

# Design, Automation and Test in Europe Conference

March 21, 2022



 POLITECNICO DI MILANO

Modern High-Level Synthesis for Complex Data Science Applications

## Productive High-Level Synthesis with Bambu

**Serena Curzel**

Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

[serena.curzel@polimi.it](mailto:serena.curzel@polimi.it)

Pacific Northwest National Laboratory

[serena.curzel@pnl.gov](mailto:serena.curzel@pnl.gov)



**Pacific Northwest**  
NATIONAL LABORATORY

- ❑ The hands-on sessions are supported by Jupiter Notebook(s) running on Google Colab
  - [Link](#) for this session and the next
  - If you don't have access to a Google account, you can run the notebook locally (provided all dependencies are met)
- ❑ The first cell of the notebook downloads Bambu and all required tools, you don't need to install anything else
- ❑ If you want to use Bambu later on your own, [installation instructions](#) are provided on the website

Explore folders and open files here

Runtime information

Run cell

The screenshot shows the Bambu IDE interface. On the left is a file explorer showing a directory structure with folders like 'bin', 'boot', 'content', 'PandA-bambu', 'bambu-tutorial', 'sample\_data', 'bambu-showcase.AppImage', 'datalab', 'dev', 'etc', 'home', 'lib', 'lib32', 'lib64', 'libx32', 'media', 'mnt', 'opt', 'proc', 'python-apt', 'root', 'run', 'sbin', 'srv', 'sys', 'tensorflow-1.15.2', 'tmp', and 'tools'. A blue circle highlights the file explorer. In the center is a code editor with a terminal cell. The cell contains the following commands:

```
[1] add-apt-repository -y ppa:git-core/ppa
apt-get update
!apt-get install -y --no-install-recommends build-essential ca-certificates gcc-multilib git iverilog verilator
!wget https://release.bambuhls.eu:8080/appimage/bambu-showcase.AppImage
!chmod +x bambu-*.AppImage
!ln -sf $PWD/bambu-*.AppImage /bin/bambu
!ln -sf $PWD/bambu-*.AppImage /bin/spider
!ln -sf $PWD/bambu-*.AppImage /bin/tree-panda-gcc
!git clone --depth 1 --filter=blob:none --branch tutorial_2021 --sparse https://github.com/ferrandi/PandA-bambu.git
%cd PandA-bambu
!git sparse-checkout set documentation/tutorial_ics_2021
%cd ..
!mv PandA-bambu/documentation/tutorial_ics_2021/ bambu-tutorial
```

A blue circle highlights the first line of the code cell. Below the code cell, there are sections for 'Introduction' and 'Exercise 1'. The 'Exercise 1' section contains the following text:

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 --simulator=VERILATOR --simulate --generate-tb=test_icrc1.xml -v2 --print-dot --pretty-print=a.c 2>&1 | tee icrc1.log
```

A blue circle highlights the code cell. On the right side of the interface, there is a 'Runtime information' panel showing 'RAM' and 'Disk' usage. A blue circle highlights this panel. Arrows point from the text labels to these elements.

Edit cell (use ! for bash commands)

## □ Basic command:

```
bambu filename.c --top-fname=name
```

□ Input: C (or C++, or LLVM IR) file

□ Output: Verilog (or VHDL) file

□ Compiler-like command-line interface

- Other options are passed as compiler flags, e.g.:

```
-v0 | -v1 | -v2 | -v3 | -v4
```

Verbosity level

```
-O0 | -O1 | -O2 | ...
```

Optimization level (next session)

```
-I<file_name>
```

Additional inputs

```
-lm
```

Functions from `math.h`

- Print all available options:

```
bambu --help
```

- For example, you can find how to select VHDL output instead of Verilog

```
--writer, -w<language>  
    Output RTL language:  
        V - Verilog (default)  
        H - VHDL
```

- Other resources:

- [Website/GitHub issues/panda-info@polimi.it](#)
- [Examples folder](#)

- ❑ To verify that the generated HDL design produces correct results, Bambu interfaces with simulation tools

```
--simulate --simulator=SIMULATOR_NAME
```

- ❑ Supported simulators are Verilator, Icarus, ModelSim (Mentor), Xsim (Xilinx), Isim (Xilinx)

- ❑ Bambu automatically produces an HDL testbench with input values
  - Random values (no option specified)
  - User provided values `--generate-tb="a=1,b=2"`
  - User provided values `--generate-tb=test.xml`
- ❑ Matching between input values and accelerator ports is *name based*
  - Exception: when the input is a `.11` file, inputs must be named `P0, P1, P2...`
- ❑ Reference outputs can be inferred from the execution of the input code, or provided by the user

- ❑ The testbench communicates with the top-level module to control the computation and collect the computed results
- ❑ The inputs are fed to both the sw input and the generated hw module
  - If the module outputs do not match with the return values of the input code, Bambu raises an error
  - If they do, Bambu reports the number of clock cycles
- ❑ *The whole process is automated*



## □ Basic command:

```
bambu filename.c --top-fname=name
```

## □ Selecting the frontend compiler:

```
--compiler=I386_GCC8|I386_CLANG7|...
```

## □ Selecting the hardware target:

```
--device-name=5SGXEA7N2F45C1 --clock-period=5
```

## ❑ Intel

- Cyclone II: EP2C70F896C6, EP2C70F896C6-R
- Cyclone V: 5CSEMA5F31C6
- Stratix IV: EP4SGX530KH40C2
- Stratix V: 5SGXEA7N2F45C1

## ❑ Lattice

- ECP3: LFE335EA8FN484C

## ❑ AMD/Xilinx

- Virtex 4: xc4vlx100-10ff1513
- Virtex 5: xc5vlx110t-1ff1136 xc5vlx330t-2ff1738 xc5vlx50-3ff1153
- Virtex 6: xc6vlx240t-1ff1156
- Artix 7: xc7a100t-1csg324-VVD
- Virtex 7: xc7vx330t-1ffg1157 xc7vx485t-2ffg1761-VVD xc7vx690t-3ffg1930-VVD
- Zynq: **xc7z020-1clg484-VVD (default)**, xc7z020-1clg484, xc7z020-1clg484-YOSYS-VVD

## ❑ NanoXplore

- Brave NG-Medium
- Brave NG-Large

## ❑ ASIC Nangate 45nm and ASAP7 (through OpenROAD)

- ❑ Different targets (FPGA device and clock period) imply:
  - Different delays (e.g., delay of a DSP)
  - Different sizes (e.g., number of LUTs)
  - Different HDL descriptions
- ❑ Target information is embedded in XML files
  - Supported devices are included in Bambu executable
  - Characterization carried out through eucalyptus (distributed in PandA)
  - New devices can be passed to the tool as XML files (see [example](#))

- ❑ Bambu can directly interface logic synthesis tools:
  - Quartus / Quartus Prime
  - ISE / Vivado
  - OpenROAD
  - Diamond
  - NxMap
- ❑ By default, Bambu generates synthesis scripts for the appropriate tool depending on the target board
- ❑ With `--evaluation` Bambu *automatically launches the synthesis script and collects information* about generated solutions

- ❑ `--no-iob` is usually required to avoid consuming all available I/O pins on an FPGA
- ❑ Users can provide additional
  - Constraint files `--backend-sdc-extensions`
  - TCL scripts `--backend-script-extensions`

## □ Bambu can also produce

- Graphical representations of the Finite State Machine and other relevant graphs

