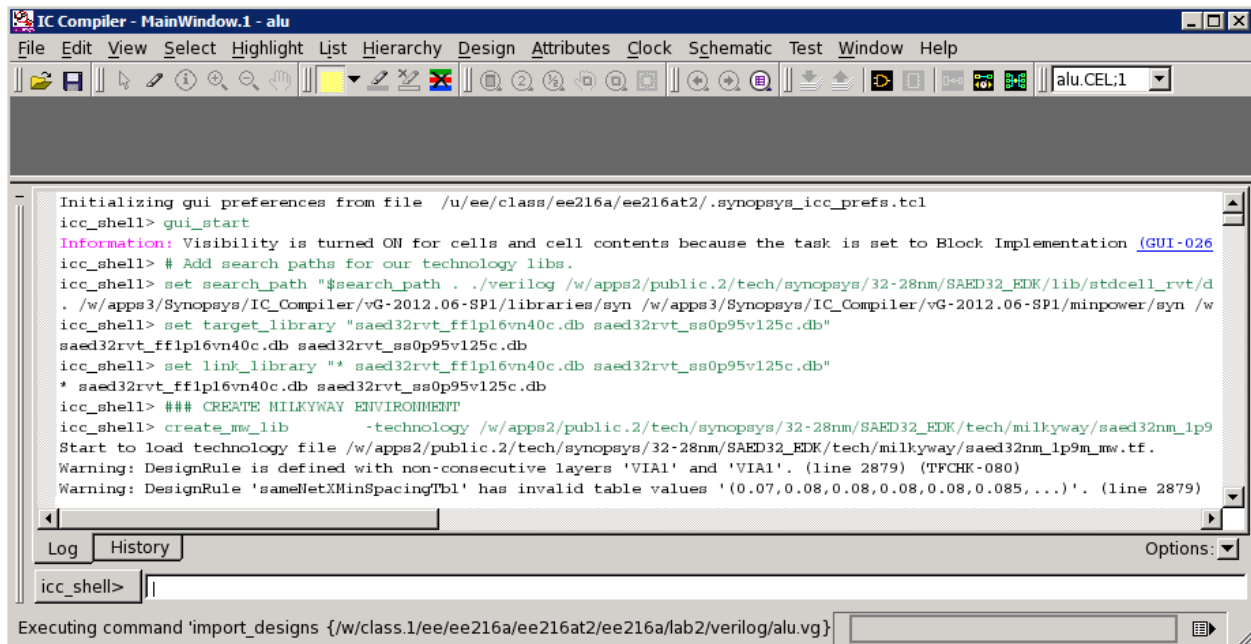


Launching IC Compiler

To start physical synthesis, you will need to source and launch IC Compiler.

```
tclsh
source /w/apps3/Synopsys/IC_Compiler/vG-2012.06-SP1/SETUP
icc_shell -gui
```

Once you do this, the IC Compiler window will popup. The “Log” tab will give you text-based feedback (i.e. warnings, errors, and area/timing/drc/lvs reports). The “History” tab will show you all of the commands that you have run/entered for the current session. IC Compiler is mainly scripting based physical synthesis tool. However, it does support and have a useful GUI interface.



Setting Up the Work Environment

You can setup the timing and parasitic libraries by going to:

