OpenPiton in Action

Princeton University

http://openpiton.org
ASIC SYNTHESIS AND BACKEND
What’s in the Box?

- **Synthesis**
  - Synopsys Design Compiler
- **Static timing analysis (STA)**
  - Synopsys Primetime
- **Formal equivalence checking (RVS)**
  - Synopsys Formality
- **Place and route (PAR)**
  - Synopsys IC Compiler
- **Layout versus schematic (LVS)**
  - Mentor Graphics Calibre
- **Design rule checking (DRC)**
  - Mentor Graphics Calibre
Why is it Useful?

• Research studies
  – Architecture, EDA, and other HW research

• ASIC tapeout

• Education
Piton ASIC

- 25 tiles
- IBM 32nm SOI
- 36 mm² (6mm x 6mm)
- 1 GHz Target Frequency
- Tested working in silicon!
Synthesis and Backend Flow
What do you need?

• OpenPiton

• Synopsys License
  – Tools and Reference Methodology (RM)

• Mentor Graphics License
  – Calibre (for LVS and DRC only)

• Standard cell library and process development kit
Getting Started

• Download Synopsys-RM

• Patch Synopsys-RM

• Familiarize with directory structure and scripts

• Port to process technology

• Running the flow
Download Synopsys-RM

• Synopsys Solvnet

• See OpenPiton Synthesis and Backend Manual
  – Specify version
  – Specify settings

• Broader support
Patching Synopsys-RM

```
[openpiton]mmckeown@hanoi$ ls
README  README.md  build  docs  piton
[openpiton]mmckeown@hanoi$ synrm_patch -h

OpenPiton generate back-end flow from Synopsys RM scripts and OpenPiton patch. This
script can also be used to generate patches from the current OpenPiton
back-end flow state.

optional arguments:
  -h, --help              show this help message and exit
  -g, --gen               Generate a patch from the current state of OpenPiton
                           back-end scripts (default is to generate back-end flow
                           from Synopsys RM scripts and OpenPiton patch)
  -d DC_RM_PATH, --dc_rm_path DC_RM_PATH
                           Path to directory containing extracted Synopsys Design
                           Compiler RM scripts
  -p PT_RM_PATH, --pt_rm_path PT_RM_PATH
                           Path to directory containing extracted Synopsys
                           Primetime RM scripts
  -i ICC_RM_PATH, --icc_rm_path ICC_RM_PATH
                           Path to directory containing extracted Synopsys IC
                           Compiler RM scripts

[openpiton]mmckeown@hanoi$ synrm_patch --dc_rm_path=../solvnet_scripts/DC-RM I
  -2013.12-SP2/ --pt_rm_path=../solvnet_scripts/PT-RM_I-2013.12/ --icc_rm_path=..
  ./solvnet_scripts/ICC-RM_I-2013.12-SP4/
```
Patching Synopsys-RM

```
[openpiton]mcmckeown@hanoi$ synrm_patch --dc_r mortar=../solvmetscripter/DC-RM_I-2013.12-SP2/ --pt_r mortar=../solvmetscripter/PT-RM I-2013.12/ --icc_r mortar=../solvmetscripter/ICC-RM_I-2013.12-SP4/
synrm_patch: #################################################################
synrm_patch: # OpenPiton Patch Synopsys RM Flow #
synrm_patch: #################################################################
synrm_patch: Checking integrity of Synopsys RM...
synrm_patch: Synopsys RM integrity check passed.
synrm_patch: Copying Synopsys RM files from '../solvmetscripter/DC-RM_I-2013.12-SP2/' to '/tank/mcmckeown/research/projects/piton/openpiton/piton/tools/synopsys/script/'
  synrm_patch: Successfully copied Synopsys RM files from '../solvmetscripter/DC-RM_I-2013.12-SP2/'.
synrm_patch: Patching Synopsys RM '../solvmetscripter/DC-RM_I-2013.12-SP2/' to OpenPiton...
synrm_patch: Successfully patched Synopsys RM '../solvmetscripter/DC-RM_I-2013.12-SP2/' to OpenPiton.
synrm_patch: Copying Synopsys RM files from '../solvmetscripter/PT-RM I-2013.12/ to '/tank/mcmckeown/research/projects/piton/openpiton/piton/tools/synopsys/script/'
  synrm_patch: Successfully copied Synopsys RM files from '../solvmetscripter/PT-RM I-2013.12/'
  synrm_patch: Patching Synopsys RM '../solvmetscripter/PT-RM I-2013.12/' to OpenPiton...
synrm_patch: Successfully patched Synopsys RM '../solvmetscripter/PT-RM I-2013.12/' to OpenPiton.
synrm_patch: Copying Synopsys RM files from '../solvmetscripter/ICC-RM I-2013.12-SP4/' to '/tank/mcmckeown/research/projects/piton/openpiton/piton/tools/synopsys/script/'
  synrm_patch: Successfully copied Synopsys RM files from '../solvmetscripter/ICC-RM I-2013.12-SP4/'
  synrm_patch: Patching Synopsys RM '../solvmetscripter/ICC-RM I-2013.12-SP4/' to OpenPiton...
synrm_patch: Successfully patched Synopsys RM '../solvmetscripter/ICC-RM I-2013.12-SP4/' to OpenPiton.
synrm_patch: Checking integrity of OpenPiton flow...
synrm_patch: OpenPiton integrity checks passed.
synrm_patch: Successfully patched Synopsys RM to OpenPiton flow.
[openpiton]mcmckeown@hanoi$
```
Directory Structure and Scripts

