

# Integration Tutorial

(Last updated: Oct. 22, 2009)

## Introduction

This tutorial will get you familiarized with the design flow of integrating your design block (made by APR tools or by custom) with the memory module and I/O pads. All files you need would be given to you for the purposes of this tutorial, but you will have to supply or modify the files in order to run your own modules in the future.

We will use the multiplier from Tutorial 2 and a 16 x 512 memory for this tutorial. After finishing all the steps, you will get a layout similar to Figure 1 below. Both DRC and LVS should be clean, which will also be the case when you turn in your final project.

**(Please make sure that you have finished the previous tutorial and have a cell called “mult” before you start this tutorial. It should have a layout view, a schematic view and a symbol view, and also it should be the only cell with that name in all of your libraries.)**

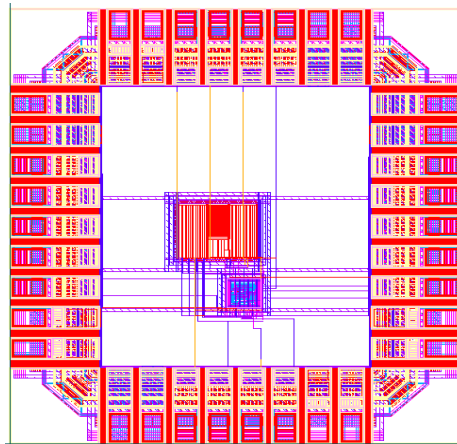


Figure 1. Final Layout

## Design Flow

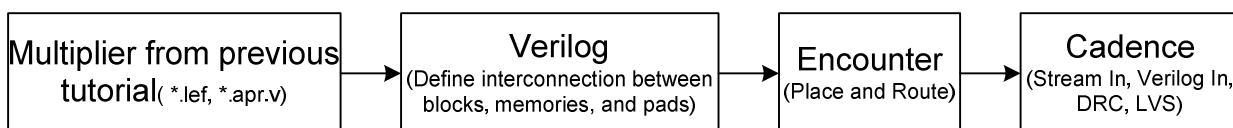


Figure 2. Design Flow

## File Preparation

### 1. Integration tutorial Folder

The integration folder is located in the following directory:

```
/afs/umich.edu/class/eecs427/ibm13/integration_tutorial
```

Use the commands below to create a new workspace and copy the integration folder over to your personal eeecs427 directory.

```
% cd /afs/umich.edu/class/eecs427/ibm13/ f09/<unique name>
% makedirs eeecs427_integration
% cd eeecs427_integration
% cp -r /afs/umich.edu/class/eecs427/ibm13/integration_tutorial .
```

### 2. Include your design block (multiplier from Tutorial 3)

You would need a verilog netlist file (\*.apr.v or \*.v) and an abstract view (\*.lef) for your design block. These two files are already provided in this tutorial.

```
./integration_tutorial/verilog/mult.apr.v
./integration_tutorial/lef/mult.lef
```

These two files are included in the *Encounter* configuration file (*./integration\_tutorial/eeecs427\_top.conf*) so that *Encounter* can do the block placement and signal connections for us. In the future, you will have to provide verilog modules (for connectivity) and LEF files (to guide routing) for each block that you want to place and connect. Then, the *Encounter* configuration file has to be modified to include these files.

### 3. Verilog files for interconnections between blocks and pads

You need to tell *Encounter* how to place your pads (\*.raw), and also the interconnections between blocks and pads (\*.v). For simplification, we provide these two files in this tutorial.

```
./integration_tutorial/verilog/eeecs427_top.raw    (define the pad placement)
./integration_tutorial/verilog/eeecs427_core.v    (define the interconnections)
```

In the future, you will have to create these two files in order to run your own modules.

Next, we use those two files to generate other verilog files we need. Change directories into the "verilog" directory ("**cd integration\_tutorial/verilog**") and type the following:

```
% make gen_padv    (generate pads warp file, eeecs427_pads.v)
% make gen_topv    (generate top level warp file, eeecs427_top.v)
% make gen_io      (generate io file, eeecs427_top.save.io)
```

Now you are ready for *Encounter*.

**Running *Encounter*:**

This step is just like what we did in Tutorial 3. Go to `./integration_tutorial/encounter/`, and all of the files are included for you.

`eeecs427_top.conf`                    (*Encounter configuration*)

`eeecs427_top.tcl`                    (*Encounter TCL script*)

Type

`% make eeecs427_top`                (*runs Encounter to do the APR for you*)

You should, at the very least, look up each command in the `eeecs427_top.tcl` script and understand them.

Next, we will import the routed top-level block into *ICFB*.

## Importing Layout and Netlist (Cadence)

### 1. Open icfb

```
% cd /afs/umich.edu/class/eecs427/ibm13/f09/<unique name>/eecs427_integration
```

```
% icfb &
```

### 2. Create a new lib

Use *IBM\_PDK* to create a new lib “*eecs427\_top*”. Don't forget to attach it to the *cmr8sf* technology library.

