ECE 4150/6250 – Cadence Tutorial: Using Cadence Innovus Digital Implementation System – Automatic Layout Place & Route Tool

Objectives:

- Automatically generate layout for a previously synthesized Verilog netlist using Cadence Digital Implementation System
- Import generated Layout into Cadence Virtuoso
- Import synthesized Verilog code into Cadence Virtuoso
- Verify our Routed design matches our schematic design

Assumptions:

- Student has a synthesized ripple carry adder from Synopsys w/ a scan-chain inserted

Introduction:

The ASIC design flow is as follows:

Specification
▼
RTL Coding and Simulation
▼
Logic Synthesis
▼
Optimization
▼
Gate Level Simulation
▼
Static Timing Analysis
▼
Place and Route
▼
Static Timing Analysis
▼
Preliminary Netlist Handoff

In this lab, we are at these stages in the ASIC Flow

In this lab we will use a Cadence Product Called: Innovus Digital Implementation System. This product is capable of taking a SYNTHESIZED Verilog netlist from Synopsys and generating a layout from it. This process is known as place & route and you can see that it is a standard part of the ASIC Design Flow listed above. Many other tools exist in the industry to accomplish the same task.

A Static Timing Analysis can be done while in the place & route tool (we will not do this in this lab), which is the next step in the ASIC Design Flow. Afterwards the placed & routed layout can be opened in Cadence Virtuoso. The synthesized Verilog code can then be used to generate a schematic within Cadence Virtuoso.

We will place & route the ripple-carry with the scan-cells inserted in Synopsys, that we created in lab5. Afterwards you can use this technique to place & route any synthesized Verilog code you may have from Synopsys. After placing and routing, we will perform verification, via LVS, against an imported Verilog schematic.

Part I: File Setup for This Lab (TA will move to the Part II in 30 mins)

Note: You can change the suggested "ece4150_6250" folder name to your own one in the labs thereafter. Any time you wish to generate a layout from synthesized Verilog code, create a directory in your ece4150_6250 folder to house all of the files that will be created during the layout process.

1. You‟ll need to have a local copy of the NCSU design kit for the Cadence Innovus Product to work correctly. Follow the following steps to create a local copy of the design kit:

- *Note: You will need 300 MB of space in order to store the NCSU design kit.*
- Login to a workstation, open up a terminal window, type the following commands:

```
mkdir ~/cadence/design_kits
cp –R $CDK_DIR ~/cadence/design_kits
```

- You should see a few permission denied errors, you may ignore these for the purposes of this lab.

2. Edit your .profile and update the location to your CDK_DIR as follows:

```
gedit ~/.bashrc
```

Add this line to the very end of the file:

```
export CDK_DIR=~/cadence/design_kits/ncsu-cdk-1.6.0.beta
```

- **Save, then log out of the workstation and log back in**
- **In a terminal window type the following command and ensure it is working:**

```
echo $CDK_DIR
```

It must return:

```
/home/ead/Netid/cadence/design_kits/ncsu-cdk-1.6.0.beta
```

(notice your name in place of the authors)

If it does not the above location, repeat step 2, ensure you've logged out and back in. If you are using VNC, kill the VNC session, restart it, and log in once again.

Note: Steps 1-2 only need to be performed once per taking ECE 4150/6250.

3. Create a directory to hold files needed for the Cadence Innovus Tool:

```
mkdir ~/cadence/innovus
```

4. Copy abstract and timing files from the OSU Standard Cell Library into your “innovus” directory:

```
cp /apps/design_kits/ncsu-cdk-1.6.0.beta/lib/NCSU_TechLib_ami06/ami05_techlib_oa.lef
   ~/cadence/innovus

cp /apps/design_kits/osu_stdcells_v2p7/cadence/lib/ami05_oa/lib/osu05_stdcells_oa.lef
   ~/cadence/innovus

cp /apps/design_kits/osu_stdcells_v2p7/cadence/lib/ami05/lib/osu05_stdcells.tlf
   ~/cadence/innovus

cp /apps/design_kits/osu_stdcells_v2p7/cadence/lib/ami05/lib/osu05_stdcells.lef
   ~/cadence/innovus
```

