Preparing a custom block for digital-on-top integration

Prof. Adam Teman

25 December 2021









Introduction







What do we need to deliver

• Early on in the project:

- Behavioral (RTL) model for RTL simulation
- At an intermediate stage
 - Layout abstract (.lef) for floorplanning and power planning
 - Timing model (.lib) for synthesis
- For signoff
 - Spice netlist (.cdl) for LVS
 - Full layout (.gds) for LVS and DRC
- In the following slides, we will overview how to prepare your custom block and create these files.

Lecture Overview





Behavioral Model





Behavioral Description

- Assuming the custom block's top hierarchy is called "topcell", your behavioral description is a Verilog file with the following properties:
 - File name: topcell.v
 - Module name: topcell
 - Interface: Input and output ports with the same names as the circuit port names.

No power/ground inout ports.

 Body of code: A description of how you want your block to behave in digital simulation. If possible, it should behave <u>equivalent to the actual behavior</u> of the custom block.

This does not have to be synthesizeable.

```
ClkxSI —
                                                                                                DOXDO
Example: Memory Macro
                                                                 WExSI -
                                                                             MEM[4096][64]
                                                                 RExSI -----
                                                              WAddrxDI ----
                                                              RAddrxDI -----
timescale 1ns/1ps
                                                                 module edram wPS(
                                               reg [WIDTH-1:0] MEM [0:DEPTH-1];
                                               integer timestamp [0:DEPTH-1];
  // Outputs
  DOxDO,
                                               initial timestamp = '{DEPTH{0}};
  // Inputs
  ClkxSI, WAddrxDI, RAddrxDI,
                                               always @(posedge ClkxSI) begin
         DIxDI, WExSI, RExSI
                                                   if (WExSI) begin
                                                       MEM[WAddrxDI] DIxDI;
   );
                                                       timestamp[WAddrxDI] <= $time;</pre>
parameter WIDTH = 64;
                                                   end
parameter DEPTH = 4096;
parameter RT IN NS = 1000000;
                                                   if (RExSI) begin
localparam DEPTH W = $clog2(DEPTH);
                                                       if (timestamp[RAddrxDI] + RT IN NS > $time)
                                                           DOxDO <= MEM[RAddrxDI];</pre>
input ClkxSI;
                                                       else begin
input [DEPTH W-1:0] WAddrxDI; // write address
                                                           DOxD0 <= {WIDTH/16{16'haa55}};</pre>
input [DEPTH W-1:0] RAddrxDI; // read address
                                                       end
input [WIDTH-1:0] DIxDI; // input data
                                                   end
input WExSI; // write enable
                                               end
input RExSI; // read enable
output reg [WIDTH-1:0] DOxDO; // output data
                                               endmodule
```



Layout Abstract





Layout Abstract (.lef)

- Well before you have finished your detailed layout, you will need to provide the backend team with a physical abstract of your block, including:
 - Size (Width x Height)
 - Pin locations and metal layers ____
 - Power connections
 - Blockages
- This is provided in as a .lef (Library Exchange Format) file, which can be:
 - Written by hand
 - Created with

10





Preparing your abstract: Pins (Ports)

- There are many guidelines that can improve your *routability*, but in general:
 - Put pins on layers according to process preferred routing direction.
 - Try to align your pins to a periodic grid, preferably the routing tracks.
 - Try to separate ports by at least one empty track.
 - Pins should reach the edge of the macro and be completely covered by routing blockage ("cover" blockage with no "pin cutouts")
 - Power (PG) Pins should be fully detailed and wide enough for a via to be dropped. This will enable <u>multiple connections</u> to power.

