POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

Fast and Accurate IR-Driven Simulation for HLS System Design: Updates from the Bambu Framework

CGO 2026 – SODA tutorial
LATHC 2026

31.01.2026 I Michele Fiorito, Serena Curzel, Fabrizio Ferrandi

Fast and Accurate IR-Driven Simulation for HLS System Design: Updates from the Bambu Framework

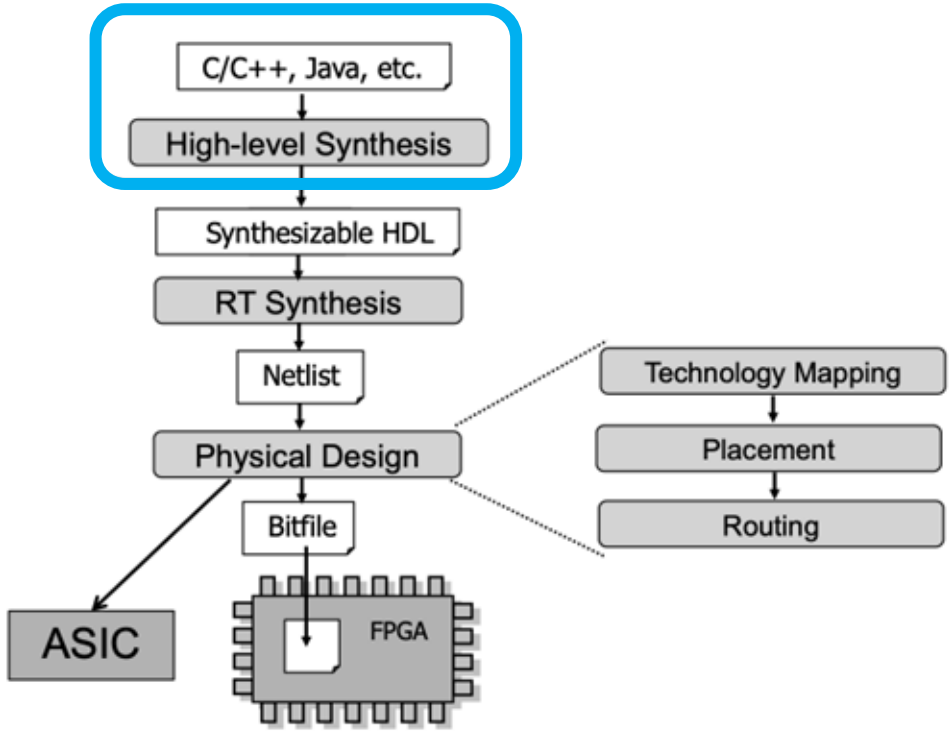
Bambu: a modern open-source HLS tool

High-Level Synthesis:

- High-level behavioral design through sw programming languages
- Automated translation to Register-Transfer Level
- Like compilers, but for hardware

→ Increased performance of FPGA/ASIC made available to software developers without hardware design expertise

Most popular tools provided within commercial EDA suites (Vitis HLS from AMD/Xilinx, Catapult from Siemens…)



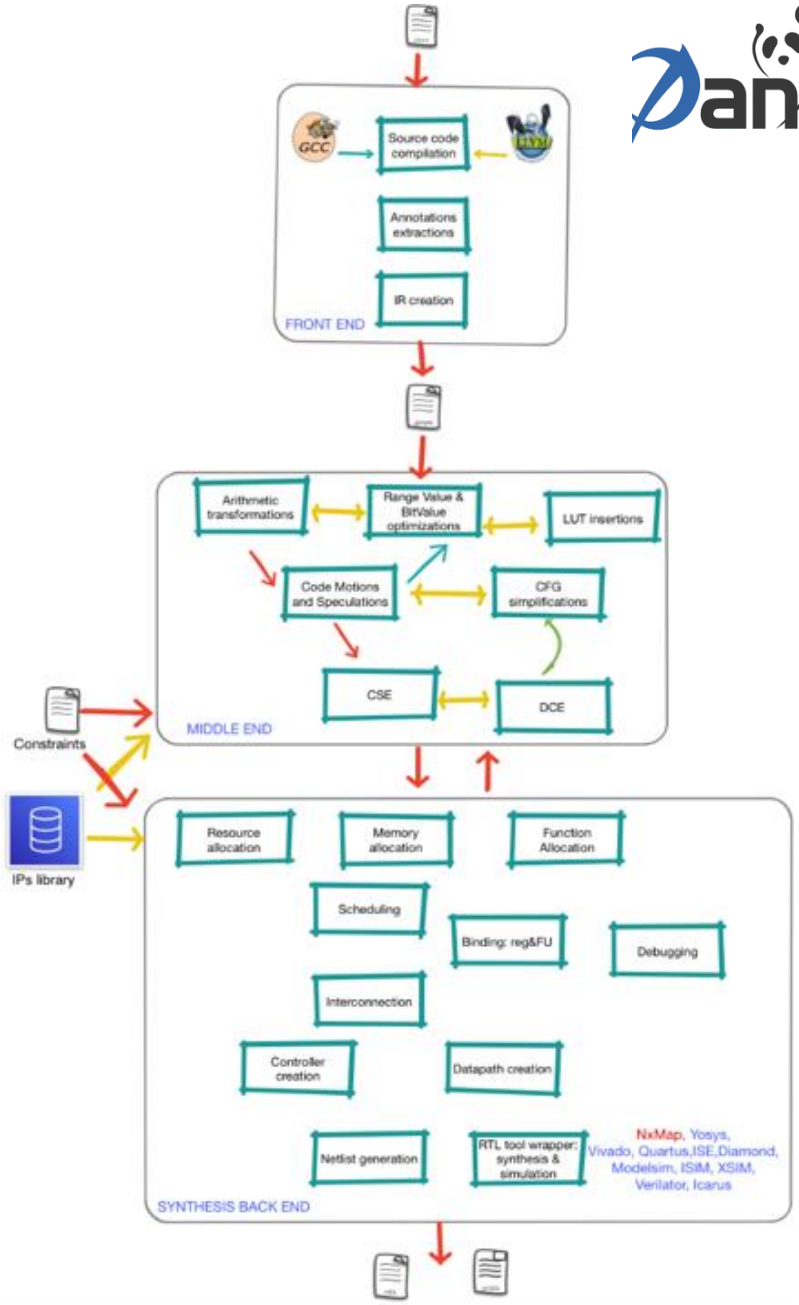
Bambu: a modern open-source HLS tool

Bambu HLS

- Developed and maintained by the PandA group at Politecnico di Milano since 15+ years
- Open-source

