

# Logic Design

Lecture 9

October 23<sup>rd</sup>, 2018

Jae W. Lee ([jaewlee@snu.ac.kr](mailto:jaewlee@snu.ac.kr))

Computer Science and Engineering

Seoul National University

***Slide credits:*** [CS:APP3e] slides from CMU; [COD5e] slides from Elsevier Inc.

# Today

**Textbook: [CS:APP3e] 4.2**

- **Overview**
- **Computation (Combinational Logic)**
- **Storage (Sequential Logic)**
- **Hardware Control Language (HCL)**

# Overview: Logic Design

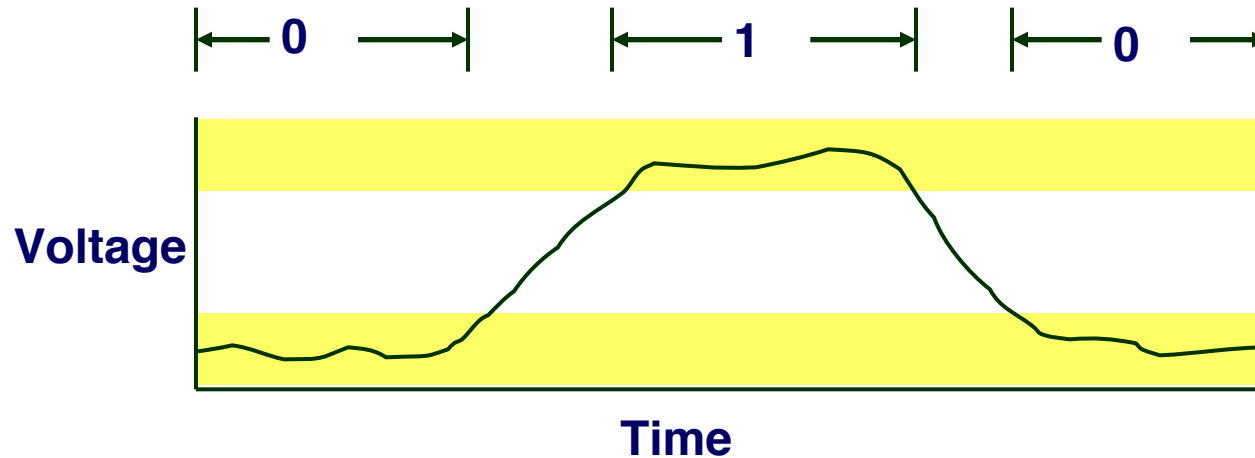
## ■ Fundamental hardware requirements

- Communication
  - How to get values from one place to another
- Computation
- Storage

## ■ Bits are our friends

- Everything expressed in terms of values 0 and 1
- Communication
  - Low or high voltage on wire
- Computation
  - Compute Boolean functions
- Storage
  - Store bits of information

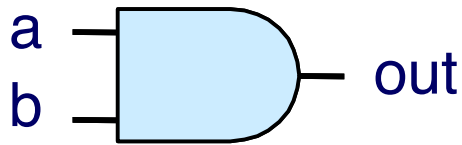
# Overview: Digital Signals



- Use **voltage thresholds** to extract discrete values from continuous signal
- Simplest version: 1-bit signal
  - Either high range (1) or low range (0)
  - With guard range between them
- Not strongly affected by noise or low quality circuit elements
  - Can make circuits simple, small, and fast

