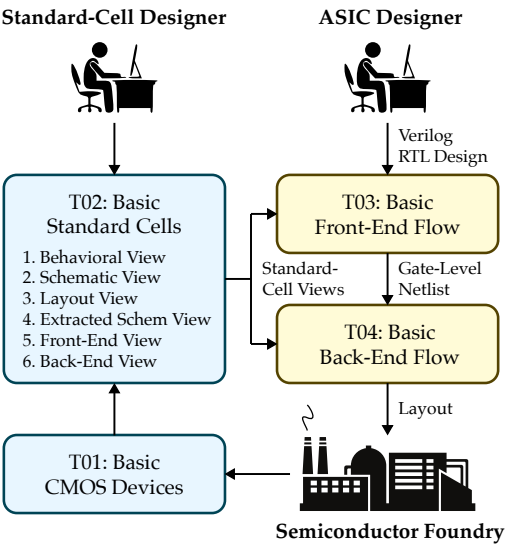
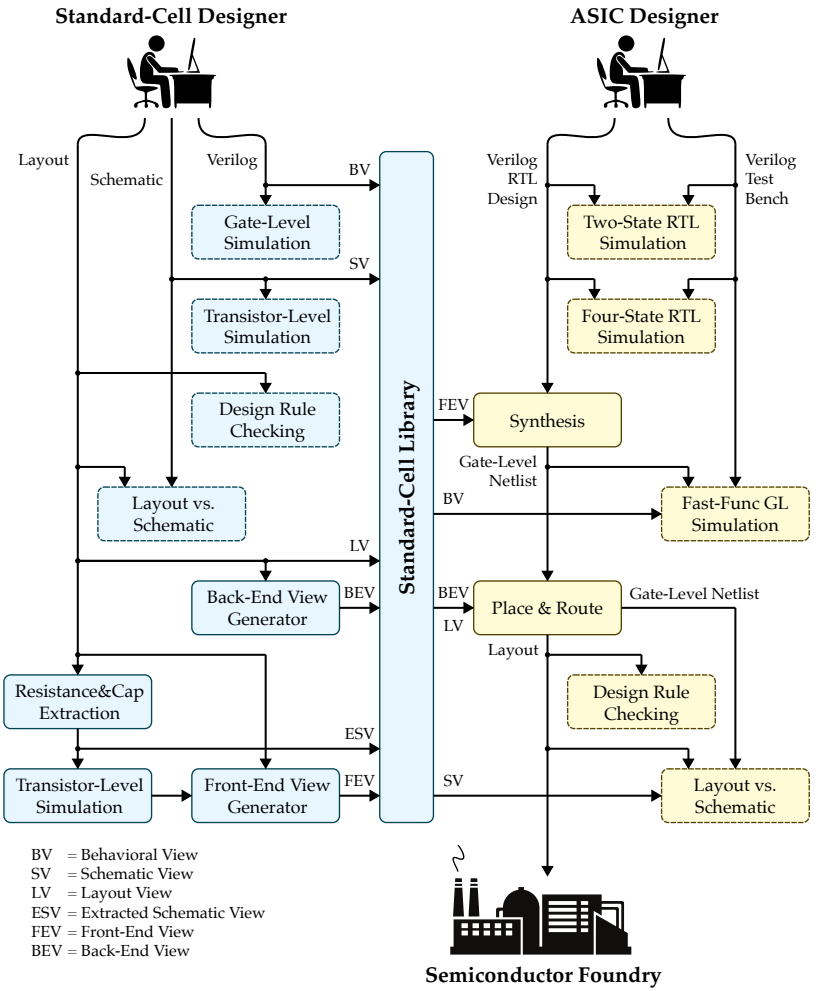
ECE 6745 Complex Digital ASIC Design
Topic 4: Basic Back-End Flow

