EE6094
CAD for VLSI Design

Chapter 1
Introduction to EDA

Spring 2024
Andy, Yu-Guang Chen
Assistant Professor, Department of EE
National Central University
andyygchen@ee.ncu.edu.tw

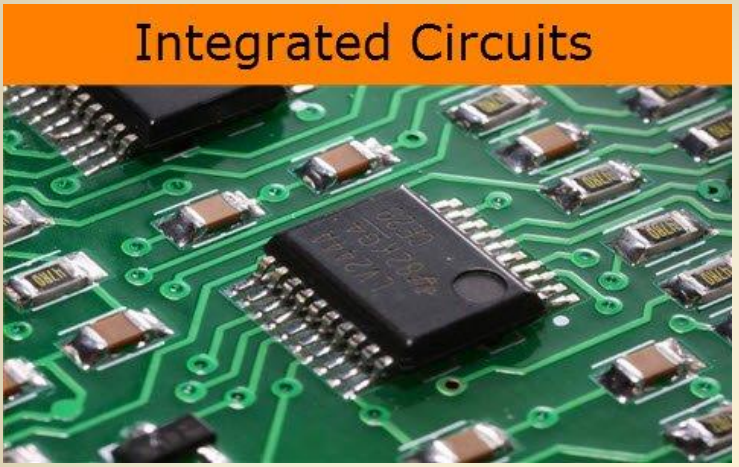
Outline

- From Semiconductor to IC
- VLSI Design Flow
- What is EDA
- Focus on VLSI Design
  - Logic Synthesis
  - Physical Design
- Focus on VLSI Fabrication
  - Lithography
  - Design Rules

Outline

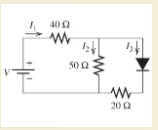
- From Semiconductor to IC
- VLSI Design Flow
- What is EDA
- Focus on VLSI Design
  - Logic Synthesis
  - Physical Design
- Focus on VLSI Fabrication
  - Lithography
  - Design Rules

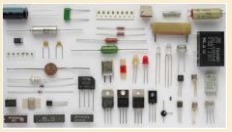
What is An Integrated Circuits (IC)



What is An Integrated Circuits (IC)

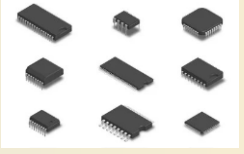
- Electronic circuit
  - An electronic circuit is composed of individual electronic components, such as resistors, transistors, capacitors, inductors and diodes, connected by conductive wires or traces through which electric current can flow.
  - The combination of components and wires allows various simple and complex operations to be performed



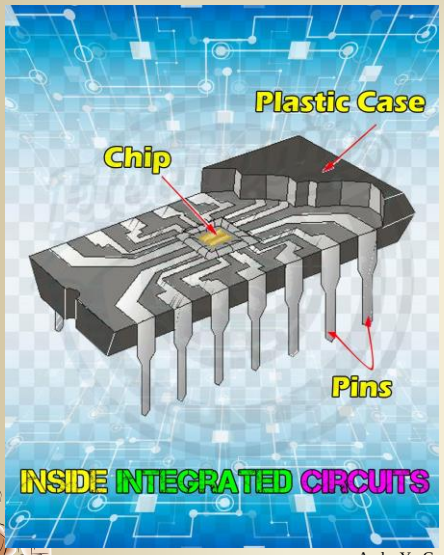


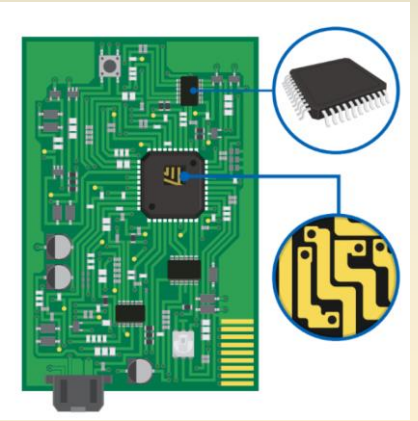
What is An Integrated Circuits (IC)

- Integrated Circuits
  - A set of electronic circuits on one small flat piece (or "chip") of semiconductor material that is normally silicon
  - The integration of large numbers of tiny MOS transistors into a small chip results in circuits that are orders of magnitude smaller, faster, and less expensive



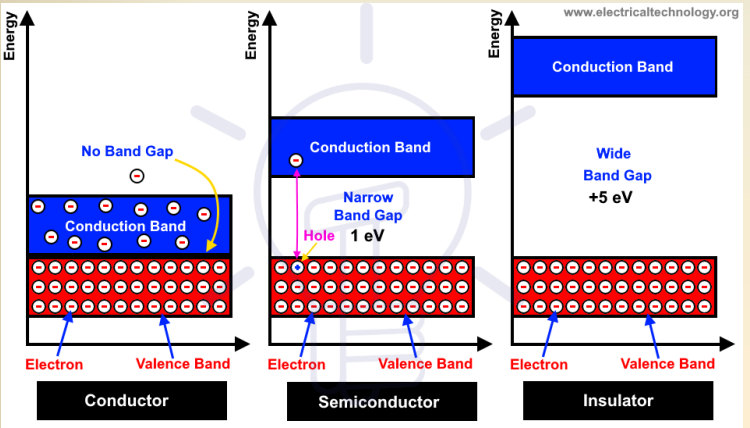
What is An Integrated Circuits (IC)





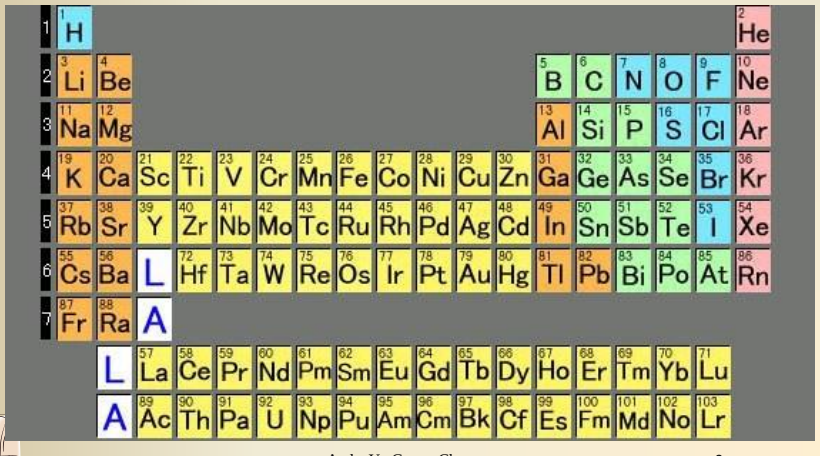
Semiconductor and IC

◆Semiconductor



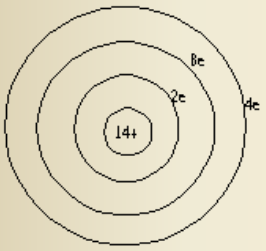
Semiconductor and IC

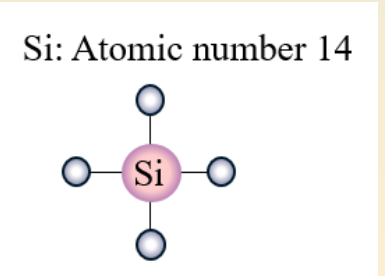
- Recall from your high school…



Semiconductor and IC

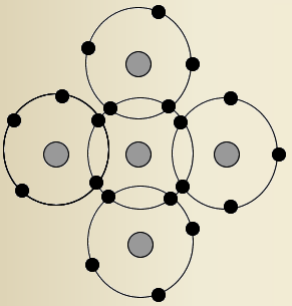
- A silicon atom has 14 electrons around the nucleus
  - there are 4 valence electrons on the outermost orbital

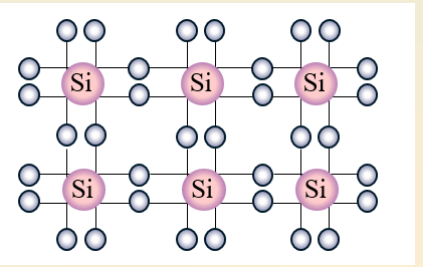




Semiconductor and IC

- When this is made into a single crystal, it can be used as a material for semiconductor products.

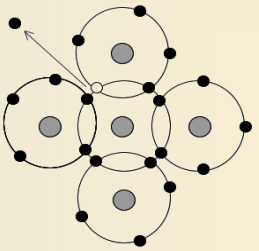




https://www.shindengen.com/products/semi/column/basic/semi/semi_basic.html

Semiconductor and IC

- Semiconductor behaves as an insulator at absolute zero temperature
- With an increase of temperature, the conductivity of semiconductor increases and resistivity decreases

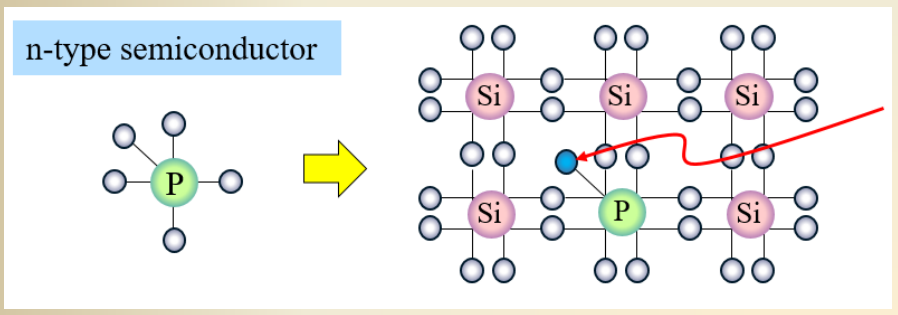


Semiconductor and IC

- Doping silicon with other impurities changes it so it is conductive
- The semiconductor is categorized as a p-type or n-type depending on the type of impurities that are doped
- Junctions based on the p-types and n-types are integrated into one chip in order to use it as an electronic component

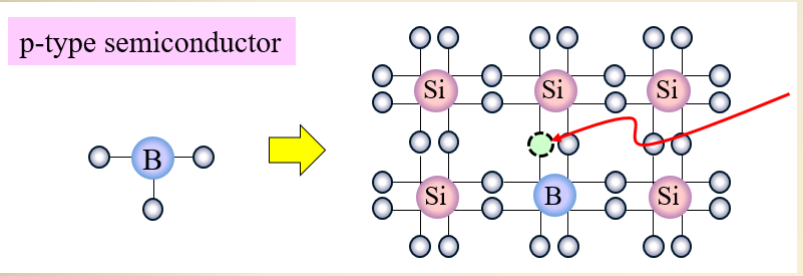
Semiconductor and IC

- Group V: extra electron (n-type)

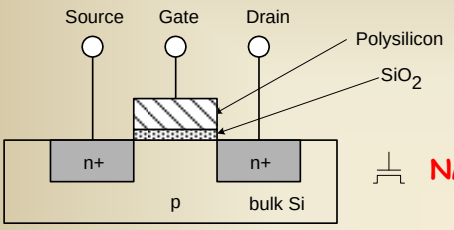


Semiconductor and IC

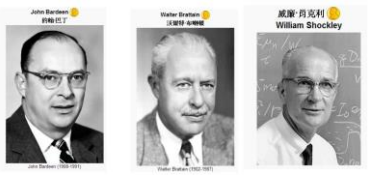
◆Group III: missing electron, called hole (p-type)

