PALACE™

Actel Edition Version 3.1

A Programmable Logic Physical Synthesis Solution

User's Manual

Magma Design Automation, Inc.
5460 Bayfront Plaza
Santa Clara, CA 95054
(408) 565-7500
http://www.magma-da.com

Contents

1. Introduction

PALACE™ (*Physical And Logical Automatic Compilation Engine*) *Actel Edition* is a physical and logic synthesis tool that supports the Actel ProASIC Plus, ProASIC 3, AX, and RTAXS device families.

The front-end interface, or the input to PALACE, takes a design netlist generated by the Actel Designer tool's EDIF interpreter and a set of design constraint files in SDC, GCF and PDC formats. On the back-end, the output from PALACE is fed to Designer for placement and routing. PALACE is a command line tool only.

2. System Requirements

This section gives system requirements and the RAM and swap space needed to run PALACE on your system.

PALACE is supported on Windows NT, 2000, XP, and Solaris systems. Tcl version 8.0 or above is required for running the scripts provided with the PALACE package.

Please note that while the following list describes the system requirements for typical designs, the unique characteristics of each individual design will affect the actual system resources required.

- A minimum of 256MB RAM and 256MB virtual memory is required to run PALACE.
- For small to medium size devices, from APA075 to APA450, 256MB RAM and 256MB virtual memory are sufficient.
- For medium to large size devices, from APA600 to APA1000, 512MB RAM and 512MB virtual memory are required.

Some designs can be implemented using less than the specified memory while other complex or large designs may require additional memory. Each designer must monitor the system resources and adjust the systems resources if necessary. Windows Task Manager for Microsoft Windows and the Top command for Solaris systems are useful for this purpose.