### 3. Import stream (\*.gds2)

On the CIW (Command Window), Hit **File**→**Import**→**Stream**.

Fill in the form as shown below (also user-defined data, and options).

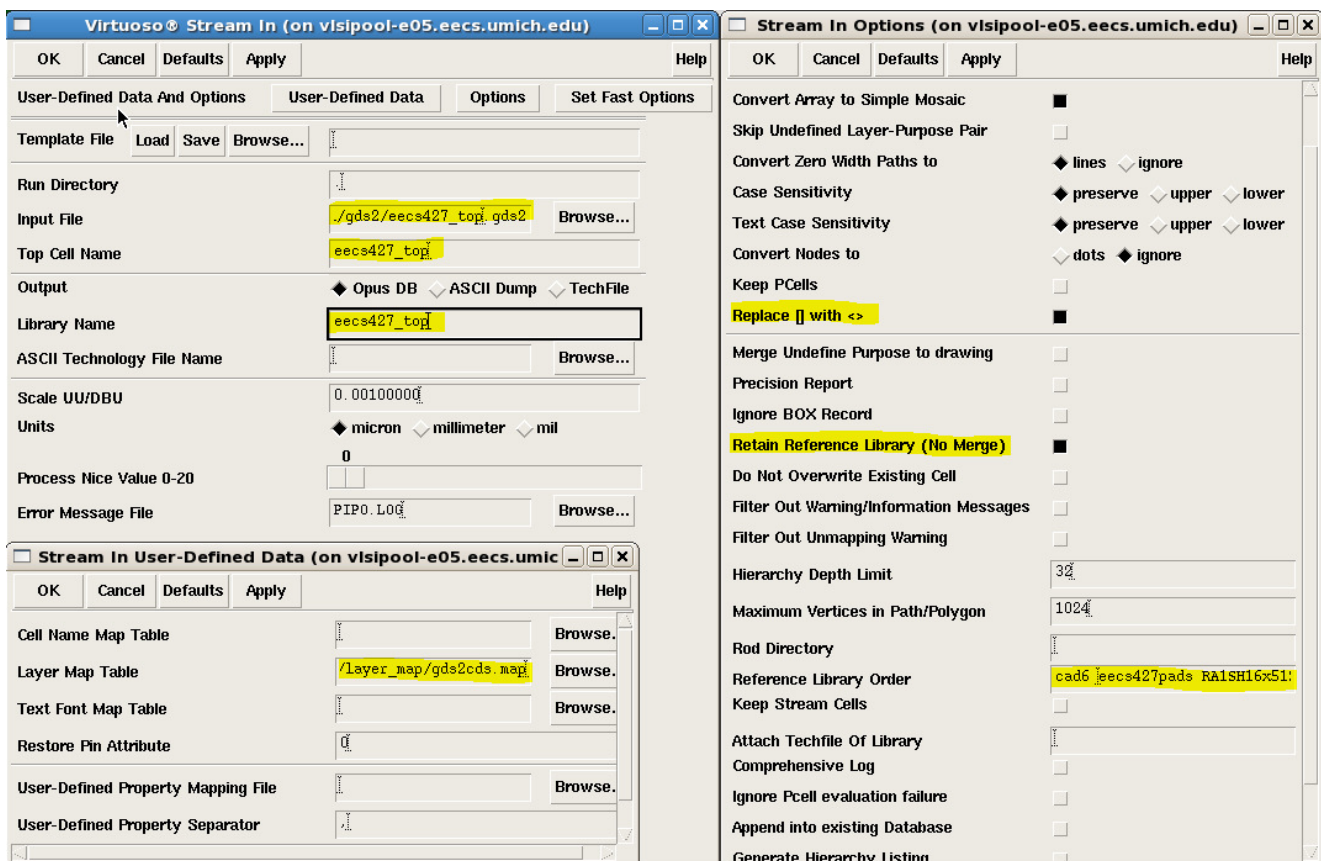


Figure 3. Stream In

Input file:

```
/afs/umich.edu/class/eecs427/f09/<unique name>/integration_tutorial/gds2/eecs427_top.gds2
```

Layer Map Table:

```
/afs/umich.edu/class/eecs427/ibm13/parts/artisan_cells/layer_map/gds2cds.map
```

Click “OK”.

#### 4. Import Netlist (create schematic)

To start the import process, go to the “CIW” Window and select **File**→**Import**→**Verilog**, and fill the form as shown below in Figure 4.

