



# CSCI 150

## Introduction to Digital and Computer System Design

### Lecture 3: Combinational Logic Design II



Jetic Gū

# Overview

- Focus: Methodology
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.1, 3.2, 3.3; v5: Ch3 3.1, 3.2
- Core Ideas:
  1. BCD-to-Seven-Segment Decoder
  2. 4-bit Equity Comparator
  3. Technology Mapping

# Systematic Design Procedures

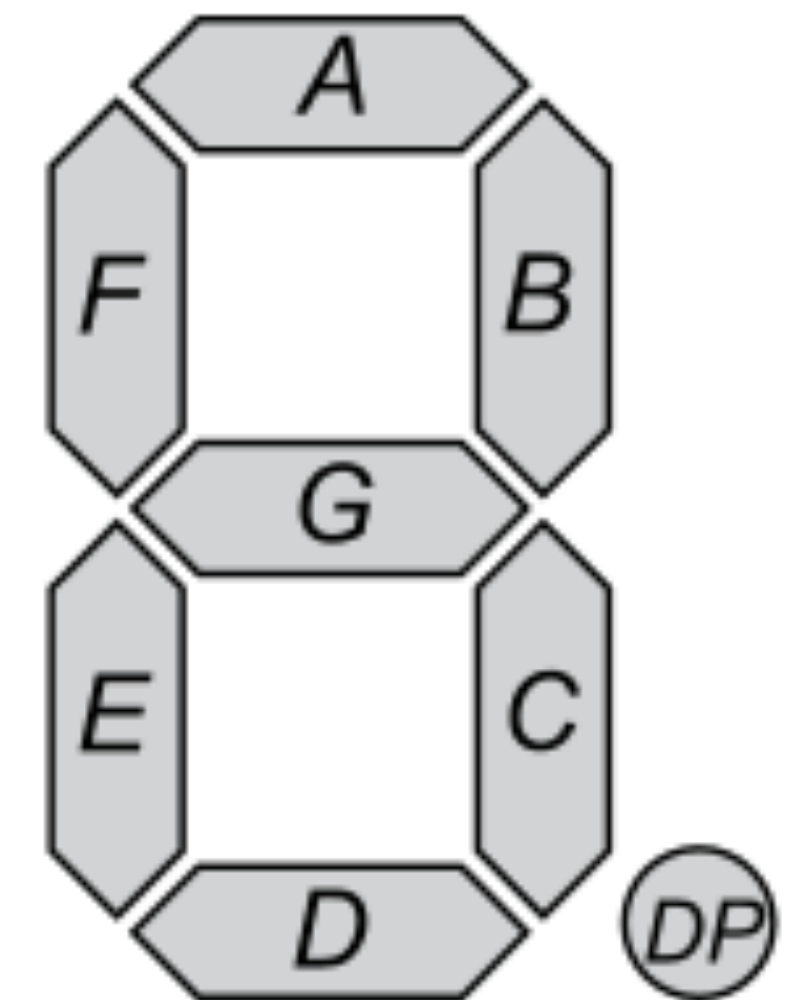
1. **Specification:** Write a specification for the circuit
2. **Formulation:** Derive relationship between inputs and outputs of the system e.g. using truth table or Boolean expressions
3. **Optimisation:** Apply optimisation, minimise the number of logic gates and literals required
4. **Technology Mapping:** Transform design to new diagram using available implementation technology
5. **Verification:** Verify the correctness of the final design in meeting the specifications

# BCD-to-Seven-Segment Decoder

Ah, not again

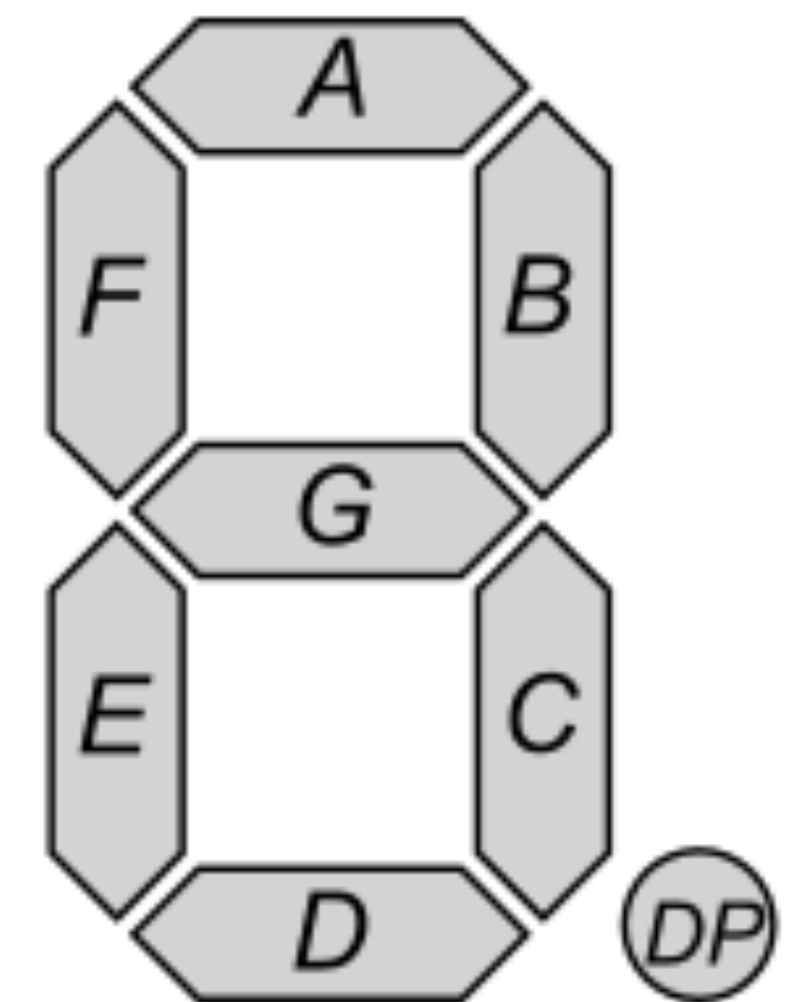
# LED Seven Segment Display

- LED: Light-Emitting Diodes
- A single digit display takes 7bit inputs (and an optional one for decimal point)



# BCD to 7 Segment Display(s)

- BCD: Each digit is represented using 4bit binary int
- BCD-to-7-segment decoder  
A Combinational circuit that
  - takes a decimal digit in BCD (4bit int  $A, B, C, D$ ); and
  - generates the appropriate control signals for the display unit ( $a, b, c, d, e, f, g$ )



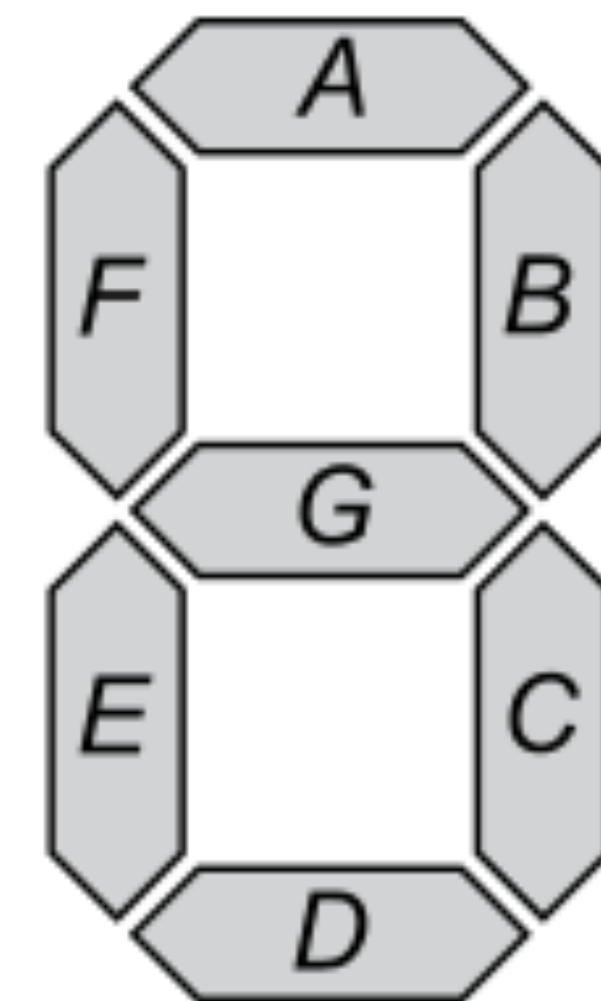
# 1. Specification

- BCD: Each digit is represented using 4bit binary int
- BCD-to-7-segment decoder  
A Combinational circuit that

**Input** • takes a decimal digit in BCD (4bit int  $A, B, C, D$ ); and

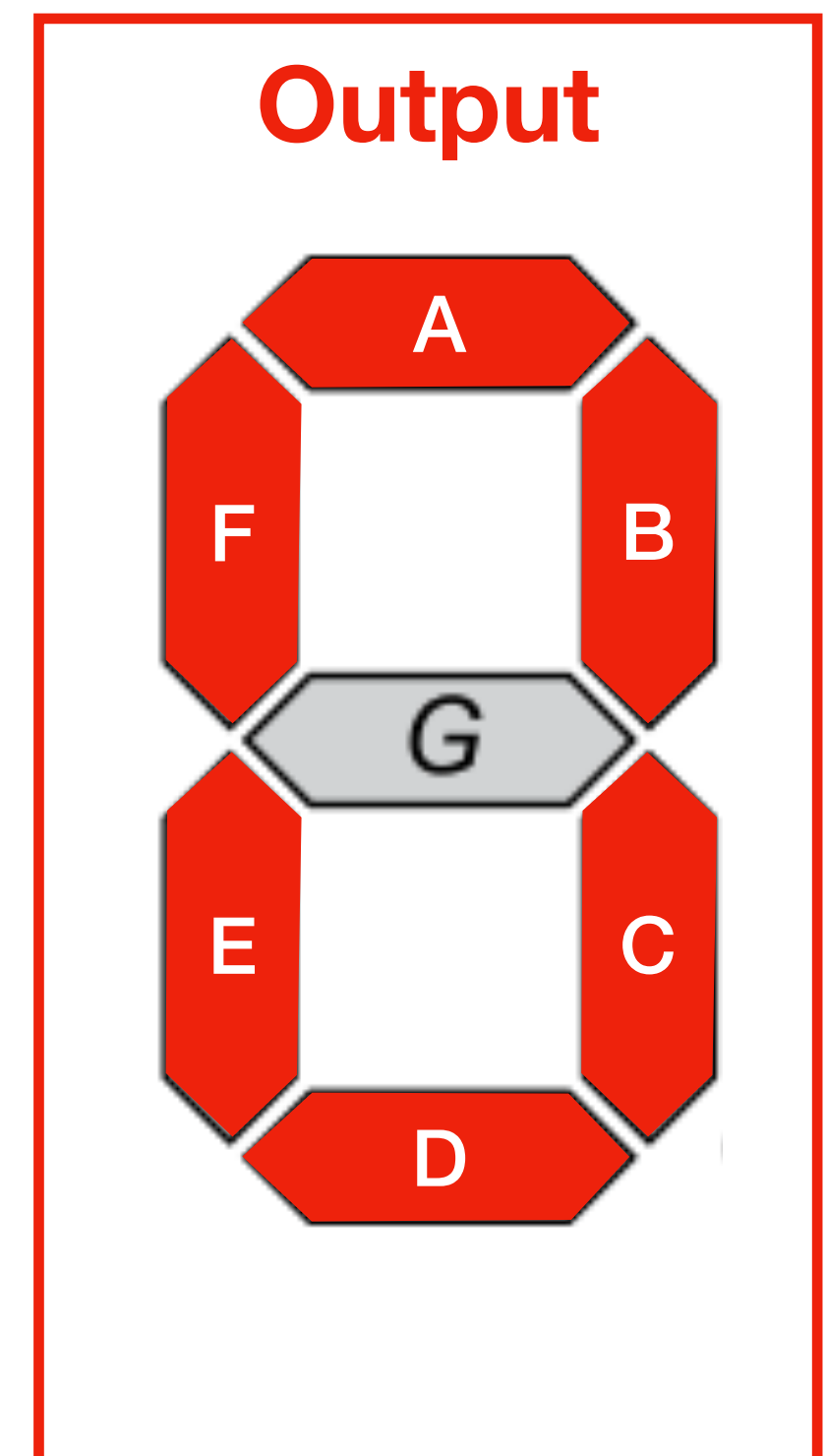
**Output** • generates the appropriate control signals for the display unit ( $a, b, c, d, e, f, g$ )

**Output**



# 2. Formulation

Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1											
2											
3											
4											
5											
6											
7											
8											
9											

