Non-traditional architectures and the future

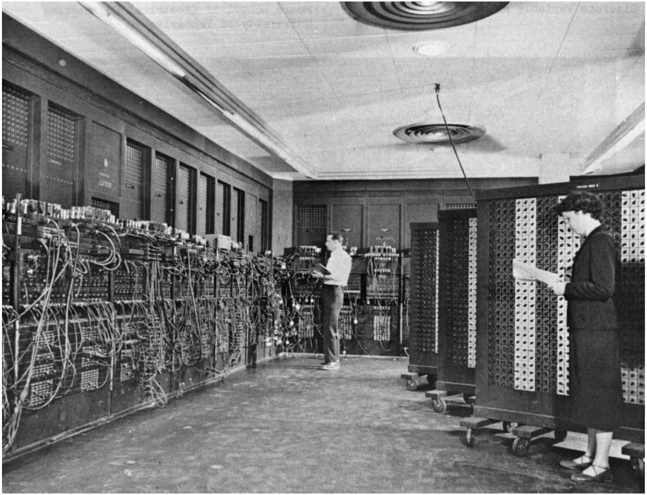
Saulius Gražulis

Vilnius, 2024

Vilnius University, Faculty of Mathematics and Informatics
Institute of Informatics

ENIAC

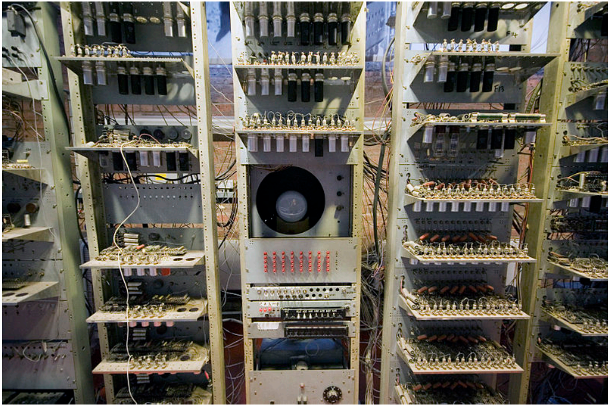
ENIAC – had to be rewired...



U.S. Army Photo, Public Domain

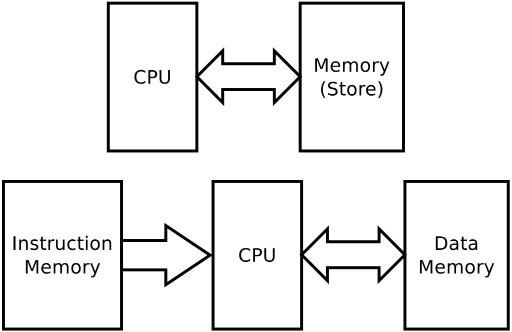
Harvard/von Neumann stored program computers

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

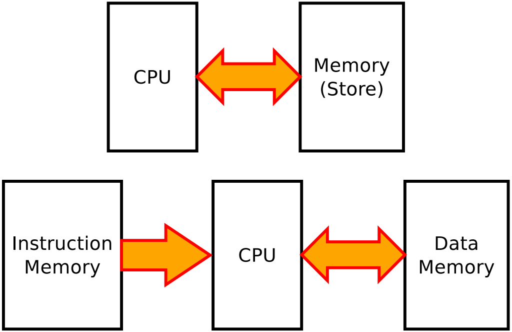


By Parrot of Doom, CC BY-SA 3.0, via Wikimedia Commons

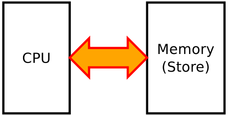
Von Neumann architecture bottleneck

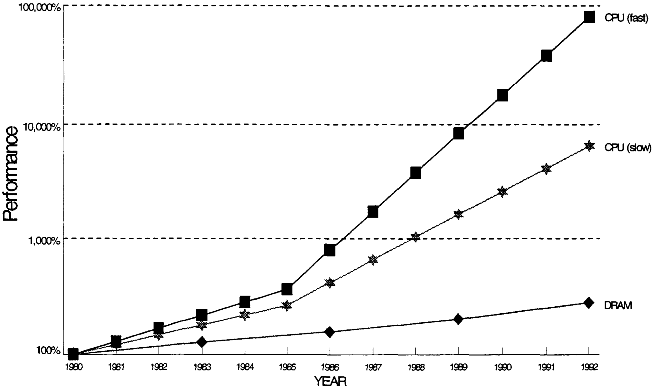


Von Neumann architecture bottleneck



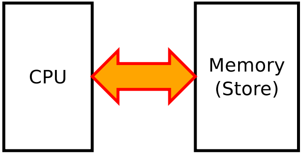
Von Neumann architecture bottleneck





(Groves 1995)

