Fast (near-) cycle-accurate SystemC Simulations

Oct 2004

Cor Schepens
Email: Cor@adveda.com

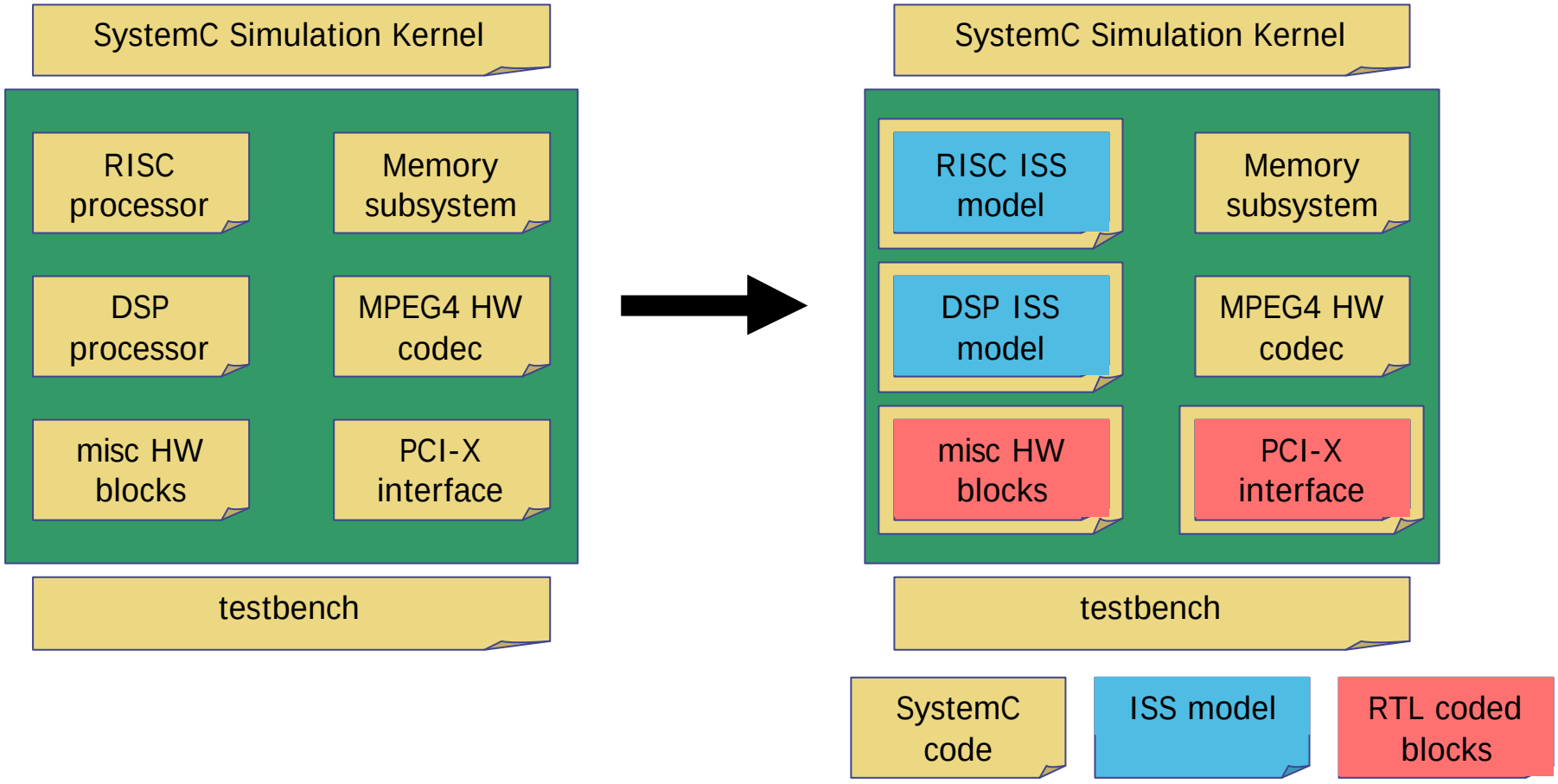
Why use SystemC?

