CAL POLY
SAN LUIS OBISPO

EE 531: ADVANCED VLSI DESIGN

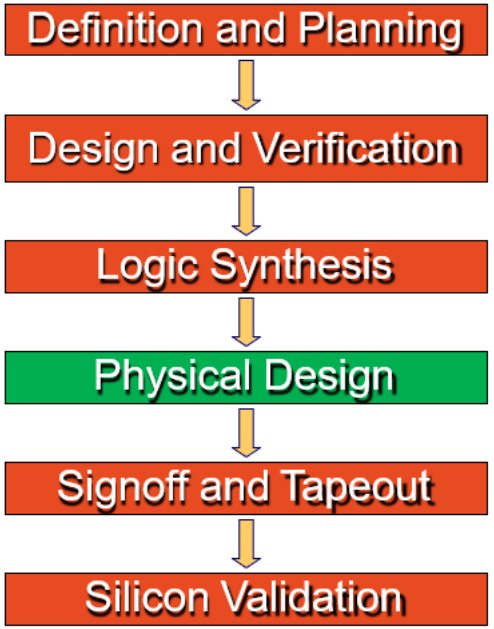
Introduction to Physical Design

Nishith N. Chakraborty

February, 2025

AUTOMATED DESIGN STEPS: NEXT STEPS

- To start, we will move between tools with a logical approach to ones with a physical approach to design implementation.
- Then, we will make a physical foundation for our design by drawing up a floorplan.
  - ➢ This will include making decisions where “big” or “important” pieces will sit, such as IPs, I/Os, Power grids, special routes, etc.
- After that, we can place our gates taking into account congestion and timing.
- With our flip-flops in place, we can go about designing a clock-tree.
- And finally, we can route all our nets, according to DRCs, timing, noise, etc.
- Before tapeout, we will clean things up, verify, etc.



