



January 21-23, 2025
Santa Clara Convention Center
ChipletSummit.com

Using Federated Simulation as a Framework for Chiplet-Based Design

Kevin Cameron, EDA Consultant

About Kevin Cameron

- Shifted from analog circuit design to EDA in the 1980s
- Spent the 1990s working on (parallel) simulation tools and (AMS) languages
- Shifted to design verification in 2000s
- Moving back into analog (DV) 2014-
- Now attempting to automate (analog) IC design with AI

What is Federated Simulation?

- Using multiple simulators together to handle different pieces or aspects of a design
- Different domains handled by specialist tools
 - **Thermal** power, heatsinking
 - **Mechanical** back-annotation, heatsinking
 - **Electrical** circuits, power, photonics, RF

Motivation for Federated Simulation (with AI)

- Does not require a simulator that does everything
- Minimal extensions to existing simulators
- Can handle some problems at the DevOps/OS level
- Potentially higher performance through parallel processing/HiL
- Falling number of HW designers, increasing complexity
 - more automation needed, AI can help
 - divide and conquer, an “agentic” approach

Legacy Problems

- Batch processing tools from EDA vendors
- Poor integration of tools
- Net-listing
- No support for new logic/power handling in digital
 - Asynchronous, Quasi-adiabatic, DVFS
- Poor support for RF, PMIC, Chiplets & Systems
- Low precision simulation (1/0) doesn't suit AI

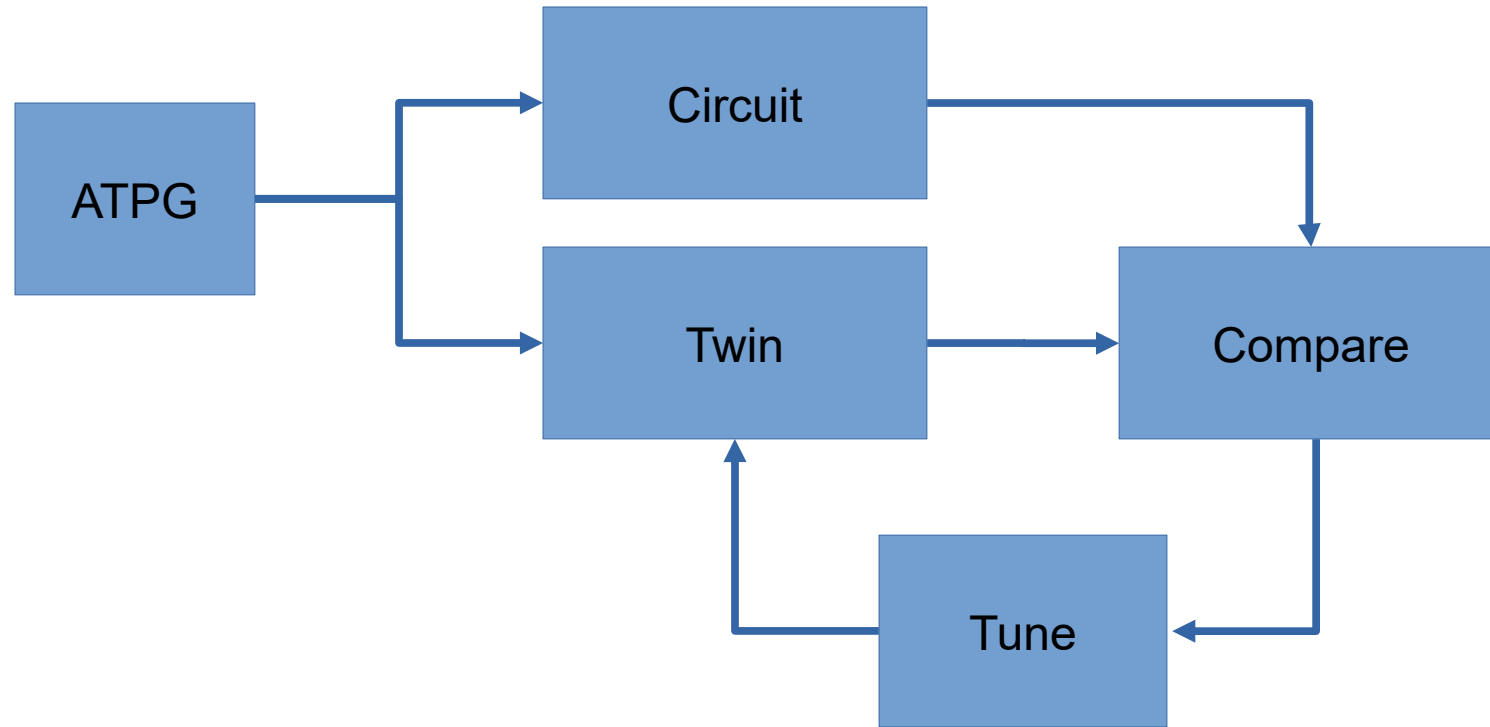
What does AI for design need?

