Tutorial 4
ECE1388 Project 3
4x4 Array Multiplier

Mustafa Kanchwala

ECE1388

What is our target?

- Example:

$$\begin{array}{l l l}
\underline{\begin{array}{r}1100\\0101\end{array}} & \begin{array}{l}: 12_{10}\\: 5_{10}\end{array} & \begin{array}{l}\text{multiplicand}\\\text{multiplier}\end{array}\\
\\
\underline{\hspace{6em}} & : 60_{10} & \text{product}
\end{array}$$

- M x N-bit multiplication

General Form

- Multiplicand: $Y = (y_{M-1}, y_{M-2}, \ldots, y_1, y_0)$
- Multiplier: $X = (x_{N-1}, x_{N-2}, \ldots, x_1, x_0)$
- Product: $$P = \overbrace{\left(\sum_{j=0}^{M-1} y_j 2^j\right)}^{\mathbf{Y}} \overbrace{\left(\sum_{i=0}^{N-1} x_i 2^i\right)}^{\mathbf{X}} = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

						y_5	y_4	y_3	y_2	y_1	y_0	multiplicand
						x_5	x_4	x_3	x_2	x_1	x_0	multiplier
						x_0y_5	x_0y_4	x_0y_3	x_0y_2	x_0y_1	x_0y_0	
					x_1y_5	x_1y_4	x_1y_3	x_1y_2	x_1y_1	x_1y_0		
				x_2y_5	x_2y_4	x_2y_3	x_2y_2	x_2y_1	x_2y_0			partial products
			x_3y_5	x_3y_4	x_3y_3	x_3y_2	x_3y_1	x_3y_0				
		x_4y_5	x_4y_4	x_4y_3	x_4y_2	x_4y_1	x_4y_0					
	x_5y_5	x_5y_4	x_5y_3	x_5y_2	x_5y_1	x_5y_0						
p_{11}	p_{10}	p_9	p_8	p_7	p_6	p_5	p_4	p_3	p_2	p_1	p_0	product

Multiplication

- Example:

```
1100 : 12₁₀
0101 :  5₁₀
```

multiplicand

multiplier

partial products

product

- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

Multiplication

- Example:

  $1100 : 12_{10}$ multiplicand

  $0101 : 5_{10}$ multiplier

  partial products

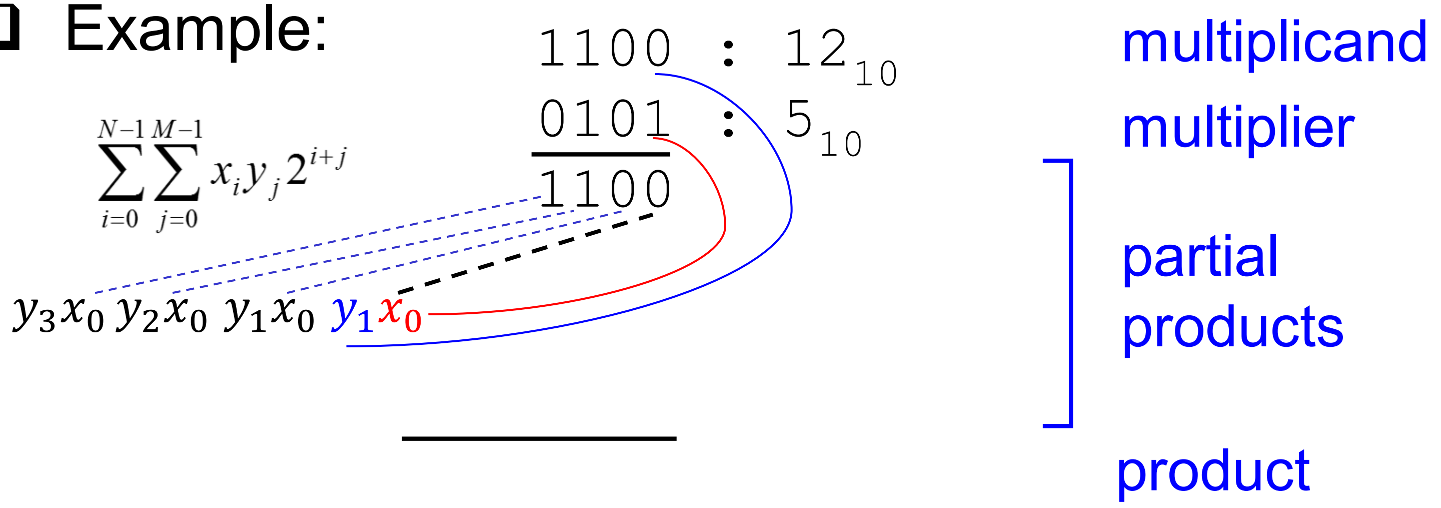
  product

- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

Multiplication

- Example:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$



- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

Multiplication

- Example:

$$\sum_{i=0}^{N-1}\sum_{j=0}^{M-1} x_i y_j 2^{i+j}$$

```
    1100 : 12₁₀    multiplicand
    0101 : 5₁₀     multiplier
    ----
    1100           partial
   0000            products
```

$y_3x_0\ y_2x_0\ y_1x_0\ y_1x_0$

$y_3x_1\ y_2x_1\ y_1x_1\ y_1x_1$

product

- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

Multiplication

- Example:

$$\begin{array}{r l l}
1100 & :\ 12_{10} & \text{multiplicand} \\
\underline{0101} & :\ 5_{10} & \text{multiplier} \\
1100 & & \\
0000 & & \text{partial} \\
1100 & & \text{products} \\
\underline{0000} & & \\
 & & \text{product}
\end{array}$$

- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

Multiplication

- Example:

$$
\begin{array}{r l l}
1100 & : 12_{10} & \text{multiplicand} \\
\underline{0101} & : 5_{10} & \text{multiplier} \\
1100 & & \left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right] \\
0000 & & \text{partial} \\
1100 & & \text{products} \\
0000 & & \\
\hline
 & & \text{product}
\end{array}
$$

- M x N-bit multiplication
  – Produce N M-bit partial products
  – Sum these to produce M+N-bit product

Multiplication

- Example:

```
    1100 : 12₁₀     multiplicand
    0101 : 5₁₀      multiplier
    ----
    1100            ┐
   0000             │ partial
  1100              │ products
 0000               ┘
--------
                    product
```

- M x N-bit multiplication
  - Produce N M-bit partial products
  - Sum these to produce M+N-bit product

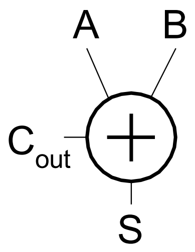
How do we perform multi-input addition?

Single-Bit Addition

Half Adder

$S = A \oplus B$

$C_{out} = A \bullet B$

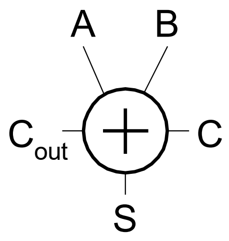


A	B	C_{out}	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Full Adder

$S = A \oplus B \oplus C$

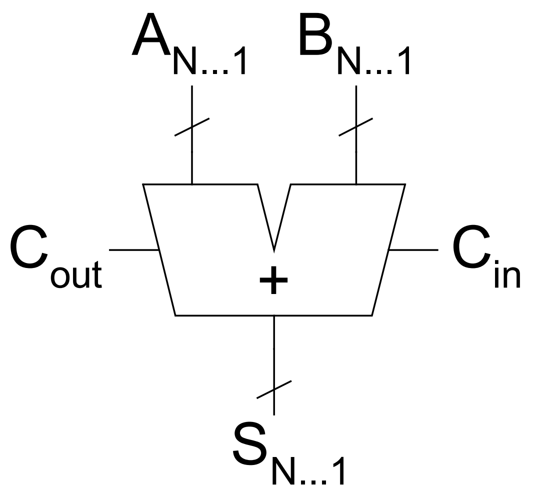
$C_{out} = MAJ(A, B, C)$

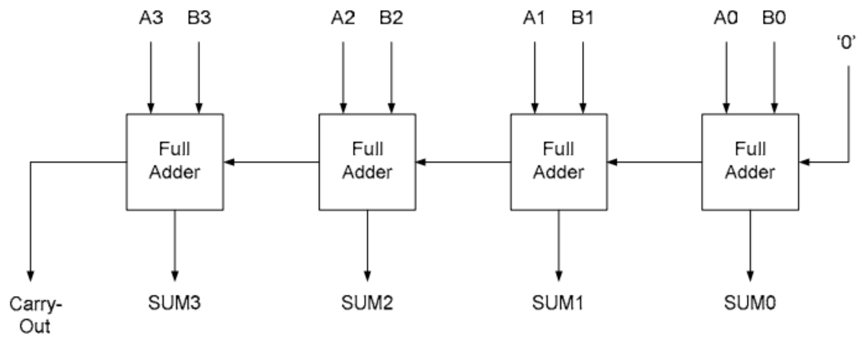


A	B	C	C_{out}	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Carry Propagate Adders

