Logic and Computer Design Fundamentals

Midterm Review

Ming Cai
cm@zju.edu.cn
College of Computer Science and Technology,
Zhejiang University

Previous Year's Final Exam

1. Fill in the blank (20 points, 2pt/per)
2. Single Choice (20 points, 2pt/per)
3. Optimization (12 points, 6pt/per)
4. Circuit Analysis (18 points)
    - Combinational circuit
    - Sequential circuit
5. Logic Design (30 points)
    - Combinational circuit
    - Sequential circuit

Course Review

HIGHLIGHTS & PROBLEMS

Chapter 1

- Conversion between number systems
  - Binary number, hexadecimal number
    - Eg. (1101 1111$)_2$ = (DF$)_{16}$

Decimal	Binary	Octal	Hexadecimal
369.3125	101110001.0101	561.24	171.5
189.625	10111101.101	275.5	BD.A
214.625	11010110.101	326.5	D6.A
62407.625	1111001111000111.101	171707.5	F3C7.A

- Conversion between binary number and decimal code
  - BCD (binary-coded decimal)
    - Eg. (1001 0101$)_{BCD}$ -> (0101 1111$)_2$ -> (5F$)_{16}$
  - Parity Bit
    - 100 0001 -> 0100 0001 (with even parity) -> 1100 0001 (with odd parity)
    - How to generate odd parity bit P for any 5-bit binary number $D_4D_3D_2D_1D_0$?
- Gray Code & ASCII Character Code

Chapter 2

- Boolean algebra
  - DeMorgan’s law: $\overline{(X + Y)} = \overline{X}\ \overline{Y}$ and $\overline{(XY)} = \overline{X} + \overline{Y}$
  - Distributive laws: $X+YZ=(X+Y)(X+Z)$
  - Dual of an algebraic expression: OR ←→AND, 0’s ←→ 1’s
  - Consensus theorem: $XY+\overline{X}Z+YZ = XY+\overline{X}Z$, $(X+Y)(\overline{X}+Z)(Y+Z)=(X+Y)(\overline{X}+Z)$
- Complement of a function
  - OR ←→AND, 0’s ←→ 1’s, $X \longleftrightarrow \overline{X}$
- Standard forms
  - Minterms, Maxterms, Canonical forms (SOM, POM)
  - Product terms, sum terms, SOP, POS
  - Relationship between SOM and POM? SOM and SOP?
- Two-level circuit optimization
  - Cost criteria: gate input cost
  - Karnaugh map (K-map), Prime Implicants, Essential Prime Implicants
  - Simplifying in SOP form (with don’t care conditions)

Chapter 2

- Other gates
  - NAND/NOR, XOR/XNOR, Odd/Even Function, Buffer, 3-state buffer
- Exclusive-OR operator and gates
  - Identities of XOR operation:
    - $X \oplus \overline{Y} = \overline{X} \oplus Y = \overline{(X \oplus Y)}$ $X \oplus \overline{(Y \oplus Z)} = \overline{(X \oplus Y)} \oplus Z = \overline{(X \oplus Y \oplus Z)}$
  - Odd function and even function
    - Use odd function to generate even parity bit
    - Use even function to generate odd parity bit
    - The even function is obtained by replacing the output gate with an XNOR gate. $P_{odd} = X \oplus Y \oplus Z$, $P_{even} = \overline{P_{odd}} = \overline{(X \oplus Y \oplus Z)}$
- High-impedance outputs
  - 3-state buffer
  - Transmission gates

EN	IN	OUT
0	X	Hi-Z
1	0	0
1	1	1

Chapter 2

- ## Problems:

1. The dual of an algebraic expression is obtained by 1) interchanging OR and AND operations and 2) replacing 1’s by 0’s and 0’s by 1’s.
2. Use DeMorgan's Theorem to complement a function: 1) interchange AND and OR operators and 2) complement each constant value and literal
3. Four variables odd function has _____C_______ “1” squares in its corresponding K-Map.

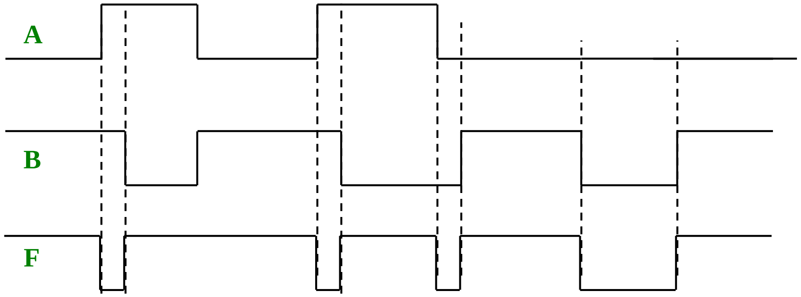
   A. 4 B. 7 C. 8 D. 14
4. The gate input cost G of function $F = AB(C+D) + C(B\overline{D} + \overline{A}D)$ is __A___.

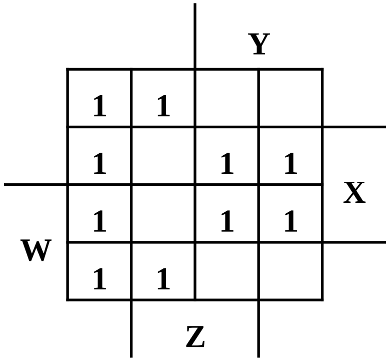
   A. 15 B. 14 C. 13 D. 12
5. Which of the following logical gates can be used as a controllable inverter? _____B______.

   A. AND gate B. XOR gate C. Buffer gate D. OR gate
6. The Essential Prime Implicants in the K-Map given below are ___B___.

   A. Y’Z’, XZ’ B. X’Y’, XY C. XY, XZ’ D. Y’Z’, X’Y’
7. Given below are the waveforms of input A, B and output F of a logic device. Then the device is a ___C____ gate.

   A. NAND B. NOR C. XOR D. OR



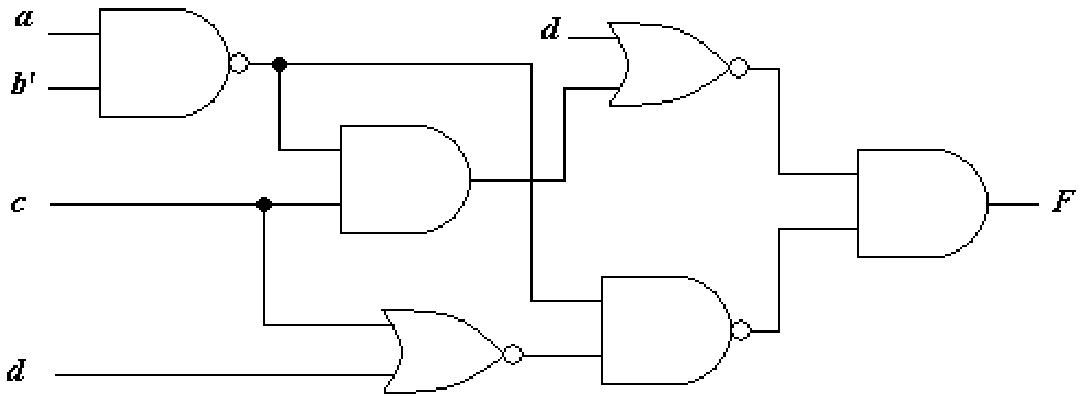


			Y	Y	
	1	1			
	1		1	1	X
W	1		1	1	X
W	1	1			
		Z	Z		

Chapter 2

- Problems:

1. According to the following logic circuit diagram, write down the corresponding Boolean function and optimize it to the form of SOP



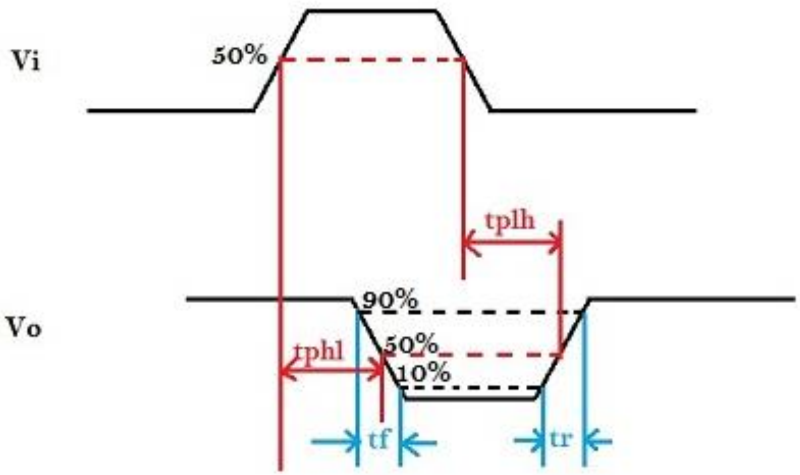
2. Given a Boolean function

$F(W, X, Y, Z) = \sum m(4, 6, 7, 8, 12, 15) + \sum d\ (2, 3, 5, 10, 11)$

Optimize F together with the don't-care conditions d using a K-map

Chapter 3

- Technology parameters
  - fan-in, fan-out, noise margin, cost, transition time, propagation delay, power dissipation

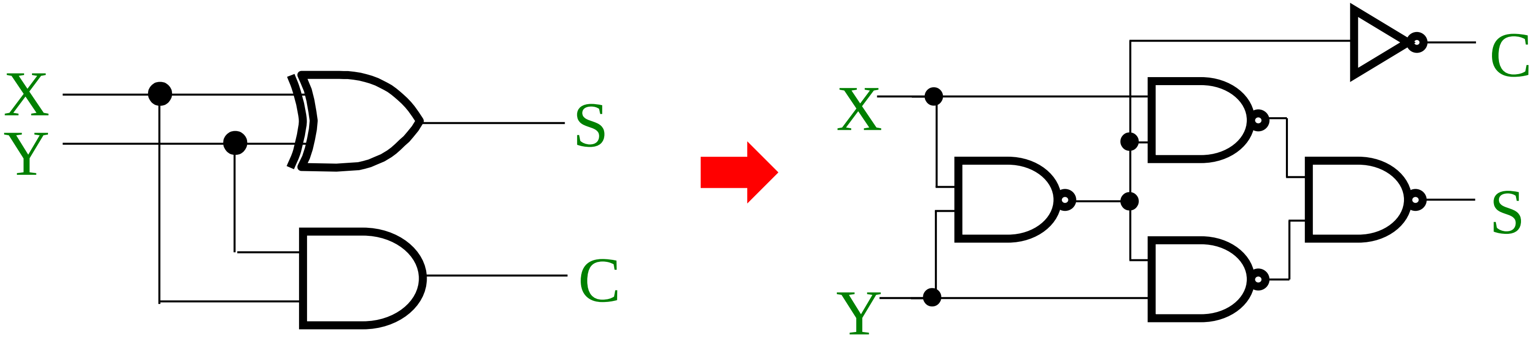


tr = Rise transition time
tf = Fall transition time
tphl = Propagation delay high-low
tplh = Propagation delay low-high

  - Delay Model: transport delay, inertial delay, rejection time
  - How to calculate gate delay based on fan-out?

Chapter 3

- Methods of Describing Logic Events
  - Truth Table, Timing Diagram, Boolean Function, Karnaugh Maps, Logic Circuit
- Design procedure: specification, formulation, optimization, technology mapping, verification
  - Hierarchical Design
- Seven-segment display
  - How to design a BCD-to-Seven-Segment decoder? example 3-2
- Technology mapping
  - How to implement a Boolean function with NAND gates?

