CS429: Computer Organization and Architecture

Logic Design

Warren Hunt, Jr. and Bill Young
Department of Computer Sciences
University of Texas at Austin

Last updated: October 27, 2014 at 08:01

Topics of this Slideset

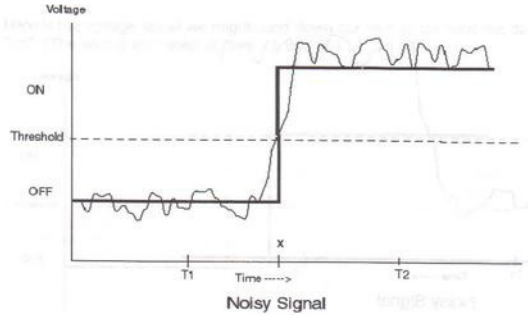
To execute a program we need:

- Communications: getting data from one place to another
- Computation: perform arithmetic or logical operations
- Memory: store the program, variables, results

Everything is expressed in terms of bits (0s and 1s).

- Communication
- Low or high voltage on a wire
- Computation
- Compute boolean functions
- Storage
- Store bits

Digital Signals



- Use voltage thesholds to extract discrete values from a continuous signal: works with light for communication.
- Simplest version: 1-bit signal
    - Either high range (1) or low range (0)
    - With a guard range between them.
- Not strongly affected by noise or low-quality elements; circuits are simple, small and fast.

Truth Tables

And: `A & B` = 1 when both A = 1 and B = 1.

0	1	&
0	0	0
0	1	0
1	0	0
1	1	1

Or: `A | B` = 1 when either A = 1 or B = 1.

<table>
<tr><th>0</th><th>1</th><th>|</th></tr>
<tr><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>1</td></tr>
</table>

Not: `~A` = 1 when A = 0.

0	~
0	1
1	0

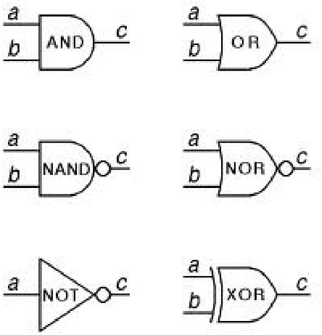
Xor: `A ^ B` = 1 when either A = 1 or B = 1, but not both.

0	1	^
0	0	0
0	1	1
1	0	1
1	1	0

Computing with Logic Gates

How are these logic functions actually computed in hardware?

- Logic gates are constructed from transistors.
- The output is a boolean function of inputs.
- The gate responds continuously to changes in input with a small delay.



How many of these do you really need?

Sets of Logic Gates

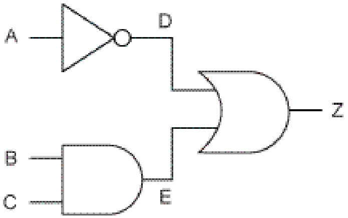
It's pretty easy to see that any boolean function can be implemented with AND, OR and NOT. Why? We call that a *functionally complete* set of gates.

You can get by with fewer gates. How would you show each of the following?

- AND and NOT is complete.
- OR and NOT is complete.
- NAND is complete.
- NOR is complete.
- AND alone is not complete.
- OR alone is not complete.

Often circuit designers will restrict themselves to a small subset of gates (e.g., just NAND gates). Why would they do that?

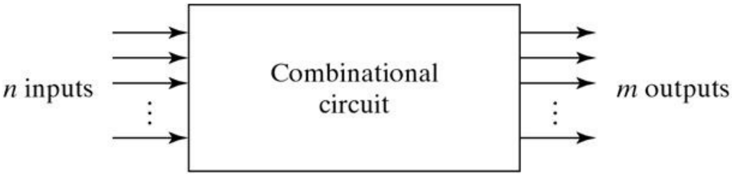
Simple boolean functions are implemented by "logic gates"; more complex functions, by combinations of gates.



A	B	C	Z
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

In C: `Z = !A || (B && C);`

Combinational Circuits

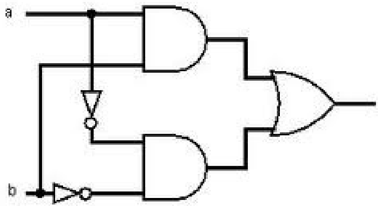


The box contains an acyclic network of logic gates.

- Continuously responds to changes in inputs.
- Outputs become (after a short delay) boolean functions of the inputs.

