



14th HiPEAC Workshop on Reconfigurable Computing (WRC'2020)
January 20th, 2020, Bologna, Italy

Scalable performance by scaling out FPGA systems via inexpensive interconnects and the open-source AXIOM software stack

Roberto Giorgi
University of Siena, Italy

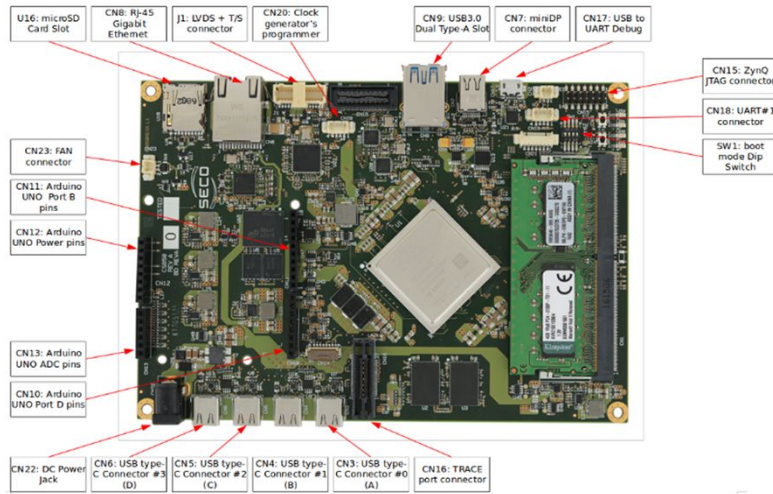
<http://www.dii.unisi.it/~giorgi>

Legal Disclaimer:

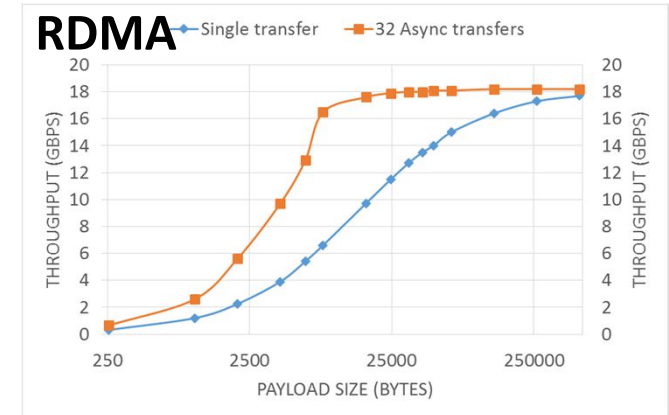
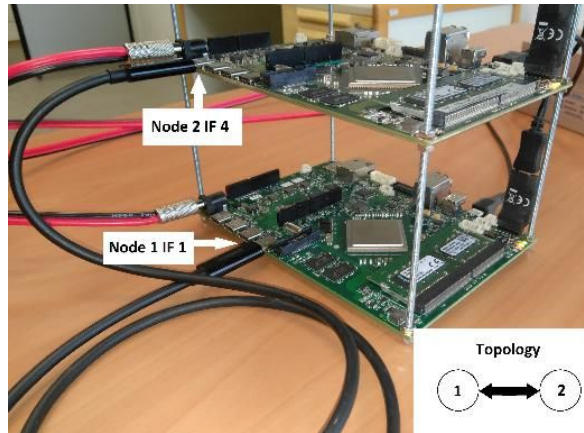
© 2015-2018 AXIOM Consortium, All Rights Reserved. All other trademarks and copyrights are the property of their respective owners. The list of author does not imply any claim of ownership on the Intellectual Properties described in this document. The authors and the publishers make no expressed or implied warranty of any kind and assume no responsibilities for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information contained in this document. The technology disclosed herein may be protected by one or more patents, copyrights, trademarks and/or trade secrets owned by or licensed to AXIOM Partners. The Partners reserve all rights with respect to such technology and related materials. Any use of the protected technology and related material beyond the terms of the License without the prior written consent of AXIOM is prohibited. This document contains material that is confidential to AXIOM and its members and licensors. Until publication, the user should assume that all materials contained and/or referenced in this document are confidential and proprietary unless otherwise indicated or apparent from the nature of such materials (for example, references to publicly available forms or documents). Disclosure or use of this document or any material contained herein, other than as expressly permitted, is prohibited without the prior written consent of AXIOM or such other party that may grant permission to use its proprietary material. The trademarks, logos, and service marks displayed in this document are the registered and unregistered trademarks of AXIOM, its members and its licensors. The copyright and trademarks owned by AXIOM, whether registered or unregistered, may not be used in connection with any product or service that is not owned, approved or distributed by AXIOM, and may not be used in any manner that is likely to cause customer confusion or that disparages AXIOM. Nothing contained in this document should be construed as granting by implication, estoppel, or otherwise, any license or right to use any copyright without the express written consent of AXIOM, its licensors or a third party owner of any such trademark.

Outline

- The bricks: AXIOM-board and AXIOM-sw_stack
- Scaling out FPGAs: how about this?
- OmpSs @ FPGA
- The Runtime system
- Optimizing data exchange via DF-Threads
- Next: the GLUON-board !



Bidirectional links



• UNIQUE FEATURES

- SCALABILITY via USB-C cable: building clusters up to 255 boards
- 4 channels on USB-C cable @ 18Gbps (custom protocol)
- PROGRAMMABILITY via OpenMP: TRANSPARENT FPGA acceleration + CLUSTER distribution
- ARDUINO-UNO socket on board for easy interfacing with the physical world
- 250MHz Trace port – Lauterbach compatible
- Open-source software stack + BSP: <https://git.axiom-project.eu/> (10⁶+ Lines of Code!)

http://www.axiom-project.eu/AXIOM_BOARD_GUIDE.pdf

Scalable Embedded Systems – A new concept?

- ▶ 1 board → ~10s, 2 board(USB-C connected)→ ~7s (no optimizations)
- ▶ SAME board type, SAME software, NO programming efforts, still EMBEDDED



Scalable Embedded Systems – A new concept?

- ▶ 1 board → ~10s, 2 board(USB-C connected)→ ~7s (no optimizations)
- ▶ SAME board type, SAME software, NO programming efforts, still EMBEDDED

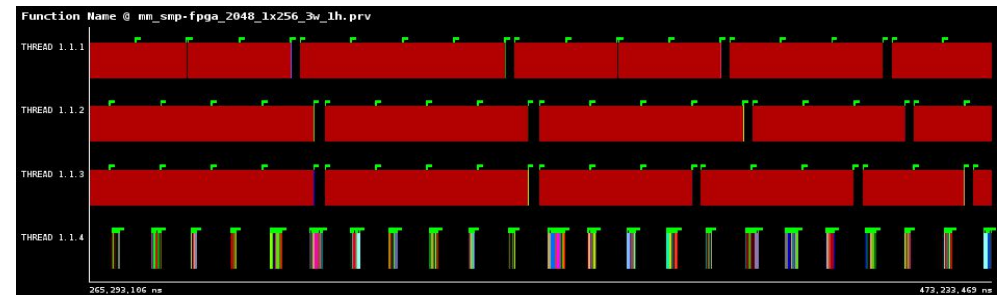


Benchmark

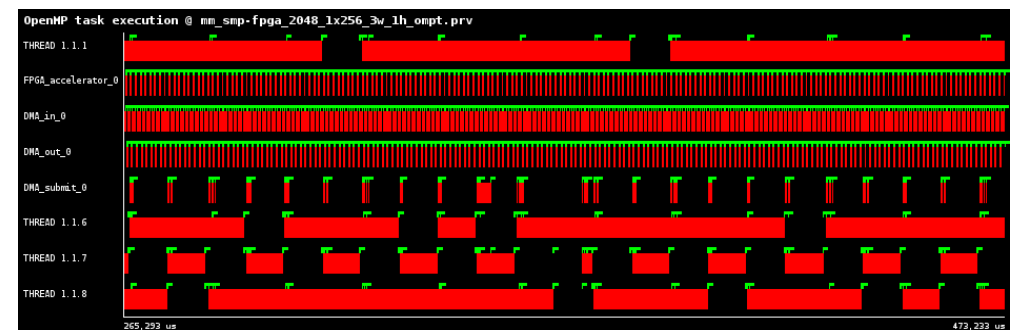
- **MatrixMultiply** with **OmpSS@cluster** over **AXIOM conduit**
 - https://git.axiom-project.eu/axiom-evi/blob/master/tests/ompss/src/ompss_evimm.c
 - Block Size = 4 - Matrix Size = 800 - Element size = 8 bytes
 - Standard OmpSs@cluster
 - 2 boards (3 working threads + 1 I/O thread per board)
 - Execution time: **6889 msec**
 - 1 board (4 working threads)
 - Execution time: **10245 msec**
 - SCHED_DEADLINE patch
 - 2 boards (4 working threads + 1 I/O thread per board)
 - Execution time: **6308 msec**

FPGA+SMP load balancing

- Originally, helper threads were exclusively dedicated to manage with the FPGA
 - Loose of performance

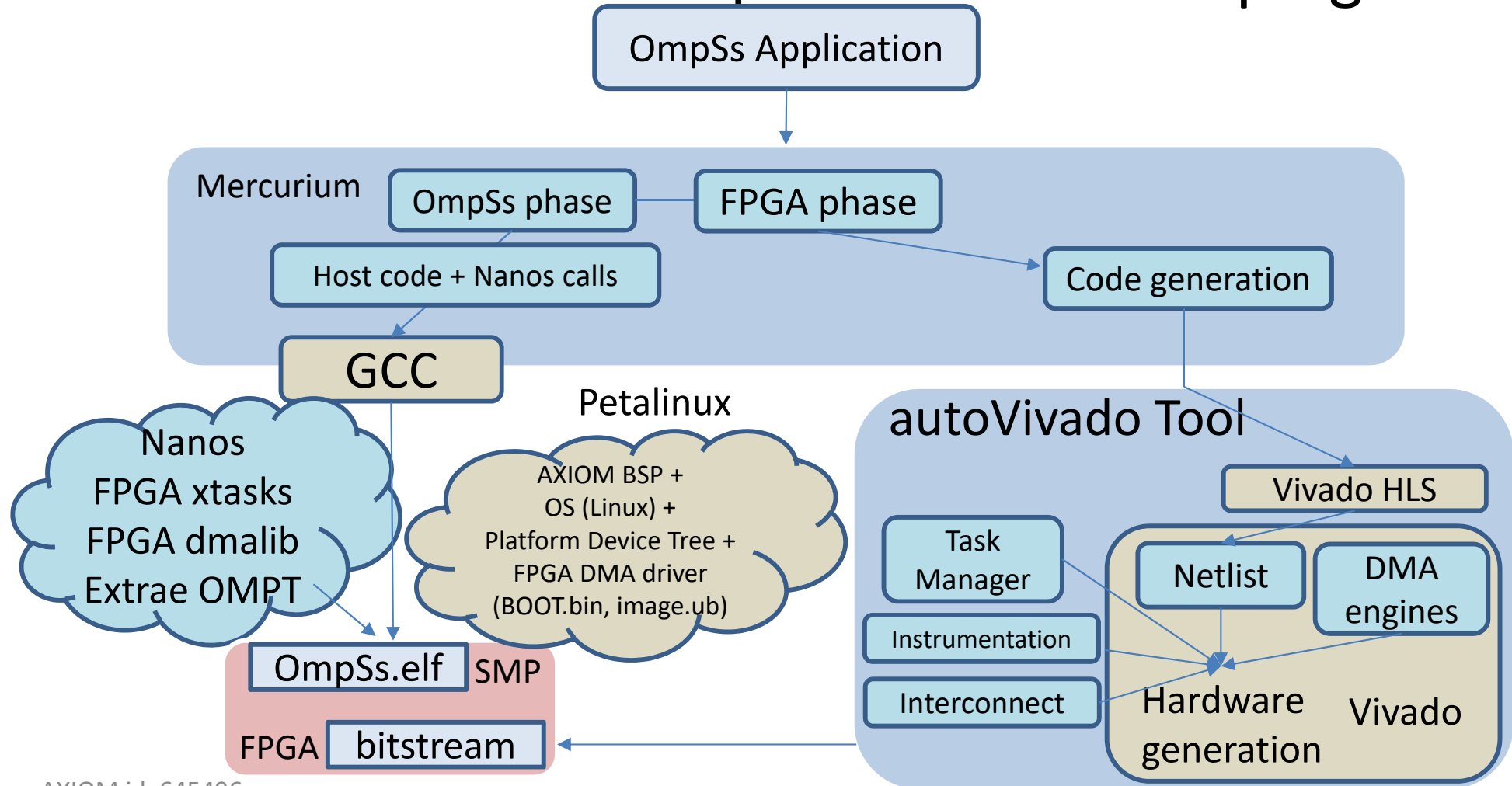


- Hybrid workers
 - Manage the FPGA
 - Execute SMP tasks



OmpSs2 compilation infrastructure

- New Mercurium compiler autoVivado plug-in



AXIOM id. 645496
<http://www.axiom-project.eu>

8

OmpSs2 autoVivado

- Directives guide the compilation & bitstream generation

```
#pragma omp target device(fpga) copy_deps num_instances(2)
#pragma omp task in(a,b) inout(c)
void matrix_multiply(float a[BS][BS], float b[BS][BS], float c[BS][BS]);
...
for (i_b=0; i_b<NB_I; i_b++)
    for (j_b=0; j_b<NB_J; j_b++)
        for (k_b=0; k_b<NB_K; k_b++)
            matrix_multiply(AA[i_b][k_b], BB[k_b][j_b], CC[i_b][j_b]);
...
#pragma omp taskwait
// Or
// other tasks depending on input output c of matrix multiply task
```