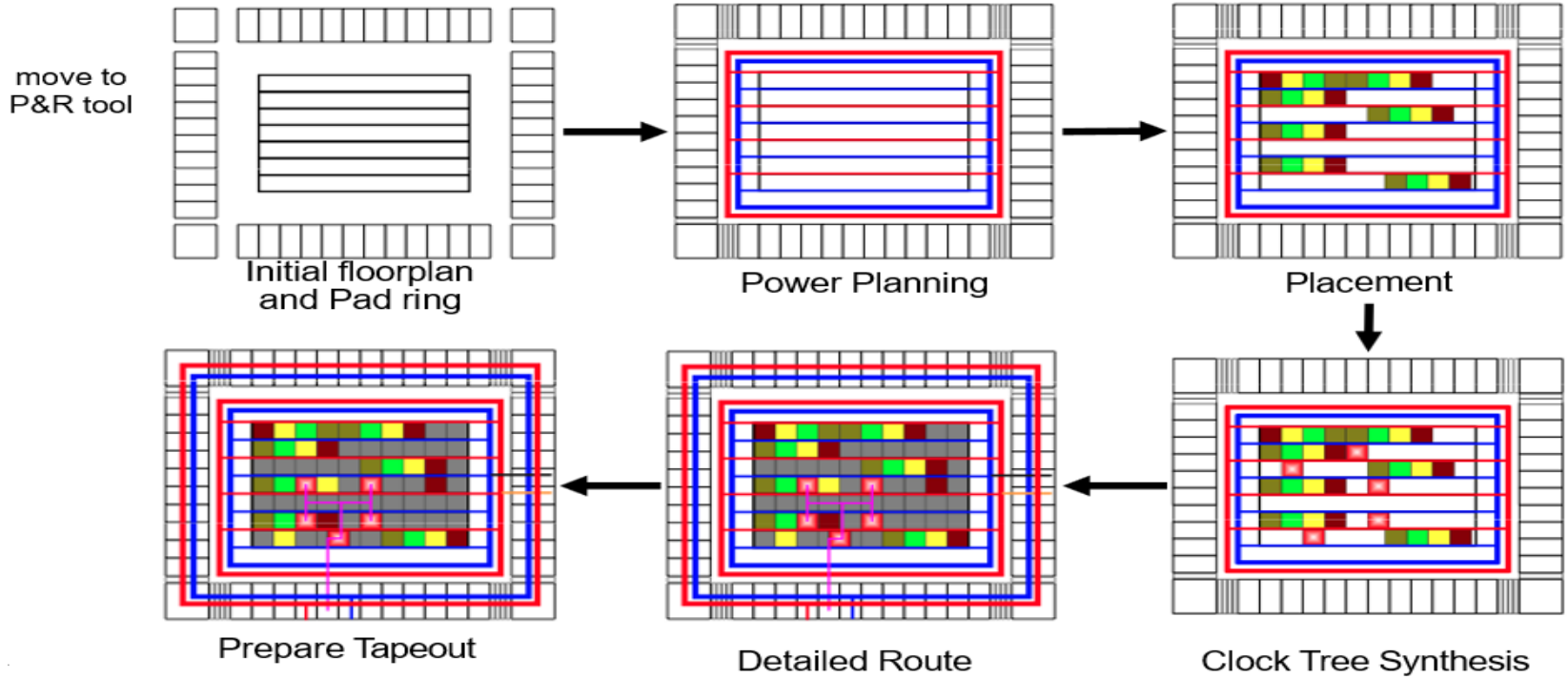
STEPS TOWARD LAYOUT

- Netlist and Partitioning
- Floorplanning
- Placement
- Clock-tree Synthesis
- Routing
- Parasitic Extraction
- Timing Analysis
- Noise Analysis
- Power Analysis

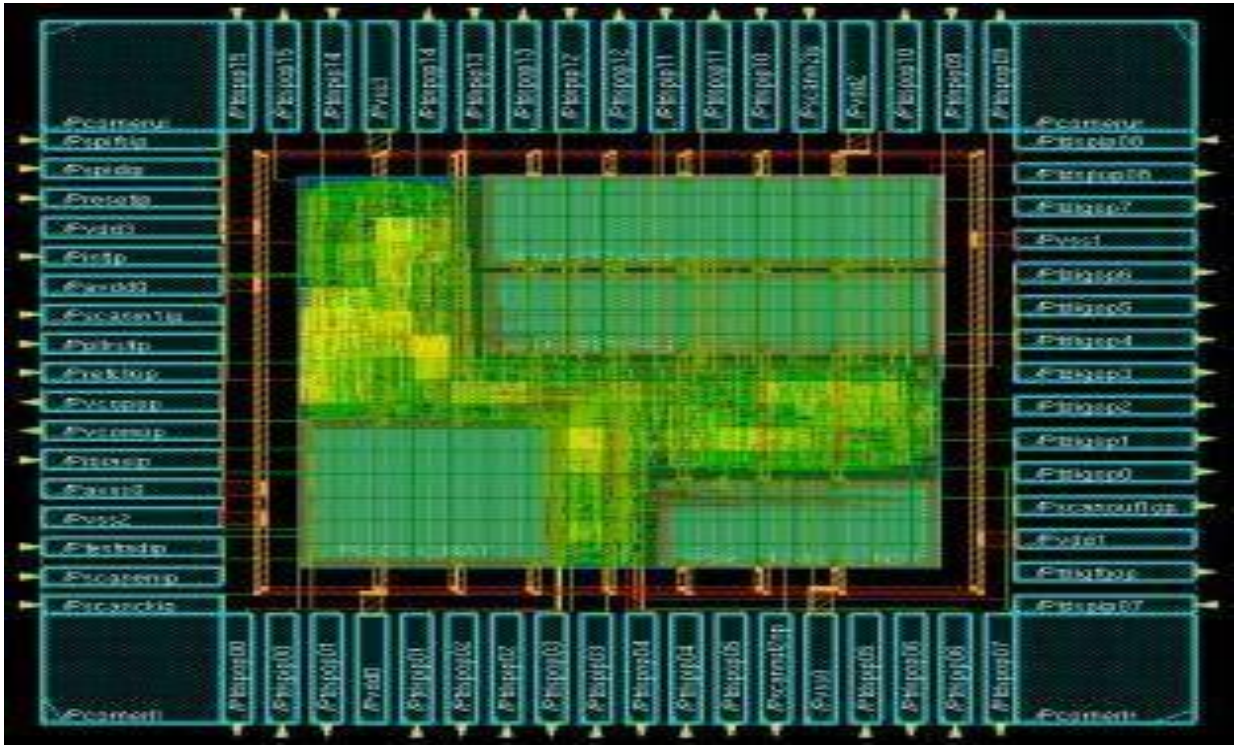
DATA EXCHANGE FORMATS

- Library Exchange Format (LEF) – contains timing information of standard cells and metal layers
- Design Exchange Format (DEF) – format (usually an ASCII file) passed between design flow stages
- Standard Delay Format (SDF) – IEEE standard for representing and interpreting timing data
- Parasitic extraction formats – used to represent parasitic information (R & C) for timing analysis
  - Extended Standard Parasitic Format (ESPF)
  - Reduced Standard Parasitic Format (RSPF)
  - Standard Parasitic Exchange Format (SPEF)