File > Create Library... and File > Setup > Application Setup

```
### ADD TECHNOLOGY SEARCH PATHS
set search_path "$search_path . /w/apps2/public.2/tech/synopsys/32-
28nm/SAED32_EDK/lib/stdcell_rvt/db_nldm"
set target_library "saed32rvt_ff1p16vn40c.db saed32rvt_ss0p95v125c.db"
set link_library "*" saed32rvt_ff1p16vn40c.db saed32rvt_ss0p95v125c.db"

### CREATE MILKYWAY LIBRARY PROJECT
create_mw_lib \
    -technology /w/apps2/public.2/tech/synopsys/32-
28nm/SAED32_EDK/tech/milkyway/saed32nm_1p9m_mw.tf \
    -mw_reference_library {/w/apps2/public.2/tech/synopsys/32-
28nm/SAED32_EDK/lib/stdcell_rvt/milkyway/saed32nm_rvt_1p9m} \
    -bus_naming_style {%d} \
    {*** PATH TO YOUR LIBRARY ***}

### SET TIMING AND CAP LIBRARIES
set_tlu_plus_files \
    -max_tluplus /w/apps2/public.2/tech/synopsys/32-
28nm/SAED32_EDK/tech/star_rcxt/saed32nm_1p9m_Cmax.tluplus \
    -min_tluplus /w/apps2/public.2/tech/synopsys/32-
28nm/SAED32_EDK/tech/star_rcxt/saed32nm_1p9m_Cmin.tluplus \
    -tech2itf_map /w/apps2/public.2/tech/synopsys/32-
28nm/SAED32_EDK/tech/milkyway/saed32nm_tf_itf_tluplus.map \

### OPEN YOUR PROJECT
open_mw_lib *** PATH TO YOUR LIBRARY ***
```

Adding Your Design

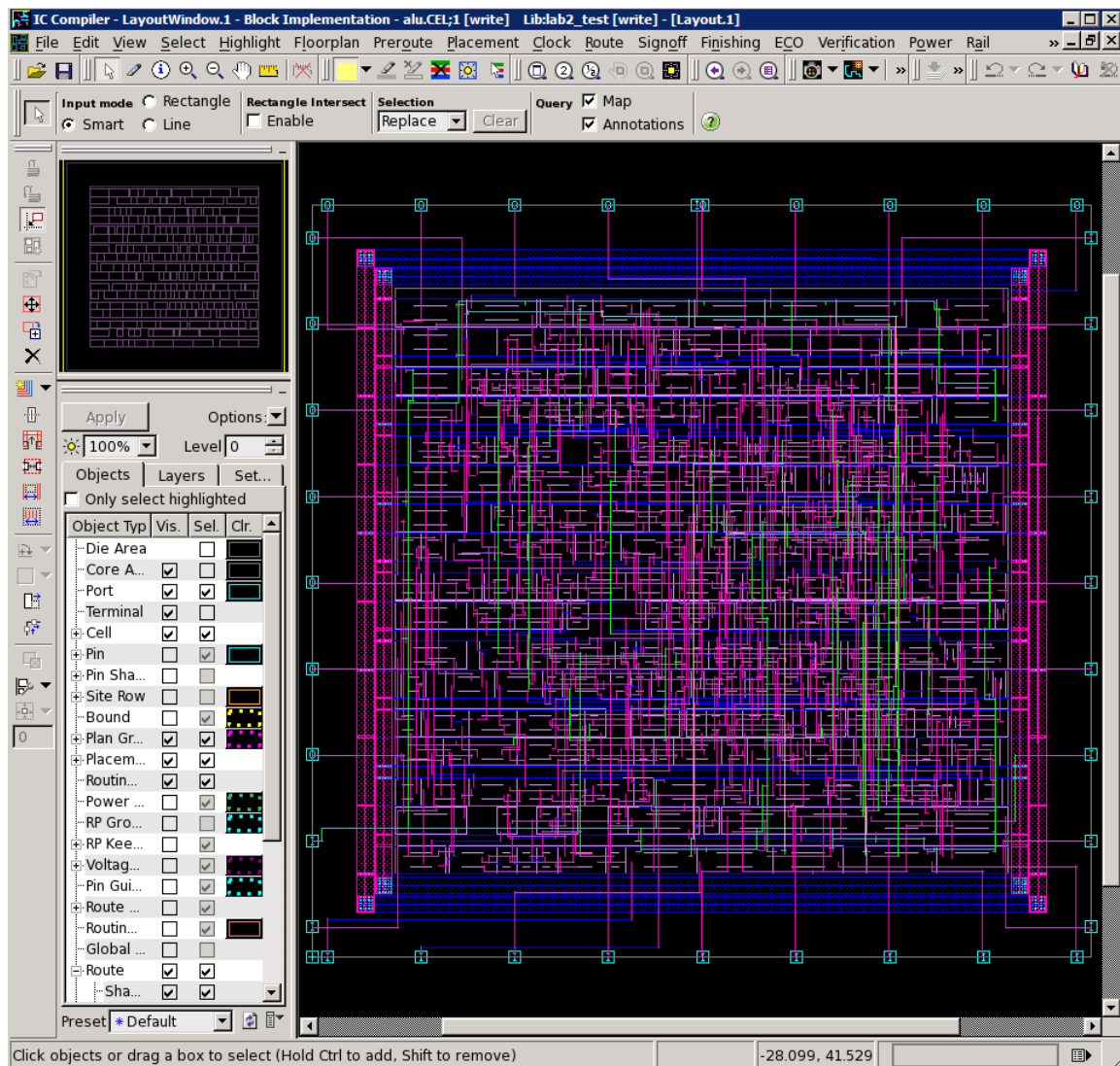
Now you add your synthesized verilog files for place and route. You can do this by going to:

File > Import Designs...

```
### ADD YOUR DESIGN
import_designs {*** PATH TO YOUR VERILOG (.VG) ***} \
    -format verilog \
    -top SGDE \
    -cel SGDE

read_sdc {*** PATH TO YOUR .SDC FILE ***}
```

This will bring up the Layout Window.



Define the Power Rails

Next we need to define the power supplies for the design. This can be accessed through the “Power” menu in the Layout Window.

```
### ADD POWER RAILS
set power "VDD"
set ground "VSS"
set powerPort "VDD"
set groundPort "VSS"
foreach net {VDD} {derive_pg_connection -power_net $net -power_pin
$net -create_ports top}
foreach net {VSS} {derive_pg_connection -ground_net $net -ground_pin
$net -create_ports top}
derive_pg_connection -tie
```

Create the Floorplan

Next create a floorplan: Floorplan > Create Floorplan...

```
### FLOOR PLAN
create_floorplan \
    -core_utilization 0.70 \
    -core_aspect_ratio 1.0 \
    -left_io2core 5 \
    -bottom_io2core 5 \
    -right_io2core 5 \
    -top_io2core 5
```

Add Power Rings

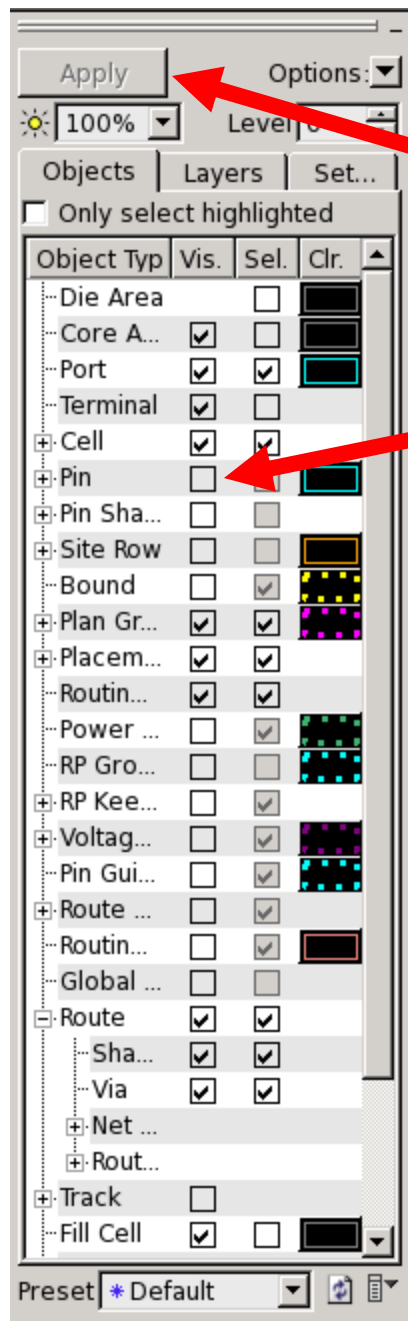
Next create power rings: Preroute > Create Rings...

```
### POWER RINGS
create_rectangular_rings \
    -nets {VDD VSS} \
    -left_offset 0.2 \
    -left_segment_width 1 \
    -right_offset 0.2 \
    -right_segment_width 1 \
    -bottom_offset 0.2 \
    -bottom_segment_width 1 \
    -top_offset 0.2 \
    -top_segment_width 1
```

Place Your Design

Place you standard cells: Placement > Core Placements and Optimization...

```
### PLACE YOUR DESIGN
place_opt
```



At this point, you should probably change the visibility of the metal layers of the standard cell pins.

