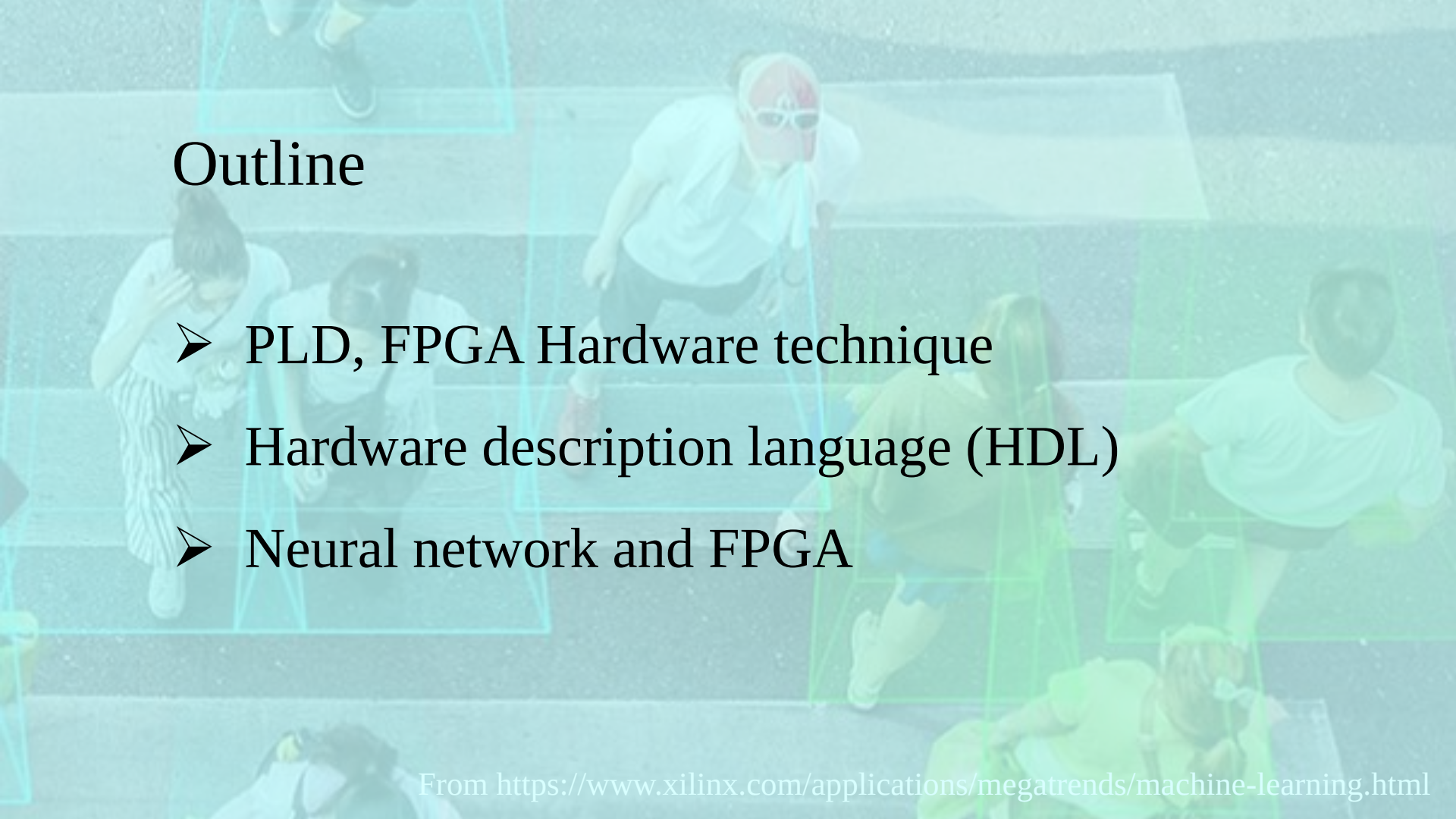
電子回路論第14回
Electric Circuits for Physicists #14

東京大学理学部・理学系研究科
物性研究所
勝本信吾
Shingo Katsumoto

Outline

- PLD, FPGA Hardware technique
- Hardware description language (HDL)
- Neural network and FPGA

