Institut
Mines-Telecom

Paris Sorbonne
University

System-Level Design for Communication-Centric Task Farm Applications

Daniela Genius, Ludovic Apvrille
daniela.genius@lip6.fr
ludovic.apvrille@telecom-paristech.fr

ReCoSoC 2017

Context

Model-oriented Design of Complex Embedded Systems

- Frequently used for the **software development** of embedded systems
- Complex MPSoC architecture with interconnect-on-chip
  - Architecture might be hierarchical

Communication-centric Applications

- Examples: telecommunication, video streaming
- Not natively supported by MDE approaches
- Our goal: **analysis of critical performance properties** such as contention on the interconnect, cache effects, ...

Related Work

Virtual Prototyping

- Kumar/Hansson/Huisken/Corporaal 2007: FPGA
- Ptolemy II, METROPOLIS, SESAME, ARTEMIS
- SoCLib: SystemC based, no graphical front-end

System-Level Design

- Batori/Theisz/Asztalo 2007
- DiNatale/Chirico/Sindico/Sangiovanni-Vincentelli 2014
- Pedroza/Knorreck/Apvrille 2011: AVATAR

AVATAR

Automated Verification of reAl Time softwARe

- SysML-based environment
- Analysis and design of embedded software
- Safety and security properties modeling and verification
- Fully supported by the free software TTool
  - Edition of AVATAR models
  - Simulation and formal verification (Internal tools, UPPAAL, ProVerif)
  - Timed functional model
    - Hardware targets are ignored (CPUs, buses, etc.)

SoCLib and MutekH

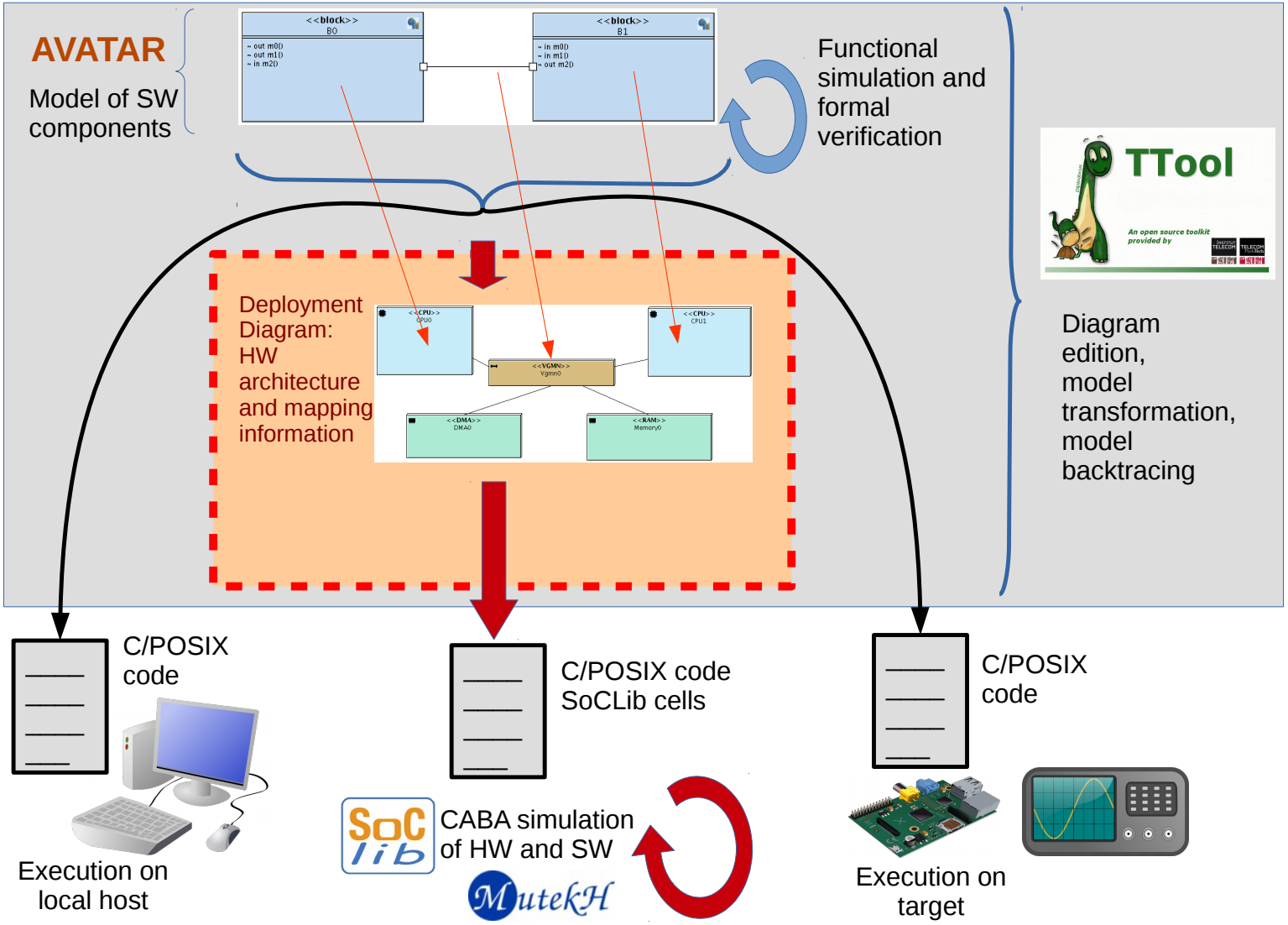
Hardware Simulation Platform based on SoCLib

- Virtual prototyping of complex Systems-on-Chip
- Supports several models of processors, buses, memories
  - Example of CPUs: MIPS, ARM, SPARC, Nios2, PowerPC
- Two simulation models:
  - TLM = Transaction Level Modeling
  - CABA = Cycle Accurate Bit Accurate

Embedded Operating System: MutekH

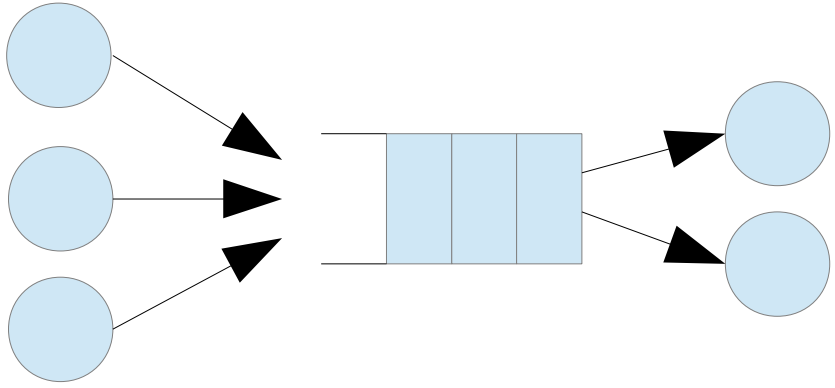
- Natively handles heterogeneous multiprocessor platforms
- POSIX thread support