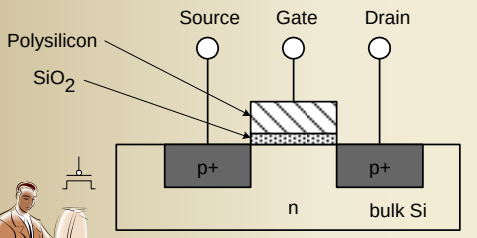


Basic Transistor Design



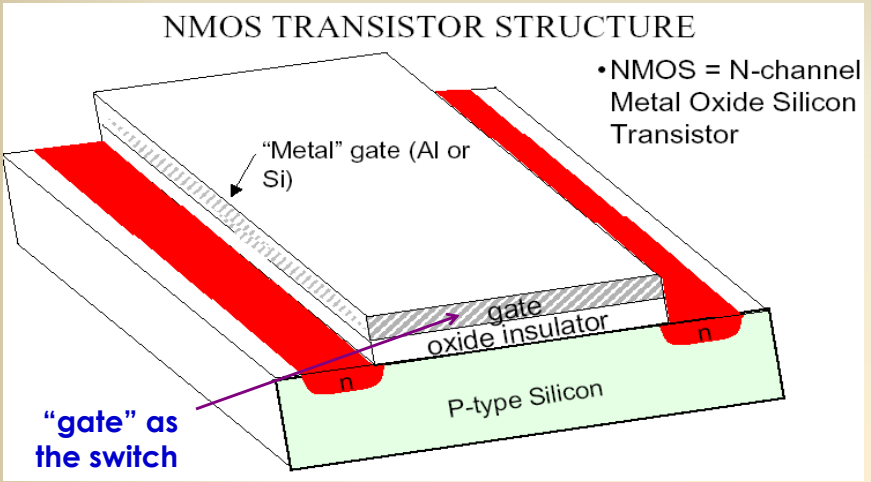
NMOS





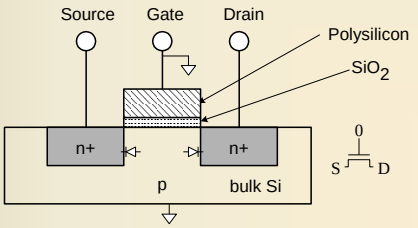
PMOS

Electronic Switch



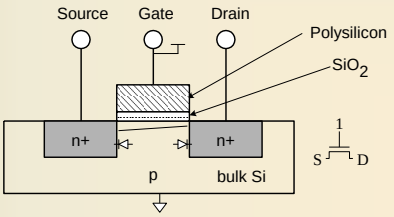
N-type Transistor Design

- Body is commonly tied to ground (0 V)
- When the gate is at a low voltage:
  - P-type body is at low voltage
  - Source-body and drain-body diodes are OFF
  - No current flows, transistor is OFF



N-type Transistor Design

- When the gate is at a high voltage:
  - Positive charge on gate of MOS capacitor
  - Negative charge attracted to body
  - Inverts a channel under gate to n-type
  - Now current can flow through n-type silicon from source through channel to drain, transistor is ON



nFET family of curves

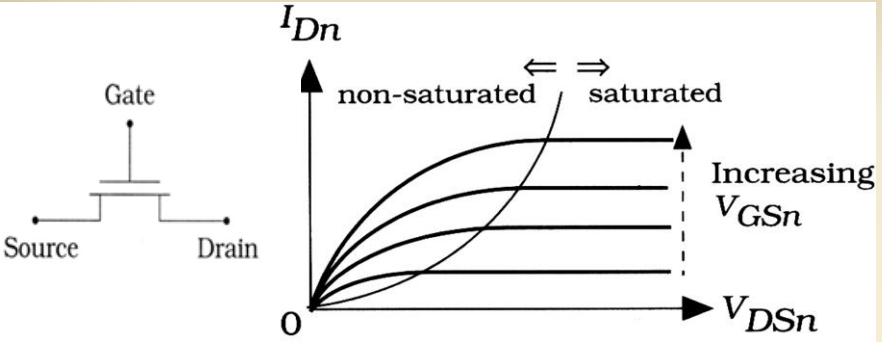


Figure 6.14 nFET family of curves

nFET Current-Voltage Equations

- I_{Dn} vs V_{DSn} (V_{GSn}fixed, $V_{GSn} > V_{Tn}$)
- Small value of V_{DSn} : parabola
  - $I_{Dn} = \frac{\beta_n}{2}[2(V_{GSn} - V_{Tn})V_{DSn} - {V_{DSn}}^2]$
- Saturation voltage
  - $V_{sat} = V_{DSn}|_{peak\ current} = V_{GSn} - V_{Tn}$
- Large value of V_{DSn} : $V_{DSn} \geq V_{sat}$
  - Saturation current
  - The largest I_{Dn} for a given V_{GSn}
  - $I_{Dn} = \frac{\beta_n}{2}(V_{GSn} - V_{Tn})^2$
- Increase slightly for $V_{DSn} \geq V_{sat}$
  - $I_{Dn} = \frac{\beta_n}{2}(V_{GSn} - V_{Tn})^2[1 + \lambda(V_{DSn} - V_{sat})]$
  - λ: empirical quantity called channel-length modulation parameter (V^{-1})
  - Use in simulation but not in hand calculation

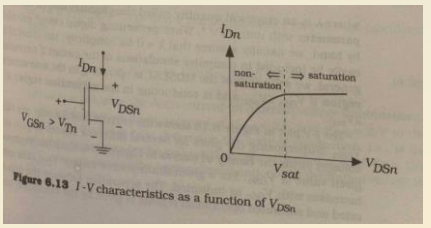
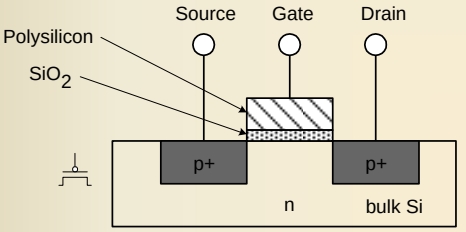


Figure 6.13 I-V characteristics as a function of V_{DSn}

P-type Transistor Design

- Similar, but doping and voltages reversed
  - Body tied to high voltage (V_{DD})
  - Gate low: transistor ON
  - Gate high: transistor OFF
  - Bubble indicates inverted behavior



Course Mapping

- In _____ course, we learned the knowledge about physical characteristics (voltage, current, …) of electronic components

Transistor As a Switch

Only two situations：turn ON、turn OFF

1 0

Physical Design

- Layout

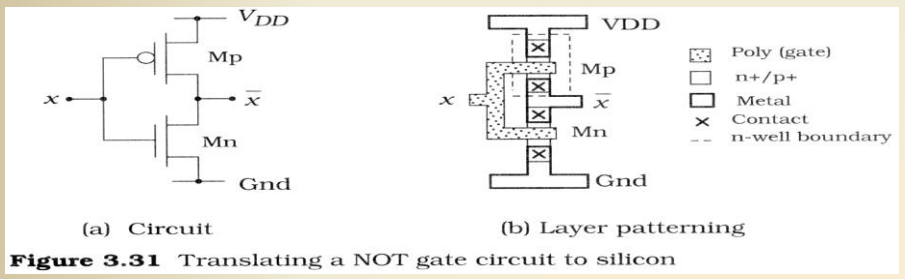


Figure 3.31 Translating a NOT gate circuit to silicon

Physical Design

- Layout

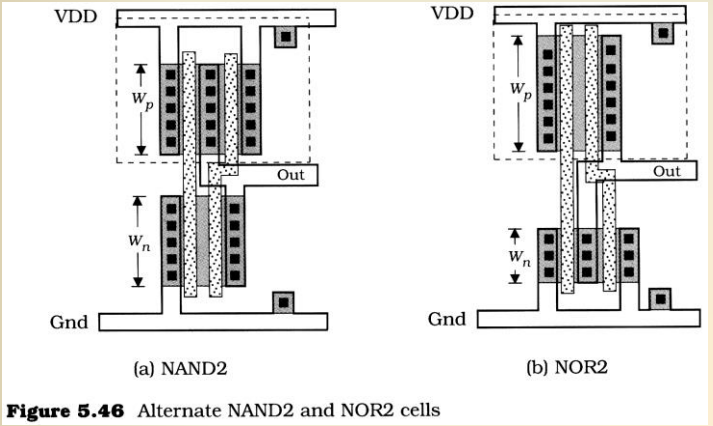


Figure 5.46 Alternate NAND2 and NOR2 cells

Physical Design

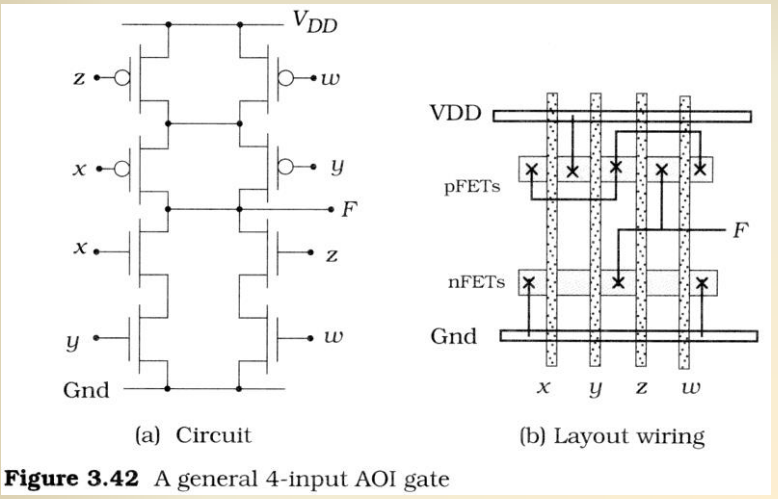
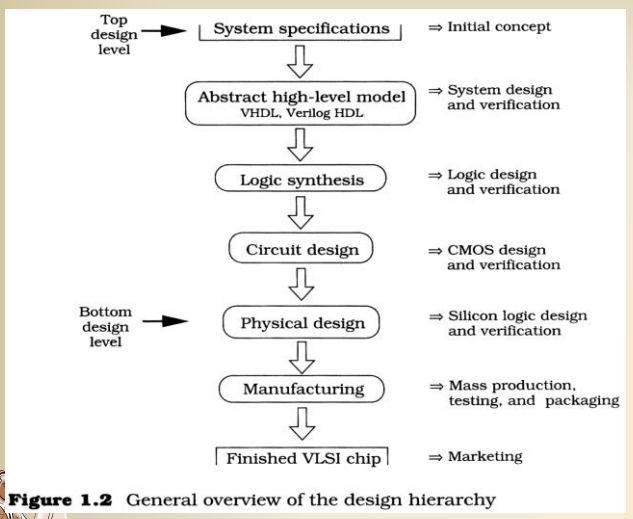


Figure 3.42 A general 4-input AOI gate

A general overview of design hierarchy



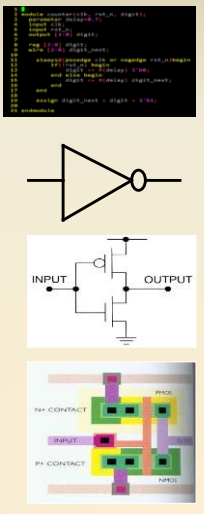
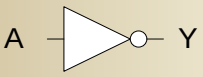


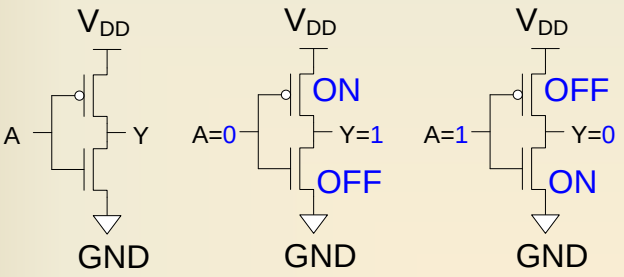
Figure 1.2 General overview of the design hierarchy

By Combining Switches…

- 1 PMOS + 1 NMOS

A	Y
0	1
1	0

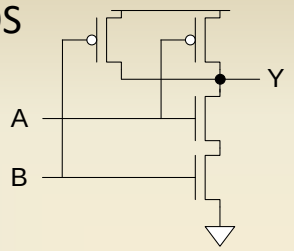


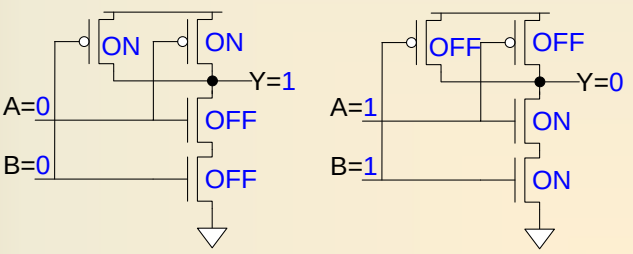


By Combining Switches…

- 2 PMOS + 2 NMOS

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

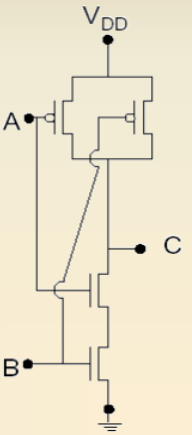




Logic Gates

- CMOS NAND:

A	B	A B	C= $\overline{A\,B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

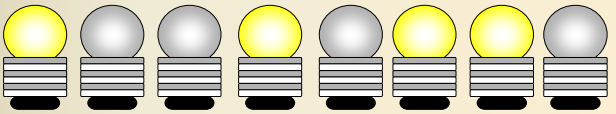


Course Mapping

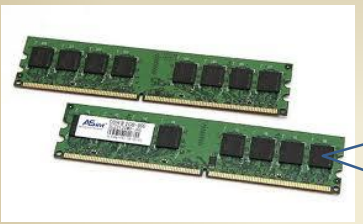
- In _____ course, we learned the VLSI design procedure, layout structure, and circuit analysis?

