TRESEMI
EMPOWERING TOMORROW'S LEADERS

VLSI Physical Design Fundamentals Course

Final Project

ASSIGNMENT: PHYSICAL DESIGN TRANSFORMATION

In this assignment, you'll execute the Physical Design steps to convert a synthesized netlist into the ultimate layout, ensuring it fulfills all requirements and constraints. Below are the goals and grading criteria:

Goals and Grading Criteria:

- **Zero DRC Errors:** Ensure there are no Design Rule Check errors.
    - Exceptions: Please ignore the density errors related to metal fills.
- **Zero DRV Violations:** Maintain a clean record with no Design Rule Violations.
    - Exceptions: Please ignore the max fanout violations.
- **No Timing Setup or Hold Violations:** Eliminate any timing setup or hold violations.
- **Minimize Total Cell Areas:** Strive for the smallest possible total cell areas.
- **Reduce Power Consumption:** Aim for the lowest power consumption achievable.

PROJECT DATA TRANSFER

To begin, copy the project data to your workspace using the following commands:

```
mkdir -p /home/<user>/works/Projects
cd /home/<user>/works/Projects
cp -r /mnt/TRESEMI/Projects/DTMF_CHIP .
```

Due to an error, if you made the copy before May 12, please recopy the scan def file to your workspace and rerun the steps starting with the reading of the file

```
cp /mnt/TRESEMI/Projects/DTMF_CHIP/INPUTS/def/scan_input.def
/home/<user>/works/Projects/DTMF_CHIP/INPUTS/def/.
```

DESIGN INFORMATION

To understand the design, take note of the following details:

DTMF (Dual Tone Multi Frequency) Design:

- Contains almost 6,000 instances, 57 I/Os, and about 6,400 nets.
- The netlist is hierarchical and in Verilog format.
- The process utilized is the 180-nanometer process technology with 6 layers of metal.

For further insights into the DTMF design, consult the `designDTMF.pdf` file in the `DOCS/` directory.

For more information about the DTMF design, refer to the `designDTMF.pdf` and `DTMF-Receiver-IC-Overview` files in the `DOCS/` directory.

Key Design Specifications

- **Top Design Name:**
  - `DTMF_CHIP`
- **Design Netlist:** Input netlist (gate-level netlist from the synthesis step):
  - `./INPUTS/verilog/dtmf_chip_ak.v`

TIMING CONSTRAINTS & REFERENCE LIBRARIES

To ensure clarity on timing constraints and reference libraries, review the following:

- **Timing Models and Constraints:** These are specified in the MMMC file:
  - `./INPUTS/mmmc/dtmf.mmmc`
- **LEF Abstracts:** All abstracts and P&R technology files are consolidated into one file:
  - `./INPUTS/lef/all.lef`

FLOORPLANNING CONSTRAINTS

- **IO Pad Placement File:**
  - `./INPUTS/fp/dtmf.io`

PLACEMENT CONSTRAINTS

- **Scan Definition File:**
  - `./INPUTS/def/scan_chain.def`
- **Filler Cell List:**

❖ `{FILL1 FILL2 FILL4 FILL8 FILL16 FILL32 FILL64}`

CTS CONSTRAINTS

- **CTS Buffer List:**

❖ `{CLKBUFX1 CLKBUFX12 CLKBUFX16 CLKBUFX2 CLKBUFX20 CLKBUFX3 CLKBUFX4 CLKBUFX8 CLKBUFXL}`

- **CTS Inverter List:**

❖ `{CLKINVX1 CLKINVX12 CLKINVX16 CLKINVX2 CLKINVX20 CLKINVX3 CLKINVX4 CLKINVX8 CLKINVXL}`

STEP 1: IMPORT

1. Create a script named `import.tcl` to perform the following tasks:
   - Read the MMMC file.
   - Import the LEF files, including the P&R technology file and all cell abstracts.
   - Load the Design Netlist.
     - Specify the names for the power and ground nets (use VDD for power and VSS for ground).
     - Connect all the power and ground pins to the PG nets.
     - Read in the IO Placement file