```
--print-dot
```

- A C version of the internal Bambu IR

```
--pretty-print=a.c
```

- VCD for waveform visualization

```
--generate-vcd
```

- All temporary files

```
--no-clean
```

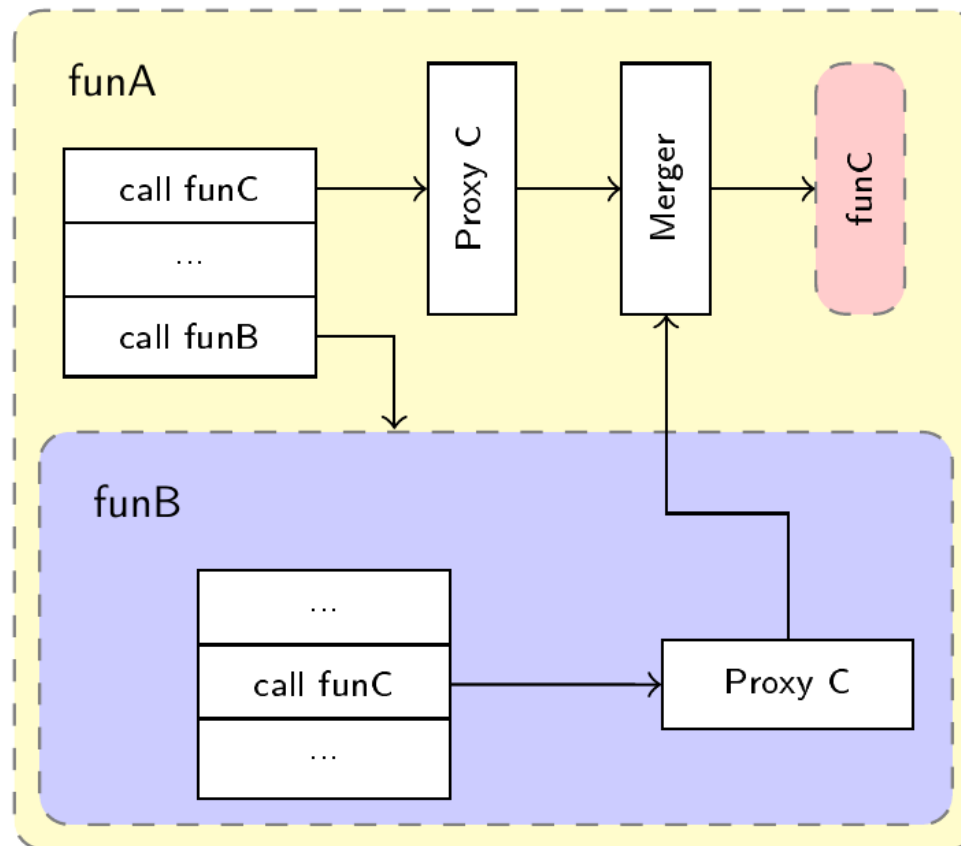
- Bambu is open-source, and it can be used to advance High-Level Synthesis research
- In the years many different algorithms and synthesis techniques have been added
  - Different allocation and binding algorithms

```
--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
    ...
```

- Multi-threaded design (last presentation today)

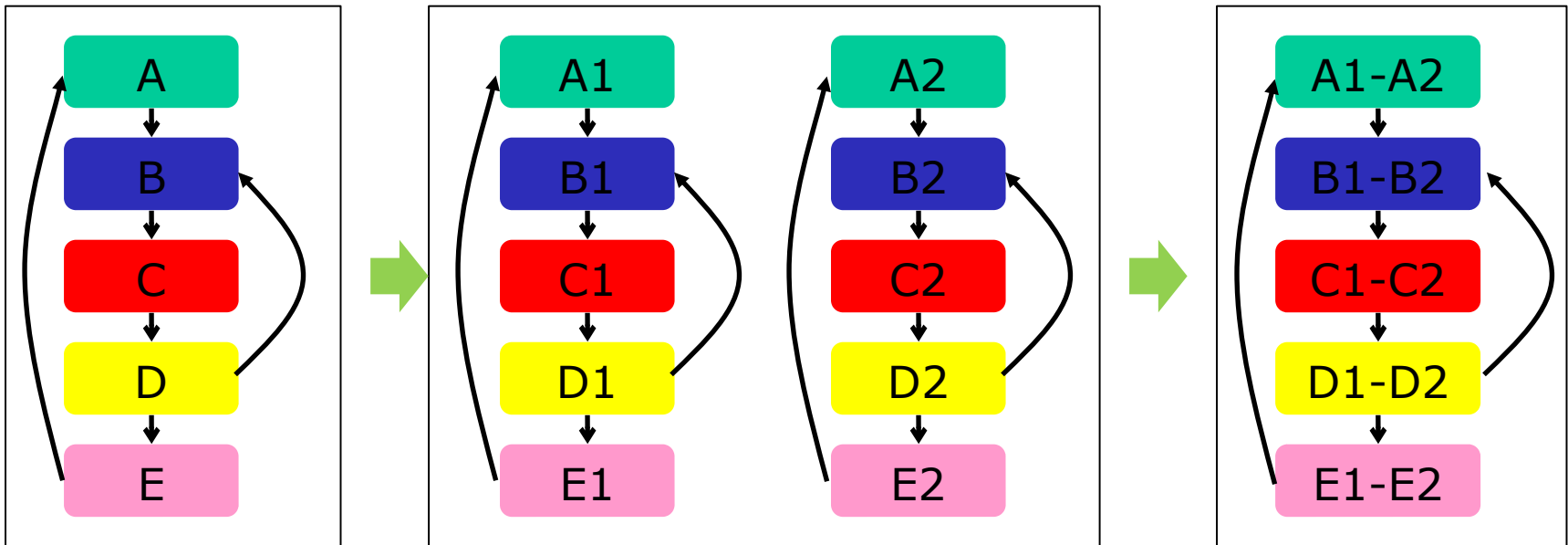
- Synthesis of function proxies



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.



- 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.