




VIPPE: Parallel simulation and performance analysis of complex embedded systems

L. Diaz, E. Gonzalez, E. Villar, P. Sanchez
University of Cantabria



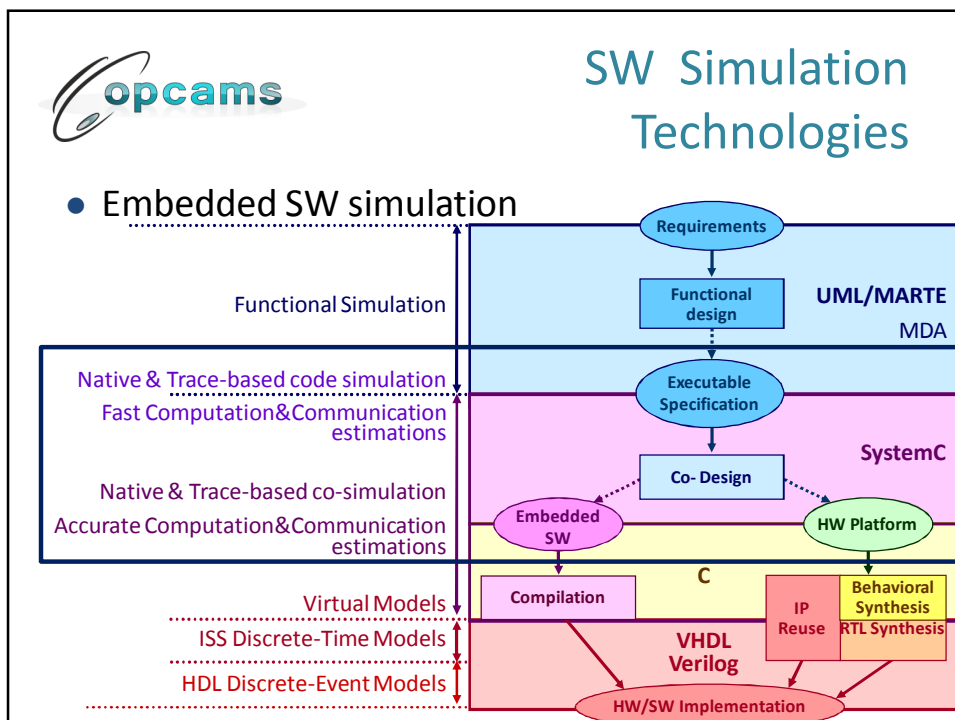
Motivation

- The MPSoC
 - Multi-processing platform
 - ASIC
 - FPGA
 - Commercial multi-processing platform
 - SW-centric design methodology
 - Most of the functionality implemented as Embedded SW
 - With 'some' application-specific HW
 - Functional and extra-functional verification
 - Simulation & Performance analysis
 - Virtual HW/SW platform
 - SW integration



State of the art

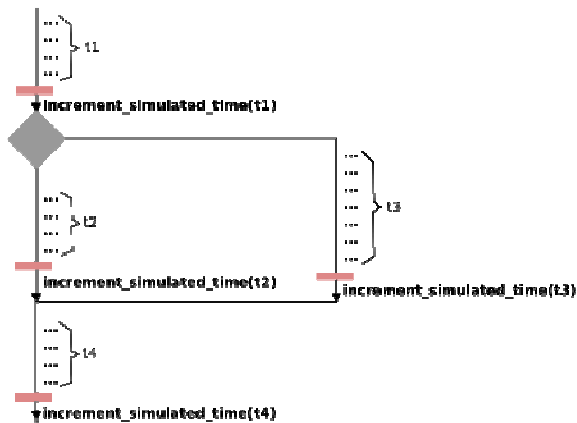
- Functional Simulation
 - Fast but inaccurate
- ↑
- Native Simulation
 - Fast and accurate enough
- ↓
- ISS
 - Accurate but (even JITC) too slow
- RTL
 - extremely slow
 - only valid for final verification