Synthesis flow:

- Compilation and optimization of the intermediate representations (IR)
- Allocation of resources
- Scheduling of operations
- Binding operations to resources
- Generation of synthesizable RTL



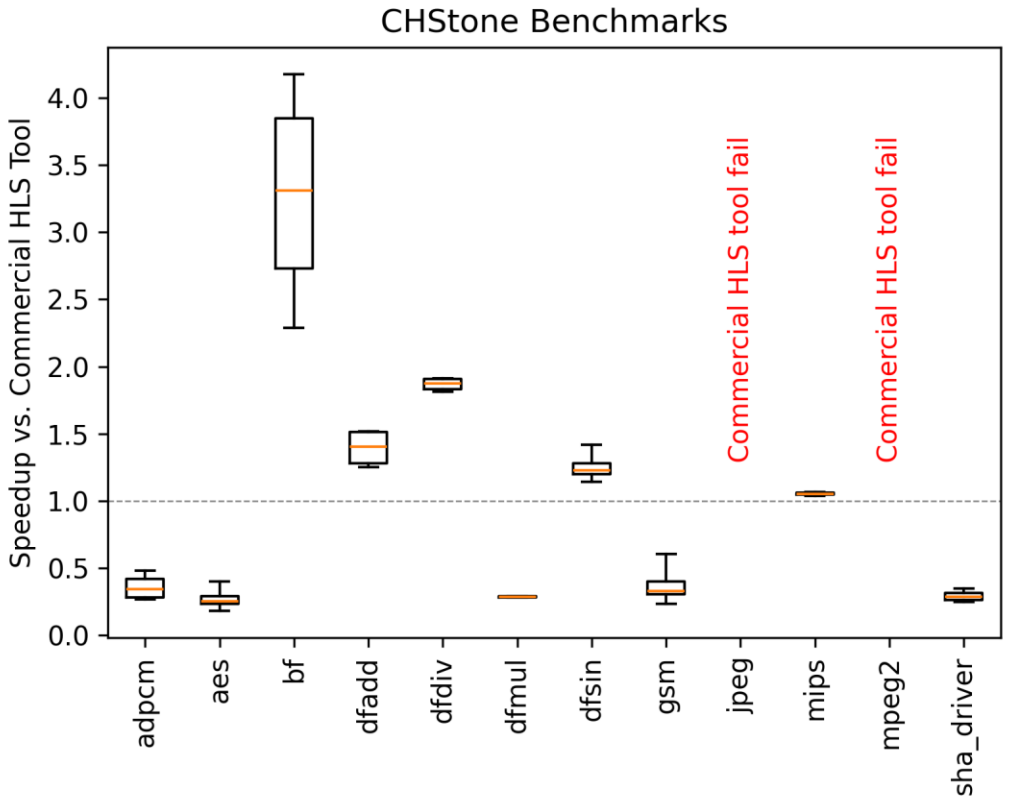
F. Ferrandi et al., Invited: Bambu: an Open-Source Research Framework for the High-Level Synthesis of Complex Applications, 58th ACM/IEEE Design Automation Conference (DAC), 2021.

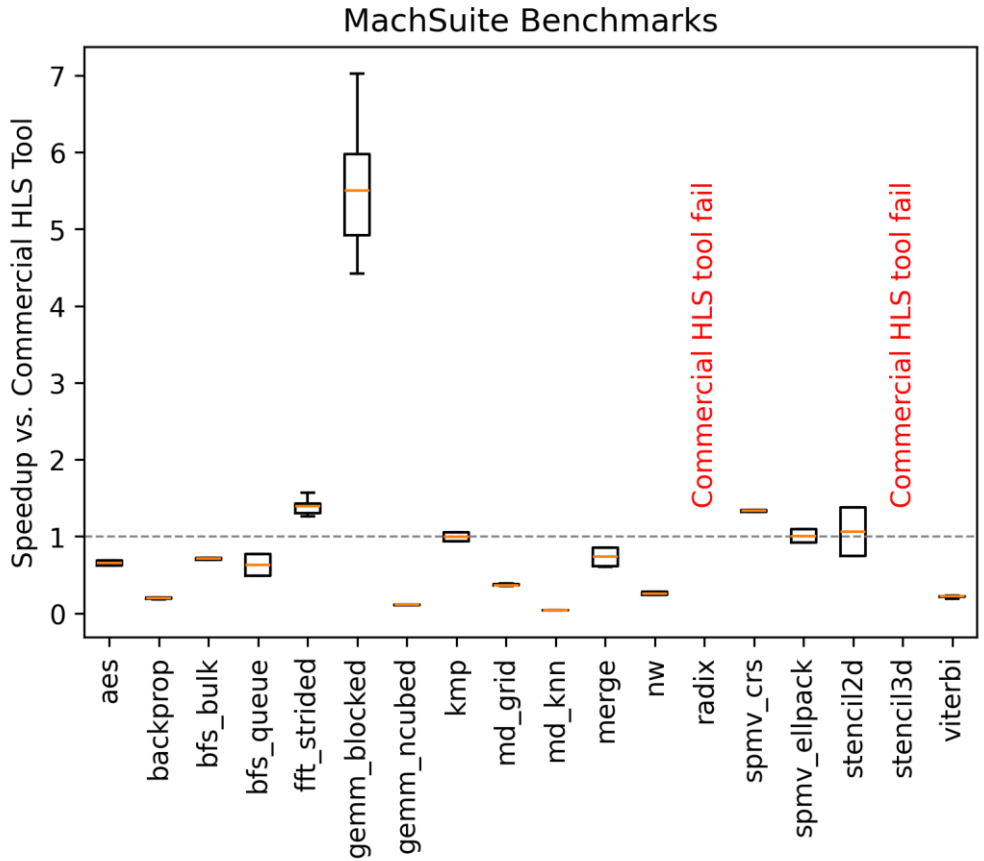
Bambu HLS Quality of Results

Speedup and area consumption over commercial HLS tool across different Bambu configurations.

