

Designing ASICs with the ADK Design Kit and Mentor Graphics Tools



Version 1.7

The purpose of this document is to provide university students with the basic flow and procedures for using Mentor Graphic' design tools with ADK design kit. The usage of this document is intended for student use only and in no way should be perceived as a complete process guide. Please refer to standard Mentor Graphics documents for complete process information.

Chapter 1: Installing the ADK	1-1
1.1. Setting up the Mentor/ADK Environment	1-2
1.1.1. Setting Your Environment (c-shell example)	1-2
1.1.2. Setting Your Location Map	1-3
1.2. Setting up Student Accounts	1-3
1.2.1. Common User Files	1-3
1.2.2. Organizing Design Data (Optional)	1-3
1.3. Setting up Leonardo	1-4
1.3.1. Setting up the Leonardo GUI	1-4
1.4. Checking your ADK Version	1-5
Chapter 2: Using as little UNIX as possible....	2-1
2.1. Setting up your Environment	2-1
2.2. Moving around the system	2-1
2.2.1. cd - Change Directory	2-1
2.2.2. pwd - Print Working Directory	2-2
2.2.3. ls - LiSt	2-2
2.2.4. cp - CoPy	2-2
2.2.5. mv - MoVe	2-3
2.2.6. rm - ReMove	2-3
2.2.7. more	2-3
2.2.8. chmod - CHange permissions	2-3
2.3. Printing your files	2-3
2.3.1. lp - printing ascii and postscript files	2-3
2.3.2. lpstat - Viewing the printer status	2-4
2.4. Processes	2-4
2.4.1. ps	2-4
2.4.2. kill	2-4
2.5. Obtaining more help on UNIX commands	2-4
Chapter 3: Simulating HDL in ModelSim	3-1
3.1. Compiling HDL code	3-1
3.2. Compiling the ADK cell libraries	3-2
3.3. Invoking ModelSim	3-2
3.4. Setting up the ModelSim windows	3-3
3.5. Applying stimulus	3-3
3.5.1. Using Force Files (interactive)	3-3
3.5.2. Force File Example 1	3-4
3.5.3. Force File Example 2 (loops)	3-4
3.5.4. VHDL Test Bench example	3-5
3.6. Running the Simulator	3-6
Chapter 4: Synthesizing VHDL or Verilog	4-1

Chapter :

4.1. Synthesizing HDL Using Leonardo (GUI Mode)	4-1
4.2. Synthesizing HDL Using Spectrum (Command-Line Mode)	4-3
4.3. Creating a EDDM Schematic from EDIF	4-4

Chapter 5: Designing for Testability 5-1

5.1. Inserting Scan chains	5-1
5.2. Generating Test Vectors	5-2

Chapter 6: Creating a Schematic 6-1

6.1. Invoking Design Architect	6-1
6.2. Creating a Schematic	6-1
6.2.1. Opening a Schematic Sheet	6-1
6.2.2. Choosing and Placing Component Symbols on a Sheet	6-2
6.2.2.1. Using the ADK_Library Palette to Place a Component Symbol	6-2
6.2.2.2. Using the Dialog Navigator to Place a Component Symbol	6-3
6.2.2.3. Using the Active Symbol Window to Place a Component Symbol	6-3
6.2.3. Drawing a Net/Bus	6-4
6.2.3.1. Naming Nets	6-4
6.2.3.2. Using Buses	6-4
6.2.4. Annotating Properties	6-5
6.2.5. Adding Input and Output Ports	6-5
6.2.6. Adding Power and Ground Symbols	6-5
6.2.7. Creating Comment Objects on a Schematic Sheet	6-6
6.2.8. Checking a Schematic for Errors	6-6
6.2.9. Saving the Schematic	6-6
6.3. Creating a Symbol from a Schematic	6-7
6.3.1. Adding or Modifying Other Symbol Properties	6-7
6.3.2. Creating Comment Objects on a Symbol	6-8
6.3.3. Checking a Symbol	6-9
6.3.4. Saving a Symbol	6-9

Chapter 7: Automated IC Layout using IC Station 7-1

7.1. Creating Design Viewpoints	7-1
7.2. Invoke IC Station	7-2
7.3. Autoplacement and Routing	7-2
7.4. Verifying Project Layout	7-5
7.5. Full Layout Verification	7-6

Chapter 8: Schematic Driven Layout using IC Station 8-1

8.1. Creating Design Viewpoints	8-1
8.2. Invoke IC Station	8-2
8.3. Creating a cell for SDL	8-2
8.4. Placing components into your layout	8-3
8.4.1. AutoPlace instances	8-3

8.4.2. Manually placing instances	8-3
8.5. Placing ports into your layout	8-3
8.6. Routing your cell	8-4
8.6.1. Semi-automatic routing	8-4
8.6.2. Routing with paths	8-4
8.6.3. Routing by placing shapes	8-4
8.6.4. Placing contacts and vias	8-5
8.6.5. Adding well contacts	8-5
8.7. Verifying your layout	8-5
8.7.1. DRC of your cell	8-5
8.7.2. LVS of your cell	8-6
Chapter 9: Generating Padframes	9-1
9.1. Top Level Layout Preparation	9-1
9.2. Padframe Layout	9-2
Chapter 10: Performing Post-Layout Verification using Mach-TA .	10-1
10.1. Preparing to Simulate the Design	10-1
10.1.1. Creating Test Vectors	10-1
10.1.2. Creating a Command File	10-1
10.2. Running Test Vectors and Viewing Results	10-2
Chapter 11: Simulating the Design Using QuickSim II	11-1
11.1. Setting up the Design Viewpoint	11-1
11.2. Invoking QuickSim II	11-1
11.3. Setting up the SimView Windows	11-2
11.3.1. Opening the Schematic View Window	11-2
11.3.2. Using a Trace Window	11-2
11.3.2.1. Opening a Trace Window	11-3
11.3.2.2. Adding a Signal to the Trace Window	11-3
11.3.2.3. Deleting a Signal from the Trace Window	11-3
11.3.3. Using a List Window	11-4
11.3.4. Using a Monitor Window	11-4
11.4. Generating Stimulus	11-5
11.4.1. Creating a Waveform Using Palette Icons	11-6
11.4.2. Creating a Waveform Using the Waveform Editor	11-7
11.4.3. Creating a Force File	11-7
11.4.3.1. Saving Waveform Stimulus in a Forcefile	11-8
11.4.3.2. Loading and Viewing Waveform Stimulus from a Forcefile	11-9
11.5. Running the Simulator	11-10
11.6. Changing the Timing Mode in the Kernel	11-10
11.7. Resetting the Simulator	11-10
11.8. Using QuickSim to Debug your design	11-11
11.8.1. Using Breakpoints	11-11
11.8.2. Back Tracing X States	11-12

Chapter :

11.8.3. Design Changes	11-12
11.8.4. Viewing a Simulation Timing Report	11-12
11.8.5. Using Cursors to Get Waveform Information	11-13

Chapter 12: Performing Static Timing Analysis12-1

12.1. Setting Up and Invoking SST Velocity	12-1
12.2. Setting Timing Constraints	12-1
12.2.1. Defining Clocks	12-2
12.2.2. Defining Constraints on Input Pins	12-2
12.2.3. Defining Constraints on Output Pins	12-2
12.2.4. Defining False Paths	12-2
12.2.5. Defining Constant Levels	12-3
12.2.6. Back Annotating from a SDF File	12-3
12.3. Timing Analysis	12-3

Chapter 13: ADK Menus and Commands13-1

Installing the ADK

FTP the tar file from *Mentor Graphics* using the ADK username/password. The file you need to obtain is `adk.tar.gz`. Place this file in the directory where you want the kit installed.

Use GNUzip to uncompress the archive and untar it in this directory:

```
gunzip adk.tar.gz ; tar -xf adk.tar
```

After installing the kit, you should be sure to change all the permissions and/or owners of the files to match your site's convention. There is no reason the files need to be owned by any particular user, so feel free to make any user the owner if you would like.

After installing the kit, verify that you have the supported version(s) of the design tools you need. Mentor only supports the library versions and scripts available on the MGC website. Currently, the kit has been verified with the following versions of the tools:

- **Mentor Tools:** C.4
(IC Station v8.7_5.4 recommended - Patch p4052)
- **Leonardo Spectrum:** 1999.1 and higher (current 2000.1a)
- **ModelSim:** 5.3 and higher (current 5.4a)

1.1. Setting up the Mentor/ADK Environment

You will need to set up the following environment variables based upon your operating system type/version and where you installed each of the components. Here is an example of a typical setup:

1.1.1. Setting Your Environment (c-shell example)

```
#####  
## Setup Mentor Software ##  
#####  
  
## MGC_HOME is set to the root of the MGC tree  
setenv MGC_HOME /idea_tree/idea_C.4-F.ss5  
  
## MODEL_HOME is set to the root of the ModelTech tree  
setenv MODEL_HOME /mti/5.3e/modeltech  
  
## EXEMPLAR is set to the root of the Exemplar tree  
setenv EXEMPLAR /exemplar/spectrum_99.1h.ss5  
  
## Add everything to the path according to your hardware  
## platform type. Also, ensure MODEL_HOME comes before  
MGC_HOME  
set path=( $\$EXEMPLAR/bin$   $\$EXEMPLAR/bin/SunOS5$  \  
 $\$MODEL_HOME/sunos5$   $\$MGC_HOME/bin$   $\$path$ )  
  
## Set your license file variables appropriately  
setenv MGLS_LICENSE_FILE 1717@lichost  
  
#####  
## Setup ADK Library ##  
#####  
  
## ADK is set to the root of your installation of the ADK.  
setenv ADK /project/adk  
  
## You must set the MGC_LOCATION_MAP variable to a map that  
## contains the necessary location map entries for the ADK.  
setenv MGC_LOCATION_MAP  $\$ADK/lib/location_map.adk$   
  
## MGC_VELOCITYLIB is set to where the technology  
## files for Velocity reside.  
setenv MGC_VELOCITYLIB  $\$ADK/technology/velocity$   
  
## Add the directory with ADK scripts and programs to your
```

```
path.  
set path=($ADK/bin $path)
```

The environment variable `$MGC_WD` should be set to the current working directory. It is suggested that the alias `swd` be set to `setenv MGC_WD "$pwd"` in the user's `.cshrc` file to make setting the variable easier.

1.1.2. Setting Your Location Map

The ADK requires the following location map entries. These are included in an example location map in `$ADK/lib/location_map.adk` and should be merged with your standard map.

```
MGC_LOCATION_MAP_2  
# Need to point to a valid installation of GENLIB  
$MGC_GENLIB  
idea_tree/libraries/gen_lib  
# Required root of ADK tree  
$ADK  
/project/ADK  
# Set MGC_WD to something that can be overridden in the envi-  
ronment  
$MGC_WD  
/tmp  
# HOME will be replaced with the user's home directory  
$HOME  
/tmp
```

1.2. Setting up Student Accounts

1.2.1. Common User Files

The environment variable `$MGC_WD` should be set to the current working directory. You may want to create an alias `swd` set to `setenv MGC_WD "$pwd"` in the users `.cshrc` file to make setting the variable easier.

1.2.2. Organizing Design Data (Optional)

Project directories contain the data that will become a System, Board, or ASIC design. This directory structure accommodates a combination of any tool use, and allows the division of design activities across multiple members and even across multiple design sites. A common directory structure for all projects is an enabler for efficient design participation of members of the design team as well as an efficient means for archival and transmission of data.

The Project Directory is the highest level of the design tree. An environmental variable `$PROJECT_NAME` is used to reference this directory, and is used as a soft

path to the design database. For the labs used during the term, create a directory called designs in home account and use the \$DESIGNS environment variable. When you start your major project, create a project directory and set a environment variable called \$PROJECT. It is also suggested to place both of these names in the location map so that the environment variables can override them and make the designs easier to manage should you find the need move or archive the data.

For each directory, only certain types of data should be allowed in it:

bin --- binaries, scripts, simulation vector files, etc.

work --- compiled HDL design objects under development

src --- source HDL code

netlist --- EDIF, VHDL, or Verilog netlists

reports --- tools transcripts and reports

1.3. Setting up Leonardo

You need to copy all the data in \$ADK/technology/leonardo to your Leonardo library directory:

```
cp $ADK/technology/leonardo/* $EXEMPLAR/lib
```

This will copy the synthesis libraries and other data necessary for the integration with the Leonardo GUI.

1.3.1. Setting up the Leonardo GUI

If you wish to add the ADK libraries to the GUI in Leonardo, perform the following steps. This is optional and unnecessary if you only wish to run Leonardo in batch mode.

1. `cd $EXEMPLAR/lib`

2. Append the `adk_devices.ini` file to the `devices.ini` file:

```
cp devices.ini d
cat d adk_devices.ini >! devices.ini
rm d
```

3. Edit the `devices.ini` file to add the following `DEVICE` lines. You will need to change the numbers to be sequential after the last number in your `device.ini` file. Also, if you are using version 2000.1a, you will need to change the `DEVICE_NUMBER` line to correctly identify the number of devices you have. Version 2000.1b removed this keyword.

```
DEVICE_152=AMI 1.2u (slow)
DEVICE_153=AMI 1.2u (typ)
DEVICE_154=AMI 1.2u (fast)
DEVICE_155=AMI 0.5u (slow)
DEVICE_156=AMI 0.5u (typ)
```

```
DEVICE_157=AMI 0.5u (fast)
```

4. Now you should run Leonardo and verify that under the ASIC section of the technology flowtab you see ADK. Under ADK you should see all six of these libraries.

1.4. Checking your ADK Version

If you are curious about what version of the ADK you are running (or if instructed to find out for support purposes), you can do this with the *adk_ver* command. The option “-a” will also show you the current versions of several Mentor tools that the ADK supports (such as IC Station, DA, ModelSim, etc.).