Periodic grid





Preparing your abstract: Power Grid

- Need a robust power delivery network
 - For preventing IR Drop and EM
 - Rule of thumb: 10% of routing resources for power
- Two main structures:

• Ring



Preparing your abstract: Others

- Make sure you provide Antenna information in your LEF
 - This will enable the router to solve antenna problems
 - Otherwise, during signoff DRC, you will have "surprises"

Prepare yourself for Metal Fill

- All layers have to have density between 30%-70% or so.
- This will be added during signoff using an automated flow.
- If you have sensitive areas, add metal fill blockage to prevent this.
- If you are sensitive to timing, add the metal fill at the macro level.
- High Voltage Markers
 - If you are using non-standard voltages, add Marker CAD layers to employ the correct DRC rules on these nets.
- Make sure your Origin is at (0,0) and your PR Boundary is correct!

Using the Abstract Generator



- The Abstract Generator is a Cadence tool, started by running abstract
- The Abstract Generator has a five-step flow
 - 1. Import Library (Layout)
 - 2. Import Logical (Interface)
 - 3. Pin Derivation
 - 4. Shape Extraction Layout
 - 5. Abstract Generation
- When this is done you can export the .lef file.

		_
	Abstract - [no current library] _ 🗆	×
	ile Bins Cells Flow He	Яþ
I		
10.	Core Core Core Corner Import Block gnore Logical Pins Extract Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Cent Ce	
	Interpreter: 💠 Tcl 🔶 Skill	
	Log Command History	
	INFO (ABS-19020): Starting @(#)\$CDS: ui version 6.1.7-64b 05/29/2018 20:01 (sjfhw308) \$, sub-version IC6.1.7-64 b.500.20 , on 3.18.26 INFO (ABS-19023): This is the OpenAccess variant of Abstract Generator.	
	abstract>	
		- 1

Using the Abstract Generator (2)



- Import Library Stage:
 - Choose a library from Virtuoso (cds.lib)
- Import Logical Stage:
 - Upload a Verilog (.v) or Liberty (.lib) file to provide the pin interface.
- Pins:
 - Correct way: define pins (shapes defined with "Create Pin") in your layout.
 - You can also have the tool extract pin shapes based on Labels.
 - In the MAP tab define pin layers and make sure all power/gnd label names are specified.
- After running, check the abstract.pin cellview in your library

-		
	Running step Pins for	the selected cell(s)
Step	Map Text Boundary Blocks	
🔶 Pins	Map text labels to pins:	
	((M5 pin)(M5 drawing))(M6 pin)(M6 d	rawing))
	Power pin names (regular expressions):	Maps text labels in the layout view to pins For example: (textLPP1 geomLPP1 geomLPP2)
	VDD VBOOST VNEG VREF	(textLPP2 geomLPP3 geomLPP4) Where textLPP has syntax of either: (textLayer textPurpose) or simply textLayer geomLPP uses the same syntax.
	, Ground pin names (regular expressions): GND	If you specify a single layer rather than a la then the abstract generator considers "all p specified as ignore purpose(s) on the Gene

Using the Abstract Generator (3)

Extract Stage

- SIGNAL tab: Only leave the layers that you want to be written as signals to the LEF.
- Power tab: Only leave the layers that you want to be written as power pins to the LEF.
- Select "EXTRACT POWER NETS" so the whole power wire will be extracted
- ANTENNA tab: Enable all options. This will provide antenna data in the LEF, otherwise, you will probably have antenna violations in full chip DRC.
- After running, check the abstract.ext cellview in your library

16



	R	unning step Extract for the sele	ected cell(s)	×			
Step	Signal Pow	rer Antenna General					
 Extract 	Layer Assignm	ent for Power Extraction					
	Layer	Geometry Specification		Create Pins			
	_ M6	M6					
	R	unning step Extract for the sele	ected cell(s)	×			
Step	Signal Pow	rer Antenna General					
🗸 Pins	Calculate hierarchical antenna						
🔶 Extract	📕 Calculate in	put pin antenna					
	📕 Calculate o	utput pin antenna <mark>.</mark>					
	📕 Calculate in	out pin antenna					
	Calculate a	ntenna metal area					
	📕 Calculate al	ntenna metal side area					
	Layer Assignm	nent for Antenna Regions					
	Layer	Geometry Specification	Region	Oxide			
	_ PO	PO and OD	Gate				
		OD andnot PO	Drain				
			A				
				HUH, ZUZ			