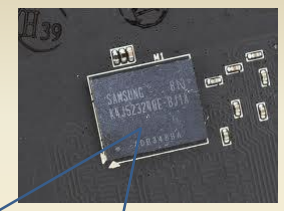
How Computer Stores Data

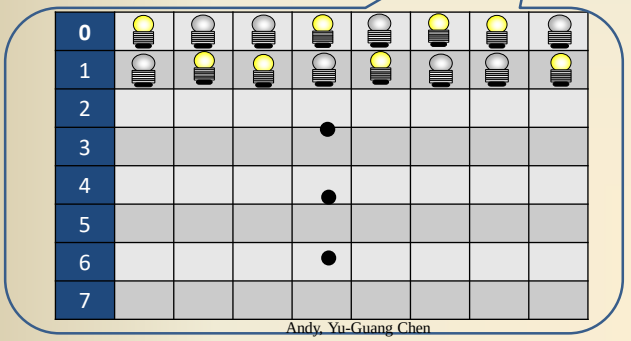
- Bit
  - binary digit
  - the basic unit of information in computing
- Byte
  - a unit of digital information in computing that consists of eight bits



How Computer Stores Data







Summary of Logic Gates (p.77)

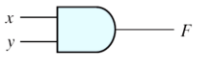
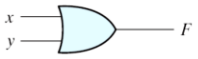
Name	Graphic symbol	Algebraic function	Truth table
AND		$F = xy$	$x\ y \mid F$: 0 0 | 0; 0 1 | 0; 1 0 | 0; 1 1 | 1
OR		$F = x + y$	$x\ y \mid F$: 0 0 | 0; 0 1 | 1; 1 0 | 1; 1 1 | 1
Inverter		$F = x'$	$x \mid F$: 0 | 1; 1 | 0
Buffer		$F = x$	$x \mid F$: 0 | 0; 1 | 1

Figure 2.5 Digital logic gates

Summary of Logic Gates (p.77)

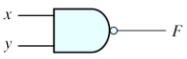
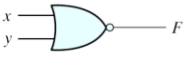
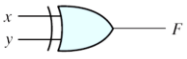
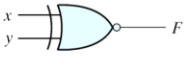
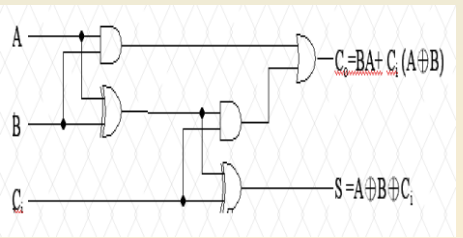
NAND		$F = (xy)'$	$x\ y \mid F$: 0 0 | 1; 0 1 | 1; 1 0 | 1; 1 1 | 0
NOR		$F = (x + y)'$	$x\ y \mid F$: 0 0 | 1; 0 1 | 0; 1 0 | 0; 1 1 | 0
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	$x\ y \mid F$: 0 0 | 0; 0 1 | 1; 1 0 | 1; 1 1 | 0
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	$x\ y \mid F$: 0 0 | 1; 0 1 | 0; 1 0 | 0; 1 1 | 1

Figure 2.5 Digital logic gates

By Combining Gates…

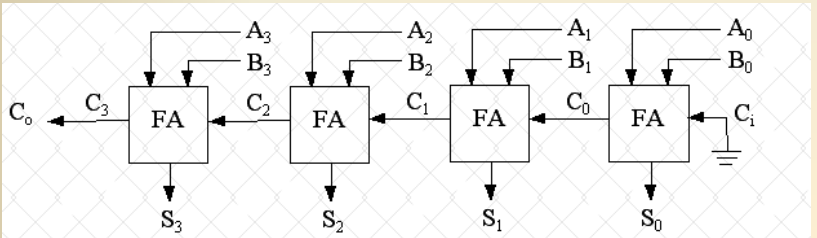
- 2 AND + 2 XOR + 1 OR

C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



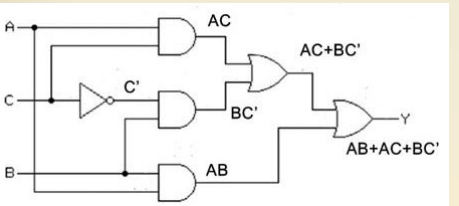
By Combining Components

- 4 Full Adders
  - Ripple Carry Adder



Digital System Design

- Function ➔ Logic hardware



$Y = ABC+ABC'+AB'C+A'BC'$

$Y' = (A'+B'+C')(A'+B'+C)(A'+B+C')(A+B'+C)$

$= 111 \quad 110 \quad 101 \quad 010$

$= M7 \bullet M6 \bullet M5 \bullet M2$

$Y = M4 \bullet M3 \bullet M1 \bullet M0$

Review: Comb. Design Proc.

- State the problem (system specification (spec.))
- Determine the inputs and outputs
- The input and output variables are assigned symbols
- Derive the truth table
- Derive the simplified Boolean functions
- Draw the logic diagram and verify the correctness

Code Conversion Example

- BCD to excess-3 code
  - The truth table

Table 4.2
Truth Table for Code-Conversion Example

Input BCD				Output Excess-3 Code			
A	**B**	**C**	**D**	**w**	**x**	**y**	**z**
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

BCD Maps

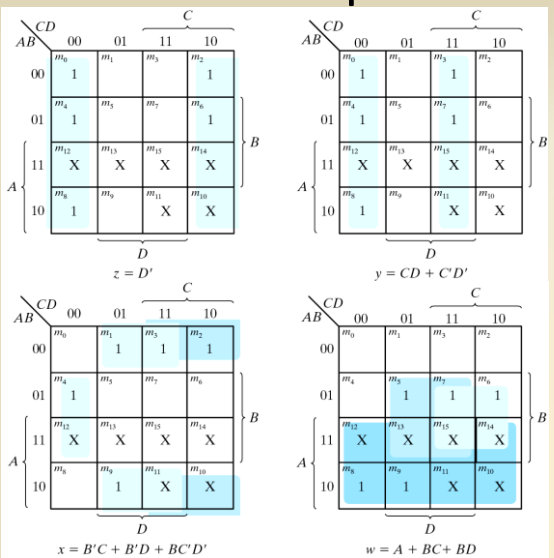
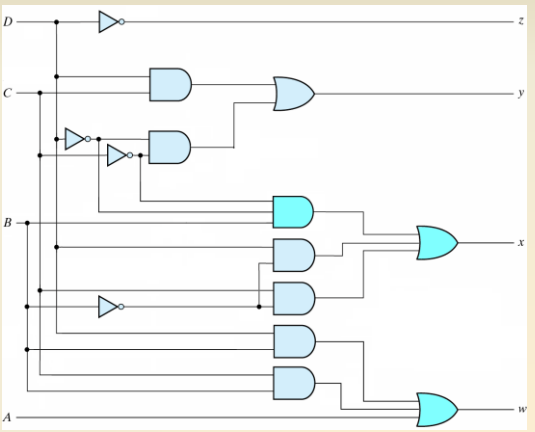


Figure 4.3 Maps for BCE to Excess-3 Code Converter

BCD Functions

- The simplified functions
  - $z = D'$
  - $y = CD + C'D'$
  - $x = B'C + B'D + BC'D'$
  - $w = A + BC + BD$
- Another implementation
  - $z = D'$
  - $y = CD + (C + D)'$
  - $x = B'(C + D) + B(C + D)'$
  - $w = A + B(C + D)$



Review: Seq. Design Proc.

- Design Procedure for sequential circuit
  - The word description of the circuit behavior to get a state diagram
  - State reduction if necessary
  - Assign binary values to the states
  - Obtain the binary-coded state table
  - Choose the type of flip-flops
  - The passcode is placement
  - Derive the simplified flip-flop input equations and output equations
  - Draw the logic diagram

Synthesis Using D flip-flops (1/4)

- An example state diagram and state table: design a circuit to detect a sequence of three or more consecutive 1's

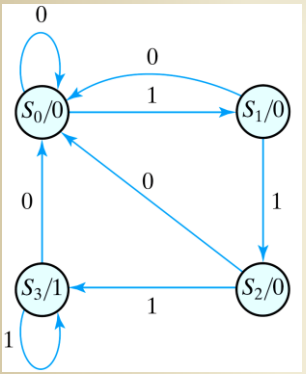


Fig. 5.27 State diagram for sequence detector

Table 5.11
State Table for Sequence Detector

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

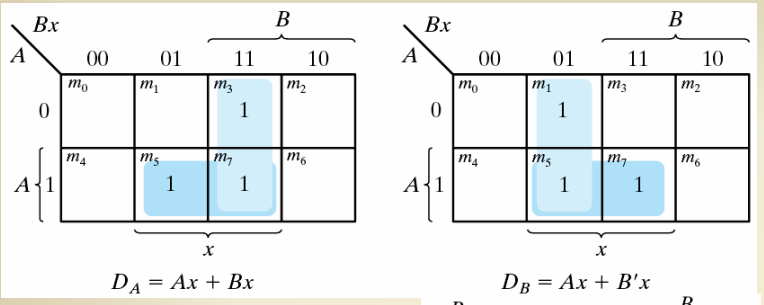
Synthesis Using D flip-flops (2/4)

Table 5.11
State Table for Sequence Detector

Present State		Input	Next State		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	1	1	1	1

- **The flip-flop input equations**
  - $A(t+1) = D_A(A, B, x) = \Sigma(3, 5, 7)$
  - $B(t+1) = D_B(A, B, x) = \Sigma(1, 5, 7)$
- **The output equation**
  - $y(A, B, x) = \Sigma(6, 7)$
- **Logic minimization using the *K*-map**
  - $D_A = Ax + Bx$
  - $D_B = Ax + B'x$
  - $y = AB$

Synthesis Using D flip-flops (3/4)



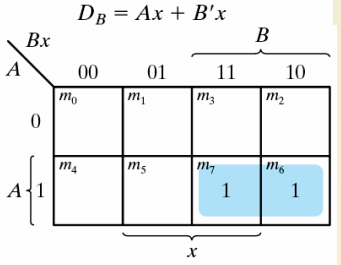
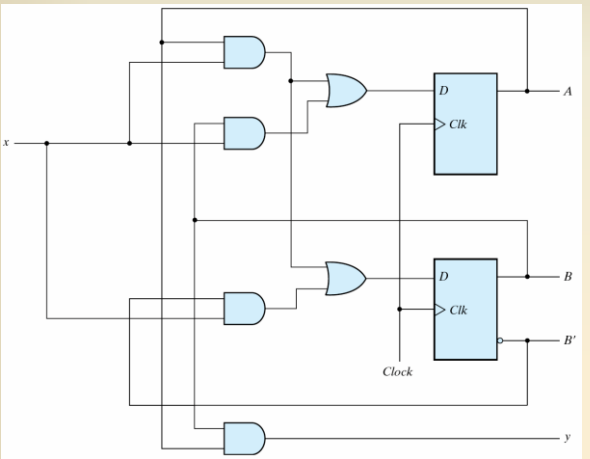


Fig. 5.28 Maps for sequence detector

Synthesis Using D flip-flops (4/4)

- The logic diagram of sequence detector



- $D_A = Ax + Bx$
- $D_B = Ax + B'x$
- $y = AB$

Fig. 5.29 Logic diagram of sequence detector

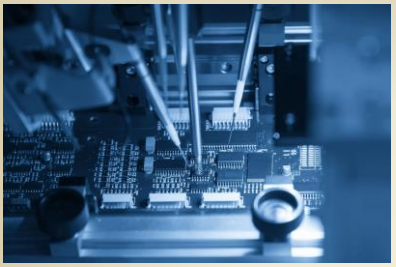
Course Mapping

- In _____ course, we learned basic elements of digital logic gates and how to implement simple combinational /sequential logics?

