18 Jan 2024
PARMA-DITAM: 15th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and 13th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms

# High-Level Synthesis developments in the context of European space technology research

**Fabrizio Ferrandi, Michele Fiorito, Claudio Barone, Giovanni Gozzi , Serena Curzel**

**Politecnico di Milano, Italy,**

## Outline

- **Introduction**
- **Bambu HLS**
- **Bambu extensions**
- **Conclusion**

# Introduction

- **Space missions today face a growing need for on-board computing performance**
  - **Low bandwidth communication links**
  - **High onboard processing capability**
  - **Complex navigation algorithms**
- **We are rapidly approaching the limits of what space-grade radiation-hardened processors and microcontrollers**

hybrid CPU-FPGA systems

## hybrid CPU-FPGA systems in Space

- **Hybrid CPU-FPGA systems offer**
  - **Improved performances**
  - **Acceptable overhead in size**
  - **Good power consumption**
  - **Low costs**
  - **In-flight reconfiguration**

Radiation-hardened FPGAs are thus a key enabling technology for space applications

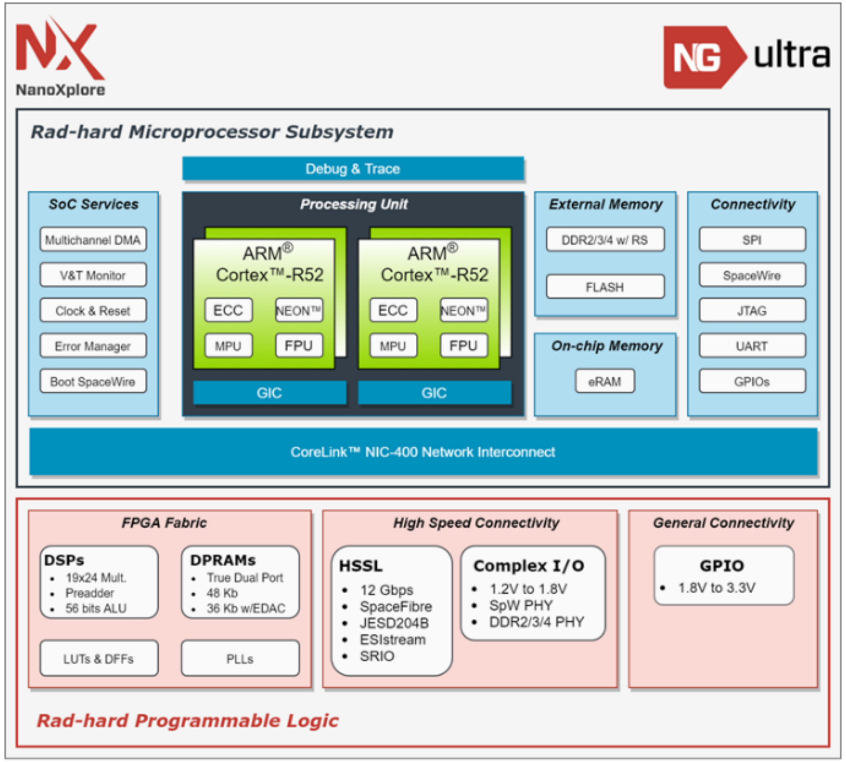
- **In Europe, there are several efforts to develop a new generation of rad-hard FPGAs**
  - **BRAVE, VEGAS, OPERA, DAHLIA, and HERMES**

# HERMES project

- **H2020-funded project started in March 2021**
- **It aims at validating and evaluating a state-of-the-art rad-hard FPGA according to the standards of the European Space Components Coordination (ESCC), and at integrating design and manufacturing technologies needed to deliver high-reliability applications running on radiation-hardened integrated circuits**
- **Consortium is composed by**
  - ***NanoXplore, France,***
  - ***Politecnico di Milano, Italy,***
  - ***Fent Innovative Software Solutions – FentISS, Spain,***
  - ***Thales Alenia Space SAS, France,***
  - ***STMicroelectronics Grenoble SAS, France,***
  - ***Airbus Defence And Space SAS, France***

# HERMES target platform

- **NG-ULTRA architecture**
  - **STM 28nm FD-SOI**
  - **rad-hard SoC**
  - **Quad-core ARM R52 processor running at 600MHz**
  - **Rad-hard eFPGA with 550k LUTs**



## HERMES development tools

- **The project aims at providing a design ecosystem integrating**
  - **NanoXplore synthesis toolkit**
  - **Open-source Bambu High-Level Synthesis tool**
  - **Support for platform virtualization based on XtratuM NextGeneration hypervisor**
  - **Generic Level 1 Boot loader**

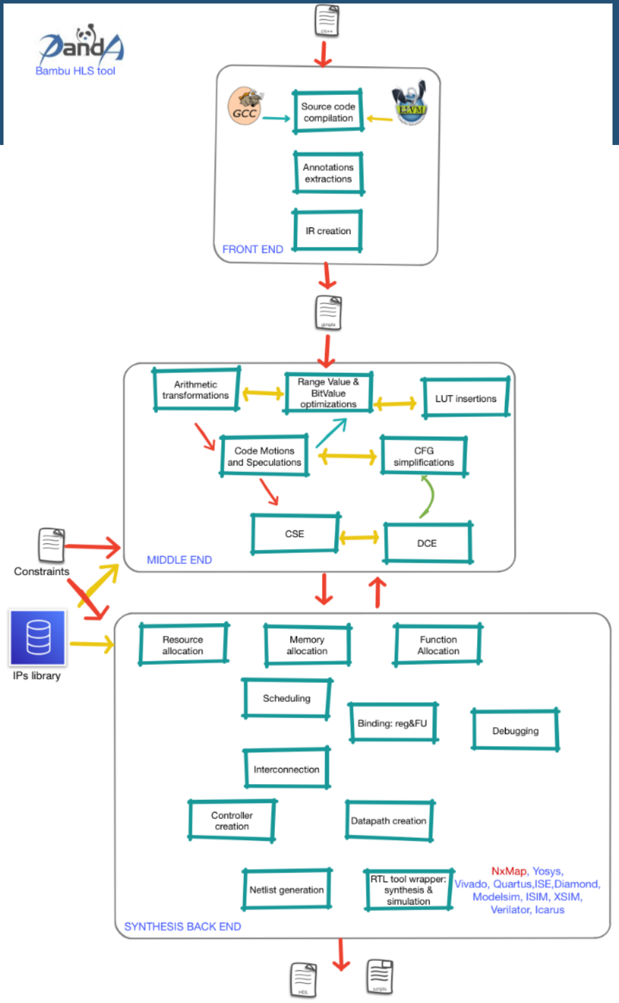
## Use Cases

- HERMES project is validating the Bambu HLS tool through typical space use cases
  - image and vision processing algorithms,
  - software-defined algorithms,
  - artificial intelligence applications

## Bambu HLS

- HLS tools simplify the implementation of accelerators on FPGA
- HLS starts from high-level languages (C/C++)
  - Optimizes the intermediate representations
  - Allocates resources
  - Schedules operations
  - Binds them to the resources
  - And generates RTL descriptions for synthesis tools

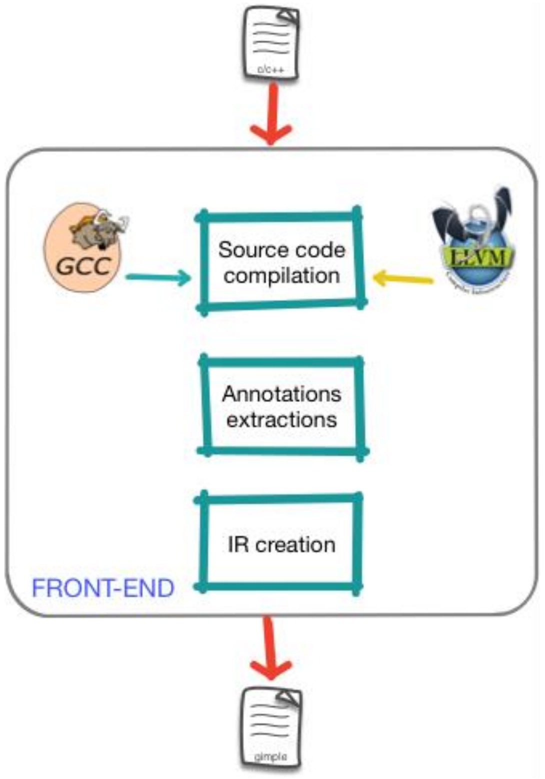
The increased performance offered by FPGAs is made available also to software developers that do not have hardware design expertise