3. Design Flow

3.1. Customer Design Flow

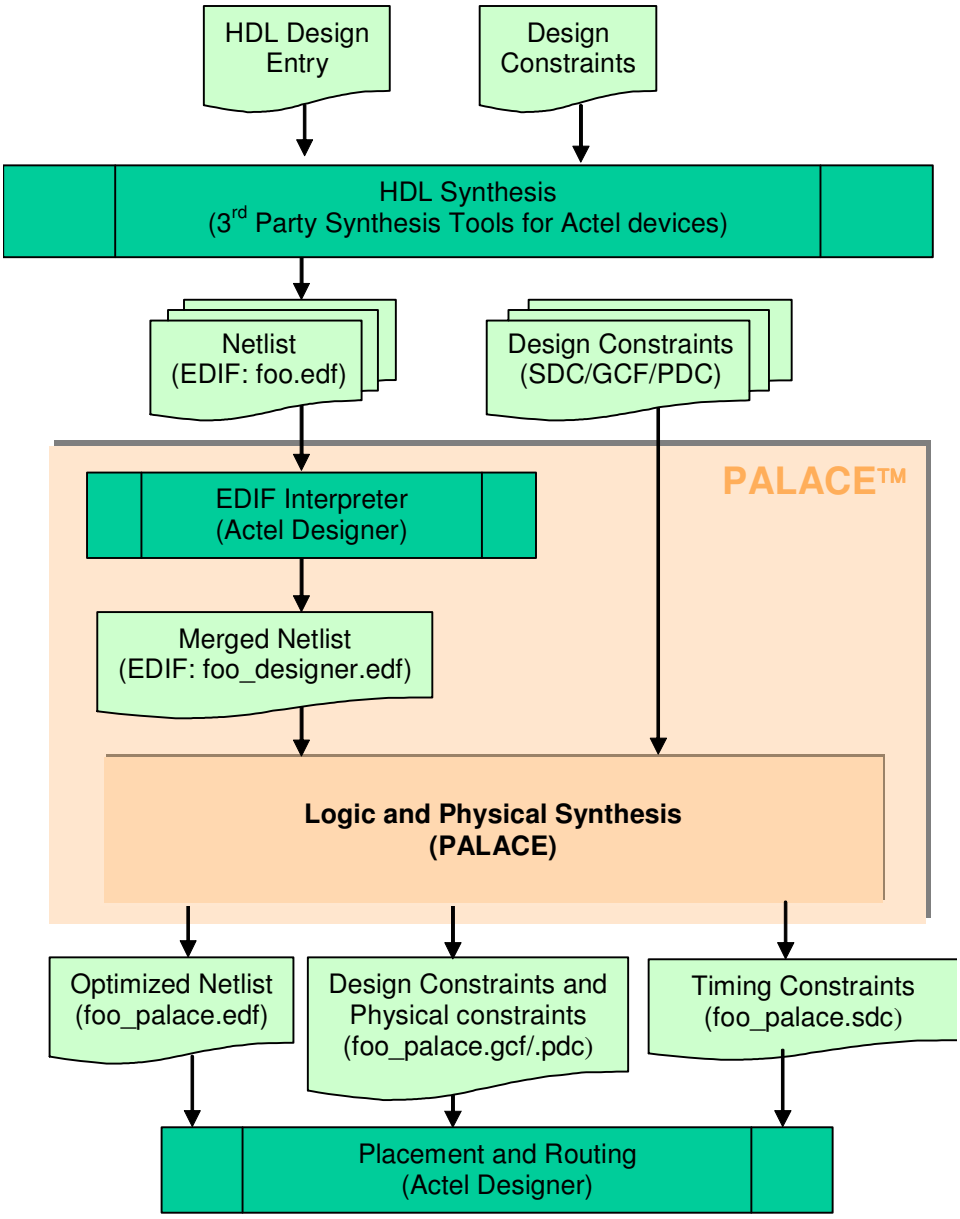

Figure 1 PALACE Customer Design Flow

In the PALACE customer design flow (see Figure 1), a front-end tool processes the customer's design and generates an EDIF netlist (*foo.edf*). PALACE then transparently launches Designer to translate the netlist to *foo_designer.edf*. Once this is done, PALACE performs logic and physical synthesis on the *foo_designer.edf* netlist while honoring the supported user

constraints in *foo.gcf* or foo.pdc. PALACE then generates an optimized and mapped netlist (*foo_palace.edf*), along with the corresponding constraint files (*foo_palace.gcf/pdc* and *foo_palace.sdc*), which serve as the inputs to Designer. Designer will then perform placement and routing and report the timing results. For the constraints supported by PALACE, please refer to Section 9.

The definitions of the major functional modules depicted in Figure 1 are as follows:

- PALACE

  PALACE (*Actel Edition*) will perform the following operations:

  1. Launch Designer to translate the EDIF designs generated by a front-end design tool (*foo.edf*) to a valid EDIF (*foo_designer.edf*) before it runs physical synthesis.
  2. Perform logic and/or physical synthesis based on the input design (foo_designer.edf) and constraints (foo.gcf).
  3. Output an optimized and mapped netlist (foo_palace.edf) and constraint files (foo_palace.gcf and foo_palace.sdc), which will serve as the inputs to Designer.

- Designer

  Designer is Actel's Placement and Routing tool. It takes PALACE outputs (foo_palace.edf, foo_palace.gcf and foo_palace.sdc) as its inputs, and performs compilation, placement and routing, and timing analysis.

3.2. User Requirements

There user requirements for PALACE are as follows:

- Designer (V4.6 and above) must be installed before launching PALACE along with a valid license. Designer should also be included in the system path, which means that the user can launch Designer from the command line without explicitly typing in the path.
  - If the Designer location is different from the system path, the user can use "-designer_location" option to specify Designer with the full path.
- The PALACE_ROOT system variable must be setup correctly and windows should be restarted after the variable setup. (Please refer to the following section for details.)
- The PALACE license must be setup correctly. (Please refer to the licensing section for more information.)
- The user may use the TCL scripts provided to run PALACE and Designer in batch mode. However, the user should make sure that the settings in aa_prun_set.tcl are updated to his/her environment. For details, please read the later sections on this topic.

4. Installation

PALACE is supported on workstations and personal computers running Microsoft Windows platform including Windows NT, Windows 2000, Windows XP Professional Edition (SP2 for XP is NOT supported), and the Sun Solaris platform.

4.1. Microsoft Windows Platform

To install the PALACE software:

1. Close all programs before you begin the installation.
2. Run the self-extracting executable or unzip the archive.
3. Follow the instructions on the screen to install the software.
4. Set the **PALACE_ROOTDIR** environment variable to the directory where PALACE is installed in Windows system environment. For example, if PALACE is installed in *C:\Magma\PALACE\1.0*, set this path for PALACE_ROOTDIR). (To set an environment variable, you can right click *system icon* on the desktop, chose *Advanced*, click *Environment Variables*, and then add a new variable.)
5. Setup the PALACE license as indicated in the licensing section.
6. Make sure you reboot your system after installation. After rebooting, you should be able to invoke PALACE anywhere on your Windows platform.

