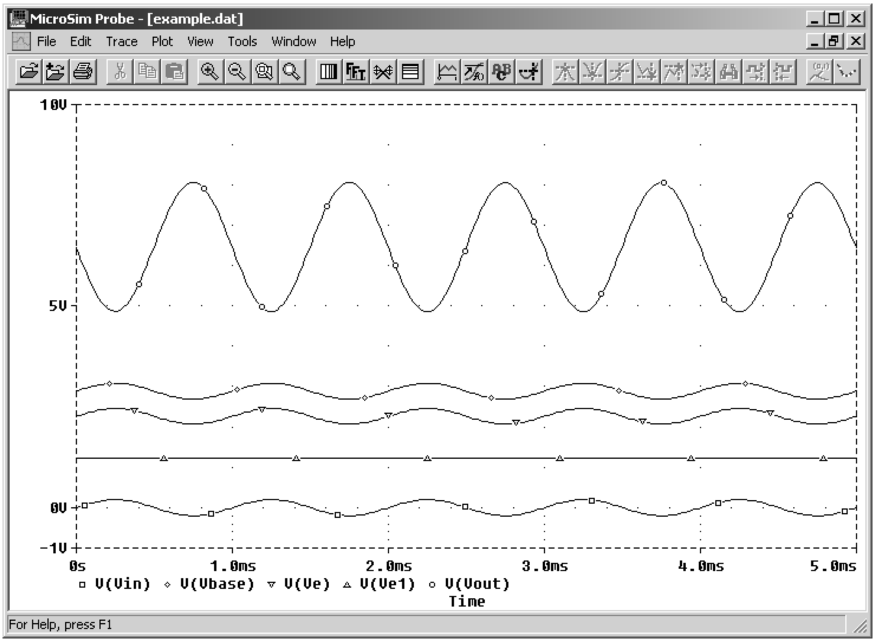

MicroSim Probe - [example.dat]
File Edit Trace Plot View Tools Window Help
10V
5V
0V
-1V
0s
1.0ms
2.0ms
3.0ms
4.0ms
5.0ms
□ V(Vin) ◇ V(Vbase) ▽ V(Ve) △ V(Ve1) ○ V(Vout)
Time
For Help, press F1

Fig. 29.1.20b. Results of the large-signal analysis

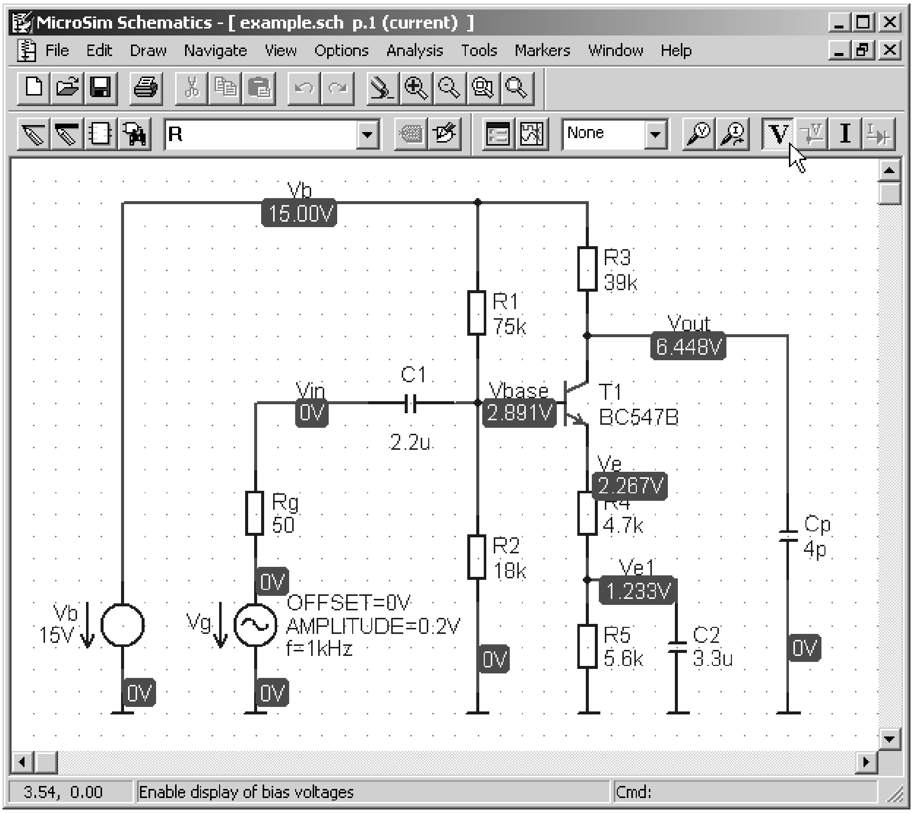
The *Tools/Cursor/Display* menu option allows you to display two markers simultaneously, which are moved with the left or right mouse button; at the same time the x and y values of the marker position are shown in an additional window. For more information refer to *Cursor* in the help index. The markers can also be turned on and off with the *Toggle Cursor* tool:

	Toggle Cursor	Switches the markers on or off

Indicating the operating point

Following a simulation the voltages and currents of the operating point can be shown in the circuit diagram (see Figs. 29.1.21 and 29.1.22); this is done with the folldue tools in the *Schematics* program:

V	*Enable Bias Voltage Display*	Displays the operating point voltages
I	*Enable Bias Current Display*	Displays the operating point currents



Enable display of bias voltages

Fig. 29.1.21b. Circuit diagram with operating point voltages

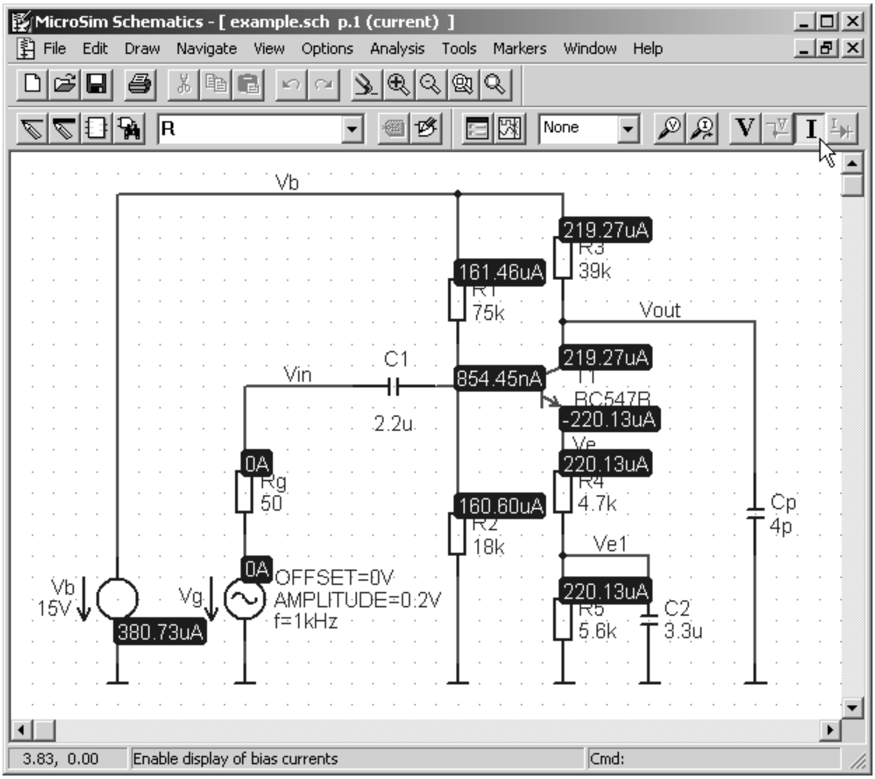


Fig. 29.1.22b. Circuit diagram with operating point currents

After an extensive circuit diagram has been entered, the first step is usually to check the operating point by performing a simulation using a *Bias Point Detail* Analysis, which is activated by default, and to examine the results. This ensures that the circuit has been entered correctly and is functional, before other analyses, which may be somewhat time-consuming, are performed. When following this procedure the *Probe* display program does not start automatically because the *Bias Point Detail* analysis does not produce any graphic data.

Netlist and Output File

The files from our example include the following contents (shortened in some instances):

- **Circuit file EXAMPLE.CIR:**

```
** Analysis setup **
.ac DEC 10 1 10MEGA
.tran 5m 5m 0 20us
.four 1kHz 5 V([Vout])
.OP
* From [SCHEMATICS NETLIST] section of msim.ini:
.lib "C:\MSimEv_8\UserLib\TSE.lib"
.lib "nom.lib"
.INC "example.net"
.INC "example.als"
.probe
.END
```

This file includes simulation instructions (`.ac/.tran/.four/.OP`), references to the model libraries (`.lib`) and instructions for including the netlist and the alias file (`.INC`).

- **Netlist EXAMPLE.NET**:

```
* Schematics Netlist *
R_R4         Ve Ve1 4.7k
R_R5         Ve1 0 5.6k
R_R3         Vb Vout 39k
Q_T1         Vout Vbase Ve BC547B
R_R1         Vb Vbase 75k
R_R2         Vbase 0 18k
C_C2         Ve1 0 3.3u
C_C1         Vin Vbase 2.2u
R_Rg         Vin $N_0001 50
V_Vb         Vb 0 DC 15V
C_Cp         Vout 0 4p
V_Vg         $N_0001 0 DC 0 AC 1V
+            SIN 0V 0.2V 1kHz 0 0
```

- **Output file EXAMPLE.OUT:**

```
****     BJT MODEL PARAMETERS
               BC547B
               NPN
          IS     7.049000E-15
          BF   374.6
          NF     1
         VAF    62.79
         IKF      .08157
         ISE    68.000000E-15
          NE     1.576
          BR     1
          NR     1
         IKR     3.924
         ISC    12.400000E-15
          NC     1.835
          NK      .4767
          RC      .9747
         CJE    11.500000E-12
         VJE      .5
         MJE      .6715
         CJC     5.250000E-12
         VJC      .5697
         MJC      .3147
          TF   410.200000E-12
         XTF    40.06
         VTF    10
         ITF     1.491
          TR    10.000000E-09
         XTB     1.5


****   SMALL SIGNAL BIAS SOLUTION    TEMPERATURE = 27.000 DEG C

   NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE     NODE   VOLTAGE
(   Vb)   15.0000  (    Ve)    2.2673  (   Ve1)    1.2327  (   Vin)    0.0000
( Vout)    6.4484  (Vbase)     2.8908  ($N_0001)   0.0000

    VOLTAGE SOURCE CURRENTS
    NAME         CURRENT
    V_Vb        -3.807E-04
    V_Vg         0.000E+00
    TOTAL POWER DISSIPATION   5.71E-03  WATTS


****   OPERATING POINT INFORMATION   TEMPERATURE = 27.000 DEG C

**** BIPOLAR JUNCTION TRANSISTORS
```

```
NAME          Q_T1
MODEL         BC547B
IB            8.54E-07
IC            2.19E-04
VBE           6.24E-01
VBC          -3.56E+00
VCE           4.18E+00
BETADC        2.57E+02
GM            8.45E-03
RPI           3.47E+04
RX            0.00E+00
RO            3.03E+05
CBE           4.02E-11
CBC           2.82E-12
CJS           0.00E+00
BETAAC        2.93E+02
CBX           0.00E+00
FT            3.13E+07

****   FOURIER ANALYSIS              TEMPERATURE = 27.000 DEG C

FOURIER COMPONENTS OF TRANSIENT RESPONSE V(Vout)

 DC COMPONENT =   6.461795E+00

 HARMONIC   FREQUENCY    FOURIER    NORMALIZED     PHASE        NORMALIZED
    NO         (HZ)     COMPONENT    COMPONENT     (DEG)       PHASE (DEG)
     1      1.000E+03   1.598E+00    1.000E+00   -1.792E+02     0.000E+00
     2      2.000E+03   1.879E-03    1.176E-03    7.745E+01     2.567E+02
     3      3.000E+03   4.460E-05    2.791E-05   -5.927E+01     1.200E+02
     4      4.000E+03   1.202E-04    7.521E-05    8.062E+00     1.873E+02
     5      5.000E+03   9.737E-05    6.093E-05    3.090E-01     1.796E+02

     TOTAL HARMONIC DISTORTION =   1.180047E-01 PERCENT
```

This file contains the parameters for the models used (here: *BJT Model Parameters*), operating point information (*Small Signal Bias Solution*) with the small-signal parameters of the components (*Operating Point Information*) and the results of the Fourier analysis (*Fourier Analysis*).

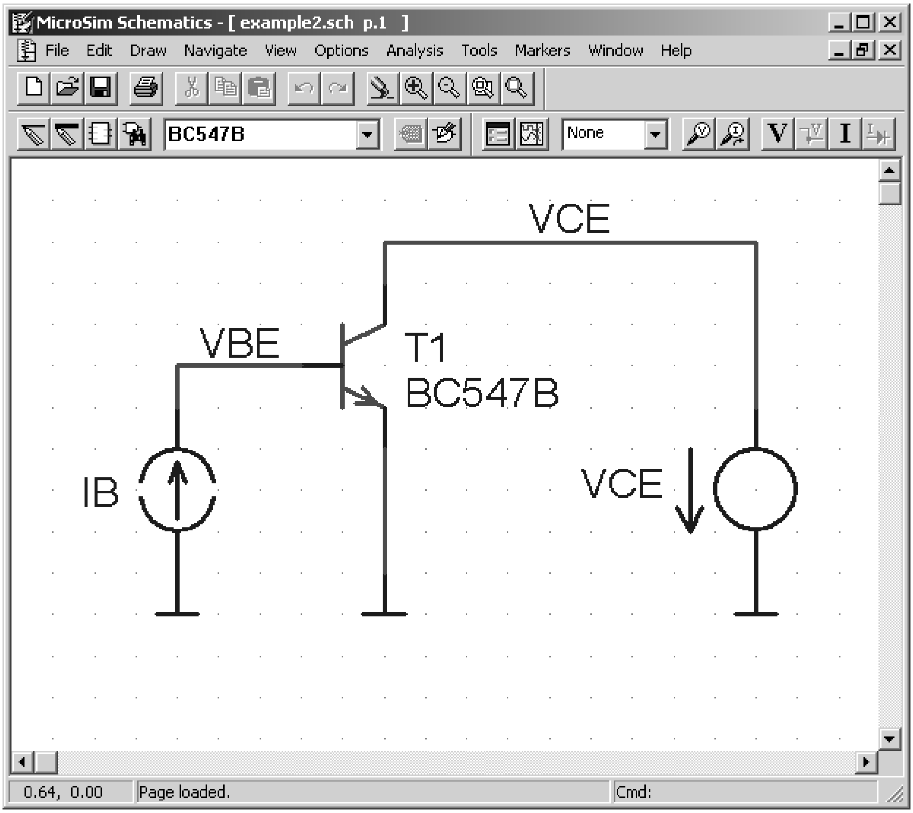
29.1.4 Further Examples

Characteristics of a Transistor

Figure 29.1.23 shows the circuit used in this example. *DC Sweep* is activated in the *Setup Analysis* dialog window (see Fig. 29.1.24). Then the parameters shown in Fig. 29.1.25 are entered:

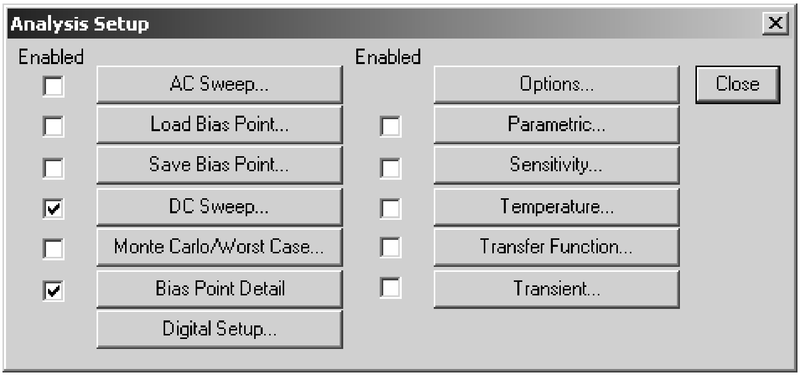
- In the internal loop *DC Sweep* the collector-emitter voltage source VCE is varied in steps of 50 mV through a range of 0…5 V.
- In the external loop *DC Nested Sweep* the base current source IB is varied in steps of 1 µA through a range of 1…10 µA.

After the parameters have been entered, the simulation is started using *Simulate* and the collector current is displayed via *Add Traces* in the *Probe* program (see Fig. 29.1.26).


MicroSim Schematics - [example2.sch p.1]
File Edit Draw Navigate View Options Analysis Tools Markers Window Help
BC547B
None
V
I
VCE
VBE
T1
BC547B
IB
VCE

0.64, 0.00 Page loaded. Cmd:

Fig. 29.1.23b. Circuit diagram for simulating the characteristics


Analysis Setup

Enabled

□ AC Sweep...

□ Load Bias Point...

□ Save Bias Point...

☑ DC Sweep...

□ Monte Carlo/Worst Case...

☑ Bias Point Detail

Digital Setup...

Enabled

Options...

□ Parametric...

□ Sensitivity...

