04.02.20 14:27

CSCI 150
Introduction to Digital and Computer System Design
Lecture 3: Combinational Logic Design I

Jetic Gū

2020 Winter Semester (S1)

Overview

- Focus: Methodology
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.1; v5: Ch3 3.1
- Core Ideas:

    1. Light Reading followed by My Lunch Break

P0
Review

Lecture 1 & 2

- Lecture 1
  - What is digital (logic) circuit
  - How information is represented in digital (logic) circuit
- Lecture 2
  - Atomic components of digital (logic) circuit: Gates, I/O
  - Boolean Algebra

Review

Overview

- Focus: **Methodology**
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.1; v5: Ch3 3.1
- Core Ideas:
  1. How to **systematically use** what you've learned in Lecture 2
  2. **practice practice practice**

Overview

- Focus: **Methodology**
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.1; v5: Ch3 3.1
- Core Ideas:

  1. How to **systematically use** what you've learned in Lecture 2
  2. **practice practice practice**

Overview

- Focus: **Methodology**
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.1; v5: Ch3 3.1
- Core Ideas:

  1. How to **systematically use** what you've learned in Lecture 2
  2. **practice practice practice**

Overview

- Focus: **Methodology**
- Architecture: Combinatory Logical Circuits
- Textbook v4: Ch3 3.1; v5: Ch3 3.1
- Core Ideas:
  1. How to **systematically use** what you've learned in Lecture 2
  2. **practice practice practice**

Design Procedure

You, design this!

Summary

P1
Design Procedure

Know the Problem

Review

P1
Design Procedure

Know the Problem

1. Specification of the Problem

Review

P1
Design Procedure

Know the Problem

1. Specification of the Problem

2. Basic Design, a functional Beta Implementation

Review

P1
Design Procedure

Know the Problem

1. Specification of the Problem

2. Basic Design, a functional Beta Implementation

3. Optimisation, Optimisation
   Harder, Better, Faster, Stronger

Review

P1
Design Procedure

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

Concept

P1
Design Procedure

Systematic Design Procedures (Universal)

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

Concept

P1
Design Procedure

Systematic Design Procedures (Universal)

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

Concept

P1
Design Procedure

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

Concept

P1
Design Procedure

Summary

Review

Summary

- Boolean Algebra III: K-Map

Review

Summary

- Boolean Algebra III: K-Map
  - Two Variable K-Map

Review

Summary

- Boolean Algebra III: K-Map
  - Two Variable K-Map
  - Three Variable K-Map

Review

P1
Design Procedure

Summary

- Boolean Algebra III: K-Map
  - Two Variable K-Map
  - Three Variable K-Map
  - Four Variable K-Map

Review

Summary

- Boolean Algebra III: K-Map
  - Two Variable K-Map
  - Three Variable K-Map
  - Four Variable K-Map
  - Don’t care optimisation

Review

An Old Friend

Curtain Motors Revisited

Summary

P2
An Example

An Old Friend

- Curtain Motor Control
  - **Sensor1: 1 when curtain is fully closed**
  - **Sensor2: 1 when curtain is fully open**
  - Button1: 1 when user wants to open the curtain
  - Button2: 1 when user wants to close the curtain
  - Output1: 1 to make the motor open the window
  - Output2: 1 to make the motor close the window
  - Light: motor is active

- Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
1. Specification

- Curtain Motor Control
  - **Sensor1: 1 when curtain is fully closed**
  - **Sensor2: 1 when curtain is fully open**
  - Button1: 1 when user wants to open the curtain
  - Button2: 1 when user wants to close the curtain
  - Output1: 1 to make the motor open the window
  - Output2: 1 to make the motor close the window
  - Light: motor is active

- Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
1. Specification

- Curtain Motor Control

Input	• **Sensor1: 1 when curtain is fully closed**
Input	• **Sensor2: 1 when curtain is fully open**
Input	• Button1: 1 when user wants to open the curtain
Input	• Button2: 1 when user wants to close the curtain
Output	• Output1: 1 to make the motor open the window
Output	• Output2: 1 to make the motor close the window
Output	• Light: motor is active

- Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
1. Specification

- Curtain Motor Control

Input	• **Sensor1: 1 when curtain is fully closed**
Input	• **Sensor2: 1 when curtain is fully open**
Input	• Button1: 1 when user wants to open the curtain
Input	• Button2: 1 when user wants to close the curtain
Output	• Output1: 1 to make the motor open the window
Output	• Output2: 1 to make the motor close the window
Output	• Light: motor is active

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
1. Specification

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to open the curtain

Switch • Button2: 1 when user wants to close the curtain

Prob • Output1: 1 to make the motor open the window

Prob • Output2: 1 to make the motor close the window

Prob • Light: motor is active

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\begin{aligned} \text{FullyClosed} &= \text{Sensor1} \\ \text{FullyOpened} &= \text{Sensor2} \\ \text{Light} &= \text{Out1} + \text{Out2} \end{aligned}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\begin{aligned} \text{FullyClosed} &= \text{Sensor1} \\ \text{FullyOpened} &= \text{Sensor2} \\ \text{Light} &= \text{Out1} + \text{Out2} \end{aligned}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{FullyOpened}}$$
$$\text{Out2} = \text{Bu2} \cdot \overline{\text{FullyClosed}}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{FullyOpened}}$$
$$\text{Out2} = \text{Bu2} \cdot \overline{\text{FullyClosed}}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{FullyOpened}} \cdot \overline{\text{BothPressed}}$$
$$\text{Out2} = \text{Bu2} \cdot \overline{\text{FullyClosed}} \cdot \overline{\text{BothPressed}}$$

$$\text{BothPressed} = \text{Bu1} \cdot \text{Bu2}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

P2
An Example

An Old Friend
2. Formulation

- Curtain Motor Control

Switch • **Sensor1: 1 when curtain is fully closed**

Switch • **Sensor2: 1 when curtain is fully open**

Switch • Button1: 1 when user wants to **open** the curtain

Switch • Button2: 1 when user wants to **close** the curtain

Prob • Output1: 1 to make the motor **open** the window

Prob • Output2: 1 to make the motor **close** the window

Prob • Light: motor is active

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{FullyOpened}} \cdot \overline{\text{BothPressed}}$$
$$\text{Out2} = \text{Bu2} \cdot \overline{\text{FullyClosed}} \cdot \overline{\text{BothPressed}}$$

$$\text{BothPressed} = \text{Bu1} \cdot \text{Bu2}$$

Cond Stop the motor when the curtain is already fully opened/ closed

Demo

An Old Friend

3. Optimisation

$$\text{FullyClosed} = \text{Sensor1}$$
$$\text{FullyOpened} = \text{Sensor2}$$
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{FullyOpened}} \cdot \overline{\text{BothPressed}}$$
$$\text{Out2} = \text{Bu2} \cdot \overline{\text{FullyClosed}} \cdot \overline{\text{BothPressed}}$$

$$\text{BothPressed} = \text{Bu1} \cdot \text{Bu2}$$

$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$
$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

Demo

An Old Friend

4. Technology Mapping

$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

Demo

An Old Friend
4. Technology Mapping

$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

- Available Components (Technology)

