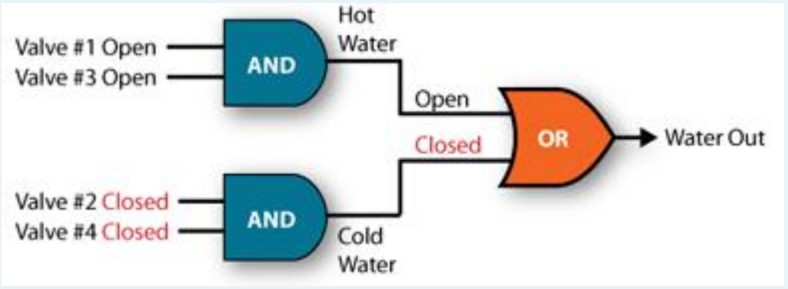
Logic and Computer Design Fundamentals

Introduction



Ming Cai

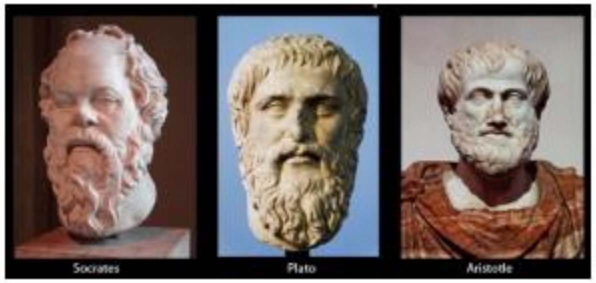
cm@zju.edu.cn

College of Computer Science and Technology
Zhejiang University

Course Information

- **Text book**
  - Mano and Kime 《Logic and Computer Design Fundamental》 5th Edition
- **QQ group: 868200982 (rename to “ID + name”)**
- **TA1: Qingqiang Zhang 951214966@qq.com**
- **TA2: Xiaoxuan Pang 21921170@zju.edu.cn**

Logic, Boolean Logic and Logic Circuit



ARISTOTLE

the ways of thinking, cognition...

Contribution: Propositional Logic and Syllogism

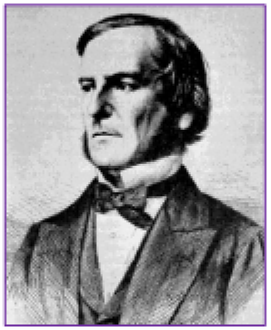
$$\begin{array}{ll}
\forall x(Man(x) \rightarrow Mortal(x)) & \text{assumptions} \\
Man(Socrates) & \big\downarrow \\
\hline
\therefore \quad Mortal(Socrates) & \text{conclusion}
\end{array}$$

Leibniz

Contribution: Leibniz studied binary numbering in 1679 and his work appeared in his article in 1703.

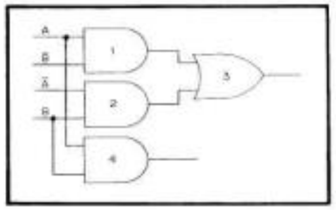
0 0 0 1	numerical value 2^0
0 0 1 0	numerical value 2^1
0 1 0 0	numerical value 2^2
1 0 0 0	numerical value 2^3

Logic, Boolean Logic and Logic Circuit

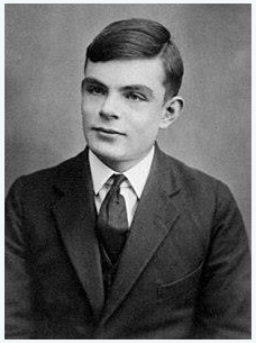


George Boole: An Investigation of the Laws of Thought (1854)

Claude Shannon: 1937 master's thesis,
A Symbolic Analysis of Relay and Switching Circuits



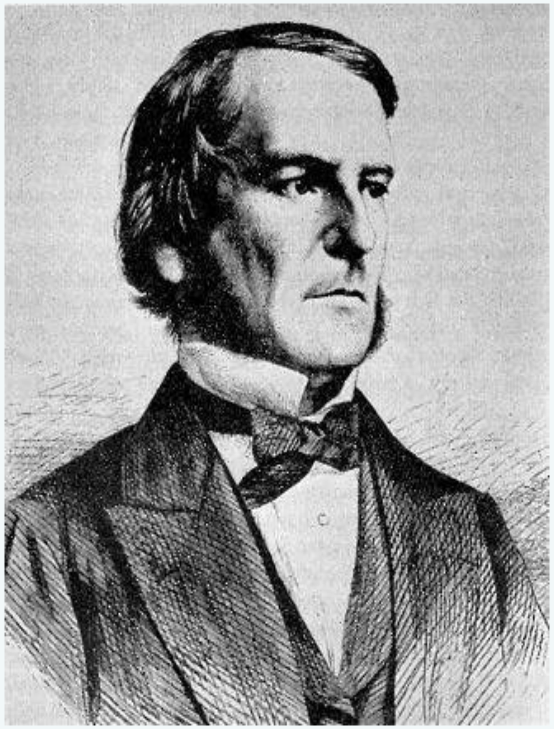
Gödel (1931)
Incompleteness theorem

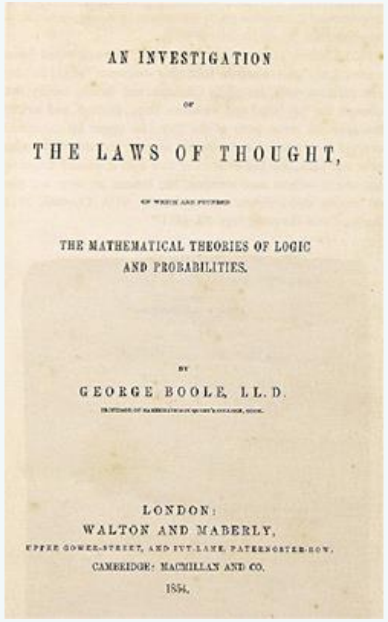


Turing (1936)
Turing machine

von Neumann
von Neumann architecture

From George Boole to….


George Boole
1815-1864


AN INVESTIGATION

OF

THE LAWS OF THOUGHT,

ON WHICH ARE FOUNDED

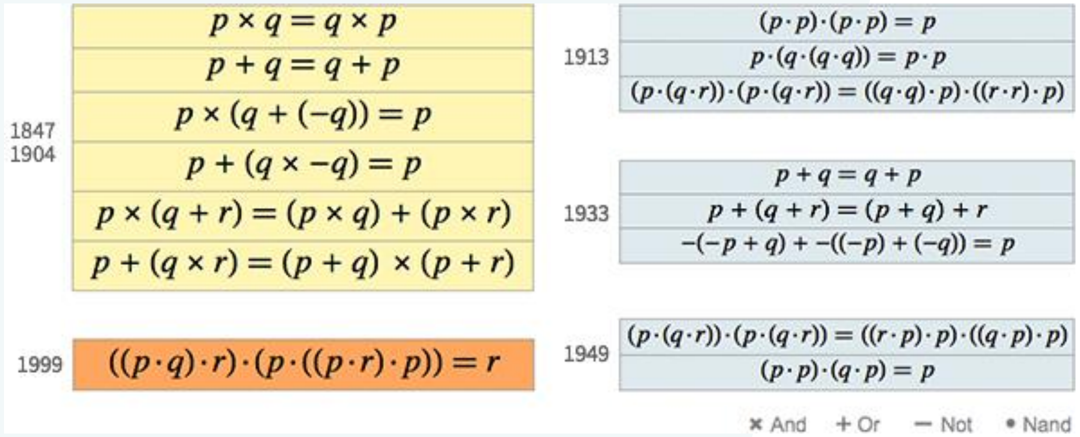
THE MATHEMATICAL THEORIES OF LOGIC
AND PROBABILITIES.

