EEDG/CE 5325: Hardware Modeling

Bill Swartz

Dept. of EE
Univ. of Texas at Dallas

Session 01

Prototyping
Verilog-for-Synthesis

Design Process

- Idea
- Implement
- Test

- Manufacture

- Subject to iteration (Refinement)
- Hopefully converges

Implementation

- Prototype
- Abstraction to Physical

Prototype

- First implementation of physical
- Useful to be flexible
- See where real world differs from theoretical
- Modify as needed

Prototypes

- Breadboards
- Printed circuit boards
- Virtual Prototypes
  - Programmable Bread boards
  - Software programmable
  - Field Programmable Gate Array (FPGA)
  - Example: Baysys2 using Xilinx Spartan 3E FPGA

Virtual Prototype Implementation

- Software programmable hardware
- Hardware description languages used to describe prototype and its behavior

Hardware Modeling Using HDL

- **HDL: Hardware Description Language -** A high level programming language used to model hardware.

- Hardware Description Languages
  - have special hardware related constructs.
  - can be used to build models for **simulation**, **synthesis** and **test**
  - have been extended to the system design level
  - **VHDL: V**HSIC **H**ardware **D**escription **L**anguage
    - VHSIC – Very High Speed Integrated Circuit Program
    - Mostly used in academia
  - **Verilog** HDL
    - Mostly used in commercial electronics industry

Hardware Modeling Using HDL

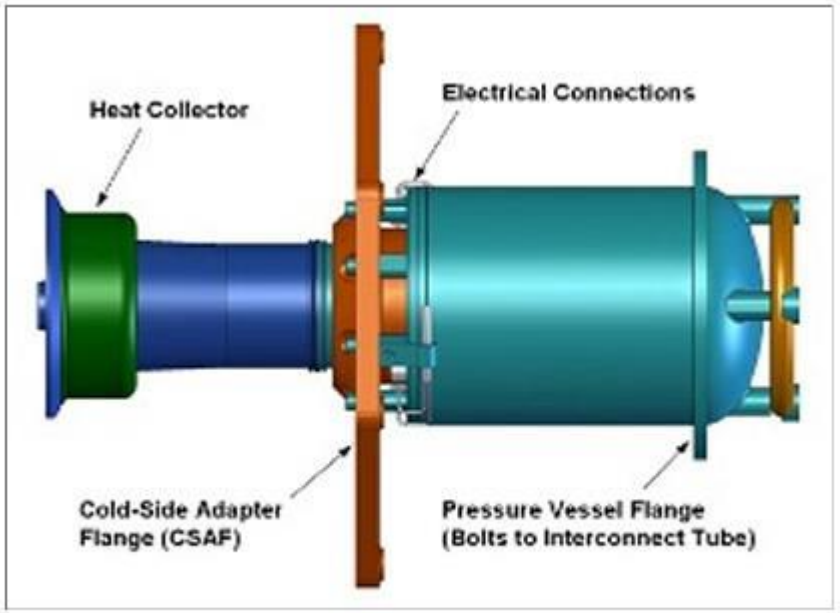
- **Verilog and VHDL may not be enough**

- Analog characteristics

  Mixed signal simulation

- Mechanical systems
  - Mixed mode simulation
  - Stirling engine simulation

Stirling Engine Converter

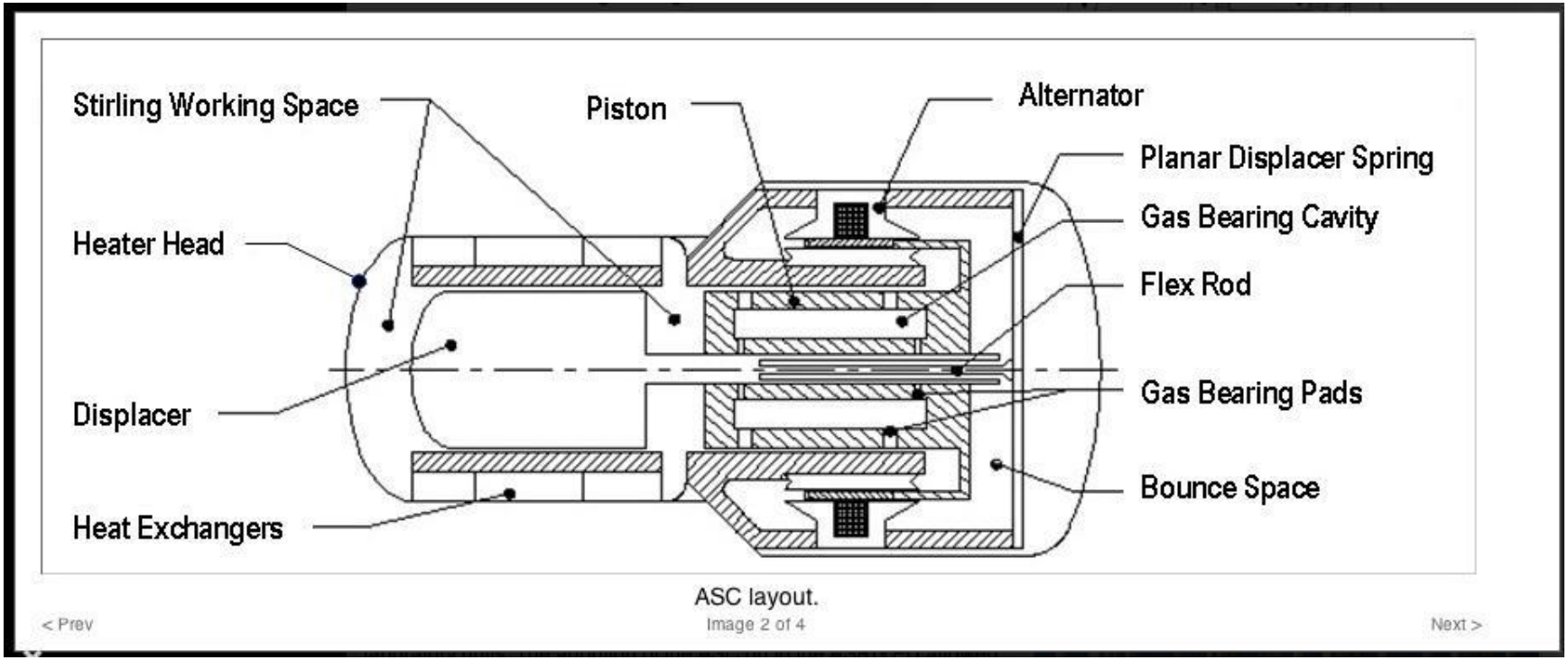


ASC-E2

Image 1 of 4 Next >

Converting heat energy to electricity – how to model this system?

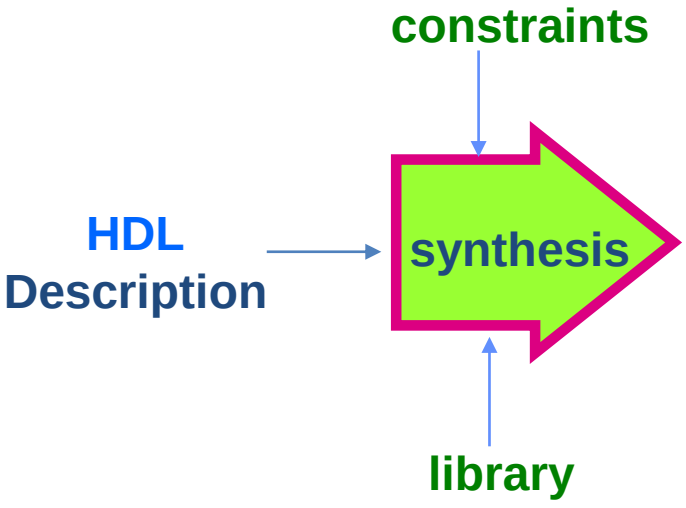
Stirling Engine Converter

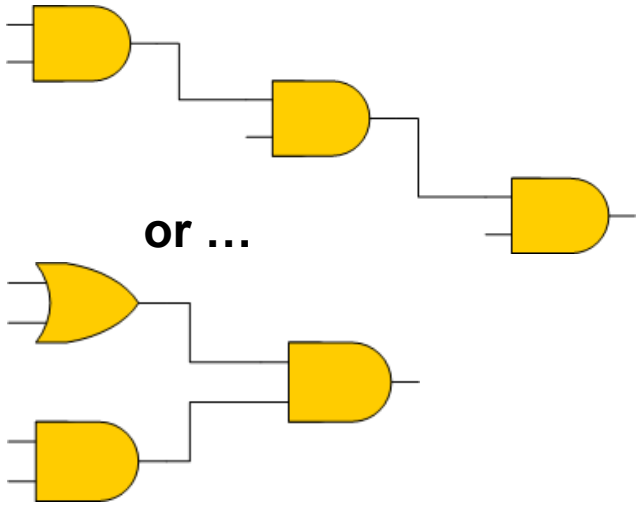


https:tec.grc.nasa.gov/rps/advanced-stirling-convertor

Concept of Synthesis

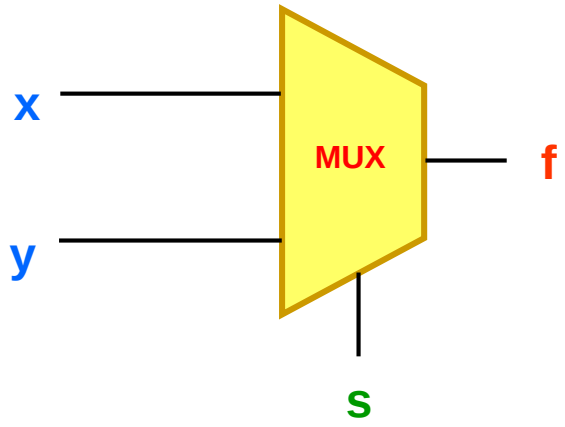
- Logic synthesis
  - A program that **"designs" logic from abstract descriptions** of the logic
    - takes constraints (e.g. size, speed)
    - uses a library (e.g. 3-input gates)
  - The aim of synthesis is to produce hardware which will do what the concurrent statements specify.
    - This includes processes as well as other concurrent statements.
- How?
  - You write an "abstract" HDL description of the logic
  - The synthesis tool provides alternative implementations

