POLITECNICO DI MILANO

The 32nd International Conference on Parallel Architectures and Compilation Techniques

Viena, Austria, October 22nd, 2023

SODA Synthesizer
Accelerating Data Science Applications with an end-to-end Silicon Compiler

Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications

Fabrizio Ferrandi (Politecnico di Milano)

A bit of history about PandA

- PandA framework development started on 2004 as a support research infrastructure for PoliMi in the context of ICODES – FP6-IST EU-funded project
  - Parsing and analysis of TLM 2.0 SystemC descriptions (gcc v.3.5)
- In the hArtes EU-funded project (2006-2010), it was used to
  - Analyzing generic C-based application annotated with pragmas (OpenMP)
  - Extracting parallel tasks
  - Estimating performance of embedded app
  - C-to-C rewriting
- Later, in Synaptic (2009-2013) and in Faster (2011-2014) EU-funded projects, logic- and high-level synthesis has been extended
  - Bambu (HLS tool) was first released in March 2012.
- In these days, the project received further funding to extend its capability and support new targets

Funding

- The following support is gratefully acknowledged:
  - Xilinx through the donation of two Nexys4 boards and for full licenses of Vivado Design Suite.
  - Altera through the donation of many DE1 CycloneII boards, one DE1-SOC, and a full Quartus software license.
  - NanoXplore through the donation of many licenses of NanoXplore software.
- European Union for funding some of this work through this list of projects:
  - ICODES – Interface and Communication based Design of Embedded Systems
  - hArtes – Holistic Approach to Reconfigurable Real-Time Embedded Systems
  - Synaptic – SYNthesis using Advanced Process Technology Integrated in regular Cells, IPs, architectures, and design platforms
  - Faster – Facilitating Analysis and Synthesis Technologies for Effective Reconfiguration
- European Space Agency for funding some of this work through this list of contracts:
  - ESA/ESTEC/Contract N. 4000100797 – Development of methodologies and tools for predictable, real-time LEON-DSP-based embedded systems.
  - ESA/ESTEC/Contract No. 22167/09/NL/JK. Cache Optimization for LEON Analysis (COLA).
  - ESA/ESTEC/Contract Call-Off Order 4 "Multicore and Schedulability Analysis" for TASTE project.
  - ESA/ESTEC/Contract No. 4000121154/17/NL/LF Compact Reconfigurable Avionics Model-Based Avionic Design (CORA-MBAD)

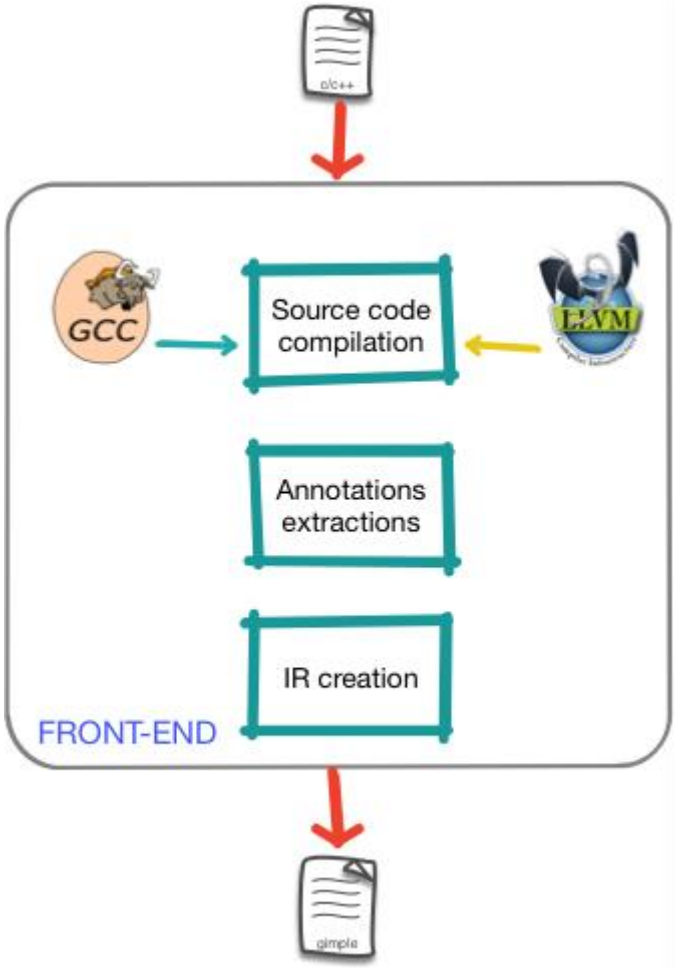
- Latest funding:
  - HERMES – qualification of High pErformance pRogrammable Microprocessor and dEvelopment of Software ecosystem

