LINXS: Using SystemC to Enable HW/SW Co-Simulation in the Real World

Luis Baldez, Juanjo Noguera, Narcis Simon, Lluis Abello
Hewlett Packard Barcelona, ICD division
ESCUG, September 2005

Motivation

- Benefits of HW/SW Co-Simulation are very clear
  - Enable earlier interaction between HW and SW
  - Enable SW code reuse for HW verification (or vice-versa)
  - Enhance functional coverage with “real world“ tests
- Need to introduce co-simulation into the HW and SW design flows
  - HW and SW teams must be fully committed to deliver results

Requirements from HP

- Minimal change to SW development environment
  - Transparent connection to “real” or “virtual” HW
  - SW tools, compilers, debuggers remain the same
- Solution independent of target CPU
  - Must work with different CPU configurations: no CPU, embedded CPU(s), external CPU(s)
- Acceptable performance of the simulation environment
  - Fast enough to enable SW development
  - Provide alternatives to improve performance

Commercial Options Analyzed

	Speed	Cost	CPU Choices	SW Changes	HDL Changes
FPGA Prototype	10-100MHz	10-100k$ + 4-8man /month	Many	Medium	High
C Model	10-100kHz	50-100k$ +2-4man /month	Many	Low	High
ISS with Simulator	1-10kHz	10-50k$ +1man /month	Few	High	Low

More on the Motivation

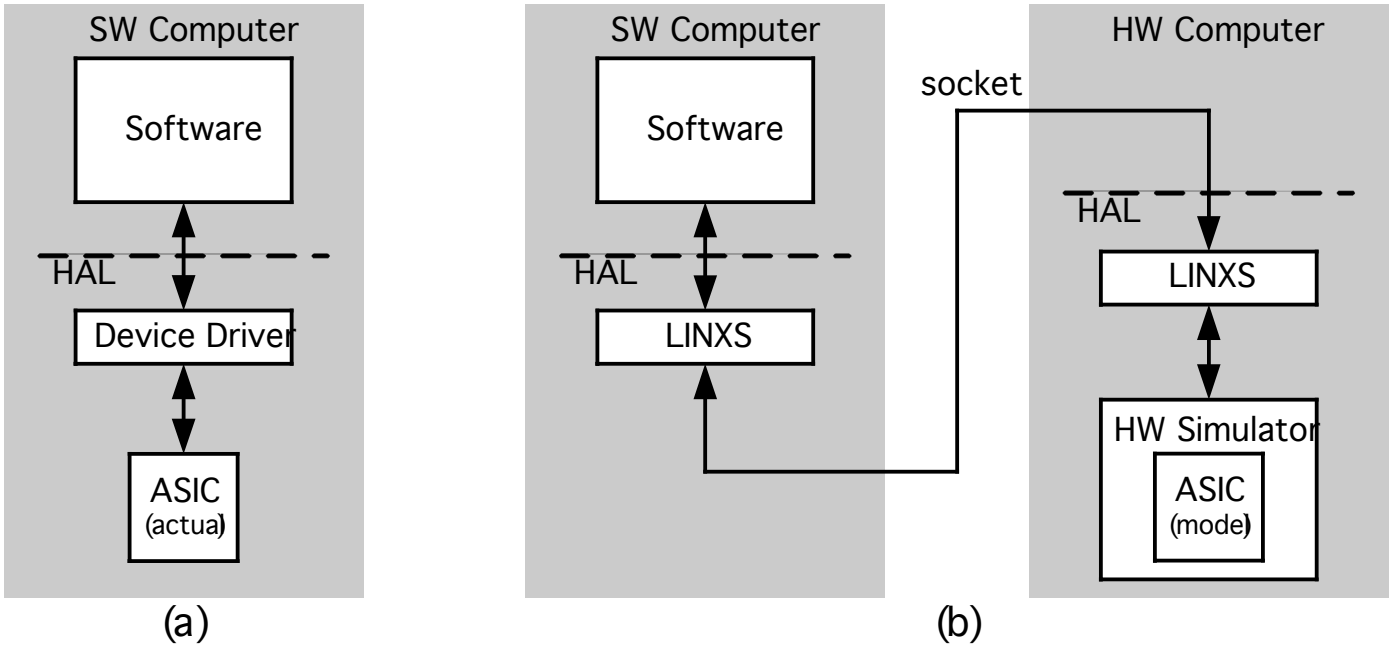
Or, why commercial options do not meet our requirements?