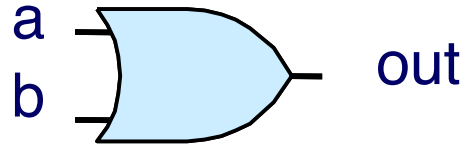
# Computing with Logic Gates

And



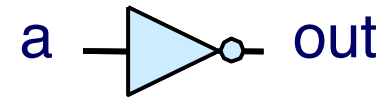
$$\text{out} = a \ \&\& \ b$$

Or



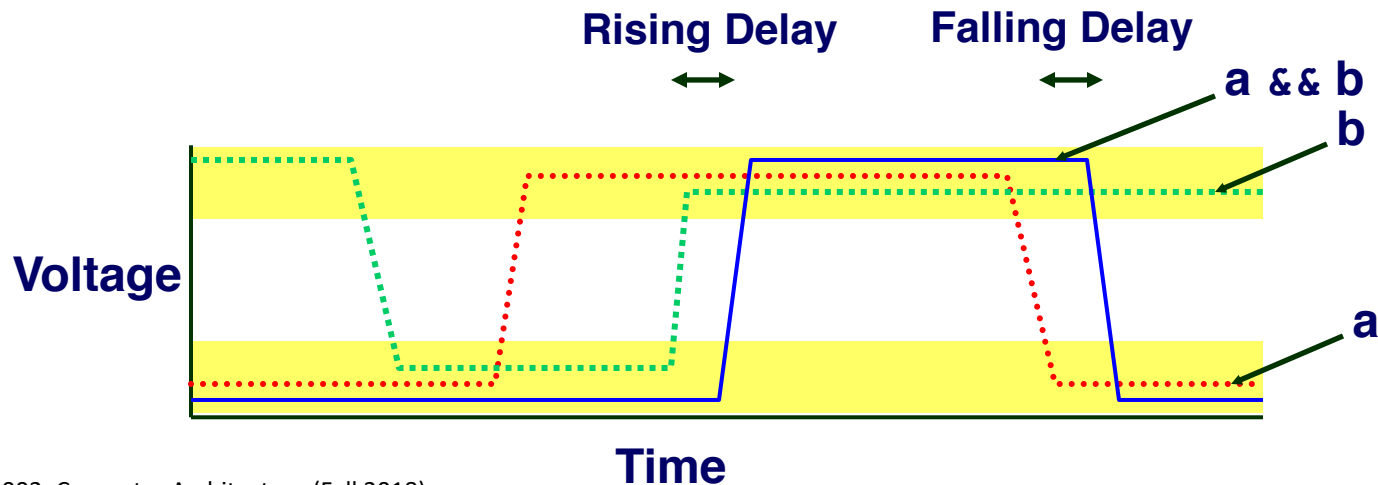
$$\text{out} = a \ || \ b$$

Not

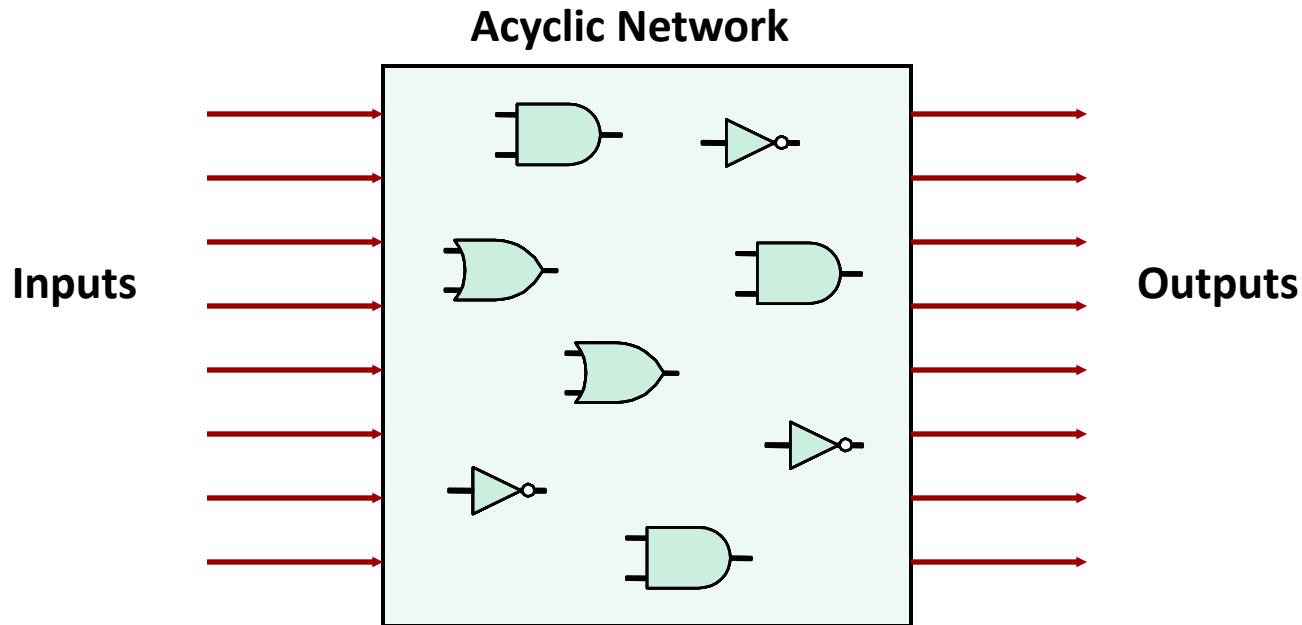


$$\text{out} = !a$$

- Outputs are Boolean functions of inputs
- Respond continuously to changes in inputs
  - With some, small delay



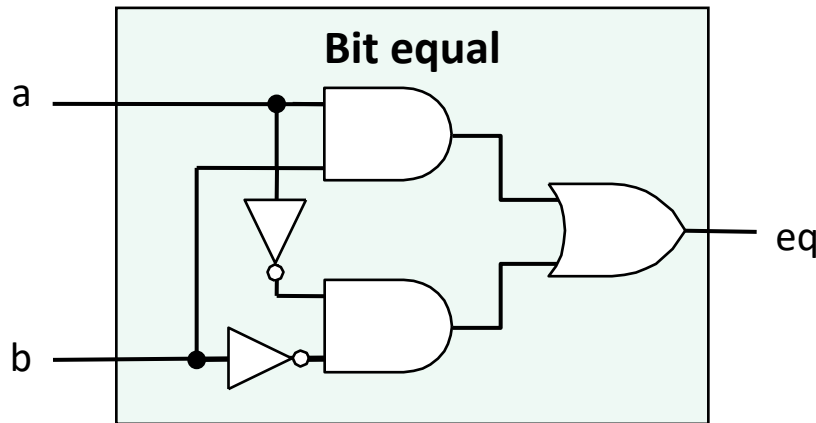
# Computation: Combinational Circuits



## ■ Acyclic network of logic gates

- Continuously responds to changes on primary inputs
- Outputs become (after some delay) Boolean functions of inputs