AN ILLUSTRATIVE VIEW OF PHYSICAL DESIGN

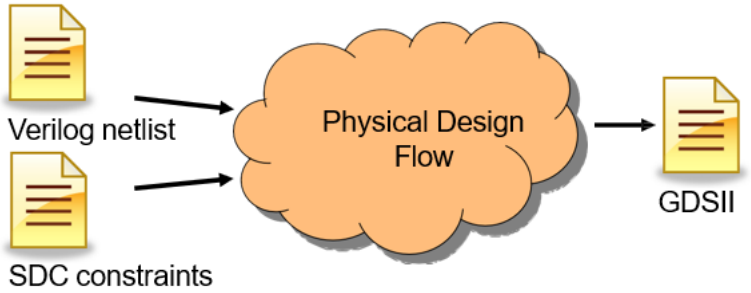


AN ILLUSTRATIVE VIEW OF PHYSICAL DESIGN



MOVING FROM LOGICAL TO PHYSICAL DESIGN

- Define design (.v)
- Define design constraints/targets (.sdc)
- Define operating conditions/modes (MMMC)
- Define technology and libraries (.lef)
- Define physical properties (Floorplan)



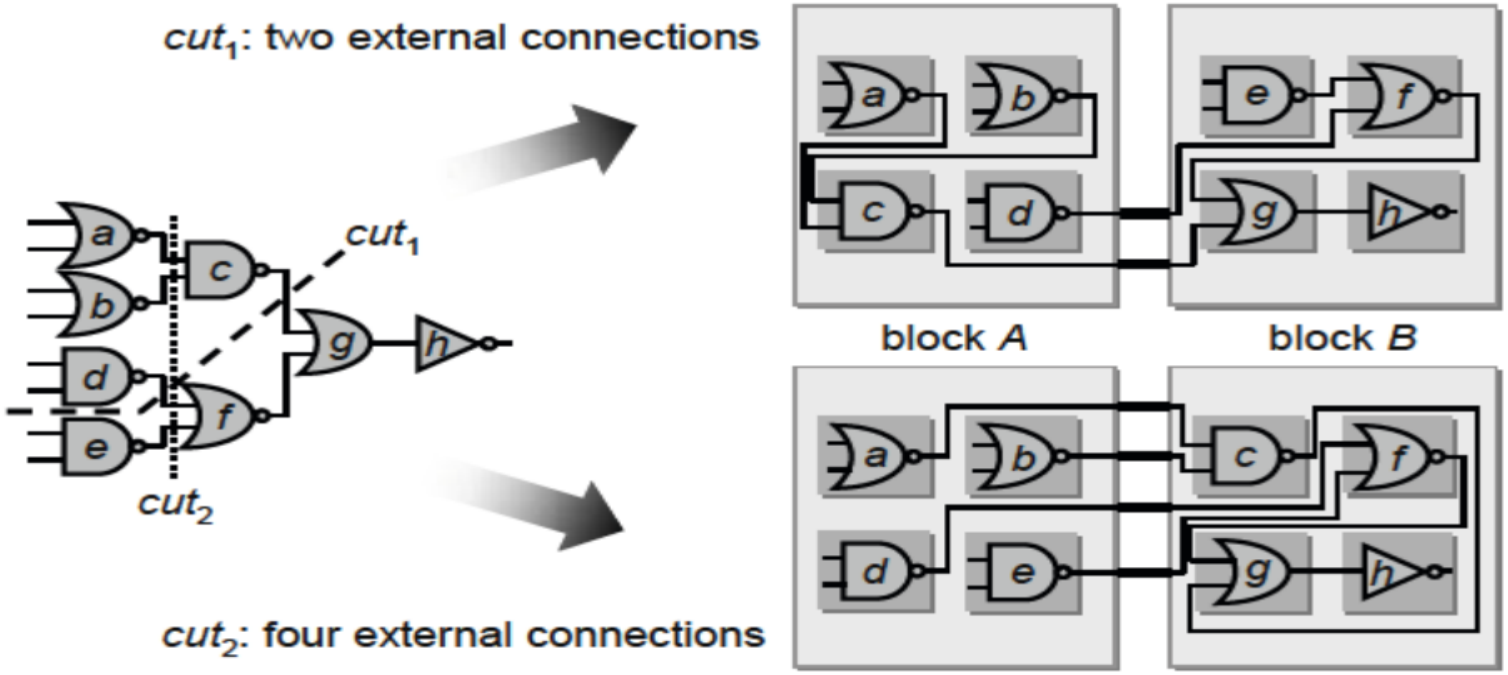
MOVING FROM LOGICAL TO PHYSICAL DESIGN

- During synthesis, our world view was a bit idealistic.
  - We didn't care about power supplies.
  - We didn't care about physical connections/entities.
  - We didn't care about clock non-idealities.
- Therefore, in order to start physical implementation:
  - Define "global nets" and how they connect to physical instances.
  - Provide technology rules and cell abstracts (.lef files)
  - Provide physical cells, unnecessary for logical functionality:
    - Tie cells, P/G Pads, DeCaps, Filler cells, etc.
  - Set up "low power" definitions, such as voltage domains, power gates, body taps, etc.

NETLIST AND SYSTEM PARTITIONING

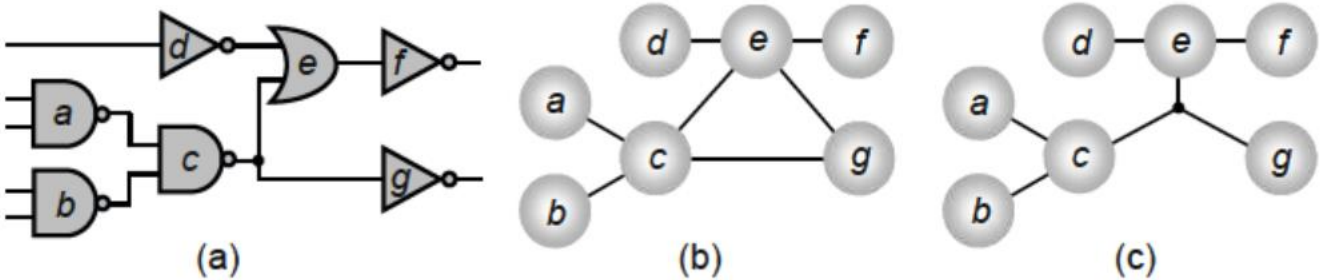
- The design complexity of modern integrated circuits has reached unprecedented scale, making full-chip layout, FPGA-based emulation and other important tasks increasingly difficult.
- A common strategy is to partition or divide the design into smaller portions, each of which can be processed with some degree of independence and parallelism.
