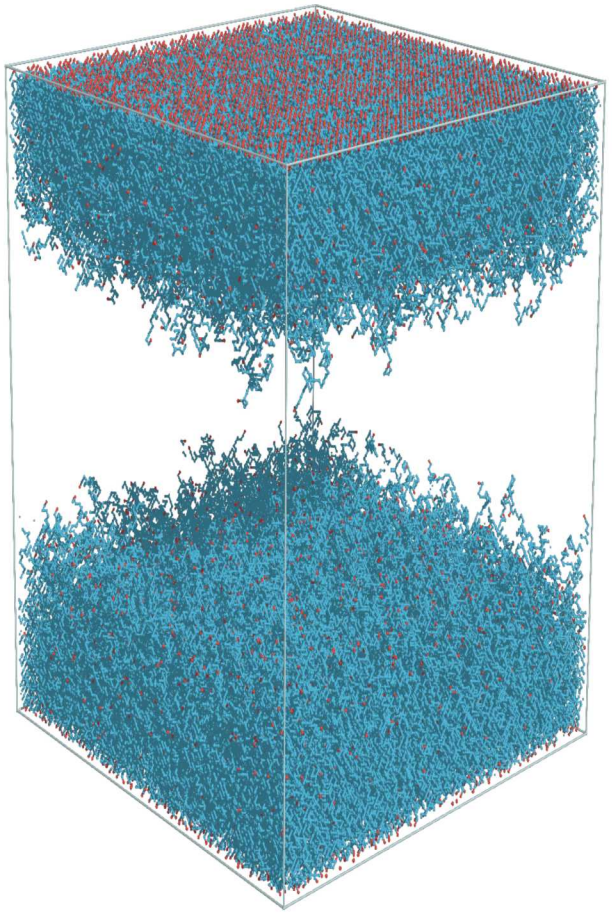
Build the largest FPGA array in the world...

How hard can it be?

by Kamil Rudnicki, PhD

FPGA Conference - Munich, 5 July 2022

POLL



ARUZ

Who am I?

15 years of FPGA experience:

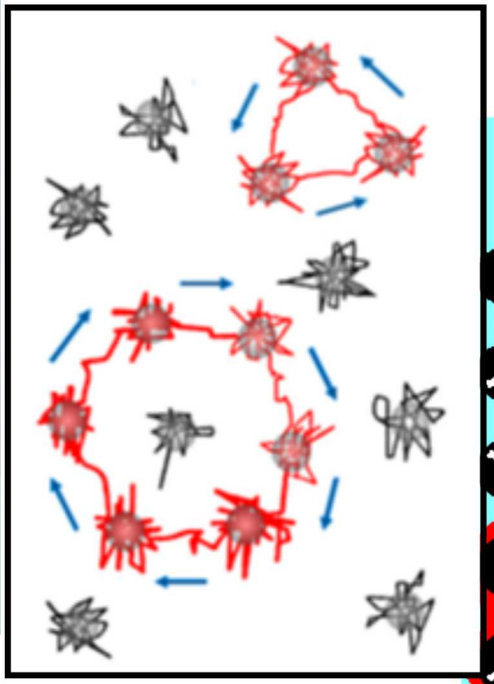
commercial (Ericsson, GE, Brightelligence, Arrow)

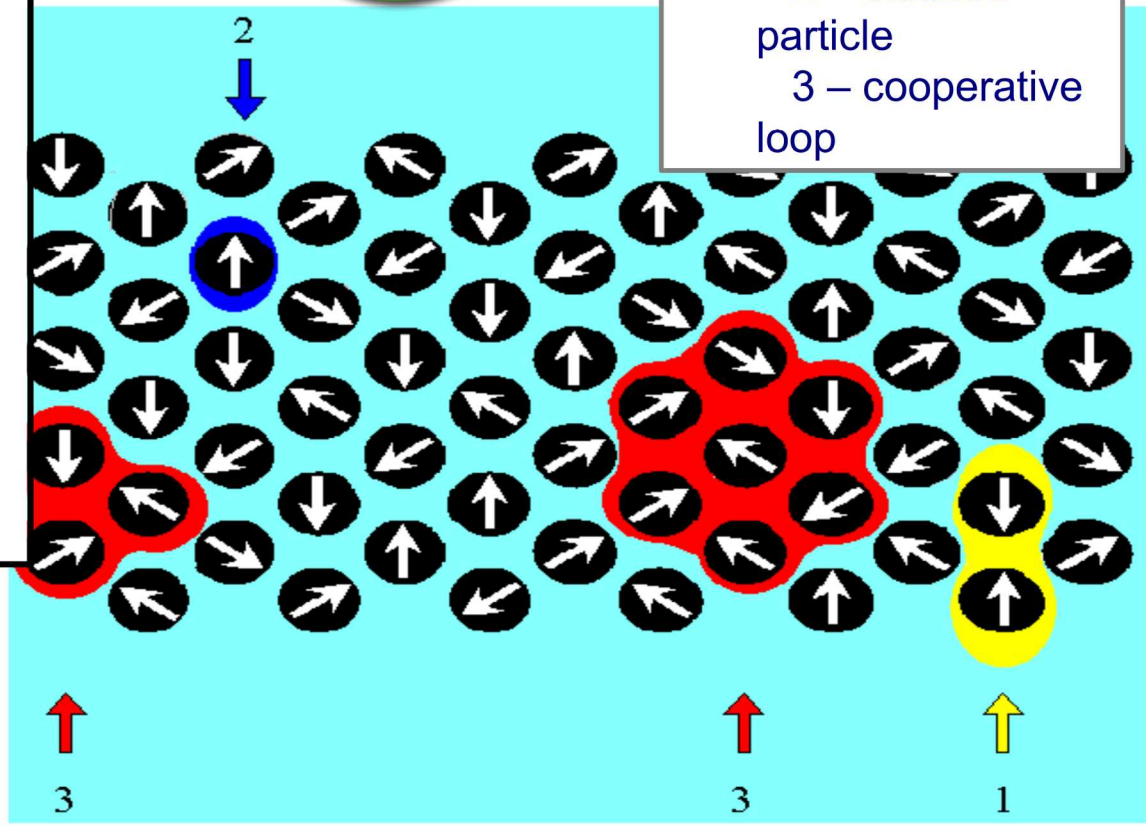
scientific

- ✔ Lodz University of Technology, Poland
- ✔ Gdansk University of Technology, Poland
- ✔ Universitat Politècnica de Catalunya, Spain
- ✔ University of Glasgow, Scotland
- ✔ Military University of Land Forces, Poland

How it all started?

Prof. T. Pakuła, LUT





Legend:

1 – collision
2 – blocked particle
3 – cooperative loop

For simulation of complex molecular systems, focused on extraction of dynamic and static properties of simulated materials.

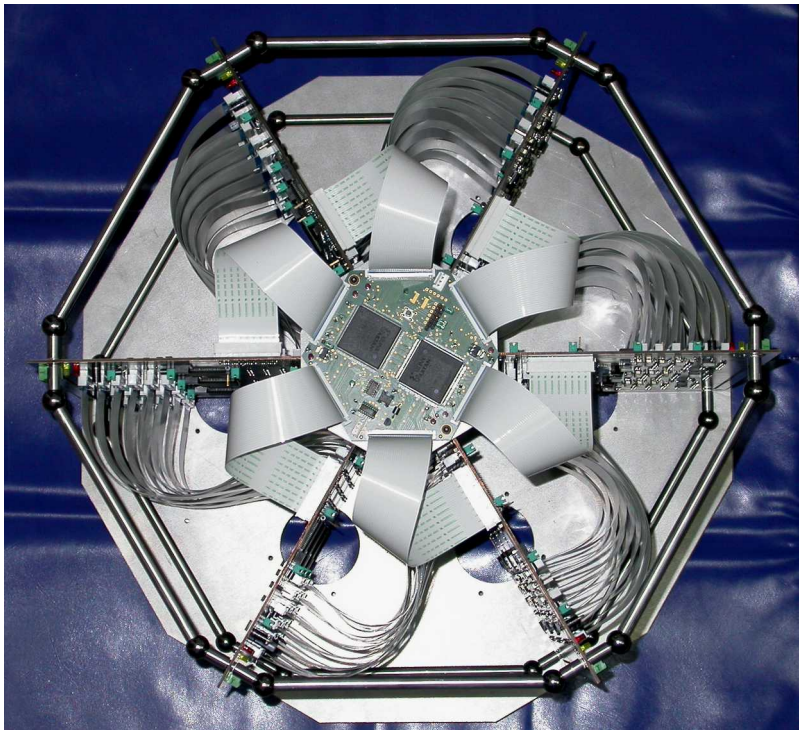
For something to work first time,
you need 10 years of experience.

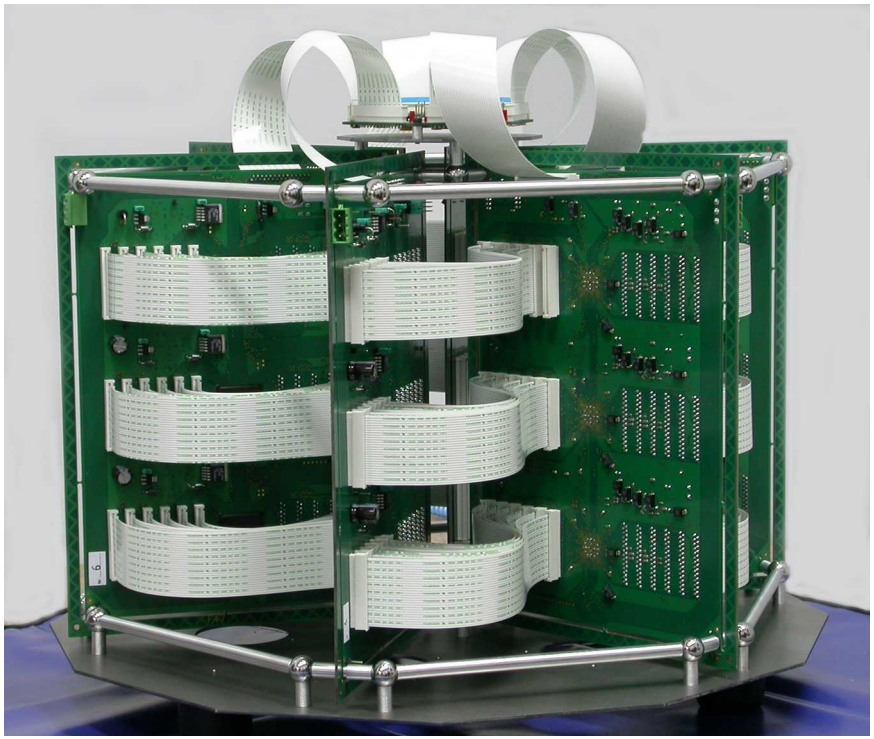
First prototype

2006 – 2008: Grant of Polish Ministry of Science and Higher Education:

„Implementation of Dynamic Lattice Liquid Algorithm by Means of Dedicated Microprogrammable Computational Cell”, 299 750 PLN (65k EUR)

Result: (**μDLL**): 6x6x6 = **216 nodes** (18 FPGAs)