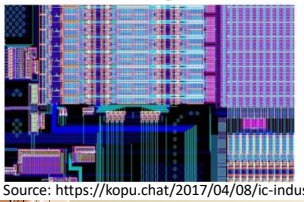
Outline

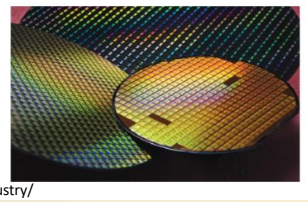
- From Semiconductor to IC
- VLSI Design Flow
- What is EDA
- Focus on VLSI Design
  - Logic Synthesis
  - Physical Design
- Focus on VLSI Fabrication
  - Lithography
  - Design Rules

How are ICs made?



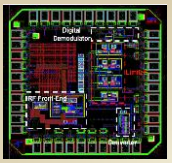
IC Design → IC Manufacture → IC Packing/Testing

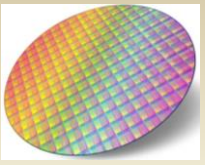




Source: https://kopu.chat/2017/04/08/ic-industry/

Global IC Supply Chain





IC Design → IC Manufacture → IC Packing/Testing

MEDIATEK

NOVATEK

REALTEK

Qualcomm

tsmc

SAMSUNG

UMC

GLOBALFOUNDRIES

日月光集團

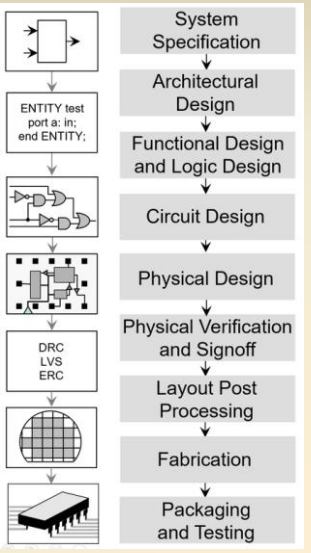
Amkor Technology

IDM (Integrated Device Manufacturer)

intel

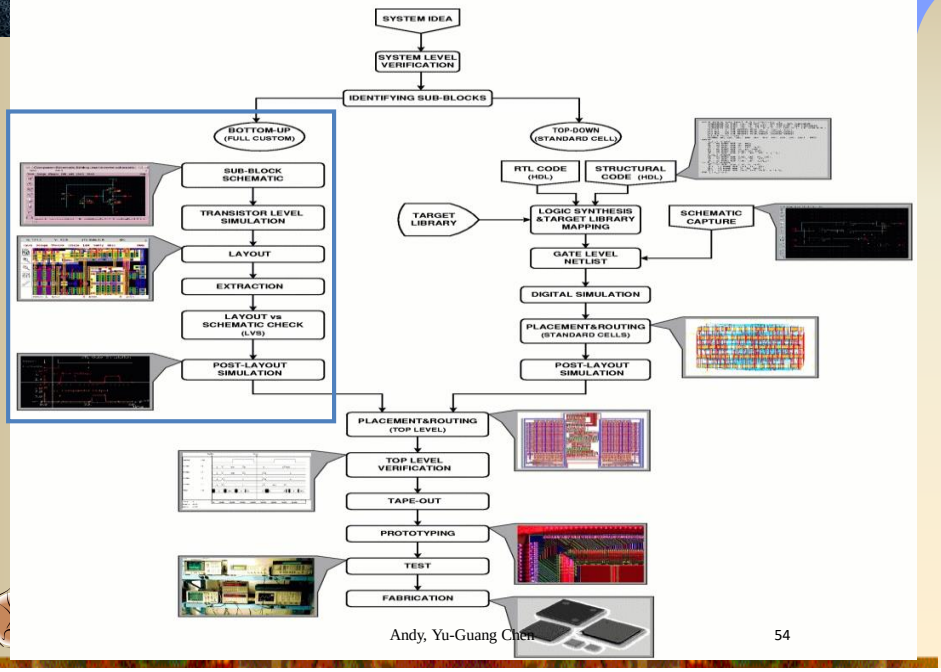
IC Design Flow

```
input in;
output out;
output [7:0] counter;

reg [7:0] counter;

wire counter_overflow = (counter == 8'hff);
wire counter_underflow = (counter == 8'h00);

always @(posedge clk or negedge xreset)
begin
        if (xreset == 0)
                counter <= 0;
        else if (en && in && !counter_overflow)
                counter <= counter + 1;
        else if (en && !in && !counter_underflow)
                counter <= counter - 1;
        else
                counter <= counter;
end
```



https://www.wikiwand.com/en/Integrated_circuit_design

VLSI DESIGN FLOW



SUB-BLOCK SCHEMATIC

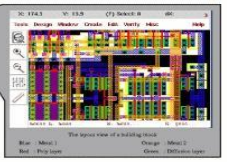
Transistor-level schematic drawings of the circuit blocks are created in Schematic Editor.

TRANSISTOR LEVEL SIMULATION

SPICE(or equivalent) simulation of circuit blocks is used to verify their functionality.

LAYOUT

Mask-layout of all circuit blocks are created in Layout Editor.

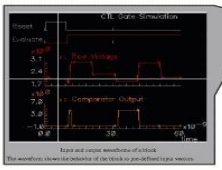


EXTRACTION

Actual device dimensions and parasitic parameters are determined from mask layout.

LAYOUT vs SCHEMATIC CHECK (LVS)

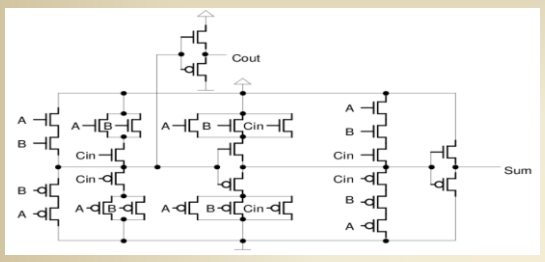
Automatic comparison of mask layout and circuit schematic.

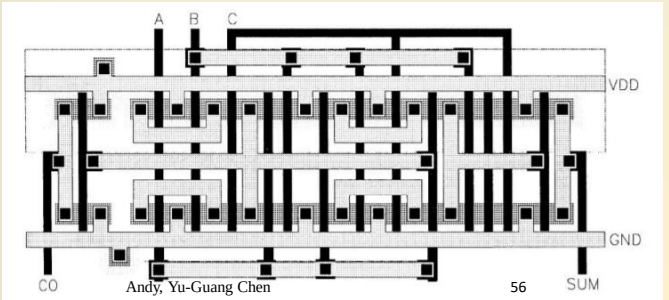


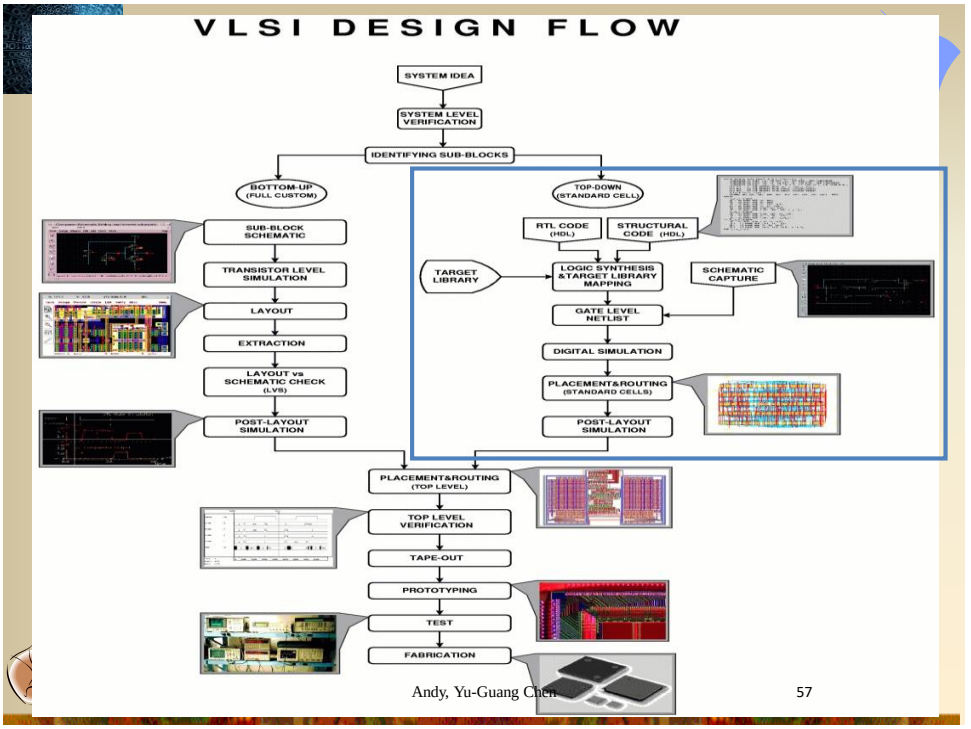
POST-LAYOUT SIMULATION

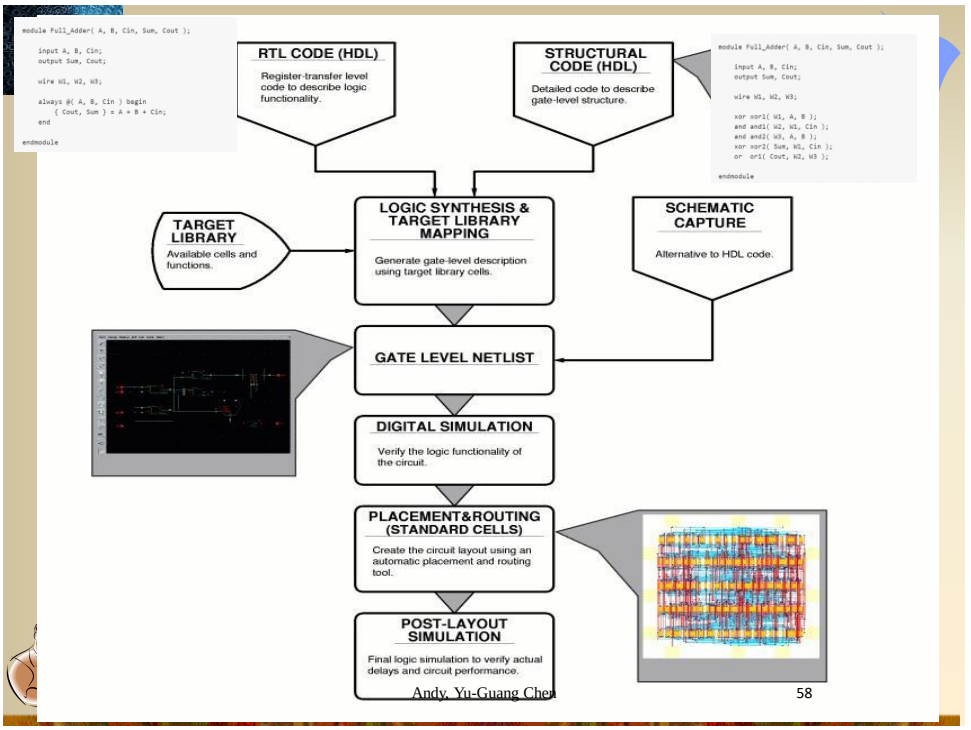
Final SPICE simulation of the circuit of the circuit blocks using extracted parameters.

