# Netradicinės ir ateities architektūros

### Saulius Gražulis

### Vilnius, 2021

Vilniaus universitetas, Matematikos ir informatikos fakultetas Informatikos institutas



Šį skaidrių rinkinį galima kopijuoti, kaip nurodyta Creative Commons Attribution-ShareAlike 4.0 International licenzijoje



Saulius Gražulis

Netradicinės ir ateities architektūros

3 🕨 🖌 3

### **ENIAC**

### ENIAC – had to be rewired...



#### U.S. Army Photo, Public Domain

Saulius Gražulis

▶ ▲ 문 ▶ ▲ 문 ▶

### Harvardo/fon Noimano kompiuteriai

SSEM "Manchester Baby" – first (?) stored program vacuum tube computer...



By Parrot of Doom, CC BY-SA 3.0, via Wikimedia Commons

Saulius Gražulis

Netradicinės ir ateities architektūros



Vilnius, 2021 4 / 26



Vilnius, 2021 4 / 26



(Groves 1995)



In its simplest form a von Neumann computer has three parts: a central processing unit (or CPU), a store, and a connecting tube that can transmit a single word between the CPU and the store (and send an address to the store). I propose to call this tube **the von Neumann bottleneck**.

John Backus, 1977 ACM Turing Award Lecture (Backus 1978)

### UMA: Uniform Memory Access (Groves 1995)



3

イロト イポト イヨト イヨト

### UMA

### NUMA: Non-Uniform Memory Access (Groves 1995)

#### NUMA node 0



### NUMA: Non-Uniform Memory Access (Groves 1995)

NUMA node 0



Saulius Gražulis

Netradicinės ir ateities architektūros

### NORMA

### NORMA: No Remote Memory Access (Groves 1995)



Saulius Gražulis

Netradicinės ir ateities architektūros

э

# Programming parallel machines

• For NORMA/MUMA/UMA:

MPI: Message Passing Interface (https://www.open-mpi.org/)

• For NUMA/UMA:

OpenMP: Open Multi-Processing API (https://www.openmp.org/)

OpenMP example:

```
#include <stdio.h>
#define N 10000000LL
int main(int argc, char *argv[]) {
   static long long a[N];
   long long i;

   #pragma omp parallel for
   for (i = 0; i < N; i++)
        a[i] = 2 * i;

   printf( "%lld\n", a[N-1LL] );
   return 0;
}</pre>
```

```
cc \
    -fopenmp \
    -Wall \
    -03 \
    -fomit-frame-pointer \
    -funroll-loops \
    -o loop \
loop.c
```

- 本間 と くき とくき とうき

### Brain wiring



http://www.flycircuit.tw (Chiang et al. 2011)

Saulius Gražulis

æ

▶ ▲ 문 ▶ ▲ 문 ▶

< A

McCulloch-Pitts neuron (McCulloch et al. 1943; Alom et al. 2018):



# Neural Networks



æ

□ > < E > < E >

### Neurons as logic gates



(Minsky 1967)

æ

12/26

AP > < = > < =

## Perceptron

# Frank Rosenblatt's Perceptron (Rosenblatt 1957):



# Deep learning ANNs (Alom et al. 2018):



### Single layer

#### Multilayer

Saulius Gražulis

Netradicinės ir ateities architektūros

Vilnius, 2021 13/26



Adapted from (Brown et al. 2000)

Saulius Gražulis

Vilnius, 2021 14/26

æ

イロト イロト イヨト イヨト



Adapted from (Brown et al. 2000)

Saulius Gražulis

Vilnius, 2021 14 / 26

æ

イロト イロト イヨト イヨト

### Lattice Semiconductor FPGA

iCE40LP/HX1K Device, Top View



#### (Lattice Semiconductor 2017)

Saulius Gražulis

イロト イポト イモト イモト

## Lattice Semiconductor FPGA

#### PLB Block Diagram



#### (Lattice Semiconductor 2017)

Saulius Gražulis

Vilnius, 2021

(日) (四) (日) (日) (日)

#### sysMEM Memory Primitives



#### (Lattice Semiconductor 2017)

Saulius Gražulis

Vilnius, 2021

< □ > < 同 > < 回 > < 回 > < 回 >

This reference design implements Convolutional Neural Network (CNN) based human face identification on Lattice's low power ECP5 FPGA using an image sensor.