Von Neumann architecture bottleneck

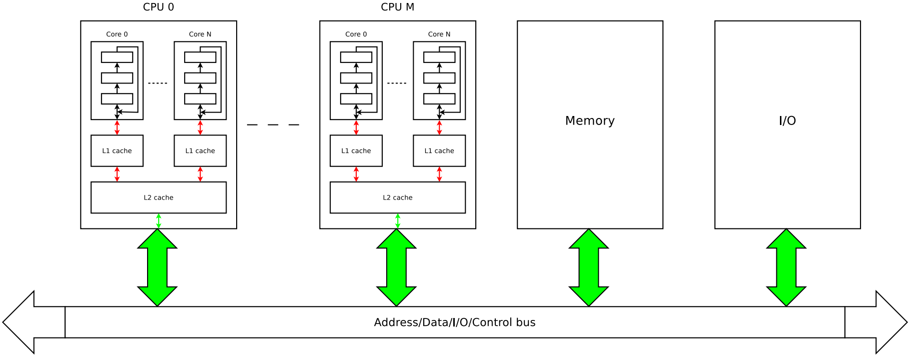


In its simplest form a von Neumann computer has three parts: a central processing unit (or CPU), a store, and a connecting tube that can transmit a single word between the CPU and the store (and send an address to the store). I propose to call this tube **the von Neumann bottleneck**.

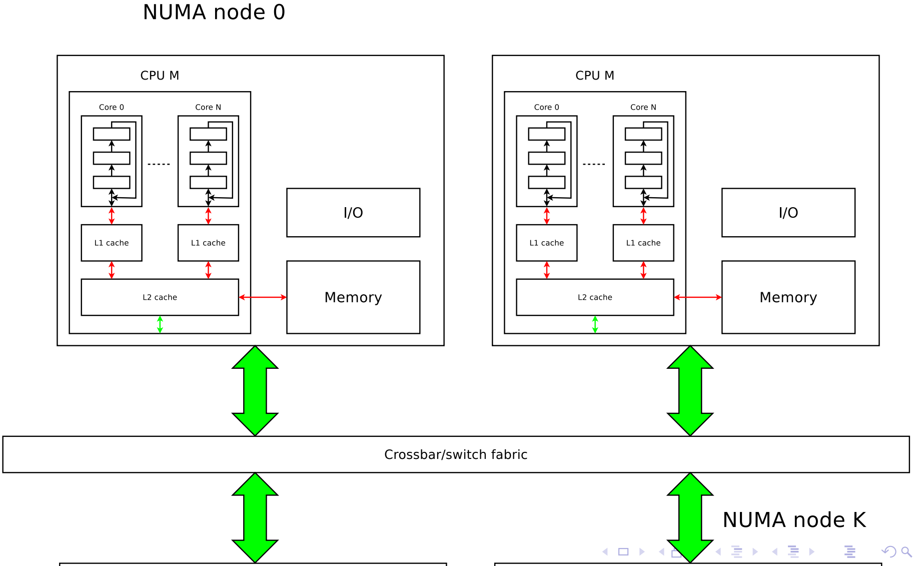
John Backus, 1977 ACM Turing Award Lecture (Backus 1978)

UMA

UMA: Uniform Memory Access (Groves 1995)