Tooling


AVATAR
Model of SW components
<<block>> B0
<<block>> B1
Functional simulation and formal verification
TTool
An open source toolkit provided by
Deployment Diagram: HW architecture and mapping information
<<CPU>> CPU0
<<CPU>> CPU1
<<VGMN>> Vgmn0
<<DMA>> DMA0
<<RAM>> Memory0
Diagram edition, model transformation, model backtracing
C/POSIX code
Execution on local host
C/POSIX code SoCLib cells
SoCLib
CABA simulation of HW and SW
MutekH
C/POSIX code
Execution on target

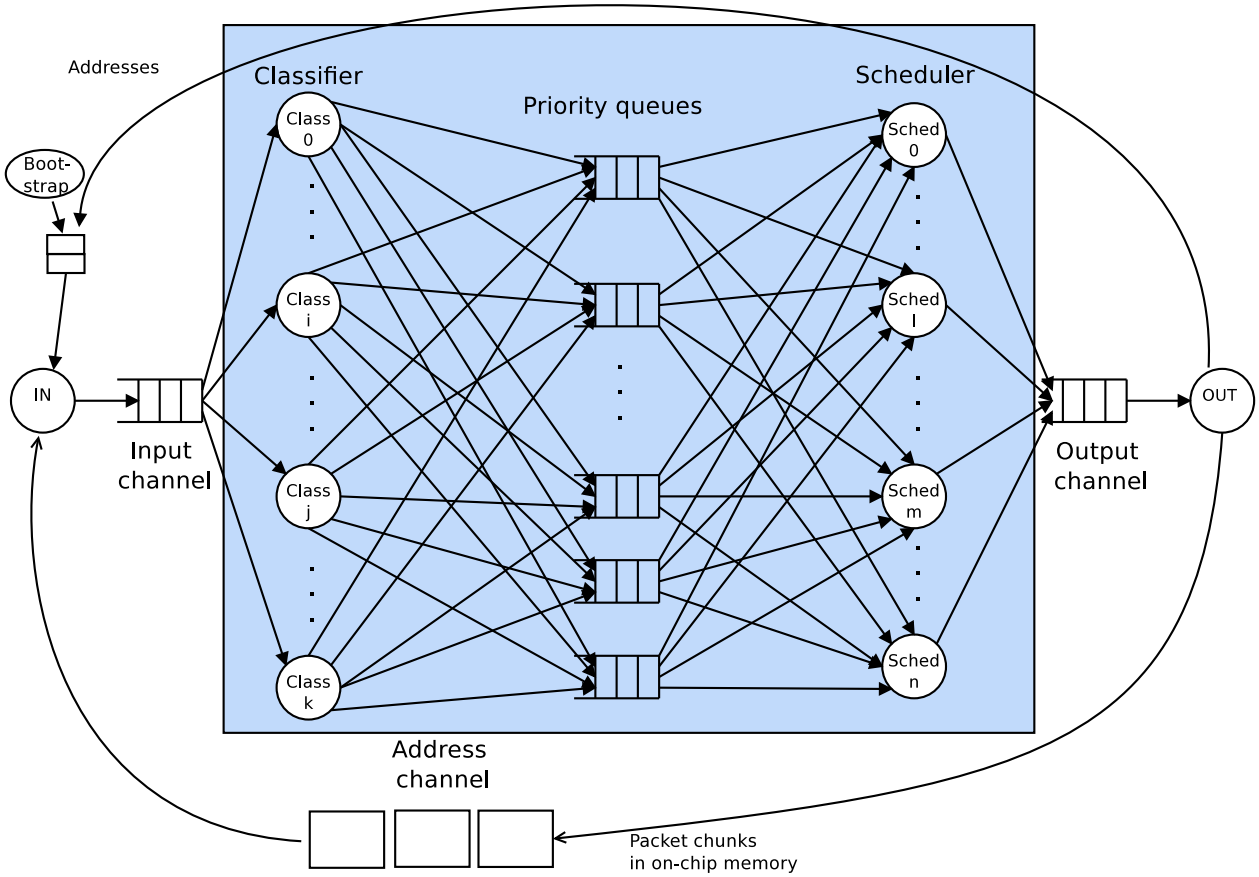
Multi-writer Multi-reader Paradigm [Faure/GreinerGenius/2006]

- Any number of reader or writer tasks can access the channel
- Reader or writer tasks can be hardware or software tasks
- Implemented as software channels in shared memory
- Blocking or non-blocking read and write primitives (read/try read, write/try write)



Telecommunication Application [Genius/Faure/Pouillon 2011]

- Task farm
- 8-byte *descriptor*
- *bootstrap task*
- *input task*
- *classification task*
- *scheduling task*
- *output task*