Note: PALACE uses the tcl84.dll file in the PALACE installation directory. If there is an older Tcl DLL file in your system path that occurs before this path, it may lead to unpredictable errors.

4.2. Sun Solaris Platform

To install PALACE software:

1. First unpack the package to an empty temporary directory. The `tar` command can be used to do this as shown in the following example

   ```
   tar -xvf archived_file.tar
   ```

2. Run the `install_palace` script. The script assumes you have access to the `gunzip` and `tar` utilities.

   Follow the instructions on the screen to install the software. Please make sure to set the environment variables as displayed by the script after it has finished installation.

   For detailed instruction, please refer to *readme.solaris* in the current directory after you unpack the package.

4.3. Release Content

The release of PALACE includes:

1. /bin

   Includes PALACE executable, and licensing daemon

/actel

Includes Actel device files

/doc

Includes PALACE documentation

/tutorials

Includes tutorials to get the user quickly started in using PALACE

/scripts

Includes sample TCL scripts to run PALACE and Designer in batch mode to automate the multiple-run process that in many cases yields better results.

5. Licensing

PALACE uses the FLEXlm® licensing manager. The license file can be saved under any name, but a feature line for the device family must be present in the license file in order to run PALACE for that device family. Table 1 lists the feature names in the license file.

Table 1: FLEXlm feature names for PALACE

FLEXlm Feature Name	Description (device families licensed)
PALACE_ACTEL_APA	ProASIC Plus, ProASIC 3, AX, and RTAXS

5.1. License types

There are two types of licenses for PALACE:

1. Stand-alone licenses are ones that are locked to a machine and enable a particular machine to run PALACE. Licenses can be locked to a hard disk, host ID (on Solaris), network card ID, or Actel Libero Dongle ID.
2. Floating licenses (network licenses) are ones that are issued by a license server to clients on a network. This type of license allows a central location for licenses, and is useful for a site that requires multiple licenses. However, it also requires the additional step of setting up a license server.

5.2. Setting the license file location

PALACE determines the location of a valid license file through either of two environment variables:

- APLUS_LICENSE_FILE
- LM_LICENSE_FILE

NOTE: APLUS_LICENSE_FILE may be more convenient to use if you have other FLEXlm enabled applications that rely on the LM_LICENSE_FILE environment variable.

For stand-alone licenses, one of these variables should point to the license file on the local file system. For floating licenses, set the value to port@host, where "port" is the port

number specified in the license server's license file, and host is the name of the license server. Multiple locations for the license file may be specified by using ';' and ':' as separators for Windows and Unix systems respectively. Some examples are:

Table 2: Example values for the environment variables

Environment variable value	Description
c:\flexlm\magma_license.dat	Stand alone license on Windows located in c:\flexlm\magma_license.dat
2201@mars	Floating licenses will be checked out from the FLEXlm license server named "mars" (for Windows or Unix)
c:\flexlm\magma_license.dat;2201@mars	Stand alone license will be checked and if invalid, the floating license will be checked

5.3. Using stand-alone (node locked) licenses

Once the license file location has been set, PALACE will be licensed for device families based on the license feature lines.

5.4. Using Floating (network) licenses

In order to use a floating license on a network, the Magma vendor daemon must be installed with your FLEXlm license software. This is done by copying the "aplus.exe" binary file on Windows, or the "aplus" binary file on Unix, to the FLEXlm directory on the license server. These files are located in the PALACE /bin directory. Once the Magma vendor daemon has been installed, the license file on the license server must be customized according to the user's settings. In the sample license file shown below, the text in between and including the '<' and '>' must be replaced.

```
#Sample license for PALACE
SERVER <host_name> DISK_SERIAL_NUM=3e3f17fd <port>
USE_SERVER
VENDOR aplus "<path>"
# License for 10 users expiring June 30, 2003
FEATURE PALACE_ACTEL_APA aplus 1.000 30-jun-2003 10 \
      SIGN="02CB 5A07 A1C1 17B7 0ADA 133D 8601 F168 C342 9274 \
      5D03 DC43 850D 118D 296A E38E D8A6 8EA6 D9E4 5B6A 441F"
```

5.5. Additional licensing Information

The licensing information contained in this document covers the aspects related to PALACE. Detailed information can be found in the FLEXlm End User Guide (http://www.macrovision.com/).

