# Logic Design and HCL

’20H2

송 인 식

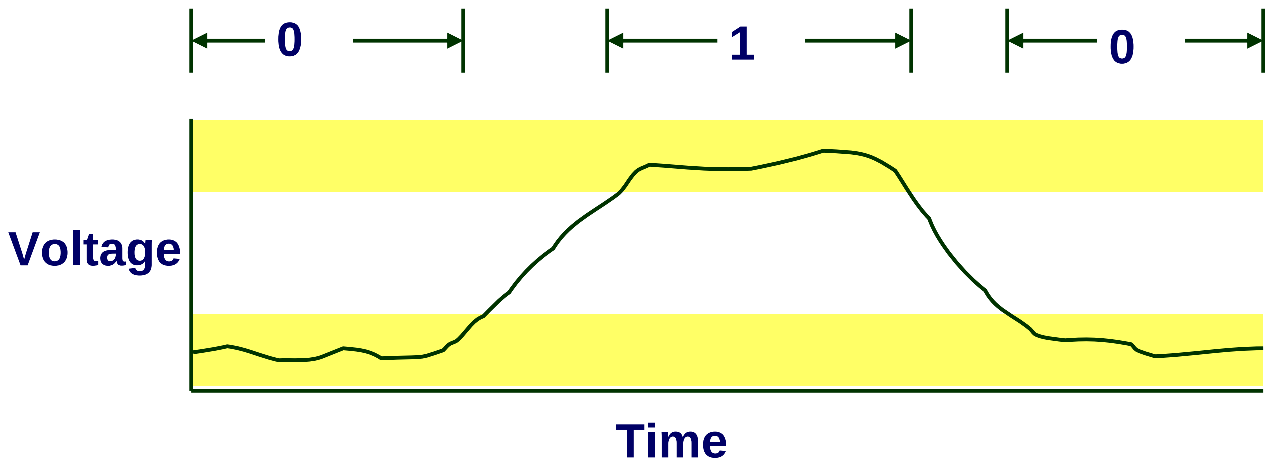
# Outline

- Combinational Logic
- Sequential Logic
- HCL(Hardware Control Language)

## Overview of Logic Design

- Fundamental Hardware Requirements
  - Computation
  - Storage
  - Communication
    - How to get values from one place to another
- Bits are Our Friends
  - Everything expressed in terms of values 0 and 1
  - Computation
    - Compute Boolean functions
  - Storage
    - Store bits of information
  - Communication
    - Low or high voltage on wire

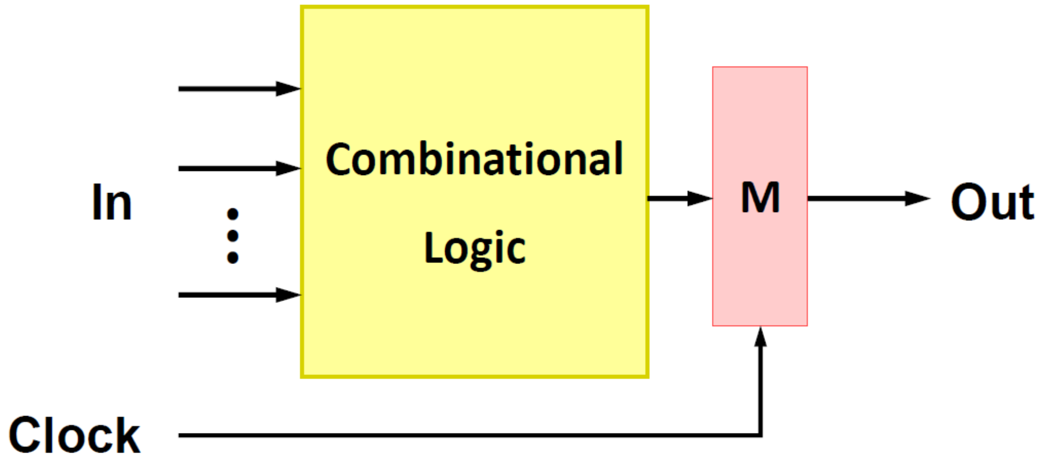
# Digital Signals



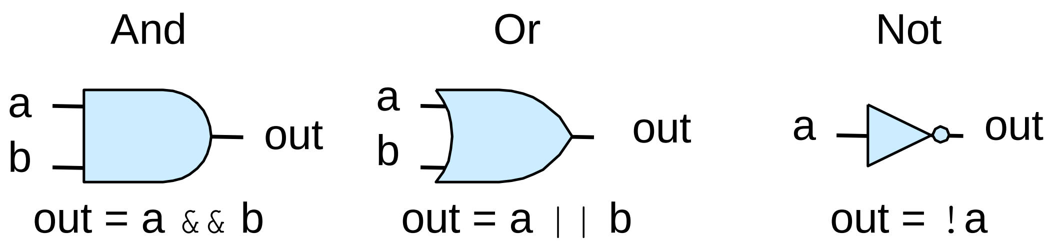
- Use voltage thresholds to extract discrete values from continuous signal
- Simplest version: 1-bit signal
  - Either high range (1) or low range (0)
  - With guard range between them
- Not strongly affected by noise or low quality circuit elements
  - Can make circuits simple, small, and fast