AVATAR Communication Channels

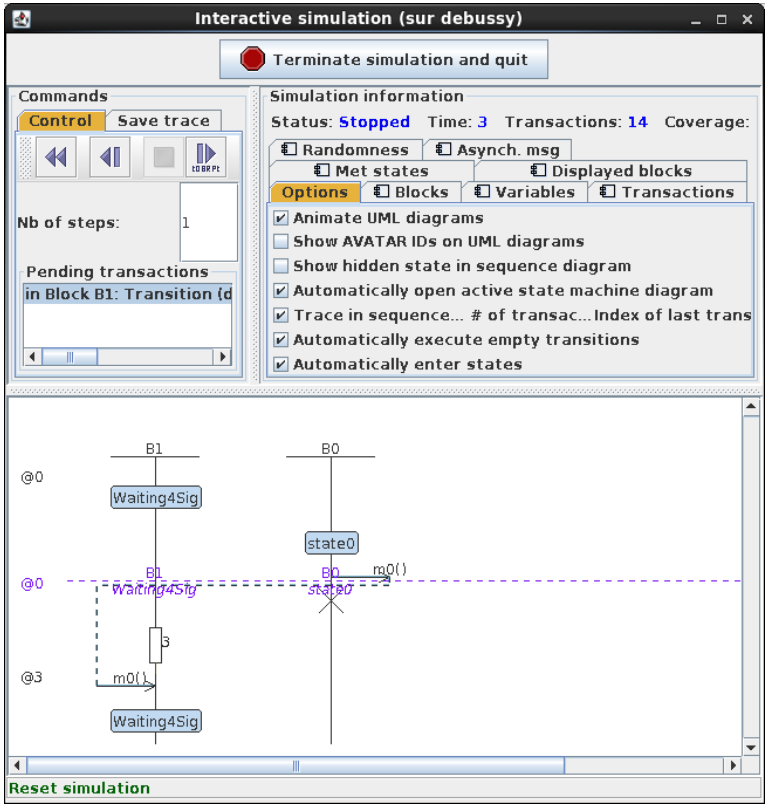
Asynchronous Semantics

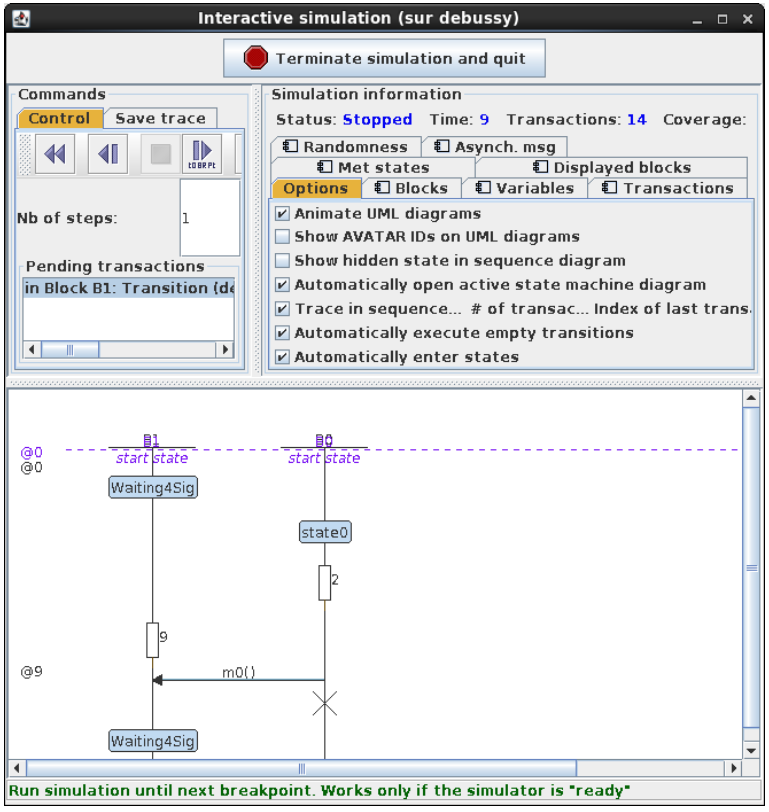
- Blocking on read
- Blocking or non-blocking on write
- Lossy

Synchronous Semantics

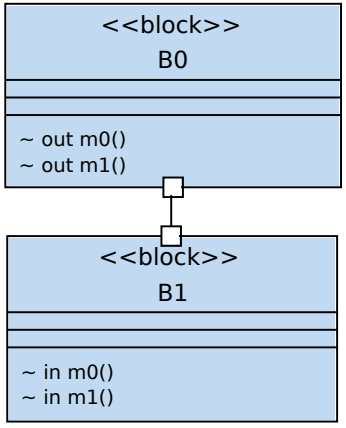
- Difficult to implement for MPSoC, require central management
- broadcast (not implemented)
- private channel, security features (not implemented)

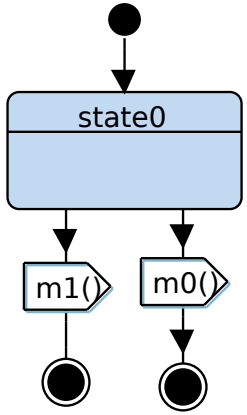
AVATAR Communication Channels (contd.)

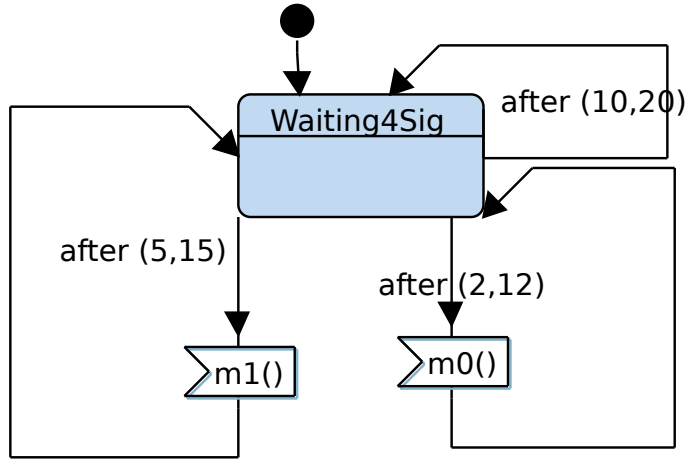




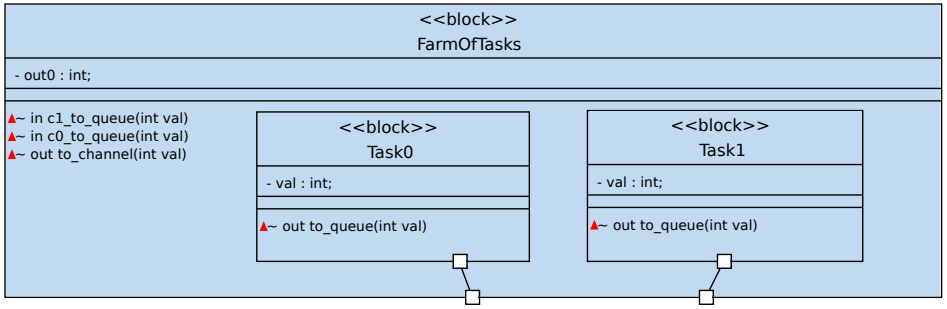
AVATAR Communication Channels: Non-determism

