





CSE 390B, Autumn 2022

Building Academic Success Through Bottom-Up Computing

Time Management & Decisions in Hardware

Time Management, Boolean Logic and Functions Review,
Implementing an **Xor** gate in HDL, Decisions in Hardware

Connect With Your CSE 390B Peers

- ❖ Download Discord from <https://discord.com/download>
- ❖ Log in or create an account
- ❖ Click on  icon in left-most column to create channel
 - Select “Create My Own”
 - Select “For me and my friends”
 - Give your server a name! (e.g., “CSE 390B 22au”)
- ❖ Use the  button to invite peers
- ❖ Create some text and voice channels. Some ideas:
 - Text: #projects, #questions, #chill, #random
 - Voice:  Study Room,  Lounge
- ❖ Feel free to connect via Slack, Messenger, etc. too

Lecture Outline

❖ Time Management

- Identifying Weekly Time Commitments

❖ Review of Boolean Logic and Functions

- Boolean Expressions, Circuit Diagrams, Truth Tables
- Boolean Function Synthesis Strategy

❖ Implementing an **Xor** gate in HDL

- Applying the Boolean Function Synthesis Strategy

❖ Making Decisions in Hardware

- Multiplexer and Demultiplexer Logical Gate

Time Management

- ❖ One of your most valuable resources in college is time

- ❖ What typically fills up your time during the quarter?
 - Lectures, quiz sections, and attending office hours
 - Part-time jobs and working
 - Studying
 - Extracurricular activities or RSOs
 - Commuting
 - Chores at home
 - Socializing with friends and family
 - Physical, mental, spiritual activities

Weekly Time Commitments

- ❖ Class meeting times and quiz sections
- ❖ Family, friends, community, extracurricular commitments
- ❖ Physical, mental, social, spiritual activities
- ❖ Studying for each of your classes
 - The number of credits for a course reflects the number of hours the class meets
 - In general, courses require two hours of homework for every one hour of class
- ❖ What else is not reflected given your specific situation?

Tracking Weekly Time Commitments

- ❖ We often don't realize what takes up our time until we manually track how we use our time
- ❖ Exercise: Complete the weekly time commitments table
- ❖ Tip: Use different colors for different activity types

Weekly Time Commitments Table

Name: _____

Time	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
7:30am							
8:00am							
8:30am							
9:00am							
9:30am							
10:00am							
10:30am							
11:00am							
11:30am							
12:00pm							
12:30pm							
1:00pm							
1:30pm							
2:00pm							
2:30pm							
3:00pm							
3:30pm							
4:00pm							
4:30pm							
5:00pm							
5:30pm							
6:00pm							
6:30pm							
7:00pm							
7:30pm							
8:00pm							
8:30pm							
9:00pm							
9:30pm							
10:00pm							
10:30pm							
11:00pm							
11:30pm							

Time Management Group Discussion

Now that you've filled out your weekly time commitments, discuss the following in groups for 4-6 minutes:

- ❖ Did anything surprise you about the way you use your time?
- ❖ What might you change about the way you utilize your time?
- ❖ How could you use this time commitments sheet in the future?

Lecture Outline

❖ Time Management

- Identifying Weekly Time Commitments

❖ Review of Boolean Logic and Functions

- Boolean Expressions, Circuit Diagrams, Truth Tables
- Boolean Function Synthesis Strategy