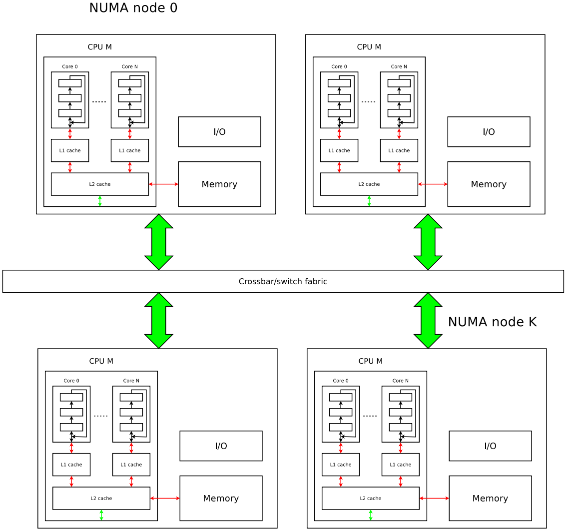


NUMA: Non-Uniform Memory Access (Groves 1995)



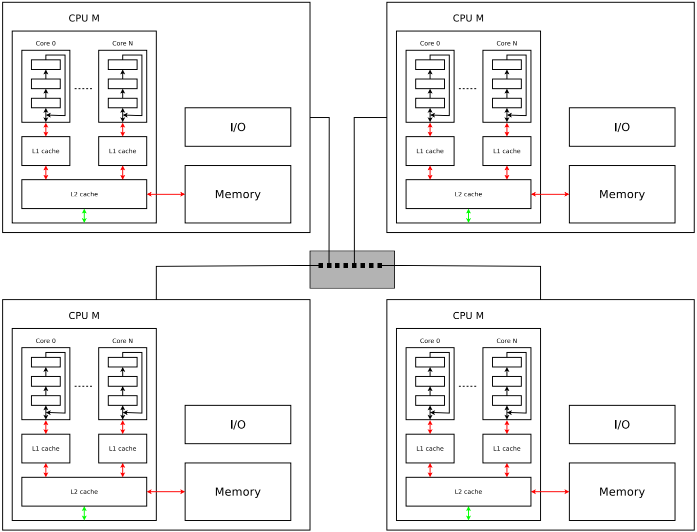
UMA

NUMA: Non-Uniform Memory Access (Groves 1995)



NORMA

NORMA: No Remote Memory Access (Groves 1995)



Programming parallel machines

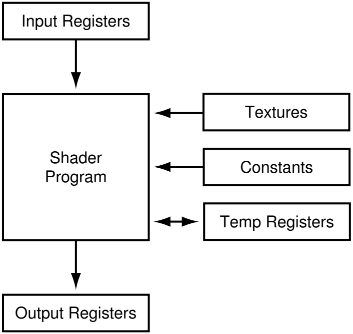
- For NORMA/MUMA/UMA:
  MPI: Message Passing Interface (https://www.open-mpi.org/)
- For NUMA/UMA:
  OpenMP: Open Multi-Processing API (https://www.openmp.org/)

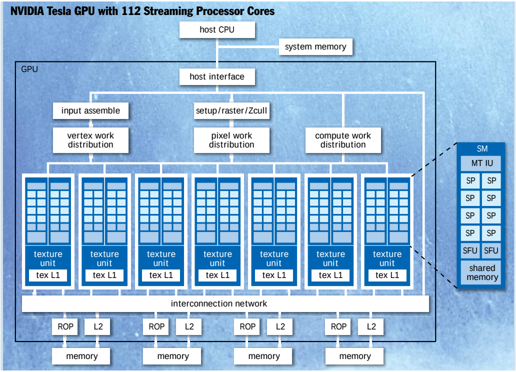
OpenMP example:

```c
#include <stdio.h>
#define N 100000000LL
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;
}
```

```
cc \
    -fopenmp \
    -Wall \
    -O3 \
    -fomit-frame-pointer \
    -funroll-loops \
    -o loop \
loop.c
```

GPU processors





(Buck et al. 2004; Nickolls et al. 2008)