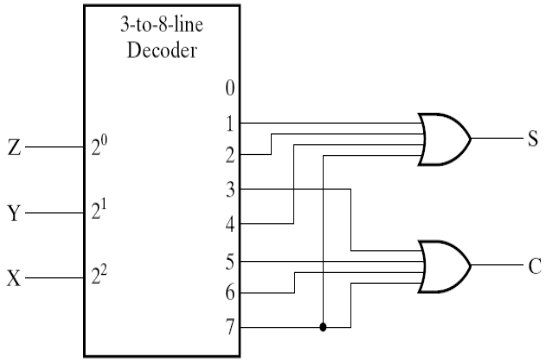


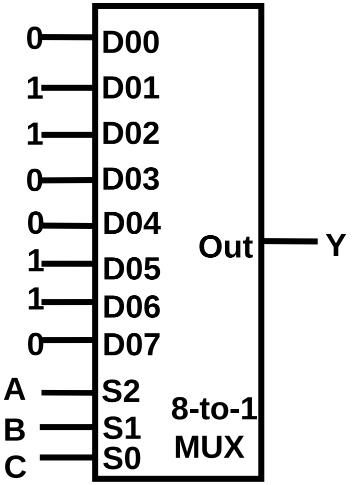
Chapter 3

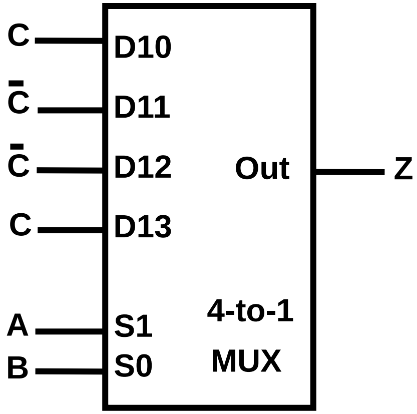
- n-to-m-Line Decoder
  - n inputs and m outputs with $n \leq m \leq 2^n$
- m-to-n-Line Encoder
  - m inputs and n outputs with $n \leq m \leq 2^n$
- Multiplexer
  - n control inputs (selection inputs), m inputs and one output with $m < 2^n$
- Demultiplexer: Decoder with Enable
- Combinational Function Implementation
  - Decoders and OR gates,
  - Multiplexers
  - ROMs
  - PALs: doesn't provide full decoding of the variables, so it doesn't generate all the minterms
  - PLAs: similar to the PALs
  - Lookup Tables

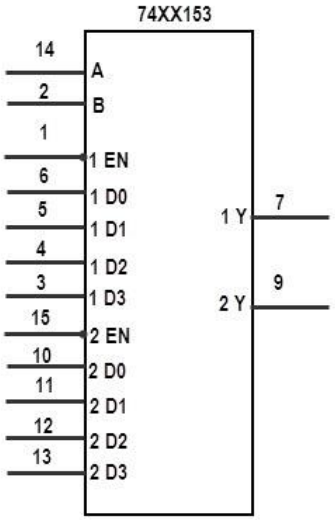
Chapter 3

- Combinational Function Implementation
  - Any combinational circuit with *n* inputs and *m* outputs can be implemented with
    - an *n*-to-2^n-line decoder, and *m* OR gates (one for each output)
  - Implement *m* functions of *n* variables with
    - an *m*-wide 2^n-to-1-line multiplexer
  - Implement *m* functions of *n+1* variables with
    - an *m*-wide 2^n-to-1-line multiplexer and a single inverter



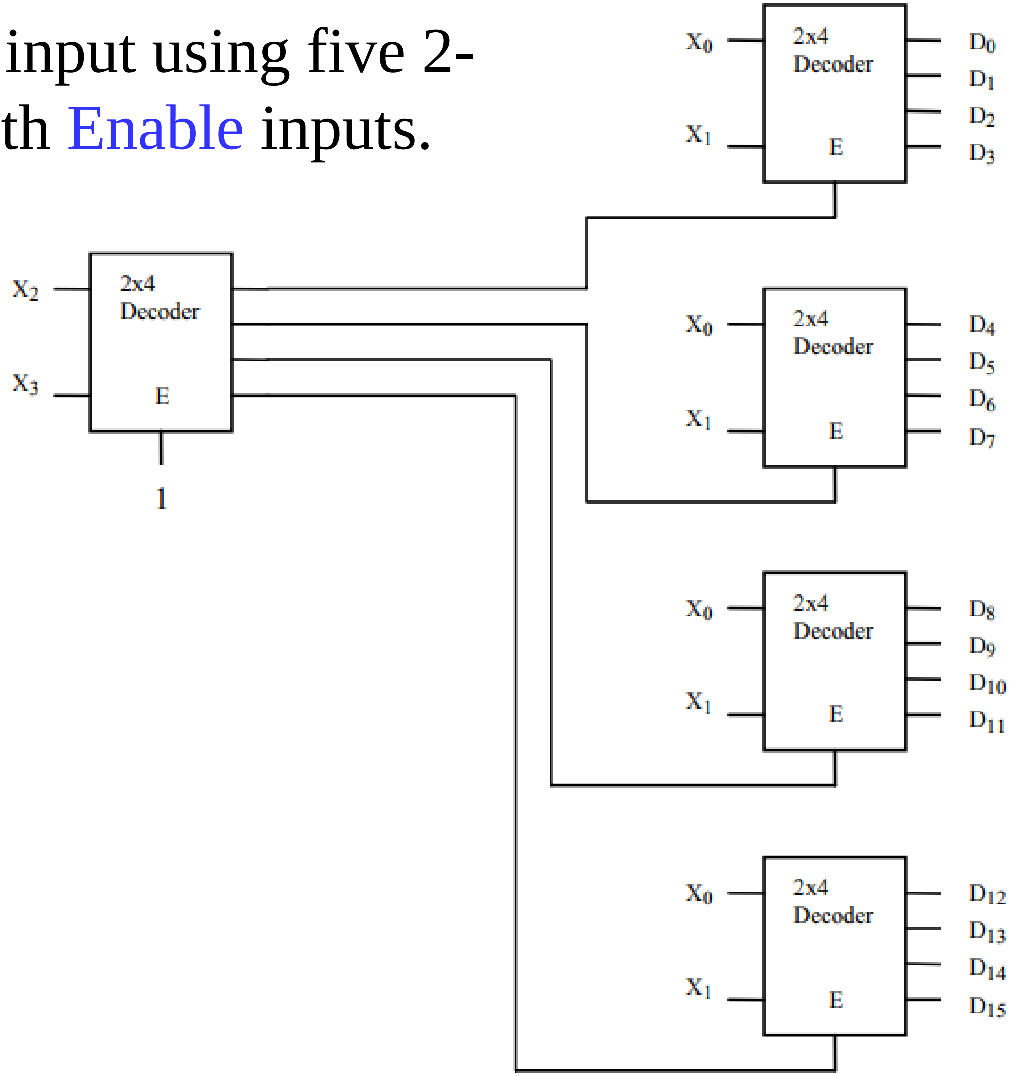






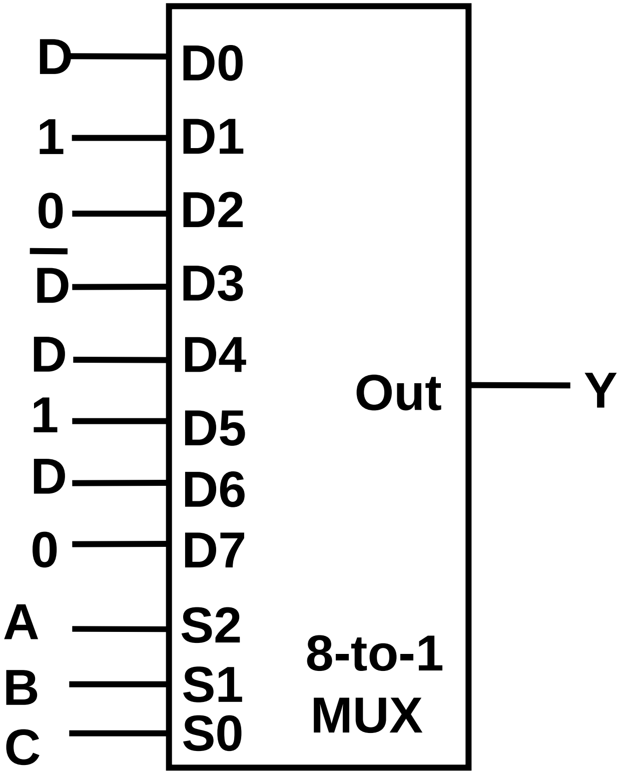
Chapter 3

- Problem: Design a 4-to-16 line decoder with Enable input using five 2-to-4 line decoders with Enable inputs.



Chapter 3

- Problem: Give the canonical sum of product expression for the function which is implemented using the following circuit.

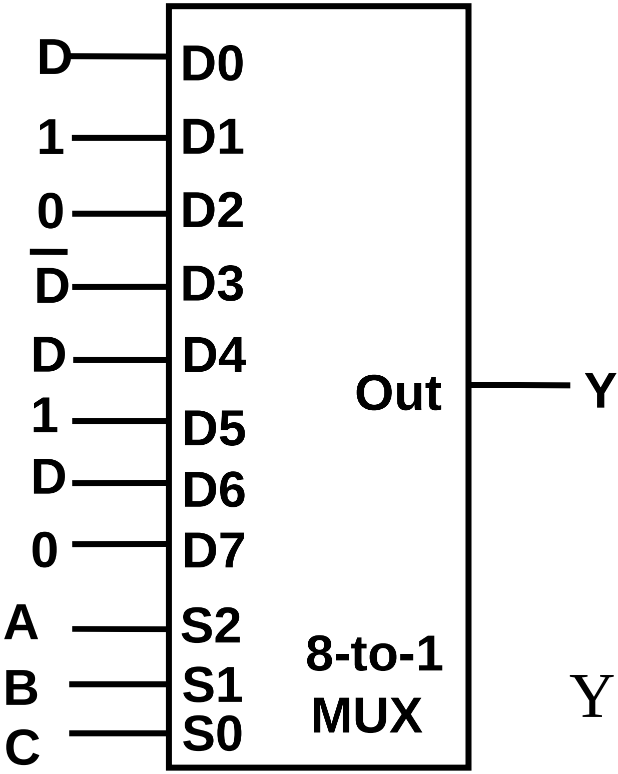


A	B	C	D	Y	
0	0	0	0	0	D
0	0	0	1	1	
0	0	1	0	1	1
0	0	1	1	1	
0	1	0	0	0	0
0	1	0	1	0	
0	1	1	0	1	$\overline{D}$
0	1	1	1	0	
1	0	0	0	0	D
1	0	0	1	1	
1	0	1	0	1	1
1	0	1	1	1	
1	1	0	0	0	D
1	1	0	1	1	
1	1	1	0	0	0
1	1	1	1	0	

$$Y(A, B, C, D) = \Sigma_m(1, 2, 3, 6, 9, 10, 11, 13)$$

Chapter 3

- Problem: Give the canonical sum of product expression for the function which is implemented using the following circuit.



- approach 2

$$Y=\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}\,\overline{B}C1+\overline{A}B\overline{C}0+\overline{A}BC\overline{D}+$$

$$A\overline{B}\,\overline{C}D+A\overline{B}C1+AB\overline{C}D+ABC0$$

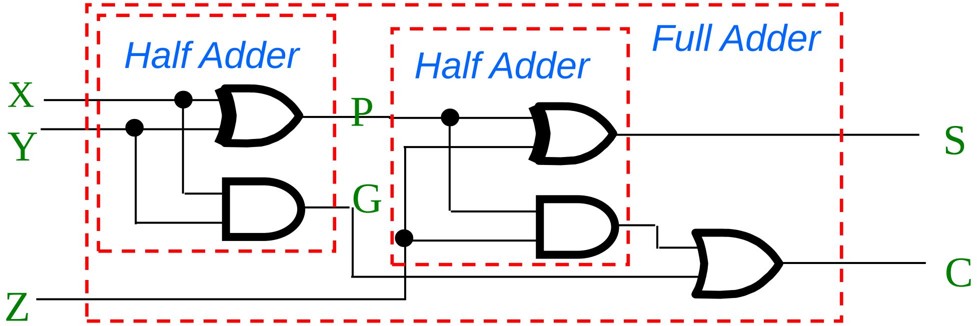
$$=\overline{A}\,\overline{B}\,\overline{C}D+\overline{A}\,\overline{B}C\overline{D}+\overline{A}\,\overline{B}CD+\overline{A}BC\overline{D}+$$