# Computation: Bit Equality

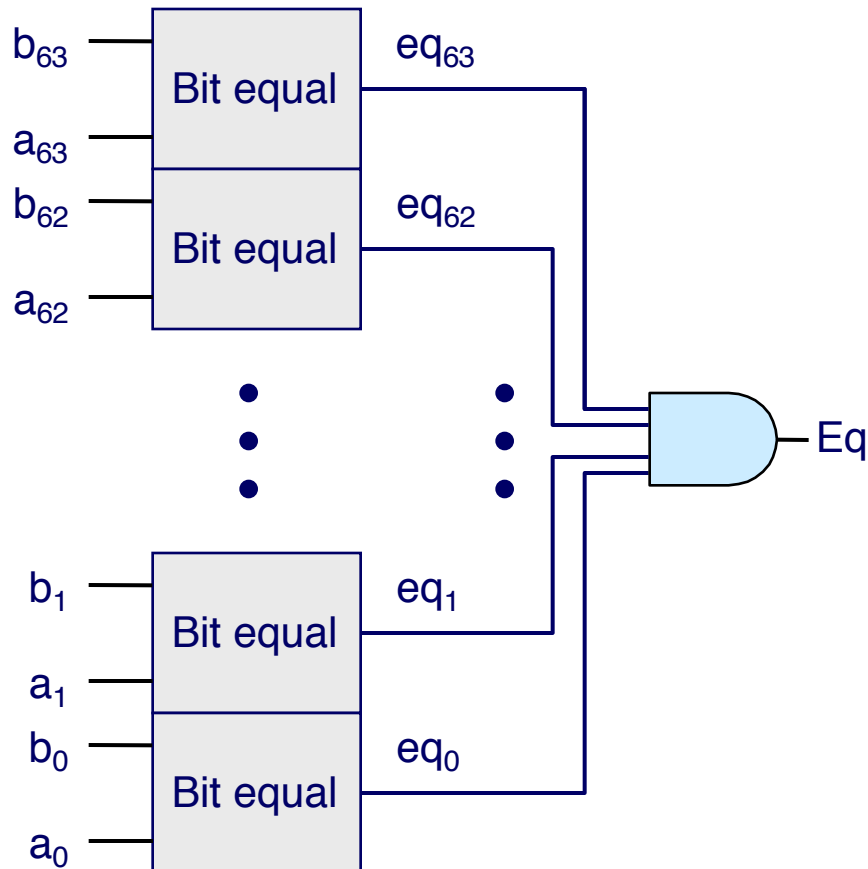


## HCL Expression

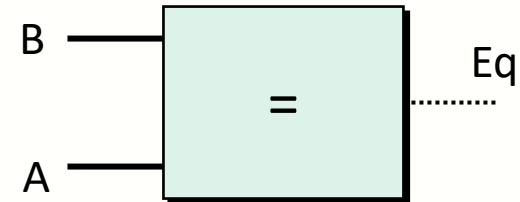
```
bool eq = (a&&b) || (!a&&!b)
```

- Generate 1 if a and b are equal
- **Hardware Control Language (HCL)**
  - Very simple hardware description language
    - Boolean operations have syntax similar to C logical operations
  - We'll use it to describe control logic for processors

# Computation: Word Equality



## Word-Level Representation



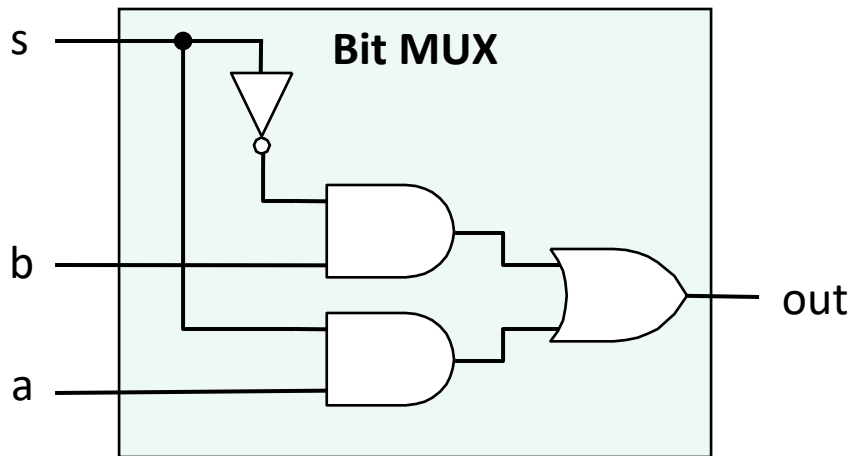
## HCL Representation

```
bool Eq = (A == B)
```

- 64-bit word size
- HCL representation
  - Equality operation
  - Generates Boolean value