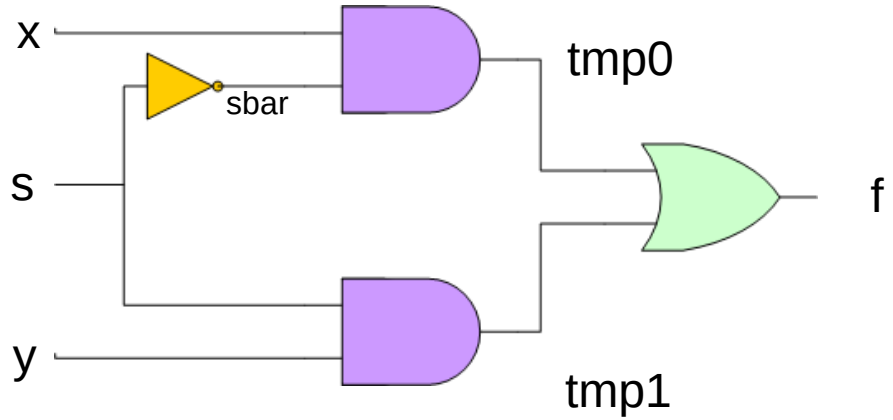




Goal

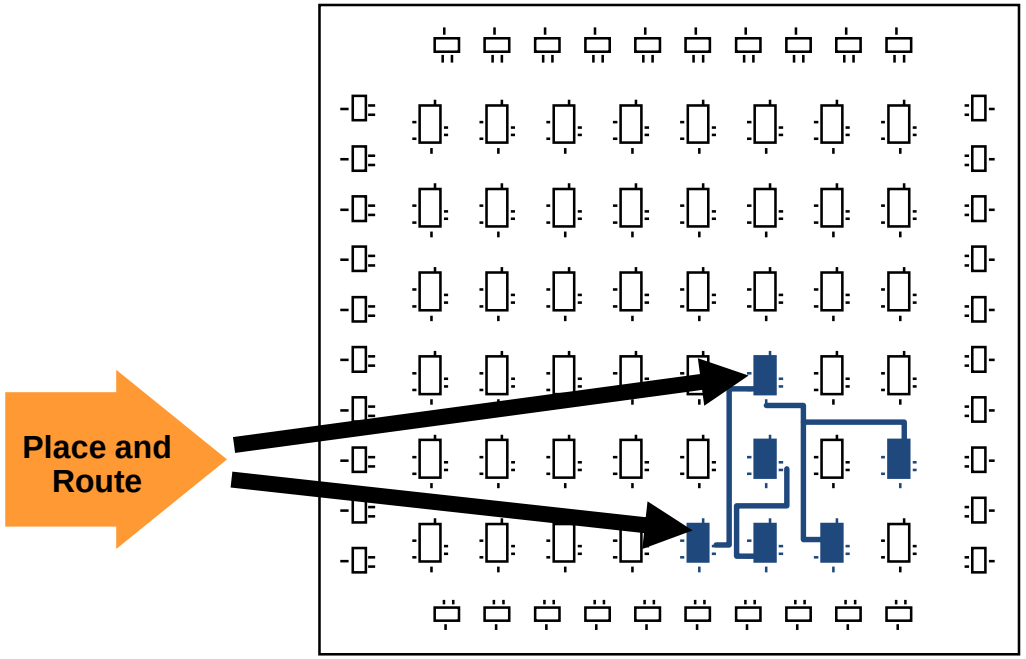
- We know the function we want, and can specify in a well-define representation.
  - … but we don't always know the exact gates (nor logic elements)…
  - … we want the tool to figure this out…





Importance of Synthesis

- In order to map an HDL code on a **FPGA**, **the code should be synthesizable!**
  - An HDL code that functions correctly in simulation, does not necessarily mean it is synthesizable.
  - Once you have gone through the synthesis tool for your HDL code **without errors**, your code is synthesizable.

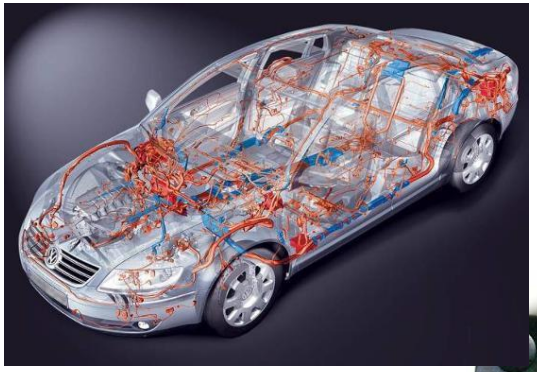


Languages Covered in this Class

- VHDL
- Verilog
- Spice
- Verilog A and Verilog AMS
- System C
- System Verilog
- UVM

The big picture

Electronics makes our life easier and productive



Integration of Electronics

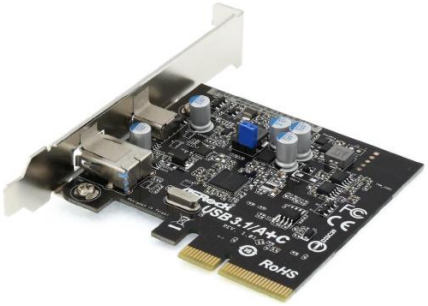
Cost reduction

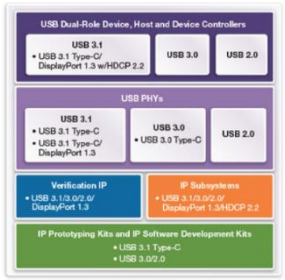
Improved performance

Improved reliability

Integration of Electronics

$$$ $$ $





USB Card

USB Single Chip

USB RTL IP

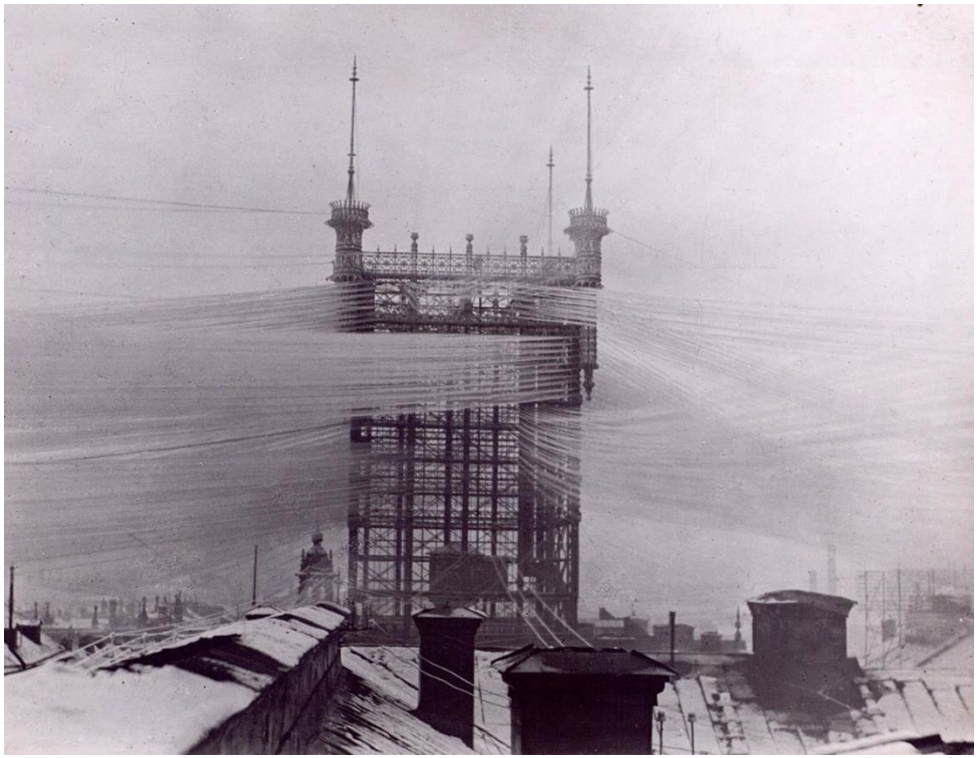
To be integrated with other circuitry

Technology scaling →

Complexity →

Integration of Electronics

How to manage the complexity of increased integration?



Complexity Mitigated

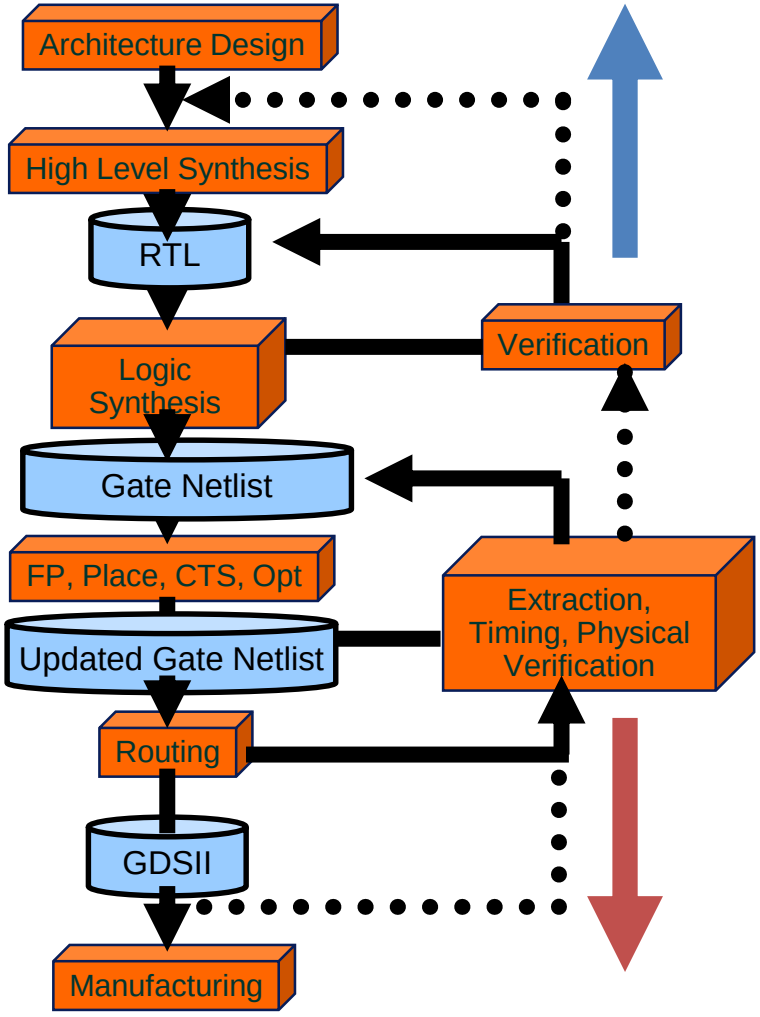
Use the computer

CAD to the rescue

Computer-Aided Design (CAD)

Chip development is complex

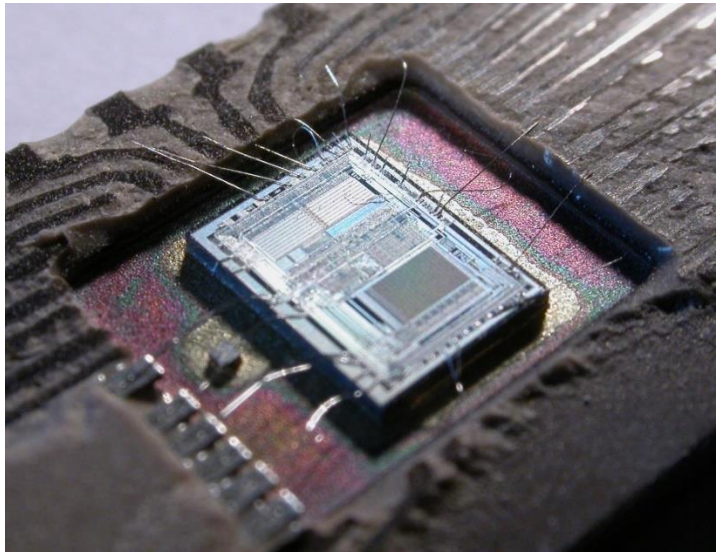
- Specification
  - What the system (or component) is supposed to do
- Architecture
  - High-level design of component
    - state definition
    - logic partitioning into blocks
- Logic
  - Logic gates, storage elements, and connections between them
- Circuit
  - Transistor-level circuits to realize logic elements
- Device, Interconnect
  - Individual circuit elements, wires
- Layout
  - Geometric shapes that define and connect circuit elements, wires
- Process
  - Steps used to fabricate circuit elements, wires



Courtesy of Prof. Andrew B. Kahng, UCSD ECE 260B / CSE 241A

Design of the physical IC

Let’s start at the “physical design” process



Physical Design

Transformation from

Structured network definition (netlist)