- A divide-and-conquer strategy for chip design can be implemented by laying out each block individually and reassembling the results as geometric partitions.
- The partitioner divides the circuit into several subcircuits (partitions or blocks) while minimizing the number of connections between partitions, subject to design constraints such as maximum partition sizes and maximum path delay.

SYSTEM PARTITIONING: EXAMPLE



SYSTEM PARTITIONING: TERMINOLOGY

- Cell: Any logical or functional unit built from components
- Partition or Block: A group of components or cell
- Partitioning problem can be abstracted using a graph representation
  - Nodes represent cells
  - Edges represent connections between cells



(a) Sample circuit. (b) Possible graph representation. (c) Possible hypergraph representation.

PARTITIONING: KERNIGHAN-LIN (KL) ALGORITHM

- The Kernighan-Lin (KL) algorithm performs partitioning through iterative improvement steps.
- It was proposed by B. W. Kernighan and S. Lin in 1970 for bi-partitioning (k = 2) graphs, where every node has unit weight.
- The KL algorithm operates on a graph representation of the circuit, where nodes (edges) represent cells (connections between cells).
- Formally, let a graph G(V,E) have |V| = 2n nodes, where each node $v \varepsilon V$ has the same weight, and each edge $e \varepsilon E$ has a non-negative edge weight.
- The KL algorithm partitions V into two disjoint subsets A and B with minimum cut cost and |A| = |B| = n.