Lattice Semiconductor Reference Designs

Features:

- VGG8 like 8x (Convolution, Batch Normalisation) + 4x Pooling + 1 fully connected CNN
- Runs at 2 frames per second with 90 x 90 RGB Input
- Total ECP5 power consumption of 850mW

### CPUs can be implemented in FPGA:



(Caska et al. 2011; Schoeberl 2011)

- Verilog (https://en.wikipedia.org/wiki/Verilog)
- VHDL (https://en.wikipedia.org/wiki/VHDL)
- Chissel (https://www.chisel-lang.org/)

Project stages/system capabilities

- Describe
- 2 Simulate
- Verify
- Synthesise (for FPGA or Silicon foundry)

# Verilog example

```
module rng (
            input clk,
            output LED1.
            output LED2,
            output LED3,
            output LED4.
            output LED5
            ):
   localparam BITS = 5;
   localparam LOG2DELAY = 22;
   reg [BITS+LOG2DELAY-1:0] counter = 0;
   reg
                             ready = 0;
   reg [31:0]
                             rng;
   always@(posedge clk)
     counter <= counter + 1;
   always@(posedge counter[LOG2DELAY-2])
     if( ready )
       begin
          rng <= ({rng[0],(rng >> 1)})^(rng | {(rng << 1),rng[31]});</pre>
     else
       begin
          rng = 32'h00010000;
          ready = 1;
       end
   assign {LED1, LED2, LED3, LED4, LED5} = rng[11:7];
endmodule
```

#### https://github.com/RGD2/icestorm\_example

3

# Verilog example

```
module rng (
                                    saulius@tasmanijos-velnias verilog/ $ make -n upload
            input clk,
                                    yosys -p "read_verilog_rng.v;_synth_ice40,_blif_rng.blif"
            output LED1.
                                    arachne-pnr -d 1k -p rng.pcf -o rng.txt rng.blif
            output LED2,
                                    icepack rng.txt rng.bin
            output LED3,
                                    iceprog rng.bin
            output LED4.
            output LED5
            ):
   localparam BITS = 5;
   localparam LOG2DELAY = 22:
   reg [BITS+LOG2DELAY-1:0] counter = 0;
   reg
                             ready = 0;
   reg [31:0]
                            rng;
   always@(posedge clk)
     counter <= counter + 1;
   always@(posedge counter[LOG2DELAY-2])
     if( ready )
       begin
          rng <= ({rng[0],(rng >> 1)})^(rng | {(rng << 1),rng[31]});</pre>
     else
       begin
          rng = 32'h00010000:
          readv = 1:
       end
   assign {LED1, LED2, LED3, LED4, LED5} = rng[11:7]:
endmodule
```

#### https://github.com/RGD2/icestorm\_example

イロト イポト イヨト イヨト

# Verilog example

```
saulius@tasmanijos-velnias verilog/ $ make simulate
module rng (
                                        iverilog simulate.v
            input clk,
                                        ./a.out
            output LED1.
                                        Begin Simulation
            output LED2,
                                                                    0, LEDS = x x x x x
                                        At time
            output LED3,
                                        At time
                                                              2097151, LEDS = 0 0 0 0 0
            output LED4.
                                        At time
                                                            23068671, LEDS = 1 0 0 0 0
            output LED5
                                                            27262975, LEDS = 0 1 0 0 0
            ):
                                        At time
                                                            31457279, LEDS = 1 1 1 0 0
                                        At time
   localparam BITS = 5;
                                                            35651583, LEDS = 0 0 0 1 0
                                        At time
   localparam LOG2DELAY = 22:
                                        ^C** VVP Stop(0) **
   reg [BITS+LOG2DELAY-1:0] counter
                                        ** Flushing output streams.
                             ready =
   reg
                                        ** Current simulation time is 39064597 ticks.
   reg [31:0]
                             rng:
   always@(posedge clk)
                                        > finish
     counter <= counter + 1;
   always@(posedge counter[LOG2DELAY-2])
     if( readv )
       begin
          rng <= ({rng[0],(rng >> 1)})^(rng | {(rng << 1),rng[31]});</pre>
       end
     else
       begin
          rng = 32'h00010000:
          readv = 1:
       end
   assign {LED1, LED2, LED3, LED4, LED5} = rng[11:7]:
endmodule
```