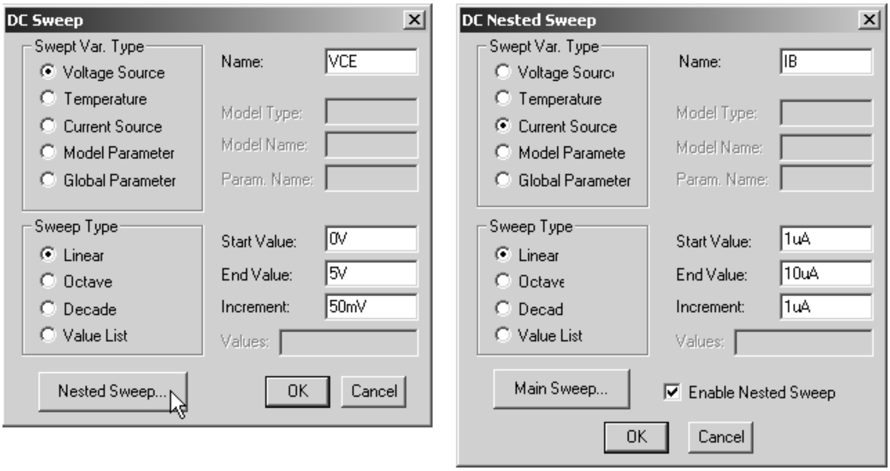
□ Temperature...

□ Transfer Function...

□ Transient...

Close

Fig. 29.1.24b. Activating the *DC Sweep* analysis



DC Sweep

Swept Var. Type
● Voltage Source
○ Temperature
○ Current Source
○ Model Parameter
○ Global Parameter

Name: VCE
Model Type:
Model Name:
Param. Name:

Sweep Type
● Linear
○ Octave
○ Decade
○ Value List

Start Value: 0V
End Value: 5V
Increment: 50mV
Values:

DC Nested Sweep

Swept Var. Type
○ Voltage Sourc
○ Temperature
● Current Source
○ Model Paramete
○ Global Parameter

Name: IB
Model Type:
Model Name:
Param. Name:

Sweep Type
● Linear
○ Octave
○ Decad
○ Value List

Start Value: 1uA
End Value: 10uA
Increment: 1uA
Values:

☑ Enable Nested Sweep

Fig. 29.1.25b. Parameters for the internal and external loop

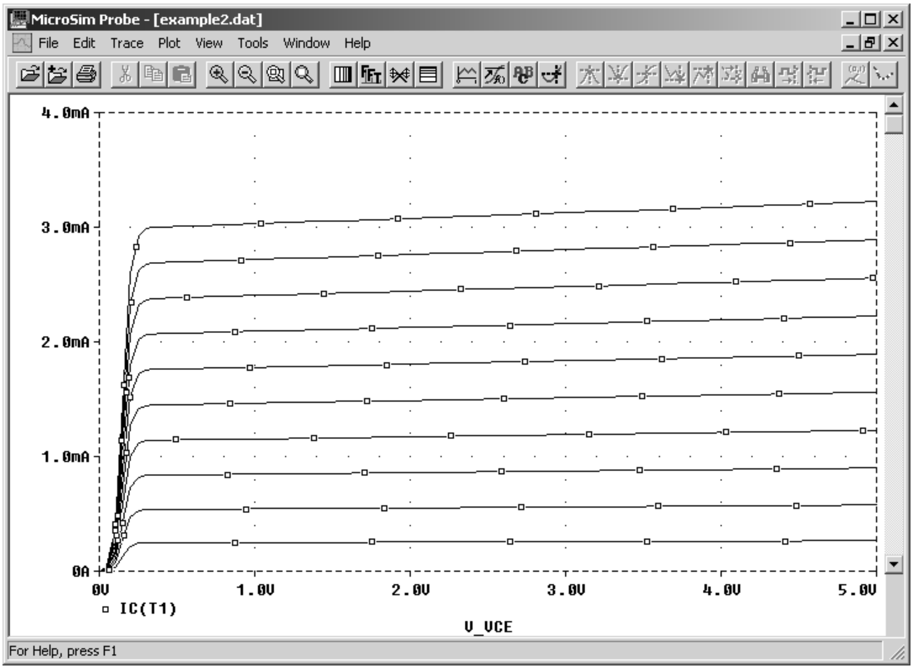


Fig. 29.1.26b. Characteristics of the transistor

Using Parameters

Often it is desirable to perform the same analysis several times while varying one circuit parameter, e.g. the value of a resistor. The example in Fig. 29.1.27 shows the circuit diagram of an inverter with variable base resistance RB. This means that the RB value has to be replaced by a parameter in curly brackets (here: R), which then has to be defined. This is done by means of the *Parameters* component that has been inserted in the upper left corner of the diagram in Fig. 29.1.27. Double clicking on the *Parameter* symbol opens the *Param* dialog window shown in Fig. 29.1.28. You will then be prompted to enter the name of the parameter and its default value; the default value is used in analyses without parameter variations.

In the *Setup Analysis* dialog, *DC Sweep* must be activated in order to simulate the characteristics; similarly, *Parametric* must be activated to vary the parameter (see Fig. 29.1.29). The corresponding parameters are shown in Fig. 29.1.30. In *DC Sweep* a given parameter can also be varied via the *Nested Sweep* option; however, this is not the most flexible of methods as the *Nested Sweep* option is only available in *DC Sweep* analyses, while *Parametric* allows variation in any analysis.

After simulation using *Simulate* the *Probe* program initially displays the window shown in Fig. 29.1.31 for selecting the curves or parameter values to be indicated; in the default

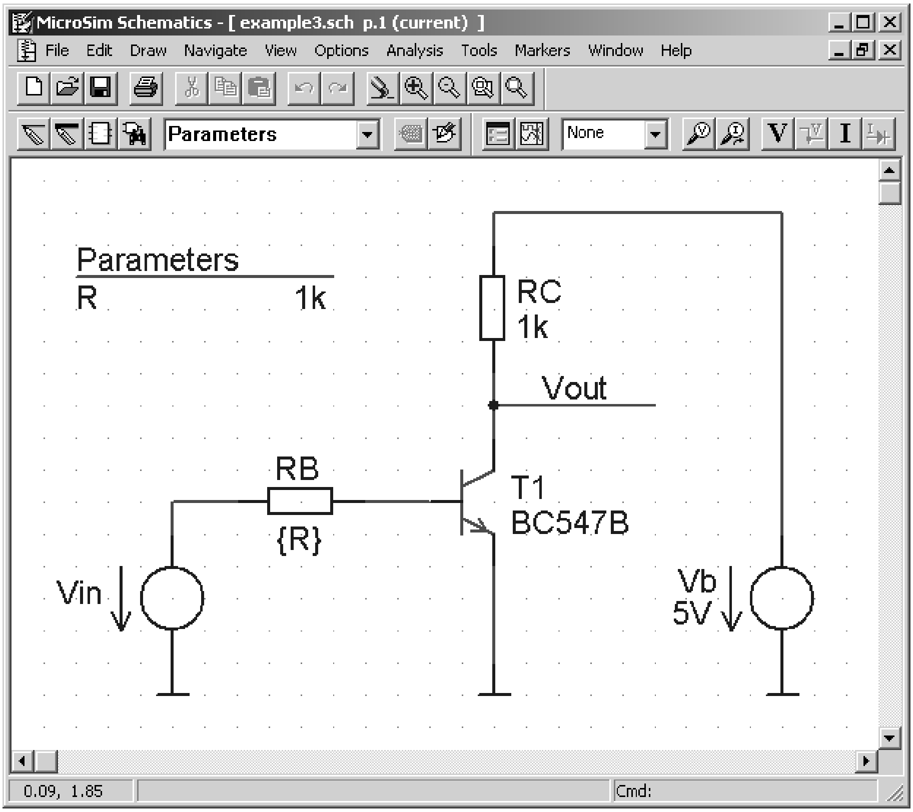
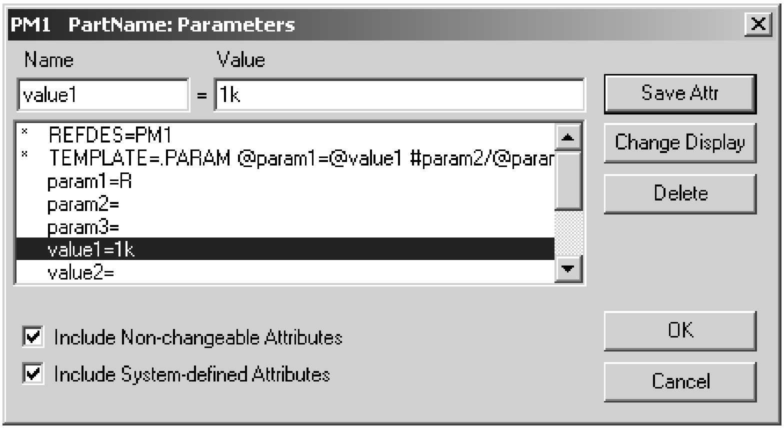

MicroSim Schematics - [example3.sch p.1 (current)]
File Edit Draw Navigate View Options Analysis Tools Markers Window Help
Parameters
None
V
I
Parameters
R 1k
RC
1k
Vout
RB
{R}
T1
BC547B
Vin
Vb
5V
0.09, 1.85
Cmd:

Fig. 29.1.27b. Circuit diagram of the inverter with parameter R



PM1 PartName: Parameters

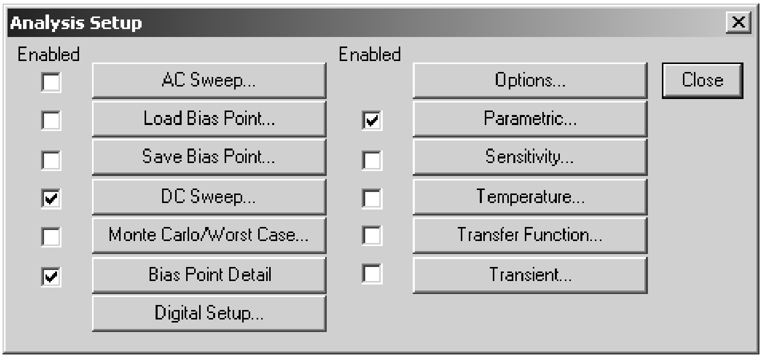
Name: value1 = Value: 1k

- * REFDES=PM1
- * TEMPLATE=.PARAM @param1=@value1 #param2/@parar
- param1=R
- param2=
- param3=
- value1=1k
- value2=

☑ Include Non-changeable Attributes

☑ Include System-defined Attributes

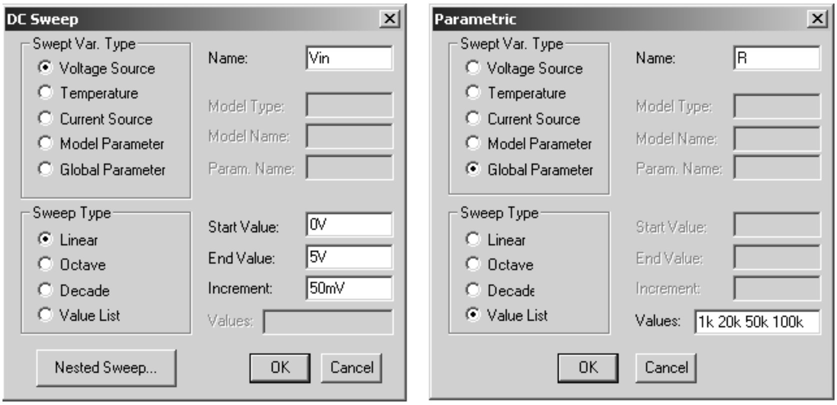
Fig. 29.1.28b. Entering the parameter in the *Param* dialog box



Analysis Setup

Enabled		Enabled	
□	AC Sweep...		Options...
□	Load Bias Point...	☑	Parametric...
□	Save Bias Point...	□	Sensitivity...
☑	DC Sweep...	□	Temperature...
□	Monte Carlo/Worst Case...	□	Transfer Function...
☑	Bias Point Detail	□	Transient...
	Digital Setup...		

Fig. 29.1.29b. Activating *DC Sweep* and *Parametric*



DC Sweep

Swept Var. Type: ● Voltage Source ○ Temperature ○ Current Source ○ Model Parameter ○ Global Parameter

Name: Vin

Model Type:

Model Name:

Param. Name:

Sweep Type: ● Linear ○ Octave ○ Decade ○ Value List

Start Value: 0V

End Value: 5V

Increment: 50mV

Values:

Parametric

Swept Var. Type: ○ Voltage Source ○ Temperature ○ Current Source ○ Model Parameter ● Global Parameter

Name: R

Model Type:

Model Name:

Param. Name:

Sweep Type: ○ Linear ○ Octave ○ Decade ● Value List

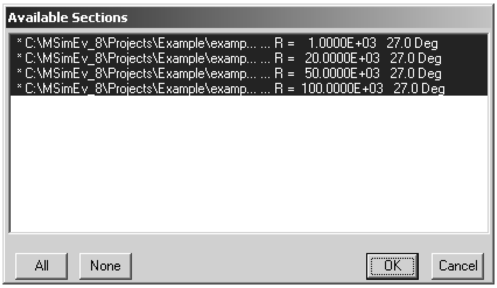
Start Value:

End Value:

Increment:

Values: 1k 20k 50k 100k

Fig. 29.1.30b. Entering the parameters for *DC Sweep* and *Parametric*



```
* C:\MSimEv_8\Projects\Example\examp... ... R =   1.0000E+03  27.0 Deg
* C:\MSimEv_8\Projects\Example\examp... ... R =  20.0000E+03  27.0 Deg
* C:\MSimEv_8\Projects\Example\examp... ... R =  50.0000E+03  27.0 Deg
* C:\MSimEv_8\Projects\Example\examp... ... R = 100.0000E+03  27.0 Deg
```

Fig. 29.1.31b. Selections for curves to be shown

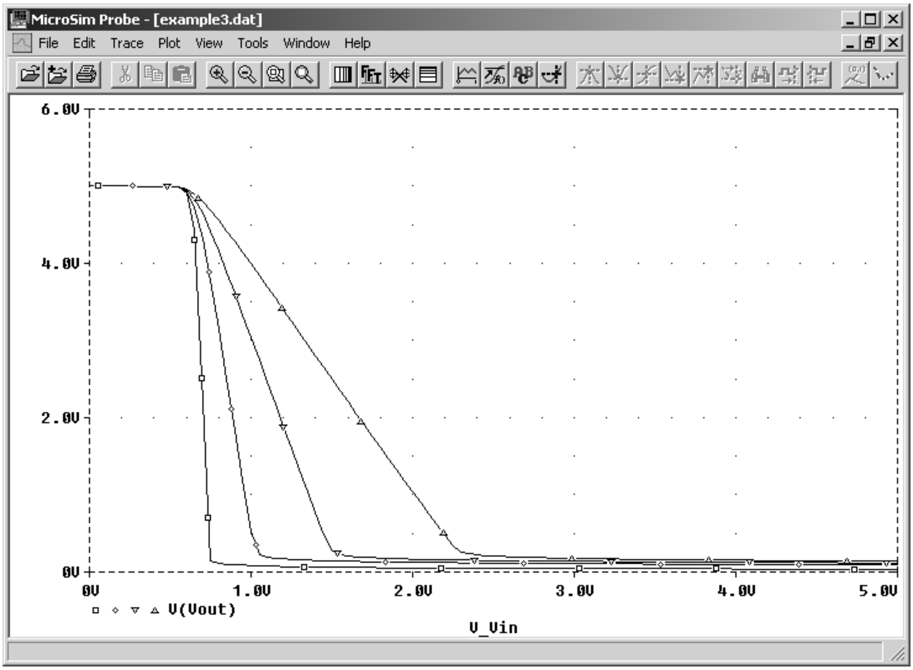


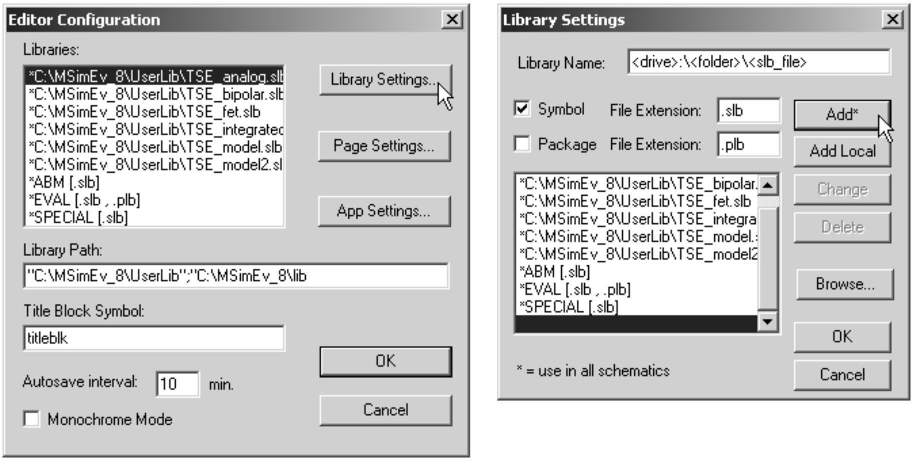
Fig. 29.1.32b. Characteristics of the inverter for R = 1 k/20 k/50 k/100 k

setting all curves are selected. Entering $V(Vout)$ results in the characteristics shown in Fig. 29.1.32. The individual characteristic curves are marked by different symbols, which are shown underneath the graph in the order of the parameter values.