- N-bit adder called CPA
  - Each sum bit depends on all previous carries
  - How do we compute all these carries quickly?

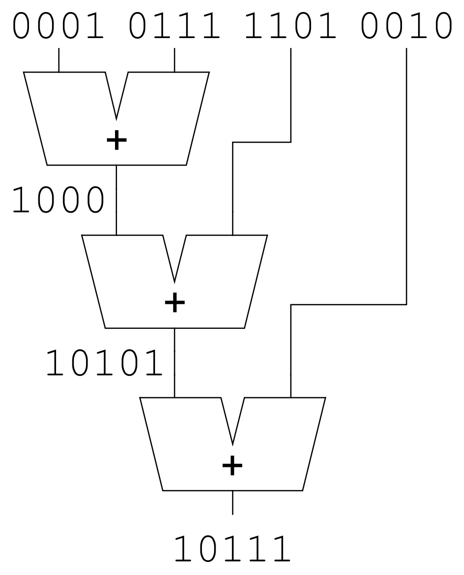




$$\begin{array}{rcrl} C_{out} \quad C_{in} & & C_{out} \quad C_{in} & \\ \boxed{0}000\boxed{0} & & \boxed{1}111\boxed{1} & \text{carries} \\ 1111 & & 1111 & A_{4...1} \\ +0000 & & +0000 & B_{4...1} \\ \hline 1111 & & 0000 & S_{4...1} \end{array}$$

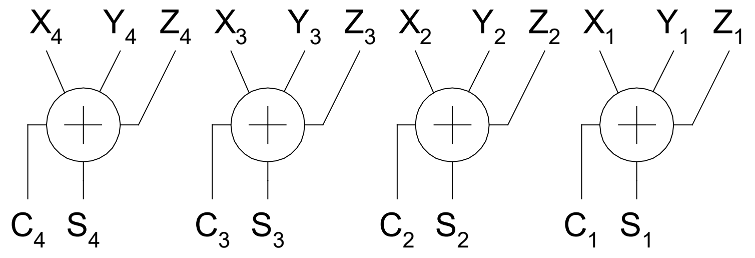
Multi-input Adders

- Suppose we want to add k N-bit words
  - Ex: 0001 + 0111 + 1101 + 0010 = 10111
- Straightforward solution: k-1 N-input CPAs
  - Large and slow

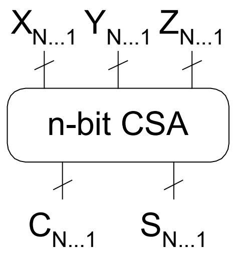


Carry Save Addition

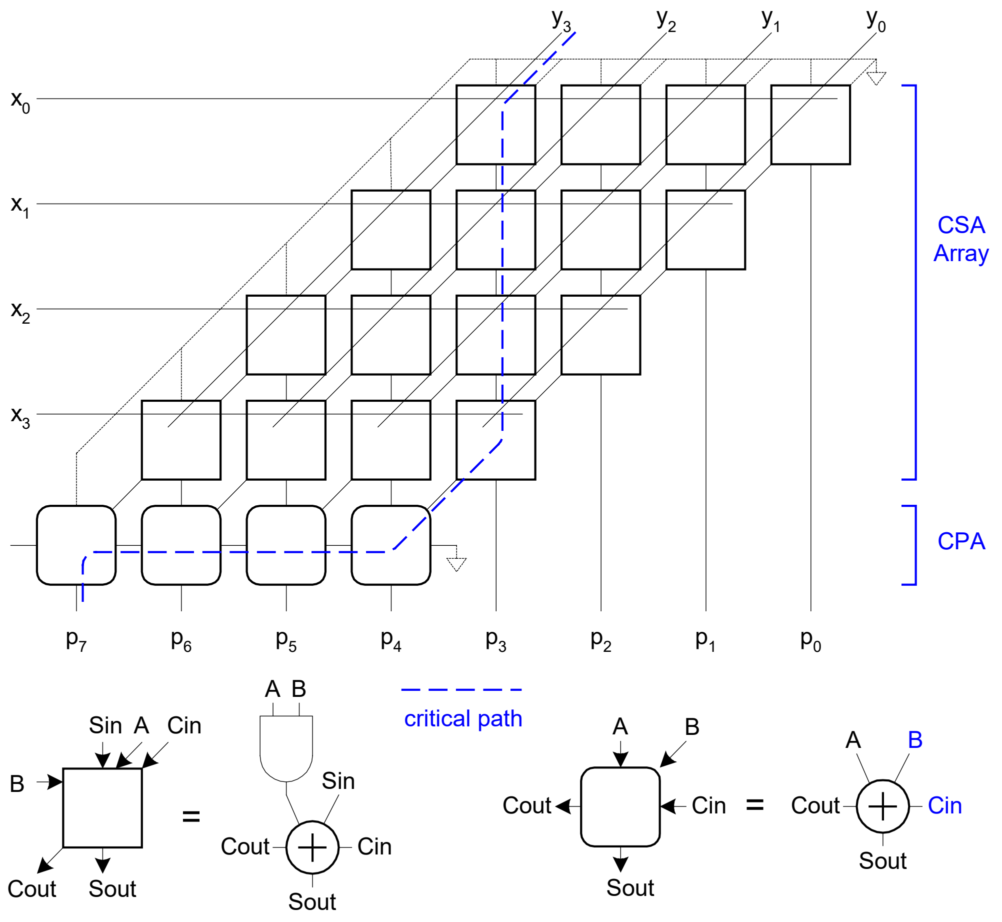
- A full adder sums 3 inputs and produces 2 outputs
  - Carry output has twice *weight* of sum output
- N full adders in parallel are called *carry save adder*
  - Produce N sums and N carry outs