School of Electrical and Computer Engineering
Cornell University

revision: 2026-02-12-12-43

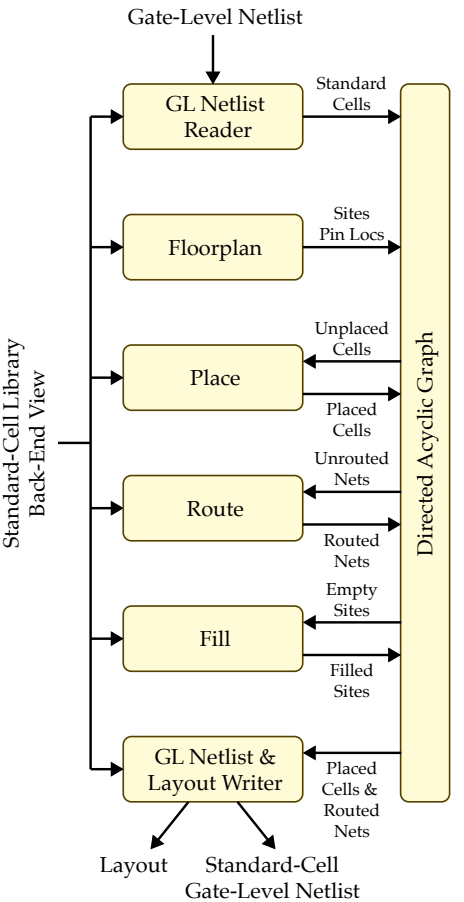


1. Back-End Flow



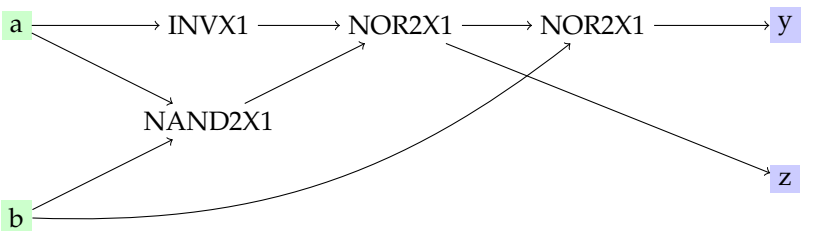
2. Place-and-Route

- Place-and-route takes as input a standard-cell gate-level netlist, places cells, routes nets, and produces the final layout
- Place-and-Route Data Structure
  - Graph of standard-cell nodes
- Place-and-Route Algorithms
  - GL Netlist Reader: Parse gate-level netlist into graph of standard-cell nodes
  - Floorplan: Creates grid of sites and position input/output pins
  - Place: Places each standard cell on the grid of sites
  - Route: Routes each net on the routing grid
  - Filler: Fills in empty space using FILL standard cells
  - Layout & GL Netlist Writer: Outputs the final layout along with an updated standard-cell gate-level netlist



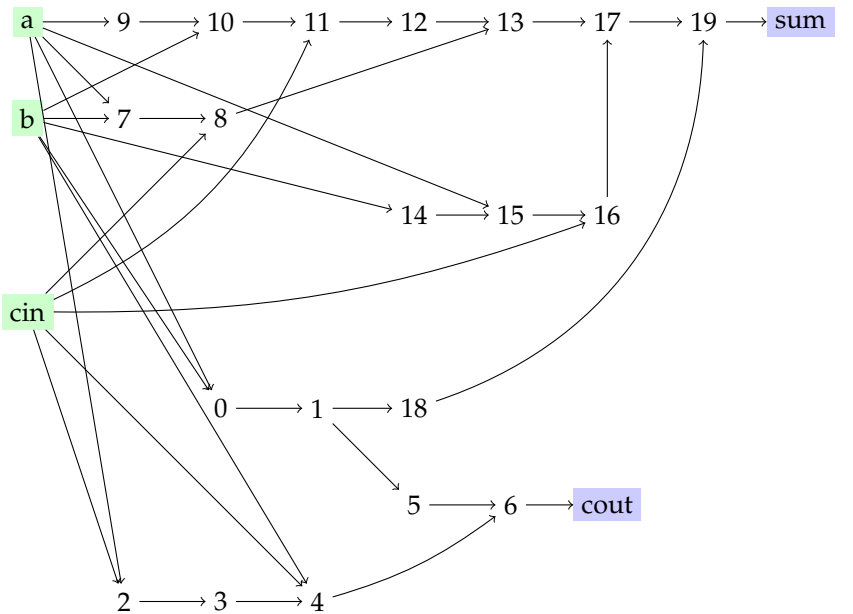
2.1. *Data Structure:* Directed Acyclic Graph

- List of input ports and output ports
- Directed Acyclic Graph
  - Nodes represent unplaced or placed standard cells
  - Edges represent unrouted or routed nets between standard cells
  - Multi-pin nets (i.e., fanout) are allowed
  - Cycles are not allowed