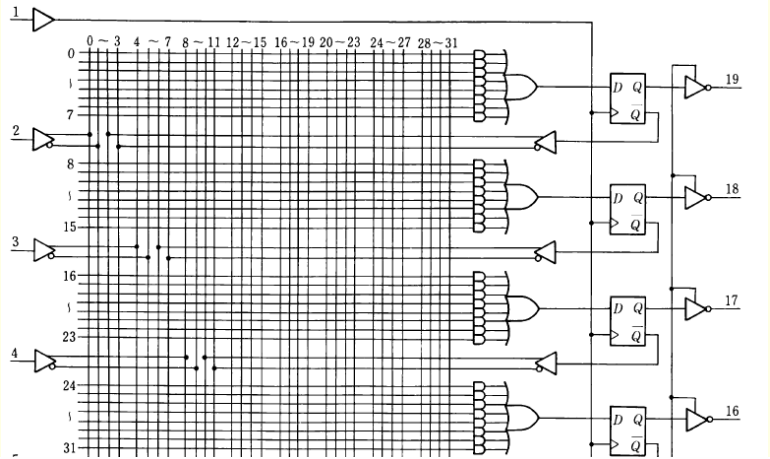


From https://www.xilinx.com/applications/megatrends/machine-learning.html

7.8 Circuit realization with Hardware Description Language (HDL)

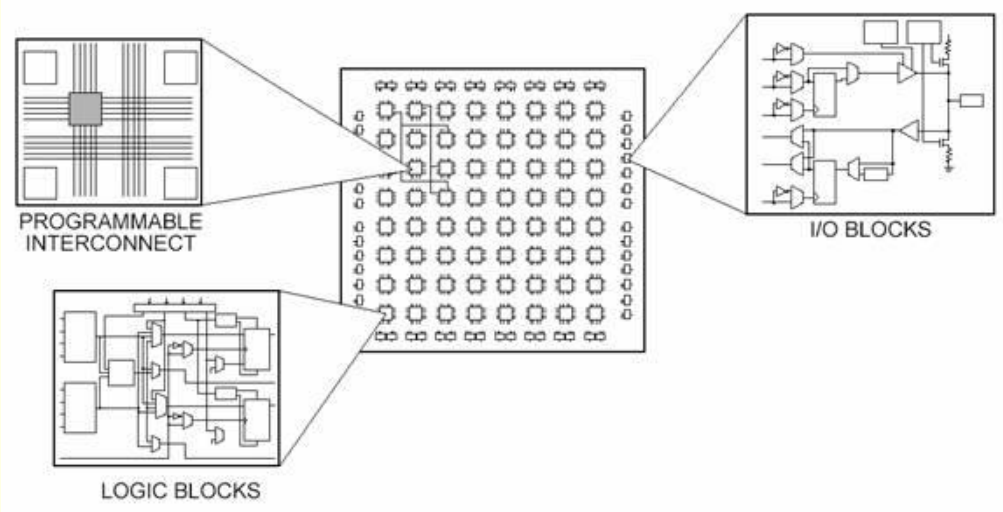
PLD/FPGA with HDL

Example of programmable logic device (PLD) circuit

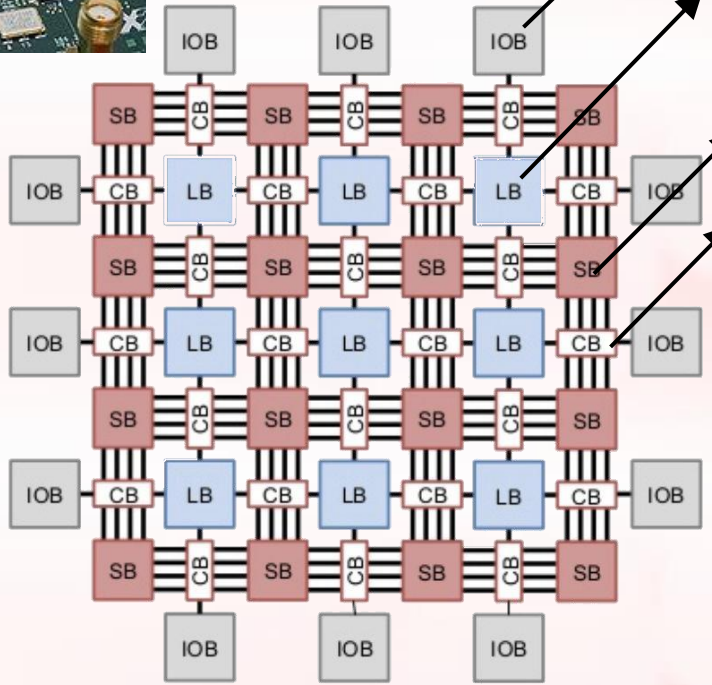


Example of field-programmable gate array (FPGA) circuit

FPGA ∈ PLD



Configuration of FPGA



Input/Output block

Logic block

Switch block

Connection block

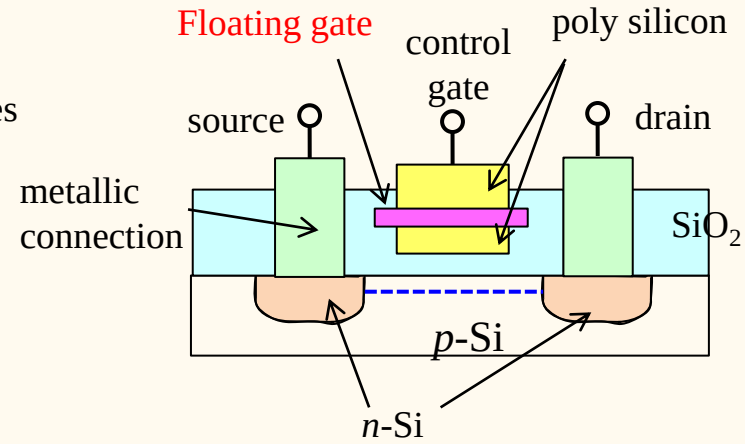
Connected to configuration memory

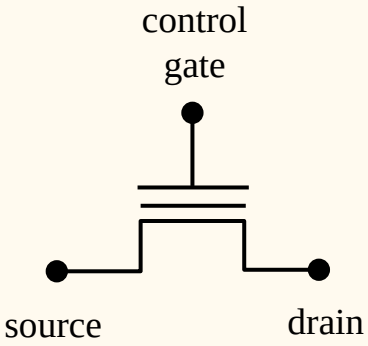
Programming technology

Flush memory

One of electrically erasable programmable read-only memories (EEPROMs)

Quantum tunneling process charge up/discharge the floating gate



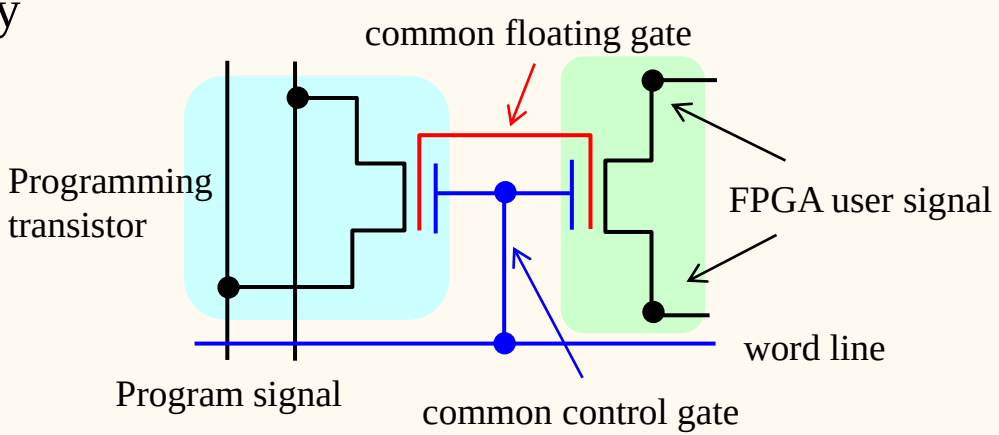


Fujio Masuoka

Programmable switch with flush memory

Programing transistor SD 5 V, control gate -11 V→ ON, SD0 V, CG16 V → OFF

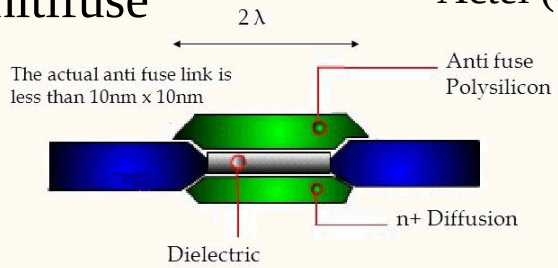
- Rewritable, non-volatile, small number of device, live at power-up
- Device size is large, on-resistance is high, load capacitance is large, memory lifetime?