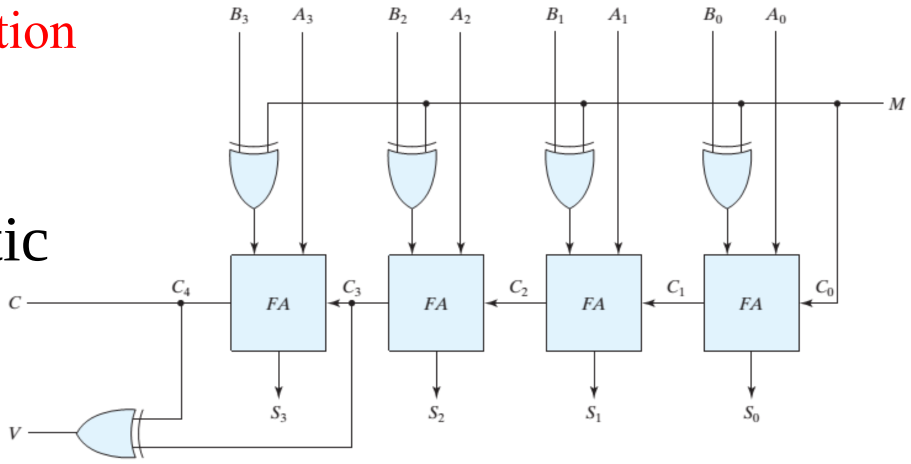
$$A\overline{B}\,\overline{C}D+A\overline{B}C\overline{D}+A\overline{B}CD+AB\overline{C}D$$

$$Y(A, B, C, D) = \Sigma_m(1, 2, 3, 6, 9, 10, 11, 13)$$

Chapter 3

- Half Adder
- Full Adder
  - *carry generate:* X ·Y
  - *carry propagate:* X⊕Y
- Binary Ripple Carry Adder
- Carry Lookahead Adder
- Binary subtraction
  - Unsigned 2’s Complement subtraction
  - Signed 2’s Complement subtraction
    - 1100 - 0011 = 1001
    - 0011 - 1100 = 0111
- Signed-Complement Arithmetic
- Binary adder-subtractors

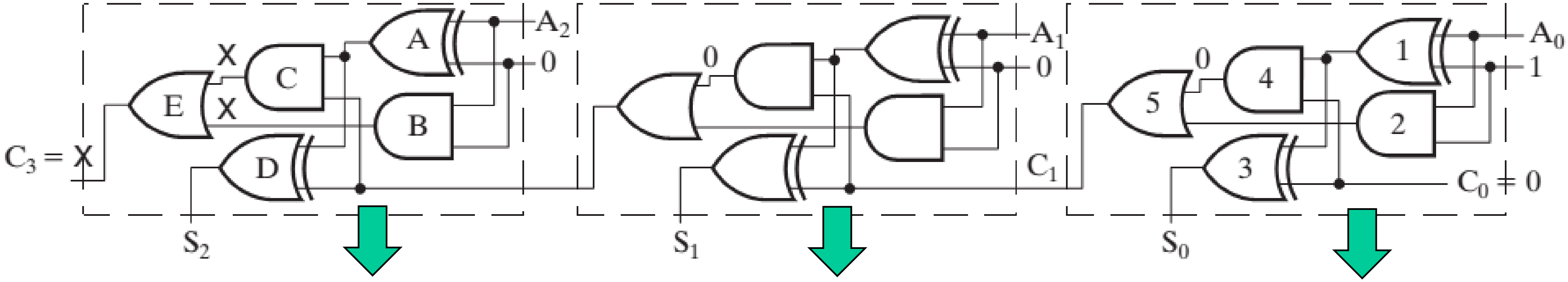




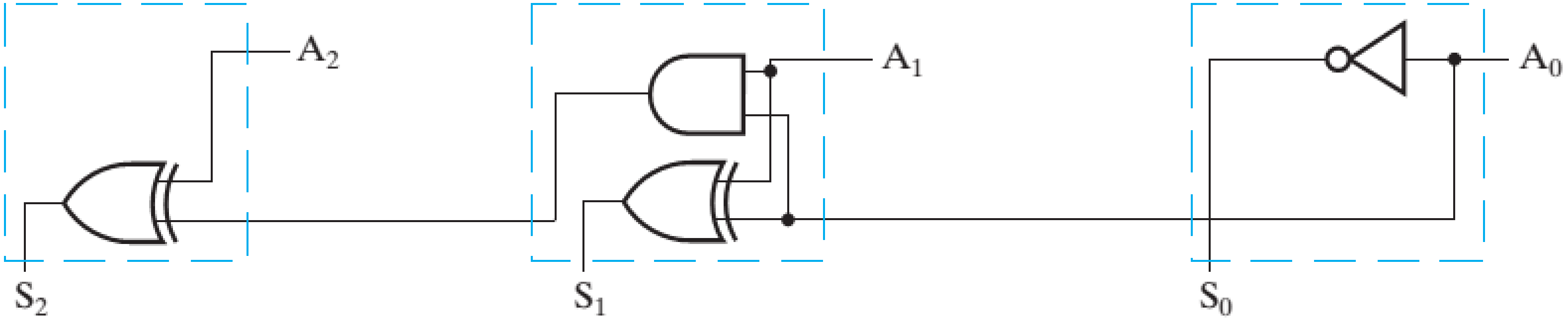
Chapter 3

- Design by Contraction

1. set B = 001 → value fixing

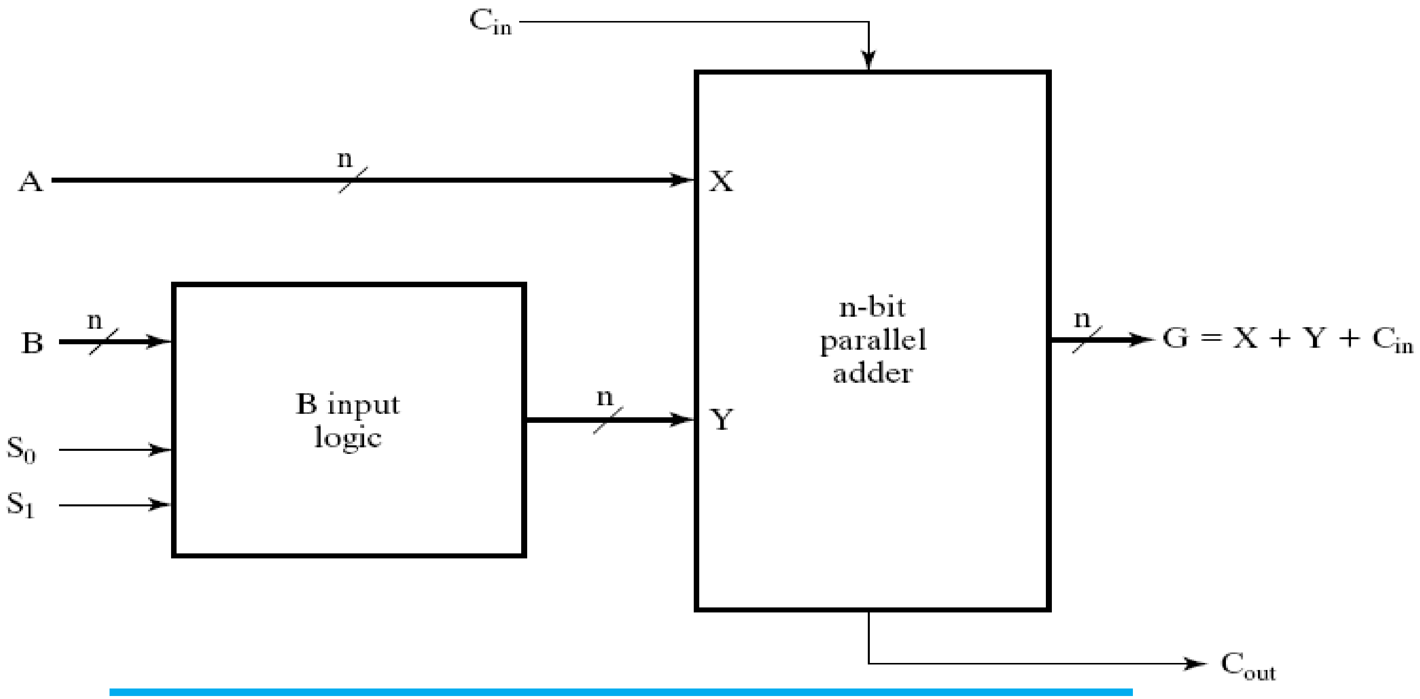


2. simplifying the logic → contracting



Chapter 3

- Arithmetic circuit design

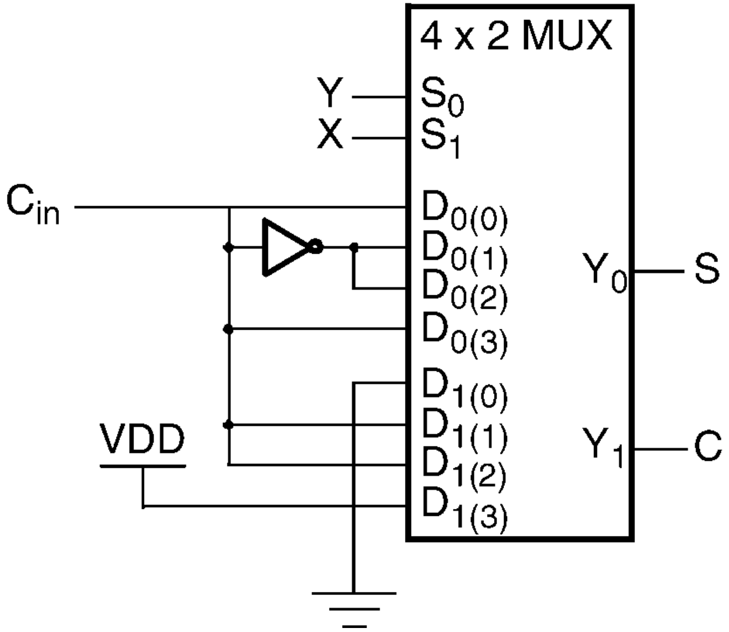


Select		Input	$G = A + Y + C_{in}$	
S_1	S_0	Y	C_{in} 0	C_{in} 1
0	0	all 0's	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1's	$G = A - 1$ (decrement)	$G = A$ (transfer)

Chapter 3

- **Problem:** Implement a binary full adder with a dual 4-to-1-line multiplexer and a single inverter.

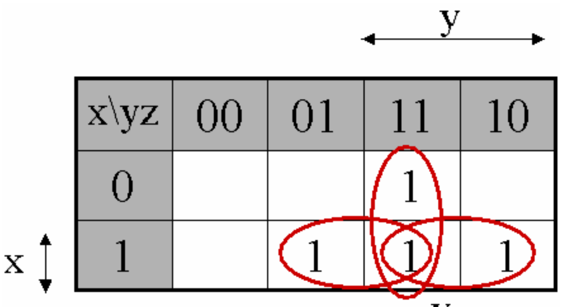
X	Y	C_{in}	S	C	
0	0	0	0	0	$S = C_{in}$
0	0	1	1	0	$C = 0$
0	1	0	1	0	$S = \overline{C}_{in}$
0	1	1	0	1	$C = C_{in}$
1	0	0	1	0	$S = \overline{C}_{in}$
1	0	1	0	1	$C = C_{in}$
1	1	0	0	1	$S = C_{in}$
1	1	1	1	1	$C = 1$



Chapter 3

- **Problem:** Using A Full Adder and logic gates to design a combinational circuit with three inputs, x, y, and z, and three outputs, A, B, and C.
- 1)When the binary input is 0, 1, 2, or 3, the binary output is one greater than the input.
- 2)When the binary input is 4, 5, 6, or 7, the binary output is one less than the input.

x	y	z	A	B	C
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	1	1	0



x\yz	00	01	11	10
0			1	
1		1	1	1

$$A = xy + xz + yz$$

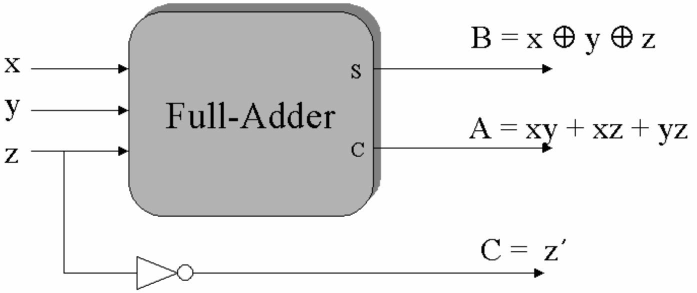
y

x

x\yz	00	01	11	10
0		1		1
1	1		1	

$$B = x \oplus y \oplus z$$

$$C = z'$$



Chapter 3

Problems

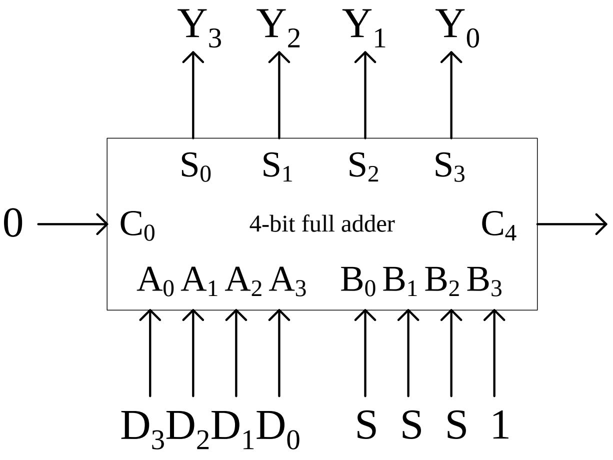
- Given a 4-bit full adder, design a 4-bit Incrementer-Decrementer circuit.