X_4 Y_4 Z_4 X_3 Y_3 Z_3 X_2 Y_2 Z_2 X_1 Y_1 Z_1
C_4 S_4 C_3 S_3 C_2 S_2 C_1 S_1



Array Multiplier



```
    1100  :  12₁₀
    0101  :  5₁₀
    ----
    1100
   0000
  1100
 0000
--------
00111100  :  60₁₀
```

multiplicand

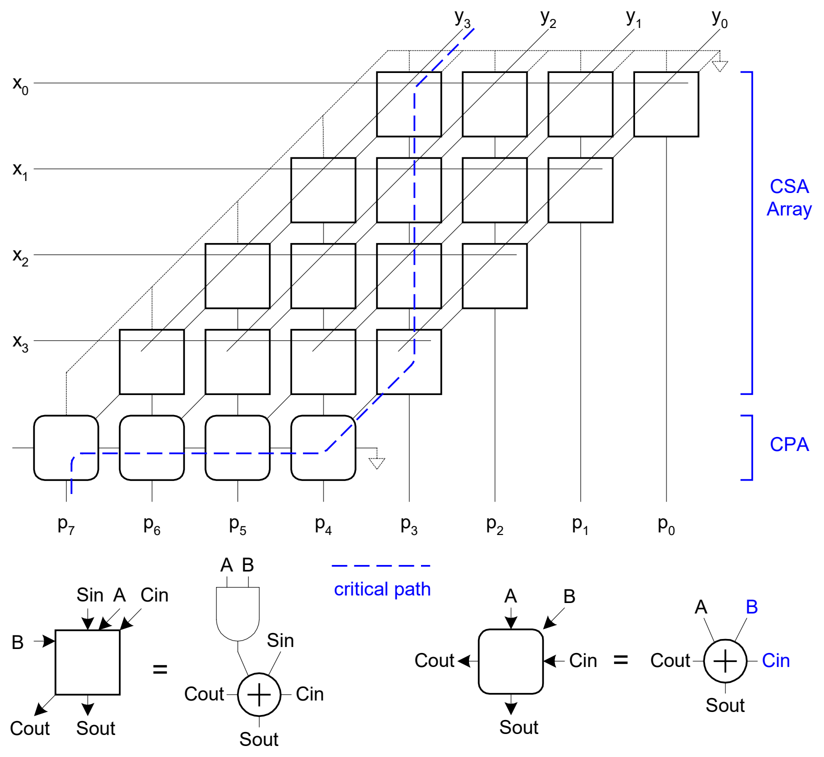
multiplier

partial products

product

Design

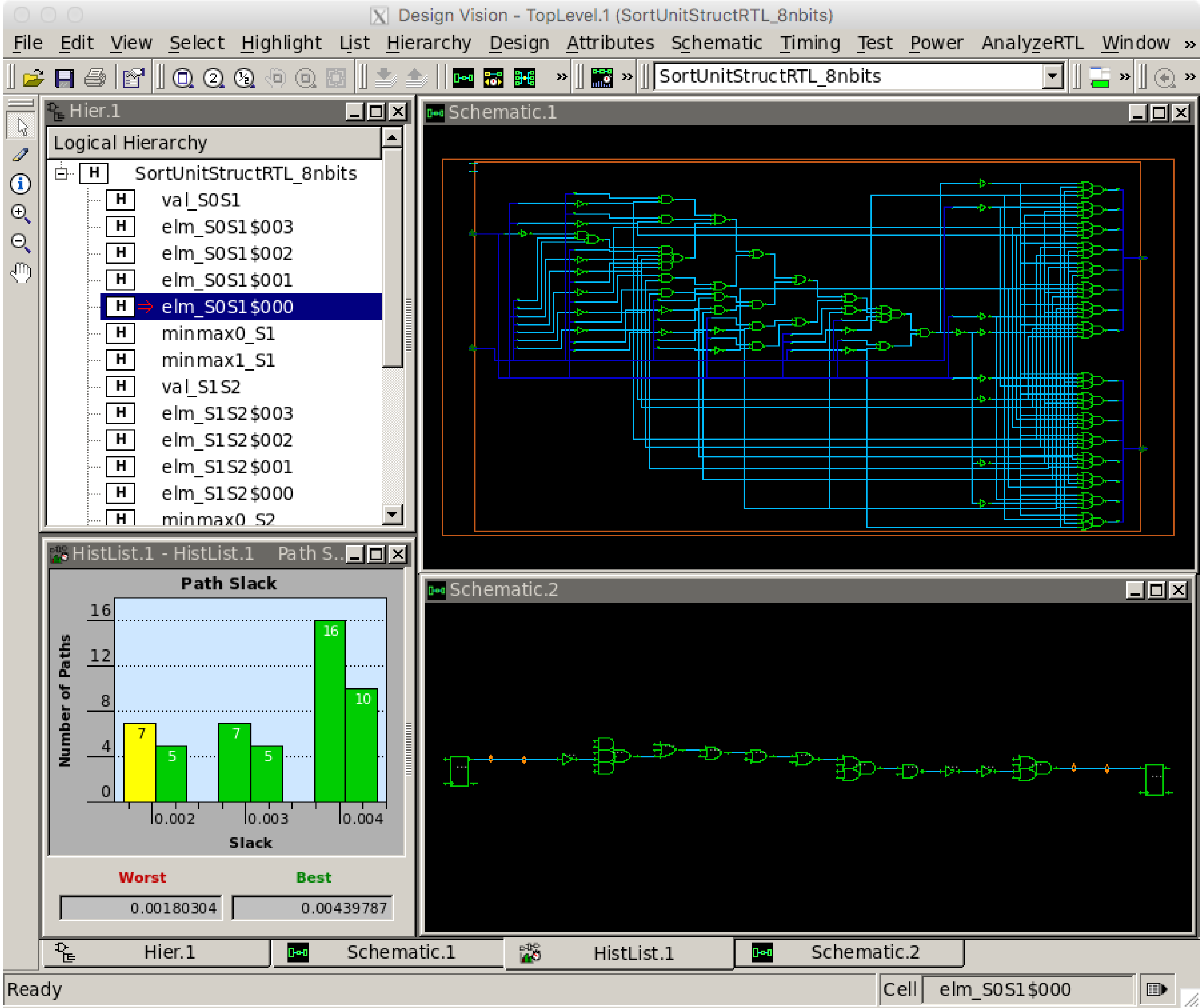
- Using Verilog, describe the digital logic of an unsigned array multiplier to calculate the unsigned product of two unsigned 4-bit numbers. Provide the full Verilog code in your report.



Simulation

- You have been provided with an exhaustive Verilog test bench to verify the functionality of your 4x4 multiplier.
- You can use NCVerilog (or Modelsim) to run the test bench with the following Linux command:
  ncverilog multiplier.v testbench.v (for NCVerilog)
  - Follow guidance in VLSI User Manual for Modelsim simulation
- You must include `timescale 1ns/1ps before your multiplier module declaration.
- Provide the output of the test bench dialogue in your report to show the functionality of your multiplier.

Synthesis


Design Vision - TopLevel.1 (SortUnitStructRTL_8nbits)
File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Window
SortUnitStructRTL_8nbits
Hier.1
Schematic.1
Schematic.2
HistList.1 - HistList.1 Path S...
Hier.1
Schematic.1
HistList.1
Schematic.2
Ready
Cell
elm_S0S1$000

Logical Hierarchy

- SortUnitStructRTL_8nbits
  - val_S0S1
  - elm_S0S1$003
  - elm_S0S1$002
  - elm_S0S1$001
  - elm_S0S1$000
  - minmax0_S1
  - minmax1_S1
  - val_S1S2
  - elm_S1S2$003
  - elm_S1S2$002
  - elm_S1S2$001
  - elm_S1S2$000
  - minmax0_S2

Path Slack

Number of Paths
16
12
8
4
0
7
5
7
5
16
10
0.002
0.003
0.004
Slack

Worst	Best
0.00180304	0.00439787

1

- Place the synthesis.tcl file in the same working directory as the Verilog test bench and your multiplier design.
- This TCL script will be launched using the Synopsys Design Compiler tool to synthesize your multiplier using the digital standard cells in the TSMC 65nm library.
- Run the Design Compiler GUI with the following Linux command: design_vision

2

- Perform the synthesis of your Verilog multiplier design by executing the synthesis script in Design Vision: File > Execute Script
- The compilation (compile_ultra) will take a few minutes to complete.
- Once complete, you will be able to view the gate-level schematic of your synthesized multiplier by right-clicking on the multiplier design in the Logical Hierarchy window and selecting schematic view.
- Save a copy of the schematic:
  - File > Print > Print to File (PDF).