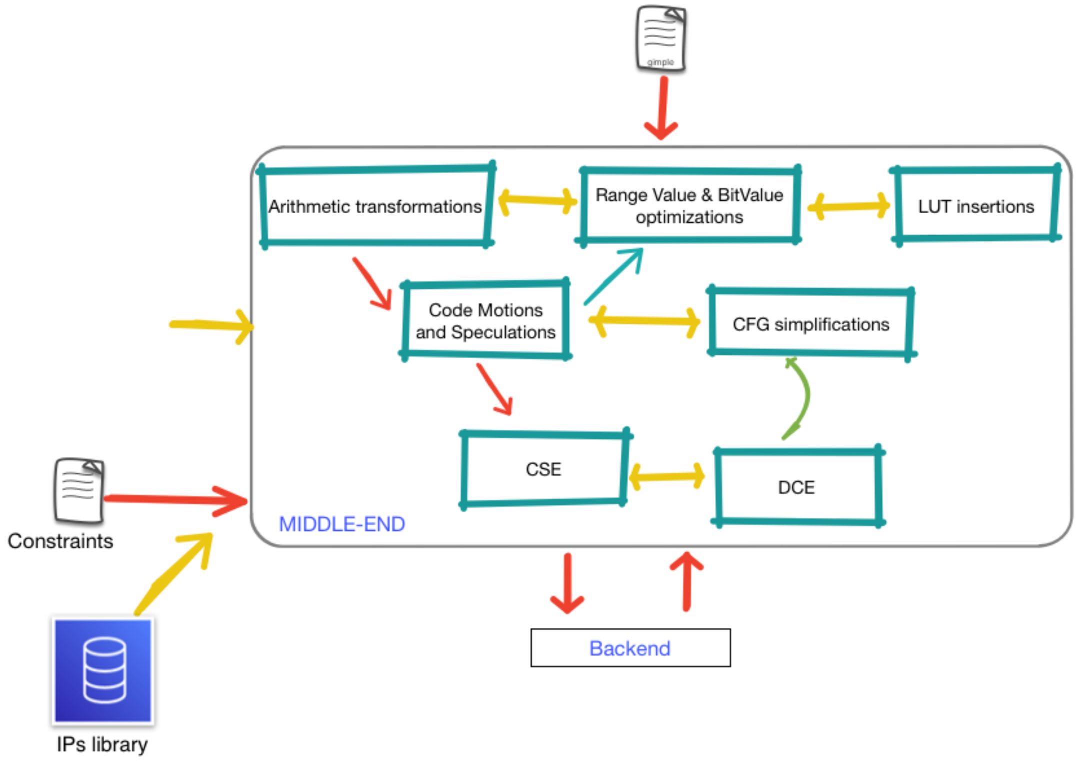
# Bambu: an example of modern HLS tools

- **Open-source HLS tool developed at Politecnico di Milano (Italy)**
  - **Front-end Input: interfacing with GCC/CLANG-LLVM for parsing C code**
    - **Complete support for ANSI C (except for recursion)**
      - **Support for pointers, user-defined data types, built-in C functions, etc..**
    - **Source code optimizations**
      - **may alias analysis, dead-code elimination, hoisting, loop optimizations, etc...**
  - **Target-aware synthesis**
    - **Characterization of the technology library based on target device**
  - **Verification**
    - **Integrated testbench generation and simulation**
      - **automated interaction with Verilator, Xilinx Xsim, Mentor Modelsim**
  - **Back-end: Automated interaction with commercial synthesis tools**
    - **FPGA: Xilinx ISE, Xilinx Vivado, Altera Quartus, Lattice Diamond, NanoXplore**
    - **ASIC: OpenRoad (Nangate 45, ASAP7)**

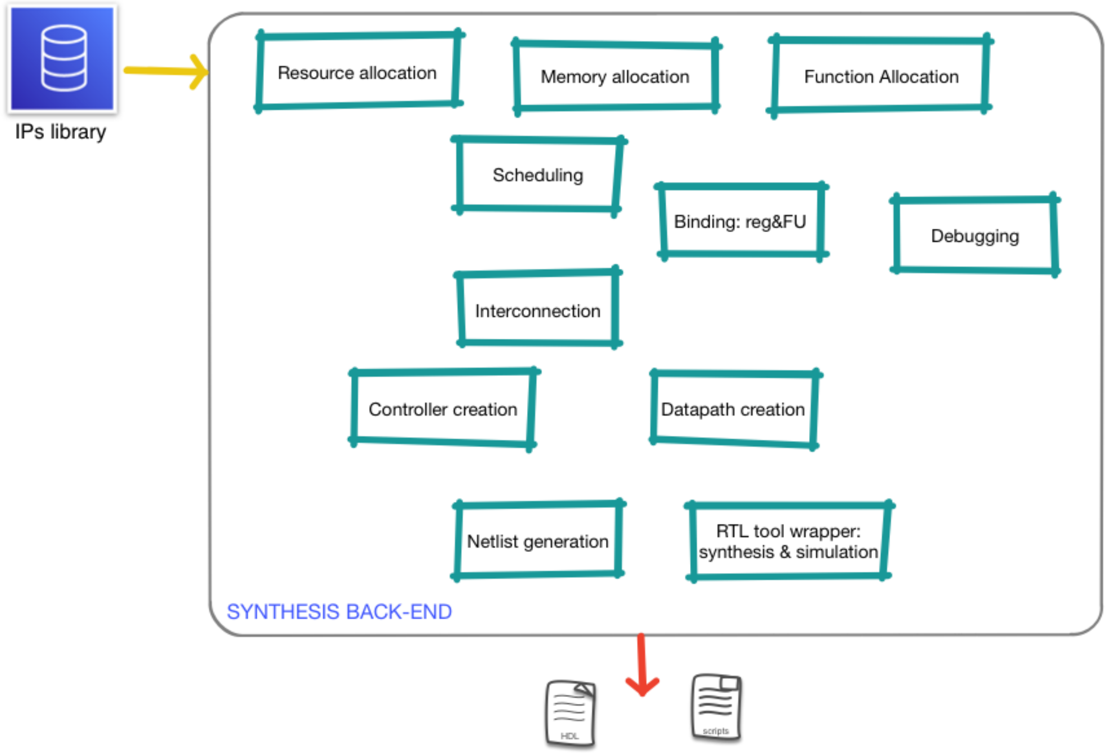
# Bambu: front-end



# Bambu: middle-end


gimple
Arithmetic transformations
Range Value & BitValue optimizations
LUT insertions
Code Motions and Speculations
CFG simplifications
CSE
DCE
MIDDLE-END
Constraints
IPs library
Backend

# Bambu: back-end



## Research opportunities

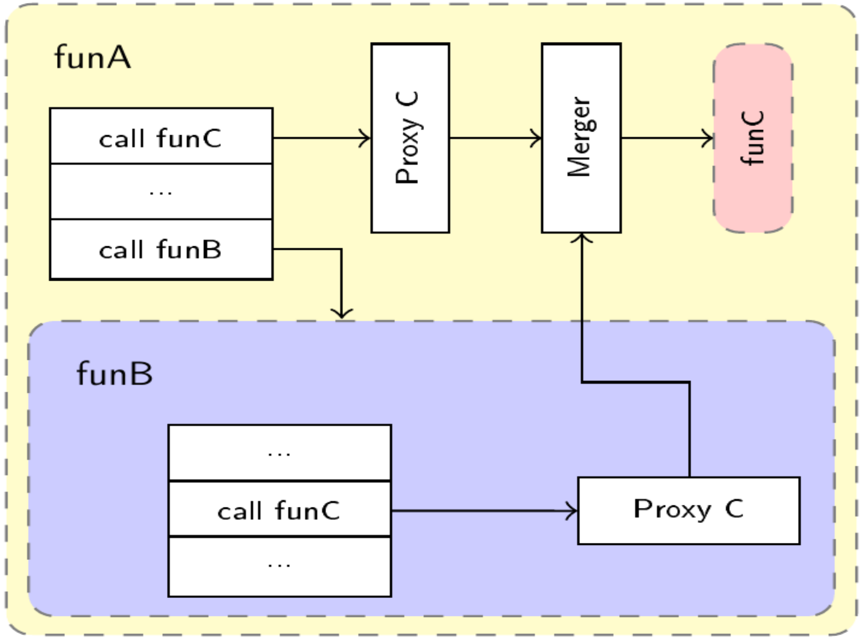
- Bambu is open-source, and it can be used to advance High-Level Synthesis research
- In the years many different algorithms and synthesis techniques have been added
  - Different allocation and binding algorithms

```
--register-allocation=<type>
      WEIGHTED_TS       - solve the weighted clique covering problem by exploiting the
                                Tseng&Siewiorek heuristics (default)
      WEIGHTED_COLORING  - use weighted coloring algorithm
      COLORING          - use simple coloring algorithm
      CHORDAL_COLORING   - use chordal coloring algorithm
      ...

--module-binding=<type>
    Set the algorithm used for module binding. Possible values for the
    <type> argument are one the following:
      WEIGHTED_TS       - solve the weighted clique covering problem by exploiting the
                                Tseng&Siewiorek heuristics (default)
      TTT_FAST         - use Tomita, A. Tanaka, H. Takahashi maxima weighted cliques heuristic
      BIPARTITE_MATCHING - bipartite matching approach
      UNIQUE           - use a 1-to-1 binding algorithm
          ...
```