6. Using PALACE in Command Mode

PALACE v3.1 is available in command mode only and can be controlled through a set of simple and intuitive command options.

6.1. Command Line Options

All inputs to PALACE should be specified as options. Some options may not take any value in which case they essentially act as switches. Options can be given in any order, but the value for an option must immediately follow the options. This section describes all options that PALACE currently supports. The user can also type in "`palace_actel -family apa -help`" in command line to get the latest information on all supported options as well as their default values.

PALACE takes the following options from the command line:

```
Usage: palace_actel -in_design <file> -family <family>
   [-in_design_format <actel_edif|generic_edif>] [in_constraint <file>]
   [-out_design <file>] [-report <file>] [-overwrite] [-edif_passed_des]
   [-device <part>] [-die <name>] [-package <name>][-
   print_device_support] [-designer_location <file>] [-speed <STD|-F>]
   [-logic_effort <0-4>] [-physical_effort <1-2>] [-relax_delay <0-50>]
   [-max_cell_perc <1-100>] [-palace_rootdir <directory>]
```

Where "-in_design" and "-family" are mandatory options, and the others are optional. The option description is as follows.

```
-in_design  <file>
      Specify the input design file name (e.g. foo.edn).
      If there is some space in the path, please use the format:
            -in_design "c:/my dir/foo.edn"

-family  <family>
      Specify device family.
      apa:    ProAsic Plus
      a3p:    ProAsic 3
      a3pe:   ProAsic 3e
      ax:     AX and RTAXS

-in_design_format  <format>
      Specify the input design format.
      actel_edif: Actel primitive EDIF format
      generic_edif: Generic gate EDIF format
      (Default: actel_edif)

-in_constraint  <file>
      Specify the input constraint file in GCF format.
```

```
      (e.g. foo.gcf) (Extension must be gcf).
      If there are several constraint files, please use this option
      for multiple times, each specifying a constraint file.
      (Note: If no timing constraint is given to PALACE,
      it will optimize delay for all register to register
      paths, i.e. to minimize the slowest clock period.)
      If there is some space in the path, please use the format:
            -in_constraint "c:/my dir/foo.gcf"

-out_design  <file>
      Specify output design file.
      (Default: <design_name>_palace.edf)

-report  <file name>
      Specify PALACE report file name.
      (Default: palace.rpt)

-overwrite
      Specify to overwrite existing files with PALACE outputs.
      Without "-overwrite", PALACE output design and output
      constraint files will not overwrite existing files.

-device  <name>
      Specify the device. (e.g. APA600-PQ208)
      It overwrites the device specified in EDIF design input.
      (Default: APA600-PQ208)

-die  <name>
      Specify the die name. (e.g. APA600)
      (If -device is used, this option will be ignored.)
      (If -die is used, -package should also be used.)

-package  <name>
      Specify the package name. (e.g. PQ208)
      (If -device is used, this option will be ignored.)
      (If -package is used, -die should also be used.)

-print_device_support
      Print out all the devices supported by this software.

-designer_location <designer_file>
      Specify designer file with full path.
      (e.g. d:/Actel/bin/designer).
      (Default: designer is not called.)

-speed  <speed_grade>
      Specify speed grade
      APA: STD (standard) "-F" (slower)
      Other families: STD (standard) "-1" (faster) "-2" (fastest)
      The option value is case insensitive.
      (Default: STD)
      (Note: Please use the fastest speed to run Palace.)


-logic_effort  <0-4>
      Specify logic synthesis effort level.
      Level 4 is the maximum effort level.
      (Default: 4 for AX
              : 3 for others )
      (Note: 0 is only for APA when physical_effort is 2)

-physical_effort  <1-2>
      Specify physical synthesis effort level (1 to 2).
      Level 2 is the maximum effort level. (For APA only:
```

```
      If logic_effort is 0, physical_effort must be 2.)
      (Default: 1)

-relax_delay  <perentage number>
      Specify delay relaxation percentage. For example, 10 means
      allowing 10% delay relaxation on the best possible delay
      PALACE can achieve.
      (Default: 0)

-keep_device
      Specify trying not to promote device.
      (Note: Sometimes Palace still has to promote device)
      (Default: do device promotion if needed)

-max_cell_perc  <perentage number>
      Specify maximum core cell utilization percentage. For
      example, 80 means allowing the maximum of 80% core cell
      utilization. When the actual utilization exceeds the
      specified percentage, PALACE will automatically choose
      to use a bigger device with the same packaging and
      speed grade that satisfies this percentage.
      (Default: 100)

-palace_rootdir <directory>
      Override the PALACE_ROOTDIR environment variable to
      <directory>.
      This directory contains supporting files for PALACE.
      Actel specific files are located in <directory>/actel.
      (Default: $PALACE_ROOTDIR/actel)

-help
      Print out the online usage
```