Demo

An Old Friend
4. Technology Mapping

$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

- Available Components (Technology)
  - AND, NAND, OR, NOT gate; Switch, Prob

Demo

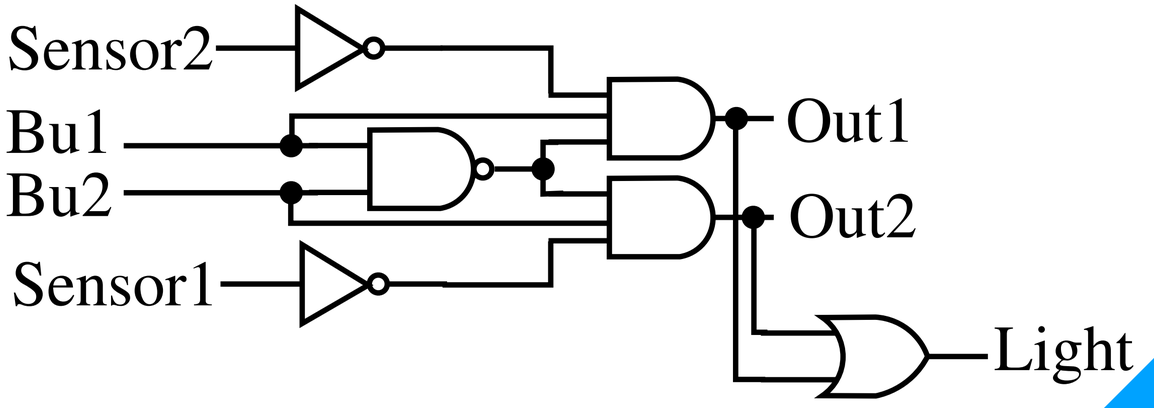
An Old Friend
4. Technology Mapping

$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

- Available Components (Technology)
  - AND, NAND, OR, NOT gate; Switch, Prob



Demo

An Old Friend
4. Technology Mapping

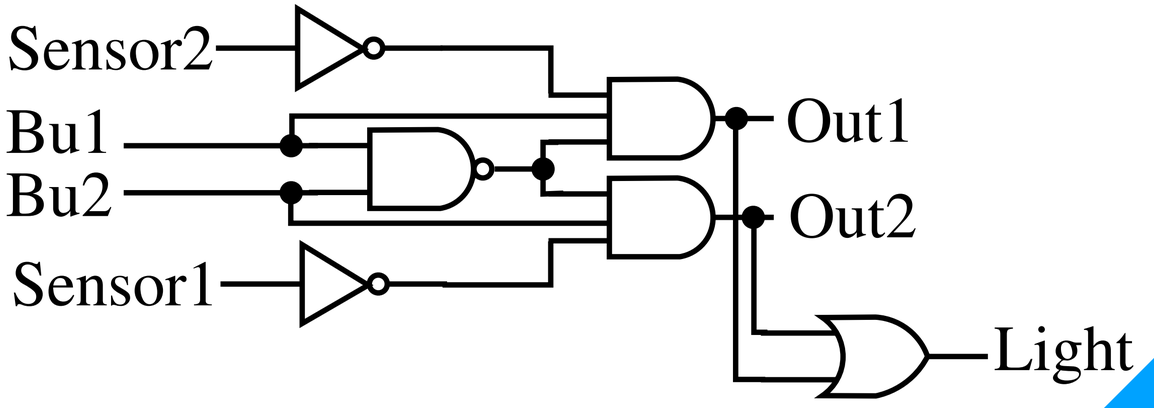
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

- Available Components (Technology)
  - AND, NAND, OR gate; Switch, Prob

NO NOT GATE!



Demo

An Old Friend
4. Technology Mapping

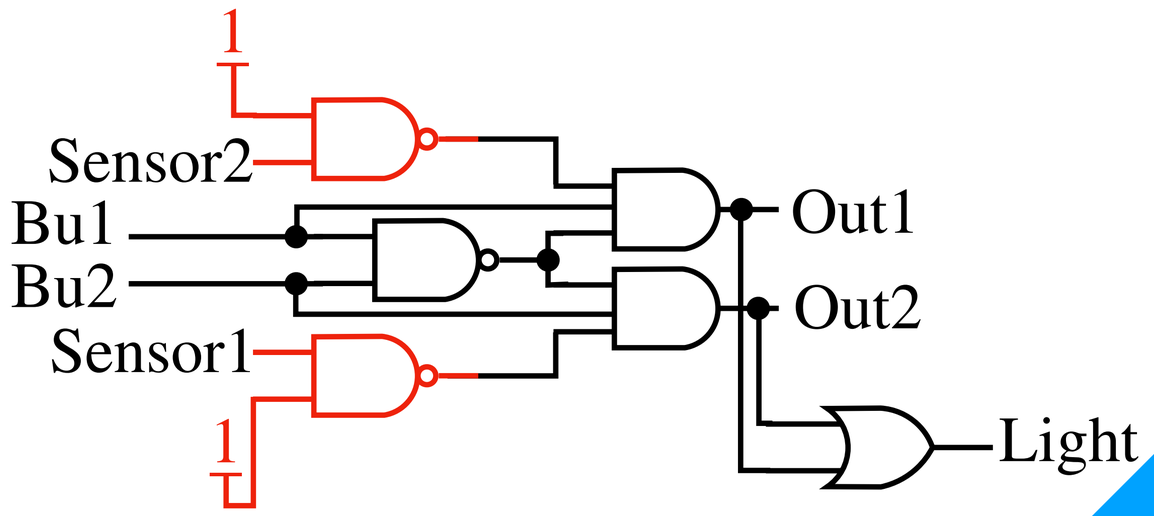
$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

- Available Components (Technology)
  - AND, NAND, OR gate; Switch, Prob

NO NOT GATE!