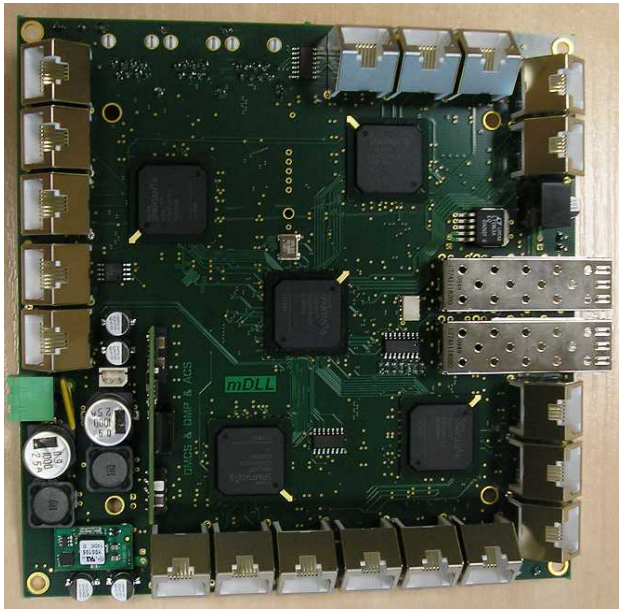


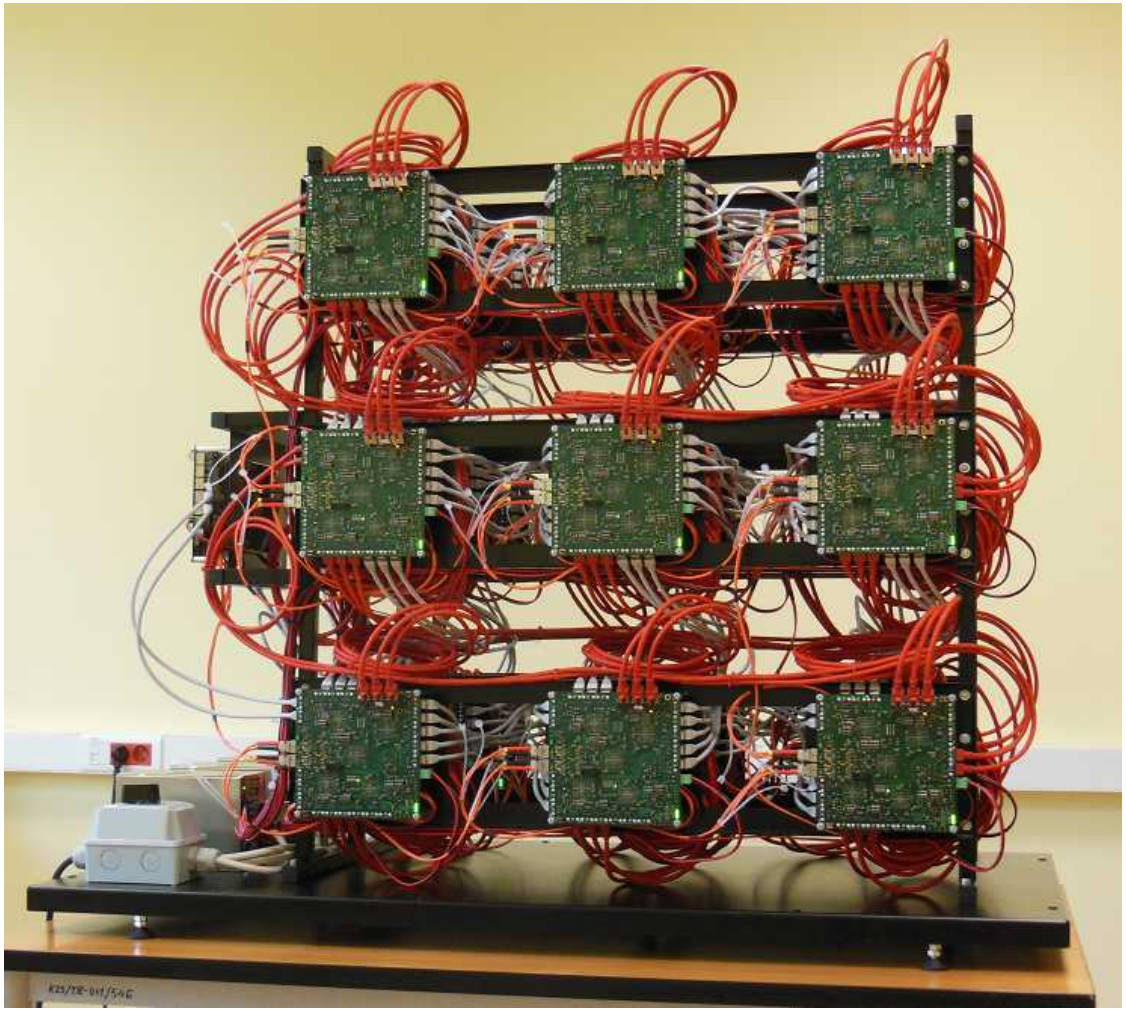


Second prototype (scalable)

2009 – 2012: Grant of Polish Ministry of Science and Higher Education:
„**Module of Dedicated Computational Cluster for Simulations Based on Dynamic Lattice Liquid Algorithm**”,
497 900 PLN (105k EUR)

Result (**mDLL**):
28 modules, 5 FPGAs in each module
(4 for simulations, 1 for controlling),
~64 nodes (in each FPGA).
In total: **~1 728 nodes** (140 FPGAs)





BioNano Park in Lodz, Poland

Challenges

We want to be able to simulate proteins, so we need at least 1 million molecules.

OK.

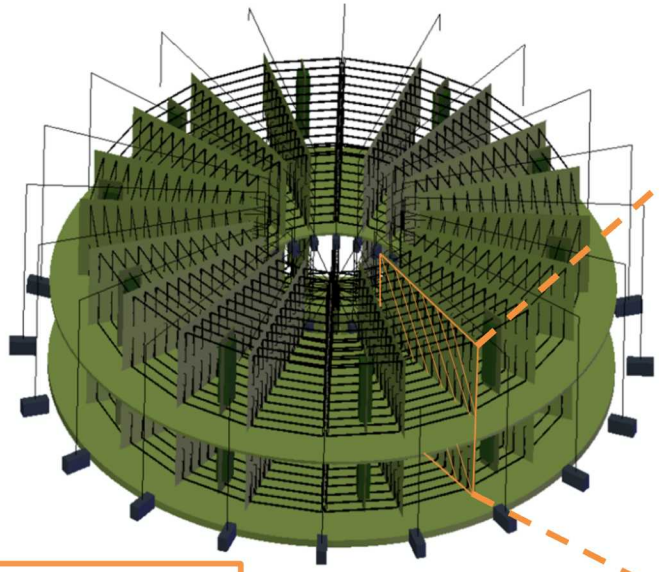
We need to simulate it reasonably quickly.

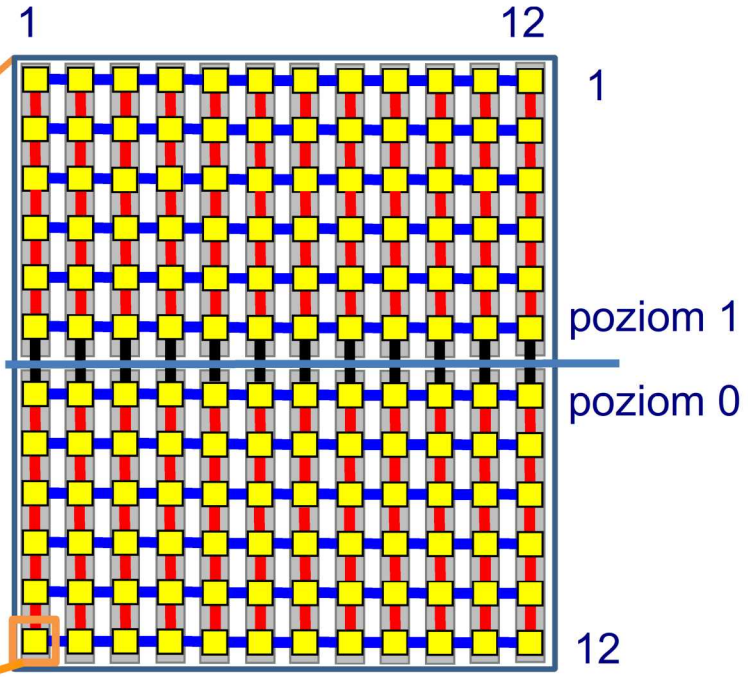
OK.

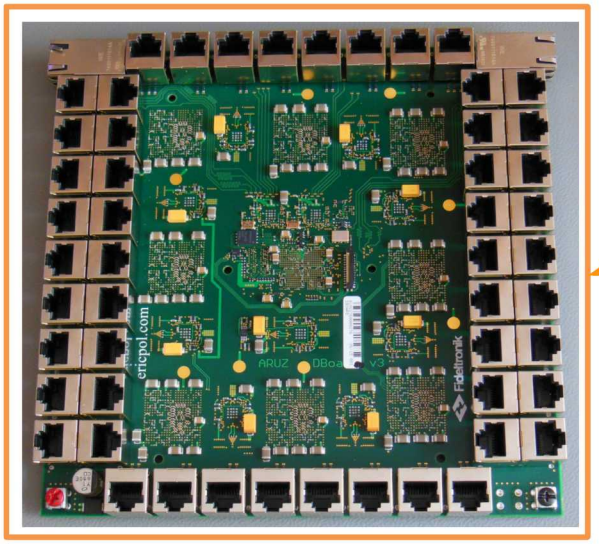
We have only 13 months to build it.

OK.

ARUZ







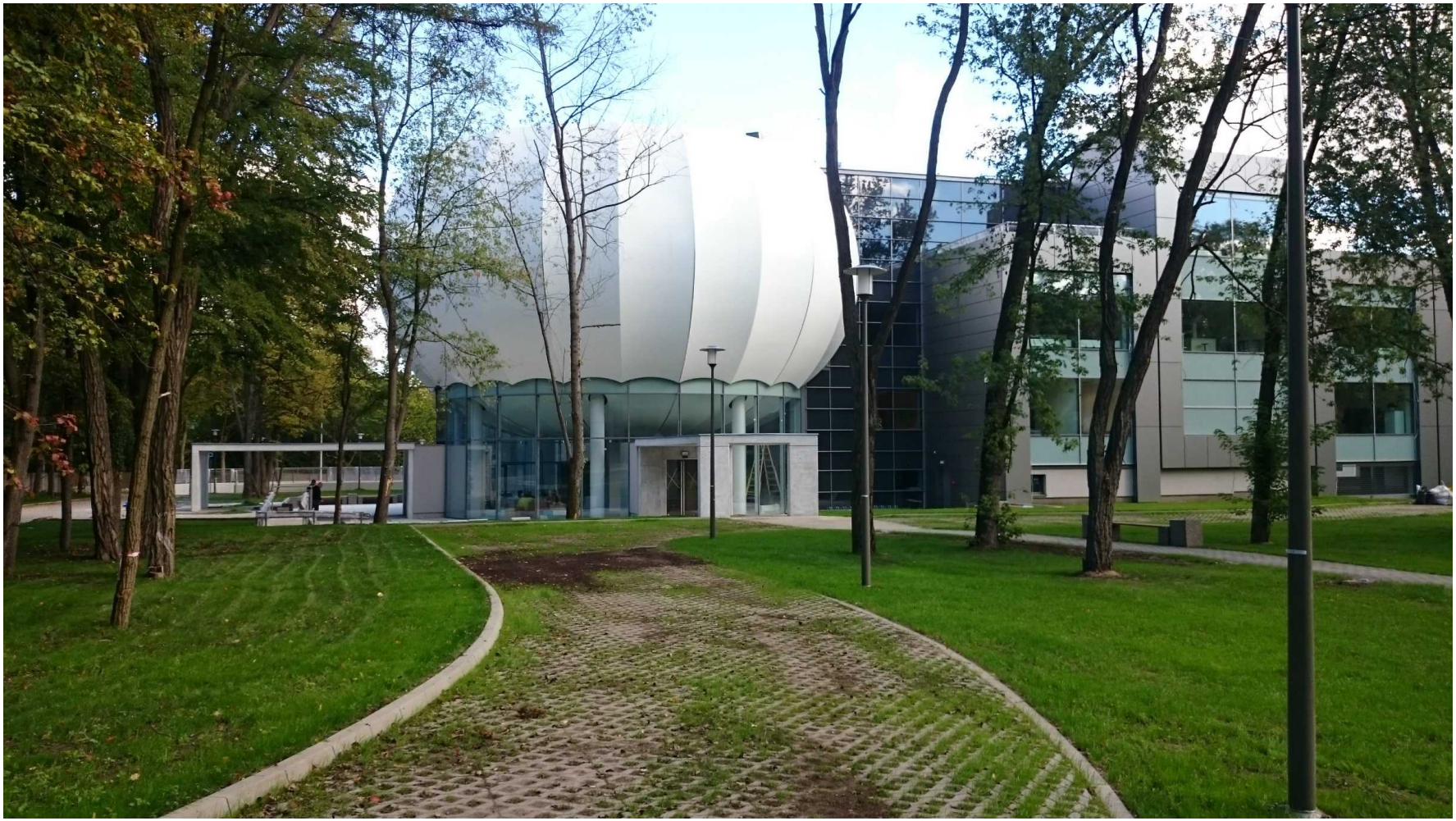
DBoard = Daughter Board

Panel = 144 x DBoard

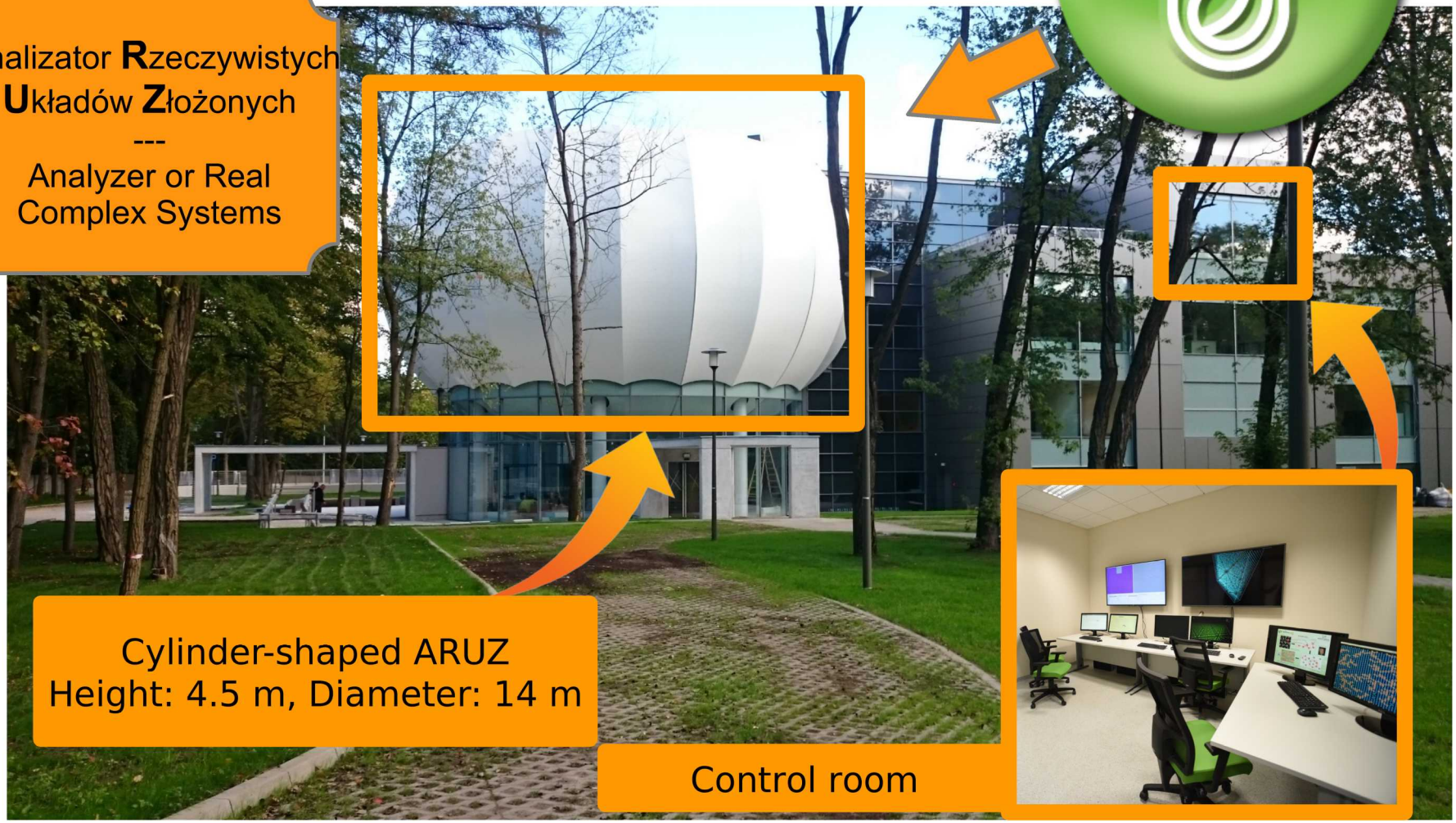
In total:

- 2 880 PCB boards
- 25 920 FPGAs
- **~1 500 000 nodes**

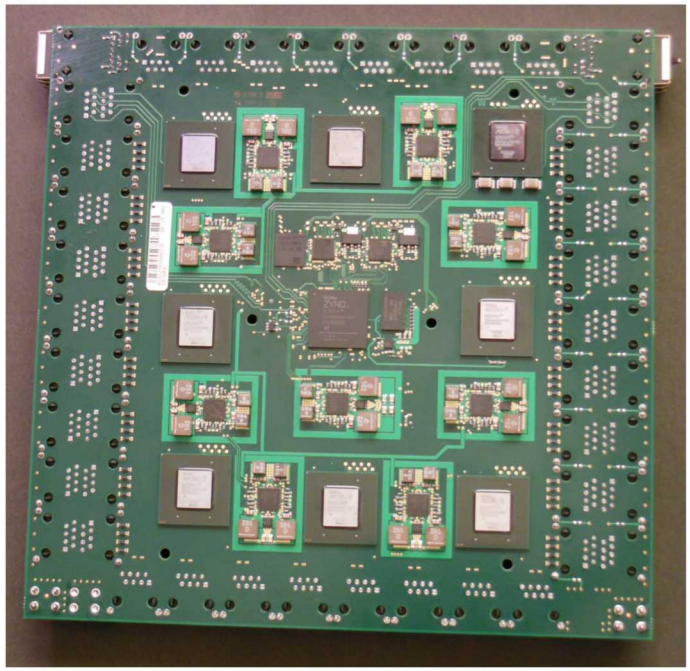
BioNano Park in Lodz, Poland

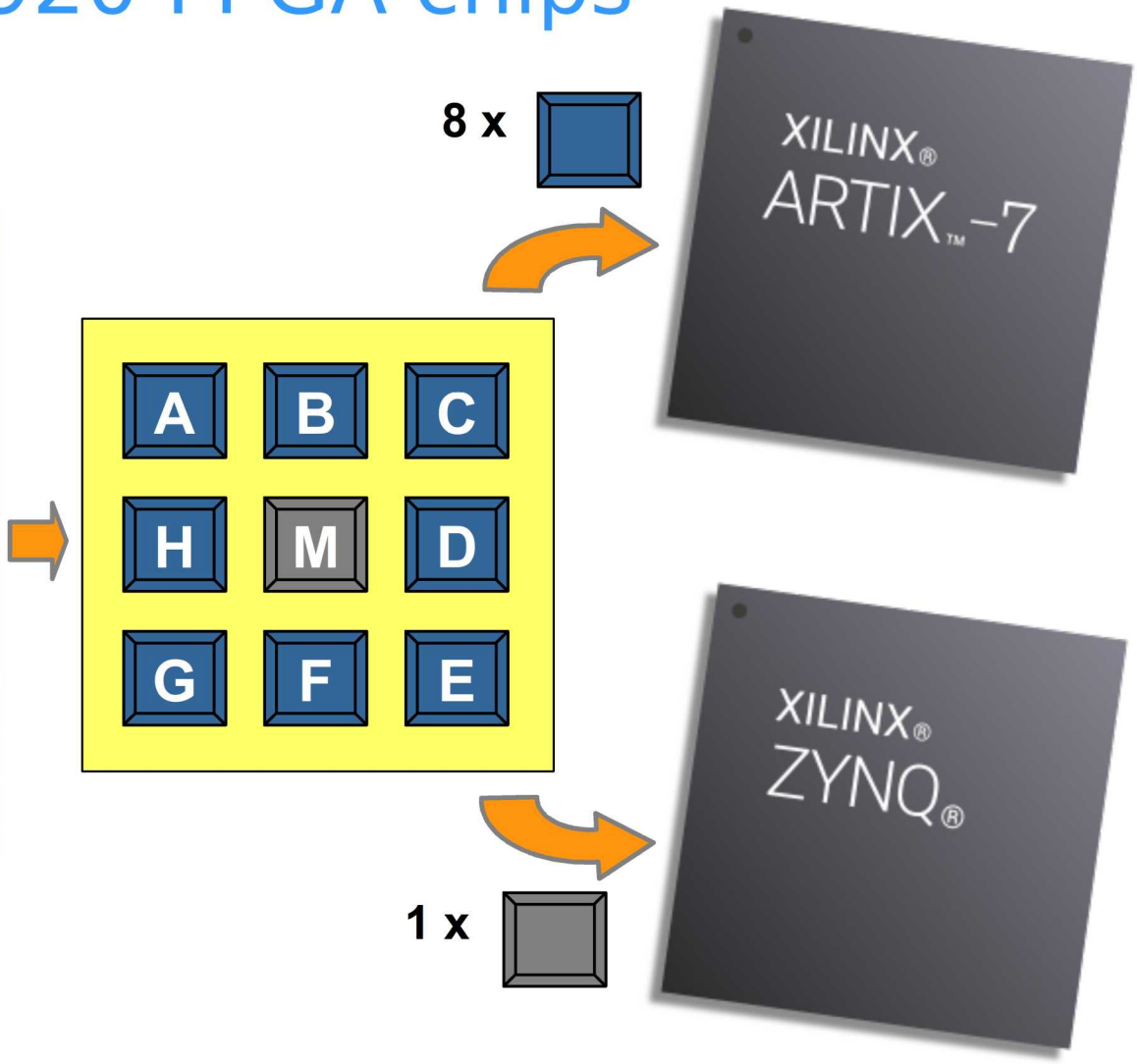


BioNano Park in Lodz, Poland

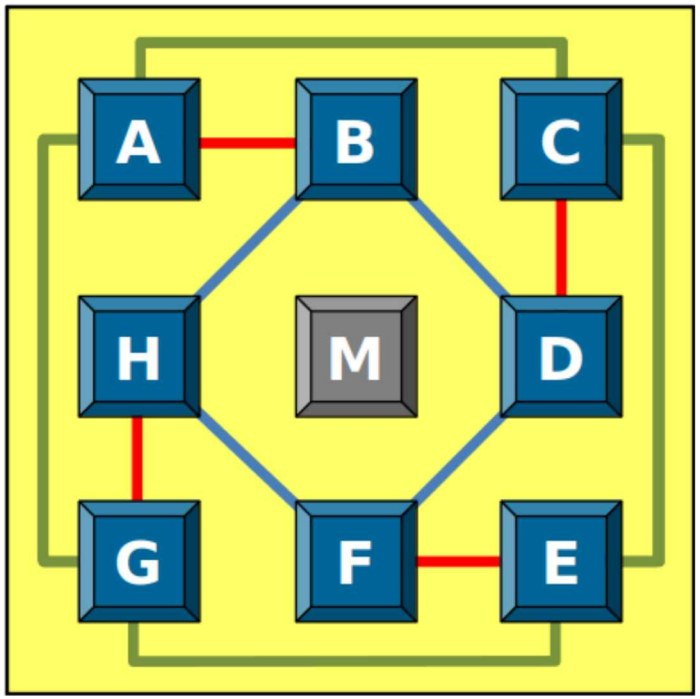


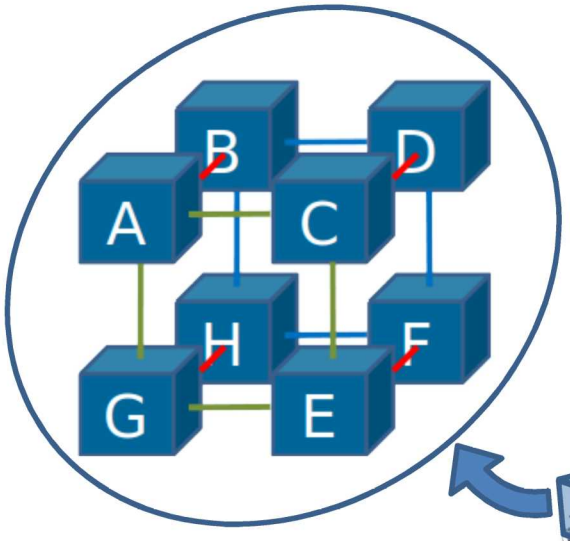
2 880 x 9 = 25 920 FPGA chips

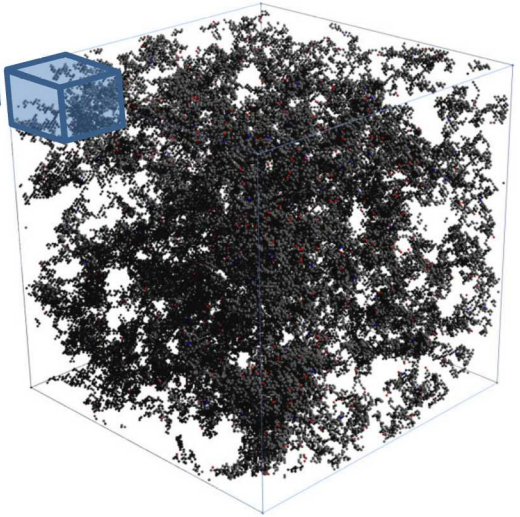


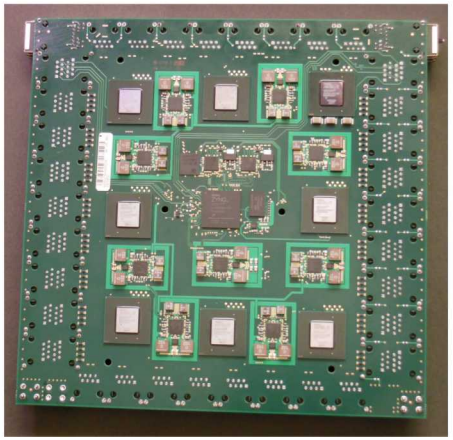


DBoard – Segment of 3D Space

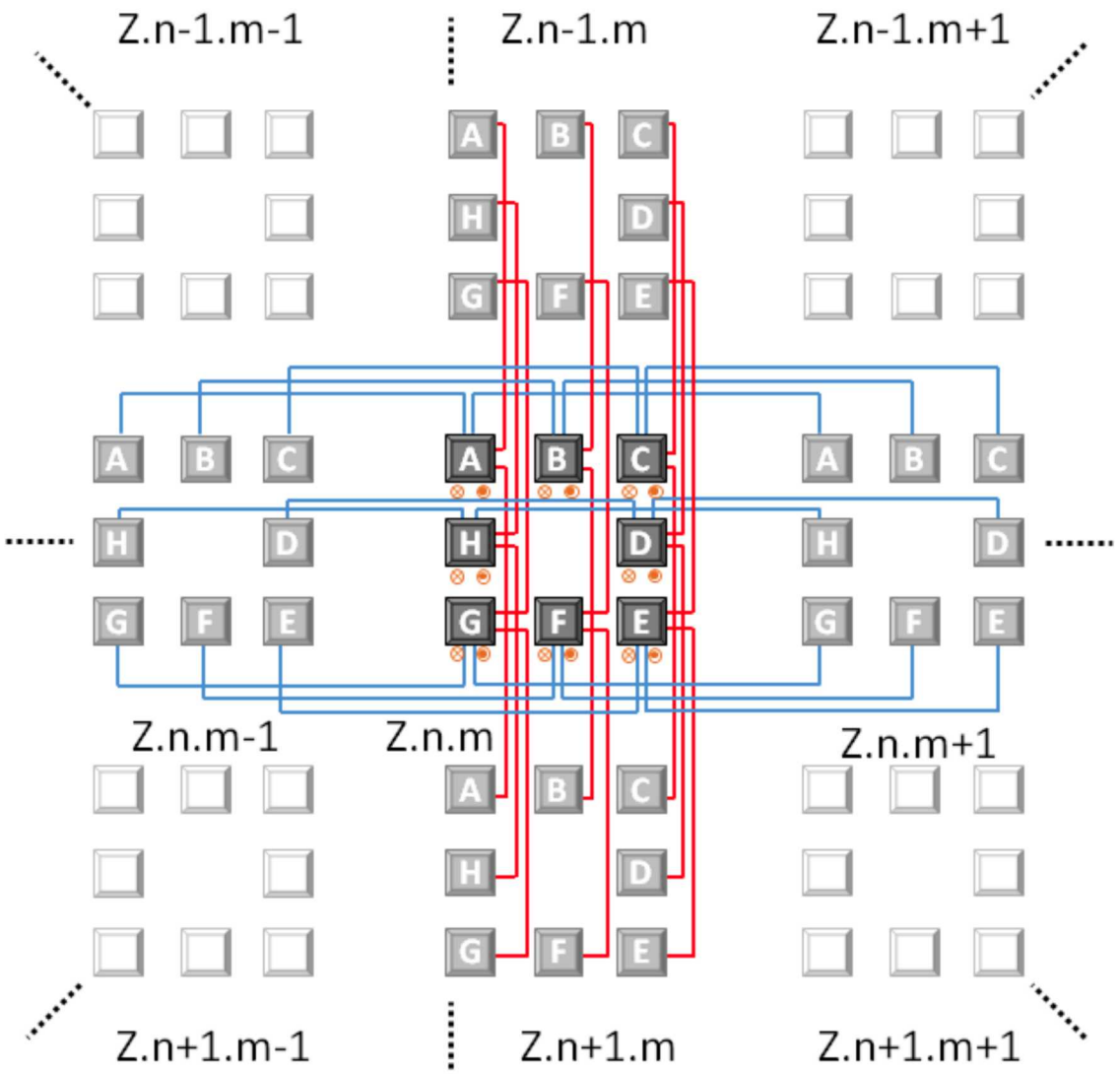


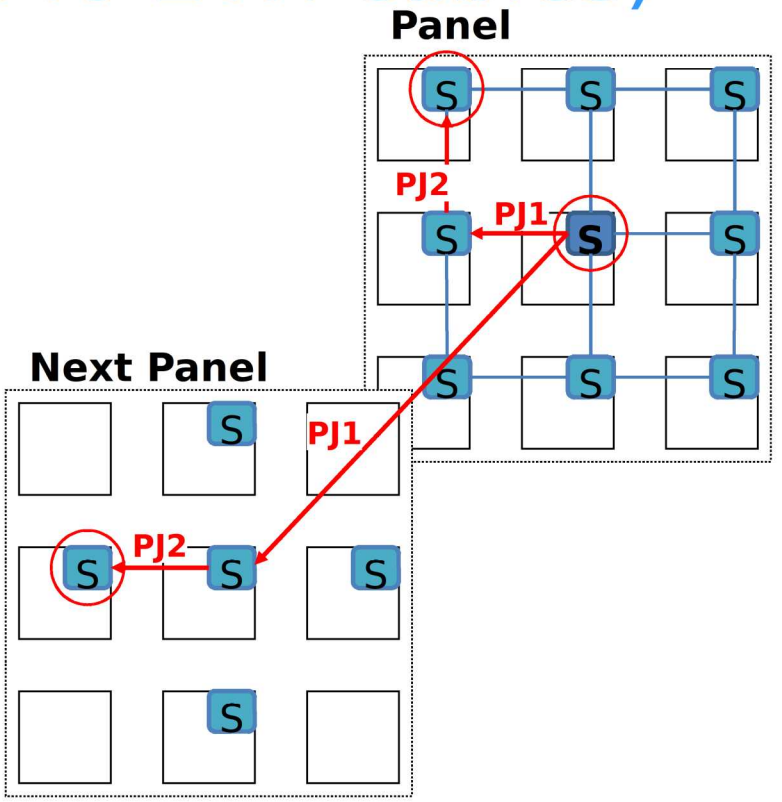






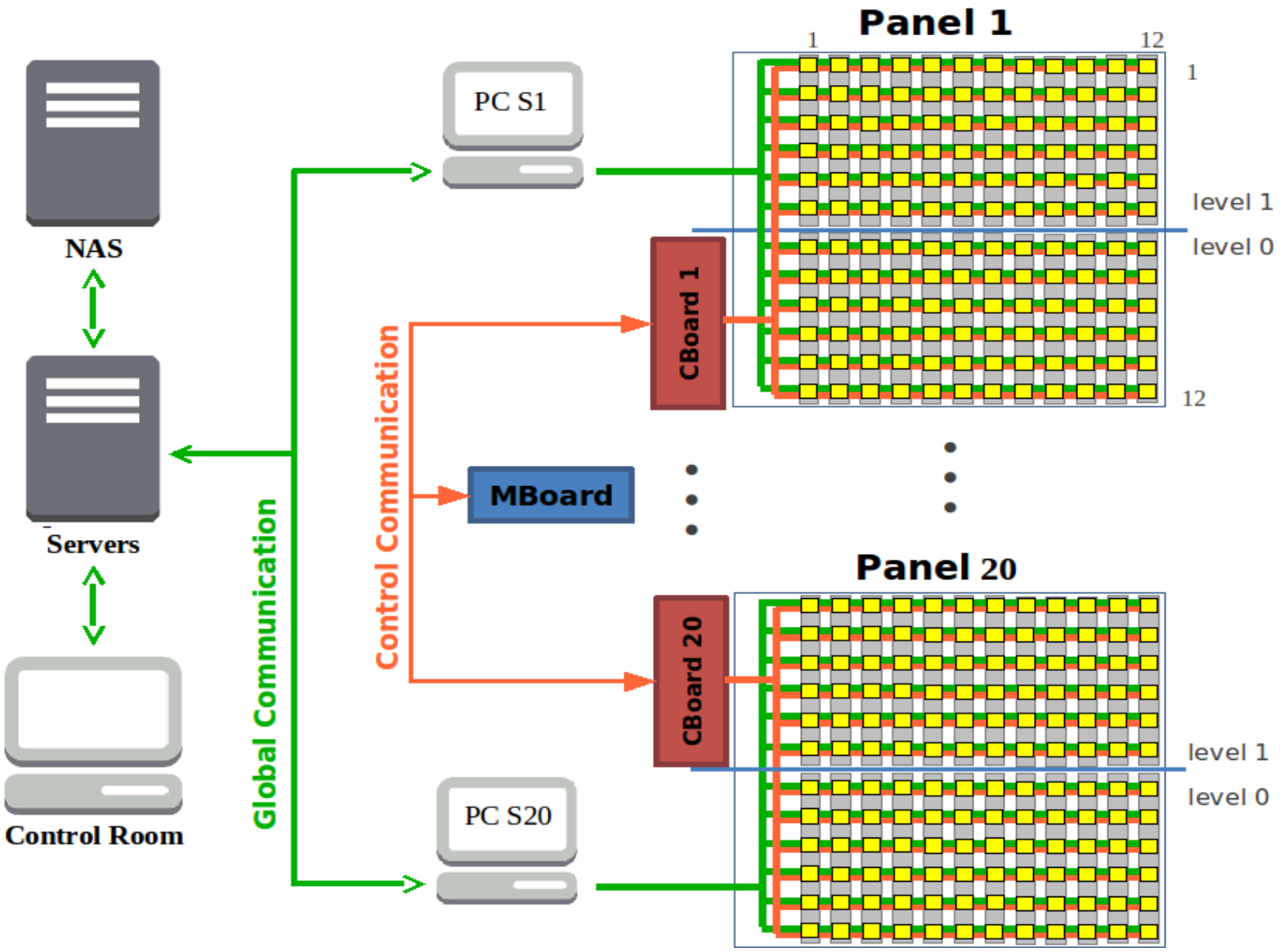
Local Communication (76 840 ETH cables)





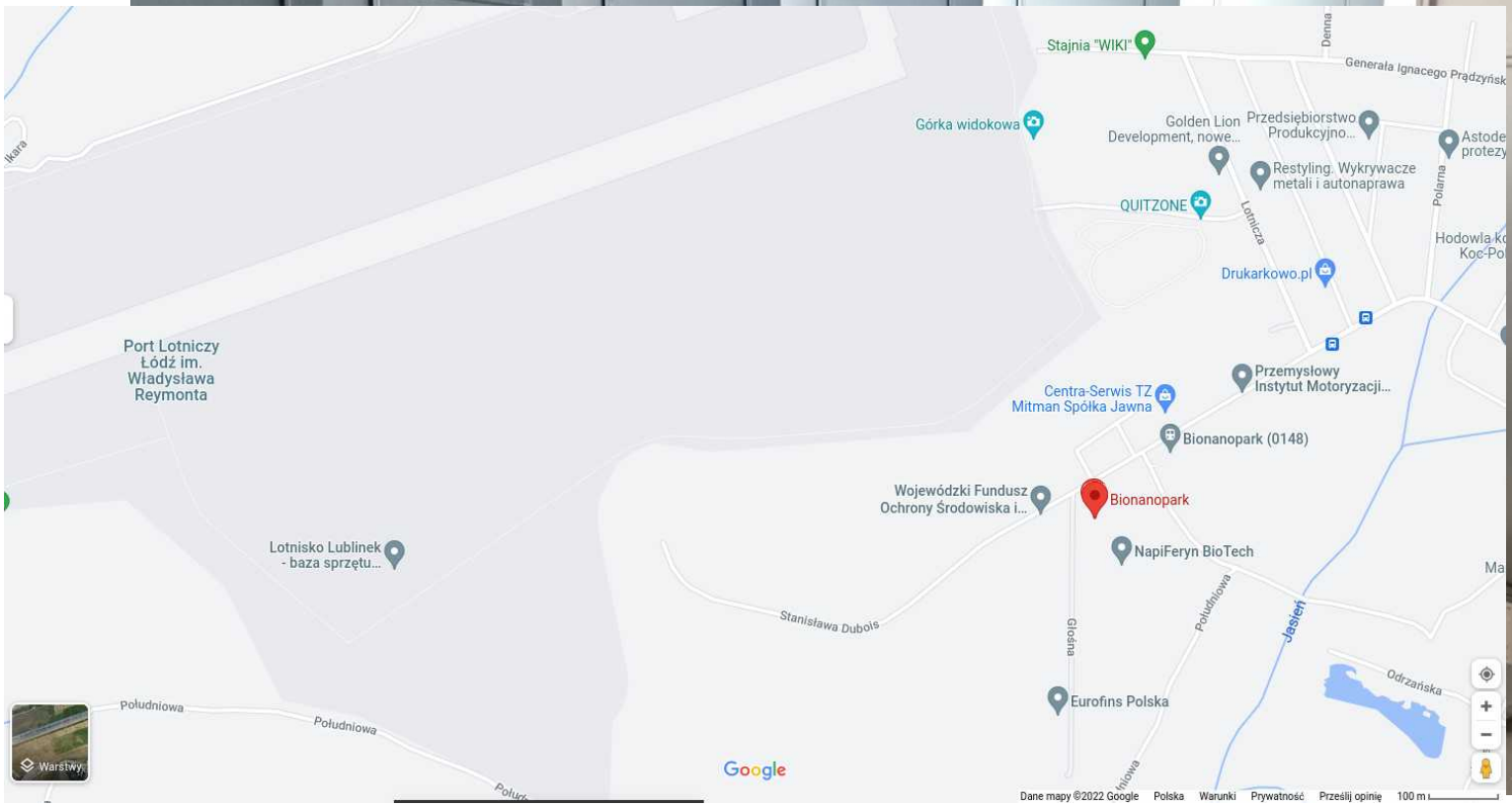
1 Gb links (full duplex)

Global & Control Communication



ARUZ