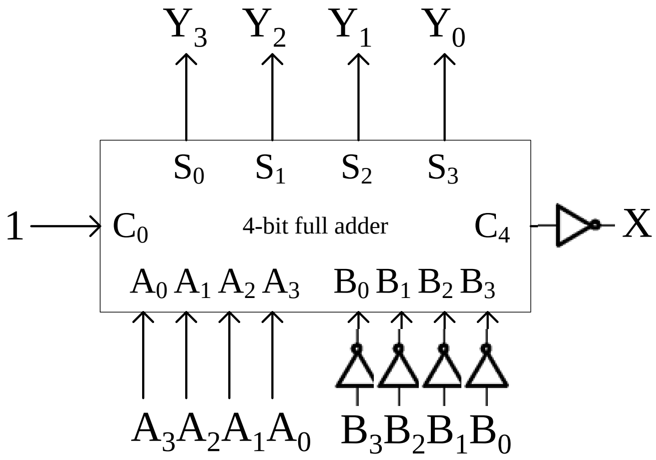
  Inputs: $D = D_3D_2D_1D_0$ and S, Outputs: $Y = Y_3Y_2Y_1Y_0$ and C.

  (1). When S = 0, circuit is an Incrementer: Y = D +1. If Y > 1111, C = 1; other C = 0.

  (2). When S = 1, circuit is a Decrementer: Y = D -1. If Y < 0000, C = 0; other C = 1.

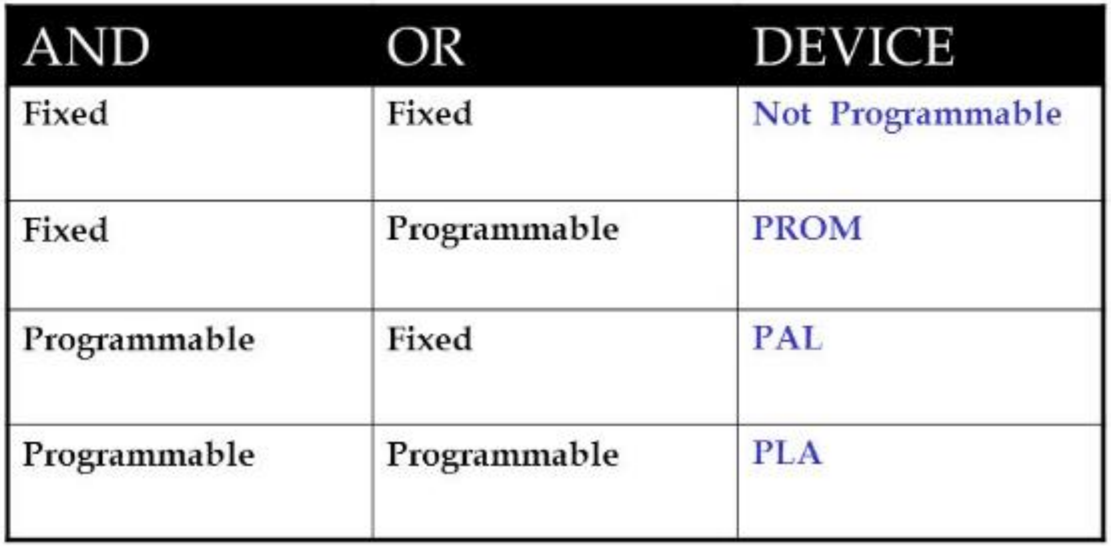
- Using a Full Adder and logic gates to design a combinational circuit that compares two 4-bit unsigned numbers A and B to see whether B is greater than A. The circuit has one output X, so that X = 1 if A < B and X = 0 if A >=B.





Chapter 5

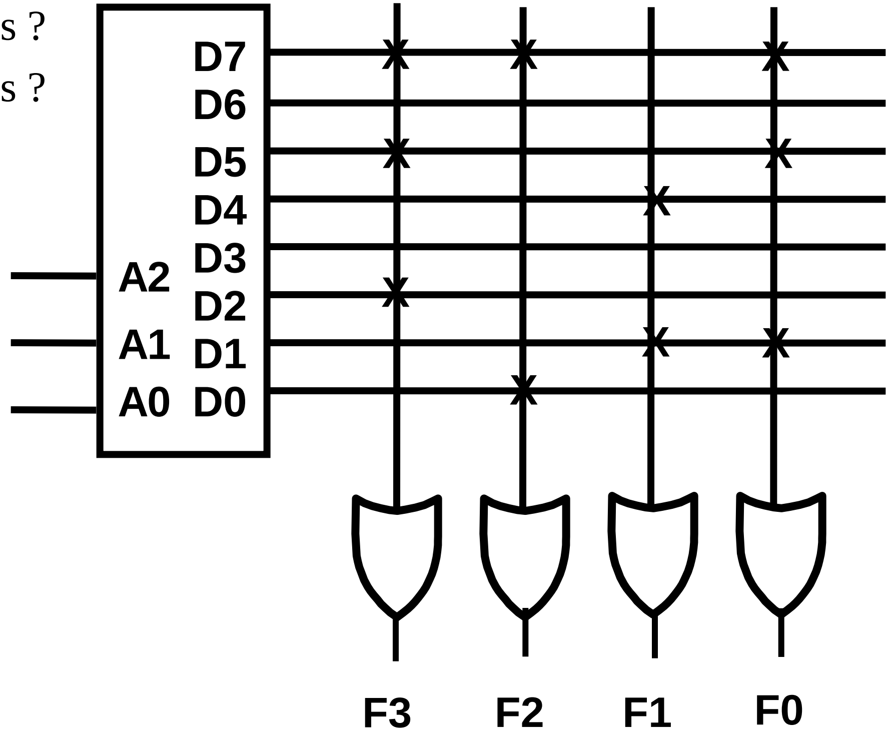
- Programmable implementation technologies
  - PROM, PAL, PLA, FPGA



AND	OR	DEVICE
Fixed	Fixed	Not Programmable
Fixed	Programmable	PROM
Programmable	Fixed	PAL
Programmable	Programmable	PLA

Chapter 5

- Programmable implementation technologies
  - PROM
  - Can any combinational circuit with *n* inputs and *m* outputs be implemented with
    - a PROM with *n* inputs and *m* outputs ?
    - a PLA with *n* inputs and *m* outputs ?
    - a PAL with *n* inputs and *m* outputs ?



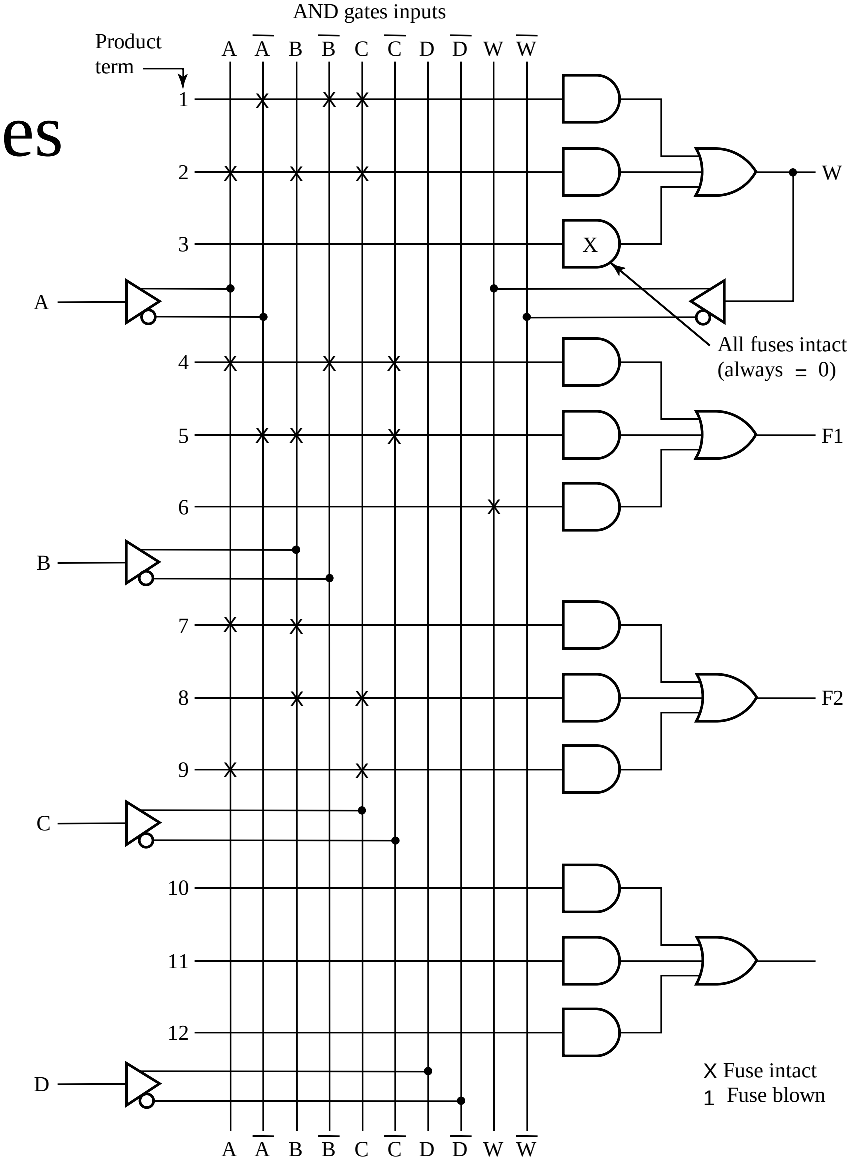
Chapter 5

- Programmable implementation technologies
  - PAL

$$\mathbf{W} = \overline{\mathbf{A}}\,\overline{\mathbf{B}}\mathbf{C} + \mathbf{ABC}$$

$$\mathbf{F1} = \mathbf{X} =$$
$$\mathbf{A}\overline{\mathbf{B}}\,\overline{\mathbf{C}} + \overline{\mathbf{A}}\mathbf{B}\overline{\mathbf{C}} + \mathbf{W}$$