# Computation: Bit-Level Multiplexor

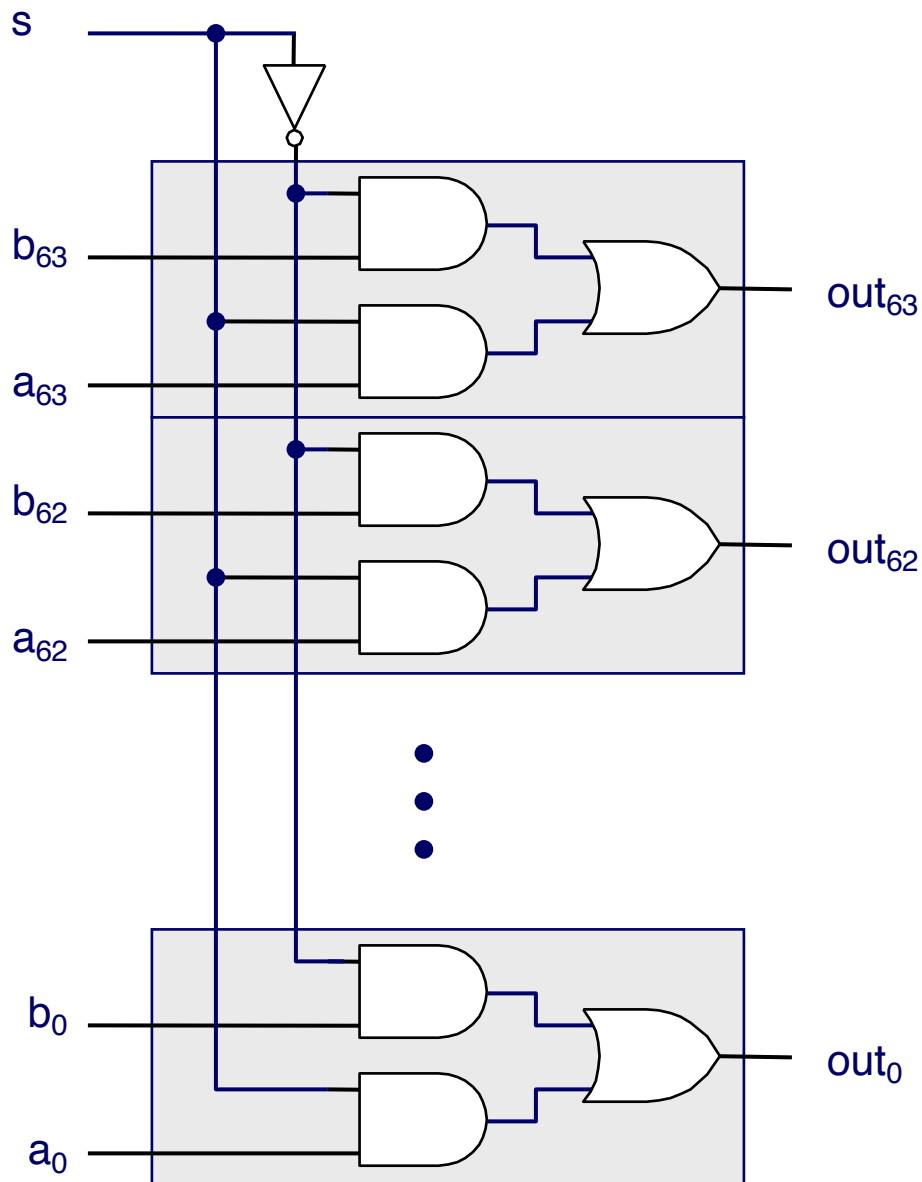


## HCL Expression

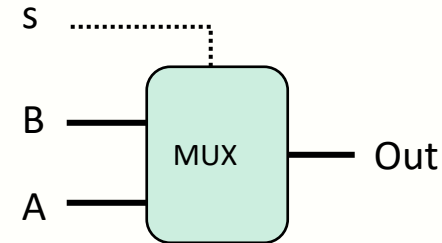
```
bool out = (s&&a) || (!s&&b)
```

- Control signal *s*
- Data signals *a* and *b*
- Output *a* when *s*=1, *b* when *s*=0

# Computation: Word Multiplexor



## Word-Level Representation



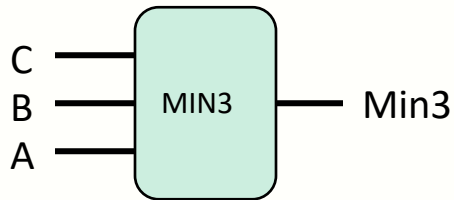
## HCL Representation

```
int Out = [
  s : A;
  1 : B;
];
```

- Select input word  $A$  or  $B$  depending on control signal  $s$
- HCL representation
  - Case expression
  - Series of test : value pairs
  - Output value for first successful test

# Computation: HCL Word-Level Examples

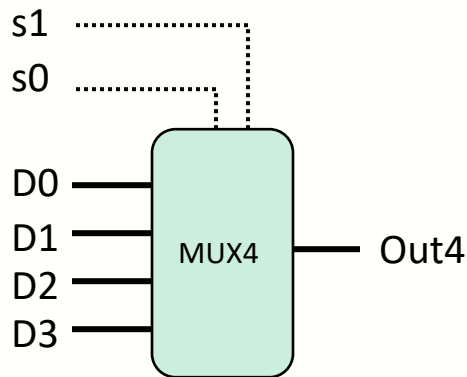
## Minimum of 3 Words



```
int Min3 = [
    A < B && A < C : A;
    B < A && B < C : B;
    1               : C;
];
```

- Find minimum of three input words
- HCL case expression
- Final case guarantees match

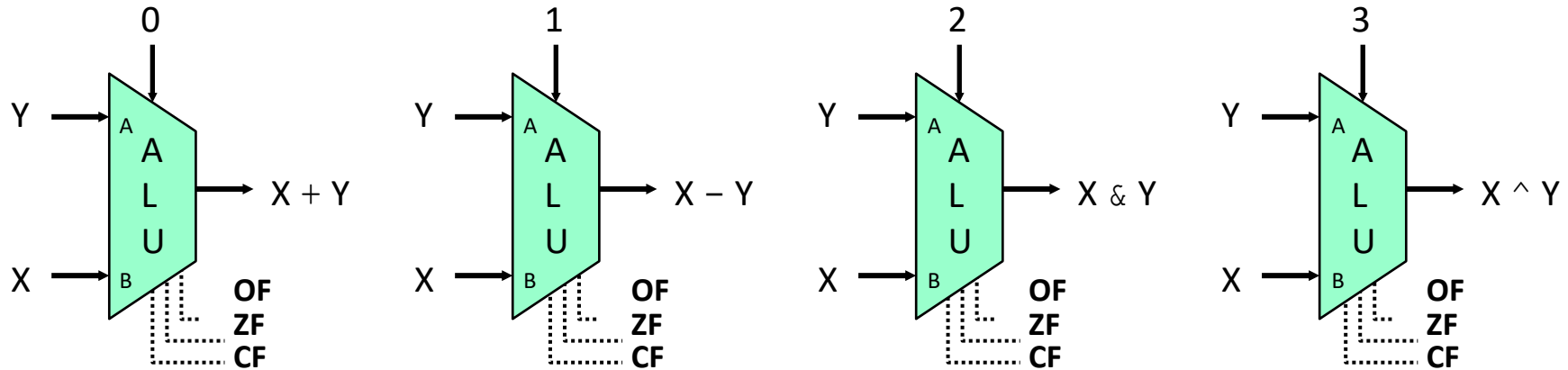
## 4-Way Multiplexor



```
int Out4 = [
    !s1 && !s0 : D0;
    !s1        : D1;
    !s0        : D2;
    1          : D3;
];
```

- Select one of 4 inputs based on two control bits
- HCL case expression
- Simplify tests by assuming sequential matching

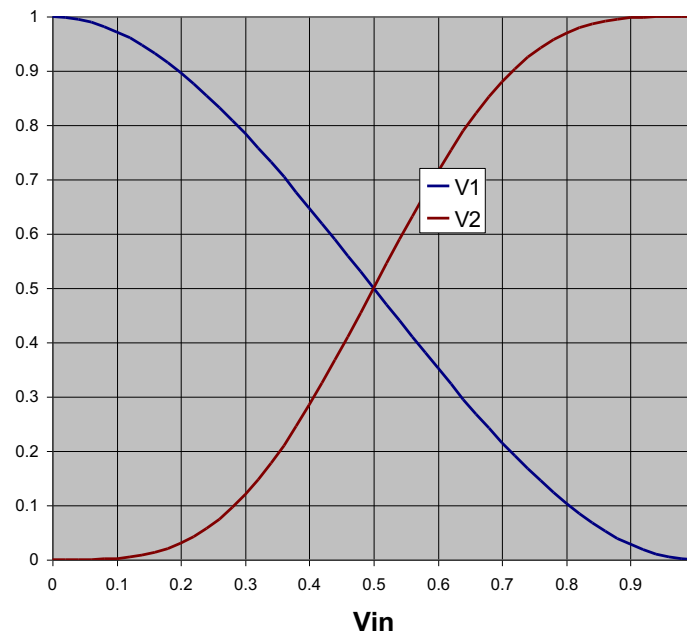
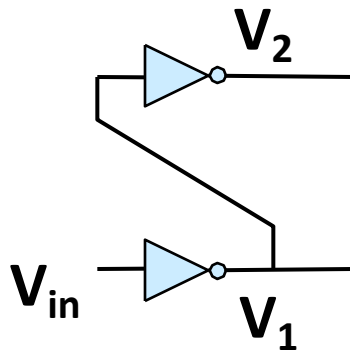
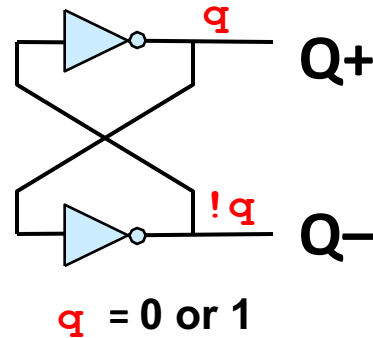
# Computation: Arithmetic Logic Unit