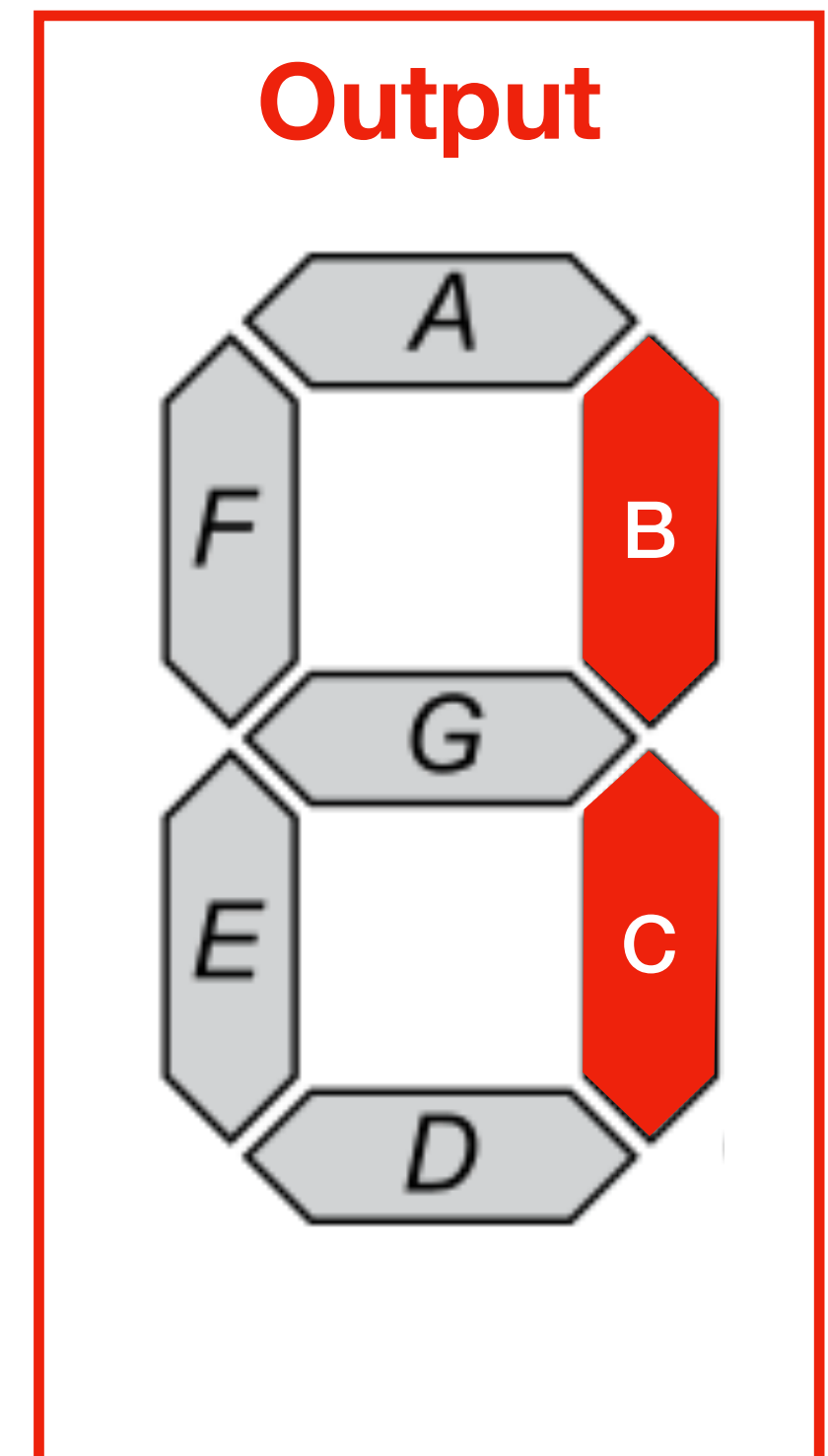


Example



# 2. Formulation

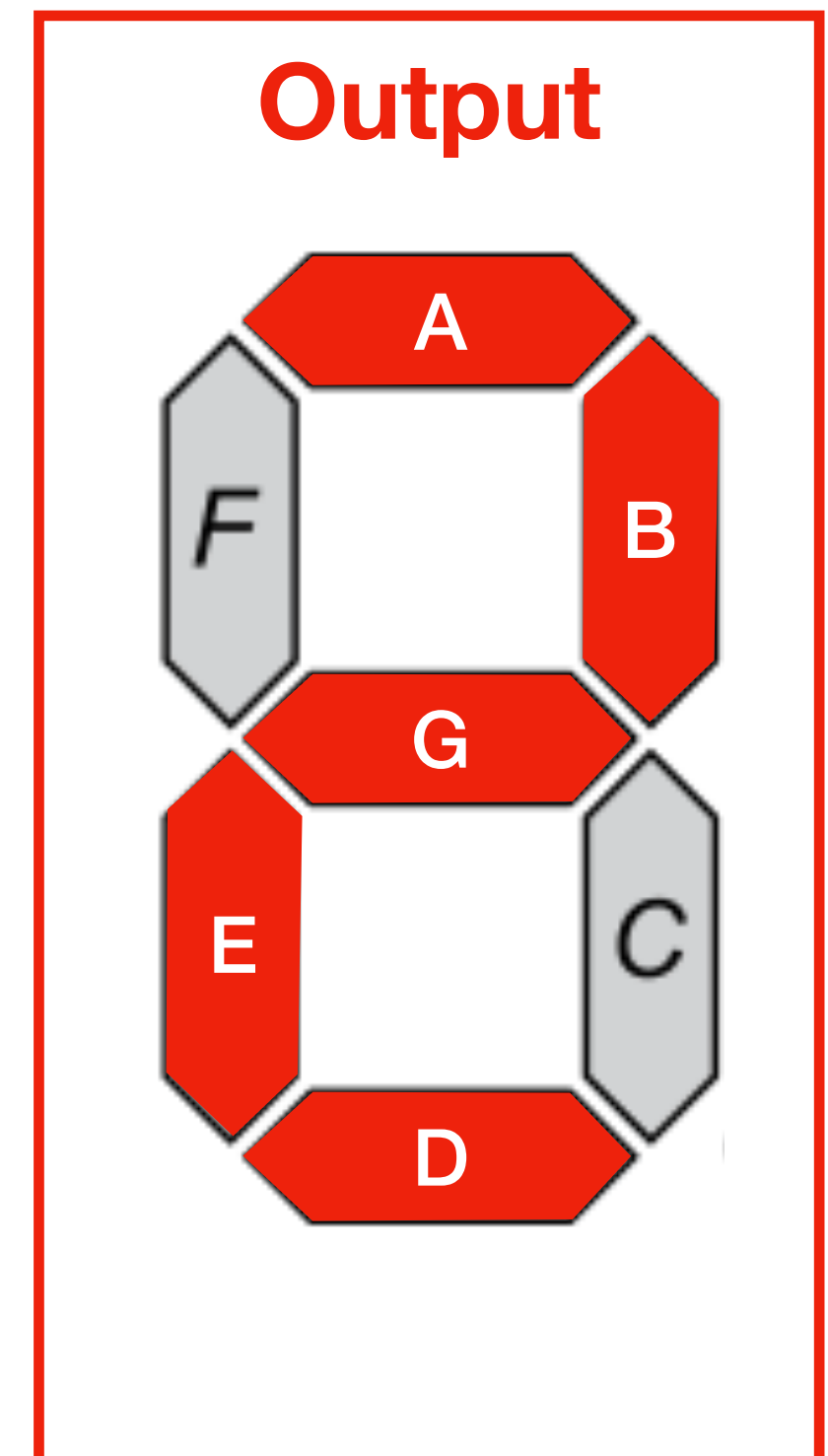
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2											
3											
4											
5											
6											
7											
8											
9											



Example

# 2. Formulation

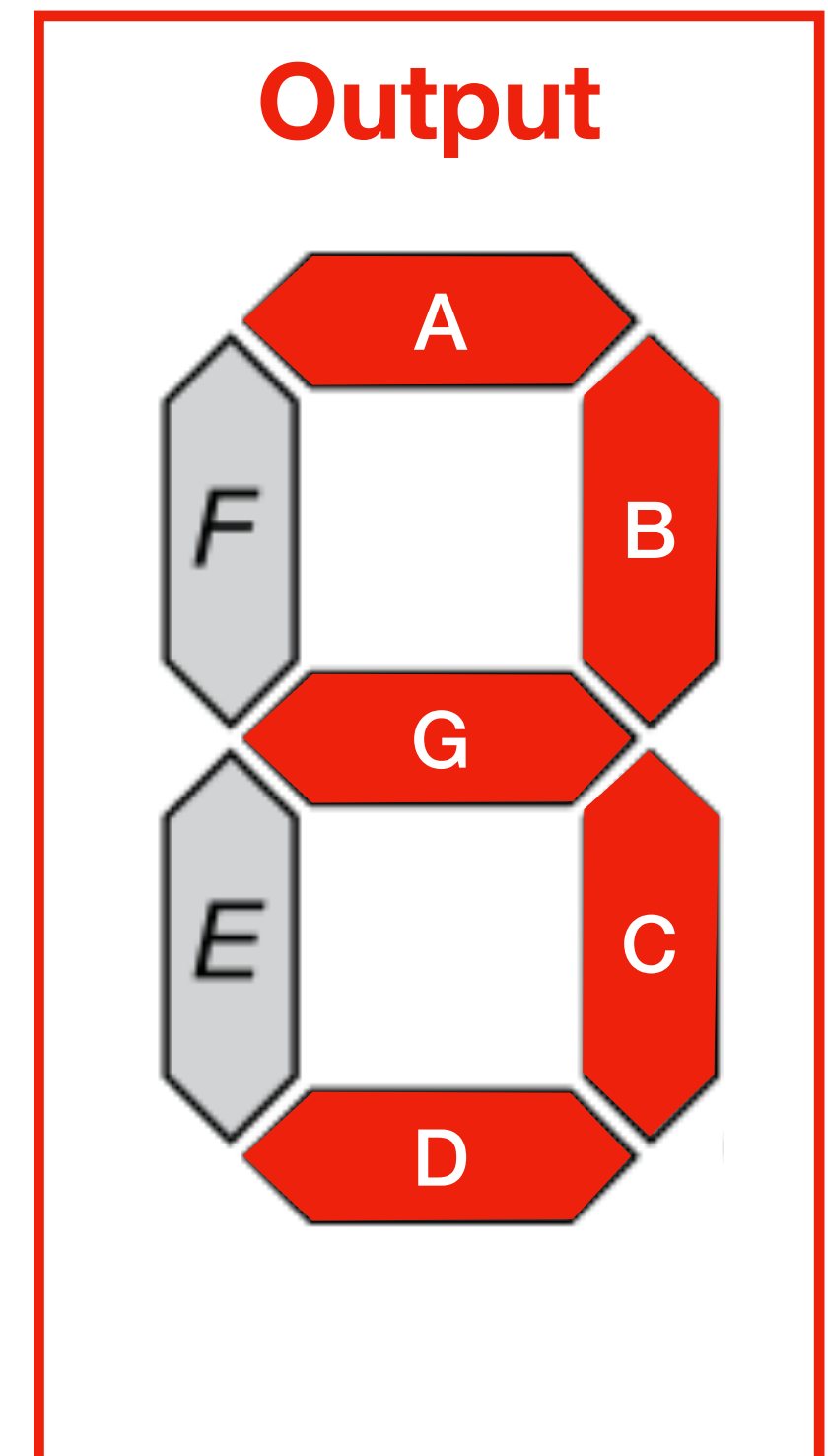
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3											
4											
5											
6											
7											
8											
9											



Example

# 2. Formulation

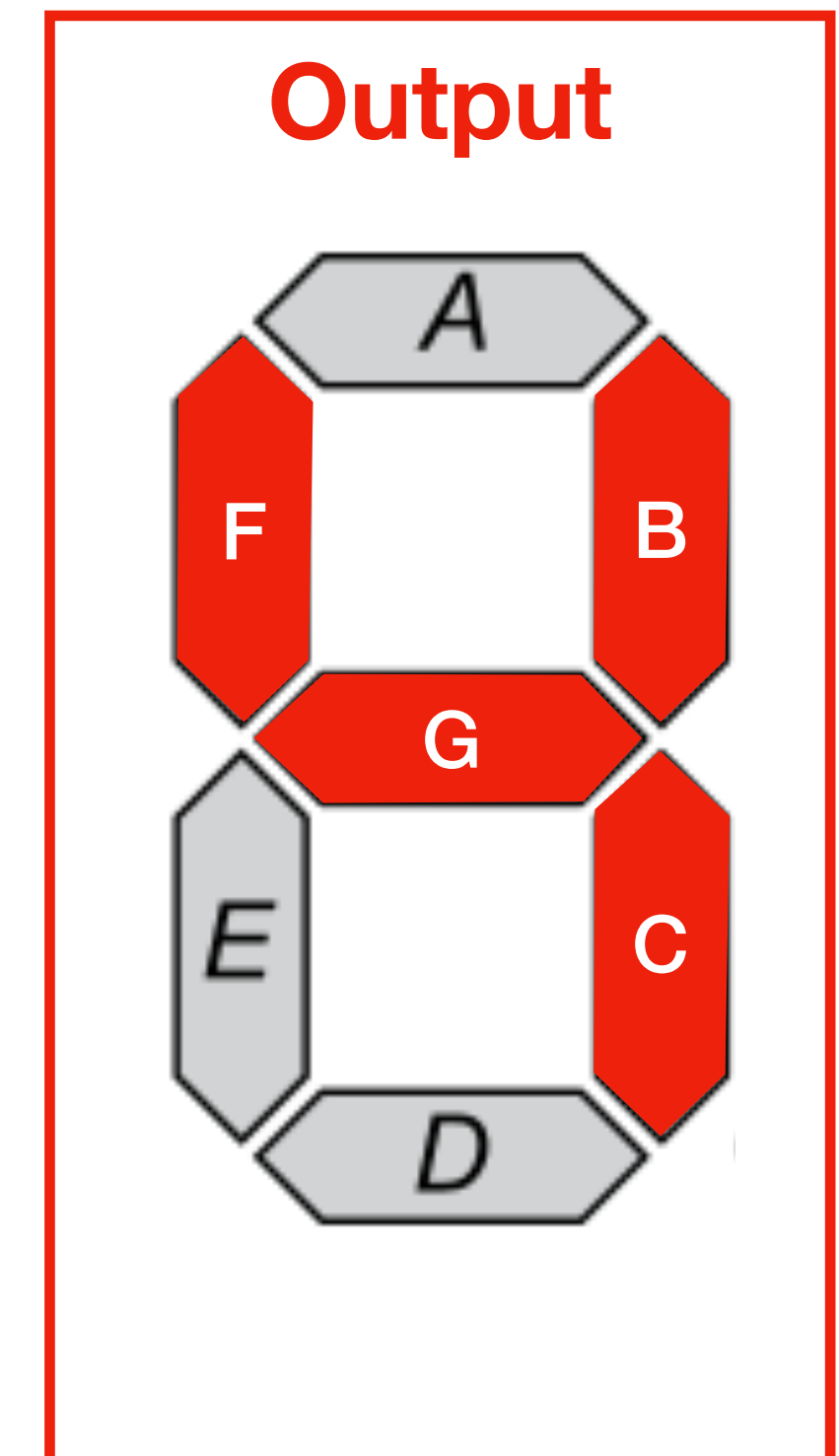
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4											
5											
6											
7											
8											
9											