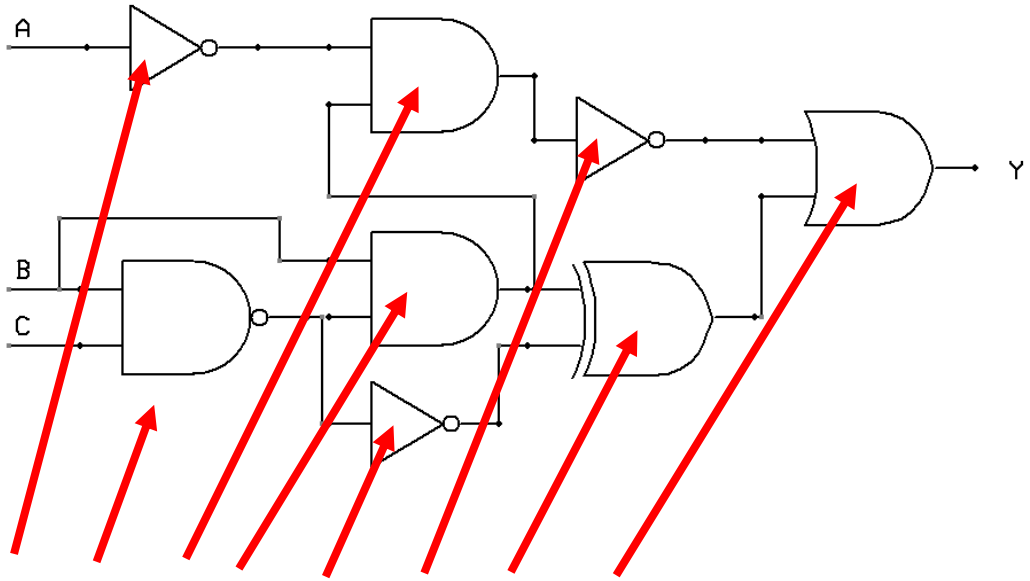
to

Geometries (artwork to generate photomasks)

Netlist  Geometries

Structured Netlist

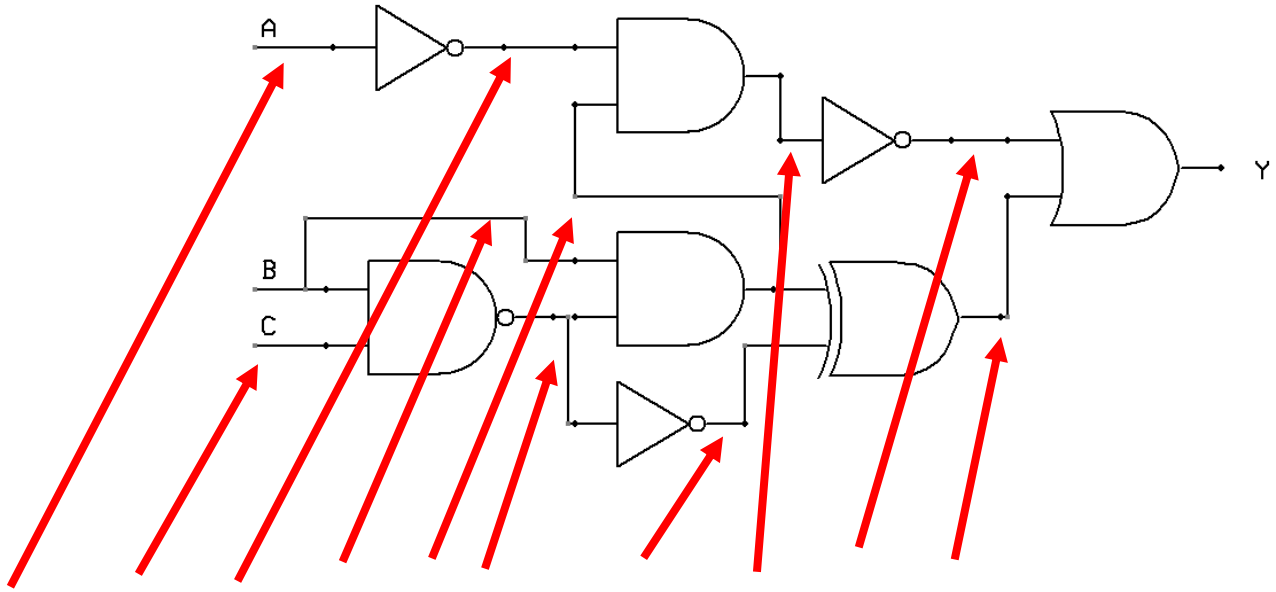
Interconnected components



Components (gates)

Structured Netlist

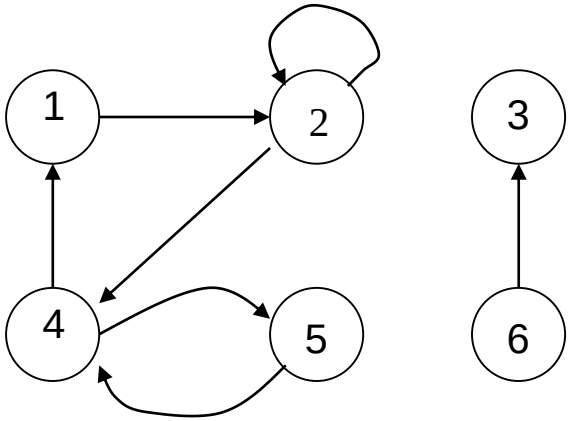
Interconnected components

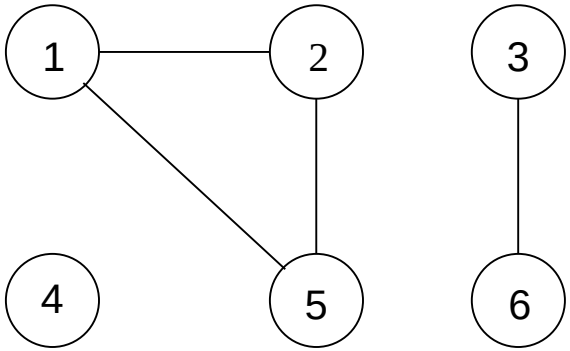


Wires interconnecting the gates

Graph Representation

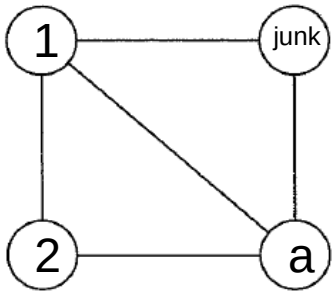
- Informally a *graph* is a set of nodes joined by a set of lines or arrows.





Graph Representation

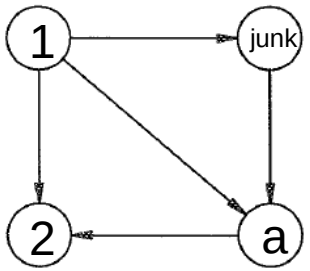
- A graph G(V,E) is a pair (V,E), where V is a set of vertices and E is a set of edges.
  - Undirected graph: the edges are unordered pairs, e.g. $\{v_i , v_j\}$



Undirected Graph

Graph Representation

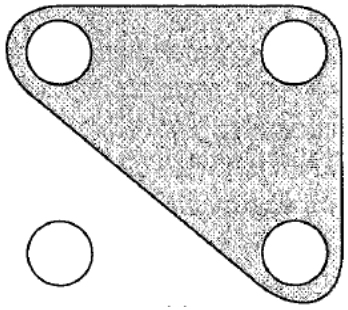
- A graph G(V,E) is a pair (V,E), where V is a set of vertices and E is a set of edges.
  - Directed graph (digraph): the edges are ordered pairs of vertices, e.g. (v_i , v_j)



Directed Graph

Graph Representation

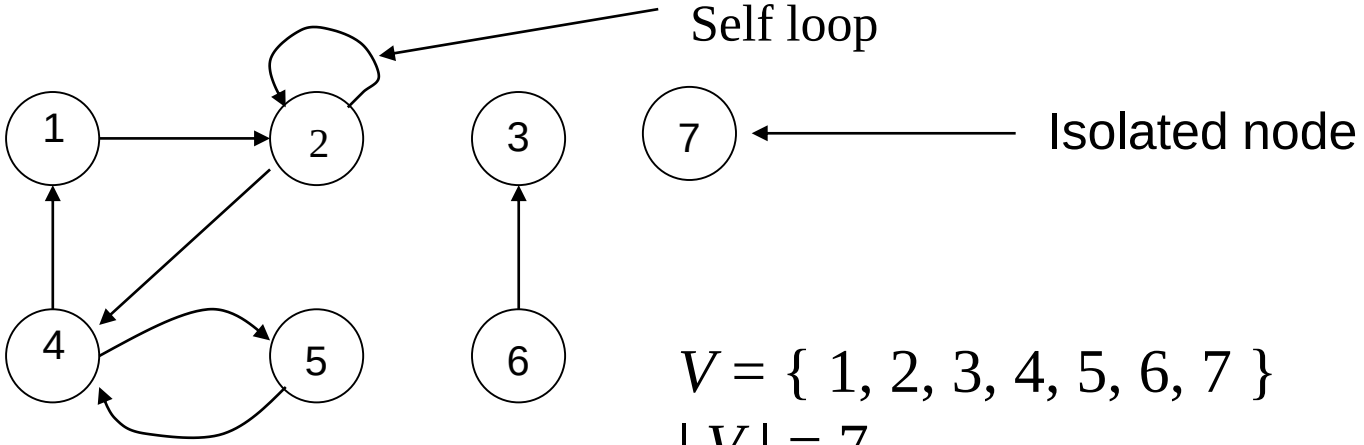
- A graph G(V,E) is a pair (V,E), where V is a set of vertices and E is a set of edges.
  - A hypergraph is an extension of a graph where edges may be incident to any number of vertices



Hypergraph

Directed Graph

- A ***directed graph,*** also called a ***digraph*** **G** is a pair (*V*, *E*), where the set *V* is a finite set and *E* is a binary relation on *V.*
- The set ***V*** is called the **vertex set** of *G* and the elements are called vertices. The set *E* is called the **edge set** of *G* and the elements are *edges* (also called *arcs*). A edge from node *a* to node *b* is denoted by the ordered pair (*a*, *b*).



$V = \{ 1, 2, 3, 4, 5, 6, 7 \}$

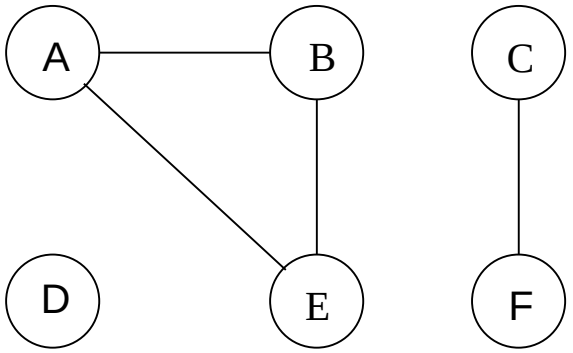
$| V | = 7$

$E = \{ (1,2), (2,2), (2,4), (4,5), (4,1), (5,4),(6,3) \}$

$| E | = 7$

Undirected Graph

- An ***undirected graph*** $G = (V , E)$, but unlike a digraph the edge set E consist of unordered pairs. We use the notation (a, b) to refer to a directed edge, and { a, b } for an undirected edge.



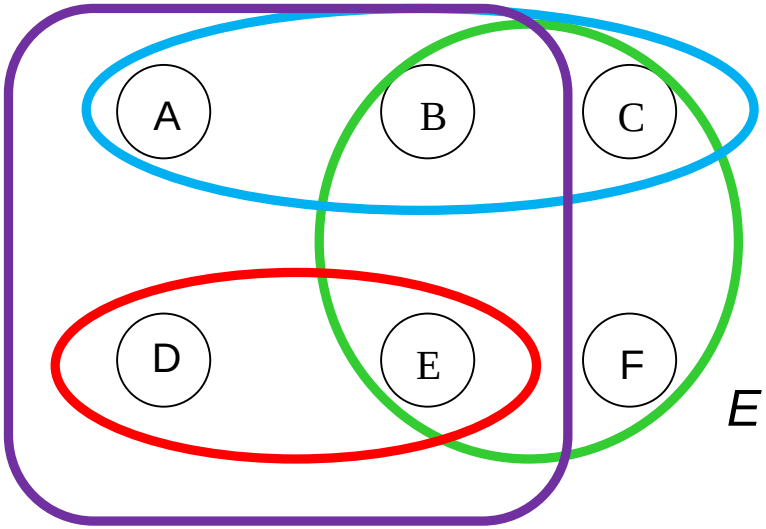
$V = \{ A, B, C, D, E, F \}$
$|V| = 6$

$E = \{ \{A, B\}, \{A,E\}, \{B,E\}, \{C,F\} \}$
$|E| = 4$

Some texts use (a, b) also for undirected edges.
So (a, b) and (b, a) refers to the same edge.