Chapter 1: *Installing the ADK*

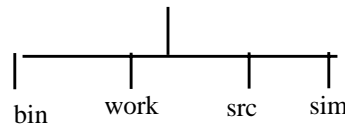
Using as little UNIX as possible....

2.1. Setting up your Environment

Project directories contain the data that will become a System, Board, or ASIC design. Figure 1 represents the development of a design tree from the System, or Board perspective. This directory structure accommodates a combination of any tool use, and allows the division of design activities across multiple members and even across multiple design sites. A common directory structure for all projects is an enabler for efficient design participation of members of the design team as well as an efficient means for archival and transmission of data.

The Project Directory is the highest level of the design tree. An environmental variable \$PROJECT_NAME is used to reference this directory, and is used as a soft path to the design database. For the labs used during the term, create a directory called designs in home account and use the \$DESIGNS environment variable. When you start your major project, create a project directory and set an environment variable called \$PROJECT.

Figure 1. Design Directory Structure



bin => binaries, scripts, etc.

work=> design under development

src => source VHDL code

sim => Simulation Force Files

The following procedure should be used to setup the files/directories to complete this course:

2.2. Moving around the system

2.2.1. cd - Change Directory

The cd command is used to move

- To a subdirectory; type: **cd** *directory*
- To another users directory; type 'cd ~*user_name*'
- To your main directory; type 'cd'

- Back one directory level; type 'cd ..'
- To a lower level directory; type 'cd ../*directory*'

2.2.2. pwd - Print Working Directory

The pwd command is used to tell you where you are located on the system. Simply type 'pwd' and it will echo something like

```
/users/u4/user_name
```

Working with your directories and files:

2.2.3. ls - LiSt

The ls command is used to list out all of your files and directories.

Flags used for this command are:

- **-a** This will show all the file in your directory including the dot files.
Usage: `ls -a`
- **-F** This will indicate directories by at / after the name and executables by a * after the name.
Usage: `ls -aF`

2.2.4. cp - CoPy

The cp command is used to copy files. The syntax is

```
cp source target
```

```
cp file1 file2 copies file1 to a new file named file2
```

The cp command can also be used to copy a file from one directory to another.

- To copy a file into a subdirectory; type:
`cp file directory_name`
This will copy the file to another file of the same name in the subdirectory of `directory_name`
- To copy a file from another users directory; type
`cp ~bob/file .`
This will copy a file from the main directory of the user, `user_name`, to the directory in which you are currently located.

Flags used for this command are:

- **-r** This will copy the directory, all of its files, and any subdirectories to the target directory.
Usage: `cp -r source_directory target_directory`

2.2.5. mv - MoVe

The **mv** command is similar to the **cp** command however it will remove file1 when file2 is created.

2.2.6. rm - ReMove

The **rm** command is used to delete a file. Type '**rm file**'

Flags used with this command:

- **-r** This will remove a directory and all of the files in that directory.

Usage: **rm -r directory**

2.2.7. more

The more command is used to view a file or output of a command one page at a time. Pressing the spacebar will continue on to the next page. Ctrl-C is used to break out.

- To view a file; type
- To view the output of a command; type:

`more file`

`command_name | more`

An example of this is 'ls | more'

2.2.8. chmod - CHange permissions

The chmod command is used to change the protections on your files and directories. The protections can be set for yourself, the group, and others. The permissions are read(4), write(2), and execute(1).

An example of this would be setting a file to be readable, writeable, and executable by yourself only. This would be done by typing:

```
chmod 755 set_dvpt.do
```

2.3. Printing your files

2.3.1. lp - printing ascii and postscript files

The lpr command is used to print out files. If not other options are specified the file will be printed to the LaserJet printer. Every computer has a default printer that it is sent to. See the information on the lpstat command for more details on this. Flags used with this command are:

```
lp -d ps115 file
```

More options are available. Please look in /usr/spool/lp/model and view the appropriate file. (ie laserjet, postscript, etc)

2.3.2. lpstat - Viewing the printer status

The lpstat command is used to check on the status of your printout. Flags used with this command are:

- -a View all status information. This is the most commonly used option.
`lpstat -a`
- -d This will show the system default destination for lpr.
`lpstat -d`
- -o This will show all jobs being output to the named printer, or all queued jobs if no printer is specified.
`lpstat -o <printername>`

2.4. Processes

2.4.1. ps

The ps command is used to report information about active processes. Flags used with this command are:

- -e Print information about all processes.
Usage: **ps -e**
- -f Print a full listing of the processes.
Usage: **ps -ef**
- -u Print information about the processes of a specified user.
Usage: **ps -u user_name** (or **ps -fu user_name**)

2.4.2. kill

The kill command is used to terminate a process(es). Use the ps command to determine the process_id (PID) of the process that you will terminate.

Usage: kill PID_from_ps_command

Flags used with this command are:

- -9 A sure kill.
Usage: **kill -9 PID**

2.5. Obtaining more help on UNIX commands

On-line help for commands is available. Simply type 'man command_name

Simulating HDL in ModelSim

You can invoke the ModelSim simulator on any compiled VHDL or Verilog design. Once you invoke ModelSim, you can apply stimulus to the design, run the simulation, analyze the results, and modify the design based on those results. You can then reset the simulator, optionally revise or apply more stimulus to the design, and start the cycle over. When the design functions correctly, you can save the stimulus and simulation results directly with the design.

The typical strategy for simulation is an iterative process that will be utilized during two steps of the overall design process:

- verify functionality of HDL code prior to synthesis
- verify functionality once the design is placed and routed.

Although the focus of each phase is different, the tasks that you perform within the simulator are very similar.

The following procedure describes how to simulate an HDL design:

3.1. Compiling HDL code

1. In a UNIX shell, change to the directory containing your HDL source code. For example:

```
cd $DESIGNS/src
```

2. If you have not already done so, map the Logical HDL library to a physical directory:

```
vlib phy_lib_path
vmap logical_lib phy_lib_path
```

```
vlib $DESIGNS/hdl/work
vmap work $DESIGNS/hdl/work
```

In this example, all files compiled into the “work” library would be saved in the \$DESIGNS/hdl/work directory. This information is saved in the modelsim.ini file.

3. Run the HDL compiler by typing the following on the UNIX command line:

```
vcom filename.vhd (for VHDL)
vlog filename.v (for Verilog)
```

To compile a VHDL file into a predefined library

```
vcom [-work logical_lib] VHDL_source
```

Chapter 3: *Simulating HDL in ModelSim*

```
vcom -work work $DESIGNS/src/count4.vhd
```

and for Verilog you would use

```
vlog [-work logical_lib] Verilog_source  
vlog -work work $DESIGNS/src/count4.v
```

If you do not specify the “*-work logical_lib*” information, then the compilers will compile the HDL code into the *work* library.

This is simply a compilation step. You can invoke the HDL simulator (*vsim*) on the compiled object in the work library. The simulator is the only tool that uses this compiled object, The synthesis tool reads the HDL code into memory without saving the compiled object to disk.

NOTE: If you get an error about can't find “work” library or running *vlib*, then execute the following two commands in a UNIX shell:

```
vlib /my_path/designs/work  
vmap work /my_path/designs/work
```

4. Fix any errors and explain all warnings
5. Determine the Next Step

After your HDL code compiles correctly you can either simulate the compiled object in ModelSim or synthesize the HDL using Leonardo.

3.2. Compiling the ADK cell libraries

If you want to compile a gate-level netlist (typically produced by Leonardo) that contains ADK (*ami05/ami12*) cells, you will need to compile the VITAL (VHDL) or Verilog cell models into a ModelSim library (similar to how you compile your design into the “work” library).

1. Create a ADK work library

```
vlib /user/student/designs/adk
```

2. Map the Logical name to the physical directory created in the previous step

```
vmap adk /user/student/designs/adk
```

3. Compile the ADK VITAL or Verilog into the ADK “work” library

```
vcom $ADK/technology/adk.vhd -work adk  
vlog $ADK/technology/adk.v -work adk
```

3.3. Invoking ModelSim

Enter the following line on the unix command line:

```
% vsim model
```

If you specify a design name (*model*), ModelSim will attempt to load the last

compiled architecture or interface of the specified model from the work library.

If you did not specify a design name on the invocation line, then ModelSim will display a dialog box that allows you to verify the library name (in most cases it will be work) and select the design that you want to simulate.

3.4. Setting up the ModelSim windows

ModelSim has a set of windows that allows you to view your design and simulation results. In most cases, you will only be using the Main, Source, and Wave windows.

1. Display the source HDL code (View > Source)

```
VSIM > view source
```

2. Displaying signal waveforms

- **To add all top-level signals:**

```
VSIM > add wave /*
```

- **To add a specific signal:**

```
VSIM > add wave signal_name
```

```
VSIM > add wave clock
```

```
VSIM > add wave curr_state
```

- **To re-order the list of signals,** Select and hold down the LMB on the signal in the wave window (it will be highlighted with a box) and then move the mouse until the skinny green box is at the point where you want to move the signal to.

NOTE: You can also define the order that signals are added by using explicitly **wave** commands when you set up the Wave window. For example:

```
add wave clock
```

```
add wave clear
```

```
add wave enable
```

```
add wave count
```

- **To remove a signal,** select the signal in the wave window (it will be highlighted with a box) and select the Edit > Cut menu item in the Wave Window

3.5. Applying stimulus

3.5.1. Using Force Files (interactive)

A signal changes value either by the user applying stimulus to the signal (force command) or by the simulator applying stimulus due to design functionality. The following list shows examples of how to apply forces to the design. Typically, you only apply forces to signals defined in the top-level entity:

Chapter 3: *Simulating HDL in ModelSim*

- To apply stimulus to a signal:

```
force signal_name value time
```

If you do not specify a time, then ModelSim applies the force at the current simulation time. Some examples:

```
force reset 1
force reset 0 100
```

- To specify a clock signal

```
force clock time_value_pair, time_value_pair -repeat period
```

There must be an even number of time_value_pair combinations. For example:

```
force clk 0 0, 1 25 -repeat 50 # 50nS clock with 50% duty
force clk 0 10, 1 20 -repeat 50 # 50 nS clock with 10nS pulse
```

3.5.2. Force File Example 1

```
## initialize design
force clear 1 0
force enable 1 0
force clock 0 0, 1 25 -repeat 50
## apply stimulus
force clear 0 10
force enable 0 430
force enable 1 530
```

3.5.3. Force File Example 2 (loops)

```
set time 0
force clock 0 0, 1 25 -repeat 50
force reset 1 0
force enter_sensor 0 0
force exit_sensor 0 0
force reset 0 30

## count up to 15
for {set num 0} {$num <= 15} {incr num 1} {
  incr time 50
  force enter_sensor 1 $time
  incr time 50
  force enter_sensor 0 $time
  incr time 100
  force exit_sensor 1 $time
  incr time 50
  force exit_sensor 0 $time
  incr time 50
}
```

```
incr time 200
force exit_sensor 1 $time
incr time 50
force exit_sensor 0 $time
incr time 100
force exit_sensor 1 $time
incr time 50
force exit_sensor 0 $time
```

3.5.4. VHDL Test Bench example

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY bcd_tb IS
END bcd_tb;

ARCHITECTURE test OF bcd_tb IS

COMPONENT bcd -pre-compiled Device to Test
  PORT (rst, clk, enable, load, up_down : IN std_logic;
        data : IN std_logic_vector(3 downto 0);
        bcd_out : OUT std_logic_vector(6 downto 0) );
END COMPONENT;

SIGNAL rst, clk, enable, load, up_down : std_logic;
SIGNAL data : std_logic_vector(3 downto 0);
SIGNAL bcd_out : std_logic_vector(6 downto 0);

BEGIN

u1 : bcd PORT MAP (clk => clk,
                  rst => rst,
                  load => load,
                  enable => enable,
                  up_down => up_down,
                  data => data,
                  bcd_out => bcd_out);

-- Initialize all input signals
clk <= '0';
rst <= '1';
load <= '0';
enable <= '1';
up_down <= '1';
data <= "0101";
```

Chapter 3: *Simulating HDL in ModelSim*

```
-- Specify 10 ns clock
clock: process (clk)
begin
  clk <= NOT (clk) after 5 ns;
end process;

-- Specify Test Vectors
stim: process ()
begin
  rst <= '0' after 15 ns;
  enable <= '0' after 63 ns;
  enable <= '1' after 87 ns;
  load <= '1' after 163 ns;
  load <= '0' after 187 ns;
  up_down <= '0' after 203 ns;
end process;

END test;
```

3.6. Running the Simulator

To run the simulator for a specified amount of time, use the following command:

```
VSIM > run time
```

For example, to run the simulator for 1000 ns, enter:

```
VSIM > run 1000
```

If you just type “run” then the simulator runs for 100 nS.

Synthesizing VHDL or Verilog

The following section describes how to synthesize the HDL code that you created, compiled and simulated in the earlier chapters. As usual, this chapter provides an overview of the flow and does not go into all the options available.

4.1. Synthesizing HDL Using Leonardo (GUI Mode)

The following list describes the basics of optimizing a design for the ADK using the Leonardo synthesis tool:

Step 1: Invoke Leonardo

```
leonardo
```

Step 2: Set some ADK-specific variables

You should set the following variables in Leonardo to help routing in IC station, avoiding timing DRC problems, and creating a tighter integration with IC station, ModelSim:

```
set vhdl_write_component_package FALSE
set vhdl_write_use_packages {library ieee,adk; use
ieee.std_logic_1164.all; use adk.all;}
set edifout_power_ground_style_is_net TRUE
set max_fanout_load 14
set force_user_load_values
```

You can set these variables globally (for all users of the Leonardo software tree) by adding the previous variable settings to the \$EXEMPLAR/data/exemplar.ini file.

Step 3: Load the technology library (*Technology FlowTab*)

1. Select the desired ADK library from the ASIC -> ADK library list in the Technology FlowTab.
2. Ignore the process variables (in the Technology FlowTab)

Although the Technology FlowTab displays the PVT derating factors, they have no affect on the timing values in the library.

FYI: The slow process corner uses a temperature of 80 degrees C, a voltage of 4.5 volts, and a process number of 3 sigmas. The fast process corner uses a temperature of 0 degrees C, a voltage of 5.5 volts, and a process number of -3 sigmas. The typical process corner uses a temperature of 27 degrees C, a voltage

of 5.0 volts and a process number of 0 sigmas.

3. Load the library by clicking on the **Load Library** button

(Optionally, you could simply type the command