3

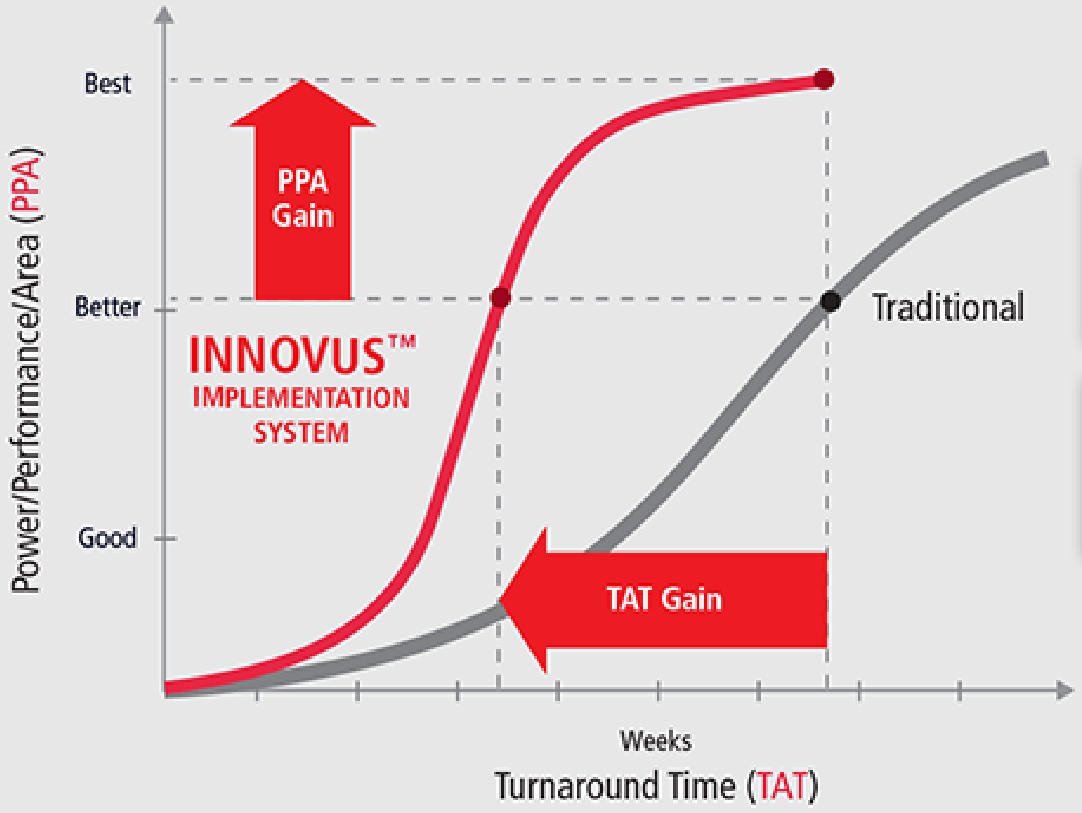
- ❑ The Synopsys Design Compiler tool will output the following files into the working directory:
  - Synthesized Verilog netlist: multiplier_syn.v
  - Area constraint file: multiplier_syn.sdc
  - Timing and Area reports
- ❑ In your report, provide the generated estimates for total area and critical path timing.
- ❑ If you were to clock the design, what would be the maximum clock frequency?
- ❑ In your report, include the schematic of the synthesized multiplier, and highlight the critical path based on the timing report.

4

- Verify that your synthesis was successful by running the multiplier test bench with the newly-synthesized netlist and the TSMC 65nm standard cell Verilog technology file:
  ncverilog testbench.v multiplier_syn.v tcbn65gplus.v
- Prior to running NCVerilog, ensure that all the module names in the generated Verilog files match.
- Provide the output of the NCVerilog test bench dialogue in your report.

Physical Implementation

Get Ahead of the Curve
with Innovus Implementation System



10-20%
Better PPA
(typical)

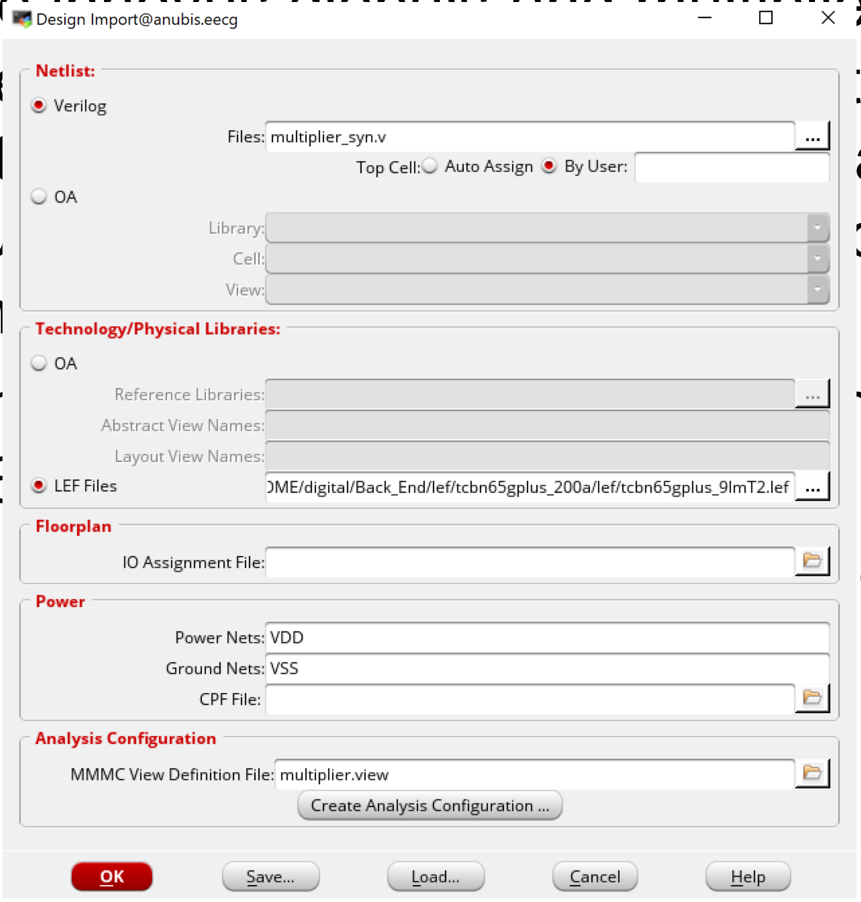
Up to 10X
Better TAT
(typical)

1

- Place the innovus.globals and multiplier.view file in the same working directory as the multiplier_syn.v and multiplier_syn.sdc files. Run the Cadence INNOVUS place & route tool with the following command: innovus
- Set up the design: File > Import Design > Load > innovus.globals
- This will load the appropriate timing and physical cell libraries for the place & route.

1

- Place the innovus globals and multiplier.view file in the same iplier_syn.v and mult adence INNOVU ollowing comman
- Set up th n > Load > innovus3
- This will d physical cell libraries


Design Import@anubis.eecg

Netlist:

[● Verilog]

Files: multiplier_syn.v

Top Cell: ○ Auto Assign [● By User:]

○ OA

Library:

Cell:

View:

Technology/Physical Libraries:

○ OA

Reference Libraries:

Abstract View Names:

Layout View Names:

[● LEF Files])ME/digital/Back_End/lef/tcbn65gplus_200a/lef/tcbn65gplus_9lmT2.lef

Floorplan

IO Assignment File:

Power

Power Nets: VDD

Ground Nets: VSS

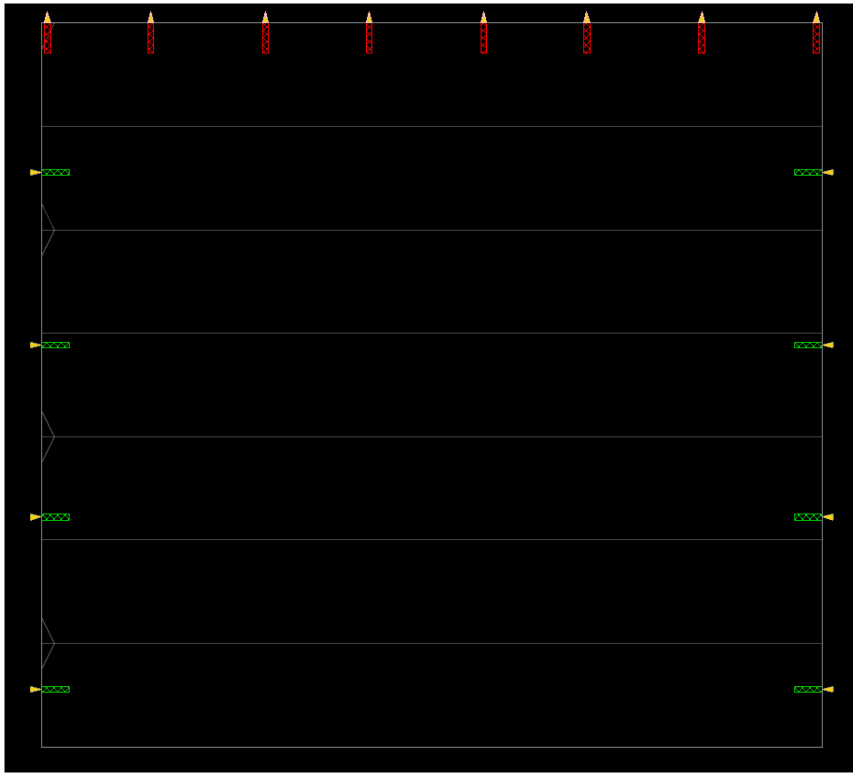
CPF File:

Analysis Configuration

MMMC View Definition File: multiplier.view

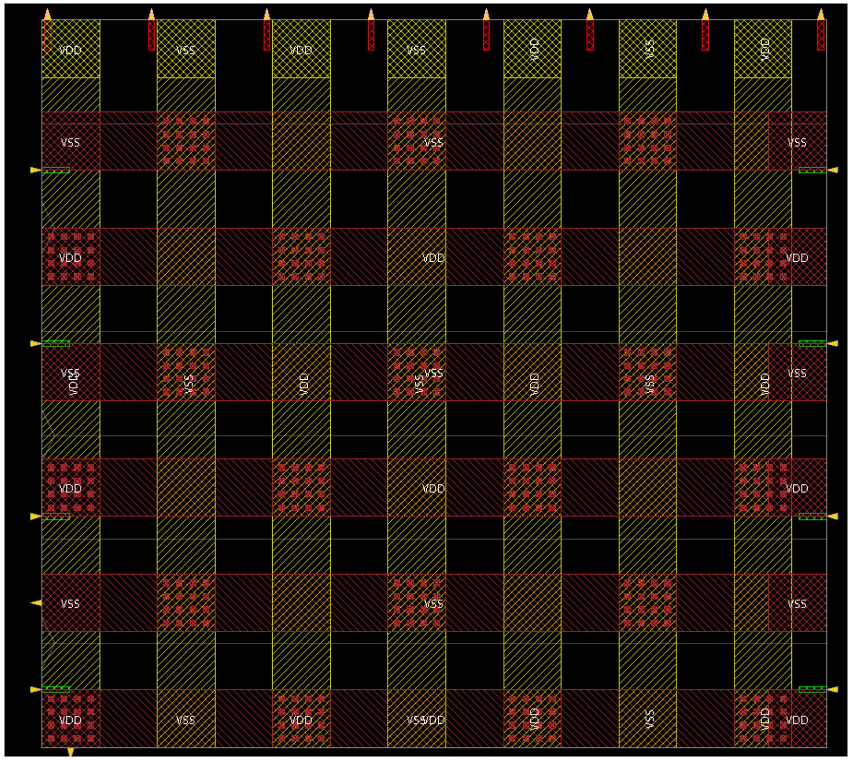
2

- Define Floorplan
- Use Edit > Pin Editor
  - Place A[] on left
  - Place B[] on right
  - Place product[] on top

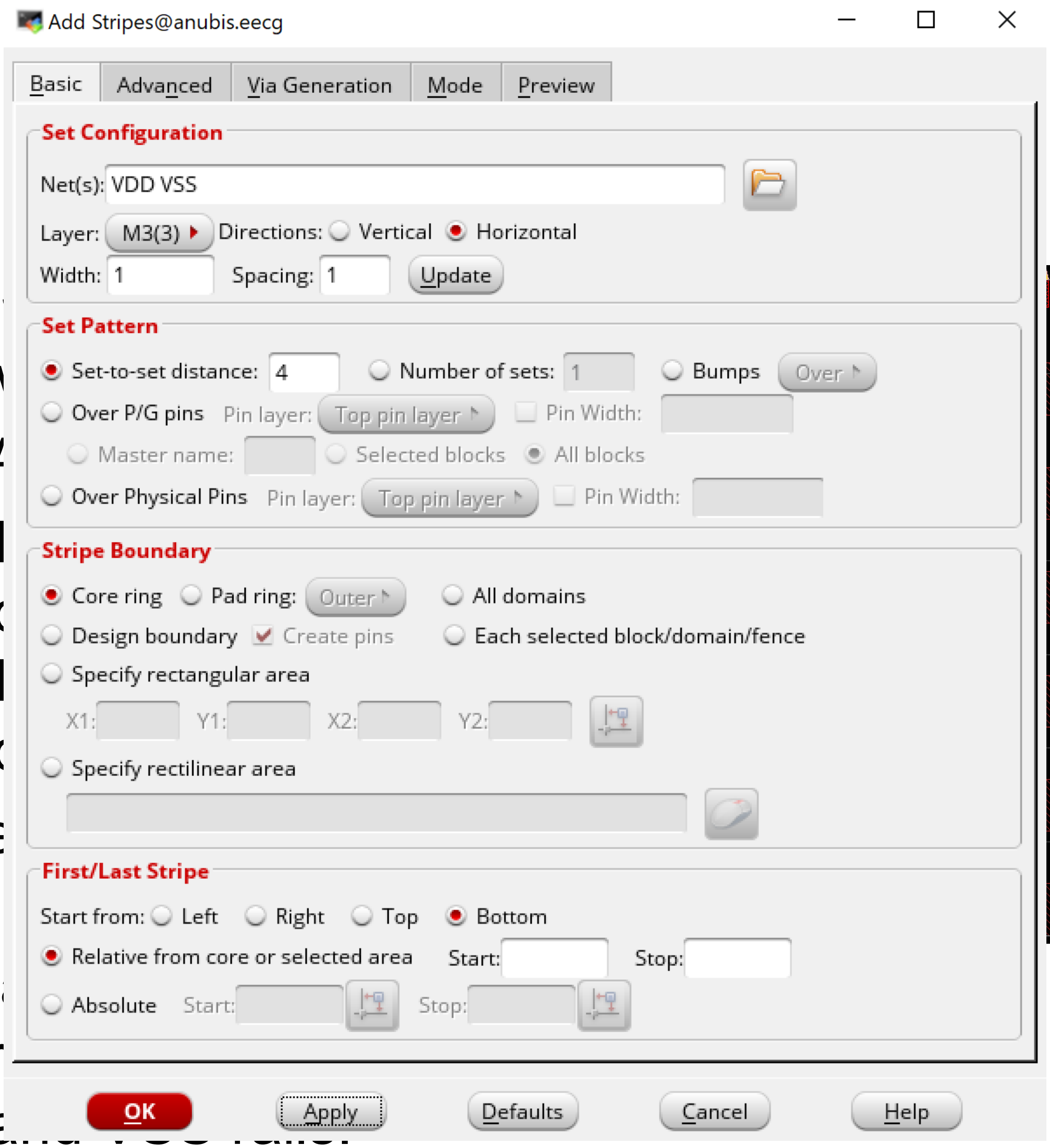


2

- Place power stripes for VDD and VSS:
  - Power > Power Planning
    > Add Stripe
  - Select the M3 Layer with: Width: 1 and Spacing: 1
    Select the M4 Layer with: Width:1 and Spacing: 1
  - Press Apply and OK.
  - Set-to-set distance = 4
- You have now set up the power connections for the VDD and VSS rails.