# Digital Systems

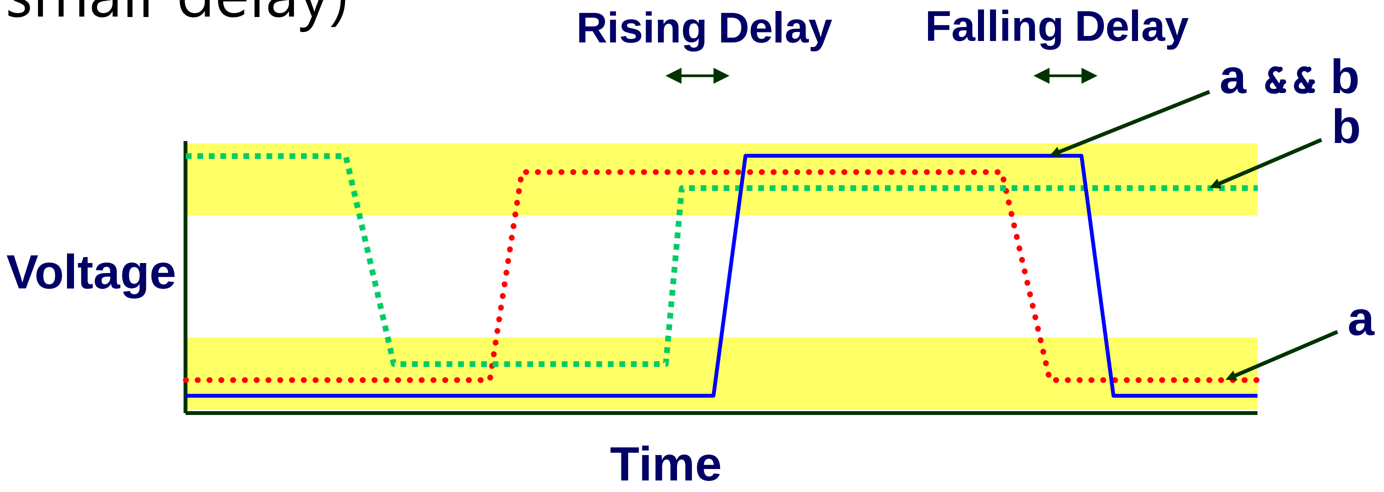
- Three components required to implement a digital system
  - Combinational logic to compute Boolean functions
  - Memory elements to store bits
  - Clock signals to regulate the updating of the memory elements



# Computing with Logic Gates

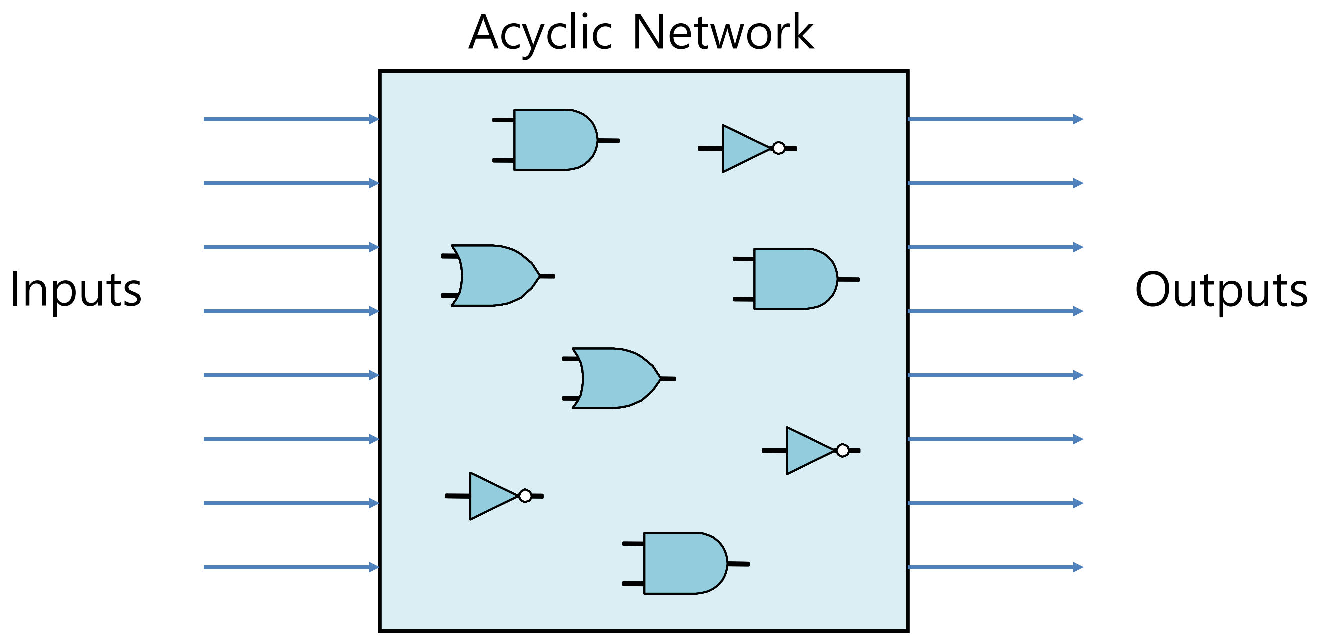


- Outputs are Boolean functions of inputs
- Respond continuously to changes in inputs (with some, small delay)

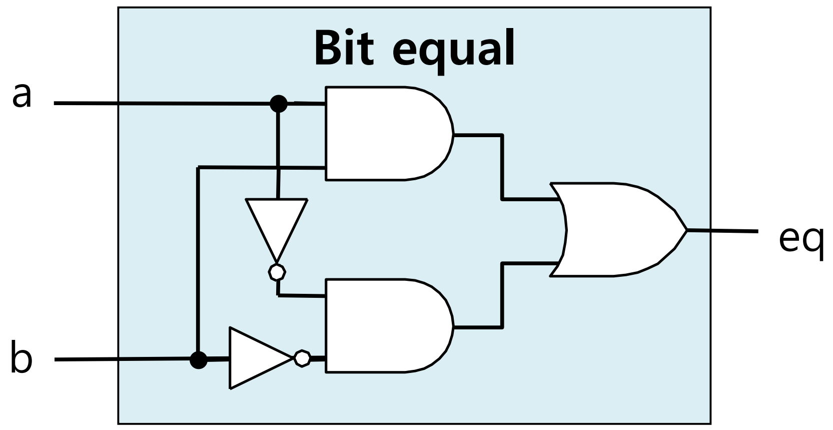


## Combinational Circuits

- Acyclic Network of Logic Gates
  - Continuously responds to changes on inputs
  - Outputs become (after some delay) Boolean functions of inputs