```
load library ami05_typ
```

You could substitute another process corner or the ami12 process, as well. The possible process corners are *slow*, *typ* and *fast*.)

Step 4: Read the VHDL file(s) (*Input FlowTab*)

1. Select the **Input** Flowtab
2. Set the working directory
3. Add the HDL files to the open files list.
4. Click on the **Read** button to synthesize your design to generic gates.

You should observe the resultant transcript for the following warnings/messages:

- Warnings about unconstrained signals (By default, all integers are 32-bits wide)
- Verify number of operators (e.g. adders) is as expected
- Verify number of sequential elements is as expected

Step 5: Optimize the design to the target technology (*Optimize FlowTab*)

The Optimize FlowTab allows you to specify the following options:

- **Target Technology:** Choose the desired technology from the picklist.
- **Run type:** Optimize a design from the specified input files or remap a previously synthesized design to a different technology.
- **Extended Optimization Effort:** If you check the extended optimization effort box, Leonardo will perform four optimization passes using different algorithms and keep the best result.
- **Optimize For:** This field informs Leonardo to keep either the smallest result (area) or the fastest result (delay).
- **Hierarchy:** Choose *preserve* to keep the design hierarchy in the synthesized design (aids in debugging), or *flatten* to merge the design into a single level of hierarchy (allows optimization across hierarchical boundaries).
- **Add I/O Ports:** MAKE SURE THIS OPTION IS NOT SELECTED. The current release of the ADK library for Leonardo does NOT include support for automatic IO insertion. You will need to manually add IO using Design Architect.
- **Run timing optimization:** This option forces Leonardo to make an additional optimization pass to help critical paths meet timing specifications.

Leonardo optimizes the original result of synthesis (reading in the design) so

running multiple optimization passes will not produce better results. The first optimization iteration is as good as it gets.

Step 6: Determine the Size and Speed of the optimized design

1. Determine the gate count (*Report FlowTab: Report Area PowerTab*)

You should examine the area report to examine the size of your design AND that leonardo used the correct number of memory resources (flip-flops).

2. Determine the critical path (*Report FlowTab: Report Delay PowerTab*)

Step 7: Save the Design (*Output FlowTab*)

Specify a filename (usually the name of the top level entity). You should also save this netlist file in a separate directory. (e.g. \$DESIGNS/netlist)

You will usually save two netlists:

- EDIF netlist for IC Layout
- VHDL or Verilog Netlist for simulation in ModelSim.

4.2. Synthesizing HDL Using Spectrum (Command-Line Mode)

If you prefer to write scripts or just want a way to automate your synthesis runs, you don't have to use the Leonardo GUI. All you need is a script and then the command

```
spectrum -file script.name
```

will run the synthesis using your script. Here is an example of a script that will read in several VHDL files and synthesize to the ADK:

```
## ADKsynthesis script

set vhdl_write_component_package FALSE
set vhdl_write_use_packages {library ieee,adk; use
ieee.std_logic_1164.all; use adk.all;}
set edifout_power_ground_style_is_net TRUE

load_library ami05_typ

analyze abmux.vhd -format vhdl -work work
analyze alu.vhd -format vhdl -work work
analyze control.vhd -format vhdl -work work
analyze datamux.vhd -format vhdl -work work
analyze top.vhd -format vhdl -work work

elaborate top -architecture structure -work work
ungroup -all -hierarchy
```

```
optimize -ta ami05_typ -effort standard -macro -area  
  
report_area -cell  
  
write ./edif/top.edf -format edif  
write ./vhdout/top.vhd -format vhd
```

4.3. Creating a EDDM Schematic from EDIF

Once you have synthesized and created your EDIF netlist, you need to generate an EDDM database and schematic to drive the IC layout tools. You do this with the *edif2eddm* script. Enter the edif directory where your EDIF output was written. Then run

```
edif2eddm design.edf model
```

where *design* is the name of your design (*top*, above), and *model* is the top-level view or architecture in your design. The output will reside in a directory called *work* within your edif directory. This is the design that has a schematic and that you can create the necessary viewpoints for before going to IC Layout.

Designing for Testability

The following sections describe how to use DFTAdvisor and FastScan with the Mentor Graphics' ADK to insert scan and generate test vectors.

5.1. Inserting Scan chains

This process has been largely automated by the DFTAdvisor tool. The following steps will get you started using a default configuration and you can then experiment with the options as you gain more familiarity with the tools.

Step 1: Create *dft.map* file in your working directory:

Using your favorite editor, create a file called *dft.map* and enter these lines:

```
standard      $MODEL_HOME/vhdl_src/std/standard.vhd
std_logic_1164 $MODEL_HOME/vhdl_src/ieee/stdlogic.vhd
adk           $ADK/technology/adk.vhd
```

Step 2: Invoke *dfta*:

Run DFTAdvisor on your structural HDL netlist for your design. The first example is for VHDL, the second for Verilog.

```
dftadvisor -vhdl design.vhd -lib $ADK/technology/adk.atpg
dftadvisor -verilog design.v -lib $ADK/technology/adk.atpg
```

Step 3: Inset Scan chain:

You can use the GUI or simply enter these commands in the command window. This is a simple configuration, but will demonstrate the basic steps necessary to insert a scan chain.

```
set system mode setup
analyze control signal
set system mode dft
run
insert test logic
```

Step 4: Write the scan netlist:

You need to write out the new netlist with your scan chain instantiated. Again, simply type these commands in the command window.

```
write atpg setup top.scan -replace
```

```
write netlist top.dfta_scan.vhd -replace
```

Step 5: Exit DFTAdvisor

```
exit -discard
```

5.2. Generating Test Vectors

Once you have inserted your scan chains, you need to generate the test vectors that will be used to test your circuit. You use FastScan to do this. Here is a general procedure for taking your scan design written from DFTAdvisor and generating your test vectors:

Step 1: Invoke FastScan in GUI mode:

You can simply type

```
fastscan
```

to bring up a dialog box that will prompt you for all the proper arguments.

Step 2: Fill out the dialog box:

For *Design*, enter the name of the scan design you output from DFTAdvisor.

Be sure the *Format* is correct for HDL source.

Enter the correct top-level name for *Top Module*.

Enter \$ADK/technology/adk.atpg for *ATPG Library*. NOTE: You must substitute your full path for \$ADK in the GUI. It will not expand your environment variable.

Enter a name for the output log file, if desired.

Click on the *Invoke FastScan* button to run FastScan.

Optionally, you can do all this from the command line:

```
fastscan -vhdl design.vhd -lib $ADK/technology/adk.atpg -top  
toy -dofile ./design.scan.dofile -log fastscan.log
```

Step 3: Generate test patterns:

At this point, you should be ready to generate patterns. Your dofile that was generated from DFTAdvisor that you ran in FastScan set up your circuit's scan and clocks. All you need to do is enter the following commands:

```
set system mode atpg  
add fault -all  
run  
save patterns top.pat  
exit -discard
```

Optionally, you can use the GUI or create a dofile with these commands in it and run that with