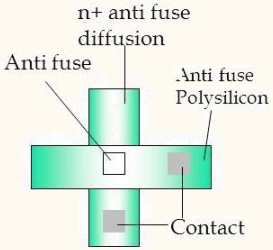


Programming technology (2)

Anitifuse

Actel (c), QuickLogic (c)





Recently metal-metal type is dominant.

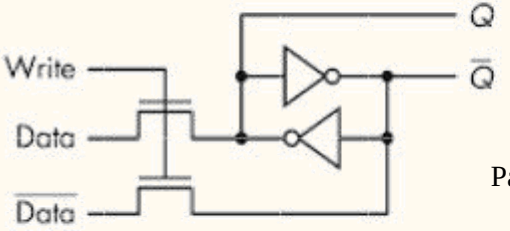
The inverse process of fusing. That is, short circuit with high voltage and current. Fusing was also used.

- Long, stable memory life. Small device size. Small on resistance.
- Not rewritable. Program needs extra transistors. Available percentage (fidelity) is low.

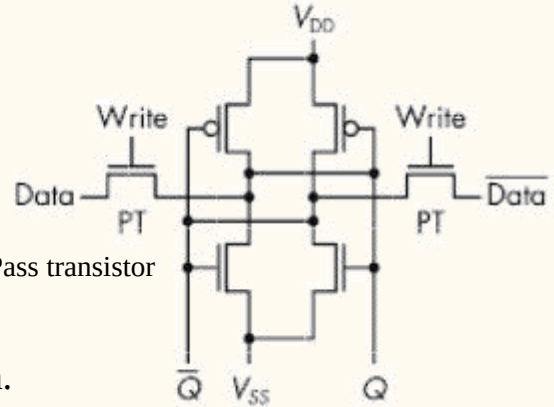
Static memory (SRAM type, mainstream method now)

SRAM does not need refreshing but volatile.

- Large device size, resources, large on resistance, stray capacitance
- State-of-art CMOS technology can be used

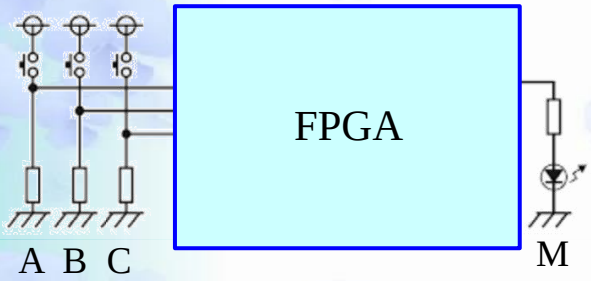


Positive feedback loop and latch.



Pass transistor

Simple example of logic block configuration



Circuit level understanding of FPGA

Let us consider a majority rule circuit. The truth table is given.

Truth table of majority rule

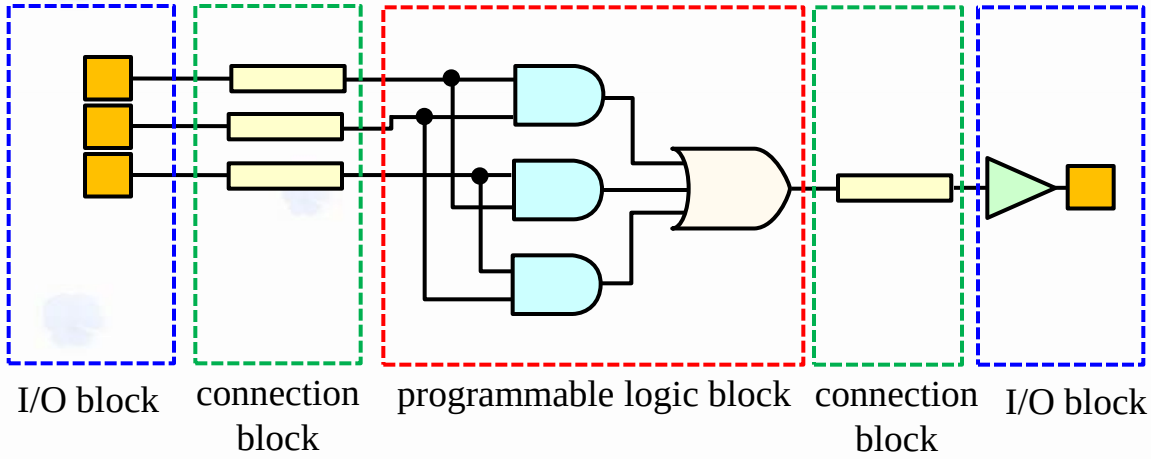
A	B	C	M
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