2.2. *Algorithm:* Gate-Level Netlist Reader

- Directed acyclic graph for full adder

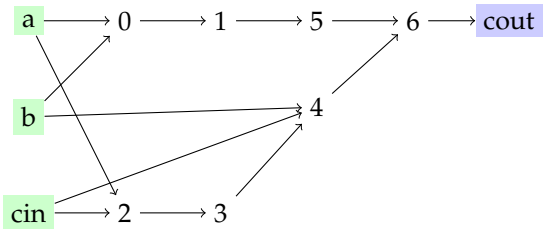


0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1

7	NOR2X1
8	NAND2X1
9	INVX1
10	NAND2X1
11	NOR2X1
12	INVX1
13	NAND2X1

14	INVX1
15	NAND2X1
16	NOR2X1
17	NOR2X1
18	NAND2X1
19	NAND2X1

- We will focus on just the cout logic for this topic



0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1

2.3. *Algorithm:* Floorplan

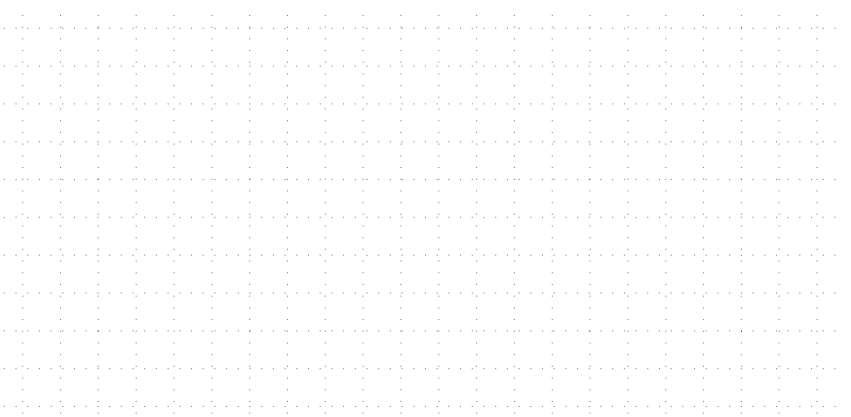
- Goal is to create a grid of sites and to position input/output pins
  - Each *site* is a one standard cell height tall and one routing track wide
  - Standard cells must be placed such that they align to sites

Fixed Floorplan

- Width, height, and input/output pins are given

Automatic Floorplan

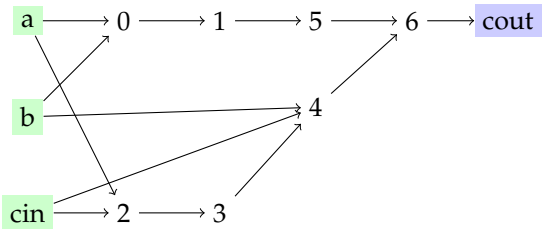
- Standard cell density and aspect ratio are given
- Calculate total area from gate-level netlist
- Divide by cell density to get total floorplan area
- Use aspect ratio to determine width and height
- Automatically position input and output pins along perimeter



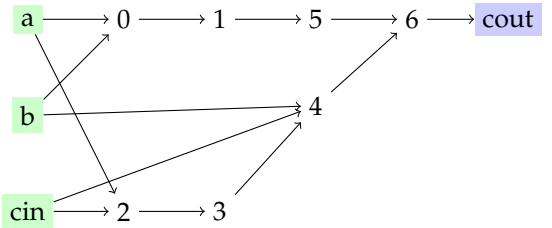
2.4. *Algorithm:* Simulated Annealing Placement

- Use widest standard cell to create a regular *placement grid*
- Randomly place cells in the placement grid
- Calculate placement cost using half-perimeter wire length (HPWL)
  - Compute bounding box of all pins in net
  - HPWL is the width + height of this bounding box
- Randomly choose one standard cell
- Randomly choose a target placement grid location
  - If target placement grid location is empty, *move* standard cell
  - If target placement grid location is not empty, *swap* standard cells
- Calculate new cost
  - If new cost is less than previous cost keep move/swap
  - If new cost is greater than previous cost do not keep move/swap
  - ... occassionally keep bad move/swaps to avoid local minimum
- Key to simulated annealing is how we occassionally keep bad move/swaps
  - Keep track of a "temperature" which gradually decreases (i.e.,g cools)
  - Propability of keeping a bad move/swap is proportional to temperature
  - Usually use $p = \exp\left(-\frac{\Delta c}{T}\right)$
  - Enables more exploration early on and as temperature gradually cools the placement settles into a good solution