PARTITIONING: KERNIGHAN-LIN (KL) ALGORITHM

- The KL algorithm is based on exchanging (swapping) pairs of nodes, each node from a different partition.
- The two nodes that generate the highest reduction in cut size are swapped.
- To prevent immediate move reversal (undo) and subsequent infinite loops, the KL algorithm fixes nodes after swapping them.
- Fixed nodes cannot be swapped until they are released, i.e., become free.
- Execution of the KL algorithm proceeds in passes. Typically, the first pass or iteration begins with an arbitrary initial partition.
- In a given pass, after all nodes become fixed, the algorithm determines the prefix of the sequence of swaps within this pass that produces the largest gain, i.e., reduction of cut cost.

PARTITIONING: KERNIGHAN-LIN (KL) ALGORITHM

- The cut size or cut cost of a graph with either unweighted or uniform-weight edges is the number of edges that have nodes in more than one partition. With weighted edges, the cut cost is the sum of the weights of all cut edges.
- The cost D(v) of moving a node $v \varepsilon V$ in a graph from partition A to B is

$$D(v) = |E_B(v)| - |E_A(v)|$$

- where $E_B(v)$ is the set of v's incident edges that are cut by the cut line, and $E_A(v)$ is the set of v's incident edges that are not cut by the cut line.
- High costs (D > 0) indicate that the node should move, while low costs (D < 0) indicate that the node should stay within the same partition.

PARTITIONING: KERNIGHAN-LIN (KL) ALGORITHM

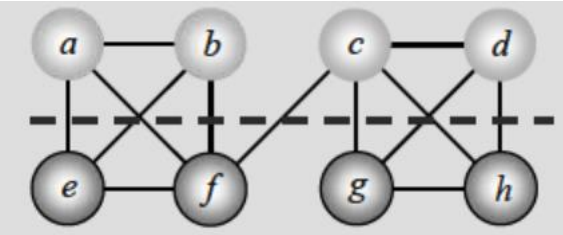
- The gain $\Delta g(a,b)$ of swapping a pair of nodes a and b is the improvement in overall cut cost that would result from the node swap. A positive gain ($\Delta g > 0$) means that the cut cost is decreased, while a negative gain ($\Delta g < 0$) means that the cut cost is increased. The gain of swapping two nodes a and b is:

$$\Delta g(a,b) = D(a) + D(b) - 2c(a,b)$$

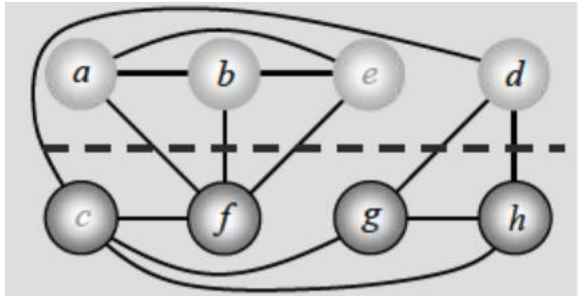
- Where D(a) and D(b) are the respective costs of nodes a and b, and c(a,b) is the connection weight between a and b. If an edge exists between a and b, then c(a,b) = the edge weight between a and b. Otherwise, c(a,b) = 0.
- The maximum positive gain G_m corresponds to the best prefix of m swaps within the swap sequence of a given pass. These m swaps lead to the partition with the minimum cut cost encountered during the pass. G_m is computed as the sum of Δg values over the first m swaps of the pass, with m chosen such that G_m is maximized.

EXAMPLE: KERNIGHAN-LIN (KL) ALGORITHM

- Given: Initial partition
- Task: Perform the first pass of KL algorithm

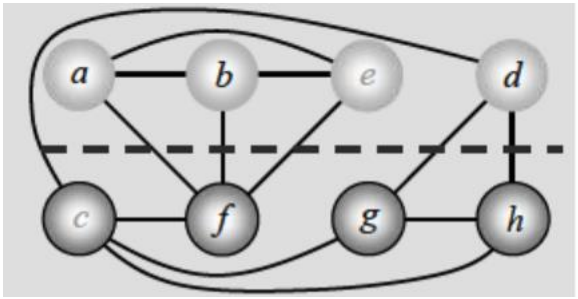


Compute $D(v)$ costs for all free nodes a-h.
$D(a) = 1,\ D(b) = 1,\ D(c) = 2,\ D(d) = 1,$
$D(e) = 1,\ D(f) = 2,\ D(g) = 1,\ D(h) = 1$
$\Delta g_1 = D(c) + D(e) - 2c(c,e) = 2 + 1 - 0 = 3$
Swap and fix nodes c and e.