5. Copy abstract and timing files for the GWU Scan Cell Library into your “innovus” directory:

```
cp /apps/design_kits/osu_stdcells_v2p7_expanded/cadence/lib/ami05_oa/lib/osu05_stdcells_expanded.lef
~/cadence/innovus

cp /apps/design_kits/osu_stdcells_v2p7_expanded/cadence/lib/ami05_oa/lib/osu05_stdcells_expanded.tlf
~/cadence/innocus
```

6. Copy the files you created in lab 5 (the ripple-carry lab) to today‟s lab:

If you have completed lab 5, copy it to be today’s lab7:

```
cp –R ~/ece128/lab5 ~/ece4150_6250/lab7
```

If you have not completed lab 5, copy it from the reference location:

(DO NOT RUN THIS COMMAND IF YOU RAN THE LAST COMMAND)

```
cp –R /home/seas/vlsi/course_ece128/lab_files/lab7 ~/ece4150_6250/lab7
```

7. To place & route future designs, you will not need to copy any files. You can simply work out of your ~/ece4150_6250/labX directory.

8. Create a Default.view file under your cadence directory:
Type command: `gedit` to open the editor.
Type the following commands:

```
# Version:1.0 MMMC View Definition File
# Do Not Remove Above Line
create_library_set -name CommonTiming -timing {innovus/osu05_stdcells_expanded.tlf innovus/
osu05_stdcells.tlf}
create_constraint_mode -name TimingConstraints -sdc_files
{../ece4150_6250/lab7/src/ripplecarry4_clk_scan.sdc} create_delay_corner -name corner_min -library_set
{CommonTiming}
create_analysis_view -name view_hold -constraint_mode {TimingConstraints} -delay_corner {corner_min}
set_analysis_view -setup {view_hold} -hold {view_hold}
```

and save it as Default.view under your cadence directory.
This Defalut.view file will be used to set timing constraints, delay corner and analysis mode.

Part II: Setup Cadence Libraries (TA will move to the next Part in 20 mins)

1. Change to your cadence working directory

```
cd ~/cadence
```

2. Start Cadence Virtuoso by typing the following commands:

```
virtuoso &
```

3. Add OSU Standard Cell (AMI .5 technology) Library to Cadence

- From the CIW Window (the small window at the bottom), select: Tools → Library Path Editor
- On the bottom line, create a new entry:
    - Library Name: `OSU_stdcells_ami05`
    - Path:
`/apps/design_kits/osu_stdcells_v2p7/cadence/lib/ami05_oa/OSU_stdcells_ami05`
- Create a 2nd entry on the bottom as follows (this is the GWU scan cell library)
    - Library Name: `OSU_stdcells_expanded_ami05`
    - Path:
`/apps/design_kits/osu_stdcells_v2p7_expanded/cadence/lib/ami05_oa/OSU_stdcells_expanded_ami05`
- From the Library Path Editor menu, choose: **File → Save As** and say make sure to check “lib.defs”. Save the changes, if prompted to overwrite any files, click YES to save the changes
- **Note1: The library names must be typed EXACTLY as above, double check this before continuing. An incorrect entry will appear to work, but later problems will surface**
- **Note2: Once these 2 libraries are added to your Library List, you never need to add it again!**

4. Update your CDS.lib and LIB.defs files:

- Anytime you create a new library in cadence, you must do the following so that the other product we will use: Cadence Innovus, will recognize the new library:
    - From the CIW Window (the small window at the bottom), select: Tools → Library Path Editor
    - From the Library Path Editor menu, choose: **File → Save As** and say make sure to check “lib.defs”. Save the changes, if prompted to overwrite any files, click YES to save the changes
    - This must be done every time you add a new library, to ensure the lib.defs file is in sync with the cds.lib file

5. Exit Cadence Virtuoso

Part III: Place and Route the Ripple Carry Adder w/ Scan Cells

Overview of Part III:

We will use Cadence Innovus to do the following:
1. Load in a synthesized Verilog netlist and constraints of the RippleCarry Adder we generated from Synopsys" Design Compiler
2. Load in AMI .5 standard cell libraries
3. ***Floorplan*** a Chip (specify chip boundary, and power grid setup)
4. Automatically route a power grid for the chip.
5. Automatically ***place*** the AMI .5 standard cells on the chip"s floorplan
6. Automatically ***route*** wires between the standard cells
7. Export the placed & routed chip to Cadence standard format: Open Access

Note: Before ever attempting to "place & route" Verilog code, you must ensure that all re-simulation of the synthesized code is complete and that it matches your design goals.