- Combinational logic
  - Continuously responding to inputs
- Control signal selects function computed
  - Corresponding to 4 arithmetic/logical operations in Y86-64
- Also computes values for condition codes

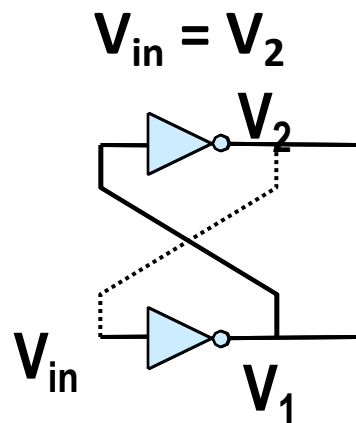
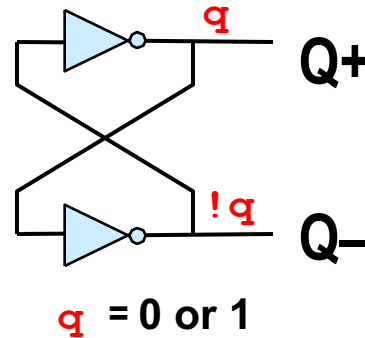
# Storage: Storing 1 Bit

## Bistable Element

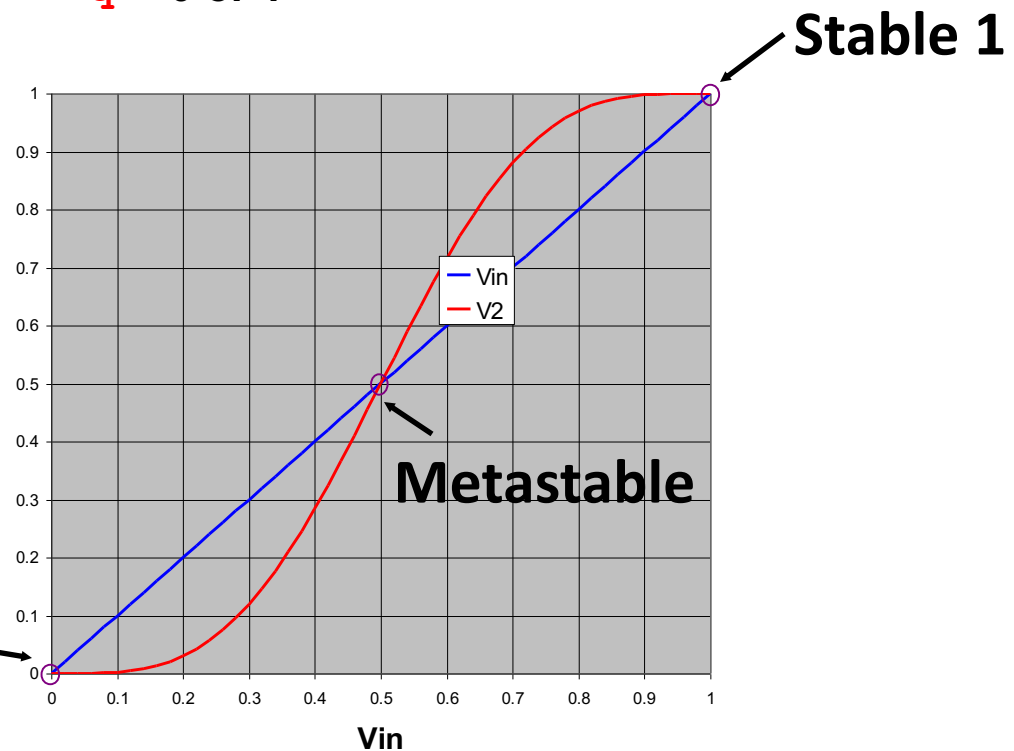


# Storage: Storing 1 Bit (cont.)

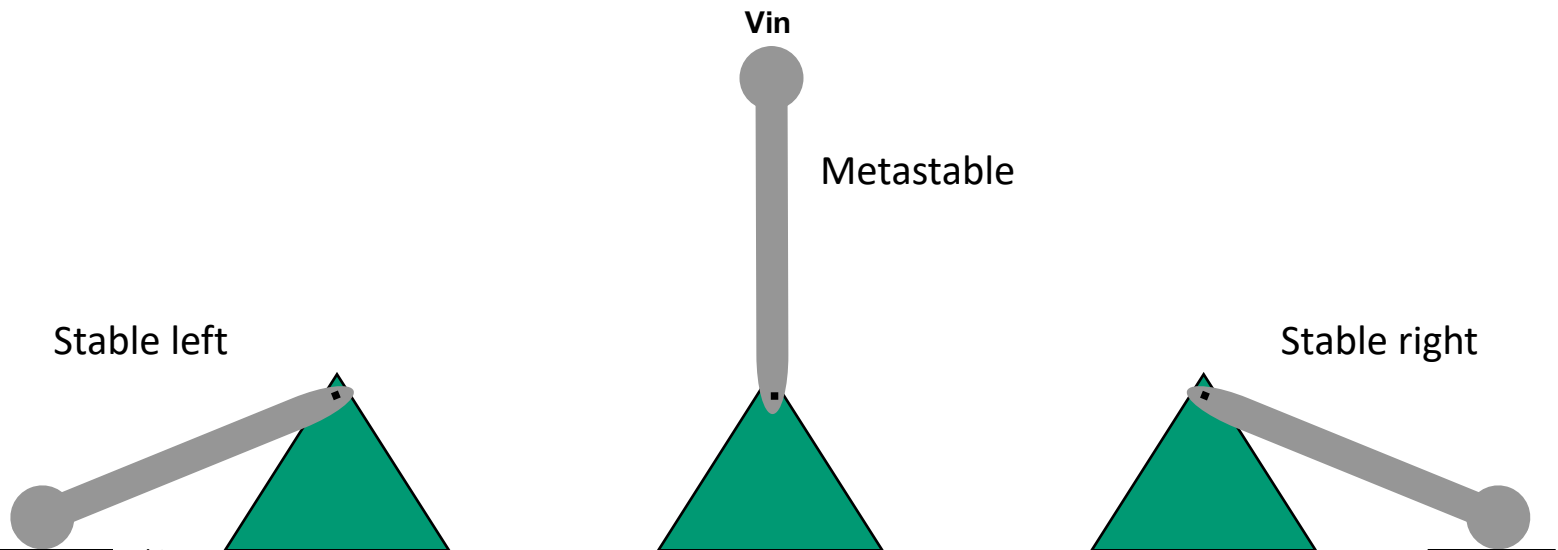
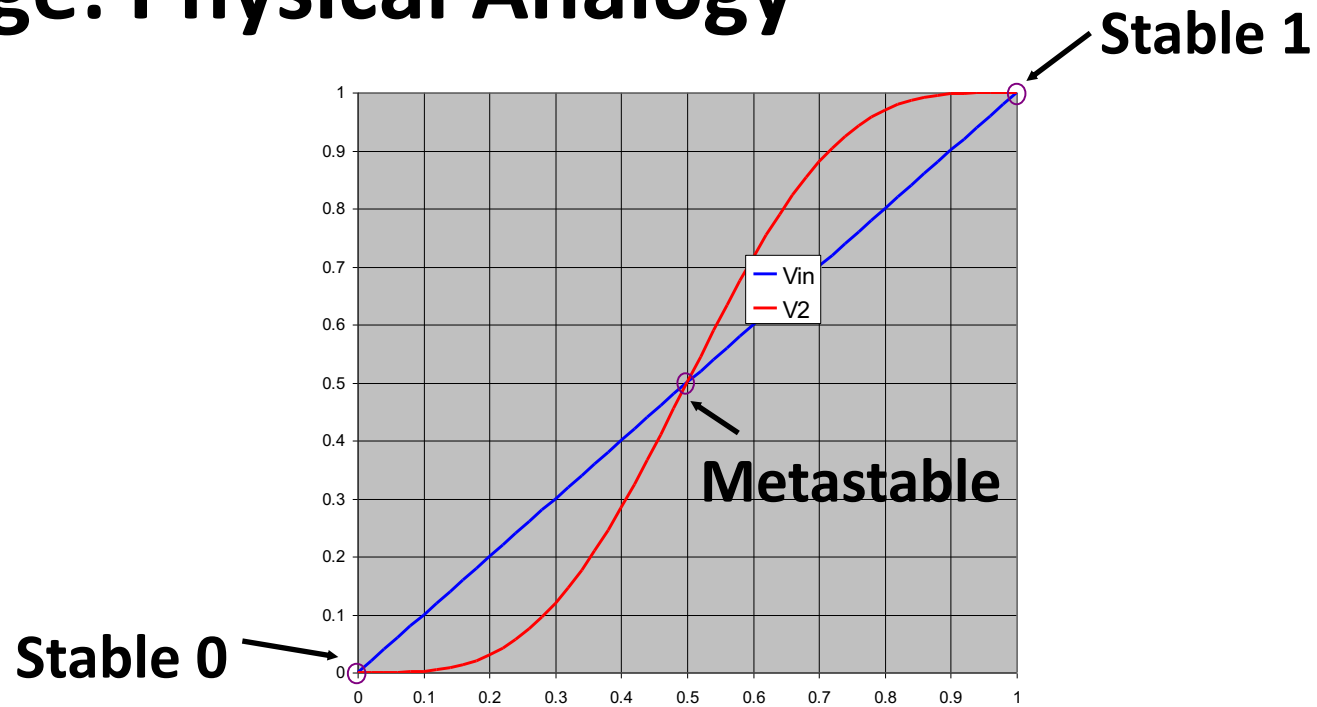
## Bistable Element



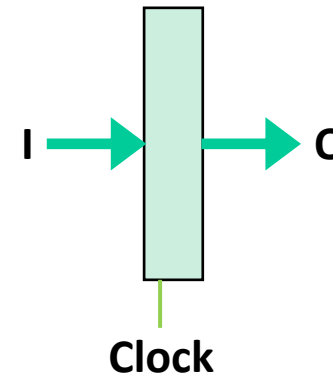
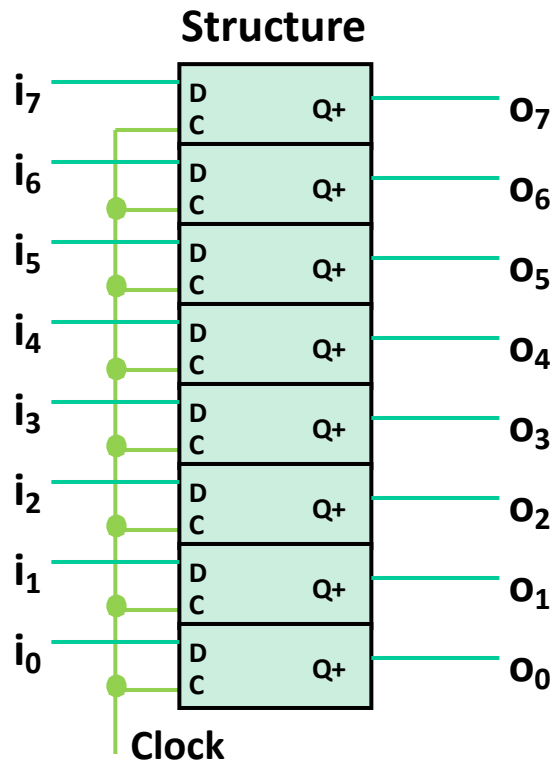
Stable 0