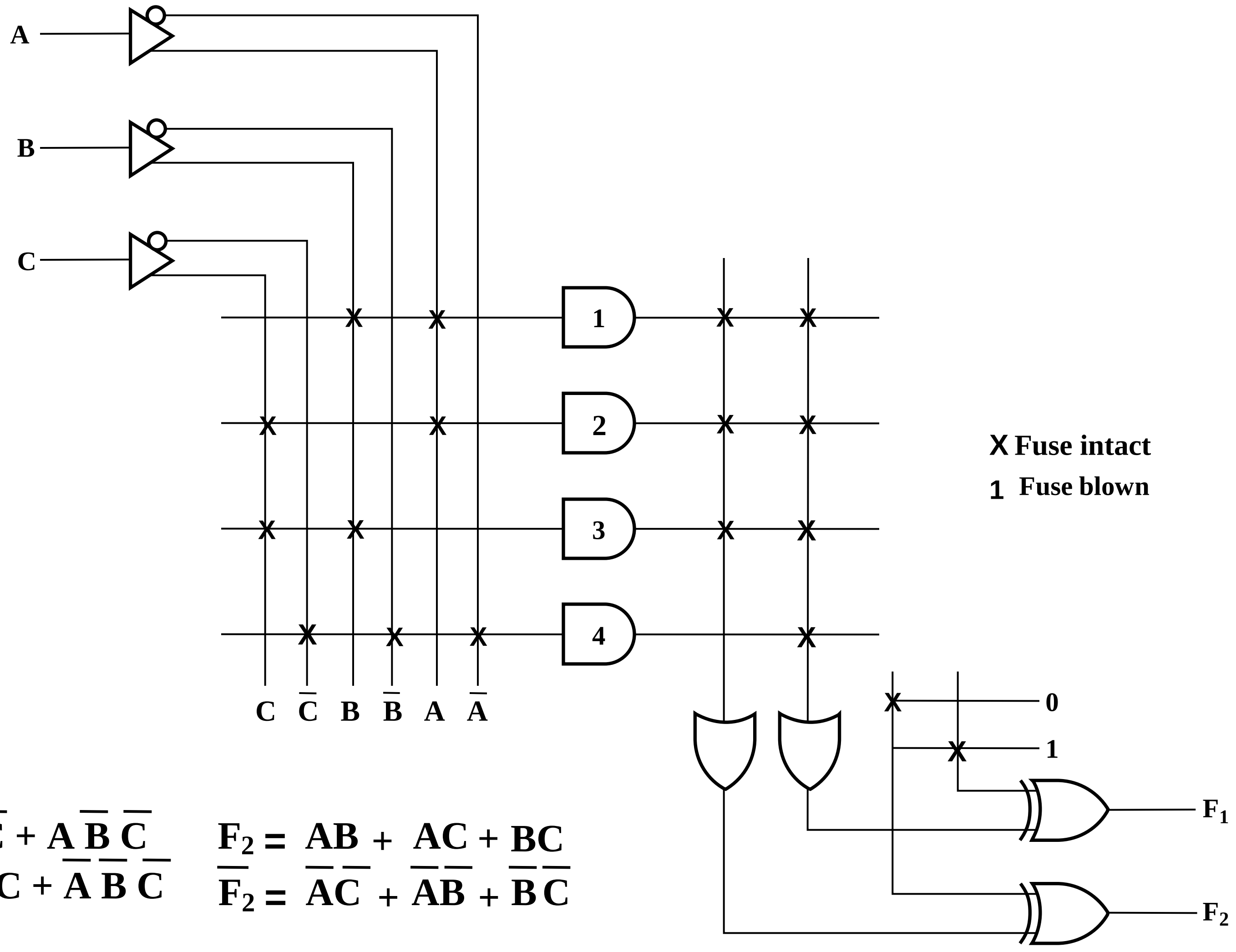
$$\mathbf{F2} = \mathbf{Y} =$$
$$\mathbf{AB} + \mathbf{BC} + \mathbf{AC}$$



AND gates inputs
Product term
A $\overline{A}$ B $\overline{B}$ C $\overline{C}$ D $\overline{D}$ W $\overline{W}$
1
2
3
X
W
A
All fuses intact (always = 0)
4
5
F1
6
B
7
8
F2
9
C
10
11
12
D
X Fuse intact
1 Fuse blown
A $\overline{A}$ B $\overline{B}$ C $\overline{C}$ D $\overline{D}$ W $\overline{W}$

Chapter 5

- Programmable implementation technologies
  - PLA



$$F_1 = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

$$\bar{F}_1 = AB + AC + BC + \bar{A}\bar{B}\bar{C}$$

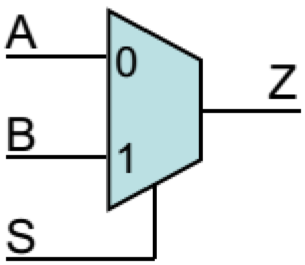
$$F_2 = AB + AC + BC$$

$$\bar{F}_2 = \bar{A}\bar{C} + \bar{A}\bar{B} + \bar{B}\bar{C}$$

Chapter 5

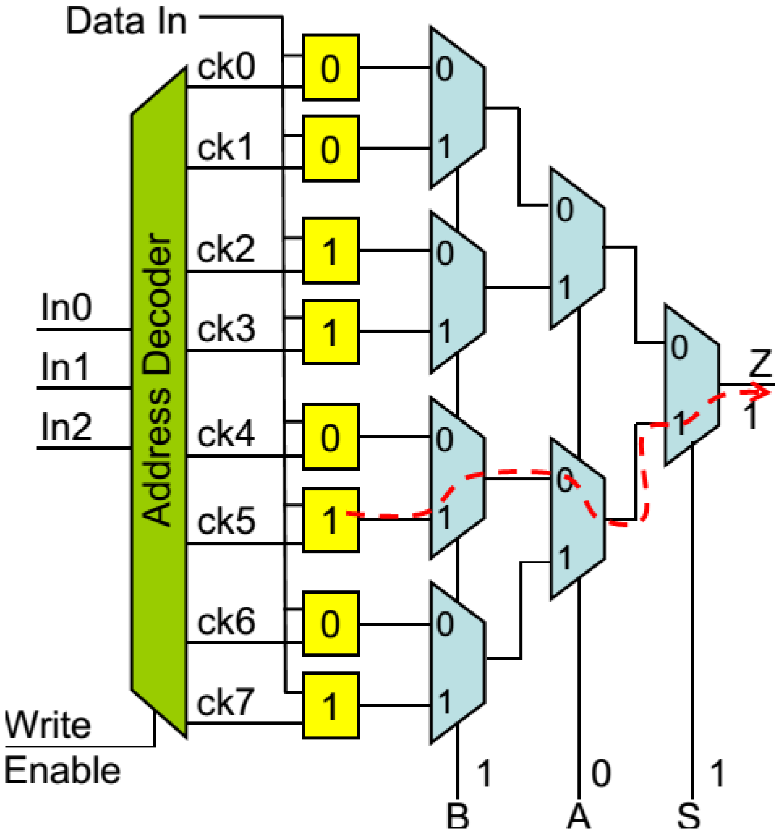
- Programmable implementation technologies
  - FPGA
  - Lookup tables (LUTs) are used for implementing FPGAs

Multiplexer



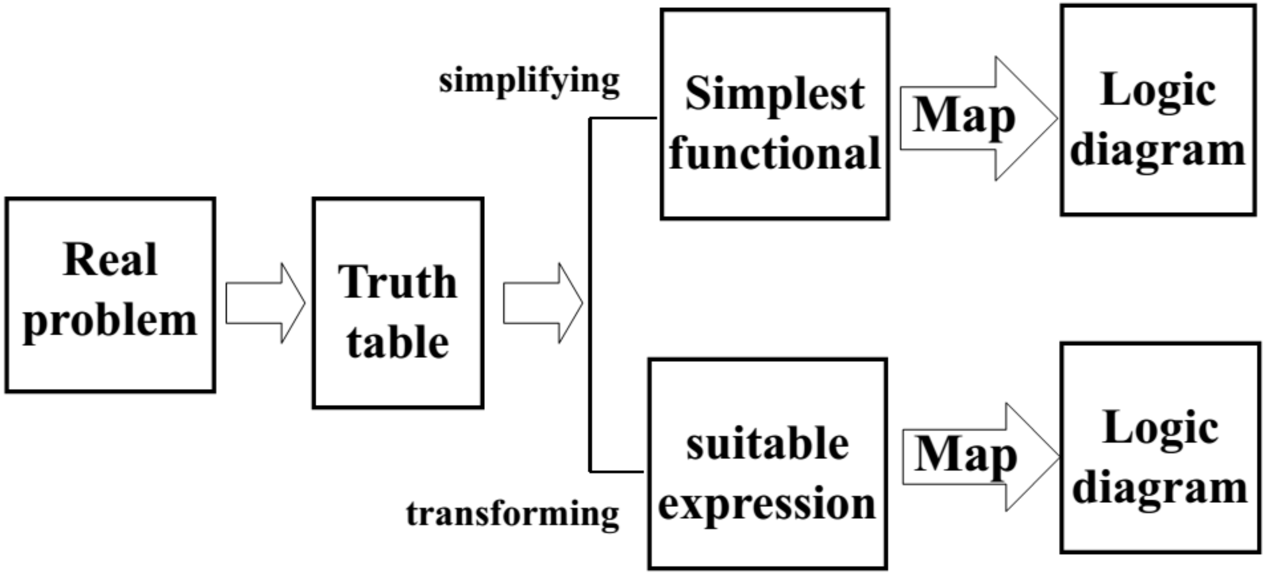
Truth table

S A B	Z
0 0 0	0
0 0 1	0
0 1 0	1
0 1 1	1
1 0 0	0
1 0 1	1
1 1 0	0
1 1 1	1



Design of Combinational Circuits

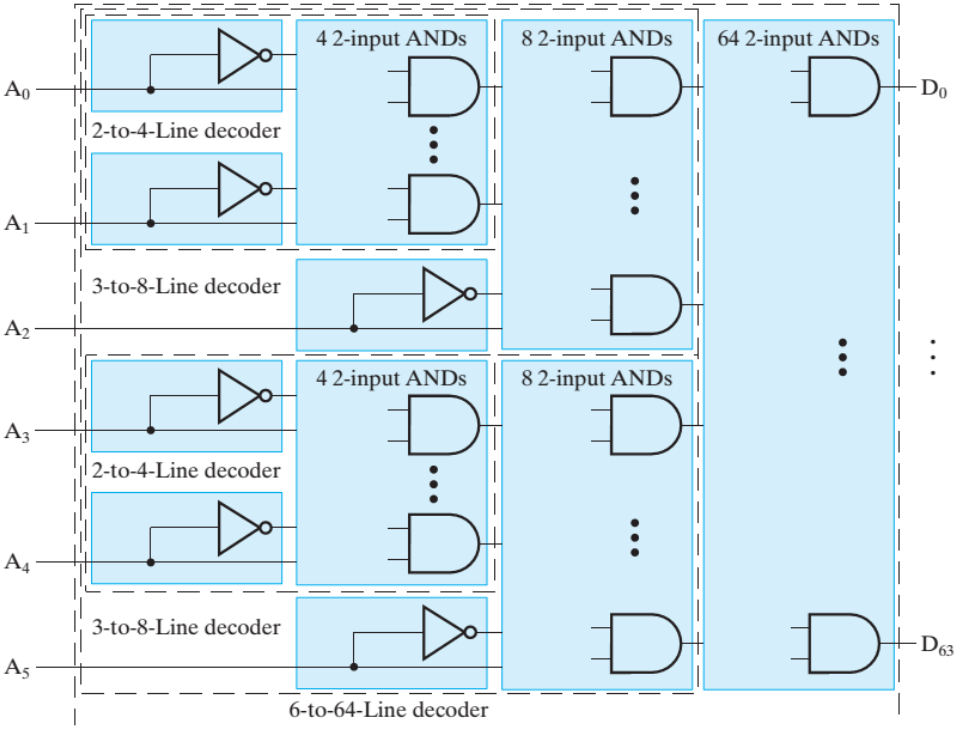
- Methods of designing combinational circuits
  - Design by truth table
  - Design by bisection
  - Design in a hierarchical structure
  - Design by iteration
  - Design by contraction/expansion