Example

# 2. Formulation

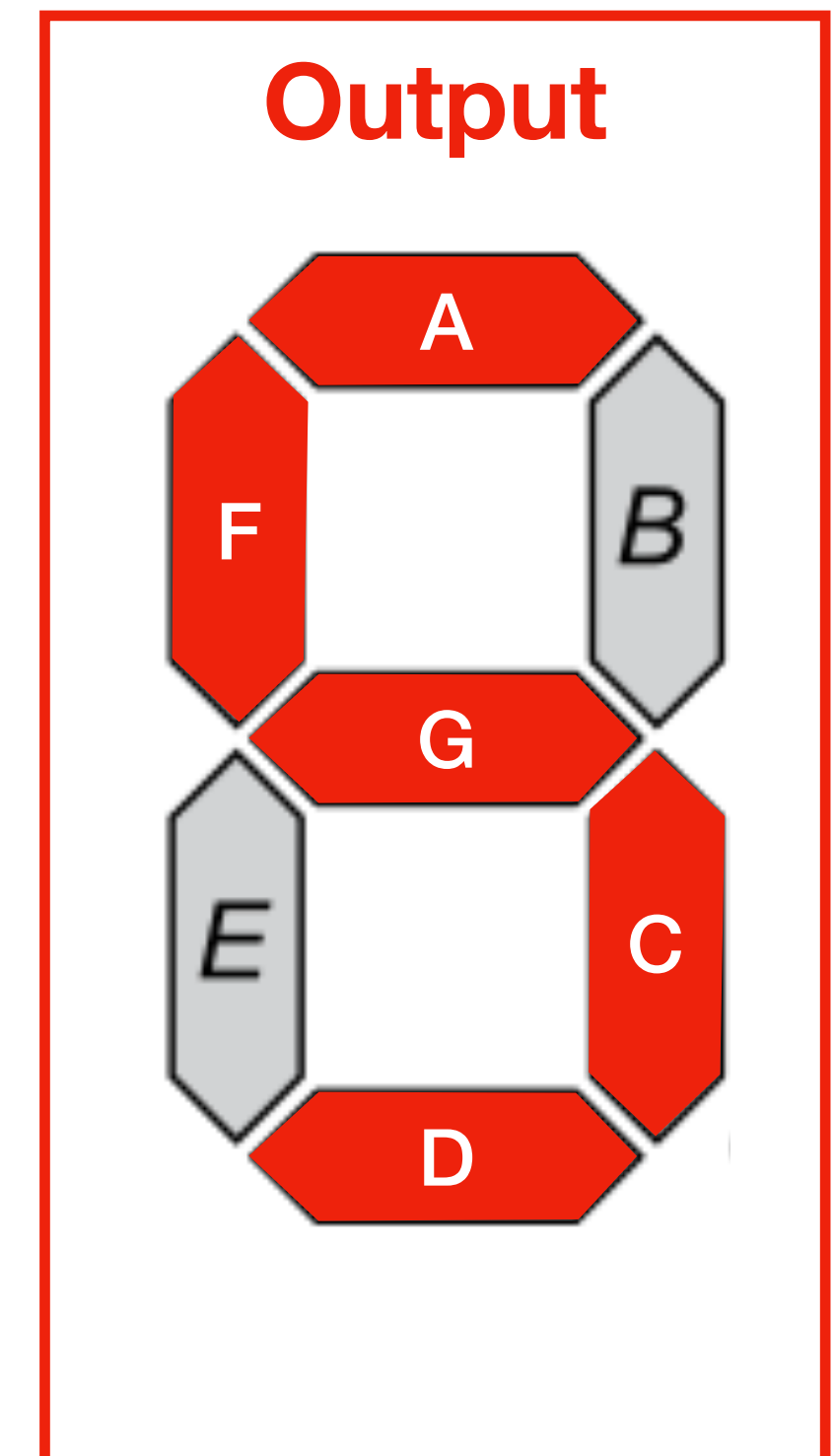
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4	0	1	0	0		1	1			1	1
5											
6											
7											
8											
9											



Example

# 2. Formulation

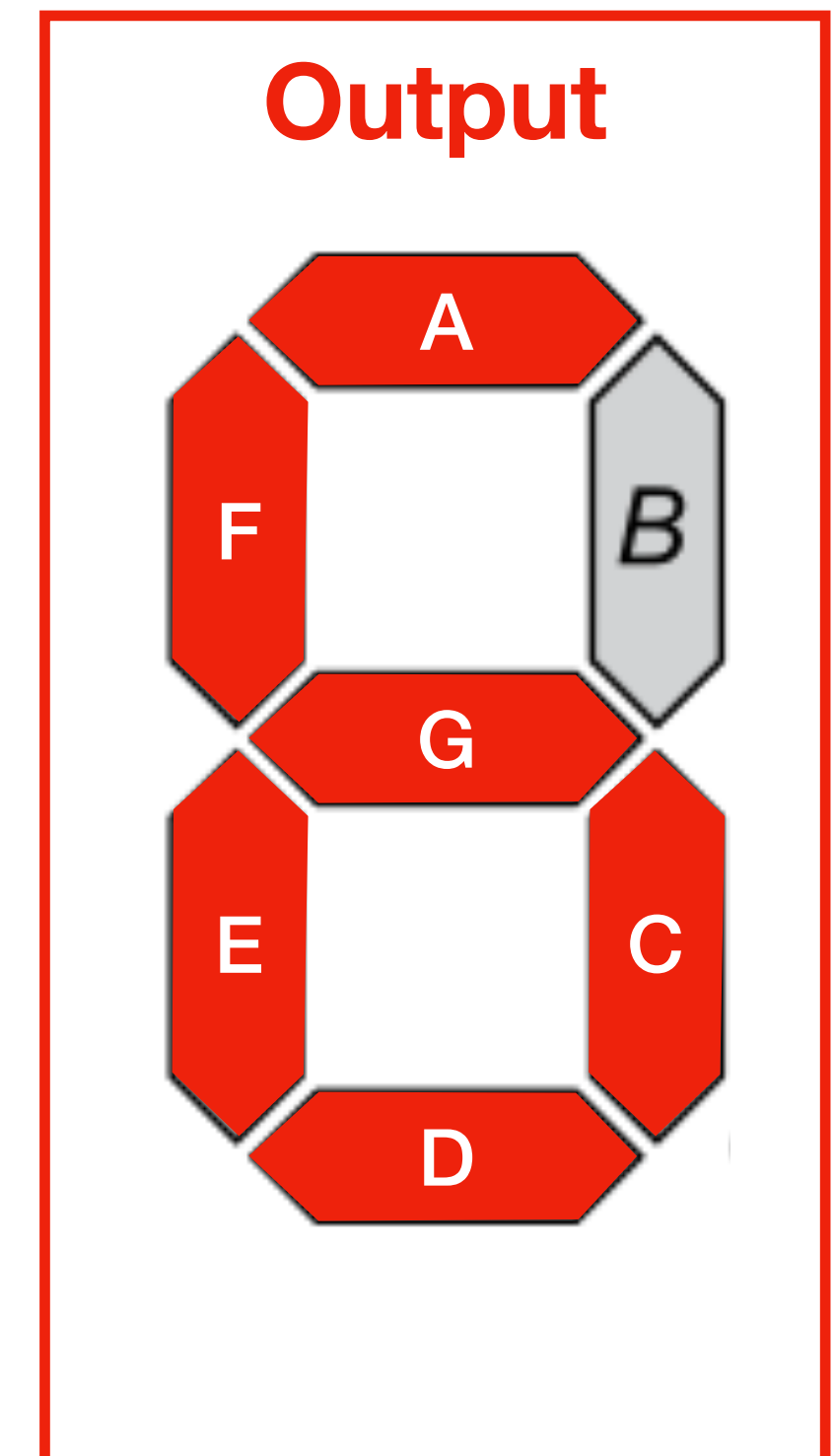
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4	0	1	0	0		1	1			1	1
5	0	1	0	1	1		1	1		1	1
6											
7											
8											
9											



Example

# 2. Formulation

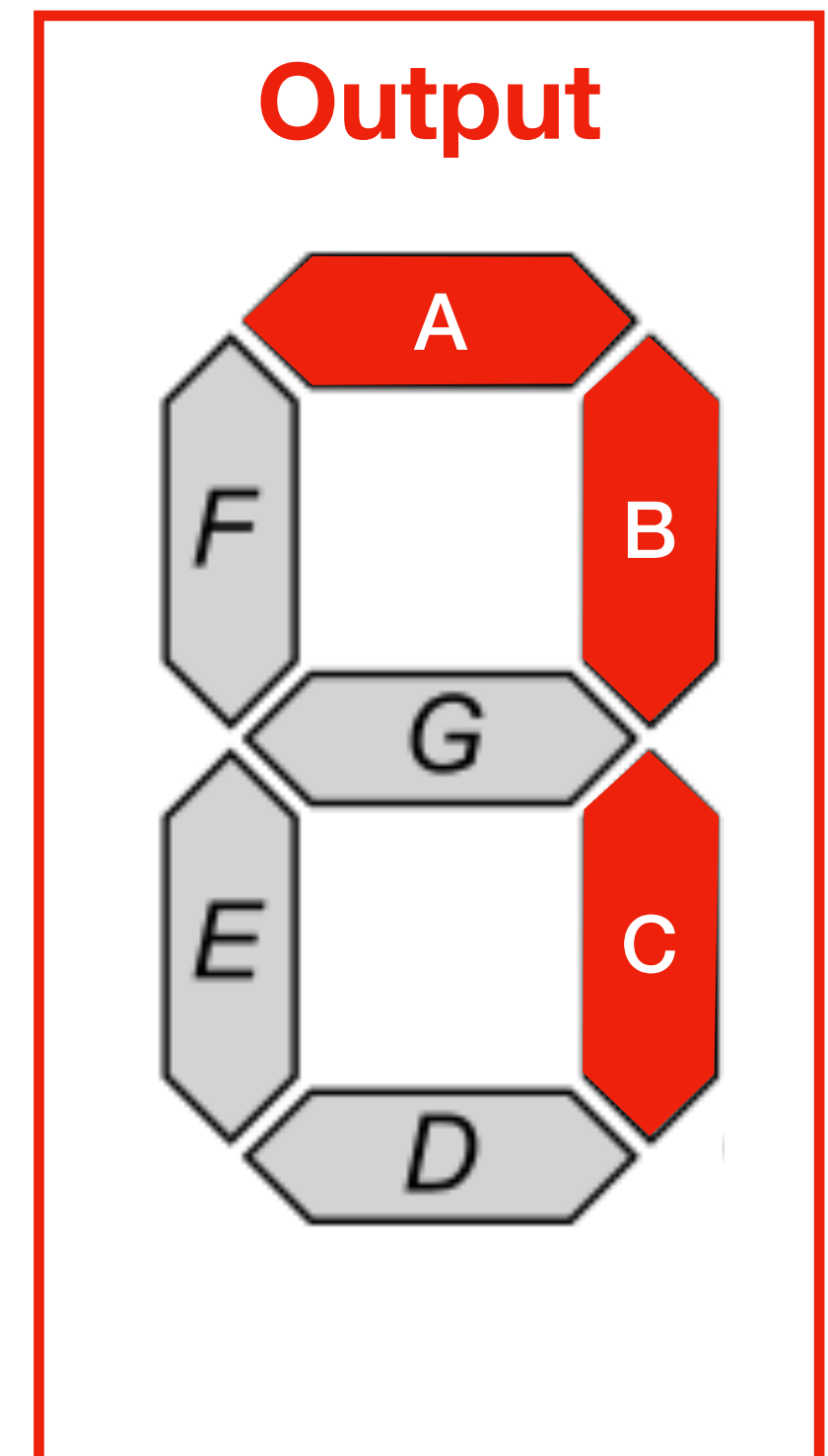
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4	0	1	0	0		1	1			1	1
5	0	1	0	1	1		1	1		1	1
6	0	1	1	0	1		1	1	1	1	1
7											
8											
9											



