CS 270: Computer Organization

Logic Design

Instructor:

Professor Stephen P. Carl

Overview of Logic Design

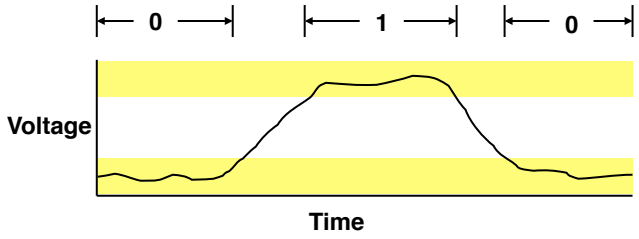
Fundamental Hardware Requirements

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

Bits are Our Friends

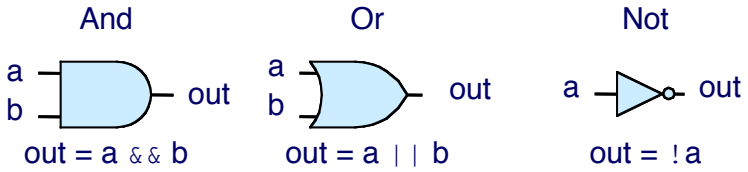
- Everything is expressed in terms of values 0 and 1
- Communication: low or high voltage on a wire
- Computation: compute fundamental Boolean functions
- Storage: Store bits of information

Digital Signals

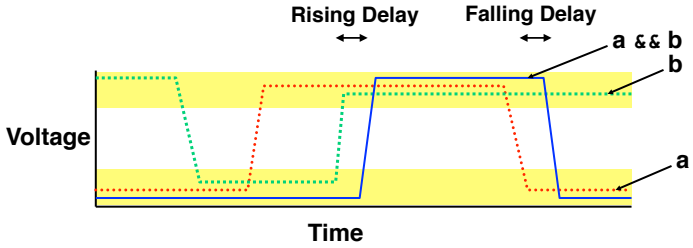


- Use voltage thresholds to extract discrete values from continuous signal
- Simplest version: 1-bit signal
  - Either high range (1) or low range (0)
  - With guard range between them
- Not strongly affected by noise or low quality circuit elements
  - Can make circuits simple, small, and fast

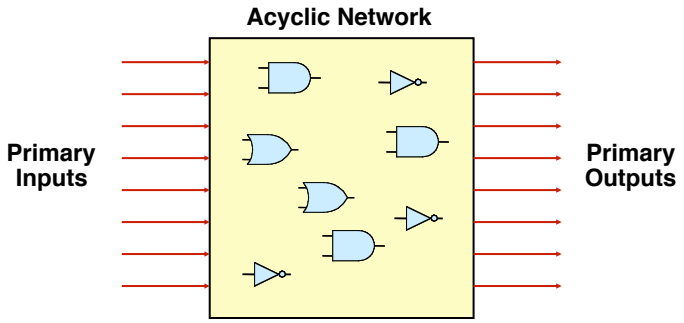
Computing with Logic Gates



- Gate outputs are Boolean functions of inputs - thanks, Claude Shannon!
- Gates respond continuously to changes in inputs
  (With some, small delay)



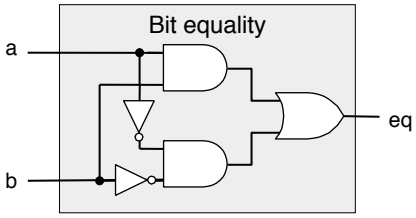
Combinational Circuits



A *combinational circuit* is an acyclic network of logic gates

- Outputs continuously respond to changes on primary inputs
- After some delay, the primary outputs become Boolean functions of primary inputs

Bit Equality



HCL Expression:

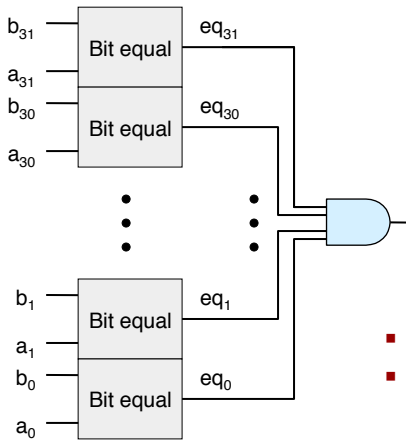
```
bool eq = (a&&b)||(!a&&!b)
```

This circuit generates a 1 as output if **a** and **b** are equal, 0 otherwise

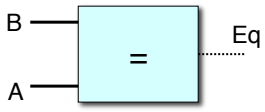
Book introduces Hardware Control Language (HCL)

- Very simple hardware description language, in which boolean operations have syntax similar to C logical operations
- You should be able to read it, but we won't write it
- Used to describe control logic for processors

Word Equality



Word-Level Representation

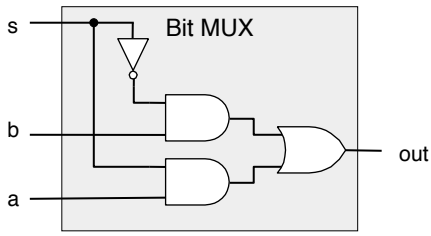


HCL Representation

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

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

Bit-Level Multiplexor

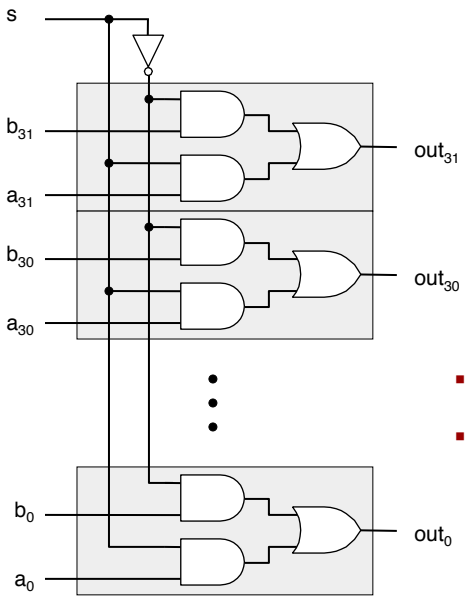


HCL Expression

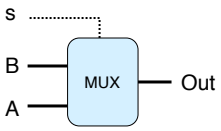
```
bool out = (s&&a)||(!s&&b)
```

- S is the *control signal*
- a and b are *data signals*
- Multiplexor outputs a when s=1, b when s=0