## Bit Equality

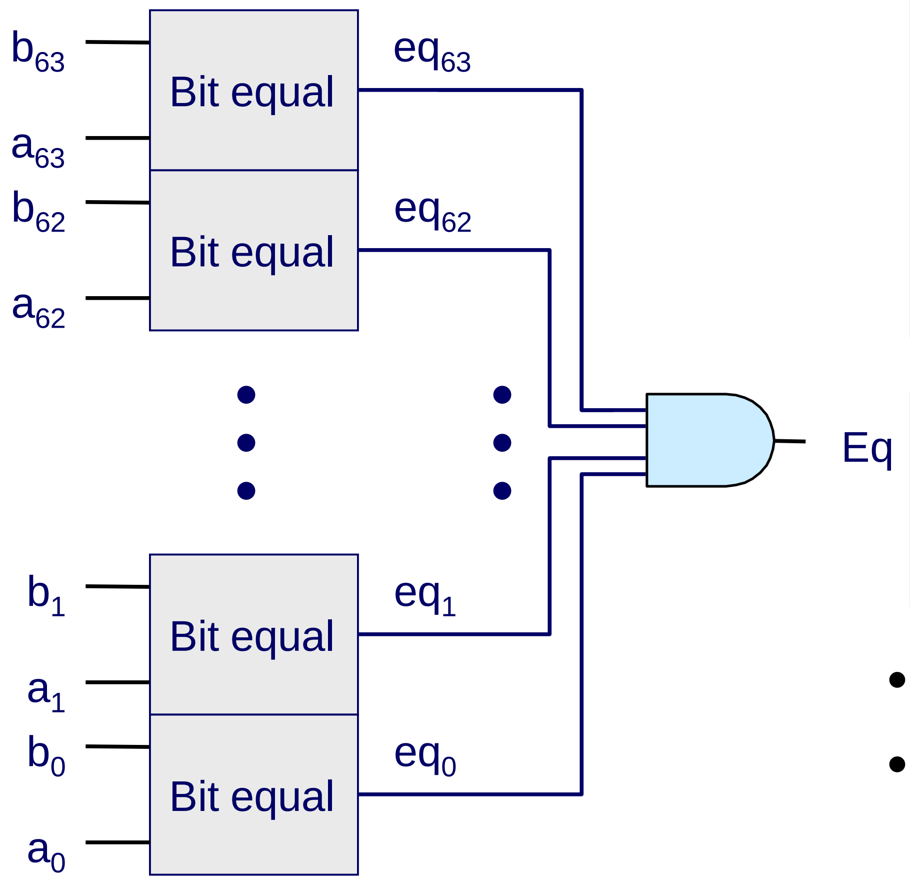


HCL Expression

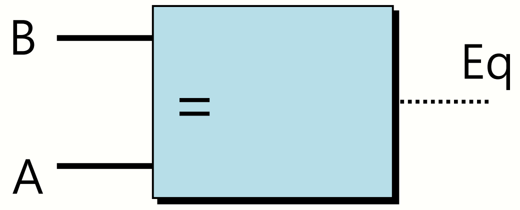
```
bool eq = (a&&b)||(!a&&!b)
```

  - 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

# Word Equality



**Word-Level Representation**

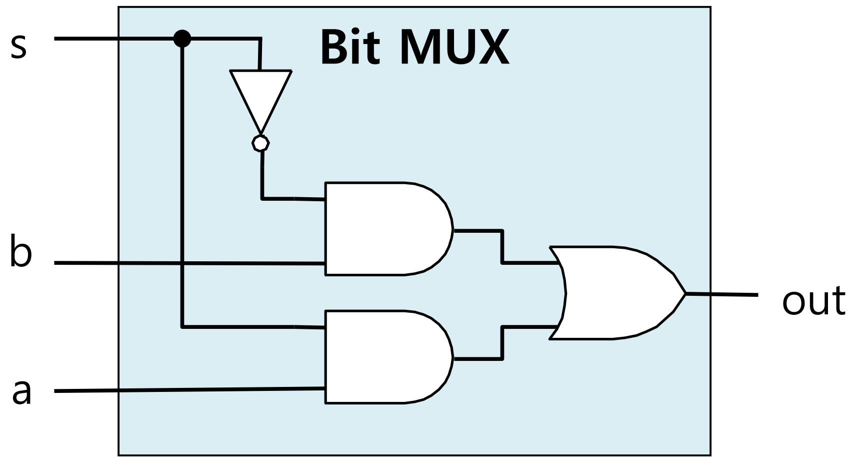


HCL Representation

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

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

# Bit-Level Multiplexor

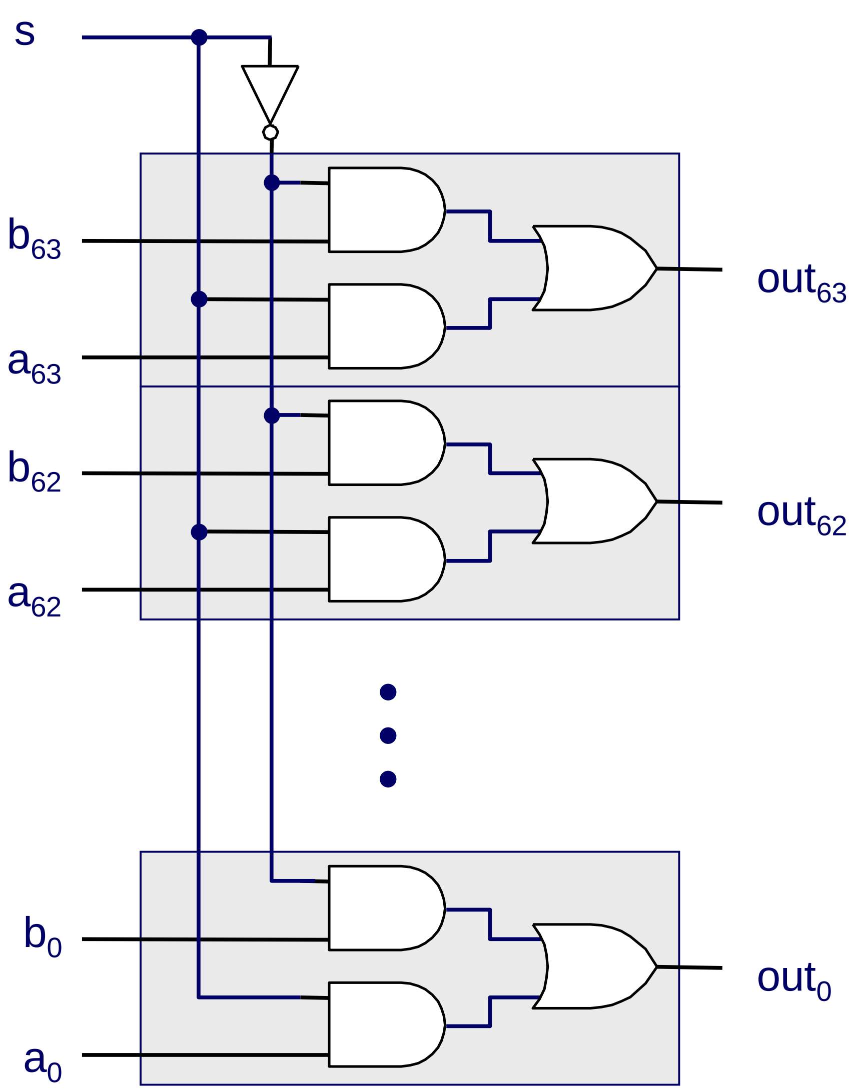


HCL Expression

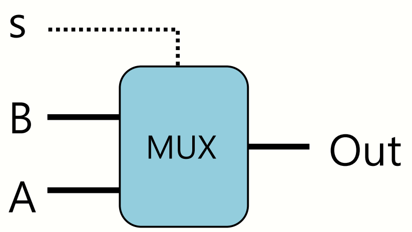
```
bool out = (s&&a)||(!s&&b)