BY

GEORGE BOOLE, LL.D.

PROFESSOR OF MATHEMATICS IN QUEEN'S COLLEGE, CORK.

LONDON:
WALTON AND MABERLY,
UPPER GOWER-STREET, AND IVY-LANE, PATERNOSTER-ROW.
CAMBRIDGE: MACMILLAN AND CO.
1854.

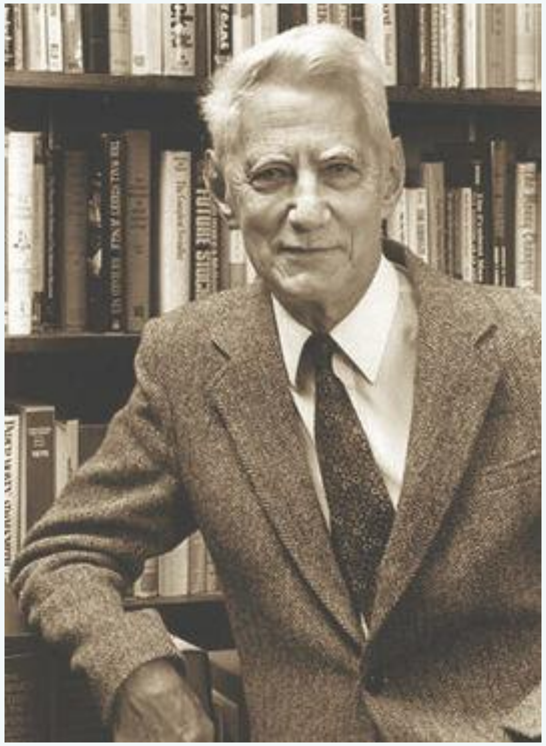


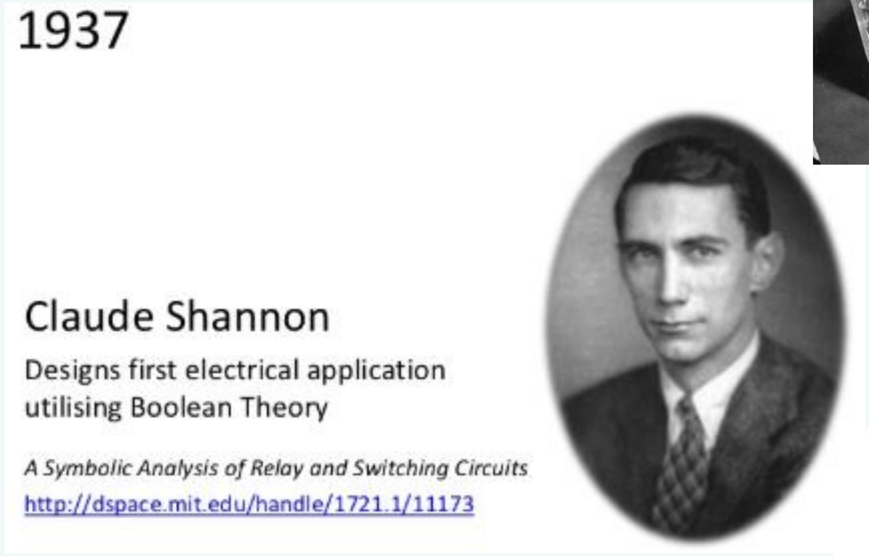
1847 1904	$p \times q = q \times p$
	$p + q = q + p$
	$p \times (q + (-q)) = p$
	$p + (q \times -q) = p$
	$p \times (q + r) = (p \times q) + (p \times r)$
	$p + (q \times r) = (p + q) \times (p + r)$
1999	$((p \cdot q) \cdot r) \cdot (p \cdot ((p \cdot r) \cdot p)) = r$

1913	$(p \cdot p) \cdot (p \cdot p) = p$
	$p \cdot (q \cdot (q \cdot q)) = p \cdot p$
	$(p \cdot (q \cdot r)) \cdot (p \cdot (q \cdot r)) = ((q \cdot q) \cdot p) \cdot ((r \cdot r) \cdot p)$
1933	$p + q = q + p$
	$p + (q + r) = (p + q) + r$
	$-(-p + q) + -((-p) + (-q)) = p$
1949	$(p \cdot (q \cdot r)) \cdot (p \cdot (q \cdot r)) = ((r \cdot p) \cdot p) \cdot ((q \cdot p) \cdot p)$
	$(p \cdot p) \cdot (q \cdot p) = p$

× And + Or − Not • Nand

From George Boole to….

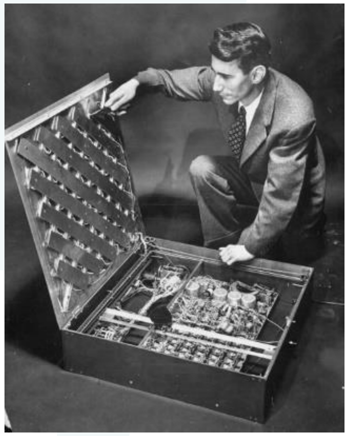

Claude Elwood Shannon
1916–2001


1937

Claude Shannon

Designs first electrical application utilising Boolean Theory

A Symbolic Analysis of Relay and Switching Circuits
http://dspace.mit.edu/handle/1721.1/11173

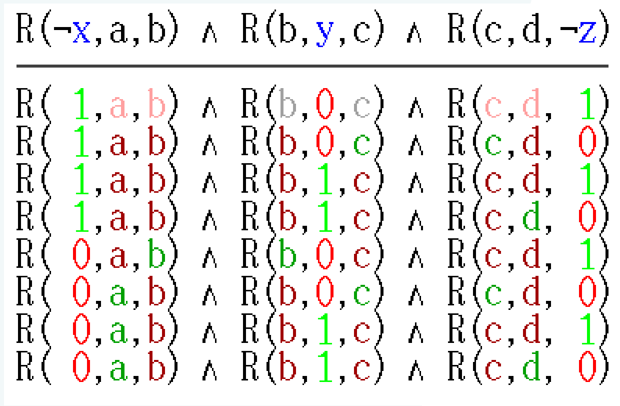


“ possibly the most important, and also the most famous, master‘s thesis of the century”

——Howard Gardner

From George Boole to….