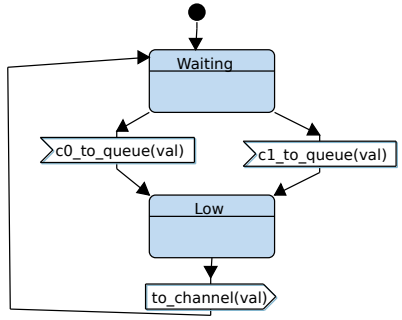


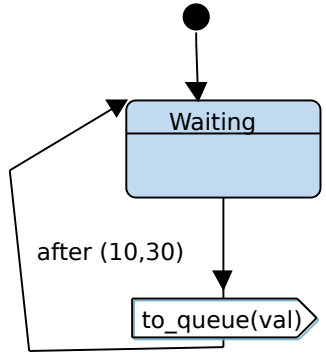




Extending the Semantics



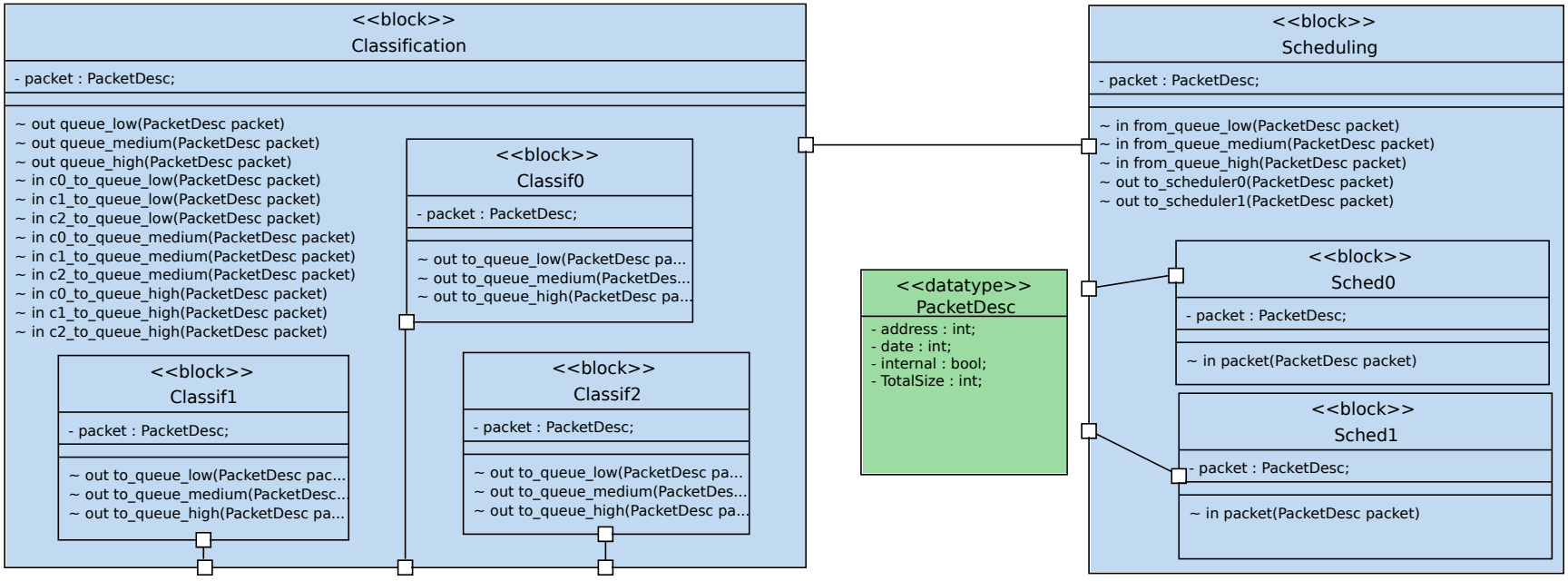




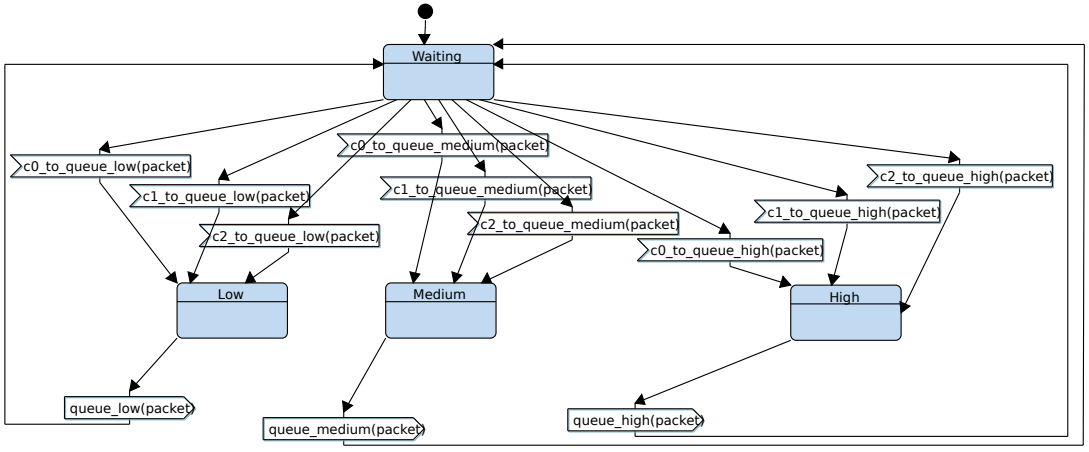
How to ensure the Task Farm property?

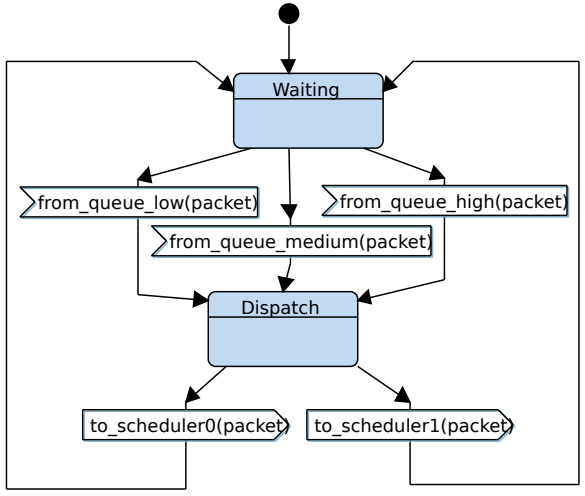
- Several readers/writers can access the channel in a non-deterministic fashion (provided by AVATAR semantics)
- Asynchronous AVATAR semantics
- Use hierarchical blocks to regroup channels
- Priority queues have *non-blocking* read and write operations: *try-read* and *try-write*

