PALACE™

Actel Edition Version 1.0

A Programmable Logic Physical Synthesis Solution

User's Manual

Magma Design Automation, Inc.
10850 Wilshire Blvd.
Suite 370
Los Angeles, CA 90024
(310) 441-9365
http://www.magma-da.com

Contents

1 Introduction

PALACE™ (Physical And Logical Automatic Compilation Engine) Actel Edition is a physical synthesis tool that supports the Actel ProASIC Plus device family.

The product interfaces on its front-end with a design netlist generated by Designer EDIF interpreter and a design constraint file in Actel GCF constraint format. The product interfaces on its back-end with Actel's suite of FPGA design tools, Designer, for placement and routing. PALACE Actel Edition is intended to run in the command line mode only.

2 System Requirements

This section gives the RAM and swap space needed to run PALACE Actel Edition on your system. 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 are 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 complicated or large designs may require additional memory. Each designer must monitor the system resources and adjust the systems resources if necessary.

3 Design Flow

3.1 Customer Design Flow

In the PALACE Actel Edition customer design flow (see Figure 1), a front-end tool will process the customer's design and generate an EDIF netlist (foo_1.edf). PALACE Actel Edition will transparently launch Designer to translate the netlist before it runs physical synthesis. In the flow, Designer creates a validated EDIF file (foo_1_designer.edf) that will be used by PALACE Actel Edition. PALACE Actel Edition will process and manage any errors generated during the process. PALACE Actel Edition will also process and honor customer generated constraints supplied in foo_1.gcf file when it performs Physical Synthesis on the netlist. The PALACE Actel Edition software will perform physical synthesis targeting the same device based on these input files supplied and generate an optimized and mapped netlist (foo_2.edf) along with the corresponding constraint files (foo_2.gcf and foo_2.sdc). The PALACE generated files will serve

as the inputs to Designer. Designer will then perform placement and routing and report the timing results. Note that in this Beta release, PALACE supports only a subset of constraints in GCF. Please refer to Section 8.5 for details.

The following is the definition of major functional modules shown in Figure 1.

- PALACE Actel Edition

  PALACE Actel Edition is a deliverable from Magma. PALACE Actel Edition will transparently launch Designer to translate the EDIF designs generated by Front-End Design Tool (foo_1.edf) to a validated EDIF (foo_1_designer.edf) before it runs physical synthesis. PALACE performs physical synthesis based on the input design (foo_1_designer.edf), input constraints (foo_1.gcf) and input library (apa.lib). In the end, it outputs an optimized and mapped netlist (foo_2.edf) and corresponding constraint files (foo_2.gcf and foo_2.sdc), which serve as the inputs to Designer. Front-End Tool(s) such as a 3rd Party Synthesis and/or schematic capture tool will generate the EDIF file initially used by PALACE Actel Edition.

- Designer

  Designer is Actel's Placement and Routing tool. It takes PALACE outputs (foo_2.edf, foo_2.gcf and foo_2.sdc) as its inputs, and performs compilation, placement and routing and timing analysis.

3.2 User Requirements

PALACE Actel Edition is a tool that is intended to run in the command line mode only. The user will install PALACE Actel Edition using a Magma supplied installation script. After PALACE Actel Edition has been installed, the user will run their Front-End Tool to generate the EDIF netlist (foo_1.edf) for the design. Additionally, the user may (optionally) enter their design constraints in the appropriate file (foo_1.gcf) for use by Designer. The user will open the Command Line prompt on their system and run PALACE Actel Edition with its constraint files. If PALACE Actel Edition processes the design successfully, it generates an output netlist (foo_2.edf) and constraint files (foo_2.gcf and foo_2.sdc) that will be used by Designer. On the final step, the user opens the Designer GUI and imports the PALACE Actel Edition generated EDIF netlist and constraint files. The user will then use Designer to compile and layout the design.

The execution of PALACE Actel Edition and Designer may be scripted. The user can perform this by writing a script that automatically executes PALACE Actel Edition and Designer in the flow listed above. PALACE release package also provides TCL scripts to automate the process.