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

Bambu: an example of modern HLS tools

- HLS tool developed at Politecnico di Milano (Italy) within the PandA framework
  - Available under GPL v3 at
    - http://panda.deib.polimi.it/
    - https://github.com/ferrandi/PandA-bambu
- Example features
  - 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)

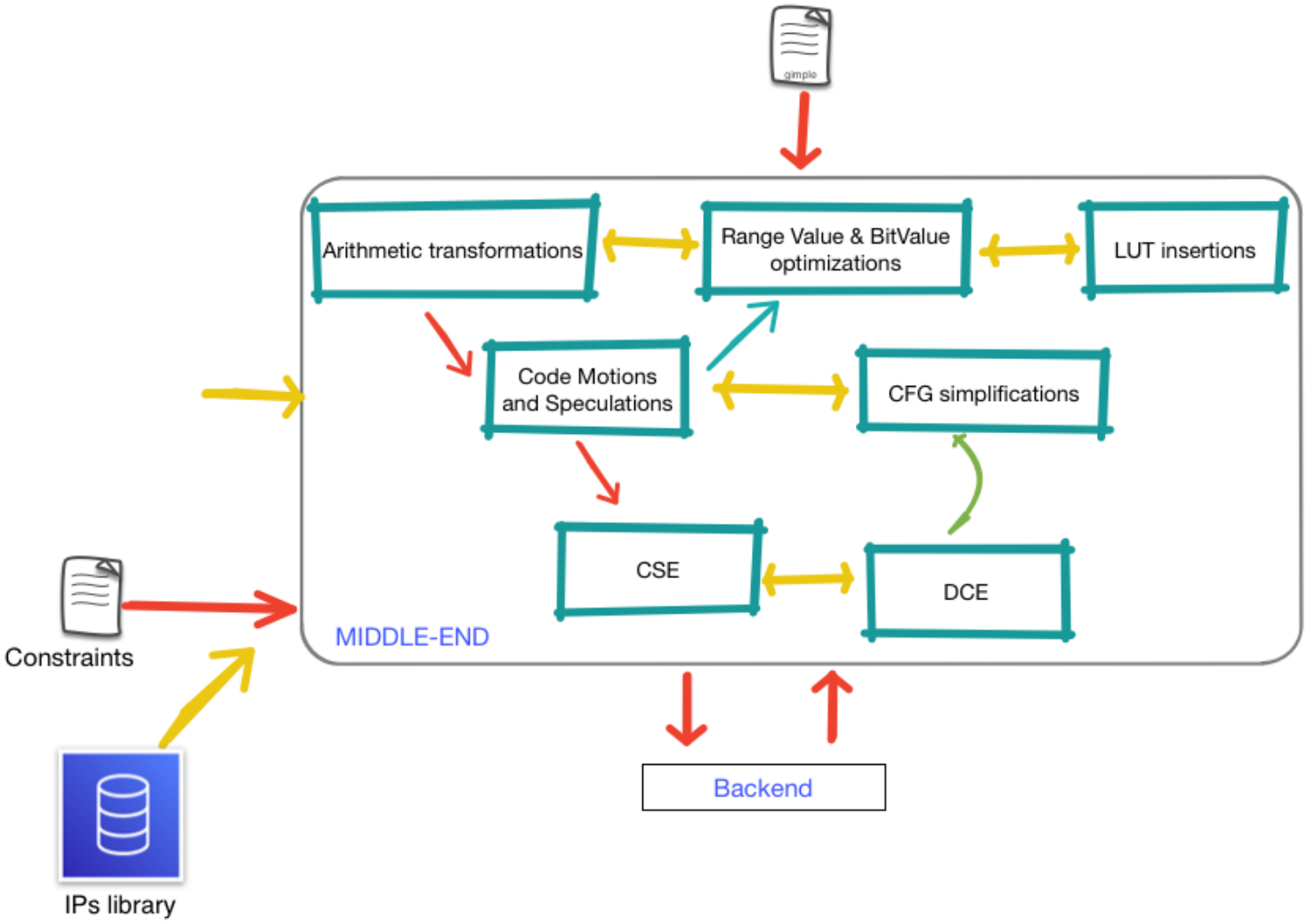
Bambu: front-end



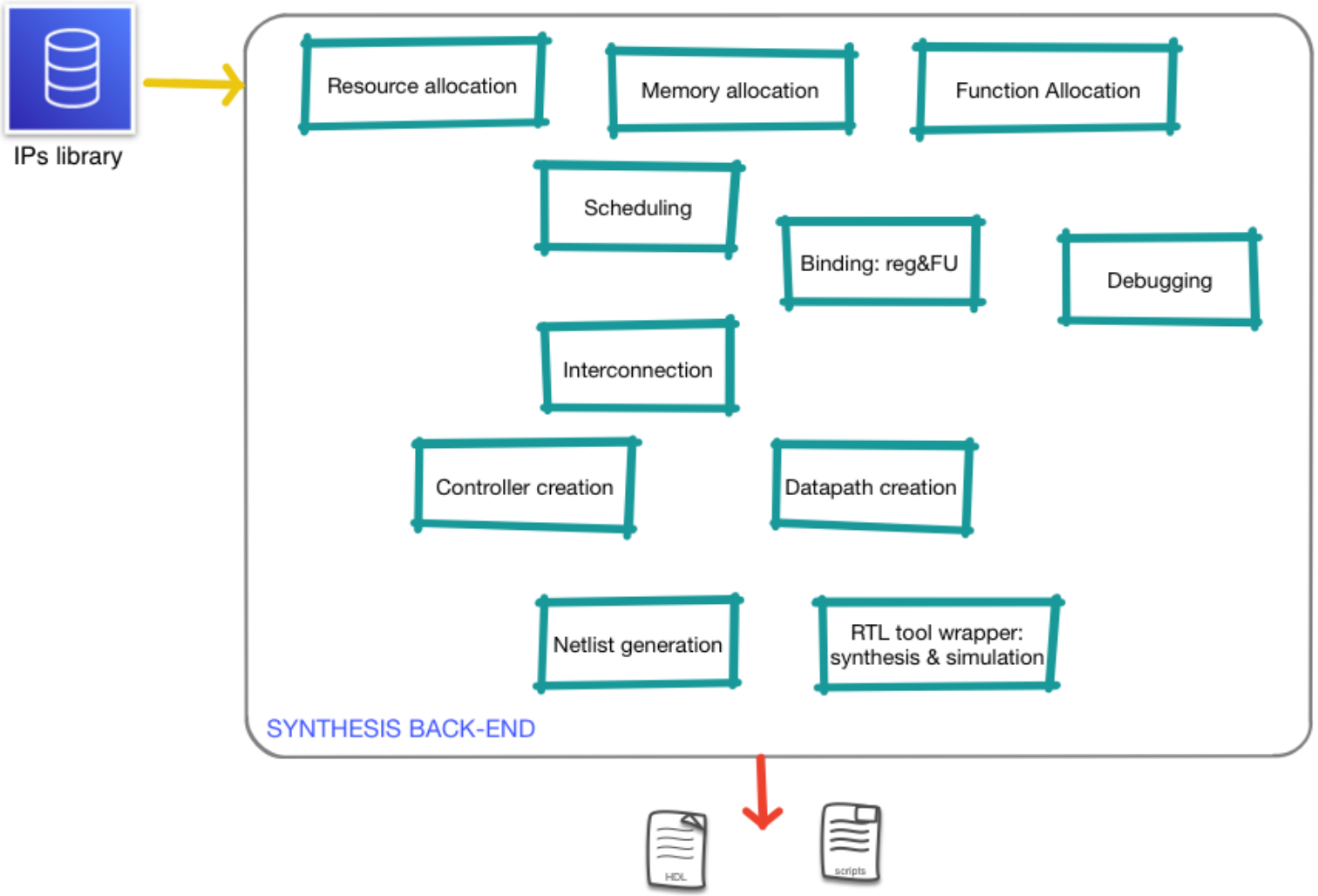
Bambu IR: behavior

- Gimple-like IR for the beviavior
  - SSA-based, 3 address form, BB, CFG, Call graph
  - Some extensions:
    - Predicated statements
      - Vectorization, Speculation/Code Motion and Tensor Optimizations
    - Code annotations coming from user directives
    - External interfaces with a specific protocol
    - Parallel annotations (OpenMP)
  - Serializable:
    - compressed gimple form (used to store pre-compiled libraries): libm, libc
    - C code that can be compiled (co-simulation)
    - LLVM .ll input supported in case a clang front-end compiler is used
    - MLIR input is supported through LLVM lowering