- SystemC is a modelling language
- GOAL:
  - ✓ High Speed System Level simulation and analysis
- REFINEMENTS:
  - ✓ Different analysis @ different speeds
  - ✓ Identify system bottlenecks
  - ✓ Verify real-time system requirements
  - ✓ Systems are a combination of embedded software and hardware
- Demand for HW/SW Co-Verification on Rise

(near-) cycle-accurate SystemC simulations

- Final verification of System
- GOAL:
  - ✓ Verify real-time requirements
  - ✓ Identify momentarily bus bottlenecks
  - ✓ Identify slow software or slow hardware
  - ✓ Identify interaction mismatches
  - ✓ Identify HW/SW interfacing issues
- REQUIREMENTS:
  - ✓ much faster than RTL simulation
  - ✓ (near-) cycle-accurate models of (most) blocks of the System
  - ✓ Good debugging of both embedded SW and HW

SystemC project example

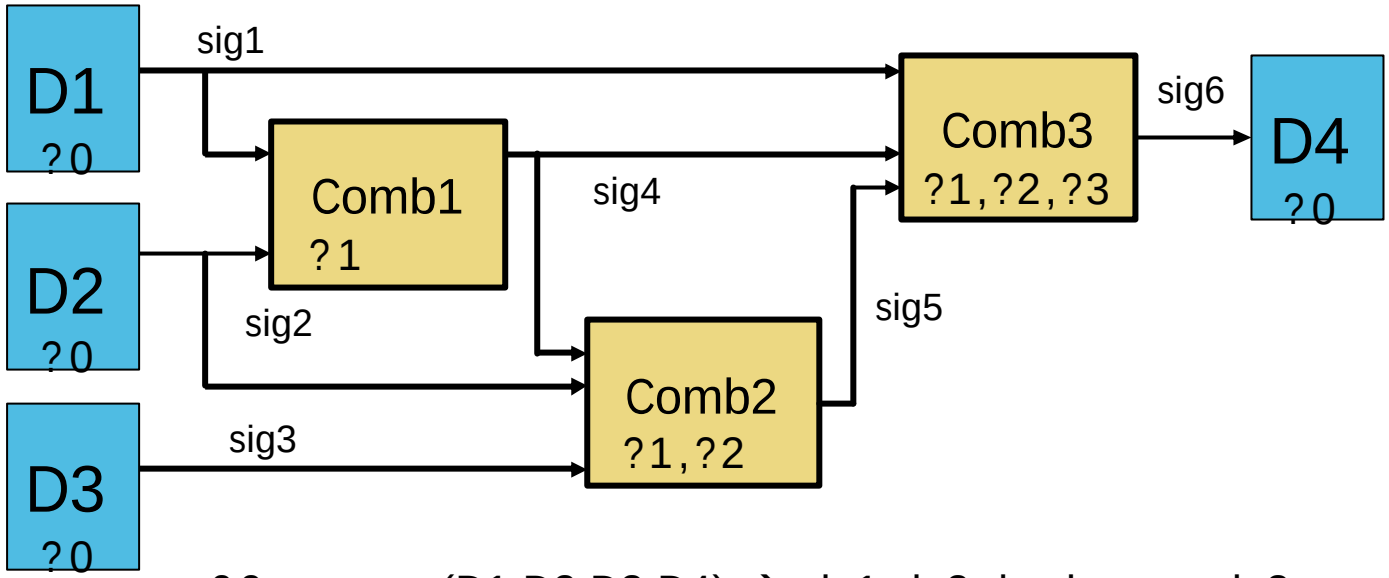


- Create fast cycle-accurate model
  - Minimize modeling effort
  - Trade-off simulation speed versus modelling effort

How to create faster HW simulations?

- Need to use a higher abstraction layer of RTL model
- Traditional simulators based on gate-level technology
  - ✓ Event-driven, Include timing (picoseconds)
  - ✓ 9-state (VHDL) or 4-state (verilog) logic models
- Simulator should be RTL-only
  - ✓ Cycle-based
  - ✓ 2-state logic models
- Build special precautions for:
  - ✓ Multiple asynchronous clocks
  - ✓ Tri-state signals
  - ✓ Undefined values when reading memory or external signals
- Uses full synthesizable RTL syntax
- Results in upto **100 TIMES** speed increase
- This technology is used by Univers Modeler