Using the Abstract Generator (4) Open Library ______

Abstract stage:

- ADJUST tab: Select "CREATE BOUNDARY PINS" for signals only! (uncheck for power!)
- BLOCKAGE tab: Define "COVER" blockage on all layers to block inside the macro.
- DO NOT select "PIN CUTOUT" for pin layers. This can lead to DRC errors during routing!
- For Power Metals, select "PIN CUTOUT"!
- After running, check the abstract cellview
 - Check that pins touch the edges
 - Check that blockages are created correctly (OBJECTS → BLOCKAGES → ROUTING BLOCKAGES).

		Import GDSII Data -		Abstract
		Import Logical Data -		Verify
-	Rur	ning step Abstract for the se	elected cell(s)	×
Step Pins Extract	Adjust Bloc	kage Density Fracture Site	Overlap	
Abstract	Boundary pin m	dary pins ax distance to boundary:		
a	Signal geometry	/ groups:		
	Power Nets			
_	Create bour Boundary pin m	dary pins ax distance to boundary:		
	Ring pin max di	oins stance to boundary:		
-	Runi	ing step Abstract for the sel	lected cell(s)	_ = ×
Step	Adjust Block	age Density Fracture Site	Overlap Uns	select!
🗸 Extract	Layer Assignme	nt for Blockages		
🔶 Abstract	Layer	Geometry Specification	Blockage	Ph Cutol t Max Spac
	M1	M2	Cover	
	M3	M3	Cover	
	M4	M4	Cover	
	, ,		1	
s 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2	• K K K K K K K K K K K K K K K K K K K			
□ OD] □ PO] □ CO]				<u>à</u>
Image: Minimum display="block">Image: Minimum display="block" Image: Minimage: Minimage: Minimum display="block"		signal	Pin M3	
M5		C) A	dam lei	man. 2021



Using the Abstract Generator (5)

- Open the file to reassure it was generated as expected:
 - You can see the MACRO name and the BLOCK class
 - The SIZE is calculated
 - The ORIGIN should be at (0,0)
 - All the PINs are created and have coordinates in the correct layers.
 - The antenna information is described.
 - "Cover" blockage should be defined, i.e., very few rectangles of type OBS.

```
edram4096x64.lef + (/tsmcproject/tsmc16ff/users/ha
File Edit Tools Syntax Buffers Window
VERSION 5.6 ;
BUSBITCHARS "[]" ;
DIVIDERCHAR "/"
PROPERTYDEFINITIONS
 MACRO CatenaDesignType STRING ;
IND PROPERTYDEFINITIONS
MACRO edram4096x6
 CLASS BLOCK ;
 ORIGIN 0 0 ;
 FOREIGN edram4096x64 0 0 ;
  SIZE 111.702 BY 295.92 ;
  SYMMETRY X Y R90
 PIN ClkxSI
    DIRECTION INPUT :
    USE CLOCK :
    ANTENNAPARTIALMETALAREA 6.35092 LAYER M5 :
     LAYER M5 :
       RECT 55.851 295.736 55.891 295.776 ;
    END
 END ClkxSI
 PIN DIXDI[0]
   DIRECTION INPUT ;
   USE SIGNAL ;
    ANTENNAPARTIALMETALAREA 11.82368 LAYER M5 :
     LAYER M5 ;
       RECT 101.017 295.736 101.057 295.776 ;
   END
 END DIXDI[0]
 PIN DIXDI[10]
   DIRECTION INPUT :
   USE SIGNAL ;
    ANTENNAPARTIALMETALAREA 11.82368 LAYER M5 ;
    PORT
     LAYER M5 ;
       RECT 87.817 295.736 87.857 295.776 ;
   END
 END DIXDI[10]
 PIN DIxDI[11]
   DIRECTION INPUT ;
   USE SIGNAL ;
    ANTENNAPARTIALMETALAREA 11.82368 LAYER M5 ;
    PORT
-- INSERT --
                       ⊎ Auum remun, zu2
```