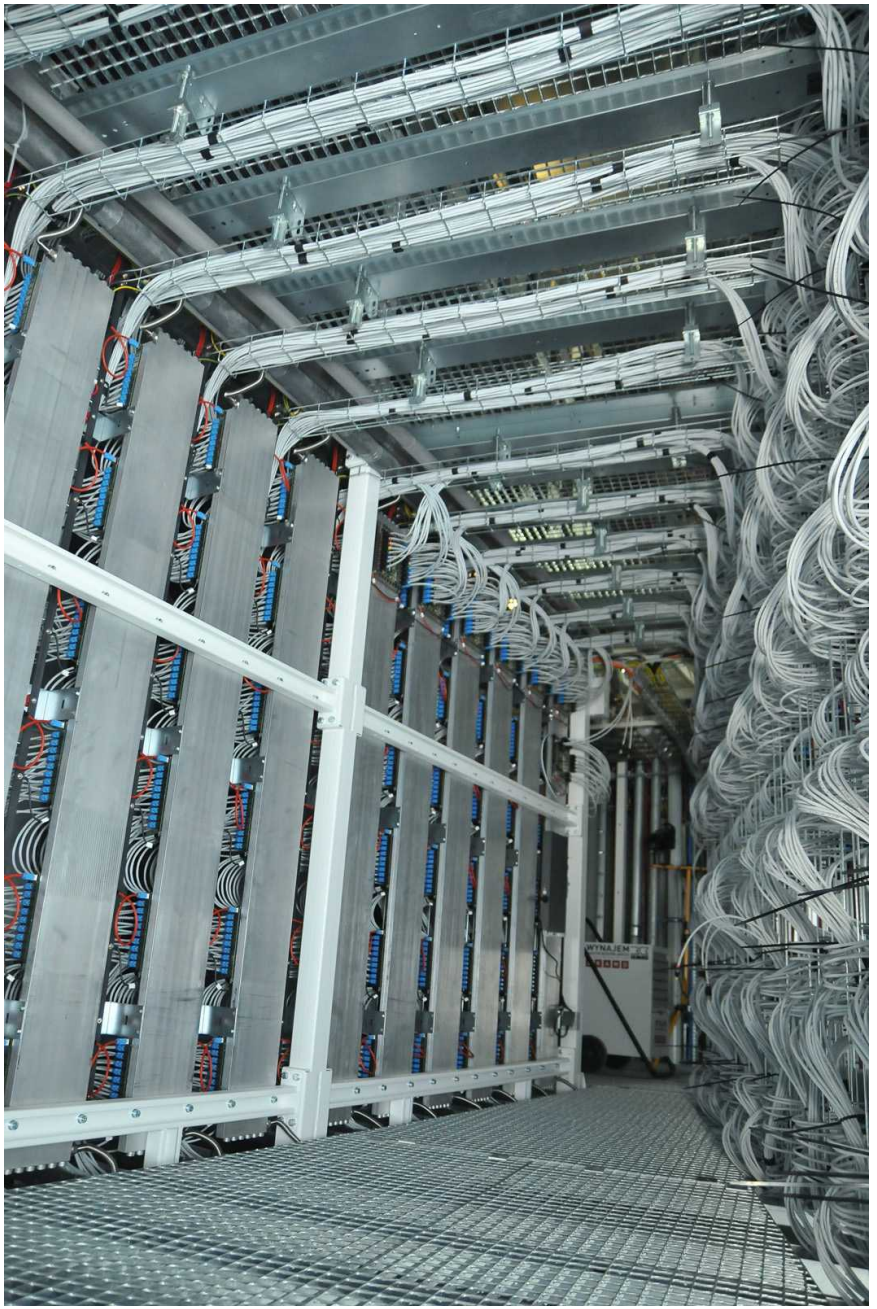
ARUZ


Stajnia "WIKI"
Denna
Generała Ignacego Prądzyńsk
Górka widokowa
Golden Lion Development, nowe...
Przedsiębiorstwo Produkcyjno...
Astode protezy
Restyling. Wykrywacze metali i autonaprawa
Polarna
QUITZONE
Lotnicza
Hodowla ko Koc-Po
Drukarkowo.pl
Port Lotniczy Łódź im. Władysława Reymonta
Przemysłowy Instytut Motoryzacji...
Centra-Serwis TZ Mitman Spółka Jawna
Bionanopark (0148)
Wojewódzki Fundusz Ochrony Środowiska i...
Bionanopark
Lotnisko Lublinek - baza sprzętu...
NapiFeryn BioTech
Stanisława Dubois
Głośna
Południowa
Jesienn
Odrzańska
Eurofins Polska
Południowa
Południowa
Warstwy
Google
Dane mapy ©2022 Google
Polska
Warunki
Prywatność
Prześlij opinię
100 m