## Research opportunities

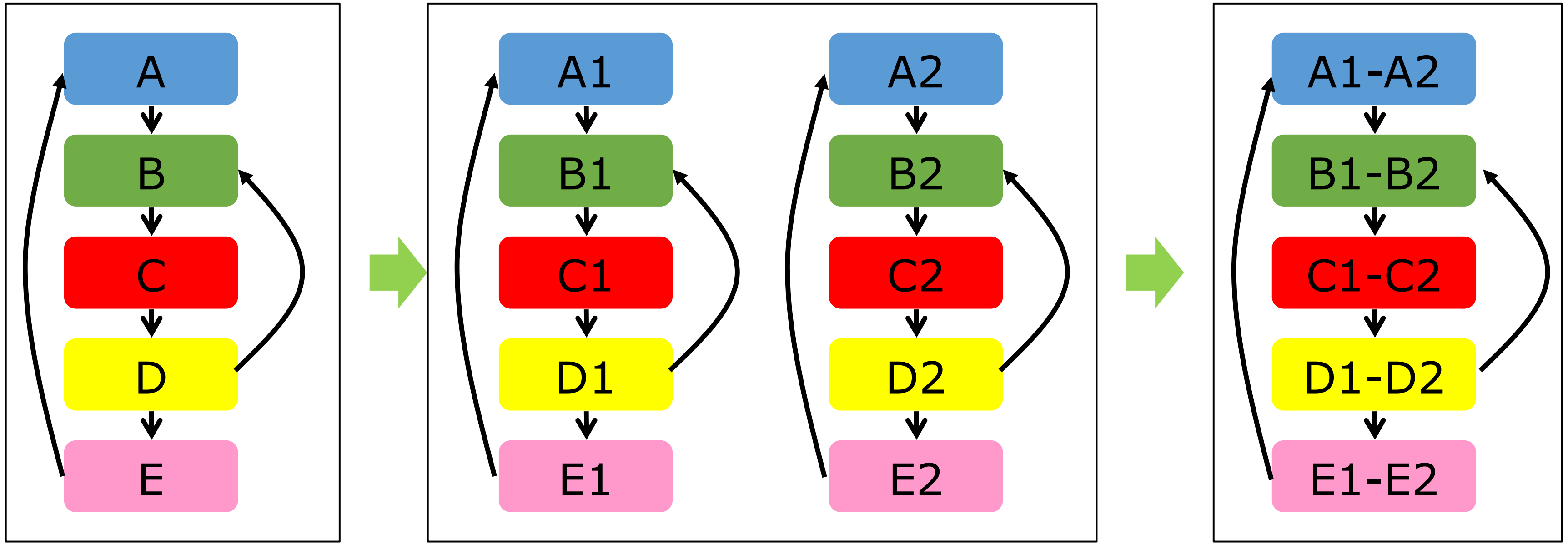
- Synthesis of function proxies



M. Minutoli, V. G. Castellana, A. Tumeo, and F. Ferrandi, “Inter-procedural resource sharing in High Level Synthesis through function proxies,” in Proceedings of the 25th International Conference on Field Programmable Logic and Applications, FPL, 2015, pp. 1-8.

## Research opportunities

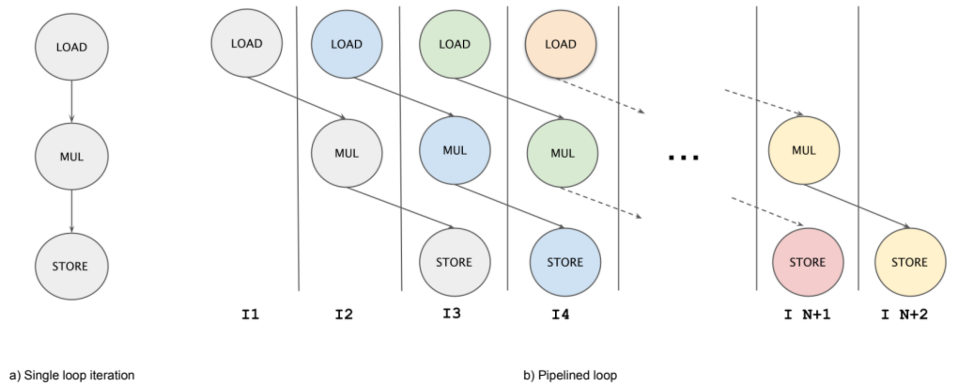
- **Outer loop vectorization**



M. Lattuada and F. Ferrandi, “Exploiting Vectorization in High Level Synthesis of Nested Irregular Loops,” Journal of Systems Architecture, vol. 75, pp. 1-14, 2017.

## Research opportunities

- **MLIR + Bambu loop pipelining**

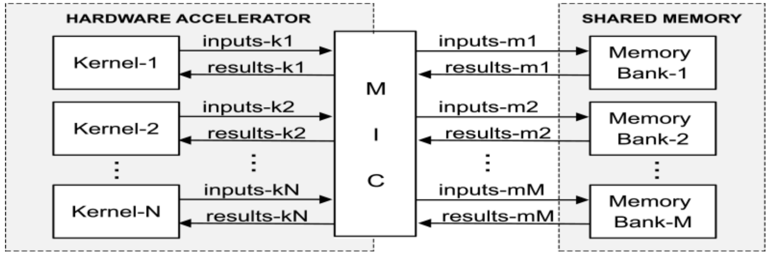


```
#map = affine_map<(d0) -> (d0 - 2)>
func @example(%arg0: memref<1000xi32>,
                         %arg1: memref<1000xi32>) {
   %c0 = arith.constant 0 : index
   %0 = affine.load %arg0[%c0] : memref<1000xi32>
   %c1 = arith.constant 1 : index
   %1 = affine.load %arg0[%c1] : memref<1000xi32>
   %2 = arith.muli %0, %0 : i32
   %3:2 = affine.for %arg2 = 2 to 1000
            iter_args(%arg3 = %1, %arg4 = %2) -> (i32, i32) {
     %5 = affine.load %arg0[%arg2] : memref<1000xi32>
     %6 = arith.muli %arg3, %arg3 : i32
     %7 = affine.apply #map(%arg2)
     affine.store %arg4, %arg1[%7] : memref<1000xi32>
     affine.yield %5, %6 : i32, i32
   }
   %4 = arith.muli %3#0, %3#0 : i32
   %c998 = arith.constant 998 : index
   affine.store %3#1, %arg1[%c998] : memref<1000xi32>
   %c999 = arith.constant 999 : index
   affine.store %4, %arg1[%c999] : memref<1000xi32>
   return
}
```

S.Curzel, S.Jovic, M.Fiorito, A.Tumeo and F.Ferrandi, “Higher-level Synthesis: experimenting with MLIR polyhedral representations for accelerator design” IMPACT workshop 2022.