2. Generate a floorplan with a `Core` box using the following dimensions:
   - **Width:** 842 µm
   - **Height:** 842 µm
   - **Core-to-IO Distance:** 100 µm
   - **Standard Cell Site:** tsm3site

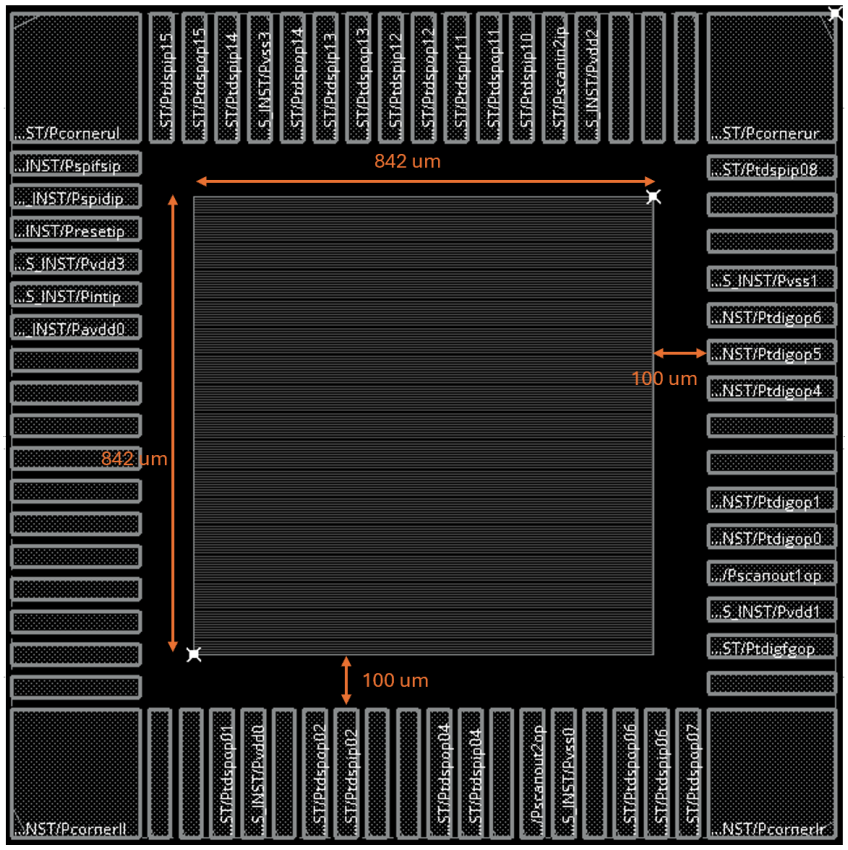
   Hint: Use the command:

   Innovus `create_floorplan -core_margins_by io -site <site_name> -core_size <width> <height> -core_margin <core_to_io_distance>`

   If you run into an error using the above command, it may be may be due to a difference in the Innovus version being used. Use this instead:

   Innovus `create_floorplan -core_margins_by io -site <site_name> -core_size <width height left bottom right top>`

   - Do a man page on create_floorplan for more details.



Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/import.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/import.tcl`

STEP 2: PLACE THE MACROS

Write CUI TCL scripts to place the all the macros, considering the following:

- Add halos as needed for DRC-clean routes.
- Utilize flightlines and other guidelines to determine the optimal location for each macro.