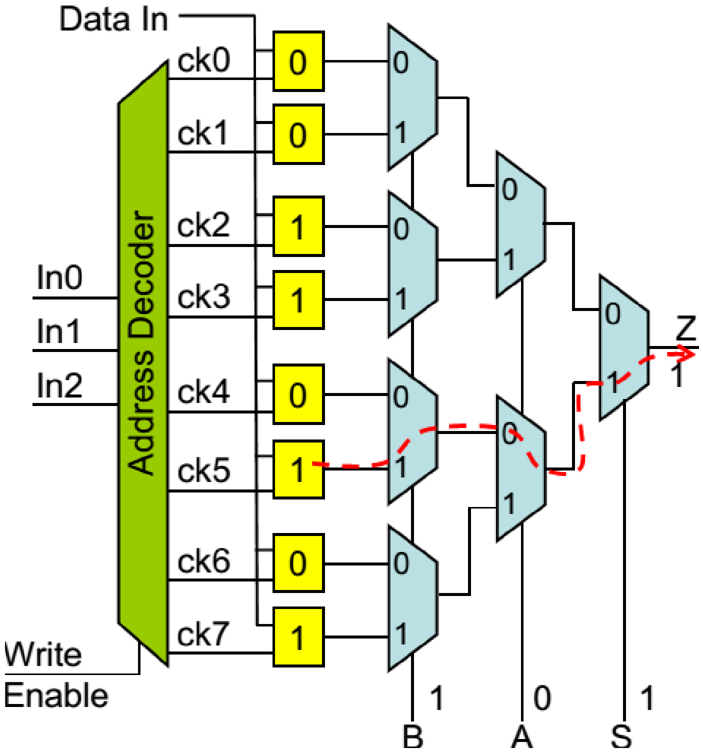
Design by truth table

Design of Combinational Circuits

- Design by bisection



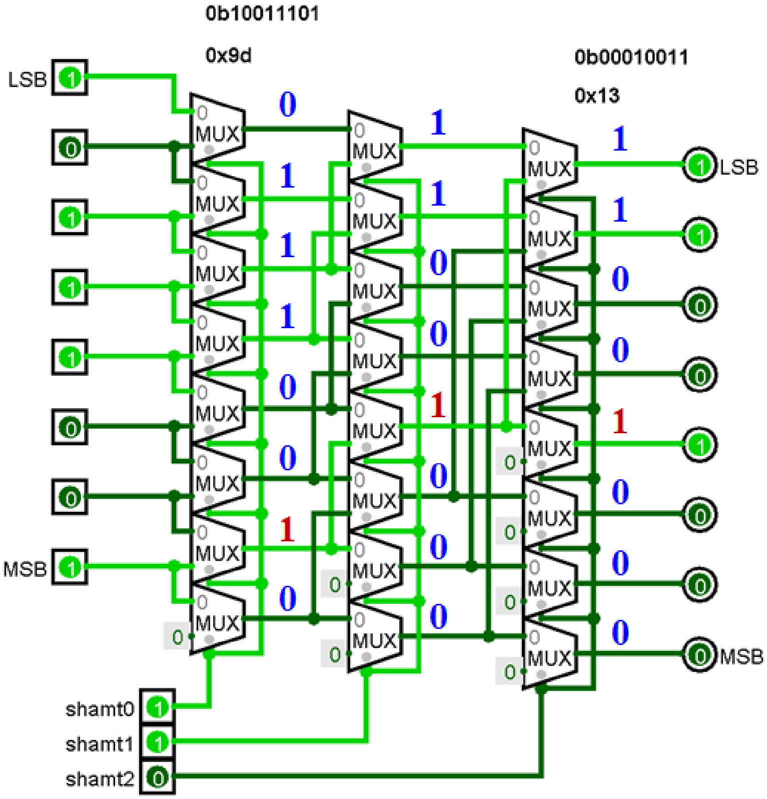
Decoder expansion



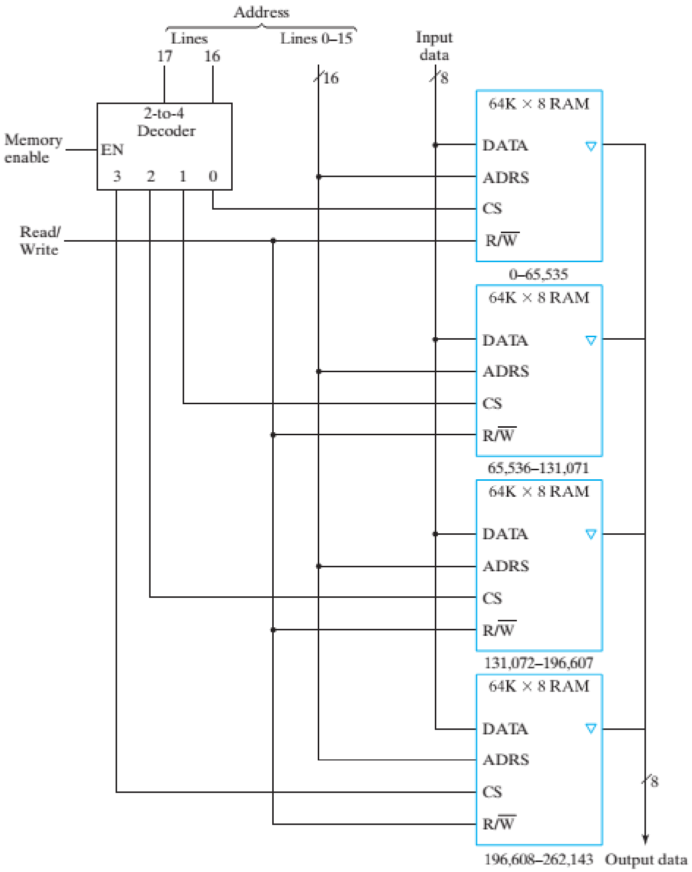
Lookup tables

Design of Combinational Circuits

- Design in a hierarchical structure



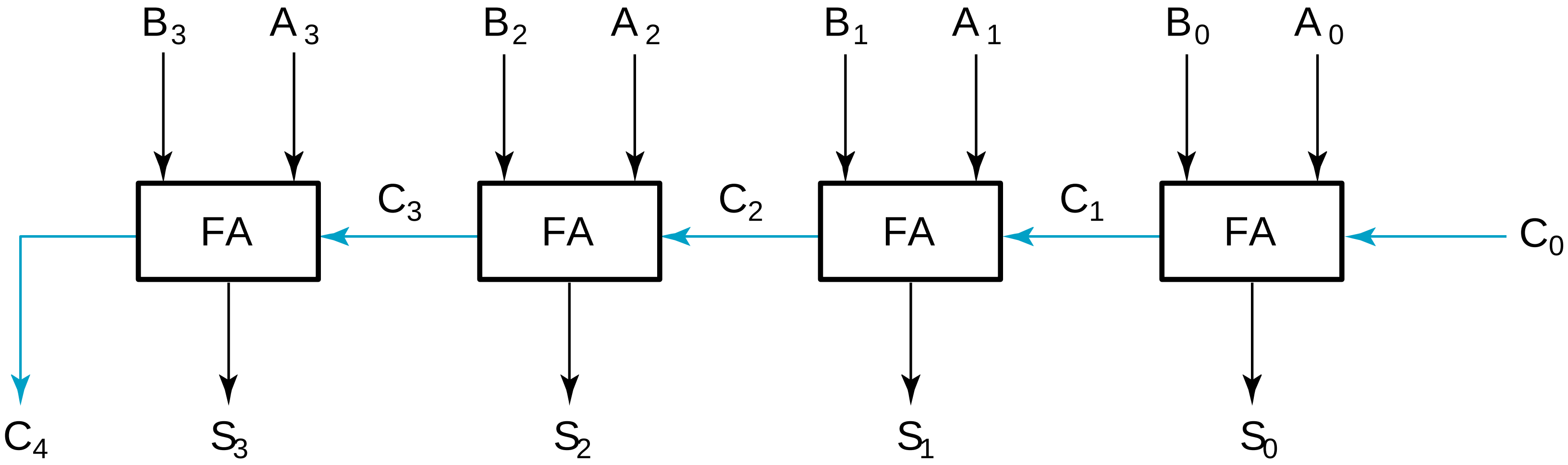
Large shifters using layers of mux



256K * 8 RAM with four 64K * 8 RAM

Design of Combinational Circuits

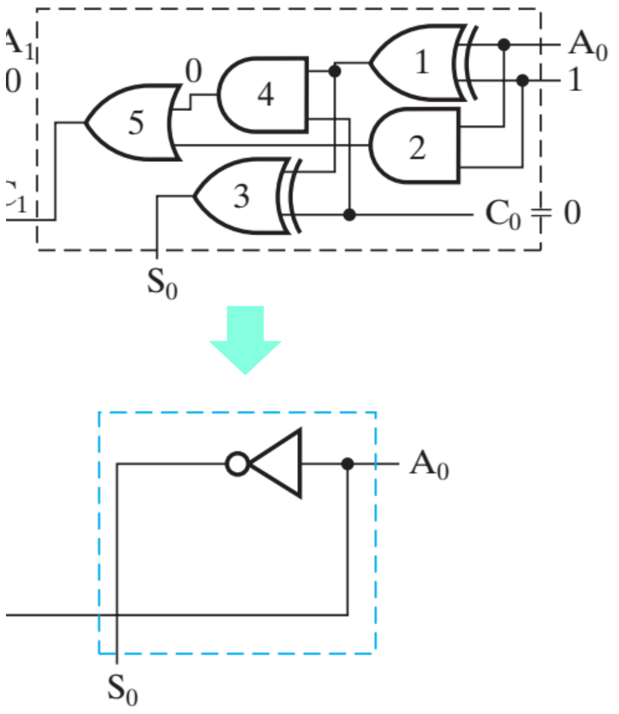
- Design by iteration



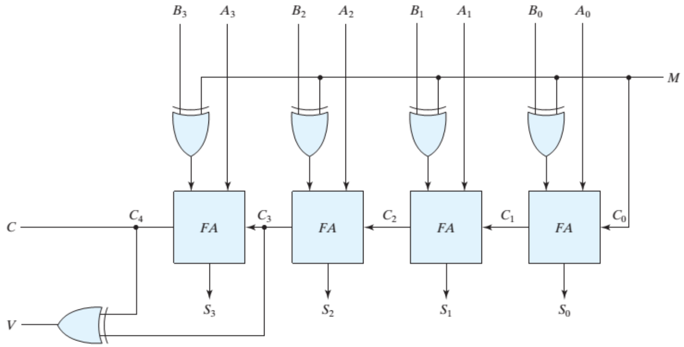
A four-bit Ripple Carry Adder

Design of Combinational Circuits

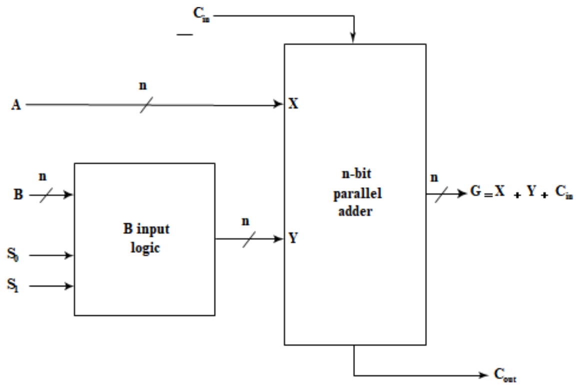
- Design by contraction/expansion



Contraction of Adder to Incrementer



Expansion of inputs for constructing substractor



Expansion of inputs for constructing ALU

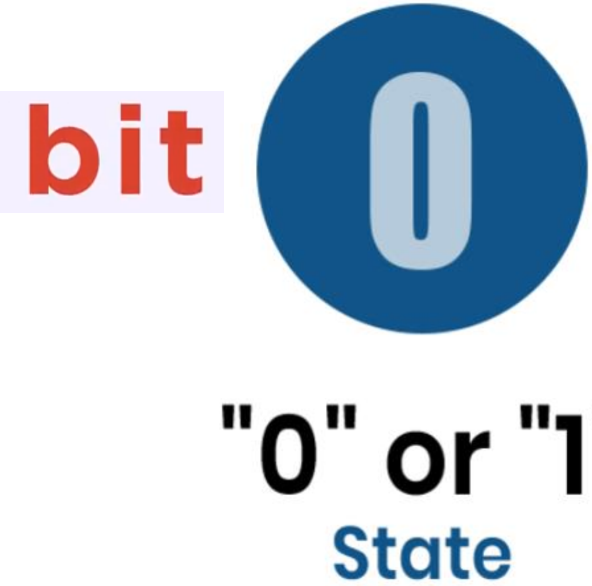
From classical computing to quantum computing

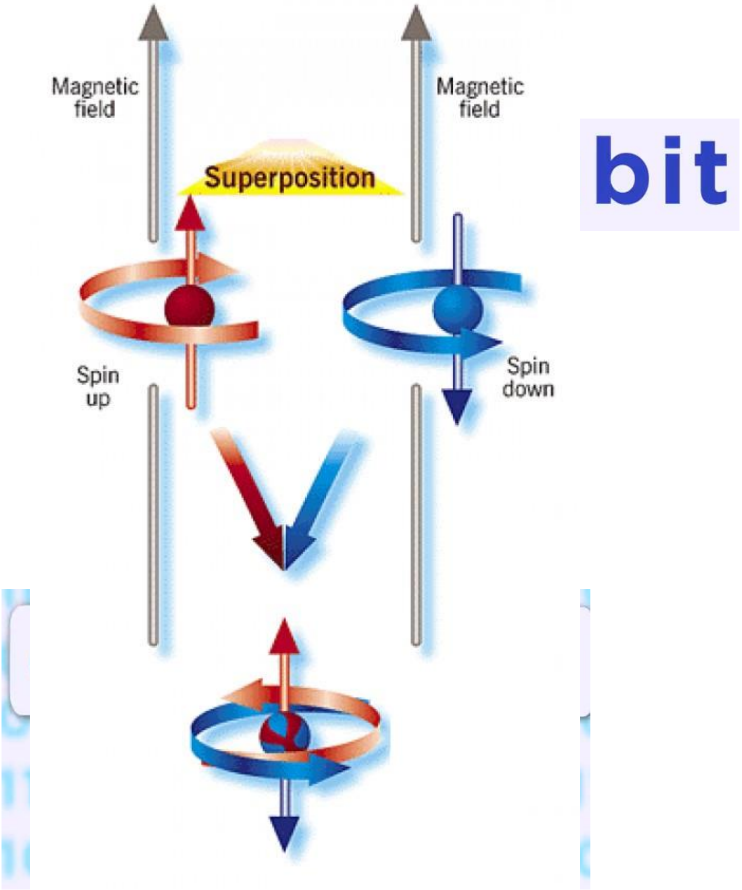
From classical computing to quantum computing