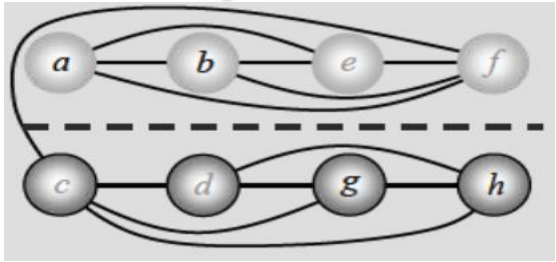


EXAMPLE: KERNIGHAN-LIN (KL) ALGORITHM

- Given: Initial partition
- Task: Perform the first pass of KL algorithm

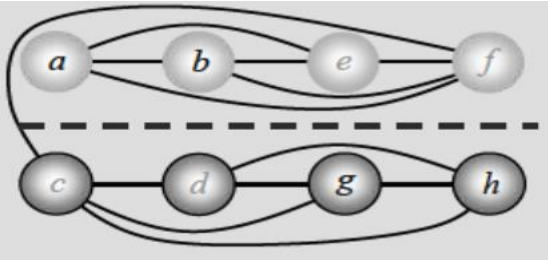


Update $D(v)$ costs for all free nodes connected to newly swapped nodes c and e: a, b, d, f, g and h.
$D(a) = -1, D(b) = -1, D(d) = 3,$
$D(f) = 2, \quad D(g) = -1, D(h) = -1$
$\Delta g_2 = D(d) + D(f) - 2c(d,f) = 3 + 2 - 0 = 5$
Swap and fix nodes d and f.

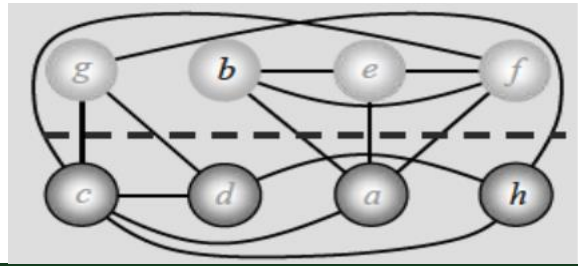


EXAMPLE: KERNIGHAN-LIN (KL) ALGORITHM

- Given: Initial partition
- Task: Perform the first pass of KL algorithm

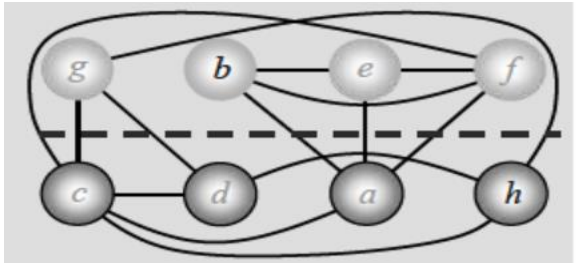


Update $D(v)$ costs for all free nodes connected to newly swapped nodes d and f: a, b, g and h.
$D(a) = -3, D(b) = -3, D(g) = -3, D(h) = -3$
$\Delta g_3 = D(a) + D(g) - 2c(a,g) = -3 + -3 - 0 = -6$
Swap and fix nodes a and g.

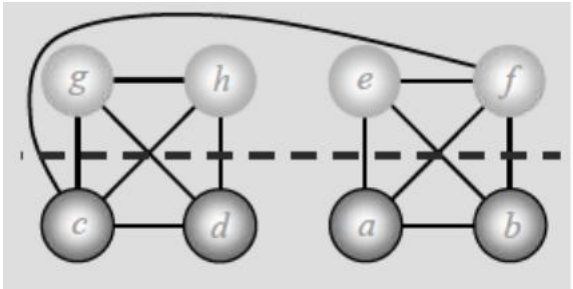


EXAMPLE: KERNIGHAN-LIN (KL) ALGORITHM

- Given: Initial partition
- Task: Perform the first pass of KL algorithm



Update $D(v)$ costs for all free nodes connected to newly swapped nodes a and g: b and h.
$D(b) = -1$, $D(h) = -1$
$\Delta g_4 = D(b) + D(h) - 2c(b,h) = -1 + -1 - 0 = -2$
Swap and fix nodes b and h.