```

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

# Word Multiplexor



Word-Level Representation



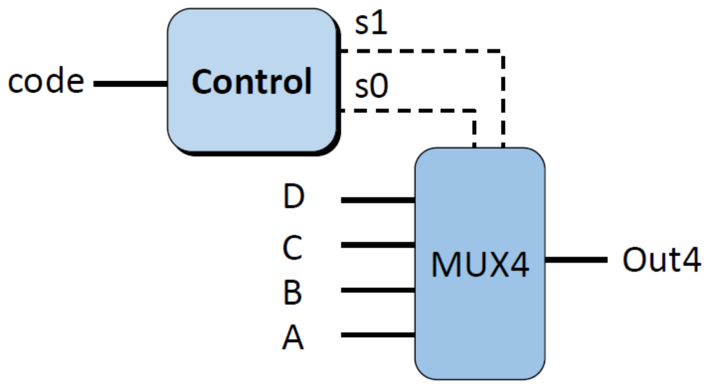
HCL Representation

```
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

# Set Membership

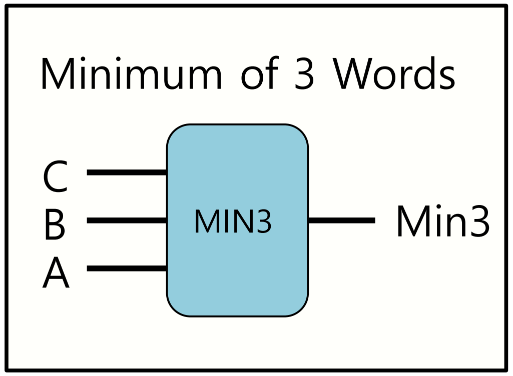
- HCL representation
  - Input code
  - Output signal s0, s1
  - s1 is 1 when code is in the set {2, 3}
  - s0 is 1 when code is in the set {1, 3}



**HCL Expression**

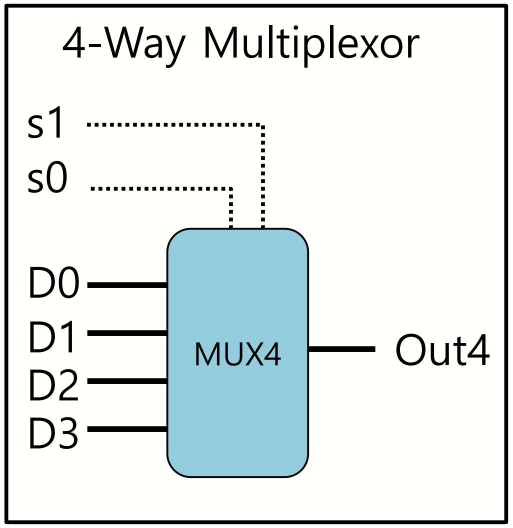
```
bool s1 = code == 2 || code == 3;
bool s0 = code == 1 || code == 3;


