Lecture 5-1: Digital System

CS10014 Computer Organization

Department of Computer Science
Tsung Tai Yeh

Thursday: 1:20 pm– 3:10 pm
Classroom: EC-022

Acknowledgements and Disclaimer

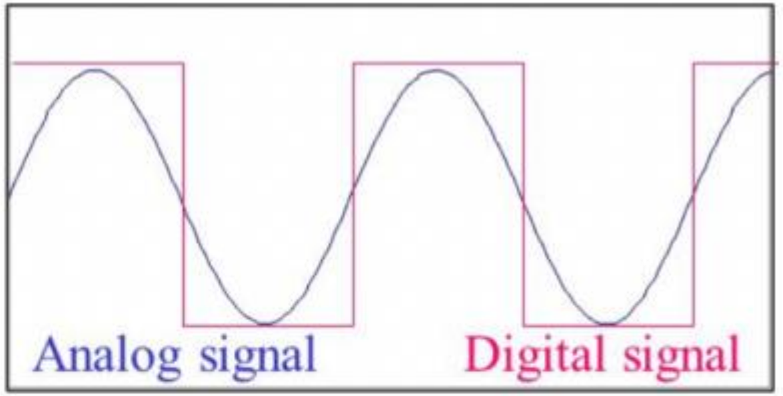
- Slides were developed in the reference with
  - CS 61C at UC Berkeley
    - https://inst.eecs.berkeley.edu/~cs61c/sp23/
  - CS 252 at UC Berkeley
    - https://people.eecs.berkeley.edu/~culler/courses/cs252-s05/
  - CSCE 513 at University of South Carolina
    - https://passlab.github.io/CSCE513/

Outline

- Logic Gate
- Arithmetic Logic Unit (ALU)
- Clock Signal
- Latch
- D Flip-Flop
- Register

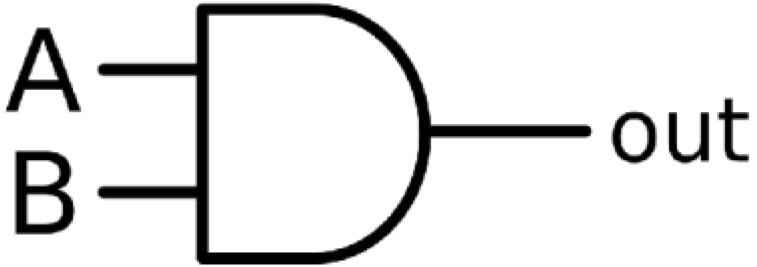
Digital Systems

- Digital
  - All values are discrete
  - A value can be on (1) or off (0)
- Analog
  - Have a continuous range of values