❖ Implementing an **Xor** gate in HDL

- Applying the Boolean Function Synthesis Strategy

❖ Making Decisions in Hardware

- Multiplexer and Demultiplexer Logical Gate

Boolean Functions

❖ Combinations of Boolean inputs resulting in single output

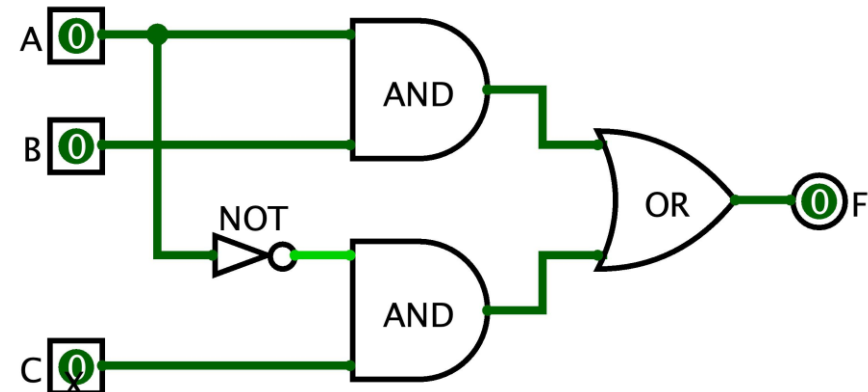
❖ Multiple ways to specify a Boolean function:

- Boolean expression: $F = (A \text{ AND } B) \text{ OR } (\text{NOT}(A) \text{ AND } C)$

- Circuit diagram with logic gates:

- Truth table:

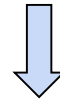
A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



Boolean Expression \rightarrow Truth Table

- ❖ We can build a truth table from an expression
 - Evaluate the Boolean expression on all possible inputs

$$F(A, B, C) = (A \text{ AND } B) \text{ OR } (\text{NOT}(A) \text{ AND } C)$$

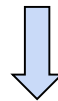


A	B	C	F
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Boolean Expression \rightarrow Truth Table

- ❖ We can build a truth table from an expression
 - Evaluate the Boolean expression on all possible inputs

$$F(A, B, C) = (A \text{ AND } B) \text{ OR } (\text{NOT}(A) \text{ AND } C)$$

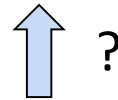


A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Boolean Expression \leftarrow Truth Table

❖ But can we do it in reverse?

$$F(A, B, C) = (A \text{ AND } B) \text{ OR } (\text{NOT}(A) \text{ AND } C)$$



A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Boolean Expression \leftarrow Truth Table

- ❖ Describe rows with an output of 1 using AND and NOT
- ❖ Then, we can combine the rows using OR operations

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

NOT(A) AND NOT(B) AND C

NOT(A) AND B AND C

A AND B AND NOT C

A AND B AND C

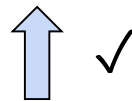
$$F = \text{NOT(A) AND NOT(B) AND C OR NOT(A) AND B AND C OR} \\ \text{A AND B AND NOT C OR A AND B AND C}$$

Boolean Expression \leftarrow Truth Table

❖ But can we do it in reverse?

- Yes, we can! The strategy we used is **Boolean Function Synthesis**

$$F(A, B, C) = (A \text{ AND } B) \text{ OR } (\text{NOT}(A) \text{ AND } C)$$



A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Lecture Outline

- ❖ Time Management
 - Identifying Weekly Time Commitments
- ❖ Review of Boolean Logic and Functions
 - Boolean Expressions, Circuit Diagrams, Truth Tables
 - Boolean Function Synthesis Strategy
- ❖ **Implementing an Xor gate in HDL**
 - **Applying the Boolean Function Synthesis Strategy**
- ❖ Making Decisions in Hardware
 - Multiplexer and Demultiplexer Logical Gate

Project 2 Overview

❖ Metacognitive Components

- Part I: Study Skills Inventory — self-assessing your skill level in various study practices and habits
- Part III: Project 2 Reflection — reflect on your experience working through the project

❖ Technical Component

- Part II: Boolean Logic — implement foundational Boolean logic gates (e.g., **Not**, **And**, **Or**, etc. gates)
- Clone your GitLab repository, write and test your code

❖ Project 2 due next Thursday (10/13) at 11:59pm

The Foundational Building Block

- ❖ It all starts with the NAND gate
- ❖ NAND is short for “Not And”
 - The same output as the AND gate, but every output bit is negated (flipped)