- Fast verification of experiments
- A lot of experiments
- Digital Twinning Frameworks
- Specifications (models) and/or test-benches

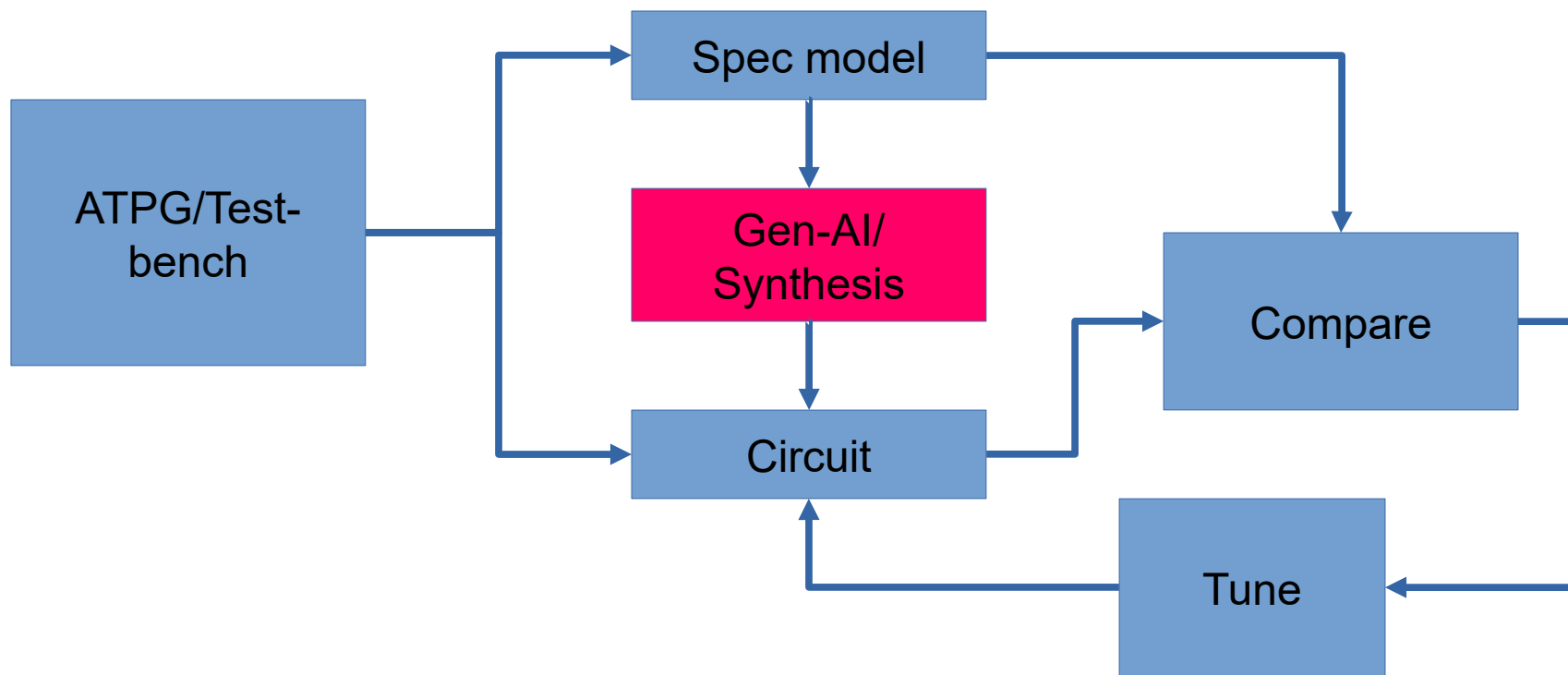
Digital Twinning

- Making one thing look like another
 - SPICE to Verilog-AMS
 - Synthesis -
 - Verilog-AMS to Schematic
 - C++ to Gates
- Methods
 - Gradient descent optimization
 - Generative AI