Information Representation

classical computing

quantum computing





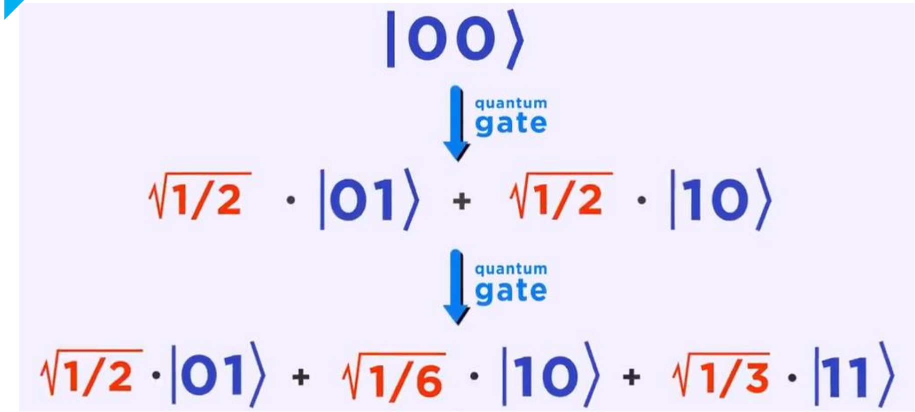
From classical computing to quantum computing

Information Representation

classical computing

quantum computing

superposition


$|00\rangle$

quantum gate

$\sqrt{1/2}\cdot|01\rangle + \sqrt{1/2}\cdot|10\rangle$

quantum gate

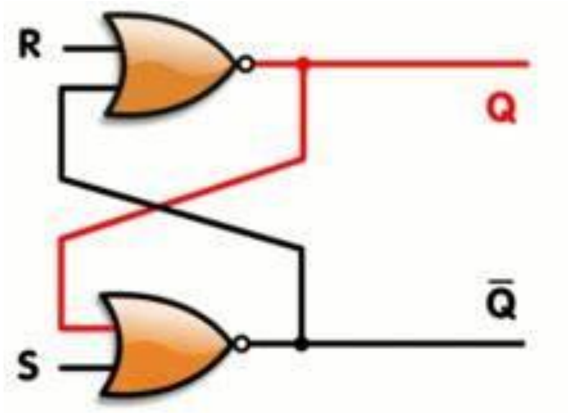
$\sqrt{1/2}\cdot|01\rangle + \sqrt{1/6}\cdot|10\rangle + \sqrt{1/3}\cdot|11\rangle$

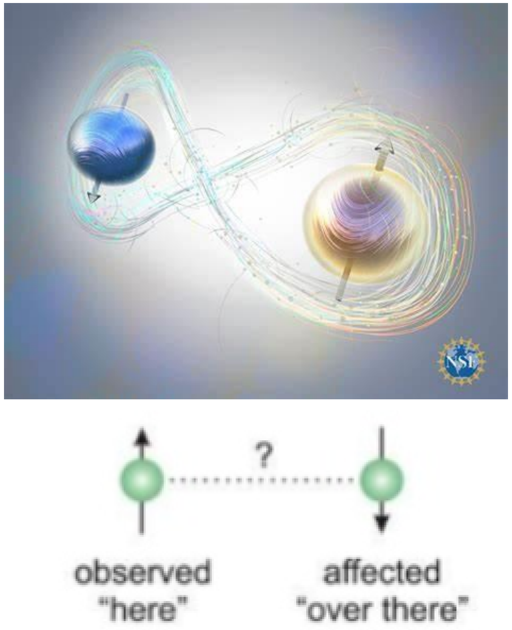
From classical computing to quantum computing

Information Representation

classical computing

quantum computing





cross-coupled gates

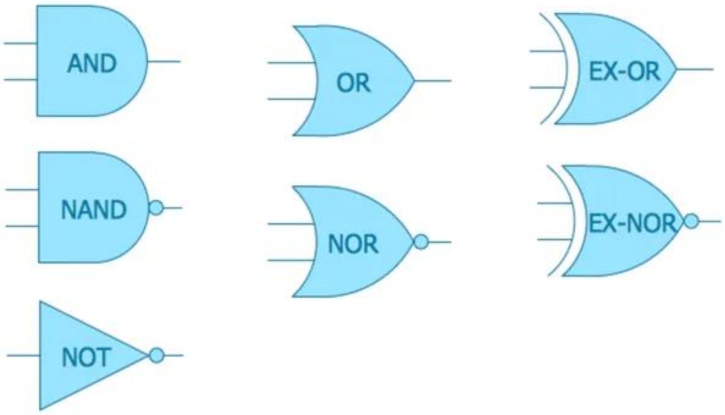
entanglement

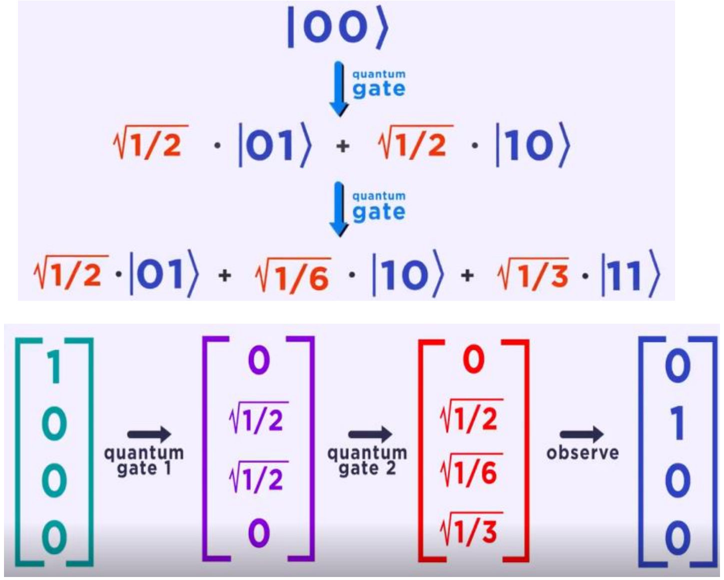
From classical computing to quantum computing

Logic Gate and Logic Operator

classical computing

quantum computing




$|00\rangle$
quantum gate
$\sqrt{1/2}\cdot|01\rangle + \sqrt{1/2}\cdot|10\rangle$
quantum gate
$\sqrt{1/2}\cdot|01\rangle + \sqrt{1/6}\cdot|10\rangle + \sqrt{1/3}\cdot|11\rangle$
$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$
quantum gate 1
$\begin{bmatrix}0\\\sqrt{1/2}\\\sqrt{1/2}\\0\end{bmatrix}$
quantum gate 2
$\begin{bmatrix}0\\\sqrt{1/2}\\\sqrt{1/6}\\\sqrt{1/3}\end{bmatrix}$
observe
$\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$

boolean operator

unitary operator

From classical computing to quantum computing

Logic Gate and Logic Operator

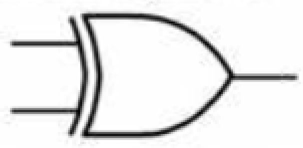
classical computing

quantum computing

boolean operator ⇒ unitary operator

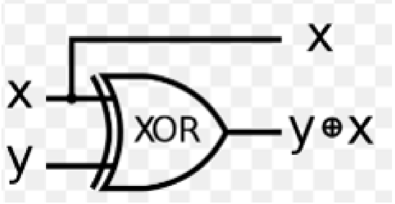
- linear transformation
- reversible logic

Inputs		Outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0



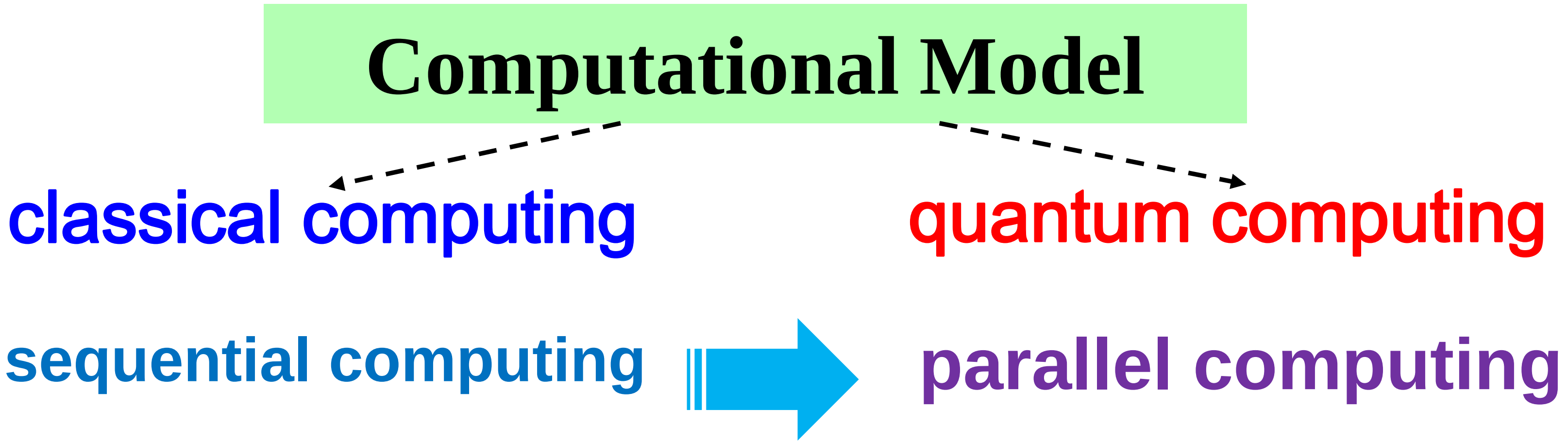
XOR gate

input		output	
x	y	x	$y \oplus x$
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0