Example

# 2. Formulation

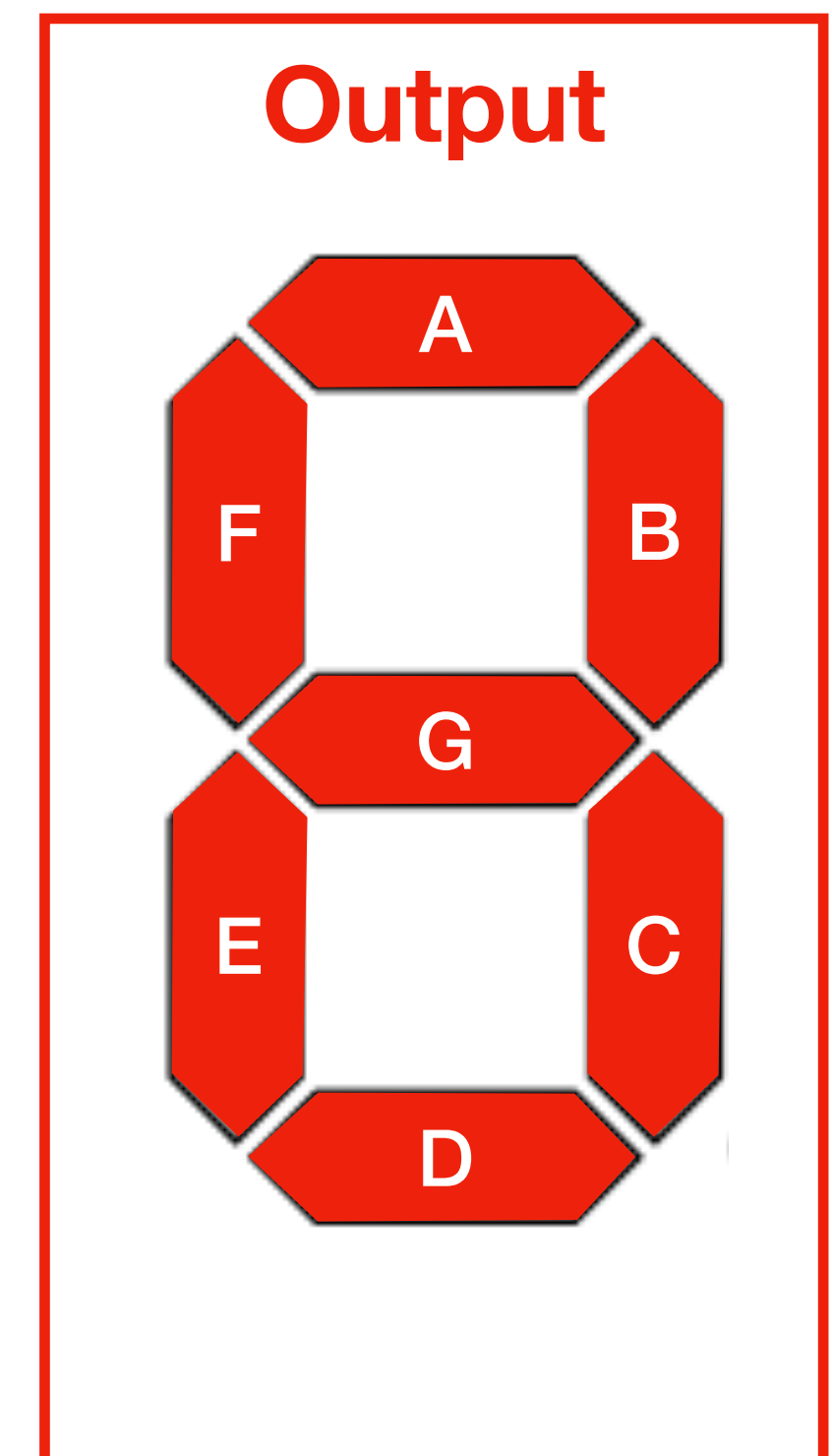
Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4	0	1	0	0		1	1			1	1
5	0	1	0	1	1		1	1		1	1
6	0	1	1	0	1		1	1	1	1	1
7	0	1	1	1	1	1	1				
8											
9											



Example

# 2. Formulation

Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4	0	1	0	0		1	1			1	1
5	0	1	0	1	1		1	1		1	1
6	0	1	1	0	1		1	1	1	1	1
7	0	1	1	1	1	1	1				
8	1	0	0	0	1	1	1	1	1	1	1
9											

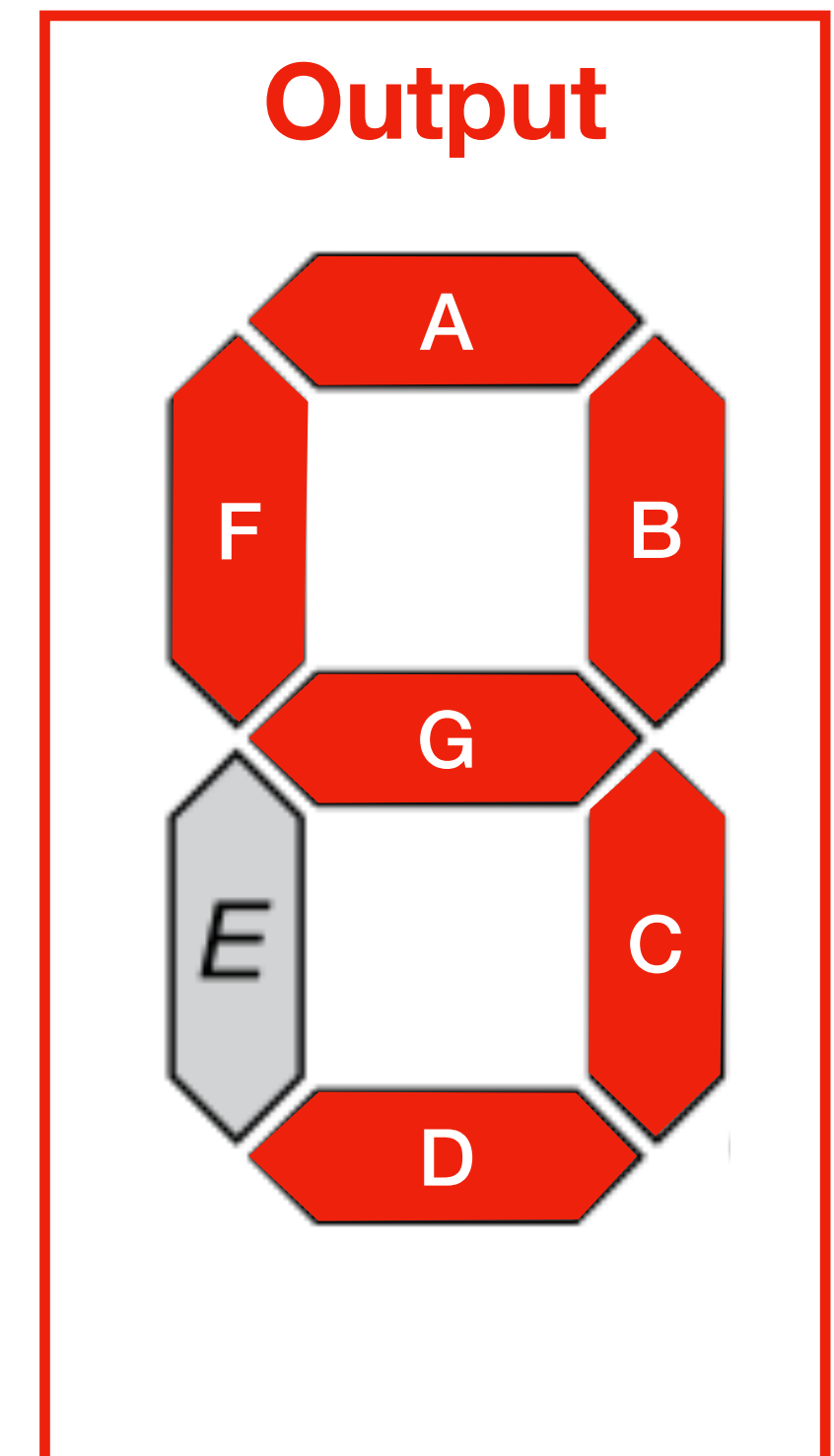


Example



# 2. Formulation

Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	
1	0	0	0	1		1	1				
2	0	0	1	0	1	1		1	1		1
3	0	0	1	1	1	1	1	1			1
4	0	1	0	0		1	1			1	1
5	0	1	0	1	1		1	1		1	1
6	0	1	1	0	1		1	1	1	1	1
7	0	1	1	1	1	1	1				
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1		1	1



Example

# 3. Optimisation

- Convert to Boolean expression

Decimal	A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

# 3. Optimisation

- Convert to Boolean expression

$$a = \bar{A}C + \bar{A}BD + \bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}$$

$$b = \bar{A}\bar{B} + \bar{A}\bar{C}\bar{D} + \bar{A}CD + A\bar{B}\bar{C}$$

$$c = \bar{A}B + \bar{A}D + \bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}$$

$$d = \bar{A}C\bar{D} + \bar{A}\bar{B}C + \bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C} + \bar{A}B\bar{C}D$$

$$e = \bar{A}C\bar{D} + \bar{B}\bar{C}\bar{D}$$

$$f = \bar{A}B\bar{C} + \bar{A}\bar{C}\bar{D} + \bar{A}B\bar{D} + A\bar{B}\bar{C}$$

$$g = \bar{A}C\bar{D} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$$

# 4. Technology Mapping

- Convert to Boolean expression
- Indie: 27 AND  
7 OR gates

$$a = \bar{A}C + \bar{A}BD + \boxed{\bar{B}\bar{C}\bar{D}} + \boxed{A\bar{B}\bar{C}}$$

$$b = \bar{A}\bar{B} + \boxed{\bar{A}\bar{C}\bar{D}} + \bar{A}CD + \boxed{A\bar{B}\bar{C}}$$

$$c = \bar{A}B + \bar{A}D + \boxed{\bar{B}\bar{C}\bar{D}} + \boxed{A\bar{B}\bar{C}}$$

$$d = \boxed{\bar{A}C\bar{D}} + \boxed{\bar{A}\bar{B}C} + \boxed{\bar{B}\bar{C}\bar{D}} + \boxed{A\bar{B}\bar{C}} + \bar{A}B\bar{C}D$$

$$e = \boxed{\bar{A}C\bar{D}} + \boxed{\bar{B}\bar{C}\bar{D}}$$

$$f = \boxed{\bar{A}B\bar{C}} + \boxed{\bar{A}\bar{C}\bar{D}} + \bar{A}B\bar{D} + \boxed{A\bar{B}\bar{C}}$$

$$g = \boxed{\bar{A}C\bar{D}} + \boxed{\bar{A}\bar{B}C} + \boxed{\bar{A}B\bar{C}} + \boxed{A\bar{B}\bar{C}}$$

# 5. Verification (Skipped here)

- Convert to Boolean expression
- Indie: 27 AND  
7 OR gates

$$a = \bar{A}C + \bar{A}BD + \boxed{\bar{B}\bar{C}\bar{D}} + \boxed{A\bar{B}\bar{C}}$$

$$b = \bar{A}\bar{B} + \boxed{\bar{A}\bar{C}\bar{D}} + \bar{A}CD + \boxed{A\bar{B}\bar{C}}$$

$$c = \bar{A}B + \bar{A}D + \boxed{\bar{B}\bar{C}\bar{D}} + \boxed{A\bar{B}\bar{C}}$$

$$d = \boxed{\bar{A}C\bar{D}} + \boxed{\bar{A}\bar{B}C} + \boxed{\bar{B}\bar{C}\bar{D}} + \boxed{A\bar{B}\bar{C}} + \bar{A}B\bar{C}D$$

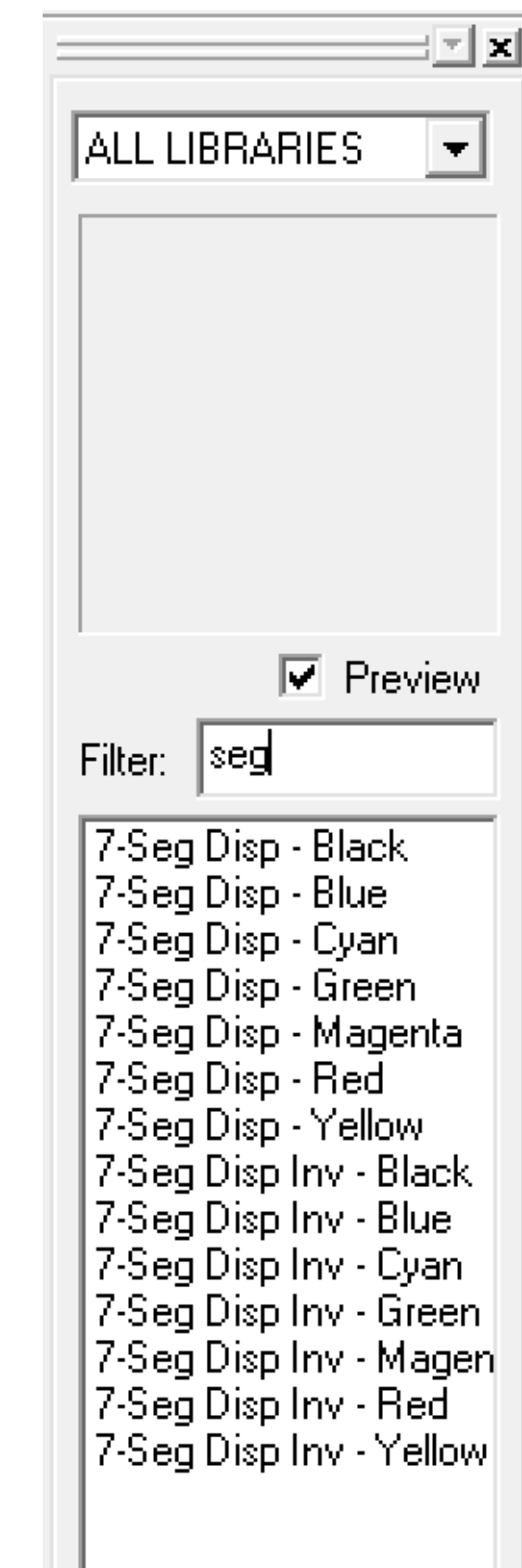
$$e = \boxed{\bar{A}C\bar{D}} + \boxed{\bar{B}\bar{C}\bar{D}}$$

$$f = \boxed{\bar{A}B\bar{C}} + \boxed{\bar{A}\bar{C}\bar{D}} + \bar{A}B\bar{D} + \boxed{A\bar{B}\bar{C}}$$

$$g = \boxed{\bar{A}C\bar{D}} + \boxed{\bar{A}\bar{B}C} + \boxed{\bar{A}B\bar{C}} + \boxed{A\bar{B}\bar{C}}$$