C \ AB	00	01	11	10
0			1	
1		1	1	1

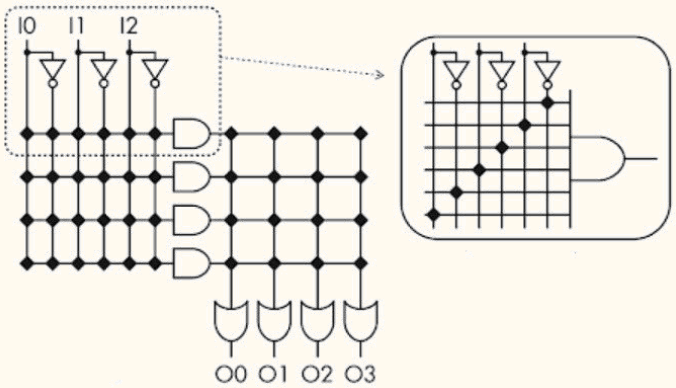
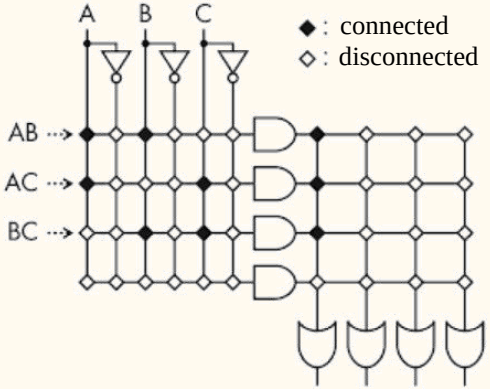
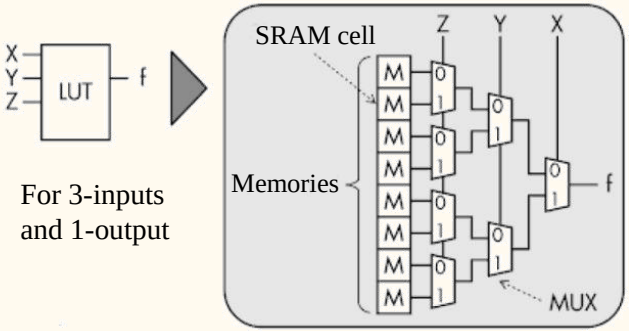
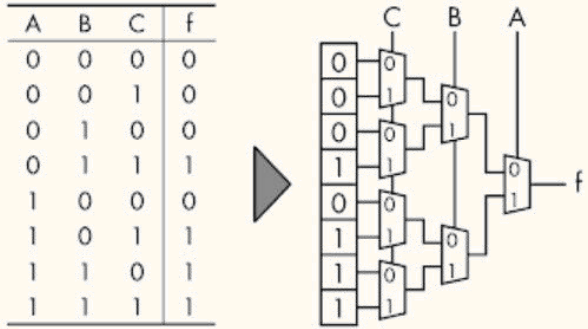
Karnaugh mapping

M=AB+AC+BC

(is not required for LUT)



Various ways to realize logic block

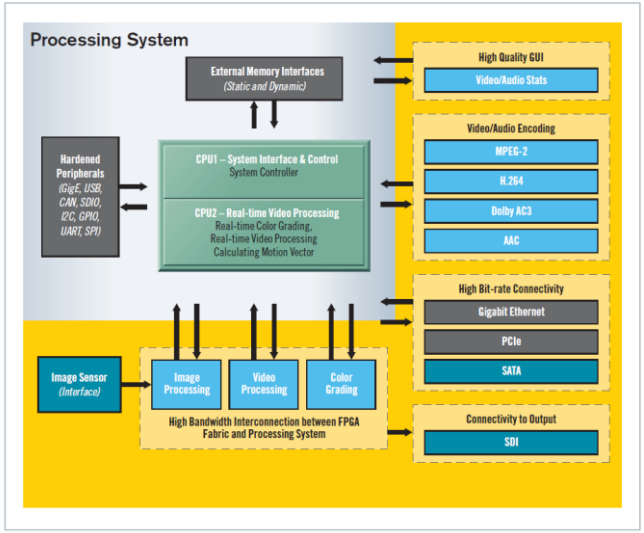
	Resource	Majority rule implementation
Product term		
Look up table	 For 3-inputs and 1-output Multiplexer	

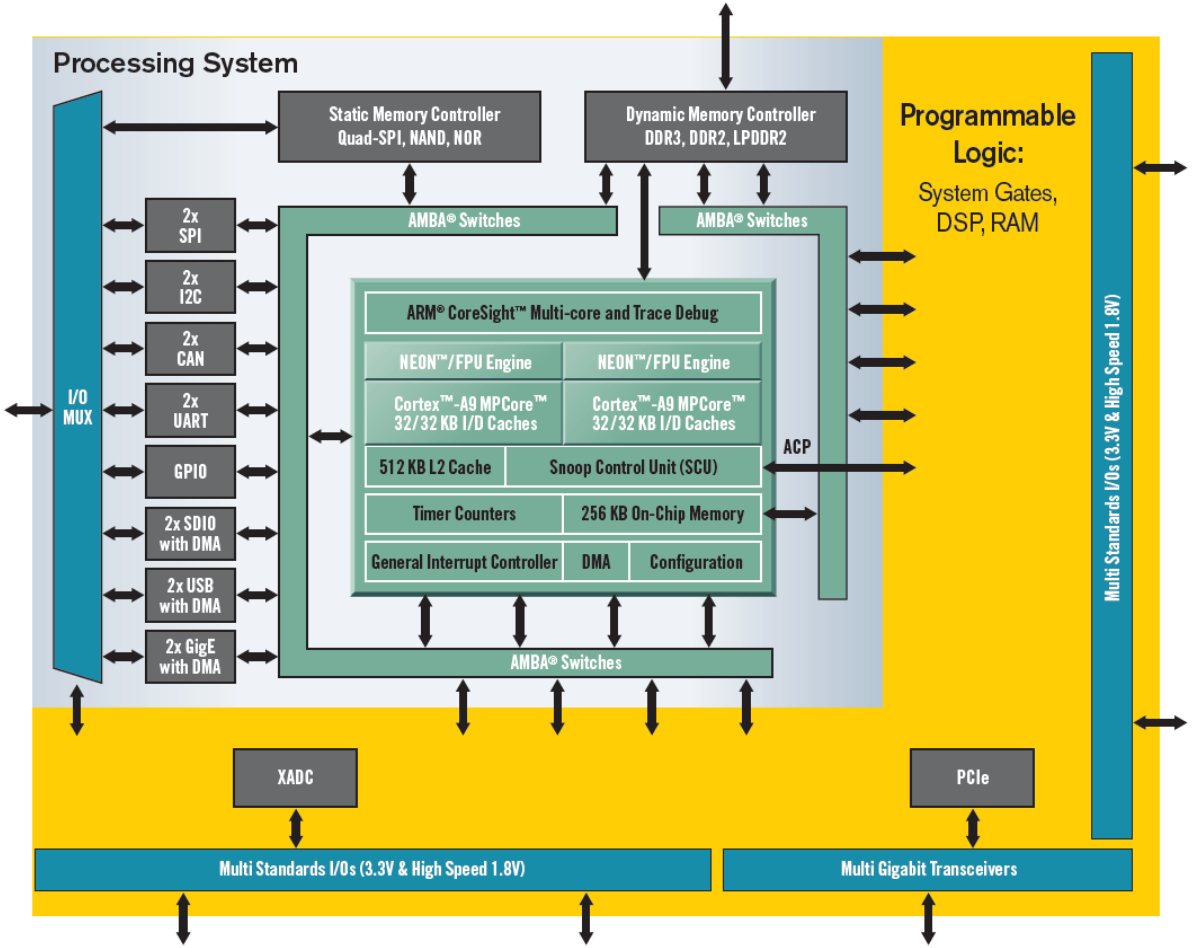
Advance type FPGA

Integration of DSP, FPU, etc.