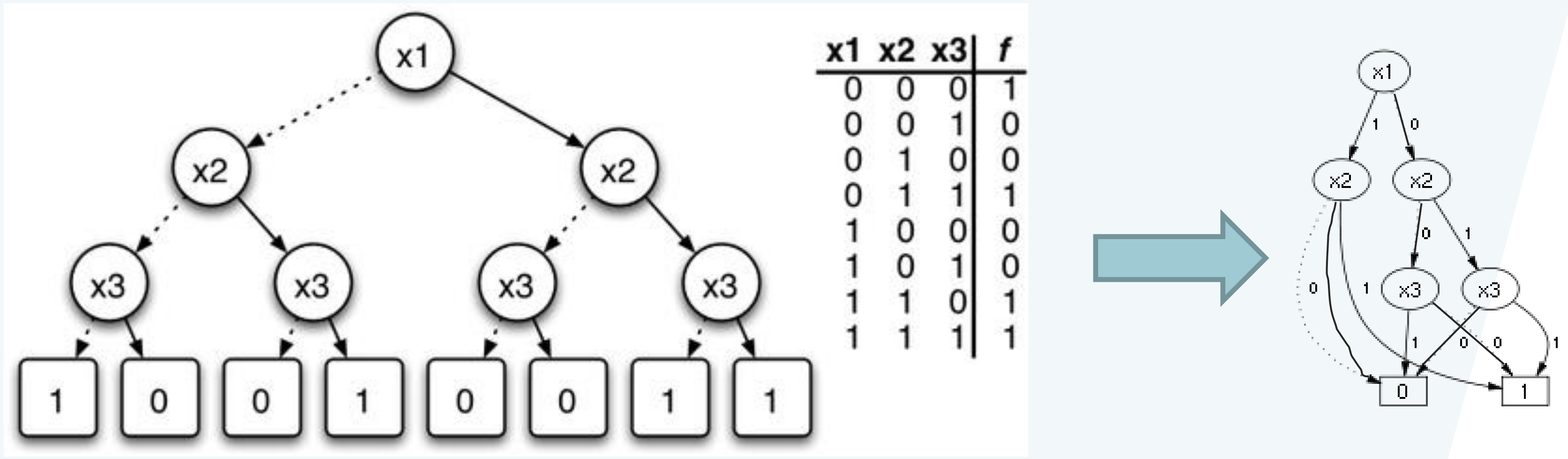
- The Boolean satisfiability problem (abbreviated SAT) is the problem of determining if there exists an interpretation that satisfies a given Boolean formula.
- SAT is the first problem that was proven to be NP-complete in 1971.
- Whether SAT has a polynomial-time algorithm is equivalent to the P versus NP problem, which is a famous open problem in the theory of computing.



```
R(¬x,a,b) ∧ R(b,y,c) ∧ R(c,d,¬z)
--------------------------------
R( 1,a,b) ∧ R(b,0,c) ∧ R(c,d, 1)
R( 1,a,b) ∧ R(b,0,c) ∧ R(c,d, 0)
R( 1,a,b) ∧ R(b,1,c) ∧ R(c,d, 1)
R( 1,a,b) ∧ R(b,1,c) ∧ R(c,d, 0)
R( 0,a,b) ∧ R(b,0,c) ∧ R(c,d, 1)
R( 0,a,b) ∧ R(b,0,c) ∧ R(c,d, 0)
R( 0,a,b) ∧ R(b,1,c) ∧ R(c,d, 1)
R( 0,a,b) ∧ R(b,1,c) ∧ R(c,d, 0)
```

From George Boole to….

- A binary decision diagram (BDD) or branching program is a data structure that is used to represent a Boolean function.

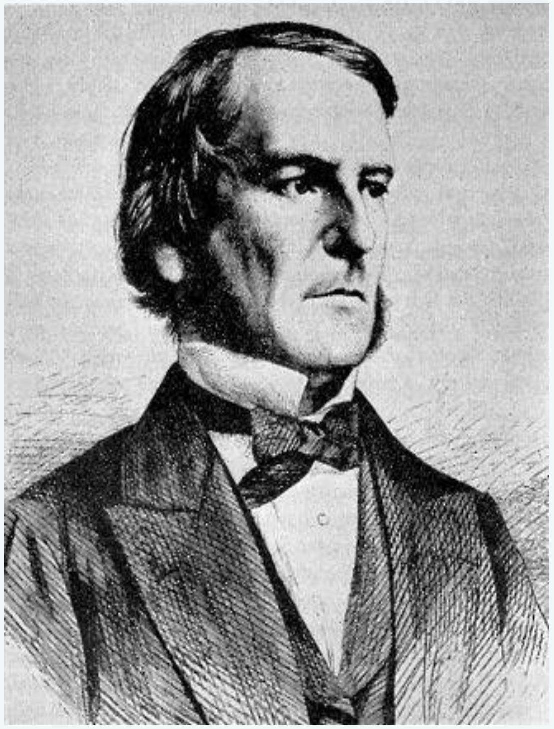


x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

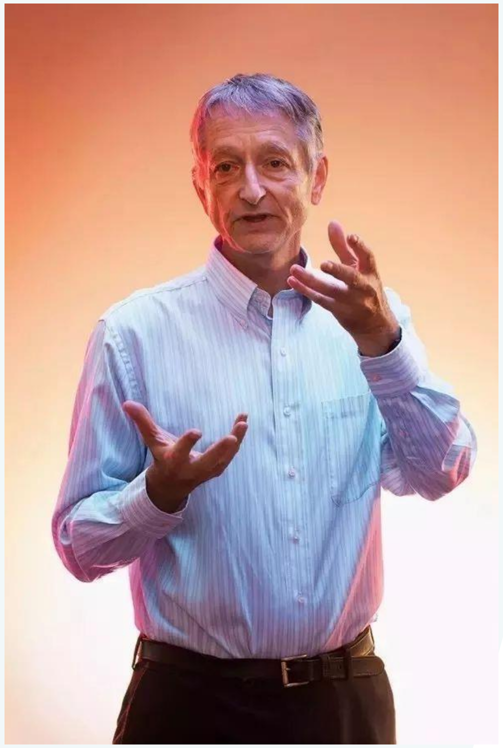
- The full potential for efficient algorithms was investigated by Randal Bryant at Carnegie Mellon University.
- “one of the only really fundamental data structures that came out in the last twenty-five years” 、 “Bryant's 1986 paper was one of the most-cited papers in computer science”

—— Donald Knuth

From George Boole to….