Hyper Graph

- A ***hyper graph*** $H = (V , E)$, is the set of vertices V and E is the set of edges which forms sets between the vertices or nodes. Any edge may contain any number of nodes



$V = \{ A, B, C, D, E, F \}$

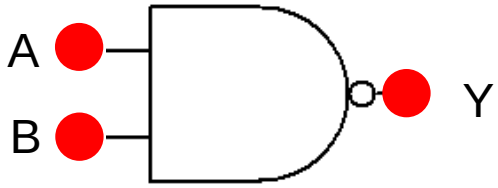
$|V| = 6$

$E = \{ \{B,C,E,F\}, \{A,B,C\}, \{D,E\}, \{A,B,D,E\} \}$

$|E| = 4$

- Natural representation of a circuit description or netlist

Component Terminals



The terminals A, B, Y connect the component to the outside world (circuit)

Component Terminals

- Input/Outputs IOs
- Terminals
- Pins
- Ports
- Connection Points

All equivalent

Connected Terminals

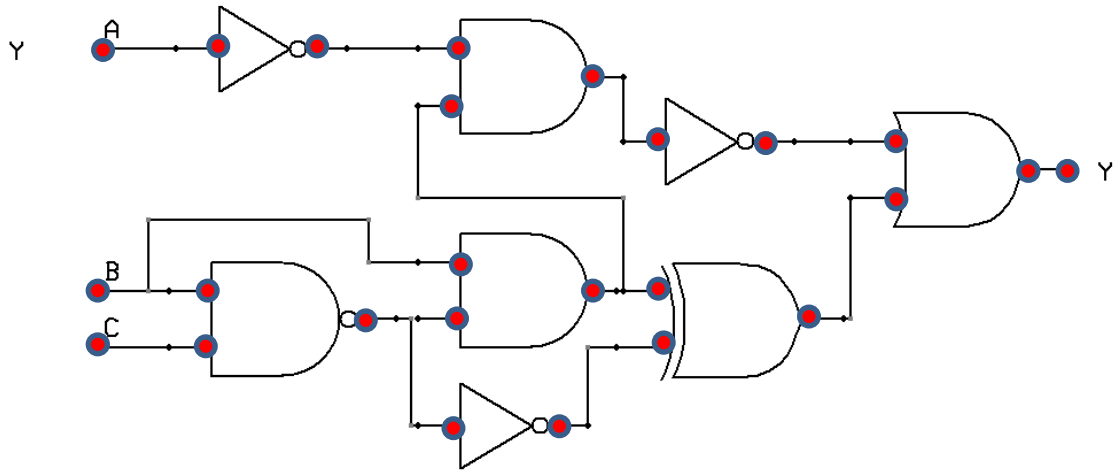
- Named Wire
- Signal
- Network
- Netname
- Net

All equivalent

Equipotential at Equilibrium t -> ∞

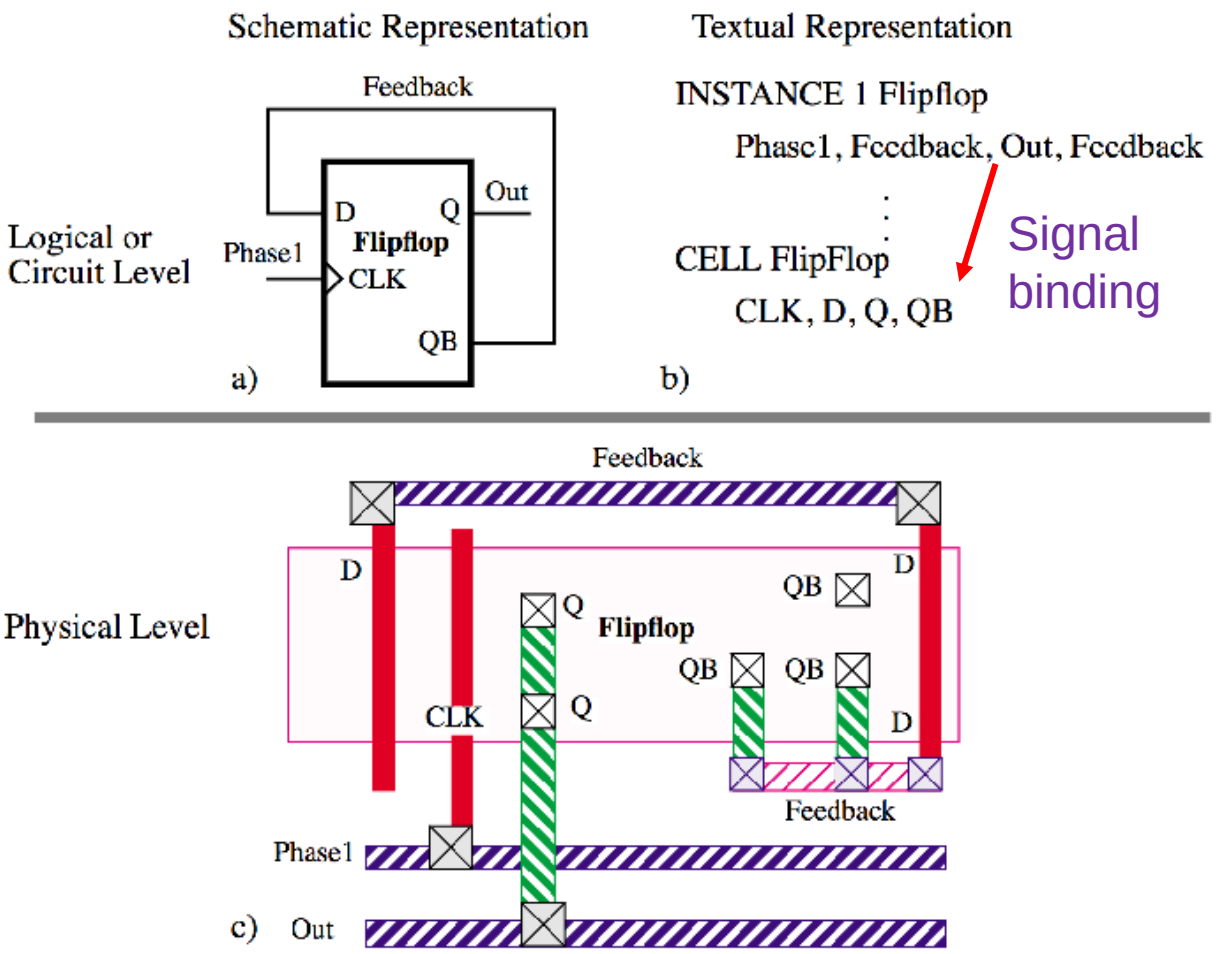
Structured Netlist

Alternate view: netlist is interconnection of component pins



Wires interconnecting the gate pins

Component Mapping / Views



Structural Verilog

Enumerate by component instances

Each component's terminals are bound to signals of the design

Structural Verilog Example

```
module design ( A, B, C, D, reset, clk, X, Y, Z ) ;
  input A, B, C, D ;
  input reset ;
  input clk ;
  output X ;
  output Y ;
  output Z ;
```

Connections to the outside world

```
  AND2 G1 ( .A(A), .B(B), .O(G1_O) ) ;
  OR2  G2 ( .A(C), .B(D), .O(G2_O) ) ;
  AND2 G3 ( .A(A), .B(FF4_Q), .O(G3_O) ) ;
  OR2  G4 ( .A(G1_O), .B(G2_0), .O(G4_O) ) ;

  S_DFF FF1 ( .D(G3_O), .CLK(clk), .R(reset), .Q( FF1_Q ) );
  S_DFF FF2 ( .D(G4_O), .CLK(clk), .R(reset), .Q( FF2_Q ) );
  S_DFF FF3 ( .D(G2_O), .CLK(clk), .R(reset), .Q( Z ) );

  AND2 G5 ( .A(FF2_Q), .B(Z), .O(G5_O) ) ;

  S_DFF FF4 ( .D(G5_O), .CLK(clk), .R(reset), .Q( FF4_Q ) );
  S_DFF FF5 ( .D(G7_O), .CLK(clk), .R(reset), .Q( FF5_Q ) );

  OR2  G6 ( .A(FF1_Q), .B(FF4_Q), .O(X) ) ;
  OR2  G7 ( .A(FF4_Q), .B(G5_O), .O(G7_O) ) ;
  OR2  G8 ( .A(FF4_Q), .B(FF5_Q), .O(Y) ) ;
```

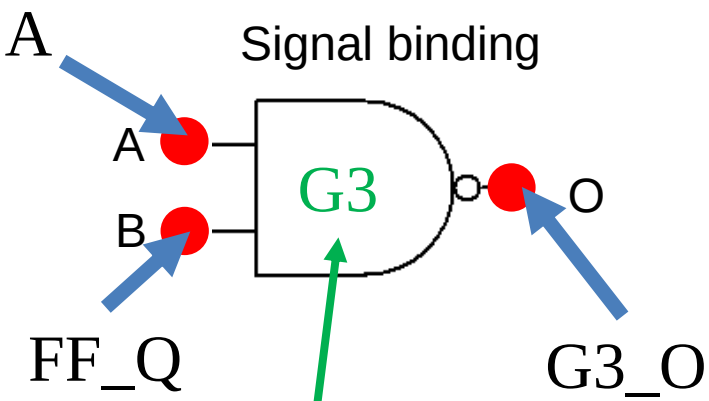
Enumerated components with signal bindings

```
endmodule
```

Structural Verilog Example

```
module design ( A, B, C, D, reset, clk, X, Y, Z ) ;
  input A, B, C, D ;
  input reset ;
  input clk ;
  output X ;
  output Y ;
  output Z ;

  AND2 G1 ( .A(A), .B(B), .O(G1_O) ) ;
  OR2  G2 ( .A(C), .B(D), .O(G2_O) ) ;
  AND2 G3 ( .A(A), .B(FF4_Q), .O(G3_O) ) ;
  OR2  G4 ( .A(G1_O), .B(G2_0), .O(G4_O) ) ;

  S_DFF FF1 ( .D(G3_O), .CLK(clk), .R(reset), .Q( FF1_Q ) );
  S_DFF FF2 ( .D(G4_O), .CLK(clk), .R(reset), .Q( FF2_Q ) );
  S_DFF FF3 ( .D(G2_O), .CLK(clk), .R(reset), .Q( Z ) );

  AND2 G5 ( .A(FF2_Q), .B(Z), .O(G5_O) ) ;

  S_DFF FF4 ( .D(G5_O), .CLK(clk), .R(reset), .Q( FF4_Q ) );
  S_DFF FF5 ( .D(G7_O), .CLK(clk), .R(reset), .Q( FF5_Q ) );

  OR2  G6 ( .A(FF1_Q), .B(FF4_Q), .O(X) ) ;
  OR2  G7 ( .A(FF4_Q), .B(G5_O), .O(G7_O) ) ;
  OR2  G8 ( .A(FF4_Q), .B(FF5_Q), .O(Y) ) ;

endmodule
```