OmpSs2 autoVivado

- Coverage

```
#define BS 128
```

```
void matrix_multiply(float a[BS][BS], float b[BS][BS], float c[BS][BS])
```

```
{
```

```
#pragma HLS inline
```

```
int const FACTOR = BS/2;
```

```
#pragma HLS array_partition variable=a block factor=FACTOR dim=2
```

```
#pragma HLS array_partition variable=b block factor=FACTOR dim=1
```

```
// matrix multiplication of a A*B matrix
```

```
for (int ia = 0; ia < BS; ++ia)
```

```
for (int ib = 0; ib < BS; ++ib) {
```

```
#pragma HLS PIPELINE II=1
```

```
float sum = 0;
```

```
for (int id = 0; id < BS; ++id)
```

```
sum += a[ia][id] * b[id][ib];
```

```
c[ia][ib] += sum;
```

```
}
```

```
}
```

Zynq-7000 Family

2x Cortex-A9 cores + FPGA
(32-bit platforms)



SECO AXIOM Board

Zynq U+ XCZU9EG-ES2



Trenz Electronics Zynq U+
TE0808 XCZU9EG-ES1

4x Cortex-A53 cores +
FPGA (64-bit platforms)



Towards
Discrete FPGAs

Xilinx ZCU102
(work in progress)

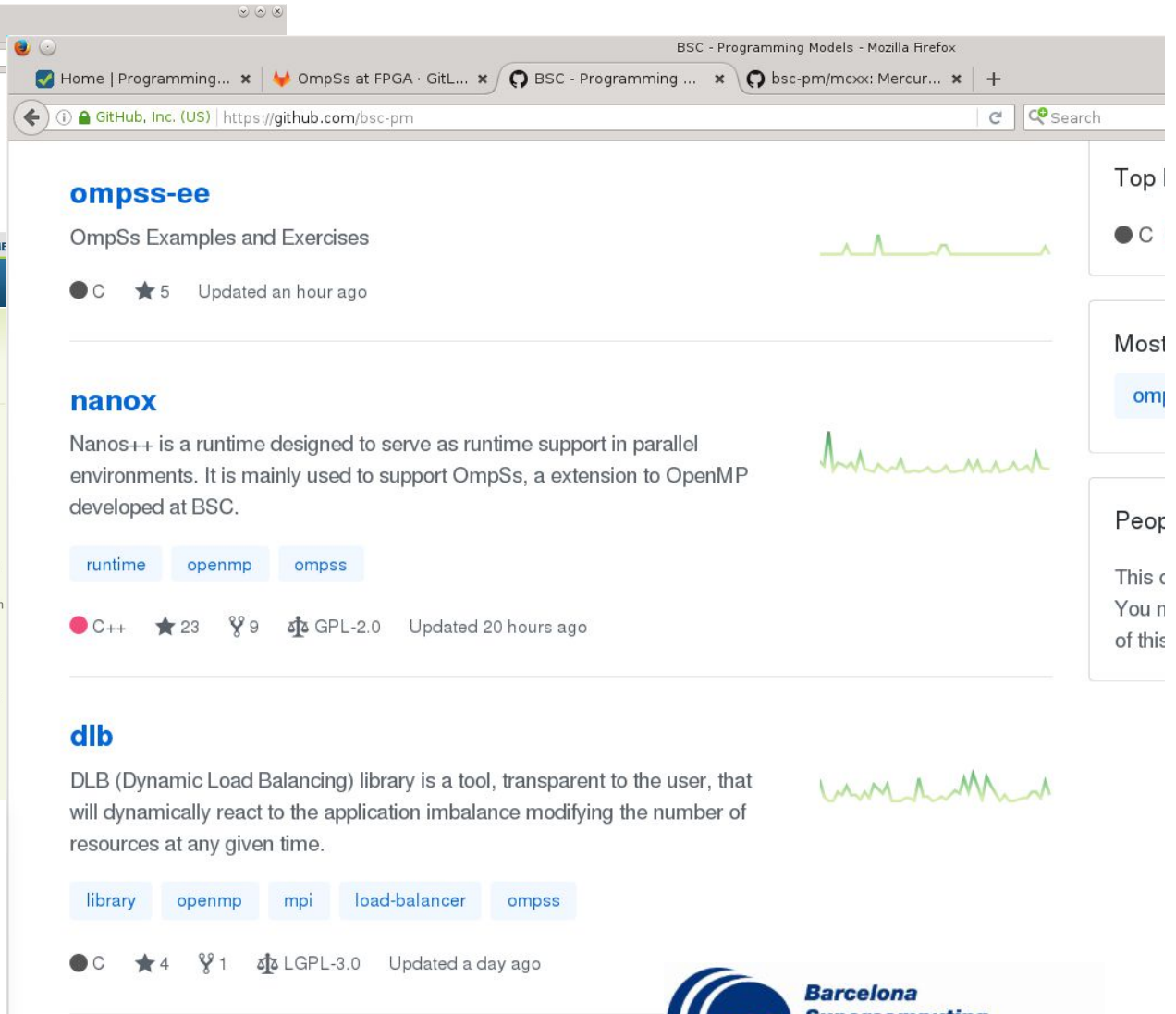


OmpSs/OmpSs2 availability

- pm.bsc.es & public github



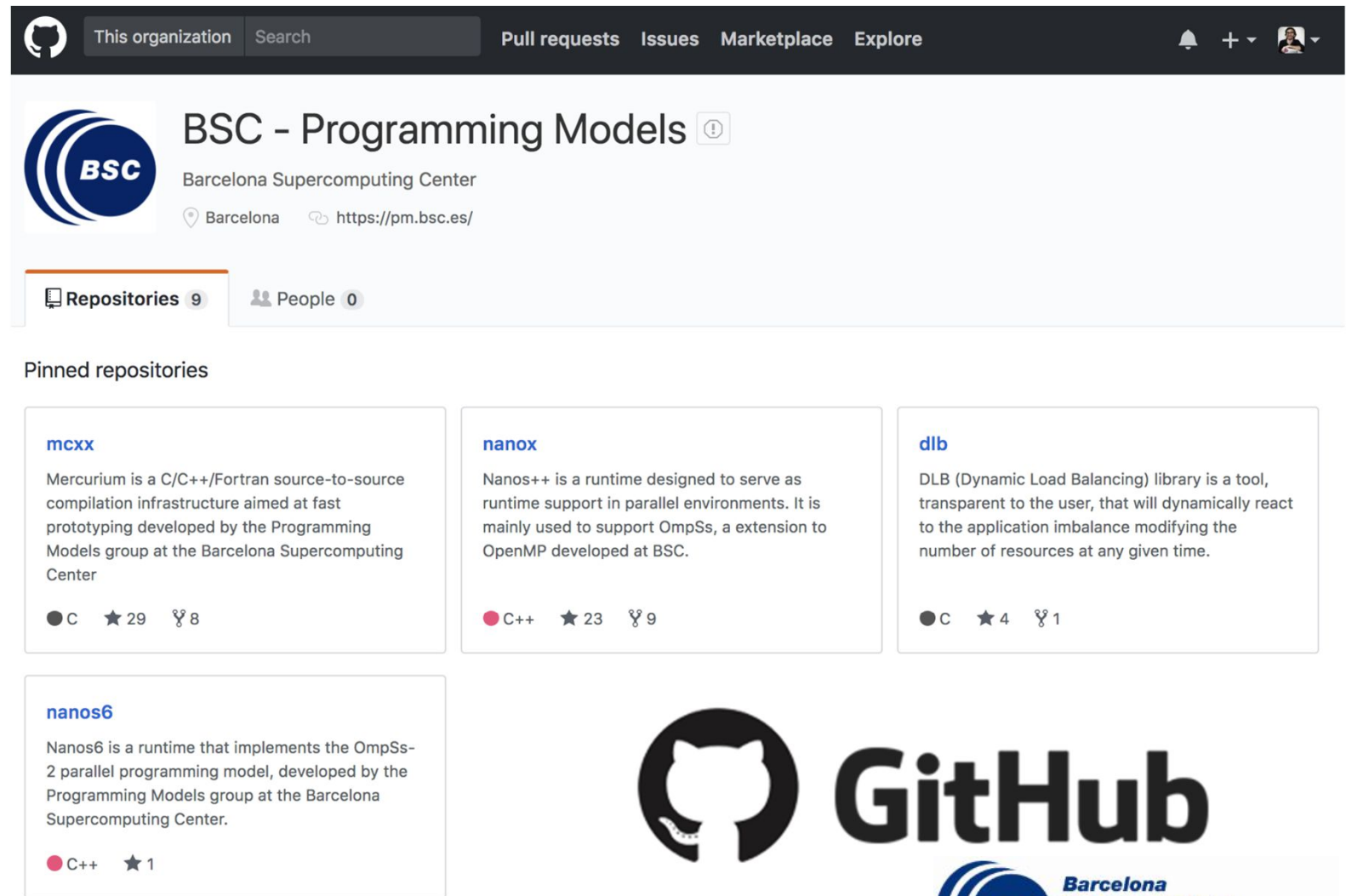
The screenshot shows the homepage of the 'Programming Models @ BSC' website. The header includes a search bar and a 'MAIN MENU' with links to HOME, OMPSS, OMPSS-2, OPENMP, and NANOS++. The main content area is titled 'The OmpSs Programming Model' and describes the project's goal to integrate features from the StarSs programming model. It mentions that OmpSs is built on top of the Mercurium compiler and Nanos++ runtime system. The text also discusses asynchronous parallelism and the use of the 'task' construct for data dependencies.



The screenshot shows the GitHub repository for 'bsc-pm' (Programming Models @ BSC). The repository is titled 'ompss-ee' and contains 'OmpSs Examples and Exercises'. It is a C++ project with 5 stars and was updated an hour ago. The repository also includes a 'nanox' project, which is a runtime designed to serve as runtime support in parallel environments. It is mainly used to support OmpSs, an extension to OpenMP developed at BSC. The repository is licensed under GPL-2.0 and was updated 20 hours ago. The repository also includes a 'dlb' (Dynamic Load Balancing) library, which is a tool transparent to the user that will dynamically react to application imbalance by modifying the number of resources at any given time. The repository is licensed under LGPL-3.0 and was updated a day ago.