Logic Gate: AND

A	B	out
0	0	0
0	1	0
1	0	0
1	1	1



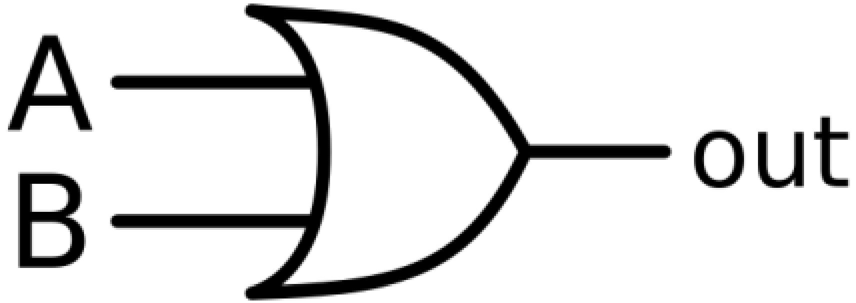
```
out = A & B
```

$\uparrow$

C syntax

Logic Gate: OR

A	B	out
0	0	0
0	1	1
1	0	1
1	1	1



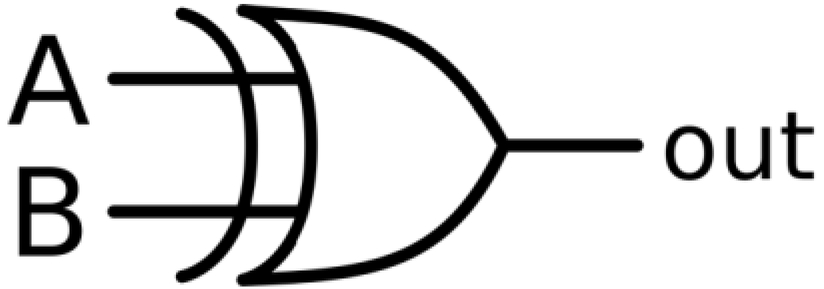
```
out = A | B
```

↑

C syntax

Logic Gate: XOR

A	B	out
0	0	0
0	1	1
1	0	1
1	1	0



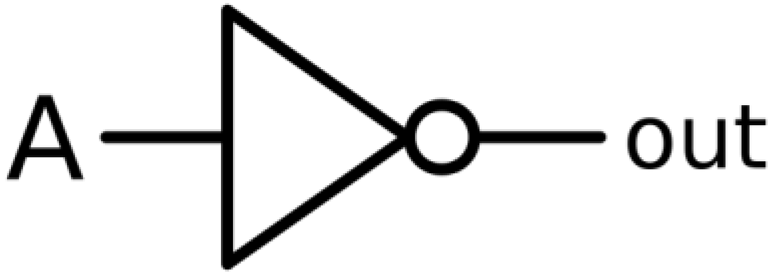
```
out = A ^ B
```

$\uparrow$

C syntax

Logic Gate: NOT

A	out
0	1
1	0

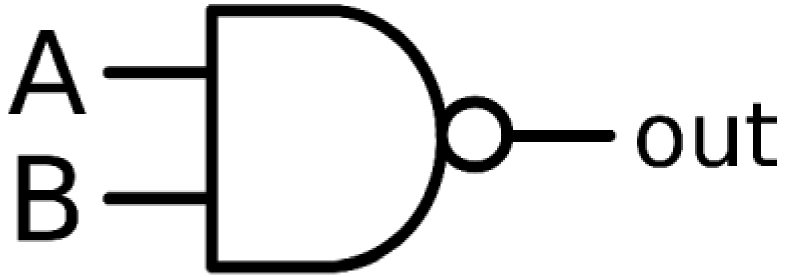


```
out = ~A
```

C syntax

Logic Gate: NAND

A	B	out
0	0	1
0	1	1
1	0	1
1	1	0

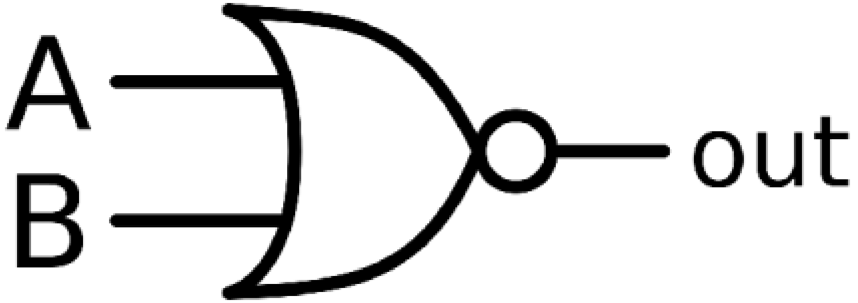


```
out = ~(A & B)
```

C syntax

Logic Gate: NOR

A	B	out
0	0	1
0	1	0
1	0	0
1	1	0

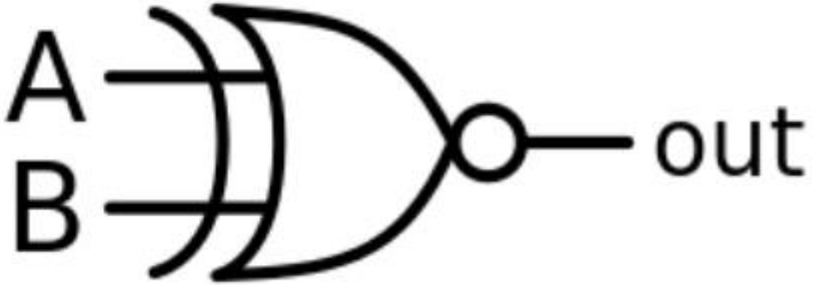


```
out = ~(A | B)
```

C syntax

Logic Gate: XNOR

A	B	out
0	0	1
0	1	0
1	0	0
1	1	1

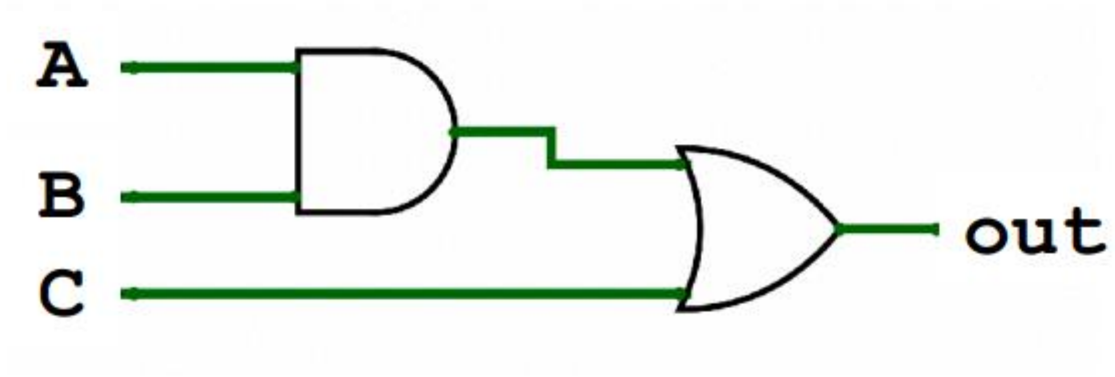


```
out = ~(A ^ B)
```

C syntax

Connecting Logic Gate

- Out = (A & B) | C
  - (A, B, and C are one bit each)



Types of Circuit

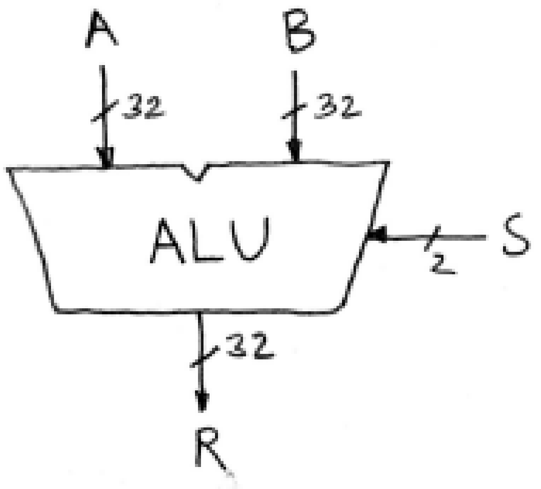
- Combinational Logic (CL) circuits
  - Output is a function of the inputs only
  - Similar to a pure function in mathematics, y = f(x)
- State Elements
  - Circuit that store information

Outline

- Logic Gate
- Arithmetic Logic Unit (ALU)
- Clock Signal
- Latch
- D Flip-Flop
- Register

Arithmetic Logic Unit (ALU) (1/12)