Component to be placed. Standard cell 2 input AND gate



Instance name G3

Physical Design Transformation

Netlist  Geometries

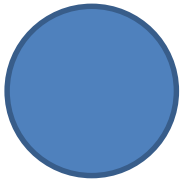
Geometries

Historically:

polygons 

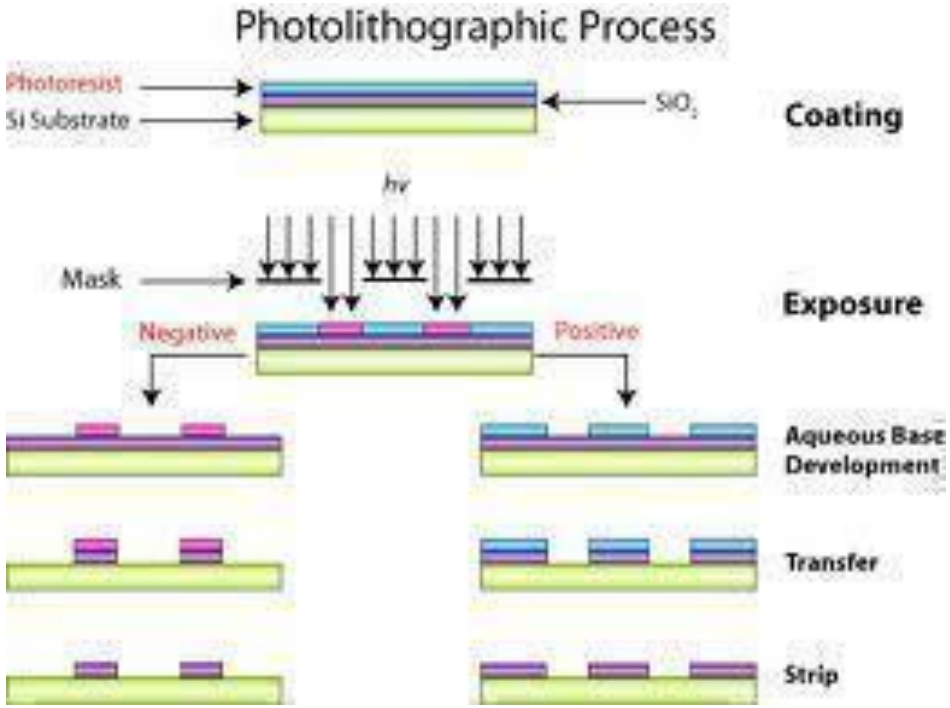
rectangles 

trapezoids 

circles 

Geometry Layers

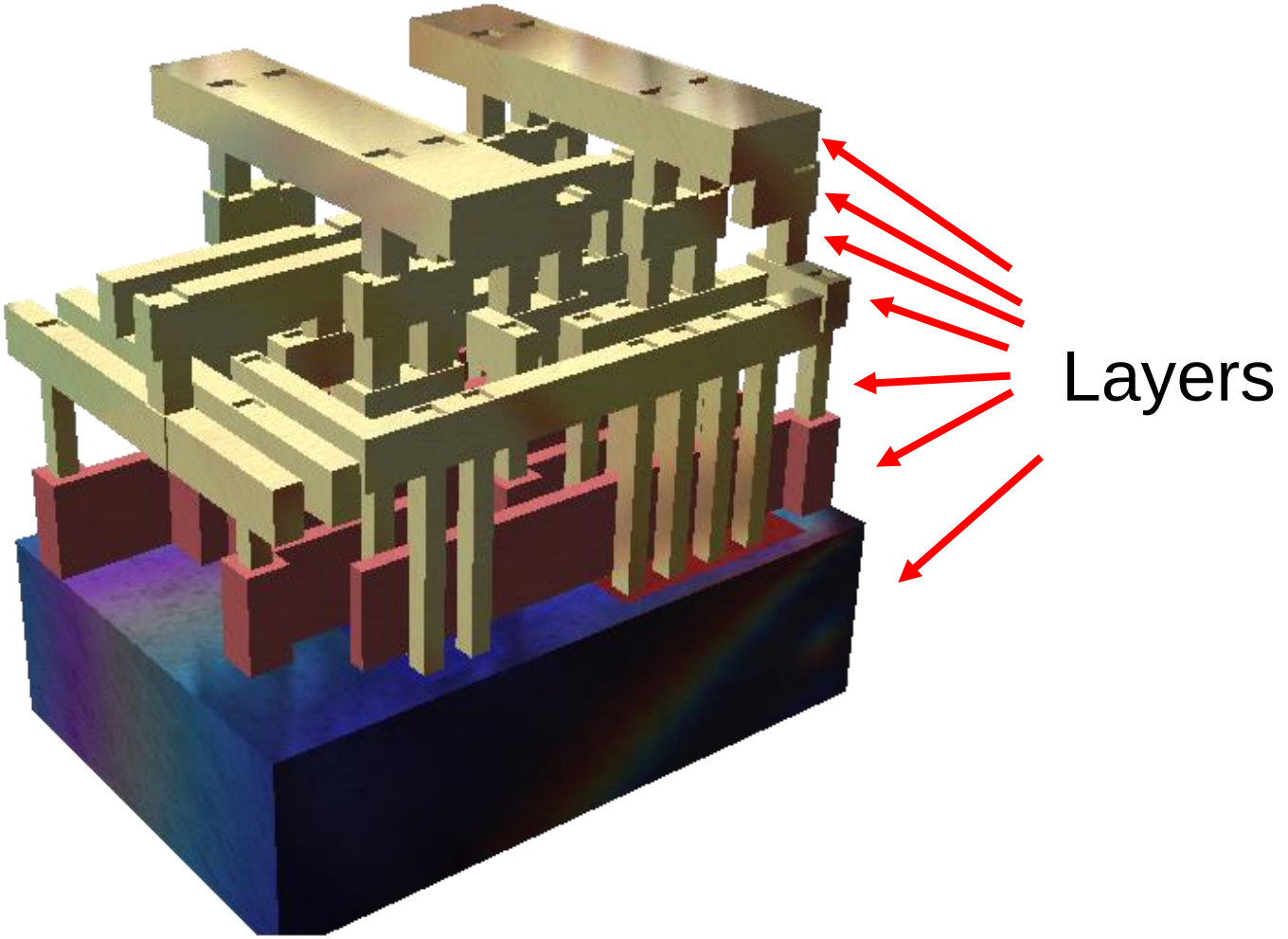
The IC manufacturing process uses photolithography



Geometry Layers

The geometries which are used to process and build a photomask are known as a “layer”

Photolithography Layers



Layer Names

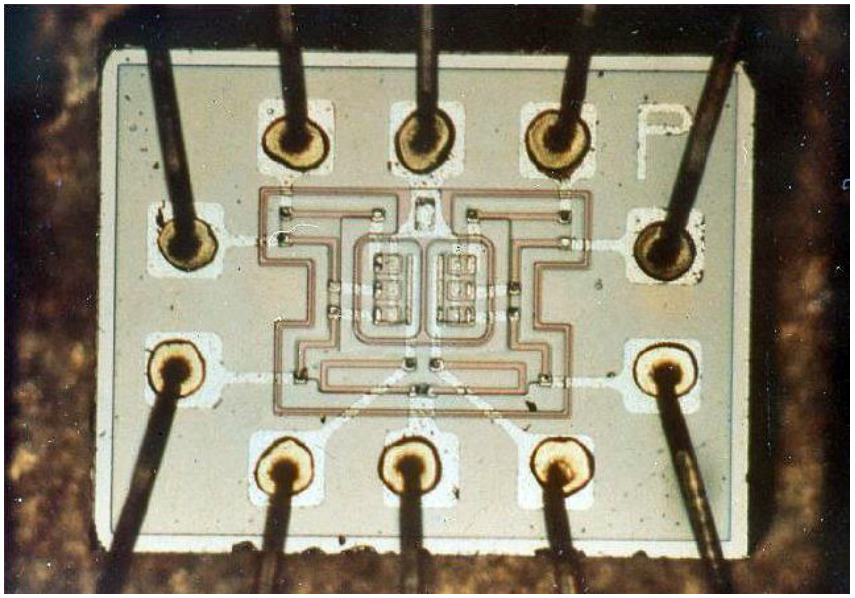
Each layer has a unique name for reference

Examples:

- Metal1
- M1
- Poly
- Ndiff

Planar Process

Integrated circuit manufacturing works on planes



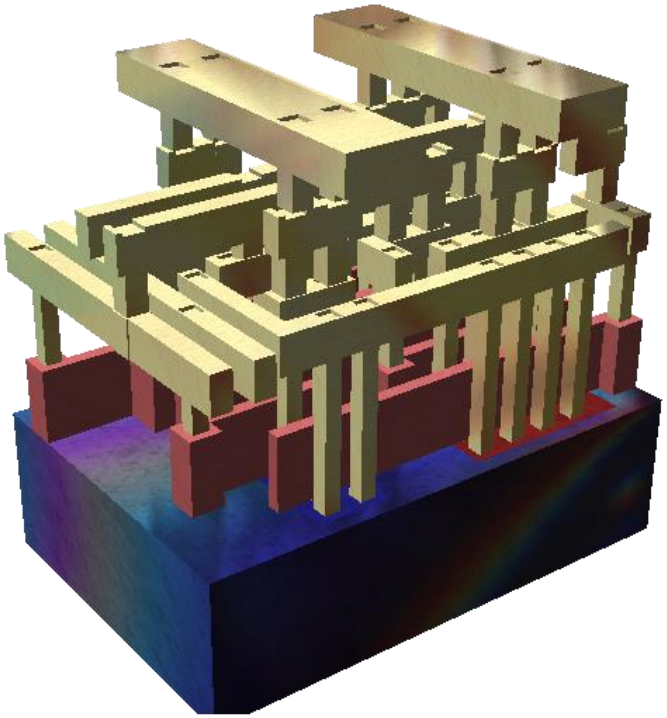
First planar NOR2 gate (Fairchild)

Layers are ordered

Layers are deposited or formed in a certain order.

Layers

Layers are ordered based on the process

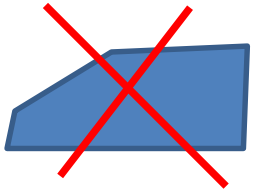


Poly < Metal1 < Metal2

Modern Geometries

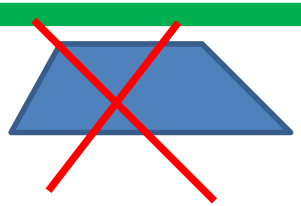
Today:

~~polygons~~

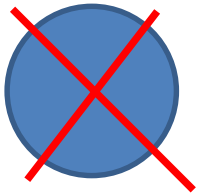


rectangles

~~trapezoids~~



~~circles~~



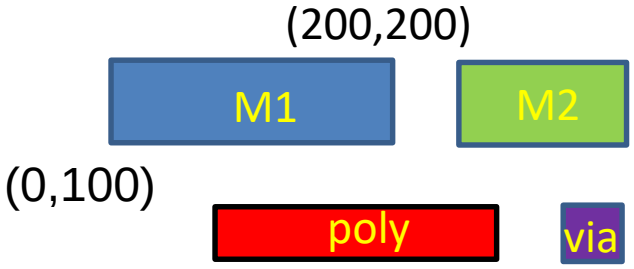
Due to manufacturing process (193nm light)

Current Physical Design

Netlist

Rectangles

$\{(x_{lb},y_{lb},layer)\ (x^{rt},y^{rt},layer)_{1\ldots}\}$



Current Physical Design

Netlist  Rectangles

$\{(x_{lb},y_{lb},\text{layer})\ (x^{rt},y^{rt},\text{layer})_{1\ldots}\}$

Billions of rectangles!

