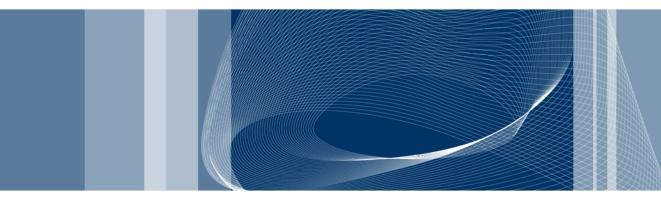
International Conference on Supercomputing

June 14 - 18, 2021. Worldwide online event



Y POLITECNICO DI MILANO





Bambu: High-Level Synthesis for Parallel Programming

Target Selection and Tool Integration

Serena Curzel

Politecnico di Milano Dipartimento di Elettronica, Informazione e Bioingegneria serena.curzel@polimi.it

- ☐ Target selection
- Integration with simulation and logic synthesis tools
- Co-simulation details

- Target = FPGA device(+synthesis tool) + clock period
 - ▶ Different delays of FPGA elements (i.e., delay of a DSP)
 - ► Different sizes of FPGA elements (i.e., size of LUTs)
 - ▶ Different HDL description of memory elements
- Target device + target clock period can be specified
 - Default is
 - Target device: xc7z020-1clg484-VVD
 - Target clock period: 10ns

- Target information is embedded in XML files
 - Supported devices are included in Bambu executable
 - New devices must be passed to the tool
- XML file mainly contains characterization of functional units
 - ► Area
 - Delay
- XML files are automatically generated by means of eucalyptus (distributed in PandA)
 - ▶ New devices can be easily added
 - ► See <u>example</u>

- 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 (experimental)

```
--device-name=<value>
```

Specify the name of the device (see previous slide) Default is xc7z020-1clg484 (Xilinx Zynq)

```
--clock-period=<value>
```

Specify the period of the clock signal (in nanoseconds)

Default is 10

Example:

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

- Bambu can directly interface synthesis tools:
 - Quartus / Quartus Prime
 - ► ISE
 - ▶ Vivado
 - Diamond
 - ▶ Nanonxpython
- By default, Bambu generates synthesis scripts for the appropriate tool
- With _--evaluation Bambu launches the synthesis script and collects information about generated solutions

- Users can provide
 - VHDL/Verilog implementation of custom modules
 - ► Constraint files | --backend-sdc-extensions

- The design flow can be modified
 - ► XML files containing custom TCL scripts

--backend-script-extensions

- Users provide input values for the tests
- Output values can be provided by the user, or inferred:
 - ▶ Input C code is executed with given inputs
 - Return values are considered the golden reference for HW
- A testbench wrapper in HDL is generated to test the design
 - ▶ It communicates with the top-level to start the computation
 - ▶ It collects the computed results
- If the result do not match Bambu emits an error message

- --simulator=SIMULATOR_NAME selects the simulatorValid values for SIMULATOR NAME are:
 - VERILATOR Verilator, an open source cycle-based Verilog simulator
 - ▶ ICARUS Icarus Verilog, an open source event-based Verilog simulator
 - ▶ MODELSIM ModelSim from Mentor (Verilog, VHDL, Mixed)
 - XSIM The Vivado Simulator from Xilinx (Verilog, VHDL, Mixed)
 - ▶ ISIM The ISim ISE Simulator from Xilinx (Verilog, VHDL, Mixed)

- Testbench is automatically generated in Verilog by Bambu starting from:
 - ► Randomly generated values
 - ► XML file | --generate-tb=<file.xml>

 - ► Annotated C file
 - Support to open, read, write of files
- Maximum allowed ULP can be set

- Matching between parameter names in the XML testbench and accelerator ports is name based
 - ► Exception: when the input is a .11 file, parameters must be named P0, P1, P2...
- XML files must respect any intrinsic assumption the code makes (e.g. array sizes)
- Syntax is similar to C initialization syntax, with comma-separated list of values to initialize the memory
- Bambu co-simulation workflow and testbench generation handle the rest, both for C and for HDL