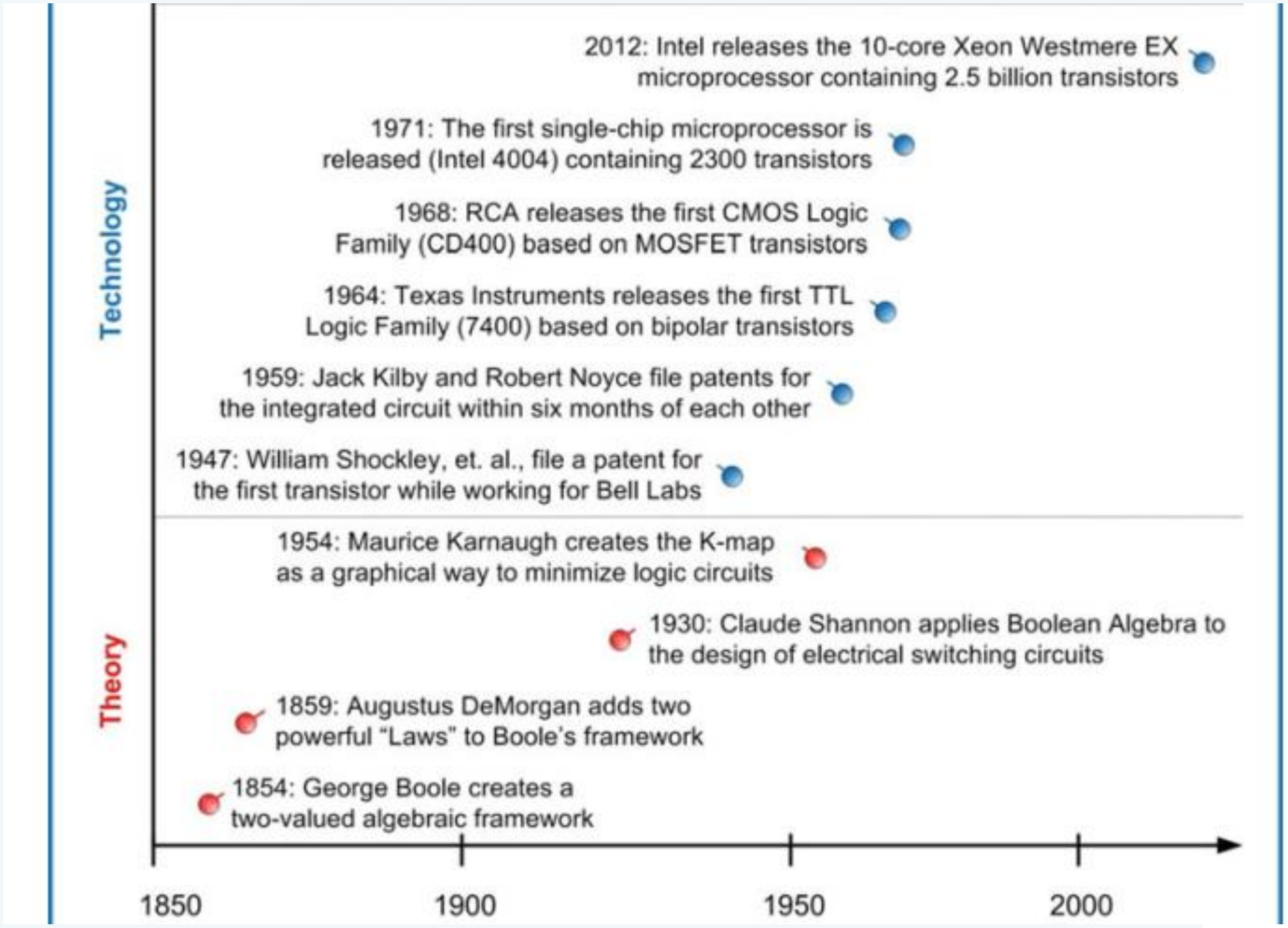


George Boole
1815-1864



Geoffrey Hinton
1947-
2019 Turing Prize

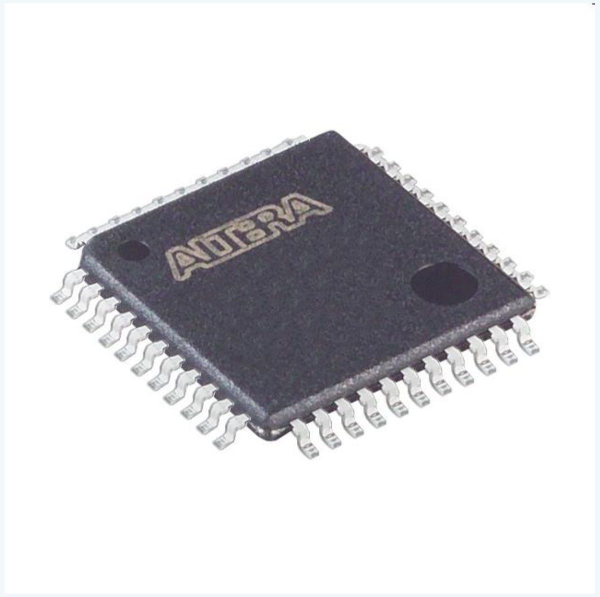
Major Milestones in Digital Logic



Technology
2012: Intel releases the 10-core Xeon Westmere EX microprocessor containing 2.5 billion transistors
1971: The first single-chip microprocessor is released (Intel 4004) containing 2300 transistors
1968: RCA releases the first CMOS Logic Family (CD400) based on MOSFET transistors
1964: Texas Instruments releases the first TTL Logic Family (7400) based on bipolar transistors
1959: Jack Kilby and Robert Noyce file patents for the integrated circuit within six months of each other
1947: William Shockley, et. al., file a patent for the first transistor while working for Bell Labs
Theory
1954: Maurice Karnaugh creates the K-map as a graphical way to minimize logic circuits
1930: Claude Shannon applies Boolean Algebra to the design of electrical switching circuits
1859: Augustus DeMorgan adds two powerful "Laws" to Boole's framework
1854: George Boole creates a two-valued algebraic framework
1850
1900
1950
2000

Course Description

- National 2003 Teaching Program
- Digital System Fundamental



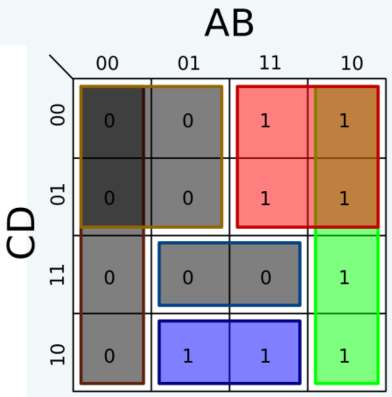
Course Objectives

- **To be able to analyze and design digital logic systems by understanding formal foundations and selected design techniques.**
  - **Introduce basic theory and design methods for digital logic.**
  - **Give students basic skills to analysis and design electronic digital computer logic circuit**
  - **Prepare for the further studies on hardware related courses, such as**
    - **Computer Organization**
    - **Computer Architecture**
    - **Embedded Systems**
    - **Communication**
    - **…**