AVATAR Model: Block Diagram

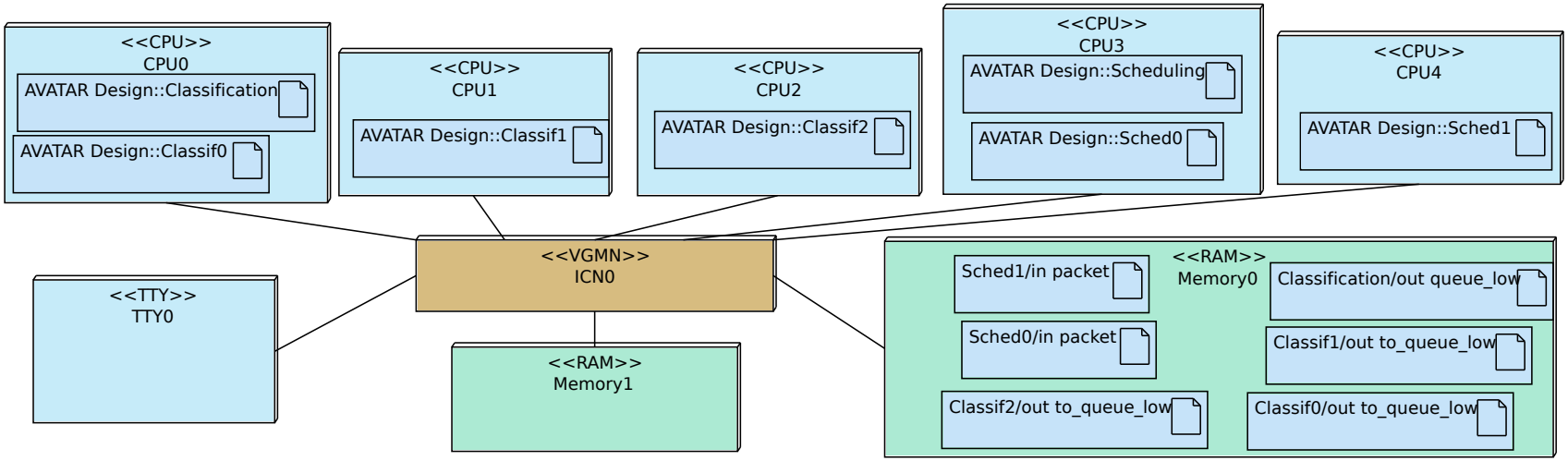


AVATAR Model: Timed Automata



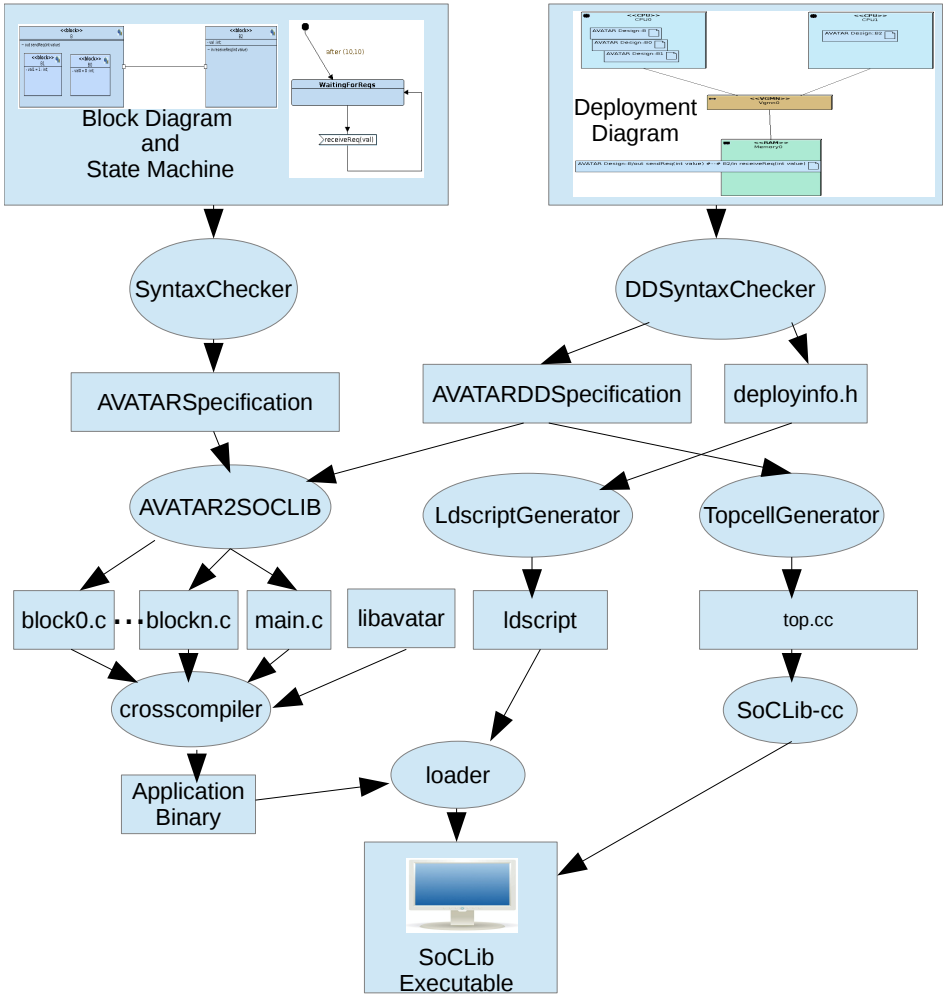


AVATAR Model: Deployment Diagram



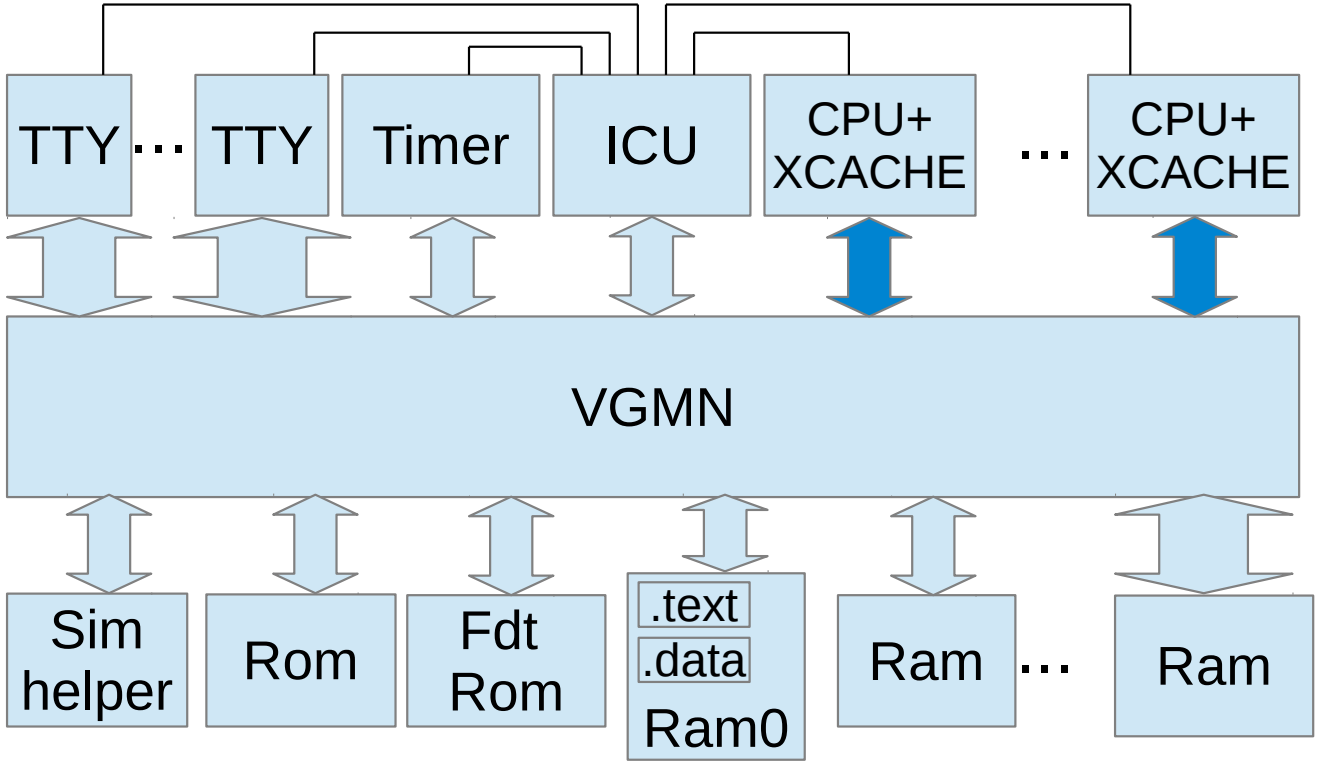
Tool Chain [ERTSS2016]

