# Batch Processing: The Complete Synthesize, Place, and Route Flow

Daniel Liu, John Lee, Puneet Gupta
*University of California, Los Angeles, NanoCAD Lab*
*{daniel,lee,puneet}@ee.ucla.edu*

## Abstract

*Design automation has been growing rapidly over the past decade, and the advancement of design tools hashelped the design of VLSI circuits vastly. This article explains the full standard synthesize, place and route (SPR) design flow of digital circuits.*

**Key Words:** RTL, Verilog, Encounter, TCL script, delay, power, area, technology library.

## 1. Introduction

Modern digital circuits contain vast amounts of cells which makes it impossible for human to manage manually. Nevertheless, thanks to computer aided design, even the design of digital circuits with millions of gates are manageable. This is enabled by the electronic design automation flow for digital circuits, Synthesis-Place and Route (SPR), which is an indispensable and crucial process for design engineer. In this report, the basics and the  of the major parts in the SPR flow will be described, as well as a demonstration of the usage of a few scripts to compare the results of some benchmark designs synthesized with different technology libraries.

## 2. The SPR Flow

The design flow of Automatically Synthesized Integrated Circuits (ASICs), and general digital design consists of three main steps, namely synthesis, placement of the standard cells, and routing of the design.

Logic synthesis is the process of converting a high-level description of a design into gate-level netlist, or RTL. Synthesis tools (i.e. RTL compiler) read in the hardware description language (HDL) of the design and construct it with standard cells from the supplied technology library with optimizations. The technology libraries are foundry-specific and are usually known and categorized by the transistor size, or minimum pitch (e.g. 60nm or 90nm). The technology library, or the so-called standard cell library, is composed of basic logic gates, such as NAND, NOR, INV, Flip-flop, XOR, MUX and etc, that are widely used in all digital circuits.

There are several advantages to designing in high-level description and compiling the design through synthesis. For example, high-level design is less prone to human errors since designs are described by a high level abstraction and need not to worry significantly on design constraints yet, but only on the functionality. Having an automated and independent logic synthesis step also makes the work of designers technology-independent and therefore portable and reusable. In addition, the synthesized netlist is almost design-style-independent and is automatically optimized by synthesis tools.

After synthesis, the gate netlist is obtained, and the next step is to place the standard cells to their optimal positions. Yet, before placing the cells, some floorplanning of the die is required, to create sites for the standard cells. The floorplanning process takes into account information such as the percent utilization of the total die and the orientation of the power rails. Once floorplanning is done, the cells are ready to be placed on the die. With Cadence Encounter, placing the standard cells can be done by a single command, and the placer automatically makes its best effort, using various algorithms, to put the cells to their optimal position for later routing.

The last step is to route the design. This includes routing of the power wires (`sroute` in Encounter), and the signal interconnects between pins and gates (`nanoroute`), and the clock lines.

Of course, it is always necessary to physically verify the validity of the design after all the processes. Design rule check (DRC) and layout versus schematic (LVS) check are two of these verifications. DRC checks whether the design violates any design rules that may cause problems after fabrication, while LVS checks if the final layout matches the netlist at the transistor level. The layout of the design is written as a GDSII, which  is a industry standard database format for IC layout. The exported file will contain the information of the actual polygons and layers that are necessary for manufacturing.