bool s1 = code in {2, 3};
bool s0 = code in {1, 3};
```

# HCL Word-Level Examples



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

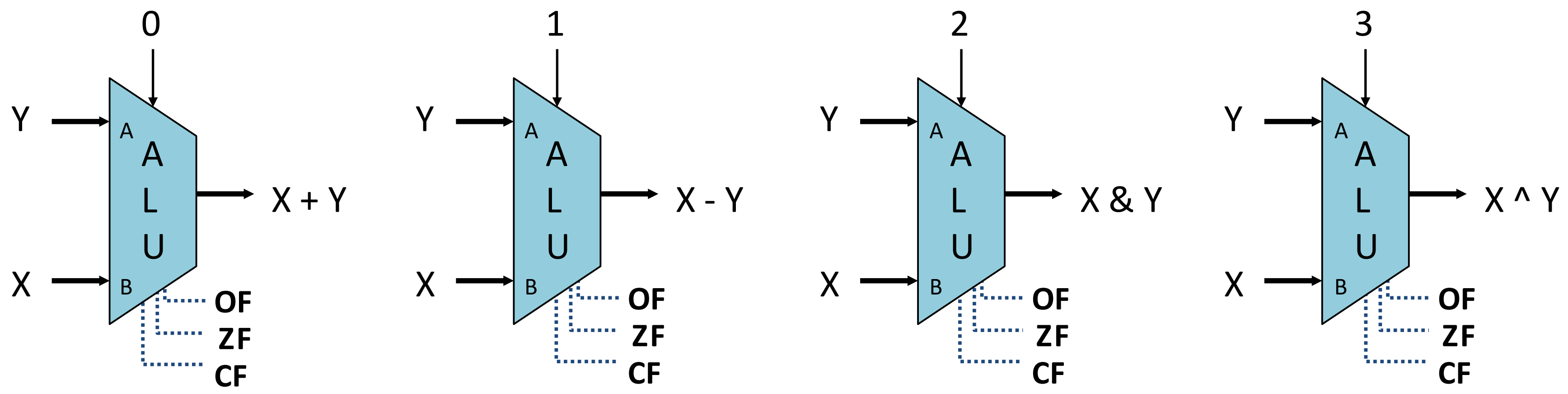
- Find minimum of three input words
- HCL case expression
- Final case guarantees match



```
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

# Arithmetic Logic Unit



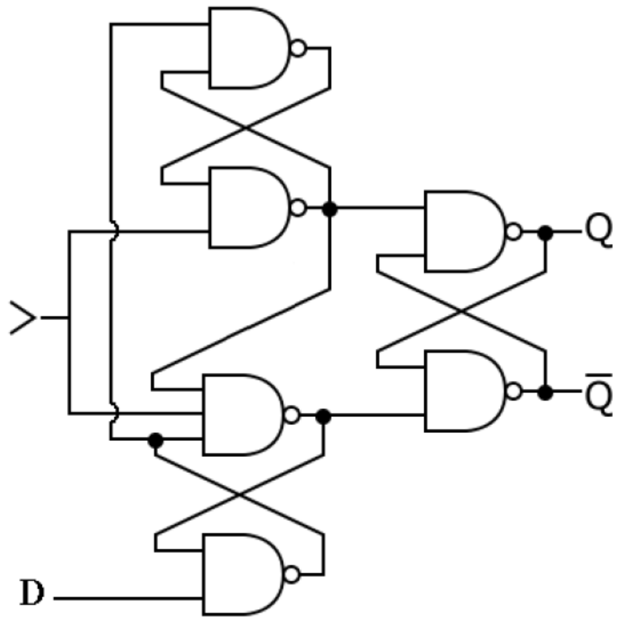
- Combinational logic
  - Continuously responding to inputs
- Control signal selects function computed
  - Corresponding to 4 arithmetic/logical operations in Y86-64
- Also computes values for condition codes

# Outline

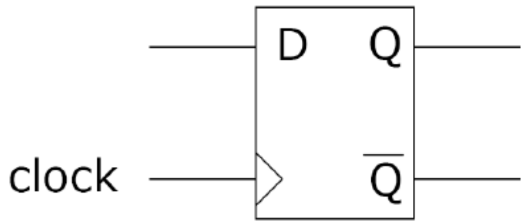
- Combinational Logic
- Sequential Logic
- HCL(Hardware Control Language)