- ❑ Place
  and V
  - – Pov
    - > A
  - – Sel
    Wic
    Sel
    Wic
  - – Pre
- ❑ You h
  power
  VDD a


Add Stripes@anubis.eecg

Basic | Advanced | Via Generation | Mode | Preview

Set Configuration

Net(s): VDD VSS

Layer: M3(3) Directions: Vertical [✓ Horizontal]

Width: 1 Spacing: 1 Update

Set Pattern

[✓ Set-to-set distance: 4] Number of sets: 1 Bumps Over

Over P/G pins Pin layer: Top pin layer Pin Width:

Master name: Selected blocks [✓ All blocks]

Over Physical Pins Pin layer: Top pin layer Pin Width:

Stripe Boundary

[✓ Core ring] Pad ring: Outer All domains

Design boundary [✓ Create pins] Each selected block/domain/fence

Specify rectangular area

X1: Y1: X2: Y2:

Specify rectilinear area

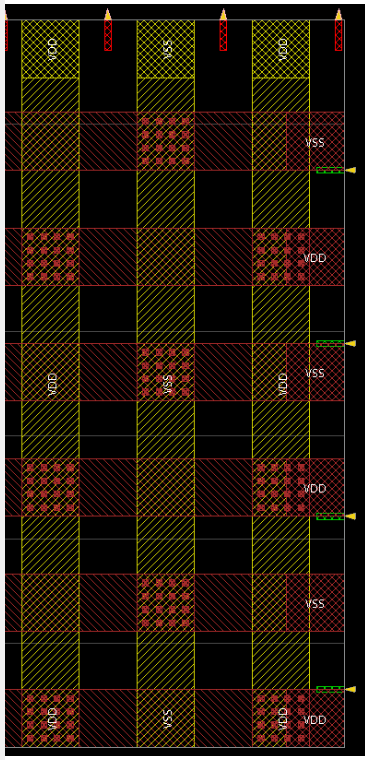
First/Last Stripe

Start from: Left Right Top [✓ Bottom]

[✓ Relative from core or selected area] Start: Stop:

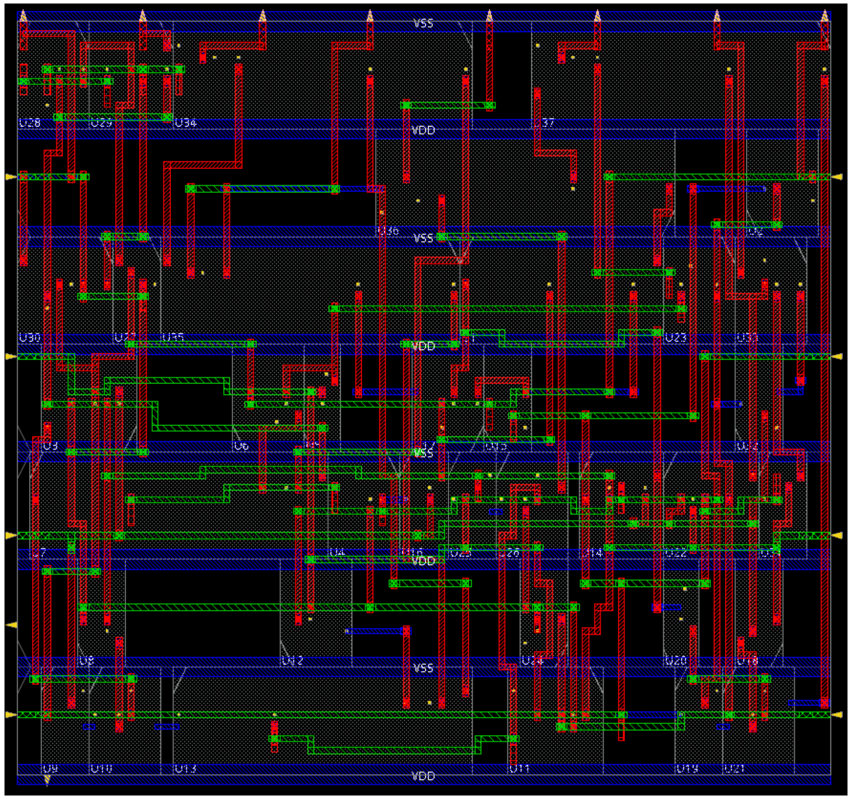
Absolute Start: Stop:

OK Apply Defaults Cancel Help



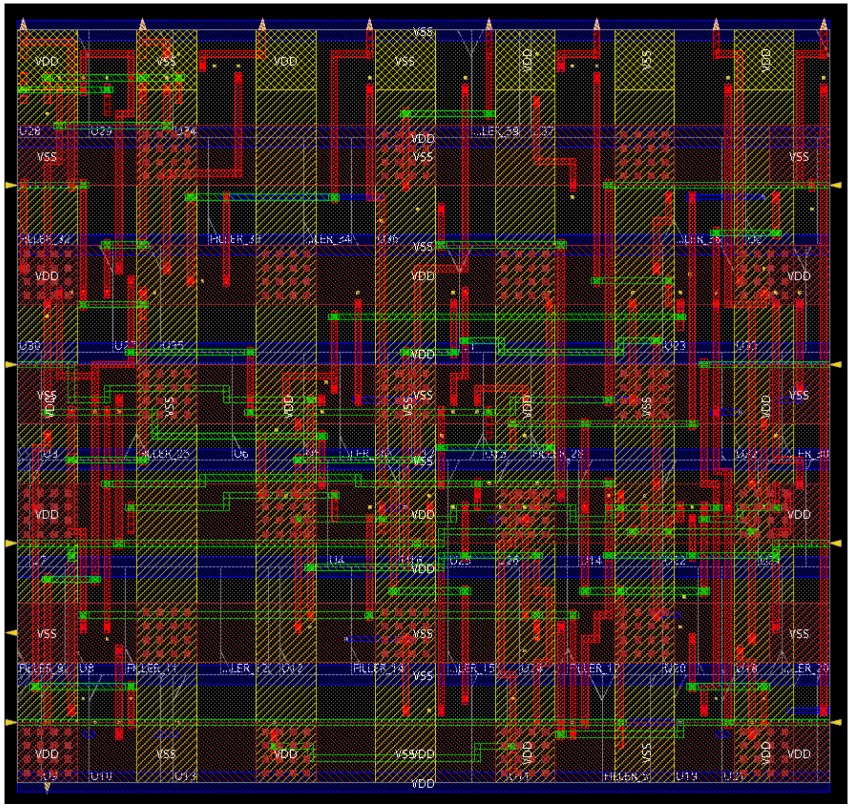
3

- Place the standard cells:
  - Place > Place Standard Cell > OK
- Once complete, you can view the placed cells:
  - Place > Display > Display Spare Cell
- Verify that the design does not have any geometry errors:
  - Verify > Verify Geometry > OK