- Very disruptive on the SW side
  - Lots of change on SW code
  - New tools, new environment
- Very disruptive on the HW side
  - Lots of change on HW code
  - High maintenance cost

"Need a new approach that justifies the ROI!!"

-My Manager

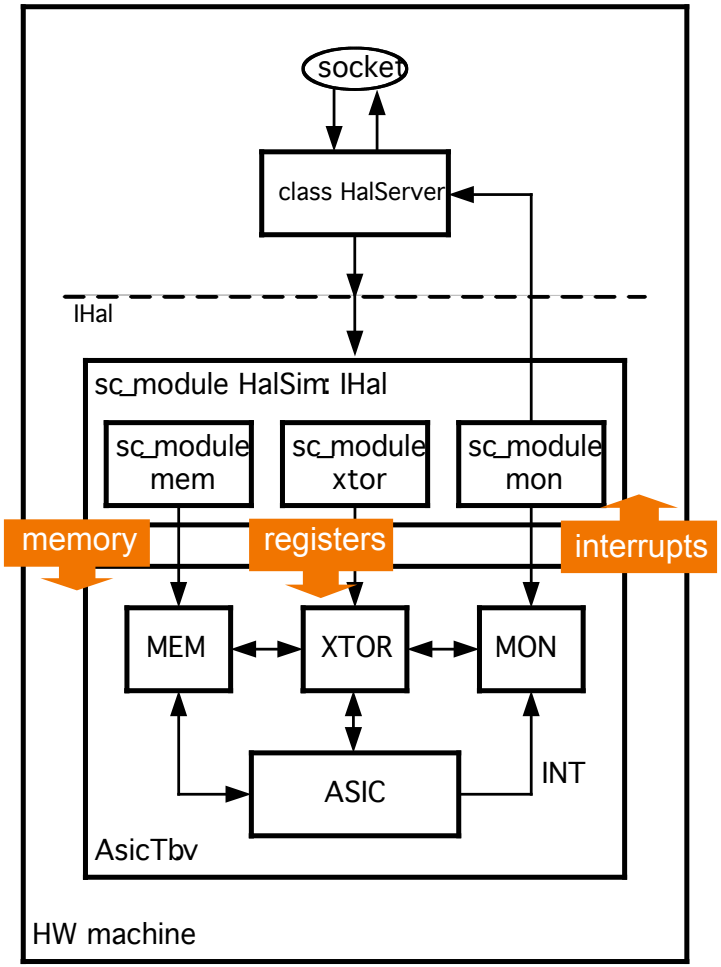
LINXS at a Glance



	Speed	Cost	CPU Choices	SW Changes	HDL Changes
LINXS	**1-10kHz**	**1man /month**	**Many**	**Low**	**Low**

Zoom in HW Machine

- HalServer receives register/memory requests and sends interrupt events
- HalSim handles requests and events inside its sub-modules:
  - XTOR gives transaction access to registers
  - MEM gives “back-door” access to memory
  - MON monitors interrupts
- Everything running inside a mixed SystemC-HDL simulator

