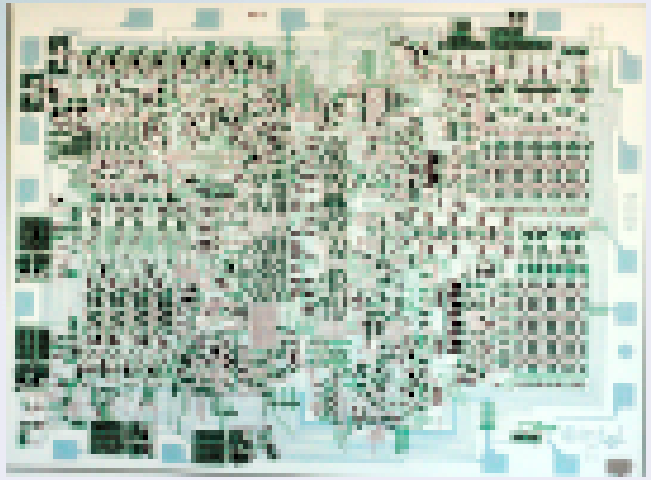


Lecture 2b: Combination Logic

CSCI11: **Computer Architecture and Organization**

Chapter 3.1-3.3.3 in Patt and Patel: Introduction to Computing Systems...

All of:

- Resources/**AspenCore Boolean Algebra**
- Resources/**AspenCore Combinational Logic**

- Revisit Wednesday lecture from last week
- Add XOR or exclusive-OR
- Expand on circuits
- Add in Combination Logic

Using logic.ly:

1. Count the number of AND, OR then add them first
2. Think of switches as binary variables
3. Think of the light bulb as a binary output
4. The value of 0 and 1 are binary constants
5. A and NOT A come from the same switch

Exclusive OR (XOR)

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive OR

- Only 1 when the inputs differ
- True if only 1 input is true (exclusive)
- "*Not equals*", true if inputs not equal
- Conditional inversion:
  - if a = 0, b = b
  - if a = 1, b = !b

for all bits in a word!

Exclusive OR

- Cryptography - Msg XOR key
- Parity checking
- Pixel graphics (80's or embedded controllers)

A Deep Dive on XOR

Combination Logic

Most of this class will be a Lab. We will be using the logic simulator to create more advanced circuits. The four circuits will be:

1. 2 input Decoder
2. 1 to 4 Demultiplexer
3. 2 bit Full Adder
4. 2-input Multiplexer

2 Input Decoder for Selecting a Memory Cell

A decoder converts (decodes) a n-bit binary value to a n-digit word.

- Decoder takes these 2 (n) binary inputs
- Activates one of four output lines (2^n)
- To select a corresponding memory word.

This setup is more efficient than using individual select lines for each memory cell, as it reduces the number of control signals needed from N (number of words) to $\log_2 N$.