Part III, Section 1 and 2: Startup Cadence Innovus & Import Design:
(TA will move to the next section in 10 mins)

1. Change to your working directory

   `cd ~/cadence`

2. Start Cadence Innovus by typing the following command:

   `innovus` *(note: do NOT put an "&" after this command)*

3. Importing Synthesized Code

   From the menu, choose: **File->Import Design**

   Under the ***Netlist*** tab, set the following options:

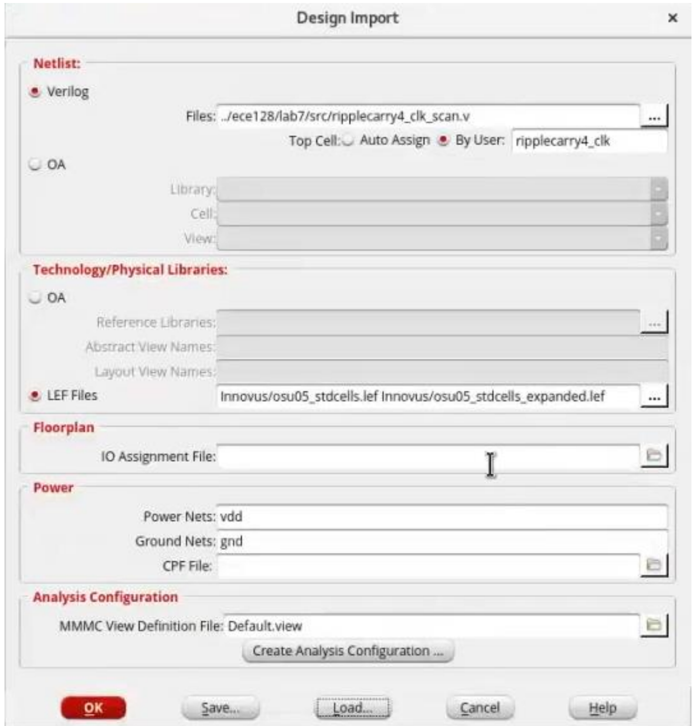
   - **Verilog ->Files**: `../ece4150_6250/lab7/src/ripplecarry4_clk_scan.v`
     - **->Top Cell->By User**: `ripplecarry4_clk`

   Under the ***Technology/Physical Libraries*** tab, set the following options:
   - **LEF Files**: `. innovus/osu05_stdcells.lef`
     `. innovus/osu05_stdcells_expanded.lef`
     (note: *`put a space between these three entries`*

   Under the ***Power*** tab, set the following options:
   - **Power->Power Nets**: `vdd`
   - **Power->Ground Nets**: `gnd`

   Under the ***Analysis Configuration*** tab, set the following options:
   - **MMMC View Definition** : Defalut.view

   Note: the Default.view file is just created by yourself.



Now, press **Save**, call the file: “~/ece4150_6250/lab7/innovus.globals”

Note: You can re-load this file any time, for future designs, you’ll only need to update the Netlist & Timing Constraint Fields

Now, press **OK** to import your code

For your own information:

- The **.lef** file contains information about the layer specifications for the technology implemented by the library (in this case AMI .5). Specs like default sizes, minimum widths, and minimum spacing dimensions. It also contains information about the layout of the standard cell library, for routing purposes.

- The **.tlf** file constraints information on the timing and power parameters of the cell library. It is used to determine delays of I/O ports and interconnects of the final design.

- The **.sdc** file contains information about timing constraints on nets, derived from the synthesis constraints. It is used to help the routing tool meet timing constraints.

- The **.conf** file contains the import settings for your code, in the event that you want to quickly reload a design. You can select **Load** from the import design menu and quickly reload your import settings.