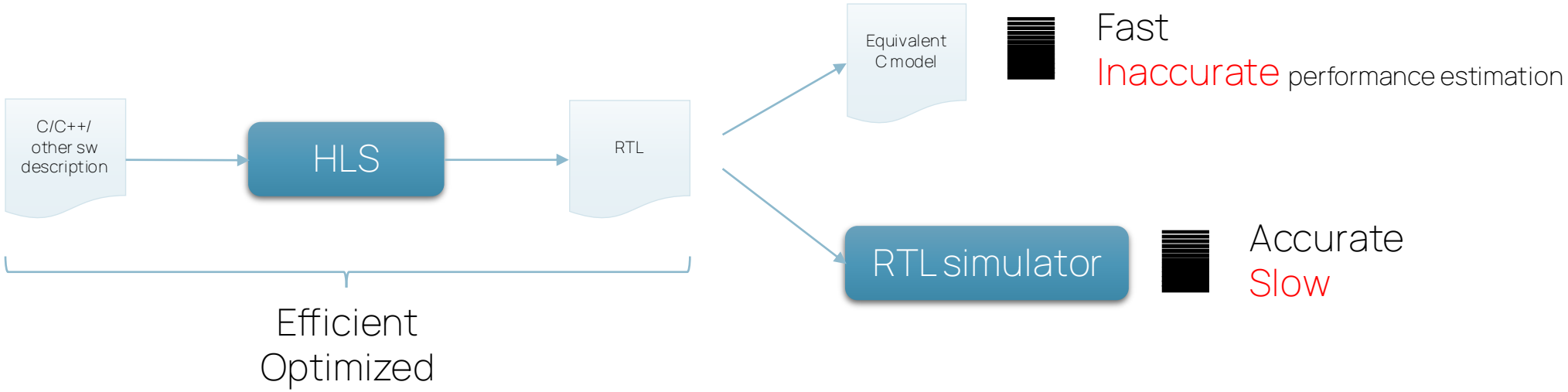
*Latency is measured in ns (clock cycles * achieved period post-implementation). > 1 is better.*

*Area is measured in Equivalent LUTs (BRAMs * 40 + DRAMs * 40 + DSPs * 40 + Registers * 0.5 + LUTs). < 1 is better.*



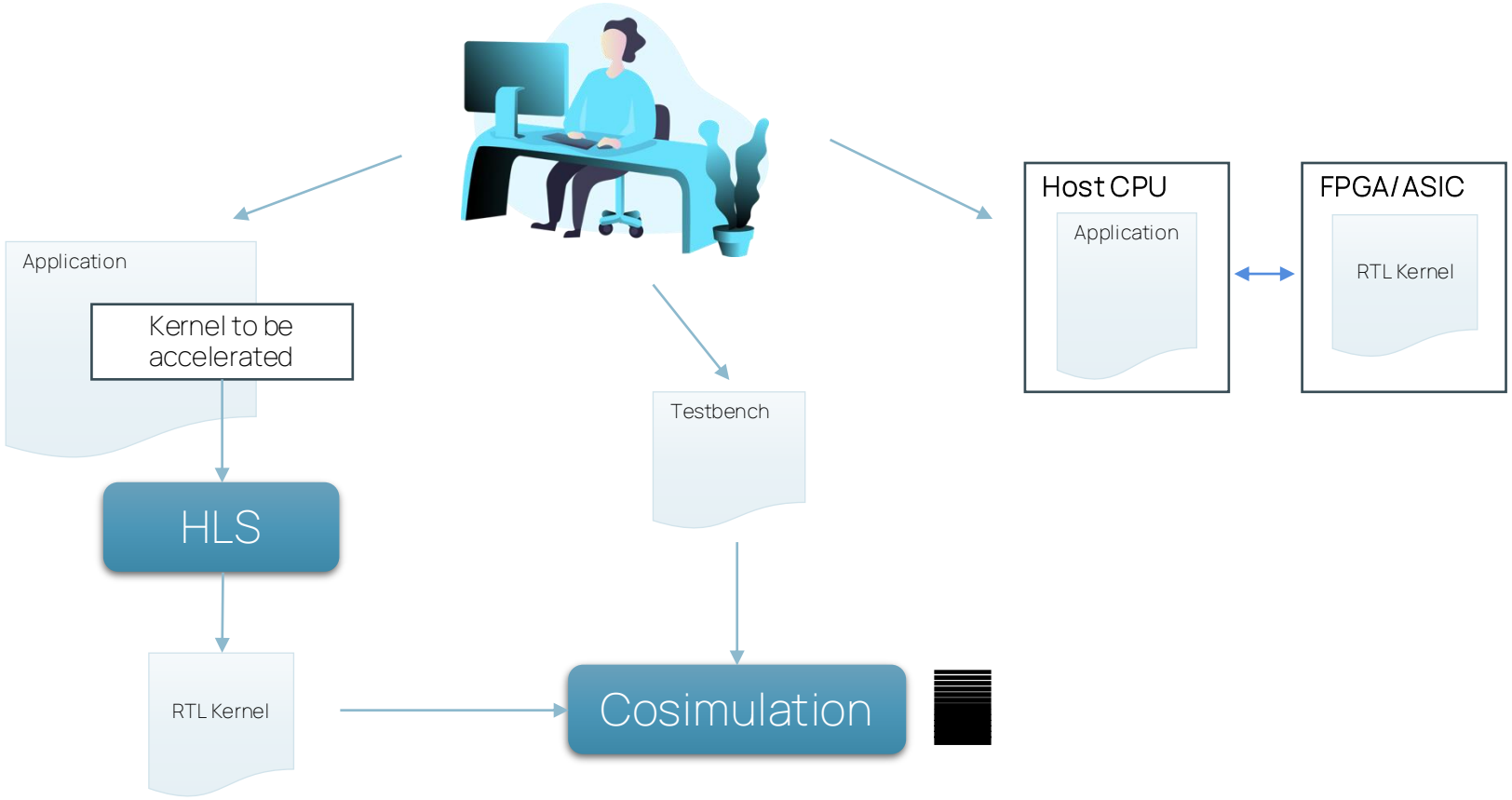


Verification of HLS-Generated Accelerators



Challenge 1:
We need a ***fast and accurate*** verification methodology for HLS-generated accelerators to assess their correctness and performance.