- All scripts written in Tcl
- Two primary locations
  - Module generic scripts
  - Module specific scripts
Porting to a Process Technology

```tcl
# DRC Runset and optional golden GDS file
OPENPITON_CALIBRE_DRC_DECK="CALIBRE_DRC_DECK_HERE (.drc.cal)"
OPENPITON_CALIBRE_GOLDEN_GDS_FILE="" ; # Optional, (.gds) ;

# LVS Runset
OPENPITON_CALIBRE_LVS_DECK="CALIBRE_LVS_DECK_HERE (.lvs.cal)"

# GDS2SP (for doing LVS) and LVS black box cells
# These can be due to internal LVS problems on IP macros,
# std cells that don't get outputted from ICC, etc.
# These should be space separated list of standard cells
# you want black boxed
OPENPITON_GDS2SP_BBOX CELLS=""
OPENPITON_LVS_BBOX CELLS=""

# Put any environment variables needed by DRC and LVS here
# Examples (should be specified by DRC/LVS deck):
# export TECHDIR=
# export LAYOUT_SYSTEM=GDSII
if [ -z ${OPENPITON_DRC RUN+x} ]; then
    # Put any DRC specific environment variables here
    :
fi

if [ -z ${OPENPITON_LVS RUN+x} ]; then
    # Put any LVS specific environment variables here
    :
fi
```
## Porting to a Process Technology

Table 3: Process specific synthesis and backend scripts. Referenced from `${PITON_ROOT}/piton/tools/synopsys/script/`