GitHub OmpSs/OmpSs2

OmpSs GitHub: <https://github.com/bsc-pm>



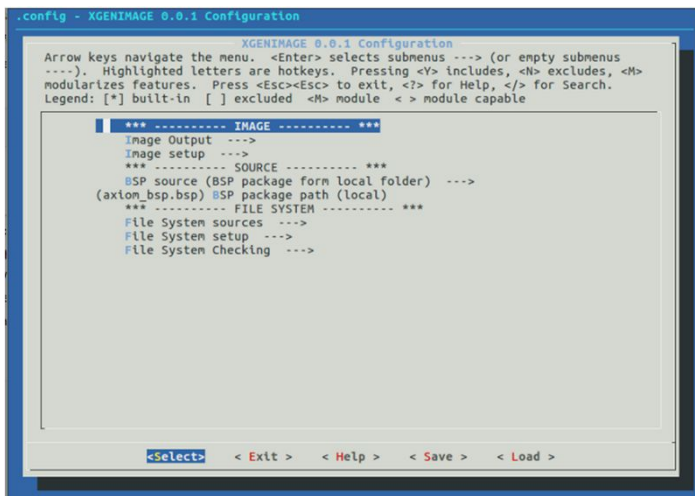
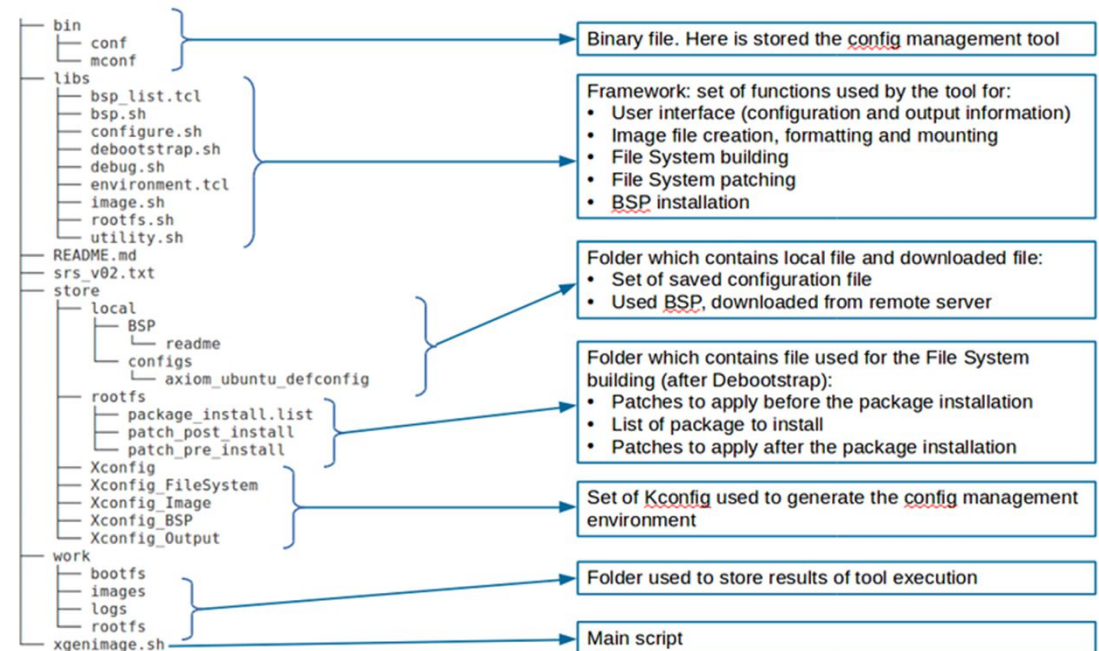
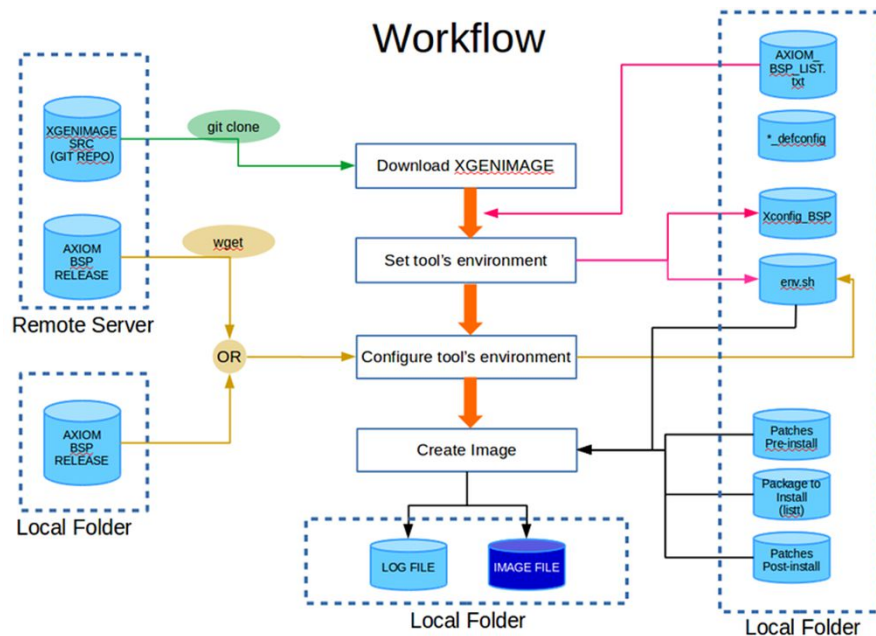
The screenshot shows the GitHub profile page for the organization "BSC - Programming Models". The header includes the GitHub logo, a search bar, and navigation links for "Pull requests", "Issues", "Marketplace", and "Explore". The organization's name "BSC - Programming Models" is displayed with a location pin for "Barcelona" and a website link "https://pm.bsc.es/". Below this, there are tabs for "Repositories" (9) and "People" (0). The "Pinned repositories" section features four cards:

- mcxx**: Mercurium is a C/C++/Fortran source-to-source compilation infrastructure aimed at fast prototyping developed by the Programming Models group at the Barcelona Supercomputing Center. It has 29 stars and 8 forks.
- nanox**: Nanos++ is a runtime designed to serve as runtime support in parallel environments. It is mainly used to support OmpSs, an extension to OpenMP developed at BSC. It has 23 stars and 9 forks.
- dlb**: DLB (Dynamic Load Balancing) library is a tool, transparent to the user, that will dynamically react to the application imbalance modifying the number of resources at any given time. It has 4 stars and 1 fork.
- nanos6**: Nanos6 is a runtime that implements the OmpSs-2 parallel programming model, developed by the Programming Models group at the Barcelona Supercomputing Center. It has 1 star.

At the bottom right, the GitHub logo and the BSC logo (Barcelona Supercomputing Center) are displayed.

Boot Image Generation: XGenImage

Tool to automate generating the boot image



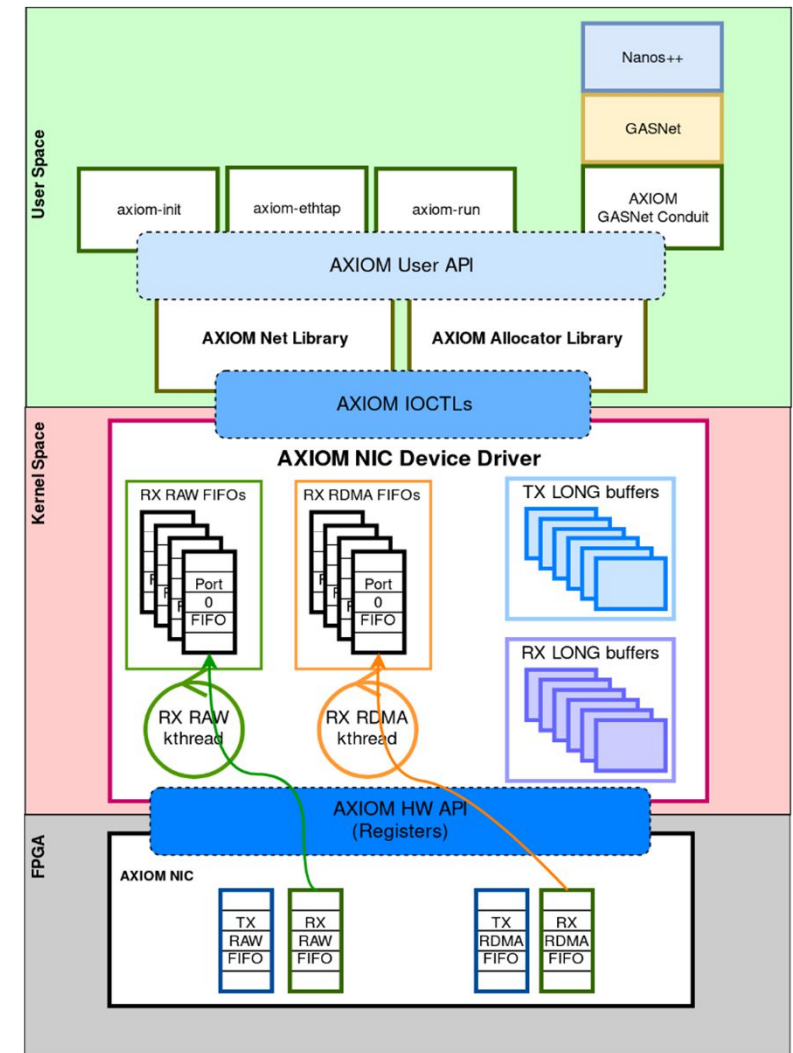
Configuration user interface

Main Tasks

Task	Function Description
Initialization	(REQUIRED) Set the tool with the default configuration
Configuration	(OPTIONAL) User configuration manipulation
Image creation	(FINAL TASK) creation of bootable SD image

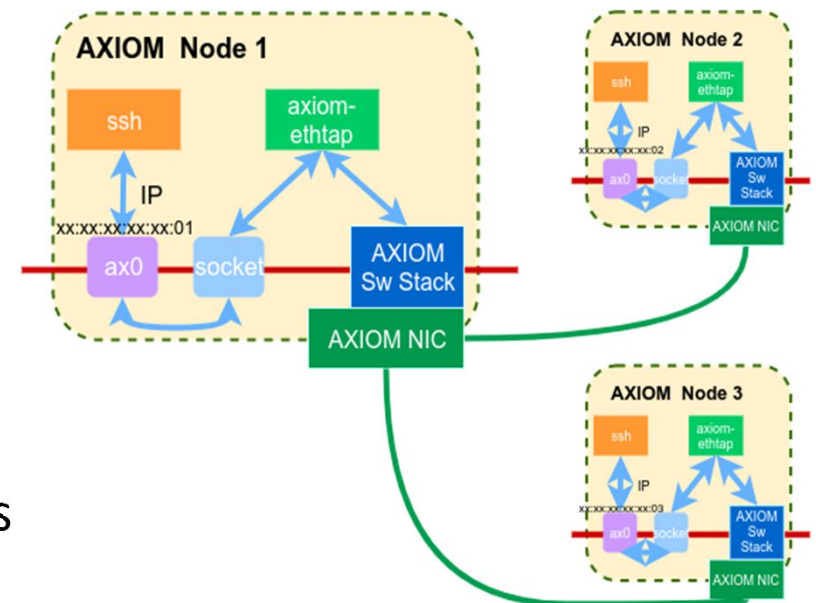
The AXIOM software stack

- Running on the AXIOM board
- AXIOM-Link software specs available
- Device drivers
- Memory allocator
- User space app management
- Utility apps
- Ethernet over AXIOM
- GASNet Spawner
- OmpSs@Cluster



Ethernet over the AXIOM Link

- axiom-ethtap (user space application)
 - creates a TAP interface (ax0)
 - TAP (namely network tap) simulates a link layer device and it operates with “layer 2 packets” like Ethernet frames
 - **Set last byte of MAC address with AXIOM NodeID**
 - Forwards ethernet frames through AXIOM-nic
 - Destination NodeID extracted from the Destination MAC address
 - init script
 - Launched from axiom-init after the end of the discovery algorithm
 - Starts axiom-ethtap and set IP on ax0 (**192.168.17.X -> X = NodeID**)
 - **iperf throughput**
 - 1 thread - 1.9 Gbps / 6 threads - 2.9 Gbps
-
- The diagram illustrates the internal components and data flow of AXIOM Node 1. It is enclosed in a dashed green box. Inside, an orange 'ssh' box is connected via a double-headed blue arrow labeled 'IP' to a purple 'ax0' box. The 'ax0' box has a MAC address 'xx:xx:xx:xx:xx:01' written next to it. A blue arrow points from 'ax0' to a light blue 'socket' box. From the 'socket', a blue arrow points to a green 'axiom-ethtap' box. The 'axiom-ethtap' box is connected via a double-headed blue arrow to a blue 'AXIOM Sw Stack' box. Finally, a green arrow points from the 'AXIOM Sw Stack' to a green 'AXIOM NIC' box outside the node boundary. A red line representing a network connection enters the 'ax0' box from the left, and a green line representing the physical network connection exits from the 'AXIOM NIC' box at the bottom.