MAKE SURE TO CHECK "PIN" AND HIT "APPLY."

Adding and Connecting Power Strips/Rails

Now we need to connect the power rails for the standard cells:

Preroute > Preroute Standard Cells...

```
### ADD/CONNECT POWER RAILS
preroute_standard_cells -nets {VDD VSS} -connect horizontal -
extend_to_boundaries_and_generate_pins
```

Clock Tree Synthesis

Next we need to do clock tree synthesis:

Clock > Core CTS and Optimization...

```
### CLOCK TREE SYNTHESIS
clock_opt

### CHECK TIMING
report_clock_tree
report_timing
```

Make sure to check your timing reports before continuing. If you have hold time errors, you will need to fix them now.

Routing

For routing: Route > Core Routing and Optimization...

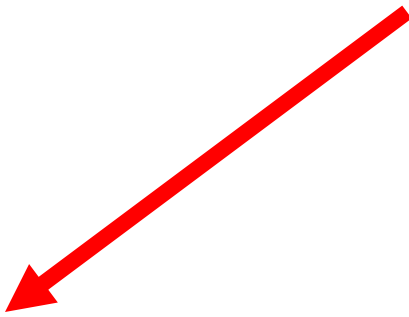
```
### ROUTE DESIGN
route_opt -effort high
```

```
Area
-----
Combinational Area:      704.995455
Noncombinational Area:  249.569412
Net Area:                143.252636
Net XLength              :      1922.25
Net YLength              :      2322.68
-----
Cell Area:              954.564867
Design Area:            1097.817503
Net Length               :      4244.93
```

Design Rules

```
-----
Total Number of Nets:      399
Nets With Violations:      0
Max Trans Violations:      0
Max Cap Violations:        0
-----
```

Before continuing, check to see if you have any DRC errors in your routing report.



Fixing DRC/LVS Errors

If you have DRC Errors, there are several options under the “Route” menu to try and fix them. Same goes for LVS Errors.

```
### FIXING DRC ERRORS
route_search_repair \
    -rerun_drc \
    -loop "10"

### CHECK/FIX LVS ERRORS
route_zrt_eco -max_detail_route_iterations 5
verify_lvs -check_open_locator -check_short_locator
```

Adding Filler Cells

Now that your design is routed and DRC/LVS clean, it is time to add filler cells:

Finishing > Insert Standard Cell Filler...

```
### ADD FILLER CELLS
insert_stdcell_filler \
    -cell_without_metal "SHFILL128_RVT SHFILL64_RVT SHFILL3_RVT
SHFILL2_RVT SHFILL1_RVT" \
    -connect_to_power {VDD} \
    -connect_to_ground {VSS}

### CONNECT ANY STRAY POWER NETS
foreach net {VDD} {derive_pg_connection -power_net $net -power_pin
$net -create_ports top}
foreach net {VSS} {derive_pg_connection -ground_net $net -ground_pin
$net -create_ports top}
```

DRC/LVS Error Check

Do one last check for DRC/LVS errors:

Verification > DRC ...

Verification > LVS ...

```
### VERIFY DRC AND LVS
verify_drc
verify_lvs
```

Stream Out the Final GDS

Finally, save your design as a GDS file: File > Export > Write Stream...

```
### ADD YOUR DESIGN
set_write_stream_options \
    -output_pin {text geometry} \
    -keep_data_type
write_stream \
    -format gds \
    -lib_name *** PATH TO YOUR LIBRARY *** \
    -cells {SGEE} SGDE.gds
```