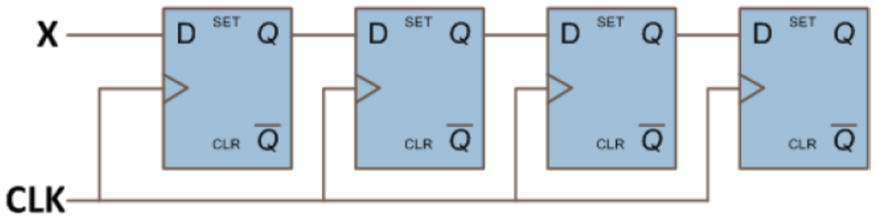
# Sequential Logic

- Flip-flops

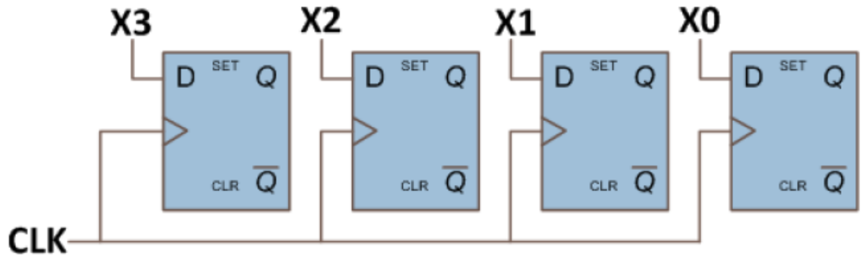


Edge triggered D flip-flop



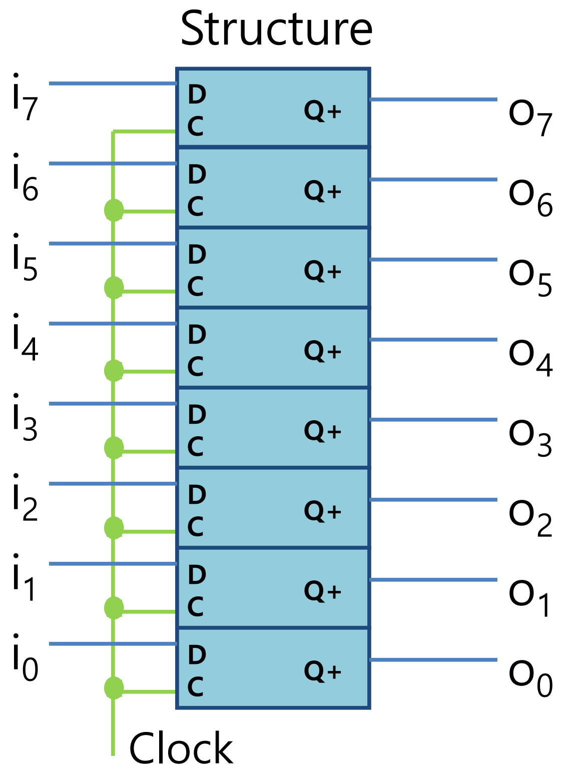


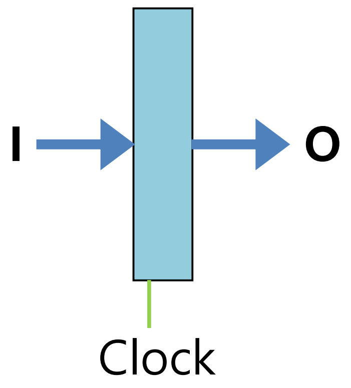
Shifter



4-bit register

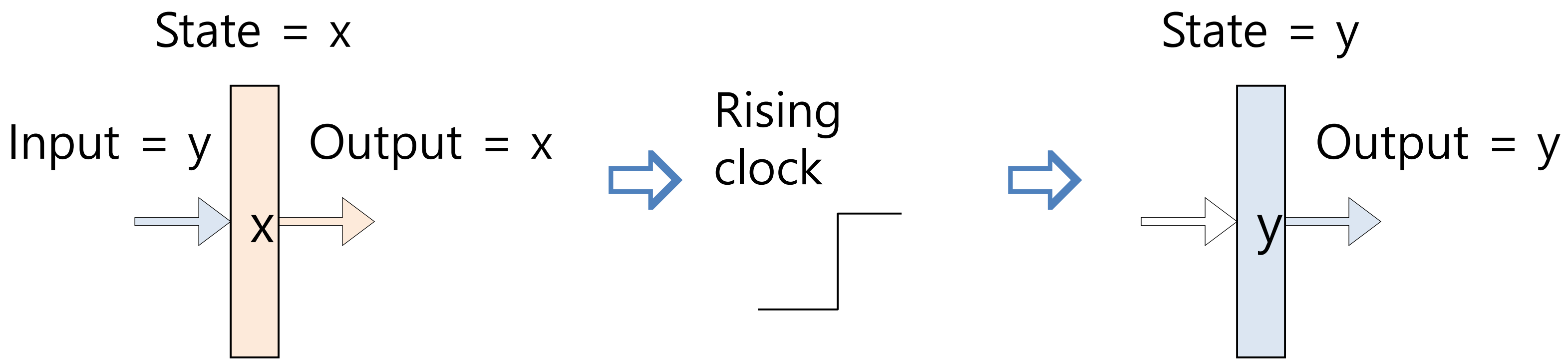
# Sequential Circuit: Registers





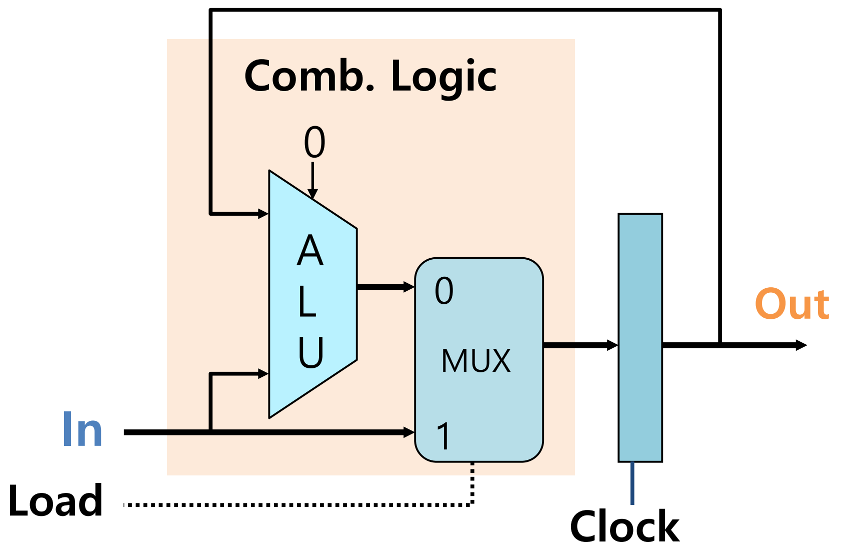
- Stores word of data
  - Different from *program registers* seen in assembly code
- Collection of edge-triggered latches
- Loads input on rising edge of clock