Lauterbach and AXIOM board

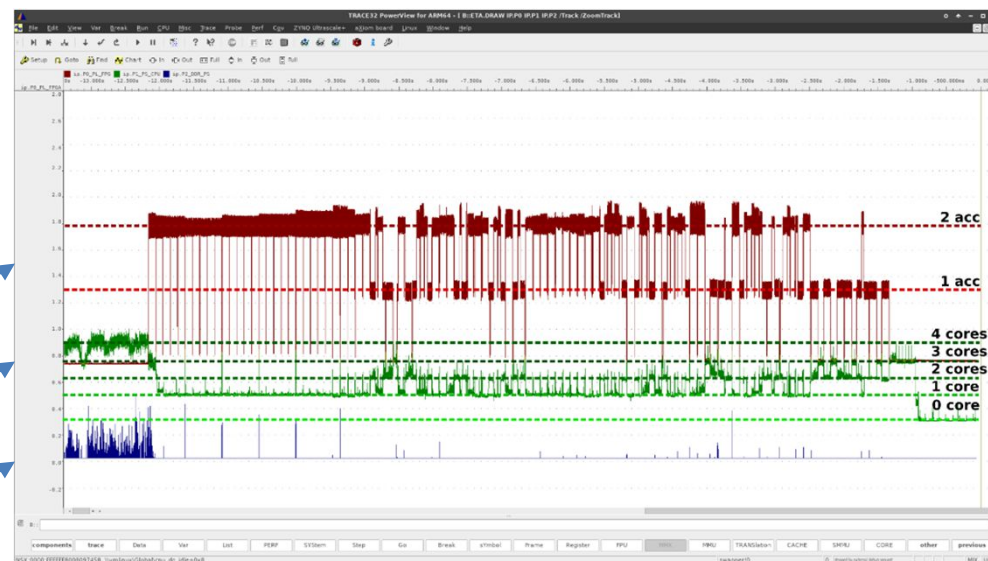
- Debug and Trace
 - U-boot
 - Linux kernel
 - Linux kernel modules
 - User-space app/libraries
- Energy profiling using the Analog Probe
 - AXIOM board modified with cables attached on shunt resistors
 - Possibility to correlate the source code with the energy consumption
- Power profiling of Vimar application (see D7.3)



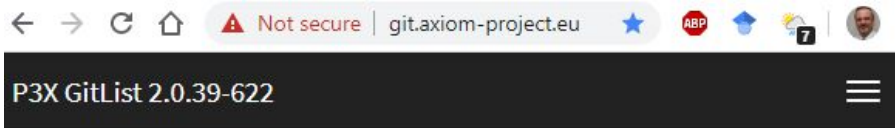
Red - FPGA

Green – CPU cores

Blue - DRAM



git.axiom-project.eu

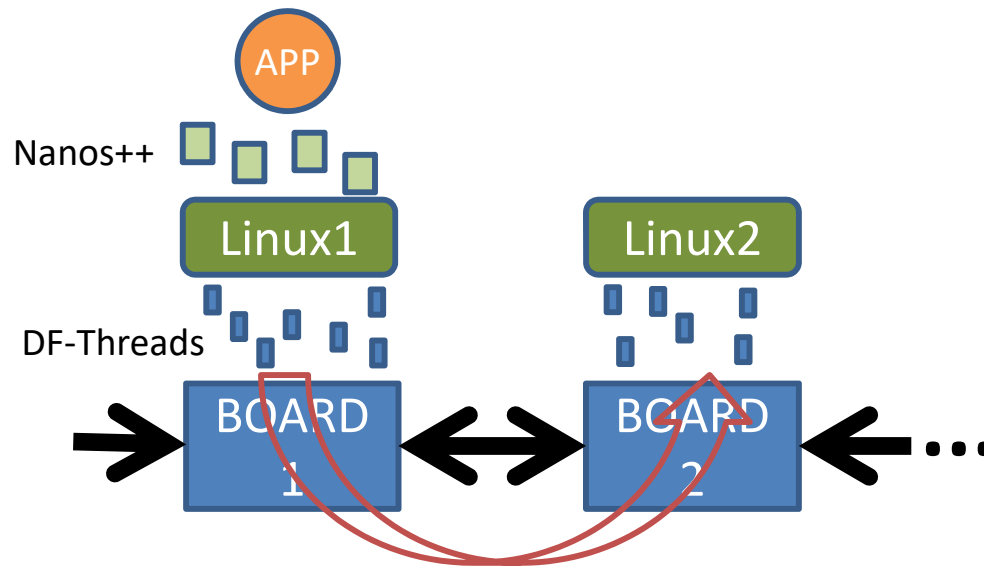


Search

 axiom-allocator 
Implementation of the three level AXIOM allocator
 axiom-evi 
Evidence main repository that includes several sub repositories (git-submodules) and scripts to compile and run emulated AXIOM boards
 axiom-evi-allocator-drv 
Implementation of the AXIOM memory device to handle virtual to physical memory mapping
 axiom-evi-allocator-lib 
Implementation of the AXIOM 3th level software allocator based on LMM
 axiom-evi-apps 
Implementation of AXIOM application and deamons (axiom-init, axiom-run, etc.)
 axiom-evi-buildroot 
Modified version of Buildroot to support AXIOM board
 axiom-evi-extrae 
Modified version of Extrae to support IOCTL and AXIOM api
 axiom-evi-gasnet 

 axiom-evi-linux 
Modified version of Linux kernel to support AXIOM board
 axiom-evi-mcxx 
Modified version of mcxx to support AXIOM GASNet conduit and cross-compilation
 axiom-evi-nanox 
Modified version of nanox to support AXIOM GASNet conduit and cross-compilation
 axiom-evi-nic 
Implementation of AXIOM NIC device driver and User Space libraries
 axiom-evi-qemu 
Modified version of QEMU to emulate the AXIOM NIC device
 axiom-evi-u-boot 
Modified version of u-boot to support AXIOM board

Distribution across nodes/boards



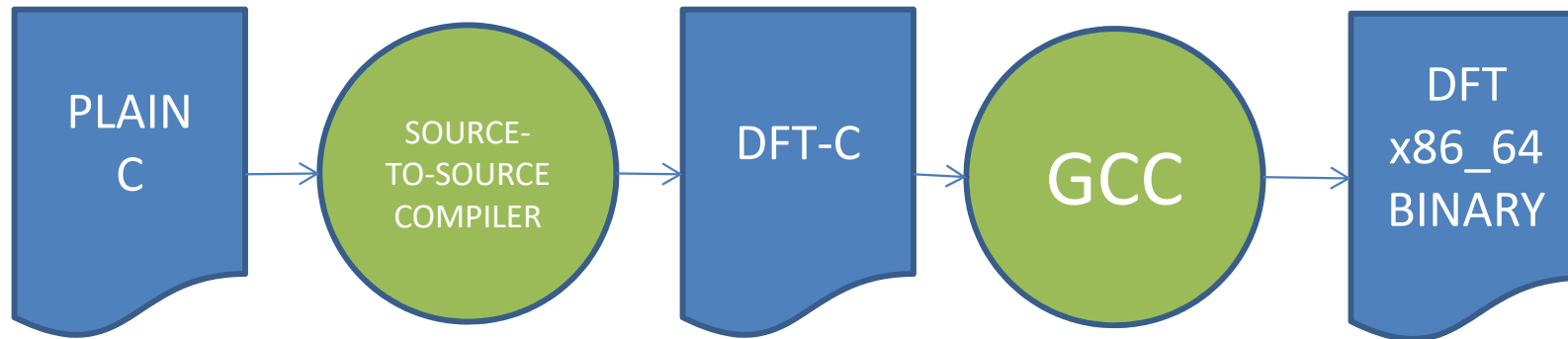
Fundamental approach: DATAFLOW

A Scheme of Computation in which
an activity is initiated by presence
of the data it needs to perform its
function

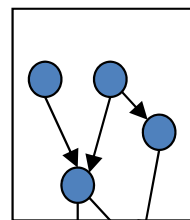
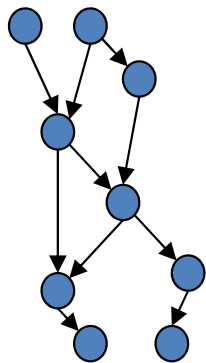


(Jack Dennis)

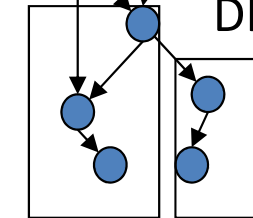
What is a DF-Thread



Program:



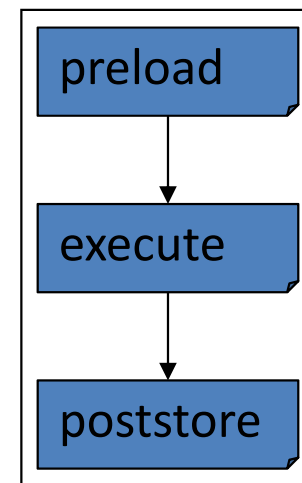
DF-thread1



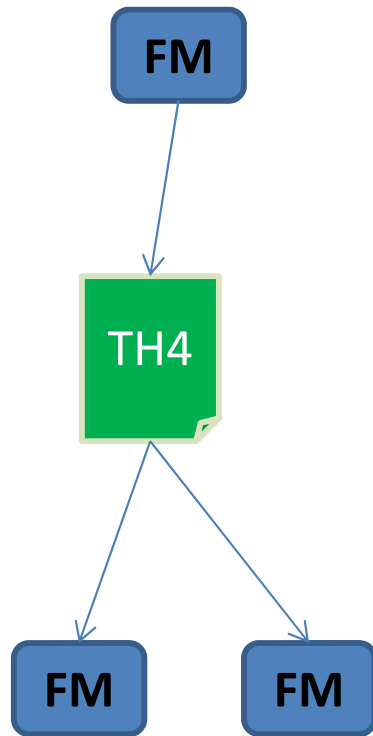
DF-thread2

DF-thread3

DF-thread



DF-Threads Memory Model



- DF-thread

- A function that expects no parameters and returns no parameters.
 - The body of this function can refer to any memory location for which it has got the pointer through DFT function calls (e.g., xpreload, xpoststor, xsubscribe, ...). A DF-thread is identified by an object of type `dftid_t` (df-thread identifier). In other words:

```
typedef void (*dft_t)(void)
```

- INPUT_FRAME, OUTPUT_FRAME

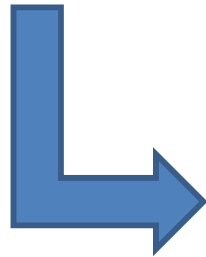
- INPUT_FRAME: A buffer which is allocated in the local memory and contains the input values for the current DF-thread.
- OUTPUT_FRAME: A buffer which is allocated in the local memory and contains values to be used by other DF-threads (consumer DF-threads)

- SYNCHRONIZATION_COUNT

- A number which is initially set to the number of input values (or events) needed by an DF-thread. The SYNCHRONIZATION_COUNT has to be decremented each time the expected data is written in an OUTPUT_FRAME.

DF-Thread Code Example

```
int fibo (int n)
{
    if (n <= 1) return n;
    return fibo(n-1)+fibo(n-2);
}
```



```
void fibo(void) { // DF-Thread Fibonacci
    uint64_t* myfp =(uint64_t*) df_ldframe();
    int n = myfp[1];
    if (n <= 1) {
        df_write(myfp[0],n);
    } else {
        uint64_t* tfib1 = df_schedule(&fibo,2);
        uint64_t* tfib2 = df_schedule(&fibo,2);
        df_write(&tfib1[1], n-1);
        df_write(&tfib2[1], n-2);
        uint64_t* tadd = df_schedule(&adder,3);
        df_write(&tadd[0], myfp[0]);
        df_write(&tfib1[0], tadd+1);
        df_write(&tfib2[0], tadd+2);
    }
    df_destroy();
}

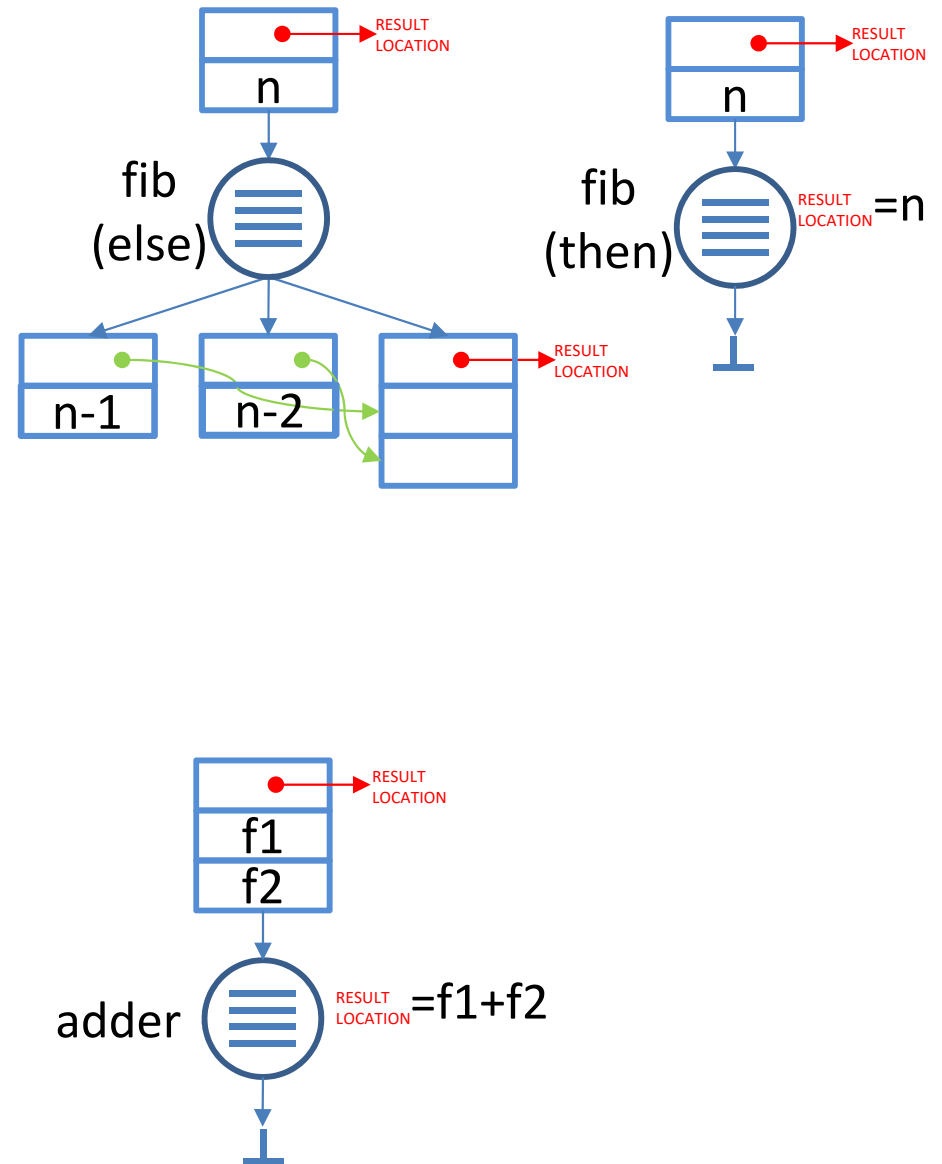
void adder(void) {
    uint64_t* myfp =(uint64_t*) df_ldframe();
    uint64_t f1 = myfp[1];
    uint64_t f2 = myfp[2];
    df_write(myfp[0], f1+f2);
    df_destroy();
}
```

- Example of translation of the Recursive Fibonacci program from C code into DF-Threads API. The two main operations (fibo and adder) are translated into the DF-Threads API.