(from Xilinx Zynq product brief)

BROADCAST CAMERA APPLICATION EXAMPLE



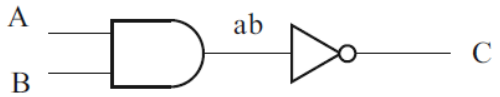


Hardware description language (HDL)

HDL
- VHDL
- Verilog HDL

```
-- Library declaration ----------
library IEEE;
use IEEE, STD_LOGIC_1164.ALL;
-- Entity declaration ---------------
entity NAND_CIRCUIT is
port(
A : in std_logic;
B : in std_logic;
C : out std_logc
);
end NAND_CIRCUIT;
-- Architecture declaration ----------
architecture RTL of NAND_CIRCUIT is
signal ab : std_logic;
begin
ab <= A and B;
C  <= not ab;
end RTL;
```

VHDL example for



RTL: register transfer level

Hardware design with high-level programming languages

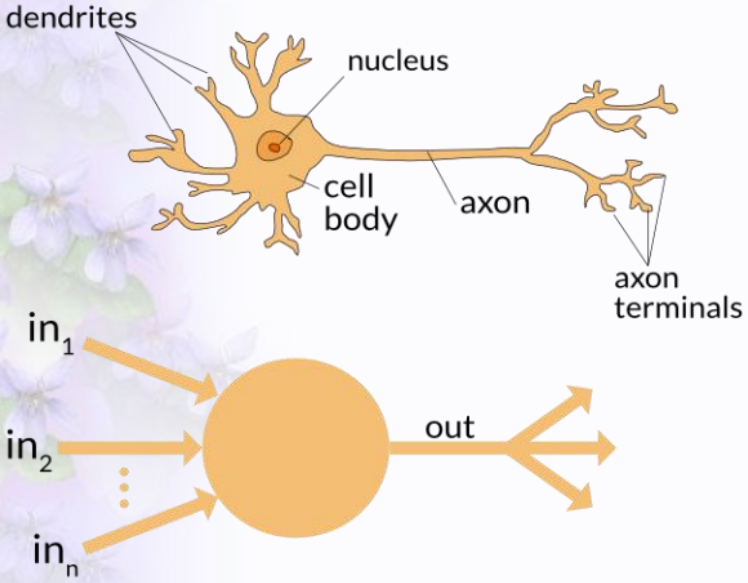
Translation or High level synthesis (高位合成) or Behavioral synthesis:

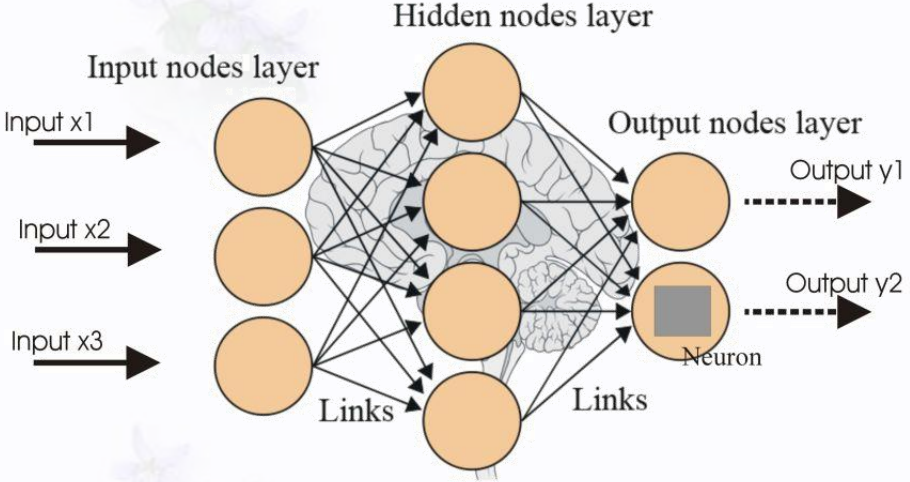
System name	Cyber Work Bench	Vivado-HLS	Catapult C	Impulse C	Synphony C Compiler	C-to-Silicon Compiler
Company	NEC	Xilinx	Mentor Graphics	Impulse Accelerated Technologies	Synopsis	Cadence
Language	System-C/ANSI-C	C/C++	ANSIC++/System C	ANSI C	C/C++	System C

Behavior description with high-level languages → Synthesis of HDL codes

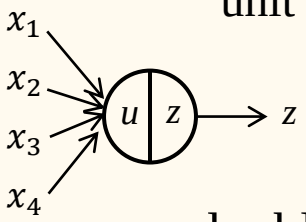
Not almighty. Because high-level languages generally use large memories but there are not such size of memories being close to hand in the case of FPGA.

Neural Network (NN)





Feedforward neural network or multi-layer perceptron



unit or perceptron 4-input example

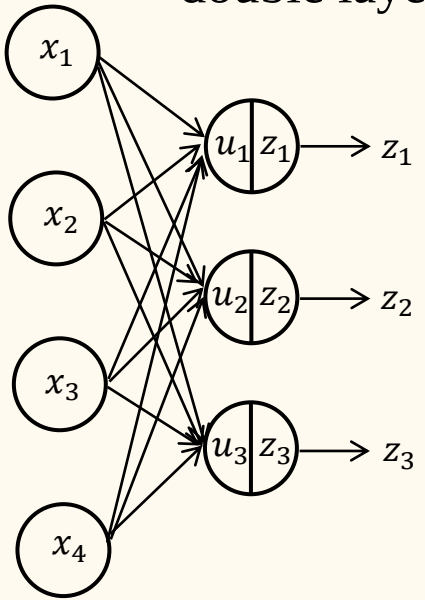
$$u = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