## Research opportunities

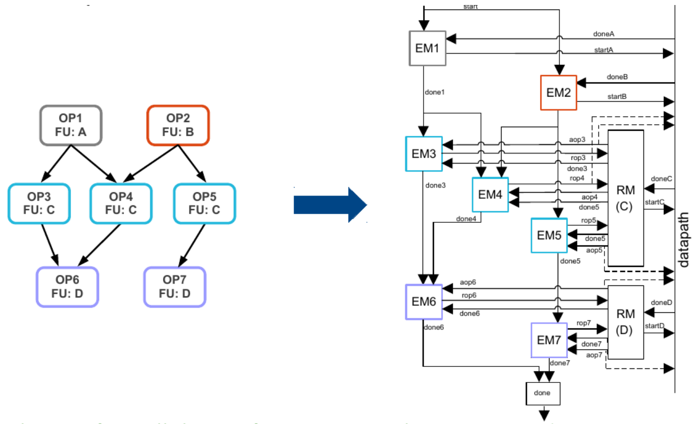
- **Architectural template that exploits both instruction level and task-level parallelism**
- **Dynamic hardware scheduler**
- **Single-cycle hardware context switching**
- **OpenMP support: omp for, omp simd**



- M. Minutoli, V. G. Castellana, N. Saporetti, S. Devecchi, M. Lattuada, P. Fezzardi, A. Tumeo, and F. Ferrandi, “Svelto: High-Level Synthesis of Multi-Threaded Accelerators for Graph Analytics,” *IEEE Transactions on Computers*, vol. 71, iss. 3, pp. 520-533, 2022.
- M. Lattuada and F. Ferrandi, “Exploiting Vectorization in High Level Synthesis of Nested Irregular Loops,” Journal of Systems Architecture, vol. 75, pp. 1-14, 2017.
- V. G. Castellana and F. Ferrandi, “An automated flow for the High-Level Synthesis of coarse-grained parallel applications,” in Proceedings of the International Conference on Field-Programmable Technology (FPT), 2013, pp. 294-301.

# Research opportunities

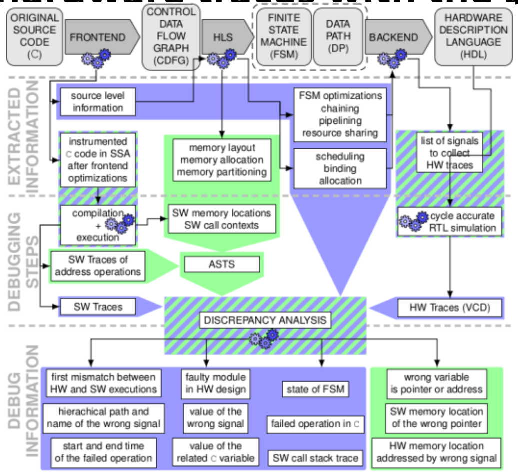
- **FSMD vs adaptive controller**
  - **Imperative vs dataflow oriented**
- **Hierarchical memory controller: parallel accesses**



- V. G. Castellana, A. Tumeo, and F. Ferrandi, “High-Level Synthesis of Parallel Specifications Coupling Static and Dynamic Controllers,” in *2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2021, pp. 192-202.
- V. G. Castellana, A. Tumeo, and F. Ferrandi, “An adaptive Memory Interface Controller for improving bandwidth utilization of hybrid and reconfigurable systems,” in Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014, pp. 1-4.

# Research opportunities

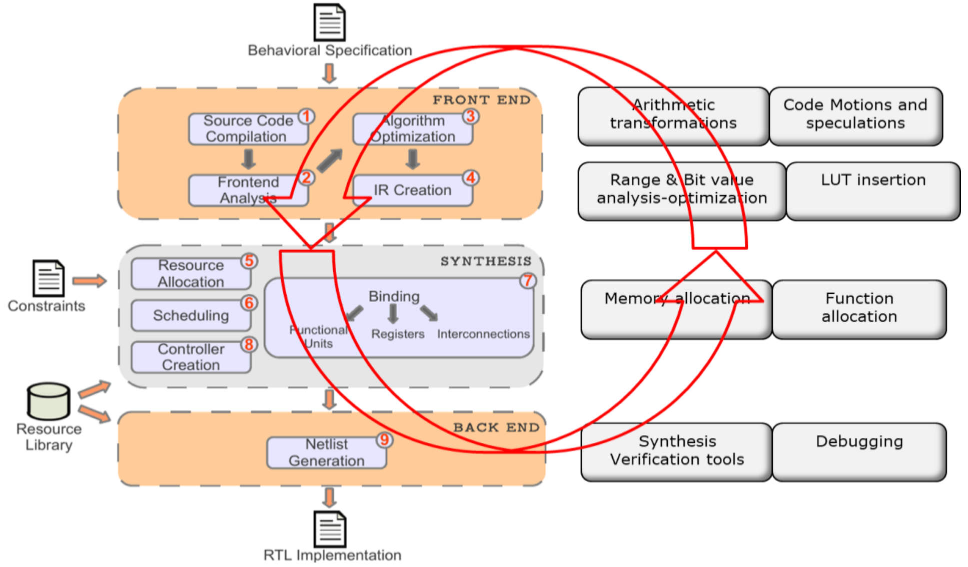
- **automated technique for bug identification: discrepancy analysis**
  - **Comparing HLS-generated hardware traces with the software traces**


ORIGINAL SOURCE CODE (C)
FRONTEND
CONTROL DATA FLOW GRAPH (CDFG)
HLS
FINITE STATE MACHINE (FSM)
DATA PATH (DP)
BACKEND
HARDWARE DESCRIPTION LANGUAGE (HDL)
EXTRACTED INFORMATION
source level information
FSM optimizations chaining pipelining resource sharing
instrumented C code in SSA after frontend optimizations
memory layout memory allocation memory partitioning
scheduling binding allocation
list of signals to collect HW traces
DEBUGGING STEPS
compilation + execution
SW memory locations SW call contexts
cycle accurate RTL simulation
SW Traces of address operations
ASTS
SW Traces
DISCREPANCY ANALYSIS
HW Traces (VCD)
DEBUG INFORMATION
first mismatch between HW and SW executions
faulty module in HW design
state of FSM
wrong variable is pointer or address
hierachical path and name of the wrong signal
value of the wrong signal
failed operation in C
SW memory location of the wrong pointer
start and end time of the failed operation
value of the related C variable
SW call stack trace
HW memory location addressed by wrong signal

- P. Fezzardi and F. Ferrandi, "Automated Bug Detection for High-Level Synthesis of Multi-Threaded Irregular Applications," *ACM Trans. Parallel Comput.*, vol. 7, iss. 4, 2020.
- P. Fezzardi, M. Lattuada, and F. Ferrandi, "Using Efficient Path Profiling to Optimize Memory Consumption of On-Chip Debugging for High-Level Synthesis," *ACM Transactions on Embedded Computing Systems – Special Issue on ESWEEK2017*, vol. 16, iss. 5s, p. 149:1–149:19, 2017.
- P. Fezzardi and F. Ferrandi, "Automated bug detection for pointers and memory accesses in High-Level Synthesis compilers," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, 2016, pp. 1-9.

# Research opportunities

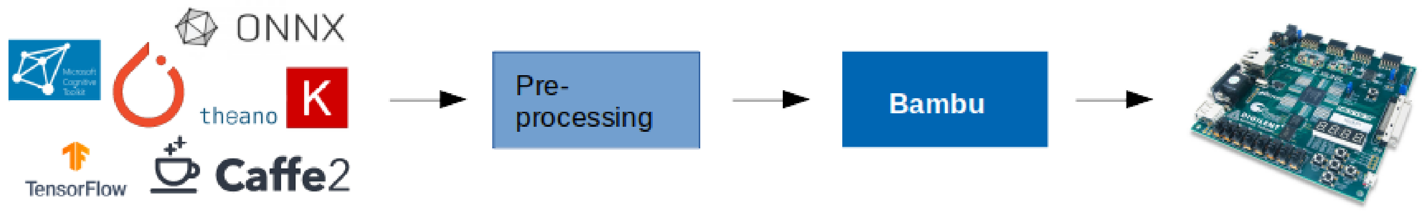
- **Design space exploration**



- M. Lattuada and F. Ferrandi, "A Design Flow Engine for the Support of Customized Dynamic High Level Synthesis Flows," ACM Trans. Reconfigurable Technol. Syst., vol. 12, iss. 4, p. 19:1–19:26, 2019.
- M. Lattuada and F. Ferrandi, "Code Transformations Based on Speculative SDC Scheduling," in *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, 2015, p. 71–77.