Shortcut for Initial LEF

- You will need to provide an initial LEF early on in the project for floorplanning.
 - After defining footprint, power connections and pin placement, you will need to freeze this and <u>only change the internals</u>.
- A quick way to do this is to use Innovus:
 - Create "empty" Verilog netlist (module) <u>only including interface</u>.
 - Load this netlist into Innovus according to technology backend flow.
 - Define floorplan size with create_floorplan command
 - Define power rails (rings/stripes) and pins (edit_pin)
 - Export LEF with the write_lef_abstract command

write_lef_abstract my_macro.lef -stripe_pins -pg_pin_layers {M5} \
 -port_for_each_stripe_pin -extract_block_pg_pin_layers {M5} -top_layer M6



Timing Model (LIB)





Timing Model (.lib)

- The digital-on-top flow is based on static timing analysis (STA) verification
 - Throughout the flow, max-delay, min-delay and DRV constraints are checked.
- For each instance in the design, STA requires:
 - Propagation delays through timing arcs
 - Capacitances (loads) on pins (mainly inputs)
 - Drive capabilities (affecting transition) of outputs
 - Type of transition (rising/falling) due to an input toggle
 - Dynamic and Static Power consumption
- This data is provided by a Liberty (.lib) file, based on:
 - Input net transition (t_{rise}, t_{fall})
 - Output Load Capacitance (C_{load})
- A separate .lib file should be provided for each corner (PVT+RCX)



Liberty Format

• A .lib file has the following hierarchy:

- Library: a collection of cells at a certain corner.
- Cell: a specific cell (e.g., macro)
- **Pin:** Timing/Power information for each pin

• Each pin has:

- General info, e.g., capacitance, functionality
- Timing: Propagation delay and output transition

pin(Y) {

- Power: Power consumption of arc
- Constraint: Setup/Hold constraints

```
lu_table_template(delay_template_5x5) {
 variable 1 : input net transition;
 variable_2 : total_output_net_capacitance;
 index_1 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0");
 index 2 ("1000.0, 1001.0, 1002.0, 1003.0, 1004.0");
```

```
Technology Library
                          Date and Revision
                          Library Attributes
                          Environmental Descriptions
                               Default Attributes
                               Nominal Operating Conditions
                               Custom Operating Conditions
                               Scaling Factors
                               Wire Load Models
                               Timing Ranges
                          Cell Descriptions
                               Cell Attributes
                               Sequential Functions
                               Bus Descriptions
                                    Default Attributes
                                    Bus Pin Attributes
                                        Pin Attributes.
                                        Combinational Function
                                         Timing.
                                              Timing Attributes
                                             Timing Constraints
cell (INVX1) {
     timing() {
      cell rise(delay template 5x5) {
         values ( \
            "0.1479, 0.2180, 0.3598, 0.922, 1.766", \
            "0.2243, 0.2929, 0.4303, 0.991, 1.831", \
            "0.3653, 0.4487, 0.5842, 1.135, 1.970", \
            "0.4620, 0.5515, 0.7010, 1.244, 2.081", \
            "0.7564, 0.8742, 1.0570, 1.628, 2.449"); }
```

Library Characterization

- The best way to create a .lib file is with a library characterization tool, such as Cadence Liberate
 - For each timing arc, a SPICE testbench is generated and run.
 - Propagation delay, output transition, power consumption, setup/hold constraints, etc. are extracted and written to a .lib file
- Cadence Liberate comes in several "flavors"
 - Liberate: Used for Standard Cell characterization
 - Liberate-MX: Used for Memory Macro characterization
 - Liberate-AMS: Used for Analog Mixed-Signal (custom) blocks
- However, setting up Liberate-AMS is often beyond our scope, and therefore, we will generate our .libs manually or semi-automatically.