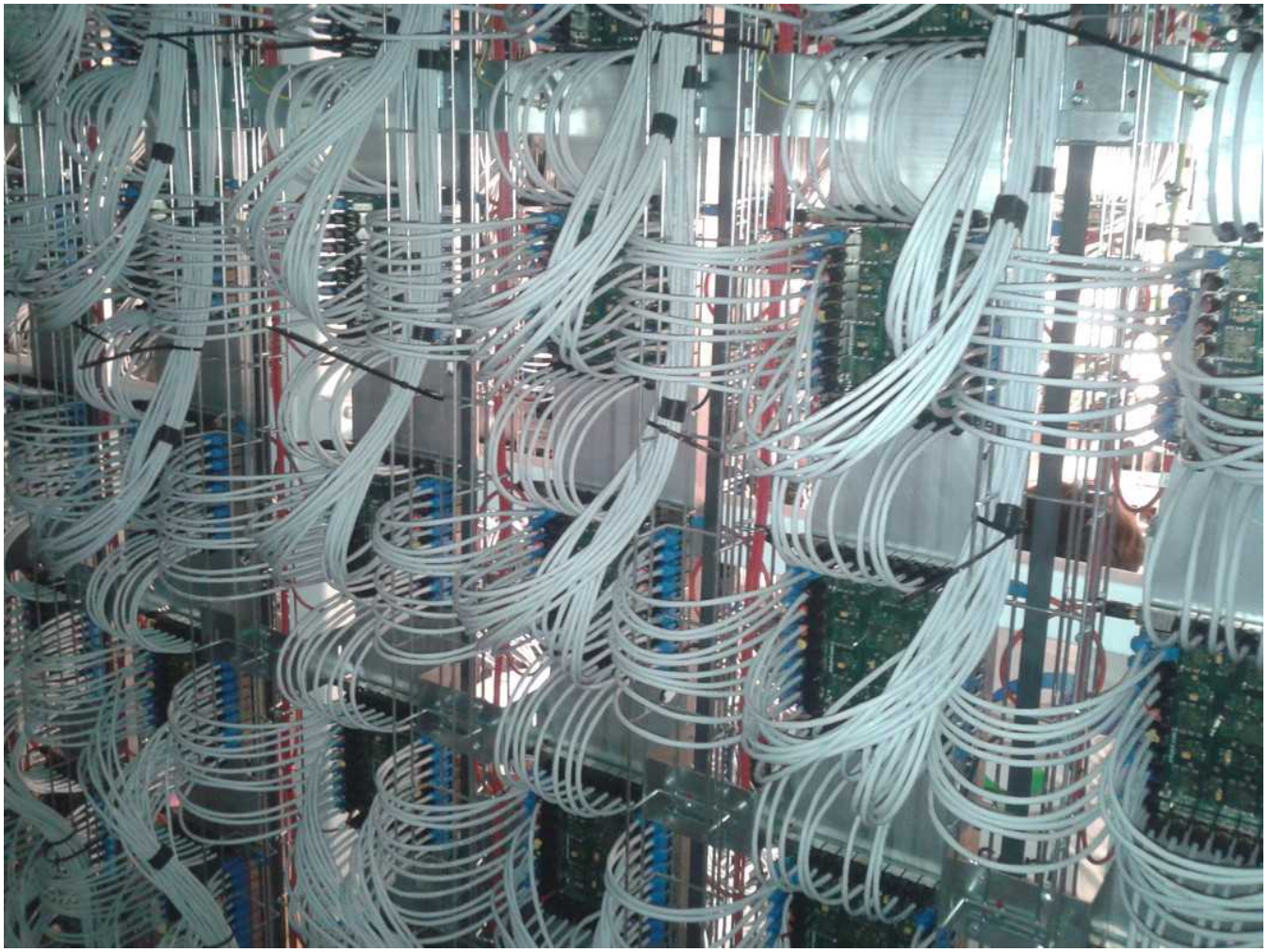
ARUZ

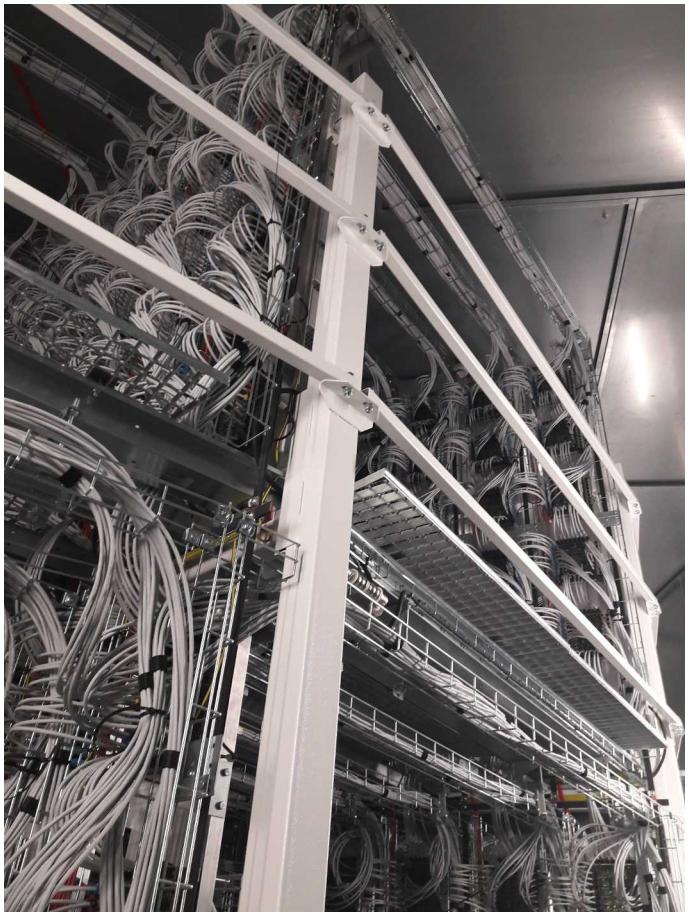
ARUZ

Kamil Rudnicki



ARUZ





ARUZ PSUs

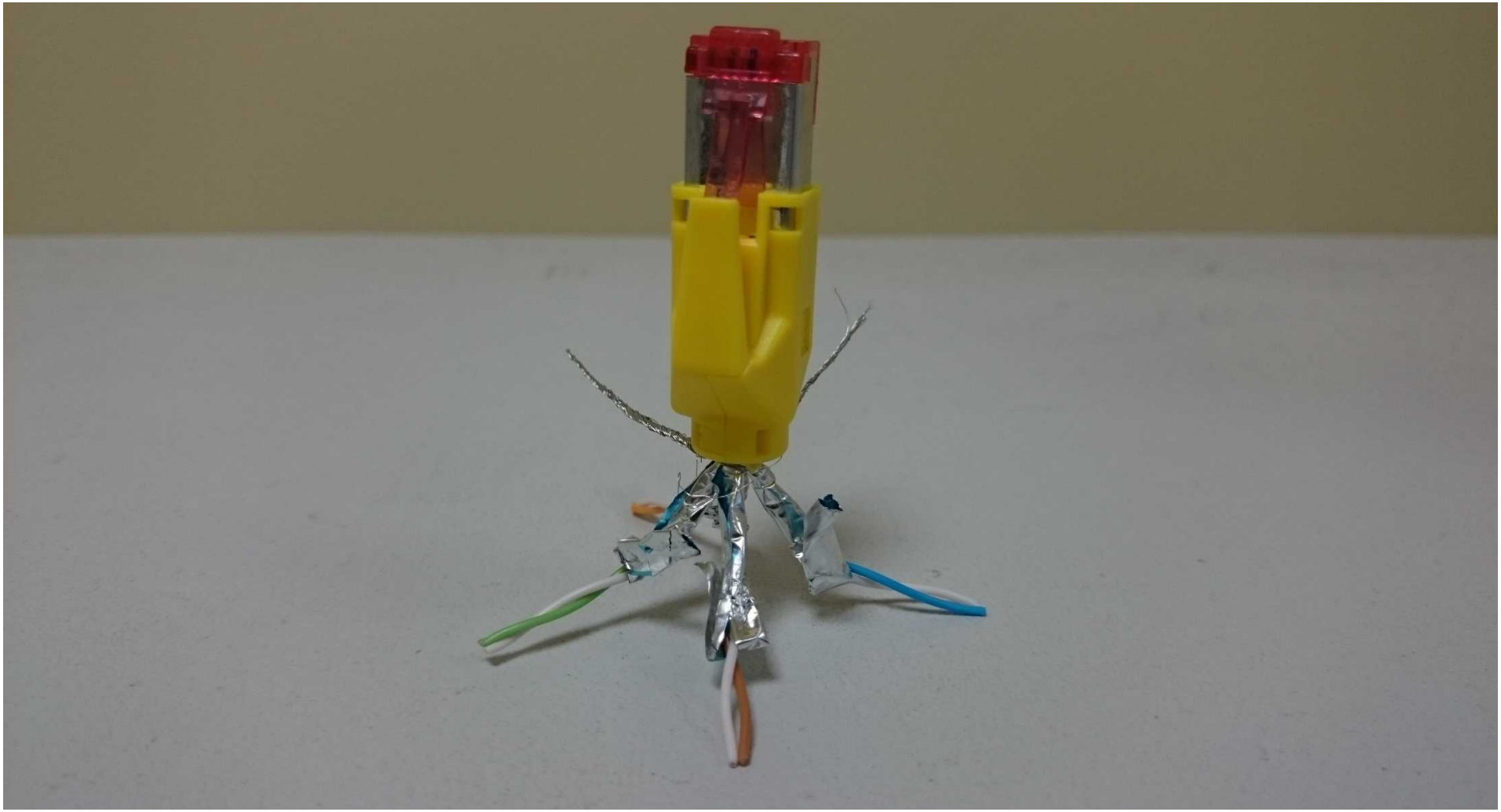
ARUZ UPS

ARUZ development

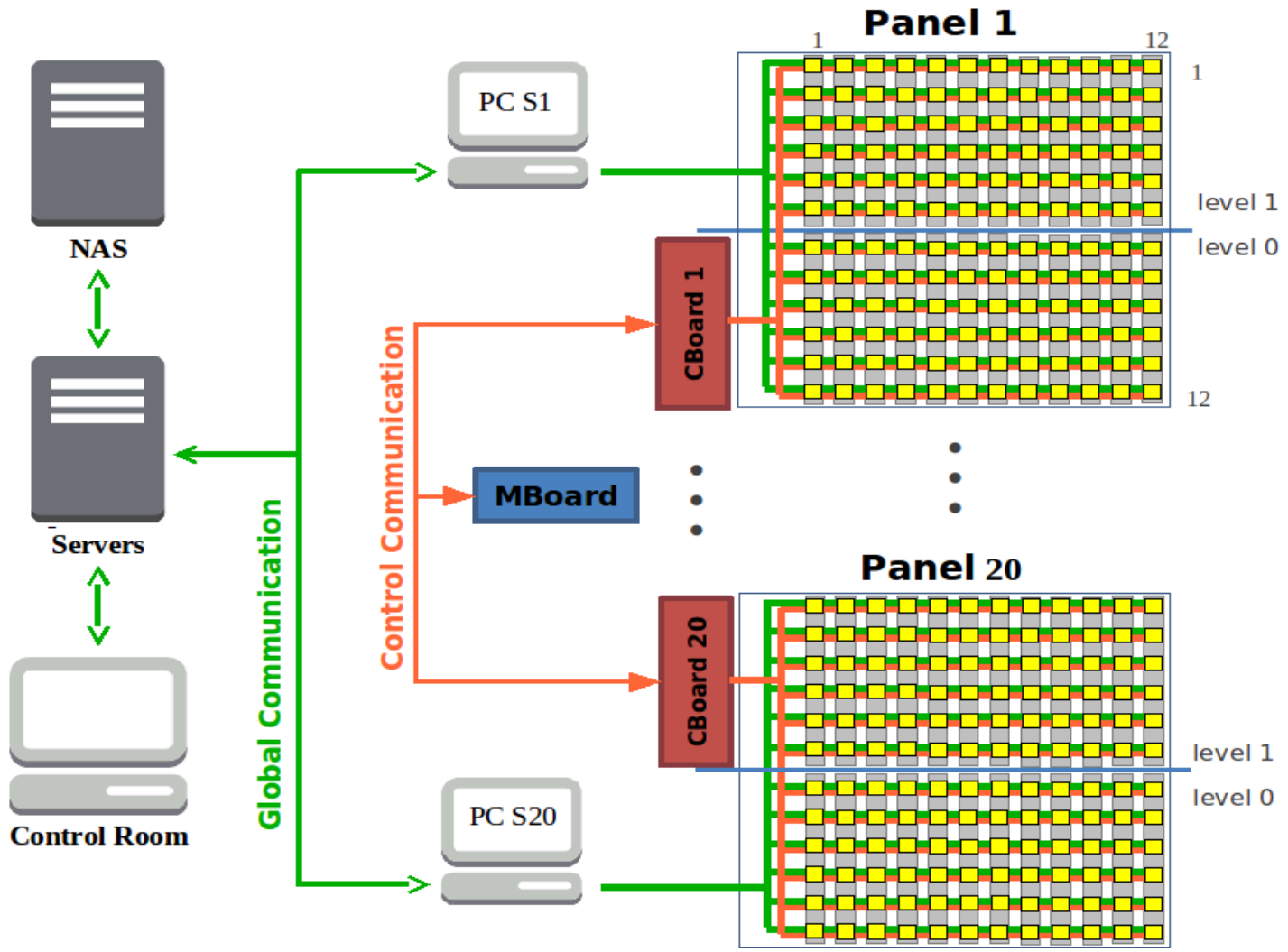
Test frame

picoARUZ and nanoARUZ

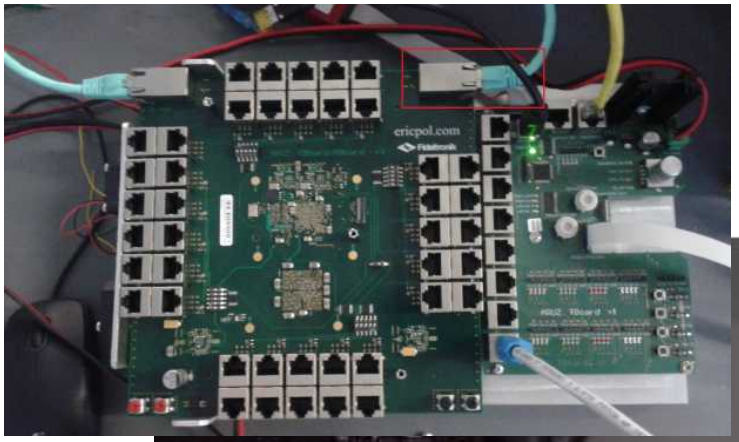
So predictable, yet so unpredictable

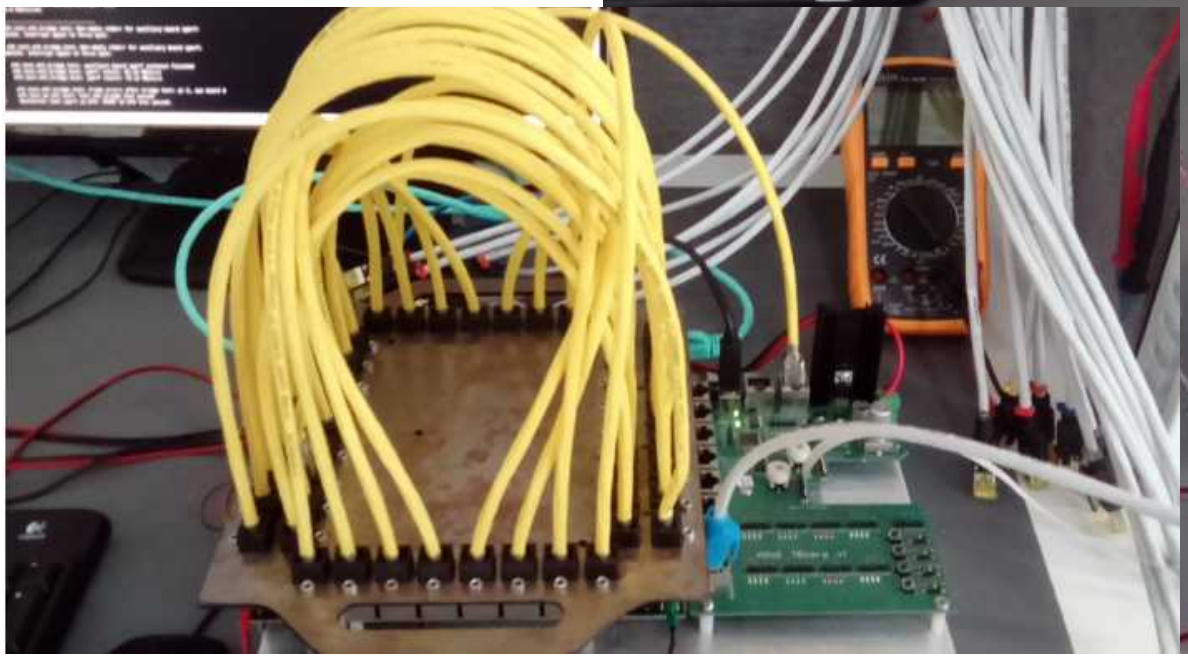


cat /dev/null


Panel 1
1
12
1
level 1
level 0
12
PC S1
CBoard 1
NAS
Servers
Control Communication
MBoard
Global Communication
Panel 20
CBoard 20
level 1
level 0
PC S20
Control Room

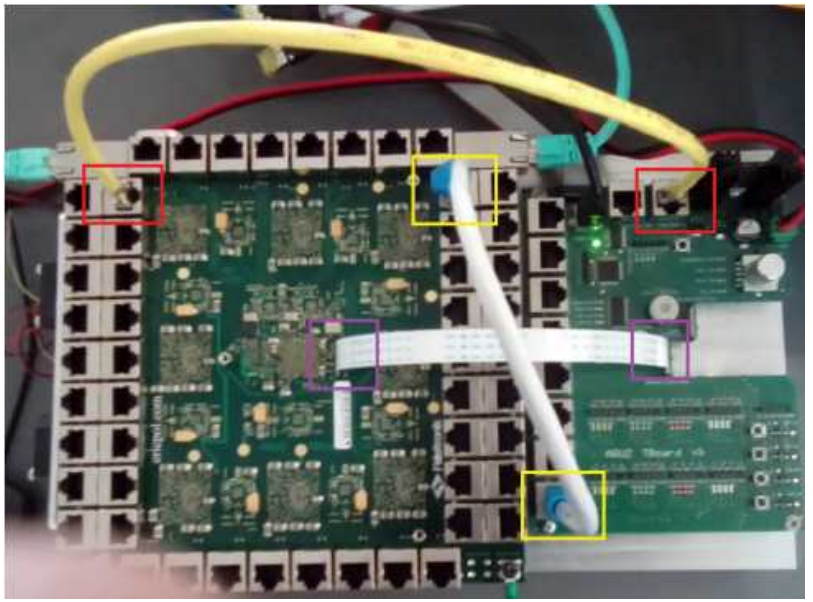
Testbed





Test procedure

1. Entering DBoard number
2. Configuration of DHCP server
   - REPEATED IP ALERT!