29.1.5 Integrating Other Libraries

A library comprises two sections (see Fig. 29.1.2):

- The *symbol library* „xxx".SLB contains the circuit symbols of the components and information on the representation of the components in the netlist.
- The *model library* „xxx".LIB contains the component models; these are either *elementary models* with the parameters given as .MODEL instructions or *macro models* which


Editor Configuration

Libraries:

*C:\MSimEv_8\UserLib\TSE_analog.slt
*C:\MSimEv_8\UserLib\TSE_bipolar.slt
*C:\MSimEv_8\UserLib\TSE_fet.slb
*C:\MSimEv_8\UserLib\TSE_integratec
*C:\MSimEv_8\UserLib\TSE_model.slb
*C:\MSimEv_8\UserLib\TSE_model2.sl
*ABM [.slb]
*EVAL [.slb , .plb]
*SPECIAL [.slb]

Library Path: "C:\MSimEv_8\UserLib";"C:\MSimEv_8\lib

Title Block Symbol: titleblk

Autosave interval: 10 min.

☐ Monochrome Mode

Library Settings

Library Name: <drive>:\<folder>\<slb_file>

☑ Symbol File Extension: .slb

☐ Package File Extension: .plb

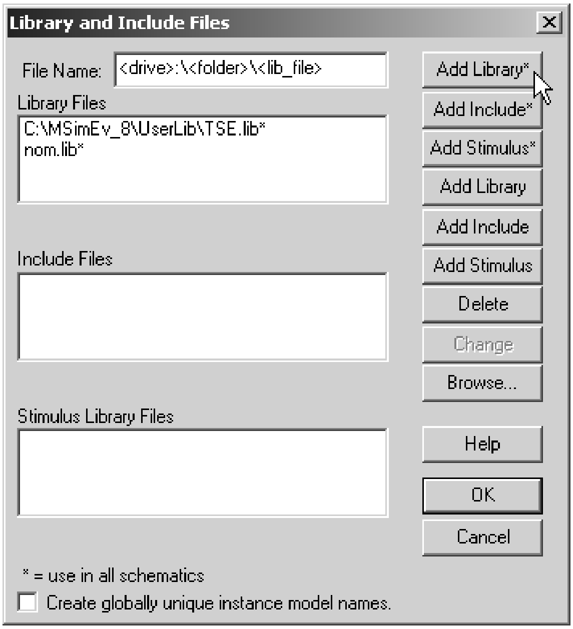
*C:\MSimEv_8\UserLib\TSE_bipolar.
*C:\MSimEv_8\UserLib\TSE_fet.slb
*C:\MSimEv_8\UserLib\TSE_integra
*C:\MSimEv_8\UserLib\TSE_model.:
*C:\MSimEv_8\UserLib\TSE_model2
*ABM [.slb]
*EVAL [.slb , .plb]
*SPECIAL [.slb]

* = use in all schematics

Fig. 29.1.33b. *Editor Configuration* and *Library Settings* dialog windows

comprise several elementary models that are combined to a *sub-circuit* and are contained in the model library in the form of *.SUBCKT„name“„terminals“„circuit“.ENDS*.

A symbol library is integrated in the *Schematics* program via the *Options/Editor Configuration* option. This opens the *Editor Configuration* dialog window shown in the left hand side of Fig. 29.1.33, which lists the existing symbol libraries and their corresponding paths. Selecting the *Library Settings* option opens the dialog box shown in the right hand side of Fig. 29.1.33. This option is used to add, edit or delete symbol libraries. It is possible


Library and Include Files

File Name: <drive>:\<folder>\<lib_file>

Library Files

C:\MSimEv_8\UserLib\TSE.lib*
nom.lib*

Include Files

Stimulus Library Files

* = use in all schematics

☐ Create globally unique instance model names.

Fig. 29.1.34b. *Library and Include Files* dialog window

to enter the name and path (drive and directory) of the library in the *Library Name* field or to search for the desired library using *Browse*. Clicking on *Add** adds the symbol library to the list. To close the dialog window click on *Ok*.

Similarly, a model library is added in the *Schematics* program via the option *Analysis/ Library and Include Files*. The name and path of the library are entered in the same way and the library can then be added with *Add Library** (see Fig. 29.1.34).

To add libraries, always use the *asterisk* commands *Add** or *Add Library** since this adds the libraries *permanently* and makes them automatically available when the program is re-activated. Since in the demo version of *PSpice* both the number of libraries and the number of library elements are limited it is necessary to *exchange* libraries if other libraries are required for further simulations and the total capacity has been used up.

29.1.6 Some Typical Errors

The typical errors are described on the basis of the circuit diagram shown in Fig. 29.1.35 which contains several errors. If an error occurs, the *MicroSim Message Viewer* appears showing the relevant error message either before or after the simulation (see Fig. 29.1.36).

- *Floating Pin:* One of a component's terminals is not connected, e.g. *R2* in Fig.29.1.35. This error occurs during netlist generation; a dialog window with the message *ERC: Netlist/ERC errors – netlist not created* appears and, after clicking on *Ok* to confirm, the *Message Viewer* with the error message *ERROR Floating Pin: R2 pin 2* appears. As a general rule, every terminal must be connected. The only exceptions are components which have been specially configured or macro models which already have *internal* circuitry at one or more terminals so that no *external* circuit is required.

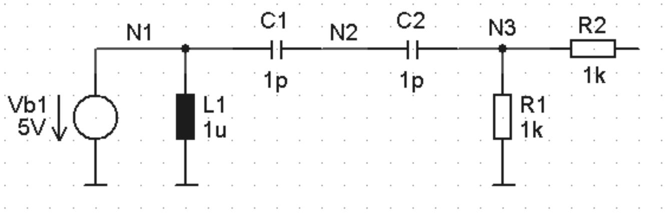
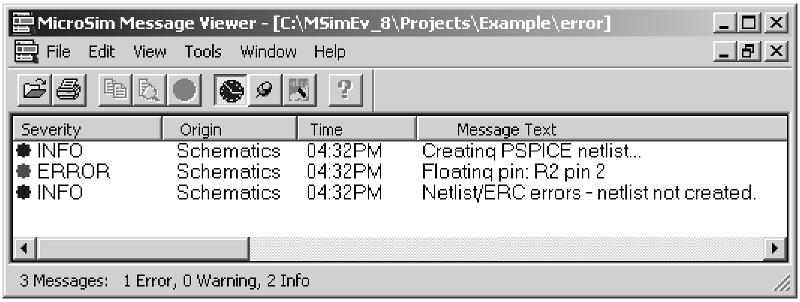


Fig. 29.1.35b. Circuit diagram containing typical errors


MicroSim Message Viewer - [C:\MSimEv_8\Projects\Example\error]
File Edit View Tools Window Help

```
Severity   Origin       Time       Message Text
INFO       Schematics   04:32PM    Creating PSPICE netlist...
ERROR      Schematics   04:32PM    Floating pin: R2 pin 2
INFO       Schematics   04:32PM    Netlist/ERC errors - netlist not created.

3 Messages:  1 Error, 0 Warning, 2 Info
```

Fig. 29.1.36b. *MicroSim Message Viewer* window

- *Node < node name> is floating*: The voltage of a node cannot be determined since it is indeterminate; this is the case for node *N2* in Fig. 29.1.35. This error message appears at any time when only capacitors and/or current sources are connected to a node; consequently, Kirchhoff's node law is not fulfilled. Every node must have a DC path to ground in order to clearly determine the nodal voltage. In the case of node *N2* in Fig. 29.1.35 the error can be rectified, for example, by adding a high-resistive resistor between *N2* and ground.
- *Voltage and/or inductor loop involving „component“:* A loop formed by voltage sources and/or inductors which goes against Kirchhoff's loop law exists; in the example shown in Fig. 29.1.35 the voltage source *Vb1* is short-circuited for DC voltages by the inductor *L1*.

29.2 ispLEVER – Brief User's Guide

29.2.1 Outline

Programming PLDs as described in Chap. 10.4 requires the creation of a so called fusemap containing a list of the desired connections. This may be done manually using a text editor or, more comfortably, with the aid of a design environment like the one provided by ispLEVER. The starter software of ispLEVER can be downloaded by Lattice (www.latticesemi.com) and can be licensed there at no cost.

This program supports the entry of the circuit by either using a programming language or by generating a circuit diagram. Furthermore, in order to test the functionality of the design simulation with graphic output is carried out. In addition to this, the propagation delay time can be tested with the help of timing analysis and optimised for different design goals.

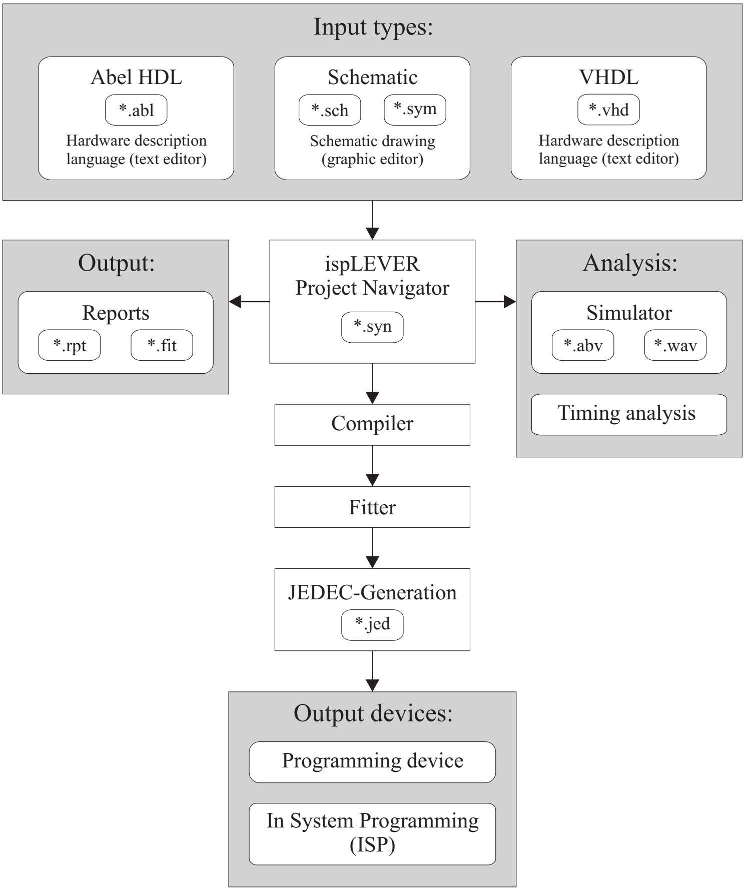


Fig. 29.2.1b. Flow chart

The design environment itself has come a long way. It was initially developed by Data I/O under the name Synario. The software was then acquired by MINC, who were bought out by Vantis, who proceeded to rename the product DesignDirect. After the merger with Lattice it was named DesignExpert and finally became ispLEVER.

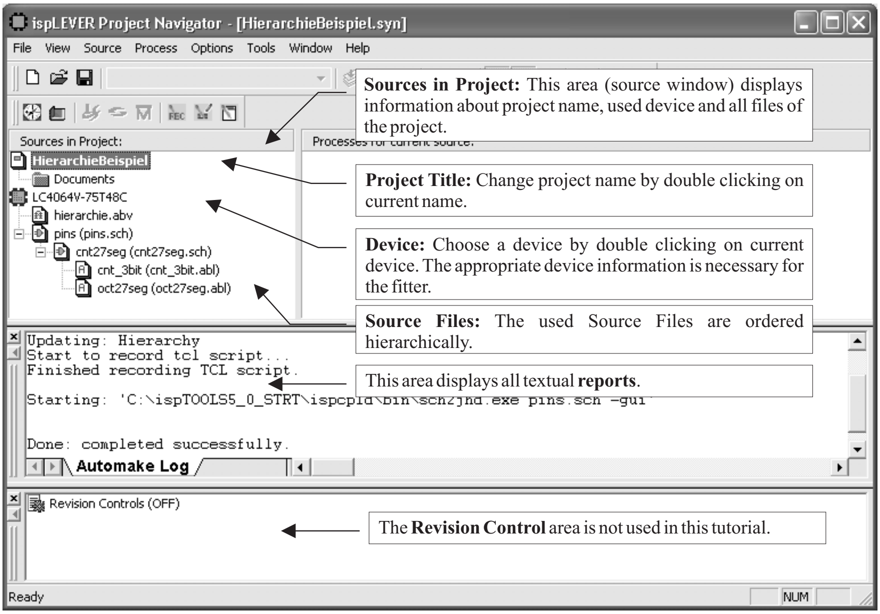
The flow chart in Fig. 29.2.1 shows the interconnections between and procedures carried out in the various input modes as well as the analysis process and the generation of output files. The project navigator is used to start all the activities and to setup the various options.

The hardware description language Abel HDL (High Definition Language) is entered via the text editor. In addition, circuit diagrams can be drawn with the help of the Schematic's input option. A description in the programming language VHDL is also possible, but this is not explained here. The interdependence of the source files is managed and represented in the form of a hierarchy in the source window.

The integrated compiler converts the source files independent of the source into a uniform machine format and generates a process-related report. Several reports are displayed during each phase of the programming process in the output window below. Such reports include not only design errors but also design analysis to show additional information such as timing analysis and utilisation of the chip resources.

The ispLEVER software package comprises various program areas. Their combined features enable the generation of complex designs. The integrated development environment is the project navigator shown in Fig. 29.2.2 which is started via the program ispLEVER. The project navigator shows all the files related to a given project and is used to start up all processes required for the project.

As is common with Windows, the menus and their various options and commands are activated by a left mouse click or a designated key combination. For reasons of simplicity



Sources in Project: This area (source window) displays information about project name, used device and all files of the project.

Project Title: Change project name by double clicking on current name.

Device: Choose a device by double clicking on current device. The appropriate device information is necessary for the fitter.

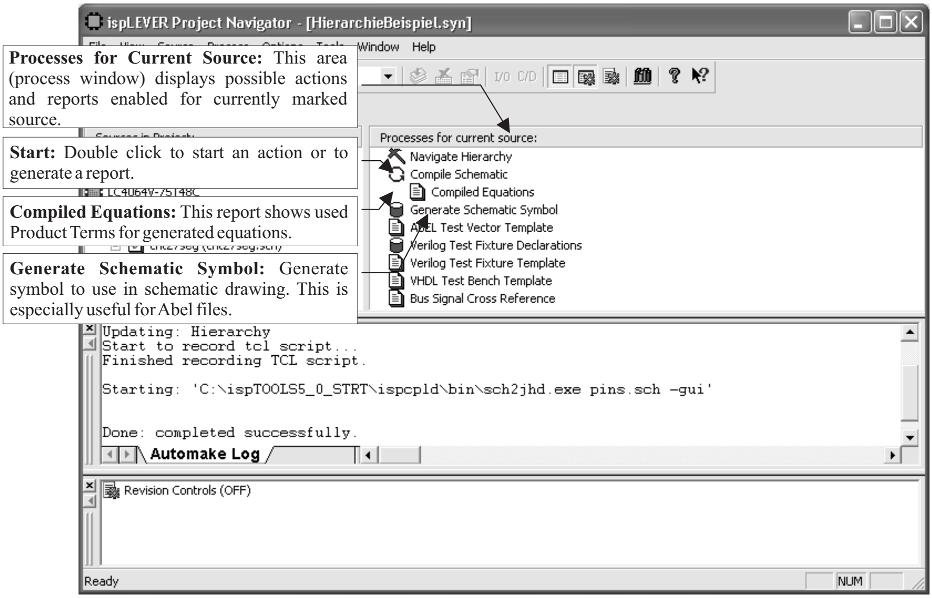
Source Files: The used Source Files are ordered hierarchically.

```
Updating: Hierarchy
Start to record tcl script...
Finished recording TCL script.

Starting: 'C:\ispTOOLS5_0_STRT\ispcpld\bin\sch2jhd.exe pins.sch -gui'


Done: completed successfully.
```

This area displays all textual **reports**.

The **Revision Control** area is not used in this tutorial.

Fig. 29.2.2b. Project Navigator source window



Processes for Current Source: This area (process window) displays possible actions and reports enabled for currently marked source.

Start: Double click to start an action or to generate a report.

Compiled Equations: This report shows used Product Terms for generated equations.

Generate Schematic Symbol: Generate symbol to use in schematic drawing. This is especially useful for Abel files.

Processes for current source:
- Navigate Hierarchy
- Compile Schematic
  - Compiled Equations
- Generate Schematic Symbol
- ABEL Test Vector Template
- Verilog Test Fixture Declarations
- Verilog Test Fixture Template
- VHDL Test Bench Template
- Bus Signal Cross Reference

```
Updating: Hierarchy
Start to record tcl script...
Finished recording TCL script.

Starting: 'C:\ispTOOLS5_0_STRT\ispcpld\bin\sch2jhd.exe pins.sch -gui'


Done: completed successfully.
```

Automake Log

Fig. 29.2.3b. Project Navigator Process window

the following description is limited to the mouse operations. In the source or the process window (see Fig. 29.2.2 and Fig. 29.2.3, respectively) you can perform a specific action or review a report by double clicking the left mouse button.

For example, a double click on the project title "Untitled" allows you to modify the project title. With a double click on the device name you can select the device, while double clicking a file name opens the file.

Both a text and a graphic editor are available for data input. The system provides text reports on the success of the individual compilation steps and the analysis results in the Automake Log window. The functional simulation results may also be presented graphically. The results of the timing analysis appear in the form of a table. Finally a fusemap is generated as a JEDEC file for device programming.