Manually Creating a .lib File

• Library:

- Copy general information (only required parts) from standard cell library
- Cell:
 - Cell name should be equivalent to cell name in Virtuoso
 - Manually fill in area and copy other fields from standard cell library
 - Provide power information in pg_pin fields
- Input Pins
 - For each input pin, provide (measured) input capacitance
 - If input pin has setup/hold constraints, measure and provide table
- Output Pins
 - Provide timing field for each timing arc of each output pin
 - For each of these generate tables (measured) for delay, transition, and power
 © Adam Teman, 2021

Wrapping with Registers

- To get around the tough measurement, surround your block with registers
 - Inputs to the design are just D pins of flip flops.
 - Setup/Hold constraints are between the clock and D of these flip flops
 - Output arcs are just clock-to-Q arcs of flip flops
- So, you can copy the cell definition of the flip flop from the standard cell library for each pin
- Or better yet, you can semi-automate this by:
 - Writing an interface-only block in Verilog with input/output sampling.
 - Loading into Innovus and writing out a .lib.



© Adam Teman, 2021

Semi-Automatic .lib Generation

Interface-only Verilog Netlist



© Adam Teman, 2021

Semi-Automatic .lib Generation

• Load into Innovus and write out .lib file

```
# Invoke by: innovus -stylus -no_gui -file gen_initial_lib.tcl
```

Read in the MMMC file for this technology
read_mmmc generic_innovus_mmmc.tcl

```
# Read in interface-only netlist
read_netlist my_block_interface.v
```

init_design

Write out .lib in one of the corners
write_timing_model -view typical_corner my_block_interface.lib

Final .lib adjustment

- Even when surrounding your block with registers, you will have different timing than the "vanilla" standard cell, due to your internal wires:
 - Measure input capacitance and update input pins.
 - Measure skew between inputs/outputs and clock: *in_skew=δci-δi out_delay= δco-δo*
 - Update input pin setup/hold constraints:
 setup -= in_skew (setup got easier)
 hold += in_skew (hold got harder)
 - Update output pin propagation delay:
 tcq += out_delay (delay got longer)





LVS Netlist





LVS Netlist (CDL)

- LVS is "Layout vs. Schematics". Therefore, we need to provide:
 - Layout Usually in the GDSII format
 - Schematic Usually in the CDL (Circuit Description Language) format
- CDL is a subset of SPICE
 - There are some subtle differences, but it is basically equivalent to SPICE.
 - You shouldn't include parasitics in your CDL file!
- The CDL is the result of netlisting your schematic
 - Netlisters are an important part of the EDA flow.
 - Virtuoso runs a netlister before running a simulation.
 - Calibre/PVS run a netlister to create an LVS run.
 - Virtuoso also has a *standalone netlister* for creating CDL files.
 - Note that these three options, don't always provide the same result!

Exporting your CDL

- Ensure that your circuit:
 - Has no GLOBAL nets (i.e., with !).
 - Uniquify your cell names
 This is essential for instantiated standard cells!
- From the CIW, select FILE → EXPORT → CDL.
 - Choose your schematic to export.
 - The exported file is defined in the OUTPUT section
 - Select Map Bus Names FROM <> TO [] to provide Verilog compatible bus names!
 - Select CONNECTION BY NAME to use the \$PINS style
 of connectivity, as opposed to connection by reference/position.
- Another option is to take the Calibre/PVS netlist generated during LVS.