- Random initial placement



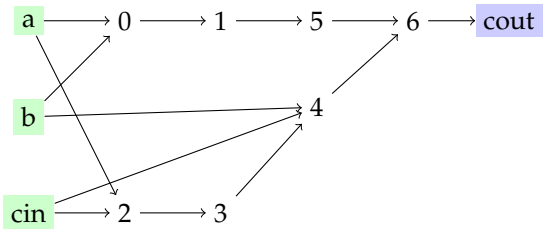
0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1



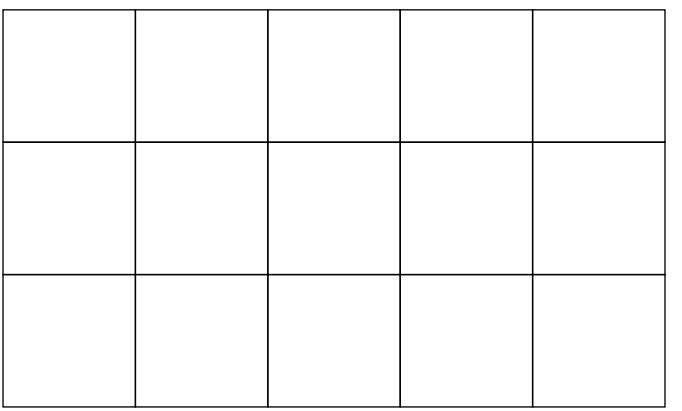
0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1

step	pair	Δc	T	$\exp\left(-\frac{\Delta c}{T}\right)$	action	c
0	$6 \leftrightarrow (4,1)$					
1	$1 \leftrightarrow 5$					
2	$3 \leftrightarrow (0,2)$					
3	$2 \leftrightarrow (3,2)$					
4	$6 \leftrightarrow (0,1)$					
5						