Event-driven simulation



?0: process(D1,D2,D3,D4) → sig1,sig2 do change, sig3 no change

?1: process(comb1,comb2,comb3) →sig4 does change, sig5, sig6 no change

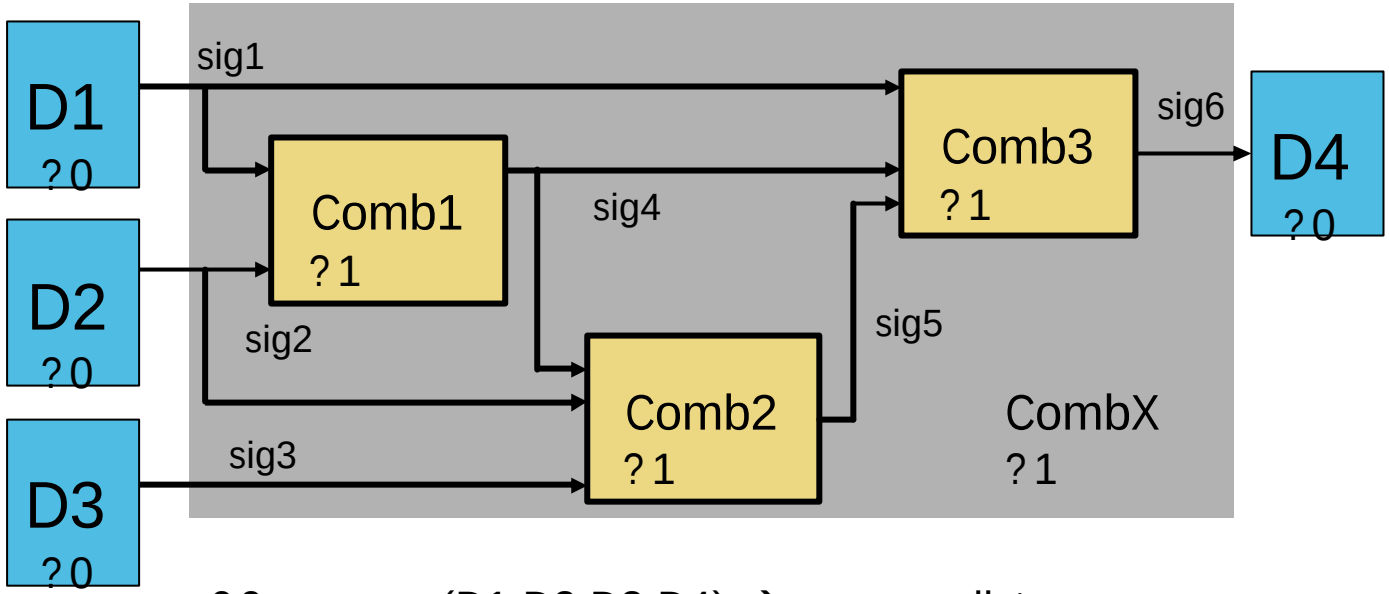
?2: process(comb2,comb3) →sig5,sig6 do change

?3: process(comb3) →sig6 no change

NOTE1: comb3 can be executed 3 times per clock cycle

NOTE2: Signal and sense list evaluation and event scheduling take a lot of time in an event-driven simulator.