#### https://github.com/RGD2/icestorm\_example

イロト イポト イヨト イヨト

## **Open Cores**

### https://opencores.org/

| → C' 🏠              | 🗊 🔒 https://opencores.org/projects?expanded=  | Arithmetic o   | ore  |                                       |                                     |                   |
|---------------------|---|--|--|---------------------------------------|-------------------------------------|-------------------|
| DuckDuckGo 🔓 Google | 💥 COD 👷 TCOD 🕲 PCOD 🧥 Redmine 🌘 Europe PMC  | 8 PubMi  | ad 🔮 wwPDB   | W Wikipedia 🛞 Wiktion                 | nary 📓 RCSB PDB 🍈 P                 | DBe 🚥 PDBj 👩 SG 🛛 |
| 1 Onon              | Comos Of  | Ve?  | A  | <b>b.</b>                             |                                     |                   |
| Opend               | lores   |  | Service Servic | 1                                     |                                     |                   |
|                     |   |  | -  |                                       |                                     |                   |
| Username:           |   |  |  |                                       |                                     |                   |
|                     |   |  |  |                                       |                                     |                   |
| Password:           |   |  |  |                                       |                                     |                   |
|                     |   |  |  |                                       |                                     |                   |
| Remember me         |   |  |  |                                       |                                     |                   |
| Login               | Written in: Any language 🗸 Stage: Any stage 🗸   | License:   | Any license  | <ul> <li>Wishbone version:</li> </ul> | Any version 🛩                       |                   |
| Register            |   |  |  |                                       |                                     |                   |
| Browse              | ASIC proven Design done FPGA proven   | Specific | ification done   | OpenCores Certifie                    | sd 👕                                |                   |
| PROJECTS            | •   |  |  |                                       |                                     |                   |
| FORUMS              | Arithmetic core 107   |  |  |                                       |                                     |                   |
| ABOUT               | Project   | Files  | Statistics   | Status                                | License                             | Wishbone version  |
| HOWTO/FAQ<br>MEDIA  | 1 bit adpcm codec   | •  | Stats  |                                       | LGPL                                |                   |
| LICENSING           | 2D FHT  | •  | Stats  |                                       | LGPL                                |                   |
| COMMERCE            | 4-bit system  | •  | Stats  |                                       | LGPL                                |                   |
| PARTNERS            | 5x4Gbps CRC generator designed with standard cells  | •  | Stats  | done                                  | GPL                                 |                   |
| MAINTAINERS         | 8 bit Vedic Multiplier  | •  | Stats  | done                                  | LGPL                                |                   |
|                     | Adder library   | •  | Stats  |                                       | Others                              |                   |
|                     | AES128  | •  | Stats  | done                                  | LGPL                                |                   |
|                     | ANN   | •  | Stats  |                                       | LGPL                                |                   |
|                     | Anti-Logarithm (square-root), base-2, single-cycle  | •  | Stats  | done                                  | LGPL                                |                   |
|                     | BCD adder   | •  | Stats  |                                       | LGPL                                |                   |
|                     | Binary to BCD conversions, with LED display driver  | •  | Stats  |                                       |                                     |                   |
|                     | Bluespec SystemVerilog Reed Solomon Decoder   | •  | Stats  |                                       | LGPL                                |                   |
|                     |   |  |  |                                       |                                     |                   |
|                     | Booth Array Multiplier  | •  | Stats  |                                       | LGPL                                |                   |
|                     | Booth Array Multiplier<br>cavic decoder   | :  | Stats<br>Stats   | done                                  | LGPL                                |                   |
|                     | Booth Array Multiplier<br>cavic decoder<br>Cellular Automata PRNG   | :  | Stats<br>Stats<br>Stats  | done<br>done                          | LGPL<br>LGPL<br>BSD                 |                   |
|                     | Booth Array Multiplier<br>cavlc.decoder<br>Cellular Automata PRNG<br>CF.Cordic  |  | Stats<br>Stats<br>Stats<br>Stats   | (done)                                | LGPL<br>LGPL<br>BSD                 |                   |
|                     | Booth Array Multiplier<br>cavic decoder<br>Cellular Automata PRNG<br>CF Cordic<br>CF FET  |  | Stats<br>Stats<br>Stats<br>Stats<br>Stats  | done<br>done                          | LGPL<br>LGPL<br>BSD                 |                   |
|                     | Booth Array Multiplier<br>cavic decoder<br>Cellular Automata PRNG<br>CF Cordic<br>CF Fioting Point Multiplier   |  | Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats   | done<br>done                          | LGPL<br>LGPL<br>BSD                 |                   |
|                     | Sooth Array Multiplier<br>Savis decoder<br>Savis decoder<br>CF Cordis<br>CF FET<br>CF Fiscation Point Multiplier<br>CF monitex Arthmetic Operations   |  | Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats   | done<br>done                          | LGPL<br>LGPL<br>BSD                 |                   |
|                     | Booth Array Multiplier<br>cavic Geoder<br>Cellular Automata PRNG<br>CF Cortillo<br>CF EFF<br>CF Filoating Point Multiplier<br>Complex Authoritic Operations<br>Complex Gaussian Resulto-random Number Generator |  | Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats<br>Stats  | done<br>done                          | LGPL<br>LGPL<br>BSD<br>LGPL<br>LGPL |                   |

Saulius Gražulis

Netradicinės ir ateities architektūros

æ

- Hall, A. Short-Read DNA Sequence Alignment with Custom Designed FPGA-based Hardware (Hall 2010);
- FPGA based molecular dynamics: (Khan et al. 2012; Yang et al. 2019; Waidyasooriya et al. 2016).