- Final placement

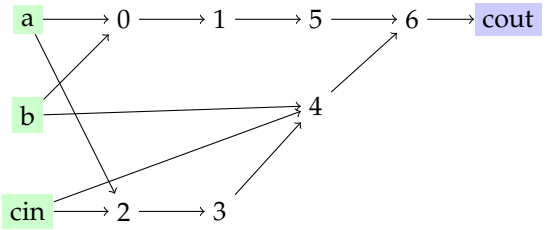


0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1

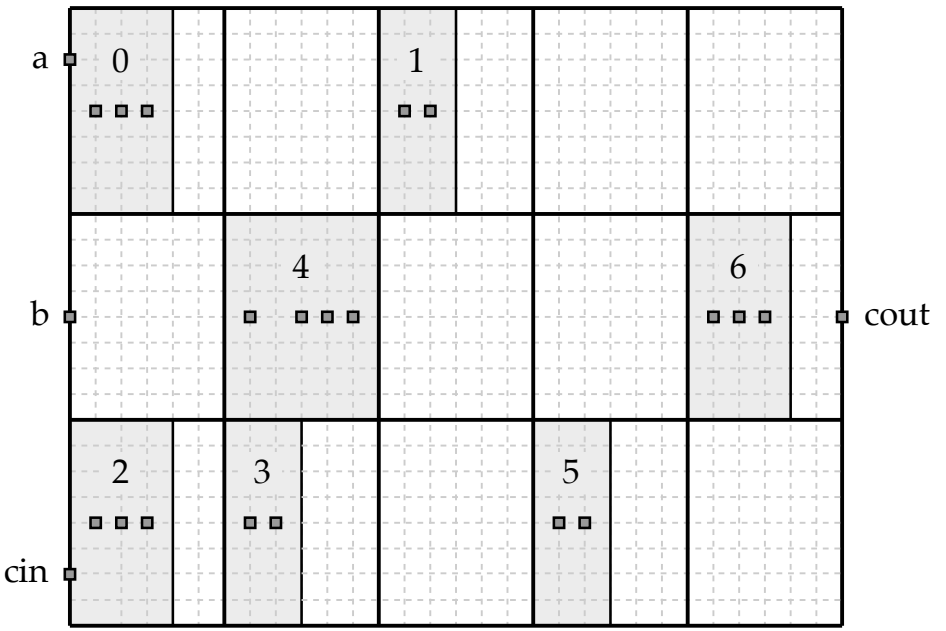


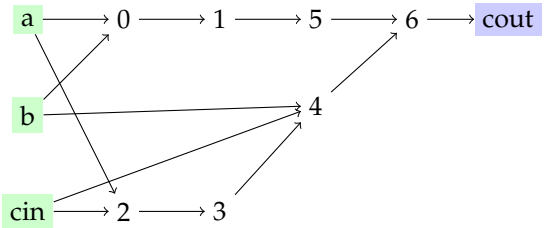
2.5. *Algorithm:* Maze Routing

- Promote pins from metal 1 to metal 2 since all routing happens on metal 2 and higher
- Sort all nets by their half-perimeter wire length (HPWL)
  - Compute bounding box of all pins in net
  - HPWL is the width + height of this bounding box
- For each net
  - Pick first two pins
  - Start from "source" pin
  - Fill in 3D routing grid with distance from source pin
  - Must route around obstacles!
  - When reach "target" pin, backtrace to recover shortest path
  - Pick next pin as the "source" pin
  - Now the target is actually the route so far (a tree)
- If a net is routable
  - Commit the route
  - Mark every grid point along the route as occupied
- If a net is unroutable
  - Place unroutable net first in the list of nets
  - Start routing process again from scratch

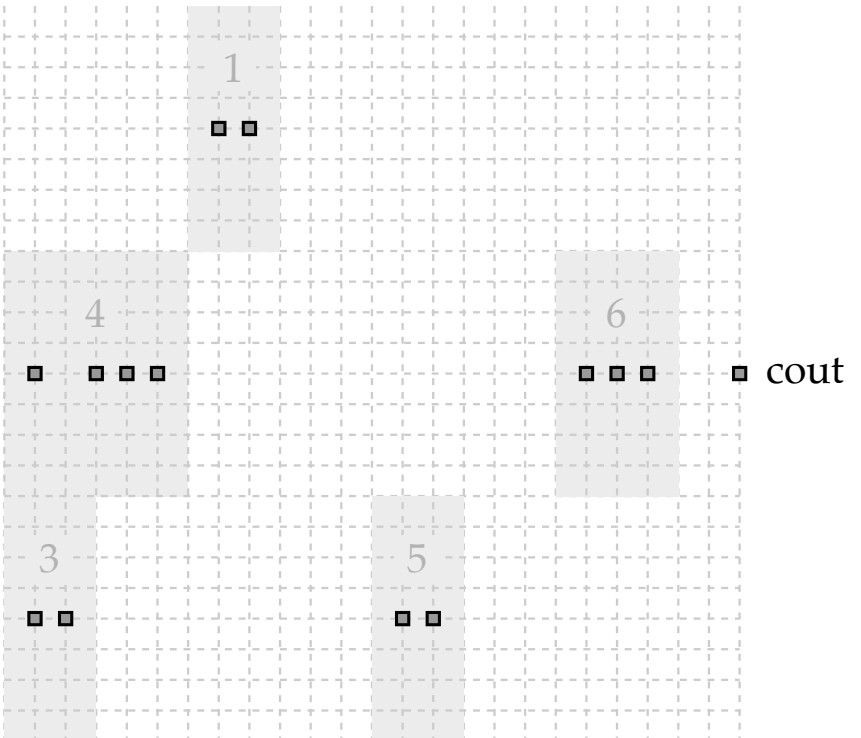


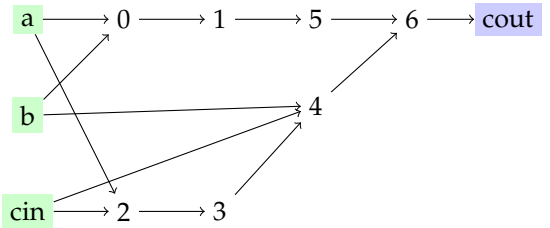
0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1