Main topics

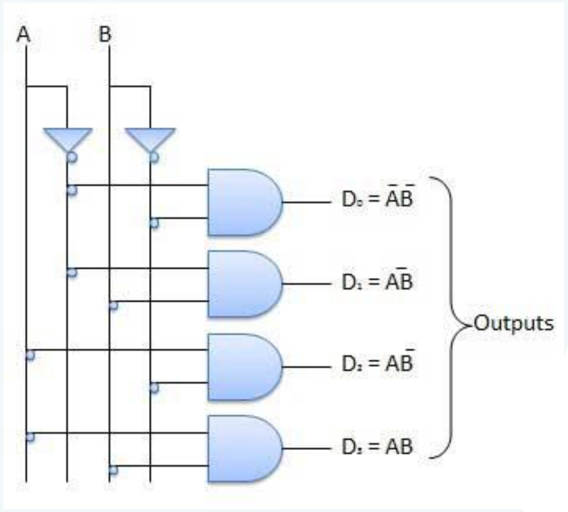
❖ Topics

$$
\begin{aligned}
F &= \overline{AB+\overline{C}}+A\overline{C}+B \\
&= (\overline{A}+\overline{B})\,C+A\overline{C}+B \\
&= \overline{A}C+\overline{B}C+A\overline{C}+B \\
&= B+C+\overline{A}C+A\overline{C} \\
&= B+C+A+\overline{A}C \\
&= A+B+C
\end{aligned}
$$

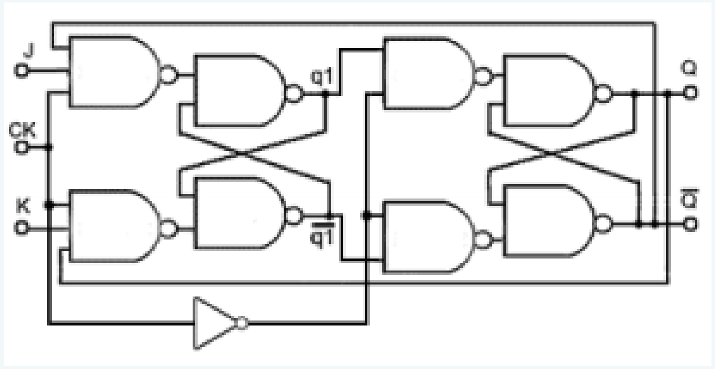


f(A,B,C,D) = E(6,8,9,10,11,12,13,14)
F=AC'+AB'+BCD'
F=(A+B)(A+C)(B'+C'+D')

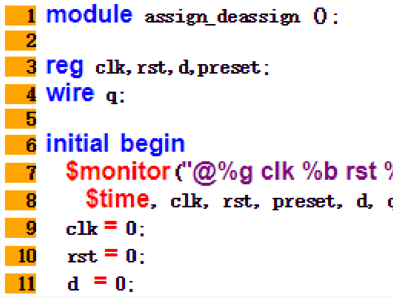
Boolean algebra and minimization



Combinational circuit



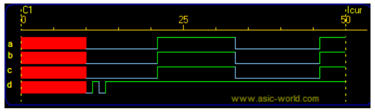
Sequential circuit



```
module assign_deassign ();

reg clk,rst,d,preset;
wire q;

initial begin
  $monitor("@%g clk %b rst %b preset %b d %b q %b",
    $time, clk, rst, preset, d, q);
  clk = 0;
  rst = 0;
  d  = 0;
```



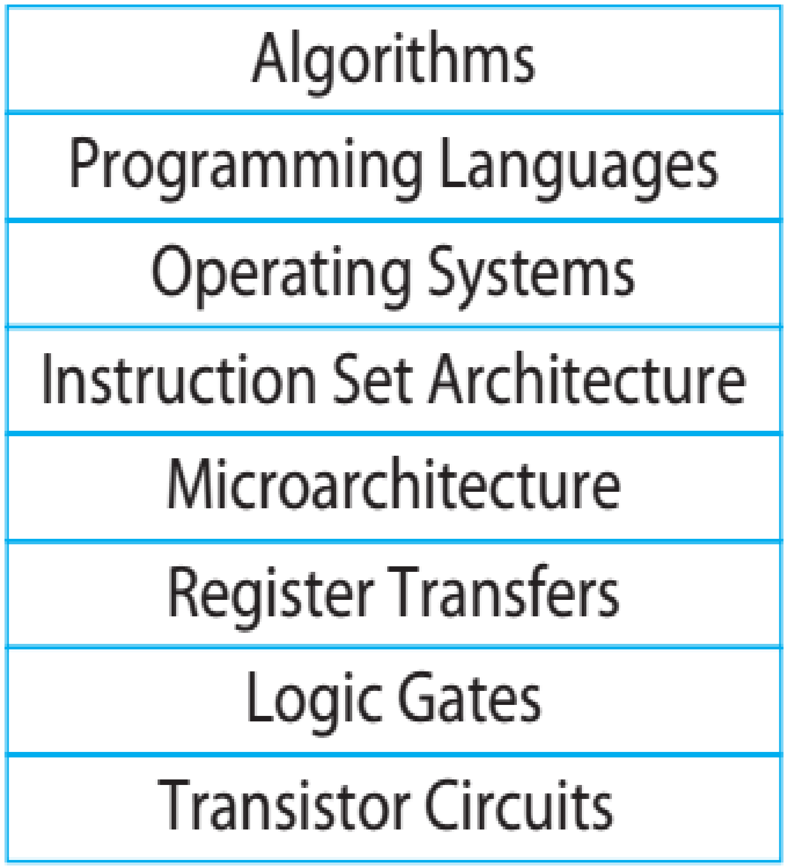
Logic design using Verilog

Topics covered

- **Topics**
  - **Number representation, digital codes**
  - **Boolean algebra and logic minimization techniques**
  - **Combinational circuit design and analysis**
  - **Sequential circuit design and analysis, timing analysis of sequential circuits**
  - **Programmable logic devices (PLD) and memories**
  - **Logic design with structural and behavioral modeling (Verilog)**

You are supposed to acquire basic skills and knowledge to analysis and design logic circuit