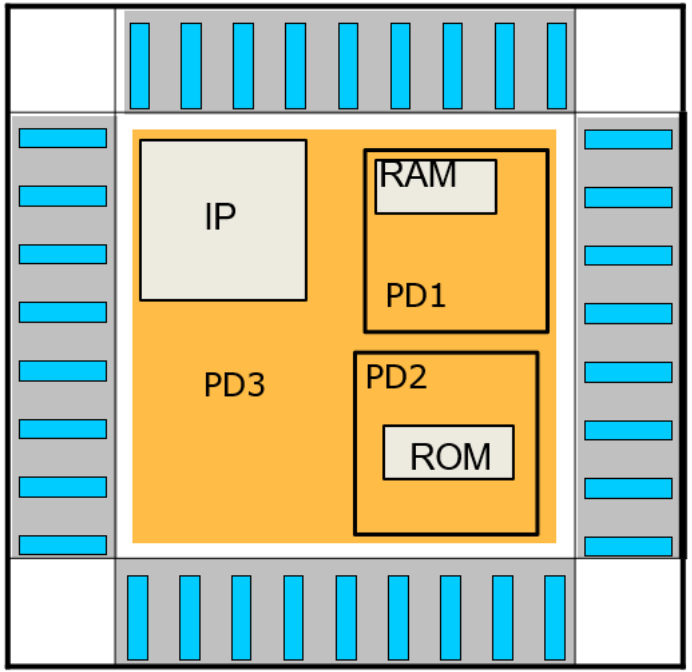
Compute maximum positive gain G_m.

$G_1 = \Delta g_1 = 3$
$G_2 = \Delta g_1 + \Delta g_2 = 3 + 5 = 8$
$G_3 = \Delta g_1 + \Delta g_2 + \Delta g_3 = 3 + 5 + -6 = 2$
$G_4 = \Delta g_1 + \Delta g_2 + \Delta g_3 + \Delta g_4 = 3 + 5 + -6 + -2 = 0$

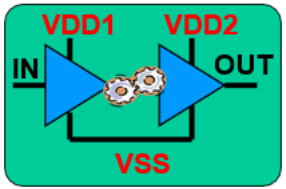
$G_m = 8$ with $m = 2$. Since $G_m > 0$, the first $m = 2$ swaps are executed: (c,e) and (d,f). Additional passes are performed until $G_m \leq 0$.

MULTIPLE VOLTAGE DOMAIN

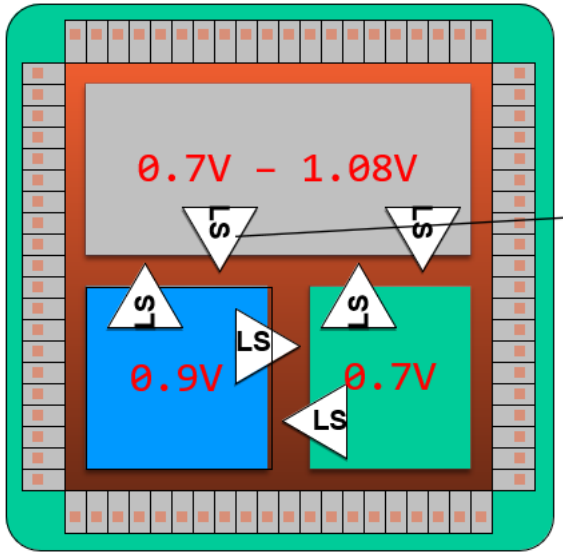
- Define power domains
  - Create power domain names
  - List of cells connected to VDD1, VDD2, GND1,…
  - Draw the power domains
- Place macros
  - Take into account:
    - Routing congestion
    - Orientation
  - Manual usually better then Auto
- Place switches
  - For the power down domains