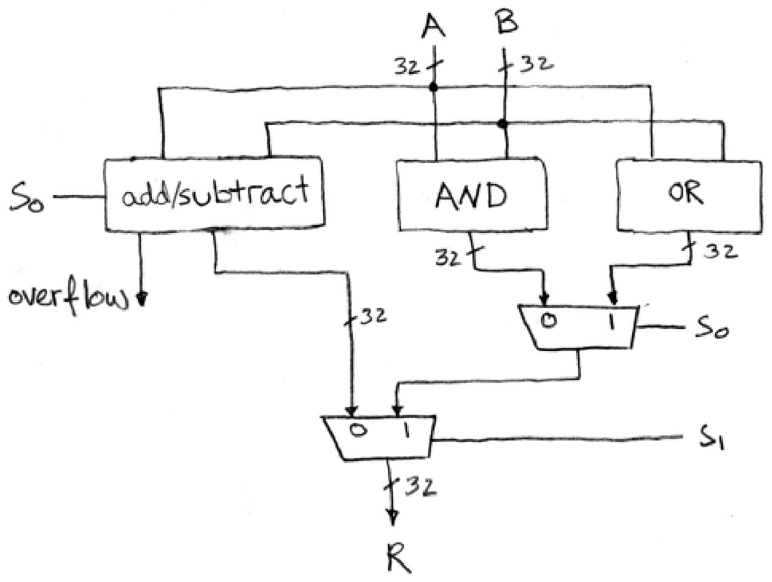
- The ALU is used to compute the result of R-type instructions (ADD, SUB, ADDI, AND, OR)



When S = 00, R = A + B
When S = 01, R = A – B
When S = 10, R = A & B
When S = 11, R = A | B

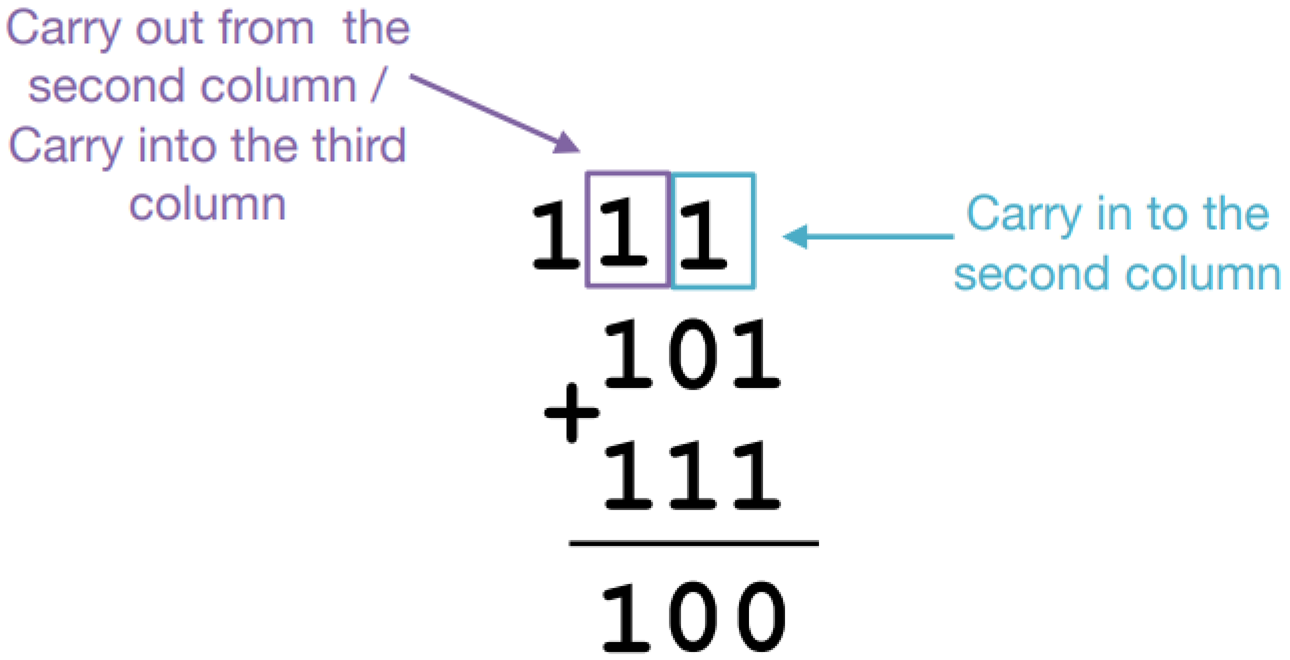
Arithmetic Logic Unit (ALU) (2/12)

- The ALU is used to compute the result of R-type instructions (ADD, SUB, ADDI, AND, OR)
  - A 32-bit bitwise AND unit
  - A 32-bit bitwise OR unit
  - A 32-bit ADD/SUBTRACT unit with a control line
  - The logic to output carry
  - Overflow
  - Zero
  - Negative



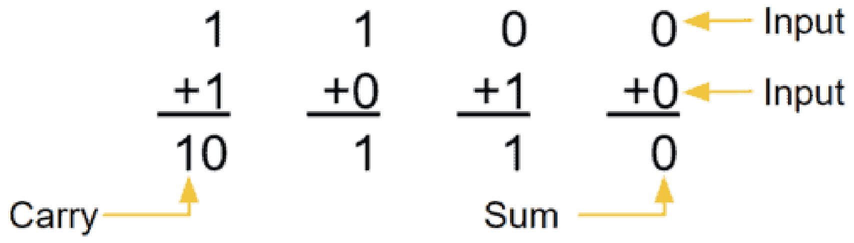
Arithmetic Logic Unit (3/12)

- Binary addition


Carry out from the second column / Carry into the third column
1 1 1
1 0 1
+
1 1 1
1 0 0
Carry in to the second column

Arithmetic Logic Unit (ALU) (4/12)

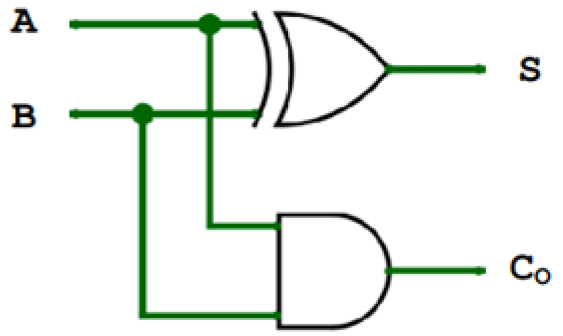
- Half adder
    - Add only two 1-bit binary numbers, sum is from 0 to 2