The window at the button of the Project Navigator is for revision control purpose, which is not reasonable for small designs and therefore neglected in this tutorial.

ispLEVER is a very comprehensive software package, which means that this description can only cover the most important commands and features. Detailed information is available in the Help function of each element of the software package.

29.2.2 Circuit Entry

Every new project begins with data input. In principle it is unimportant which device is used at this stage of the design process. But the chip-family should suit because this determines which libraries are available. ispLEVER supports almost every device from Lattice, beginning with the simple PLDs to the complex CPLDs and FPGAs. For our example, we have opted for the model ispM4A5-64/32 to illustrate the various design steps, starting with an empty project and proceeding to circuit input and design analysis and finishing with the creation of the JEDEC file. This device is a CPLD that consists of four PLDs of the 33V16 type and an additional programmable interconnection matrix. The programming logic is located on the chip; it is thus programmable within the circuit (ISP – In System Programmable) as is the case with all newer devices. The only thing required is a passive download cable and a download program as described in Sect. 29.2.5.

The following table shows the meaning of the file name extensions (also refer to Fig. 29.2.1). To archive a project only the files shown in bold print must be copied. The remaining files will be regenerated by the program as required.

abl	Abel HDL file	**sch**	Schematic file
abv	Abel test vectors	**sym**	Symbol for schematic
fit	Fitter report	**syn**	ispLEVER project file
jed	JEDEC file (Fusemap)	**wav**	simulation output
rpt	Report		

Hardware Description Language Abel

The process of describing a circuit function using Abel HDL is explained below using the example of a 3 bit counter that counts from zero to five and then restarts from zero. The state diagram and the schematic symbol are shown in Fig. 29.2.4.

A new project is begun by selecting “New Project …” in the File menu of the Project Navigator. This opens the Project Wizard window (see Fig. 29.2.5a) in which you first type in the desired project name and select or create the desired directory. Furthermore the design entry type is chosen. Use the “Project type” Schematic/Abel as shown. Following the dialog a screen for selecting the desired PLD pops up (see Fig. 29.2.5b). After selecting the device’s family a list with all devices of this family is displayed on the right. Don’t forget to set the correct package type underneath and to check the part number.

The next dialog is for adding source files of former projects (see Fig. 29.2.6.). If you don’t want to import any modules go on without any entry.

The Project Navigator does not automatically adopt the project file name as the project title; the project title can, however, be edited manually if desired by double clicking at the first entry in the source window of the project navigator.

To generate a new Abel data record use the “New …” button from the source menu of the project navigator. Select the Abel HDL module from the list (see left figure in Fig. 29.2.7). Clicking on OK opens a dialog window which prompts the user to enter the Module Name, the File Name and the Title (see right Figure in Fig. 29.2.6). After these informations have

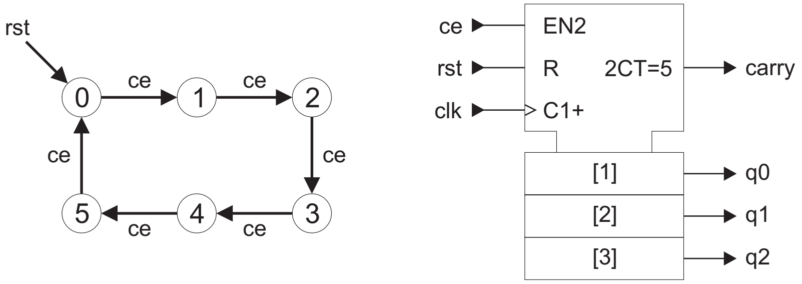
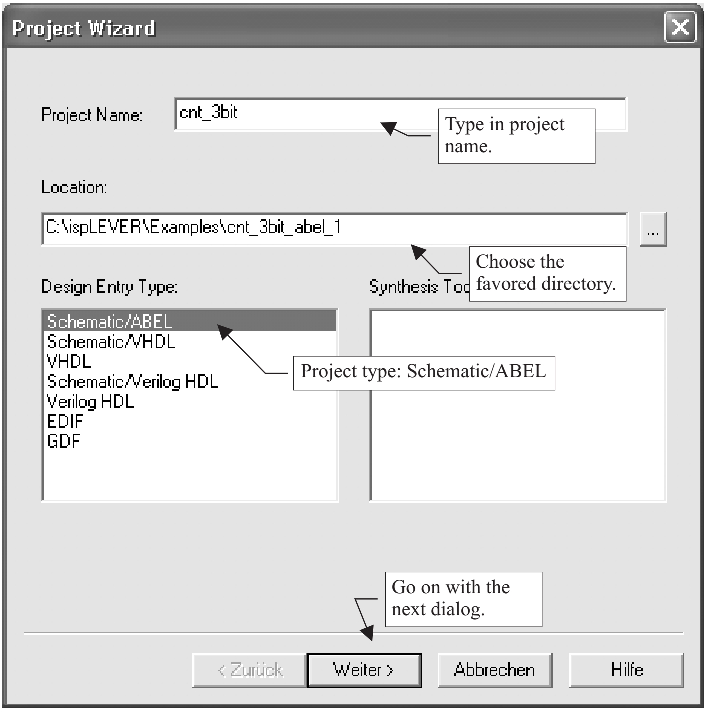


Fig. 29.2.4b. State diagram and schematic symbol of the counter



Project Name: cnt_3bit

Type in project name.

Location: C:\ispLEVER\Examples\cnt_3bit_abel_1

Choose the favored directory.

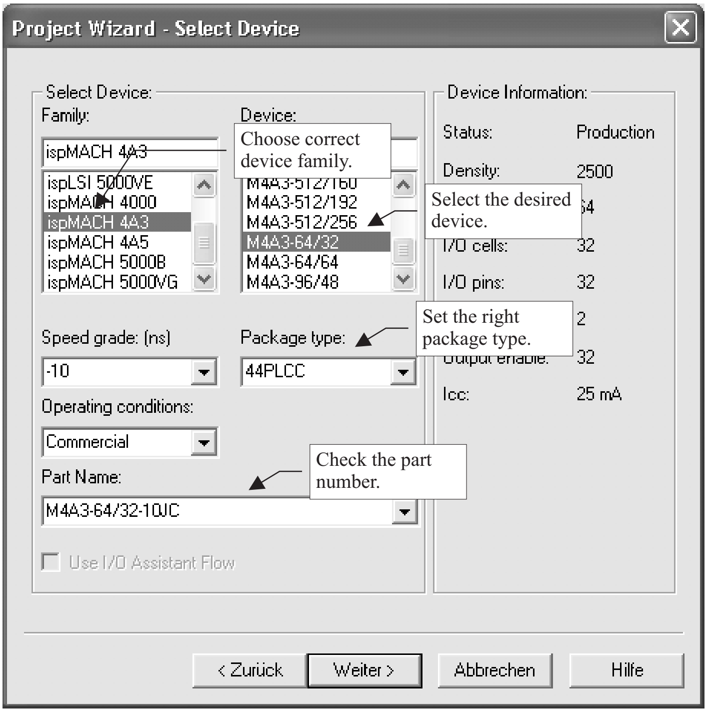
Design Entry Type: Schematic/ABEL, Schematic/VHDL, VHDL, Schematic/Verilog HDL, Verilog HDL, EDIF, GDF

Synthesis Too

Project type: Schematic/ABEL

Go on with the next dialog.

Fig. 29.2.5a. New project



Select Device:

Family: ispMACH 4A3

ispLSI 5000VE, ispMACH 4000, ispMACH 4A3, ispMACH 4A5, ispMACH 5000B, ispMACH 5000VG

Choose correct device family.

Device: M4A3-512/160, M4A3-512/192, M4A3-512/256, M4A3-64/32, M4A3-64/64, M4A3-96/48

Select the desired device.

Speed grade: (ns) -10

Package type: 44PLCC

Set the right package type.

Operating conditions: Commercial

Part Name: M4A3-64/32-10JC

Check the part number.

□ Use I/O Assistant Flow

Device Information:

Status: Production

Density: 2500

I/O cells: 32

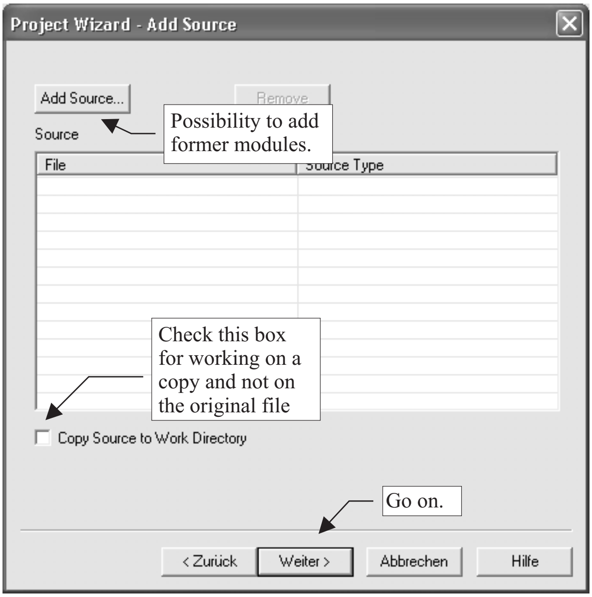
I/O pins: 32

2

Output enable: 32

Icc: 25 mA

Fig. 29.2.5b. Choose device



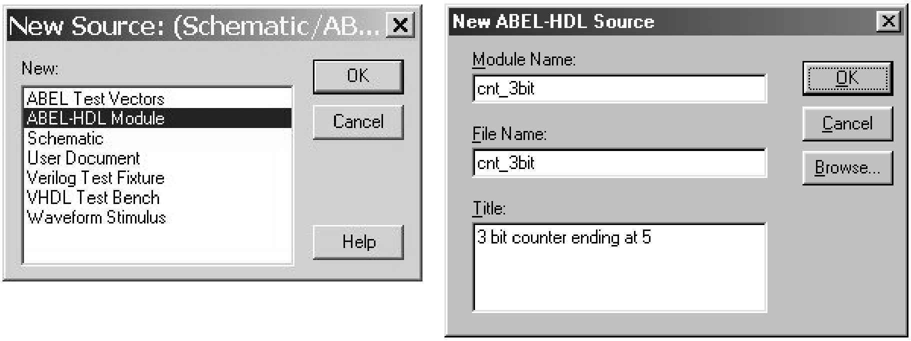
□ Copy Source to Work Directory

Possibility to add former modules.

Check this box for working on a copy and not on the original file

Go on.

Fig. 29.2.6. File import



New: ABEL-HDL Module

Module Name: cnt_3bit

File Name: cnt_3bit

Title: 3 bit counter ending at 5

Fig. 29.2.7. New Abel source

been entered and confirmed via the OK button, the text editor for entering the Abel code will open automatically.

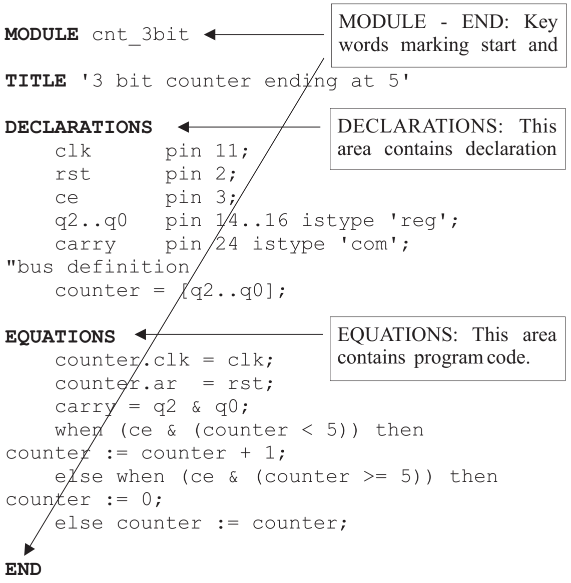
The module name is the identification code within each project, while the file name is the name under which the data record is saved in the directory. Both entries are mandatory; the title, on the other hand, is an optional description of the function. It makes sense to use identical names for module and file in order to facilitate later retrieval. The title should be as detailed as possible so that the function of the module can be readily identified later.

The following program sample shows the complete generation of an Abel HDL module. Of particular importance are the key words shown in bold print, which must exist in every Abel file. If you wish to format the text as shown in the example below spaces or tabs should be used since these are ignored by the compiler.

Enter the example from Fig. 29.2.9 using the names from Fig. 29.2.8 and save the project. If you want to avoid typing you can import the source files from the sample directory (my files\ispLEVER examples that the TS-installer has generated) using the

Project name:	cnt_3bit
Directory:	cnt_3bit_abel_1
Abel module name:	cmt_3bit
Abel file name:	cmt_3bit
Abel title:	3 bit counter ending at 5

Fig. 29.2.8. Suggested names for example 1



```
MODULE cnt_3bit

TITLE '3 bit counter ending at 5'

DECLARATIONS
    clk      pin 11;
    rst      pin 2;
    ce       pin 3;
    q2..q0   pin 14..16 istype 'reg';
    carry    pin 24 istype 'com';
"bus definition
    counter = [q2..q0];

EQUATIONS
    counter.clk = clk;
    counter.ar  = rst;
    carry = q2 & q0;
    when (ce & (counter < 5)) then
counter := counter + 1;
    else when (ce & (counter >= 5)) then
counter := 0;
    else counter := counter;

END
```

MODULE - END: Key words marking start and

DECLARATIONS: This area contains declaration

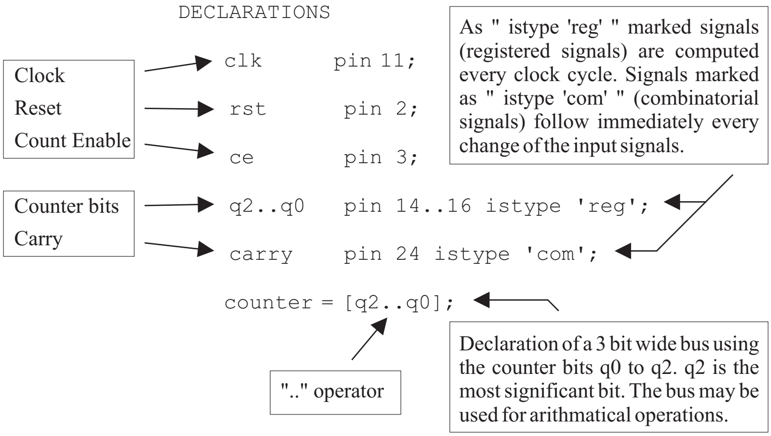
EQUATIONS: This area contains program code.

Fig. 29.2.9. Example 1. Abel module for cnt_3bit

Source/Import option in the Project Navigator. The files can be used to avoid typing and to prevent possible errors. However, the project should not be opened in the existing directory as this would prevent you from being able to follow the individual steps.

Inputs and outputs in the Abel module: The description of the behaviour of a circuit (see Fig. 29.2.9) starts immediately underneath the module name and title with a list of the input and output signals and their interrelations. Pin numbers may also be assigned at this point, which will be explained further below.

Abel recognises output signals from the key word “istype”. It would be more correct to speak here of a resulting signal since it is not necessary available at a pin (see Fig. 29.2.10). The outputs q2 to q0 were declared “istype ‘reg’ ”, which means that each of these outputs is made available via a register (to build a sequential circuit). The carry signal is in the form of a combinatorial circuit which calls for a corresponding designation. The syntax here is “istype ‘com’ ”. The “istype” command has many more variations. But only the “,‘reg’ ” (registered) and “ ‘com’ ” (combinatorial) types are required for this brief user’s guide. A list of some more variations is contained in the on-line help menu of the Text Editor. The outputs are not necessarily available at a pin of the chip; several signals are only intended for internal feedback. Therefore it would be more accurate to speak of resulting signals.



```
DECLARATIONS
clk      pin 11;
rst      pin 2;
ce       pin 3;
q2..q0   pin 14..16 istype 'reg';
carry    pin 24 istype 'com';
counter = [q2..q0];
```

As " istype 'reg' " marked signals (registered signals) are computed every clock cycle. Signals marked as " istype 'com' " (combinatorial signals) follow immediately every change of the input signals.

Declaration of a 3 bit wide bus using the counter bits q0 to q2. q2 is the most significant bit. The bus may be used for arithmatical operations.

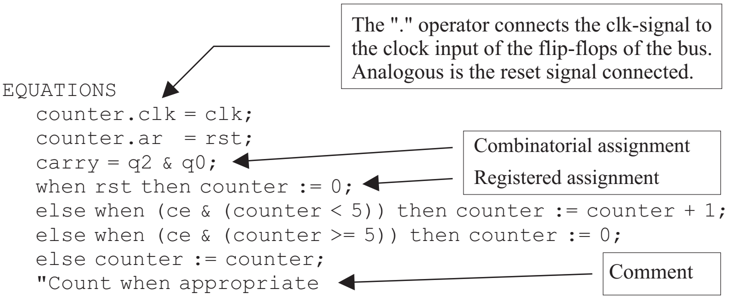
Fig. 29.2.10. Declaration of inputs and outputs

The “..” operator is an abbreviation for sequences, meaning that not every element of a sequence needs to be described in detail. In this example (Fig. 29.2.9) the benefit is minimal since only the element q1 needs not to be entered.

Behaviour of the circuit in the Abel module: The following syntax of the functional description is only one among many, but it is nevertheless a very powerful method for a counter. Figure 29.2.11 shows this method of defining the counter. The second and third program lines tell the compiler to permanently transmit the input signals “clk” and “rst” to the bus “counter”. Normally a decimal value is assigned to the contents of the bus.