<table>
<thead>
<tr>
<th>Process Specific File Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>common/env_setup.tcl</td>
<td>Environment variable setup. Gets environment variables that are used throughout the scripts.</td>
</tr>
<tr>
<td>common/process_setup.tcl</td>
<td>Main process specific setup file. Includes standard cell libraries, technology files, layer mapping files, Milkyway libraries, etc.</td>
</tr>
<tr>
<td>common/calibre_env</td>
<td>Calibre process specific setup file. Sets the DRC and LVS decks and any environment variables used by these decks.</td>
</tr>
<tr>
<td>common/design_setup.tcl</td>
<td>Setup variables specific to a design. Includes things like clock frequency, allowed routing metal layers, etc.</td>
</tr>
<tr>
<td>common/floorplan/common_pgn.tcl</td>
<td>Preroutes the power and ground network (PGN) for modules without SRAMs or other IP macros. For modules with SRAMs or other IP macros, a module specific PGN script will be required.</td>
</tr>
<tr>
<td>common/floorplan/core_pgn_mesh.tpl</td>
<td>Templates for PGN meshes. Called from <code>common/floorplan/common_pgn.tcl</code> and module specific PGN scripts.</td>
</tr>
<tr>
<td>common/floorplan/common_post_floorplan.tcl</td>
<td>Post floorplan steps that are common to all modules. This is usually process specific but is also sometimes optional. We used this to place certain cells at a specific density throughout a module after floorplanning is complete, per a requirement from the foundry.</td>
</tr>
<tr>
<td>common/dbl_via_setup.tcl</td>
<td>Double via definition script. This is usually process specific but is also sometimes optional. Can define which vias are used for double via insertion.</td>
</tr>
<tr>
<td>common/pt_eco_drc_buf.tcl</td>
<td>This is also optional, but specifies a list of standard cell buffers that Synopsys PT can use for DRC fixing.</td>
</tr>
<tr>
<td>common/vt_group_setup.tcl</td>
<td>Optional, can group target libraries into threshold voltage groups for reporting. Can generate a report of what percentage of different groups were used, useful for knowing things like what portion of your critical path is made up of the lowest threshold voltage standard cells.</td>
</tr>
</tbody>
</table>
Porting to a Process Technology

• Module specific scripts suggested for review:
  – module_setup.tcl
  – floorplan.tcl
  – <design_name>.constraints.tcl
Running the Flow

Table 4: OpenPiton Synthesis and Back-end Flow Run Commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Flow Step</th>
<th>Tool</th>
<th>Checking Script</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsyn</td>
<td>Synthesis</td>
<td>Synopsys Design Compiler</td>
<td>csyn</td>
</tr>
<tr>
<td>rsta</td>
<td>Static Timing Analysis</td>
<td>Synopsys Primetime</td>
<td>csta</td>
</tr>
<tr>
<td>rrvs</td>
<td>RTL vs. Schematic Equivalence Checking</td>
<td>Synopsys Formality</td>
<td>crvs</td>
</tr>
<tr>
<td>rpar</td>
<td>Place and Route</td>
<td>Synopsys IC Compiler</td>
<td>cpar</td>
</tr>
<tr>
<td>reco</td>
<td>Run ECO</td>
<td>Synopsys IC Compiler</td>
<td>cpar</td>
</tr>
<tr>
<td>merge_gds</td>
<td>Merge GDSII Designs</td>
<td>Synopsys IC Workbench Edit/View Plus</td>
<td>cmerge_gds</td>
</tr>
<tr>
<td>rdrc</td>
<td>Design Rule Checking</td>
<td>Mentor Graphics Calibre</td>
<td>cdrc</td>
</tr>
<tr>
<td>rlvs</td>
<td>Layout vs. Schematic Checking</td>
<td>Mentor Graphics Calibre</td>
<td>clvs</td>
</tr>
<tr>
<td>rftf</td>
<td>Full tool flow</td>
<td>All of the above</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Table 5: OpenPiton Synthesis and Back-end Flow Supported Modules

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Description</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ffu</td>
<td>OpenSPARC T1 core floating-point front-end unit</td>
<td>Small module with one SRAM macro</td>
</tr>
<tr>
<td>sparc</td>
<td>OpenSPARC T1 core</td>
<td>Large module with many SRAM macros</td>
</tr>
<tr>
<td>dynamic_node</td>
<td>OpenPiton on-chip network router</td>
<td>Small module with no IP macros</td>
</tr>
<tr>
<td>tile</td>
<td>OpenPiton tile</td>
<td>Large, hierarchical module with many SRAM macros</td>
</tr>
<tr>
<td>chip</td>
<td>OpenPiton top-level chip</td>
<td>Large, top-level hierarchical module</td>
</tr>
</tbody>
</table>