Programming GPUs

Computing y ← ax + y with a Serial Loop

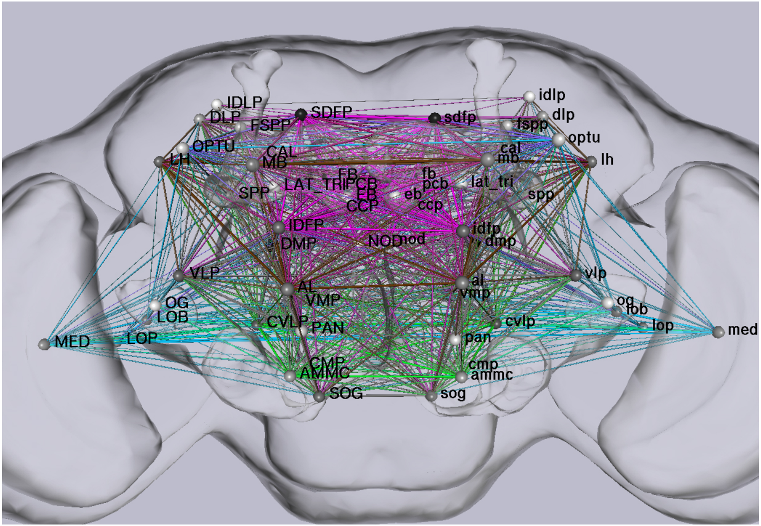
```
void saxpy_serial(int n, float alpha, float *x, float *y)
{
    for(int i = 0; i<n; ++i)
        y[i] = alpha*x[i] + y[i];
}

// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);
```