For example, the following command,

"palace_actel -family apa -in_design *foo.edf* -logic_effort 3 -physical_effort 1 -in_constraint *foo.gcf*"

will turn on logic effort level of 3 and physical effort level of 1. The output design will be saved to `foo_palace.edf`. The output constraints will be saved to `foo_palace.gcf` and `foo_palace.sdc`. PALACE will also output a TCL file named `foo_palace_compile.tcl` to be used for Designer Compile.

When no timing constraints are given to PALACE (through –in_constraint foo.gcf), PALACE will optimize the delay for register to register paths to minimize clock period.

6.2. Running Designer after PALACE

Upon successful completion, PALACE will generate following files.

1. The GCF file *foo_palace.gcf* contains all the content in the input constraint file `foo.gcf` and additional (physical) constraints generated by PALACE
2. The SDC file *foo_palace.sdc* contains all timing constraints from the input constraint file *foo.gcf*

3. TCL file *foo_palace_compile.tcl* that can be used for Designer Compile. The command to run Designer Compile with this TCL file is: `designer script:foo_palace_compile.tcl`

These files will be the input to the Designer implementation tool for placement and routing.

7. PALACE Report

PALACE reports the settings and options used, optimization progress, and a summary of results in a report file. By default report file is named `palace.rpt and placed` in the working directory. The PALACE report consists five types of information, namely

- Setting and option summary
- Progress report
- Device utilization report
- Clock report
- Timing report

Setting and option summary

At the beginning of the report file, PALACE reports the setting and options that the PALACE run used. Information such as device family, part name, and optimization effort levels are reported in this section.

Progress Report

PALACE reports the status on input interface, physical synthesis, and output interface in the report file, as well as the run time for each major operation. At the end, it reports the total run time and peak memory usage.

Device Utilization Report

As part of result summary, PALACE reports the utilization information of the final design on the target device. The following is an example of utilization report:

```
Section 1: Device utilization summary
-------------------------------------
```

```
Device name: APA075-PQ208
Core cells:   217 out of   3072      7%
RAM/FIFOs:      0 out of     12      0%
IOBs:          61 out of    156     39%
PLLs:           0 out of      8      0%
```

Clock Report

In the clock report section, PALACE reports all the clocks in the design, global and local, and the number of flip-flops that each of them drives. The following is an example of clock report:

```
Section 2: Clock report
-----------------------


------------------------------------------------------------
           clock name            | resource type | # fanouts
------------------------------------------------------------
               CLK_1             |     Global    |      1175
------------------------------------------------------------
               CLK_2             |     Global    |      3152
------------------------------------------------------------
```

Timing Report

As part of result summary, PALACE reports the estimated timing of final design. The timing is based on the assumption that the implementation tool will be run in a timing-driven mode. Magma recommends using Designer's timing driven place and route option.

The report provides the following information, all based on multiple clock timing analysis:

- Circuit level minimum period if no timing constraints are specified
- Circuit level minimum slack in case of timing constraints
- Clock period for each constrained clock
- Clock to output pad delay for each constrained clock
- Input pad setup delay for each constrained clock
- Pad to pad delay for each constrained path

The following is an example of timing report:

```
Section 3: Timing report
------------------------

* Minimum slack
-------------------------------------------------------------------
       initial      |      post PALACE      |   slack improvement
-------------------------------------------------------------------
        -4.79 ns    |         1.25 ns       |        6.04 ns
-------------------------------------------------------------------
* Clock period
-------------------------------------------------------------------
   clock name  |    initial    |    required   |  post PALACE
-------------------------------------------------------------------
   CLK         |    14.79 ns   |    10.00 ns   |     8.75 ns
-------------------------------------------------------------------
```

8. Scripting

The TCL scripts provided with the PALACE release are very helpful in automating the PALACE-to-Designer design flow. These scripts are especially useful in exploring multiple runs of the flow to achieve better quality of results. For each run, PALACE will process the design with a specific logic and physical effort option. Designer will then run place and route using a particular seed. From the multiple runs, the one with the best FMAX will be chosen and reported. The logic effort level and physical effort level used by PALACE, and the seed for Designer place and route for each run can all be specified using the scripts. The following sub sections further explain in detail how to use these scripts.