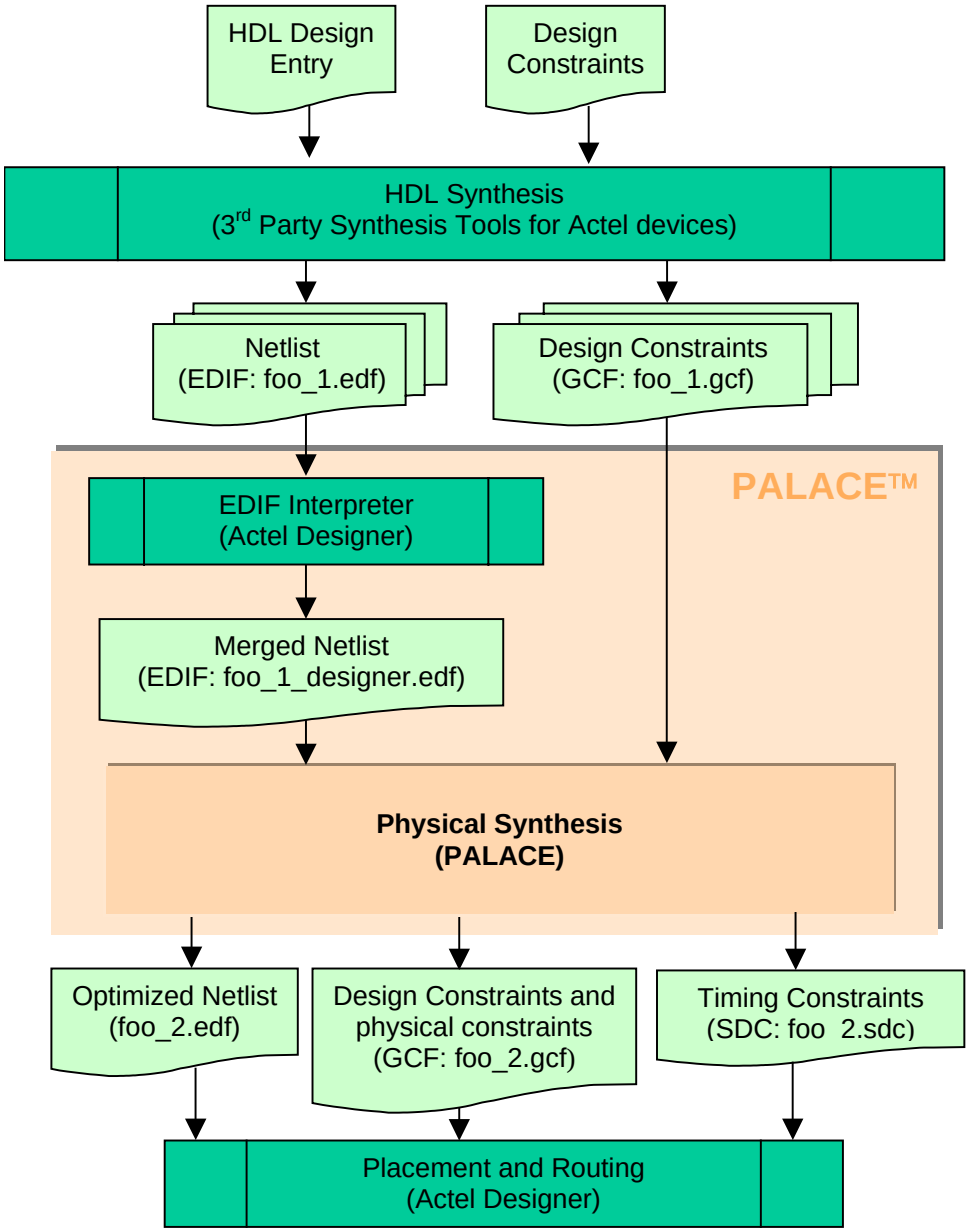

Figure 1 PALACE Actel Edition Customer Design Flow

4 Installation

PALACE is supported on workstations and personal computers running Microsoft Windows platform including Windows NT, Windows 2000, Windows XP Professional Edition, and Sun Solaris platform.

4.1 Microsoft Windows Platform

To install PALACE software:

1. Close all programs before you begin installation.
2. If obtained on a CD, insert PALACE Actel Edition CD in the CD drive and enter the `nt` directory. If the software is downloaded from the web, first unzip the package to an empty temporary directory.
3. From the CD drive (`nt` directory) or the temporary directory that contains PALACE Actel Edition release, run installation program `setup.exe`.

   Follow the instructions on the screen to install the software.

Make sure you reboot your system after installation. After rebooting, you should be able to invoke PALACE anywhere on your Windows platform.

4.2 Sun Solaris Platform

To install PALACE software:

1. If obtained on a CD, insert PALACE CD in the CD drive and enter the `solaris` directory. If the software is downloaded from the web, 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. From CD drive (`solaris` directory) or the temporary directory that contains PALACE release, 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.

4.3 Release Content

The release of PALACE Actel Edition includes:

1. /bin

   Includes PALACE executable, licensing daemon, and supporting files

2. /doc

   Includes PALACE documentation

3. /tutorials

   Includes tutorials to get the user quickly started in using PALACE

3. /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 Actel Edition

FLEXlm Feature Name	Description (device families licensed)
PALACE_ACTEL_APA	ProASIC Plus

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.