# BCD to 7 Segment Display

- 'BCD-to-seven-segment *decoder*'
- '*decodes*' binary code for decimal digit
- usually decoders mean something different, which we'll discuss later



# 4-bit Equity Comparator

Ah, not again

# 4-bit Equity Comparator

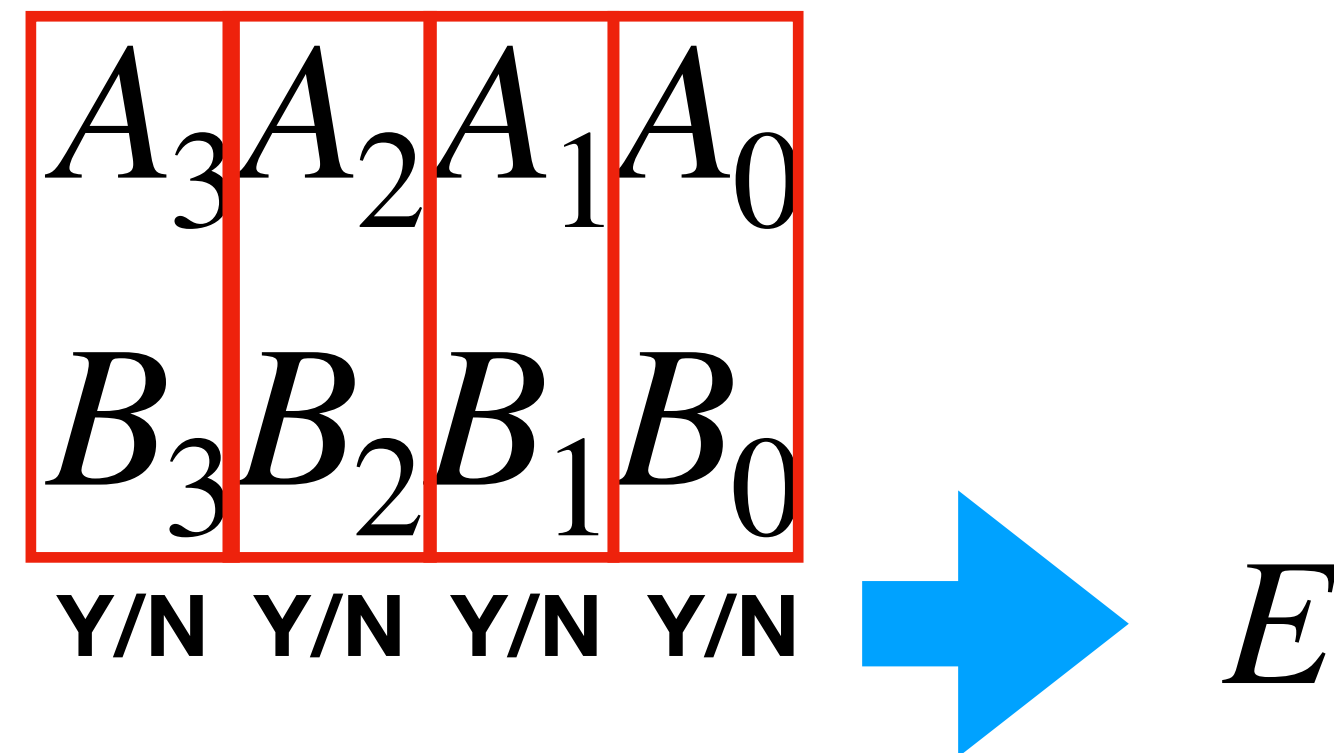
- Compare two numbers
- Equal or not



# 1. Specification

- Input:  $A_3A_2A_1A_0, B_3B_2B_1B_0$
- Output:  $E = 1$  for equal, 0 for not

# 2. Formulation

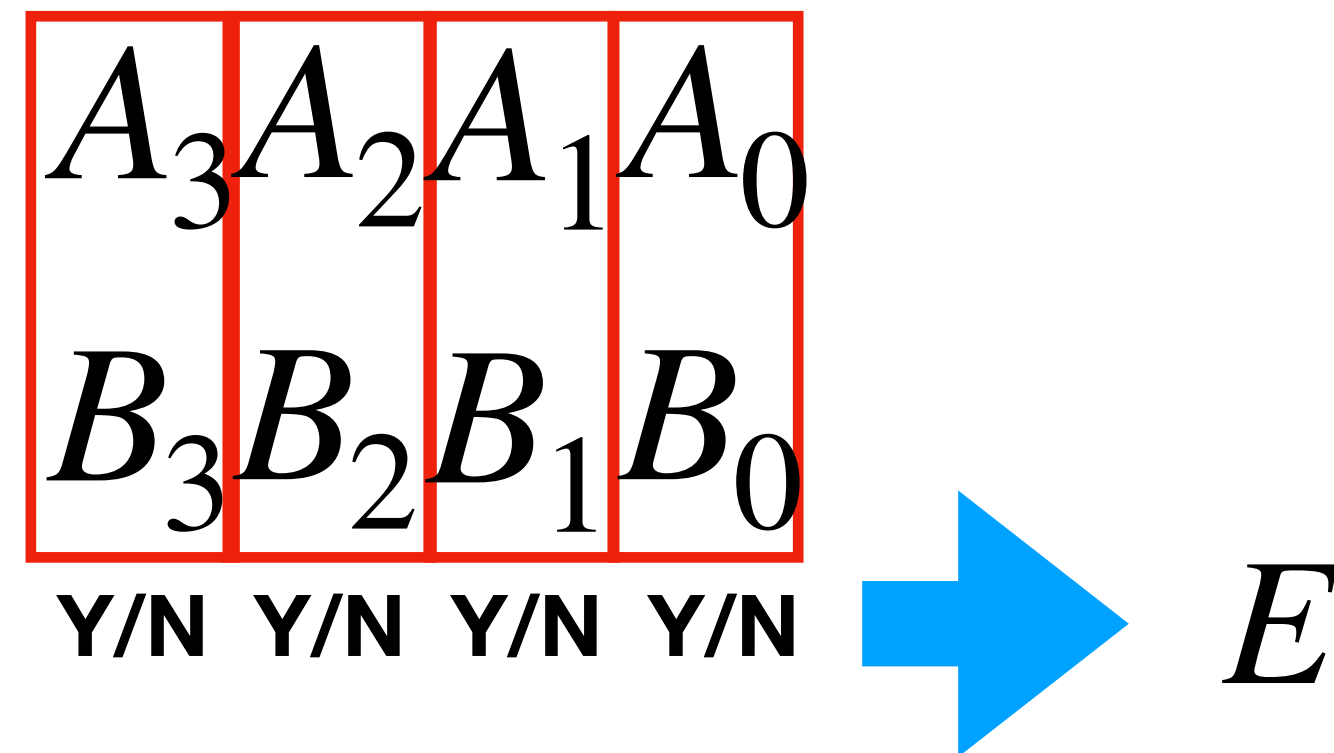


- 2 things are happening
  - We need to compare pairs of binary values and see if they are equal
  - We need to combine the comparison of all these different bits to make up  $E$

# Hierarchical Design

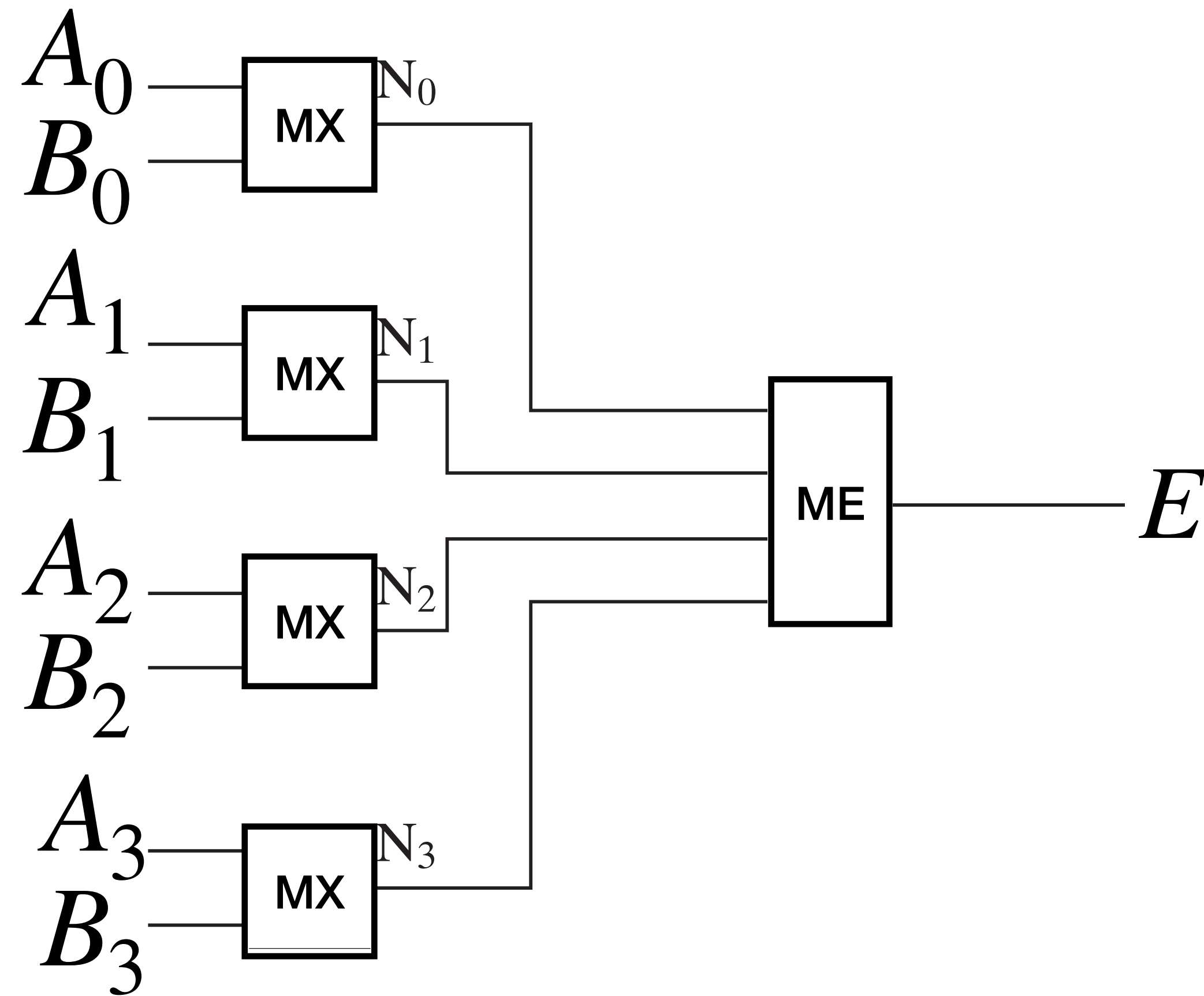
- "divide-and-conquer"
- Circuit is broken up into individual functional pieces (blocks)
  - Each block has explicitly defined **Interface** (I/O) and **Behaviour**
  - A single block can be **reused** multiple times to simplify design process
  - If a single block is too complex, it can be **further divided into smaller blocks**, to allow for easier designs