## 3. Synthesis Script Explained

The following is a sample script for Cadence RTL Compiler for synthesizing benchmark circuit c17. Comments have been added to improve the readability and understanding of the code.

```
# Script for Cadence RTL Compiler synthesis
# To run: rc < rc.tcl

# All HDL files, separated by spaces
set hdl_files {c17.v, gatelib.v}

# Name of the design
set DNAME c17
# The Top-level Module
set DESIGN c17

# Set clock pin name in design. If `clk` just leave untouched,
# otherwise change clk
set clkpin clk

# Target delay in ps for optimization
set delay 5

# Path in which the synthesizer looks for the design HDL files
set_attribute hdl_search_path {./} /
# Path in which the synthesizer looks for the technology librar
set_attribute lib_search_path ~/NangateOpenCellLibrary/liberty

# On a scale of 0 to 9, set the level of information to be shown for
# the synthesis process (9 max, 0 min, 6 is recommended)
set_attribute information_level 6 /

# Set target technology library liberty file (contains delay / power info)
set_attribute library FreePDK45_lib_v1.0_typical.lib

# Read in the hardware description files (HDL)
read_hdl -v2001 ${hdl_files}

# Elaboration is only required for top-level design
# - build data structures
# - infers registers in design
# - performs high-level HDL optimization (e.g. dead code removal)
# - checks semantics
elaborate $DESIGN

# Apply Constraints
set clock [define_clock -period ${delay} -name ${clkpin} [clock_ports]]
    external_delay -input  0 -clock ${clkpin} [find / -port ports_in/*]
    external_delay -output  0 -clock ${clkpin} [find / -port ports_out/*]

# Sets transition to default values for Synopsys SDC format,
# fall/rise 400ps
dc::set_clock_transition .1 ${clkpin}

check_design -unresolved
report timing -lint

#*Synthesis
# The process of transforming HDL design into a gate-level netlist,
# given all the specified constraints and optimization settings
synthesize -to_mapped

# Write out the report of timing, power, and the cells used
report timing > reports/timing_synth.rep
report gates  > reports/cell_synth.rep
report power  > reports/power_synth.rep

# Write out the synthesized file
write_hdl -mapped > output/${DNAME}_synth.v
# SDC is the synopsis design constraints file
write_sdc >  ${DNAME}.sdc

report timing -lint -verbose
puts \n
puts "Synthesis Finished!"
puts \n
puts "Check timing.rep, area.rep, gate.rep and power.rep for synthesis results"
puts \n
quit
```

## 4. Floorplan, Place and Route Script Explained

The next step after synthesis is the floorplanning, cell placement, and wire routing:

```
# Script for Cadence SOC Encounter
# To run: source this TCL script in Encounter
# see the "Encounter Text Command Reference (fetxtcmdref.pdf)
# for more information on the commands

# Setup design and create floorplan
# Load in config file "enc.conf"
loadConfig ./enc.conf

# Create Initial Floorplan
# Aspect ratio 1.0, 70% utilization, and 20 units
# core to left/bottom/right/top distance
# ( Aspect ratio specifies the chip's core dimensions as the ratio
# of the height divided by the width)
floorplan -r 1.0 0.7 20 20 20 20

# Define Global Power Nets
globalNetConnect VDD -type pgpin -pin VDD -inst * -module {}
globalNetConnect VSS -type pgpin -pin VSS -inst * -module {}

# Create Power structures (only after floorplanning)
# Create a powering with metal 1 and 2
addRing -spacing_bottom 5 -width_left 5 -width_bottom 5 -width_top 5 -spacing_top 5 -layer_bottom metal1 -width_right 5 -around core -center 1 -layer_top metal1 -spacing_right 5 -spacing_left 5 -layer_right metal2 -layer_left metal2 -nets { VDD VSS }

# Place standard cells
# Runs placement based on the global settings for placement,
# RC extraction, timing analysis, and trial routing
placeDesign

# Route power nets
# Routes power structures (after placing power rings)
sroute -noBlockPins -noPadRings -noPadPins

# Perform trial route and get initial timing results
# trialRoute does not guarntee DRC, only used to estimate parasitic
# values for timing analysis
trialroute
buildTimingGraph
setCteReport
report_timing -nworst  10 -net > timing.rep.1.placed
```

```
# Run Clock Tree Synthesis
createClockTreeSpec -output encounter.cts -bufferList BUF_X1 BUF_X2 BUF_X4 BUF_X8 BUF_X16 BUF_X32 INV_X1 INV_X2 INV_X4 INV_X8 INV_X16 INV_X32
specifyClockTree -clkfile encounter.cts
# cts.rguide is the name of the routing guide file
ckSynthesis -rguide cts.rguide -report report.ctsrpt -macromodel report.ctsmdl -fix_added_buffers

# Output Results of CTS (Clock Tree Synthesis)
trialRoute -highEffort -guide cts.rguide
extractRC
reportClockTree -postRoute -localSkew -report skew.post_troute_local.ctsrpt
reportClockTree -postRoute -report report.post_troute.ctsrpt

# Run Post-CTS Timing analysis
setAnalysisMode -setup -async -skew -autoDetectClockTree
buildTimingGraph
setCteReport
report_timing -nworst  10 -net > timing.rep.3.cts

# Perform post-CTS IPO
setIPOMode -highEffort -fixDrc -addPortAsNeeded -incrTrialRoute -restruct -topomap
setExtractRCMode -default -assumeMetFill
extractRC

# Run Post IPO-2 timing analysis
buildTimingGraph
setCteReport
report_timing -nworst  10 -net > timing.rep.4.ipo2

# Connect all new cells to VDD/GND
globalNetConnect VDD -type tiehi
globalNetConnect VDD -type pgpin -pin VDD -override

globalNetConnect VSS -type tielo
globalNetConnect VSS -type pgpin -pin VSS -override

# Run global Routing
# utilizes the nano router
globalDetailRoute


# Write the final gds file
streamOut outputs/i2c_master_route.gds -libName DesignLib -structureName i2c_master -stripes 1 -units 2000 -mode ALL

# save the netlist
saveNetlist -excludeLeafCell i2c_master_final.v

puts "**************************************"
puts "* Encounter script finished         *"
puts "* --------                          *"
puts "* Layout:  final.gds2               *"
puts "* Netlist: final.v                  *"
puts "* Timing:  timing.rep.5.final       *"
puts "**************************************"
exit
```