Cycle-based simulation



?0: process(D1,D2,D3,D4) →no sense list

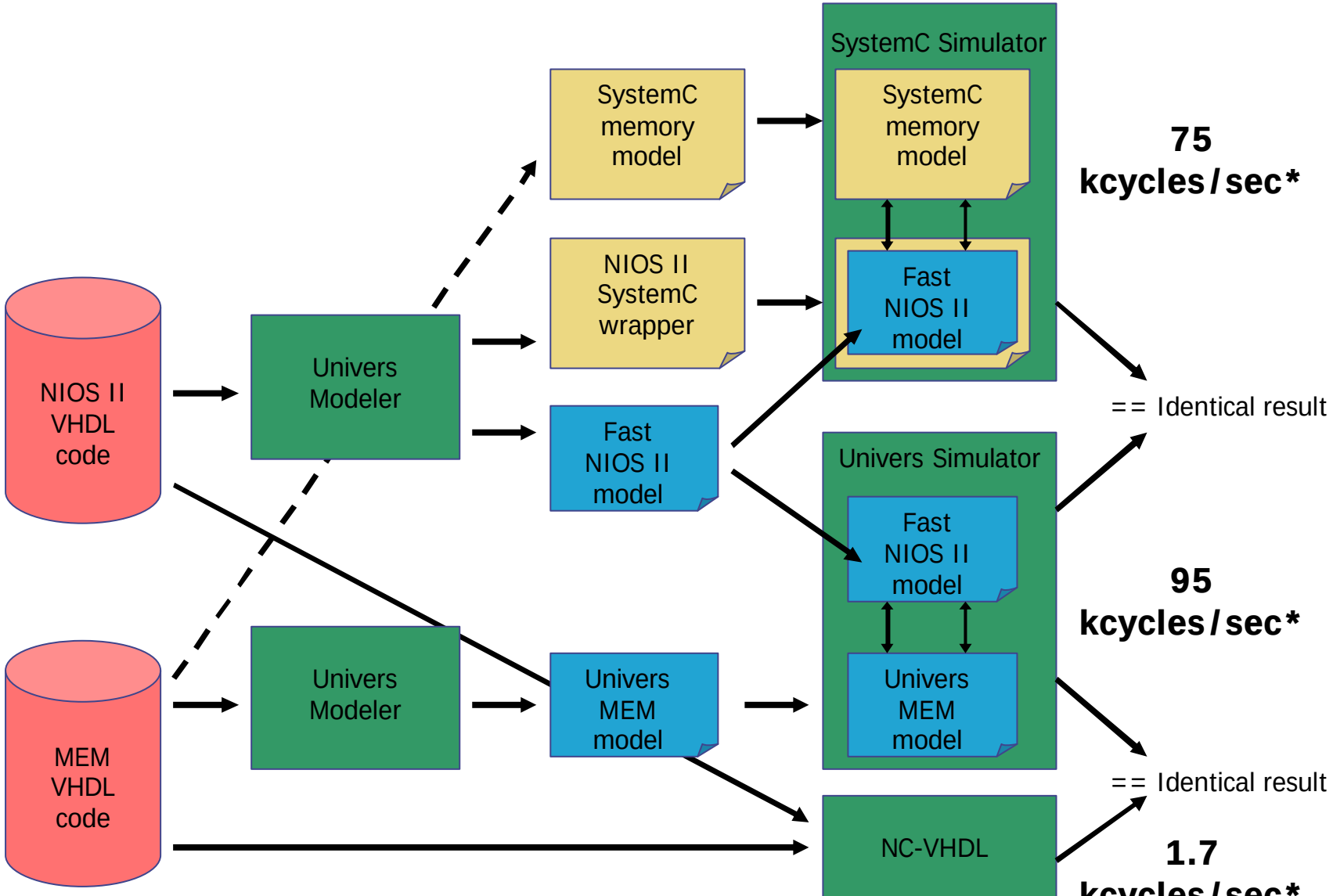
?1: process(comb1), process(comb2), process (comb3)

NOTE1: The order of the combinatorial blocks are compile-time defined

NOTE2: **NO** signal and sense list evaluation and **NO** event scheduling

NOTE3: always executes combinatorial logic once per cycle

Univers Modeler: NIOS II example



* = Benchmark run on 1.5GHz Pentium4M

Customer Example:

- Handmade DDR-SDRAM controller SystemC model:
  - ✓ First SystemC model of this engineer
  - ✓ 3 manmonths
  - ✓ Result: still not cycle-accurate
  - ✓ Speed: 3x over RTL
- Automatic SystemC Generation with Univers Modeler:
  - ✓ First usage of this tool: Install, Play, Use, simulate, verify
  - ✓ <1 day
  - ✓ Result: 100% cycle-accurate
  - ✓ Speed: 79x over RTL

Efficient coding styles for simulation speed

- Minimize the number of ‘events’:
  - ✓ Only transactions create an event
  - ✓ For cycle-accurate, clocks do trigger transactions
  - ✓ Try to be smart in reducing the events
  - ✓ Make sure that events that happen most often are dealt with in optimized way.
- Carefully select datatypes:
  - ✓ Avoid sc_lv, sc_bv etc.
  - ✓ Use int or unsigned int, whenever possible
  - ✓ Use ANSI-C or ANSI-C style whenever possible