3. Configuration of power supply
4. Configuration of DBoard



Test procedure

Tests starts automatically with:

1. QSPI Flash Programming
2. UART Test
3. JTAG Setup
4. eMMC Setup
5. OS Image download
6. DBoard Master (Zynq) bitstream download
7. Reset Status test
8. IPERF test
9. IBERT test

3000 boards → 140 boards
in need of a fix (4.7%)

Firmware

Chemical mechanisms

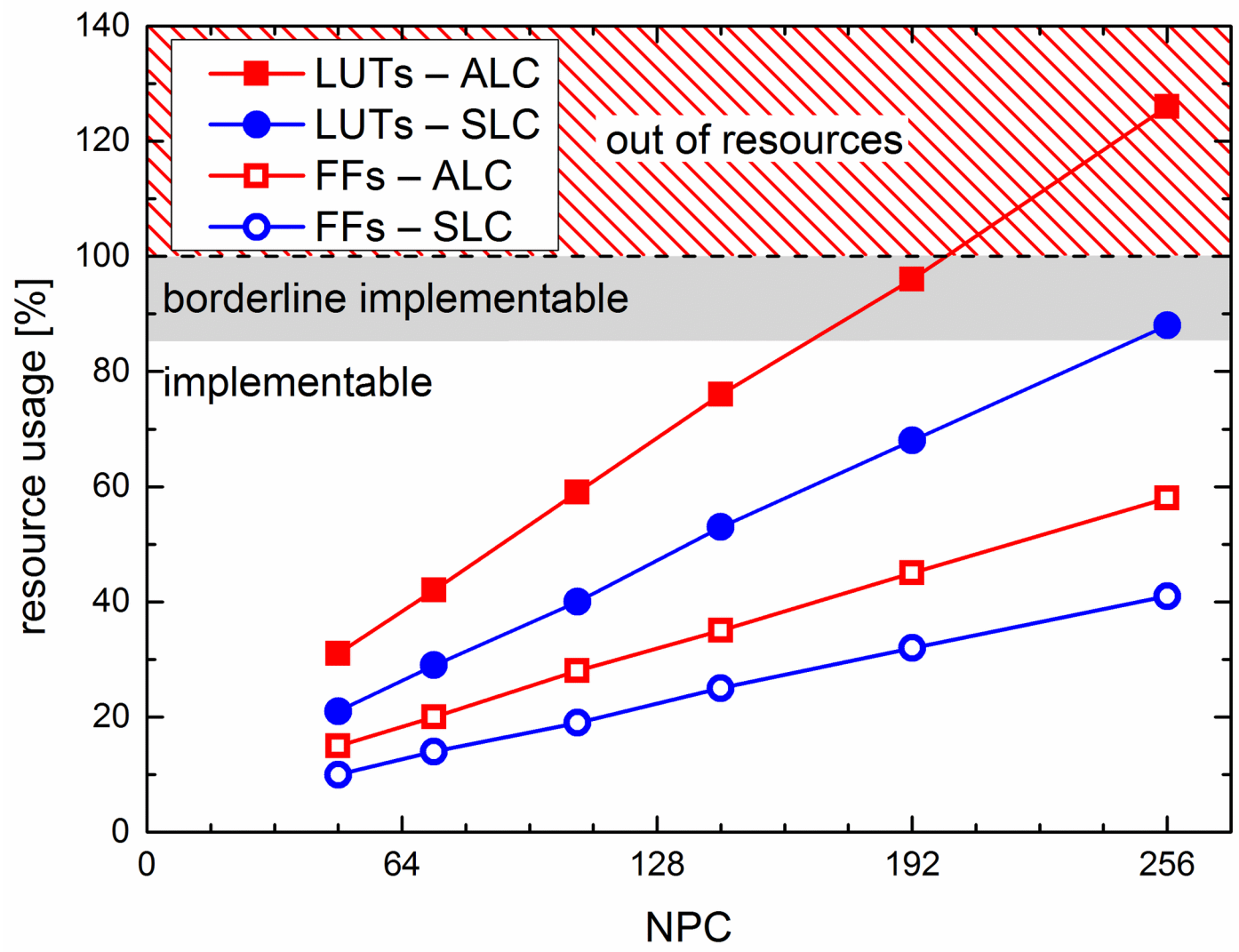
Mechanisms:

- LOOPS
- WAYS
- BONDS
- BOND_BINDS
- BOND_BREAKS
- TERMINATION
- MOBILITY
- ENERGY
- REACTION
- APERIODIC
- VECTOR
- REORIENTATION

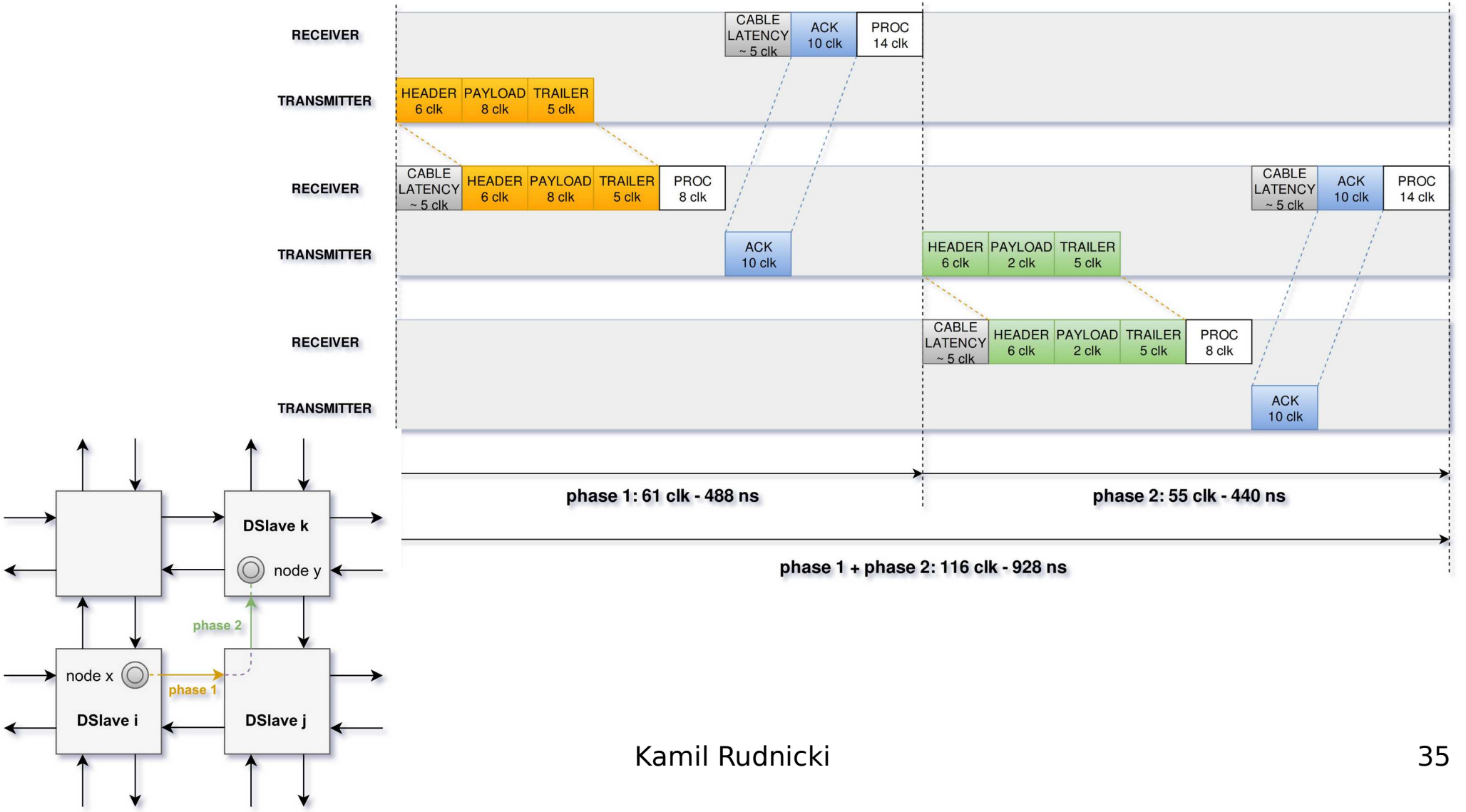
For each required combination of mechanism dedicated FPGA configuration is prepared

Hardware (configuration) adapted to the task (requirements)!