1. **Libavatar** Runtime for SoCLib, implements AVATAR operators
2. **DDSyntaxChecker** checks syntax of deployment diagrams and identifies their elements
3. **AVATAR2SOCLIB** translates AVATAR blocks into C POSIX tasks, generates main program
4. **TopcellGenerator** generates SystemC top cell
5. **LdscriptGenerator** generates linker script



Top Cell Generation

- SystemC instanciation of components, netlist, table associating memory segments, charging of code
- Some components transparent to the user (interrupts, simulation infrastructure)



Code Generation: Extract from Main Program

```
#define CHANNEL0 __attribute__((section("section_channel0")))
#define LOCK0 __attribute__((section("section_lock0")))
...
  pthread_t thread__Classif0;
  pthread_t thread__Classif1;
  pthread_t thread__Classif2;
  pthread_t thread__Sched0;
  pthread_t thread__Sched1;
  ...
  struct mwmr_s *channels_array_Classif[3];
  ...
  ptr =malloc(sizeof(pthread_t));
  thread__Classif2= (pthread_t)ptr;
  attr_t = malloc(sizeof(pthread_attr_t));
  pthread_attr_affinity(attr_t, 2);
```

Code Generation: Extract from Topcell

```
maptab.add(Segment("cram0", 0x10000000, 0x800, IntTab(2), true));
maptab.add(Segment("uram0", 0x10200800, 0x800, IntTab(2), false));
maptab.add(Segment("cram1", 0x20000000,  0x800, IntTab(7), true));
maptab.add(Segment("uram1", 0x20200800 ,  0x800, IntTab(7), false));
...
soclib::caba::VciRam<vci_param>Memory1("Memory1", IntTab(2), maptab);
soclib::caba::VciRam<vci_param>Memory0("Memory0", IntTab(7), maptab);
...
    data_ldr.load_file(std::string(kernel_p)
        + ";.data;.channel0; ...
        .channel14;.cpudata;.contextdata");
```

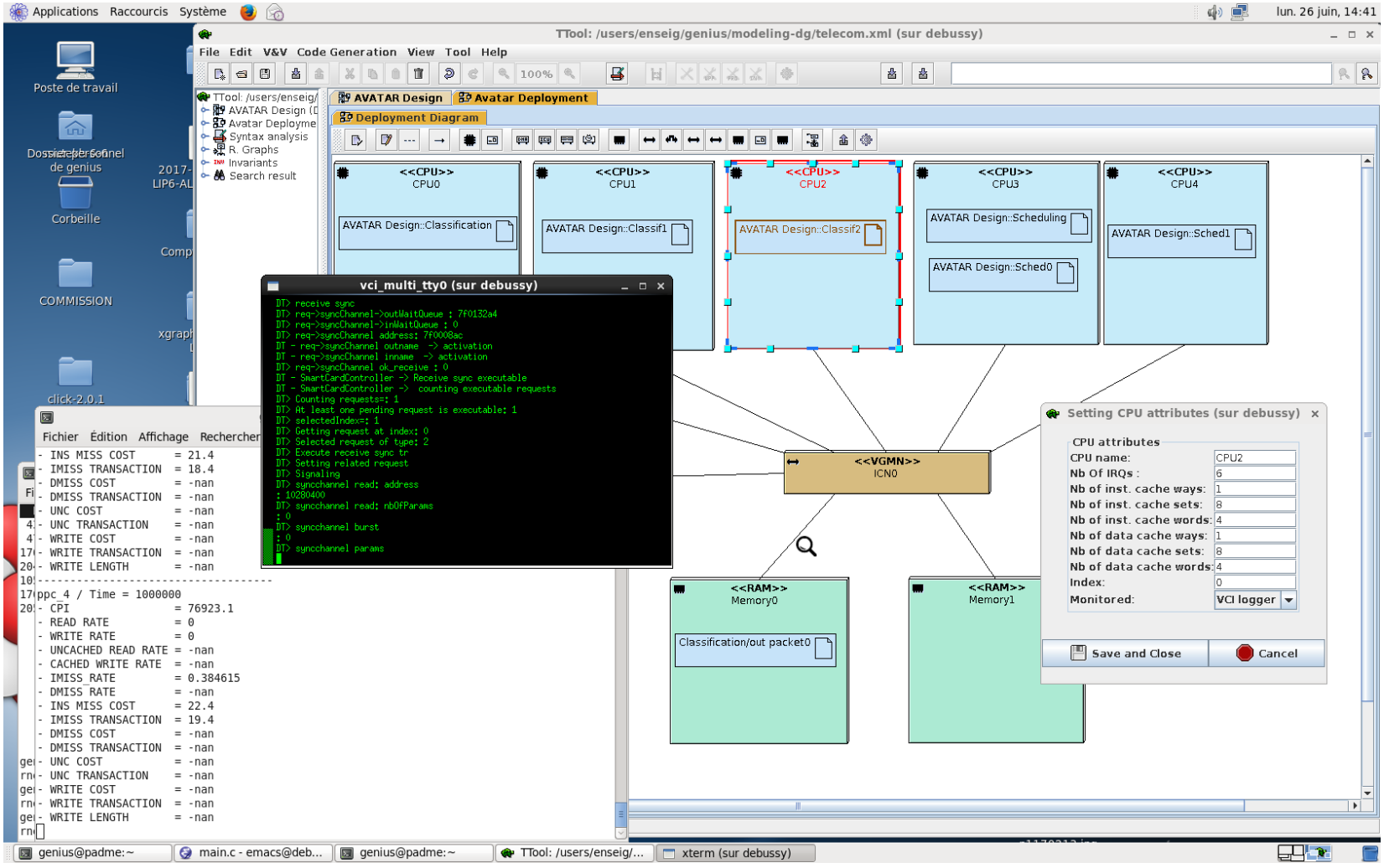
Code Generation: Extract from ldscript

```
uram0 (RWAL) : ORIGIN = 0x10280000, LENGTH = 0x80000
cram0 (RWAL) : ORIGIN = 0x10000000, LENGTH = 0x80000
...
.channel0 : {*(section_channel0)} > uram0
.channel14 : {*(section_channel14)} > uram1
.lock0 : { *(section_lock0)} > uram0
...
.lock14 : { *(section_lock14)} > uram1
```