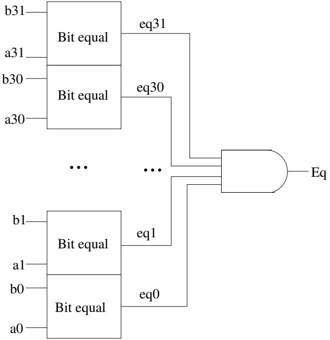
Bit Equality

The following circuit generates a 1 if a and b are equal.

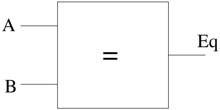


In C: `int eq = (a&&b) || (!a&&!b);`

Word Equality

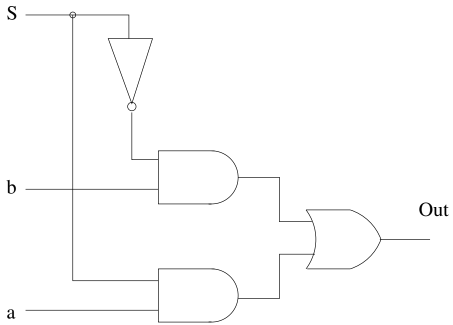


Word-level representation:



- Equality operation
- Generates Boolean value

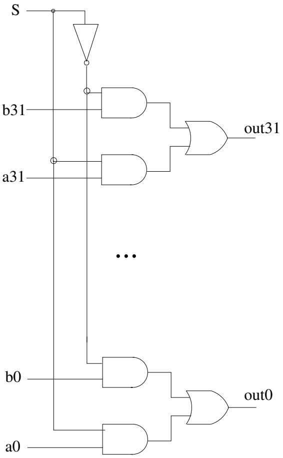
Bit Multiplexor



In C: `int out = (s && a) || (!s && b);`

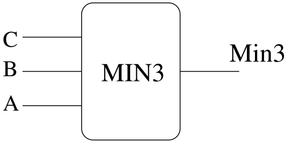
- Control signal s selects between two inputs a and b.
- Output is a when `s == 1`, and b otherwise.

Word Multiplexor

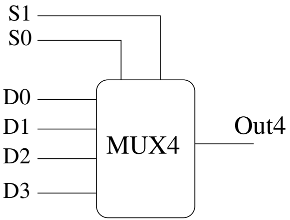


Select input word A or B depending on control signal S.

Minimum of 3 words



4–way Multiplexor

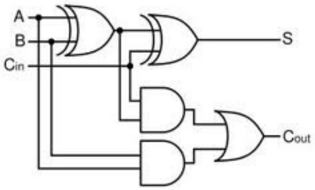


An ALU is an Arithmetic Logic Unit

- Multiple functions: `add`, `subtract`, `and`, `xor`, others
- Combinational logic to perform functions.
- Control signals select function to be performed.
- Modular: multiple instances of 1-bit ALU

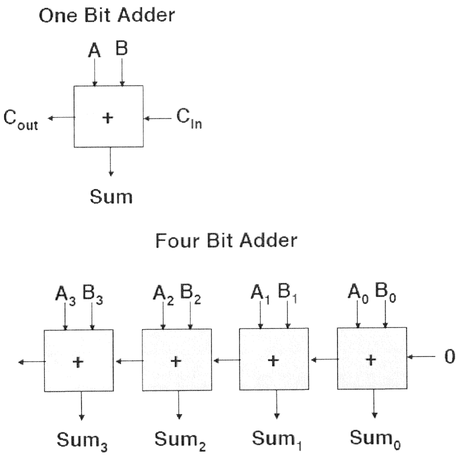
1-Bit Adder

The following circuit is a 1-bit adder:



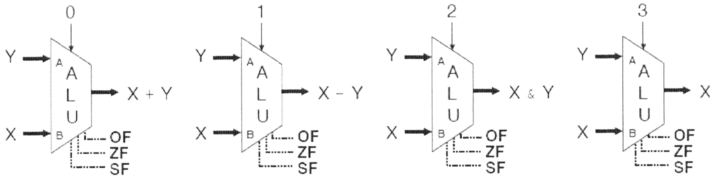
A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Adding a Pair of 4-bit Ints

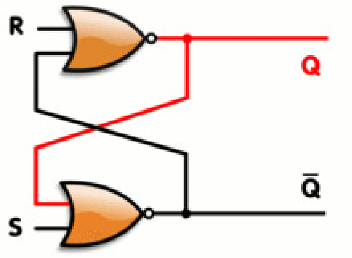


How do you subtract? How do you multiply?