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

$$F = A \text{ AND } B$$

A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

$$F = A \text{ NAND } B$$

Building Gates From Nand

- ❖ Recall the **Boolean Function Synthesis** strategy
 - We saw how we can represent any truth table in terms of three gates: **Not**, **And**, **Or**
- ❖ First, we can represent **Not** directly from **Nand**
 - $\text{Not } a = a \text{ Nand } a$
- ❖ Then, we can represent **And** in terms of **Not** and **Nand**
 - $a \text{ And } b = \text{Not}(a \text{ Nand } b)$
- ❖ Represent **Or** in terms of **Not** and **And**
 - Apply De Morgan's Law
 - $a \text{ Or } b = \text{Not}(\text{Not}(a) \text{ And } \text{Not}(b))$ [De Morgan's Law]

Implementing an **Xor** gate: Overview

- ❖ Let's walk through an example of building a gate that you will work on in Project 2, the **Xor** gate
- ❖ Together, we'll implement the **Xor** gate

```
/**  
 * Xor gate:  
 * out = not(a == b)  
 */  
CHIP Xor {  
    IN a, b;  
    OUT out;  
  
    PARTS:  
    // Put your code here:  
}
```

Implementing an Xor gate: Overview

❖ Plan of action:

- Step 1: Create the logic operation's **truth table**
- Step 2: Use truth table to generate a **Boolean function** using strategies we've learned, such as the Boolean Function Synthesis
- Step 3: Convert Boolean function to **HDL**

```
/**  
 * Xor gate:  
 * out = not(a == b)  
 */  
CHIP Xor {  
    IN a, b;  
    OUT out;  
  
    PARTS:  
    // Put your code here:  
}
```

Implementing an Xor gate: Step 1

❖ Step 1: Create the truth table for Xor

- Interpret the specification: $F = \text{NOT}(A == B)$

A	B	F
0	0	
0	1	
1	0	
1	1	

$$F = A \text{ XOR } B$$

Implementing an Xor gate: Step 1

❖ Step 1: Create the truth table for Xor

- Interpret the specification: $F = \text{NOT}(A == B)$

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

$$F = A \text{ XOR } B$$

Implementing an Xor gate: Step 2

- ❖ Step 2: Use truth table to generate a Boolean function
 - Let's use the **Boolean Function Synthesis** strategy
 - **Row 2 = NOT(A) AND B**

A	B	F	
0	0	0	(Row 1)
0	1	1	(Row 2)
1	0	1	(Row 3)
1	1	0	(Row 4)

$$F = A \text{ XOR } B$$

Implementing an Xor gate: Step 2

- ❖ Step 2: Use truth table to generate a Boolean function
 - Let's use the **Boolean Function Synthesis** strategy
 - Row 2 = NOT(A) AND B
 - Row 3 = A AND NOT(B)

A	B	F	
0	0	0	(Row 1)
0	1	1	(Row 2)
1	0	1	(Row 3)
1	1	0	(Row 4)

$$F = A \text{ XOR } B$$

Implementing an Xor gate: Step 2

- ❖ Step 2: Use truth table to generate a Boolean function
 - Let's use the **Boolean Function Synthesis** strategy
 - Row 2 = NOT(A) AND B
 - Row 3 = A AND NOT(B)
 - $F = ?$

A	B	F	
0	0	0	(Row 1)
0	1	1	(Row 2)
1	0	1	(Row 3)
1	1	0	(Row 4)

$$F = A \text{ XOR } B$$



Vote at <https://pollev.com/cse390b>

What is the unsimplified Boolean expression result from performing Boolean function synthesis on $F = A \text{ XOR } B$?

- A. $(A \text{ AND NOT}(B)) \text{ AND } (\text{NOT}(A) \text{ AND } B)$
- B. $(\text{NOT}(A) \text{ AND } B) \text{ AND } (A \text{ AND } B)$
- C. $(A \text{ AND } B) \text{ OR } (\text{NOT}(A) \text{ AND NOT}(B))$
- D. $(\text{NOT}(A) \text{ AND } B) \text{ OR } (A \text{ AND NOT}(B))$
- E. We're lost...