A	B	C_O	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$$S = A \oplus B$$

$$C_O = AB$$



Arithmetic Logic Unit (ALU) (5/12)

- Full adder (single bit)
  - Add three 1-bit binary numbers
  - Sum is from 0 to 3, which can be expressed with two output bits (“11”)

S is 1 when 1 or 3 of the bits are 1

$$S = A \oplus B \oplus C_i$$

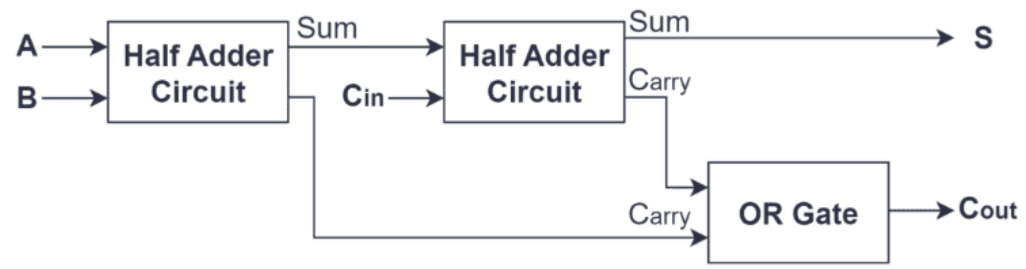
A	B	C_i	C_o	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

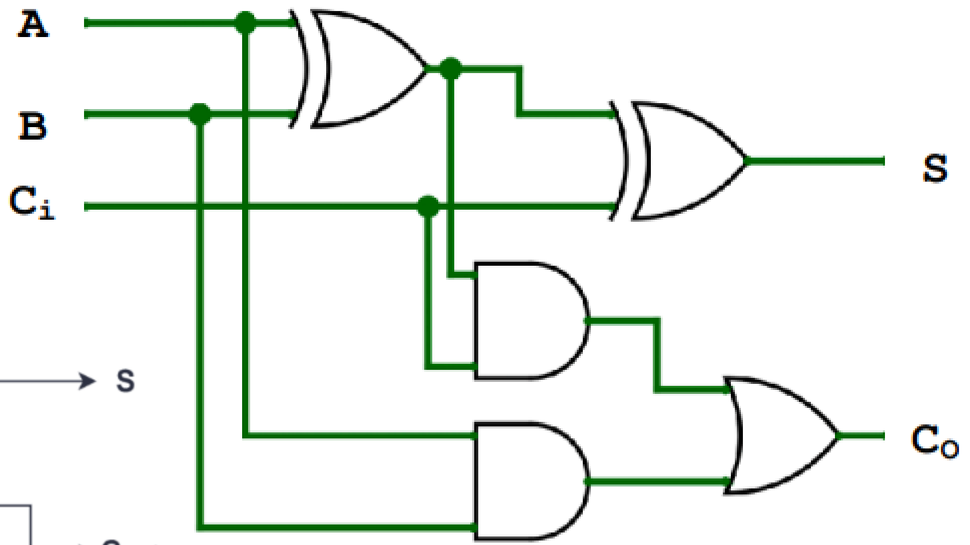
Arithmetic Logic Unit (ALU) (6/12)

- Full adder (single bit)

$$S = A \oplus B \oplus C_i$$

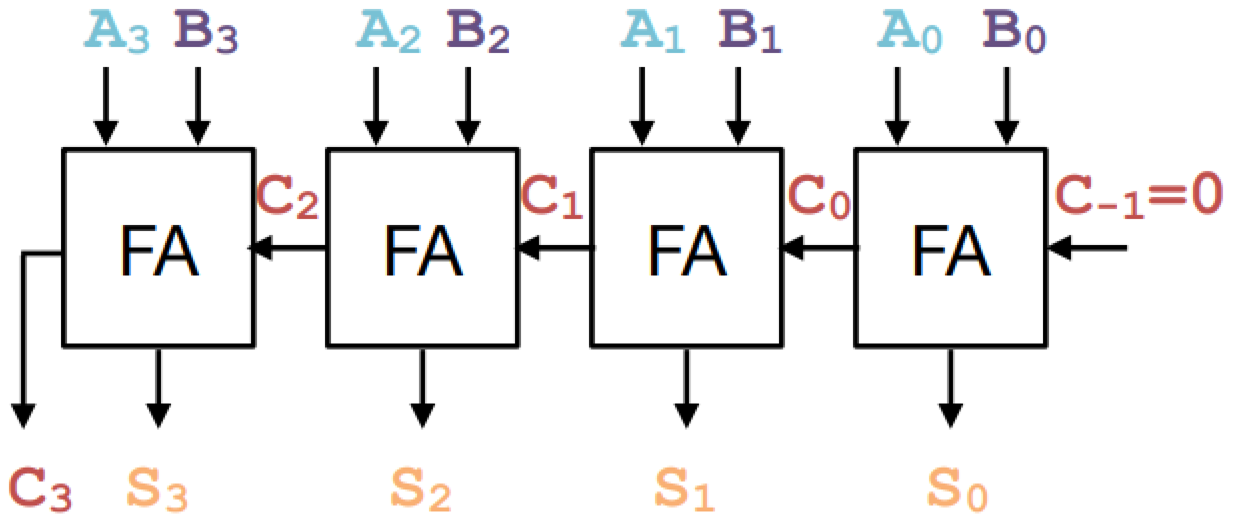
$$C_o = AB + C_i(A \oplus B)$$





Arithmetic Logic Unit (ALU) (7/12)