# Format: # BlockID BlockPath BlockSynMemReq BlockPARMemReq FTFPasses ECO
chip
  chip     | chip     | 48000       | 128000       | 1 | Y
  tile     | chip/tile| 32000       | 128000       | 1 | Y
dynamic_node | chip/tile/dynamic_node | 86000       | 168000       | 1 | N
sparc     | chip/tile/sparc | 64000       | 128000       | 1 | Y
ffu       | chip/tile/sparc/ffu | 8000        | 12000        | 1 | Y
Launch Flow

```
[openpiton]mmckeown@hanoi$ rftf sparc --slurm

Submitted batch job 147185
Submitted batch job 147186
Submitted batch job 147187
Submitted batch job 147188
Submitted batch job 147189
Submitted batch job 147190
Submitted batch job 147191
Submitted batch job 147192
Submitted batch job 147193
[openpiton]mmckeown@hanoi$ squeue

JOBID   NAME     USER   ST  TIME  NODES  CPUS QOS   NODELIST(REASON)
147188  eco1_sparc mmckeown PD  0:00  1   10 short (Dependency)
147192  drc_sparc mmckeown PD  0:00  1   10 short (Dependency)
147193  lvs_sparc mmckeown PD  0:00  1   10 short (Dependency)
147183  stal_dc_sparc mmckeown PD  0:00  1   1 short (Dependency)
147184  rvs1_dc_sparc mmckeown PD  0:00  1   1 short (Dependency)
147186  stal_icc_sparc mmckeown PD  0:00  1   1 short (Dependency)
147187  rvs1_icc_sparc mmckeown PD  0:00  1   1 short (Dependency)
147189  stal_eco_sparc mmckeown PD  0:00  1   1 short (Dependency)
147190  rvs1_eco_sparc mmckeown PD  0:00  1   1 short (Dependency)
147191  mgds_sparc mmckeown PD  0:00  1   1 short (Dependency)
147185  par1_sparc mmckeown PD  0:00  1   10 long (Dependency)
147182  syn1_sparc mmckeown R  0:04  1   6 short hanoi
```
Flow Runtimes

<table>
<thead>
<tr>
<th>Submodule</th>
<th>SYN</th>
<th>PAR</th>
<th>STA</th>
<th>ECO</th>
<th>DRC</th>
<th>LVS</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoC Router</td>
<td>0.32</td>
<td>2.35</td>
<td>0.02</td>
<td>N/A</td>
<td>0.04</td>
<td>0.04</td>
<td>2.80</td>
</tr>
<tr>
<td>L1.5</td>
<td>0.40</td>
<td>15.45</td>
<td>0.05</td>
<td>3.55</td>
<td>0.24</td>
<td>0.12</td>
<td>34.18</td>
</tr>
<tr>
<td>Core</td>
<td>1.16</td>
<td>36.82</td>
<td>0.19</td>
<td>3.51</td>
<td>1.09</td>
<td>0.32</td>
<td>78.04</td>
</tr>
<tr>
<td>Tile</td>
<td>1.11</td>
<td>22.95</td>
<td>0.12</td>
<td>2.69</td>
<td>2.10</td>
<td>0.49</td>
<td>55.89</td>
</tr>
<tr>
<td>Chip</td>
<td>4.20</td>
<td>79.19</td>
<td>0.67</td>
<td>11.36</td>
<td>9.04</td>
<td>0.93</td>
<td>104.91</td>
</tr>
</tbody>
</table>

Table 5: Time durations (in hours) of selected synthesis and back-end stages for selected submodules. If any stage executes more than once (STA or for multi-pass flow), the maximum duration is shown.