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\aplus_license.dat	Stand alone license on Windows located in c:\flexlm\aplus_license.dat
2201@mars	Floating licenses will be checked out from the FLEXlm license server named "mars" (for Windows or Unix)
c:\flexlm\aplus_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 devices families based on the license feature lines.

5.4 Using Floating (network) licenses

In order to use a floating license on a network, the Aplus 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 Aplus 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"
```

4.5 Additional licensing Information

The licensing information contained in this document covers the aspects related to PALACE only, and is not a comprehensive information source for FLEXlm and PALACE licensing. Detailed information can be found in Globetrotter's end user guide (www.globetrotter.com).

6 Using PALACE in Command Mode

PALACE Actel Edition v1.0 Beta 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 Actel Edition currently supports. One can also type in "`palace –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 -family <apa> [-device <part>] -in_design <file>
[-in_design_format <actel_edif|generic_edif>] [in_constraint <file>]
[-out_design <file>] [-out_constraint <file>] [-report <file>] [-
overwrite] [-keep_hierarchy] [-logic_effort <1-4>] [-physical_effort
<1-2>] [-relax_delay <0-50>] [-max_cell_perc <1-100]
```

Where:

`-in_design` *input design file name*

Specify the input design file name.

[`-in_design_format` *input_design_format*]

Specify the input design format:

actel_edif: Actel primitive EDIF format

generic_edif: Generic gate EDIF format

(Default: Actel primitive EDIF)

[`-in_constraint` *input constraint file name*]

Specify the input constraint file name in GCF. If no timing constraint is given to PALACE, the tool will optimize delay for register to register paths, i.e. to minimize clock period.

[`-out_design` *output design file name*]

Specify file name for the output design.

(Default: <input_design_name>_palace.edf)

[`-out_constraint` *output constraint file name*]

Specify file name for the output constraint in GCF.

(Default: <deisgn_name>_palace.gcf and <design_name>_palace.sdc)

[`-report` *report file name*]

Specify file name for PALACE report.

(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.

`-family` *device family*

Specify device family. Choose from the following devices:

apa (for ProAsicPlus)

(Default: apa)

[`-device` *device part*]

Specify the device. It overwrites the device specified in EDIF design input.

(Default: APA600-BG456)

[`-keep_hierarchy`]

Specify to keep the design hierarchy.

(Default: flatten the design hierarchy)

[-logic_effort *logic synthesis effort level*]

Specify logic synthesis effort level (1 to 4). Level 4 is the maximum logic effort level. Logic effort 1 is to minimize area. The following describes what is being done and what is expected for each logic effort level:

-logic_effort 0: This effort level does no logic change at all.

-logic_effort 1: This effort level is primarily targeted for area. It applies various area oriented optimization techniques.

-logic_effort 2: This effort level is primarily targeted for timing. It applies basic timing optimization techniques.

-logic_effort 3: This effort level is primarily targeted for timing. It applies more extensive timing optimization techniques comparing to logic effort 2. This is also the default effort level and one should start with this effort level if the object is timing.

-logic_effort 4: This effort level is primarily targeted for timing. It applies area minimization before applying the various timing optimization techniques.

(Default: 3)

[-physical_effort *physical synthesis effort level*]

Specify physical synthesis effort level (1 to 2). Level 2 is the maximum physical effort level.

(Default: 1)

[-relax_delay *delay relaxation percentage*]

Specify delay relaxation percentage to reduce area. For example, 10 means allowing 10% delay relaxation on the best possible delay PALACE can achieve.

(Default: 0)

[-max_cell_perc *maximum core cell utilization percentage*]

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)

[-help]

Print out the above usage.

For example, the following command,

```
"palace –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, i.e. to minimize clock period.

6.2 Running Designer after PALACE

Upon successful completion, PALACE will generate two files

1. Optimized netlist `foo_palace.edf`
2. GCF file `foo_palace.gcf` that contains all the content in the input constraint file `foo.gcf` and additional (physical) constraints generated by PALACE
3. SDC file `foo_palace.sdc` that contains all timing constraints from the input constraint file `foo.gcf`
4. 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 together with any binary IP will be the inputs to 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 as `palace.rpt` in the working directory. PALACE report consists of five types of information, namely

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

7.1.1 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.

7.1.2 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.

7.1.3 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%
```