Storage Instructions:

- **Database Saving:**
  - Save the database to:

- data/dbs/place_macro.db

- **Script Location:**
  - Save the scripts to:

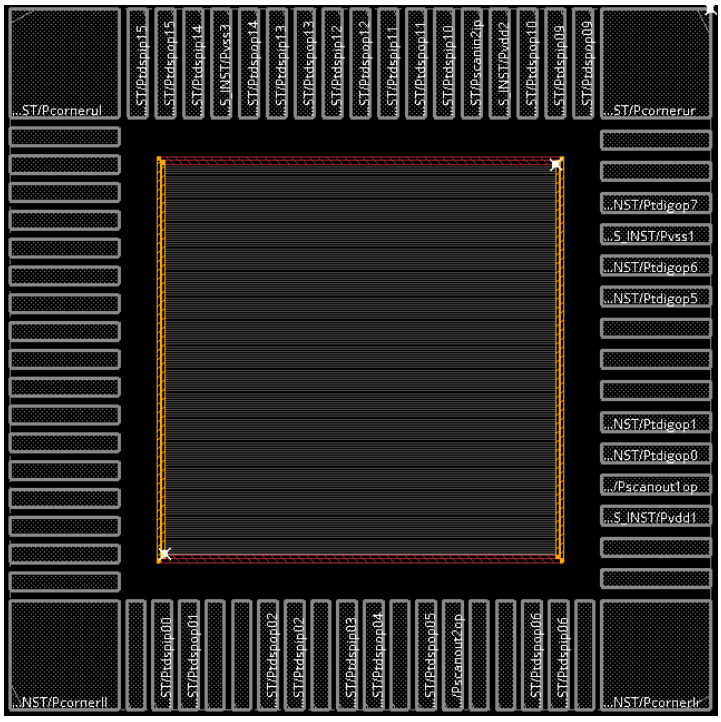
- SCRIPTS/place_macro.tcl

STEP 3: CREATE THE POWER GRID

To establish the power grid, follow these steps:

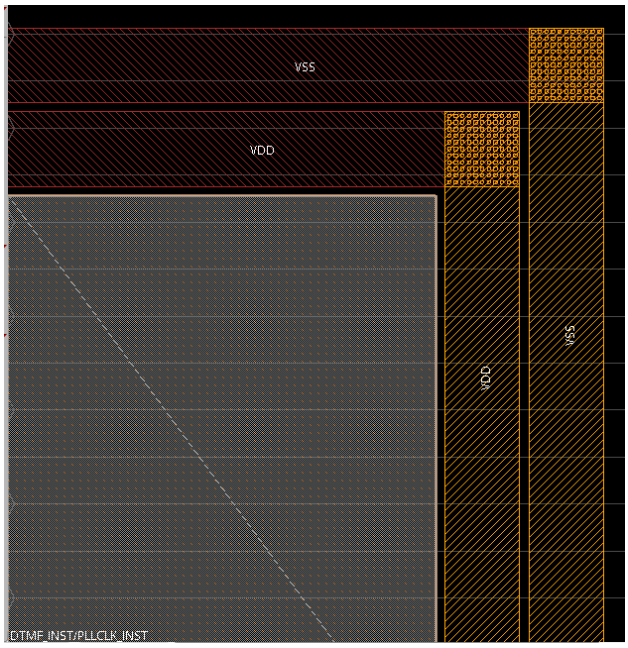
- **Add a Ring around the Core boundary for the Power and Ground Nets:**

- **Layers:** Top/Bottom: Metal5 | Left/Right: Metal6
- **Width:** 8 µm
- **Spacing Between Power and Ground Rings:** 1 µm
- **Offsets From the Core Boundary:** 1 µm for all sides.