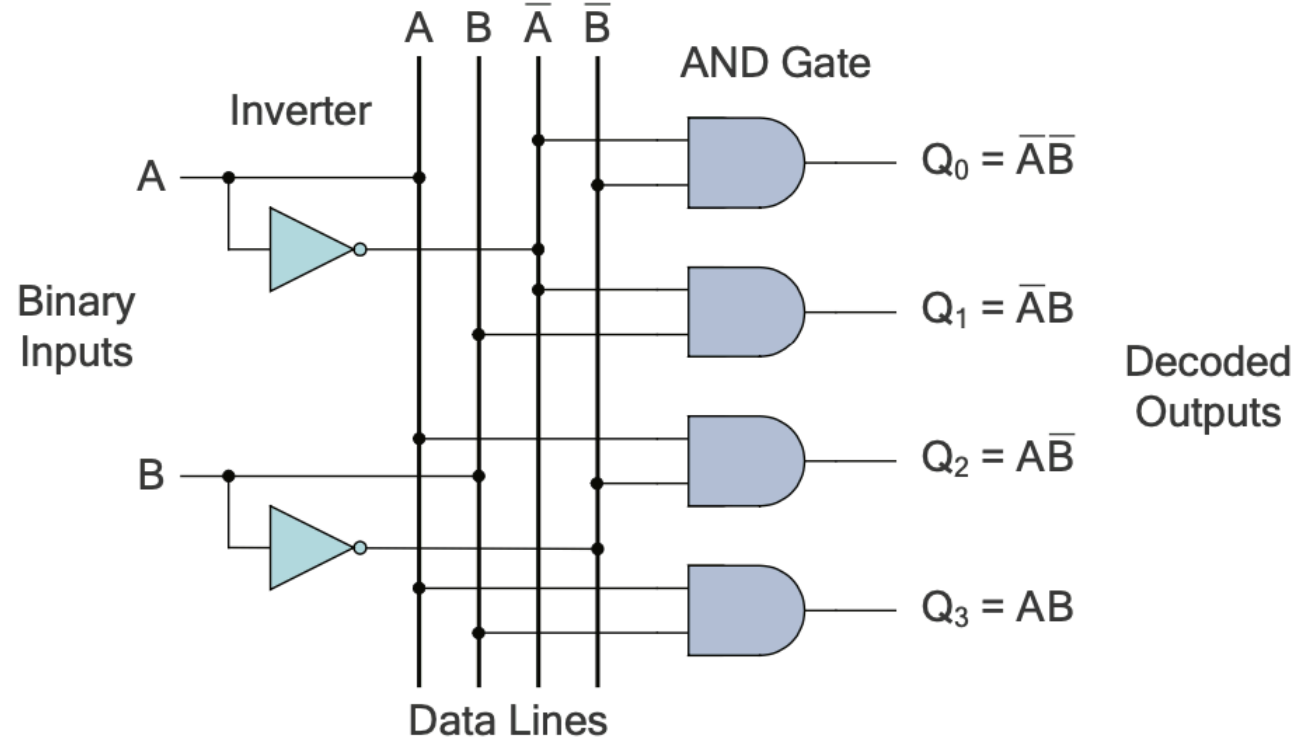
Memory Structure:

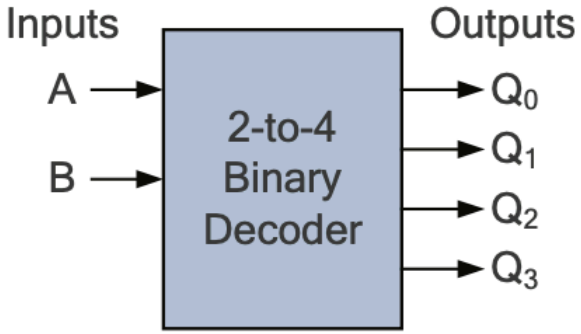
- **4 memory cells** (words)
- **2 address lines** (inputs) needed to select one of the four cells
- The **2:4 decoder** converts these 2 inputs into 4 select lines

Input Address Lines → Output Selected Cell or Register

- 00 → Cell 0 or Register 0
- 01 → Cell 1 or Register 1
- 10 → Cell 2 or Register 2
- 11 → Cell 3 or Register 3

FIGURE 3. A 2-TO-4 BINARY DECODER





Truth Table

B	A	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

1 to 4 Demultiplexer (1 to n demultiplexer)

This demultiplexer has a 1 input and 2 select lines to distribute the input to 1 of 4 outputs.

A 1-to-4 demultiplexer is a device that:

- Takes 1 input data line (*F*)
- Uses 2 select lines (*a* and *b*)
- Distributes the input to 1 of 4 output lines (**A** through **D**)

Light an 8 segment LED:

- Human eye can't detect lights blinking faster than 60Hz - 120H
- Use a demux to select a segment led then apply a PWM signal to light the led
- Uses 4 lines, 1 PWM and 3 select lines
- Must cycle faster than 8 * 120Hz or ~1kHz.