# Register Operation

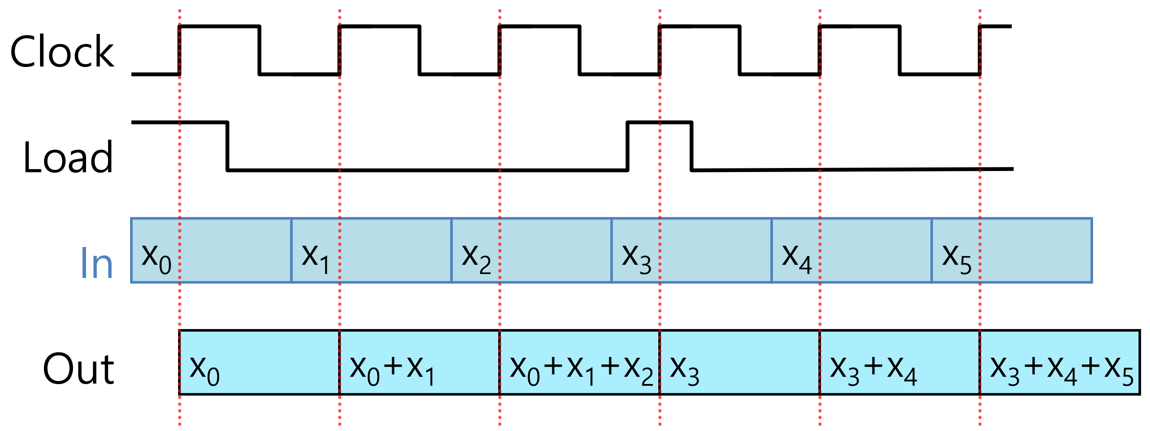


- Stores data bits
- For most of time acts as barrier between input and output
- As clock rises, loads input