The next five lines contain the actual program for the desired function. The carry signal is the AND operation of the first and third bit and therefore becomes active at $5_{\text{dec}} = 101_{\text{dual}}$. Afterwards the counter reading is checked to determine whether or not it has reached the final count of 5. If it is true the counter is reset to 0 otherwise it is incremented.

The “.” operator allows the sub-elements of signals and busses to be accessed. The “:=” operator assigns a signal that is synchronous to the clock (registered), while “=” sets the signal directly (combinatorial) to the given value. This works only with the appropriate variables, e.g. a combinatorial signal cannot be assigned to a registered signal (combinatorial).



```
EQUATIONS
   counter.clk = clk;
   counter.ar  = rst;
   carry = q2 & q0;
   when rst then counter := 0;
   else when (ce & (counter < 5)) then counter := counter + 1;
   else when (ce & (counter >= 5)) then counter := 0;
   else counter := counter;
   "Count when appropriate
```

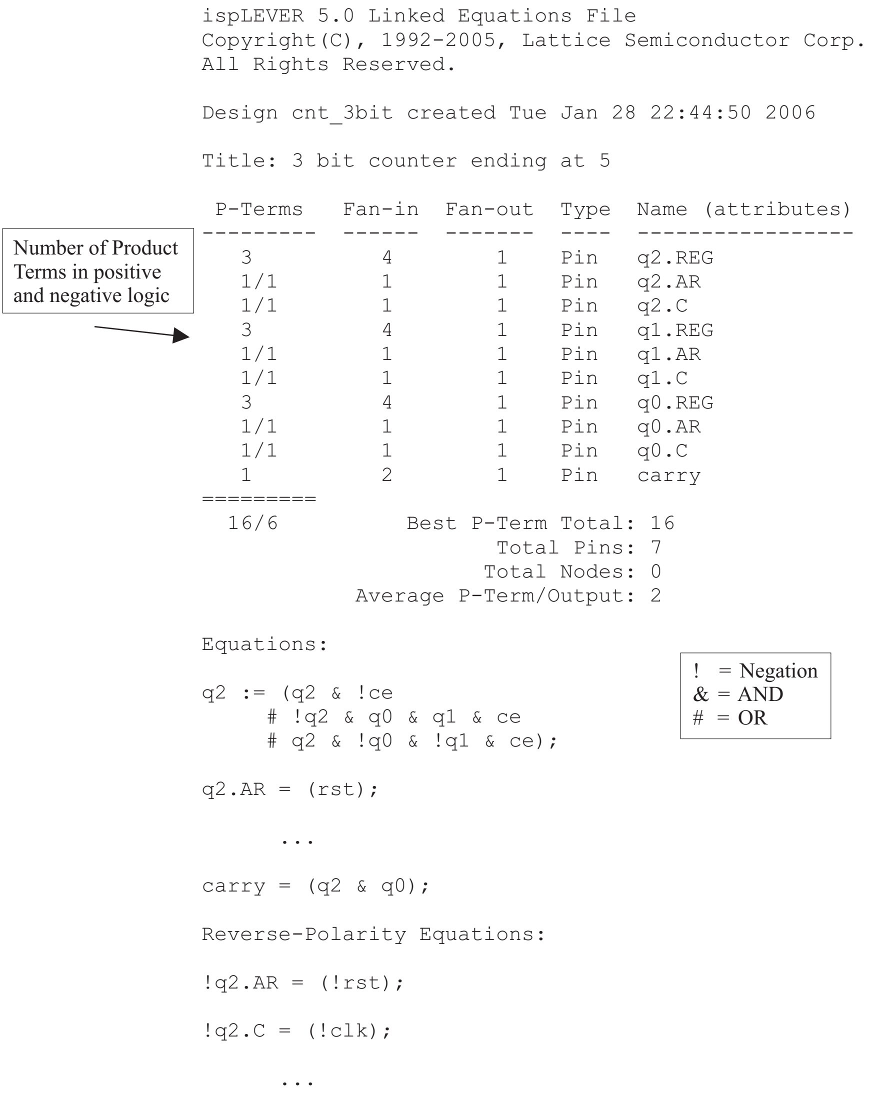
The "." operator connects the clk-signal to the clock input of the flip-flops of the bus. Analogous is the reset signal connected.

Fig. 29.2.11. Functional description

The "when then else" expression is used to distinguish between different situations. Any logic expression can serve as a condition; ensure that parentheses are used intelligently to avoid erroneous operations. The operators are processed in the sequence given by the compiler; but parentheses make it easier to understand the intention of the programmer.

After closing the text editor you can compile the Abel module by double clicking on "Compile Logic" in the process window of the project navigator. The compiler automatically checks the syntax.

The results of the compilation can be checked in the report window by opening the desired object in the "Processes for current source" window in the Project Navigator.



```
ispLEVER 5.0 Linked Equations File
Copyright(C), 1992-2005, Lattice Semiconductor Corp.
All Rights Reserved.

Design cnt_3bit created Tue Jan 28 22:44:50 2006

Title: 3 bit counter ending at 5

 P-Terms   Fan-in  Fan-out  Type  Name (attributes)
---------  ------  -------  ----  -----------------
   3          4        1    Pin   q2.REG
   1/1        1        1    Pin   q2.AR
   1/1        1        1    Pin   q2.C
   3          4        1    Pin   q1.REG
   1/1        1        1    Pin   q1.AR
   1/1        1        1    Pin   q1.C
   3          4        1    Pin   q0.REG
   1/1        1        1    Pin   q0.AR
   1/1        1        1    Pin   q0.C
   1          2        1    Pin   carry
=========
  16/6          Best P-Term Total: 16
                       Total Pins: 7
                      Total Nodes: 0
            Average P-Term/Output: 2

Equations:

q2 := (q2 & !ce
     # !q2 & q0 & q1 & ce
     # q2 & !q0 & !q1 & ce);

q2.AR = (rst);

      ...

carry = (q2 & q0);

Reverse-Polarity Equations:

!q2.AR = (!rst);

!q2.C = (!clk);

      ...
```

Number of Product Terms in positive and negative logic

! = Negation
& = AND
= OR

Fig. 29.2.12. Compiled equations for example 1

You can view the reports in the Automake Log window (see Fig. 29.2.2) that imforms you about the success of compiling your design. If you follow this procedure for the “Compiled Equations”, you will receive a report as shown in Fig. 29.2.12. Please note that the Project Navigator automatically performs all preparatory steps. In our example the Project Navigator would cause the compiler to begin compilation (Compile Logic) as soon as the report is requested. Use “Generate Schematic Symbol” to create a symbol for later use.

```
MODULE cnt_3bit

TITLE '3 bit counter ending at 5'

DECLARATIONS
    clk      pin 11;
    rst      pin 2;
    ce       pin 3;
    q2..q0   pin 14..16 istype 'reg';
    carry    pin 24 istype 'com';
"bus definition
    counter = [q2..q0];

EQUATIONS
    counter.clk = clk;
    carry = q2 & q0;

STATE_DIAGRAM counter;
    State 0:
      if (!rst & ce) then 1;
      else 0;
    State 1:
      if (rst) then 0;
      else if (ce) then 2;
      else 1;
    State 2:
      if (rst) then 0;
      else if (ce) then 3;
      else 2;
    State 3:
      if (rst) then 0;
      else if (ce) then 4;
      else 3;
    State 4:
      if (rst) then 0;
      else if (ce) then 5;
      else 4;
    State 5:
      if (rst) then 0;
      else if (ce) then 0;
      else 5;
    State 6:
      goto 0;
    State 7:
      goto 0;
end;
```

Equivalent to first example

State variable

if(rst = 0 & ce = 1) then 1

State number

Go to state 0 if rst is true.

Remain in current state

Go to state 0

Fig. 29.2.13. Counter with status commands

Project name:	ent_3bit
Directory:	ent_3bit_abel_2
Abel module name:	ent_3bit
Abel file name	ent_3bit
Abel title:	3 bit counter ending at 5

Fig. 29.2.14. Suggested names for example 2

State Diagram in Abel

The hardware description language Abel HDL also offers the possibility of creating a sequential logic as a state diagram in text form.

With the "STATE_DIAGRAM" key word (see Fig. 29.2.13) you can enter each transient after making the usual declarations of module name, inputs/outputs and buses (vectors).

Create a new project. Use the names from Fig. 29.2.14 and generate the new directory. Enter the Abel HDL module or import it with the Project Navigator: Source/Import option from my files\ispLEVER examples.

The basic difference between the two sample counter designs is obvious: When describing the counter using a loop, only the maximum counter reading needs to be changed in order to create a counter of any desired count cycle.

In the state diagram every state must be listed individually. For this reason this type of entry is suitable only for sequential logic circuits with few states. On the other hand it enables complex transient conditions to be described.

Truth Table in Abel

Another possible entry option in Abel HDL is the truth table. It is primarily intended to create combinatorial circuits. In our example we will create a seven-segment decoder. This is meant to translate the output of the counter in the previous examples into a seven-segment display (shown in Fig. 29.2.15). The suggested names for the third example are listed in Fig. 29.2.16. Figure 29.2.17 shows the data record to be entered. Again observe the "compiled equations" and generate a schematic symbol using "Generate Schematic Symbol".

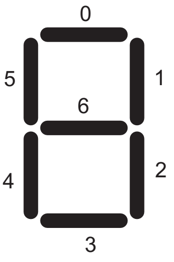
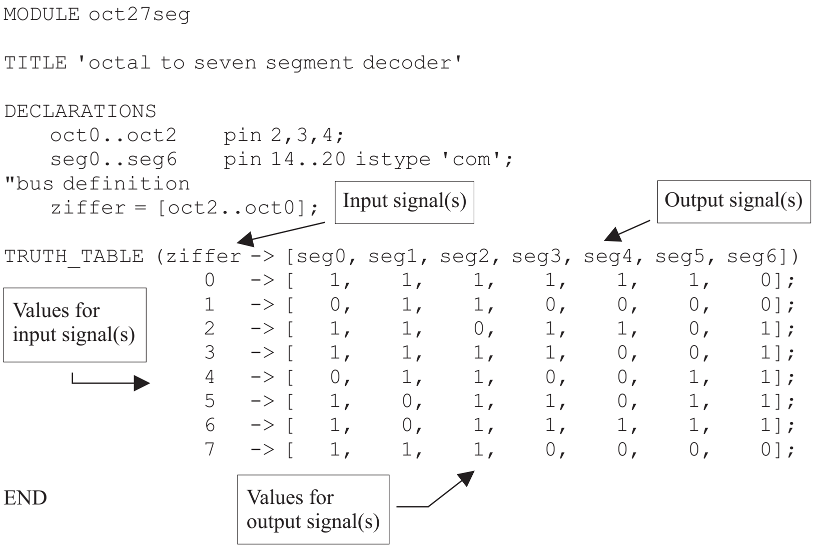


Fig. 29.2.15. Seven-segment display

Project name:	oct27seg
Directory:	oct27seg
Abel module name:	oct27set
Abel file name	oct27seg
Abel title:	octal to seven segment decoder

Fig. 29.2.16. Suggested names for example 3



```
MODULE oct27seg

TITLE 'octal to seven segment decoder'

DECLARATIONS
    oct0..oct2    pin 2,3,4;
    seg0..seg6    pin 14..20 istype 'com';
"bus definition
    ziffer = [oct2..oct0];

TRUTH_TABLE (ziffer -> [seg0, seg1, seg2, seg3, seg4, seg5, seg6])
                 0   -> [   1,    1,    1,    1,    1,    1,    0];
                 1   -> [   0,    1,    1,    0,    0,    0,    0];
                 2   -> [   1,    1,    0,    1,    1,    0,    1];
                 3   -> [   1,    1,    1,    1,    0,    0,    1];
                 4   -> [   0,    1,    1,    0,    0,    1,    1];
                 5   -> [   1,    0,    1,    1,    0,    1,    1];
                 6   -> [   1,    0,    1,    1,    1,    1,    1];
                 7   -> [   1,    1,    1,    0,    0,    0,    0];

END
```

Input signal(s)
Output signal(s)
Values for input signal(s)
Values for output signal(s)

Fig. 29.2.17. Data record for the seven-segment decoder

Circuit Diagram with Schematic

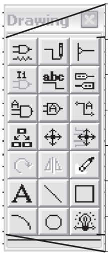
The Schematic Editor is a tool used to produce graphic designs for digital circuits. The design process for such circuit diagrams is supported by several libraries of logic cells, registers, input/output buffers, etc. (graphic symbols with inputs and outputs) to which the designer has access. These libraries allow the user to add his own components created either in Abel HDL or with the Schematic Editor.

If you choose the Source/New command "Schematic" a dialog box opens that prompts the entry of a name for the new schematic. After a suitable name has been fed, the Schematic Editor is ready for the circuit to be entered.

When creating a circuit diagram in Schematic it is very important to know the tool box commands described in more detail in Fig. 29.2.18. Each description in the table corresponds to a button in the tool box.

Again the design example is the 3 bit counter from Fig. 29.2.9. Three toggle flip-flops with reset input are used and connected accordingly. The corresponding circuit is shown in Fig. 29.2.19.

Enter the example using the names from Fig. 29.2.20 or use the Project Navigator: Source/Import option to import the information from my files\ispLEVER examples.



Add Symbol	Add Wire	Add Bus Tap
Add Instance	Add Net Name	Add I/O Marker
Edit Pin	Edit Symbol	Edit Net
Duplicate	Move	Drag
Rotate	Mirror	Delete
Draw Text	Draw Line	Draw Rectangle
Draw Arc	Draw Circle	Highlight

Fig. 29.2.18. Tool box of the Schematic Editor

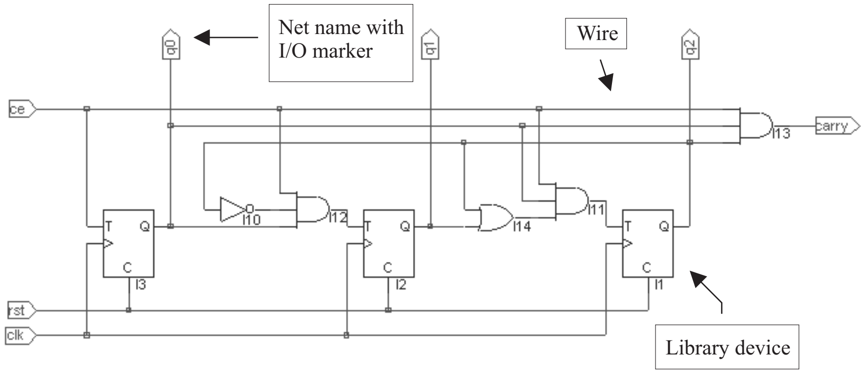


Fig. 29.2.19. Counter Schematic

Project name:	cnt_3bit
Directory:	cnt_3bit_sch
Schematic file name:	cnt_3bit

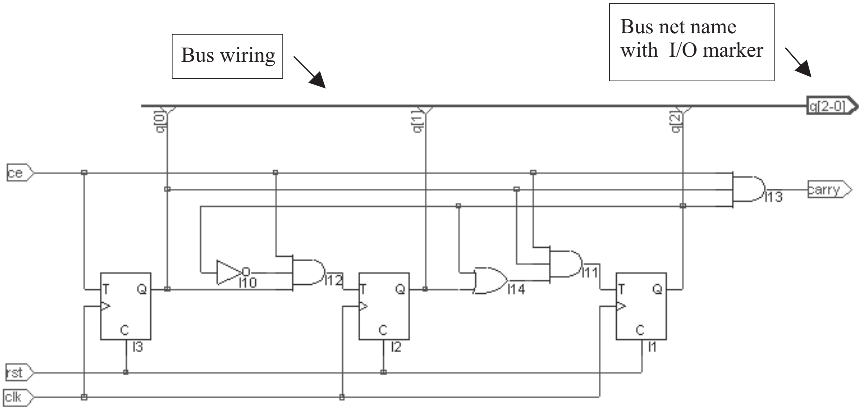
Fig. 29.2.20. Suggested names for example 4

Only the most important steps for the design of a circuit diagram will be outlined here. For further information refer to the Help menu. To perform an operation first select a command from the tool box and then the object that is to be used. If you want to terminate a command, click on the circuit diagram with the right mouse button. If an entire region is marked, the command is applied to several objects simultaneously. To do so, press and hold the left mouse button and draw a box around the desired objects. If you want to cancel the last step click on Undo (either in the Edit menu or in the symbol bar).

The following steps must be executed to create the sample design:

- Insert the toggle flip-flops (library REGS.LIB, component G_TC) selected from the Symbol Libraries (Add Symbol) into the diagram. The desired library can be selected in the upper section of the popped up window. Then mark the component in the lower section and place it three times in the desired positions in the circuit diagram.
- Add the required logic gates to the diagram as shown in Fig. 29.2.19. You find them in the GATES.LIB; select the gates G_INV, G_2OR and G_3AND.
- Use Add Wire to connect components at the respective red dots.
- Assign "net names" to all inputs and outputs so they can be addressed in the superordinate modules or for testing. After selecting the command (Add Net Name) enter the desired name at the bottom of the Schematic Editor, confirm it by clicking on Enter and click on the red dot at the end of the wire.
- Now use the I/O markers (Add I/O Marker) to define the signals as input or output signals. In the case of several I/Os, this can also be done for all of them in one step by marking the relevant region.
- Save the schematic diagram.
- Use File/Matching Symbol to create a schematic symbol.