# 3. Optimisation

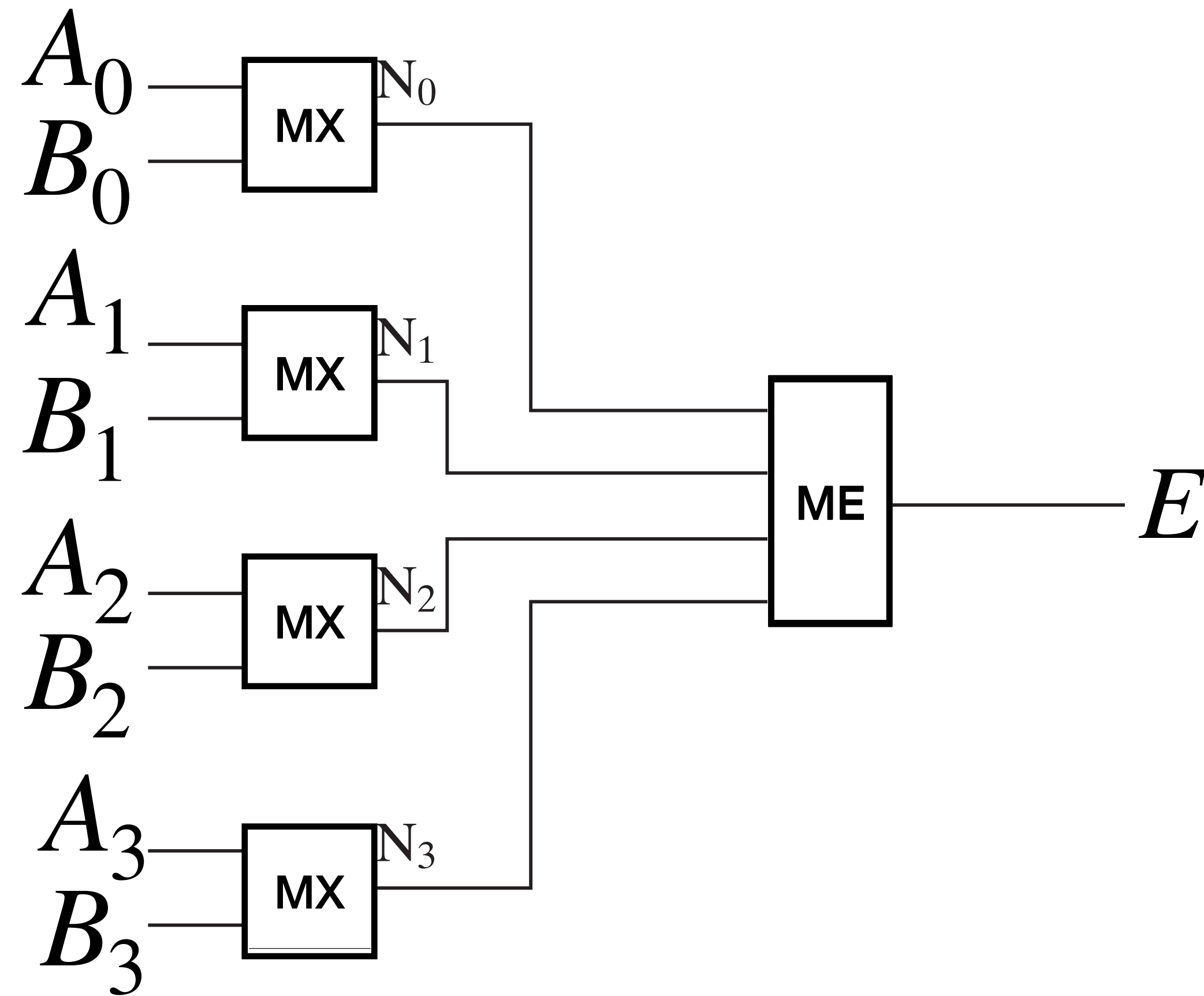


- 2 things are happening
  - We need to compare pairs of binary values and see if they are equal MX
  - We need to combine the comparison of all these different bits to make up  $E$  ME

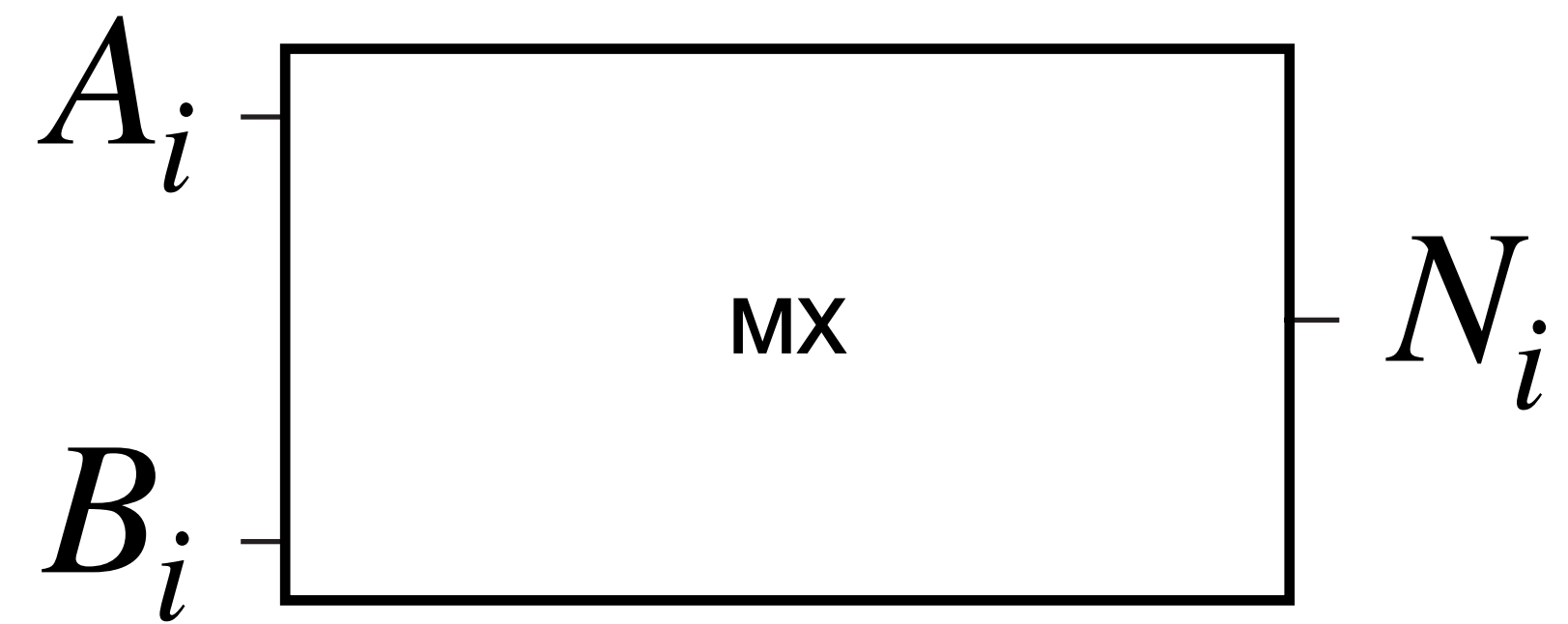
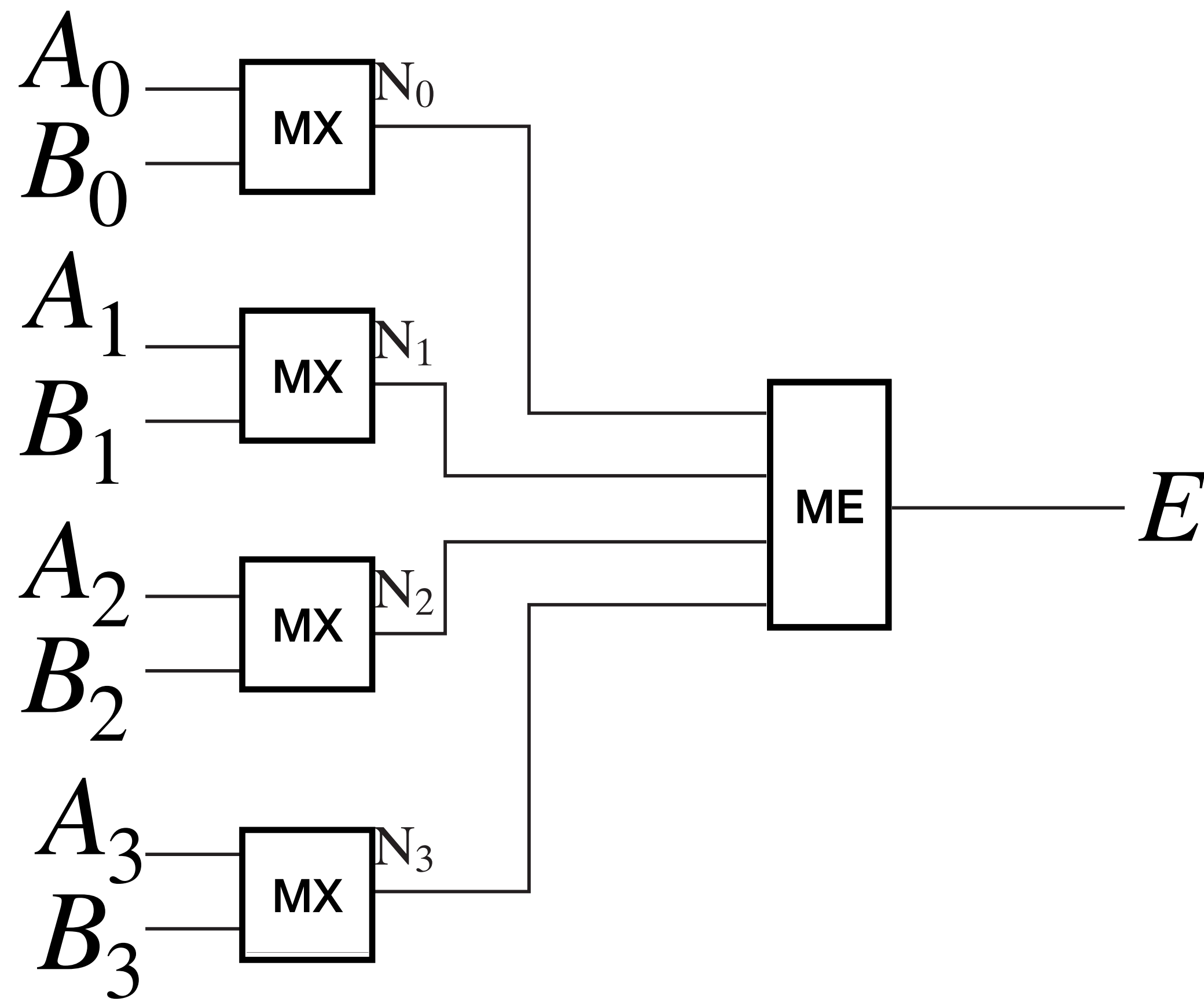
# 3. Optimisation