Word Multiplexor



Word-Level Representation

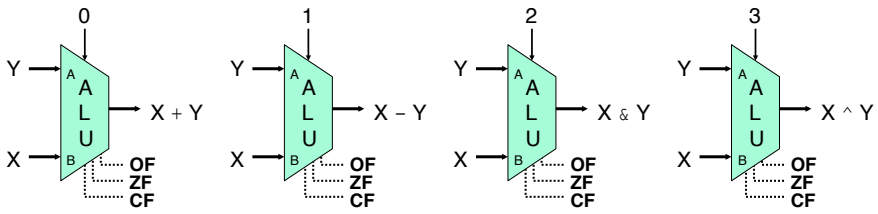


HCL Representation

```
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

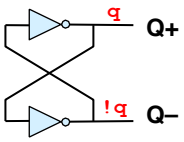
Arithmetic Logic Unit



- ALU is made up of Combinational logic
  (recall: such logic continuously responds to inputs)
- Control signal selects function computed
  (this corresponds to the 4 arithmetic/logical operations in Y86)
- Also computes values for condition codes

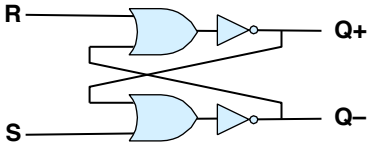
Storing and Accessing 1 Bit

Bistable Element

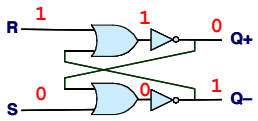


q = 0 or 1

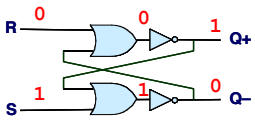
R-S Latch



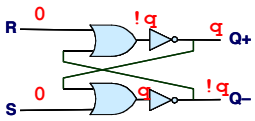
Resetting



Setting

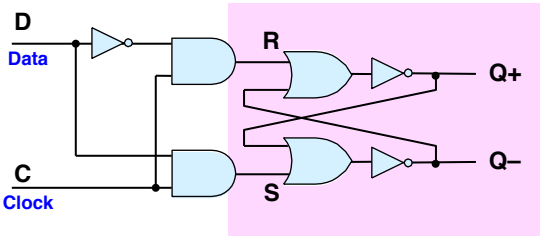


Storing

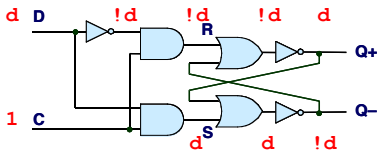


1-Bit Latch

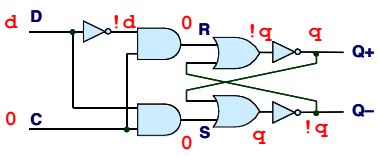
D Latch



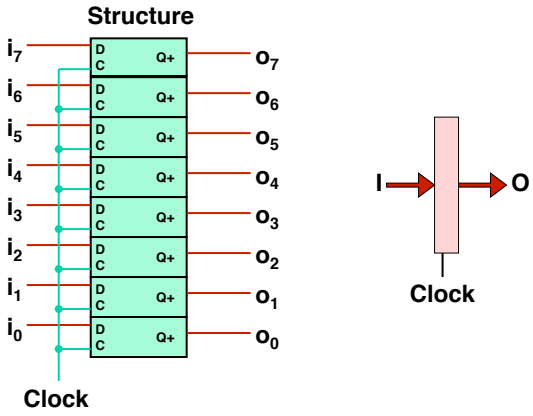
Latching



Storing

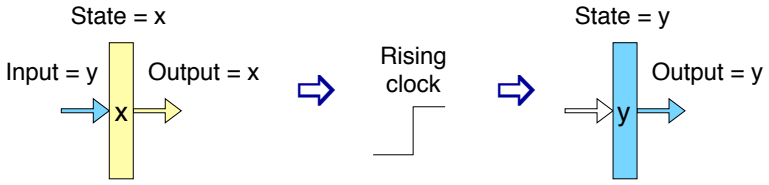


Registers



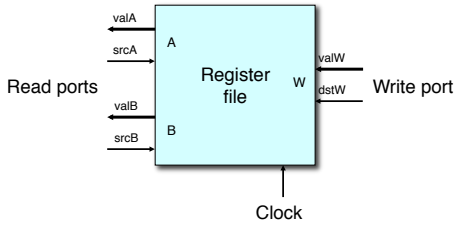
- A *register* stores a word of data
  (note: this is different from *program registers* seen in assembly code)
- Collection of latches
- Loads input on rising edge of clock

Register Operation



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

Random-Access Memory



- Stores multiple words of memory
  - Address input specifies which word to read or write
- Register file
  - Holds values of program registers
  - `%eax`, `%esp`, etc.
  - Register identifier serves as address
- Multiple Ports
  - Can read and/or write multiple words in one cycle
  - Each has separate address and data input/output

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
    - Possible multiple read or write ports
    - Read word when address input changes
    - Write word as clock rises