## 5. The Batch Scripts

In order to synthesize, place, and route a batch of designs automatically, two script were created: `synth.sh` and `placeRoute.sh` . These scripts work similarly as the scripts above, but the batch script will run the same settings to all the target design.

The read-me files are shown below, and for brevity of this report, the actual scripts are not shown. The scripts are located on the server `dfm.ee.ucla.edu` .

```
--------------------------------------------------------------------
Documentation for `synth.sh` (referred to as "the script")
--------------------------------------------------------------------
Summer 2010
Author: Daniel Liu, UCLA undergraduate
Purpose: To synthesize a batch of verilog designs by simply running one script.
Dependency: Cadence RTL logic synthesis tool (RC), synthesize.tcl (needs to be in the same directory)

--------------------------------------------------------------------
How to run the script?
--------------------------------------------------------------------
To run the script, have all you designs verilog files in a directory and put the directory in the same folder as the script (synth.sh) .

Then, simply execute the command `./synth.sh` in the terminal.

After the script is done executing, two folders will be created in each of the design directories. The two folders are: `output` and `reports`

--------------------------------------------------------------------
What do the folders contain?
--------------------------------------------------------------------
After running the script.

->Folder `output`: This folder will hold the synthesized verilog files after running the script (with a flattened version).
->Folder `reports`: This folder will hold the files which contains the information of the synthesized design (e.g. power and etc..)
```

```
--------------------------------------------------------------------
Documentation for `placeRoute.sh` ("the script")
--------------------------------------------------------------------
Summer 2010
Author: Daniel Liu, UCLA undergraduate
Purpose: To run placement and routing on a batch of sythesized verilog designs by simply running one script.
Dependency: Cadence Encounter, sythesized verilog file
```

```
-------------------------------------------------------------------
How to run the script?
-------------------------------------------------------------------
This script is intended to ru  after the `synth.sh` script is ran. It looks for the synthesized verilog file in the `output` directory and runs placement and routing using that file.

To run, simply execute the command `./placeRoute.sh` in the terminal.

After the script is done executing, some files will appear in each of the design's `output` directories, including the Encounter file, DEF, GDSII, and so on.
```

## 6. Simulation Results on Benchmark Circuits

Some benchmark circuits were chosen arbitrarily to be synthesized, placed, and routed using two different technology libraries. The technology libraries used are `Nangate` and `ST`. The comparison results are shown in the tables on the right.