# 3. Optimisation



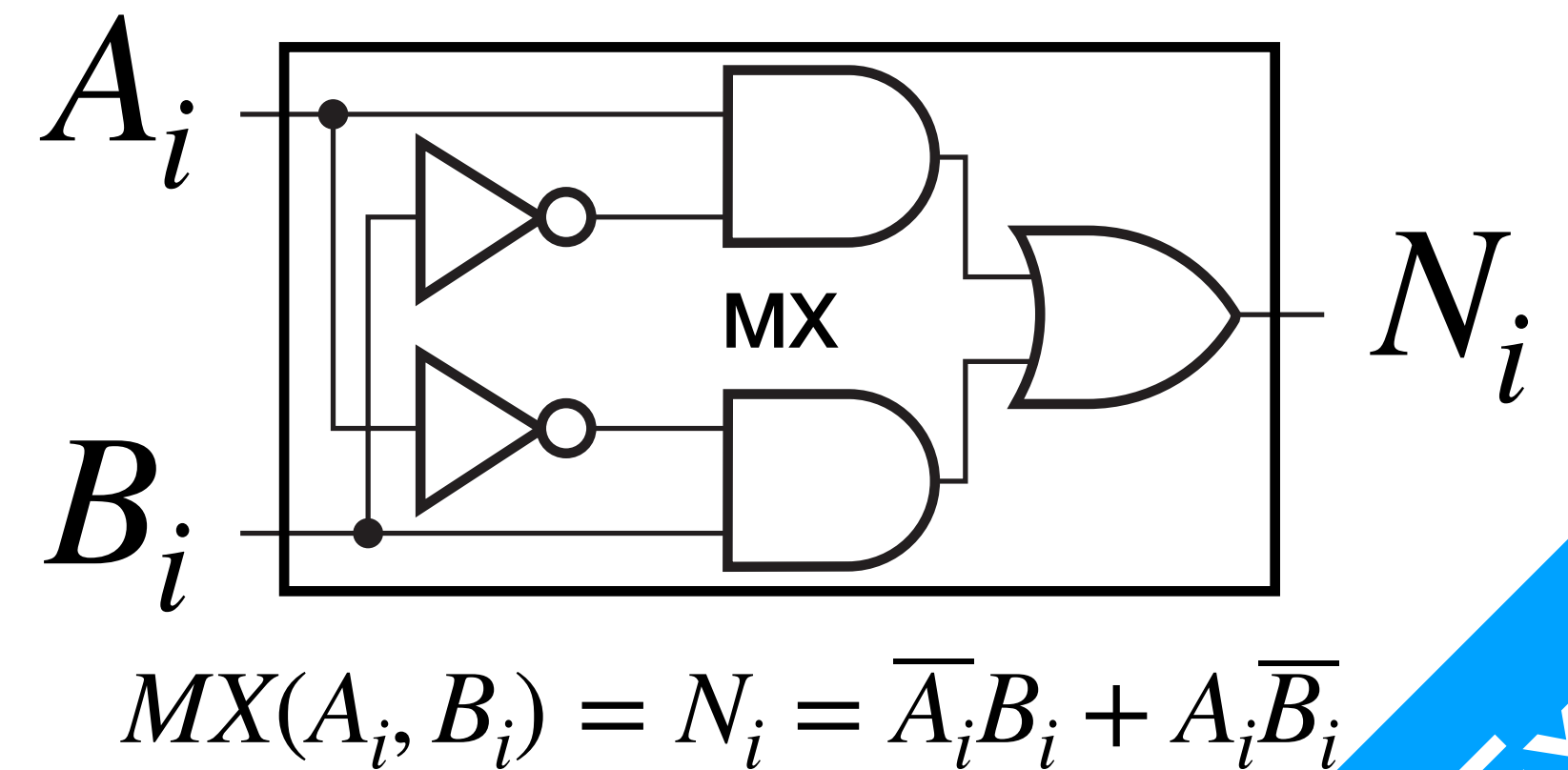
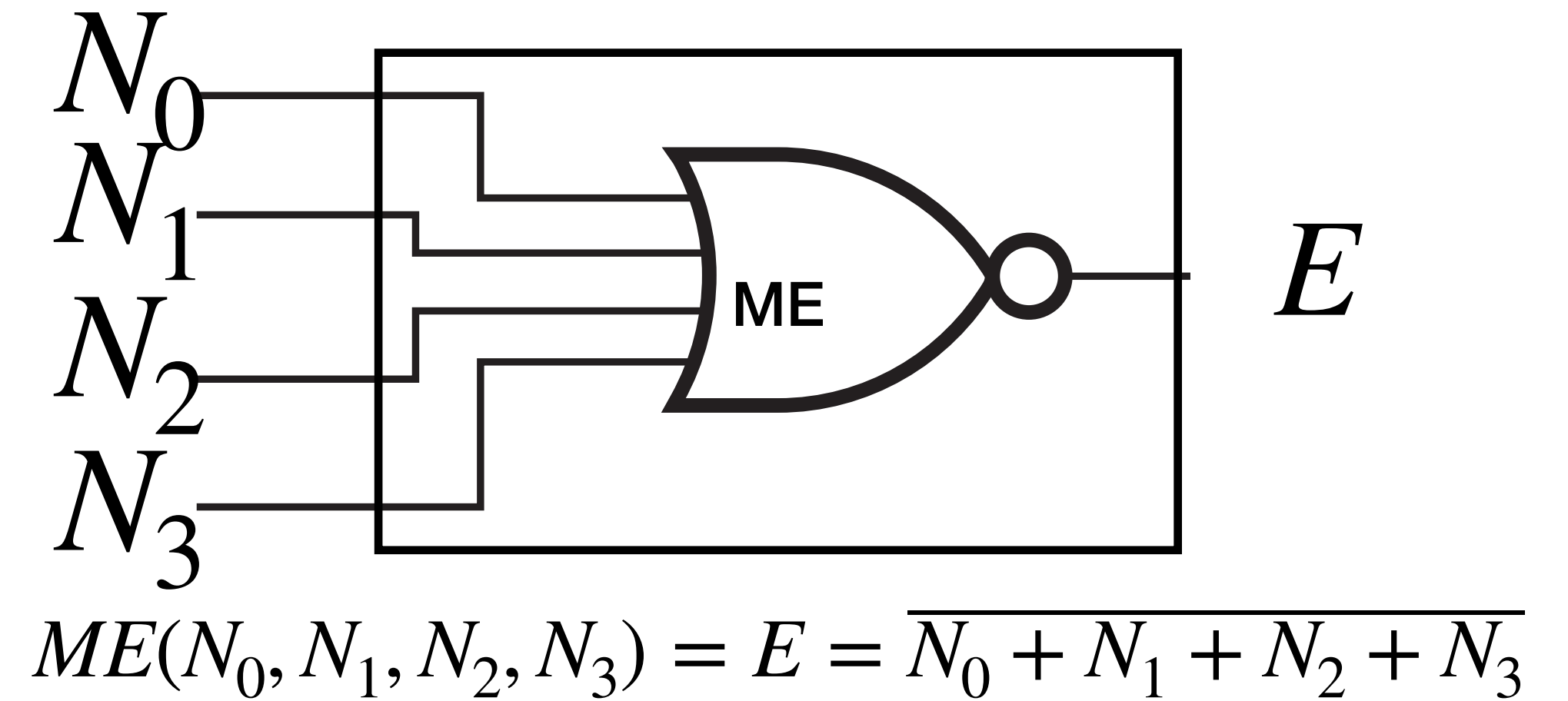
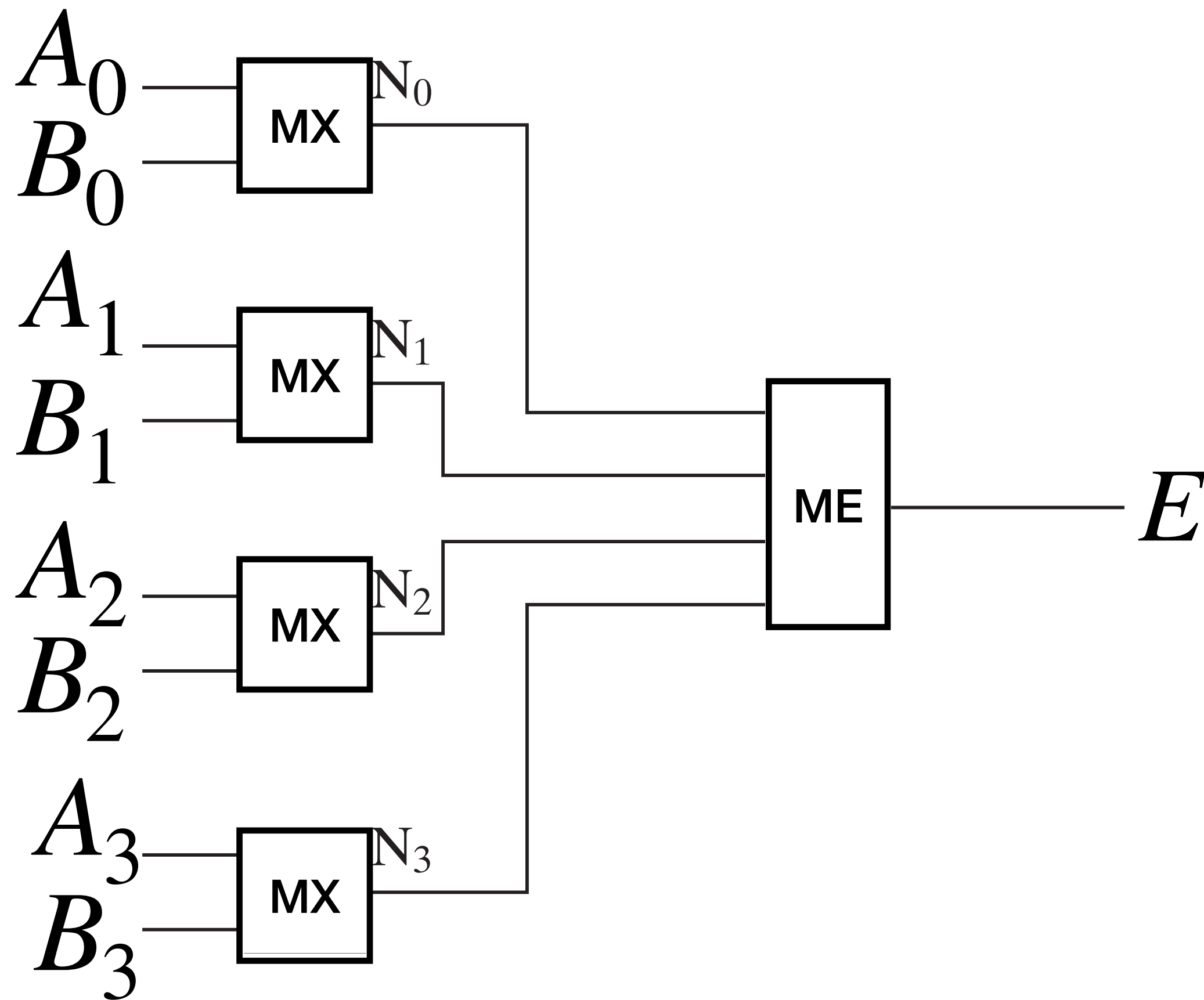
# 3. Optimisation



$$MX(A_i, B_i) = N_i = 0 \text{ if } A_i = B_i$$

$$MX(A_i, B_i) = N_i = \bar{A}_i B_i + A_i \bar{B}_i$$

# 3. Optimisation





# Technology Mapping

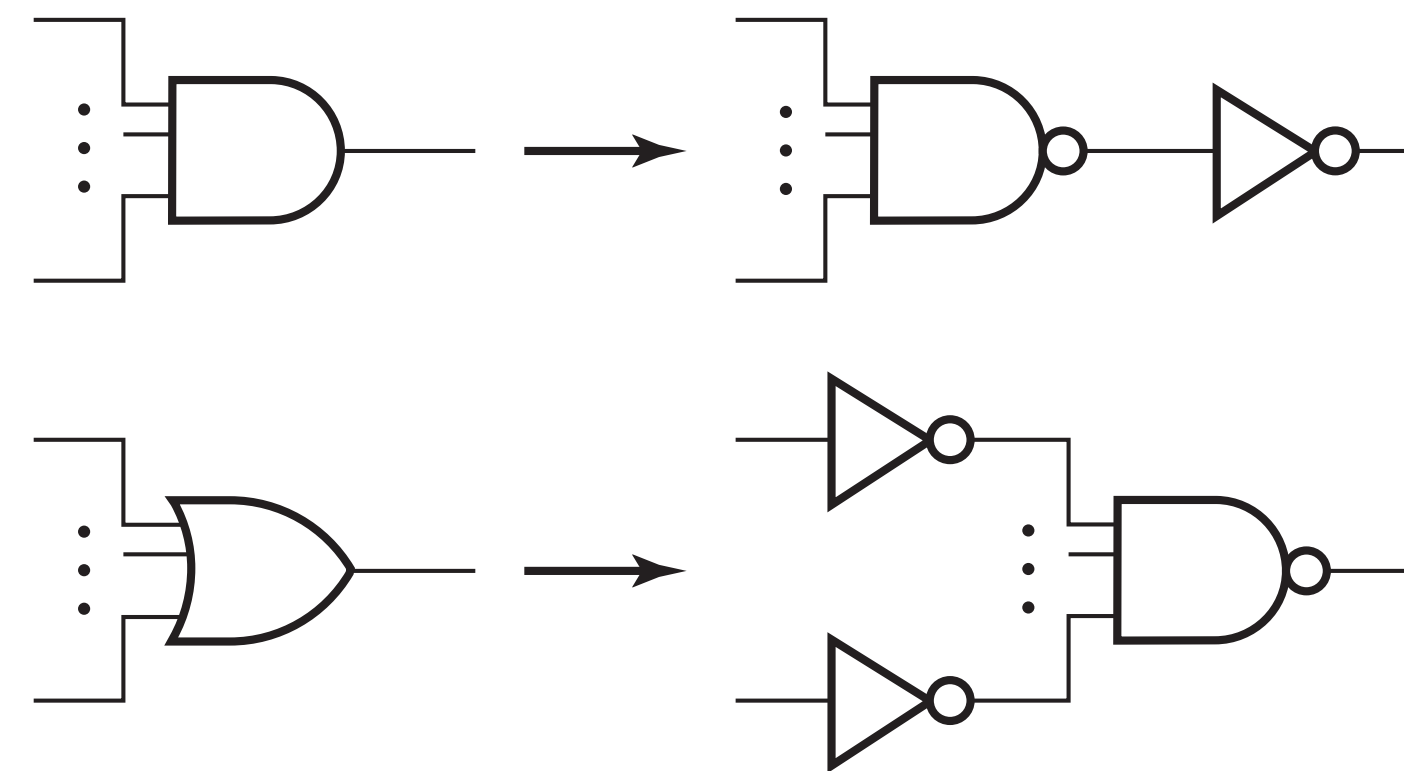
Mostly Old Tech, no new iPhones

# Technology Mapping

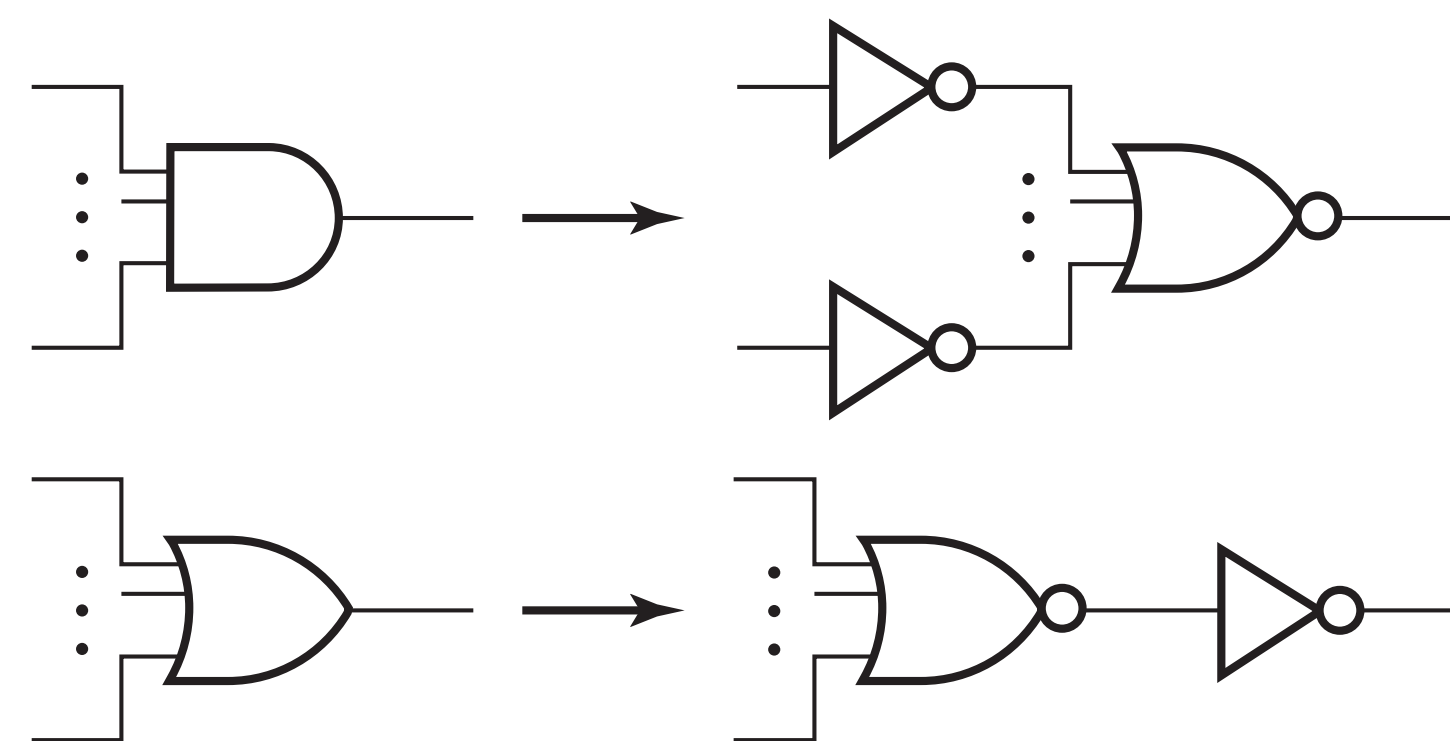
- Technology
  - Available physical components
  - Programmable implementation technology e.g. Field-Programmable Gate Array (FPGA)
  - VHDL; Verilog Language