...in other words...

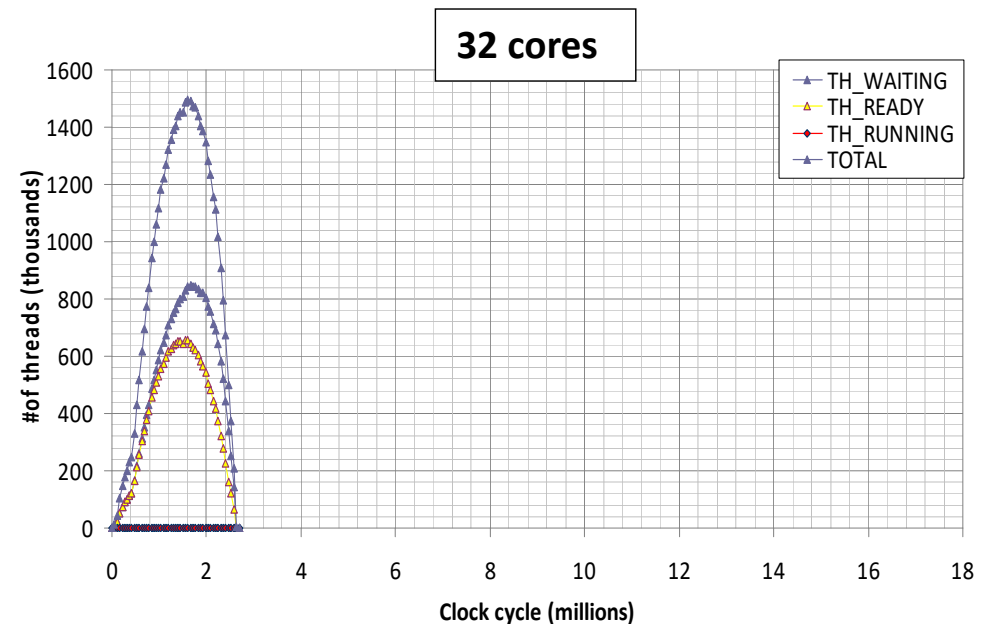
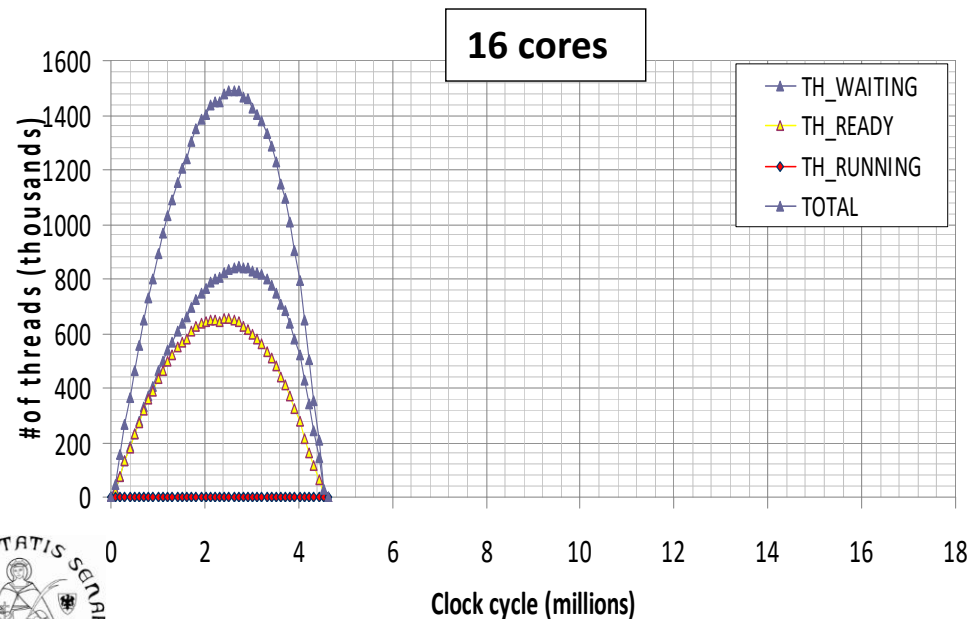
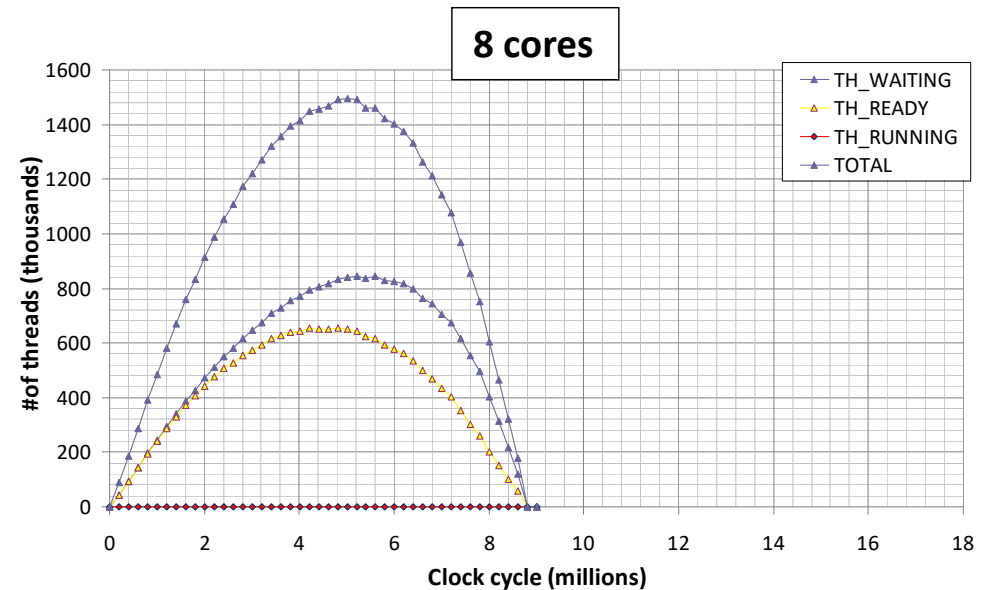
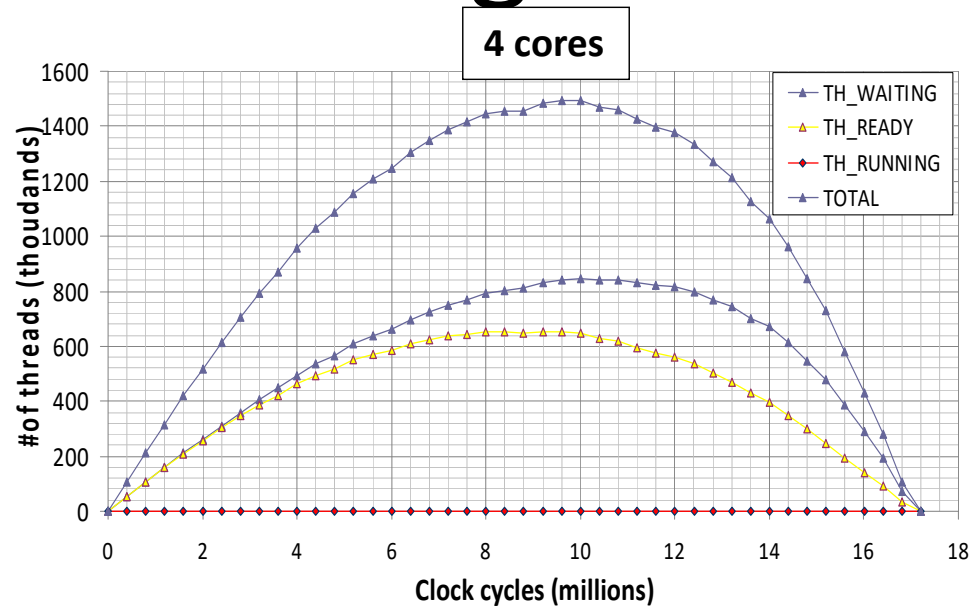
```
void fibo(void) { // DF-Thread Fibonacci
    uint64_t* myfp = (uint64_t*) df_ldframe();
    int n = myfp[1];
    if (n <= 1) {
        df_write(myfp[0], n);
    } else {
        uint64_t* tfib1 = df_schedule(&fibo, 2);
        uint64_t* tfib2 = df_schedule(&fibo, 2);
        df_write(&tfib1[1], n-1);
        df_write(&tfib2[1], n-2);
        uint64_t* tadd = df_schedule(&adder, 3);
        df_write(&tadd[0], myfp[0]);
        df_write(&tfib1[0], tadd+1);
        df_write(&tfib2[0], tadd+2);
    }
    df_destroy();
}

void adder(void) {
    uint64_t* myfp = (uint64_t*) df_ldframe();
    uint64_t f1 = myfp[1];
    uint64_t f2 = myfp[2];
    df_write(myfp[0], f1+f2);
    df_destroy();
}
```