Overcoming the complexity

Abstraction

Divide and conquer

Overcoming the complexity

Abstraction

- How to model and instantiate the components?
- How to group the underlying transistors?
- What is the granularity?
- How to assemble the final die?

Divide and conquer

Design Style

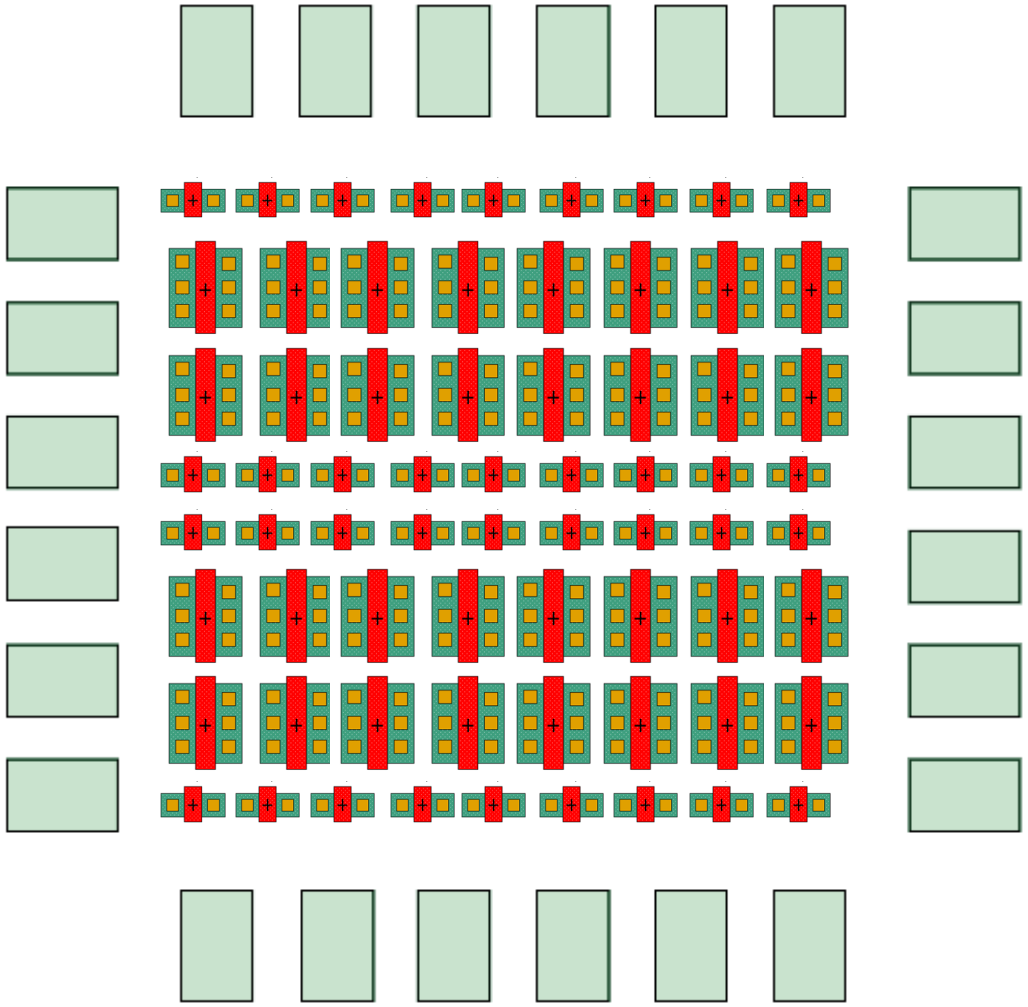
Grouping transistors induces a “design style”

Forms a “cell” to be manipulated

Design styles

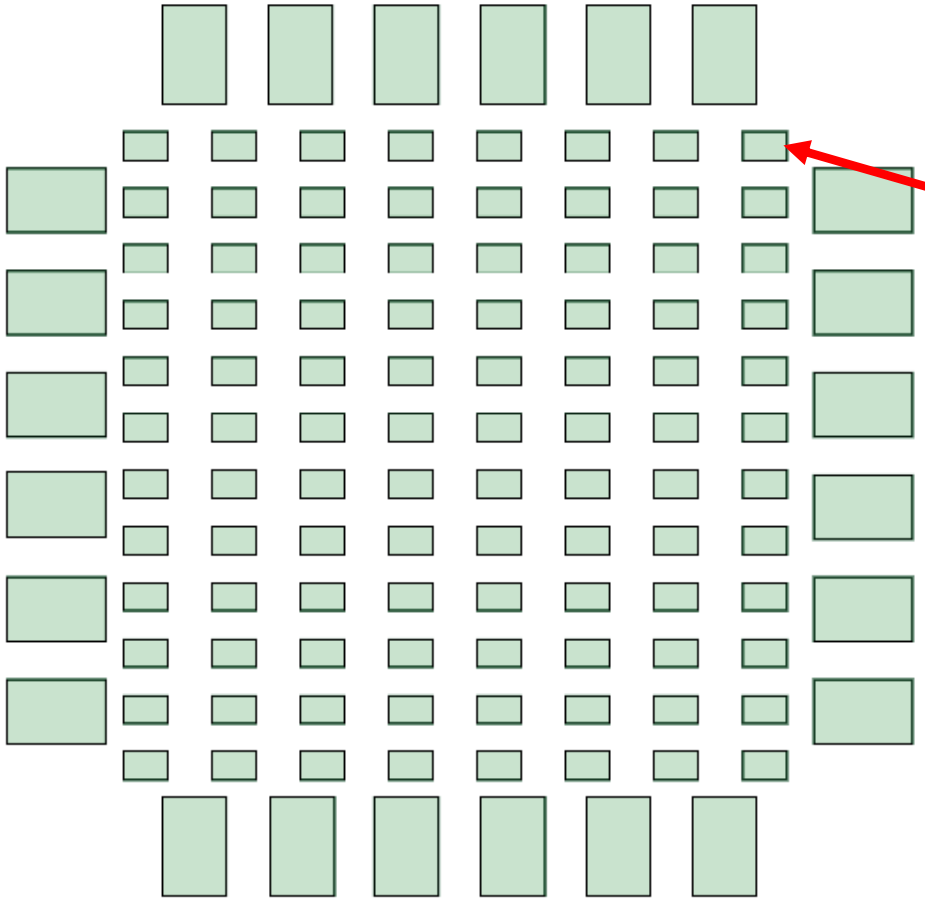
Sea of Transistors
Gate Array
Standard Cell
Block / Macro
Mixed Style

Sea of Transistors



The cell is the transistor itself

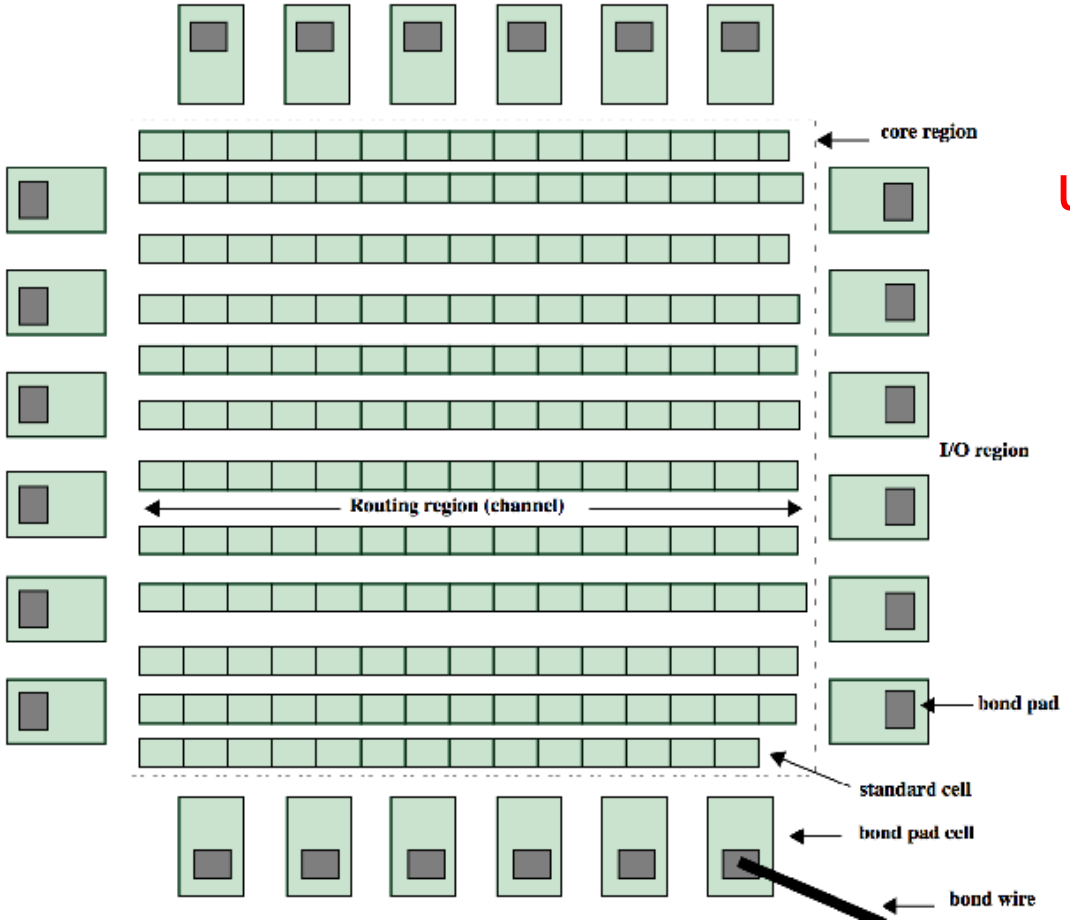
Gate Array



Group transistors into Boolean gates such as NOR2

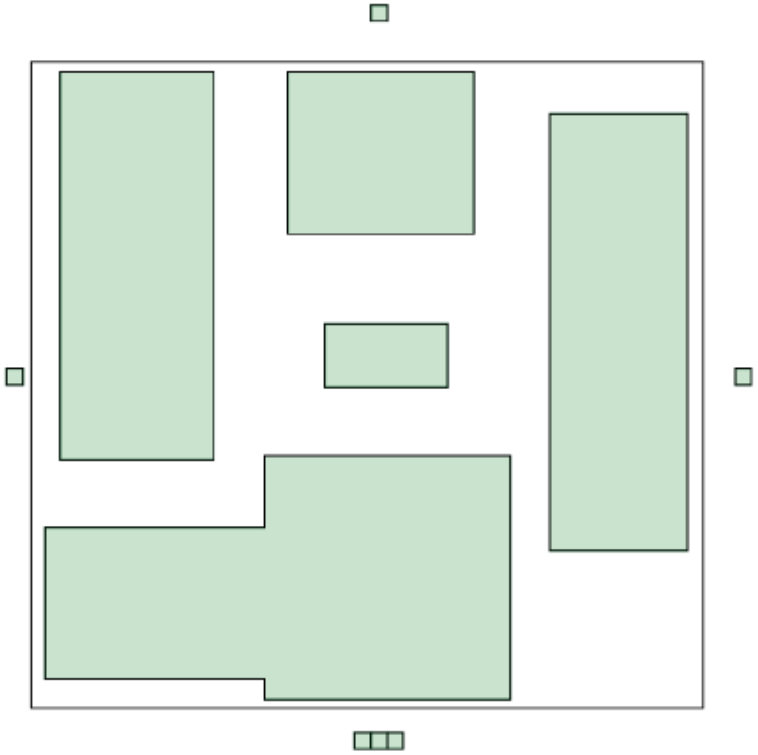
Transistors are fabricated beforehand at known locations