In order to illustrate how a bus is generated the outputs q0 up to q2 are connected to one bus (see Fig. 29.2.21).


Bus wiring

Bus net name with I/O marker

Fig. 29.2.21. Circuit diagram of counter with bus

Follow the description below:

- First remove the output wires and markers from the diagram (Delete). To do so you can either click on the element to be deleted or mark and delete an entire region completely.
- Draw a horizontal wire above the circuit.
- Designate the right end of the label with a net name called q[0-2]. This makes the line a bus which then appears as a bolder line.
- Define the bus as an output using an I/O marker.

The next step is to establish the bus connections, i.e. to connect the output signals to the bus. This is done in the following manner:

- First enter the range of bus connections via Add Net Name (q[0-2] in this example) and confirm by pressing Enter.
- Then hold Shift and press the right mouse button. The name assigned to the first connection (here q[0]) will appear at the cross-hair cursor.
- Hold down the left mouse button and draw a line **from** the desired output point to the bus. Releasing the mouse button will place a bus tap and at the cross-hair cursor appears the label of the next connection (here q[1]).
- Repeat this procedure for all other connections to be made. For necessary corrections use the Undo function.

Alternatively you can do the connections using the Bus Tap Tool from the toolbox:

- Click the bus and drag a horizontal line from the bus to the pin to be connected.
- Choose the naming tool and click with the left mouse button at the bus. The bus name will appear at your cursor tip.
- Then click right for changing the bus name into connection names and place them with a left click at the bus taps.

Component libraries

In Schematic the options File/Matching Symbol and Add Symbol make it possible to reuse an existing design. For example the 3 bit counter could be connected with the 7-segment decoder. In this context we wish to draw your attention to the existing component libraries.

Library	Description
vanprim.lib	selguide.pdf
vanttl.lib	vanttl.pdf
vanfunc.lib	vanfunc.pdf

Fig. 29.2.22. Component libraries for MACH PLDs

They contain not only simple gates but also complex components such as entire counters, multiplexers and adders. The use of these modules saves a lot of time in the designing and testing process; in this way the design process is performed on the same level as before when using complex TTL devices.

Which of the libraries are available depends on the PLD used. Figure 29.2.22 shows a brief list of the libraries that are most important for the MACH modules. Libraries for the ispLSI1k...8k family are described in the files ispmacro.pdf and 58kmcr.pdf.

In our exercise a BCD counter and a 7-segment decoder from the TTL library are used. Again we generate a counter with a 7-segment output. The corresponding circuit diagram is shown in Fig. 29.2.23.

Enter the sample component with the names from Fig. 29.2.24 or use the Source/Import option to get it from the directory my files\ispLEVER examples. Follow the steps described below to create the circuit diagram:

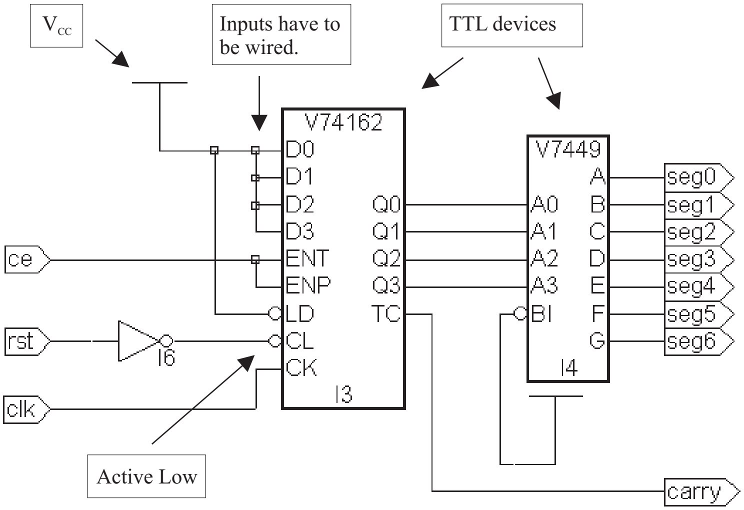


Fig. 29.2.23. TTL modules with wiring

Project name:	library
Directory:	library
Schematic file name:	library

Fig. 29.2.24. Suggested names for example 5 with component libraries

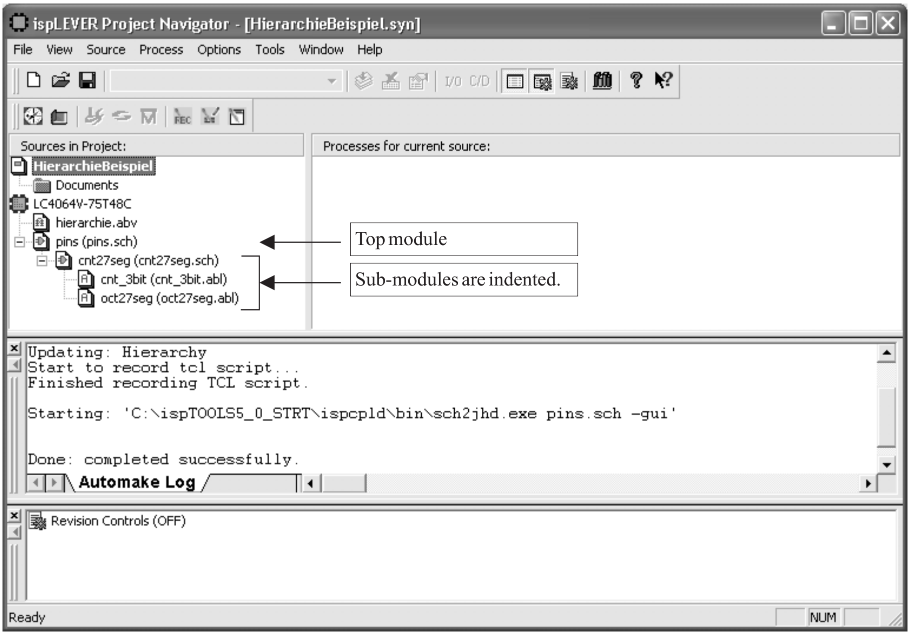
- Insert the TTL modules V74162 and V7449 (library VANTTL.LIB) selected from the Symbol Libraries (Add Symbol) into the diagram.
- Add the required logic gates, inverters and VCC (library GATES.LIB, G_INV and VCC) to the diagram according to Fig. 29.2.23.
- Draw the wires using Add Wire.
- Assign net names to all inputs and outputs. After selecting the Add Net Name option enter the desired names, confirm using Enter and click on the wire; ensure that you click onto the red dot at the end of the line. For the segment designations use the name "seg0+". This causes the program to increase the final digit by one after each designation process.
- Define the designated I/O signals as inputs or outputs using the I/O marker (Add I/O Marker). In the case of several I/Os, all of these can be defined in one step by marking an entire region.

Hierarchy

With ispLEVER you can link or interleaf several modules that have been designed in Abel or Schematic. This is similar to a C or Pascal program with various different functions and procedures. ispLEVER shows the hierarchy using a tree structure.

The simplest method of linking hierarchy levels is to develop the upper-most module as a schematic as shown in Fig. 29.2.25. The desired sub-modules are then integrated into this. In the hierarchy tree these modules appear below the main module. To do so, it may be necessary to import the Abel files used.

In this exercise we will use the counter and the octal-to-seven-segment decoder to create the design shown in Fig. 29.2.26.



Top module

Sub-modules are indented.

```
Updating: Hierarchy
Start to record tcl script...
Finished recording TCL script.

Starting: 'C:\ispTOOLS5_0_STRT\ispcpld\bin\sch2jhd.exe pins.sch -gui'


Done: completed successfully.
```

Fig. 29.2.25. Hierarchy in the Project Manager

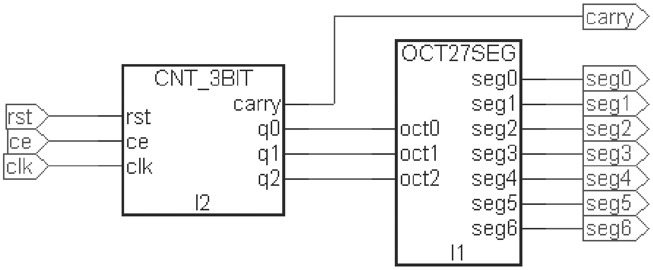


Fig. 29.2.26. Top module in Schematic

Project name:	cnt27seg
Directory:	cnt27seg
Schematic file name:	cnt27seg

Fig. 29.2.27. Suggested names for example 6

Enter the example with the name from Fig. 29.2.27 or use the Project Navigator: Source/Import option to import it from the ispLEVER examples.

Import the previously created Oct27Seg decoder and the first Abel counter (from cnt_3bit_abel_1). Use menu option Source/Import in the Project Navigator and select the corresponding Abel file. Then open a new Schematic design using Source/New…/ Schematic. Add the two symbols from the "local" library to the design.

Finally draw the necessary wires, define them as inputs/outputs and mark them with the corresponding I/O markers.

Use the File/Matching Symbol to generate a symbol for the entire design before quitting the Schematic Editor. This is effectively the same as using Generate Schematic Symbol in the Project Navigator.

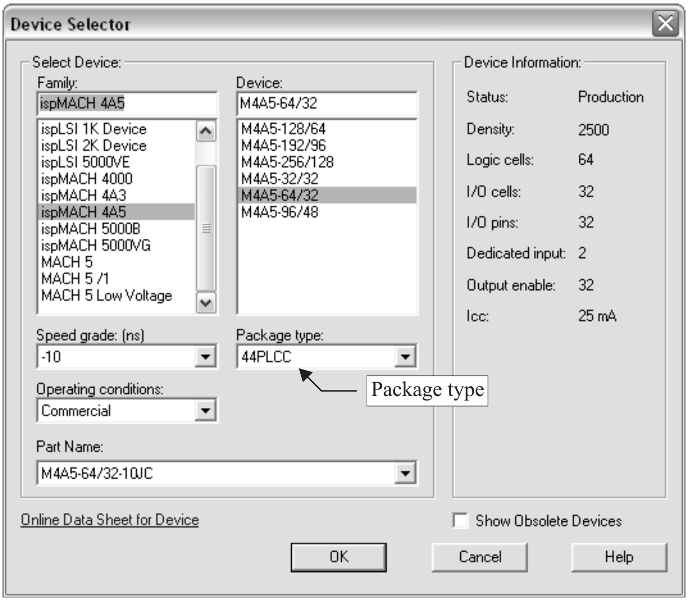
29.2.3 Pin Assignment

If no pins are assigned, the Device Fitter will perform the pin assignment itself when generating the JEDEC file, so that it is best for the internal wiring (also refer to Sect. 29.2.5 Optimization). Manual pin assignment is done either directly in the source file (e.g. Abel or Schematic) as in the examples described or by entering the values in the Constraint Editor.

If the pin assignment is to be taken from a source file, the import option must be chosen in the dialog popping up during the fitting procedure. The fitter then adopts the assignments from the top-level module, i.e. all pin assignments in the sub-modules are discarded.

The information of the specific chip is first fetched from the chip-library when it is used for simulation and for converting the net lists into JEDEC format. Only in such cases does it actually make sense to assign pins.

The device selection window is opened by double clicking on the current device name in the source window. This opens the Device Selector window (Fig. 29.2.28). Don't forget to choose the correct package type.



Family: ispMACH 4A5

Device: M4A5-64/32

Speed grade: (ns) -10

Package type: 44PLCC

Operating conditions: Commercial

Part Name: M4A5-64/32-10JC

Device Information:

Status:	Production
Density:	2500
Logic cells:	64
I/O cells:	32
I/O pins:	32
Dedicated input:	2
Output enable:	32
Icc:	25 mA

□ Show Obsolete Devices

Package type

Fig. 29.2.28. Device Selector window

Pin definition is entered directly with I/O declaration and depends on the current device.

```
DECLARATIONS

    oct0..oct2    pin 2,3,4;
    seg6..seg0    pin 14..20 istype 'com';
```

Fig. 29.2.29. Pin assignment in Abel

Abel

In Abel pins are assigned simply by entering the desired pin number after the key word “pin” (see Fig. 29.2.29). The Oct27Seg decoder is used as an example; here, the pin numbers are shown in bold print. The numbers can be entered either individually or by using the “..” operator.

By assigning the pin numbers directly in the Abel HDL module, which is also used for the function, the designer is forced to adapt the pin numbers accordingly when changing to another module or when importing the module into another project.

Schematic

Pin assignment can be done in any Schematic file. However, it is easier and more common to combine the entire circuit in a single block and to assign the pins in a top-level schematic (see Fig. 29.2.30).

Create a new project and import the files shown in Fig. 29.2.31. Generate a new Schematic module named “pins” and perform the following steps:

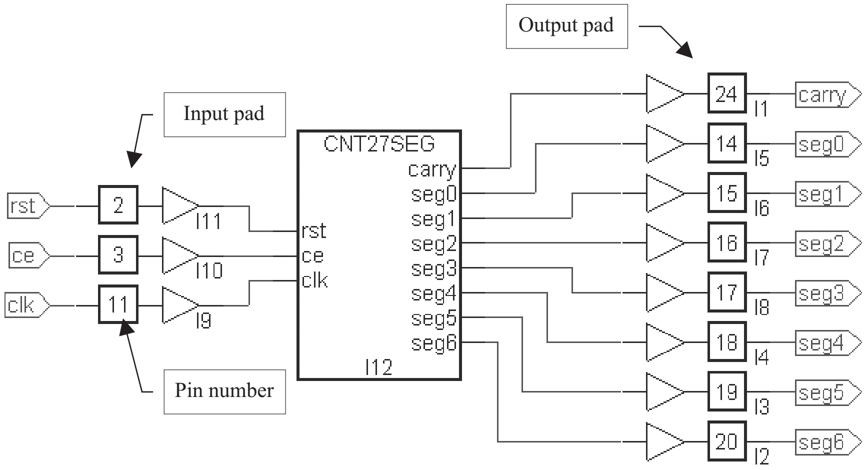
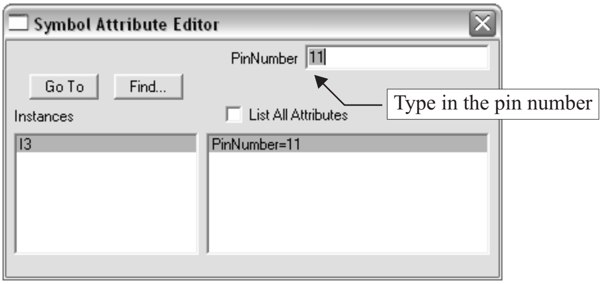


Fig. 29.2.30. Pin assignment in Schematic

Project name:	cnt27seg
Directory:	cnt27seg
Schematic file name:	cnt27seg
Import files:	cnt27seg\cnt27seg.sch
	cnt27seg\cnt_3bit.abl
	cnt27seg\oct27seg.abl

Fig. 29.2.31. Suggested names for example 7

- Insert the circuit symbol cnt27seg from the library ("local")
- Insert the I/O pads (library I/OPADS.LIB)
- Draw wires to connect the I/O pads to the module cnt27seg
- Add short wires to the outside of the I/O pads
- Add the input and output names at the end of the wiresProvide the inputs/outputs with I/O markers
- Open the Symbol Attribute Editor from the tool box (see Fig. 29.2.32)
- Mark one I/O pad after the other and assign the according pin number.



PinNumber 11

Type in the pin number

I3

PinNumber=11

Fig. 29.2.32. Symbol Attribute Editor

Constraint Editor

By the use of the Constraint Editor the pin numbers are assigned to the signals by hand. To start this editor mark the device entry in the source window and double-click at Constraint Editor in the process window.

Double click at the "carry" signal in the left window (see Fig. 29.2.33). The selected signal appears as entry on the right. Then double click t the "pin" cell and enter the pin number 25. In the GLB cell the logic block of the device is assigned automatically, which indicates the geometrical placement of the signal on the chip.

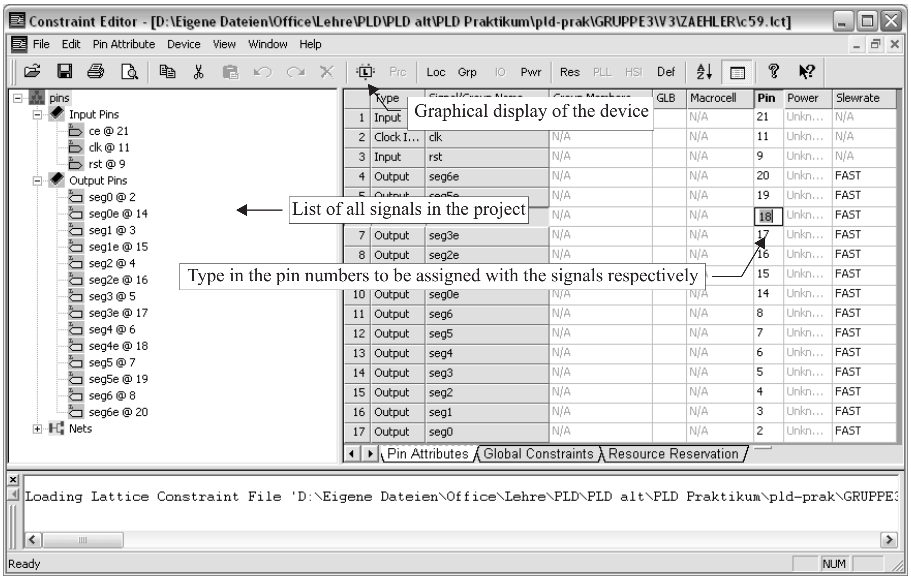
The "carry" signal is now assigned to pin 25. In order to undo an assignment click at the signal with the right mouse button and choose "clear selected".If you have assigned the pin numbers in the schematic or ABEL file these numbers are taken for initializing the constraint editor. You are also able to change the pin numbers in this case by the help of the constraint editor.