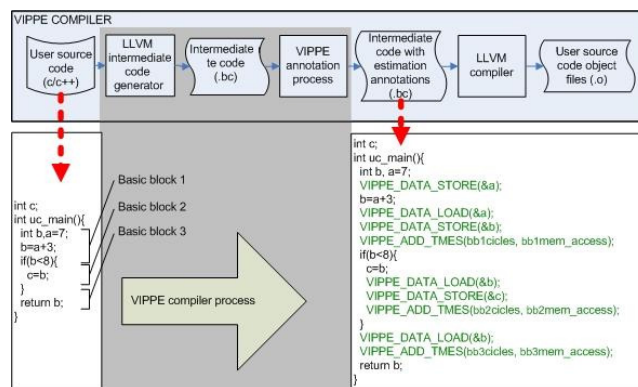
Native simulation methodology

- Based on basic blocks



VIPPE annotation process

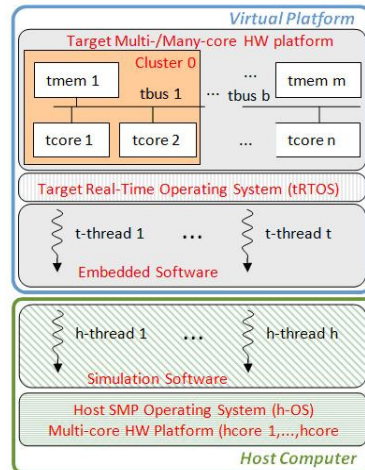
- Native simulation





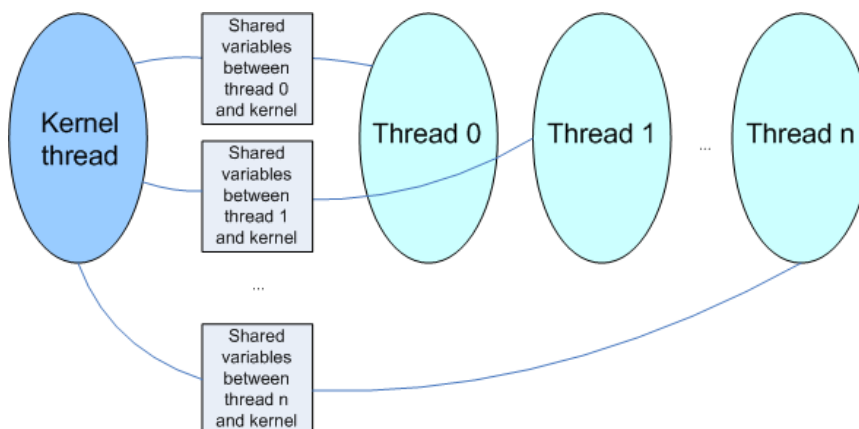
VIPPE Kernel

- Each target thread modeled as a host thread
- Target RTOS as an additional thread
- No need for thread locking
- Threads locking
 - shared variable
 - synchronization element
 - i.e. semaphore



VIPPE Kernel

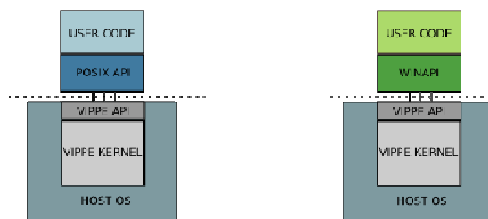
- Informing the kernel thread about their estimations of individual thread times, access to memory, etc. via shared variables.





VIPPE API

- Optimal reduced set of primitives
 - Kernel services
- OS API implementation



VIPPE API

MANAGEMENT OF PROCESSES	MANAGEMENT OF TIMES
process_create	time_real_watch time_user_watch
MANAGEMENT OF THREADS	MANAGEMENT OF SEMAPHORES
get_prio set_prio get_id thread_create thread_exit thread_delete thread_wait_for_end	semaphore_create semaphore_wait semaphore_post semaphore_watch
MANAGEMENT OF SIGNALS	MANAGEMENT OF AFFINITIES
vippe_signal vippe_kill	uc_PE_mapping uc_PE_dismapp uc_take_OS_id



POSIX API

- POSIX API
 - Events
 - Concurrency
 - Synchronization
 - Timing and I/O

POSIX API

Events	vippe_signal vippe_kill
Concurrency	process_create thread_create thread_delete thread_wait_for_end get_id get_prio set_prio
Synchronization	semaphore_create semaphore_wait semaphore_post semaphore_watch
Timing	time_real_watch time_user_watch
I/O	semaphore_wait semaphore_post



POSIX API annotation

- POSIX API
 - Annotated in the same way as user's code
- VIPPE API
 - Annotated at function-level
 - Execution time and power depending on target processor



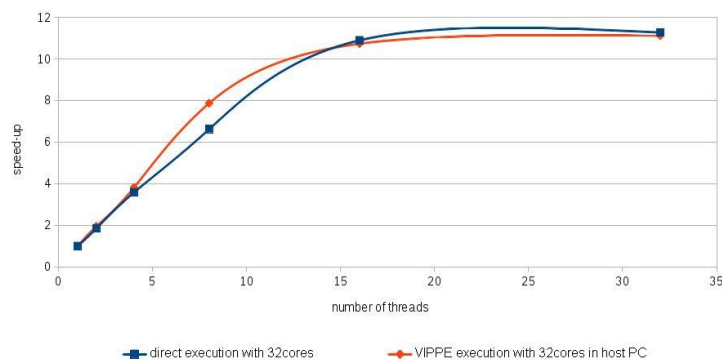
Experimental results

- Evaluated with the **PARSEC** benchmark suite
 - Princeton Application Repository for Shared-Memory Computers
- Host
 - 8 Intel Xeon E5-2687W at 3.10GHz
 - Each processor has 8 cores, thus the host platform integrates 64 cores with SMP capability
 - 64Gb of RAM and 20Mb of cache