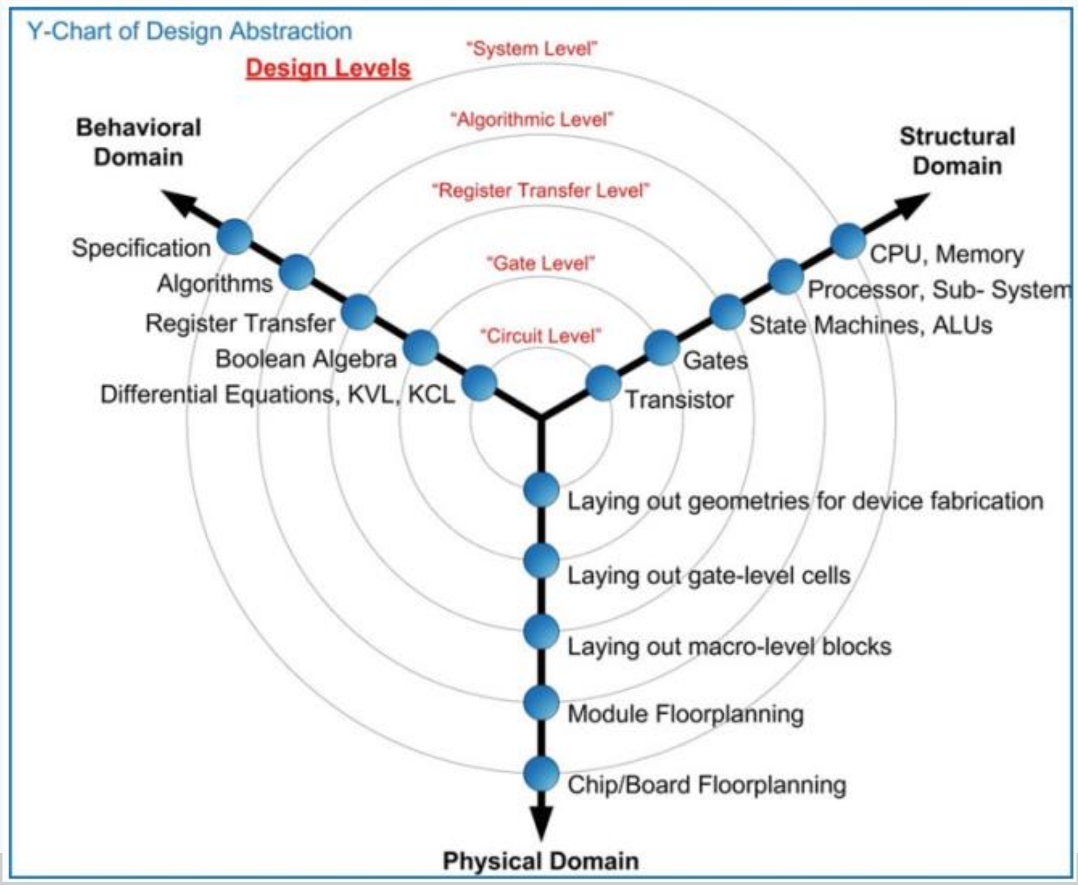
Abstraction Layers in Computer Systems



- **greedy, heuristic, LP, DP**
- **C/C++, Java, Python**
- **Linux, Windows, Android**
- **X86, ARM, MIPS, PPC**
- **Pipeline, OOE, Multiprocessing**
- **Register, Datapath, Control Unit**
- **AND, OR, NOT, NAND, NOR**
- **BJT, JFET, IGFET**

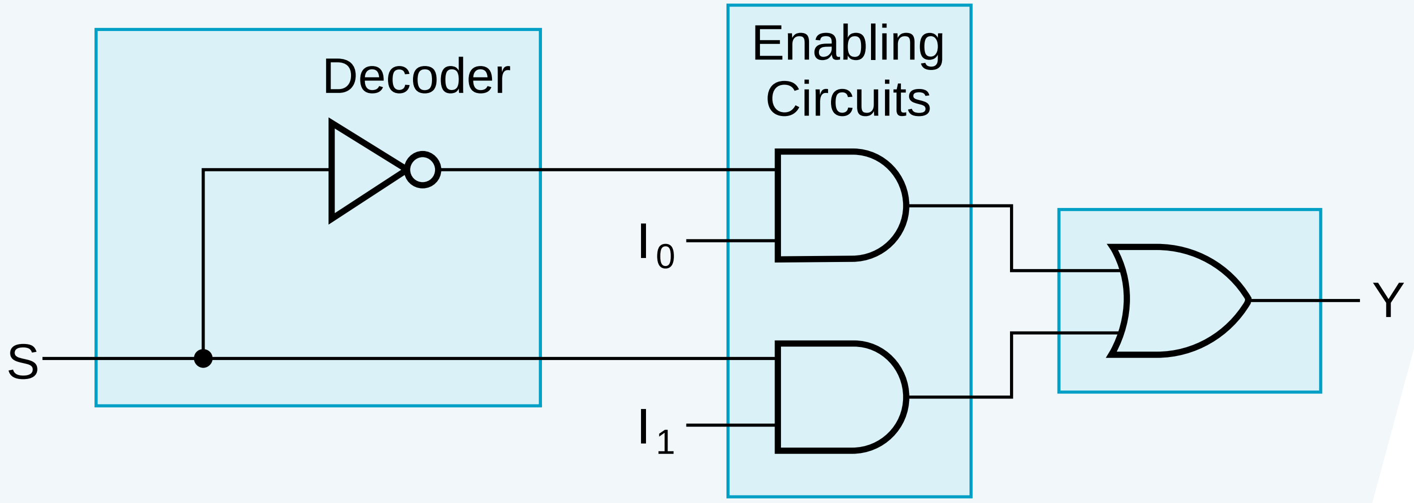
Top picture of the logic design

- **Behavioral**: specifies what a specific system does
- **Structural**: specifies how entities are connected together
- **Physical**: specifies how to build a structure



Behavioral and Structural Modeling

- 2-to-1-Line Multiplexer ($Y = Mux(S, I_0, I_1)$)



- **Behavioral modelling**

```
if (!S)
    Y = I0;
 else
     Y = I1;
```

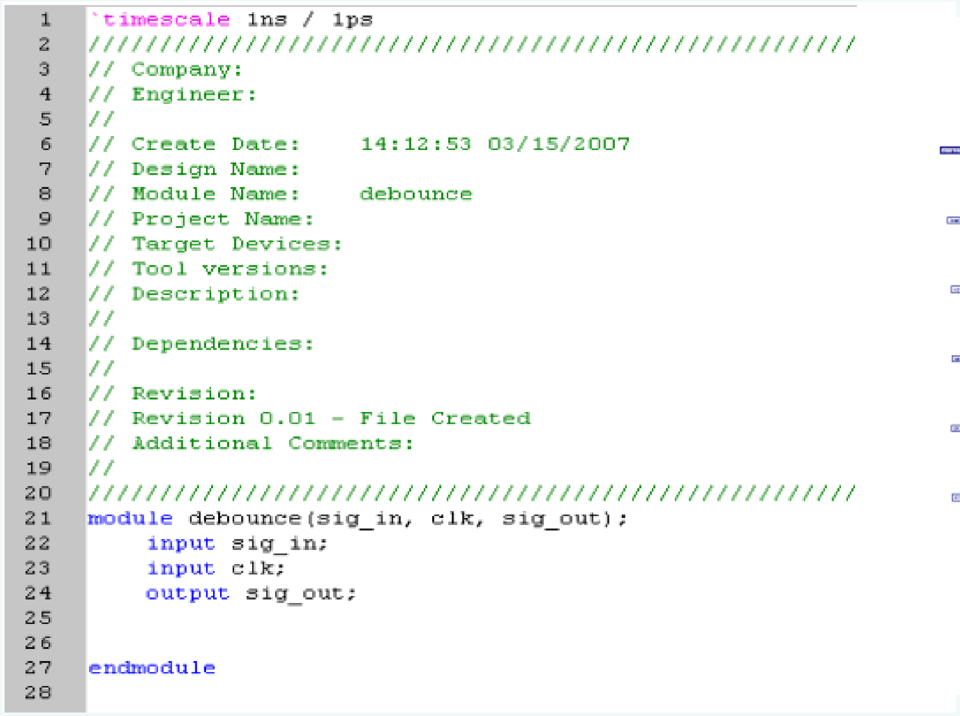
Structural modelling

```
not   u1(NS, S);
and  u2(sel0, I0, NS);
and  u3(sel1, I1, S);
or     u4(Y, sel0, sel1);
```