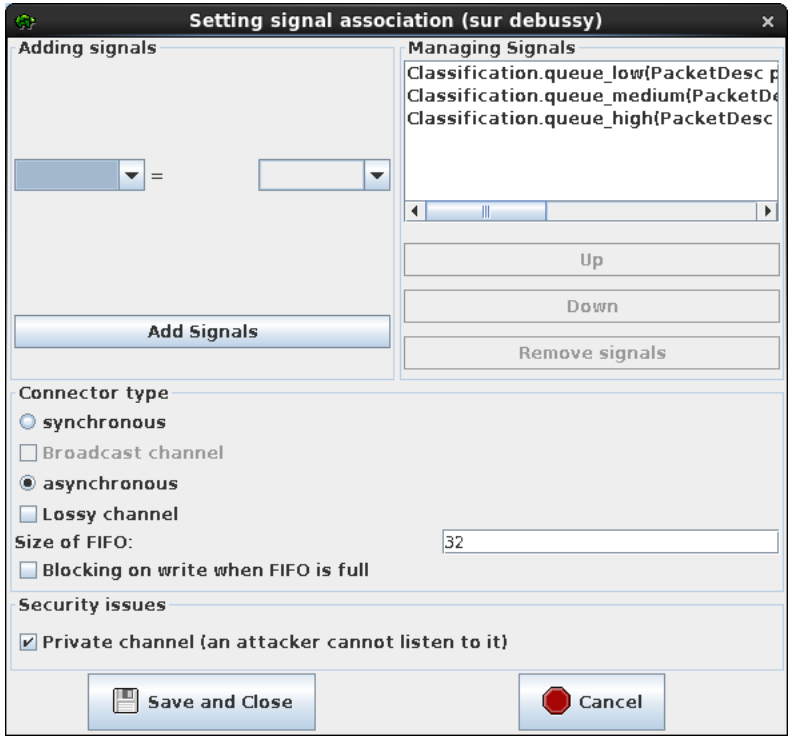
Experimental Setup

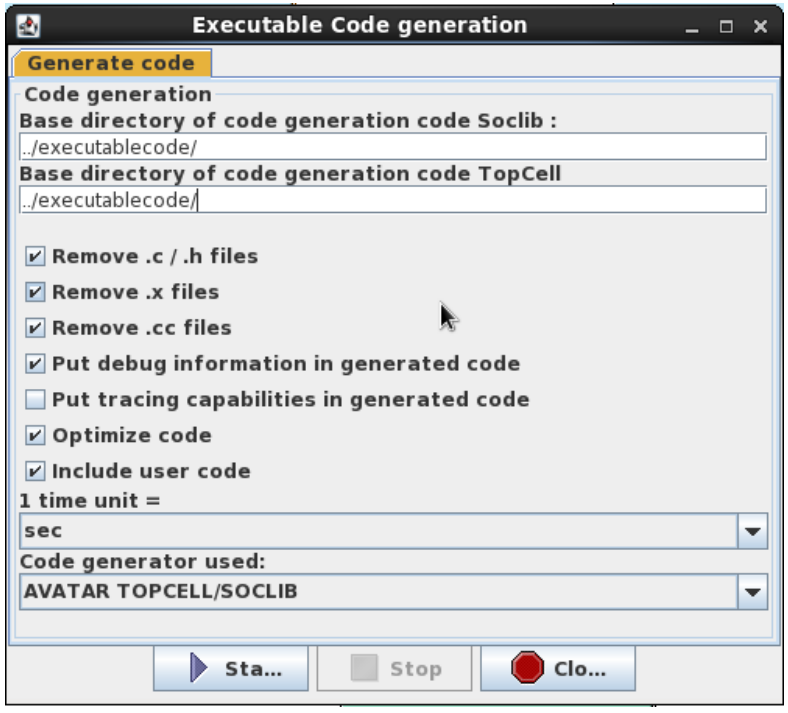
- CABA (Cycle Accurate/Bit Accurate) level simulation
- *PowerPC405* processors
- MutekH
- SYSTEMCASS