Controlled NOT gate

From classical computing to quantum computing



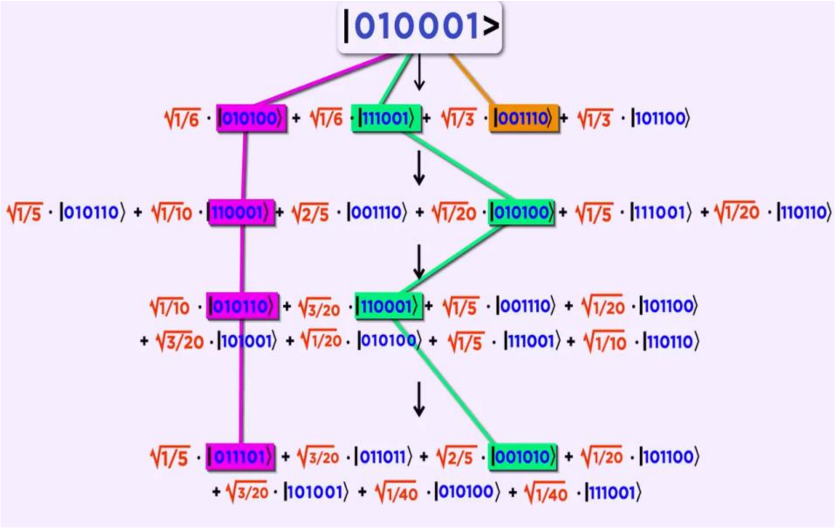
Computational Model

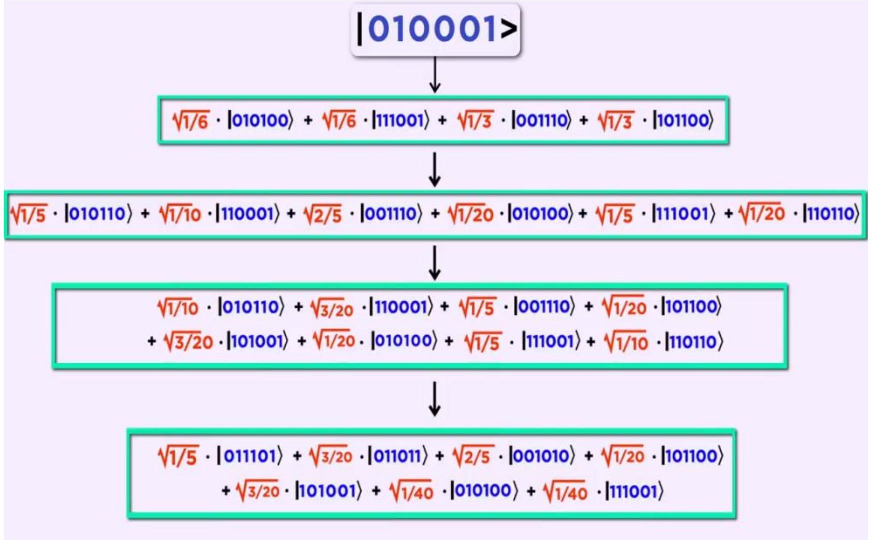
classical computing

quantum computing

sequential computing

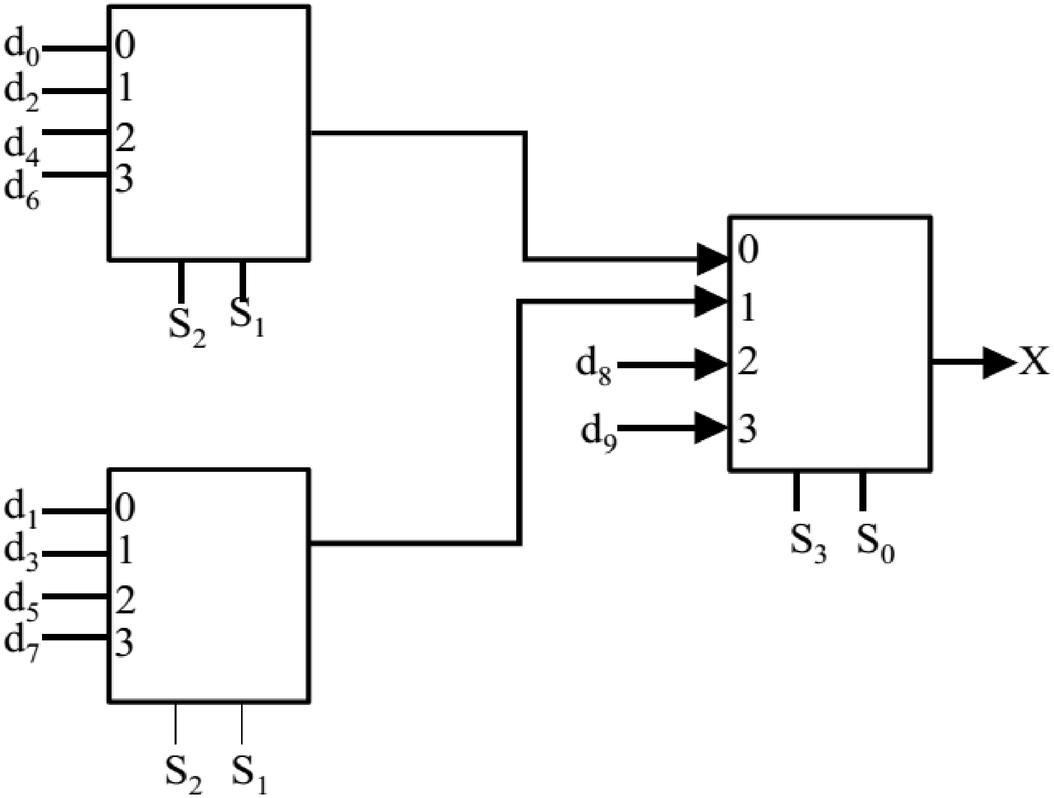
parallel computing





Appendix A: Chapter 3

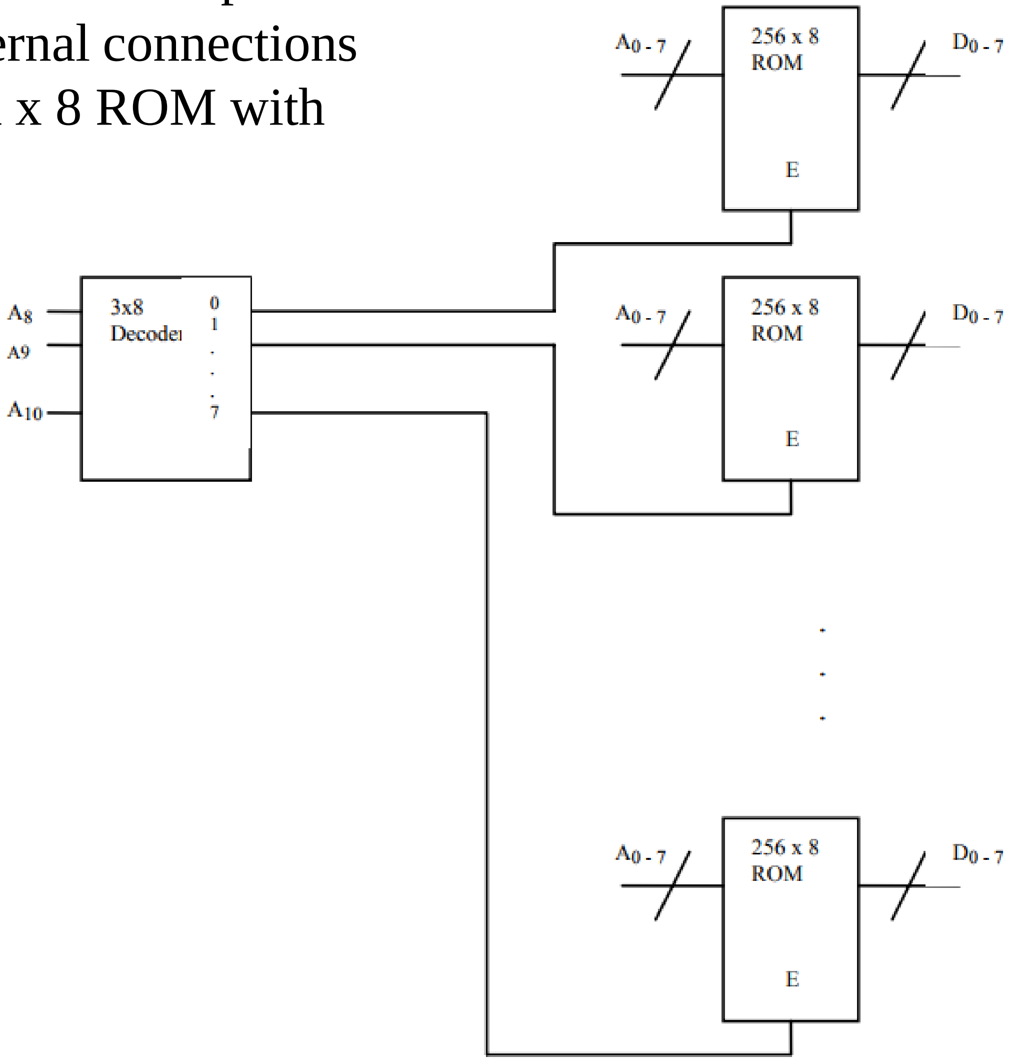
- **Problem:** Construct a 10-to-1 line multiplexer with three 4-to-1 line multiplexers. The selection codes 0000 through 1001 can be directly applied to the multiplexer selections inputs without added logic.



S3	S2	S1	S0	d
0	0	0	0	d0
0	0	0	1	d1
0	0	1	0	d2
0	0	1	1	d3
0	1	0	0	d4
0	1	0	1	d5
0	1	1	0	d6
0	1	1	1	d7
1	0	0	0	d8
1	0	0	1	d9

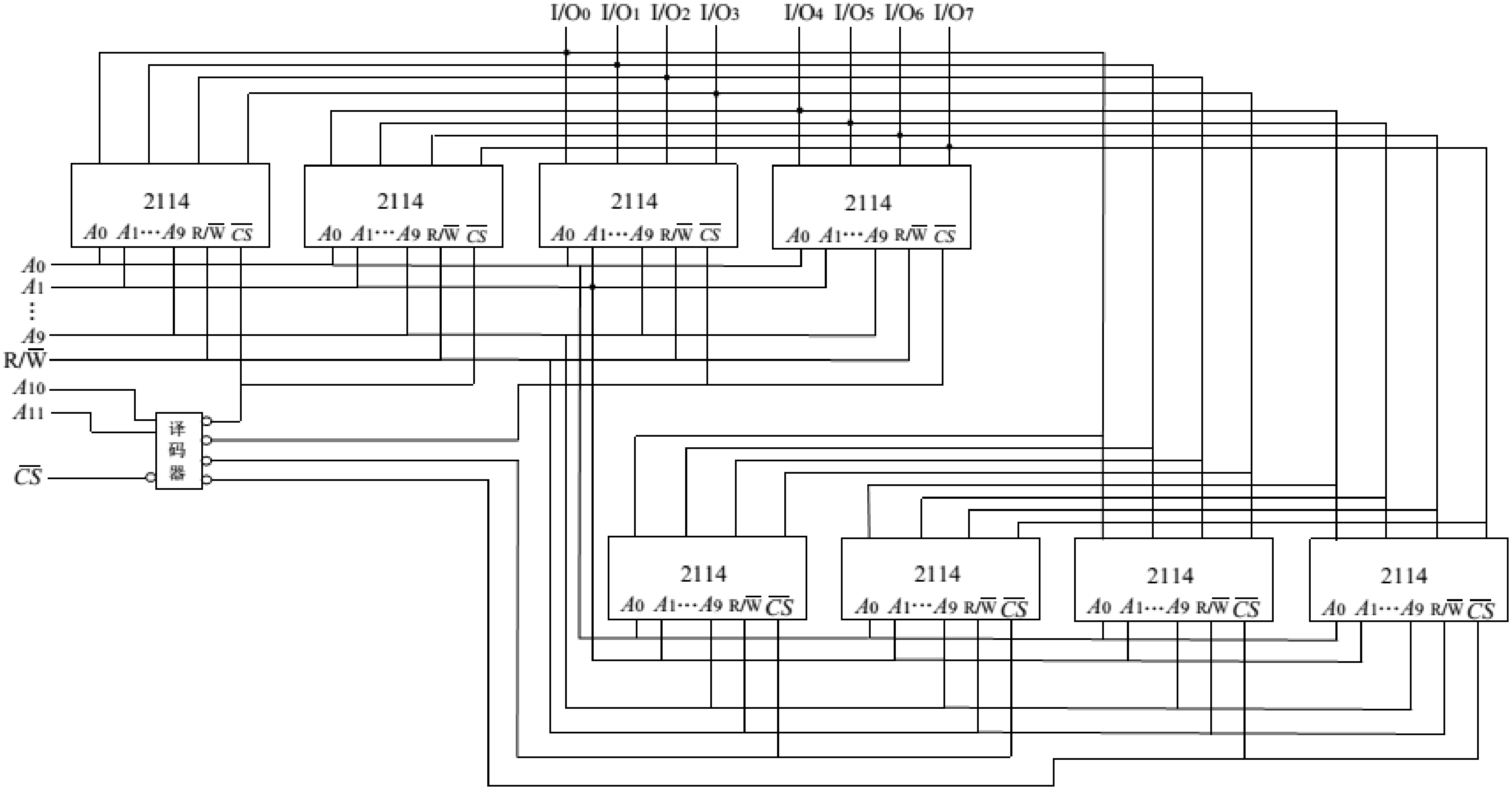
Appendix B: Chapter 5

- Problem: Given a 256 x 8 ROM chip with Enable input, show the external connections necessary to construct a 2K x 8 ROM with eight chips and a decoder.



Chapter 5

- **Problem:** Given a 1K x 4 ROM chip with Enable input, show the external connections necessary to construct a 4K x 8 ROM with eight chips and a decoder.



I/O0 I/O1 I/O2 I/O3
I/O4 I/O5 I/O6 I/O7
2114
A0 A1⋯A9 R/W̄ C̄S̄
A0
A1
A9
R/W̄
A10
A11
C̄S̄
译码器