# State Machine Example

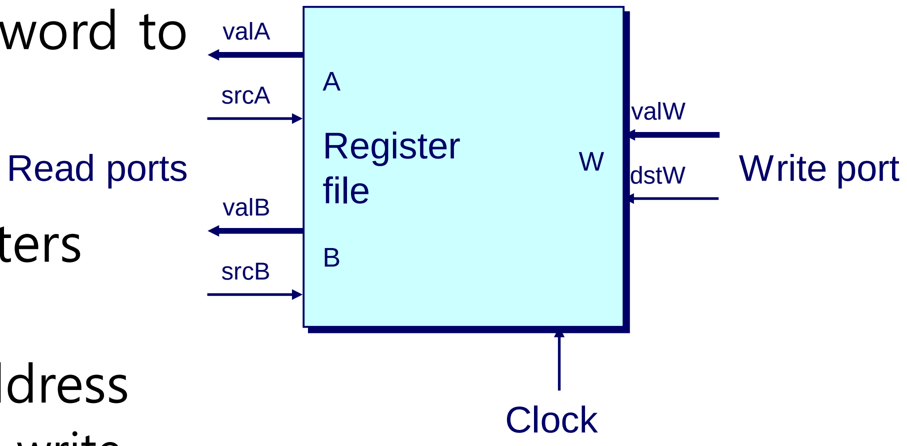


- Accumulator circuit
- Load or accumulate on each cycle

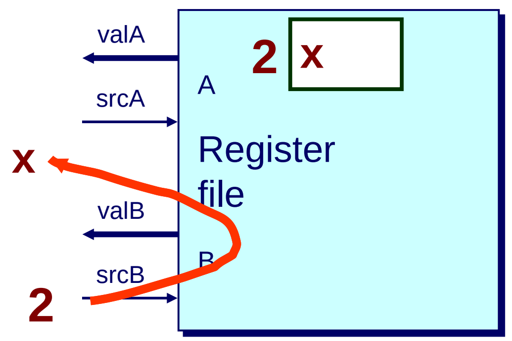


# Register File

- 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



# Register File Timing

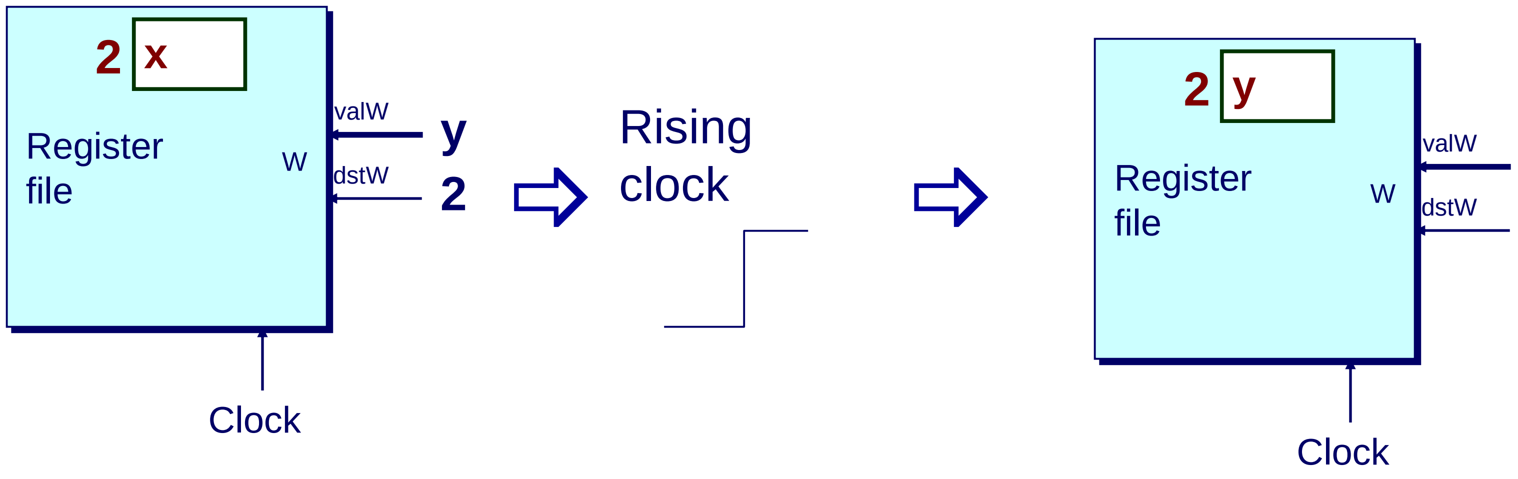


## Reading

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

## Writing

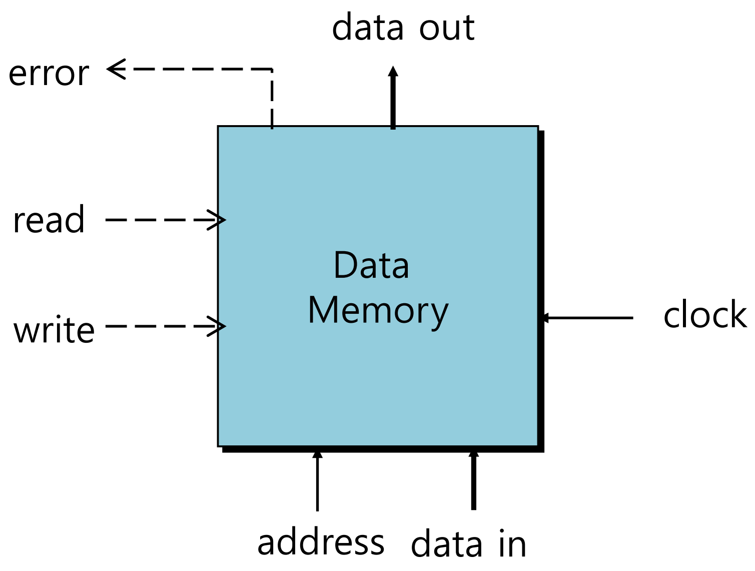
- Like register
- Update only as clock rises



# Data Memory

- Hold instructions and data for program
- Read or write, one at a time
- Read is like combinational logic (like register file)
- Write is performed only at the rising clock edge
- Error signal will be set if the address is out of range

- Another read-only memory for instructions
- Dual-port memory
  - One read port for instructions, another read or write port for data



## Outline

- Combinational Logic
- Sequential Logic
- HCL(Hardware Control Language)

# Hardware Control Language

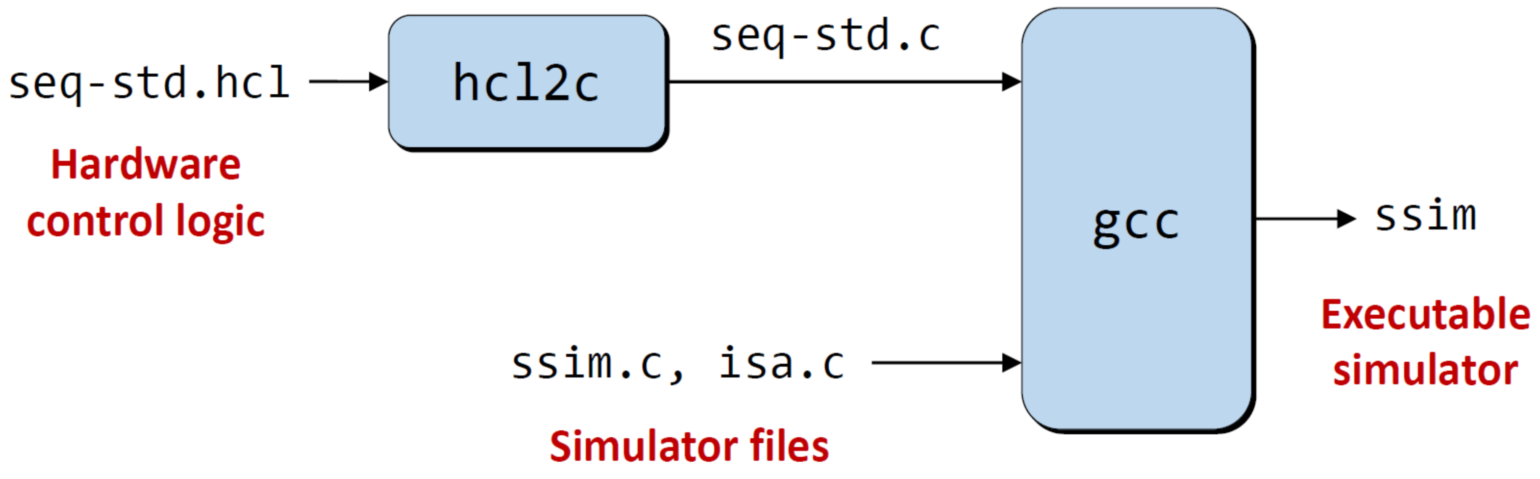
- Very simple hardware description language
- Can only express limited aspects of hardware operation
- More info at http://csapp.cs.cmu.edu/3e/waside/waside-hcl.pdf

- Data Types
  - `bool`: Boolean
    - `a`, `b`, `c`, …
  - `int`: words
    - `A`, `B`, `C`, …
    - Does not specify word size---bytes, 32-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

# Using HCL


seq-std.hcl
Hardware control logic
hcl2c
seq-std.c
gcc
ssim
Executable simulator
ssim.c, isa.c
Simulator files

## Summary

- Computation
  - Performed by combinational logic
  - Computes Boolean functions
  - Continuously reacts to input changes
- Storage
  - Registers
    - Hold single words, Loaded as clock rises
  - Random-access memories
    - Hold multiple words
    - Possibly multiple read or write ports
    - Read word when address input changes
    - Write word as clock rises

# Questions?