Two design methods

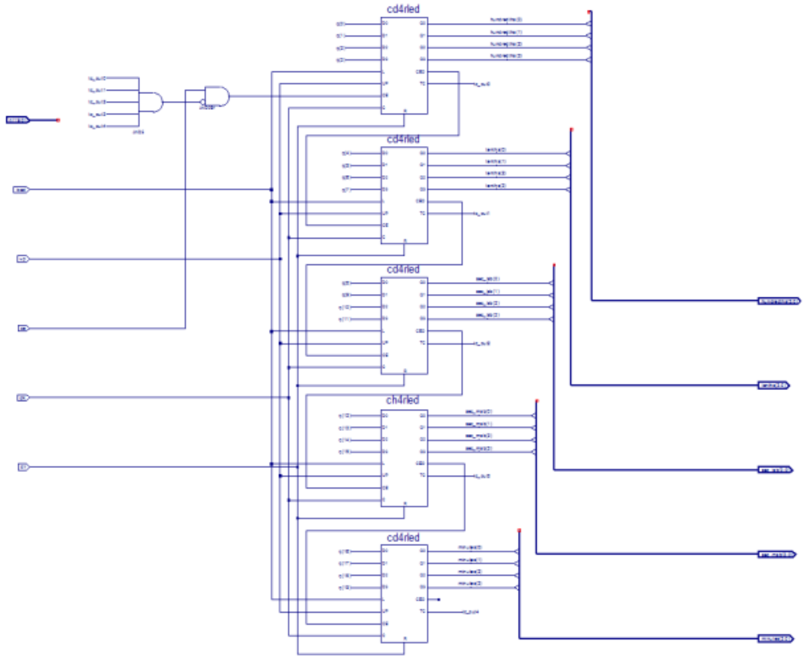
❖ **HDL-Based Design**

❖ **Schematic-Based Design**



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    14:12:53 03/15/2007
// Design Name:
// Module Name:    debounce
// Project Name:
// Target Devices:
// Tool versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
module debounce(sig_in, clk, sig_out);
     input sig_in;
     input clk;
     output sig_out;