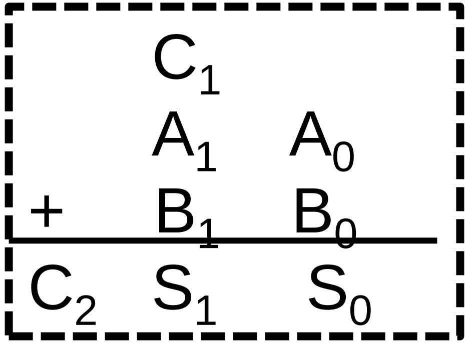
- 4-bit adder



$$\begin{array}{ccccc} C_3 & C_2 & C_1 & C_0 & C_{-1} = 0 \\ & A_3 & A_2 & A_1 & A_0 \\ & B_3 & B_2 & B_1 & B_0 \\ \hline & S_3 & S_2 & S_1 & S_0 \end{array}$$

Arithmetic Logic Unit (ALU) (8/12)

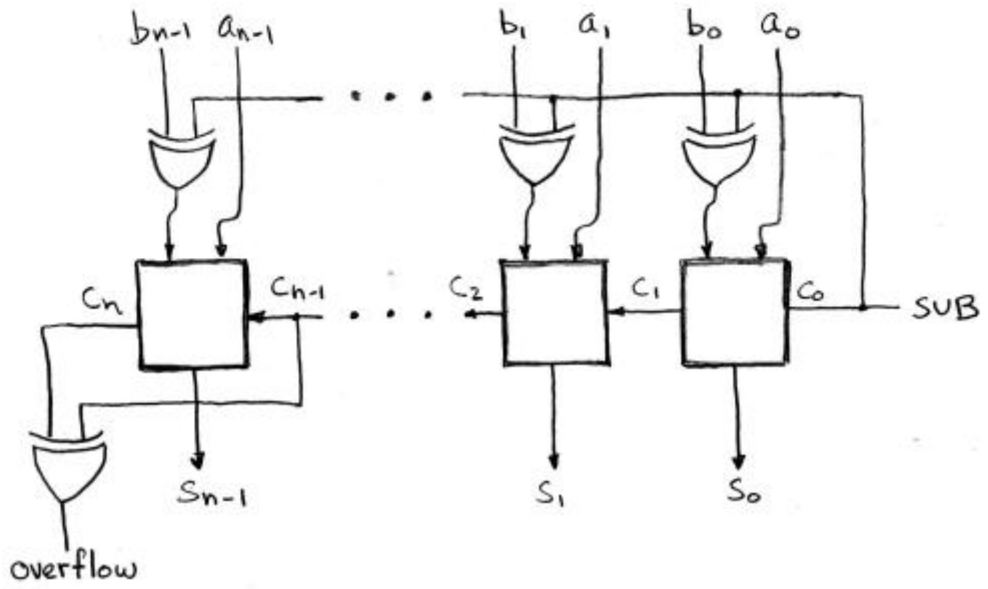
- Overflow
  - Indicate when the result cannot fit into n bits
- Highest adder
  - C_{in} = Carry-in = C_1
  - C_{out} = Carry-out = C_2
  - No C_{out} or C_{in} -> No overflow!
  - C_{in} and C_{out} -> No overflow!
  - C_{in}, but no C_{out} -> **Overflow!**
  - C_{out}, but no C_{in} -> **Overflow!**



Overflow = c_n XOR C_{n-1}

Arithmetic Logic Unit (ALU) (9/12)

- Subtractor (A – B = A + (-B))
    - XOR serves as conditional inverter!



A	B	out
0	0	0
0	1	1
1	0	1
1	1	0

XOR Truth Table

Arithmetic Logic Unit (ALU) (10/12)

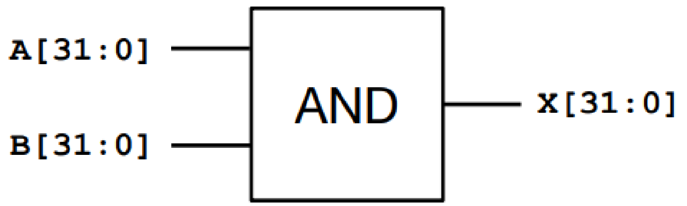
- Logical AND on 32-bit number

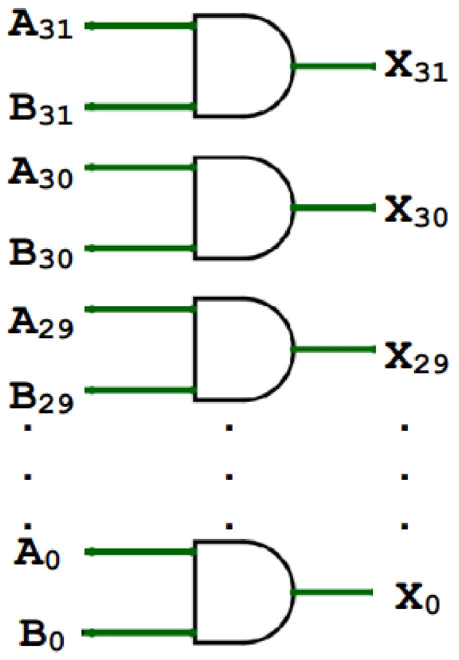
$$X = AB$$

$$A = A_{31}\ A_{30}\ A_{29} \ldots A_0$$

$$B = B_{31}\ B_{30}\ B_{29} \ldots B_0$$

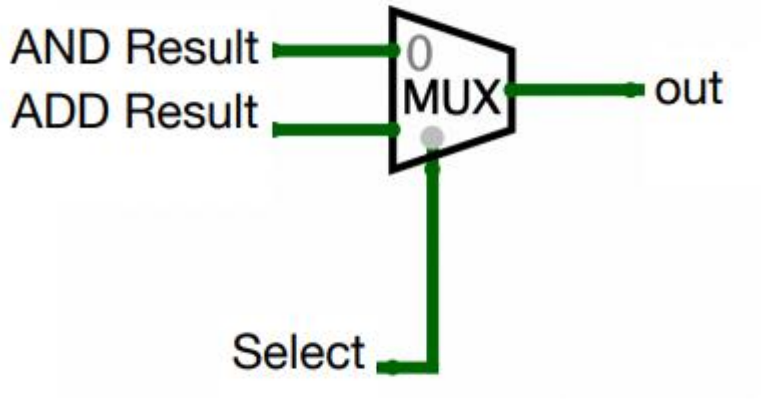
$$X = X_{31}\ X_{30}\ X_{29} \ldots X_0$$