- Row 2 = $\text{NOT}(A) \text{ AND } B$
- Row 3 = $A \text{ AND NOT}(B)$

A	B	F = A XOR B	
0	0	0	(Row 1)
0	1	1	(Row 2)
1	0	1	(Row 3)
1	1	0	(Row 4)

Implementing an Xor gate: Step 2

- ❖ Step 2: Use truth table to generate a Boolean function
 - Let's use the Boolean function synthesis strategy
 - Row 2 = NOT(A) AND B
 - Row 3 = A AND NOT(B)
 - $F = \text{Row 2 OR Row 3}$
 $= (\text{NOT}(A) \text{ AND } B) \text{ OR } (A \text{ AND NOT}(B))$

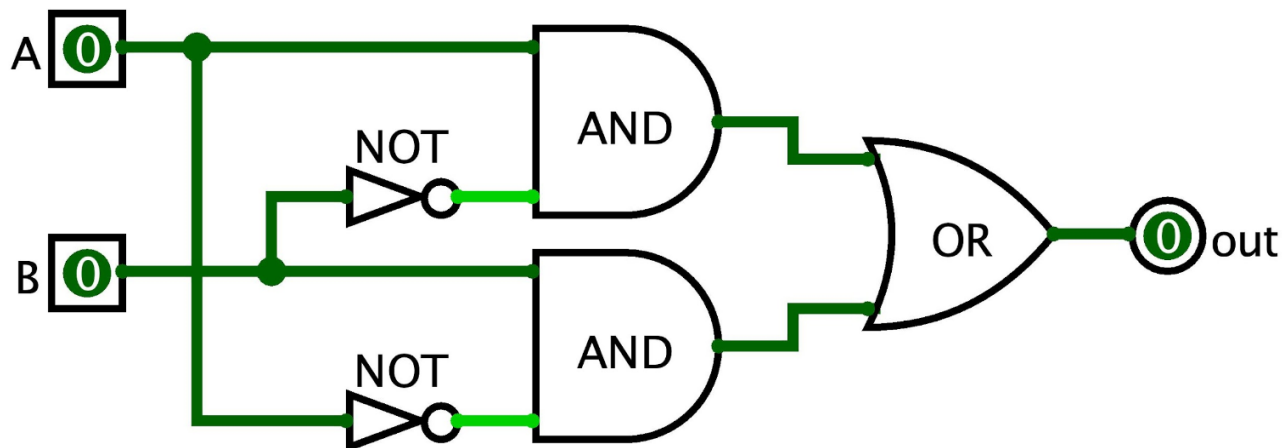
A	B	F	
0	0	0	(Row 1)
0	1	1	(Row 2)
1	0	1	(Row 3)
1	1	0	(Row 4)

$$F = A \text{ XOR } B$$

Implementing an Xor gate: Step 3

- ❖ Now that we have a Boolean expression, we can implement the **Xor** gate in HDL
- ❖ Optionally, it can help to express the Boolean expression as a circuit diagram

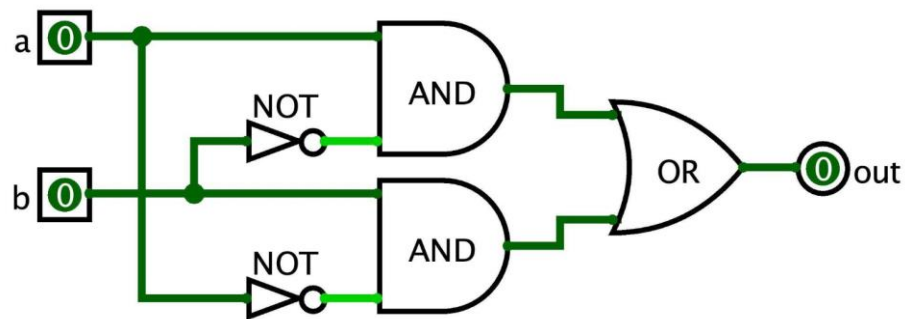
$$A \text{ XOR } B = (\text{NOT}(A) \text{ AND } B) \text{ OR } (A \text{ AND } \text{NOT}(B))$$