Bambu: middle-end



Bambu: back-end



Bambu IR: circuit

- Circuit for the structure
  - Standard RTL description with: signals, signals_vectors, ports, port_vectors, components
  - Ports and signals are typed
  - Support of global signals (interfaces and proxys)
  - Annotated with area, delay, initiation time, latency
  - Serializabile:
    - Input Functional unit libraries are described in XML format
    - Output in SystemVerilog and VHDL

Bambu: command-line interface

- Minimal command
  - `$ bambu filename.c`
- Select the top component
  - `$ bambu filename.c -top-fname=top_function_name`
- Controlling the clock period (100Mhz)
  - `$ bambu filename.c --clock-period=10`
- Select the device
  - `$ bambu filename.c -device-name=xc7z020,-1,clg484,VVD`

Subset of synthesizable C (1)

- We support what standard compilers accept as input (CLANG/LLVM and GCC)
- Supported features:
  - Expressions of any kind: arithmetic, logical, bitwise, relational, conditional, comma-based expressions.
  - Types: integers, single- and double-precision floating point, _Bool and Complex, struct-or-union, bitfields, enum, typedef, pointers and arrays, type qualifiers.
  - Variable declarations, initialization, storage-specifiers
  - Functions definition and declaration, extern or static, pointer to functions, parameters passed by copy or reference, tail recursive functions.
  - Statements and blocks: labeled (case), compound, expression, selection (if,switch), iteration(while,do,for), jump (goto,continue,break,return)
  - All preprocessor directives
  - Unaligned memory accesses and dynamic pointers resolution
  - GCC vectorization

Subset not supported

- struct returned by copy
- Non-tailing recursive functions

libbambu

- assert, puts, putchar, read, open, close, write, printf, exit, abort

- libc functions: bswap32,memcmp, memcpy, memmove, memset, malloc, free, memalign, alloca_with_align, calloc, bcopy, bzero, memchr, mempcpy, memrchr, rawmemchr, stpcpy, stpncpy, strcasecmp, strcasestr, strcat, strchr, strchrnul, strcmp, strcpy, strcspn, strdup, strlen, strncasecmp, strncat, strncmp, strncpy, strndup, strnlen, strpbrk, strrchr, strsep, strspn, strstr, strtok

- libm functions: acos, acosh, asin, asinh, atan, atan2, atanh, cbrt, ceil, cexpi, copysign, cos, cosh, drem, erf, exp, exp10, expm1, fabs, fdim, finite, floor, lfloor, fma, fmax, fmin, fmod, fpclassify, frexp, gamma, lgamma, tgamma, hypot, ilogb, infinity, isinf, isnan, j0, j1, jn, ldexp, log, log2, log10, log1p, modf, nan, nearbyint, nextafter, pow, pow10, remainder, remquo, rint, lrint, llrint, round, lround, llround, scalb, scalbln, scalbn, signbit, significand, sin, sincos, sinh, sqrt, tan, tanh, trunc.

Synthesis per function

- one component per function
  - function interface
  - start and done
  - parameter passing
    - wires
    - memory interaction
    - none (ap_none), acknowledge (ap_ack), valid (ap_vld), ovalid (ap_ovld), handshake (ap_hs), fifo (ap_fifo) and array (ap_memory), AXI
- hierarchy based on call graph
  - no-recursion
  - proxy

C++ support

- Support of C++ is ongoing:
  - templates
  - C++11 and beyond
  - ac_types from Mentor Graphics could be used
  - ap_types from Xilinx support by wrapping ac_types

```
#include <algorithm>
int gcd(int x, int y )
 {
        if( x < y )
           std::swap( x, y );

        while( y > 0 )
        {
           int f = x % y;
           x = y;
           y = f;
        }
        return x;
 }
```

Integration of hand- written components in the HLS flow

- Function mapped on IPs has to be declared as extern:
  - `extern void module1(uint32_t input1, uint16_t input2, module1_output_t *outputs);`
- C code has to be passed with the following option
  - `--C-no-parse=module1.c,…`
- Binding between function **module1** and component **module1** has to be specified with a XML file and passed as an option to bambu
  - `$ bambu … module_lib.xml`
- Check these examples:
  - `examples/IP_integration`
  - `examples/breakout`
  - `examples/pong`
  - `examples/led_example`