Arithmetic Logic Unit (ALU) (11/12)

- 2:1 Multiplexors
    - Selects an input to propagate to the output

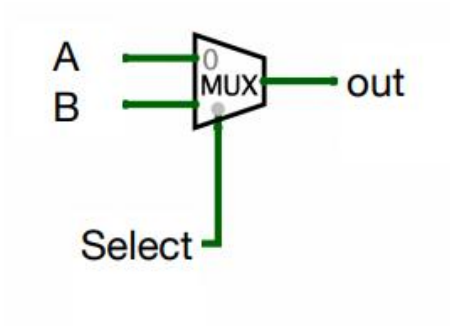


If select = 0, out = AND Result
If select = 1, out = ADD Result

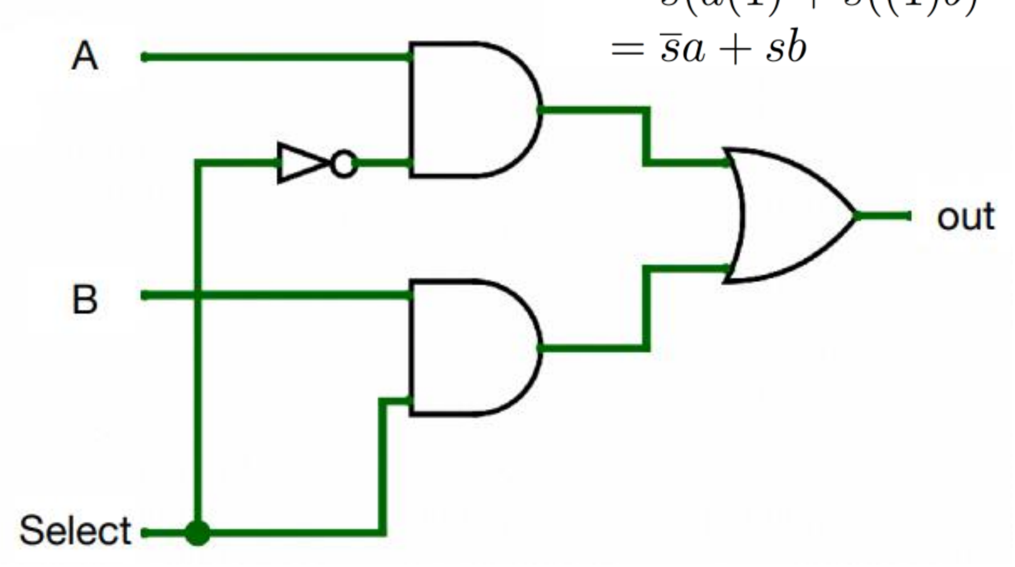
Arithmetic Logic Unit (ALU) (12/12)

- 2:1 Multiplexors
  - Selects an input to propagate to output

$$
\begin{aligned}
c &= \overline{s}a\overline{b} + \overline{s}ab + s\overline{a}b + sab \\
&= \overline{s}(a\overline{b} + ab) + s(\overline{a}b + ab) \\
&= \overline{s}(a(\overline{b} + b)) + s((\overline{a} + a)b) \\
&= \overline{s}(a(1) + s((1)b) \\
&= \overline{s}a + sb
\end{aligned}
$$



$\texttt{out} = \texttt{A}\overline{\texttt{S}} + \texttt{BS}$

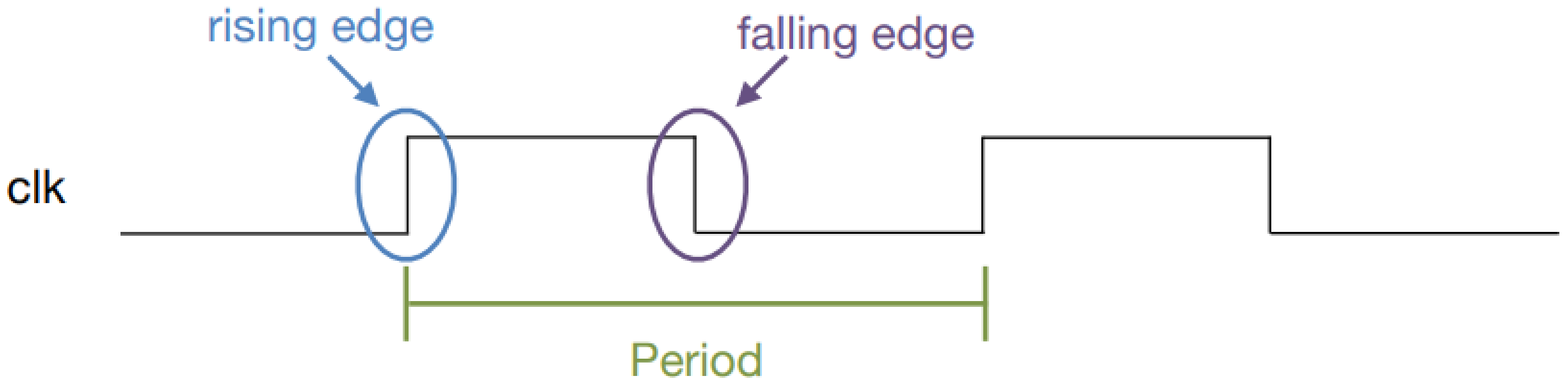


Outline

- Logic Gate
- Arithmetic Logic Unit (ALU)
- Clock Signal
- Latch
- D Flip-Flop
- Register