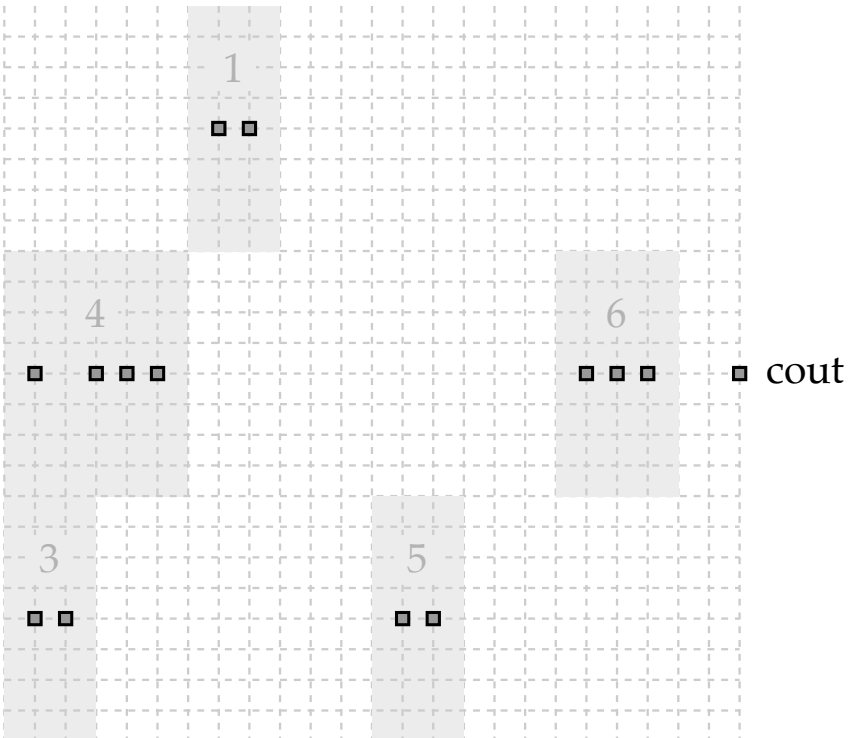


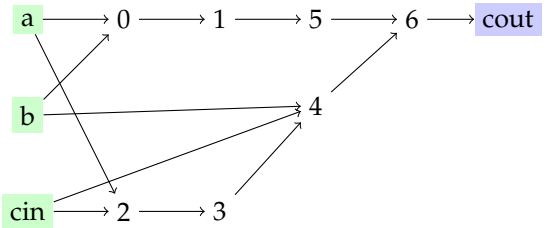
0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1



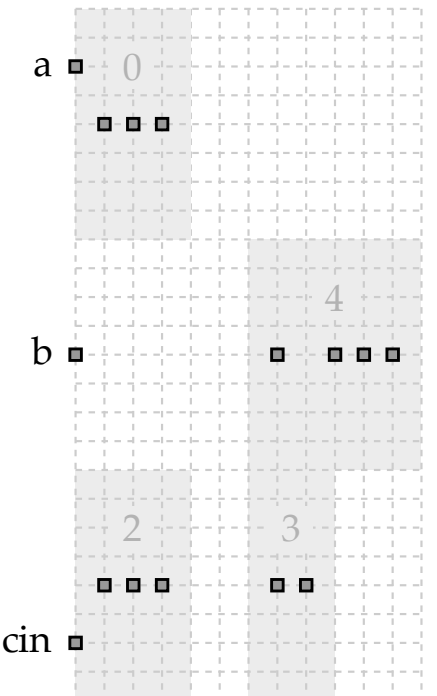


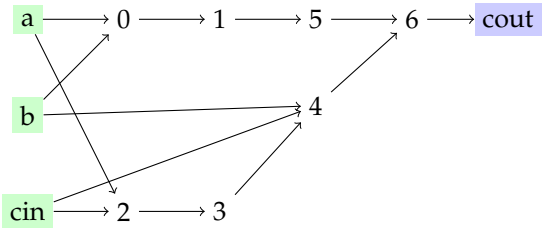
0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1