Verification of HLS-Generated Accelerators



Challenge 2:
We need an ***automated*** framework for ***system-level*** HW/SW simulation that can model interactions with the host application.

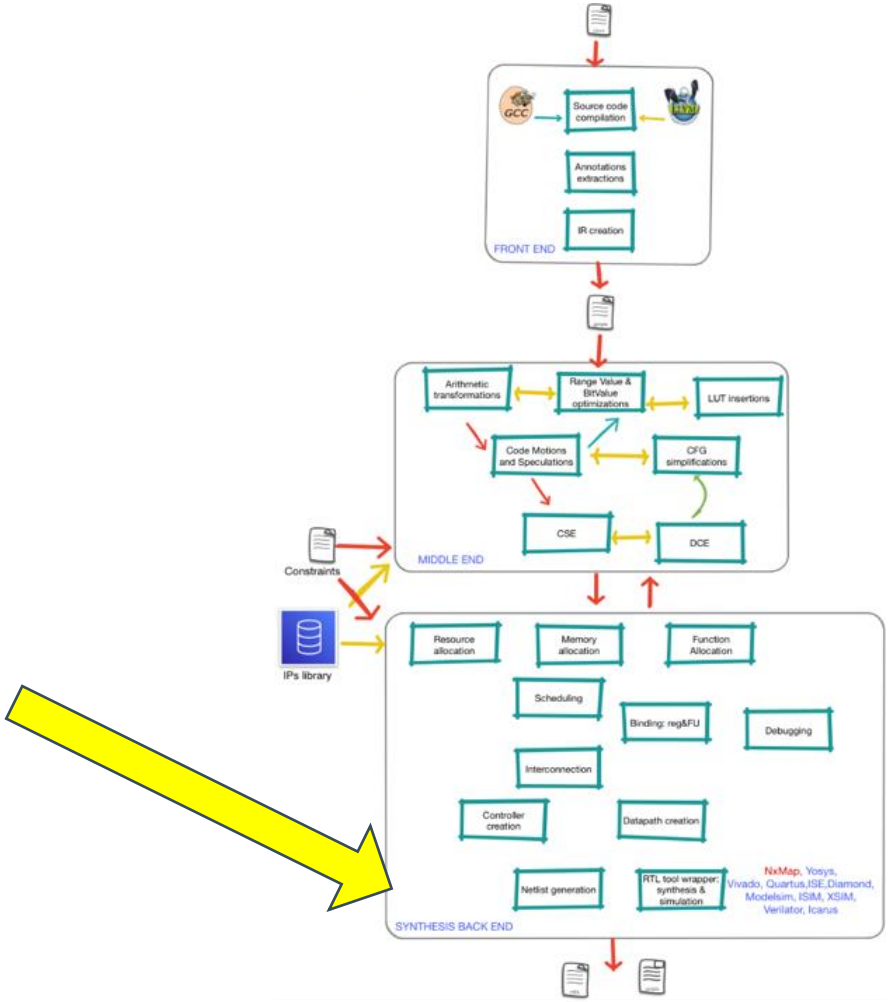
Fast and Accurate Functional Verification

Key insight:

- Use an HLS IR (see also FLASH [TCAD '20], LightningSimV2 [FPGA '24])

Our solution:

- Extract the Bambu IR just before it is translated to Verilog/VHDL
  - Open-source HLS tool
  - All optimizations have already been applied!
- Generate a C version, compile, and run
  - More human-readable than RTL
  - Can be used to debug HLS errors
  - Annotated with timing information from the scheduling phase
  - Bit-accurate **and** orders of magnitude faster than RTL simulation



[TCAD'20] Y.-K. Choi, Y. Chi, J. Wang, and J. Cong, "FLASH: Fast, parallel, and accurate simulator for hls," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4828– 4841, 2020.
[FPGA '24] R. Sarkar, R. Paul, and C. C. Hao, " LightningSimV2: Faster and Scalable Simulation for High-Level Synthesis via Graph Compilation and Optimization ," in *2024 IEEE 32nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2024, pp. 104–114.

Fast and Accurate Functional Verification

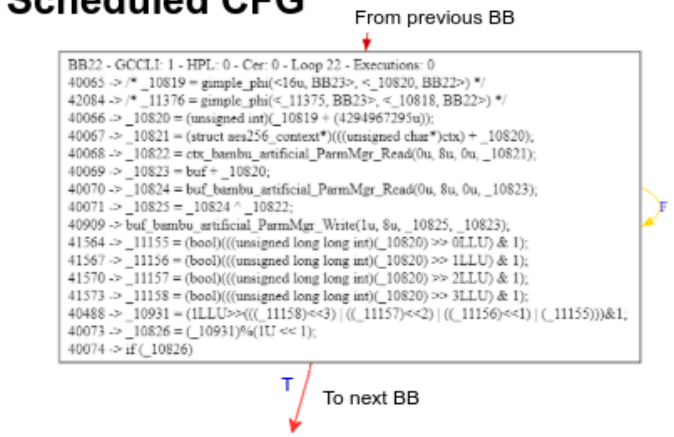
IR instrumentation:

- At conditional statements, to update a clock cycle count *online* during execution

Input source

```
void aes_addRoundKey(uint8_t *buf, uint8_t *key)
{
        register uint8_t i = 16;
        addkey: while (i--) buf[i] ^= key[i];
}
```