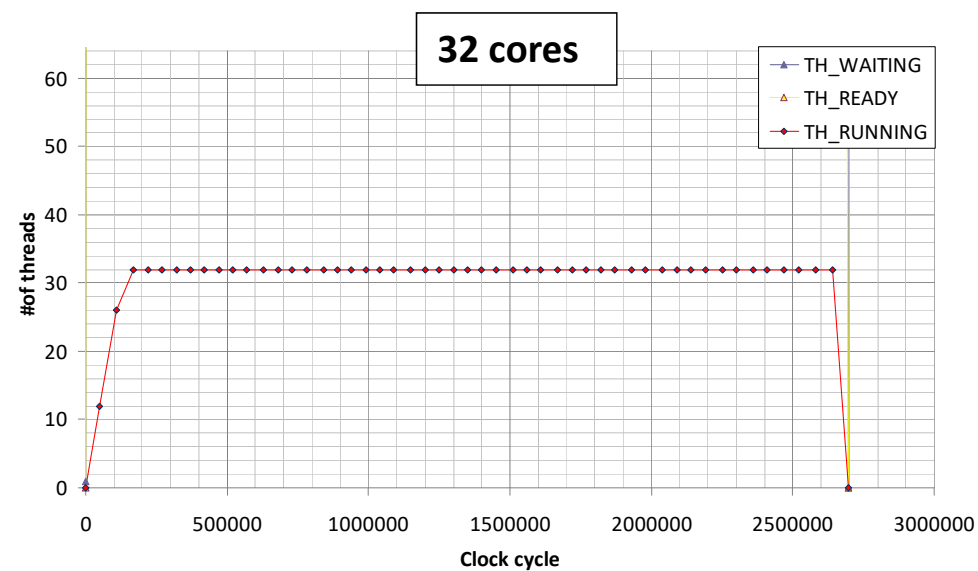
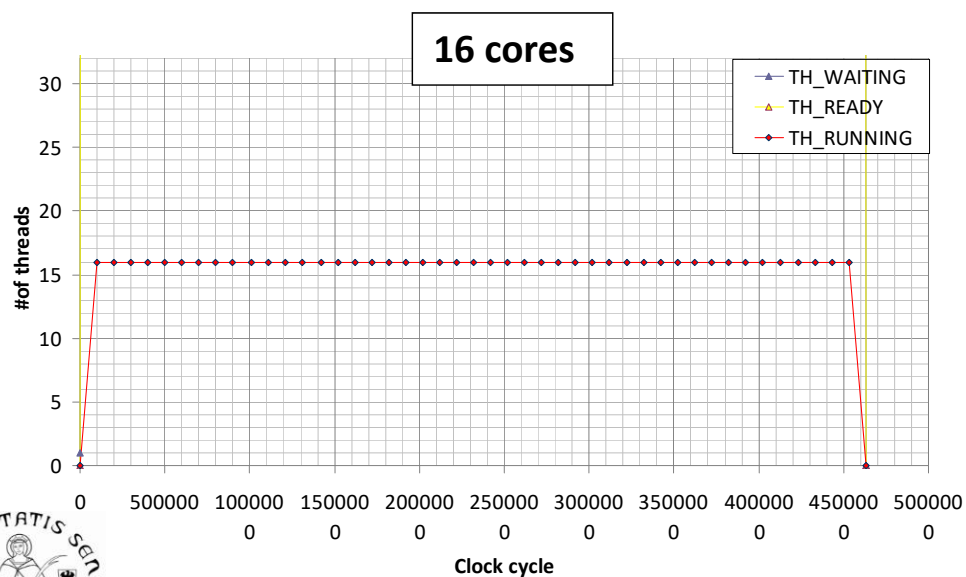
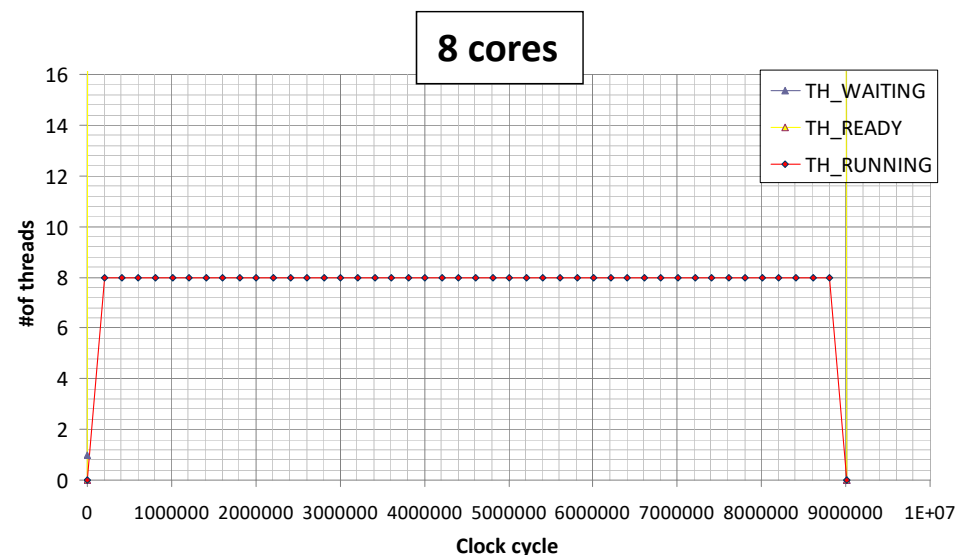
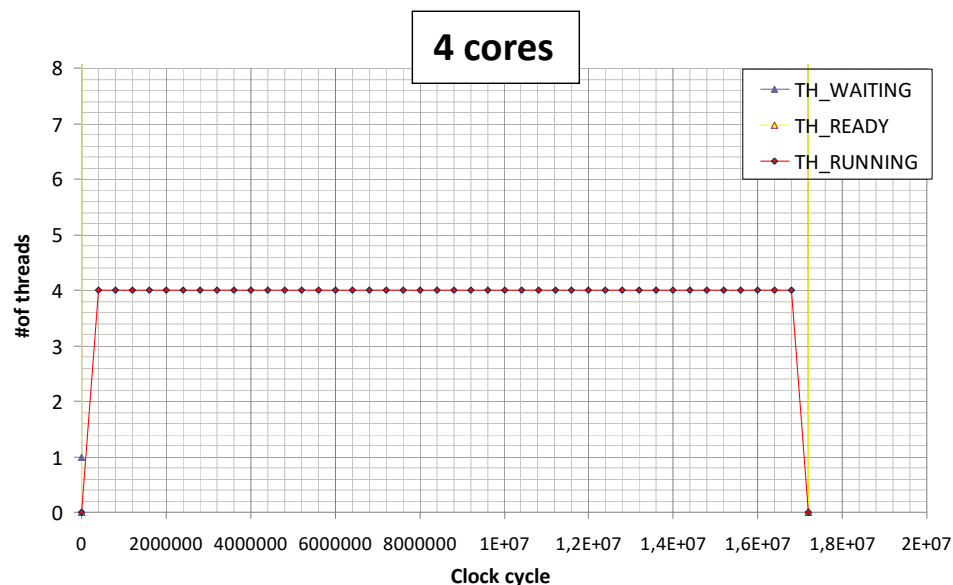


- A simple dataflow runtime to test DF-Threads program: <http://drt.sf.net>

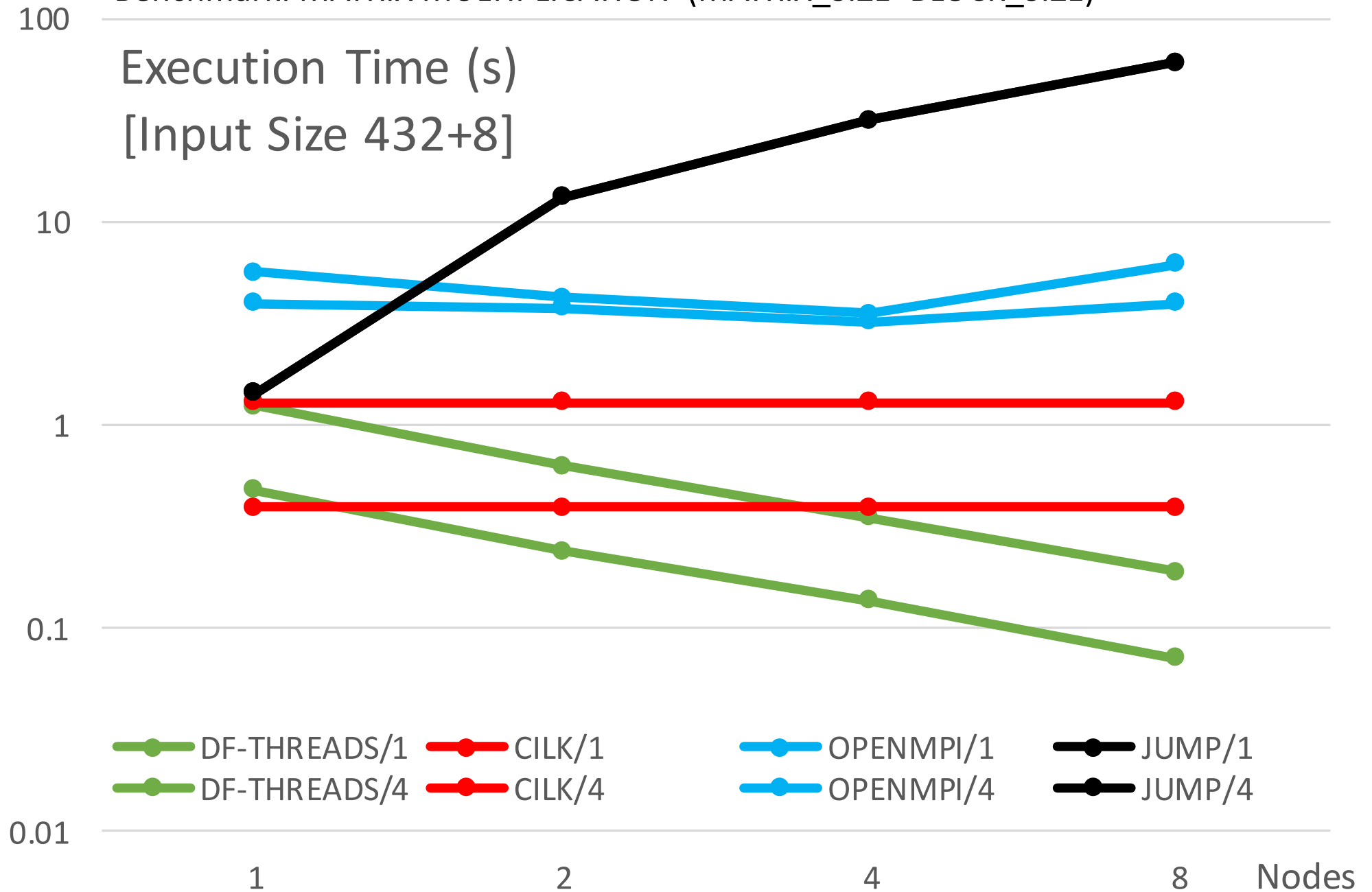
Running fibonacci(35) - simulation



Running fibonacci(35) (zoom)



Benchmark: MATRIX MULTIPLICATION (MATRIX_SIZE+BLOCK_SIZE)



Benchmark: MATRIX MULTIPLICATION (MATRIX_SIZE+BLOCK_SIZE)

1000

Execution Time (s)

[Input Size 832+8]