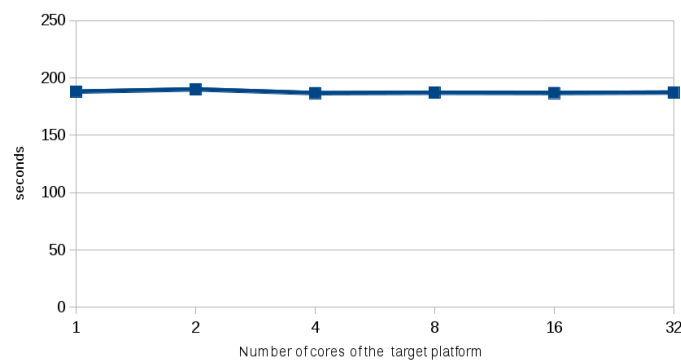
Speed-up vs number of target threads

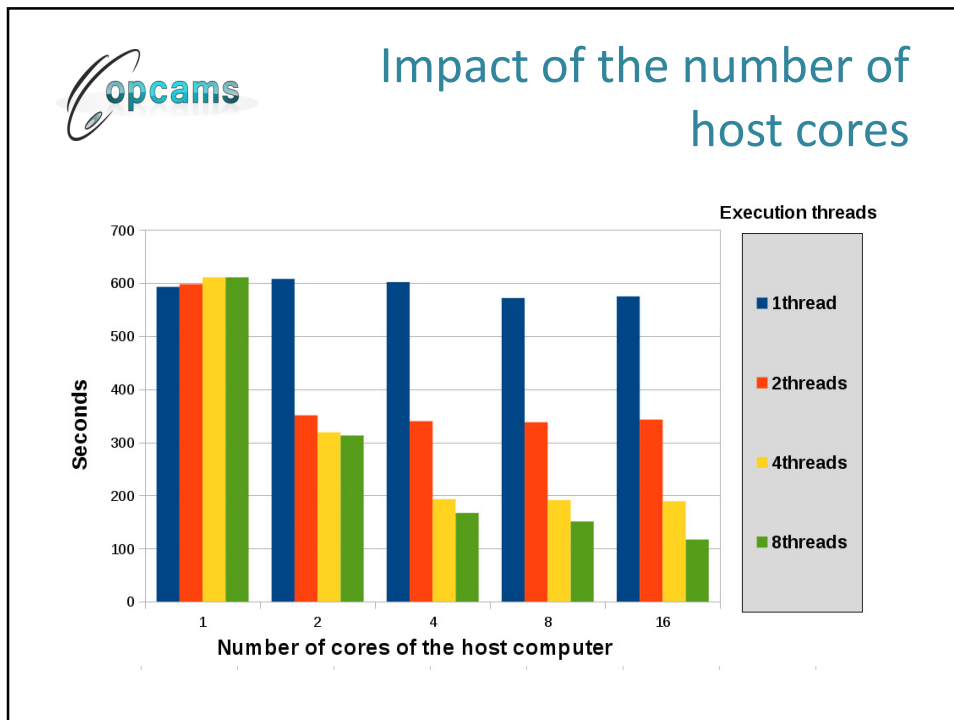
- The number of cores of the target platform has been limited to 4
- Although the native simulation requires more host time than the original benchmark, the speed-up is similar
- maximum difference between original code and simulation is about 15%.




Execution time vs number of cores in the target platform

- Performance improvement in simulation execution time is independent of the number of processors in the target platform
- Depends only on the number of host processors and application threads





 Impact of the number of host cores

- When the number of threads is higher than the number of host processors, the time penalty due to thread creation, scheduling time (i.e. context switching), etc. increases execution time
- When the number of threads is equal to or lower than the number of host processors, it can be seen that the execution time of the simulation is reduced proportionally to the number of threads in the application



Conclusions

- VIPPE: Native simulation tool
 - Efficient
 - Parallel
- Simulation of application SW
 - multi-processing platforms
- VIPPE API
 - optimized, general-purpose
 - supports complex OS APIs (i.e. POSIX)
- Experimental results
 - advantages in SW simulation on multi-core workstations