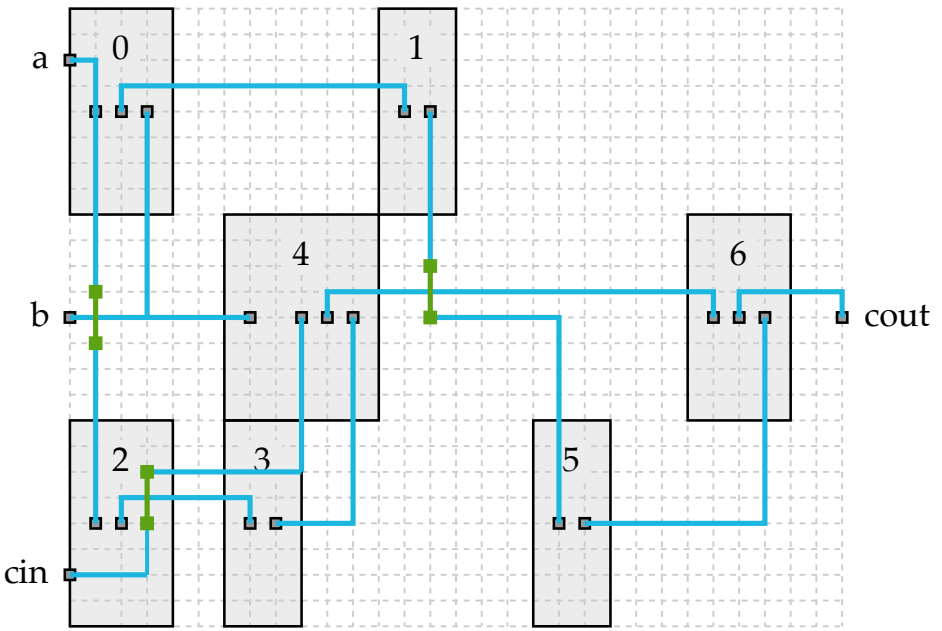


0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1



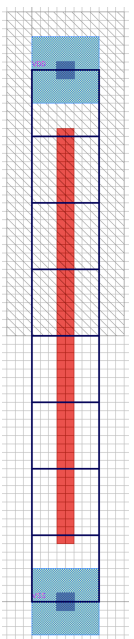


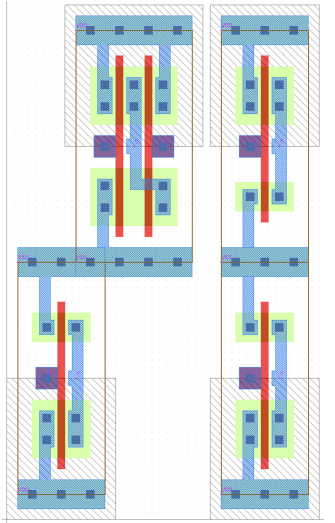
0	NAND2X1
1	INVX1
2	NAND2X1
3	INVX1
4	AOI21X1
5	INVX1
6	NAND2X1

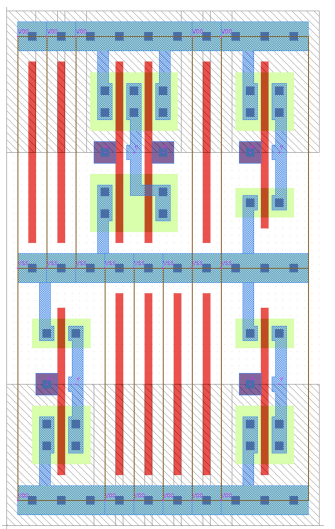


2.6. *Algorithm:* Fill

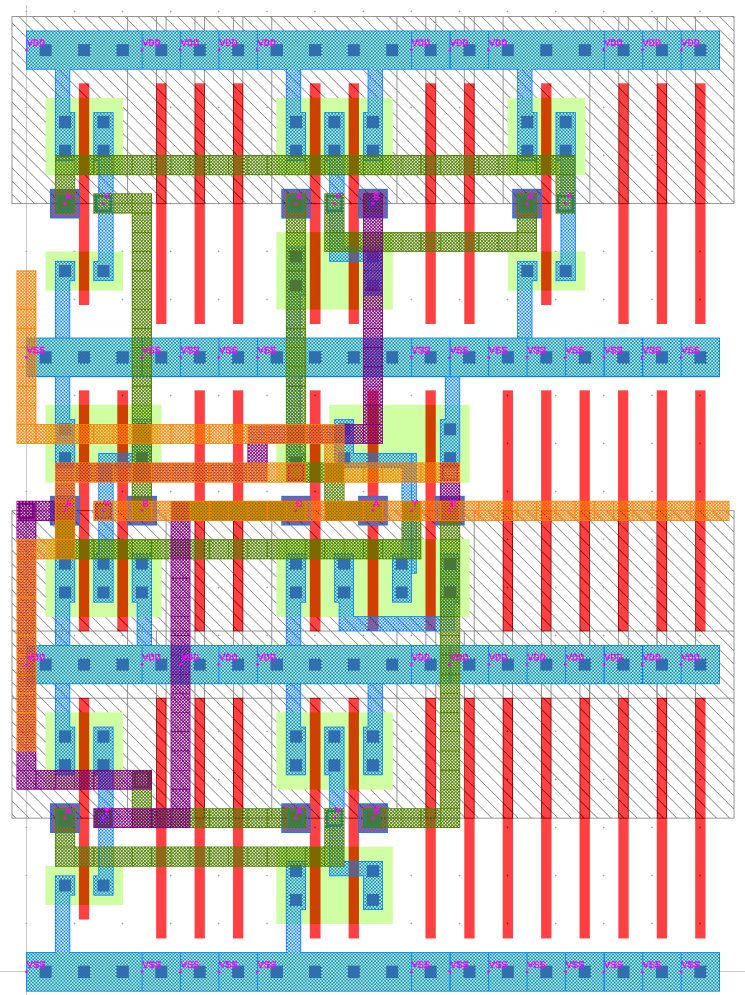
- Standard cell for filling in empty space along each row
- Needs to connect the power rail, ground rail, and n-well
- Neesd to include substrate and n-well contacts
- Needs polysilicon to satisfy polysilicon density design rules