Implementing an Xor gate: Step 3

- ❖ Step 3: Convert Boolean function to HDL syntax
 - $A \text{ XOR } B = (\text{NOT}(A) \text{ AND } B) \text{ OR } (A \text{ AND } \text{NOT}(B))$
 - Assumes **Not**, **And**, and **Or** are already implemented
 - Note the use of intermediary wires: **nota**, **notb**, **x**, and **y**

```
CHIP Xor {  
  IN a, b;  
  OUT out;  
  
  PARTS:  
    Not (in=a, out=nota);  
    Not (in=b, out=notb);  
    And (a=a, b=notb, out=x);  
    And (a=nota, b=b, out=y);  
    Or (a=x, b=y, out=out);  
}
```



Lecture Outline

❖ Time Management

- Identifying Weekly Time Commitments

❖ Review of Boolean Logic and Functions

- Boolean Expressions, Circuit Diagrams, Truth Tables
- Boolean Function Synthesis Strategy

❖ Implementing an **Xor** gate in HDL

- Applying the Boolean Function Synthesis Strategy

❖ Making Decisions in Hardware

- Multiplexer and Demultiplexer Logical Gate

Making Decisions in Hardware

- ❖ We write if/else statements in Java with the understanding that *only one* of the branches will run
 - For example, in the following code, we expect to compute one of $a \ \& \ b$ or $a \ | \ b$ (not both)

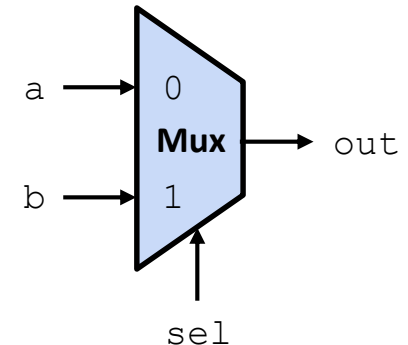
```
if (c == 0) {  
    out = a & b;  
} else {  
    out = a | b;  
}
```

- ❖ In hardware, the entire circuit is always executing
 - We can't "turn off" a part of a circuit based on a condition
 - Instead, we create circuits for different conditions and choose which output based on a condition instead

Decisions in Hardware: Mux Gate

❖ We can use a **Multiplexer (Mux)** gate to choose which singular input to output

- Takes three inputs: a , b , and sel
- If $sel == 0$, then $out = a$
- Otherwise, $out = b$



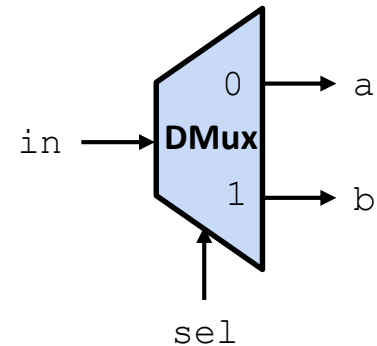
❖ Mux Gate Truth Table:

a	b	sel	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Decisions in Hardware: DMux Gate

❖ A **Demultiplexer (DMux)** gate passes one input to one of two outputs and 0 to the rest

- Takes two inputs: `in` and `sel`
- If `sel == 0`, then `a = in` and `b = 0`
- Otherwise, `a = 0` and `b = in`



❖ DMux Gate Truth Table:

in	sel	a	b
0	0	0	0
0	1	0	0
1	0	1	0
1	1	0	1

Example of Applying the Mux Gate

- ❖ Draw the circuit diagram that corresponds to the following pseudocode:

```
if (sel == 0) {  
    out = ~a | ~b;  
} else {  
    out = a | b;  
}
```

Implementation a Mux and DMux Gate

- ❖ We can follow the same process as implementing an **Xor** gate from earlier

- ❖ Plan of action:
 - Step 1: Create the logic operation's **truth table**
 - Step 2: Use truth table to generate a **Boolean function** using strategies we've learned, such as the Boolean Function Synthesis
 - Step 3: Convert Boolean function to **HDL**

- ❖ Exercise for you to practice in Project 2

Post-Lecture Reminders

- ❖ **Project 1 due tonight (10/6) at 11:59pm**
- ❖ Project 2 (Study Skills Inventory & Boolean Logic) released today, due next Thursday (10/13)
- ❖ Course Staff Support
 - Eric has office hours in CSE2 153 today after lecture
 - Post your questions on the Ed discussion board