Scheduled CFG



C simulation model

= next BB takes 22 cycles

```
__bambu_csim_trace(38438, 22, 0);
BB_LABEL_22_0:
__t__42120_0 = _10819;
/* __t__42120_0 = gimple_phi(<16u, BB23>, <_10820, BB22>) */
_10820 = (unsigned int)(__t__42120_0 + (4294967295u));
_10821 = (struct aes256_context*)((unsigned char*)ctx + _10820);
_10822 = ctx_bambu_artificial_ParmMgr_Read(0u, 8u, 0u, _10821);
_10823 = buf + _10820;
_10824 = buf_bambu_artificial_ParmMgr_Read(0u, 8u, 0u, _10823);
__11155 = (bool)(((unsigned long long int)(_10820) >> 0LLU) & 1);
__11156 = (bool)(((unsigned long long int)(_10820) >> 1LLU) & 1);
__11157 = (bool)(((unsigned long long int)(_10820) >> 2LLU) & 1);
__11158 = (bool)(((unsigned long long int)(_10820) >> 3LLU) & 1);
_10825 = (((((__11158)<<3) | ((__11157)<<2)) | ((__11156)<<1)) | (__11155))&1;
_10826 = (_10931)<<(1u << 1);
_10825 = _10824 ^ _10822;
buf_bambu_artificial_ParmMgr_Write(1u, 8u, _10825, _10823);
_10819 = _10820;
if (_10826)
{
    __bambu_csim_trace(38438, 25, 0);
}
else
{
    __bambu_csim_trace(38438, 22, 0);
    goto BB_LABEL_22_0; /*goto7*/
}
```

M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

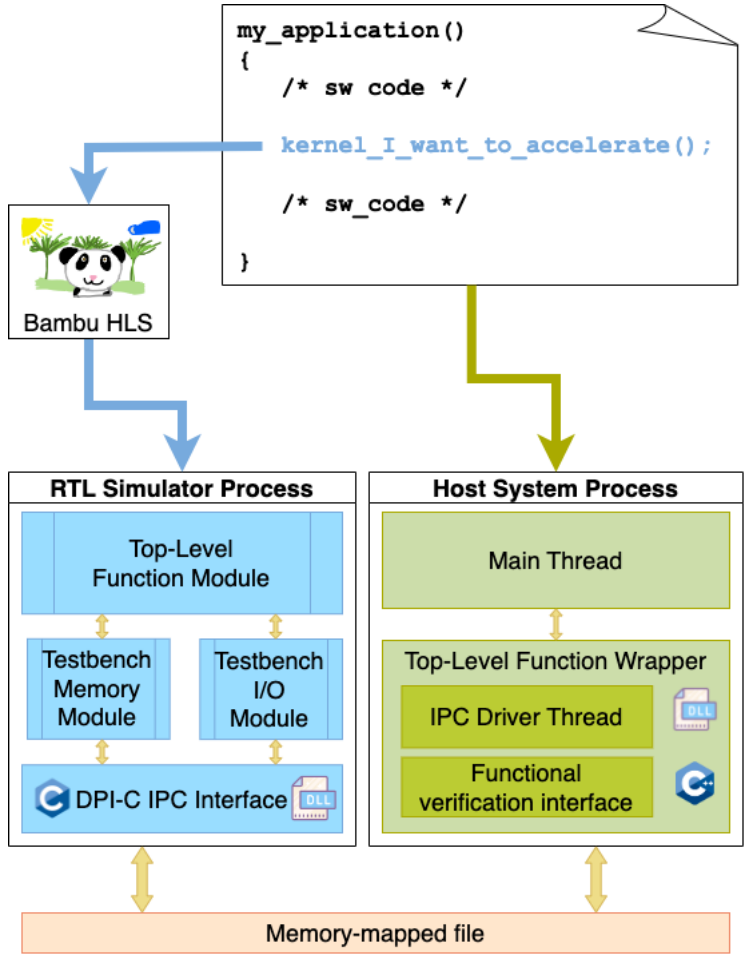
System-Level Verification

Key insight:

- Inter-Process Communication (IPC) to integrate host application and RTL simulation

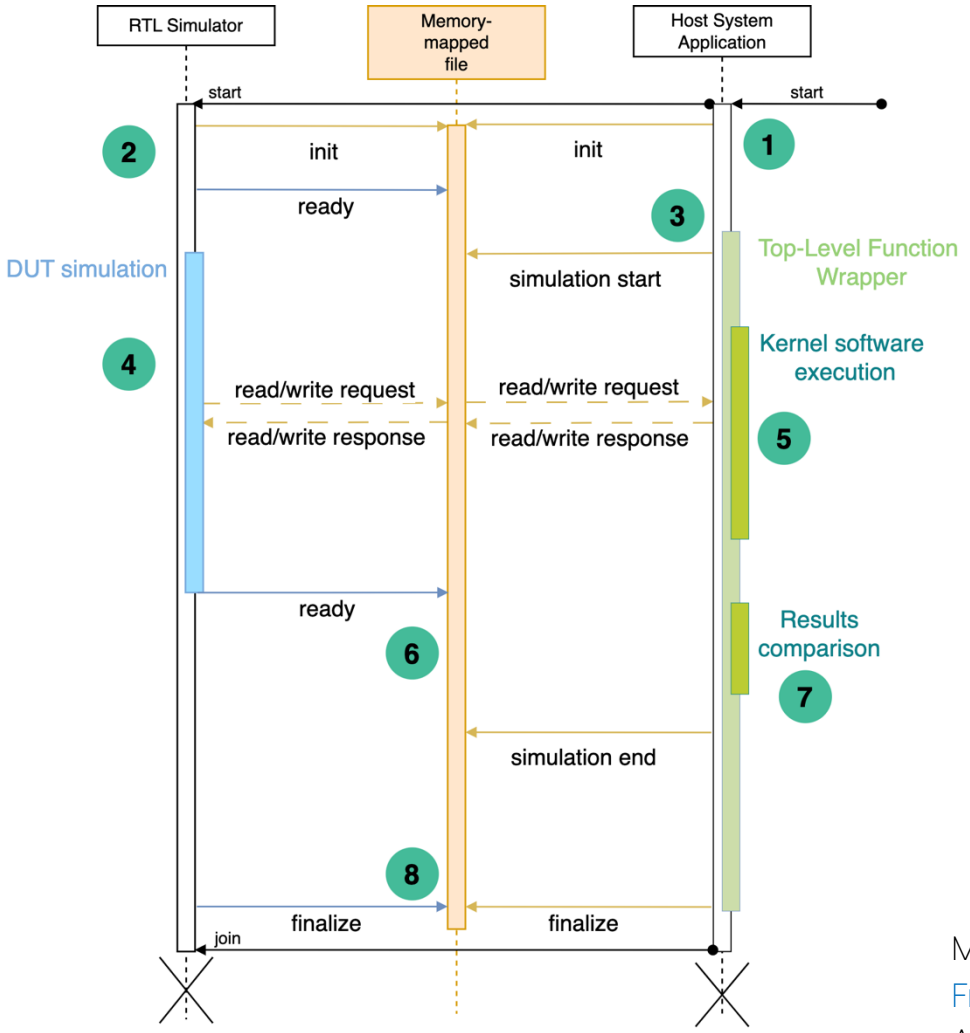
Our solution:

- Use the same application to
  - Extract kernels to be synthesized
  - Verify correctness of results
  - Simulate host-accelerator interactions
- Required infrastructure is automatically generated by Bambu
  - Kernel function replaced by wrapper communicating with simulator process
  - RTL testbench, I/O modules, DPI-C interface
  - Automated comparison against software execution



M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

System-Level Verification



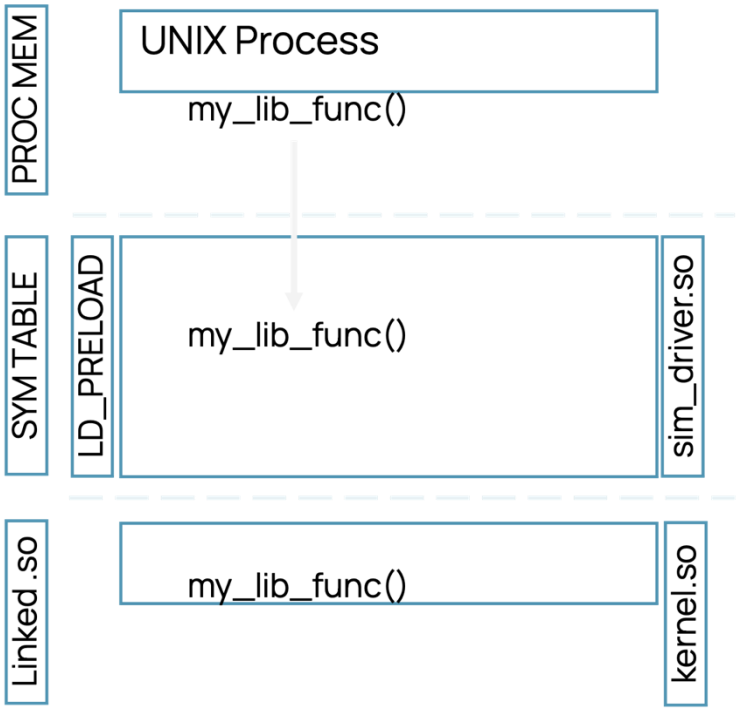
1. IPC environment setup, RTL simulator started as a child process.
2. RTL simulator synchronizes with the shared memory file, resets the DUT and waits for a request.
3. The host application runs until it calls the Top-Level Function that has been synthesized and requests the simulator to start.
4. The simulation runs the DUT, possibly interacting with the host to exchange data.
5. The host process runs the original software version of the Top-Level Function.
6. At the end of the DUT simulation, the FSM finalizes the execution and waits for the next request.
7. The host process compares results generated from the DUT simulation and from the software execution; if there are discrepancies, the co-simulation terminates with an error.
8. Finalization routines dispose of the generated environment and report timing information to the user.

M. Fiorito et al., POSTER: A System-level HW/SW Co-simulation Framework for HLS-generated Accelerator, Proceedings of the 22nd ACM International Conference on Computing Frontiers. 2025.

System-Level Verification

Bonus:

- Use the generated kernel wrapper to run co-simulation with an application written in a ***different language*** (must have C bindings)
  - The Top-Level Function wrapper replaces the original implementation
  - Dynamically linked executable looks for dynamic symbols at runtime
  - Shared object preload forces the system to initialize the symbols table of a process with given symbols
  - When the process looks for the dynamically linked symbol, the preloaded symbol is executed.



M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

Experimental results

Our work improves

- *Developer productivity*, e.g.:
  - No manual testbench development
  - Easier HLS debugging

Qualitative improvement

- Performance estimation *speed*

Needs to consider differences between generated accelerators

- Clock cycles reported by XSIM
- Slices as a proxy for design complexity
- Two metrics: Simulation Speed (SC/s) and Simulation Power (AC/s)

Experimental setup:

- MachSuite HLS benchmark kernels
- Bambu 2024.10 (modified)
- XSIM for RTL simulation
- Virtex7 FPGA target
- Comparison against:
  - LightningSimV2 (LSV2)
  - Vitis HLS 2022.2 cosim

M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

Experimental results

Comparison between HLS IR simulation methodologies in terms of performance estimation accuracy with respect to the RTL simulation (XSIM) baseline:

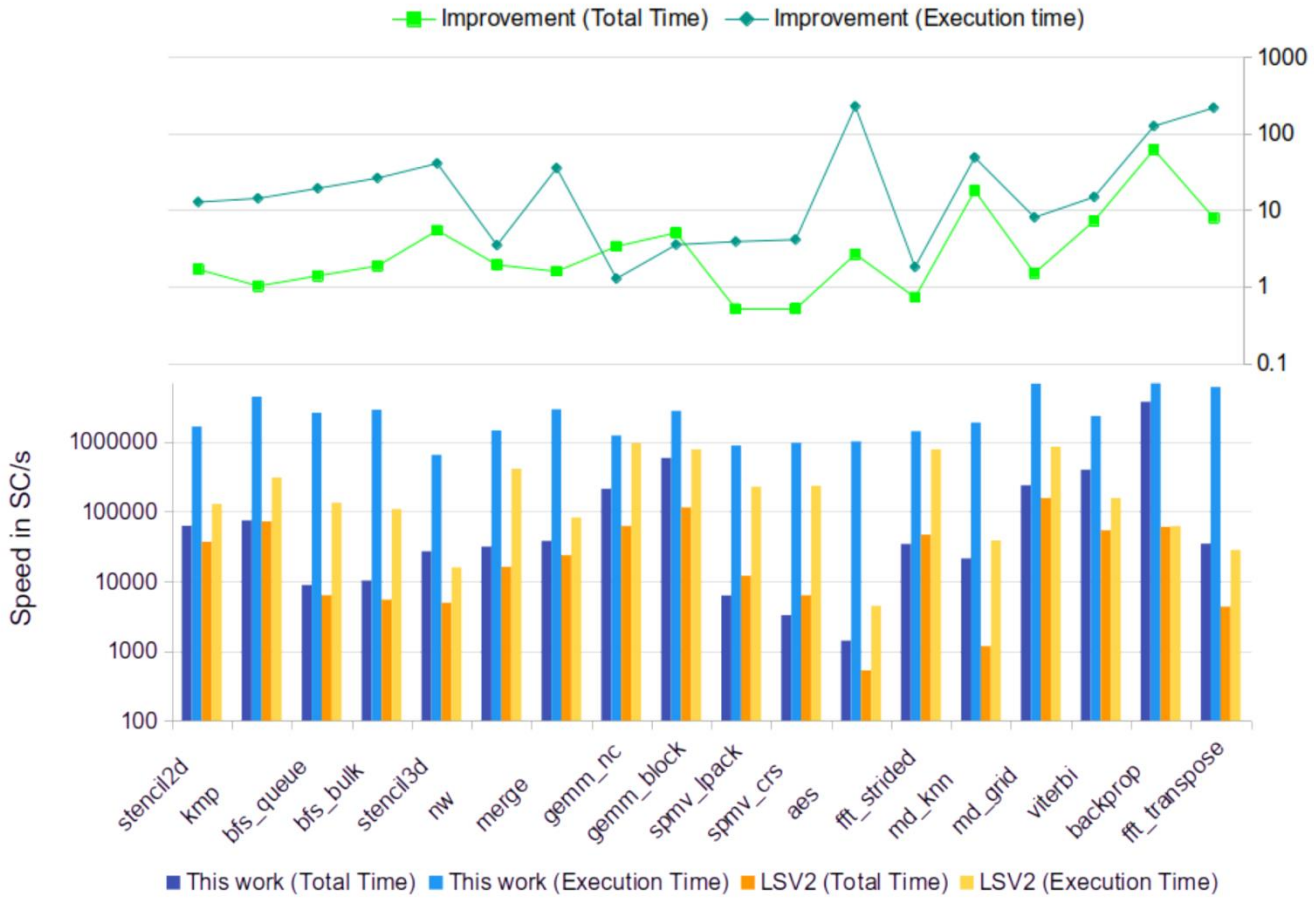
	LightningSimV2		*This work*	
Benchmark	Cycles	Error	Cycles	Error
aes	1230	-8.35%	3039	0.00%
backprop	4474188	0.00%	21932644	-3.49%
bfs_bulk	11965	-0.05%	20164	0.00%
bfs_queue	13687	0.00%	18248	0.00%
fft_strided	97301	0.00%	76822	0.00%
fft_transpose	12340	0.00%	104216	-3.46%
gemm_block	262154	0.00%	1642570	0.00%
gemm_nc	131366	0.00%	548930	0.00%
kmp	195362	0.00%	163981	0.00%
md_grid	395321	-0.28%	600020	0.00%
md_knn	2462	0.00%	54786	0.00%
merge	65560	3.22%	81926	0.00%
nw	33931	0.00%	67018	0.00%
spmv_crs	13087	-0.02%	6793	0.00%
spmv_lpack	25689	0.00%	13340	0.00%
stencil2d	101557	0.00%	132932	0.00%
stencil3d	22722	0.00%	55210	0.00%
viterbi	294447	0.00%	1202134	0.00%
Average	-	0.66%	-	0.39%

(Average among absolute error values)

- Consistently small errors
- LSV2 makes mistakes because it evaluates an IR before scheduling
- Our method makes mistakes because of slightly oversimplified assumptions about pipelining

M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

Experimental results



Comparison between our work and LSV2 in terms of Simulation Speed:

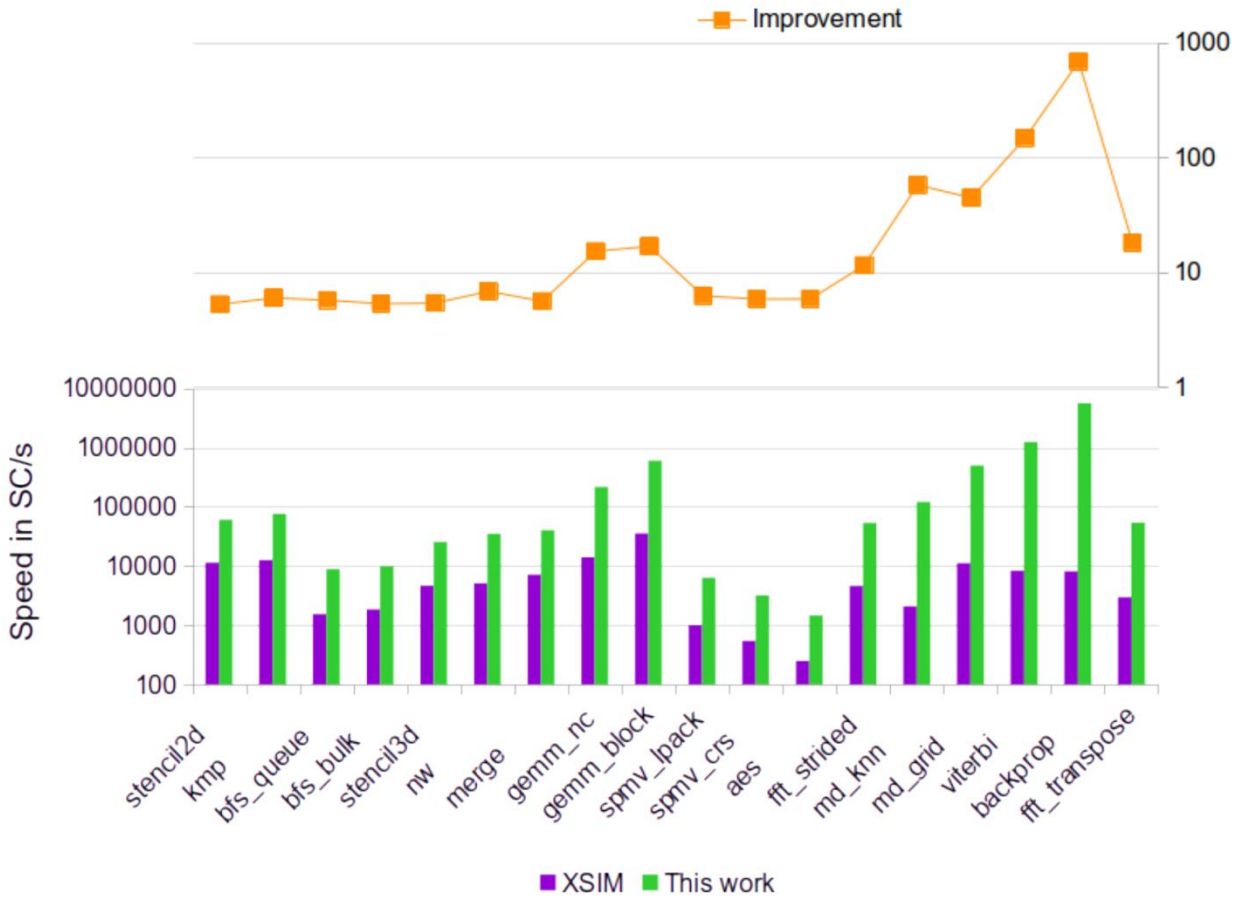
- *Execution Time* (run simulation, generate and parse traces)
- *Total Time* (Execution Time + compilation)
- Benchmarks ordered by area