Cell Matrix



https://cellmatrix.com/entryway/entryway/branchAbout.html https://www.cellmatrix.com

Saulius Gražulis

Netradicinės ir ateities architektūros

Vilnius, 2021

### Cell Matrix



FIGURE 3 - A SINGLE CELL MATRIX CELL

https://cellmatrix.com/entryway/entryway/branchAbout.html https://www.cellmatrix.com

Saulius Gražulis

Netradicinės ir ateities architektūros

Vilnius, 20<u>21</u>

- Cellular automata (e.g. J. H. Conway's "Life"); Turing complete!
- DNA data storage
- DNA computing

# Šaltiniai I

Alom, Md Zahangir et al. (Mar. 3, 2018). "The history began from AlexNet: a comprehensive survey on deep learning approaches". In: *arXiv*, pp. 1–39. arXiv: 1803.01164 [cs.CV]. URL: https://arxiv.org/abs/1803.01164v1.

- Backus, John (1978). "Can programming be liberated from the von Neumann style?: a functional style and its algebra of programs". In: *Communications of the ACM*. Vol. 21. 8. Association of Computing Machinery, pp. 613–641. URL: https://dl.acm.org/doi/10.1145/1283920.1283933.
- Brown, Stephen et al. (2000). Architecture of FPGAs and CPLDs: A Tutorial. Tech. rep. Department of Electrical and Computer Engineering, University of Toronto. URL: http://www.eecg.toronto.edu/~jayar/pubs/brown/survey.pdf.
- Caska, James et al. (2011). "Java dust: how small can embedded Java be?" In: Proceedings of the 9th International Workshop on Java Technologies for Real-Time and Embedded Systems - JTRES '11. ACM Press, pp. 1–5. DOI: 10.1145/2043910.2043931. URL: https://www.jopdesign.com/doc/lerosjvm.pdf.
- Chiang, Ann-Shyn et al. (Jan. 2011). "Three-dimensional reconstruction of brain-wide wiring networks in Drosophila at single-cell resolution". In: *Current Biology* 21.1, pp. 1–11. DOI: 10.1016/j.cub.2010.11.056.
- Groves, R. (1995). "Brief history of computer architecture evolution and future trends". en. In: 18th CERN School of Computing. CERN, pp. 147–159. DOI: 10.5170/CERN-1995-005.147. URL: https://cds.cern.ch/record/399391/files/p147.pdf.

イロト イポト イヨト 「ヨ

# Šaltiniai II

- Hall, Adam (Nov. 2010). "Short-Read DNA Sequence Alignment with Custom Designed FPGA-based Hardware". MA thesis. THE FACULTY OF GRADUATE STUDIES (Bioinformatics) THE UNIVERSITY OF BRITISH COLUMBIA (Vancouver). URL: https://open.library.ubc.ca/cIRcle/collections/ubctheses/24/items/1.0071441.
- Khan, M. A. et al. (2012). FPGA-accelerated molecular dynamics. URL: http://www.bu.edu/caadlab/Khan13.pdf.
- Lattice Semiconductor (2017). *iCE40<sup>TM</sup> LP/HX family data sheet*. Tech. rep. Lattice Semiconductor. URL: http://www.latticesemi.com/~/media/LatticeSemi/Documents/DataSheets/iCE/ iCE40LPHXFamilyDataSheet.pdf.
- McCulloch, Warren S. et al. (Dec. 1943). "A logical calculus of the ideas immanent in nervous activity". In: *The Bulletin of Mathematical Biophysics* 5.4, pp. 115–133. DOI: 10.1007/bf02478259. URL: https://www.cs.cmu.edu/~./epxing/Class/10715/reading/McCulloch.and.Pitts.pdf.
- Minsky, Marvin Lee (1967). Computation: finite and infinite machines. Prentice-Hall.
- Minsky, Marvin Lee and Seymour Papert (1969). *Perceptrons: an introduction to computational geometry*. MIT Press.
- Rosenblatt, Frank (1957). *The Perceptron: a perceiving and recognising automaton*. Tech. rep. Cornaell Aeronautical Laboratory, Inc., pp. 1–33. URL:
  - https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf.

25/26

イロト 不得 トイヨト イヨト 二日

Schoeberl, Martin (2011). Leros: a tiny microcontroller for FPGAs. URL: https://www.jopdesign.com/doc/leros.pdf.

Waidyasooriya, Hasitha Muthumala et al. (June 2016). "Architecture of an FPGA accelerator for molecular dynamics simulation using OpenCL". In: 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS). IEEE, p. 7550743. DOI: 10.1109/icis.2016.7550743.

Yang, Chen et al. (May 14, 2019). "Fully integrated on-FPGA molecular dynamics simulations". In: ArXiv, p. 190505359. arXiv: http://arxiv.org/abs/1905.05359v1 [cs.DC]. URL: https://arxiv.org/pdf/1905.05359.pdf.

• • = • • =