Resources utilization



Dedicated local communication protocol



Verification and emulation

Software emulation

Hardware implementation

Simulation step accurate

DLLDesigner
software
emulator

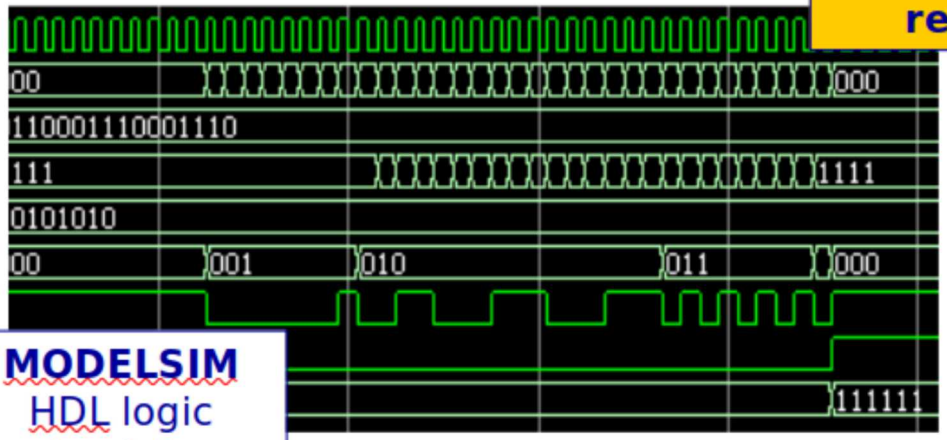
```
0001 1000 1001 1101
1111 0101 1010 1101
0000 1000 0010 0000
1011 1110 1011 0000
1101 1100 1000 0100
0001 1001 1000 1010
1010 1010 1011 1010
0010 0100 1100 1000
1010 0100 0100 1111
0000 1101 0101 0000
1001 0100 1001 0010
```

Binary results

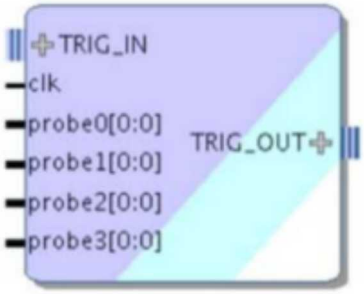
nanoARUZ
developer
prototype

Clock cycle accurate

Waveform results



MODELSIM
HDL logic
simulator



CHIPSCOPE
Integrated
Logic Analyzer

DLLDesigner grid generation

$$\begin{bmatrix} \lambda^s_{11} & \lambda^s_{12} \\ \lambda^s_{21} & \lambda^s_{22} \\ \lambda^s_{31} & \lambda^s_{32} \end{bmatrix} = \begin{bmatrix} (s-2)\cdot\left(\overline{\overline{s}}-\overline{s}\right)-\overline{\overline{s}} & \frac{1}{2}\left(\overline{\overline{4s}}+2\overline{s}\overline{s}-s-\overline{s}\right) \\ 2\overline{s}-s+3\overline{\overline{s}} & \frac{1}{2}\left(s-\overline{s}-2\overline{\overline{s}}\right) \\ \overline{s}-\overline{\overline{s}} & \overline{\overline{s}} \end{bmatrix} \quad gdzie \quad \begin{matrix} \overline{\overline{s}} = \frac{1+(-1)^s}{2} \\ \overline{s} = \frac{1-(-1)^s}{2} \end{matrix}$$

$$a = \frac{x-\lambda^s_{12}\cdot(M+1)}{\lambda^s_{11}} - (j-1)\cdot\alpha \;;$$

$$b = \frac{y-\lambda^s_{22}\cdot(N+1)}{\lambda^s_{21}} - (i-1)\cdot\alpha \;;$$

$$c = \frac{z-\beta\cdot(k-1)-\lambda^s_{32}\cdot(\beta+1)}{\lambda^s_{31}}$$

$$\left[x_s, y_s, z_s\right] = \begin{bmatrix} \lambda^s_{11}\left[(j-1)\cdot\alpha+a\right]+\lambda^s_{12}(M+1), \lambda^s_{21}\left[(i-1)\cdot\alpha+b\right] \\ +\lambda^s_{22}(N+1), (k-1)\beta+\lambda^s_{31}c+\lambda^s_{32}(\beta+1) \end{bmatrix}$$

$$\Gamma = \overline{\overline{\beta}}\cdot\left[\overline{\overline{\alpha}}\cdot\overline{\overline{(A+B+C)}}+\overline{\alpha}\cdot\overline{\overline{(A+B+C+I+J)}}\right] + \overline{\beta}\cdot\left[\overline{\overline{\alpha}}\cdot\overline{(A+B+C+k)}+\overline{\alpha}\cdot\overline{(A+B+C+I+J+k)}\right]$$

$$\left[A,B,C\right] = \left[\lambda^s_{11}a+\lambda^s_{12}(\alpha+1), \lambda^s_{21}b+\lambda^s_{22}(\alpha+1), \lambda^s_{31}c+\lambda^s_{32}(\beta+1)\right]$$

$$\left[J,I\right] = \left[\lambda^s_{11}j+\lambda^s_{12}(M+1), \lambda^s_{21}i+\lambda^s_{22}(N+1)\right]$$

DLLDesigner

```
REG_222_p_in_1 <= (NODE_211_out(2),NODE_211_out(6),NODE_211_out(7),NODE_211_out(10),
                   NODE_211_out(11),NODE_411_out(2),NODE_411_out(6),NODE_411_out(7),
                   NODE_411_out(10),NODE_411_out(11),NODE_611_out(2),NODE_611_out(6),
                   NODE_611_out(7),NODE_112_out(2),NODE_112_out(3),NODE_112_out(10),
                   NODE_112_out(11),NODE_312_out(2),NODE_312_out(3),NODE_312_out(6),
                   NODE_312_out(7),NODE_312_out(10),NODE_312_out(11),NODE_512_out(2),
                   NODE_512_out(3),NODE_512_out(6),NODE_512_out(7),NODE_512_out(10),
                   NODE_512_out(11),NODE_213_out(3),NODE_213_out(7),NODE_213_out(11),
                   NODE_413_out(3),NODE_413_out(7),NODE_413_out(11),NODE_613_out(3),
                   NODE_613_out(7),NODE_121_out(2),NODE_121_out(10),NODE_121_out(11),
                   NODE_321_out(2),NODE_321_out(6),NODE_321_out(7),NODE_321_out(10),
                   NODE_321_out(11),NODE_521_out(2),NODE_521_out(6),NODE_521_out(7),
                   NODE_521_out(10),NODE_521_out(11),NODE_222_out(2),NODE_222_out(3),
                   NODE_222_out(6),NODE_222_out(7),NODE_222_out(10),NODE_222_out(11),
                   NODE_422_out(2),NODE_422_out(3),NODE_422_out(6),NODE_422_out(7),
                   NODE_422_out(10),NODE_422_out(11),NODE_622_out(2),NODE_622_out(3),
                   NODE_622_out(6),NODE_622_out(7),NODE_123_out(3),NODE_123_out(11),
                   NODE_323_out(3),NODE_323_out(7),NODE_323_out(11),NODE_523_out(3),
                   NODE_523_out(7),NODE_523_out(11),NODE_231_out(6),NODE_231_out(10),
                   NODE_431_out(6),NODE_431_out(10),NODE_631_out(6),NODE_132_out(10),
                   NODE_332_out(6),NODE_332_out(10),NODE_532_out(6),NODE_532_out(10));

(NODE_211_in(2),NODE_211_in(6),NODE_211_in(7),NODE_211_in(10),NODE_211_in(11),
NODE_411_in(2),NODE_411_in(6),NODE_411_in(7),NODE_411_in(10),NODE_411_in(11),
NODE_611_in(2),NODE_611_in(6),NODE_611_in(7),NODE_112_in(2),NODE_112_in(3),
NODE_112_in(10),NODE_112_in(11),NODE_312_in(2),NODE_312_in(3),NODE_312_in(6),
NODE_312_in(7),NODE_312_in(10),NODE_312_in(11),NODE_512_in(2),NODE_512_in(3),
NODE_512_in(6),NODE_512_in(7),NODE_512_in(10),NODE_512_in(11),NODE_213_in(3),
NODE_213_in(7),NODE_213_in(11),NODE_413_in(3),NODE_413_in(7),NODE_413_in(11),
NODE_613_in(3),NODE_613_in(7),NODE_121_in(2),NODE_121_in(10),NODE_121_in(11),
NODE_321_in(2),NODE_321_in(6),NODE_321_in(7),NODE_321_in(10),NODE_321_in(11),
NODE_521_in(2),NODE_521_in(6),NODE_521_in(7),NODE_521_in(10),NODE_521_in(11),
NODE_222_in(2),NODE_222_in(3),NODE_222_in(6),NODE_222_in(7),NODE_222_in(10),
NODE_222_in(11),NODE_422_in(2),NODE_422_in(3),NODE_422_in(6),NODE_422_in(7),
NODE_422_in(10),NODE_422_in(11),NODE_622_in(2),NODE_622_in(3),NODE_622_in(6),
NODE_622_in(7),NODE_123_in(3),NODE_123_in(11),NODE_323_in(3),NODE_323_in(7),
NODE_323_in(11),NODE_523_in(3),NODE_523_in(7),NODE_523_in(11),NODE_231_in(6),
NODE_231_in(10),NODE_431_in(6),NODE_431_in(10),NODE_631_in(6),NODE_132_in(10),
NODE_332_in(6),NODE_332_in(10),NODE_532_in(6),NODE_532_in(10)) <= REG_222_p_out_1;
```

Length of generated VHDL file:
128 nodes per chip: **~10000** lines
64 nodes per chip: **~5500** lines

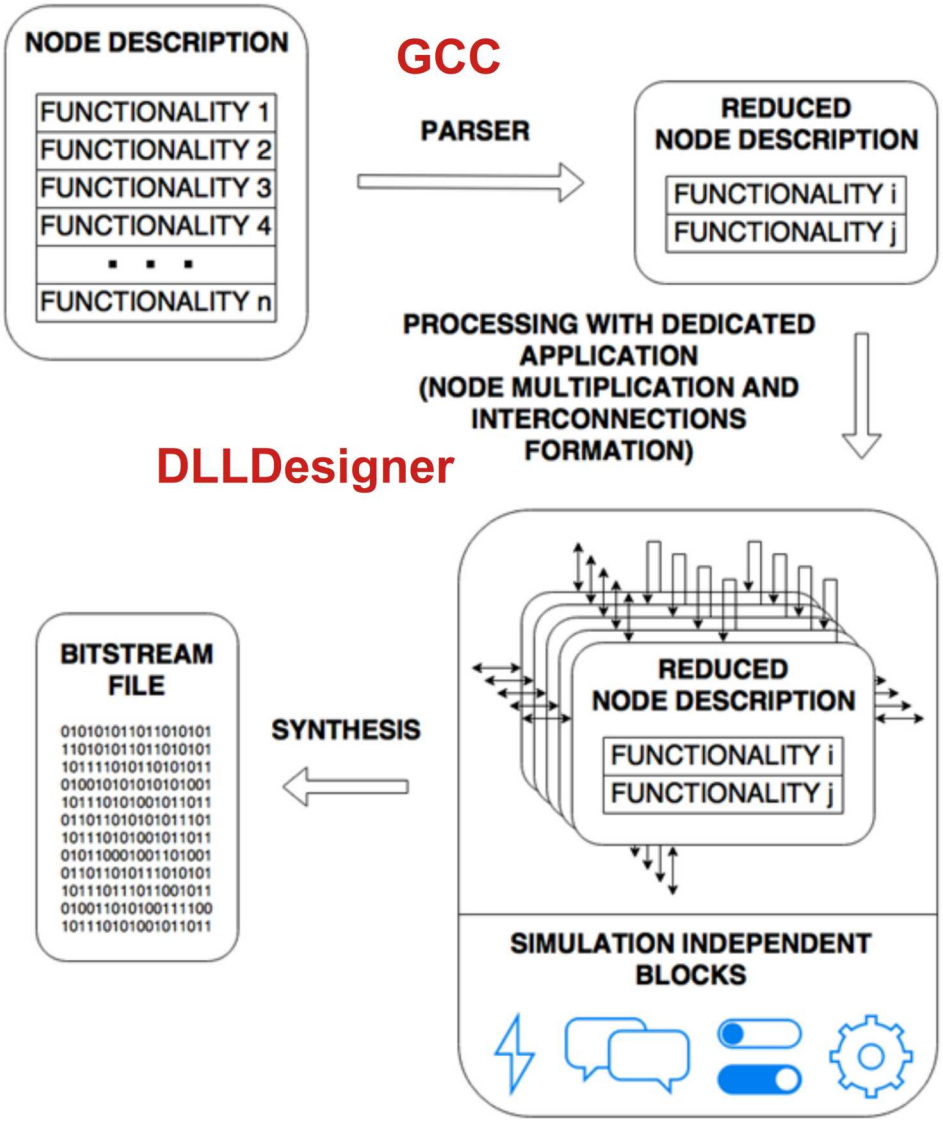
Firmware generation

```
511     r_bonds(10) when x"A",
512     r_bonds(11) when x"B",
513     '-' when others;
514 #endif /* SIMDEF_BOND_BINDS+SIMDEF_BOND_BREAKS
515
516 #if SIMDEF_WAYS+SIMDEF_BOND_BREAKS+SIMDEF_BOND_
517     DIRECTION_PRIORITY_SEQ_INST: entity work.dir
518     port map (
519     pi_clk => pi_clk,
520     pi_start => pi_inverse_
521     pi_dir_priority => s_rng_dir,
```