Depending on the device selected other location assignments are possible, e.g. the allocation to a special PAL-block or the macro cell which is to calculate the signal.

Open the "pins" project and start the Constraint Editor. Click on the "Loc" button and mark the "carry" signal. Then mark pin number 25. Confirm the assignment by clicking on Add. The "carry" signal is now included in the list underneath. In order to undo an assignment use the cancel command. Alternative you mark the "carry" signal, select pin number 24 and confirm this with the "Update" button. If the Import Source Constraint option is active, the signals are shown instantly in the list underneath.

The assignment can be changed in 3 steps:

- mark the signal in the window at the bottom
- Modify transfers the signal to the upper window
- here it can be connected to the desired pin
- confirm with update



File
Edit
Pin Attribute
Device
View
Window
Help
Prc
Loc
Grp
IO
Pwr
Res
PLL
HSI
Def
Graphical display of the device
List of all signals in the project
Type in the pin numbers to be assigned with the signals respectively
Pin Attributes
Global Constraints
Resource Reservation
NUM

Constraint Editor - [D:\Eigene Dateien\Office\Lehre\PLD\PLD alt\PLD Praktikum\pld-prak\GRUPPE3\V3\ZAEHLER\c59.lct]

- pins
  - Input Pins
    - ce @ 21
    - clk @ 11
    - rst @ 9
  - Output Pins
    - seg0 @ 2
    - seg0e @ 14
    - seg1 @ 3
    - seg1e @ 15
    - seg2 @ 4
    - seg2e @ 16
    - seg3 @ 5
    - seg3e @ 17
    - seg4 @ 6
    - seg4e @ 18
    - seg5 @ 7
    - seg5e @ 19
    - seg6 @ 8
    - seg6e @ 20
  - Nets

	Type	Signal/Group Name	Group Members	GLB	Macrocell	Pin	Power	Slewrate
1	Input	[illegible]	[illegible]	[illegible]	N/A	21	Unkn...	N/A
2	Clock I...	clk	N/A		N/A	11	Unkn...	N/A
3	Input	rst	N/A		N/A	9	Unkn...	N/A
4	Output	seg6e	N/A		N/A	20	Unkn...	FAST
5	Output	seg5e	N/A		N/A	19	Unkn...	FAST
[illegible]	[illegible]	[illegible]	N/A		N/A	18	Unkn...	FAST
7	Output	seg3e	N/A		N/A	17	Unkn...	FAST
8	Output	seg2e	N/A		N/A	16	Unkn...	FAST
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	15	Unkn...	FAST
10	Output	seg0e	N/A		N/A	14	Unkn...	FAST
11	Output	seg6	N/A		N/A	8	Unkn...	FAST
12	Output	seg5	N/A		N/A	7	Unkn...	FAST
13	Output	seg4	N/A		N/A	6	Unkn...	FAST
14	Output	seg3	N/A		N/A	5	Unkn...	FAST
15	Output	seg2	N/A		N/A	4	Unkn...	FAST
16	Output	seg1	N/A		N/A	3	Unkn...	FAST
17	Output	seg0	N/A		N/A	2	Unkn...	FAST

```
Loading Lattice Constraint File 'D:\Eigene Dateien\Office\Lehre\PLD\PLD alt\PLD Praktikum\pld-prak\GRUPPE3
```

Ready

Fig. 29.2.33. Constraint Editor

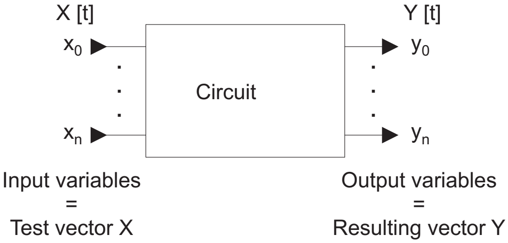


Fig. 29.2.34. Test vector

29.2.4 Simulation

Trouble shooting is the prime purpose of design analysis. The behaviour of the newly designed circuit is checked by way of simulation. On the other hand the timing/frequency analysis is suitable for checking the functionality under given conditions, such as the maximum frequency. ispLEVER features an integrated simulator which applies the input signals described in a test vector file to the design and calculates the resulting signal forms (see Fig. 29.2.34).

These input signals are the test vectors that are applied to the design one after the other. Another option is to compare the simulation results with a given result vector.

Whether the results of such simulation are useful and the extent (in terms of percentage) to which they test the design depends on the selection of suitable input signals. Therefore it is important to define suitable start conditions. For example the registers should be reset prior to initial use.

Flip-flops accept input signals only in the event of a rising clock pulse edge. For this reason, signals that occur after a positive test pulse edge and disappear before the next pulse have no effect. Furthermore, the input signals of flip-flops must not change during the positive clock pulse edge since otherwise they do not comply with the set-up and hold time, which would result in undefined conditions.

The results of the simulation can be viewed in the Waveform Viewer. All signal forms can be examined here.

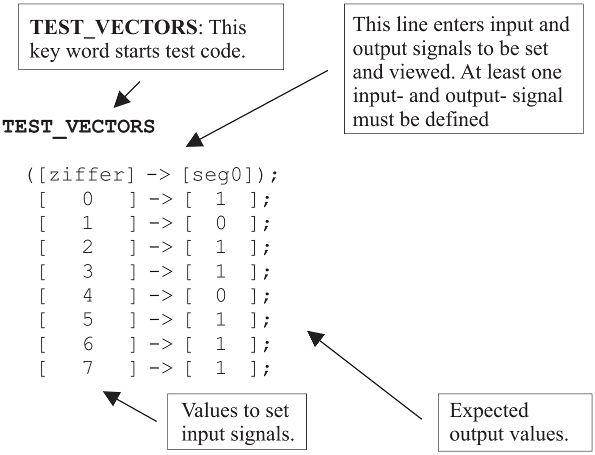
Test vectors

Test of combinatorial circuits: Open the Oct27Seg project and open the Abel file. Insert the text after the truth table as shown in Fig. 29.2.35. Only one input and one output signal must be given in the test vector. All the other output signals can be viewed in the Waveform Viewer.

Now mark the Oct27Seg vectors in the Project Navigator and start the functional simulation. In the Simulator Control Panel the simulation is activated via the Simulate/Run button. The Waveform Viewer opens and shows the simulation result. A more detailed description of the operation is given further below.

Testing sequential logic: In this example a test vector file that is separate from the Abel file is to be created in order to test the counter.

Open the "cnt_3bit" project in the "cnt_3bit_abel_1" directory, create a new source test vectors module and enter the program according to Fig. 29.2.36. The test vectors are saved in a separate file, which provides the advantage that the test vector file can be used



TEST_VECTORS: This key word starts test code.

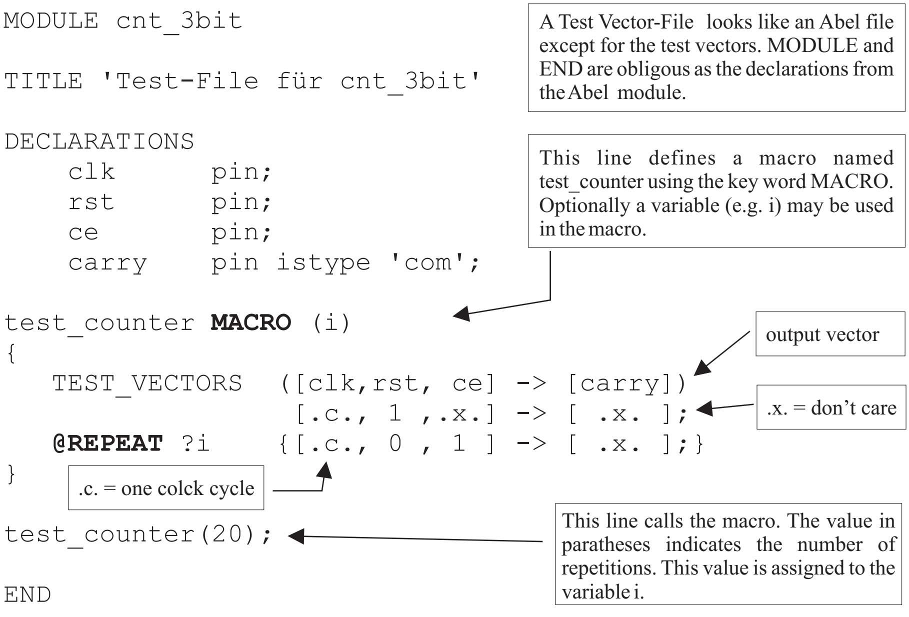
This line enters input and output signals to be set and viewed. At least one input- and output- signal must be defined

```
TEST_VECTORS

 ([ziffer] -> [seg0]);
  [   0   ] -> [  1  ];
  [   1   ] -> [  0  ];
  [   2   ] -> [  1  ];
  [   3   ] -> [  1  ];
  [   4   ] -> [  0  ];
  [   5   ] -> [  1  ];
  [   6   ] -> [  1  ];
  [   7   ] -> [  1  ];
```

Values to set input signals.

Expected output values.

Fig. 29.2.35. Structure of test vectors



```
MODULE cnt_3bit

TITLE 'Test-File für cnt_3bit'

DECLARATIONS
    clk      pin;
    rst      pin;
    ce       pin;
    carry    pin istype 'com';

test_counter MACRO (i)
{
   TEST_VECTORS  ([clk,rst, ce] -> [carry])
                  [.c., 1 ,.x.] -> [ .x. ];
   @REPEAT ?i    {[.c., 0 , 1 ] -> [ .x. ];}
}

test_counter(20);

END
```

A Test Vector-File looks like an Abel file except for the test vectors. MODULE and END are obligous as the declarations from the Abel module.

This line defines a macro named test_counter using the key word MACRO. Optionally a variable (e.g. i) may be used in the macro.

output vector

.x. = don't care

.c. = one colck cycle

This line calls the macro. The value in paratheses indicates the number of repetitions. This value is assigned to the variable i.

Fig. 29.2.36. Test vector file for the counter

in different projects. In other words, this test vector file can be used for all three "cnt_3bit" projects.

Test vectors are also used for stimulating counters. In this case macros are used. This allows longer simulations to be set up without every time increment having to be specified explicitly.

First you define the "test_counter" macro via the key word "macro". Please note that the entire macro is in parentheses. The variable (i in this example) is used to specify the number of repetitions. The macro itself has a structure similar to that of a common test vector. In order to avoid having to specify every single step, the repeat command ensures that the test vector is repeated i number of times. Several test vectors could also be used – the important thing is to pay close attention to the use of parentheses. To activate the macro, its designated name must be entered; the value in parenthesis specifies the number of repetitions desired.

The simulator knows how to interpret the values ".c." and ".x.". ".c." means that the simulator has to generate a clock pulse, while ".x." (don't care) instructs the simulator to ignore the values.

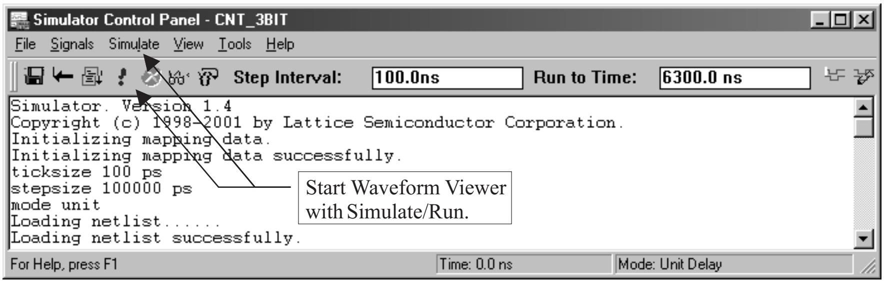
The test vector must again contain at least one input and one output element. Other signals can be selected and viewed in the Waveform Viewer. The test vector file created can now be used for the two other counter projects and can be imported into the library, hierarchy and pins projects; these projects are then ready for testing. Observe: there must exist only one test vector file in a project.

Waveform Viewer

Open the hierarchy project, import the test vector file of the counter and start the simulation. This opens the Simulator Control Panel (see Fig. 29.2.37), which allows several settings to be made before the actual simulation is started (Simulate/Run). The Waveform Viewer (see Fig. 29.2.38) is the most suitable tool in the ispLEVER software package for signal representation.

All signals available for viewing (see Fig. 29.2.39) can be displayed via the Show command in the Edit menu. Instances indicates the current level of signals; the corresponding signals can be found under **Nets**. Double clicking on a **net** displays the given signal. Alternatively you can mark one or more signals and view them via the **Show** option. A marked signal shown in the plot-window can be deleted when it is marked by the edit/hide command.

In order to show a summary of the corresponding output signals when viewing a counter, you have to use the bus option. In our example we wish to view the internal counter output only. This is done by double clicking on D underneath Instances. This displays additional internal signals. Expand the window by pressing the **Bus** button and enter a bus name. Then mark the **Nets** N_1, N_2 and N_3. The signals marked are added to the active bus via **Add Net(s)**. The sequence of the signals, i.e. the significance within the bus, can be reversed using the **Reverse** button. With **Save Bus** the bus can be saved and subsequently displayed via **Show**. The sate of the bus is shown in decimal notation. **New Bus** allows you to create additional buses.



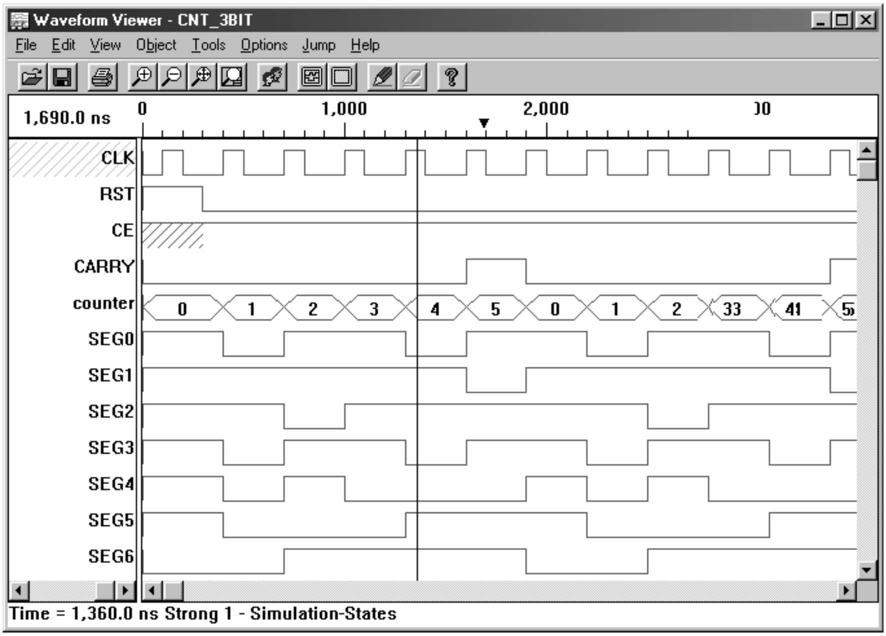
Step Interval: 100.0ns
Run to Time: 6300.0 ns

```
Simulator. Version 1.4
Copyright (c) 1998-2001 by Lattice Semiconductor Corporation.
Initializing mapping data.
Initializing mapping data successfully.
ticksize 100 ps
stepsize 100000 ps
mode unit
Loading netlist......
Loading netlist successfully.
```

For Help, press F1 | Time: 0.0 ns | Mode: Unit Delay

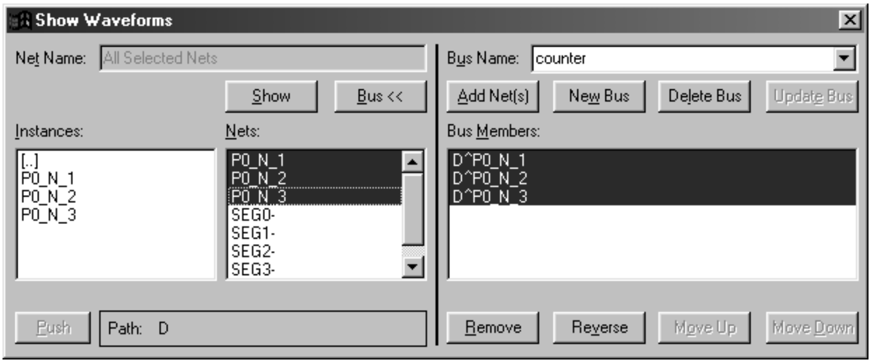
Start Waveform Viewer with Simulate/Run.

Fig. 29.2.37. Simulator Control Panel



Time = 1,360.0 ns Strong 1 - Simulation-States

Fig. 29.2.38. Signal forms in the Waveform Viewer



Net Name: All Selected Nets

Bus Name: counter

Instances:
[..]
P0_N_1
P0_N_2
P0_N_3

Nets:
P0_N_1
P0_N_2
P0_N_3
SEG0-
SEG1-
SEG2-
SEG3-

Bus Members:
D^P0_N_1
D^P0_N_2
D^P0_N_3

Path: D

Fig. 29.2.39. Show Waveforms dialog window

The Waveform Viewer allows you to zoom in on an image. Click on the Zoom In option in the View menu. Mark the desired area in the display or simply click in the display to increase the zoom.