# Technology Mapping

- $\text{NAND} == \text{AND} + \text{Inverter};$
- $\text{AND} == \text{NAND} + \text{Inverter};$
- $\text{NOR} == \text{OR} + \text{Inverter};$
- $\text{OR} == \text{NOR} + \text{Inverter};$
- DeMorgan's Rule



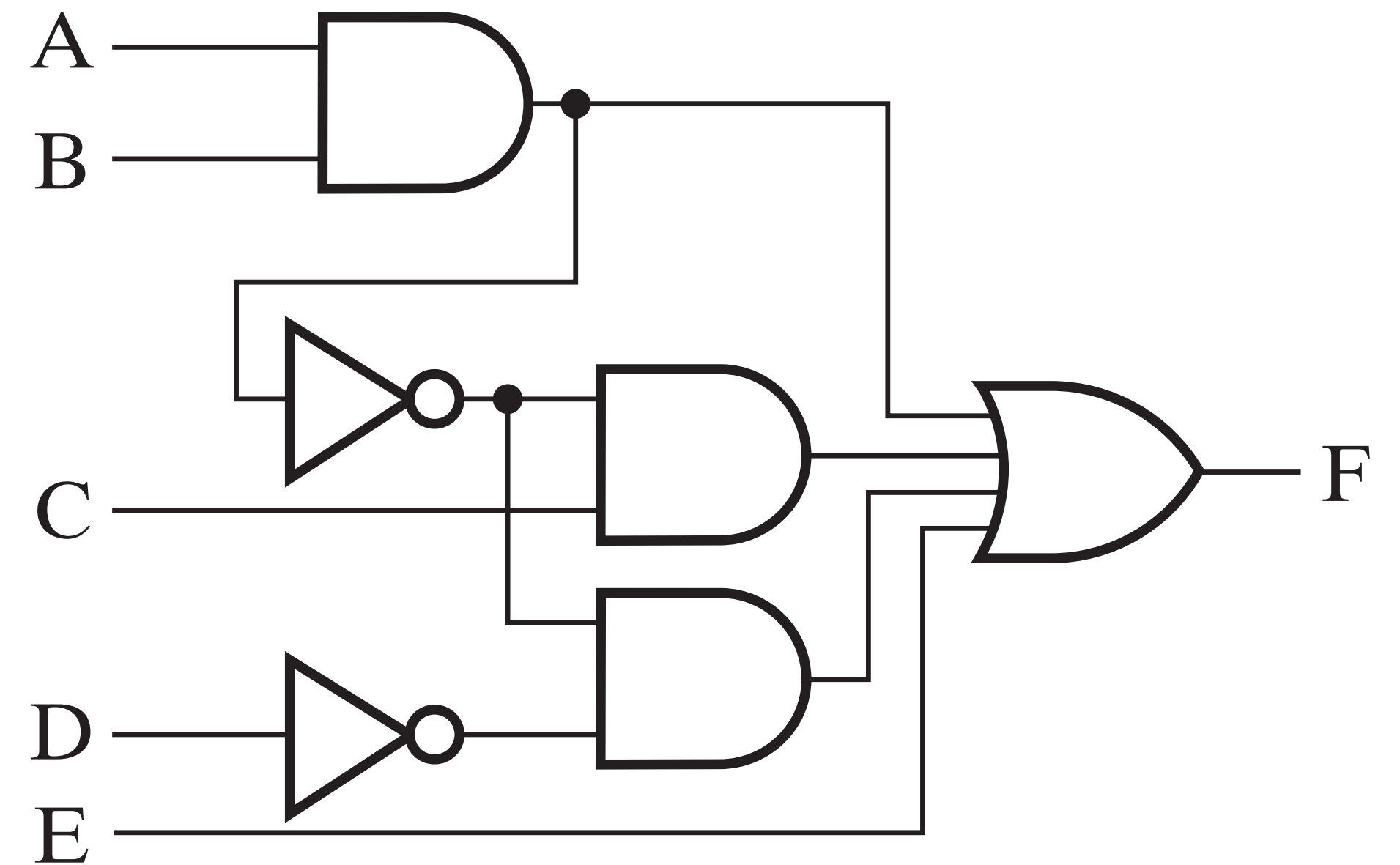
(a) Mapping to NAND gates



(b) Mapping to NOR gates

# Technology Mapping

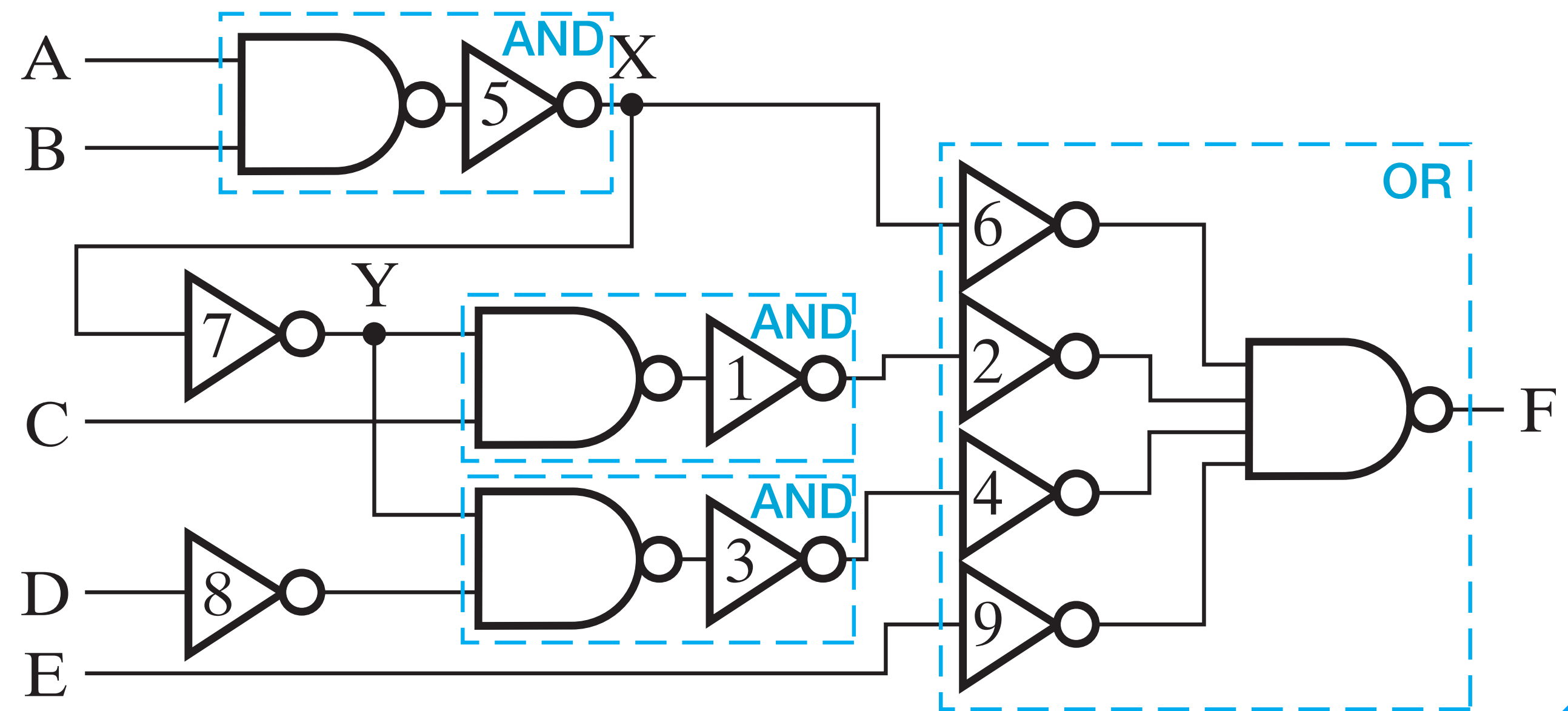
- Implement the circuit diagram with NAND gates and Inverters only



# Technology Mapping

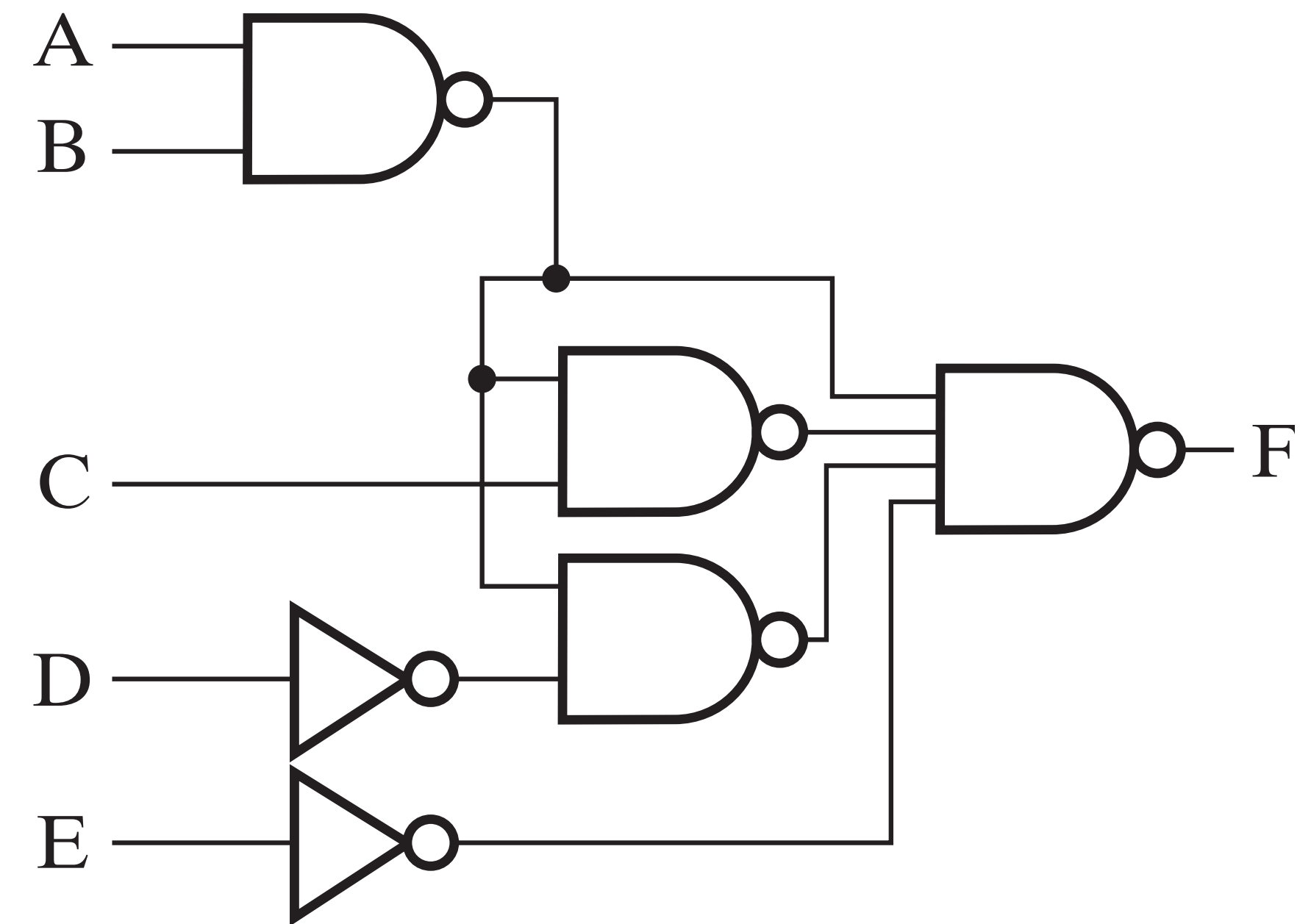
- Implement the circuit diagram with NAND gates and Inverters only

## 1. Replacement



# Technology Mapping

- Implement the circuit diagram with NAND gates and Inverters only
  1. Replacement
  2. Simplification



# Verification

Manual vs Auto

# Verification

- Manual
  - Use the truth table, row by row
- Auto
  - Use computer simulation to go through the truth table