A 4-bit ALU



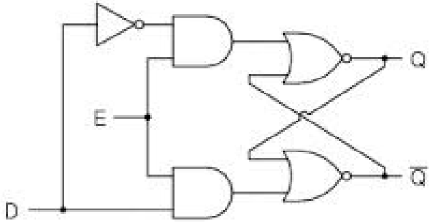
- Combinational logic: continuously responding to inputs.
- Control signal selects function computed: corresponding to the 4 arithmetic/logical operations in Y86.
- Also computes values of condition codes:
  - OF: overflow flag
  - ZF: zero flag
  - SF: sign flag



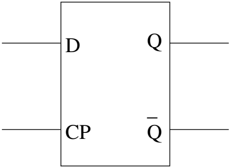
Characteristic table

S	R	Qnext	Action
0	0	Q	hold state
0	1	0	reset
1	0	1	set
1	1	X	not allowed

Gated D Latch: Store and Accesss One Bit



Higher level representation



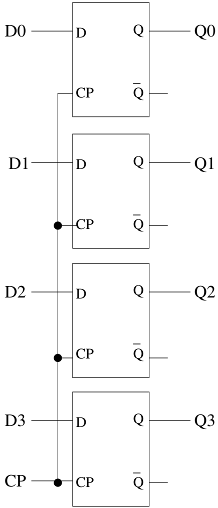
D Latch Truth table

E/C	D	Q	$\overline{Q}$	Comment
0	X	Q	$\overline{Q}$	No change
1	0	0	1	Reset
1	1	1	0	Set

A 4-bit Register

4 D latches:

- All share the E (aka WE or Write Enable) input
- D0–D3 are the data input
- Q0–Q3 are the output

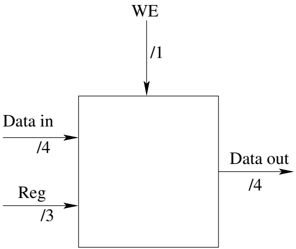


Register File Abstraction

Register file provides the CPU with temporary, fast storage.

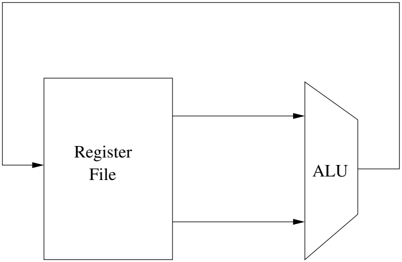
- N registers.
- Each of K bits.
- L output ports.

Suppose we want eight 4-bit registers and one output port.



Write Enable (WE) must be held at "1" long enough to allow:

- Data to be read;
- Operation (e.g., addition) to be performed;
- Result to be stored in target register.



Edge Triggered Flip Flops

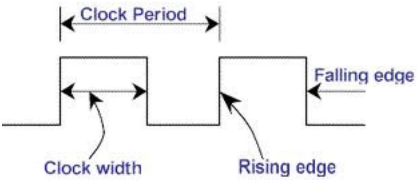
An edge-triggered flip-flop changes states either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse on the control input.

- A register is made up of several flip flops, each providing storage and access for an individual bit.
- A register file is made up of several registers and control logic

Clocking

The clock acts to enforce timing control on the chip.

- An integral part of every synchronous system.
- Can be global



Clock Frequency = 1 / clock period

- Measured in cycles per second (Hertz)
- 1 KHz = 1000 cycles / second
- 1ns (10^{-9} seconds) = 1GHz (10^{9}) clock frequency
- Higher frequency means faster speed.

Random Access Memory (RAM)

Stores many words

- Conceptually, a large array where each row is uniquely addressable.
- In reality, much more complex to increase throughput.
- Multiple chips and banks, interleaved, with multi-word operations.

Many implementations

- Dynamic (DRAM) is large, inexpensive, but relatively slow.
    - 1 transistor and 1 capacitor per bit.
    - Reads are destructive.
    - Requires periodic refresh.
    - Access time takes hundreds of CPU cycles.
- Static (SRAM) is fast but expensive.
    - 6 transistors per bit.
    - Streaming orientation.

Summary

Computation

- Performed by combinational logic.
- Implements boolean functions.
- Continuously reacts to inputs.

Storage

- Registers: part of the CPU.
  - Each holds a single word.
  - Used for temporary results of computation.
  - Loaded on rising clock.
- Memory is much larger.
- Variety of implementation techniques.