Note:
Nan* -> NangateOpenCellLibrary_PDKv1_3_v2009_07 (45nm) ;
ST->CORE65LPSVT (65nm)

### ISCAS85

POWER

| | Leakage Pwr (nW) | | Switching Pwr (nW) | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| c17 | 125.38 | 3.985 | 671.22 | 2649 |
| c432 | 1862.45 | 74.32 | 11176.52 | 50843.46 |
| c499 | 9058.12 | 223.98 | 123168.46 | 504059.31 |
| c880 | 5505.35 | 111.7 | 33042.86 | 83568 |
| c1355 | 9098.4 | 227.7 | 121594.22 | 484819.63 |
| c1908 | 10160.78 | 296.39 | 118342.82 | 454048.78 |
| c2670 | 13777.8 | 310.8 | 96593.31 | 256835 |
| c3540 | 23267.42 | 583.34 | 160874.55 | 475153.73 |
| c5315 | 26723.09 | 629.9 | 254770.95 | 763669.34 |
| c6288 | 46050.87 | 1160 | 797042.6 | 2344400 |
| c7552 | 39450.99 | 927.08 | 362362.14 | 1057355.83 |

TIMING

| | Delay (pico-second) | |
|---|---|---|
| | Nan* | ST |
| c17 | 60 | 27 |
| c432 | 416 | 340 |
| c499 | 487 | 392 |
| c880 | 402 | 318 |
| c1355 | 484 | 391 |
| c1908 | 793 | 705 |
| c2670 | 600 | 527 |
| c3540 | 1097 | 989 |
| c5315 | 1099 | 978 |
| c6288 | 2110 | 1756 |
| c7552 | 895 | 762 |

CELL INFORMATION

| | Number of Cells | | Total Cell Area (dbu) | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| c17 | 9 | 5 | 7.45 | 35.36 |
| c432 | 112 | 151 | 104.01 | 680.68 |
| c499 | 499 | 385 | 484.92 | 2188.16 |
| c880 | 323 | 295 | 334.89 | 1245.4 |
| c1355 | 492 | 402 | 476.94 | 2216.76 |
| c1908 | 658 | 731 | 588.13 | 2761.72 |
| c2670 | 1021 | 1014 | 824.87 | 2968.16 |
| c3540 | 1595 | 1559 | 1396.23 | 5292.04 |
| c5315 | 2033 | 2036 | 1904.03 | 6664.32 |
| c6288 | 2883 | 2570 | 2704.95 | 11395.28 |
| c7552 | 2952 | 2984 | 2546.68 | 9105.2 |

WIRE LENGTH

| | Total Wire Length (um) | | Total # of VIAs | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| c17 | 85.69 | 44.9 | 14 | 5 |
| c432 | 448.18 | 1463.74 | 219 | 417 |
| c499 | 1834.44 | 4452.81 | 1193 | 1110 |
| c880 | 2118.92 | 4708.98 | 1113 | 1262 |
| c1355 | 2030.38 | 4695.36 | 1205 | 1236 |
| c1908 | 2408.85 | 7842.55 | 1644 | 2029 |
| c2670 | 5482.58 | 11231.38 | 2501 | 3276 |
| c3540 | 5697.51 | 22609.77 | 3910 | 6083 |
| c5315 | 11076.16 | 33152.49 | 6384 | 9704 |
| c6288 | 6805.69 | 51165.28 | 6359 | 11901 |
| c7552 | 12740.29 | 43185.36 | 7466 | 11417 |

### ISCAS89

POWER

| | Leakage Pwr (nW) | | Switching Pwr (nW) | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| s1196 | 6825.63 | 94.3 | 21740055.13 | 35552616.99 |
| s1238 | 6585.08 | 92.83 | 4372778.16 | 35807965.14 |
| s13207 | 26473.67 | 421.15 | 369016693 | 632872177.17 |
| s1423 | 10303.43 | 220.54 | 108029318.84 | 151306761.87 |
| s1488 | 9749.8 | 169.93 | 9737424.42 | 12447953.21 |
| s1494 | 9134.86 | 199.9 | 9736492.41 | 12463243.5 |
| s15850 | 11714.06 | 193.73 | 194984050.06 | 268259588.09 |
| s27 | 391.05 | 5.34 | 3792229.34 | 6068937.11 |