Computing y ← ax + y in parallel using CUDA

```
__global__
void saxpy_parallel(int n, float alpha, float *x, float *y)
{
    int i = blockIdx.x*blockDim.x + threadIdx.x;

    if( i<n )  y[i] = alpha*x[i] + y[i];
}

// Invoke parallel SAXPY kernel (256 threads per block)
int nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

(Nickolls et al. 2008)

Brain wiring

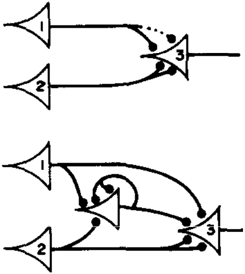


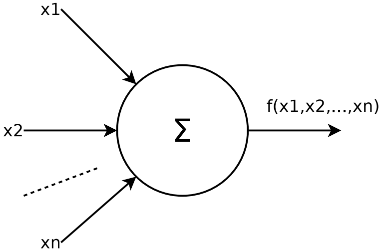
http://www.flycircuit.tw (Chiang et al. 2011)

Neuron models

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

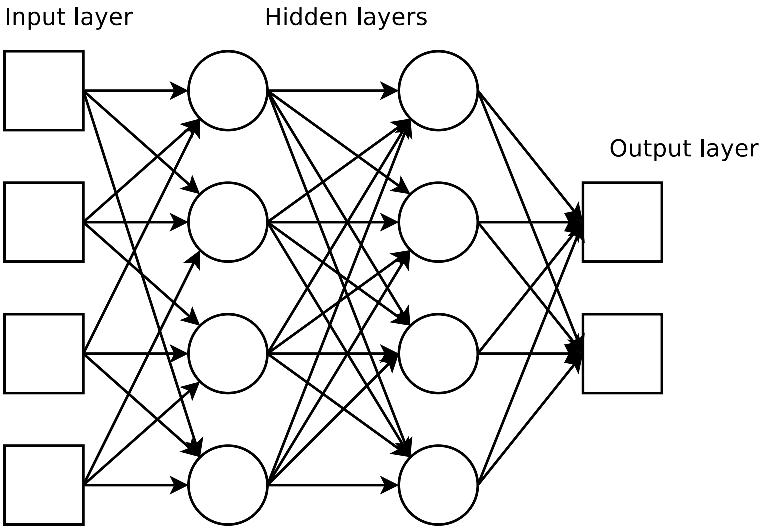
(i)



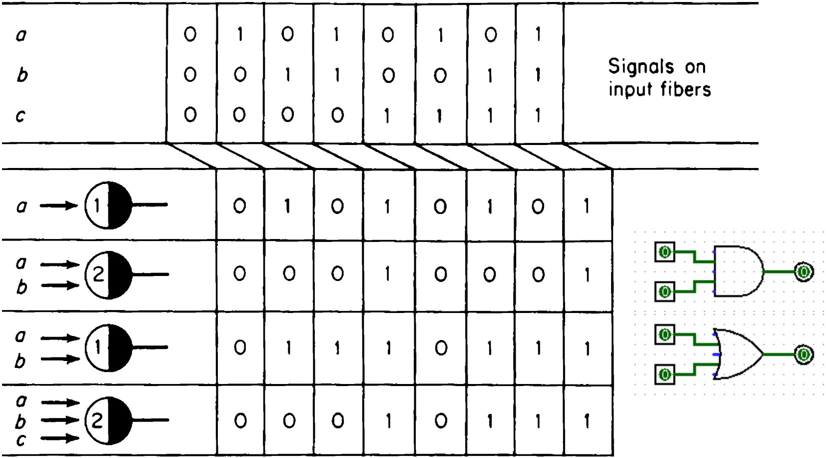


$$f(x_1, x_2, \ldots, x_n) = \varphi\left(b + \sum_{i=1}^{n} w_i x_i\right)$$

Neural Networks



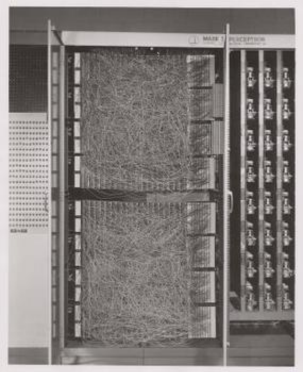
Neurons as logic gates



(Minsky 1967)

Perceptron

Frank Rosenblatt's Perceptron (Rosenblatt 1957):



Deep learning ANNs (Alom et al. 2018):

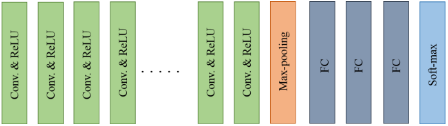


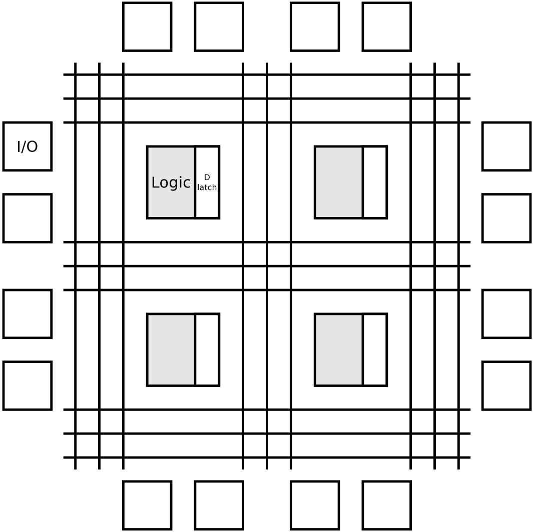
Fig. 15. Basic building block of VGG network: Convolution (Conv) and FC for fully connected layers

Single layer

Multilayer