Research opportunities

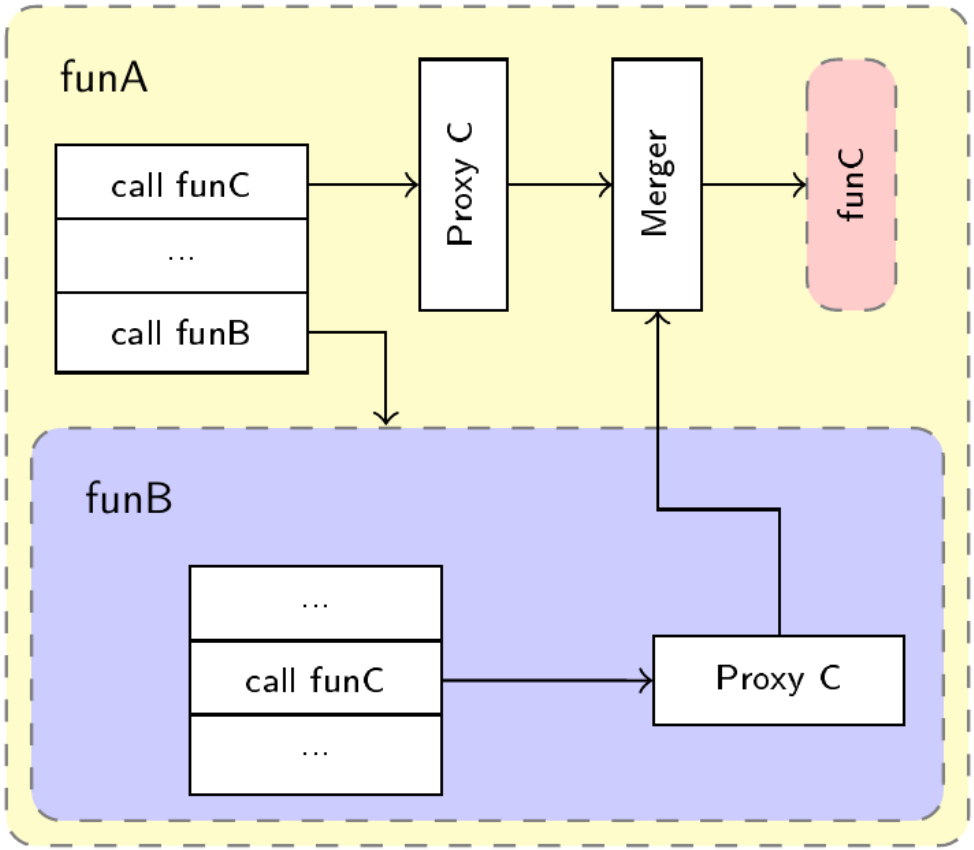
- Bambu is open-source, and it can be used to advance High-Level Synthesis research
- In the years many different algorithms and synthesis techniques have been added
  - Different allocation and binding algorithms

```
--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

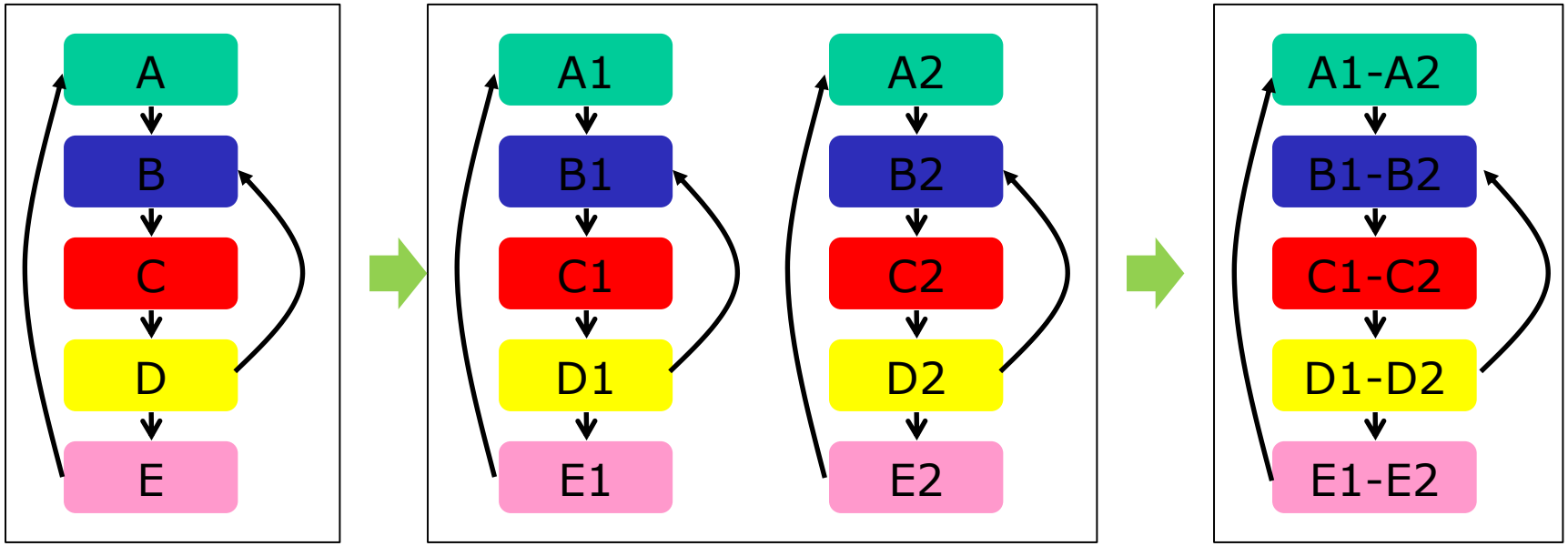
- Synthesis of function proxies



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

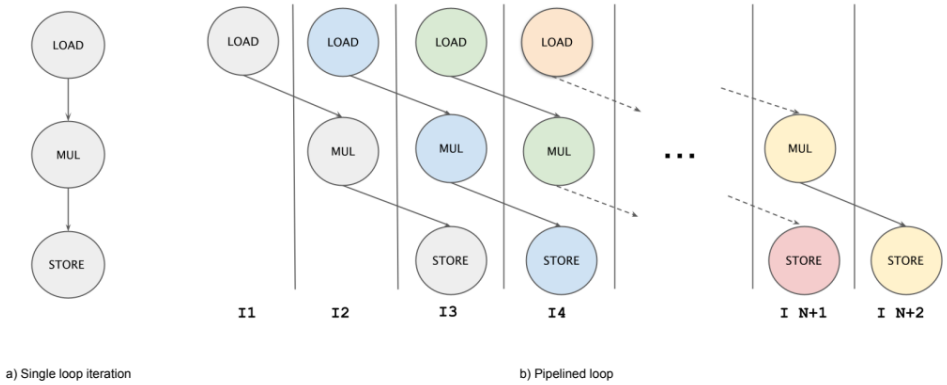
- Outer loop vectorization



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

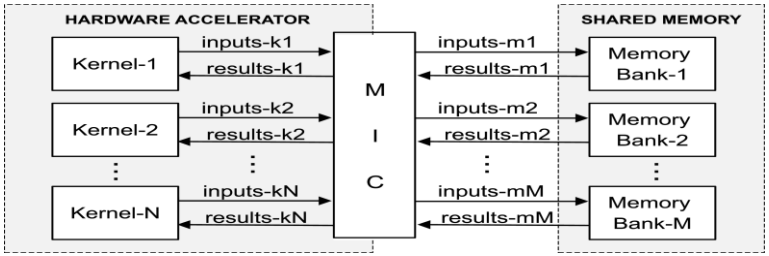