100

10

1

DF-THREADS/1 CILK/1 OPENMPI/1 JUMP/1
 DF-THREADS/4 CILK/4 OPENMPI/4 JUMP/4

0.1

1

2

4

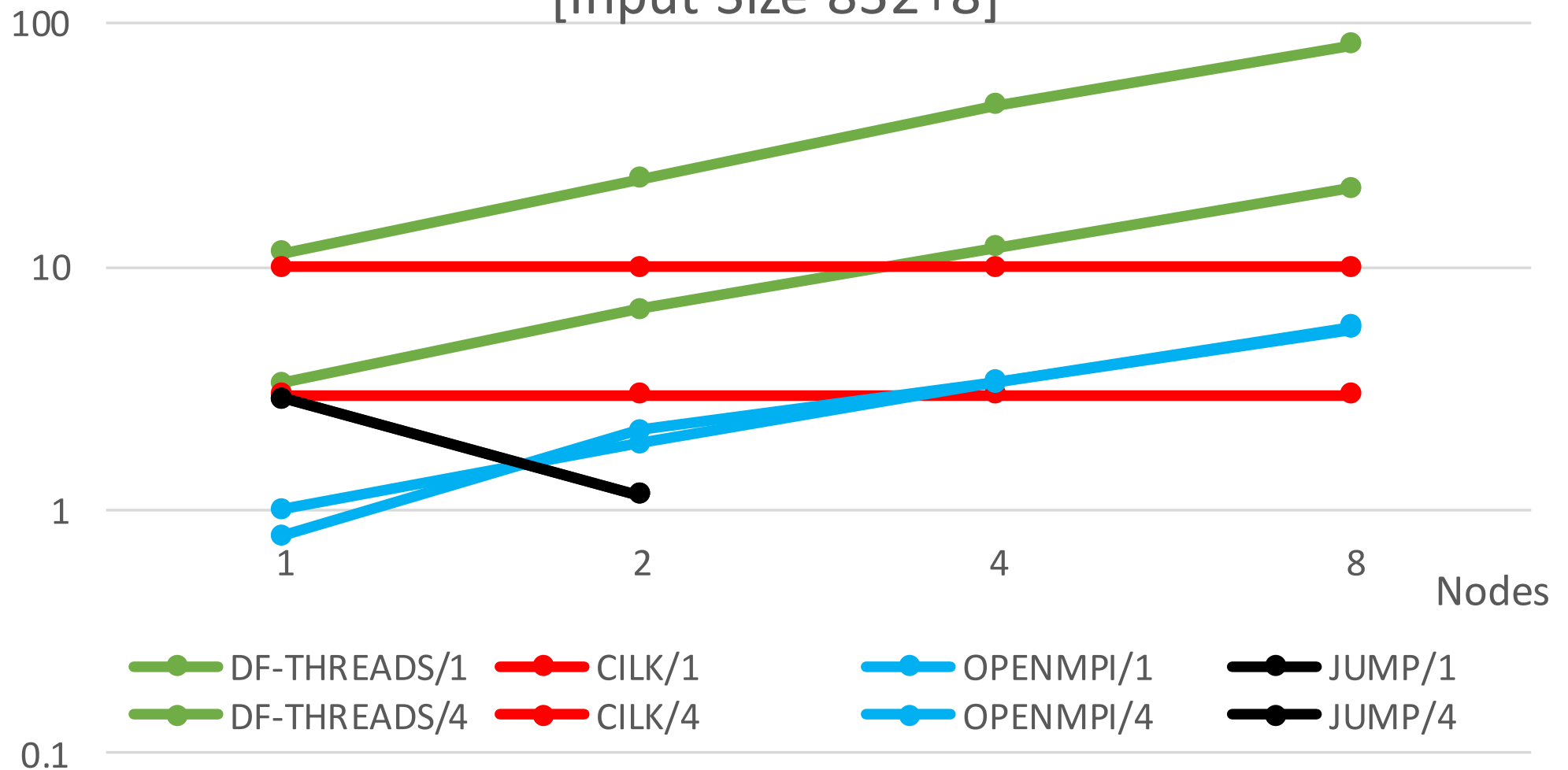
8

Nodes

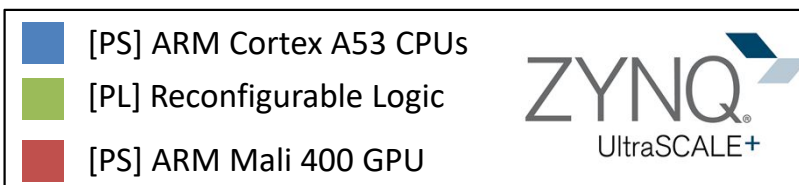
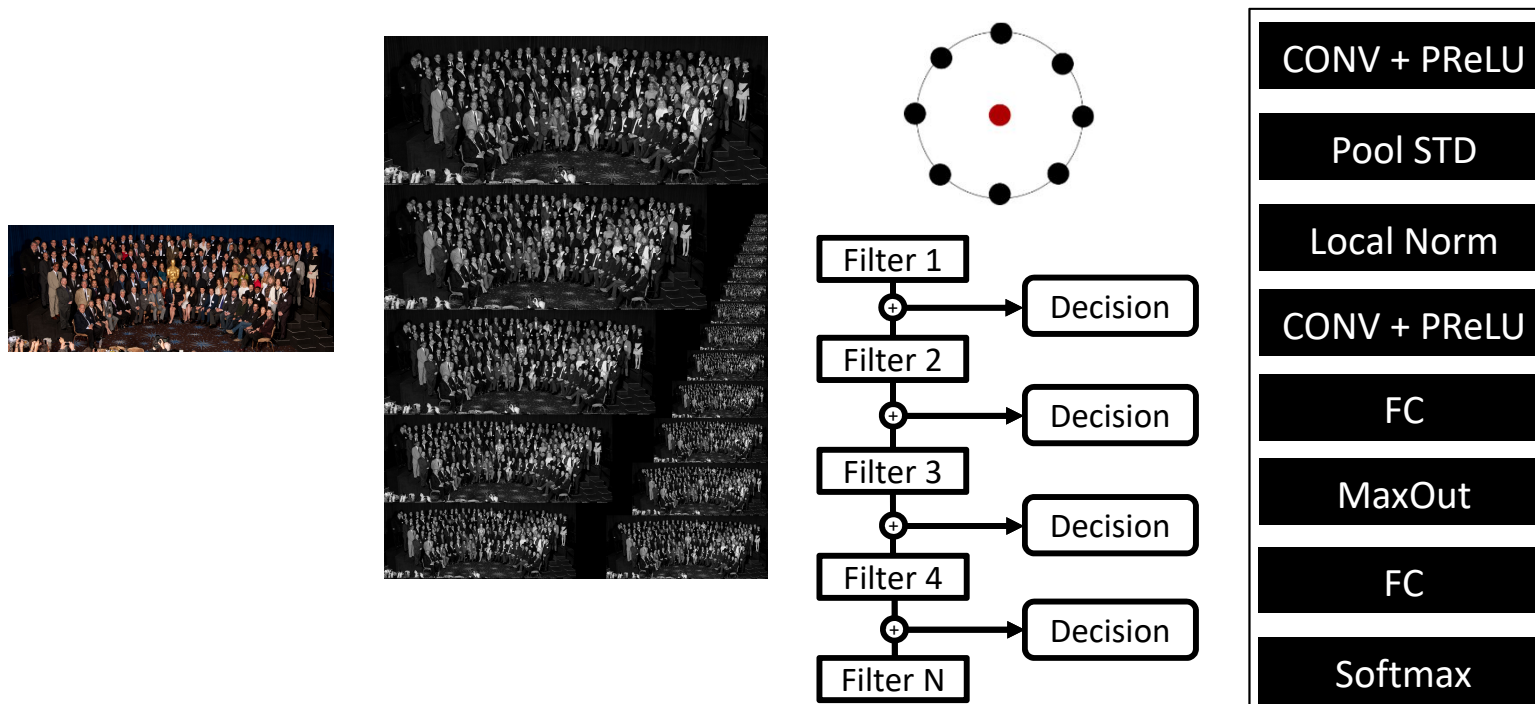
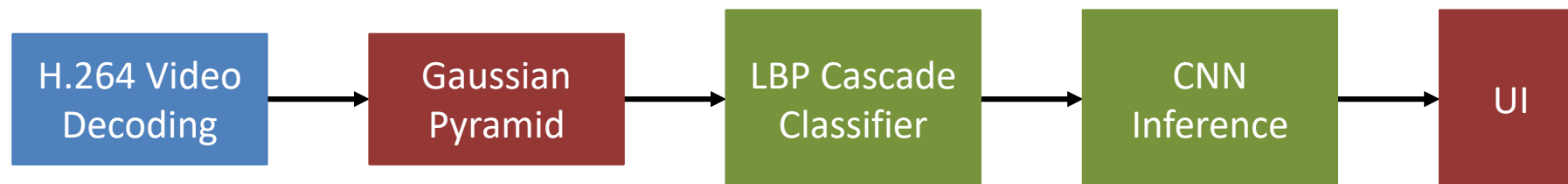


SpeedUp vs. 1node/1core OpenMPI

[Input Size 832+8]

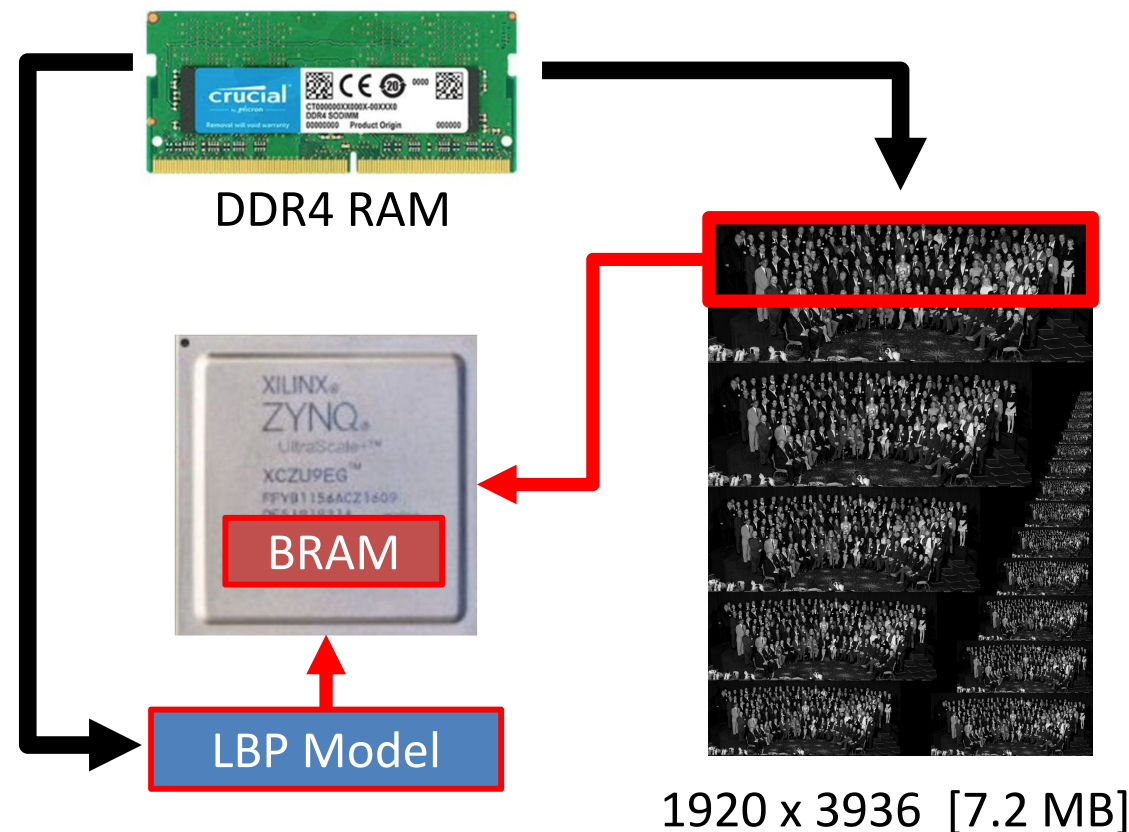


Implementation on the AXIOM platform



LBP Kernel Offloading to the PL

- The lack of a cache memory hierarchy in the PL domain forced us to split the **image pyramid** into chunks, and later fit them in the on-die **BlockRAM** [4 MB].



LBP Kernel Offloading to the PL (II)

- A single LBP accelerator consumed 75% of the ZU9EG BlockRAM, and exploited only 8 FPGA ALUs.
 - Therefore, it was not feasible to synthesize several accelerators in the PL.

- The achieved clock frequency of the generated bitstream was **200 MHz**

- Conclusions:

- The LBP kernel was challenging for the Vivado HLS compiler. The evaluation of the classifiers was input dependent → Very hard to optimize at compile time for HLS.
- Very low ~~FPGA~~ DSP/ALU utilization
- Latency-

	BlockRAM	DSP48E	FF	LUT
Used	1380	8	11599	113376
Total	1824	2520	548160	274080
(%)	75	0.3	2.1	41.3

Demographics CNN Inference

- Herta's proprietary CNN inference engine ported to the AXIOM board architecture (originally implemented in CUDA).
 - SGEMM backend replaceable by BSC's OmpSs@FPGA matrix multiply kernel.
- The CNN models developed during Year 2 were updated with latest ML advances (**batch normalization** layers).

AXIOM Face Dataset

- All face pictures collected by UNISI [14 GB] were released to the public, and are now stored and indexed at **Zenodo** for enriching existing deep learning facial datasets:

<https://doi.org/10.5281/zenodo.1187127>

Publication date:
March 2, 2018

DOI:
DOI 10.5281/zenodo.1187127

Keyword(s):
face recognition facial database
demographics estimation age gender ethnicity

Grants:
European Commission:
• AXIOM - Agile, eXtensible, fast I/O Module for the cyber-physical era (645496)

License (for files):
[Creative Commons Attribution 4.0](#)

Files (14.0 GB)		
Name	Size	
AXIOM-Annotator.zip	33.5 kB	Preview Download
md5:689ffec8bafc5e8f673befcc04c51976 ?		
AXIOM-Face-Database.pdf	663.6 kB	Preview Download
md5:e03fb2f7869aab0c0d5ede6e2b38ee60 ?		
AXIOM-Face-Database.zip	14.0 GB	Preview Download
md5:03bcd115e0b508802b48b2df71958257 ?		



System Integration



VIMAR-ELVOX
Video Door
Entry System

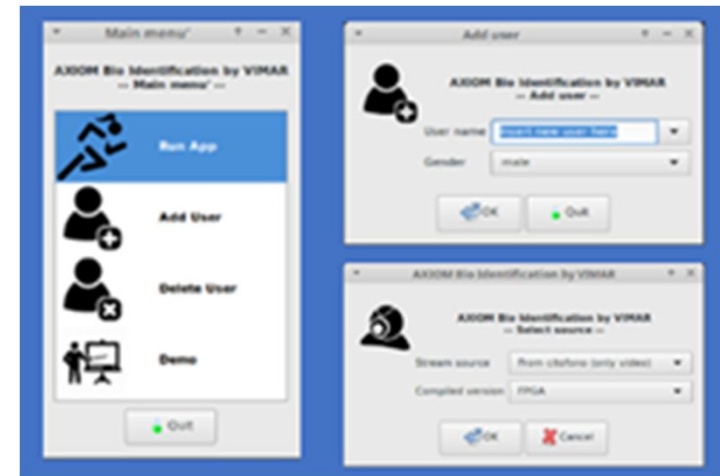
VIMAR
User Interface

AXIOM
Board

**Edge Computing prototype for
Smart Home solution**

Customization of the video door entry system and the AXIOM_Bio_ID Application

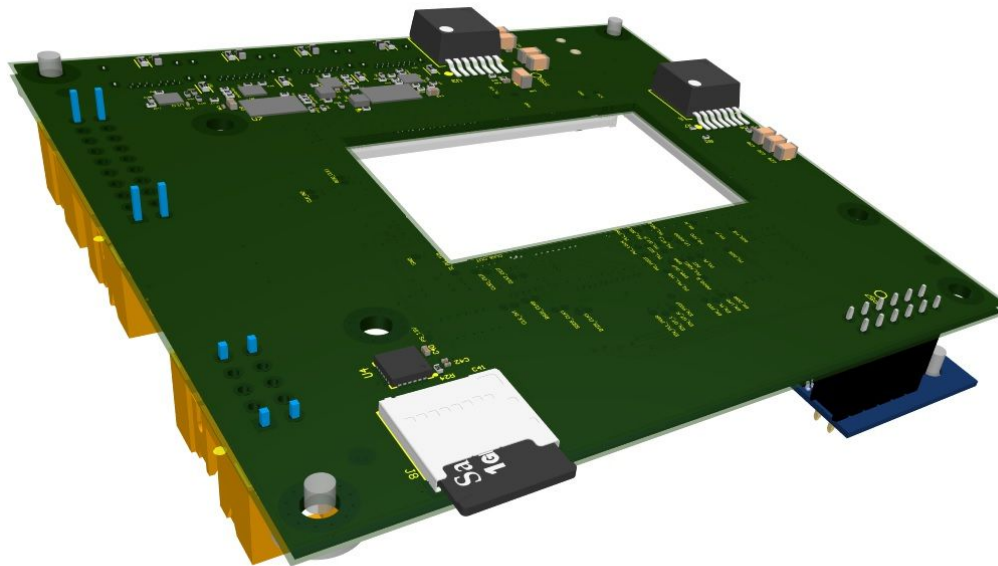
- HW: Define the correct Lens
- HW: Illumination/LEDs
- SW: Input video resolution (720x578 px)
- SW: image rectification
- SW: Graphical User Interface