# Research opportunities

## Machine Learning accelerators

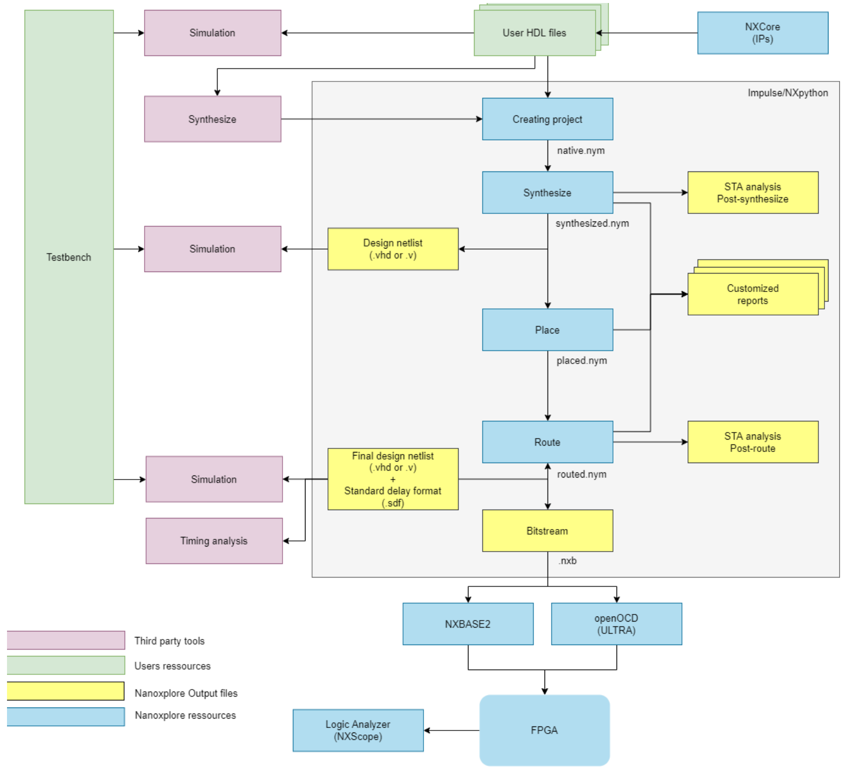


- S. Curzel; N. B. Agostini; V. G. Castellana; M. Minutoli; A. Limaye; J. Manzano; J. J. Zhang; D. Brooks, G. Wei, F. Ferrandi, A. Tumeo. "End-to-End Synthesis of Dynamically Controlled Machine Learning Accelerators", IEEE Transactions on Computers, 2022.
- S. Curzel, S. Jovic, M. Fiorito, A. Tumeo, F. Ferrandi, "Higher-Level Synthesis: experimenting with MLIR polyhedral representations for accelerator design“, Impact 22.
- S. Curzel, N. B. Agostini, S. Song, I. Dagli, A. Limaye, C. Tan, M. Minutoli, V. G. Castellana, V. Amatya, J. Manzano, A. Das, F. Ferrandi, and A. Tumeo, “Automated Generation of Integrated Digital and Spiking Neuromorphic Machine Learning Accelerators,” in *Proceedings of the 40th International Conference on Computer-Aided Design*, New York, NY, USA, 2021.
- M. Siracusa and F. Ferrandi, “Tensor Optimization for High-Level Synthesis Design Flows,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Best Paper Candidate of CODES+ISSS 2020*, vol. 39, iss. 11, pp. 4217-4228, 2020.

## HERMES extensions

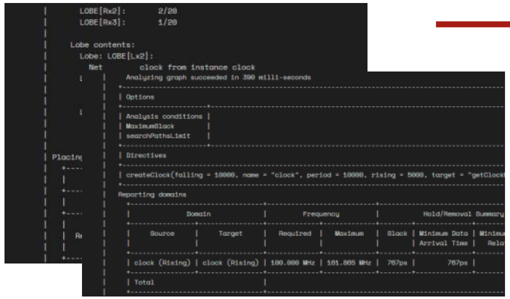
# Impulse design flow

- **In HERMES NanoXplore provides the Impulse design suite**
- **Impulse is an RTL synthesis tool for NX rad-hard FPGAs**
- **It is integrated with third-party tools for simulation and verification**
- **It starts from RTL Verilog/VHDL and goes through the synthesis steps till the bitstream is generated**

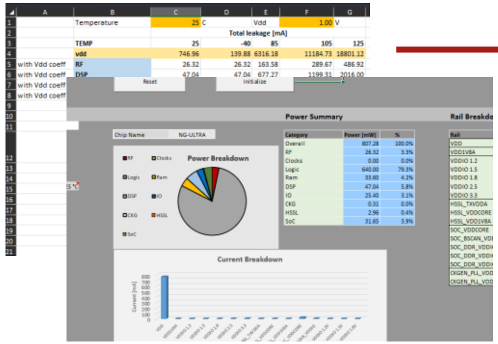


# Bambu – NX integration

- **Seamless integration between Bambu and Impulse from NX**
  - **automatic generation of backend synthesis scripts**
- **Bambu IP resource library characterized with respect to rad-hard NX FPGAs**
  - **Resource occupation and latency under different design constraints**
  - **Automatically performed with PandA Eucalyptus tool**



```
|      LOBE[Rx2]:       2/20
|      LOBE[Rx3]:       1/20
|
|    Lobe contents:
|      Lobe: LOBE[Lx2]:
|        Net        clock from instance clock
|      |   Analyzing graph succeeded in 300 milli-seconds
|      |  +---------------------------------------------------------------------
|      |  | Options
|      |  +---------------------+-----------------------------------------------
|      |  | Analysis conditions |
|      |  | MaximumSlack        |
|      |  | searchPathsLimit    |
|      |  +---------------------+-----------------------------------------------
| Placing |  | Directives
|  +---|  +---------------------------------------------------------------------
|  |   |  | createClock(falling = 10000, name = "clock", period = 10000, rising = 5000, target = "getClock
|  +---|  +---------------------------------------------------------------------
|  |   |  Reporting domains
|  +---|   +-----------------------------+-----------------------+---------------------------
|  |   |   |           Domain            |       Frequency       |      Hold/Removal Summary
|  |   |   +--------------+--------------+-----------+-----------+-------+--------------+-----
|  | Re|   |    Source    |    Target    | Required  |  Maximum  | Slack | Minimum Data | Minimu
|  |   |   |              |              |           |           |       | Arrival Time |  Rela
|  +---|   +--------------+--------------+-----------+-----------+-------+--------------+-----
       |   | clock (Rising) | clock (Rising) | 100.000 MHz | 161.865 MHz | 767ps |    767ps |
       |   +--------------+--------------+-----------+-----------+-------+--------------+-----
       |   | Total                       |
       |   +-----------------------------+-------------------------------------------------------
```



| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | | Temperature | 25 | C | Vdd | 1.00 | V |
| 2 | | | | Total leakage [mA] | | | |
| 3 | | TEMP | 25 | -40 | 85 | 105 | 125 |
| 4 | | vdd | 746.96 | 139.88 | 6316.18 | 11184.73 | 18801.12 |
| 5 | with Vdd coeff | RF | 26.32 | 26.32 | 163.58 | 289.67 | 486.92 |
| 6 | with Vdd coeff | DSP | 47.04 | 47.04 | 677.27 | 1199.31 | 2016.00 |
| 7 | with Vdd coeff | | | | | | |
| 8 | with Vdd coeff | | | | | | |

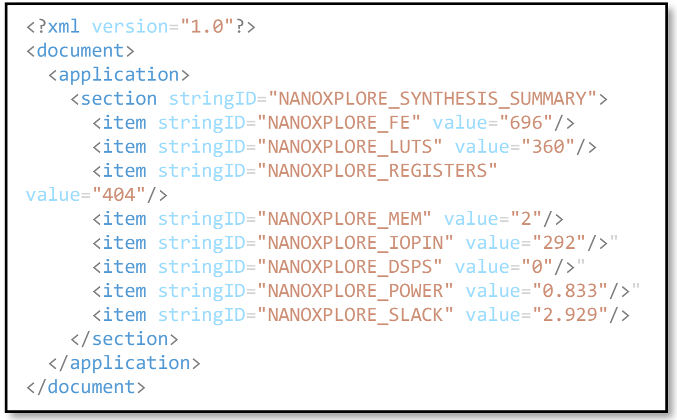
Power Summary

| Chip Name | NG-ULTRA |
|---|---|

| Category | Power [mW] | % |
|---|---|---|
| Overall | 807.28 | 100.0% |
| RF | 26.32 | 3.3% |
| Clocks | 0.00 | 0.0% |
| Logic | 640.00 | 79.3% |
| Ram | 33.60 | 4.2% |
| DSP | 47.04 | 5.8% |
| IO | 25.40 | 3.1% |
| CKG | 0.31 | 0.0% |
| HSSL | 2.96 | 0.4% |
| SoC | 31.65 | 3.9% |