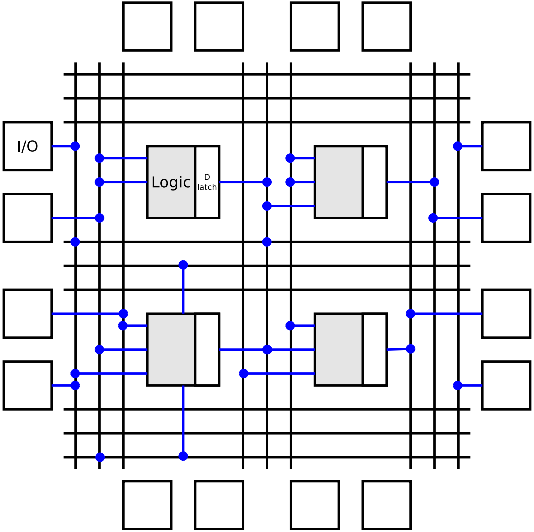
Single layer can not do XOR: (Minsky and Papert 1969)

FPGA



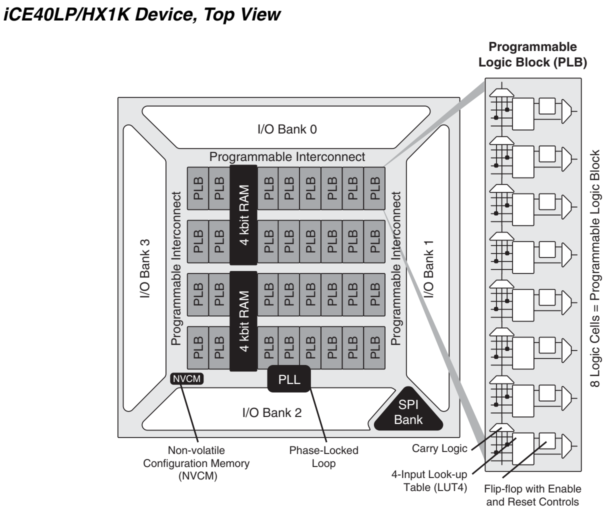
Adapted from (Brown et al. 2000)

FPGA



Adapted from (Brown et al. 2000)

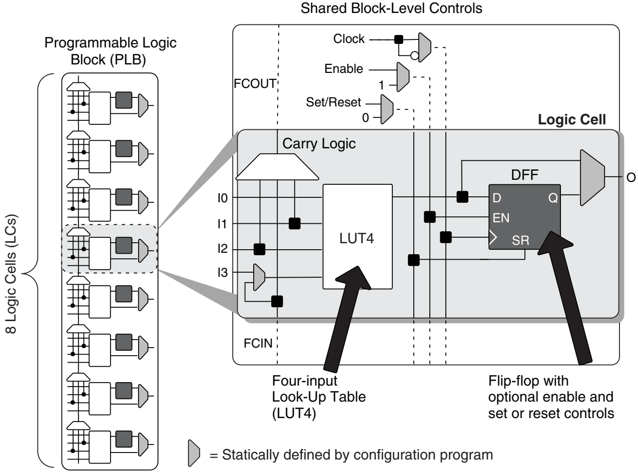
Lattice Semiconductor FPGA



(Lattice Semiconductor 2017)

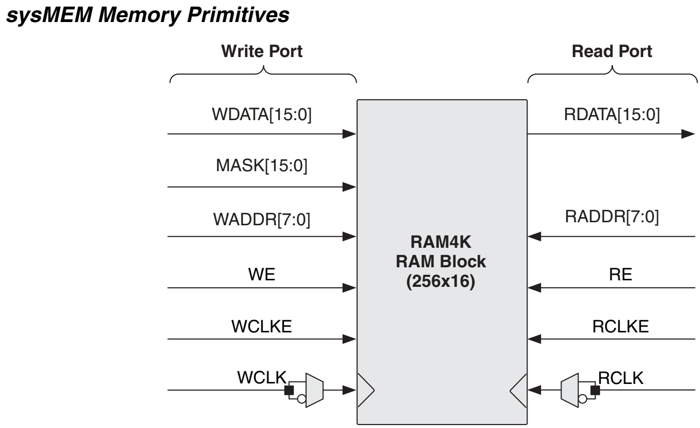
Lattice Semiconductor FPGA

PLB Block Diagram



(Lattice Semiconductor 2017)

Lattice Semiconductor FPGA



(Lattice Semiconductor 2017)

FPGA ANN

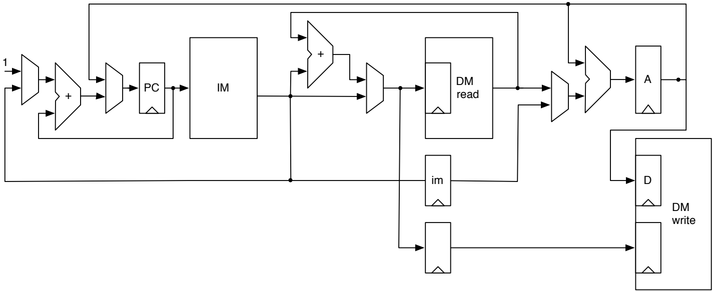
> *This reference design implements Convolutional Neural Network (CNN) based human face identification on Lattice's low power ECP5 FPGA using an image sensor.*

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)

Hardware description languages

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

Project stages/system capabilities

1. Describe
2. Simulate
3. Verify
4. 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]});
       end
     else
       begin
          rng = 32'h00010000;
          ready = 1;
       end
   assign {LED1, LED2, LED3, LED4, LED5} = rng[11:7];
endmodule
```