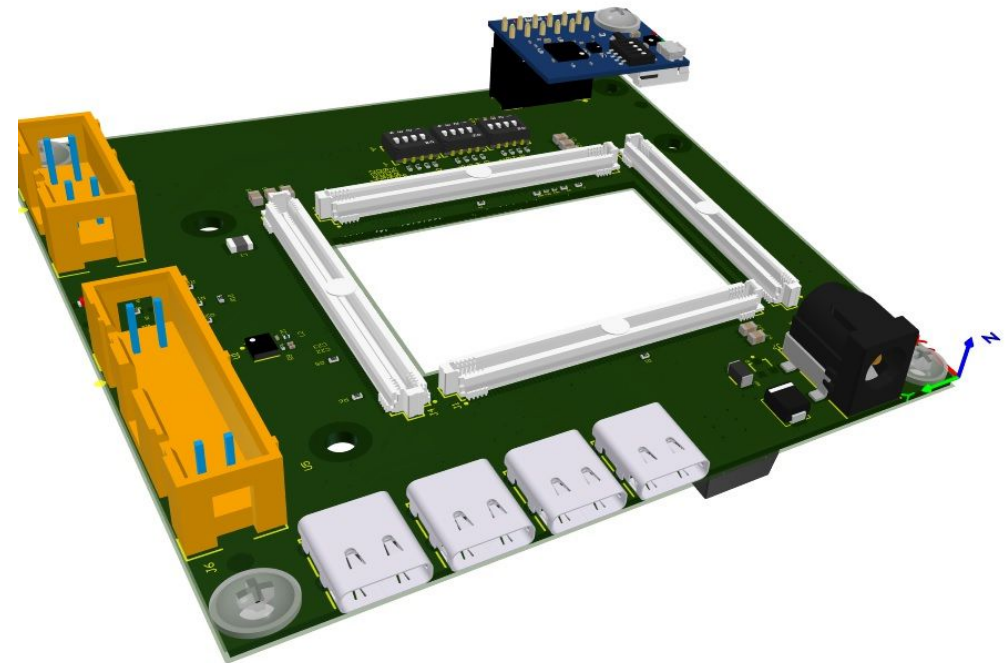
User Interface to test user interaction

The GLUON-board (preview)

Bottom view



Top view



Specifications

PL GT Transceivers

SD Card

JTAG/UART header

10 Pin I2C header for “Silabs” Clock Builder Field Programmer

Done, Error/Status LEDs

PS I/O loopbacks

PL I/O loopbacks

Description

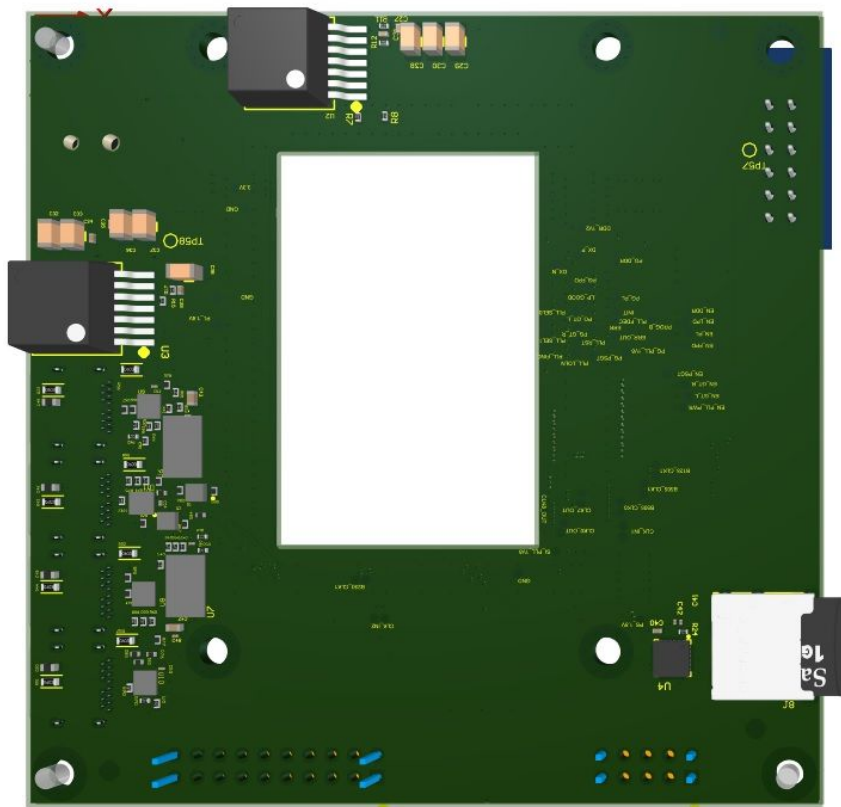
8 Differential Pair in total, 16.3 Gb/s

Boot the Operating System

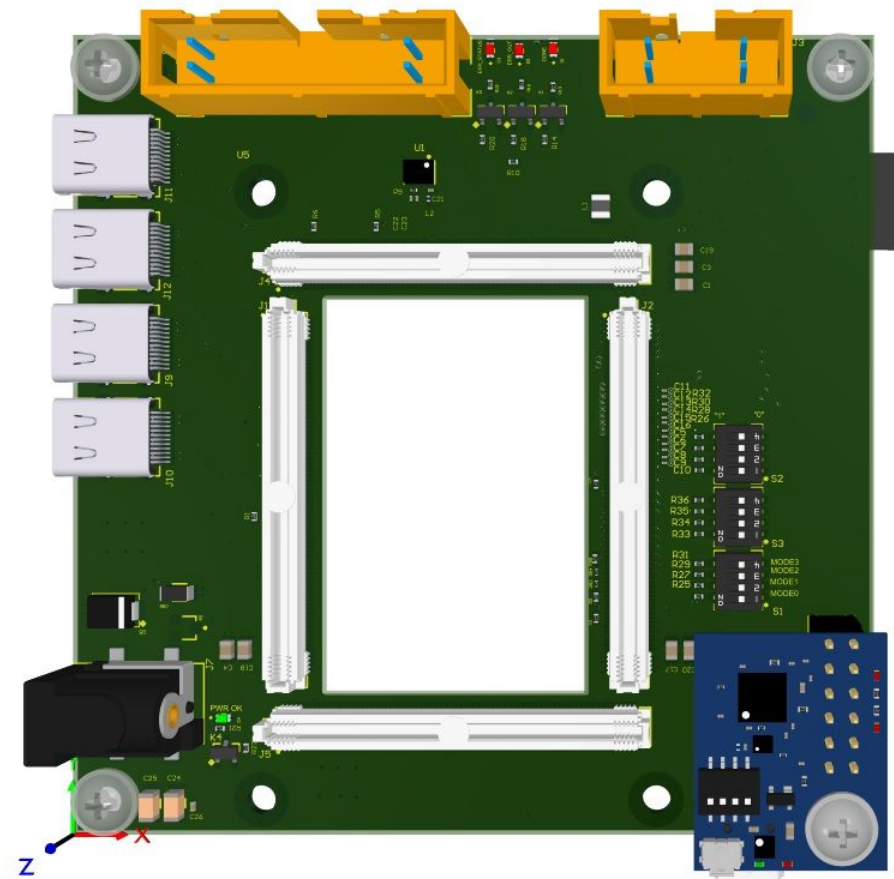
To configure the Si5345 Clock Generator

The GLUON-board (preview)

Bottom view



Top view



GLUONs connect QUARKs

(i.e., TE0808 Trenz-modules based on Zynq Ultrascale+)



The module will go on top
of the GLUON-board

CONCLUSIONS

- This presentation introduced the "AXIOM software stack", which provides the possibility to distribute computations across a cluster of FPGAs by using the OmpSs programming model
- OmpSs allows the programmer to increase the performance by either parallelizing the application on cores and nodes/boards of the cluster or by offloading a computation on an FPGA accelerator
- The AXIOM project explored both the options and the combination of them.
- This open-source software, along the full software stack based on a standard Linux/Ubuntu 16.04 LTS distribution has been demonstrated on the AXIOM-board, which is based on the Zynq ZU9EG Ultrascale+ Soc
- To further simplify the deployment and provide an inexpensive mean for connecting up to 255 boards, the AXIOM-board provides four USB-C connectors, each one capable of up to 18Gbps data transfer rate
- This model permits to address the need of a scalable performance depending on the size of the problem, e.g., in the Smart Home in order to serve a building or an apartment, by using the same hardware and software platform, or in the Smart Videosurveillance
- The new GLUON board will provide a more cost effective solution for implementing AXIOM project concepts



Facebook: <https://www.facebook.com/theaxiomproject?ref=hl>

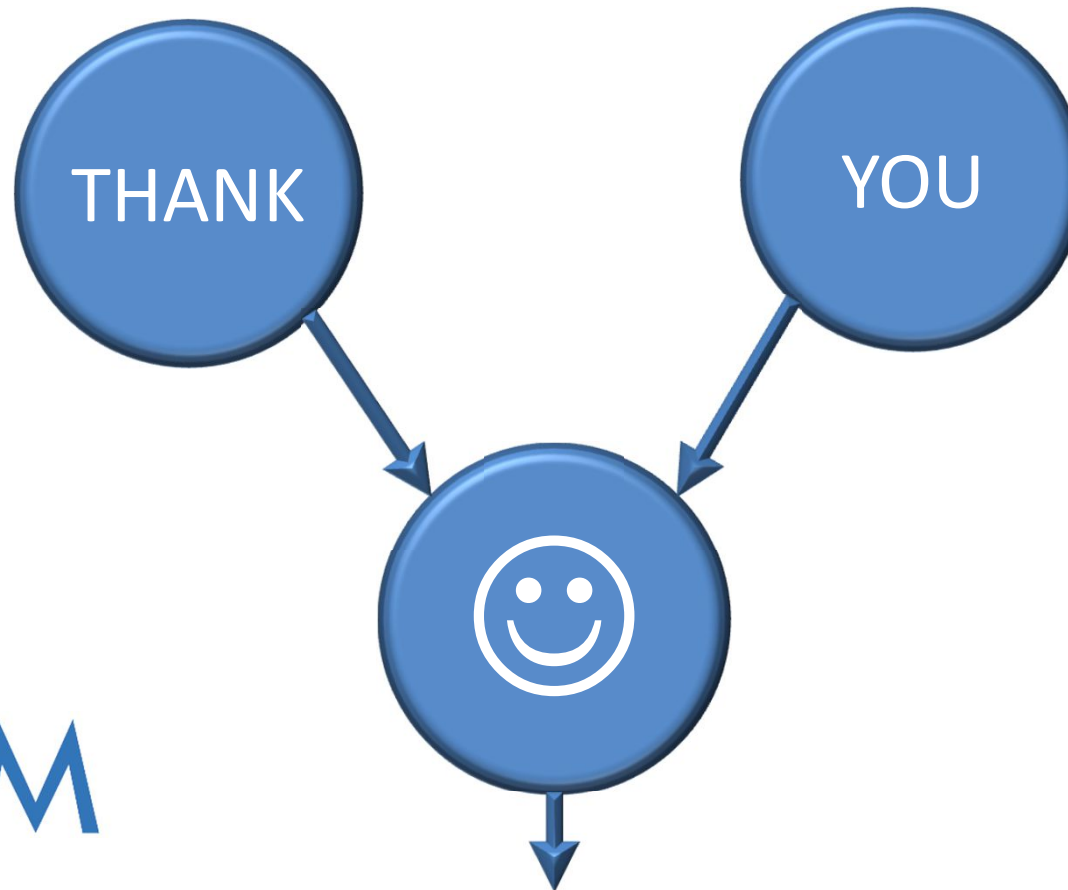
Twitter: https://twitter.com/axiom_project @axiom_project

Google+: google.com/+Axiom-projectEu



LinkedIn: <https://www.linkedin.com/grp/home?gid=8294592>

Youtube Channel: <https://www.youtube.com/channel/UCsQmVv98aPV0vzSjn7y1bMw>



Questions?