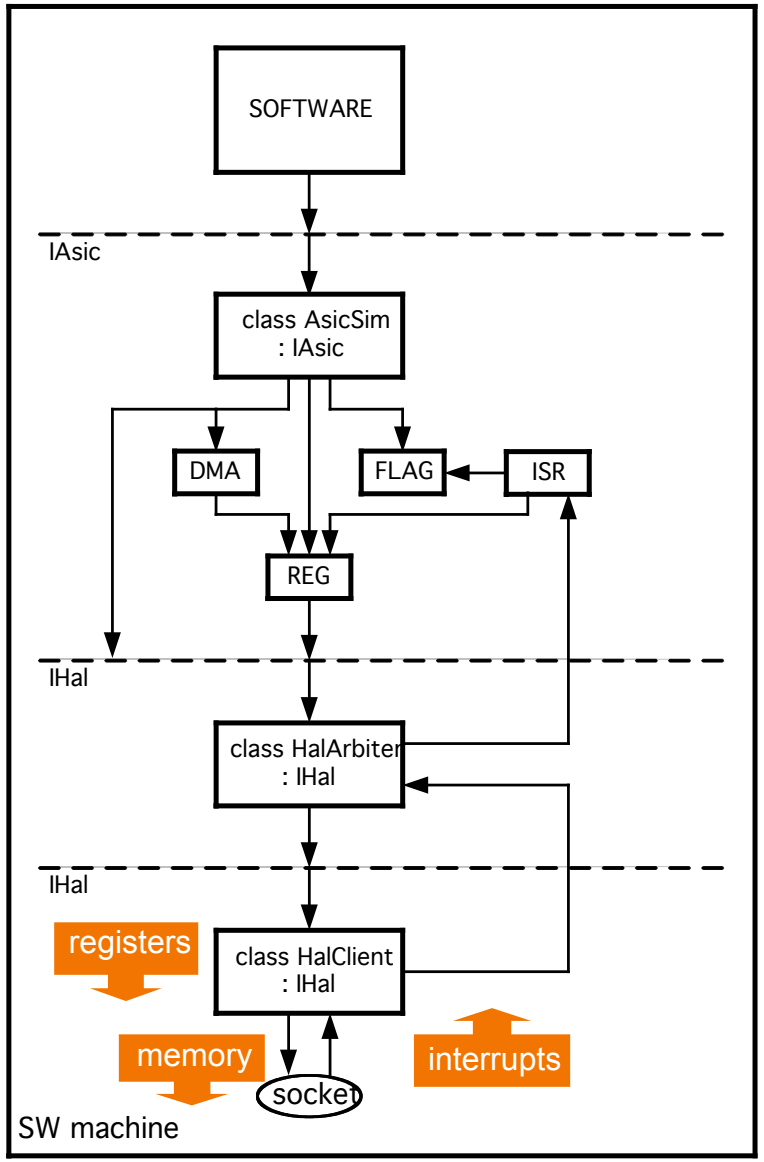


No changes are required on the ASIC model or testbench

Zoom in SW Machine

- Software is a multi-threaded, huge and C++ application
- IAsic is the interface SW uses to access registers, memory and interrupts
- HalArbiter arbitrates between threads and interrupt service routines
- HalClient requests register/memory accesses and receives interrupt events

Everything below the IAsic interface is transparent to SW



SystemC in LINXS

- Standard language supported by many vendors
  - Solution is vendor-independent
- Easy integration with HDL models
  - Able to access signals, memories and tasks
- Easy integration with Linux libraries
  - Berkeley socket I/O implementation
- Improve performance with higher-level SystemC models of HW

Case Study: Large-Format Printer

- HW machine: ASIC reused from real product
  - PCI interface, SDRAM controller and several image processing blocks
  - Cadence IUS5.3 in Linux PC
- SW machine: application reused from real product
  - Multithreaded code that sends images to printheads
  - Pentium CPU in Linux P

	Simulation Time (seconds)	
Data Size	**HW and SW running in the same machine**	**HW and SW running in different machines**
1Kbyte	1.02	0.42
10Kbyte	2.95	2.51
100Kbyte	23.66	22.61
500Kbyte	115.15	112.30

LINXS Advantages on the SW side

- Only change on the IAsic implementation, everything else remains the same
- SW engineers keep their own environment, just need to add a C++ module to his makefiles
- Very accurate model of the HW is available early in the project, no need to write stub code
- Able to run the entire code (not just parts of it)
- Code must be ported to PC, but this is already part of their development flow
- Performance is acceptable and can be improved

LINXS Advantages on the HW side

- Just need to add a SystemC wrapper on the top of the top-level testbench
- ASIC is tested with real testcases, improving verification coverage
- Complete visibility of HW nodes for debugging
- Compatible with other verification strategies
- No extra cost apart from the simulator licenses

Conclusions

- HW/SW co-simulation tools available in the market require many changes on the current HW/SW environment and source code
- “Real world” users will not completely change their environment – it’s too risky and time consuming!
- LINXS and SystemC enable HW and SW engineers to maximize the benefits of HW/SW co-simulation by
  - Decoupling the HW and SW machines with sockets
  - Minimizing the changes on environment and code
  - Minimizing the acquisition and maintenance costs

Thank You!