Design to be Netlisted	
Library Name	my_library Library Browser
Top Cell Name	my_topcell
View Name	schematic
Switch View List	auCdl schematic
Stop View List	auCdl
Output	
Output CDL Netlist File	my_cdl.cdl
Run Directory	. Browse
Netlisting Mode	🔾 Digital 🖲 Analog
Run in Background	Z
Renetlist	
Other Inputs	
Analog Netlisting Type	Connection By Order
Resistor Threshold Value	2000
Resistor Model Name	
Equivalents	
nclude File	Browse
Check Resistors	● value ○ size ○ none
Check Capacitors	\odot value \bigcirc area \bigcirc perimeter \bigcirc both \bigcirc none
Check Diodes	\bigcirc area \bigcirc perimeter \textcircled{o} both \bigcirc topology \bigcirc none
Scale	● meter ⊖ micron ⊖ none
Shrink Factor for Width and Length	0
Check LDD	
Display Pin Information	¥
Map Bus Name from < > to []	Preserve '!' in the Netlist
Global Power Signals	
Global Ground Signals	
Print Inherited Connections	
r	



LVS Layout Stream





LVS Layout (Stream)

- There are two options to extract a GDSII file from your layout:
 - Standalone XStream Out (strmout) tool
 - Take it from Calibre/PVS extraction
- From the CIW, select FILE → EXPORT → STREAM .
 - Select the layout view (LIBRARY/TOP CELL/VIEW) of the cell you want to export
 - The exported GDS is defined in the STREAM FILE field
 - The TECHNOLOGY LIBRARY should be the techlib of the PDK.
 - The LAYER MAP is a file that translates between the layout layer and the GDS layer number. It should be provided in your PDK.

• Select **TRANSLATE** to export your GDS

• But not before you follow the steps on the next slide!

	XStream Out (on enicsw04.local) _ 🗆 ×				
Stream File	my_layout.gds				
Library	my_library				
Top Cell(s)	my_topcell				
View(s)	layout				
echnology Library	tsmcN65				
Template File					
	Stream Out from Virtual Memory				
Layer Map	/data/tsmc/65LP/pdk/tsmcN65/tsmcN65.layermap				
Object Map					
Log File	strmOut.log				
<u>[</u> ranslate <u>A</u>	pply <u>Cancel</u> <u>Reset All Fields</u> <u>More Options</u> <u>H</u> elp				
e cell you want to export					

Stream Out – Important Points

- In the XSTREAM OUT form, choose the MORE OPTIONS button and:
 - Under Cell Mapping add Cell NAME PREFIX (or SUFFIX), such as "my_cell_" and select IGNORE CELL NAME PREFIX AND SUFFIX FOR TOP CELL.
 - Under MAPPING -> GENERAL select REPLACE <> WITH [].

xs	tream Out More Options (on enicsw04.local)	×		XS	tream Out More Op	otions (on enicsw04.local)	×
Search Mapping General Geometry Transformation Limit General Messages Report General	Cell Map File Name ☐ Wildcard ✓ Ignore C ☐ Respect GDS Cell Name Length Cell Name Prefix my_cell_ Cell Name Suffix Sub-Master Separator _CDNS_ Cell Case Sensitivity preserve	d in Cell Map Cell Name Prefix and Suffix for Top Cel 32 Characters Name Limit		Sear	rch Q V Mapping 	Ref Lib File Name Prop Map File Name Font Map File Name Label Map File Name Label Translation Depth	Use All Libraries as Ref Lib Output Property Value Only Replace < > with []	
	<u>о</u> к	<u>Apply</u> <u>D</u> efaults <u>C</u> ancel	Help			•	OK Apply Defaults	ancel <u>H</u> elp



Summary





Summary

• Following all these steps, you should provide the following early on:

- my_block.v file: behavioral model for digital simulation
- my_block_initial.lef: early stage LEF file with frozen size, pins and power.
- my_block_initial.lib: early stage LIB file with registers around interface
- Your final delivery for signoff should include:
 - my_block.lef: final .lef that should be equivalent to initial .lef.
 - my_block.lib: final .lib that should be equivalent to initial .lib, if you used the register wrapping approach.
 - my_block.cdl: uniquified SPICE (.cdl) netlist without parasitics or globals.
 - my_block.gds: uniquified, DRC/LVS clean GDSII file with metal fill blocks, where required, or after local metal fill.