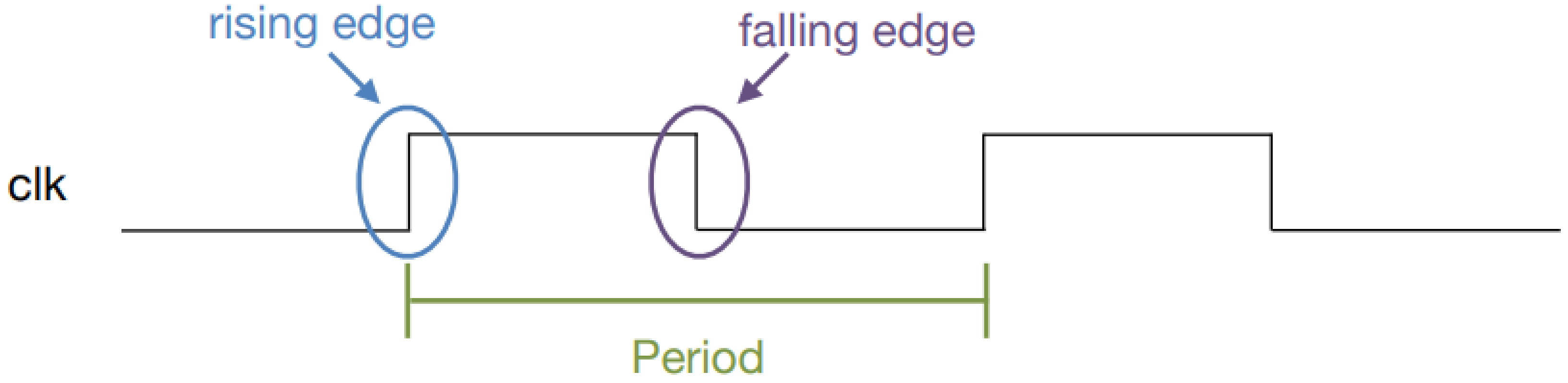
Clock Signal (1/2)

- Clock Signal
  - Oscillates between a high and low state
  - Period = time between one rising edge to the next rising edge
  - Frequency = 1/period



Clock Signal (2/2)

- Clock
  - The unit for frequency is Hertz (Hz)
  - E.g. clock frequency is 4 GHz
    - The clock goes through 4 billion cycles every second
    - Period = 1/frequency = 1/4GHz = 0.25 ns

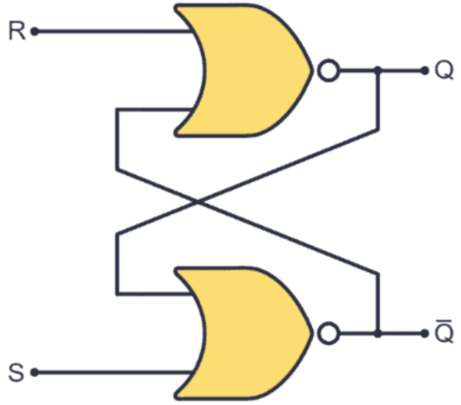


Outline

- Logic Gate
- Arithmetic Logic Unit (ALU)
- Clock Signal
- Latch
- D Flip-Flop
- Register

Latch (1/2)

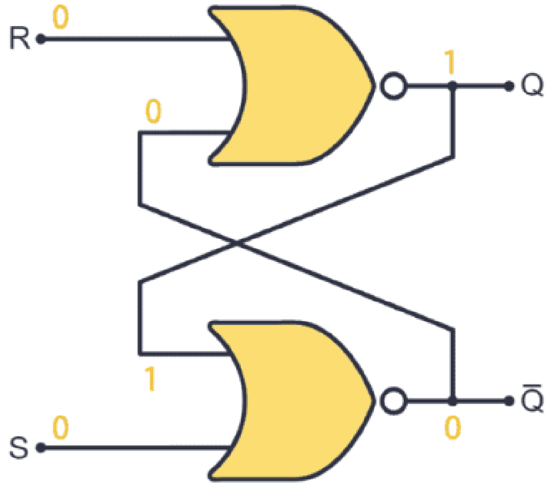
- **Latch**
  - An asynchronous circuit (it doesn't require a clock signal to work)
  - Has two stable states, HIGH ("1"), and LOW ("0")
  - Can be used for storing binary data
- **Set-Reset (S-R) latch**
  - Two NOR gates with a cross-feedback loop
  - The feedback path stores one bit of data as long as the circuit is powered
  - Two inputs (R and S)
  - Two outputs (Q and inverted Q)



S-R Latch with NOR gates

Latch (2/2)

- Set-Reset (S-R) latch
  - NOR gate gives 1 when both of its inputs are "0"
  - When both inputs S and R are equal to "0", the output Q remains the same as it was (saves the previous value)
  - If one input is "1", the output will be "0"

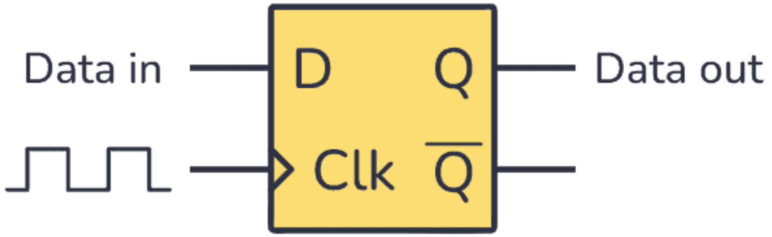


Input S	Input R	Output Q
0	0	Previous State
0	1	0
1	0	1
1	1	0 (Invalid)

Outline

- Logic Gate
- Arithmetic Logic Unit (ALU)
- Clock Signal
- Latch
- D Flip-Flop
- Register

D Flip Flop (1/3)