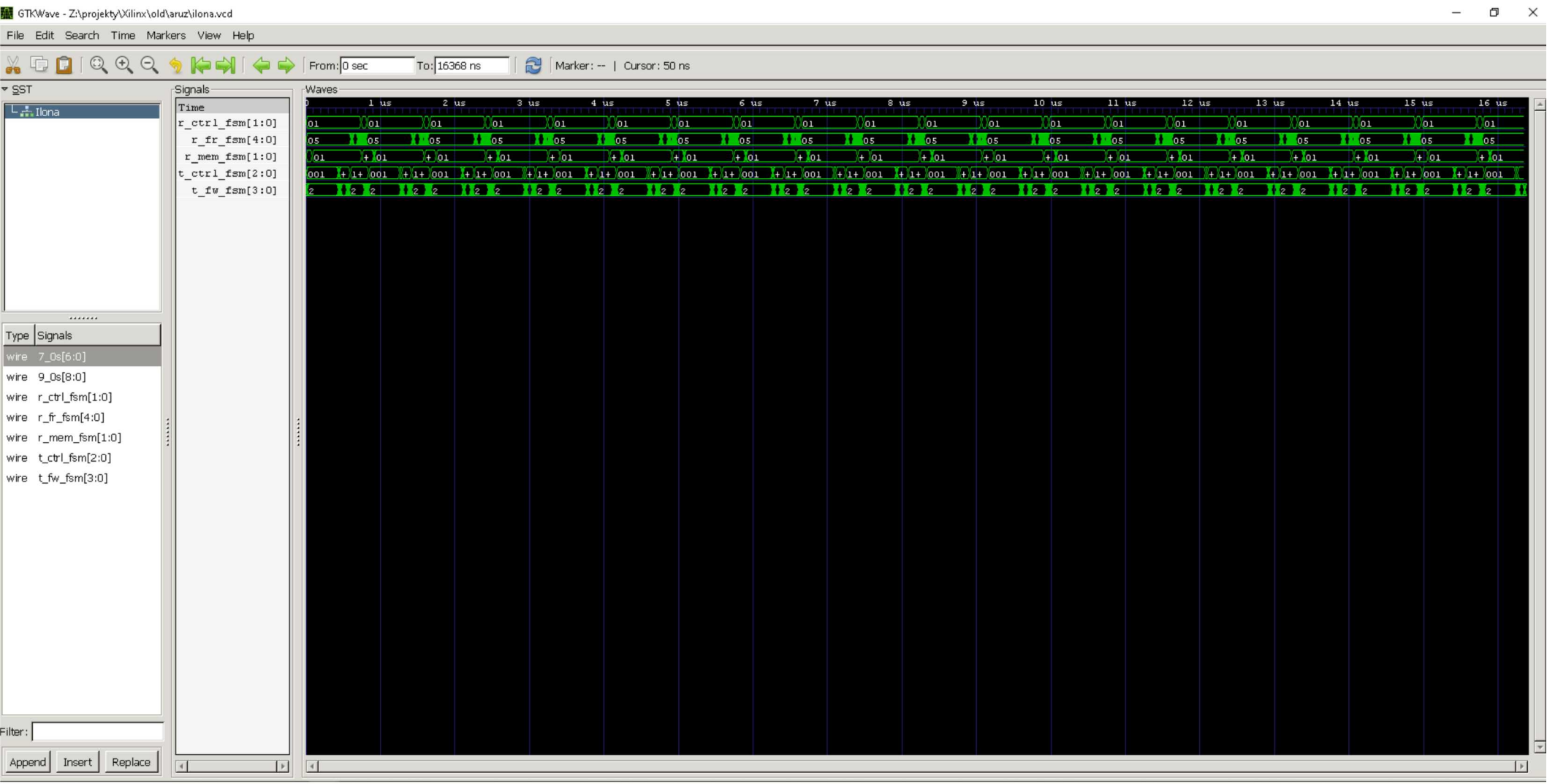
For each algorithm variant dedicated FPGA configuration is prepared. In this way

hardware (configuration) is always optimized to the task (requirements)!

Such an approach increases the efficiency and reduces power consumption.



Large scale debugging



At large scale - rules and recommendations

1) Go hybrid… top-down + bottom-up + scale-up, "divide and conquer", one problem at a time
2) automate everything to avoid human errors as much as possible
3) add traceability (add checksums, version control, timestamps, user id, etc.)
4) everything becomes a needle in a haystack
   - ✔ so always to try scale down the problem space first
   - ✔ for magical issues be prepared to go BIG (example: metastability)
5) every fix must be tested – solving over masking
6) you must know the system - the system must be predictable, observe changes

Systematic approach to systematic issues.

At large scale - rules and recommendations

7) prevention is faster, so leave no stone unturned, look for the problems before they come to light (example: misconnected cables)

8) keep the code readable and tidy (notation, indentation, etc)

9) document everything

10) backup everything (3 backups on various medium / 2 locations)

11) only one project at a time, incentivise the employees, get only the best people, everyone must contribute – small team

12) support the team, keep the spirit up especially when the mood is low

At large scale - rules and recommendations

13) protect the team from any external distractions (especially related to any work for higher management)

14) organize to the last detail

15) short decision path

16) ergonomy and efficiency - allows focusing on what to do next

17) we had 2 week sprints and weekly meeting to check the progress

18) experiment

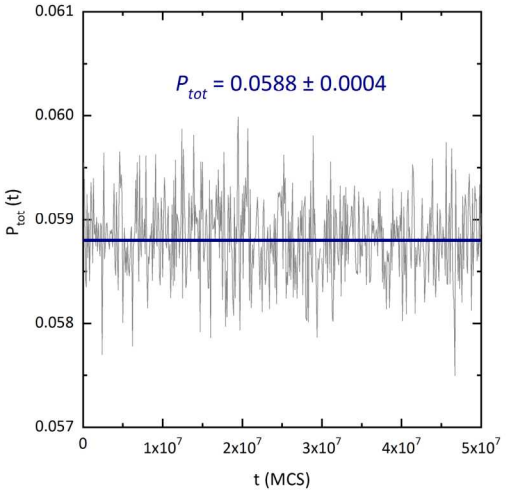
and more...

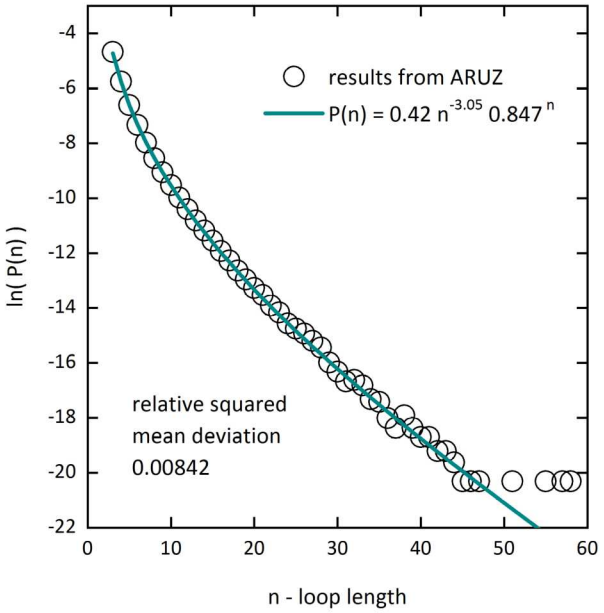
ARUZ vs PC

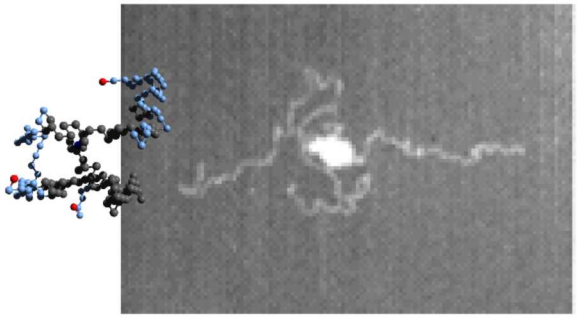
Performance Metric	ARUZ		PC	
	ALC	SLC	x 1	x 6
Speed [cycles/s]	*10 000*	4 100	8	6
Speed [LUPS]	*15e+9*	6.15e+9	12e+6	(9e+6) x 6
Power [kW]	*100*	100	0.14	0.21/6 = 0.035
Energy [GJ]	***100***	244	**175**	58
Time [days]	***12***	28	**14 468 (40 years)**	19 290 (58 years)

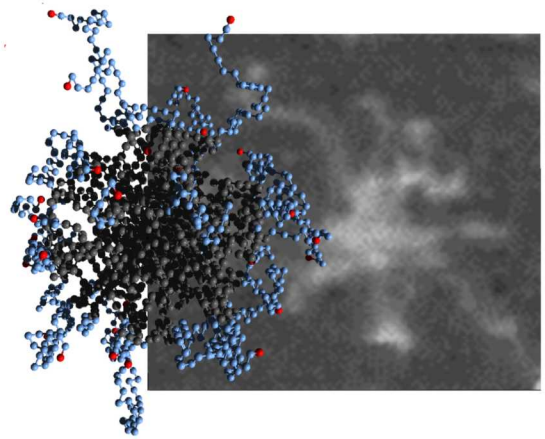
- ✔ **Simulation**: 1.5 million molecules, extended DLL (all mechanisms), 1e+10 cycles.
- ✔ **ARUZ**: 72 NPC, 18x12x12 Dboards
- ✔ **PC**: i7-4930K@4.1 GHz (6 cores with Hyper Threading), 16 GB of DDR3 2400 MHz
- ✔ **ALC** - Accelerated Local Communication, **SLC** - Standard Local Communication
- ✔ **Regular fonts** – measurements, **Italics** – estimations
- ✔ "**x 1**" – one core used, "**x 6**" – 6 independent simulations running simultaneously

Compliance of theory and practice









Comparison with AFM picture (form: Kumaki, J.; Nishikawa, Y.; Hashimoto, T.; J. Am. Chem. Soc. 1996, 118, 3321-3322 and Tsitsilianis, C.; Soft Matter 2010, 6, 2372-2388)

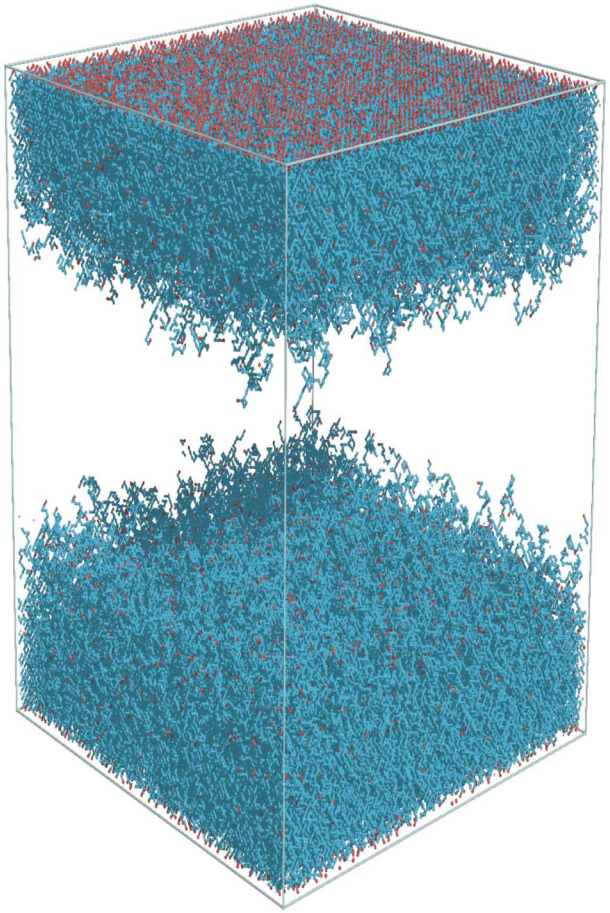
ARUZ publications

1. KIELBIK R., HALAGAN K., ZATORSKI W., JUNG J., ULANSKI J., NAPIERALSKI A., RUDNICKI K., AMROZIK P., JABLONSKI G., STOZEK D., POLANOWSKI P., MUDZA Z., KUPIS J, PANEK P.: “ARUZ - Large-scale, massively parallel FPGA-based analyzer of real complex systems”, Computer Physics Communications, 2018, DOI: 10.1016/j.cpc.2018.06.010

2. JUNG J., KIELBIK R., RUDNICKI K., HALAGAN K., POLANOWSKI P., SIKORSKI A.: „From the Dynamic Lattice Liquid Algorithm to the Dedicated Parallel Computer - mDLL Machine”, Computational Methods In Science And Technology, ISSN 1505-0602, eISSN 2353-9453, 2018, Volume 24 (4) 2018, p. 235–247

3. KIELBIK R., RUDNICKI K., MUDZA Z., JUNG J.: “Methodology of Firmware Development for ARUZ—An FPGA-Based HPC System”, MDPI Electronics 2020, 9(9), 1482;

3 more on the way + a few in the pipeline...



Thank you.