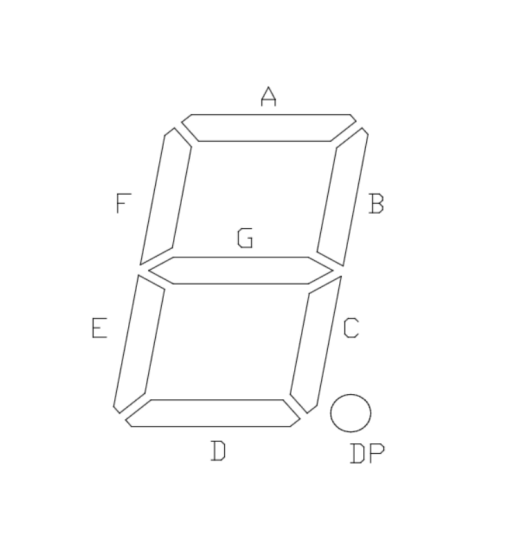
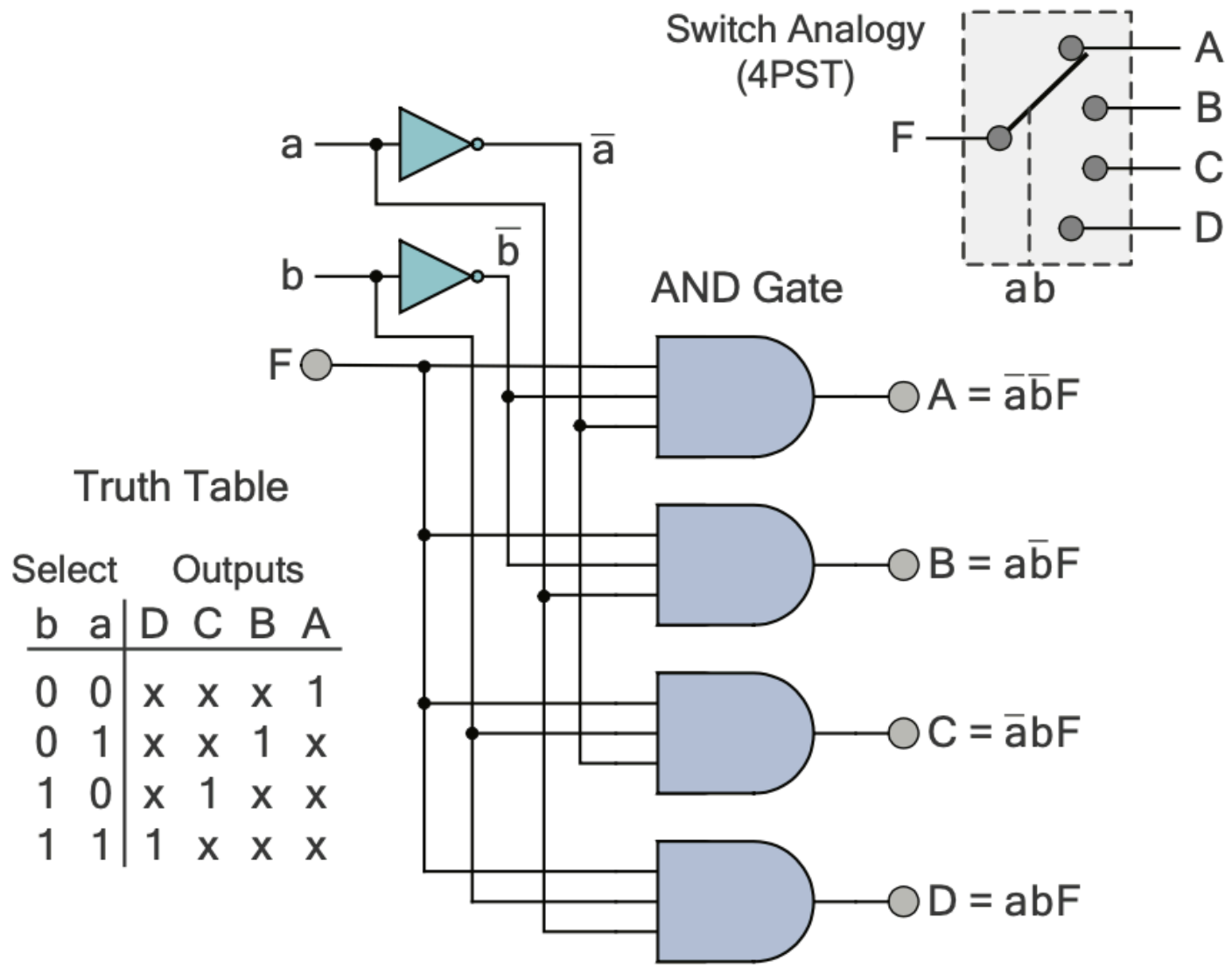