Standard Cells



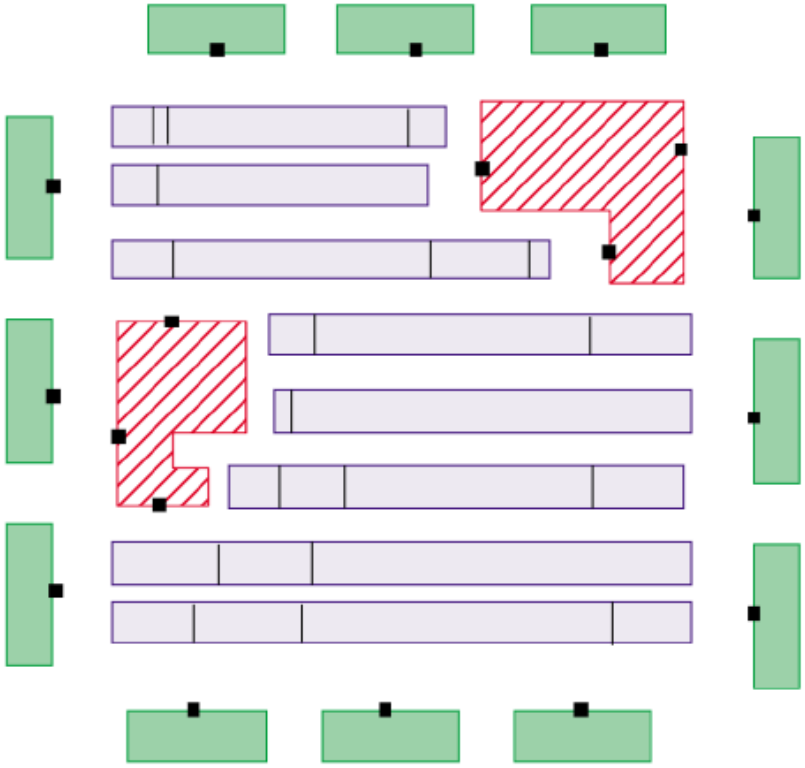
Non uniform cells which are unique to a given design

Block / Macro



Transistors are grouped to build large special purpose functions like RAM, ALUs, etc

Mixed Style



Standard cells + macro blocks

Assembly methods

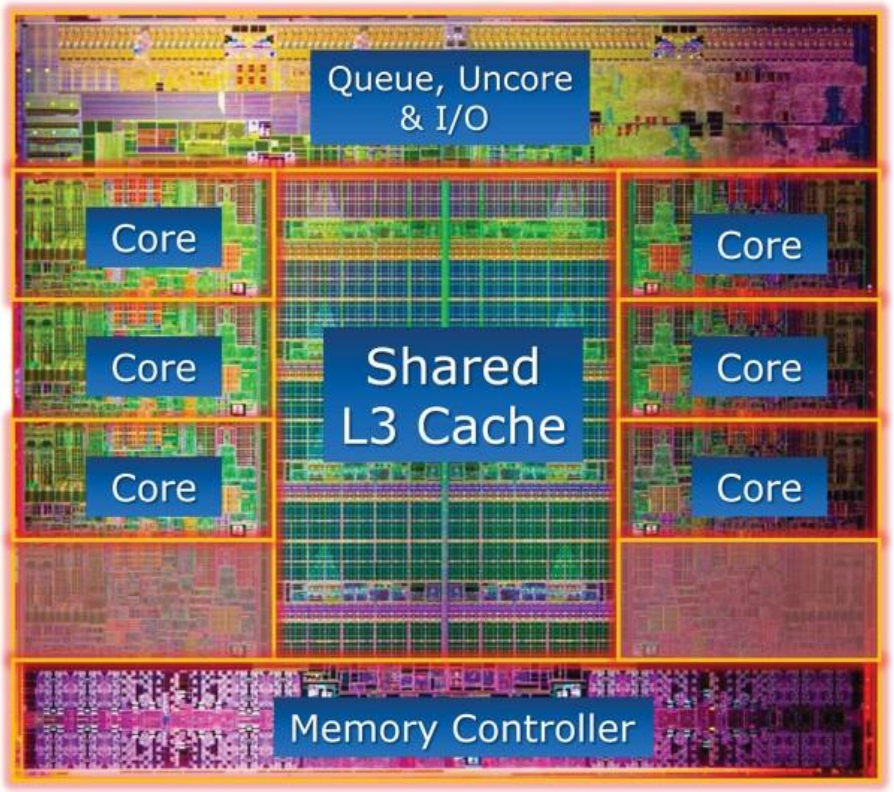
Fixed die
Variable die
Constrained die

Fixed Die

- Dominates industry and academia
- Placement consists of two stages:
  - Coarse
  - Detail
- Global routing
  - Timing / Congestion Driven
- Detail routing
  - Multilevel to increase capacity and fix problems due to abstraction

Fixed Die

- Placement set positions of cells for all time



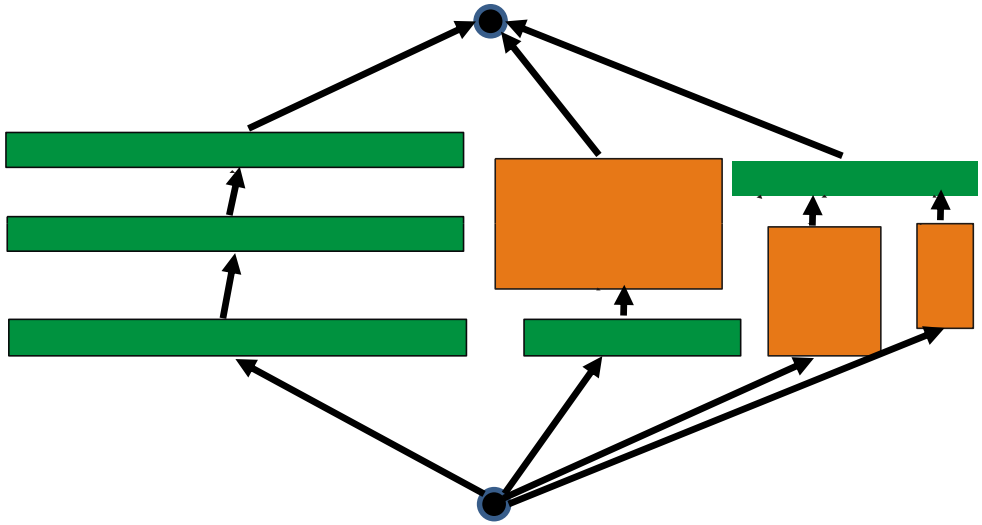
THE INTEL CORE i7-3960X (Sandy Bridge), 2011

Fixed Die Problems

- Rows abut to minimize space
- What happens when detailed router does not complete?
- Where do we add space?

Variable Die

White space is manipulated to insure no overflow
Placement is not fixed.
Relative placement may change as well
Dynamic compaction problem



Y direction constraint graph

Variable Die

- Virtual pin assignment determines orthogonal (vertical) resources
- Partition into virtual regions to determine horizontal resources
- If all pins at center of standard cell
  - Perfect decomposition
  - Global router decouples virtual regions
  - Perfect parallelism – all core regions at once.
  - Blocks/macros complicate things

Variable Die

- Applicable to block level
- Early stages of chip level
- Minimize area
- Trade off # of routing layers versus cost
- Guarantees completion
- Minimizes congestion (controversial)
- Original P&R algorithm using channels

Constrained Die

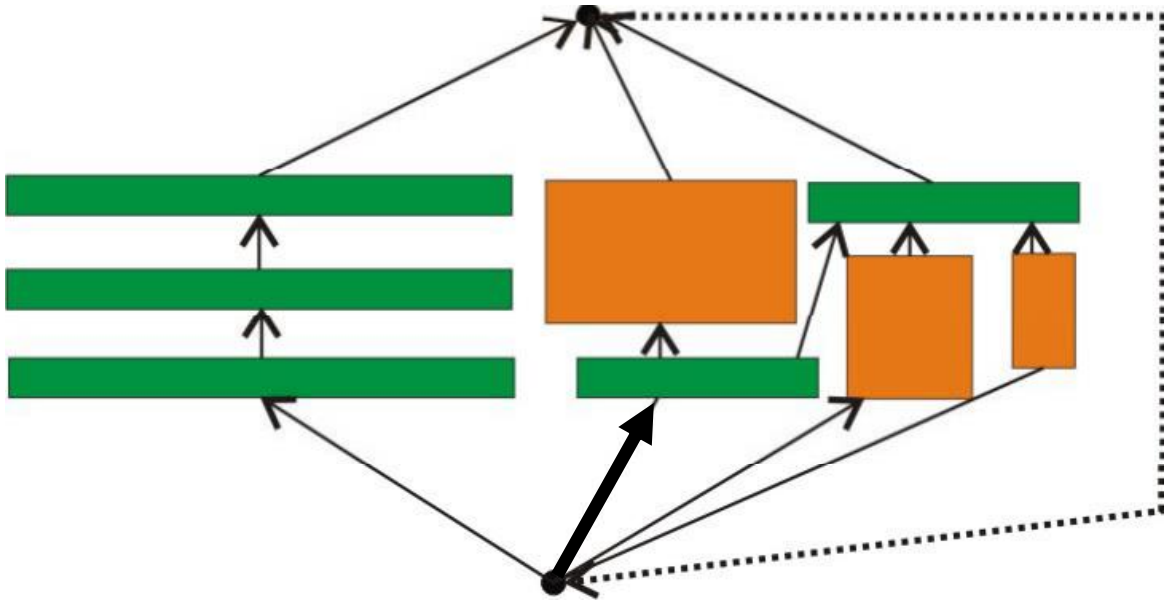
- Chip design process uses iterative refinement not one-shot
- Design space exploration at beginning
  - What is possible?
- As knowledge is acquired, more and more aspects become fixed
  - Die size become fixed
  - I/Os become fixed

Constrained Die

- Designer cares only about interface of block or chip remain constant during iterations
  - Footprint important
  - Physical location of internals irrelevant
  - Design convergence is important (timing)

- Propose Constrained Die Methodology

Constrained Die



Constrained die compaction graph in y direction.
Dotted line shows constrained die fixed edge.

Assembly methods

Fixed die

Variable die

Constrained die

Only Fixed Die Assembly Today

Overcoming the complexity

Abstraction

- How to model and instantiate the components?
- How to group the underlying transistors?
- What is the granularity?
- How to assemble the final die?