Demo

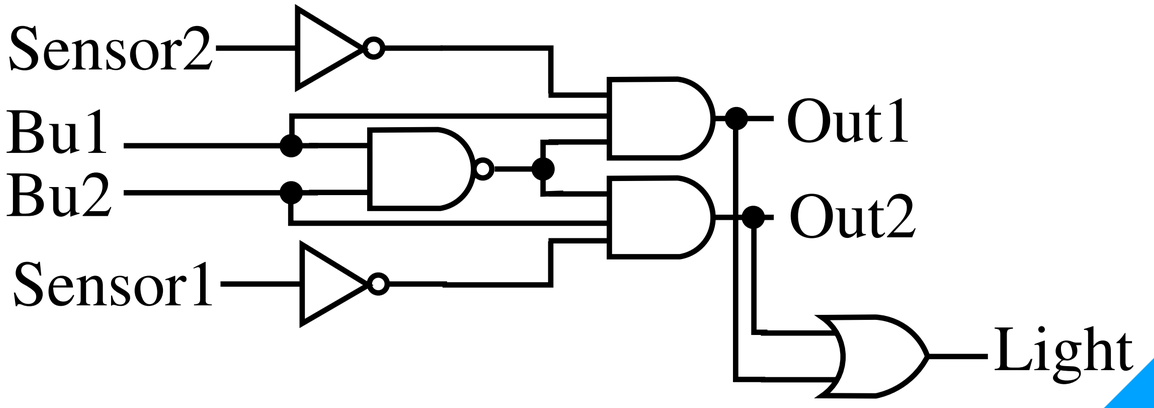
An Old Friend
4. Technology Mapping

$$\text{Light} = \text{Out1} + \text{Out2}$$

$$\text{Out1} = \text{Bu1} \cdot \overline{\text{Sensor2}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

$$\text{Out2} = \text{Bu2} \cdot \overline{\text{Sensor1}} \cdot \overline{\text{Bu1} \cdot \text{Bu2}}$$

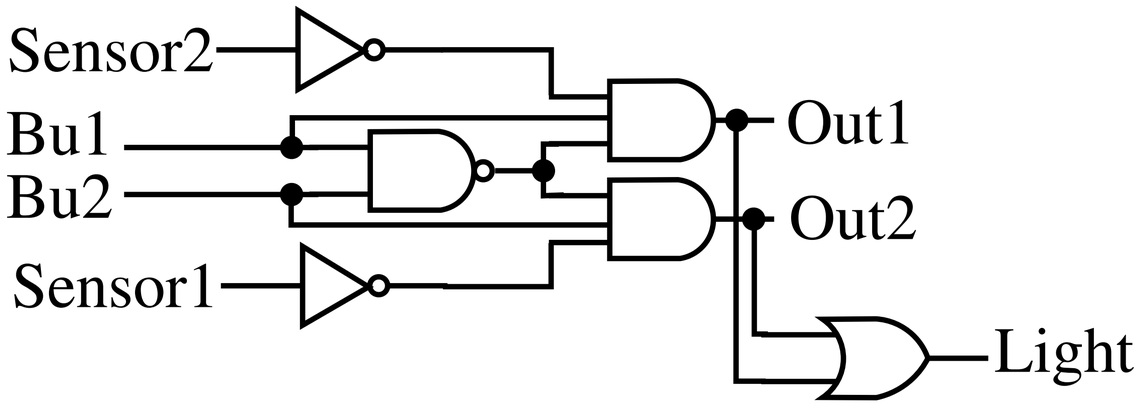
- Available Components (Technology)
  - AND, NAND, OR, NOT gate; Switch, Prob



Demo

An Old Friend

5. Verification



- Design test cases
  - Sensor2, Bu1 on
  - Sensor1, Bu2 on
  - Bu1, Bu2 on
  - Bu1 on
  - Bu2 on
  - etc…

Demo