8.1. Customizing the Script Settings

Before running the script, the following variables in file aa_prun_set.tcl must be properly set based on your environment and preferences:

1. **SRC_DIR:** specify the root directory for the source design and constraint files

   NOTE, the design file and constraint files for each design must be in an individual directory under the directory of $SRC_DIR. For example, if SRC_DIR is set as:

   ```
   set SRC_DIR "c:/my_test "
   ```

For a design named foo, the files foo.edf, foo.gcf, and foo.sdc must be put in directory named c:/my_test/foo/

2. **SRC_EXT:** specify source file's extension

   For example, if your source design file is foo.edf, then set the variable to be:

   ```
   set SRC_EXT ".edf"
   ```

3. **ACTEL_DIR:** specify the location of the Designer software
4. **PALACE,** specify the location of PALACE and options

   Example:

   ```
   set PALACE "exec c:/tools/bin/palace_actel -overwrite -family
   apa"
   ```

   Note that you should NOT specify "–in_design", "-in_constraints", "-logic_effort" and "-physical_effort" options in this variable as they can be explicitly specified in the script.
5. **USE_CONSTRAINT:** tell PALACE whether to pick up design constraints in GCF file
6. **NUM_PAL_DES_ITERATION:** specify the number of iterations to run PALACE followed by a run of Designer
7. **PALACE_LOGIC_EFFORT_LIST:** specify the "-logic_effort" levels for multiple PALACE runs
8. **PALACE_PHYSICAL_EFFORT_LIST:** specify the "-physical_effort" levels for multiple PALACE runs
9. **PALACE_SEEDS:** specify the seeds for Designer to place and route the design generated by PALACE for multiple runs
10. **LOG:** specify the root name of log file

8.2. Executing the Script

After all the variables in aa_prun_set.tcl are properly set, the script can be executed by launching a TCL shell and invoking the "`run_pal_des_file`" command to perform multiple PALACE+Designer runs. Follow these steps:

```
% source aa_prun.tcl
% run_pal_des_file <ckt-dev-file>
```

Where <*ckt-dev-file*> is a file containing the source design names and their corresponding target devices. A sample file may look like this:

```
my_design1  "APA075-PQ208"
my_design2  "APA450-BG456"
…
```

8.3. Script Commands

The following list describes the commands in the aa_prun.tcl script:

– run_palace
  - Run PALACE on all circuits specified in variable **CKTS**

- Run PALACE only once. Using the **first** value of PALACE_LOGIC_EFFORT_LIST, PALACE_PHYSICAL_EFFORT_LIST, and PALACE_SEEDS
- The original designs are in **SRC_DIR**

– run_palace_file < *ckt-dev-file* >
- Similar to the above
- All circuits are specified in < *ckt-dev-file* >, which is defined in the previous section

– run_des
- Run all circuits specified in variable **CKTS** with Designer.
- The original designs are in **SRC_DIR**
- Number of designer iterations are specified by **NUM_DESIGNER_LAYOUT_RUNS**
- Designer seeds are specified by **DESIGNER_SEEDS**

– run_des_file < *ckt-dev-file* >
- Similar to the above
- All circuits are specified in < *ckt-dev-file* >, which is defined in the previous section

– run_pal_des
- Run PALACE + Designer on all circuits in **CKTS**
- Palace logic efforts are specified in **PALACE_LOGIC_EFFORT_LIST**
- Palace physical efforts are specified in **PALACE_PHYSICAL_EFFORT_LIST**
- Number of iterations is specified by **NUM_PAL_DES_ITERATION**
- The seeds for placement & route are specified in **PALACE_SEEDS**
- The original designs are in **SRC_DIR**

– run_pal_des_file < *ckt-dev-file* >
- Similar to the above
- All circuits are specified in < *ckt-dev-file* >, which is defined in the previous section.