FIGURE 13. 4-CHANNEL DEMULTIPLEXER CIRCUIT



Truth Table

Select		Outputs			
b	a	D	C	B	A
0	0	x	x	x	1
0	1	x	x	1	x
1	0	x	1	x	x
1	1	1	x	x	x

Full Adder 2-bit

Description

A **2-bit full adder** can be used to create a counter which could serve as a program counter.

1. The **2-bit full adder** can count from 0 to 3 (00 to 11 in binary)
2. To create a simple program counter:
    - Initial state starts at 00
    - One input is permanently set to 1 (01)
    - Each clock cycle, the sum output is fed back as the other input
    - The carry output can be used to detect overflow

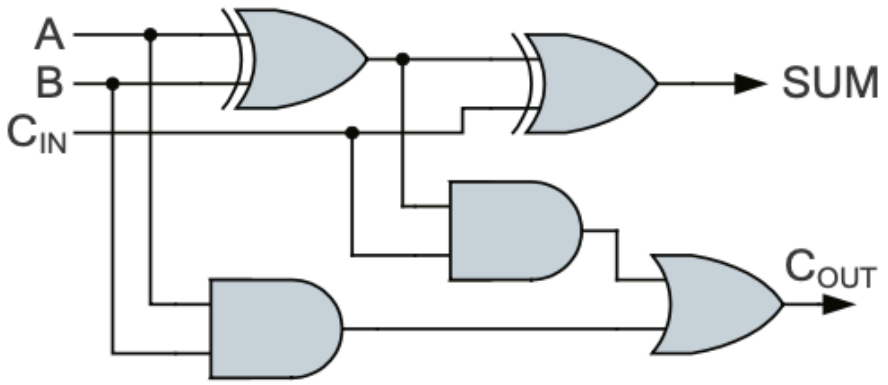
The counting sequence would be:

```
Initial state: 00
After 1st cycle: 01  (00 + 01)
After 2nd cycle: 10  (01 + 01)
After 3rd cycle: 11  (10 + 01)
After 4th cycle: 00  (11 + 01, with overflow)
```

- Creates a simple modulo-4 counter
- Continuously cycles through 0-3 values
- Can be used for basic program sequencing or timing purposes.
- The carry output (overflow) can be used to trigger other circuit elements (divide by 4)

One could take this program counter and add the *2 to 4 Decoder* and have a very simple state machine.

FIGURE 21. FULL-ADDER AND TRUTH TABLE

Symbol	Truth Table				
	Carry-In	B	A	SUM	Carry-Out
	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

2-input multiplexer

Description

A **2-input multiplexer** (also called a **2:1 mux**) is like a digital switch that selects between two input signals based on a selection line.