```
#map = affine_map<(d0) -> (d0 - 2)>
func @example(%arg0: memref<1000xi32>,
              %arg1: memref<1000xi32>) {
  %c0 = arith.constant 0 : index
  %0 = affine.load %arg0[%c0] : memref<1000xi32>
  %c1 = arith.constant 1 : index
  %1 = affine.load %arg0[%c1] : memref<1000xi32>
  %2 = arith.muli %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 LINES: Parallelization

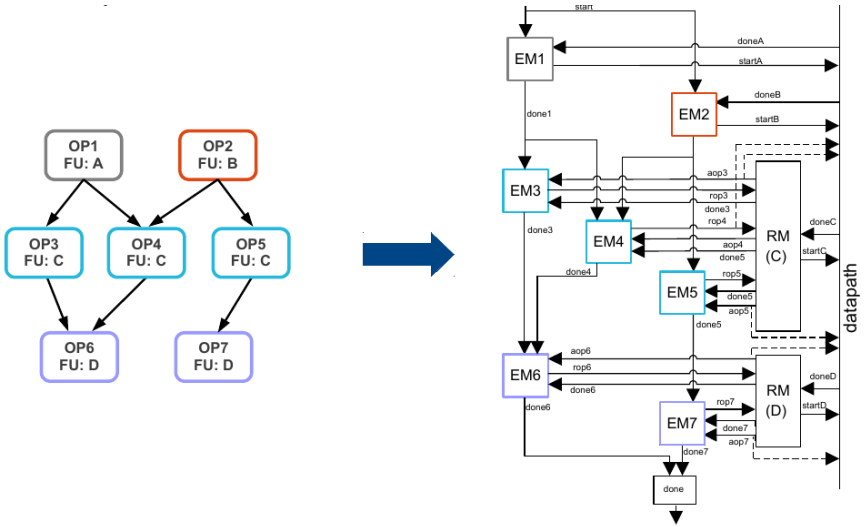
- architectural template that exploits both instruction level and task-level parallelism
- Dynamic hardware scheduler
- single-cycle hardware context switching
- OpenMP support: omp for, omp simd