One-bit Mirror Adder Layout

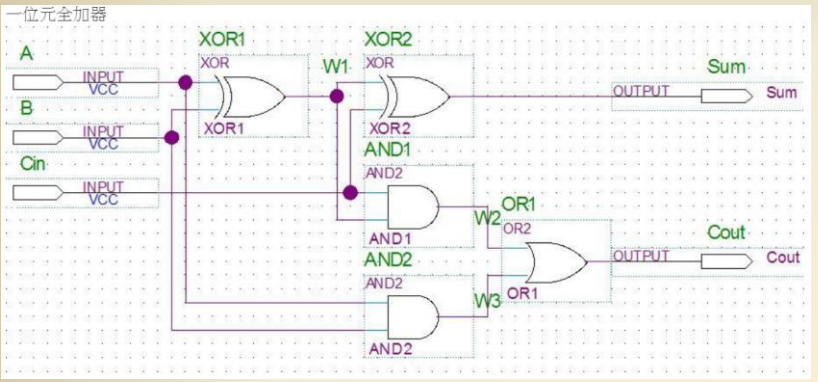


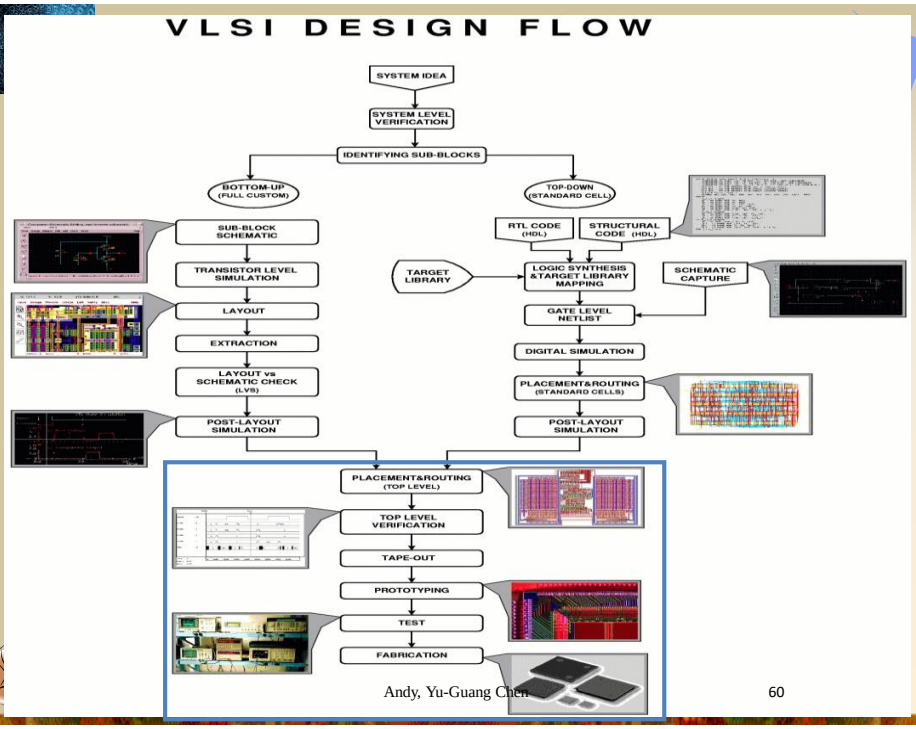


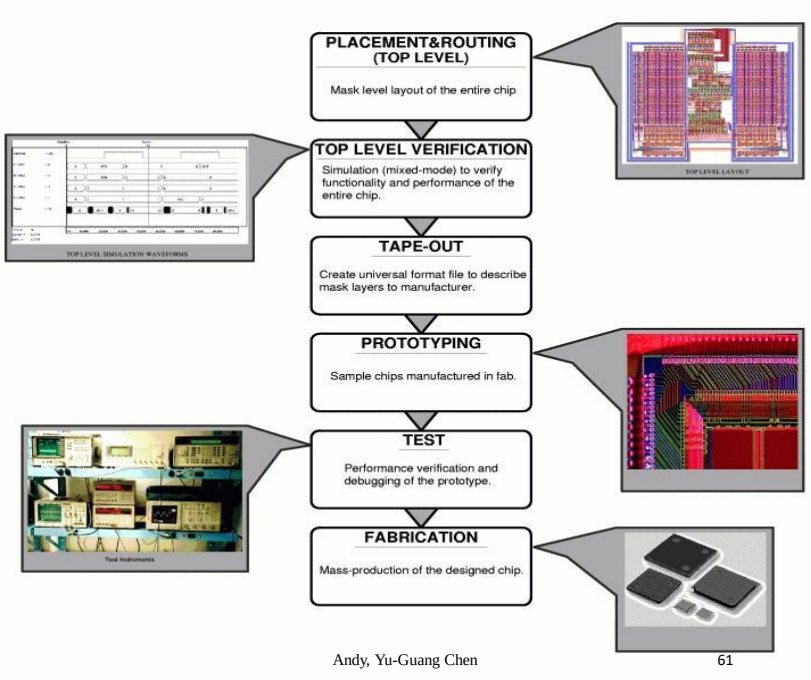

VLSI DESIGN FLOW
SYSTEM IDEA
SYSTEM LEVEL VERIFICATION
IDENTIFYING SUB-BLOCKS
BOTTOM-UP (FULL CUSTOM)
SUB-BLOCK SCHEMATIC
TRANSISTOR LEVEL SIMULATION
LAYOUT
EXTRACTION
LAYOUT vs SCHEMATIC CHECK (LVS)
POST-LAYOUT SIMULATION
TOP-DOWN (STANDARD CELL)
RTL CODE (HDL)
STRUCTURAL CODE (HDL)
TARGET LIBRARY
LOGIC SYNTHESIS &TARGET LIBRARY MAPPING
SCHEMATIC CAPTURE
GATE LEVEL NETLIST
DIGITAL SIMULATION
PLACEMENT&ROUTING (STANDARD CELLS)
POST-LAYOUT SIMULATION
PLACEMENT&ROUTING (TOP LEVEL)
TOP LEVEL VERIFICATION
TAPE-OUT
PROTOTYPING
TEST
FABRICATION

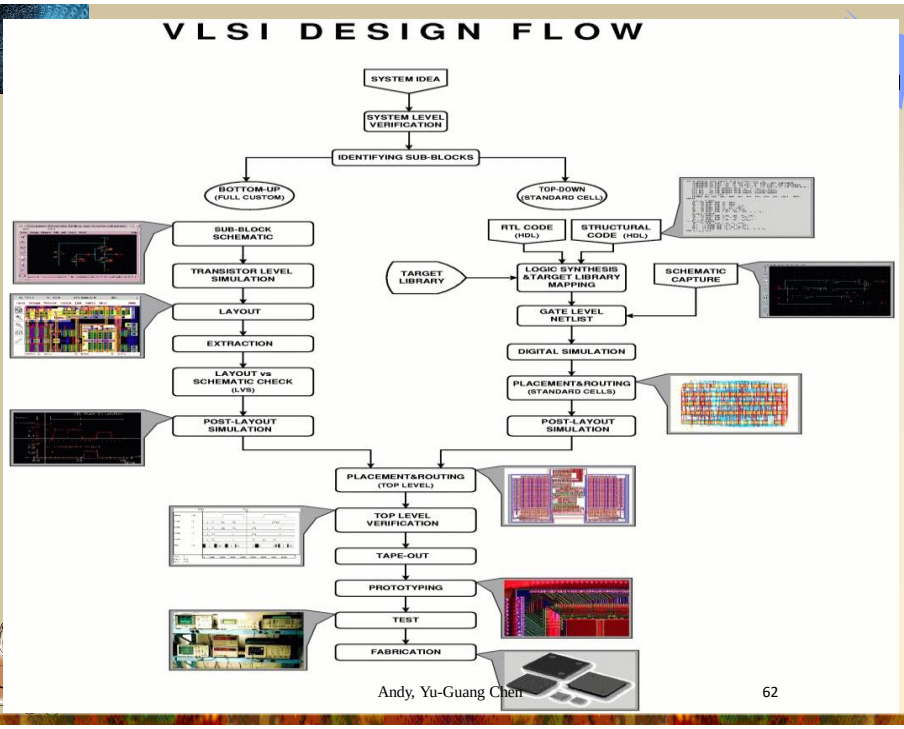

RTL CODE (HDL)
Register-transfer level code to describe logic functionality.
STRUCTURAL CODE (HDL)
Detailed code to describe gate-level structure.
TARGET LIBRARY
Available cells and functions.
LOGIC SYNTHESIS & TARGET LIBRARY MAPPING
Generate gate-level description using target library cells.
SCHEMATIC CAPTURE
Alternative to HDL code.
GATE LEVEL NETLIST
DIGITAL SIMULATION
Verify the logic functionality of the circuit.
PLACEMENT&ROUTING (STANDARD CELLS)
Create the circuit layout using an automatic placement and routing tool.
POST-LAYOUT SIMULATION
Final logic simulation to verify actual delays and circuit performance.

One Bit Full Adder






PLACEMENT&ROUTING
(TOP LEVEL)
Mask level layout of the entire chip
TOP LEVEL VERIFICATION
Simulation (mixed-mode) to verify functionality and performance of the entire chip.
TAPE-OUT
Create universal format file to describe mask layers to manufacturer.
PROTOTYPING
Sample chips manufactured in fab.
TEST
Performance verification and debugging of the prototype.
FABRICATION
Mass-production of the designed chip.


VLSI DESIGN FLOW
SYSTEM IDEA
SYSTEM LEVEL VERIFICATION
IDENTIFYING SUB-BLOCKS
BOTTOM-UP (FULL CUSTOM)
TOP-DOWN (STANDARD CELL)
SUB-BLOCK SCHEMATIC
TRANSISTOR LEVEL SIMULATION
LAYOUT
EXTRACTION
LAYOUT vs SCHEMATIC CHECK (LVS)
POST-LAYOUT SIMULATION
RTL CODE (HDL)
STRUCTURAL CODE (HDL)
TARGET LIBRARY
LOGIC SYNTHESIS &TARGET LIBRARY MAPPING
SCHEMATIC CAPTURE
GATE LEVEL NETLIST
DIGITAL SIMULATION
PLACEMENT&ROUTING (STANDARD CELLS)
POST-LAYOUT SIMULATION
PLACEMENT&ROUTING (TOP LEVEL)
TOP LEVEL VERIFICATION
TAPE-OUT
PROTOTYPING
TEST
FABRICATION

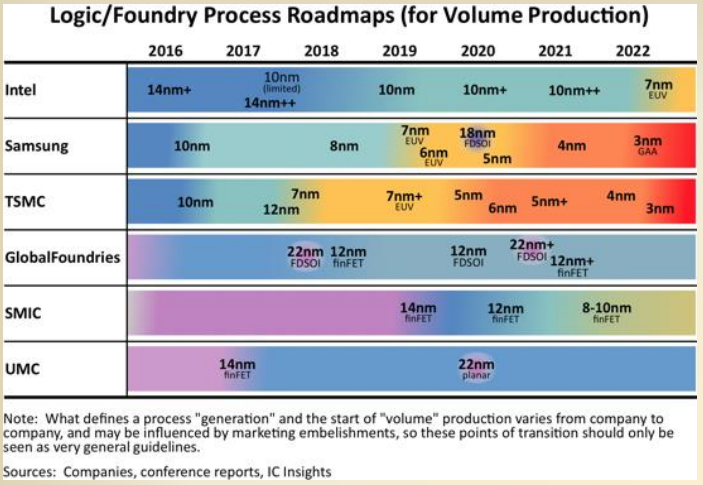
Design Actions

- **Synthesis:** increasing information about the design by providing more detail (e.g., logic synthesis, physical synthesis).
- **Analysis:** collecting information on the quality of the design (e.g., timing analysis).
- **Verification:** checking whether a synthesis step has left the specification intact (e.g., layout verification).
- **Testing:** checking whether a fabricated chip has left all functions intact

Design Actions

- **Optimization:** increasing the quality of the design by rearrangements in a given description (e.g., logic optimizer, timing optimizer).
- **Design Management:** storage of design data, cooperation between tools, design flow, etc. (e.g., database).