- Target Library Name = “eecs427\_top”
- Reference Libraries = “RA1SH16x512 eeecs427pads cad6 basic cmrf8sf” (libraries where modules are located)
- Verilog Files To Import = (any Verilog files needed for import)
- Power Net Name, Ground Net Name, Global Net Names = (shown below)

The screenshot shows the 'Verilog In' dialog box with the following fields and values:

- File Filter Name:** A list of files including 'Makefile', 'eecs427\_core.v', 'eecs427\_pads.v', 'eecs427\_top\_apr.v' (highlighted), 'eecs427\_top.raw', and 'eecs427\_top.save.io'.
- Target Library Name:** 'eecs427\_top' (highlighted).
- Reference Libraries:** 'f8sf basic cad6 eeecs427pads RA1' (highlighted).
- Verilog Files To Import:** 'xial/verilog/eecs427\_top\_apr.v' (highlighted).
- f Options:** (empty)
- v Options:** (empty)
- y Options:** (empty)
- Library Extension:** (empty)
- Library Pre-Compilation Options:**
  - Pre Compiled Verilog Library:** (empty)
  - HDL View Name:** 'hdl' (highlighted)
  - Target Compile Library Name:** (empty)
  - Compile Verilog Library Only:**
- Ignore Modules File:** (empty)
- Import Structural Modules As:** 'schematic' (highlighted)
- Structural View Names:**
  - Schematic:** 'schematic' (highlighted)
  - Netlist:** 'netlist' (highlighted)
  - Functional:** 'functional' (highlighted)
  - Symbol:** 'symbol' (highlighted)
- Log File:** './verilogIn.log' (highlighted)
- Work Area:** '/tmp' (highlighted)
- Name Map Table:** './verilogIn.map.table' (highlighted)
- Overwrite Existing Views:**  (highlighted)
- Verilog Cell Modules:**  Create Symbol Only,  Import,  Import As Functional
- Global Nets:**
  - Power Net Name:** 'VDD' (highlighted)
  - Ground Net Name:** 'VSS' (highlighted)
  - Global Signals:** 'VDD VSS' (highlighted)

Figure 4. Verilog In

## Design Rule Check (DRC)

DRC checks should be the same as any previous DRC. We will have some DRC violations here (in this tutorial, we have 26 DRC violations). Look at the violations and fix them (adding more “V3” geometries, more *metal4*, and fixing the “V2” spacing errors).

**Note:** Pads and memories do not follow the design rules we used in Calibre, and we exclude them when running the DRC check. That is why we got *metal4* DRC violations. By adding *metal4*, those DRC violations will be fixed.

## Layout versus Schematic (LVS)

Before we do the LVS, we need to add two labels on the layout, *DVDD* and *DVSS*.

1. Use “Ctrl + f” to change display levels to 0 and you will see the sub-cell names.
2. Pick any one of the four “PDVSS” cells, and zoom in.
3. Choose the “M6” label layer in the LSW window; press ‘L’ (lower case) to create the label.
4. Type *DVSS* in the label, and put the label in the position of the cell as shown in the figure below.
5. Repeat for one of the “PDVDD” cells. Use *DVDD* for the label.

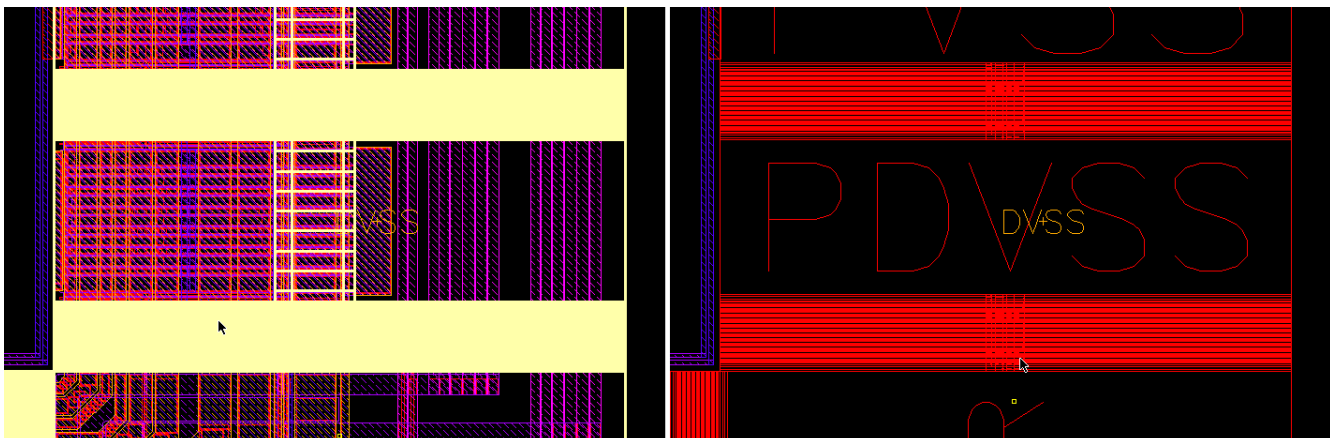


Figure 5. *DVSS* Label

Don't use \*.cdl for this LVS. Execute the same procedure from Tutorial 1, using all of the default values (exporting the schematic from the viewer), and press “Run LVS”. Hopefully, you will get the smiley face.



Congratulations!