Part III: Section 3: Floor Plan the Chip: (TA will move to the next section in 10 mins)

Floor planning is the process of arranging the overall chip. Setting up the overall area, where main power rails will travel, arranging clock trees, etc. We will perform this process in the next few steps.

4. Setup the chip area and boundaries

- From the menu, choose: **Floorplan → Specify Floorplan**
  - In the "**Core Margins by**" section, change "Core to" *left*, *right*, *top* and *bottom* to: **30**
  - This creates a 30 micron space on all sides of the chip that we'll use later for I/O ports
  - **Dimension**: Width:**200** Height:**120**
  - Leave all other defaults and press OK
- You will now see the overall boundary of the chip in the main window

- For your own information, there are many options here which affect how your core is floorplanned
  - On the **basic** menu the following options are available:
    - Aspect ration: Determines the height/width ratio of the design.
    - Core Utilization: This lets the tool know how densely to pack the cells together, to leave room during optimization phases.
    - Core Margins: This sets the margins for placing power and ground rings,
  - On the **advanced** menu, there are some interesting options for Standard Cell Placement**:**
    - Double Back Rows – Rows can either be placed with their power rails back to back, or not.
    - Row Spacing – Spacing between rows, to allow for routing channels.
- For your final project, you may wish to experiment with these options to balance speed, area, etc.

5. Save your design (frequently)

- Innovus does not have an „un-do‟ option, it is buggy and tends to crash when you make a mistake
- It is good practice to save your design often, but use different file name at each step
- Save your floorplanned design as follows: **File -> Save Design:** Data type: **Innovus**
- **Use filename: ~/ece4150_6250/lab7/innovus_fplan.enc**
- You can reload your design at any time, by going to: **File -> Restore Design**: Data type: **Innovus**

Part III: Section 4: Power Planning the Chip: (TA will move to the next section in 10 mins)

6. Add Main Power Rings Around Chip
- From the menu, choose: **Power → Power Planning → Add Rings**
  - Net(s): **gnd** and **vdd**
  - **Ring Configuration**:
    - Set the **width** of each metal layer to: **9.9**
    - Set the **spacing** of each metal layer to: **1.8**
    - Click on: **Center in channel**
    - *This will center the wires in the boundary you defined in the floorplanning step*
  - Leave all other defaults, and press OK

  - You should now see a blue and red 'ring' around the perimeter of the chip (width 9.9)
  - These are layers M1 and M2 which will distribute VDD and GND around the chip
  - It is possible to create larger power rings, as long as the width of your wires is divisible by 0.3 (For AMI 0.5 um processes)
  - *Note: the wider the metal, the more current that wire can carry to your chip, pwr hungry chips will require wider metal power rings*

7. Add Vertical Power Rails Across You Chip
- You may skip this step for the small ripplecarry adder, but you must do it for your large CPU project
- From the menu, choose: **Power → Power Planning → Add Stripes**
  - **Layer:** Metal2
  - **Direction:** Vertical
  - **Width:** 4.8
  - **Spacing:** 1.8
  - **Set Pattern->Set-to-set distance:** 99 *(must divisible by 0.3 for AMI 0.5)*
  - **Advanced Tab: Snap wire center to routing grid:** *choose*: GRID
- Leave all other defaults, and press OK

- You will now see vertical strips of Metal 2 across your chip, every 99 microns. These will supply a current hungry chip with a more direct connection to VDD and GND
- Note: We use M2 for the vertical stripes, because our standard cells use M1 for VDD and GND

8. Add Horizontal Power Rails Across You Chip
- This step is **NOT** optional!
- From the menu, choose: **Route → Special Route**
- Leave all defaults, and press OK

- You will now see rows of Metal 1 across your chip, spaced 30um on center.
- This distance is the default because the AMI .5 standard cells from OSU, have a standard “pitch” (distance from VDD to GND) of 30um for each of their cells.
- If you placed vertical strips of metal in step 4, notice the horizontal and vertical strips are via‟ed together where they intersect.
- *Once the 'standard cells' are placed, you will see that this grid will connect all the VDD and GNDs for each cell.*

9. Save your design (frequently)
- **File -> Save Design**: Data type: **Innovus**
- Use filename: ~/ece4150_6250/lab7/_**pwr.enc** *(notice a different name is used)*

Part III: Section 5 and 6: Place the Standard Cells and Route the Connections Between Them:
(TA will move to the next section in 10 mins)

10. Enter the following commands in **Innovus terminal** to **Place** the standard cells

```
specifyScanChain -start test_si -stop test_so my_chain
setPlaceMode -place_global_place_io_pins true
setPlaceMode -fp false
place_design -noPrePlaceOupt
```

Click on the 'physical view" icon in the upper right part of the editor window. You will now see an 'outline' or what is called an 'abstract view' of the perimeters of the standard cells.