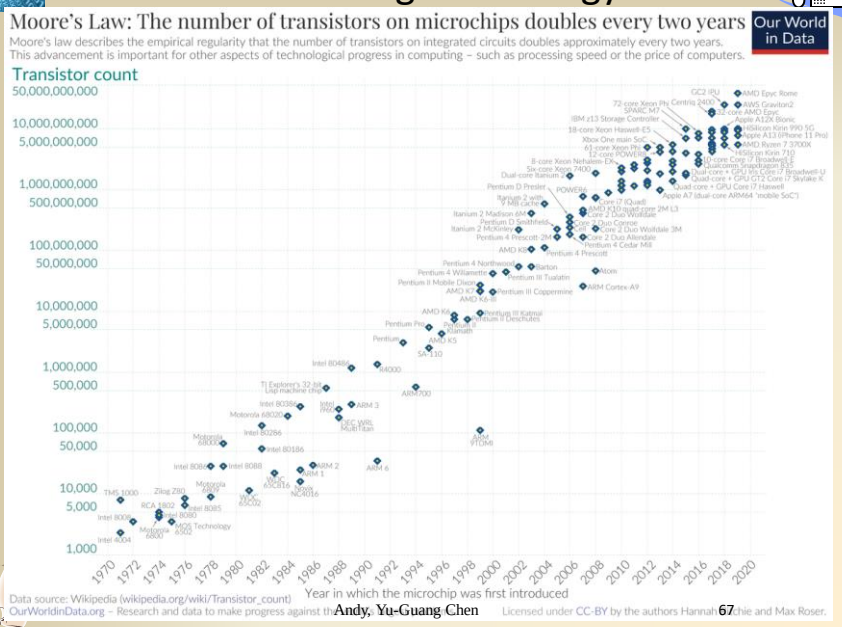
Design Technologies



Moore's Law

- Logic capacity doubles per IC at a regular interval.
- Moore: Logic capacity doubles per IC every two years (1975).
- D. House: Computer performance doubles every 18 months (1975)
- Consequences of smaller transistors:
  - Faster transistor switching
  - More transistors per chip
- True for 40+ years!
- And it will be true in at least another 10 years
  - Need smarter and more powerful CAD tools than ever

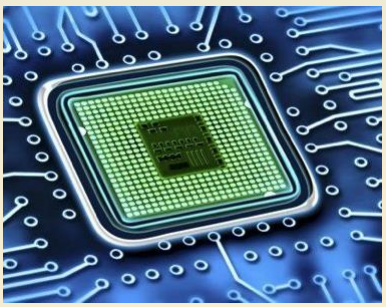
"Moore's" Law: Driving Technology Advances



When IC Design becomes more complicated……

- We need **software** for IC design!!!

```
module PISO #(parameter N_Bits = 4)
             (input clk,
              input [N_Bits-1:0] data_in,
              input load,
              input reset,
              output reg data_out);

    reg [N_Bits-1:0] shift_reg = 0;

    always @ (posedge clk)
    begin
        if(reset)
            shift_reg = 0;
        else if (load)
            shift_reg = data_in;
        else begin
            data_out = shift_reg[0];
            shift_reg = {1'b0, shift_reg[N_Bits-1:1]};
        end
    end

endmodule
```



Outline

- From Semiconductor to IC
- VLSI Design Flow
- What is EDA
- Focus on VLSI Design
  - Logic Synthesis
  - Physical Design
- Focus on VLSI Fabrication
  - Lithography
  - Design Rules

What is EDA?

- Electronic Design Automation (EDA)
  - 電子設計自動化
- VLSI Computer-Aided Design (VLSI-CAD)
  - 積體電路電腦輔助設計

What is EDA?

- A category of *software tools* for designing electronic systems such as integrated circuits and printed circuit boards.
- The tools work together in a design flow that chip designers use to design and analyze entire semiconductor chips.
- Since a modern semiconductor chip can have billions of components, *EDA tools are essential for their design*

Silicon Compiler

High-level language program (in C)

```
swap(int v[], int k)
{int temp;
   temp = v[k];
   v[k] = v[k+1];
   v[k+1] = temp;
}
```

Compiler

Binary machine language program (for MIPS)

```
00000000101000010000000000011000
00000000000011000000110000100001
10001100011000100000000000000000
10001100111100100000000000000100
10101100111100100000000000000000
10101100011000100000000000000100
00000011111000000000000000001000
```

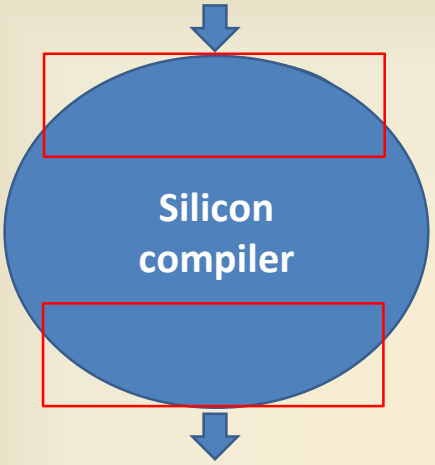
Hardware description language (Verilog/VHDL)

Silicon compiler

Layout (GDSII)

Silicon Compiler

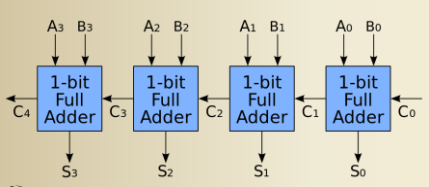
Hardware description language
(Verilog/VHDL)

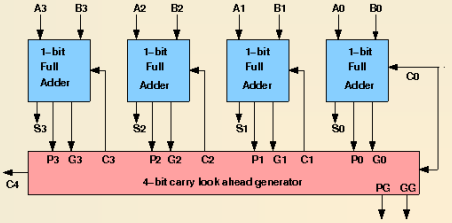


Layout
(GDSII)

Why EDA?

- The complexity of IC design increases
- We need software to help IC designers
- Most EDA problems are optimization problems
  - Maximize performance (P)
  - Minimize power (P)
  - Minimize area (A)

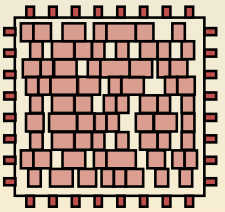




https://commons.wikimedia.org/wiki/File:4-bit_ripple_carry_adder.svg
https://en.wikipedia.org/wiki/Carry-lookahead_adder
http://616pic.com/tupian/wenhaoxiaoren.html

Example of EDA Problems (Placement)

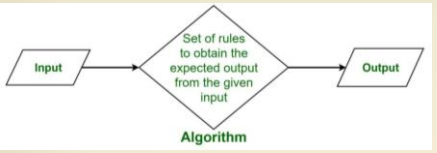
- Input:
  - Blocks (standard cells and macros) B_1, ... , B_n
  - Shapes and Pin Positions for each block B_i
  - Nets N_1, ... , N_m
- Output:
  - Coordinates (x_i , y_i) for block B_i.
- Constraint:
  - No overlaps between blocks
- Optimization goal
  - The total wire length is minimized
  - The area of the resulting block is minimized or given a fixed die
- Other consideration: timing, routability, clock, buffering and interaction with physical synthesis



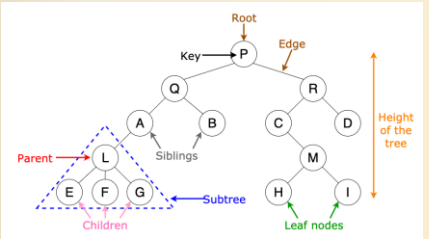
75

What is behind EDA?

- Algorithms



- Data Structures

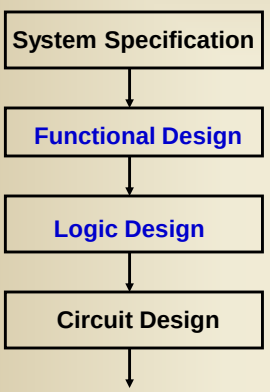


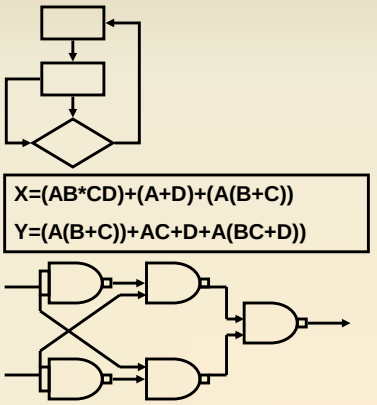
76

Outline

- From Semiconductor to IC
- VLSI Design Flow
- What is EDA
- Focus on VLSI Design
  - Logic Synthesis
  - Physical Design
- Focus on VLSI Fabrication
  - Lithography
  - Design Rules
- AI for EDA

Front End





Focus of this Course (1)

- CAD tools for synthesis and verification at logic-level of abstraction
- Theory behind: functions representation and manipulation
  - representation ⇔ data structures
  - manipulation ⇔ algorithms
- In-depth course:
  - You should be able create a small CAD-tool

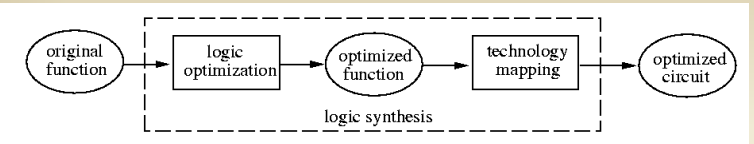
Why Logic Level?

- Logic-level synthesis is the core of today's CAD flows for IC and system design
  - course covers many algorithms that are used in a broad range of CAD tools
  - basis for other optimization techniques
  - basis for functional verification techniques

Why Logic Level?

- Most algorithms are computationally hard
  - covered algorithms and flows are good example for approaching hard algorithmic problems
  - course covers theory as well as implementation details
  - demonstrates an engineering approaches based on theoretical solid but also practical solutions
    - very few research areas can offer this combination

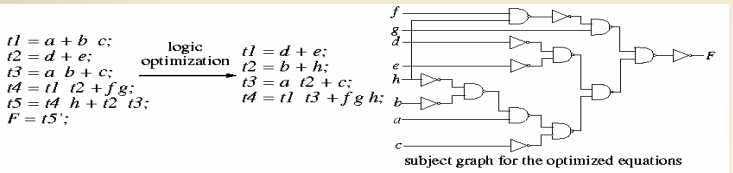
Logic Design/Synthesis



- **Logic synthesis** programs transform Boolean expressions into logic gate networks in a particular library.
- Optimization goals: minimize area, delay, power, etc
- **Technology-independent** optimization: logic optimization
  - Optimizes Boolean expression equivalent.
- **Technology-dependent** optimization: **technology mapping/library binding**
  - Maps Boolean expressions into a particular cell library.

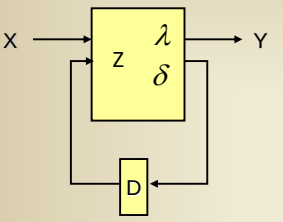
Logic Optimization Examples

- **Two-level:** minimize the # of product terms.
  - $F = \bar{x_1}\bar{x_2}\bar{x_3} + \bar{x_1}\bar{x_2}x_3 + x_1\bar{x_2}\bar{x_3} + x_1\bar{x_2}x_3 + x_1x_2\bar{x_3} \Rightarrow F = \bar{x_2} + x_1\bar{x_3}$.
- **Multi-level:** minimize the #'s of literals, variables.
  - E.g., equations are optimized using a smaller number of literals.



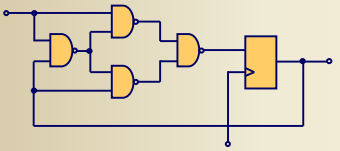
- Methods/CAD tools: Quine-McCluskey method (exponential-time exact algorithm), Espresso (heuristics for two-level logic), MIS (heuristics for multi-level logic), Synopsys Design Compiler, Cadence Genus, etc.