– run_des_for_pal
- Run Designer for PALACE result on all circuits in **CKTS**
- The sub-directory under each design where PALACE results are stored is specified by **SUB_DIR**
- All layout script to run designer is in <ckt_name>_palace_compile.tcl
- The original designs are Palace designs in the working directory

8.4. Obtaining the result

After running the script, for each design foo, a directory foo/ is created under the directory where you ran the script. To check the result, in directory foo/:

1. The complete log information for all runs of PALACE+Designer is recorded in the file $LOG.log

In the $LOG.log file, for each PALACE+Designer run, the frequency of the slowest clock and slowest constrained clock are reported as follows:

```
----------------------
- Result of iteration 1
----------------------
--------------------------------
- clk = CLK1 fmax = 113.16
--------------------------------
-------------------------------------------
-   [constrained]clk = CLK2, fmax = 103.16
-------------------------------------------
```

The best of the multiple runs is reported at the end of the file:

```
============================================================
= ===>Slowest clock : CLK1
= ===>Fmax of slowest clock : 117.01
============================================================
============================================================
= ===>Slowest constrained clock : CLK2
= ===>Fmax of slowest constrained clock : 127.01
============================================================
```

Check the saved detailed result of each PALACE+Designer run in directory ite_<n>/, where n is the run number.

9. Getting Familiar with PALACE through an Example

1. Launch **cygwin** or **DOS Command Prompt**. (cygwin is downloadable at http://www.cygwin.com/) .

2. Check if Designer is executable from the window by typing **designer**. (Then quit designer)

3. Go into the PALACE installation directory. Under **tutorials**/**Actel**/, there are three directories. The "**designs**" directory contains two subdirectories, each containing a design called foo for the APA and PA3 families. The other directories are named "**<family>_test**", which are the test directory for each device family.

4. Go into the **<family>_test** directory of your choice. Open **aa_prun_set.tcl** to check the following settings and make necessary changes according to your environment:

- **set ACTEL_DIR {d:/Actel}**

  Change "d:/Actel" to your Designer installation directory

- **set BINFILE "d:/test/apa/bin/palace_actel"**

  Change the path to your path.

- **set SRC_DIR "../../designs"**

  You may need to change this setting if you use another design directory.

5. Type tclsh (or the tcl command you are using.)

6. In the Tcl shell, type the following commands sequentially

- *source aa_prun.tcl*

- *run_pal_des_file des_dev.txt*

The screen shot for the APA test is depicted below:

```
$ tclsh
% source aa_prun.tcl
% run_pal_des_file des_dev.txt

circuits: foo


>>>>>>>>>>>>>>>>>> foo <<<<<<<<<<<<<<<<<<
oooooo iteration 1/3
o running PALACE(w/ GCF)(logic_effort=3, physical_effort=1)
o Running Designer compile o Running Designer layout (on PALACE design, seed=78381505)
o   === this iteration: clk, fmax=149.97, BEST FMAX=149.97
      [constrained]: clk, fmax=149.97, BEST FMAX=149.97
oooooo iteration 2/3
o running PALACE(w/ GCF)(logic_effort=4, physical_effort=1)
o Running Designer compile o Running Designer layout (on PALACE design, seed=78381505)
o   === this iteration: clk, fmax=128.47, BEST FMAX=149.97
      [constrained]: clk, fmax=128.47, BEST FMAX=149.97
oooooo iteration 3/3
o running PALACE(w/ GCF)(logic_effort=0, physical_effort=2)
o Running Designer compile o Running Designer layout (on PALACE design, seed=78381505)
o   === this iteration: clk, fmax=155.01, BEST FMAX=155.01
      [constrained]: clk, fmax=155.01, BEST FMAX=155.01

%
```

7. Go into foo/ and check the results.

- **test_result.log** contains all messages from PALACE and Designer

- **ite_1**/, **ite_2**/ and **ite_3**/ store PALACE and Designer results for each iteration.

10. Constraints Support

In this release, PALACE supports a subset of input constraints. Please refer to the table below for the constraints that are supported.