Save your design: File -> Save Design: Data type: Innovus

File name: ~/ece4150_6250/lab7/**Innovus_placed.enc**

11. **Route** the wires (connections you defined in your Verilog code) between the standard cells
From the menu, choose: **Route** → **NanoRoute** → **Route**
Leave all defaults, and press OK

All of the I/O ports on the standard cells will be connected together to match how the 'synthesized' Verilog code dictated

Save your design: Design -> Save Design As -> SoCE: ~/ece4150_6250/lab7/**innovus_routed.enc**

12. **Fill** empty space between cells
From the menu, choose: **Place** → **Physical Cells**→ **Add Filler**
Press the top ***'select'*** button
Select **FILL** in the right-hand pane, press the **ADD** button, then close the window
Leave all defaults, and press OK

A 'filler' cell is inserted between any blank space throughout your chip
It is also possible to, at this stage, to insert Spare cells into your chip, instead of just using FILL cells. This tutorial will not cover how to do that. That is done under **Place -> Place Spare Cells**.

Save your design: File -> Save Design: Data type: Innovus

File name: ~/ece4150_6250/lab7/**Innovus_filled.enc**

13. View the final chip
The place and route sequence is complete. You can zoom around the chip (the commands and short-cut keys are similar to cadence). Notice how the VDD and GND wires connect to the ring and how the vias are used.

Part III: Section 7: Exporting the Design:
(TA will move to the next section in 10 mins)

We must export the design from Cadence Innovus, so that we can open it up in Cadence Virtuoso

14. Export as Open Access Format Design

- From the ternimal, Type the following commands:

```
set init_lef_file ""
set init_oa_ref_lib {OSU_stdcells_ami05 OSU_stdcells_expanded_ami05}
set init_oa_default_rule LEFDefaultRoutSpec
set init_layout_view layout
set init_abstract_view abstract
```

- Go back to Innovus window to create OA library
  From the menu, choose: **File → Create OA Library**
  OpenAccess Library: **ripplecarry4_clk**
  Select **Reference existing technology libraries**
  For Technology Libraries**,** add **Rapid_MSOA_PDK**

- Save Design
  Data Type: **OA**
  Library: **Lab7**
  Cell: **ripplecarry4_clk**
  View: **Layout**

- You may see warnings, ignore them
- *Note: We could have exported this design as a GDS file, but by using the Open Access Format, we could 'reopen' this design later (after we do some work on it in cadence)*
- *Note: We have to download and add **Rapid_MSOA_PDK** in Virtuoso (in library path editor)*

15. You may now exit from Cadence Innovus

The rest of this page is intentionally left blank.

Part VI: Open Layout and Generate Schematics in Cadence Virtuoso

(TA will move to the next section in 20 mins)

Startup Cadence Virtuoso & View Layout:

1. Change to your working directory

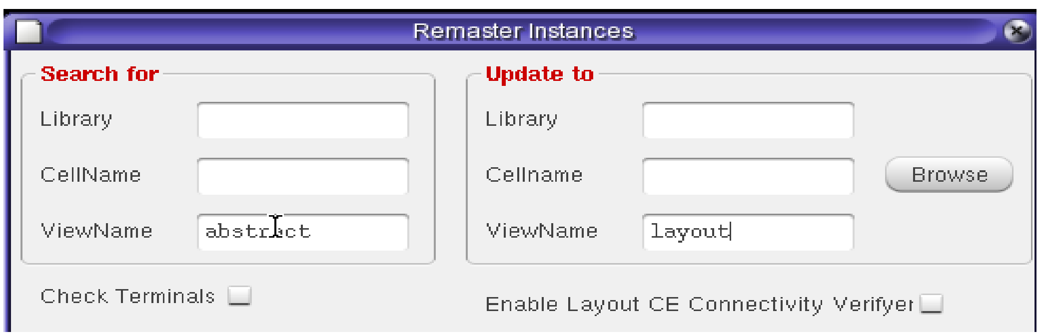
```
cd ~/cadence
```

2. Start Cadence Virtuoso:

```
virtuoso &
```

3. Open Layout
   - From the Library Manager, open up the layout view for the **ripplecarry4_clk** cell
   - You should see the placed & routed layout for the **ripplecarry4_clk** circuit

4. Re-master Cells
   - From the layout menu choose: **Tools → Remaster Instances**
   - Fill in the form as follows:



   - Press OK
   - Press <shift> f to see the complete layout with the standard cells placed & routed

5. DRC and EXTRACT the Design
   - **Choose: Verify->DRC**
     Rules File: **/apps/design_kits/ncsu-cdk-1.6.0.beta/lib/NCSU_TechLib_ami06/divaDRC.rul**
     Uncheck **Rules Library**
     Ensure that there are 0 errors. If you see errors, you have made a mistake while in Silicon Innovus

   - **Choose: Verify->Extract, Add parasitic capacitance, press OK** *(ensure no errors have occurred)*
     Rules File: **/apps/design_kits/ncsu-cdk-1.6.0.beta/lib/NCSU_TechLib_ami06/divaEXT.rul**
     Uncheck **Rules Library**
     Zoom in on the design, verify that things look as you might expect for a ripple carry adder circuit
   - Close and Save the layout view

6. Generate the schematic view
   - From the CIW Windows (the small window below the library manager), type this command "`impHdlDisplay(impHdlOptionsFormMain)`" to open the window for setting parameters.
   - Set the following fields:
     - Target Library Name: **ripplecarry4_clk**
     - Reference Libraries (delete what is there, type): **OSU_stdcells_ami05 OSU_stdcells_expanded_ami05**

- Verilog Files To Import: **../ece4150_6250/lab7/src/ripplecarry4_clk_scan.v**
- Import Structural modules As: **schematic and functional**
- Leave all other fields to their defaults and press **OK**

- Go back to Library Manager, choose: **View** → **Refresh**
- View the log file window, you may see warnings like:
  - “Unable to find the Verilog Module AND2X2. Therefore using library OSU_stdcells_ami05...”
  - This is OK, because we didn't 'create' a module called, AND2X2, we want it to you use the standard cell library's reference module
  - Ensure that no errors exist and press OK
- From the Library Manager, you can open up the ripplecarry4_clk → **schematic view** and notice it will be exactly as you defined it in the Verilog code. You will even notice the 'sub-modules' are there!
- Also, you can open up ripplecarry4_clk → **functional view** and you will see the imported synthesized Verilog that the schematic and your layout are based off of.

7. Run an LVS

- Open up the **extracted view** for the ripple carry adder and from the menu choose: Verify → LVS
- Run an LVS check between the **schematic** view and the **extracted** view (Both from ripplecarry4_clk)
- Rules File**: /apps/design_kits/ncsu-cdk-1.6.0.beta/lib/NCSU_TechLib_ami06/divaLVS.rul**
- Uncheck Rules Library
- Run

8. To show that you’ve completed this section of the lab, fill in the answers below regarding the ripple carry adder, print this page, and hand it in to your GTA before next lab:Run an LVS

References

- Virginia Tech ECE Department Web Publication: http://www.vtvt.ece.vt.edu/vlsidesign/tutorialCadence_socEncounter.php
- George Mason University ECE Department Web Publication: http://www.egr.msu.edu/classes/ece410/mason/files/Tutorial%20PnR-sp08.pdf
- Oklahoma State University ECE Department Web Publication: http://avatar.ecen.okstate.edu/projects/scells/flow/encounter_gui/index.html
- Joannes Grad Open Access Tutorial: http://www.chiptalk.org/modules/wfsection/article.php?articleid=12
- Digital VLSI Chip Design with Cadence and Synopsys CAD Tools, 2010, Erik Brunvand. Person Education Inc.