# Storage: Physical Analogy



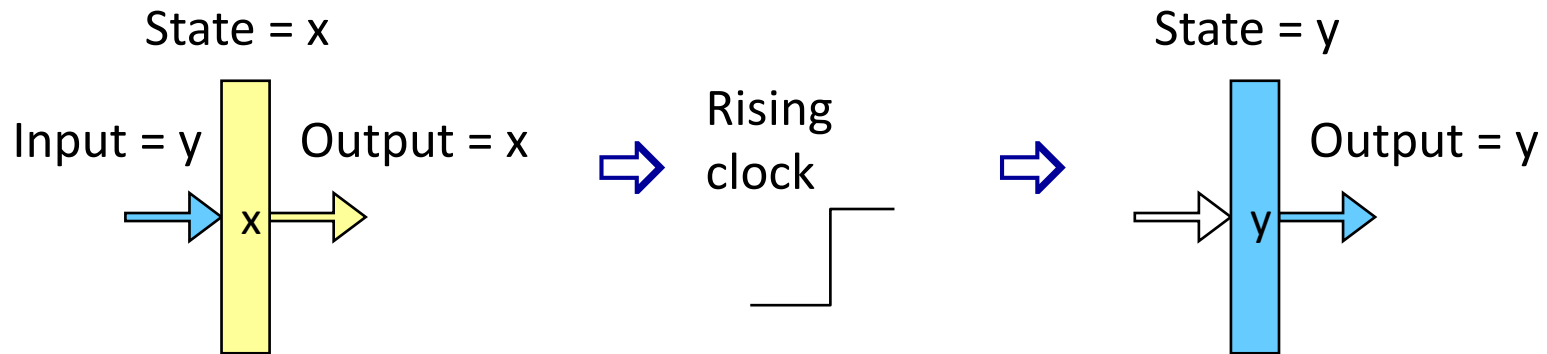
# Storage: Sequential Circuit - Registers



- Stores word of data (hardware registers)
  - (Slightly) different from *program registers* seen in assembly code
- Collection of edge-triggered latches (so-called flip-flops)
- Loads input on rising edge of clock

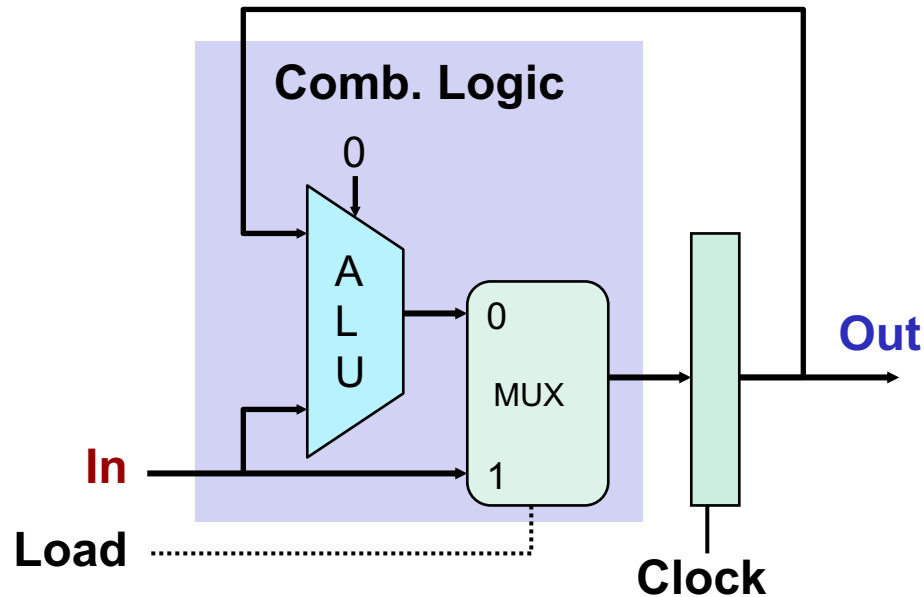


# Storage: Register Operation

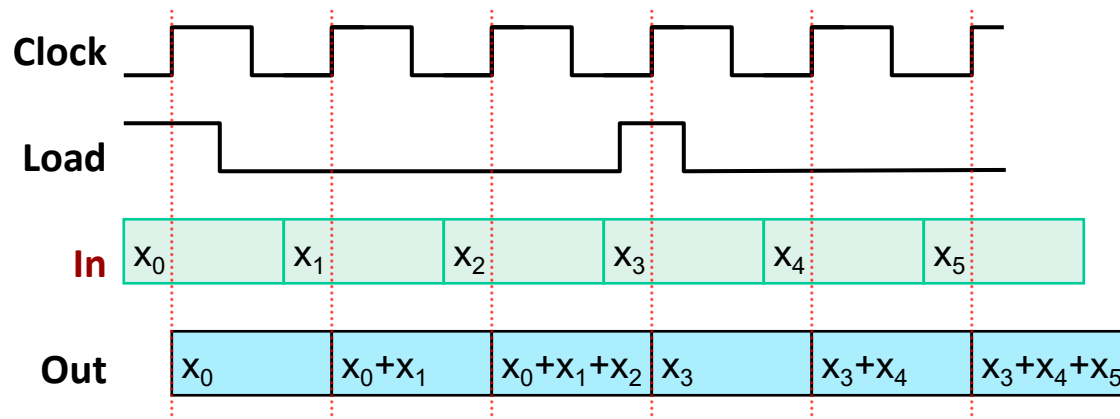


- Stores data bits
- For most of time acts as barrier between input and output
- As clock rises, loads input