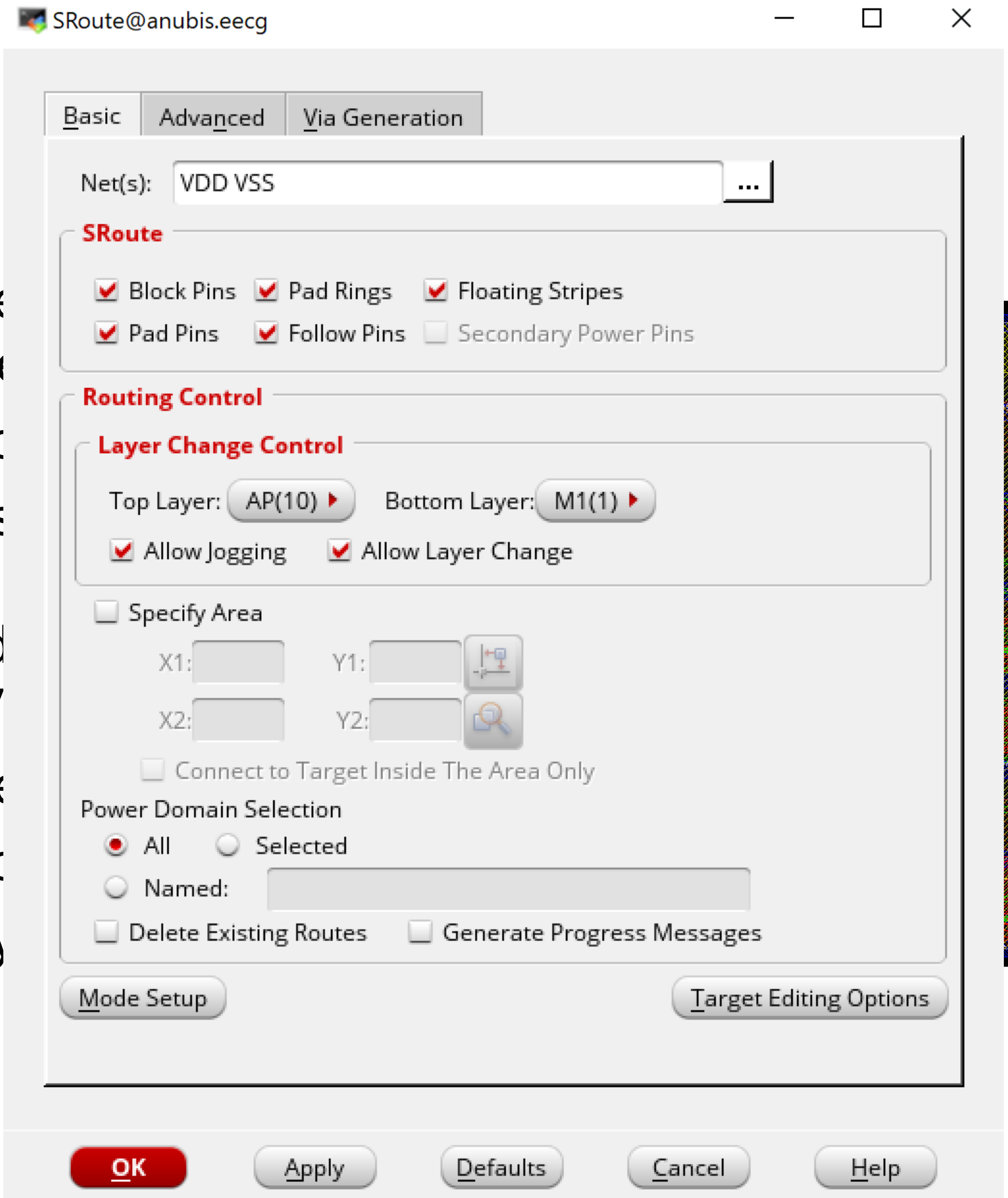


4

- Route the power connections:
  - Route > Special Route
- You should now see VDD and VSS rails of standard cells connected to the VDD and VSS stripes
- Route the signal connections:
  - Route > NanoRoute
- Verify > Verify Geometry



- ❑ Route
  conn
  – Ro
- ❑ You s
  VDD
  stand
  the V
- ❑ Route
  – Ro
- ❑ Verify


SRoute@anubis.eecg

Basic | Advanced | Via Generation

Net(s): VDD VSS

SRoute

- [✓] Block Pins
- [✓] Pad Rings
- [✓] Floating Stripes
- [✓] Pad Pins
- [✓] Follow Pins
- [ ] Secondary Power Pins

Routing Control

Layer Change Control

Top Layer: AP(10) Bottom Layer: M1(1)

- [✓] Allow Jogging
- [✓] Allow Layer Change

- [ ] Specify Area

X1: Y1:
X2: Y2:

- [ ] Connect to Target Inside The Area Only

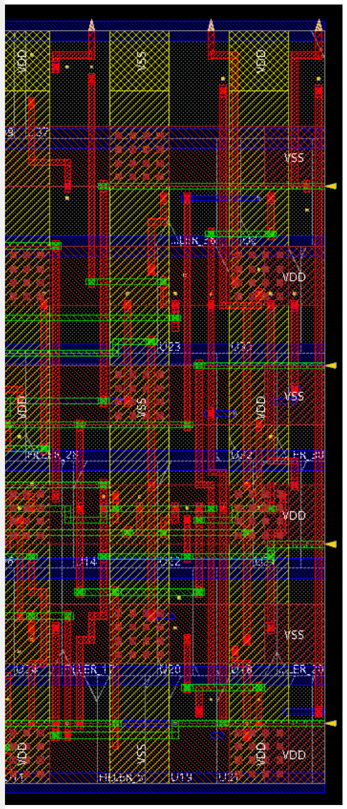
Power Domain Selection

[✓ All] Selected
Named:

- [ ] Delete Existing Routes
- [ ] Generate Progress Messages

Mode Setup Target Editing Options

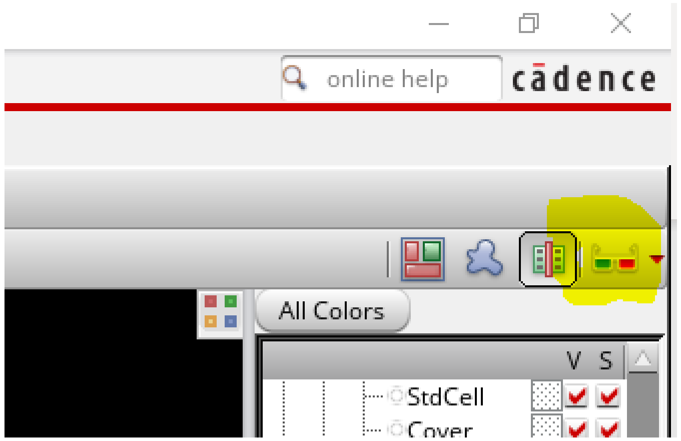
OK Apply Defaults Cancel Help

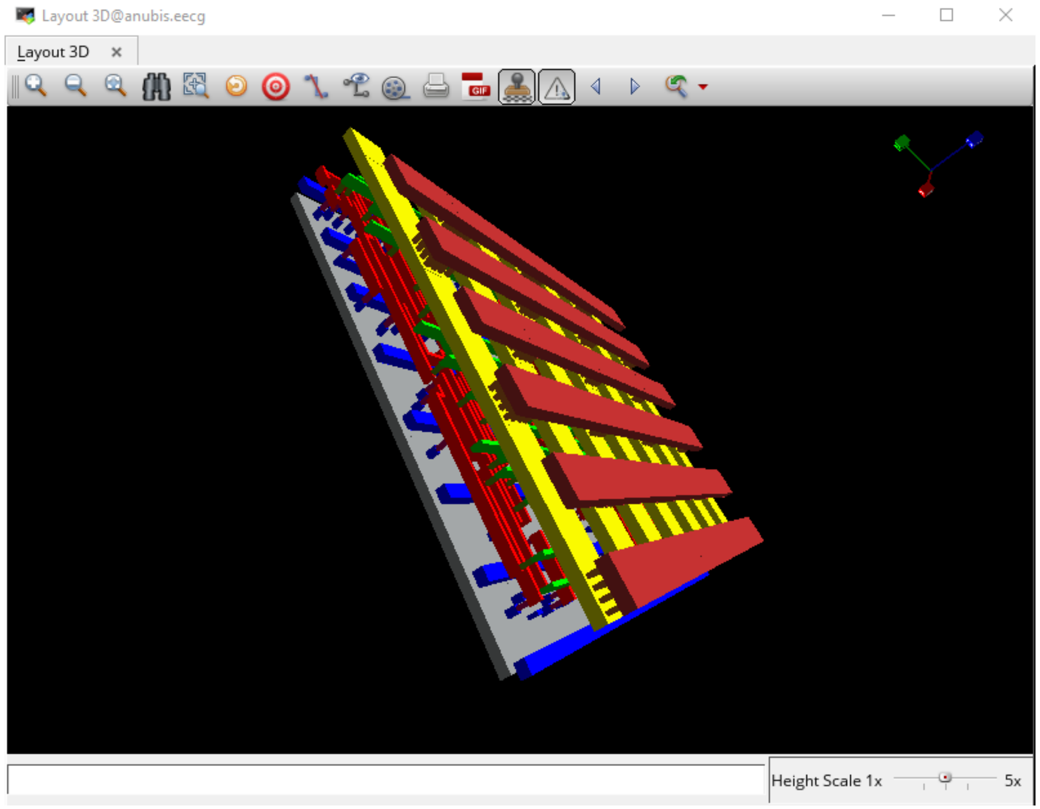


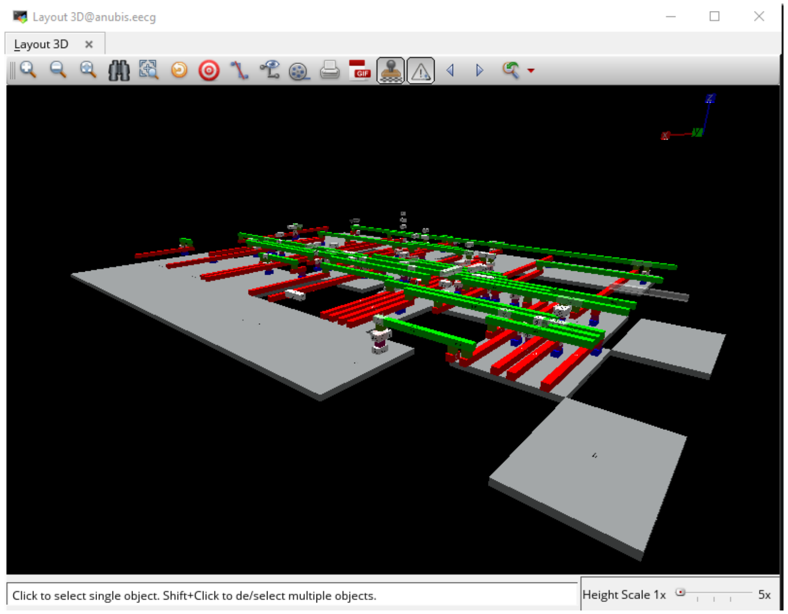
5

- ❑ Add filler cells to fill the unused area of your design:
  - – Place > Physical Cell > Add Filler
  - – Under Cell Name(s), select all the available filler cells from the cell list.
- ❑ Verify that the design does not have any geometry errors:
  - – Verify > Verify Geometry > OK

- A cool feature to see congestion







Click to select single object. Shift+Click to de/select multiple objects.

6

- Perform a power analysis:
  - Power > Power Analysis > Run
- Perform static power analysis for a constant input activity factor of 0.2, over the frequency range of 100MHz to 1GHz.
- Plot your results for total internal power and total switching power vs. frequency on the same graph.
- Comment on these results with respect to the synthesized timing results obtained in Part 2.2.
- What do the power analysis estimates suggest is the maximum clock frequency?

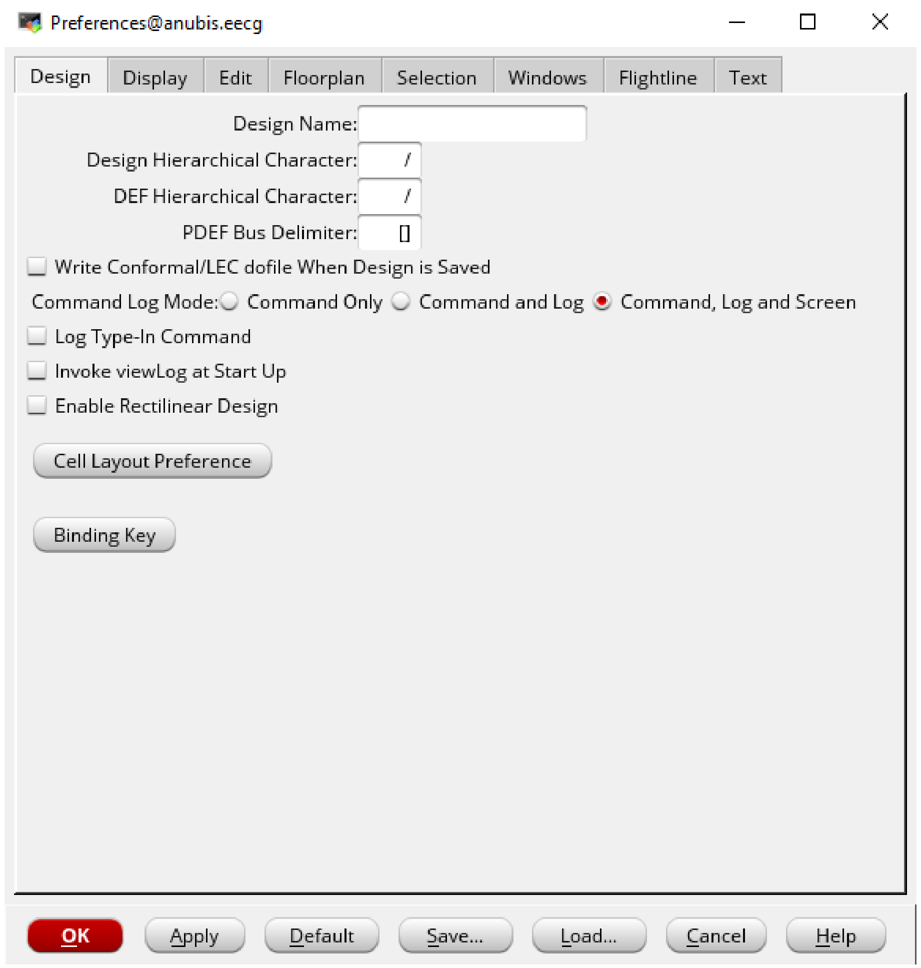
7

- Obtain the total area of your placed & routed design: File > Report > Summary
  Compare the resulting total core area after place & route with the estimated synthesis area from Part 2.2. Comment on the discrepancy.

8

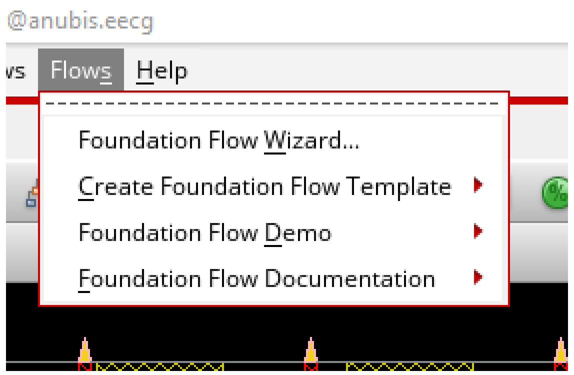
- ❑ Export the Verilog netlist of the placed & route design: File > Save > Netlist
- ❑ Save the file as multiplier_netlist.v, and verify that your synthesis was successful by running the multiplier test bench with the newly-synthesized netlist and the TSMC 65nm standard cell Verilog technology file:
  ncverilog testbench.v multiplier_netlist.v tcbn65gplus.v
- ❑ Provide the output of the NCVerilog test bench dialogue in your report.

Some Tips


Preferences@anubis.eecg

Design | Display | Edit | Floorplan | Selection | Windows | Flightline | Text

Design Name:
Design Hierarchical Character: /
DEF Hierarchical Character: /
PDEF Bus Delimiter: []
☐ Write Conformal/LEC dofile When Design is Saved
Command Log Mode: ○ Command Only ○ Command and Log [✓ Command, Log and Screen]
☐ Log Type-In Command
☐ Invoke viewLog at Start Up
☐ Enable Rectilinear Design


@anubis.eecg
Flows
Help
Foundation Flow Wizard...
Create Foundation Flow Template
Foundation Flow Demo
Foundation Flow Documentation

END