https://github.com/RGD2/icestorm_example

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]});
       end
     else
       begin
          rng = 32'h00010000;
          ready = 1;
       end
   assign {LED1, LED2, LED3, LED4, LED5} = rng[11:7];
endmodule
```

```
saulius@tasmanijos-velnias verilog/ $ make -n upload
yosys -p "read_verilog rng.v; synth_ice40 -blif rng.blif"
arachne-pnr -d 1k -p rng.pcf -o rng.txt rng.blif
icepack rng.txt rng.bin
iceprog rng.bin
```

https://github.com/RGD2/icestorm_example

Verilog example

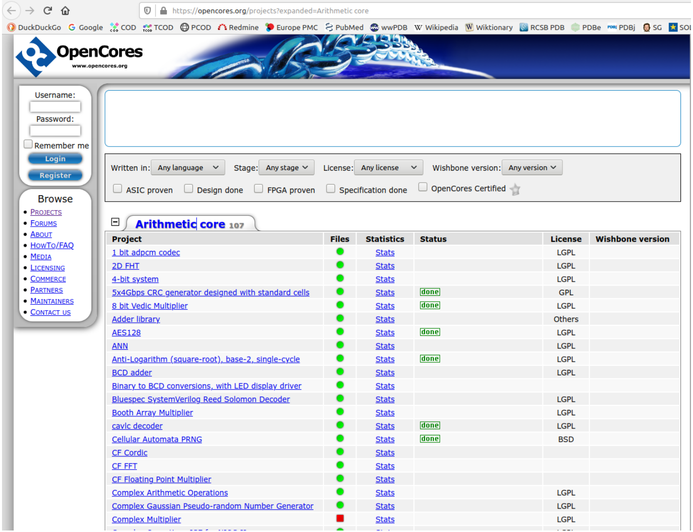
```
module rng (
            input  clk,
            output LED1,
            output LED2,
            output LED3,
            output LED4,
            output LED5
            );
   localparam BITS = 5;
   localparam LOG2DELAY = 22;
   reg [BITS+LOG2DELAY-1:0] counter =
   reg                      ready = (
   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]});
       end
     else
       begin
          rng = 32'h00010000;
          ready = 1;
       end
   assign {LED1, LED2, LED3, LED4, LED5} = rng[11:7];
endmodule
```