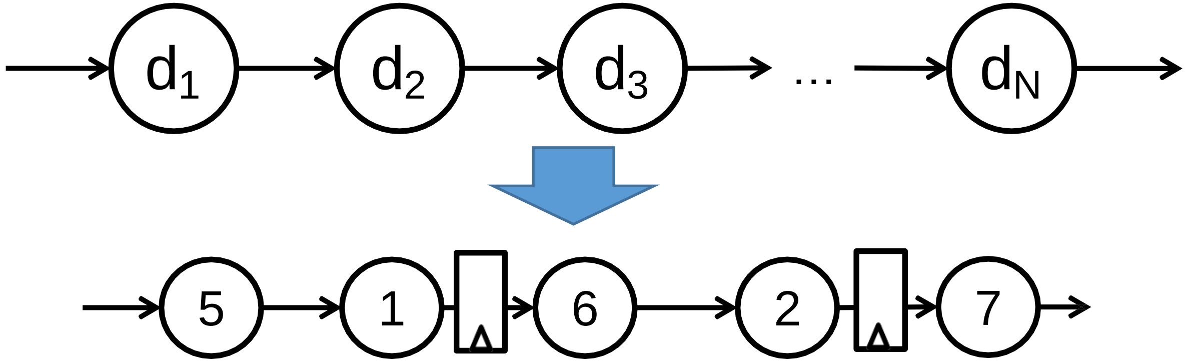
Rail Breakdo

| Rail |
|---|
| VDD |
| VDD1V8A |
| VDDIO 1.2 |
| VDDIO 1.5 |
| VDDIO 1.8 |
| VDDIO 2.5 |
| VDDIO 3.3 |
| HSSL_TXVDDA |
| HSSL_VDDCORE |
| HSSL_VDD1V8A |
| SOC_VDDCORE |
| SOC_BSCAN_VD |
| SOC_DDR_VDD |
| SOC_DDR_VDD |
| SOC_DDR_VDD |
| CKGEN_PLL_VDD |
| CKGEN_PLL_VDD |



```
<?xml version="1.0"?>
<document>
  <application>
    <section stringID="NANOXPLORE_SYNTHESIS_SUMMARY">
      <item stringID="NANOXPLORE_FE" value="696"/>
      <item stringID="NANOXPLORE_LUTS" value="360"/>
      <item stringID="NANOXPLORE_REGISTERS"
value="404"/>
      <item stringID="NANOXPLORE_MEM" value="2"/>
      <item stringID="NANOXPLORE_IOPIN" value="292"/>"
      <item stringID="NANOXPLORE_DSPS" value="0"/>"
      <item stringID="NANOXPLORE_POWER" value="0.833"/>"
      <item stringID="NANOXPLORE_SLACK" value="2.929"/>
    </section>
  </application>
</document>
```

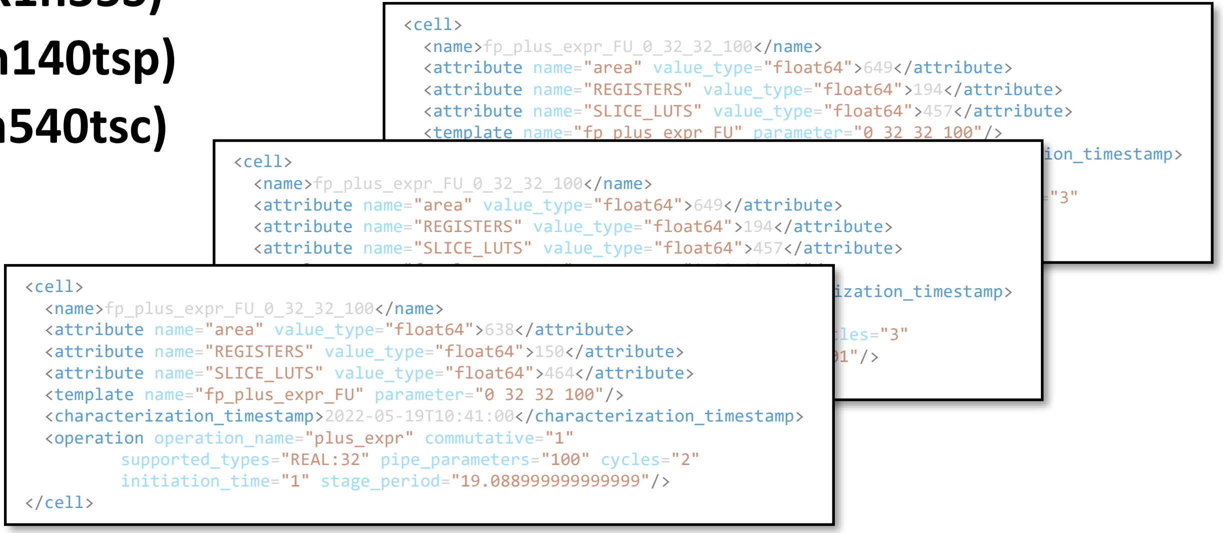
# Bambu – NX integration



- **Performance estimation essential for**
  - **Aggressive scheduling**
  - **Pipelining**
  - **Code transformations**
- **Bambu IP library customized with respect to**
  - **DSPs**
  - **NG-ULTRA fabric True Dual Port RAMs**

# Bambu – NX integration

- **Added support for NanoXplore radiation-hardened FPGAs with device-specific RTL component library characterization for:**
  - **NG-MEDIUM (nx1h35S)**
  - **NG-LARGE (nx1h140tsp)**
  - **NG-ULTRA (nx2h540tsc)**



```
<cell>
  <name>fp_plus_expr_FU_0_32_32_100</name>
  <attribute name="area" value_type="float64">649</attribute>
  <attribute name="REGISTERS" value_type="float64">194</attribute>
  <attribute name="SLICE_LUTS" value_type="float64">457</attribute>
  <template name="fp_plus_expr_FU" parameter="0 32 32 100"/>
                                                                    ion_timestamp>
                                                                    "3"
```

```
<cell>
  <name>fp_plus_expr_FU_0_32_32_100</name>
  <attribute name="area" value_type="float64">649</attribute>
  <attribute name="REGISTERS" value_type="float64">194</attribute>
  <attribute name="SLICE_LUTS" value_type="float64">457</attribute>
                                                                    ization_timestamp>
                                                                    les="3"
                                                                    1"/>
```

```
<cell>
  <name>fp_plus_expr_FU_0_32_32_100</name>
  <attribute name="area" value_type="float64">638</attribute>
  <attribute name="REGISTERS" value_type="float64">150</attribute>
  <attribute name="SLICE_LUTS" value_type="float64">464</attribute>
  <template name="fp_plus_expr_FU" parameter="0 32 32 100"/>
  <characterization_timestamp>2022-05-19T10:41:00</characterization_timestamp>
  <operation operation_name="plus_expr" commutative="1"
         supported_types="REAL:32" pipe_parameters="100" cycles="2"
         initiation_time="1" stage_period="19.088999999999999"/>
</cell>
```

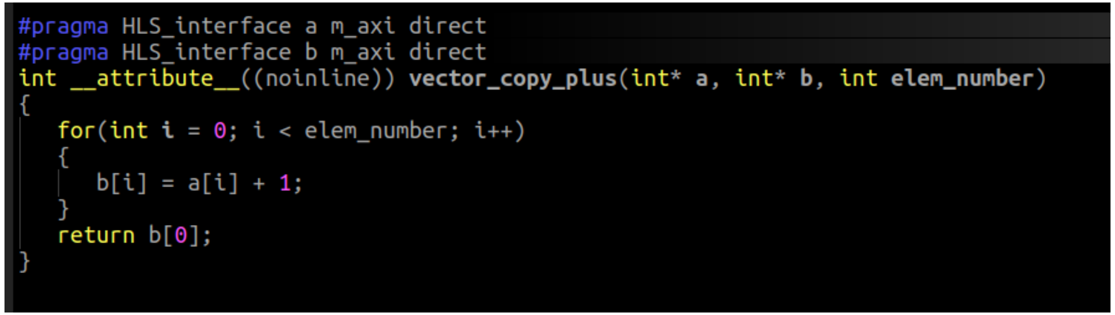
## Bambu – extended features

- **Added support to AXI4 master and AXIS interfaces**
  - **Easier integration with ARM processor on the NG-ULTRA board**
  - **Easy way to integrate existing IPs**