endmodule
```



Verilog File in Xilinx ISE
Structural and Behavioral Modeling

Schematic in ISE Editor
Structural Modeling

Experiment

编号	实验名称	实验方式
1	常用数字仪器的使用	硬件实验
2	二极管与三极管开关电路	硬件实验
3	集成逻辑门电路的功能及参数测试	硬件实验
4	EDA实验平台与实验环境运用	绘图实验
5	变量译码器设计与应用	绘图实验
6	7段数码管显示译码器设计与应用	绘图实验
7	多路选择器设计及应用	绘图实验，学习模块调用，模块化使用按键数据输入模块
8	全加器的设计实现	绘图实验，学习模块封装，模块化使用按键数据输入模块
9	加减法器和ALU基本原理与设计	绘图实验 + Verilog结构化描述
10	锁存器与触发器基本原理	绘图实验
11	同步时序电路设计	绘图实验 + Verilog结构化描述
12	寄存器堆及寄存器传输设计	Verilog结构化描述
13	计数器/定时器设计与应用	Verilog结构化描述 + 硬件实验
14	移位寄存器设计与应用	Verilog结构化描述

Course Assessment

- **Experiments: 30%**
- **Project: 15%**
  - Source code, source project and technical report should be submitted
  - Deadline: before the final examination

Practice is a vital process of the class

- **Quiz: 25%**
  - Random, without notification, 4 – 5 times
  - Topics from textbook and home assignments
- **Final: 30%**
  - Score of Final Examination: ≥ 50

Course Requirements

- **Home assignments:**
  - Need not submit, but spot check
  - Answer sheet will be published
- **should be finished in English**
  - Home assignments, project technical report, classroom tests and final paper examination

Course Resources

- **Software & Online practice**
  - HDLBits-Verilog Practice: https://hdlbits.01xz.net/wiki/Problem_sets#Getting_Started
  - Logisim: https://sourceforge.net/projects/circuit/?source=typ_redirect
  - online Karnaugh map generator: http://www.32x8.com
- **References**
  - R. Mano and Ciletti, 《Logic design with an introduction to the Verilog》 Fifth edition
  - Vaibbhav Taraate, 《Digital Logic Design Using Verilog》 ISBN: 978-81-322-2789-2, Springer, 2016
  - Brock J. LaMeres, 《Introduction to Logic Circuits & Logic Design with Verilog》 ISBN: 978-3-319-53882-2, Springer, 2017
  - 夏宇闻. Verilog数字系统设计教程. 北京：北京航空航天大学出版社
- **Reading Supplements**
  - http://wps.pearsoned.com/ecs_mano_lcdf_5/248/63706/16308896.cw/index.html

Appendix A: Insights from 2017 Turing Award Winners

❖ Turing Lecture

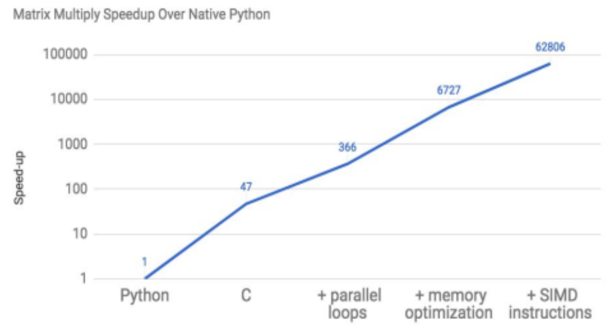
A New Golden Age for Computer Architecture:

Domain-Specific Hardware/Software Co-Design, Enhanced Security, Open Instruction Sets, and Agile Chip Development

John Hennessy and David Patterson
Stanford and UC Berkeley
June 4, 2018

What's the Opportunity?

Matrix Multiply: relative speedup to a Python version (18 core Intel)

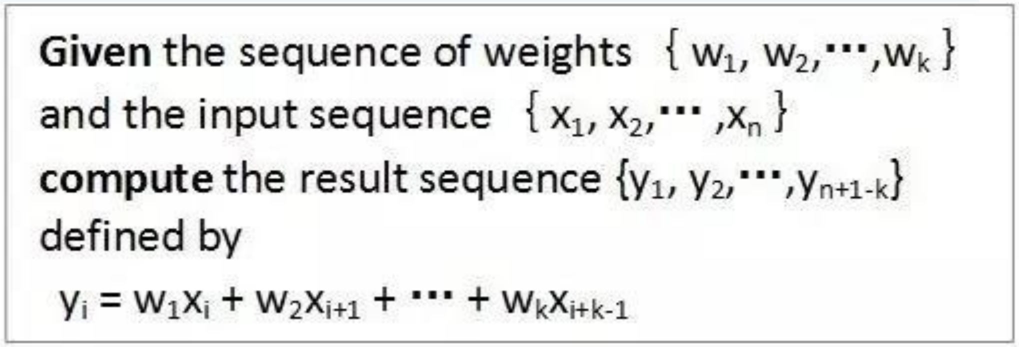


Why DSAs Can Win (no magic)
Tailor the Architecture to the Domain

- More effective parallelism for a specific domain:
  - SIMD vs. MIMD
  - VLIW vs. Speculative, out-of-order
- More effective use of memory bandwidth
  - User controlled versus caches
- Eliminate unneeded accuracy
  - IEEE replaced by lower precision FP
  - 32-64 bit bit integers to 8-16 bit integers
- Domain specific programming language

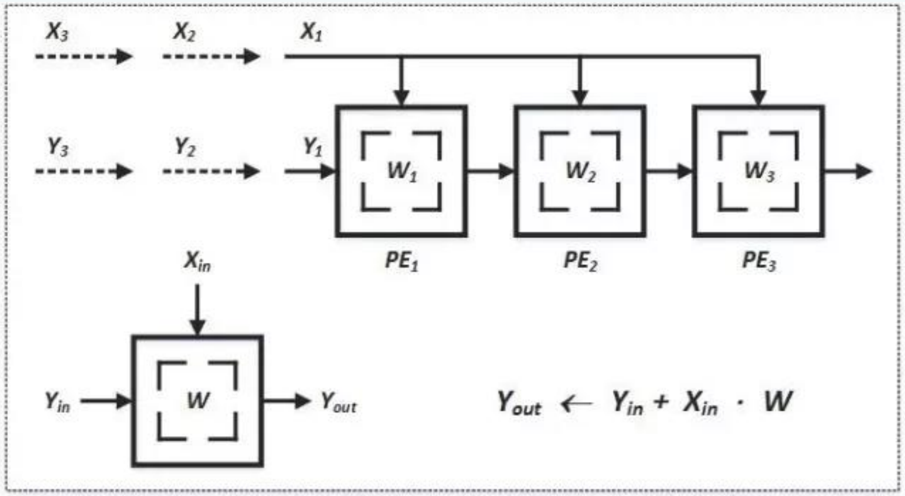
Appendix B: Some Motivating Examples (1/2)

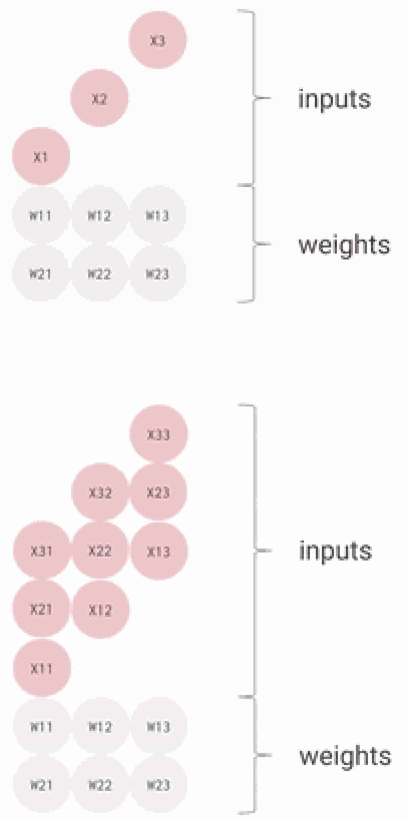
- **Hardware acceleration for Artificial Intelligence(AI)**



Given the sequence of weights $\{ w_1, w_2, \cdots, w_k \}$
and the input sequence $\{ x_1, x_2, \cdots, x_n \}$
compute the result sequence $\{y_1, y_2, \cdots, y_{n+1-k}\}$
defined by

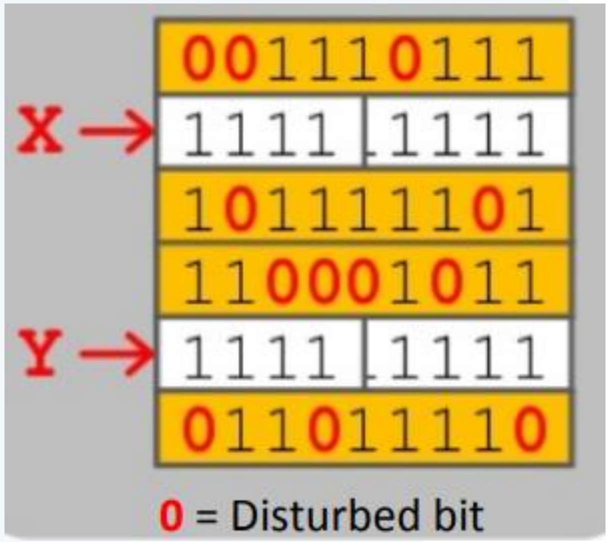
$$y_i = w_1 x_i + w_2 x_{i+1} + \cdots + w_k x_{i+k-1}$$



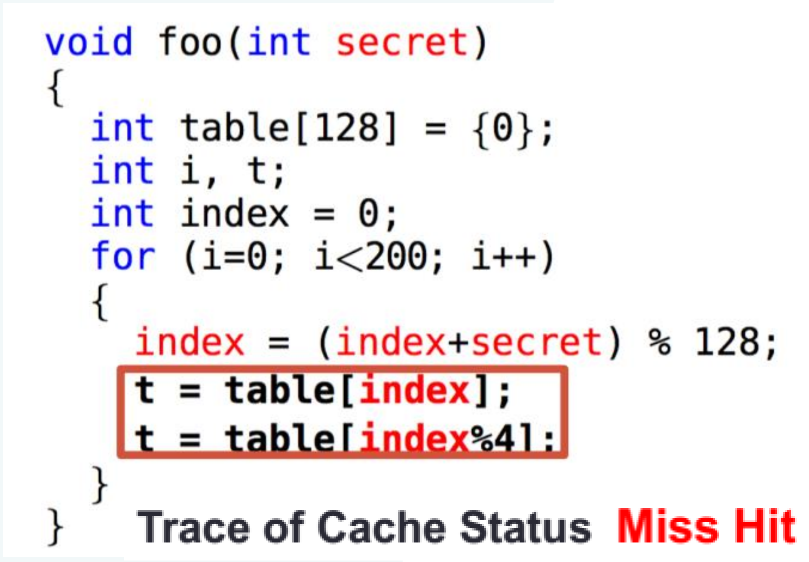


Appendix B: Some Motivating Examples (2/2)

- **Row Hammer**
  - Kim Y, Daly R, Kim J, et al. Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors. ISCA 2014



- **Cache Difference**
  - Wang S, Wang P, Liu X, et al. CacheD: Identifying Cache-Based Timing Channels in Production Software. USENIX security symposium, 2017



```
void foo(int secret)
{
  int table[128] = {0};
  int i, t;
  int index = 0;
  for (i=0; i<200; i++)
  {
    index = (index+secret) % 128;
    t = table[index];
    t = table[index%4];
  }
}
```