```
saulius@tasmanijos-velnias verilog/ $ make simulate
iverilog simulate.v
./a.out
Begin Simulation
At time                    0, LEDS = x x x x x
At time              2097151, LEDS = 0 0 0 0 0
At time             23068671, LEDS = 1 0 0 0 0
At time             27262975, LEDS = 0 1 0 0 0
At time             31457279, LEDS = 1 1 1 0 0
At time             35651583, LEDS = 0 0 0 1 0
^C** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 39064597 ticks.
> finish
```

https://github.com/RGD2/icestorm_example

Open Cores

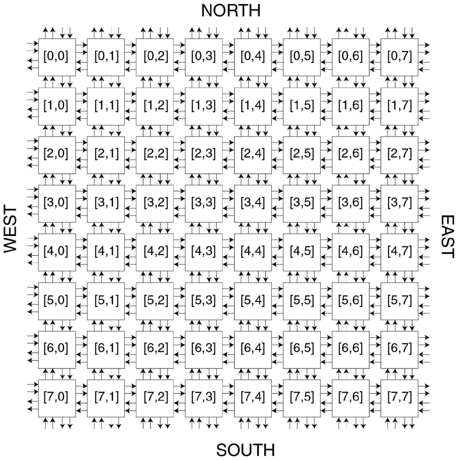
https://opencores.org/



FPGA for bioinformatics

- 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

Cell Matrix

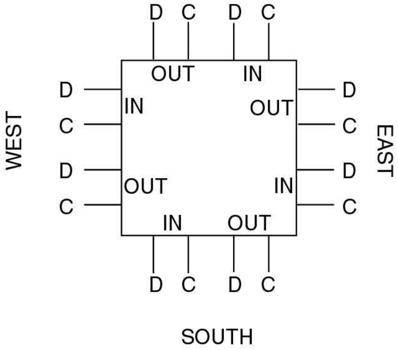


FIGURE 3 - A SINGLE CELL MATRIX CELL

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

https://www.cellmatrix.com

Systolic processors

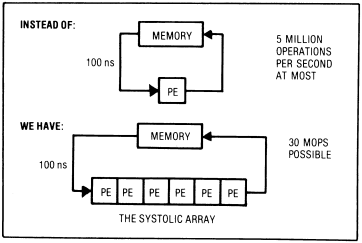


Figure 1. Basic principle of a systolic system.

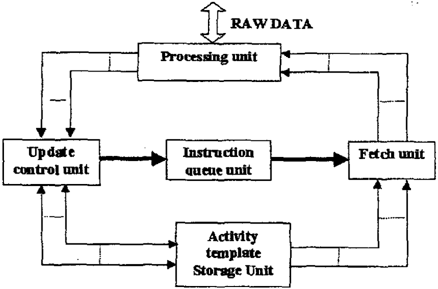
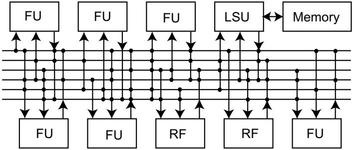


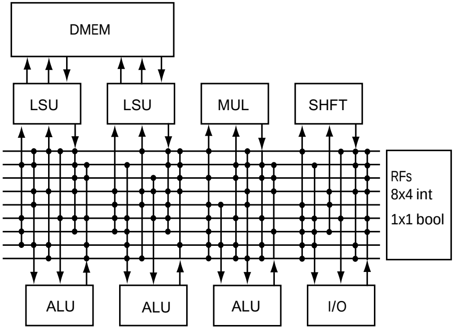
Figure 1: Architecture of the Hybrid DF-SIMD Machine

(Kung 1982; Sinha et al. 2002)

Transport-triggered architecture



g. 1. Principal block diagram of TTA. FU: Function unit. RF: Register
e. LSU: Load-store unit. Dots represent socket connections.



(Janssen 2001; Heikkinen et al. 2002)

Other possibilities...

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

References 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.

Buck, Ian et al. (Aug. 2004). "Brook for GPUs: stream computing on graphics hardware". In: *ACM Transactions on Graphics* 23.3, pp. 777–786. ISSN: 1557-7368. DOI: 10.1145/1015706.1015800.

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.

References II

- 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.
- 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.
- Heikkinen, J. et al. (2002). "Design of transport triggered architecture processor for discrete cosine transform". In: *15th Annual IEEE International ASIC/SOC Conference*. ASIC-02. IEEE, pp. 87–91. DOI: 10.1109/asic.2002.1158036. URL: http://openasip.org/move/doc/ASICSOC02.pdf.
- Janssen, Johan (2001). "Compiler strategies for transport triggered architectures". PhD thesis. URL: http://ce-publications.et.tudelft.nl/publications/1171_compiler_strategies_for_transport_triggered_architectures.pdf.
- Khan, M. A. et al. (2012). *FPGA-accelerated molecular dynamics*. URL: http://www.bu.edu/caadlab/Khan13.pdf.
- Kung (Jan. 1982). "Why systolic architectures?" In: 15.1, pp. 37–46. DOI: 10.1109/mc.1982.1653825.

References III

Lattice Semiconductor (2017). *iCE40^{TM} 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 andinfinite machines*. Prentice-Hall.

Minsky, Marvin Lee and Seymour Papert (1969). *Perceptrons: an introduction to computational geometry*. MIT Press.

Nickolls, John et al. (Mar. 2008). "Scalable Parallel Programming with CUDA: Is CUDA the parallel programming model that application developers have been waiting for?" In: *Queue* 6.2, pp. 40–53. ISSN: 1542-7749. DOI: 10.1145/1365490.1365500.

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.

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

References IV

Sinha, A. et al. (2002). “A reconfigurable data-flow architecture for a class of image processing applications”. In: *ICCSC’02. 1st IEEE International Conference on Circuits and Systems for Communications. Proceedings (IEEE Cat. No.02EX605)*. ICCSC-02. St. Petersburg State Polytech. Univ, pp. 460–463. DOI: 10.1109/occsc.2002.1029140.

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.