Using TTool/SoCLib

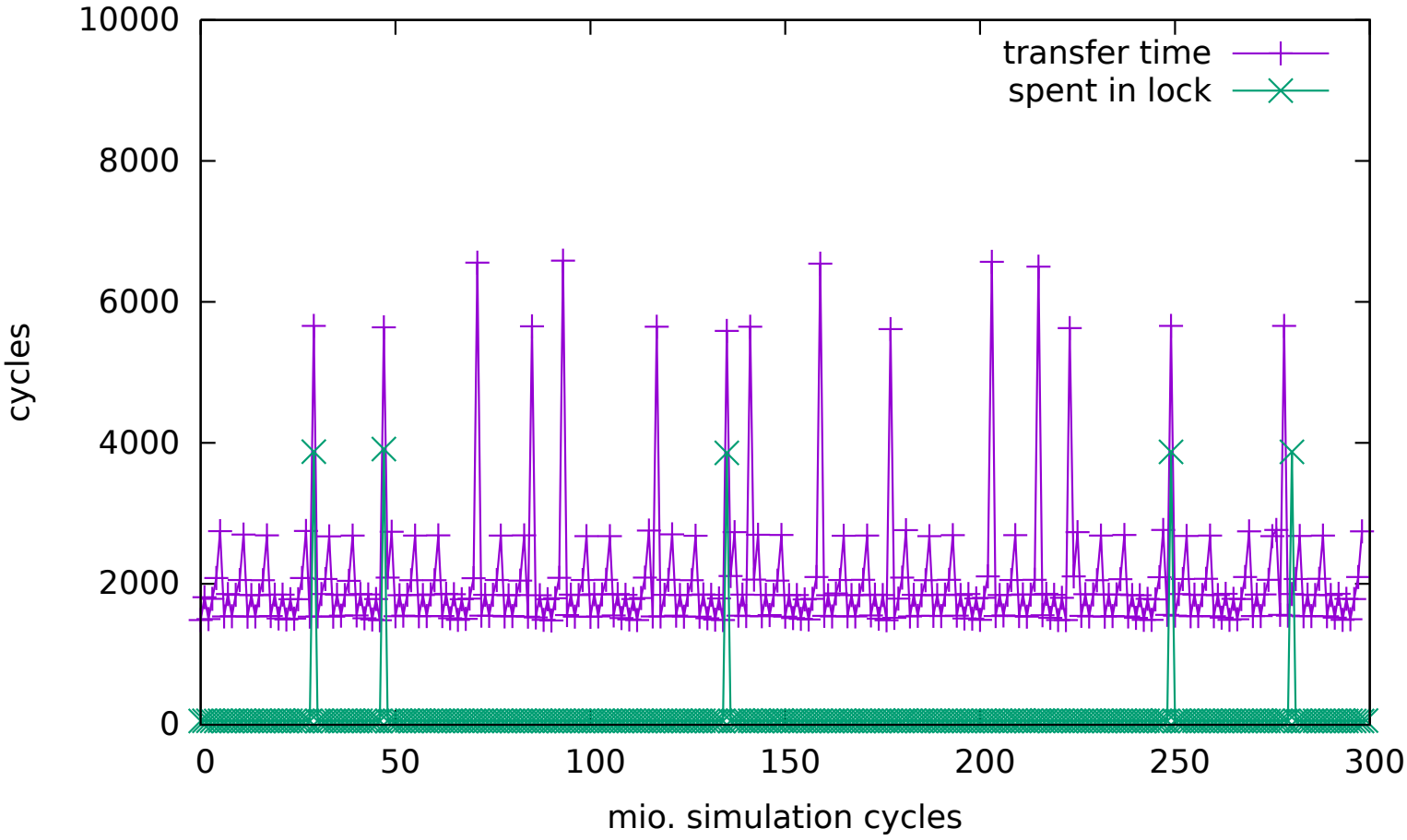


Using TTool/SoCLib: Channel Configuration and Code Generation Dialogues



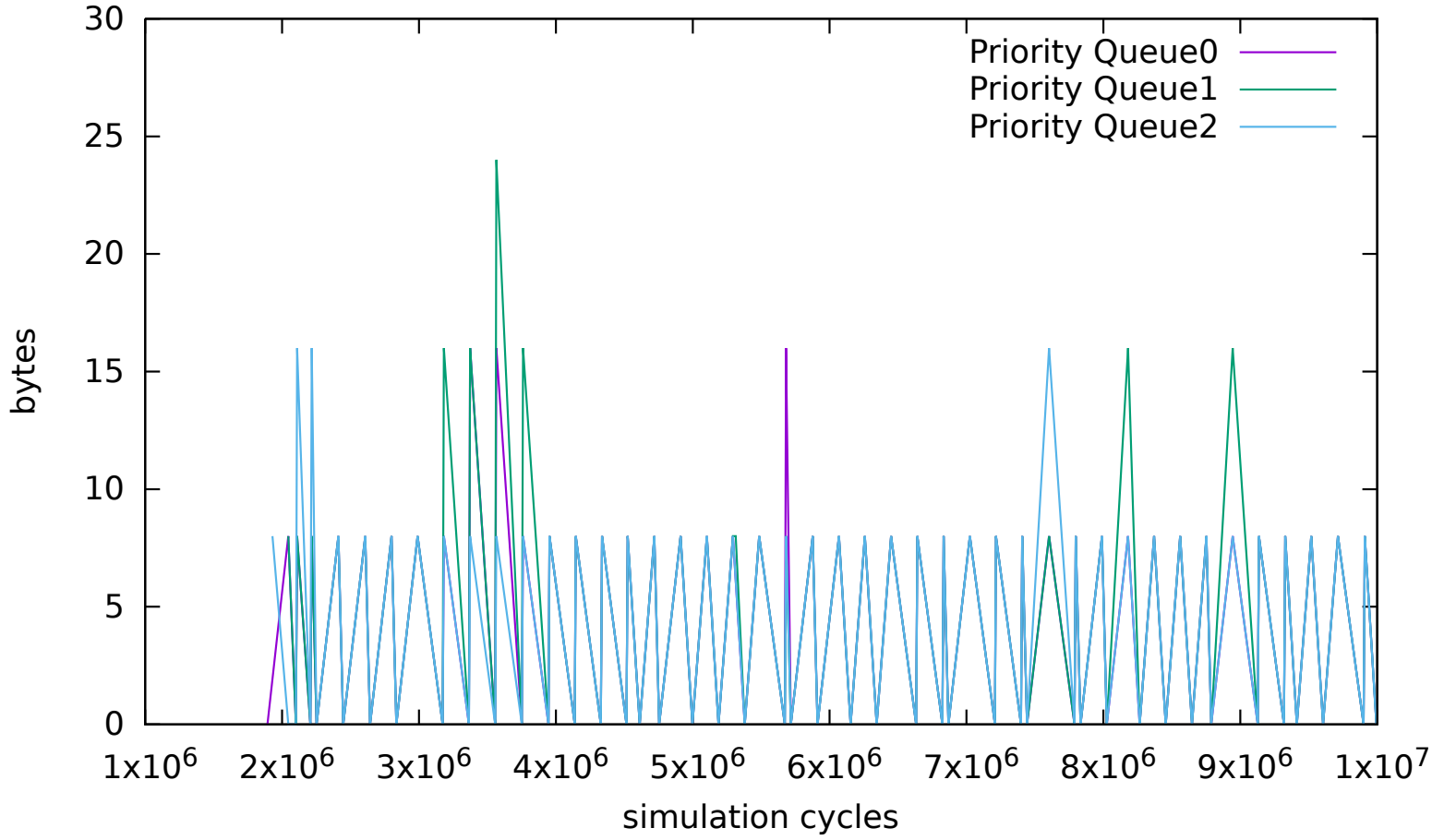


Experiments



Duration of a transfer in the priority queue (SoCLib simulation cycles)

Experiments



Fill state of the priority queues

Conclusion

Limitations

- AVATAR cannot model application specific co-processors
- Better capture latencies and automatically evaluate them during the prototyping stage
- Support more interconnects (CAN, clustered etc.)

Perspectives

- Increase simulation speed (TLM?)
- Performance evaluation tools
- Long term: Design Space Exploration with low-level simulations

Questions?

To try/download TTool:

ttool.telecom-paristech.fr

To try/download SoCLib/MutekH:

www.soclib.fr
www.mutekh.fr