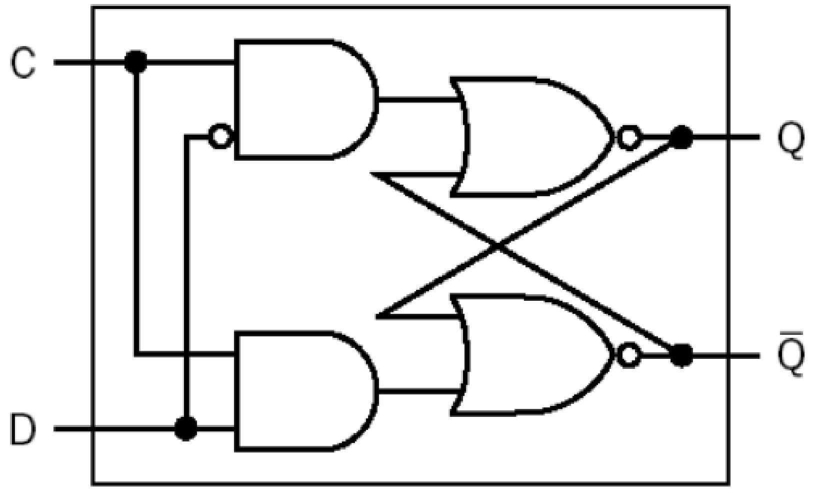
D Flip-Flop

- Flip Flops
  - A flip-flop is a state element
  - State element: A circuit component that can hold a value
  - Store one bit (1 or 0) on flip flop output
  - Synchronous circuit that need a clock signal (Clk)
- D Flip-Flop
  - The D Flip-Flop will only store a new value from the D input when the clock goes from 0 to 1 (rising edge) or 1 to 0 (falling edge)
  - Commonly used as a basic building block to create counters or memory blocks such as shift register

D Flip-Flop (2/3)

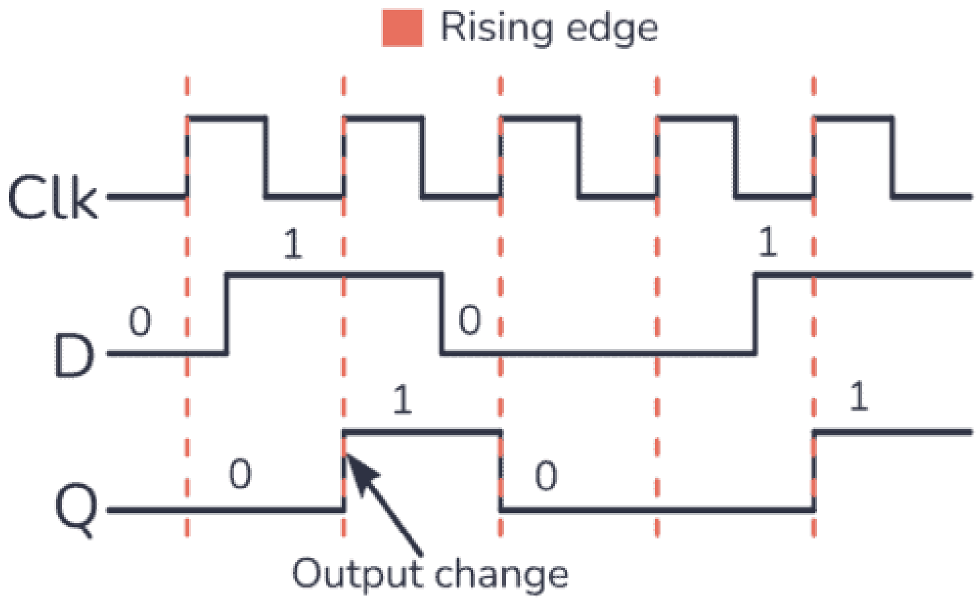
- Inputs C (clock) and D
- When C is 1, latch open, output = D
- When C is 0, latch closed, output = store value

C	D	AND
0	0	0
0	1	0
1	0	0
1	1	1



D Flip-Flop (3/3)

- The output Q only changes to the value of the D input at the moment the clock goes from 0 to 1

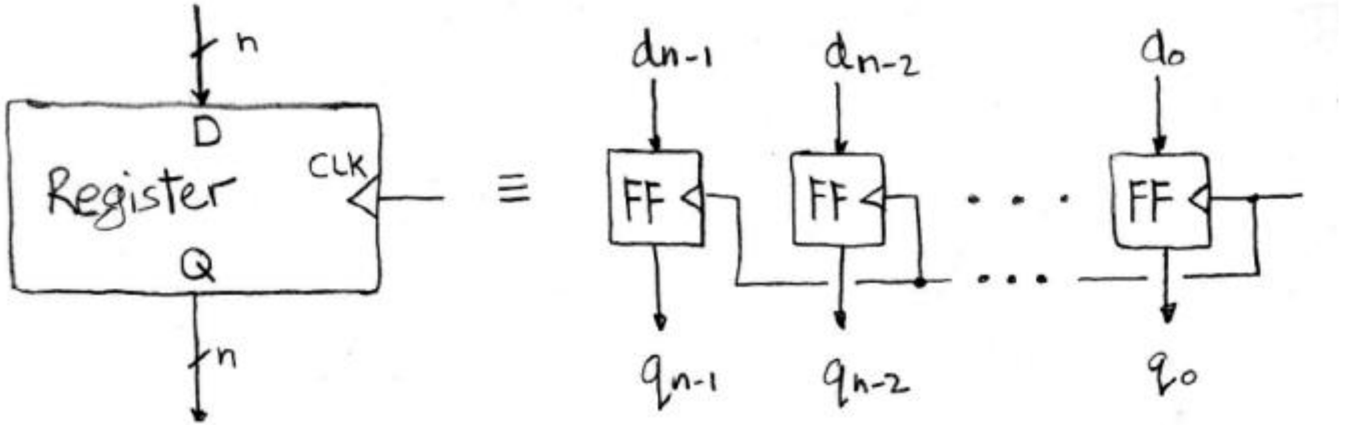


Outline

- Logic Gate
- Arithmetic Logic Unit (ALU)
- Clock Signal
- Latch
- D Flip-Flop
- Register

Register

- How to store a 32-bit number?
  - Put 32 flip flops together
  - A register is a state element



Summary

- ALU can be implemented using a mux
  - Couple with basic block elements
- N bit adder-subtractor done using N 1-bit adders with XOR gates on input
  - XOR serves as conditional inverter