Logic Optimization Examples



Given: Finite-State Machine F(X,Y,Z,λ,δ) where:

X: Input alphabet
Y: Output alphabet
Z: Set of internal states
$\lambda : X \times Z \rightarrow Y$ (output function)
$\delta : X \times Z \rightarrow Z'$ (next state function)



Target: Circuit C(G, W) where
G: set of circuit components g (Boolean gates flip-flops, etc}
W: set of wires connecting G

Finite State Machine

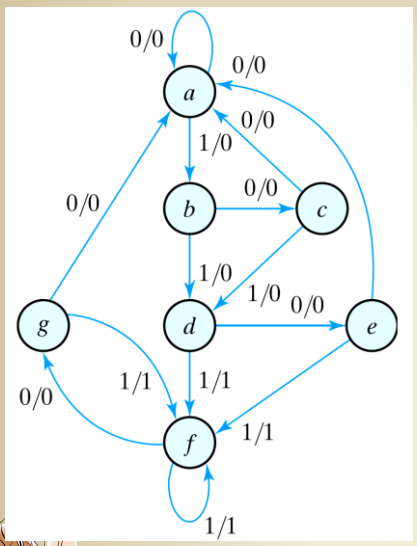


Fig. 5.25 State diagram

State:	**a**										
Input:	**0**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**0**	**0**
Output:											

Typical Synthesis Scenario

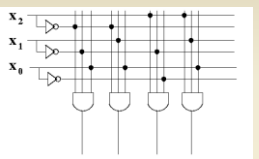
RTL to Network Transformation	- Read Verilog/VHDL - Control/data flow analysis
Technology Independent Optimizations	- Basic logic restructuring - Crude measures for goals
Technology Mapping	- Use logic gates from target cell library
Technology Dependent Optimizations	- Timing optimization - Physically driven optimizations
Test Preparation	- Improve testability - Test logic insertion

Objective Function for Synthesis

- Minimize area
  - in terms of literal count, cell count, register count, etc.
- Minimize power
  - in terms of switching activity in individual gates, blocks, etc.
- Maximize performance
  - in terms of maximal clock frequency of synchronous systems, throughput for asynchronous systems
- Any combination of the above
  - combined with different weights
  - formulated as a constraint problem
    - Ex: minimize area for a clock speed > 300MHz

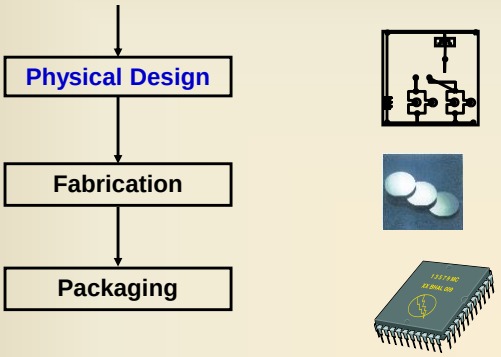
Constraints on Synthesis

- Given implementation style:
  - two-level implementation (PLA)
  - multi-level logic
  - FPGAs

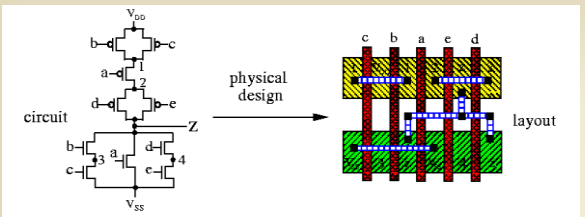


- Given performance requirements
  - minimal clock speed requirement
- Given cell library
  - set of cells in standard cell library
  - fan-out constraints (maximum number of gates connected to another gate)

Back End

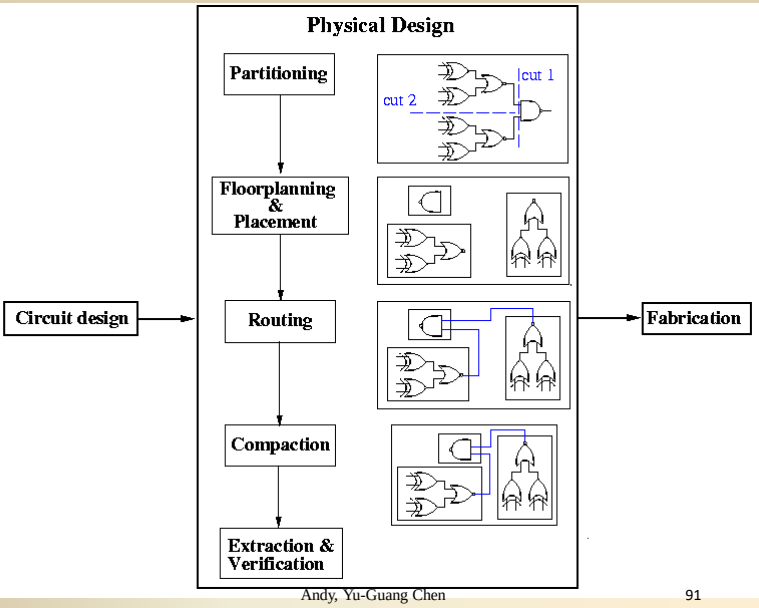


Physical Design

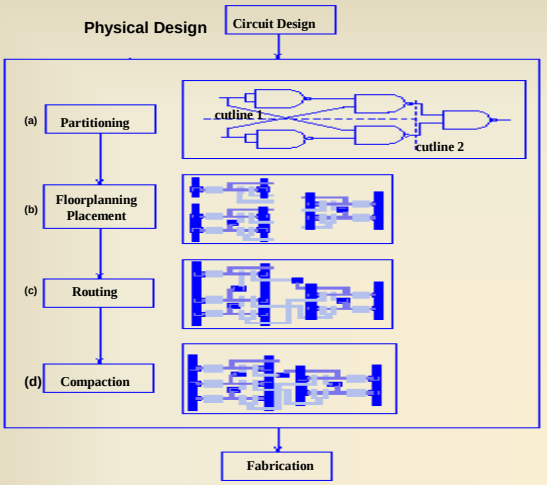


- Physical design converts a circuit description into a geometric description.
- The description is used to manufacture a chip.
- Physical design cycle:
  1. Logic partitioning
  2. Floorplanning and placement
  3. Routing
  4. Compaction
- Others: circuit extraction, timing verification and design rule checking

Physical Design Flow



Physical Design Cycle



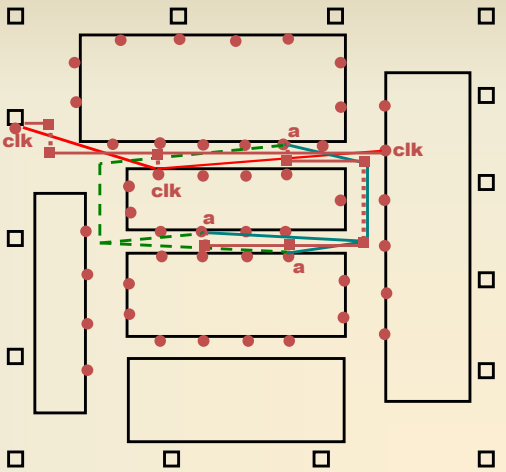
Physical Design Process

Design Steps:

- Partition & Clustering
- Floorplan & Placement
- Pin Assignment
- Global Routing
- Detailed Routing

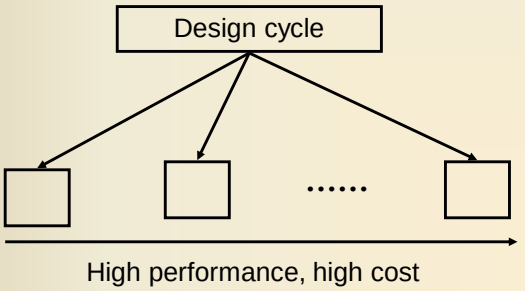
Methodology:

- Divide-and-Conquer



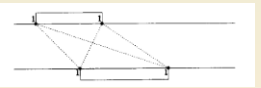
Complexities of Physical Design

- More than 100 million transistors
- Performance driven designs
- Power-constrained designs
- Time-to-Market



History of Physical Design

- Born in early 60's (board layout)
- Passed teenage in 70's (standard cell place and route)
- Entered early adulthood in 80's (over-the-cell routing)
- Declared dead in late 80's !!!
- Found alive and kicking in 90's



- Physical Design (PD) has become a dominant force in overall design cycle,
  - thanks to the deep submicron scaling
  - expand vertically with logic synthesis and interconnect optimization, analysis…. => Design closure!

Design Closure

- A part of the development workflow by which an integrated circuit design is modified from its initial description to meet a growing list of design constraints and objectives
- Looks at the overall design closure process, which takes a chip from its initial design state to the final form in which all of its design constraints are met

Focus of this Course (2)

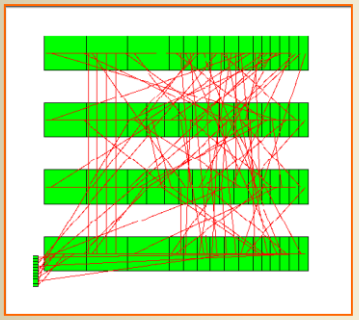
- Many existing solutions are still very suboptimal
  - Ex: placement
- Interconnect dominates
  - No physical layout, no accurate interconnect
- More new physical and manufacturing effects pop up
  - Crosstalk noise, Electromigration, …
  - Optical Proximity Correction, OPC (manufacturability), etc.
- More vertical integration needed
- Physical design is the KEY linking step between higher level planning/optimization and lower level modeling

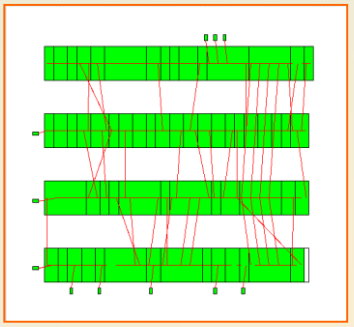
Placement Challenge

- Placement, to large extend, determines the overall interconnect
- If it sucks, no matter how well you interconnect optimization engine works, the design will suck
- Placement is a very old problem, but got renewed interest
  - Mixed-size (large macro blocks and small standard cells)
  - Optimality study shows that placement is still a bottleneck
  - Not even to mention performance driven, and coupled with buffering, interconnect optimizations, and so on (all you name)

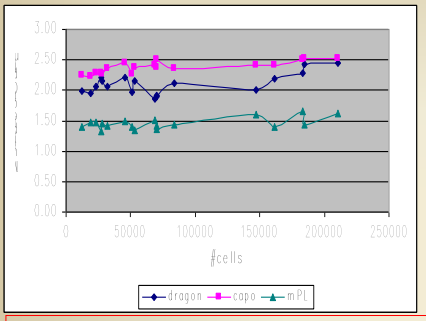
VLSI Global Placement Examples

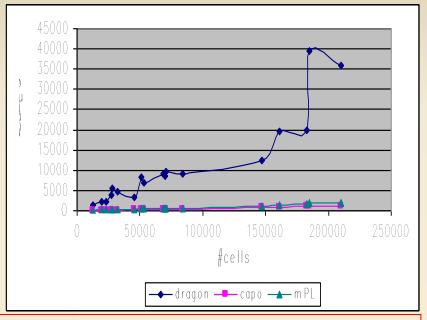
- Which one of the two placement results is better?





Comparison with Optimal



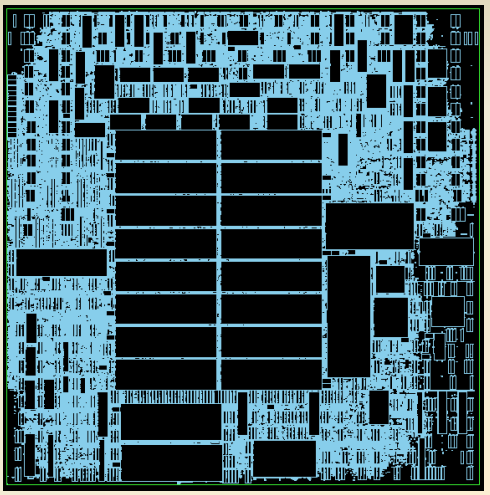


- Capo: Based on recursive min-cut (UCLA-UMich)
- Dragon: Recursive min-cut + SA refinement at each level (NWU-UCLA)
- mPL: multi-level placer (UCLA)

There is significant room for improvement in placement algorithms: existing algorithms are 50-150% away from optimal!

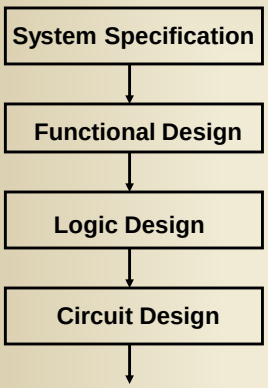
FloorPlacer (Mix-mode Placement)

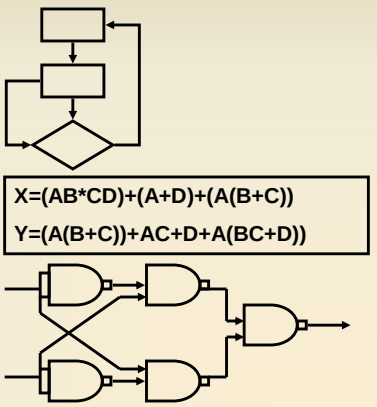
- Many macros
- data paths + dust logic
- I/O constraint
  (area I/O or wirebond)



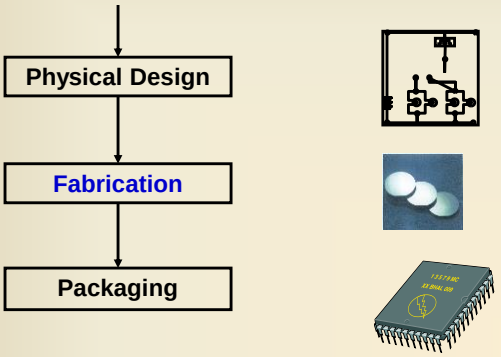
(source: IBM)

VLSI Design Cycle





VLSI Design Cycle (cont.)