| | | | | |
|---|---|---|---|---|
| s298 | 2193.6 | 40.7 | 20837491.46 | 28879281.45 |
| s344 | 2818.81 | 46.03 | 17733193.91 | 29592536.14 |
| s349 | 2753.87 | 44.51 | 18538082.68 | 29609543.12 |
| s35932 | 175475.32 | 3531.15 | 2524902627.13 | 3525519287.32 |
| s382 | 3036.98 | 52.96 | 31243005.95 | 42670468.12 |
| s38417 | 176701.67 | 2954.21 | 1842011936.4 | 3098272066.39 |
| s38584 | 139508.68 | 2087.58 | 1409407547.86 | 2301113264.86 |
| s386 | 2698.92 | 43.5 | 9048620.17 | 12084195.98 |
| s400 | 2866.36 | 55.84 | 32404103.66 | 43117293.62 |
| s444 | 3063.78 | 57.17 | 31775246.6 | 42831903.01 |
| s526 | 3776.2 | 74.25 | 31918175 | 42621131.52 |
| s5378 | 19390.2 | 364 | 180778137.74 | 353535213.15 |
| s641 | 2878.32 | 50.68 | 2748260.43 | 36217718.77 |
| s713 | 2938.47 | 51.8 | 27233697.13 | 36256792.65 |
| s820 | 5011.63 | 103.11 | 7707411.56 | 10521690.63 |
| s832 | 5036.86 | 96.03 | 7707727.08 | 10469147.27 |

TIMING

| | Delay (pico-second) | |
|---|---|---|
| | Nan* | ST |
| s1196 | 384 | 314 |
| s1238 | 362 | 374 |
| s13207 | 422 | 363 |
| s1423 | 696 | 598 |
| s1488 | 474 | 422 |
| s1494 | 446 | 450 |
| s15850 | 459 | 428 |
| s27 | 236 | 235 |
| s298 | 347 | 350 |
| s344 | 378 | 354 |
| s349 | 368 | 359 |
| s35932 | 428 | 372 |
| s382 | 371 | 362 |
| s38417 | 819 | 681 |
| s38584 | 500 | 454 |
| s386 | 384 | 342 |
| s400 | 404 | 355 |
| s444 | 367 | 354 |
| s526 | 356 | 385 |
| s5378 | 482 | 474 |
| s641 | 409 | 370 |
| s713 | 389 | 366 |
| s820 | 403 | 385 |
| s832 | 408 | 377 |

CELL INFORMATION

| | Number of Cells | | Total Cell Area (dbu) | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| s1196 | 423 | 385 | 481.99 | 1352.52 |
| s1238 | 415 | 363 | 471.62 | 1316.64 |
| s13207 | 1063 | 1206 | 2530.19 | 6169.84 |
| s1423 | 557 | 639 | 878.07 | 2730.52 |
| s1488 | 566 | 514 | 557 | 1791.4 |
| s1494 | 529 | 544 | 532.53 | 2022.28 |
| s15850 | 511 | 555 | 1073.04 | 2670.2 |
| s27 | 23 | 17 | 32.98 | 69.16 |
| s298 | 117 | 115 | 169.97 | 479.44 |
| s344 | 137 | 122 | 198.7 | 525.2 |
| s349 | 139 | 121 | 194.18 | 526.24 |
| s35932 | 8071 | 9753 | 15700.12 | 42509.48 |
| s382 | 166 | 173 | 248.44 | 666.64 |
| s38417 | 8023 | 7752 | 15304.84 | 38503.4 |
| s38584 | 7173 | 7224 | 11970.53 | 29354.52 |
| s386 | 161 | 135 | 175.83 | 489.84 |
| s400 | 156 | 169 | 236.47 | 702.52 |
| s444 | 167 | 172 | 247.38 | 709.28 |
| s526 | 222 | 210 | 288.61 | 871 |
| s5378 | 978 | 1045 | 1652.13 | 4735.12 |
| s641 | 170 | 167 | 235.94 | 669.24 |
| s713 | 187 | 174 | 247.91 | 668.2 |
| s820 | 320 | 293 | 306.7 | 1107.6 |
| s832 | 303 | 282 | 303.77 | 1034.28 |