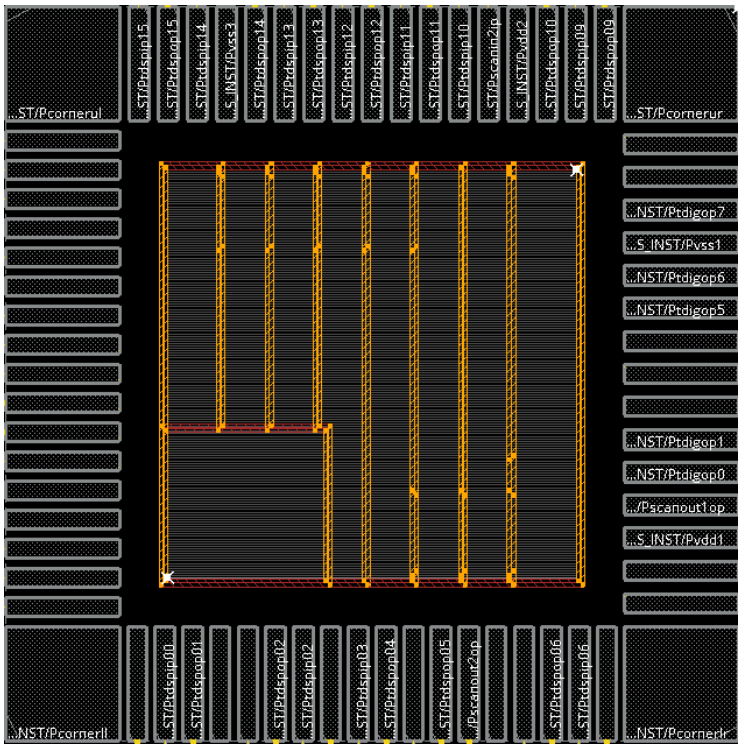
- **Add a Ring around the DTMF_INST/PLLCLK_INST Macro:**

- **Layers:** Top/Bottom: Metal5 | Left/Right: Metal6
- **Width:** 8 µm
- **Spacing Between Power and Ground Rings:** 1 µm
- **Offsets From the Core Boundary:** 1 µm for all sides.



- **Add Power/Ground Straps:**

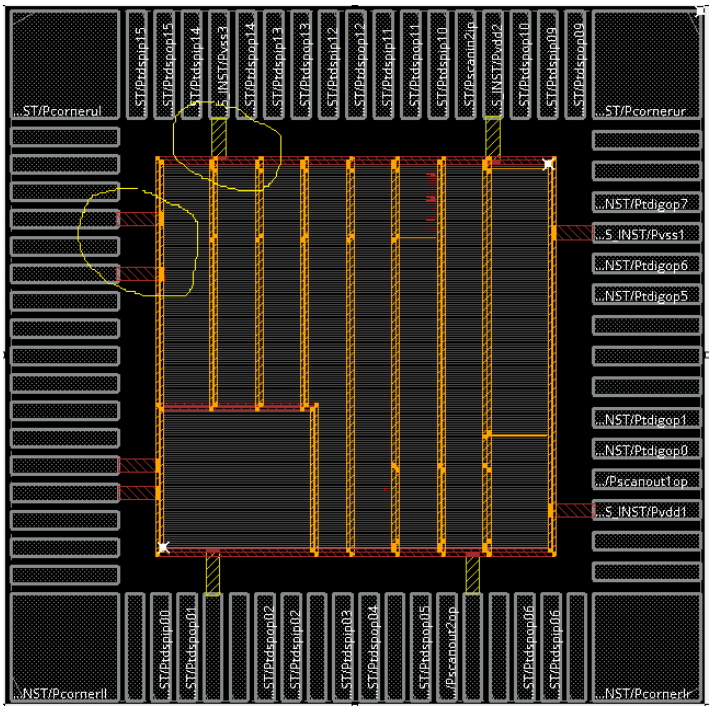
❖ Direction: Vertical
❖ Layers: Metal6
❖ Width: 8 µm
❖ Spacing Between Power and Ground Straps: 1 µm
❖ Spacing Between Power-Ground Sets: 100 µm
❖ Offsets From the Left Core Boundary: 100 µm.
❖ Connections: Top Layer: Metal6 | Bottom Layer: Metal1