7.1.4 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
-------------------------------------------------------------
```

7.1.5 Timing Report

As part of result summary, PALACE reports the estimated timing of final design. The timing is estimated based on the assumption that the implementation tool will be run in a timing-driven mode. It is recommended that users use Designer 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 PALACE release is very helpful to automate the PALACE followed by Designer design flow. It is the most useful to use these scripts to explore multiple runs of this flow to achieve better Quality of Results. In each run, PALACE will be run with a specific option on its logic effort level and physical effort level, and then Designer place and route will be run using a particular seed. Of the multiple runs, the best FMAX will be chosen and reported. The logic effort level and physical effort level to 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 Customize the setting

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, for each design, its design file and constraint files 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 foo, the files foo.edf, foo.gcf, and foo.sdc must be put in directory of 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 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 Execute the script

After all the variables in aa_prun_set.tcl are properly set, the only thing left is to launch TCL shell and invoke "`run_pal_des_file`" command to perform multiple PALACE+Designer runs:

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

Where <ckt-dev-file> is a file containing the information of 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 Commands in Script

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

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

– run_palace_file `< ckt-dev-file >`
- Similar as above
- All circuits are specified in `< ckt-dev-file >`, which is defined in previous section.

– run_designer
- Run all circuits specified in variable CKTS with Designer only.
- The original designs are in **SRC_DIR**
- Use Designer script, $ACTEL_DIR/scripts/sh_iterate.tcl, which is described in page 118, of "Designer Users Guide" V4.6 R1-2003.

– run_designer_file `< ckt-dev-file >`
- Similar as above
- All circuits are specified in `< ckt-dev-file >`, which is defined in 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 iteration 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 as above
- All circuits are specified in `< ckt-dev-file >`, which is defined in previous section.

– run_extdes
- Run Designer extened layout with specified runs and seeds on all circuits in CKTS
- Number of run is specified by NUM_LAYOUT_RUN_AFTER_PALACE
- Seeds are specified in LAYOUT_RUN_SEEDS_AFTER_PALACE
- The original designs are Palace designs in the working directory

– run_extdes_file `< ckt-dev-file >`
- Similar as above
- All circuits are specified in `< ckt-dev-file >`, which is defined in previous section.

– run_pal_extdes
- Run PALACE + Designer extened layout (1 ~ 5 runs) on all circuits in CKTS

- Run PALACE only once. Using the **first** element of PALACE_LOGIC_EFFORT_LIST, PALACE_PHYSICAL_EFFORT_LIST, and PALACE_SEEDS for PALACE
- Number of run is specified by NUM_LAYOUT_RUN_AFTER_PALACE
- Seeds for Designer are specified in LAYOUT_RUN_SEEDS_AFTER_PALACE
- The original designs are in **SRC_DIR**

– run_pal_extdes_file file `< ckt-dev-file >`

- Similar as above
- All circuits are specified in `< ckt-dev-file >`, which is defined in previous section.

– gen_designer_edf

- Given EDIF generated by 3rd synthesis tool, run Designer to comile and export EDIF of Designer

8.4 Obtain 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 are recorded in the file $LOG.log
2. In $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
-------------------------------------------
```

And 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
============================================================
```

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

8.5 Solaris Notes

It is recommended to source aa_prun_sun.tcl instead of aa_prun.tcl.

- aa_prun_sun.tcl is modified from aa_prun.tcl and fully tested on Solaris 5.7 with tclsh8.0 and above version.
- The commands for aa_prun_sun.tcl are same as that of aa_prun.tcl.

9 Known Limitations in this Version

In this release, PALACE supports a sub set of input constraints. Please refer to Table 3 for the constraints that are supported. We welcome your feedbacks on the additional set of constraints you would like to see to be supported in the near term.

"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.

GCF Constraints	Support in PALACE	Notes
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	
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	
Netlist optimization constraints		
dont_optimize	Honor	'dont_optimize buffer' is ignored
optimize	Honor	
set net_region	Honor	
set_max_fanout	Honor	
dont_touch	Honor	
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	

Table 3 – PALACE Actel Edition Constraints Support

10 Getting the Best Results Using PALACE

It is recommended to read User's Manual and follow a simple process to achieve the best possible quality of results:

10.1 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`), run PALACE with:

   `"-in_design foo.edf –in_constraint foo.gcf –logic_effort 3 –physical_effort 1"`

2. 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` among 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:

3. Use `–relax_delay <5-30>` to run PALACE one more time to reduce area.
4. 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 PALACE release is very helpful to automate the PALACE followed by Designer design flow. It is the most useful to use these scripts to explore multiple runs of this flow to achieve better Quality of Results on performance. In each run, PALACE will be run with a specific option on its logic effort level and physical effort level, and then Designer place and route will be run using a particular seed. Of the multiple runs, the best FMAX will be chosen and reported. The logic effort level and physical effort level to 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.

10.2 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"`

10.3 Constraints support

In this release, PALACE supports most of the GCF 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.
- Too tight constraints, especially tight location constraints, may over constrain the tool, hence may hurt PALACE's physical synthesis results.

11 Technical Support

Magma Design Automation, Inc. provides superior technical support to customers through:

Support hotline — (310) 441-9365 x 33, support e-mail — palace_support@magma-da.com.

If you have any questions and comments on PALACE, please do not hesitate to contact Magma Design Automation. Any comments that you may have are highly appreciated.