WIRE LENGTH

| | Total Wire Length (um) | | Total # of VIAs | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| s1196 | 3312.55 | 7003.14 | 2149 | 2294 |
| s1238 | 3109.76 | 6031.11 | 2158 | 2134 |
| s13207 | 6541.18 | 23068.71 | 4531 | 6833 |
| s1423 | 2608.26 | 9177.17 | 1795 | 2780 |
| s1488 | 4661.25 | 8139.55 | 2981 | 2651 |
| s1494 | 5116.87 | 8895.45 | 2842 | 2698 |
| s15850 | 2534.37 | 7380.59 | 1902 | 2426 |
| s27 | 133.09 | 127.99 | 46 | 63 |
| s298 | 603.37 | 1328.88 | 429 | 498 |
| s344 | 721.41 | 1413.22 | 464 | 515 |
| s349 | 689.47 | 1365 | 394 | 505 |
| s35932 | 70318.6 | 366367.02 | 32225 | 61471 |
| s382 | 721.94 | 1229.6 | 548 | 516 |
| s38417 | 60654.36 | 256954.18 | 36963 | 53079 |
| s38584 | 72234.55 | 318853.84 | 34268 | 59325 |
| s386 | 724.08 | 1689.35 | 487 | 622 |
| s400 | 683.68 | 1901.09 | 515 | 714 |
| s444 | 695.82 | 1670.57 | 524 | 657 |
| s526 | 794.74 | 2429.24 | 659 | 887 |
| s5378 | 7528.72 | 21261.4 | 4216 | 5998 |
| s641 | 924.57 | 2298.41 | 521 | 673 |
| s713 | 1105.29 | 2257.22 | 595 | 686 |
| s820 | 1785.71 | 3890.34 | 1098 | 1268 |
| s832 | 1747.9 | 3623.19 | 1115 | 1159 |

## Other

POWER

| | Leakage Pwr (nW) | | Switching Pwr (nW) | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| cpu | 59858.25 | 1164.54 | 293245453 | 921708289 |
| fpu | 1063538.81 | 28122.22 | 575937089 | 1394867607 |
| nova | 1461508.62 | 17318.76 | 37981512.5 | 107178839 |

TIMING

| | Delay (pico-second) | |
|---|---|---|
| | Nan* | ST |
| cpu | 1042 | 934 |
| fpu | 10454 | 9411 |
| nova | 2724 | 2453 |

CELL INFORMATION

| | Number of Cells | | Total Cell Area (dbu) | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| cpu | 2743 | 2452 | 4407.09 | 11992.76 |
| fpu | 63348 | 60660 | 63965.55 | 262837.64 |
| nova | 53143 | 47231 | 123008.77 | 254010.64 |

WIRE LENGTH

| | Total Wire Length (um) | | Total # of VIAs | |
|---|---|---|---|---|
| | Nan* | ST | Nan* | ST |
| cpu | 19256.54 | 83845 | 12898 | 23246 |
| fpu | 405090.03 | 6699556.94 | 243524 | 1088722 |
| nova | 751634.95 | TBA | 387889 | TBA |

## 7. Conclusion

As we can see from the data, the results from 65nm(ST) and 45nm(Nangate) libraries are reasonable; bigger transistors produce less leakage power because it suffers less from short channel effects and Vth scaling, making it less unlikely for electrons to leak through. On the other hand, since transistors are bigger, it takes more energy to switch the gates, which results in a larger switching power in the ST library.

The other two measures, cell area and wire length also increase with the dimension of the gates or transistors. Again, as we can see, circuits synthesized using the ST library tend to use up more area as well as longer wire length.

## 8. Ongoing Project

Currently, I am working on a project will John Lee from NanoCAD Lab on multi-row-height cell placement. We wish to find the wire length, area, etc trade-offs using standard cells of different heights.

John and I have tried modifying the LEF of the Nangate technology library so that gates above 4X are stretched to have a higher height and smaller width. We wanted to see if Encounter can be “tricked” to place the stretched cells on different rows automatically. However, this did not succeed. Then we also tried alternating the DEF file so we specifically ask Encounter to create different row heights and let it place the cells automatically. In this case, Encounter complains about the row heights are not integer multiples of a single row. Therefore, we still haven’t successfully found the trade-offs of multi-row-height cell placement.

We will continue to work on the project. Since we did not get Encounter to place the cells correctly, we will try conducting the project using a different tool, mPL. The tool mPL is an open-source program, and we shall be able to do what we desire.

## 10. References

[1] P. Gupta, “EE209S The VLSI Design Manufacturing Interface”, *EE209S,* University of California