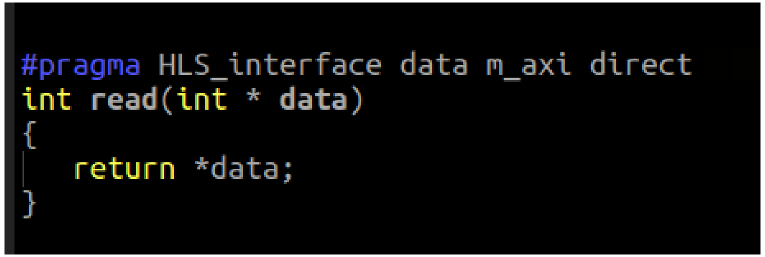
  - **Automatic generation of AXI testbench supported**
  - **Memory latency can be configured**
  - **Unaligned accesses are supported**
  - **AXI4 burst transactions supported**

# Bambu – extended features

| | AXI4 Memory Copy | AXI4 Memory Read |
|---|---|---|
| LUTS | 1072 | 276 |
| Registers | 910 | 142 |
| Frequency | 114 MHz | 189.43 MHz |



```
#pragma HLS_interface a m_axi direct
#pragma HLS_interface b m_axi direct
int __attribute__((noinline)) vector_copy_plus(int* a, int* b, int elem_number)
{
   for(int i = 0; i < elem_number; i++)
   {
      b[i] = a[i] + 1;
   }
   return b[0];
}
```



```
#pragma HLS_interface data m_axi direct
int read(int * data)
{
   return *data;
}
```

# Bambu – extended features

- **Support for caches on AXI interfaces**
  - **Customizable cache and line size**
  - **Support for different write policies**
  - **Support for associative caches**
  - **Support for different replacement policies**
  - **Support for larger AXI xDATA signal size**
  - **Support for pipelined write transactions**
  - **Automatic cache flush at the end of the computation**
  - **Includes simulation-only hit/miss counters**

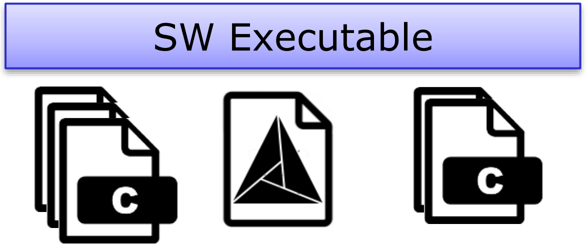
## Bambu – extended features

- **C++ FIFO interface support:**
  - ac_channel<T>
  - hls::stream<T>

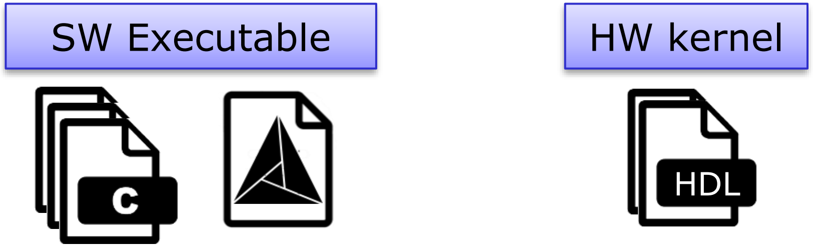
```
void sum3numbers(ac_channel<ap_uint<64>>& a,
                 ac_channel<ap_uint<64>>& b,
                 ac_channel<ap_uint<64>>& c,
                 ac_channel<ap_uint<64>>& d)
{
   int i;
   for(i = 0; i < 8; ++i)
      d.write(a.read() + b.read() + c.read());
}
```

# Standard verification flow

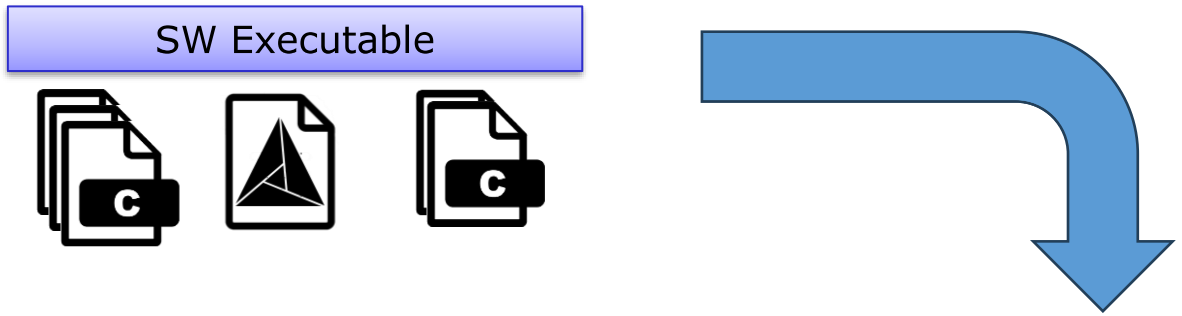
- **Starting point**



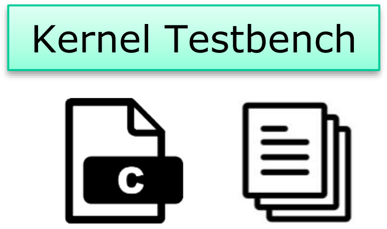
- **Expected result**
  - **Verified software application with verified hardware-accelerated kernel**



# Standard Verification flow

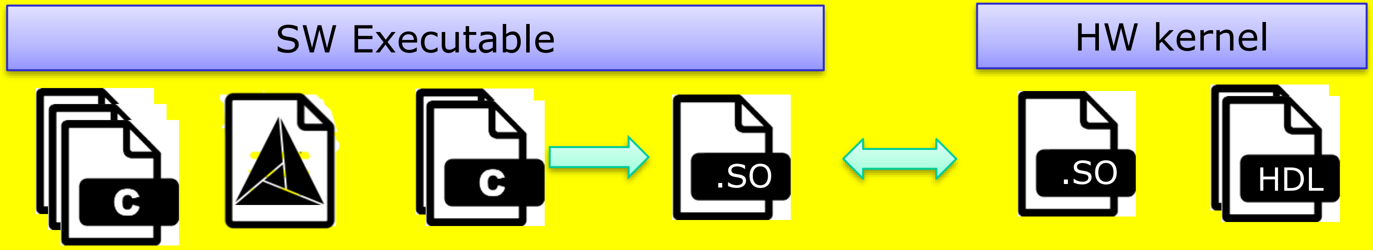


- Testbench built as unit testing module
- Kernel I/O test file generation
- Error-prone
- Separated from use case application



# Bambu verification approach

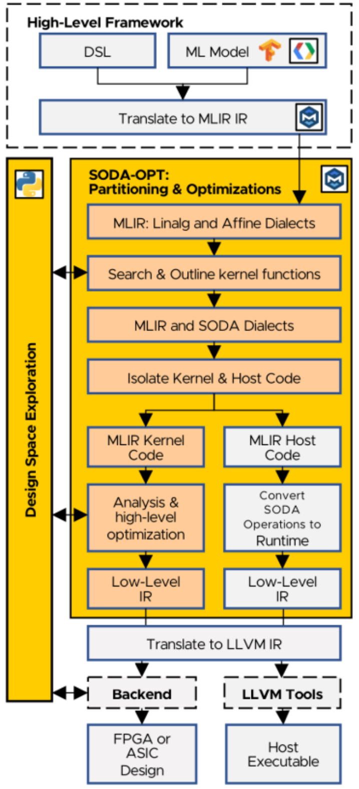
1. **Standard verification flow is now supported even by Bambu**
2. **Bambu additional verification approach**



- **Application instrumented to compare software and HW execution using the original application**
- **DPI-C interface used to connect the SW and the HW worlds**