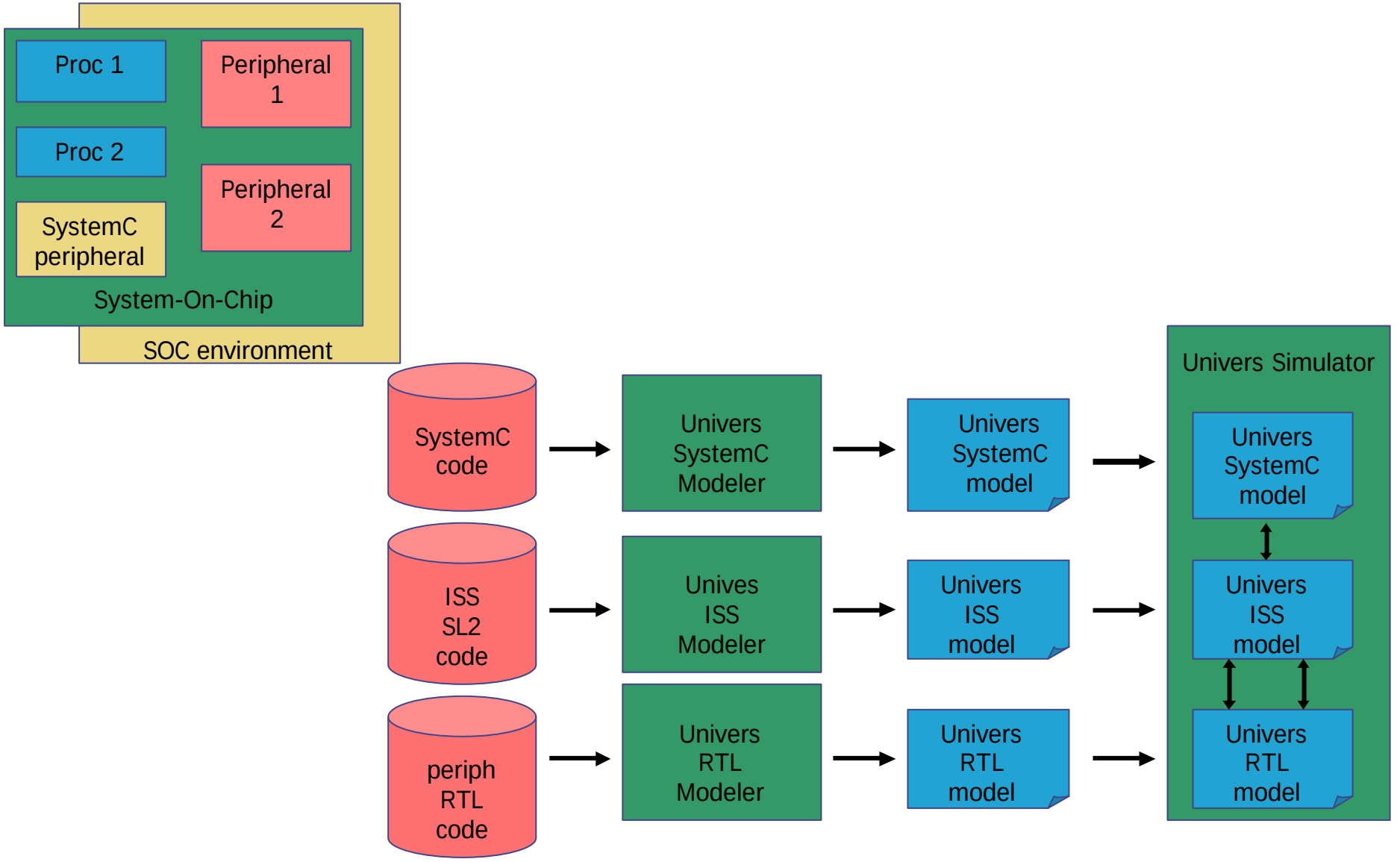
Translation example

- VHDL: 4-state: array of bits, event-driven
  - ✓ signal A,B,C: std_logic_vector(19 downto 0);
  - ✓ A <= B + C;
- SystemC: 2-state: class-defined widths, event-driven
  - ✓ sc_bv<20> A,B,C;
  - ✓ A = B + C;
- ANSI-C: 2-state: integer-mapped, cycle-based
  - ✓ unsigned int A,B,C;
  - ✓ A = (B + C) & 0x000FFFFF;

Conclusions:

- Usage of SystemC at different levels
- When you write models manually:
  - ✓ Minimize events
  - ✓ Use integers when possible
- Use automatic modeling tools when possible
  - ✓ Reduces the modeling effort
  - ✓ Increases the reliability of the model
  - ✓ Provides maximum cross-correlation with RTL
- Possible for:
  - ✓ Legacy blocks (also in earlier high level simulations)
  - ✓ bottom-up instead of top-down
  - ✓ Final system-level functional verification

Univers with SystemC support: Future



Screenshot of multi-core+RTL project