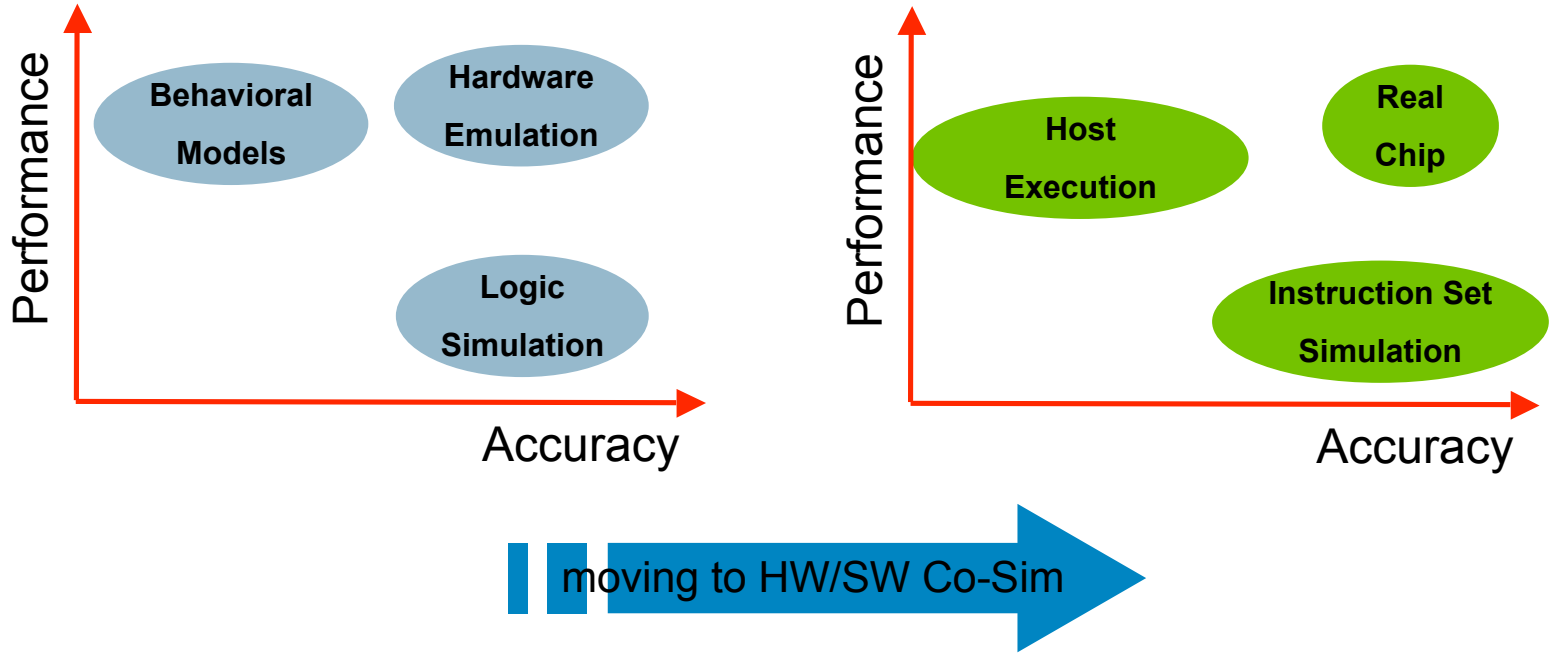

invent

Backup Slides

Future Steps

- Implement final versions of HAL components with advanced features
- Enable system engineers to interact with HW model
- Improve performance by writing high-level models or acceleration boxes

HW/SW Validation Options



- Emulation is a good option when it is representative of real HW
  - Cost and complexity are usually high
- If this is not the case, host code execution becomes a better option