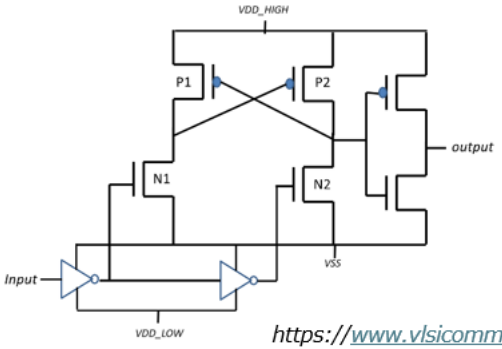
MULTIPLE DOMAIN DESIGN: LEVEL SHIFTERS



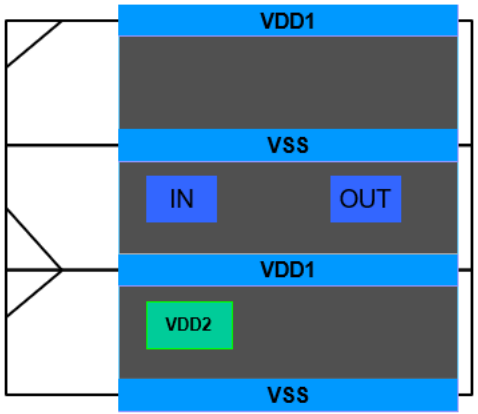
logic model



Dual High-to-Low and
Low-to-Hight level shifter

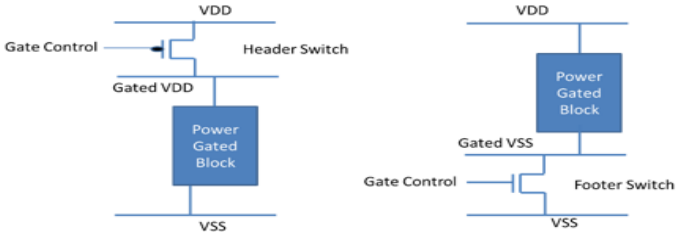


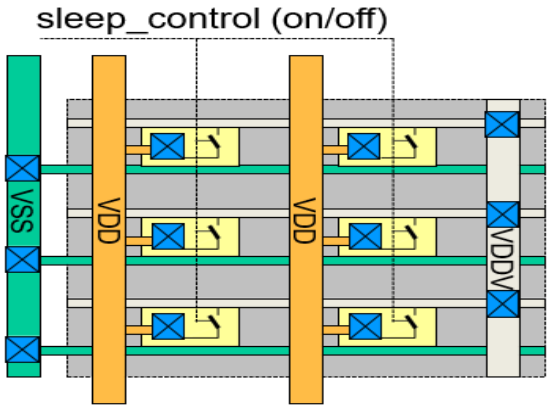
Double row Shifter Floorplan

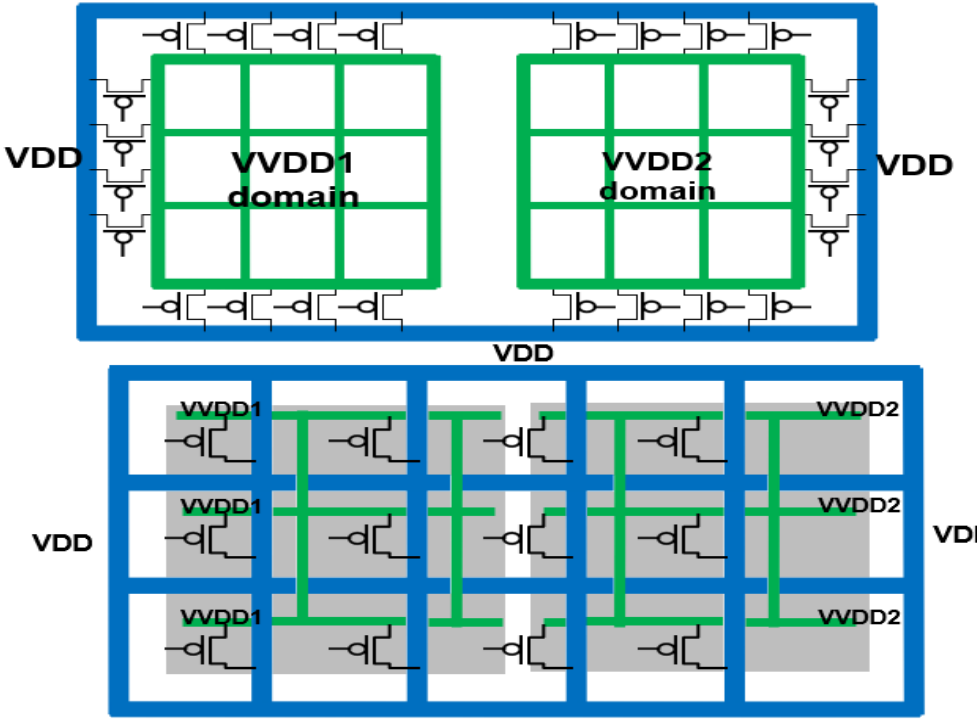


https://www.vlsicommunity.com

MULTIPLE DOMAIN DESIGN: POWER GATING







Thank you!