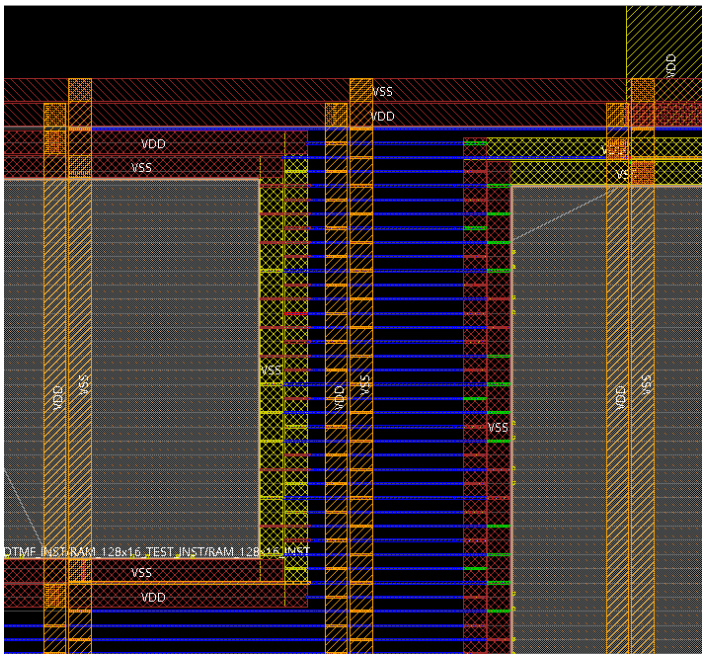


- **Route Standard Cell Rails, Macros' Power-Ground (PG) Pins, and IO cells' PG Pins to PG Nets:**

- Utilize the command `route_special` to connect all PG pins together.

❖ **Target Layers:** `Metal1 to Metal6`
❖ **Connections:** `to block_pin, pad_pin, pad_ring, core_pin, floating_stripe`





Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/create_power_grid.db`

- **Script Location:**

- Save the scripts to:

❖ `SCRIPTS/create_power_grid.tcl`

STEP 4: RUN PLACEMENT OPTIMIZATION

Execute the placement of standard cells and optimize the design for timing. Generate a timing report at the conclusion of this step.

Timing Optimization Setup:

Do the following before running the placement step:

- **Analysis Type:**
  - Choose on-chip variation mode for accuracy:

```
set_db timing_analysis_type ocv
```

- **Process Node:**
  - Set process node to 180 for parasitic extraction accuracy (TSMC18_6LM is the target process):

```
set_db design_process_node 180
```

- **Pessimism Removal:**
  - Set the removal of clock reconvergence pessimism:

```
set_db timing_analysis_cppr both
```

Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/placeopt.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/placeopt.tcl`

- **Timing Reports:**
  - Save the timing reports with a "`placeopt`" prefix to the `./reports` directory. For example:

❖ `./reports/placeopt_reg2reg.tarpt.gz`
❖ `./reports/placeopt_*`

STEP 5: RUN CTS

Generate the clock trees. Produce a timing report at the completion of this step.

Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/cts.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/cts.tcl`

- **Timing Reports:**
  - Save the timing reports with a "`cts`" prefix to the `./reports` directory. For example:

❖ `./reports/cts_reg2reg.tarpt.gz`
❖ `./reports/cts_*`

STEP 6: RUN CTS OPTIMIZATION

Perform post-CTS optimization to address setup and hold timing violations. Generate a timing report upon completion of this step.

Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/ctsopt.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/ctsopt.tcl`

- **Timing Reports:**
  - Save the timing reports with a "`ctsopt`" prefix to the `./reports` directory. For example:

❖ `./reports/ctsopt_reg2reg.tarpt.gz`
❖ `./reports/ctsopt_*`

- **Clock Tree Reports:**

- Save the clock tree reports with a "`clocktree`" prefix to the `./reports` directory. For example:
- Save output of `report_clocks` to:

❖ `./reports/clocktree.rpt`

- Save output of `report_skew_group` to:

❖ `./reports/clocktree.skew-group.rpt`

STEP 7: ROUTE THE DESIGN

Route all signals throughout the design. Conduct a DRC (Design Rule Check) and timing report at the conclusion of this step. Ensure there are no DRC errors.