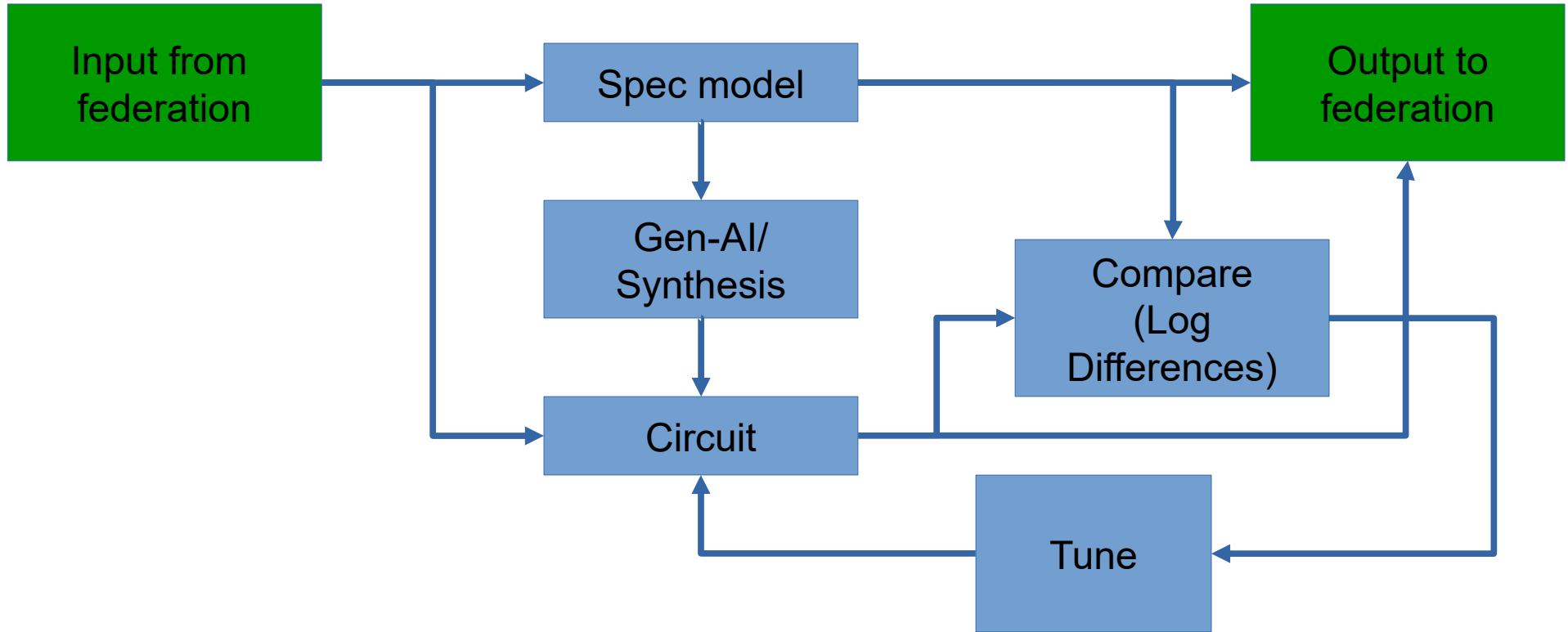
Simple Digital Twinning



Digital Twinning for Synthesis



Twinning Environment as Simulation Component



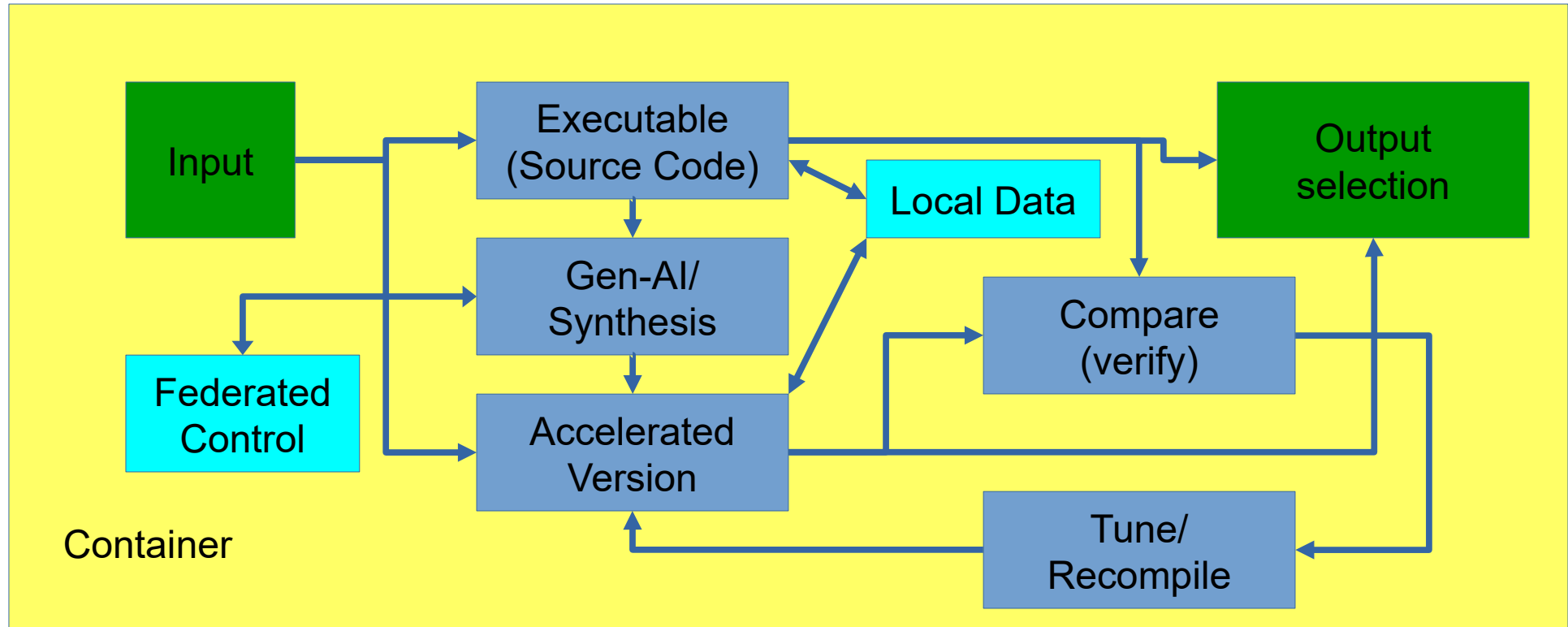
Traditional Hardware Design

- Loose English specs
- Bottom-up/middle-out with RTL and SDKs
 - Multiple issues tackled in parallel
- Emulation for software development

Hardware from Software

- ***The code is the spec, the workload is the test-bench***
- Code comes at the granularity of routines and shared libraries
- Every routine can be considered a separate “computing context” in a neural-network -like graph
- Routine call/return can be viewed as message-passing
- Parallelize first, then accelerate the pieces

Twinning Environment for SW Component



HW/SW Co-design Methodology

- Starting on X86/ARM, target heterogeneous mix of:
 - DSP, GPU, Extended RISC-V, FPGA, ASIC
 - AI Accelerators
- Compile and analyze per routine.
- Drop tightly coupled routines into same container
- Map containers to Chiplet-accelerators/ICs/soft-IC
 - Chiplet vendors provide tailored versions of shared libraries

Communication Design

- Evaluate communication cost between containers (SDN), assign logical channels (between IPV6 end-points) to physical channels
- Simulate physical channels similarly to HW twinning
- Trade off latency/throughput/power
- Group containers with high cost together

Federated Design

- Adjacent container groups can trade members during design/synthesis
- Optimize for:
 - Power
 - Communication
 - Cost
 - Yield

Runtime Optimizations

- Dynamic migration of containers
 - cloning/redundancy
- Short circuit IPv6 for static channels
- Neuromorphic code
 - adapt to workload
- Scaling
 - move containers as size/conditions change

Conclusions

- Multi-domain federated simulation will be used for verification
- Federated simulation will use AI digital twinning for acceleration of hardware and software
- AI synthesis is just a reversal of the twinning process using gen-AI and tuning

Q & A



Implementation of an FS Framework

- Objects need hierarchical names and addresses
 - IPv6/DNS in virtual network
- C/C++ APIs and/or micro-services
 - May need to cross OS boundaries
- Handle conversions outside slave simulators
 - Simulators attach drivers and receivers to the network objects
- Speculative scheduling
 - Check-pointing and back-tracking
- Drop and restart members (pluggable like the real world)