"Honor" means that PALACE will consider the constraints during its physical synthesis.		
"Pass" means that PALACE will take in the constraint and pass it on to the output of PALACE, however, it may not generate impact to PALACE.		
"Ignore" means that PALACE will remove the constraint.		
Constraints	**Support in PALACE**	**Notes**
GCF Timing constraints		
create_clock	Honor	
generate_paths	Pass	
set_false_path	Honor	
set_input_to_register_delay	Honor	
set_multicycle_path	Honor (Ignore -through, assuming through all ports)	
set_register_to_output_delay	Honor	
net_critical_ports	Ignore	
set_critical	Ignore	
set_critical_port	Ignore	
set_max_path_delay	Honor	
set_switch_threshold	Ignore	
GCF Global resource consraints		
set_auto_global	Honor	
set_auto_global_fanout	Honor	
set_global	Honor	
set_noglobal	Honor	
dont_fix_globals	Honor	
use_global	Honor	
GCF Netlist optimization constraints		
dont_optimize	Honor	'dont_optimize buffer' is ignored
Optimize	Honor	
set net_region	Honor	
set_max_fanout	Honor	
dont_touch	Honor	
GCF Placement constraints		
set_empty_location	Honor	
set_empty_io	Honor	
set_initial_io	Honor	
set_io	Honor	
set_location	Honor	If location constraints are applied for hierarchical nodes, the leaf nodes' locations will be honored while the login hierarchy is ignored.
set_initial_location	Honor	
Macro	Ignore	

PDC constraints		
assign_global_clock	Honor	
unassign_global_clock	Honor	
set_io	Honor or Pass	Only honors "-pinname" and "-fixed" options, other options are passed
reset_io	Honor	
set_iobank	Honor	
reset_iobank	Honor	
set_location	Honor	
set_vref	Honor	
set_vref_defaults	Honor	
assign_local_clock	Honor	
unassign_local_clock	Honor	
define_region	Honor	
move_region	Ignore	
assign_region	Honor	
unassign_macro_from_region	Ignore	
assign_net_macros	Honor	
unassign_net_macros	Ignore	
dont_touch_buffer_tree	Honor or Pass	
delete_buffer_tree	Honor	
reset_floorplan	Honor	

Table 3 – PALACE Constraints Support

11. Getting the Best Results Using PALACE

Magma recommends reading the User's Manual and following a simple process to achieve the best possible quality of results as outlined in this section.

11.1. If Timing is the major objective

1. If there are timing constraints, and all the timing requirements are specified in the input constraint file (`foo.<gcf/sdc>`), run PALACE with:

   `"-in_design foo.edf -in_constraint foo.<gcf/sdc> -logic_effort 3 -physical_effort 1"`

   If there is no timing constraint for the design and clock period is to be optimized, run PALACE with:

   `"-in_design foo.edf -logic_effort 3 -physical_effort 1"`

If the result is not satisfactory, you can first vary `-logic_effort` between 4 and 2 **(in that order)** and after that increase `-physical_effort` to 2 (when physical effort level is 2, logic effort level will be automatically reset to 0, so there is no need to vary `-logic_effort` when physical effort level is 2) to try to get better results.

If PALACE reports good performance improvement in its report file, but the PALACE generated design cannot be routed due to bigger area, you can try two things:

2. Use `-relax_delay <5-30>` to run PALACE one more time to reduce area.

3. Use `-max_cell_perc` to specify maximum core cell utilization percentage. For example, 80 means allowing the maximum of 80% core cell utilization. When the actual utilization exceeds the specified percentage, PALACE will automatically choose to use a bigger device with the same packaging and speed grade that satisfies this percentage. The default is 100.

***Scripting*:** The TCL scripts provided with the PALACE release are very helpful to automate the PALACE to Designer design flow. Using these scripts can be very useful in exploring multiple runs of this flow to achieve better quality of results in performance. For each run, PALACE will run with specific options on its logic and physical effort levels, and then Designer place and route will run using a particular seed. The best FMAX will be chosen and reported from the multiple runs. The logic effort level and physical effort level used by PALACE, and the seed for Designer place and route for each run, can all be specified using the scripts. The "Scripting" Section in User's Manual explains in detail how to use these scripts.

11.2. If Area is the major objective

The logic synthesis effort level 1 (`-logic_effort 1`) is designed to minimize the area. Run PALACE with: "`-in_design foo.edf -logic_effort 1`"

11.3. Constraints support

In this release, PALACE supports most of the GCF and PDC constraints. Refer to the "Known Limitation" Section in User's Manual for details. The following are a few tips.

- Timing constraints are very helpful to improve the clocks user wants.
- Overly tight constraints, especially tight location constraints, may over constrain the tool, and degrade PALACE's physical synthesis results.

12. Technical Support

Technical support for this version of PALACE is provided by Actel. Please contact Actel technical support for any PALACE technical issues through either of the following:

- Email: tech@actel.com
- Phone: 1-800-262-1060