<table>
<thead>
<tr>
<th>Submodule</th>
<th>SYN</th>
<th>PAR</th>
<th>STA</th>
<th>ECO</th>
<th>DRC</th>
<th>LVS</th>
<th>Peak</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoC Router</td>
<td>5.38</td>
<td>3.55</td>
<td>0.57</td>
<td>N/A</td>
<td>1.40</td>
<td>1.33</td>
<td>5.38</td>
</tr>
<tr>
<td>L1.5</td>
<td>19.91</td>
<td>5.97</td>
<td>1.43</td>
<td>5.00</td>
<td>1.73</td>
<td>1.46</td>
<td>19.91</td>
</tr>
<tr>
<td>Core</td>
<td>27.19</td>
<td>7.99</td>
<td>1.96</td>
<td>15.05</td>
<td>2.54</td>
<td>4.03</td>
<td>27.19</td>
</tr>
<tr>
<td>Tile</td>
<td>28.04</td>
<td>7.71</td>
<td>1.90</td>
<td>13.62</td>
<td>2.48</td>
<td>8.84</td>
<td>28.04</td>
</tr>
<tr>
<td>Chip</td>
<td>8.37</td>
<td>64.54</td>
<td>1.34</td>
<td>64.65</td>
<td>8.54</td>
<td>&gt;41</td>
<td>64.65</td>
</tr>
</tbody>
</table>

Table 6: Peak memory usage (in GByte) of selected synthesis and back-end stages for selected submodules. If any stage executes more than once (STA or for multi-pass flow), the maximum peak usage is shown.
# Flow Reports

Table 6: OpenPiton Synthesis and Back-end Flow Results Locations. Referenced from module specific synopsys directory.

<table>
<thead>
<tr>
<th>Command</th>
<th>Results Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>rsyn</td>
<td>reports/dc_shell, reports/dc_shell_pass*</td>
</tr>
<tr>
<td>rsta</td>
<td>reports/pt_shell, reports/pt_shell_dc_pass*, reports/pt_shell_icc_pass*</td>
</tr>
<tr>
<td>rrvs</td>
<td>reports/fm_shell, reports/fm_shell_dc_pass*, reports/fm_shell_icc_pass*</td>
</tr>
<tr>
<td>rpar</td>
<td>reports/icc_shell, reports/icc_shell_pass*</td>
</tr>
<tr>
<td>reco</td>
<td>reports/eco_shell, reports/echo_shell_pass*</td>
</tr>
<tr>
<td>merge_gds</td>
<td>results/</td>
</tr>
<tr>
<td>rdrc</td>
<td>reports/&lt;design_name&gt;.drc.summary (bottom of file)</td>
</tr>
<tr>
<td>rlvs</td>
<td>reports/&lt;design_name&gt;.lvs.report (second result)</td>
</tr>
</tbody>
</table>
Flow Outputs

```
[synopsys]mmckeown@hanoi$ pwd
/tank/mmckeown/research/projects/piton/piton_master/  [synopsys]mmckeown@hanoi$ ls
WORK
  alib-52
  command.log
  dc_shell_pass1.log
  drcRun
  drc_sparc
  ecol_sparc
  eco_shell_pass1.log
  filenames_16101_D20160412.log
  filenames_17779_D20160412.log
  filenames_25593_D20160412.log
  filenames_27689_D20160413.log
  filenames_30063_D20160413.log
  filenames_31848_D20160413.log
  filenames_38009_D20160413.log
  filenames_45017_D20160413.log
  filenames_58085_D20160414.log
  filenames_58451_D20160414.log
  filenames_58476_D20160414.log
  filenames_59231_D20160414.log
  filenames_63773_D20160414.log
  filenames_64960_D20160413.log
  fm_shell_command.log
  fm_shell_dc_pass1.log
  fm_shell_eco_pass1.log

  formality.log
  formality_svf
  icc_shell_pass1.log
  legalizer_debug_plots
  logs_eco_pass1
  logs_icc_pass1
  lvssRun
  lvs_sparc
  merge_gds.log
  mgds_sparc
  net.acts
  parl_sparc
  parasitics_command.log
  pna_output
  pt_shell_command.log
  pt_shell_dc_pass1.log
  pt_shell_eco_pass1.log
  pt_shell_icc_pass1.log
  reports
  results
  rvs1_dc_sparc
  rvs1_eco_sparc
  rvs1_icc_sparc
  rvs_status_dc_pass1.log
  syn1_sparc
  syn_status_pass1.log
```

```
Flow Outputs
Opening the Design

[synopsys]mmckeown@hanoi$ icc_shell -gui
Opening the Design
Opening the Design
Opening the Design
Opening the Design
Opening the Design