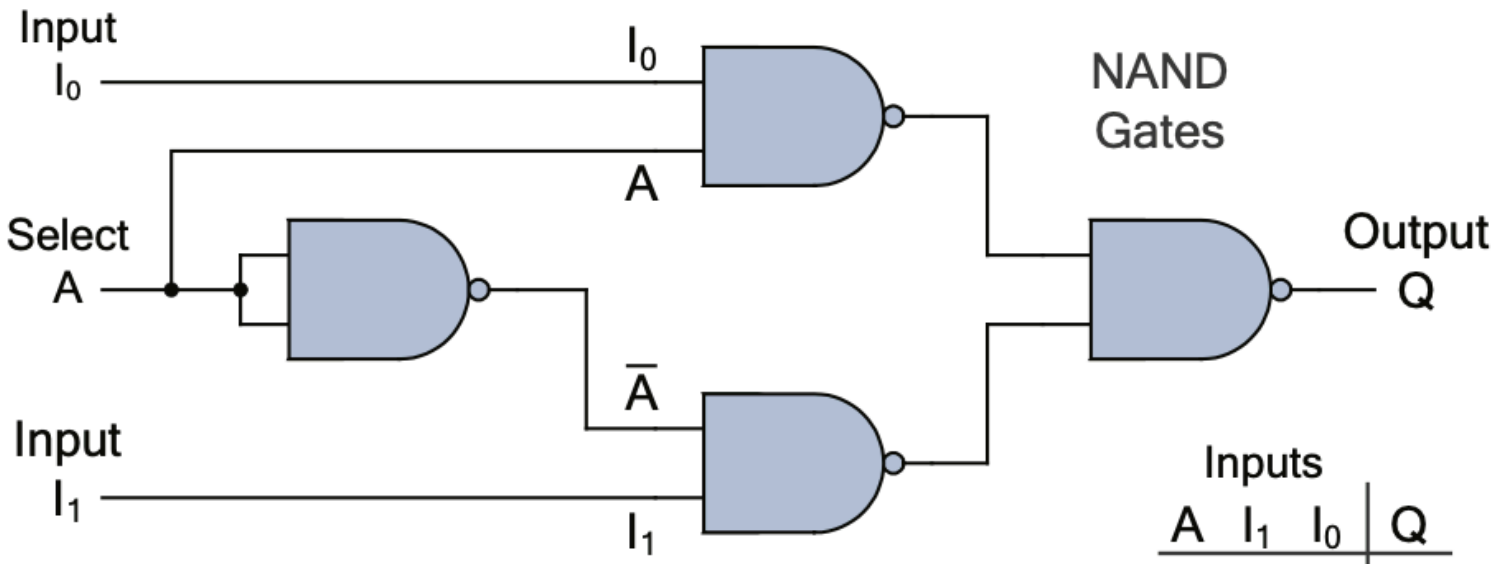
Here's how it works:

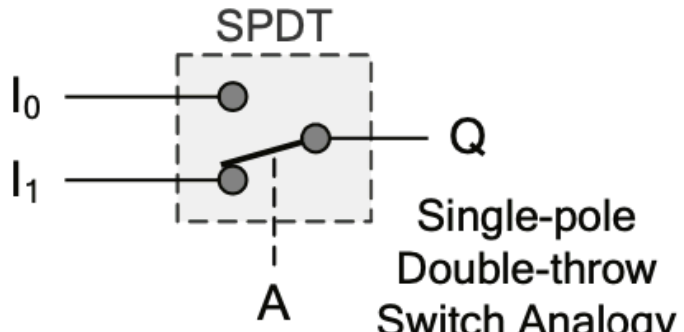
- When **A** (selection line) is **0**, the output will be whatever value is on input **I0**
- When **A** is **1**, the output will be whatever value is on input **I1**

Application

A common practical application would be selecting between two different clock sources. Assume you only have a 16-bit counter, which means at 1MHz (*1 tick per usec, one million ticks per second*), you can count 65,535 ticks or .065 seconds. However, if you had two clocks, high-speed at 1MHz and low speed at 1kHz (1000 ticks per second), you could have both a high precision timer and a long timer, up to 1 minute and 5.53 seconds.:

FIGURE 9. BASIC 2-INPUT MULTIPLEXER USING NAND GATES





Truth Table

Inputs			
A	I_1	I_0	Q
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

Combination Logic

1. **1 to 4 Demultiplexer**
    - data *distributor* - same input, multiple outputs
    - passes 1 input to 1 of 4 outputs
    - register or memory selection with data
2. **2-input Multiplexer**
    - data *selector*, multiple inputs, single output
    - passes 2 inputs to 1 output
    - select a specific clock signal for timing

Combination Logic (cont.)

1. **2 input Decoder**
    - converts a n-bit value to a string of 2^n bits
    - register or memory selection
    - (*yes, a demux w/o the data*)
2. **2 bit Full Adder**
    - n-bit counter
    - program counter
    - divides by 2^n-bits