Please make sure filler cells are inserted.

Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/route.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/route.tcl`

- **Timing Reports:**
  - Save the timing reports with a "`route`" prefix to the `./reports` directory. For example:

❖ `./reports/route_reg2reg.tarpt.gz`
❖ `./reports/route_*`

STEP 8: RUN ROUTE OPTIMIZATION

Perform post-Route optimization to address setup and hold timing violations. Generate a timing report upon completion of this step. Conduct a DRC (Design Rule Check) and timing report at the conclusion of this step. Ensure there are no DRC errors.

Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/routeopt.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/routeopt.tcl`

- **Timing Reports:**
  - Save the timing reports with a "`routeopt`" prefix to the `./reports` directory. For example:

❖ `./reports/routeopt_reg2reg.tarpt.gz`
❖ `./reports/routeopt_*`

STEP 9: RUN CHIP FINISHING

Execute the chip finishing steps. Generate a timing report upon completion of this step.

Storage Instructions:

- **Database Saving:**
  - Save the database to:

❖ `data/dbs/chip_finishing.db`

- **Script Location:**
  - Save the scripts to:

❖ `SCRIPTS/chip_finishing.tcl`

- **Timing Reports:**
  - Save the timing reports with a "`chip_finishing`" prefix to the `./reports` directory. For example:

❖ `./reports/chip_finishing_reg2reg.tarpt.gz`
❖ `./reports/chip_finishing_*`

STEP 10: SUBMISSION

A shared folder has been created for each user:

❖ `/home/[USER]/Share_Folder`

When you are ready to submit your project, please copy your entire run directory to this shared foler:

❖ `cp -Rp /home/[USER]/works/Projects/DTMF_CHIP /home/[USER]/Share_Folder/.`

Please note that you can make changes to this shared foler before the project deadline, but the directory will be locked after the deadline has passed.

EXTRA RECOGNITION

Students have the option to attempt the following for additional recognition:

- **1/ Increase vclk1 Frequency:** Maintain the same floorplan and maximize the vclk1 frequency to the highest achievable level without introducing DRC or other violations.
- **2/ Maximize Core Density:** Keep the same operating frequencies and optimize the core density to the highest possible level without encountering DRC or timing violations.

Perform you extra-recognition runs in the `DTMF_CHIP_MAX_FREQUENCY` and `DTMF_CHIP_MAX_CORE_DENSITY` directories:

Max Frequency Run:

- `cp -Rp /home/[USER]/works/Projects/DTMF_CHIP /home/[USER]/works/Projects/DTMF_CHIP_MAX_FREQUENCY`

- Modify INPUTS/constraints/dtmf.sdc and increase the speed.
    - `cd /home/[USER]/works/Projects/DTMF_CHIP_MAX_FREQUENCY`
    - Run all steps.
    - If all clean. Repeat the process with faster speed until you can't produce a clean design any more.
- Submit the results:
    - `cp -Rp /home/[USER]/works/Projects/DTMF_CHIP_MAX_FREQUENCY /home/[USER]/Share_Folder/.`

Max Core Density Run:

- `cp -Rp /home/[USER]/works/Projects/DTMF_CHIP /home/[USER]/works/Projects/DTMF_CHIP_MAX_CORE_DENSITY`

- Modify the core area to make it smaller.
- Run all steps.
- If all clean. Repeat the process with smaller core areas until you can't produce a clean design any more.
- Submit the results:
    - `cp -Rp /home/[USER]/works/Projects/DTMF_CHIP_MAX_CORE_DENSITY /home/[USER]/Share_Folder/.`

© Tresemi 2024