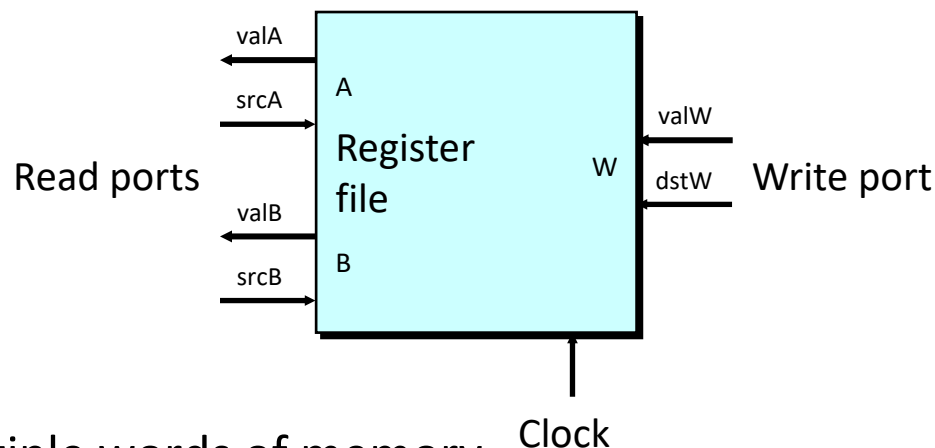
# Storage+Computation: State Machine Example



- Accumulator circuit
- Load or accumulate on each cycle

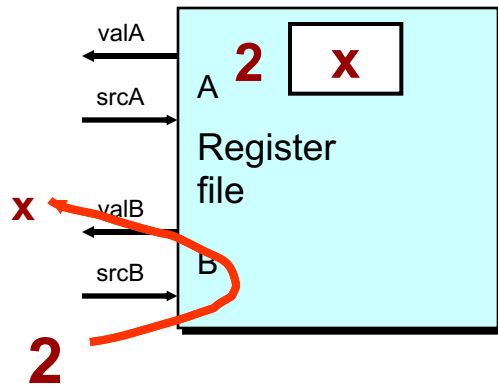


# Storage: Register File



- Stores multiple words of memory
  - Address input specifies which word to read or write
- Register file
  - Holds values of program registers
  - **%rax**, **%rsp**, etc.
  - Register identifier serves as address
    - ID 15 (0xF) implies no read or write performed
- **Multiple Ports**
  - Can read and/or write multiple words in one cycle
    - Each has separate address and data input/output

# Storage: Register File Timing

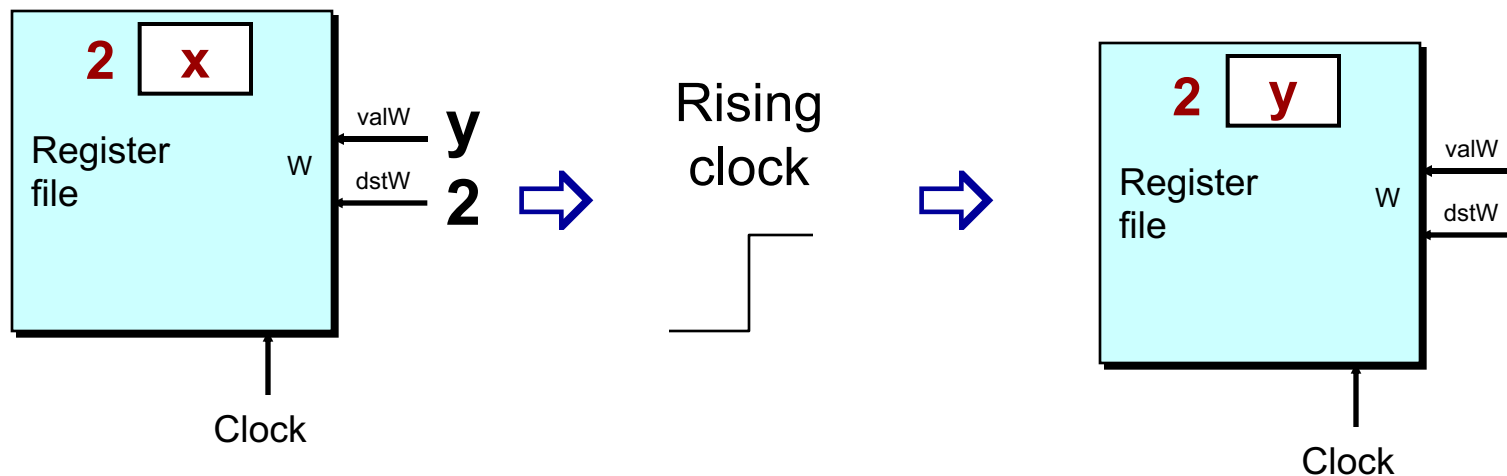


## ■ Reading

- Like combinational logic
- Output data generated based on input address
  - After some delay

## ■ Writing

- Like register
- Update only as clock rises



# Hardware Control Language

- Very simple hardware description language
- Can only express limited aspects of hardware operation
  - Parts we want to explore and modify

## ■ Data Types

- `bool`: Boolean
  - `a, b, c, ...`
- `int`: words
  - `A, B, C, ...`
  - Does not specify word size---bytes, 64-bit words, ...

## ■ Statements

- `bool a = bool-expr ;`
- `int A = int-expr ;`

# HCL Operations

- Classify by type of value returned

## ■ Boolean Expressions

- Logic Operations
  - `a && b, a || b, !a`
- Word Comparisons
  - `A == B, A != B, A < B, A <= B, A >= B, A > B`
- Set Membership
  - `A in { B, C, D }`
    - Same as `A == B || A == C || A == D`

## ■ Word Expressions

- Case expressions
  - `[ a : A; b : B; c : C ]`
  - Evaluate test expressions `a, b, c, ...` in sequence
  - Return word expression `A, B, C, ...` for first successful test

# Summary

## ■ Computation

- Performed by combinational logic
- Computes Boolean functions
- Continuously reacts to input changes

## ■ Storage

- Registers
  - Hold single words
  - Loaded as clock rises
- Register file (random-access memories)
  - Hold multiple words
  - Possible multiple read or write ports
  - Read word when address input changes
  - Write word as clock rises