$$z = f(u)$$: activation function (generally non-linear)

vector: $\boldsymbol{w}$ weight

b bias

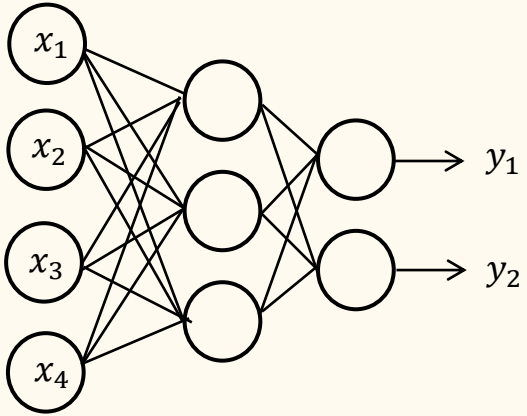
double layer



$$u_j = \sum_{i=1}^{I} w_{ji}x_i + b_i, \quad z_j = f(u_j) \qquad \text{or}$$

$$\boldsymbol{u} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \quad \boldsymbol{z} = \boldsymbol{f}(\boldsymbol{u})$$

Now it is (conceptually) easy to extend to multi-layer



Many layers

→ Deep neural network

Learning

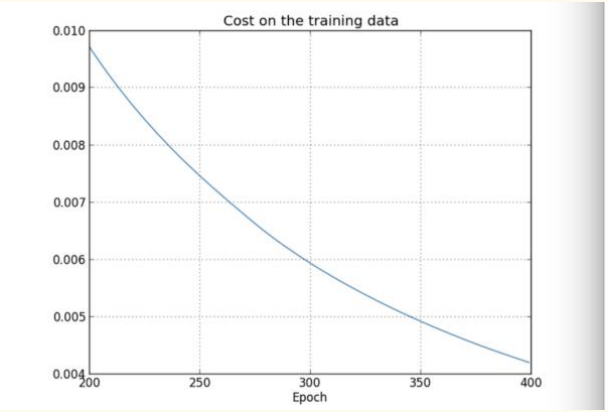
input $\boldsymbol{x} \rightarrow$ desired output $\boldsymbol{d}$ training data $\{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{d}_N)\}$

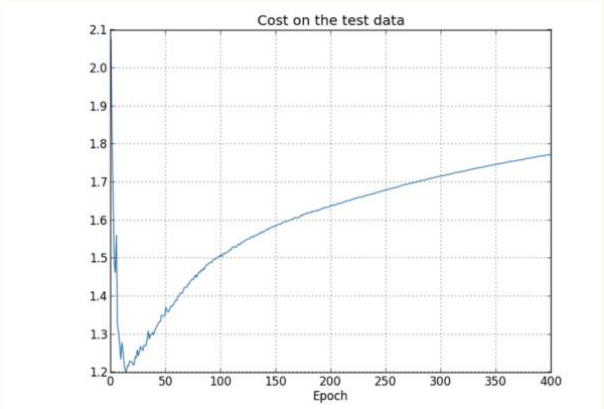
training samples

Learning: tune $\boldsymbol{w}$ to minimize distance between $\{\boldsymbol{d}_j\}$ and the network output $\{\boldsymbol{y}(\boldsymbol{x}_j, \boldsymbol{w})\}$

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{n=1}^{N} ||\boldsymbol{d}_n - \boldsymbol{y}(\boldsymbol{x}_n, \boldsymbol{w})||^2$$: typical error function → minimize

Overfitting (overlearning) problem:





Trapping to local minima

https://towardsdatascience.com/

Various remedies for overfitting

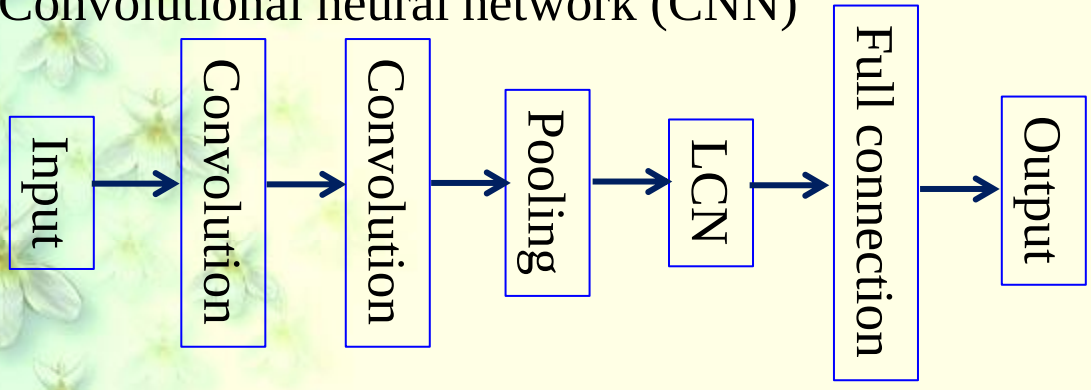
- Back propagation → However this leads to the vanishing gradient problem

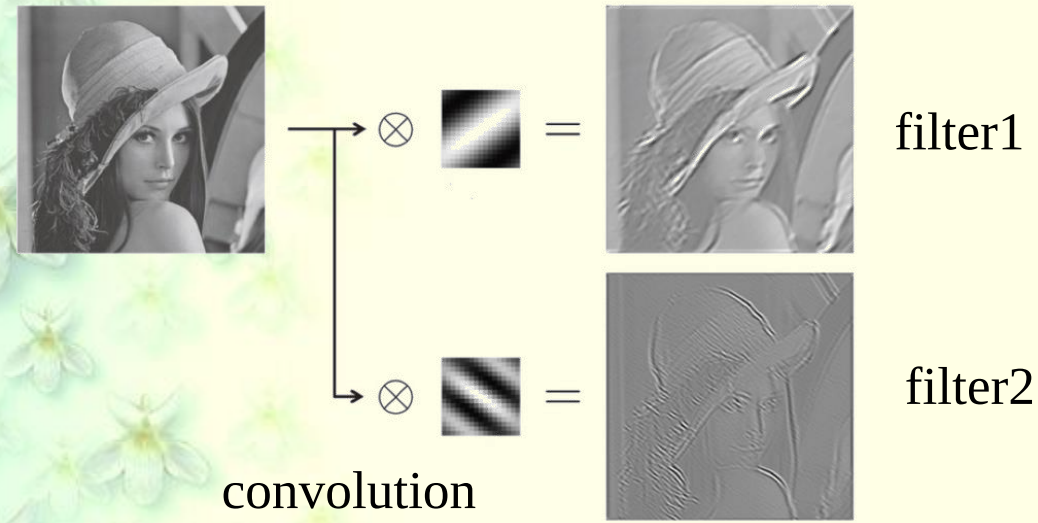
- Pretraining
  Deep belief network (DBN)
  → decompose into restricted Boltzmann machine (RBM)
  → learning for each RBM
  → Transfer to feedforward NN