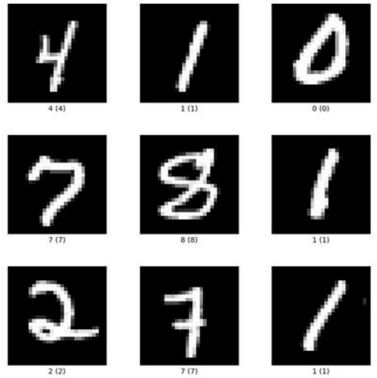
# Integrating ML-design flow in Bambu

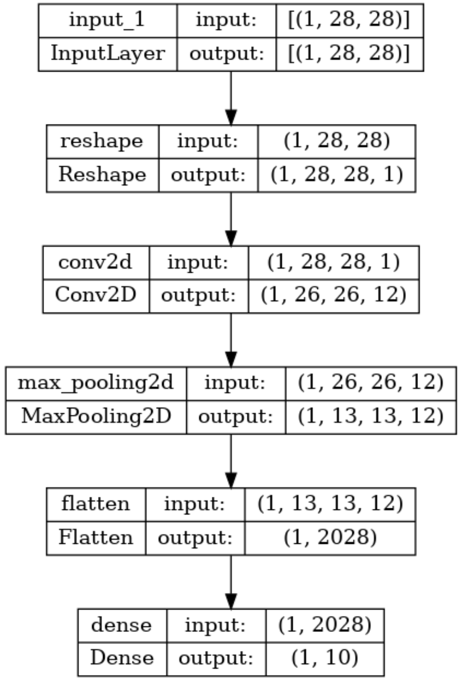
- **Research ongoing on DSLs to synthesize accelerators for machine learning-based inference.**
- **We did some experiments with SODA-OPT framework**
- **SODA-OPT jointly developed by PNNL, Northwestern University and Politecnico di Milano**



Serena Curzel, Nicolas Bohm Agostini, Vito Giovanni Castellana, Marco Minutoli, Ankur Limaye, Joseph B. Manzano, Jeff Zhang, David Brooks, Gu-Yeon Wei, Fabrizio Ferrandi, Antonino Tumeo: End-to-End Synthesis of Dynamically Controlled Machine Learning Accelerators. IEEE Trans. Computers 71(12): 3074-3087 (2022)

# MNIST example

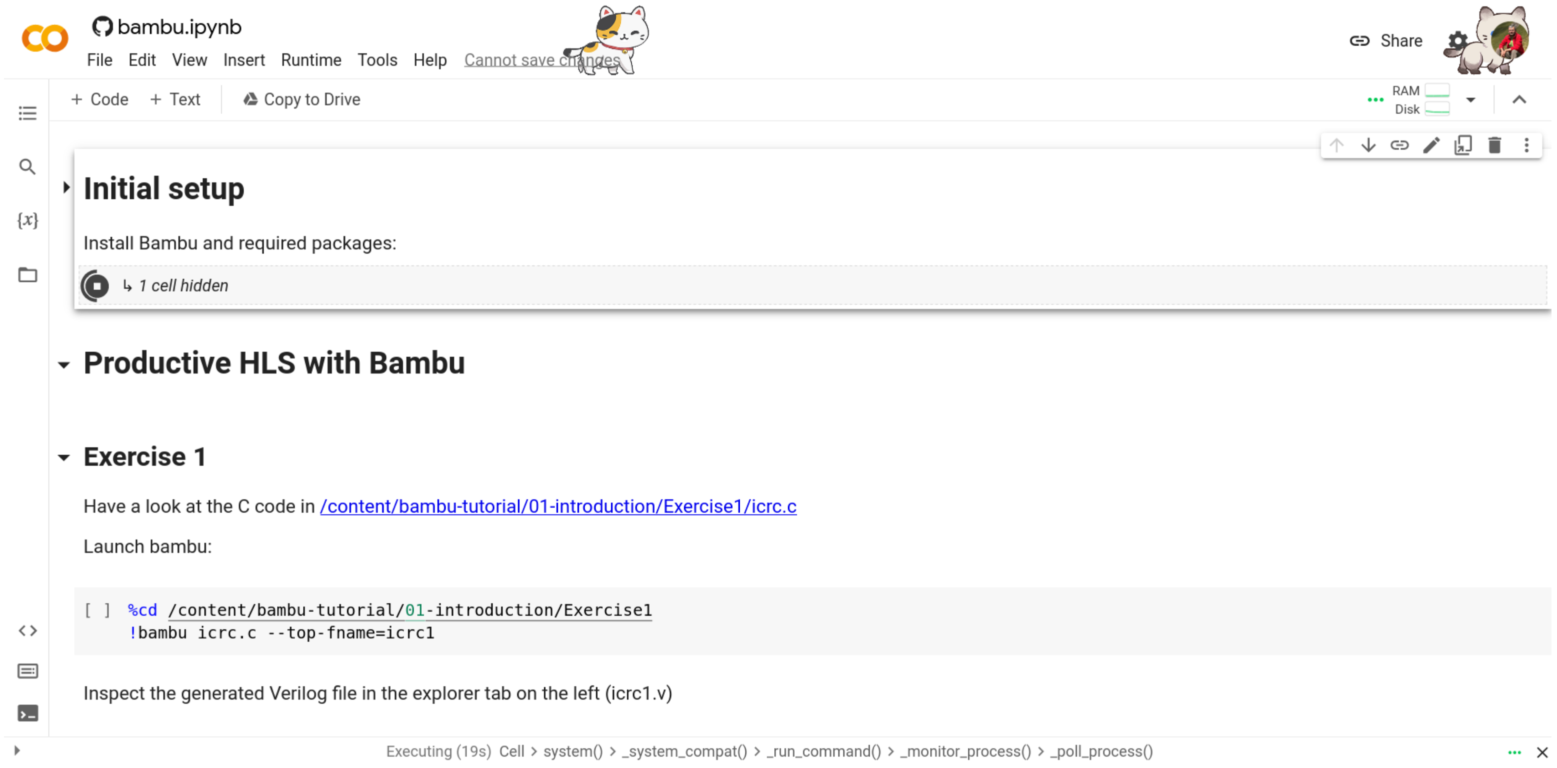



input_1 | input: | [(1, 28, 28)]
InputLayer | output: | [(1, 28, 28)]
reshape | input: | (1, 28, 28)
Reshape | output: | (1, 28, 28, 1)
conv2d | input: | (1, 28, 28, 1)
Conv2D | output: | (1, 26, 26, 12)
max_pooling2d | input: | (1, 26, 26, 12)
MaxPooling2D | output: | (1, 13, 13, 12)
flatten | input: | (1, 13, 13, 12)
Flatten | output: | (1, 2028)
dense | input: | (1, 2028)
Dense | output: | (1, 10)

| | NG-Ultra embedded |
|---|---|
| LUTS | 4627 |
| Registers | 5714 |
| Frequency | 45.7 MHz |
| DSP | 54 |
| MEM | 34 |
| cycles | 169,649 |

- **8-bit quantized**
- **Tensorflow 2.15**
- **Clang 18**
- **MLIR based design flow**

# Google Colab notebook



bambu.ipynb
File Edit View Insert Runtime Tools Help Cannot save changes
Share
+ Code + Text Copy to Drive
RAM
Disk

## Initial setup

Install Bambu and required packages:

↳ *1 cell hidden*

## Productive HLS with Bambu

### Exercise 1

Have a look at the C code in /content/bambu-tutorial/01-introduction/Exercise1/icrc.c

Launch bambu:

```
[ ] %cd /content/bambu-tutorial/01-introduction/Exercise1
    !bambu icrc.c --top-fname=icrc1
```

Inspect the generated Verilog file in the explorer tab on the left (icrc1.v)

```
Executing (19s) Cell > system() > _system_compat() > _run_command() > _monitor_process() > _poll_process()
```

## Conclusion

- **FPGAs are very versatile and suitable for many markets**
- **Integrating HLS will improve productivity**
- **Raise the level of abstraction to develop rad-hard FPGA-based applications**
- **Raise the Technology Readiness Levels (TRL) of the Bambu HLS tool**

# Questions

HERMES

HERMES PROJECT – H2020

Qualification of High-pErformance pRogrammable Microprocessor and dEvelopment of Software ecosystem

https://www.hermes-h2020project.eu/

https://panda.dei.polimi.it

https://github.com/ferrandi/PandA-bambu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 101004203