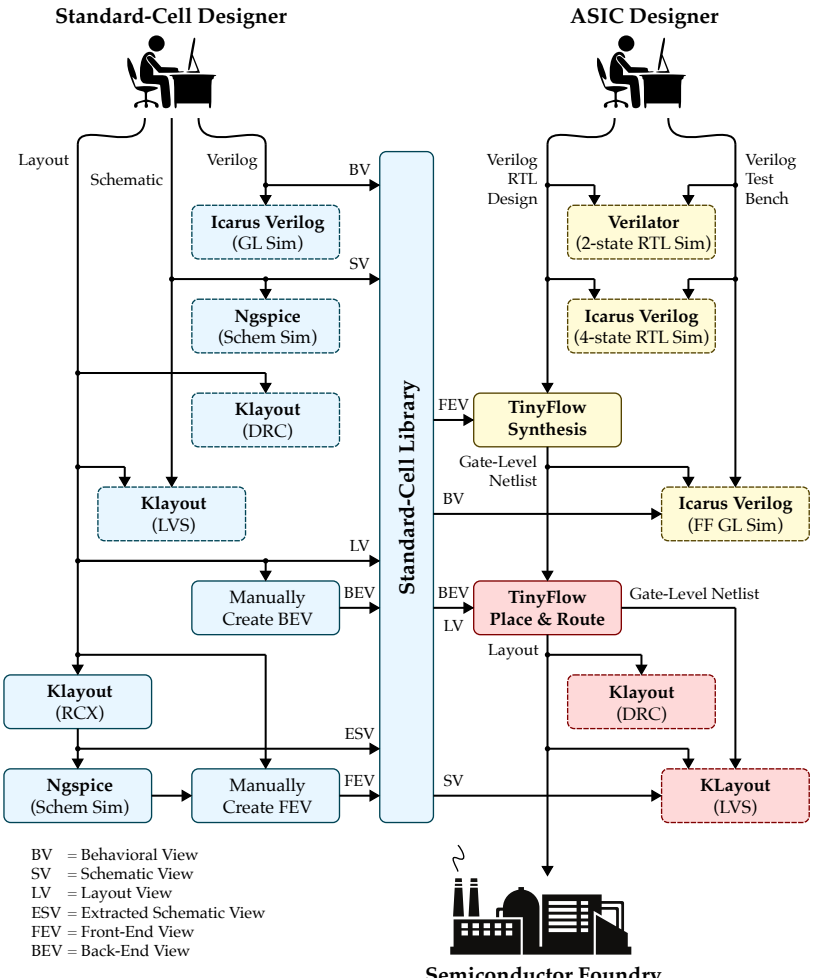




2.7. *Algorithm:* Gate-Level Netlist & Layout Writer



3. Tiny Back-End Flow



4. Summary of Basic ASIC Flow

```
1  module FA
2  (
3    input  wire a,
4    input  wire b,
5    input  wire cin,
6    output wire cout,
7    output wire sum
8  );
9
10   wire g;
11   assign g = a & b;
12
13   assign cout = (a & cin)  | (b & cin) | g;
14
15   assign sum = (~a & ~b &  cin)
16              | (~a &  b & ~cin)
17              | ( a & ~b & ~cin)
18              | ( g      &  cin);
19
20 endmodule
```