Our method clearly outperforms LSV2

- Avg 7.0x, geomean 2.7x higher simulation speed
- Small kernels, less than 1s simulation time
- 1.8x faster Total Time and 15.4x faster Execution Time even considering raw simulation time

M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

Experimental results



Comparison between our work and RTL simulation in terms of Simulation Speed:

- XSIM only reports Total Time
- Benchmarks ordered by area

Our method is significantly faster than RTL simulation

- Avg 36.2x, geomean 10.9x higher simulation speed
- Advantage increasingly evident as complexity (~area) increases

M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

Experimental results

Comparison between our system-level cosimulation and Vitis HLS cosim:

- Benchmarks in bold have similar characteristics synthesized by Bambu HLS and Vitis HLS

	Total Time (s)			*Simulation Speed (SC/s)*			*Simulation Power (AC/s)*		
Benchmark	Vitis cosim	This work	Speedup	Vitis cosim	This work	Speedup	Vitis cosim	This work	Speedup
aes	32.82	12.28	2.67x	40.89	247.39	6.05x	0.82	1.04	1.26x
backprop	23727.90	2812.11	8.44x	188.56	8081.35	42.86x	56.89	866.08	15.22x
bfs_bulk	31.00	10.97	2.83x	386.16	1838.48	4.76x	0.72	3.59	4.99x
bfs_queue	30.64	11.91	2.57x	446.70	1532.71	3.43x	0.51	2.24	4.39x
fft_strided	153.89	16.73	9.20x	632.28	4593.17	7.26x	6.54	33.99	5.20x
fft_transpose	120.34	36.47	3.30x	102.54	2960.35	28.87x	14.42	892.10	61.87x
gemm_blocked	600.37	46.54	12.90x	436.65	35295.39	80.83x	3.03	142.59	47.05x
gemm_ncubed	712.70	39.12	18.22x	184.32	14032.79	76.13x	4.65	48.97	10.54x
kmp	32.62	13.14	2.48x	5989.03	12483.31	2.08x	8.32	14.36	1.72x
md_grid	808.32	59.58	13.57x	490.45	11078.39	22.59x	15.44	427.29	27.67x
md_knn	261.14	26.43	9.88x	9.43	2072.74	219.85x	1.30	75.92	58.31x
merge	31.15	11.56	2.69x	2038.94	7086.56	3.48x	4.69	21.61	4.61x
nw	30.27	13.23	2.29x	1120.94	5065.09	4.52x	3.40	13.83	4.07x
spmv_crs	69.15	12.59	5.49x	189.30	539.67	2.85x	0.75	2.06	2.76x
spmv_ellpack	81.34	13.32	6.11x	315.82	1001.64	3.17x	1.26	3.85	3.05x
stencil2d	32.10	11.75	2.73x	3163.77	11314.17	3.58x	1.30	4.30	3.31x
stencil3d	41.64	11.87	3.51x	545.68	4651.38	8.52x	58.39	6.93	0.12x
viterbi	363.63	144.78	2.51x	809.74	8302.91	10.25x	75.57	760.55	10.06x
Average	1508.946	183.575	6.189x	949.512	7343.193	29.505x	14.333	184.516	14.789x
Geometric mean	129.439	26.879	4.816x	386.309	4065.859	10.525x	4.362	26.371	6.046x

Our method outperforms Vitis HLS cosim

- Vitis cosim is closed-source, so unclear where exactly the advantage comes from

M. Fiorito et al., Augmented Co-Simulation for Fast Functional and System-Level Verification of HLS Accelerators, 44th IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2025.

Conclusions

Limitations of current verification methods for HLS-generated accelerators:
- Trade-off between **speed and accuracy**
- Limited **automation**

What we propose:
- Simulation of **C model derived from HLS IR** after all optimizations have been applied
- **System-level** HW/SW co-simulation with all infrastructure automatically generated

What we achieved:
- C simulation: average **7.0x improvement in simulation speed** compared to state-of-the-art tools
- System-level co-simulation: average **6.2x reduced simulation time** compared to commercial tools
- Increase in **productivity** for HLS users and developers

Next steps:
- Fix discrepancies in accuracy
- Apply to larger designs
- Include coverage measurement

POLITECNICO MILANO 1863 | DIPARTIMENTO DI ELETTRONICA INFORMAZIONE E BIOINGEGNERIA

Thank you!

Contacts

{michele.fiorito, serena.curzel, fabrizio.ferrandi}@polimi.it
www.polimi.it
https://panda.deib.polimi.it/
https://github.com/ferrandi/PandA-bambu