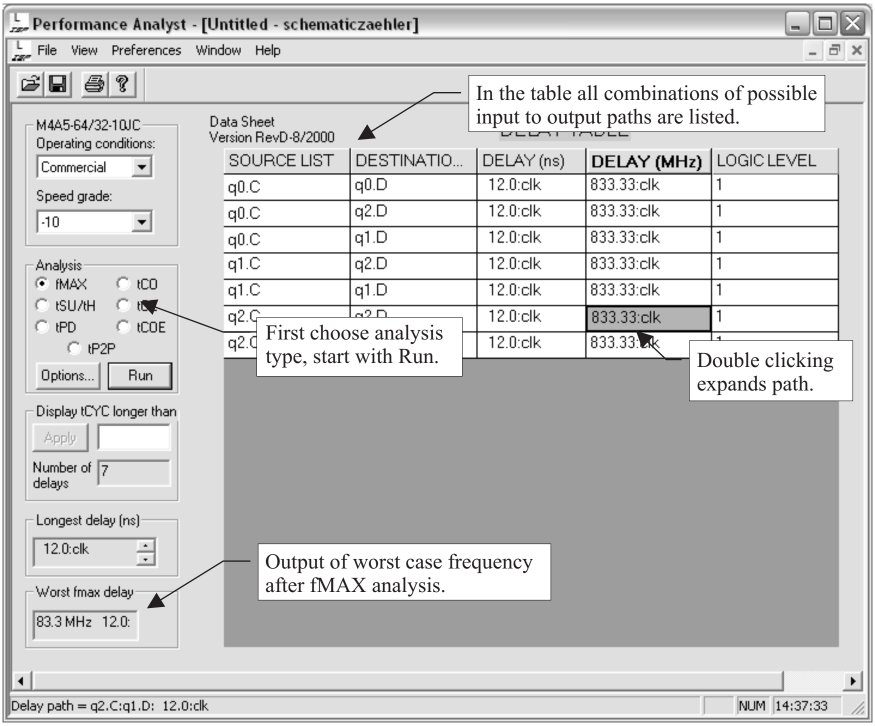
Functional simulation performs solely a functional analysis, while timing simulation takes the real chip and rooting into consideration. This requires that the signals are connected to pins. However, calculations become far more complex, which take much computation time in large-scale projects.

Time and Frequency Analysis

With ispLEVER you can calculate the delay times of the signals in the circuit, although this is only possible with newer components and does not work with old GALs.

Timing analysis can be started as soon as a MACH component has been selected and marked in the source window. Double click on the Timing Analysis button in the process window.

On the left hand side you can choose from several analysis types (see Fig. 29.2.40). The analysis of the maximum frequency is most important for obtaining a first impression. This illustrates the general usability of the design. If you continue to point the cursor to



Performance Analyst - [Untitled - schematiczaehler]
File View Preferences Window Help

M4A5-64/32-10JC
Operating conditions: Commercial
Speed grade: -10

Analysis
● fMAX ○ tCO
○ tSU/tH ○ tC
○ tPD ○ tCOE
○ tP2P

Display tCYC longer than

Number of delays: 7

Longest delay (ns): 12.0:clk

Worst fmax delay: 83.3 MHz 12.0:

Data Sheet
Version RevD-8/2000

SOURCE LIST	DESTINATIO...	DELAY (ns)	DELAY (MHz)	LOGIC LEVEL
q0.C	q0.D	12.0:clk	833.33:clk	1
q0.C	q2.D	12.0:clk	833.33:clk	1
q0.C	q1.D	12.0:clk	833.33:clk	1
q1.C	q2.D	12.0:clk	833.33:clk	1
q1.C	q1.D	12.0:clk	833.33:clk	1
q2.C	q2.D	12.0:clk	833.33:clk	1
q2.C		12.0:clk	833.33:clk	1

Delay path = q2.C:q1.D: 12.0:clk
NUM 14:37:33

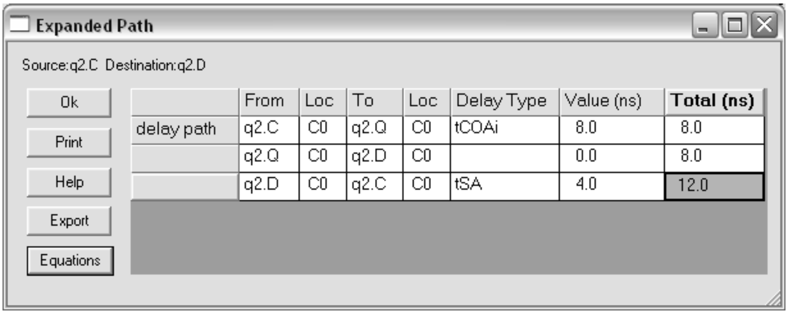
In the table all combinations of possible input to output paths are listed.

First choose analysis type, start with Run.

Double clicking expands path.

Output of worst case frequency after fMAX analysis.

Fig. 29.2.40. Timing analysis



Source:q2.C Destination:q2.D

	From	Loc	To	Loc	Delay Type	Value (ns)	Total (ns)
delay path	q2.C	C0	q2.Q	C0	tCOAi	8.0	8.0
	q2.Q	C0	q2.D	C0		0.0	8.0
	q2.D	C0	q2.C	C0	tSA	4.0	12.0

Fig. 29.2.41. Timing analysis, logic path information

one of the seven points for several seconds, you will receive more information. The Run option starts the analysis chosen.

The analysis result is presented in the window to the right. Detailed information on the transit times in the chip can be obtained by double clicking on one of the elements (see Fig. 29.2.41).

29.2.5 Optimization

There are different ways of improving a design. First you have to define the intended design goal. Is it to consume as few resources as possible or is it to achieve the highest possible clock frequency?

With the information obtained from the analysis you can alter the allocated modules in order to improve the design.

A simple way of increasing the permissible frequency is to choose a faster chip. However, such a chip may not be readily available on the market or may draw a too much current or may be too expensive.

Often the only remaining option is to change the existing design. This may involve changing the pin assignment (Constraint Editor) or redefining the operating conditions of the Fitter in the "Optimization Constraint Editor" in the process window of the device entry.

Maximum frequency versus minimum space requirement: By activating the "Optimization Constraint Editor" from the process window of the device entry you are able to configure the fitting constraints. By double clicking each table entry respectively you are able to select the possible values from the drop down menu. By changing line 6 ("node_collapsing_mode") from "Speed" to "Area" the fitter is optimized for bringing big designs in a small chip: The logic cells of the chips are used to full capacity. Use this option if the fitting process failes with "need more pins". Be aware that the area option optimized packing the design at the cost of signal transition time.

In small designs, as in the examples described here, these effects are not noticeable as the module capabilities are not fully utilised.

Without fixed pin assignment: If the desired maximum frequency cannot be achieved by the methods described above, you still have the possibility of not defining the pin assignment. This enables the fitter to spread the design more suitably on the chip. But this means that the printed circuit board has to be adapted to the pin assignment of the PLD.

Constraint Editor: The Constraint Editor not only allows you to change the pin assignment in some chip types but also to influence the assignment of the macrocells, blocks and segments. So you can force corresponding functions in one PAL-block and reduce transmission times on the chip.

29.2.6 Programming

Generally, the newer PLDs are programmable in the circuit (ISP – In System Programmable). This makes programming equipment unnecessary since the required programming logic is located on the chip. Sometimes "ISP" is included in the model designation,

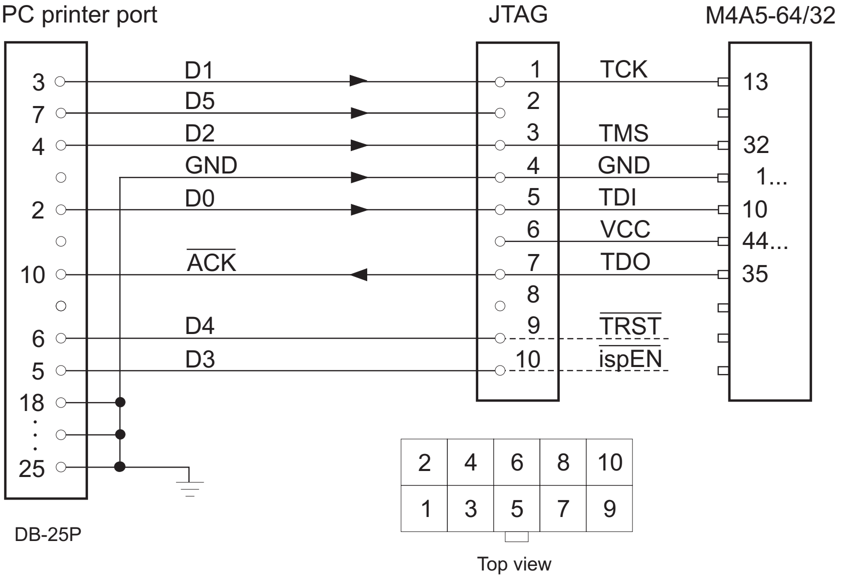


Fig. 29.2.42. Passive JTAG download cable with connections according to Lattice

as is the case with the chip from the ispM4A family used in our example. Programming is done via the standardised JTAG interface[1] which is also used to test the circuits.

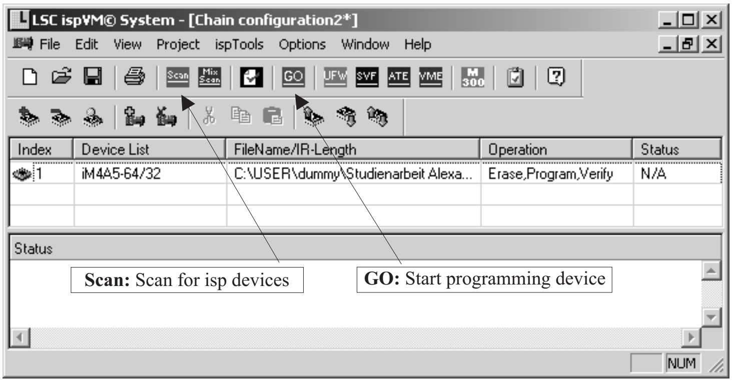
To program the component all you need is a download program to transfer the JEDEC file to the chip via a download cable. The download program for Lattice-products is named ispVM. ispLEVER software contains the download program ispVM which uses the parallel port (printer interface). The necessary connections are shown in Fig. 26.2.42.

A download cable can be ordered from Lattice under the order number HW-DL-3C and is used to connect the PC parallel interface to the standardised JTAG plug on the printed board of the PLD. As an extra measure, the Lattice cable also features integrated drivers (74VHC244) to guarantee the correct levels at the PLD even in unfavourable conditions. In most cases a simple passive cable like the one shown in Fig. 29.2.42 is sufficient. For

JTAG pin	Signal name	Meaning
1	TCK	Test clock
2		not used
3	TMS	Test mode select
4	GND	Ground
5	TDI	Test data in
6	VCC	Interface supply
7	TDO	Test data out
8		not used
9	TRST	Test reset
10	ispEN	Enable programming

Fig. 29.2.43. Signals in the JTAG connector

[1] IEEE 1149.1 Boundary Scan Test Interface from the Joint Test Action Group (JTAG)

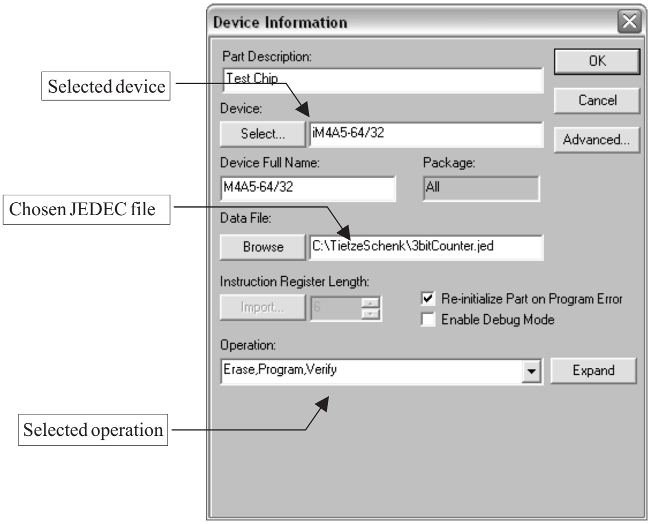

LSC ispVM© System - [Chain configuration2*]
File Edit View Project ispTools Options Window Help
Scan
Mix Scan
GO
UFW
SVF
ATE
VME
M 300
Status
NUM

Index	Device List	FileName/IR-Length	Operation	Status
1	iM4A5-64/32	C:\USER\dummy\Studienarbeit Alexa...	Erase,Program,Verify	N/A

Scan: Scan for isp devices

GO: Start programming device

Fig. 29.2.44. ispVM-System



Part Description: Test Chip

Device: iM4A5-64/32

Device Full Name: M4A5-64/32

Package: All

Data File: C:\TietzeSchenk\3bitCounter.jed

Instruction Register Length: 6

☑ Re-initialize Part on Program Error

□ Enable Debug Mode

Operation: Erase,Program,Verify

Selected device

Chosen JEDEC file

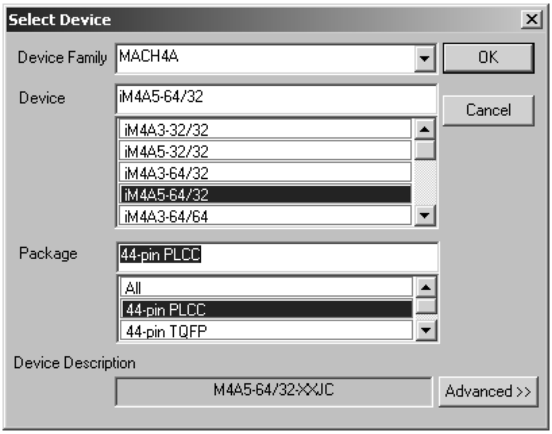
Selected operation

Fig. 29.2.45. JTAG Part Properties

the JTAG connection the download cable has a 10-pin socket like those commonly used on interface cables for PC mainboards. The matching 10-pin plug is located on the printed board of the PLD to be programmed. The names and meaning of the signals in the JTAG interface are listed in Fig. 29.2.43. As seen in our example, the TRST and the ENABLE signals are not required in many PLDs.

To program a device start the program ispVM (Fig 29.2.44).

- At first install the download-cable, insert the PLD and switch the power on for the board.
- Press the “Scan” button in Fig. 29.2.44. Now all PLDs in the chain of the circuit are scanned and displayed in the right order.
- Double click on the Device in the list to be changed. Insert in the dialog the information on the Device by choosing Select (resulting in the menu in Fig. 29.2.46), the Data File (JEDEC-file) and programming operation as seen in Fig. 29.2.45.


Select Device

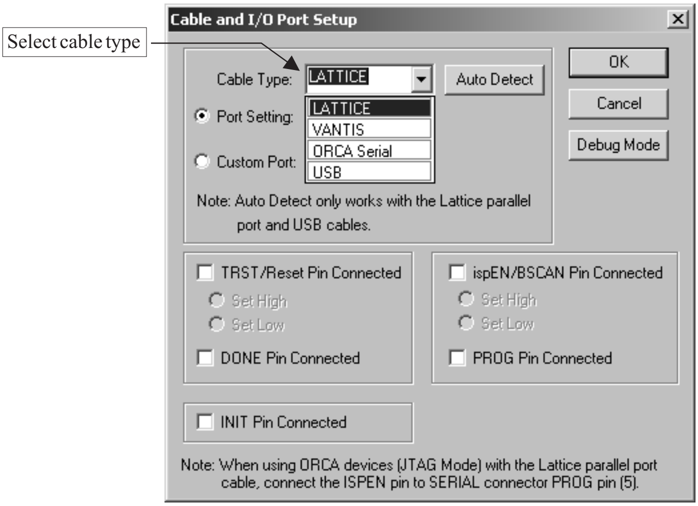
Device Family: MACH4A

Device: iM4A5-64/32

Package: 44-pin PLCC

Device Description: M4A5-64/32-XXJC

Fig. 29.2.46. Select Device


Select cable type

Cable and I/O Port Setup

Cable Type: LATTICE

● Port Setting:

○ Custom Port:

Note: Auto Detect only works with the Lattice parallel port and USB cables.

□ TRST/Reset Pin Connected
○ Set High
○ Set Low
□ DONE Pin Connected

□ ispEN/BSCAN Pin Connected
○ Set High
○ Set Low
□ PROG Pin Connected

□ INIT Pin Connected

Note: When using ORCA devices (JTAG Mode) with the Lattice parallel port cable, connect the ISPEN pin to SERIAL connector PROG pin (5).

Fig. 29.2.47. Programming Cable Selection

- With Options/Cable and I/O Port Setup you select the PC port and the type of the download-cable (Fig. 29.2.47).
- Start programming with "Go" from the main window (Fig. 29.2.44)

29.2.7 Outlook

We hope that the information provided in this chapter will enable you to design your own circuits using ispLEVER and program them in a PLD. For those of you requiring extra or additional information we suggest using the help menue in ispLEVER.

If a design is not only to be simulated but also to be tested in a circuit, the evaluation board from Lattice is particularly useful. The provided configuration differs for the mounted devices CPLDs and FPGAs.