- 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 LINES: Dynamic vs static scheduling

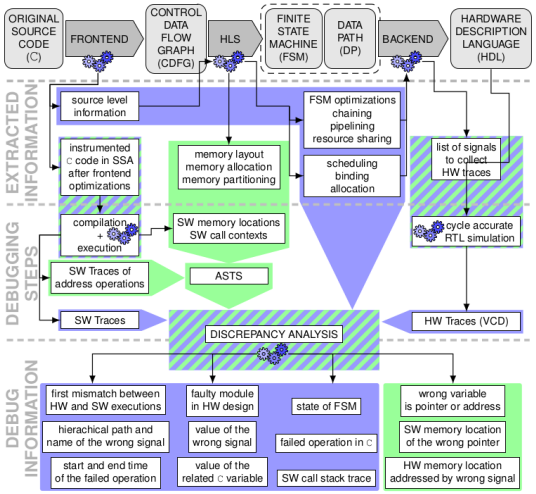
- FSMD vs adaptive controller
  - Imperative vs dataflow oriented
- Hierarchical memory controller: parallel accesses



- 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.

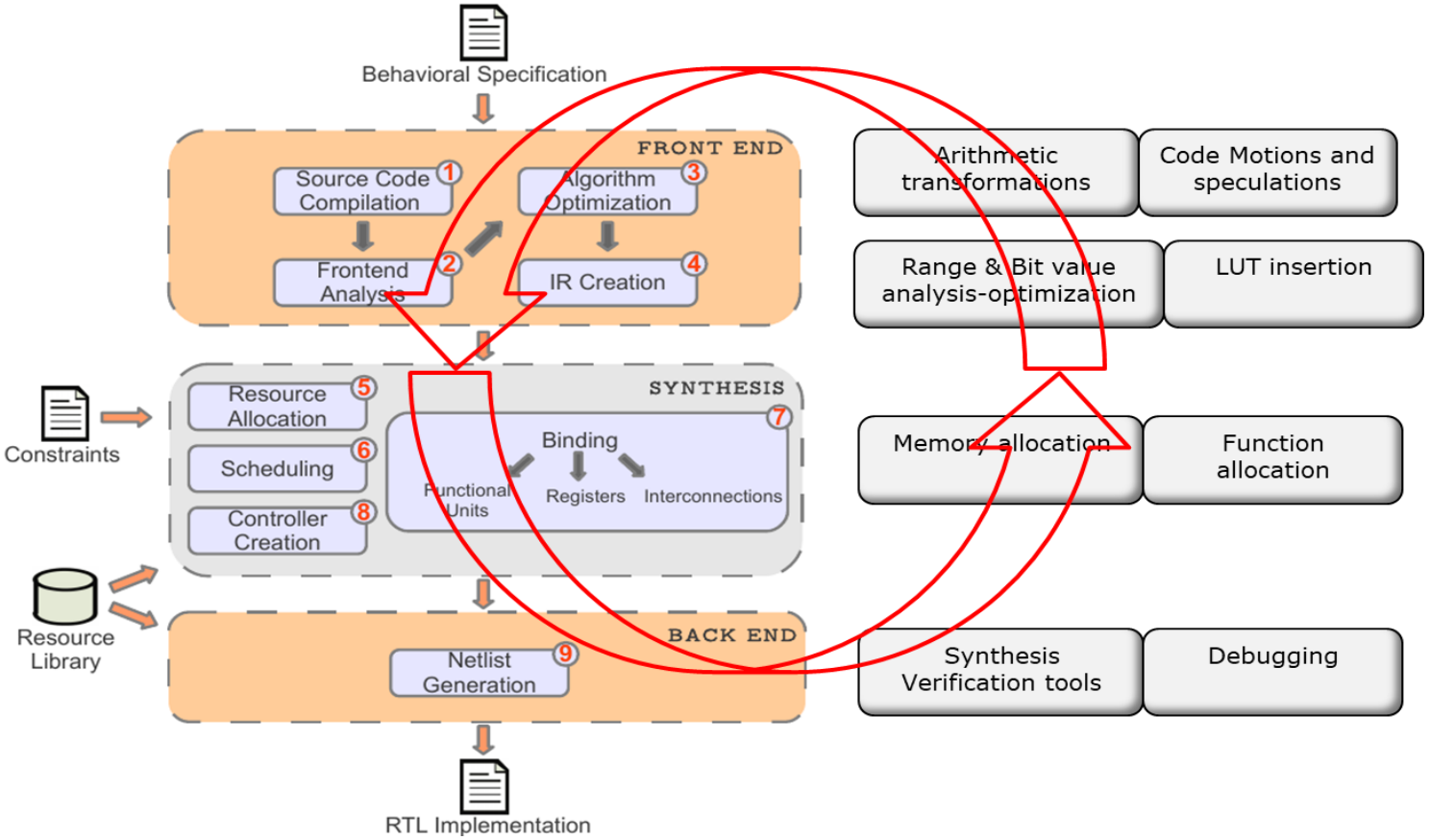
RESEARCH LINES: Verification and Debug

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



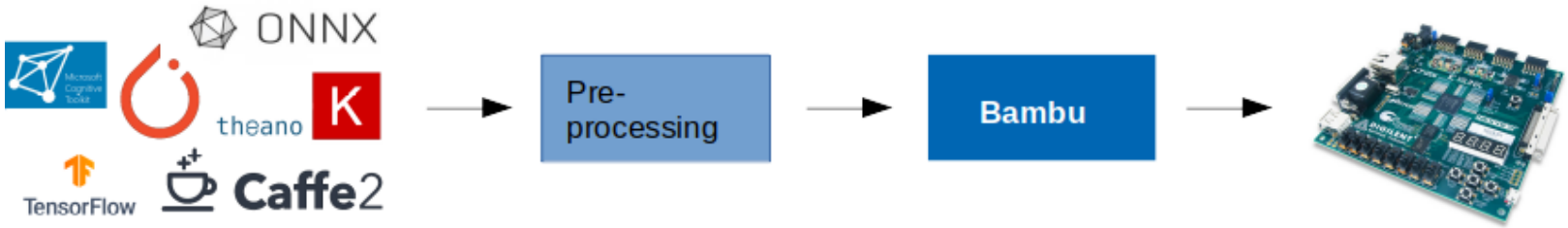
- P. Fezzardi and F. Ferrandi, "Automated Bug Detection for High-Level Synthesis of Multi-Threaded Irregular Applications," *ACM Trans. Parallel Comput.*, vol. 7, iss. 4, 2020.
- P. Fezzardi, M. Lattuada, and F. Ferrandi, "Using Efficient Path Profiling to Optimize Memory Consumption of On-Chip Debugging 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 LINES: Design space exploration for HLS



- 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 LINES: Machine Learning accelerators



hls4ml/FINN produce C++ specific to Vivado HLS and Xilinx FPGAs

- generalize the library so that Bambu can produce efficient designs
- use bambu to target FPGAs from any vendor

onnx-mlir produces LLVM code for CPU

- direct synthesis may be inefficient
- combine with SODA Synthesizer to make it hardware-oriented

RESEARCH LINES: MLIR integration

- MLIR dialects -> LLVM dialect -> LLVM IR -> bambu
  - any design written in MLIR can be synthesized
- *Higher-Level* synthesis
  - MLIR as higher level of abstraction
  - apply high-level optimizations (e.g. loop pipelining connected with SODA Synthesizer)

- 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.

Binary distribution now available

- The code is available as AppImage on https://panda.dei.polimi.it
  - bambu-x86_64.AppImage
  - AppImage is a format for distributing portable software on Linux without needing superuser permissions to install the application.
  - Once downloaded just add the execution rights with this command:

```
chmod +x bambu-x86_64.AppImage
```

  - Source code is available on GitHub:
    - https://github.com/ferrandi/PandA-bambu/

Questions?

http://panda.deib.polimi.it

F. Ferrandi, V. G. Castellana, S. Curzel, P. Fezzardi, M. Fiorito, M. Lattuada, M. Minutoli, C. Pilato, and A. Tumeo, "Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications," in 2021 58th ACM/IEEE Design Automation Conference (DAC), 2021, pp. 1327-1330.