  Autoencorder

Exceptionally “learnable” NN without pretraining

Convolutional neural network (CNN)





Input: $W \times W$ image

elements: x_{ij}

Filter: $H \times H$ image

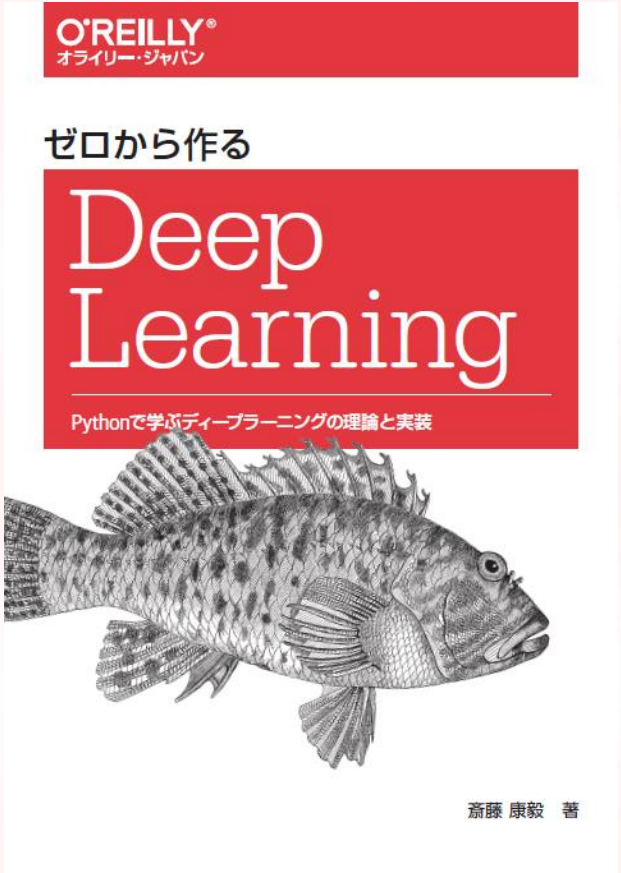
elements: h_{ij}

$$u_{ij} = \sum_{p=0}^{H-1} \sum_{q=0}^{H-1} x_{i+p,j+q} h_{pq}$$

Pooling

- Average
- Max
- Lp (some point between above)

Deep learning textbook (Japanese only)

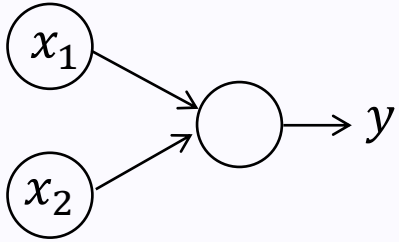


Deep learning from scratch

- Explains from introduction of Python to your PC
- Concise and to the points
- Python examples

Perceptron expression for logic gates

Perceptron



$$y = \begin{cases} 0 & (w_1 x_1 + w_2 x_2 \leq \theta), \\ 1 & (w_1 x_1 + w_2 x_2 > \theta). \end{cases}$$

AND

$(w_1, w_2, \theta) = (0.5, 0.5, 0.7)$: just an example

NAND

$(w_1, w_2, \theta) = (-0.5, -0.5, -0.7)$

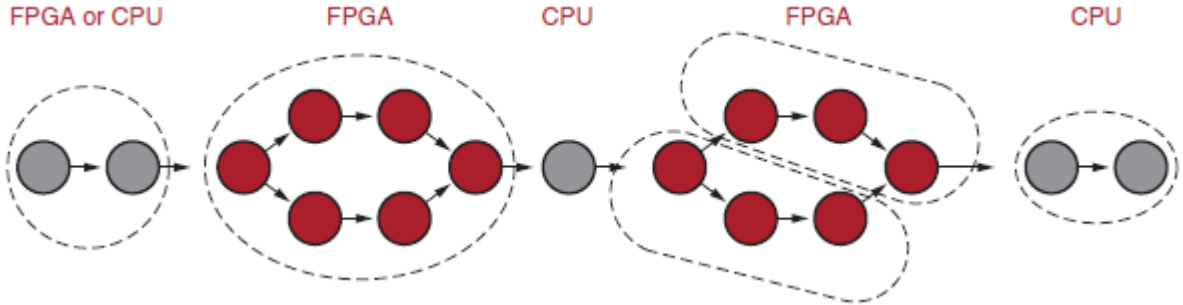
OR

$(w_1, w_2, \theta) = (0.6, 0.6, 0.5)$

Why FPGA fits for DNNs?