Outline

- From Semiconductor to IC
- VLSI Design Flow
- What is EDA
- Focus on VLSI Design
  - Logic Synthesis
  - Physical Design
- Focus on VLSI Fabrication
  - Lithography
  - Design Rules

Lithography

- Photolithography (照相平版印刷)
  - optically project the shadow of the pattern onto the surface of the chip
- The same process to make printed circuit boards
- Start with a desired pattern defined for each layer
- Mask reticle is typically 5-10x the size of the actual chip.
  - Transparent no metal and opaque metal

Lithography

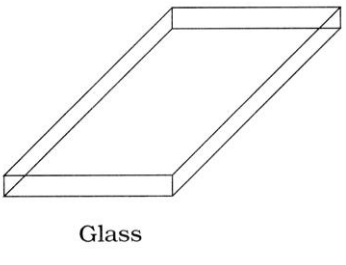


Figure 4.9 A reticle is a glass plate with a chromium pattern

Lithography

- First, coat the wafer with a light-sensitive liquid plastic material photo resist. (sensitive to light)
- After the resist is developed, hardened layers remain in the regions that were shielded from the light
  - Hardened layer: protect underlying regions from the etching process

Lithography

- Coat the surface with light-sensitive liquid material
- Spin the wafer for uniform coating

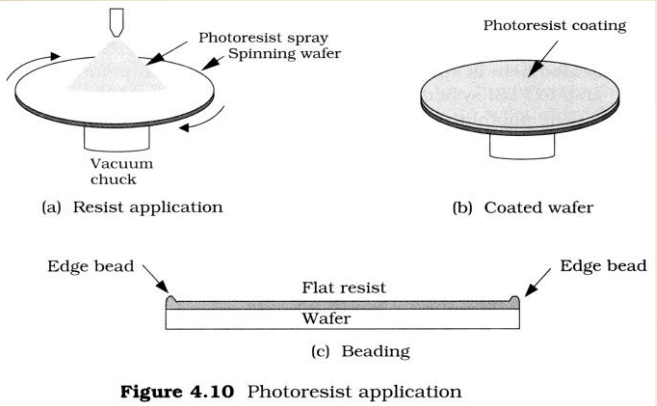


Figure 4.10 Photoresist application

Lithography

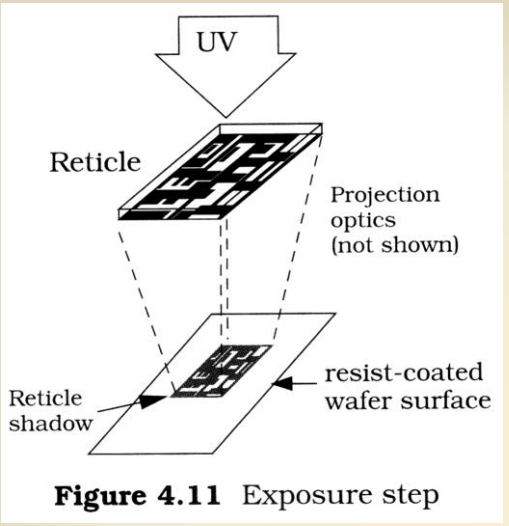


Figure 4.11 Exposure step

Lithography

- The region shield from light is harden and others are rinsed away.

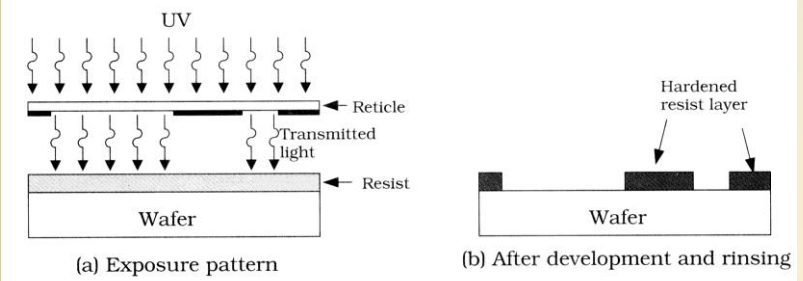


Figure 4.12 Characteristics of positive photoresist

Lithography

- Hardened resist layer is used protect underlying regions from the etching process
- The etching step removes oxide in the unprotected regions.
- This technique is used to pattern layer including poly, chemical vapor deposition (CVD) oxides, metals.

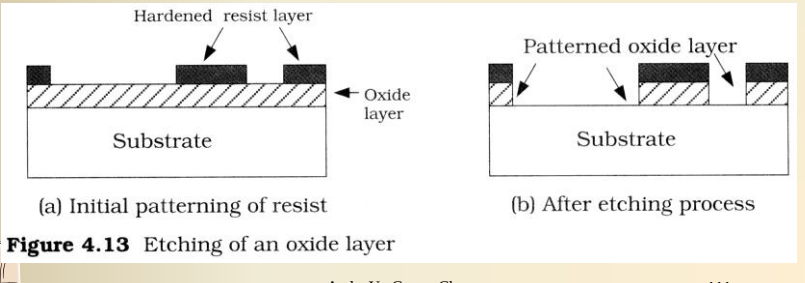


Figure 4.13 Etching of an oxide layer

Lithography

- The resist-oxide layers are used to shield from an ion implantation step.

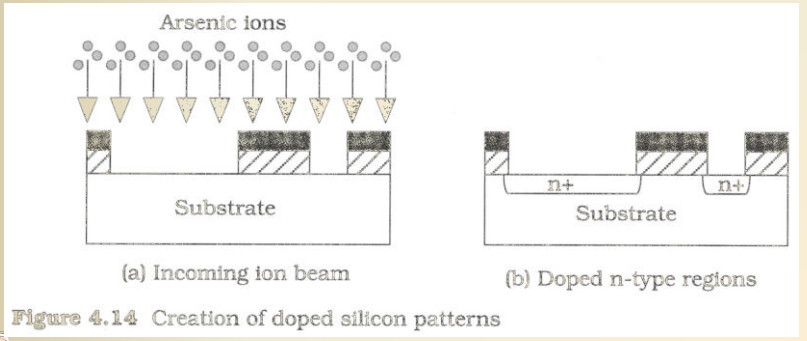


Figure 4.14 Creation of doped silicon patterns

Lithography

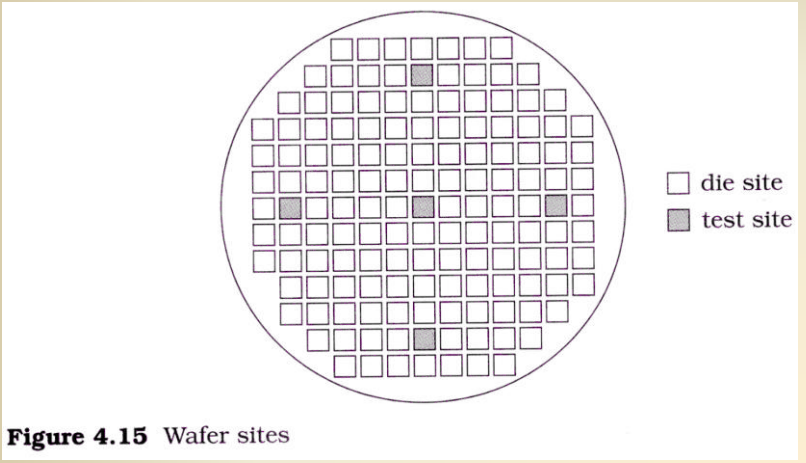
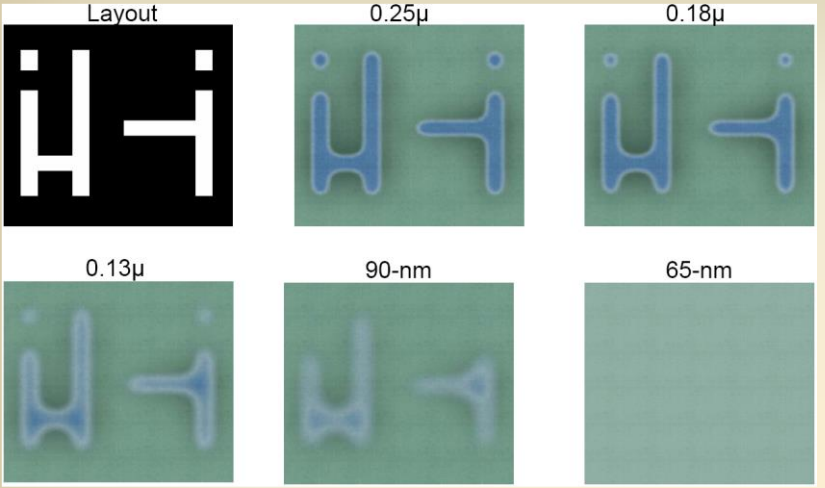
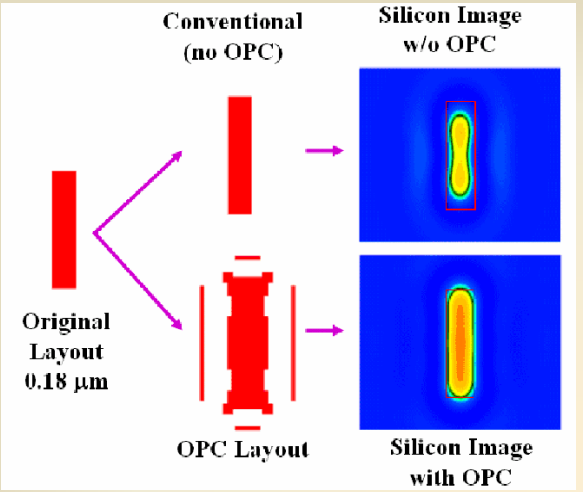


Figure 4.15 Wafer sites

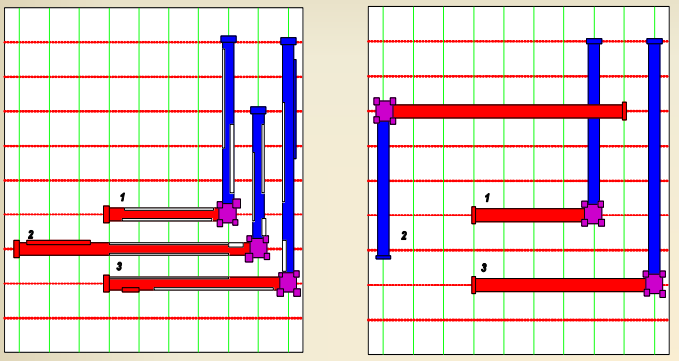
Lithography



Optical Proximity Correction (OPC)



OPC-Aware Routing



More OPC friendly

Design Issues and Tools

- System-level design
  - Partitioning into hardware and software, co-design, co-simulation, etc.
  - Cost estimation, design-space exploration
- Algorithmic-level design
  - Behavioral descriptions (e.g. in Verilog, VHDL)
  - High-level simulation
- From algorithms to hardware modules
  - High-level (or architectural) synthesis
- Logic design:
  - Schematic entry
  - Register-transfer level and logic synthesis
  - Gate-level simulation (functionality, power, etc)
  - Timing analysis
  - Formal verification

Design Issues and Tools

- Transistor-level design
  - Switch-level simulation
  - Circuit simulation
- Physical (layout) design:
  - Partitioning
  - Floorplanning and Placement
  - Routing
  - Compaction
  - Layout editing
  - Design-rule checking
  - Layout extraction
- Design management
  - Data bases, frameworks, etc.
- Silicon compilation: *from algorithm to mask patterns*
  - The *idea* is approached more and more, but still far away from a single *push-button* operation

Conclusion: We Need Algorithms

- To optimize design among different objectives, area, power, performance, and etc.
- Fundamental questions: How to do it smartly?
- Definition of algorithm in a board sense: A step-by-step procedure for solving a problem.
- Examples:
  - Cooking a dish
  - Making a phone call
  - Sorting a hand of cards
- Definition for computational problem: A well-defined computational procedure that takes some value as input and produces some value as output

Q&A

Andy, Yu-Guang Chen
Assistant Professor, Department of EE, NCU
Email: andyygchen@ee.ncu.edu.tw
FB: Yu-Guang Chen
IG: ncu.eda.andy
Google account: andyygchen.ncu