```
dofile dofile.name
```

At this point, you have an HDL netlist with scan chains and a pattern file with the test patterns used to test your circuit.

This is only a very basic introduction to using the DFT tools. See the Mentor documentation for more features and on how to debug your circuit should you have any problems with these tools on your netlist.

Creating a Schematic

6.1. Invoking Design Architect

The design creation process consists of symbol, HDL and schematic creation, hierarchical interpretation, design rule verification, and netlist generation. *Design Architect* is the primary tool that you use to create schematic, and symbol models. Design Architect is a design creation environment that provides you with the following functionality:

- **Schematic capture.** You can draw a schematic using components from both an ASIC library and your own symbol library.
- **Symbol creation.** You can create a symbol to represent any collection of connected components to create a hierarchical block.

For ADK design kit parts, invoke the ADK-modified Design Architect: \$ADK/bin/adk_da. Design Architect has been customized to add cell menus for the ADK supported technologies.

To exit from Design Architect, double-click the Select mouse button on the Window Menu button, which is located at the top left corner of the Design Architect Session window.

You use the Schematic Editor in Design Architect to capture schematic information that describes your design. A *schematic* is both a graphical and behavioral description of a circuit. You can include detailed information about instances, nets, connectors, test points, timing, and engineering notes. To create a schematic model, perform the steps that are detailed in the following sections.

6.2. Creating a Schematic

6.2.1. Opening a Schematic Sheet

To open a Schematic sheet from the Design Architect Session window, perform the following steps:

1. Click the **Open Sheet** icon in the Session palette menu.
An Open Sheet dialog box appears.
2. Enter the name of a new or existing component in the Component Name entry box. You can click on the Navigator button to locate a component.
3. Click on the **OK** button to execute the dialog box.

NOTE: DA opens \$schematic by default. If you generated the schematic using AutoLogic, change schematic name to “opt” in Open Sheet (options) and use Open Design Sheet. This overrides default value (\$schematic) with Model property from Back Annotation file

6.2.2. Choosing and Placing Component Symbols on a Sheet

Perform any of the following procedures to choose and place component symbols on a schematic sheet.

6.2.2.1. Using the ADK_Library Palette to Place a Component Symbol

You perform this procedure if you want to instantiate and place a component that resides in the ADK library.

1. Activate the ADK_Library palette menu by performing the following steps:
Choose the **Libraries** pulldown menu from the menu bar and press and hold the Select mouse button.
Slide the mouse pointer down to ‘ADK Libraries’ in the Libraries pulldown menu.
The ADK_Libraries palette appears, which displays the components in the ADK library.
2. Move the mouse pointer over the component that you want to instantiate and click the Select mouse button.
3. Move the mouse pointer in the Schematic sheet window and click to place the ghost image of the component to the position that you desire.

6.2.2.2. Using the Dialog Navigator to Place a Component Symbol

You perform this procedure if the symbol you want to instantiate is located in a directory of user-created component symbols.

1. Choose the following menu path from the popup menu (**Instance > Choose Symbol**)

The Add Instance dialog box appears.

2. Click on a component symbol name in the list box. If the contents of the list box do not display the name you want, use the Navigator buttons to display the directory where the component symbol resides and then click on it.
3. If the name of the component symbol you want is not the default name, perform the following steps:
 - a. Click on the Explore Contents Navigator button.

The list box displays the contents of the component.

- b. Click on the name of the component symbol you want.
4. Click the **OK** button to execute the dialog box.

The ADD IN prompt bar appears and the mouse pointer becomes a location cursor.
5. Move the cursor into the Schematic Editor window.

A ghost image of the symbol moves with your mouse pointer.
6. Move the ghost image to the desired location and click the Select mouse button to instantiate the symbol.

6.2.2.3. Using the Active Symbol Window to Place a Component Symbol

The active symbol is the symbol that the Active Symbol window displays. To place a symbol that is in this window, perform the following steps:

1. Choose **Active Symbol** menu item from the Instance popup menu:

The PLA AC S prompt bar appears.
2. Move the mouse pointer inside the Schematic sheet window and click the Select mouse button to place the symbol in the desired location.

6.2.3. Drawing a Net/Bus

To draw a net, perform the following steps:

1. Display the Schematic palette menu by moving the mouse pointer into the palette menu area and choosing the following popup menu item:

Display Schematic Palette

A schematic palette appears.

2. View the **schematic_add_rou** palette by clicking on the **ADD/ROUTE** common command button in the schematic palette menu.

The **schematic_add_rou** palette appears.

3. Click on the **ADD WIRE** icon in the **schematic_add_rou** palette menu.

The ADD WI prompt bar appears.

4. Position the mouse pointer where you want the net to begin (usually, this is at an instance pin) and click the Select mouse button.

The initial point of the net becomes fixed and a ghost net image rubber bands as you move your mouse.

5. Move the end of the net segment to the location that you desire and click the Select mouse button.

Design Architect instantiates the net segment between the initial and final points that you specified.

6. To continue adding segments to the net, move the mouse pointer to the next position and click the Select mouse button.

7. To complete the net, double click the Select mouse button.

The ADD WI prompt bar remains active so that you can begin drawing another net.

8. Click on the **Cancel** button to remove the ADD WI prompt bar.

6.2.3.1. Naming Nets

To name a net in the schematic:

1. Select Net
2. Display the Name Net prompt bar ((popup) > Name Net)

6.2.3.2. Using Buses

To connect buses to single nets and vice versa:

1. Click on the **ADD BUS** icon in the **schematic_add_rou** palette menu.

2. Place your bus by clicking on the initial point and on each bend in the bus you desire.
3. Double-click at the end of the bus.
4. Give the bus a name. It should be of the form *bus(15:0)* where the numbers represent the bits in the bus.
5. Now add wires from your components to the bus. To connect a wire to the bus click on the bus or double-click if the wire should terminate on the bus.
6. A bus ripper will appear and you will be prompted to enter the bit number for this net. You can change these later if you change your mind.

If the Check Sheet function reports unconnected pins warnings, you can add a Class property with a value of dangle to inform DA that this pin is supposed to dangle.

7. Select pin
8. Add “Class” property with value of “dangle”

6.2.4. Annotating Properties

Property annotation is the process of adding design information in the form of properties to both schematics and symbols. To annotate properties on an instance, follow these steps:

1. Setup Selection Filter
2. Select the Object to which you want to add a property
3. Select the Properties > Add > Single Item menu in the Schematic Popup menu to add a new property to the symbol
4. Select the Properties > Modify menu item in the Schematic Popup menu to change an existing property on a symbol
(If you have multiple objects selected, Design Architect should rotate through the selected objects, allowing you to modify the property on each occurrence.)

6.2.5. Adding Input and Output Ports

Go to the ADK_Library palette and use the SDL menu to select the ‘portin’ or ‘portout’ symbol.

Make sure that you give each port a unique name.

6.2.6. Adding Power and Ground Symbols

If you desire to explicitly place power and ground symbols, you must use the symbols in the SDL menu from the ADK_Library palette. These have the proper properties to drive the layout and simulation tools with the other models.

6.2.7. Creating Comment Objects on a Schematic Sheet

You can add comment text and graphics directly to a sheet in the Schematic Editor. To create graphical comment objects on a schematic sheet, perform the following steps:

1. View the **schematic_draw** palette by clicking on the **DRAW** common command button in the schematic palette menu.

The **schematic_draw** palette appears.

2. Click on any of the icons in the palette menu to create a graphical object.

To create a comment text object, perform the following steps:

3. Choose the following menu path from the Add popup menu (**Draw > Text**)

The ADD TE prompt bar appears.

4. Enter your comment text in the Text entry box.

5. Click on the At Location button.

A ghost image of your comment text moves when you move the mouse.

6. Move the ghost image to the location you want to place the text.

7. Click on the Select mouse button.

Design Architect places the text comment on your schematic sheet.

6.2.8. Checking a Schematic for Errors

You must check your schematic sheet before you can use it. You can check your design at various points in the development cycle. In Design Architect, you can perform checks of symbols, schematics, and entire designs. You can direct Design Architect to enforce optional design rules in addition to the required ones. Your design must conform to certain design rules before you can successfully use downstream tools to either analyze or simulate it.

If your design violates a required design rule, a downstream tool may issue a warning upon invocation; therefore, it is important that you first execute the design checks to ensure that the results of analysis and simulation are both valid and accurate.

To check your design, choose the following pulldown menu path from the menu bar:

Check > Sheet

6.2.9. Saving the Schematic

To save the schematic, choose the following menu path from the menu bar:

File > Save Sheet

Design Architect saves the schematic sheet.

6.3. Creating a Symbol from a Schematic

For student use, you should not manually create symbols. It is generally easier to have DA create the default symbol with all the required properties and then manually move the pins to their proper location.

You use the Symbol Editor in Design Architect to create and modify symbols. By creating symbols that represents a portion of circuitry and then using them in another schematic, you can create a logical hierarchy.

To generate a symbol from an existing schematic, perform the following steps from within the Schematic Editor:

1. Select the MISC > Generate Symbol menu item in the schematic editor
2. Use the default values in the resulting form.
3. After DA generates the symbol, you can move (DON'T DELETE or RENAME) the pins.

A symbol with a rectangular symbol body and pins on the edges of the symbol body should appear in a Symbol Editor window. All input pins are on the left side and all output pins are on the right.

DO NOT DELETE ANY PROPERTIES ON THE SYMBOL!!!!

You can move the location of the pins by selecting the pin (diamond shape on the symbol body), the pin name (e.g. clock), and the pintype property (e.g. IN). If you do not move all three of these items when you change the pin ordering on the symbol, bad things may happen with downstream tools (e.g. QuickSim).

If you can not select these three items, you may need to change the selection filter (**Edit > Selection Filter**) with the symbol window active.

6.3.1. Adding or Modifying Other Symbol Properties

In most cases, you will not need to add properties. The symbol generation function in Design Architect will add all the necessary properties and pins.

To add one or more properties, perform the following steps:

1. Select one or more object diamonds.
2. Choose the following popup menu path (Properties > Add > Add Multiple Properties)
An Add Multiple Properties dialog box appears.
3. Enter the property name and value pair for each property that you want add.
4. Click on the **OK** button to execute the dialog box.
An ADD PR prompt bar appears.
5. Move the mouse pointer to the location that you want to add the new property and click on the Select mouse button.

6. Repeat the previous step until you have added all of the properties that you specified in the Add Multiple Properties dialog box.

To modify one or more properties, perform the following steps:

7. Select the object diamond(s) that have attached properties you want to modify.
8. Choose the following popup menu path (**Property > Change Values**)
The Modify Properties dialog box appears.
9. Select one or more property name/value pairs in the list box.
10. Click on the **OK** button to execute the dialog box.
A Modify Property dialog box appears, which contains the values and attributes of a single property.
11. Change any of the property attributes in the dialog box.
12. Click on the **OK** button to execute the dialog box.
A new Modify Property dialog box will appear for the next selected property.
13. Repeat step 5 until you are finished examining and modifying the selected properties.

6.3.2. Creating Comment Objects on a Symbol

To create a comment text object, perform the following steps:

1. Choose the **Text** menu item from the Add popup menu.
2. Enter your comment text in the Text entry box.
3. Click on the At Location button.
A ghost image of your comment text moves when you move the mouse.
4. Move the ghost image to the location you want to place the text.
5. Click on the Left mouse button.

To convert symbol body graphics and text into comment objects, perform the following steps:

6. Select the objects that you want to convert.
7. Choose the following pulldown menu path from the menu bar (Edit > Convert to Comment)

Design Architect converts the selected object(s) into a comment object(s). When you unselect the object, the comment object will be a different color than either the symbol body graphics or text.

6.3.3. Checking a Symbol

All symbols must pass a set of required checks before you can place them on a schematic sheet. You can check your design at various points in the development cycle. You can direct Design Architect to enforce optional design rules in addition to the required ones. Your design must conform to certain design rules before you can successfully use downstream tools to either analyze or simulate it.

If your design violates a required design rule, a downstream tool may issue a warning upon invocation; therefore, it is important that you first execute the design checks to ensure that the results of analysis and simulation are both valid and accurate.

To check your symbol choose the following menu path from the menu bar:

Check > With Defaults

Design Architect displays any errors and warnings in the Check status window.

6.3.4. Saving a Symbol

To save your symbol, execute the following menu path from the menu bar:

File > Save

Design Architect saves your symbol to disk.

Automated IC Layout using IC Station

You will now be guided step-by-step through IC layout . You will use automatic place and route tools to establish the internal layout. You will also learn how to perform LVS and DRC checks. You will then learn how to automatically generate then program a padframe for your design. Finally, you will learn how to manually edit the layout to connect the I/O ports of your design to the pad cells.

NOTE: This chapter is appropriate only for standard-cell designs. If you are doing transistor-based designs (using SDL, for example), please go to Chapter 8 for instructions on using IC Station for that.

**** PLEASE NOTE **** Several key steps in this manual cite "ample" shortcut macros that are directory-structure dependent. Its assumed that the administrator responsible for installing the development environment kit has updated the macros for the directory-structure the library is being created in. Fields in the documentation below such as "process_mnemonic" and "process" are dependent on the macro updates.

7.1. Creating Design Viewpoints

Before the actual layout process begins, the design needs to be prepared for use with the layout tools. In the same way QuickSim viewpoints were created for the part, two more are required for layout creation and verification. The viewpoints are created for the ICTrace and ICBlocks tools, specifically. This process is automated by calling the script `adk_dve` from your design-project directory.

For ADK parts: run `$ADK/bin/adk_dve <design>`

The script creates the following viewpoints for IC station:

1. Viewpoint Creation for the ICblocks place-and-route tool.

A viewpoint called `sdl` is created, where primitives called "element" and "comp" are added. A string parameter named "lambda" is also added and assigned the value of lambda for the process specified.

2. Viewpoint for ICTrace.

The viewpoint called `LVS` is created and the primitive called `element` is appended. It does not matter what the value or the type the primitive is.

The `element` primitive in ICTrace serves the same function as the `model` primitive for QuickSim, namely that of identifying the leaf parts of the design.

7.2. Invoke IC Station

The ICgraph tool is the main entry point into the IC Station environment. Before running the tool, however, we need to set the \$MGC_WD environment variable to the directory where the design resides. This can be performed with the Unix alias-macro "swd". To invoke IC Station from the unix command prompt, simply type "adk_ic". This is a special version of IC Station that has been enhanced with special features for the ADK.

7.3. Autoplacement and Routing

Autoplacement and routing is the process of automatically generating a layout from a previously existing schematic or design. To increase our confidence in the correctness of a generated result, we setup ICgraph (the layout editor) to carefully check every action taken by ICblocks (the autoplacement/router). We will now describe ICblocks use to automatically place and route a layout. These steps will walk you through a relatively simple, direct path to an automatically placed and routed layout.

Step 1: Create an IC cell

Select "CREATE" from the menu on the right hand side of the screen. A form titled "CREATE CELL" will appear. Enter the following, then click OK:

```
Cell Name : $PROJECT/layout/[design_name]
Attach Library : $ADK/technology/ic/[process_cell_lib]
Process: $ADK/technology/ic/[process_file]
Rules File: $ADK/technology/ic/[process.rules]
Angle Mode : 45
Logic Source : $PROJECT/layout
Logic Source Type : EDDM
Logic Loading: Flat
```

\$PROJECT is your path to the part you created. Table 1 shows the valid options for the other fields. All other options should be left at default values. **It is very important to change the logic loading to flat if you have any logic other than standard cells in your design.** It is always safe to load flat so it is recommended to do so, always.

Once you click OK on this dialog box a cell window will appear having the name of the design.

Table 1: Variable Fields for Cell Creation

Variable Name	Possible Values	Description
process_cell_lib	ami05 ami12 tsmc035	These are the libraries of standard cells and pads. Currently, only pads are available for tsmc035. The library for ami15 and ami12 are both ami12. No characterization has yet been done for ami15
process_file	ami05 ami12 ami15 tsmc035	These are the process files for the currently available processes.
process_rules	ami05.rules ami05.accusim.rules ami12.rules ami12.accusim.rules ami15.rules ami15.accusim.rules tsmc035.rules tsmc035.accusim.rules	The standard rules files are for all flows except Accusim backannotation. If you want to extract for backannotation to Accusim, use the *.accusim.rules file.

Step 2: AutoFloorplan the IC cell

Select Place & Route from the IC Palette (or P&R from the ADK Edit Palette) then choose autofloorplan (Autofp) in the Place & Route Palette. Allow all options to default simply by clicking on OK in response to the Autofloorplan Options form. Use View -> All from the top menu bar to see the automatically generated floor plan. You will see a series of boxes enclosed by solid bars along each edge. The boxes indicate the rows into which cells will be organized. The solid bars indicate edges of the cells along which physical ports will be placed.

Step 3: AutoPlace the standard cells

Choose Autoplace Std Cells (StdCel under Autopl) in the Place & Route palette, and click on OK in the form, once again leaving all options to their default values. You should see individual cells placed in the floorplan boxes. Cell locations are determined by their interconnectivity. Cells which share connections are placed near one another. You may wish to experiment with the results obtained thus far by selecting different Autofloorplan and Autoplace options.

Step 4: AutoPlace Ports

Select Autoplace Ports (Ports under Autopl) in the Place & Route palette. You can allow the options to default for now, but you may wish to experiment with

them later also. You will see lightly shaded areas along the port bars at the edge of the layout.

At this point, it is assumed that you have arrived at a satisfactory initial placement for all cells of the layout. Prior to autorouting the interconnect, you may wish to observe a "rats nest" view of the signals connecting the various cells. This is sometimes useful to the layout technician for determining sources of routing congestion. To observe a rats nest of signal connections, select Connectivity -> Net -> Restructure -> All signal from the top menu bar. It may look a little messy, but keep in mind that it doesn't change the layout whatsoever.

Step 5: AutoRoute the IC cell (Autoroute All from the Place & Route palette)

From the palette menu select "All" from the "Autorou" subsection of the menu. A submenu will appear in the editing window. Select "Options" and "Expert" options and select "Channel Over Cell Routing". From the "OCR" options menu set the step size to .5 and the "Operation Mode Type" to "Center Weighted". OK all the forms and begin routing. Depending on the size of the design this may take several minutes. When the process has completed the mouse pointer changes back from an hourglass to an arrow and the results of the process are in the transcript. Several small overflows may still exist, which can be expected for larger designs. These overflows are addressed in the next step.

Step 6: Find Overflows and Route Them

This step is necessary even if overflows don't immediately appear in the routed layout. When zoomed out small overflows may not be viewable, but may still exist. To select all overflows in the design type "check over". In the form window that appears select "All" and OK the form. Next, from the Place and Route palette menu select "Overflow". If the response "An object of type Overflow must be selected" appears in the status block, there are no overflows to route. Otherwise, the overflows should be routed.

Step 7: Save the IC cell

Save the layout by selecting File -> Cell -> Save Cell -> Current Context. You may save the layout and exit the ICgraph session at any time. To re-load the layout later, choose open from the IC station palette. If you wish to make changes to the cell, you must also select File -> Cell -> Reserve Cell -> Current Context.

7.4. Verifying Project Layout

For layout verification, the physical layout data is interpreted in at least two important ways.

- The first determines if the geometries within the layout meet a set of physical design rules established by the IC foundry. This helps insure a more manufacturable and reliable chip.
- The second determines whether the layout precisely conforms to the schematic developed during the front end design phase. This of course helps guarantee that the actual fabricated circuit will perform as much like the front end simulation as possible.

Although the automatic tools performed the layout in Correct by Construction mode, it is always good to verify the layout for correctness in terms of both layout design rules and connectivity to insure consistency between the various tools used. This establishes a system of "checks and balances" that increases the overall confidence in the design. First check for layout design rule errors using ICrules, then verify the layout by double checking it against the transistor level representation of the logic diagram.

Step 1: Check Layout Design Rules

From the main ICstation palette select ICrules, then select Check from the ICrules palette. A prompt box will appear at the lower left of the screen. Click on OK to proceed with the check.

Optionally, if you are using the ADK Edit menu, you can use Drc->Check.

Step 2: Fix any Design Rule Errors

When the check is complete, design rule errors which exist in the layout will be reported in the message bar at the bottom of the ICstation window. The first one can be shown by selecting "First" in the palette (or DRC->First in the ADK Edit palette). Then rest can be viewed by clicking on "Next". As an example, one or more design rule errors of the type "DRC4_4_p_space: P+ select space = 2L" may exist. The design rule checker is complaining about p-select mask layers being too close in adjacent cells. A similar error may occur with n-select masks.

Step 3: Verify connectivity between schematic and layout

Verify the layout for consistency with the transistor level schematic of your design. Do this by returning to the main ICstation palette and selecting the ICtrace(M) (mask level LVS) option. Click on LVS in the ICtrace(M) palette and in the form which appears enter the "Source Name" \$PROJECT/LVS and click on the Setup LVS button. In the Setup LVS form change the following items and click OK (If you're already in the ADK Edit menu, you can go directly to this dialog box by clicking on LVS.):

Ground Names : VSS GND Recognize Gates : Yes

Step 4: View the connectivity check results

When the check is complete, the bottom message bar will read "Mask results database loaded". To view the results, you can select Report -> LVS from the palette menu. Look for that magic smiley face. If a nasty X appears instead, go back and make certain that the LVS viewpoint has been properly setup and follow the above procedures again. When doing so, look for potential sources of discrepancies between the schematic and how the layout was generated.

Step 5: Save the IC layout

Save your completed project layout by selecting File -> Cell -> Save Cell -> Current Context.

7.5. Full Layout Verification

You may check your layout for design rule violations at any time using ICrules. You can do this the same way you checked your cell layout, using Check in the ICrules palette, with one exception. When you check the entire layout you will find some DRC errors in the pad frame as well as any DRC errors that still exist in the cell layout. The pad frame errors are inconsequential and are a result of using an optional set of design rules. Focus on the design rules that exist independent of the pad frame by excluding certain cells from the DRC.

1. Select Check from the ICrules palette.
2. Click on options in the resulting prompt bar, and enter the following under exclude cell: PadOut PadSpace PadBidir PadInC PadGnd PadVcc PadFC (and any other pad names you see in your padframe). If you also did a DRC on your core logic, you can even exclude it from the checks.

This will exclude all pad cells from the DRC check, and any errors will be exclusively related to your cell and the top level interconnect.

Schematic Driven Layout using IC Station

In addition to the automated tools for layout that let you do standard-cell place and route, you can use *schematic driven layout (SDL)* to place and route your cells or device-level circuits. This section will show you the basic features of SDL and how the ADK supports SDL for device and hierarchical cell layout. For more complete documentation on all of the SDL features of IC Station, see the manual *IC Station Device Level Automation (DLA) Manual*.

**** PLEASE NOTE **** Several key steps in this manual cite “ample” shortcut macros that are directory-structure dependent. Its assumed that the administrator responsible for installing the development environment kit has updated the macros for the directory-structure the library is being created in. Fields in the documentation below such as “process_mnemonic” and “process” are dependent on the macro updates.

8.1. Creating Design Viewpoints

Before the actual layout process begins, the design needs to be prepared for use with the layout tools. In the same way simulation viewpoints were created for the part, two more are required for layout creation and verification. The viewpoints are created for the ICTrace and ICsdl tools, specifically. This process is automated by calling the script `adk_dve` from your design-project directory.

For ADK parts: run `$ADK/bin/adk_dve <design>`

The script creates the following viewpoints for IC station:

1. Viewpoint Creation for the ICsdl.

A viewpoint called `sdl` is created, where the primitive called “`elemen`” is added. A string parameter named “`lambda`” is also added and assigned the value of `lambda` for the process specified. Some other properties are made visible to the IC tool, as well

2. Viewpoint for ICTrace.

The viewpoint called `LVS` is created and the primitive called `element` is appended. It does not matter what the value or the type the primitive is.

The `element` primitive in ICTrace serves the same function as the `model` primitive for QuickSim, namely that of identifying the leaf parts of the design.

8.2. Invoke IC Station

The ICgraph tool is the main entry point into the IC Station environment. Before running the tool, however, we need to set the \$MGC_WD environment variable to the directory where the design resides. This can be performed with the Unix alias-macro “swd”. To invoke IC Station from the unix command prompt, simply type “adk_ic”. This is a special version of IC Station that has been enhanced with special features for the ADK.

8.3. Creating a cell for SDL

There are two ways to create a cell for SDL. It is important to create you cell in one of these two ways else the logic and ICStation databases might not correlate, correctly.

Method 1: Use Create Cell

Just as in creating a cell for automatic place and route, you can select “CREATE” from the menu on the right hand side of the screen. A form titled “CREATE CELL” will appear. Enter the following, then click OK:

```
Cell Name: $PROJECT/layout/[design_name]
Process: $ADK/technology/ic/[process_file]
Rules File: $ADK/technology/ic/[process.rules]
Angle Mode: 45
Logic Source: $PROJECT/sdl
Logic Source Type: EDDM
Logic Loading: Flat
```

\$PROJECT is your path to the part you created. Table 1 in the previous Chapter shows the valid options for the other fields. All other options should be left at default values. **It is very important to change the logic loading to flat.** Since you are not using a standard cell library for your design, you should leave the library line blank. If you wish to use the default technology of AMI05, you can leave the process and rules lines blank, also. They are loaded by default.

Once you click OK on this dialog box a cell window will appear having the name of the design.

Method 2: Use Create SDL

If you wish to use the default technology of AMI05 or want to change the technology/process information later, you can use Create-SDL from. Click on “Create-SDL” on the palette menu to bring up the dialog box. Enter the following data and then click OK:

```
Already have viewpoint
Path to Viewpoint: $PROJECT/layout/[design_name]/sdl
```

It is important that you specify the SDL viewpoint.

8.4. Placing components into your layout

Once you have opened your cell you can start performing your layout. You will need to place your instances (transistors, resistors, capacitors, ports). You can use auto-placement features, manual placement or a combination of both. You will need to have your logic opened to do either, so if it is not yet open you should go to the ADK Edit menu and then click on Open to open your logic.

8.4.1. AutoPlace instances

You can quickly place all of your instances in your schematic into your cell by clicking on AutoInst in the ADK edit menu. The positions of the devices will resemble the placement of the schematic instances. That is, if a transistor is above another one in your schematic, it will be in your layout, also.

Once the devices are placed, you can move them to fine-tune the placement. You will see overflows between the pins of the devices that will help you to see the connectivity in your design. Use these as guides when moving your devices around.

If you only want to place some of the components, select them in the schematic and then click on AutoInst to instantiate only the selected item(s).

8.4.2. Manually placing instances

If you prefer more control over initial placement of your devices, you can place them, manually. To do this, select the instance(s) that you wish to place in your schematic. Then click on Inst in the palette menu. A ghost image of each component, one at a time, will appear in your layout window. Click where you want to place this instance and then the next one will appear and you continue this procedure until all selected instances are placed.

Again, you will see overflows to help you locate the best placement. In addition, you will see the current instance highlighted in your schematic. Once placed, the devices can be moved around your cell as any object.

8.5. Placing ports into your layout

Before placing the ports, you will need to select a port style. You do this via the Setup->SDL pulldown menu. One of the options you can select is "SDL Port Style." You should then select the "Process Port" button and select the "default" port from the list. This is a minimum-sized port on metal2. By default, all ports are on metal2. You can change this after placement if you desire.

Unlike placing instances, you must place your ports manually. To do this, simply select the port(s) you wish to place and click on Port. This will prompt you, one at a time, to place the ports like you did the instances. If you wish to place all the ports at this time, you can select no ports and then click Port. This is a semi-automatic mode that will prompt you to place all ports in the design.

8.6. Routing your cell

Once you have placed your components (or at least as many as you want to) you can route the overflows. There are many ways to route your design. All are manual or semi-automatic. There is no fully-automatic autorouting for SDL designs.

8.6.1. Semi-automatic routing

You can use the AddR command from the ADK Edit menu to add a route. A route is a semi-automatic way to place metal connections within your circuit. DRC rules are followed and you are shown boundary-lines where you are allowed to place your route. You are restricted to only using valid routing layers (such as metal1 and metal2) so this does not work for poly, for example.

To use the AddR command, simply click on AddR in the palette menu and then select one pin that has an overflow going to/from it. You then click at each point where the wire should bend. If you want to change layers, and put down a via where appropriate, press the space bar to cycle through the valid layers. When done, double-click and the route will end. You can continue this procedure for all the metal connections in your cell. The routes placed are really paths so you can edit them as you would any path.

8.6.2. Routing with paths

For some layers, like poly, you cannot use the semi-automatic routing tools. Here you must use a manual tool. One such way to manually route is to use paths. Paths are of a set width and you simply click at each point at which the path should touch. You can set the path to be defined from a center point or the left or right edge of the path.

To add a path, click on Path in the ADK Edit menu. The path prompt bar will appear and you will be able to specify the layer and width of the path via the Options button. You can also specify other options for how the path should be drawn. Once you have these options set, simply click at each point where you want the path to go. You will see the path being drawn as you go. Double-click at the end and the path will be complete. You can then go on and repeat this process for other paths you wish to place.

8.6.3. Routing by placing shapes

Optionally, you can have complete control over the size, shape and placement of the routes by placing geometric shapes, explicitly. You do this with the Shape command on the ADK Edit menu.

When you click on Shape you are prompted for points that will define the corners of a polygon. You can use the Options button to select the layer on which the shape should be drawn.

8.6.4. Placing contacts and vias

Sooner or later you will need to place contacts from metal to poly or vias between metal layers. If you are using the routing tool then vias within your routes are placed automatically. Contacts to poly and other methods of routing will require manual contact/via placement, however.

There are some handy macros that have been defined in the ADK that will make placing contacts and vias easier. The *pc* (or poly contact) macro will place a poly contact to metal1 centered at the current cursor position. The *pp* (or place port) macro will let you place a port contact (a via from metal1 to metal2 and some blockage layers) where ever you click once the ghost image appears.

Keep in mind that unless you change the port layer, all ports will be on metal2. You will need to place a port contact (or via) to get to metal1 to connect your port to metal1. Don't forget to do this!

8.6.5. Adding well contacts

You will almost certainly want to add well (or bulk) contacts to your cells as well. These will tie your wells or bulk to metal so you can properly bias it as necessary. Again, there are two macros that make this a little easier.

- *nwc* will place an n-well contact into your cell where you click
- *pwc* will place a p-well contact into your cell where you click

Both macros place a cell that has been predefined with proper rules to pass DRC. The bottom of the n-well contact is designed to abut to your n-well and the top of the p-well contact is designed to abut to your p-well. Keep this in mind if you decide to rotate or flip your contacts. You can always peek down into the contacts to see the full layout to better align your cells.

8.7. Verifying your layout

After placing components and ports and then wiring your circuit up, you need to verify it's correctness. There are three areas to check: Design Rule Checks (DRC), Layout Versus Schematic (LVS) and functional correctness/performance. The last item will be discussed in the next section on extraction.

8.7.1. DRC of your cell

You need to run a DRC of your cell to verify that you have not violated any design rules with the placement of your devices or routes. This is most easily done from the ADK Edit menu via the DRC->Check item. When you select this and click OK on the prompt bar, a complete DRC of your entire cell is performed and the results presented in the prompt bar at the bottom of your screen.

The first error (if any) can be shown by selecting DRC->First in the ADK Edit palette. Then rest can be viewed by clicking on "Next". As you are checking you can fix errors and continue showing the next one as long as you'd like. Keep in

mind that errors in nearby structures might be fixed with one change, however, so it is wise to rerun the DRC after every few fixes.

8.7.2. LVS of your cell

In addition to verifying the design rules with DRC, you also will want to verify that your layout matches your schematic. You want to verify that all components were placed and that you have not shorted anything together or left things unwired. This is easy to do with LVS.

- a. First, close your logic using Logic->Close from the ADK Edit menu.
- b. Then, from the ADK Edit menu, click on LVS which will bring up the LVS dialog box. You need to enter the viewpoint to use for LVS. You should use the LVS viewpoint for LVS.
- c. Now click on Setup Lvs and change the ground name to “ground” and OK the dialog box.
- d. OK the LVS dialog box to run LVS.
- e. To view the results, use Results->LVS next to the LVS item on the ADK Edit menu. If you see a checkmark, a smiley face and “Correct” then you are good! If not, read the report to see what types of problems you might have. If you have many errors you can go to the ICtrace (M) menu off the main IC Station menu for many more LVS options.

At this point you should have verified that your cell has no DRC errors and matches your schematic. The final step in verification is extraction and simulation of your design.

Generating Padframes

The ADK has a facility for automatically generating a TinyChip padframe for fabrication through MOSIS. If you wish to automatically generate and route to your padframe, you need to prepare your design data in a particular manner to facilitate this feature.

9.1. Top Level Layout Preparation

To begin, you need a top level part containing the design inside a padframe. Start by creating a top level schematic for the whole chip that includes both the pads and the design core. The core logic needs to have a symbol for each piece and a property called *phy_comp* for “physical component” and **no comp** property. Simply place your core logic and the pads which will drive the automatic padframe generator in IC Station.

Step 1: Create a symbol for your core logic:

In Design Architect you need to generate a symbol (Miscellaneous -> Generate Symbol) for each piece of core logic that you want to instantiate in the pad frame. You will have to have a completed IC layout for each component for which you have a symbol at this top level. These can be SDL or standard cell designs or a combination of both.

Once you have generated your symbol, you need to add the *phy_comp* property-string to the symbol body. The value of *phy_comp* needs to be the name of the IC cell that you created (or will create). Do not place a *comp* property on this symbol.

Step 2: Use Choose Symbol to bring in the design part(s)

and wire them together if necessary. Perform a check -> sheet and just ignore the unconnected pin warnings. You can also eliminate the reporting of these warnings by selecting check -> sheet -> set defaults and setting dangles to errors only.

Step 3: In the ADK Library menu choose the pads to use

from the appropriate library for your technology. Based on the technology, there may be different types of pads available. You can only use one set of technology pads and it must agree with the technology of your design.

Add the pads to your schematic and wire them up to your core logic. Change the property PINXX (shift+F7) to replace the XX with the pin number on the package

you wish to connect this signal to in the final design. Be sure to add power/VDD/GND pads as necessary. Any pads not in your schematic will not appear in your completed padframe. Check and save your schematic.

Note: For all processes, the corner pads will be automatically instantiated for you. For ami12, this includes the power corner pads which power the padframe. You still must instantiate core-logic power and ground, however.

Step 4: Use `adk_dve` to create proper viewpoints for the top-level design.

Run `adk_dve [design]` like you did for other schematics. This will create the necessary viewpoints for layout and verification.

9.2. Padframe Layout

In IC station (`adk_ic`) you can create a new cell for the top-level logic and the padframe that will be generated based on your schematic.

Note: The automatic padframe generator will only generate 40-pin tiny-chip padframes at this time.

1. Create a new layout cell for the whole chip.

Enter the following in the new cell form, allowing all else to default:

```
Cell Name: $PROJECT/layout/[design_name]
Attach Library: $ADK/technology/ic/[process_cell_lib]
Process: $ADK/technology/ic/[process_file]
Rules File: $ADK/technology/ic/[process.rules]
Angle Mode: 45
Logic Source: $PROJECT/layout
Logic Source Type: EDDM
Logic Loading: Flat
```

- 2. Go to the ADK Edit palette menu and click on *Open* to open the logic source window. You should see your schematic in a new window.**
- 3. Select the core logic (if you have more than one piece, select them all). Then click on *Inst* on the DLA Logic palette to place these cells into your layout window. Remember, you must have previously designed these cells and they must have the name of the `phy_comp` property on your symbol.**

If you wish to store your completed cells in a location other than your current working directory, you will need to set your search path from the IC Station Setup->SDL menu. The search path should look something like this:

```
$ADK/technology/ic/ami05_via $PROJECT/cells.
```

The last “.” is important for it includes the current, or working directory. Simply

add any other paths you wish to search for your cells.

4. Once you have placed the core logic block(s), you are ready to generate your padframe. Do not place the pads, yourself. Use the ADK->Generate Padframe->AMI 0.5 (or the appropriate technology for your design) pulldown menu to generate the padframe.

Make life easy on yourself. NEVER EDIT OR RELOCATE PAD CELLS.

5. View the entire cell, now, to see the padframe and your logic (Shift-F8).
You should see the entire padframe with corner pads, the pads you asked for and spacer pads for all the empty spaces (or analog pads for the case of the ami12 technology). You will also see overflows showing which ports get wired to which pins
6. At this point you might notice that your core logic should be rotated, flipped or moved to make routing easier. Do that, now, but be careful not to move the pads.
7. After finalizing the placement of your core logic, you can route the pads to the core. You can do this manually or automatically.
8. To autoroute, click on P&R on the ADK Edit palette menu to bring up the Place & Route palette menu. Then click on All under *Autorou*. On the prompt-bar that appears click *Options* and unselect *Expand Channels* from the menu. Click *OK* on the menu and prompt-bar to autoroute the pads.
9. Now take a look at the final layout and fix any problems. For example, you might not like the width or placement of the power busses. You can edit these routes (or opt to manually route them, initially) until you are satisfied with the final result. You should also check, that the pads did not move during an autoroute. If they did, you need to unplace all the pads, then rerun the generator and autoroute, again, verifying that *Expand Channels* is unselected. This is the most common reason for this problem.

Performing Post-Layout Verification using Mach-TA

Mach TA is an analog simulation tool for performing timing analysis and functional verification of digital designs. Its role in the ADK-kit design flow is to perform analysis on netlists extracted from a design layout in IC station.

10.1. Preparing to Simulate the Design

Step 1: Extracting the SPICE Netlist from IC Station

Mach TA uses SPICE netlists as its input. To get a SPICE netlist for your design, do an IC Extract(M)->Lumped from the IC Palettes menu in IC Station (or click on Extract in the ADK Edit palette menu). In the form that appears select the Netlist button, enter the filename for the netlist, and enter the ground-node name as 'GND'. OK the form to generate the netlist.

An incongruence exists between the format of the netlist that IC Station provides and the format expected by Mach TA. Specifically the netlist generated by IC Station is defined as a subcircuit while Mach TA expects a flat file with no defined subcircuits. To correct this an awk script \$ADK/bin/mta_prep.awk has been provided in the kit to reformat the netlist. To use the script invoke:

```
cat [design].sp | nawk -f $ADK/bin/mta_prep.awk | sed
's/\+/\*\/g' >![design].mta.sp
```

The second pipe though sed may not be necessary as that it only removes '+' line wrapping in netlist files with an excessive number of ports defined in the subcircuit.

10.1.1. Creating Test Vectors

To apply stimulus to a design-netlist Mach TA requires a test vector (.tv) file in Lsim format. This can be acquired from early Modelsim test results be either using the VTRAN conversion utility (www.sourceiii.com) or writing a customized script to convert the simulation results.

10.1.2. Creating a Command File

A command dofile for Mach TA makes it very convenient to automate test-vector analysis. Such a dofile is automatically generated when using the mta_layout_prep script for test vector creation. An example script would look like:

```
.dc
plot v(clear) v(clock) v(enable) v(output(0)) v(output(1))
v(output(2)) v(output(3))
```

```
run -tvend -dc count4.tv
```

This script performs initial DC analysis, selects which port pins signals to plot, and starts test vector analysis with test-vectors in a file called count4.tv

10.2. Running Test Vectors and Viewing Results

Step 1: Invoking Mach TA

To invoke Mach TA on a netlist using the nominal ami05 technology corner case

```
mta -t $ADK/technology/mta/ami05 -tc NOM [design].mta.sp
```

This brings up the application and loads the design. To execute a pre-created dofile type 'dof [design_dofile_name].do' at the command-line to begin analysis. Commands can also be typed in manually on the comandline in the lower command window. See the previous section on dofiles for example commands.

Step 2: Viewing Results

Once test-vector analysis has passed (and hopefully completed) the simulation waveforms selected by the 'plot(node_name)' command can be viewed from Tools->Wave Viewer drop-down menu. If the display doesn't initially look correct select the equal button on the top row of the window to view the entire simulation period.

Given the case where the test vectors were derived from the Modelsim simulation results, the simulation output from both tools should be comparable.

Simulating the Design Using QuickSim II

By using QuickSim II, you can apply stimulus to the design, run the simulation, analyze the results, and modify the design based on those results. You can then reset the simulator, optionally revise or apply more stimulus to the design, and start the cycle over. When the design functions correctly, you can save the stimulus and simulation results directly with the design. You can perform various design simulation tasks, including the following items:

- Examine the logic states of signals by tracing, listing, monitoring, and break-pointing them
- Modify your design during simulation and then re-simulate
- Simulate your design using any of the following timing modes: unit delay, linear timing, linear timing with constraint checking, full timing, and full timing with constraint checking.
- Calculate your design's timing as a function of pin loading and environmental effects, such as temperature, voltage, and process

The typical strategy for simulation is an iterative process that has three main phases: verify functionality without regard for timing, verify functionality accounting for timing effects, and verify functionality once the design is placed and routed. Although the focus of each phase is different, the tasks that you perform within the simulator are very similar.

11.1. Setting up the Design Viewpoint

For ADK parts: `run $ADK/bin/adk_dve <-t technology> design`

This command will setup viewpoints for quicksim in addition to viewpoints for IC layout and analog simulation. Specify the technology you wish to use (ami05, ami12, or ami15). If you do not specify a technology, ami05 is assumed.

11.2. Invoking QuickSim II

You can invoke QuickSim II by issuing the **quicksim** command from an operating system shell. SimView is automatically invoked when you invoke QuickSim II. Do not attempt to invoke SimView independently. The following sections detail these two invocation methods.

To invoke QuickSim II from a shell window, enter the following command:

quicksim design/ami05/ami12

You must use an existing Design Viewpoint and add any additional switches to

specify a variety of modes and setup conditions. When you invoke the QuickSim II simulator, it displays a default window called the Session window. The Session window, which is managed by SimView, controls the interactions between the data in the Session's sub-windows.

You can open up the following windows inside a Session window: Schematic View, List, Trace, Monitor, Waveform Database, and Waveform.

After you invoke QuickSim, you setup the SimView environment, as described in the following sections:

11.3. Setting up the SimView Windows

QuickSim II uses the SimView common interface for all user-interactions with the simulation kernel. SimView is used by all "Classic Mentor" simulation products (QuickHDL does not use SimView).

11.3.1. Opening the Schematic View Window

The Schematic View window displays the schematic of your design. Perform the following steps to view your design in a Schematic View window:

1. Click on the **Setup** palette button.

The Setup palette icons appear in the palette menu.

2. Click on the **Open Sheet** icon in the palette menu.

The schematic of the design that you invoked QuickSim II on appears in a Schematic View window.

The Schematic View window contains a schematic sheet. This window enables you to view the design hierarchically. You can graphically select and unselect design objects in the Schematic View window. If you open a schematic sheet on a VHDL design, an interactive list window appears. Back annotations appear in the Schematic View window and are highlighted; you can also hide them from view.

11.3.2. Using a Trace Window

The Trace window contains a waveform display of signal, bus, and expression values. You can view stimulus in the Trace window before and after running a simulation. This window uses line color, style, and position to show signal states and strengths. A Trace window displays signal names along its left edge. A row of tic marks that are located to the right of each signal name helps you to identify the signal's logic level. The following list describes the logic levels and their corresponding positions:

- 0.** Below the tic marks
- 1.** Above the tic marks
- X.** Through the tic marks

The following list describes the drive strengths and their default appearances:

- S.** Solid or light blue line
- R.** Dashed or medium blue line
- Z.** Dotted or green line
- I.** Bold solid or yellow line

The following sections contain procedures that you perform to open a Trace window and both add and delete signals from it.

11.3.2.1. Opening a Trace Window

The Trace window displays waveforms of signal activity. To open a Trace window, perform the following steps:

1. Open the Schematic View window, if it is not already open, by performing the “Opening the Schematic View Window” procedure in this manual.
2. Select either a net or bus in the Schematic View window or a VHDL description in a VHDL Description window by moving the mouse pointer over it and clicking the Select mouse button.

SimView highlights the object that you selected.

3. Click on the **Trace** common command button in the palette menu.
The Trace window appears along the bottom of the Session window.

11.3.2.2. Adding a Signal to the Trace Window

To add a signal to the Trace window, perform the following steps:

1. Select a net or bus in the Schematic View window or a VHDL description in a VHDL Description window by moving the mouse pointer over it and clicking the Select mouse button.

SimView highlights the object that you selected.

2. Choose the following menu path from the menu bar (Add > Traces)
The Add Traces dialog box appears.
3. Click on the **OK** button to execute the dialog box.
The signal that is on the selected pin, net, or bus appears in the Trace window.

11.3.2.3. Deleting a Signal from the Trace Window

To delete a signal from the Trace window, perform the following steps:

1. Select one or more signals in the Trace window by moving the mouse pointer over each signal label and clicking the Select mouse button.
2. Press the Menu (right) mouse button and choose the following menu path from the popup menu (**Delete > Selected**)

11.3.3. Using a List Window

The List window contains a listing of signal, bus, and expression values. This window enables you to view signal activity in tabular form. The following sections contain procedures that you perform to open a List window and both add and delete signals from it.

Opening the List Window

The List window displays a tabular listing of signal activity. To open a List window, perform the following steps:

1. Click on the **Setup** palette button.
2. Click on the **List** common command button in the palette menu.

If you want signals to automatically appear in the List window, select the corresponding pin, net, or bus in the Schematic View window before you open the List window.

For more procedural information, refer to both the “List Window” and the “Creating a List Window” sections of the *Getting Started with QuickSim II Training Workbook*.

Adding Signals to the List Window

To add signals to the List window, perform the following steps:

1. Select a pin, net, or bus in the Schematic View window.
2. Press the Menu (right) mouse button and choose the following menu path from the popup menu: (**Add > Lists > Selected**)

Deleting Signals from the List Window

To delete signals from the List window, perform the following steps:

1. Press the Mouse Menu button and choose the following menu path from the popup menu: (**Delete > Lists**)
2. Enter the name(s) of the signal(s) that you want to delete in the Signal Name entry box.
3. Click on the **OK** button to execute the dialog box.

11.3.4. Using a Monitor Window

The Monitor window displays signal and expression values at the current time. The following sections contain procedures that you perform to open a Monitor window and both add and delete signals from it.

Opening the Monitor Window

The Monitor window displays both the current simulation time and the current

values of the signals that you have added to the window. To open the Monitor window, perform the following steps:

1. Select one or more pins, nets, or busses in the Schematic View window.
2. Press the Menu mouse button and choose the following menu path from the popup menu (**Add > Monitors > Specified**)
3. Click on the **OK** button to execute the dialog box.

For more information, refer to the “Opening and Adding Data to Windows” section of the *SimView Common Simulation User’s Manual*.

Adding Signals to the Monitor Window

To add a signal to the Monitor window, perform the following steps:

1. Select one or more pins, nets, or busses in the Schematic View window.
2. Press the Menu mouse button and choose the following menu path from the popup menu (**Add > Monitors > Selected**)

Deleting a Signal from the Monitor Window

To delete one or more signals from the Monitor window, perform the following steps:

1. Activate the Monitor window by moving the mouse pointer inside the Monitor window and clicking the Stroke/drag (middle) mouse button.
2. Select the signal(s) you want to delete.
3. Press the Menu mouse button, while keeping the mouse pointer inside the Monitor window, and choose the following path from the popup menu (**Delete > Selected**)

11.4. Generating Stimulus

QuickSim II uses stimulus that is stored in a waveform database. SimView, which manages a variety of waveform databases, conveys the stimulus to QuickSim II, applies the stimulus to your design and calculates the results. QuickSim II sends the results back to SimView to be stored in the Results Waveform Database.

You can use a number of formats to create waveform stimulus, including Forcefiles, Logfiles, and Mentor Interactive Stimulus Language (MISL) files. You can easily translate Forcefiles, Logfiles and MISL files into a waveform database by using SimView to load the stimulus file into program memory. You can also translate a waveform database into either a Forcefile or a Logfile.

The following procedures describe various methods that you can use to create and manipulate stimulus that QuickSim II can use to simulate your design.

11.4.1. Creating a Waveform Using Palette Icons

You can create waveforms by using a variety of Stimulus palette icons. Perform the following steps to create a new waveform that you want to apply to either a pin, net, or bus:

Step 1: Click on the red STIMULUS palette button.

Step 2: Select pins, nets, or busses

In the Schematic View window, move the mouse pointer over it and click the left button.

Step 3: Click on one of the following Stimulus palette icons and perform its related procedure, if any, to create a waveform:

- **Edit Waveform.** Adds the waveform to the default waveform database and to the Trace window.
- **Force To State.** Applies the state value that you select from the State value cascade menu at the current simulation time to the default force target waveform database.
- **Add Force.** Applies the time/value pairs and force type to the selected signal. Perform the following steps to add a force:
 - a. Click on the Add Force icon.
 - b. Enter the time/value pairs and force types that you want to apply in the dialog box.
 - c. Click on the **OK** button to execute the dialog box.
SimView adds the force to the selected signal(s).
- **Add Clock.** Applies a clock waveform that has the period, time/value pair, and force type to the selected signal. Perform the following steps to add a clock waveform:
 - d. Click on the Add Clock icon.
 - e. Enter the time/value pairs and force type that you want to apply in the dialog box. A typical clock signal has at least two time/value pairs for both low and high values.
 - f. Click on the **OK** button to execute the dialog box.
SimView adds the clock to the selected signal.
- **Pattern Generator.** Generates a pattern waveform that it applies to the selected signal. Perform the following steps to apply a pattern:
 - g. Click on the Pattern Generator icon.

- h. Click on the pattern type label button that describes the pattern that you want to generate.
- i. Describe the details of the pattern you want to generate in the entry boxes.
- j. Click on the **OK** button to execute the dialog box.

SimView generates the pattern and adds it to the signal that you either selected or specified.

11.4.2. Creating a Waveform Using the Waveform Editor

You can use the Waveform Editor to create a variety of both simple and complex waveforms. To create a simple waveform that transitions from an Undefined (X) to a High (1) logic value by using the Waveform Editor, perform the following steps:

1. Open the Trace window, if it is not already open, by performing the “Opening the Trace Window” procedure in this manual.
2. Click on the red **WF EDITOR** palette button.
3. Select a pin, net, or bus in the Schematic View window.
4. Click on the Edit Waveform icon in the Waveform Editor palette menu.
The signal that corresponds to the selected object appears in the Trace window. The Trace window displays any stimulus waveforms that you previously created for that signal.
5. Activate the Trace window by moving the mouse pointer inside the Trace window and clicking the middle mouse button.
The Trace window becomes the active context window and all of the icons in the Waveform Editor palette menu become usable.
6. Click on the **Add Edge** icon in the Waveform Editor palette menu.
7. Activate the Value entry box by moving the mouse pointer over the signal in the trace window and clicking the Select mouse button.
8. Enter **1** in the Value entry box.
9. Click on the Location icon in the prompt bar.
10. Move the mouse pointer which now appears as a set of cross hairs to a point on the selected waveform in the Trace window.

SimView adds an X to 1 logic level transition to the signal in the Trace window.

11.4.3. Creating a Force File

The following example shows a force file for a State Machine design with a clock (clk) and reset and an input signal called “x”. In addition to stimulus information, you can also include other AMPLE commands such as OPEN SHEet, ADD TRAcE, and ADD

LISt to the force file.

The following list shows the typical development process to create a force file:

1. Generate the stimulus using the SimView tools (e.g. palette icons and waveform editor).
2. Verify the stimulus is correct by simulating the design
3. Delete the clock signal stimulus

If you do not delete the clock signal, every clock transition is added to the force file which makes the resulting force file difficult to edit.

4. Save the stimulus into a force file (as described in the next section).
5. Edit the force file to include the clock signal definition

The following example shows a completed force file to test (simulate) a finite state machine:

```
//Define Clock signal
Set Clock Period 50
Force /clk 0 20 -Repeat
Force /clk 1 40 -Repeat
FORCe /clk 0 0.0 -Abs

//Initialize flip-flops
FORCe /rst 0 0.0 -Abs
FORCe /rst 1 1.0 -Abs

// Test State Machine
FORCe /x 1 12.0 -Abs
FORCe /x 0 75.0 -Abs
FORCe /x 1 175.0 -Abs
FORCe /x 0 225.0 -Abs
FORCe /x 1 325.0 -Abs
FORCe /x 0 525.0 -Abs
FORCe /x 1 575.0 -Abs
FORCe /x 0 725.0 -Abs
```

11.4.3.1. Saving Waveform Stimulus in a Forcefile

To save the waveforms that you have created in a Forcefile, perform the following steps:

1. Click on the red **STIMULUS** palette button.
The Stimulus palette icons appear in the palette menu.
2. Click on the **SAVE WDB** icon in the palette menu.
A Save Waveform DB dialog box appears.

3. Enter a pathname in the Pathname entry box for the Forcefile that you want SimView to create.

You should save this file in the \$DESIGNS/sim directory. Typically, you name the file *design.do*. To execute the file within Quicksim II, type:

```
DOFile $DESIGNS/sim/design.do
```

If the filename already exists, click on the **Replace** button.

4. In the File type field, click on the Forcefile choice button.
5. Specify both a start time and stop time in the Start time and Stop time entry boxes for the waveform activity that you want to save.
6. Click on the **OK** button to execute the dialog box.

SimView saves the waveform activity in the Forcefile at the pathname that you specified.

11.4.3.2. Loading and Viewing Waveform Stimulus from a Forcefile

To save the waveforms that you have created, perform the following steps:

1. Reset the simulator to time zero by performing the following steps:
 - a. Click on the **RESET** common command button in the palette menu.

The Reset dialog box appears.

- b. Click on the **State** radio button.
 - c. Click on the **OK** button to execute the dialog box.

SimView resets the simulator time to zero.
2. Press the Menu mouse button and choose the following menu path from the popup menu (**Force > From File**)
The Load Forcefile dialog box appears.

3. Enter the name of the Forcefile in the Pathname entry box.
4. Click on the **OK** button to execute the dialog box.
SimView executes the Force commands that were in the Forcefile.

To view the stimulus waveforms, perform the following steps:

5. Click on the red **WF EDITOR** palette button.
The Waveform Editor palette icons appear in the palette menu.
6. Select the pins, nets, and busses in the Schematic View window that correspond to your Forcefile stimulus you want to view.
SimView highlights the selected objects.
7. Click on the **EDIT WAVEFORM** icon in the Waveform Editor palette menu.

All of the signals that corresponds to the selected objects appear in the Trace window.

11.5. Running the Simulator

To begin a simulation run, enter the following command in the popup command line:

```
run time
```

If you do not specify a time, the simulator will run indefinitely. You can stop the simulator at any time by pressing the CONTROL and C key at the same time.

If you have found problems in your circuit or do not fully understand the behavior of the design, you will need to make changes to both the design and stimulus before you proceed with further simulation. The following sections briefly describe how you reset the simulator to time 0, set breakpoints, back trace X states, and change your design.

11.6. Changing the Timing Mode in the Kernel

Unless you set the kernel differently, you will be running in unit-delay mode. That is, all gates will have a unit delay and you will only be performing a functional test. If you wish to see actual timing for the gates, you will need to set the timing mode in the kernel.

Use the **Setup->Kernel->Change->Timing Mode...** menu to change the timing mode. You can select from unit-delay, full delays or linear approximation delays. Currently, there are no linear models so if you choose that option, the full delays will be used, instead.

You can choose to use *slow*, *fast* or *typ* timing values for your circuit. The timing mode can be changed for all or selected instances.

11.7. Resetting the Simulator

You can reset your simulation environment to start a simulation again at a time domain of zero or to reinstate the setup conditions for SimView and/or QuickSim II that were in effect when you invoked the simulator.

You can use any or all of the following choices to reset a simulation:

- **State.** Resets the simulation time to zero. No other conditions that previously existed are reinstated; stimulus are maintained, but the data in the Results WDB is removed.
- **SimView setup.** Reinstates the SimView setup conditions that were in effect when you invoked the simulator. SimView closes all windows (except the Session window), deletes action lists and expression definitions, and resets all bus definitions, synonyms (probes), groups, and selection filters to their original settings.
- **QuickSim setup.** Reinstates the kernel setup conditions that were in effect

when you invoked the simulator, such as the timing modes, delay modes, checking modes, settings, keep list, run setup and breakpoint settings list.

To reset the simulator to time =0 and re-initialize the nodes in the design, type:

```
reset state -d
```

11.8. Using QuickSim to Debug your design

11.8.1. Using Breakpoints

Breakpoints provide you with a powerful means to troubleshoot problems in your design. You can use a number of conditions as breakpoints during the simulation to isolate specific problems. You can interrupt the simulation based on either a simulation expression or signal state. You can also interrupt the simulation based on the activation of a VHDL object. The following sections describe the procedures for adding, reporting, and deleting breakpoints.

Adding Breakpoints

To add a breakpoint, perform the following steps:

1. Choose the following pulldown menu path from the menu bar (**Add > Breakpoint**)
The Add Breakpoint dialog box appears.
2. Click on the **Expression** choice button.
3. Enter a signal name of either a pin, net, or bus in the Expression entry box.
4. Click on the **On change** button to specify that the breakpoint occurs when the evaluation of the signal changes.
5. Specify in the On occurrence entry box how many times the breakpoint condition must occur before QuickSim II interrupts the simulation.
6. Click on the **OK** button to execute the dialog box.

To view a report on all of the defined breakpoints, choose the following pulldown menu path from the menu bar: (**Report > Setup > Breakpoints**)

Deleting Breakpoints

To delete a breakpoint, perform the following steps:

1. Choose the following pulldown menu path from the menu bar (Delete > Breakpoints)
The Delete Breakpoints dialog box appears.
2. Click on the **Expression** choice button.

3. Enter the name in the Expression entry box of a signal that currently has a breakpoint on it.
4. Click on the **OK** button to execute the dialog box.
SimView removes the breakpoint from the breakpoint list.

11.8.2. Back Tracing X States

If your design produces X signal states during simulation, you can use the Debug Gates palette to find the instances that are generating them. You can easily search back through a circuit to find the cause of an X value. First, the simulator examines the inputs of the instance that is driving the selected net. If the signal value of any input to the instance is X, it then selects that net and notifies you that the backtrace has succeeded.

11.8.3. Design Changes

You can change your design within a simulation session and then resimulate it. You can make the following types of design changes without exiting the simulator:

- **Design property changes.** You can add or change property values. This activity is referred to as “annotating the design.” You can also import ASCII back annotation files, which contain a set of property changes.
- **Swapping models.** You can substitute one model representation for another by changing the Model property. By swapping models, you can try using different manufacturing processes.
- **Reload models.** You can reload models that you have changed in Design Architect. For example, if you find a mistake in a simulation of a design containing VHDL models, you can open a Design Architect Session window, fix the model, and then reload the modified model in the QuickSim II Session window. You can perform the entire modify and reload process without re-invoking QuickSim II. The simulator keeps track of all incremental design changes. This tracking ensures compatibility with related design information that you save. The simulator automatically checks the data objects that are dependent on the design configuration and prohibits them from being used if they are not compatible.

11.8.4. Viewing a Simulation Timing Report

To view a timing report, which contains all of the timing related information for selected signals, perform the following steps:

1. Select one or more pins, nets and busses in the Schematic View window.
SimView highlights both the selected objects in the Schematic View window and the related signals in both the Trace and Monitor windows.
2. Press the Menu mouse button and choose the following menu path from the popup menu: (**Report > Timing > Selected**)

SimView opens a Timing Info window that describes the selected signals.

11.8.5. Using Cursors to Get Waveform Information

Cursors are domain markers that provide you with a quick way of getting information at a point on a waveform. To add a trace cursor to the Trace window, perform the following steps:

1. Activate the Trace window by moving the mouse pointer inside it and clicking on the Stroke/drag (middle) mouse button.
2. Click on the red **DBG GATES** palette button.
The Debug Gates palette menu appears.
3. Click on the **ADD CURSOR** icon in the palette menu.
The Add Cursors dialog box appears.
4. Enter any cursor name that you want to assign to the new cursor.
5. Click on the **OK** button to execute the dialog box.
SimView adds the cursor to the Trace window at time zero.
6. To move the trace cursor to a different point in the time domain, perform the following steps:
 - a. Click on the **SLIDE CURSOR** icon in the palette menu.
 - b. Move the mouse pointer to a new location in the Trace window and click the Select mouse button.

SimView moves the trace cursor to the new time domain location.

After you setup the schematic, trace, list, etc. windows within SimView, you are ready to simulate the design by generating stimulus

Performing Static Timing Analysis

One of the most critical aspects of VLSI design is timing analysis. The ADK includes the necessary library files to use the SST Velocity static timing analysis tool to perform slack analysis, find longest or shortest delay paths, or examine clock statistics. The ADK also supports post-layout back annotation of designs for static timing analysis.

12.1. Setting Up and Invoking SST Velocity

Before working with a VHDL design, you must define a velocity.map file in your local directory that will map the logical names of design units and standard packages to their physical locations. The velocity.map file must include the locations of all design subunit files which are referenced in the top level file. For a description of creating a velocity.map file and examples, refer to the VHDL section of the “Setting Up a Design” chapter in the SST Velocity User’s Manual.

After creating the velocity.map file (VHDL users only), you must then set the target technology. In the pull-down menu ‘File’, choose the ‘Set Technology’ option. Choose your desired technology from the list of available libraries.

When you have chosen a target technology, you will now be able to choose File>Open>Design. Navigate to your top level design and click on the ‘OK’ button.

12.2. Setting Timing Constraints

SST Velocity requires that you set the following constraints before executing timing analysis. Setting constraints is the most important part of timing analysis. If you set the constraints too aggressively, SST Velocity will report inaccurate timing violations. If you set constraints that are too relaxed, the resultant design might not meet the timing requirements of the system outside the design being analyzed. A more in-depth discussion of timing parameters can be found in the “Specifying Timing Parameters” and “Backannotating a Design” chapters of the SST Velocity User’s Manual.

Using the ‘Specify’ Menu

For a given level of the design hierarchy, the ‘Specify’ menu enables you to specify various timing constraints for either ports or nets. The design must be fully constrained before meaningful results can be obtained. To determine whether or not you have specified all timing constraints, click on the pull-down menu choice Report>Missing Timing Specifications>All.

12.2.1. Defining Clocks

Clocks define the timing between registers. Without clocks defined, all registers are assumed unconstrained and all combinational logic between registers will be ignored during timing analysis. When you define a clock, you have effectively constrained the maximum delay between all registers to one clock period.

You can describe clocks in SST Velocity using the Specify>Clock Definitions pull-down menu. When defining a clock, be sure to specify the clock period, duty cycle (edge positions), arrival time variance (edge region), and slew.

Setting a Single Clock

If your design only has one clock, you define the clock using the commands in the previous example. Your design will have only one clock domain.

Setting Multiple Synchronous Clocks

If you have multiple clocks in your design that are not in isolated clock domains (a register in one domain drives logic connected to a register in another domain), then you may define multiple clock signals within the same clock domain.

12.2.2. Defining Constraints on Input Pins

The input arrival time attribute specifies the maximum delay from outside the present design to the input port and the minimum and maximum slew that signal may exhibit. You can specify input arrival constraints at any input ports in your design, at any level of the hierarchy. All input arrival times start at time zero and cannot be specified relative to a particular clock edge. Arrival times are specified by choosing 'Input Arrival Times...' from the 'Specify' pull-down menu.

12.2.3. Defining Constraints on Output Pins

The output required time specifies the maximum time allowed from the register (or primary input) to the output port. The time value is always relative to time zero and cannot be specified relative to a particular clock edge. You can specify output constraints at any output ports in your design, at any level of the hierarchy. Required times are specified by choosing 'Output Setup/Hold Times...' from the 'Specify' pull-down menu.

12.2.4. Defining False Paths

False paths are paths in the design that you want SST Velocity to ignore during timing analysis. To eliminate a false path from consideration during timing analysis, designate the path as a false path by indicating the 'from' and 'to' points or "through" points on the path. These points can be listed in the 'False Paths...' option in the 'Specify' pull-down menu.

12.2.5. Defining Constant Levels

It may be necessary in certain designs to define pins as being held at a constant logic level. Such paths may include set, reset, or enable pins that would put the design in an undesired state for timing analysis. To define a constant level on a pin, choose the 'Assign Constant Level.' option from the 'Specify' pull-down menu.

12.2.6. Back Annotating from a SDF File

If you intend to back annotate a design from a post-layout SDF file, you must first load the file by specifying the desired SDF file in the File>Open>Back Annotation screen. This will replace the library delay values with those calculated based on the actual physical layout of the design.

12.3. Timing Analysis

Now that the timing constraints of your design have been described, you can perform several different types of timing analysis.

To perform a slack analysis, you can select either the 'Setup/Recovery Constraints', 'Hold/Removal Constraints' or 'Clock Pulse Widths' option from the Report>Timing Violations pull-down menu. These options let you select a specific path by indicating To and From pins or you can analyze the entire design by clicking on the 'OK' button.

To determine the longest or shortest delay paths in a design, select the 'Longest Delay Paths' or 'Shortest Delay Paths' option from the Report>Delay Paths pull-down menu. Again you can either specify a particular path or analyze the entire design.

Clock information such as clock tree details, gating locations, merged clock locations, clock skew statistics, clock latency statistics, clock pulse width statistics, and asynchronous domain crossings can be determined by selecting the desired option from the Report>Clock Information pull-down menu.

A more in-depth discussion of timing analysis can be found in the "Analyzing a Design" chapter of the SST Velocity User's Manual.

ADK Menus and Commands

The ADK provides some enhanced functionality over the standard Mentor tools. There are several new menus, palettes and commands that have been created to ease the use of the tools for ADK designs. You will only see these enhancements if you use the ADK version of the tools (such as *adk_da* or *adk_ic*). If you use the standard tool, it will not have any of the ADK enhancements.

M.1. ADK_DA

adk_da is the command that will start Design Architect with the ADK extensions. If you use this command, you will have the following additional functionality in Design Architect:

M.1.1. ADK Libraries menu

Under the standard DA *Libraries* menu you will see a new item **ADK Libraries**. This selection will display the ADK Libraries palette menu which will have all of the libraries specific to the ADK on it.

- **Basic Logic Gates:** This library contains all of the schematic models for the basic logic standard cell gates and their variants: AND, NAND, NOR, OR, FADD, MUX. The number following the gate name indicates the number of inputs for the AND, NAND, NOR and OR gates. The number after FADD indicates the number of bits in the full-adder and for the MUX indicates the number of inputs and outputs (e.g. MUX21 has two inputs and one output).
- **Inverters and Buffers:** This library contains all of the schematic models for the inverters and buffers. Each in multiple drive strengths. The number following each gate is the drive strength for the gate. These are relative numbers so an INV02 has twice the drive strength as an INV01. See the datasheets on the cells for more specific, absolute, drive strength information.
- **AO Gates:** This library contains all of the schematic models for the AND-OR gates. The numbers following the name indicate the groupings of inputs to the AND gates. An AO21 gate has a two-input AND gate and a one-input AND gate (a wire) all feeding into a two-input OR gate. An AO221 would have two, two-input AND gates and a wire feeding into a three-input OR gate.
- **AOI Gates:** This library contains all of the schematic models for the AND-OR-INVERT gates. The numbers following the name indicate the groupings of inputs to the AND gates. An AOI21 gate has a two-input AND gate and a one-input AND gate (a wire) all feeding into a two-input NOR gate. An AOI221 would have two, two-input AND gates and a wire feeding into a three-input

NOR gate.

- **OAI Gates:** This library contains all of the schematic models for the OR-AND-INVERT gates. The numbers following the name indicate the groupings of inputs to the OR gates. An OAI21 gate has a two-input OR gate and a one-input OR gate (a wire) all feeding into a two-input NAND gate. An OAI221 would have two, two-input OR gates and a wire feeding into a three-input NAND gate.
- **Flip-Flops/Latches:** This library contains all of the schematic models for the d-type flip-flops, d-type latches and scan, d-type flip-flops. The *r* and *s* letters following the cell name indicate the presence of any asynchronous reset and/or set inputs, respectively.
- **Pads:** This library contains all of the schematic models for the pads in each supported technology. This library is subdivided into a separate library for each technology. Be sure to select the appropriate technology for the pad libraries so as to match your target technology. See the MOSIS web page for more information on the pads: <http://www.mosis.org>.
- **SDL Parts:** This library contains schematic models for the parts that may be used for schematic driven layout (SDL). These include four-terminal FETs for all technologies and capacitor and resistors for some technologies. In addition, some generic parts are supplied, such as GND, VDD and ports.

M.2. ADK_IC

`adk_ic` is the command that will start IC Station with the ADK extensions. If you use this command, you will have the following additional functionality in IC Station:

M.2.1. ADK pulldown menu

Once you open a cell, you will see a new, additional menu called *ADK* in the menu bar. This menu allows you to do several tasks specifically related to ADK designs:

- **Show ADK Palette:** This will change the palette to display the ADK palette menu. This is a quick way to display that menu no matter which palette is currently displayed.
- **Scale:** This will display a submenu that will allow you to scale your design based on your target technology. You must have your cell reserved for edit and you must be in GE mode to scale. It is recommended that you instantiate your final layout into a new GE mode cell and then scale that new cell. In this way you can more easily retarget to a different technology should you later decide to do so.
- **Generate Padframe:** This will display a submenu where you can automatically generate a padframe for a 40-pin tinychip based on your target technology. See the chapter Automatic Padframe Generation for more information.

M.2.2. ADK palette menu

A new palette menu has been created which contains most of the commonly used operations that you will need for doing designs with the ADK. You can show this palette by selecting ADK->Show ADK Palette from the pulldown menu or by clicking on ADK Edit at the bottom of the session palette.

The first section of the ADK Edit menu will let you gain quick access to other palette menus.

- **Top:** will take you to the top-level, session menu
- **Edit:** will take you to the standard edit menu for more editing functions than are presented on the ADK Edit menu
- **P&R:** will take you to the place and route palette menu for placement and routing of your standard cell designs.

The next section, labeled *SDL* has functions to help you preform your schematic-driven layout.

- **Logic:** This will display a submenu that will let you set your logic source, close the logic or update your logic should you have made an update outside of IC Station.
- **Open:** opens the current logic source in a schematic window
- **Place:** This will display a submenu to allow you to select instances or ports, and to place unplaced instances and ports.
- **Inst:** Instantiates all selected instances into your cell or all unplaced instances should you not have any schematic instances selected.
- **AutoInst:** will automatically instantiate all unplaced instances into your cell.
- **Port:** place selected ports or all ports if none currently selected.
- **DEdit:** This will display a submenu that will give you options on editing placed devices. You will have options to move MOS pins, split and join MOS devices, define the folding for MOS devices and modify capacitor and resistor properties as well.
- **Clean:** This will clean up your schematic should a placement abort due to an error. If your schematic updating seems to be stuck, try to clean it up.

The section on routing has many useful functions for routing your devices and cleaning up your layout.

- **Route:** This is a menu with options to help you create overflows and manage your routing database.
- **SsNet:** This function will select all nets that are shorted or of the same net to the currently selected object.
- **MkPort:** This will make the selected object a port. You will be prompted for the type and name of the port.
- **SsPort:** The nets connected to the selected port will be selected for you.
- **ResAll:** This function will restructure all nets in your design and update the overflows to display the updated database. Useful after moving many objects or if you see stray overflows in your cell.
- **ResSel:** Restructure selected net(s) only.
- **AutoR:** Autoroute the currently selected overflow. This will only route on the

metal routing layers. It will not route on poly.

- **Comp:** Compact your design. This runs the IC Station compactor.

The next section has functions to add various shapes and text.

- **Shape:** This is the Add Shape command to allow you to add arbitrary shapes on any layer you wish.
- **Path:** This function allows you to place paths in your cell.
- **AddR:** Add Route. This is the routing tool to manually route your design. This will only route on the metal routing layers defined in the process.
- **Text:** Allows you to add text to your cell.

The Edit section provides several functions for editing your cell and the objects in the cell.

- **Copy:** Copy selected object(s) to location you specify.
- **Move:** Move the selected object(s) to the location you specify.
- **Align:** Align selected object(s).
- **Rotate:** Rotate the selected object(s).
- **Flip:** Flip the selected object(s).
- **Stretch:** Stretch the selected objects (cannot be used on devices).

The verification section of the palette menu allows your to easily perform DRC/LVS and extraction of your cell without leaving the palette.

- **DRC:** This menu allows you to run the DRC check and then to scan through the results and view your DRC errors, if any.
- **Load Rules:** If you need to change the rules that you are using, use this option to load a new rules file.
- **Extract:** This function will perform a lumped, mask-level extraction of your circuit suitable for simulation with ELDO, Mach-TA or Accusim.
- **LVS:** Run a mask-level LVS of your cell.
- **Report:** LVS reporting functions. The options in this menu allow you to view the LVS report and to check your LVS errors, if any.

Finally, the last item is **Repeat**. This is a toggle switch that will turn on or off the repeatable functions. Functions with an asterisk (*) by them will repeat. If you do not want those functions to repeat, click on the **Repeat** item to turn it off, and click on it again to turn it back on.

M.2.3. ADK Commands

In addition to the new menus, there are some new commands (or macros) that have been included with `adk_ic`. Some of these commands are specific to creating and maintaining your standard cells, and are offered as examples that you can modify for your own needs.

To use any of these commands, simply type them in the cell window. A prompt bar will appear with your command in it.

Table M-1.

<i>Function</i>	<i>Description</i>
pc	Place a poly contact; places a poly contact centered at the current cursor position.
pp	Place a port contact; creates a port and places it where you click.
pr	Places standard cell power rails; Places a VDD and a GND power rail in metal 1 with the height of a standard cell. Also creates proper port and net properties for the rails. These rails are short, but can be stretched to and desired width for your cell.
nwc	Place an n-well contact; creates and places an n-well contact on the standard cell power rails.
pwc	Place a p-well contact; creates and places p-well contact on the standard cell power rails.
pfp	Place an fp1 layer; For standard cells, you need to specify an fp1 (floor plan) layer. This command automatically generates and places one to the extent of the cell.
p	Place a poly path of width 2 (minimum width); Prompts you to place a poly path.
m	Place a metal1 path of width 3 (minimum width); Prompts you to place a metal1 path.
v	Move unconstrained; Moves the selected object(s).
ec5	Lumped, mask extraction for AMI 0.5; This command scripts the extraction to a specific location that probably does not exist at your site. Modify this command if you wish to use it.
ec12	Lumped, mask extraction for AMI 1.2; This command scripts the extraction to a specific location that probably does not exist at your site. Modify this command if you wish to use it.
ab1	Add metal1 blockages to entire cell; This command will place a metal1 blockage on top of all metal1 shapes in your cell. This is useful for autorouting with hierarchy.

Table M-1.

<i>Function</i>	<i>Description</i>
rab1	Remove metal1 blockages; This command will remove all metal1 blockages in your cell.
s12	Set cell process to AMI 1.2
s5	Set cell process to AMI 0.5
cp	Checkpoint cell; save the cell and then reserve the cell.
dup	Delete unplaced ports; If LVS reports unplaced ports but you see that they are all placed from your schematic, the database might have duplicate ports. This command will remove them.