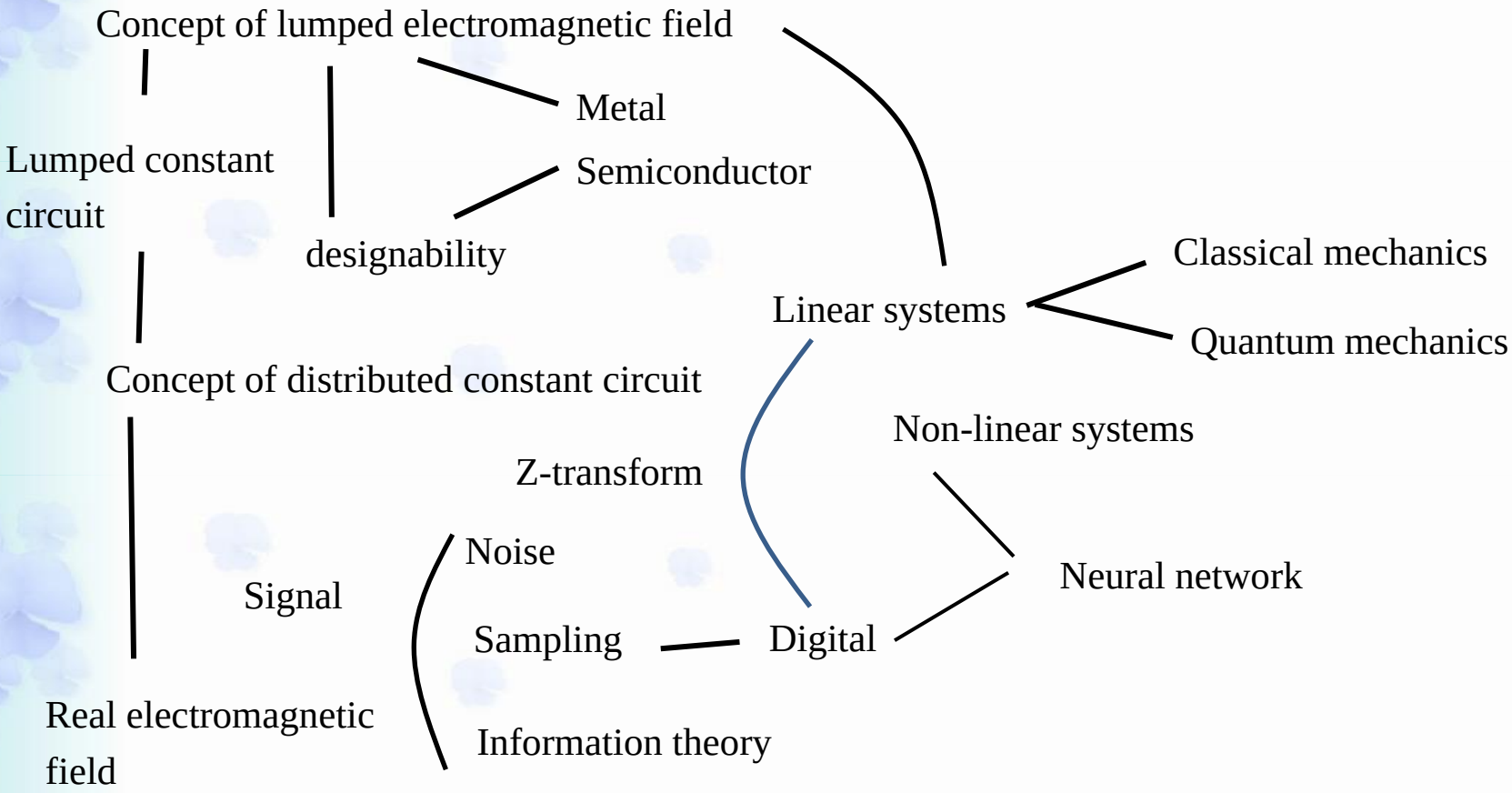
Layer by layer learning calculation → can be done in parallel naturally.
Calculation along transition lines → also can be done in parallel.

FPGA card specialized for DNNs (Xilinx)



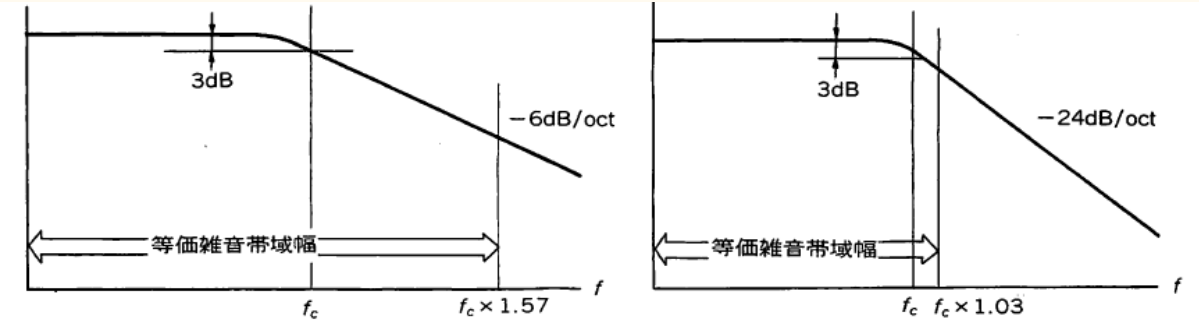
Overview

Electric Circuits: Treasury of Languages and Concepts



Supplement for amplifier and noise

Equivalent noise band width (ENBW)

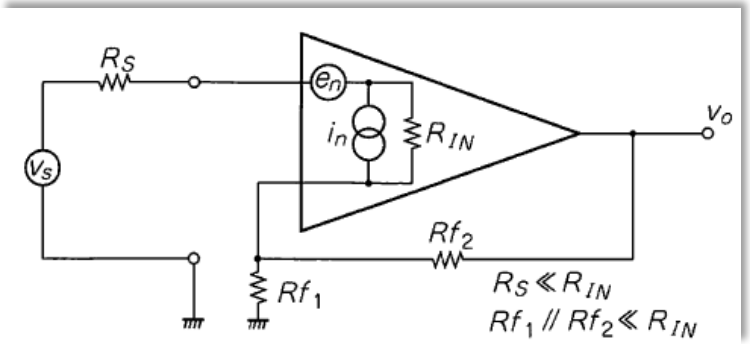


Attenuation gradient	k
-6 db/oct	1.57
-12 db/oct	1.11
-18 db/oct	1.05
-24 db/oct	1.03

gain band width GBW $= A \times f_c$

OP amp. spec. sheet: (open loop gain) × (cut off freq.)

Supplement amplifier and noise



1. R_S thermal noise = $\sqrt{4kTR_S}$
2. $Rf_1 \parallel Rf_2$ thermal noise = $\sqrt{4kT(Rf_1 \parallel Rf_2)}$
3. Input voltage noise e_n, current noise i_n
4. Current noise times source resistance = $i_n \times R_S$
5. Current noise times feedback resistance = $i_n \times (Rf_1 \parallel Rf_2)$

$$\text{(output noise)} = \sqrt{{e_1}^2 + {e_2}^2 + {e_3}^2 + {e_4}^2 + {e_5}^2} \times G \times \sqrt{\text{ENBW}}$$