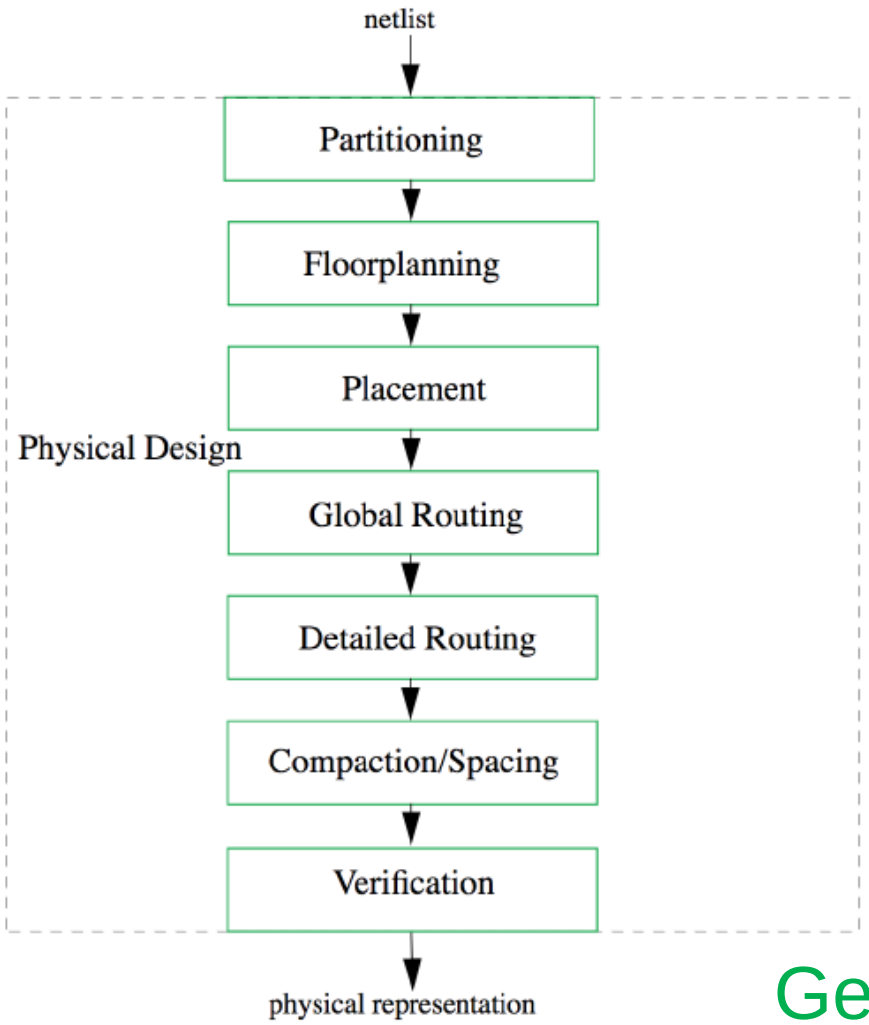
Divide and conquer

Divide and conquer

Break physical design process into steps

Physical Design Flow

- Simplified view
- Does not include clock tree synthesis, buffering, global and detailed placement sub-steps, gate-sizing, or cross-talk minimization, track assignment, …

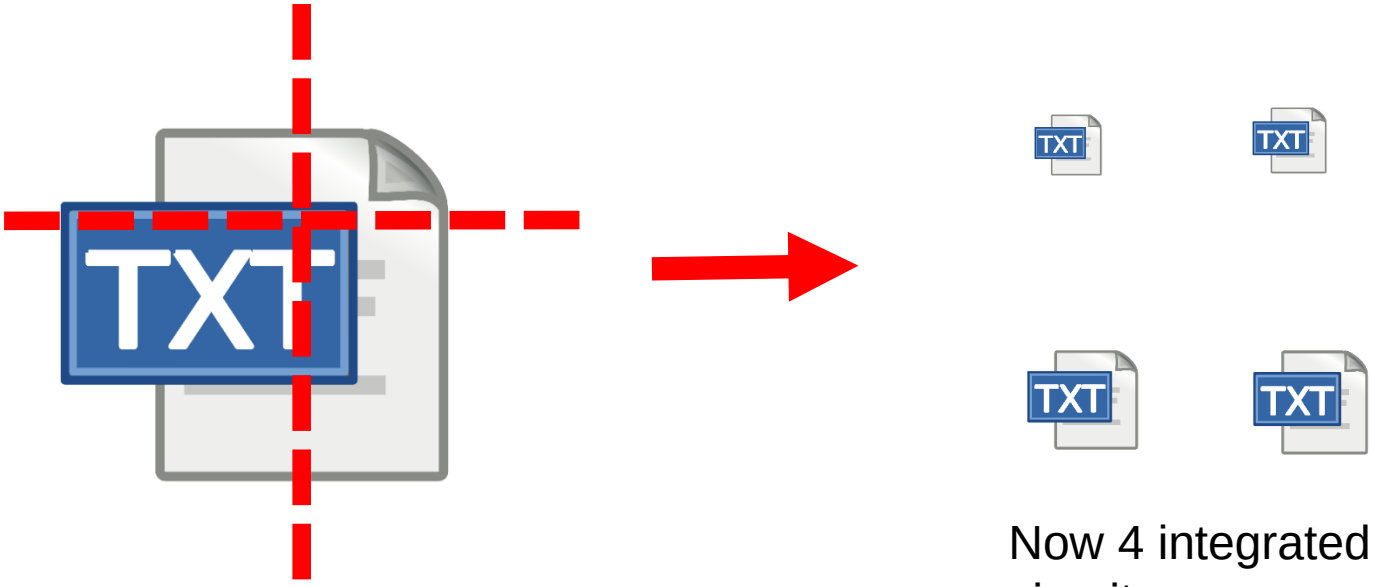


Netlist

Geometries

Partitioning

Cut system such that it fits on a set of dies



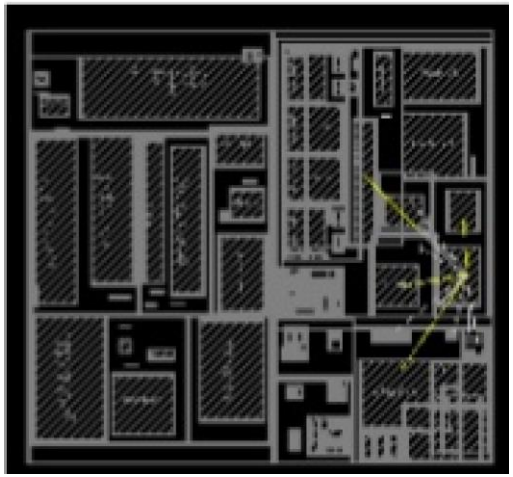
Now 4 integrated circuits

Floorplanning

Tentative placement of major blocks of design

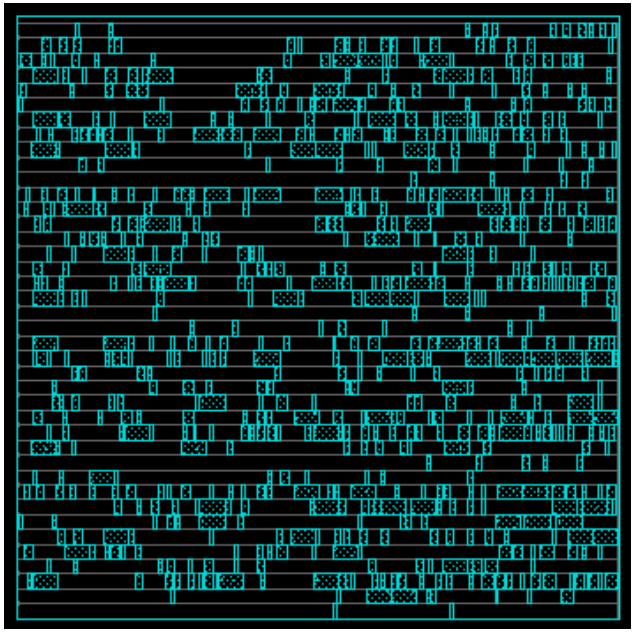
Block placement tool

May change depending on methodology



Placement

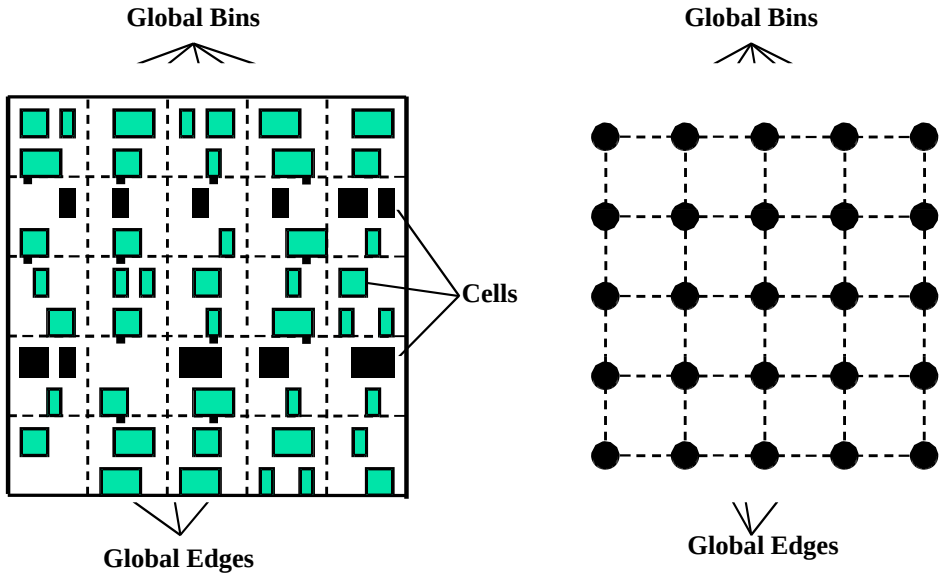
Determine the location of the abstracted cell such that performance specifications are met



Global Routing

Divide wiring of cells into manageable pieces

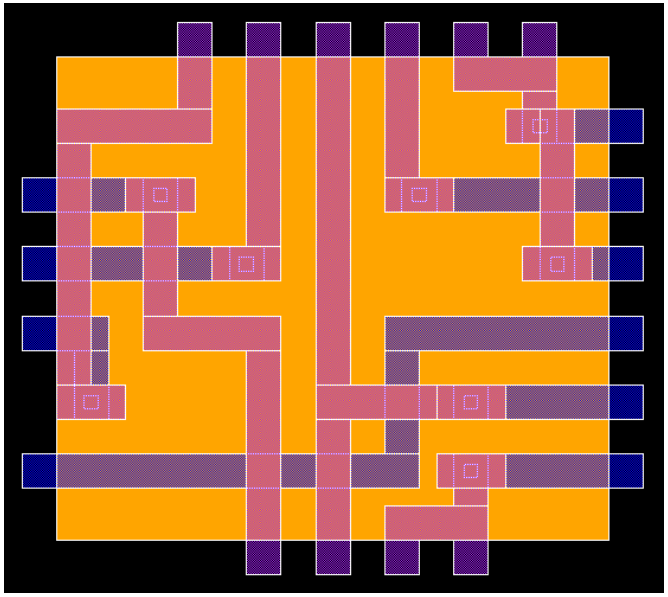
- Assign net segments to physical regions.
- Model regions as a graph.
- Nets are embedded into graph
- Minimize the total overflow on all global edges



From Pan M, Chu C.

Detailed Routing

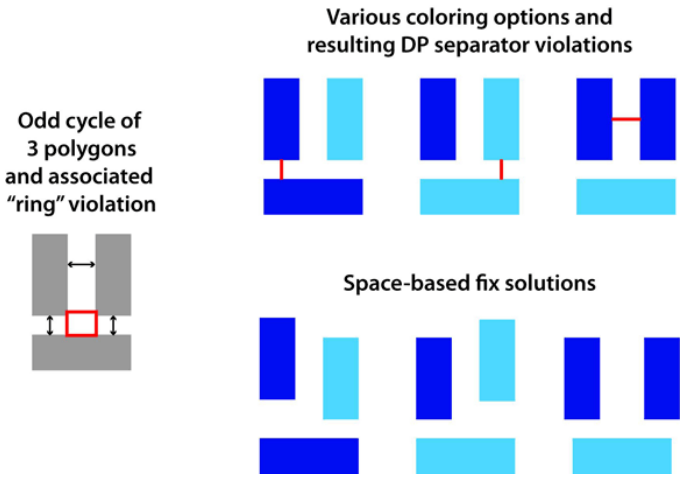
Connect the terminals of cells with rectangles on a set of layers



Switchbox : pins on all four sides of region

Compaction / Spacing

Modify / Ready layout for a given technology
Include fill patterns
Double / Triple patterning stitch and cutting



David Abercrombie, "To Cut Or Not To Cut? That Is The Double Patterning Question", Semiconductor Engineering, Dec 14, 2011

Verification

Make sure specifications are met

Design must be correct

Physically	design rule check (DRC)
Logically	layout versus schematic (LVS)
Temporally	timing analysis
Electrically	electromigration / power delivery
Behaviourally	formal verification