A schedulerless semantics of TLM models written in SystemC via translation into Lotos

Olivier Ponsini and Wendelin Serwe

INRIA / VASY

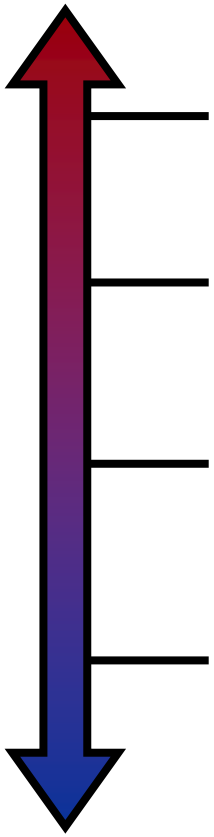
http://www.inrialpes.fr/vasy

Outline

- TLM (Transaction Level Modeling)
- Untimed TLM in SystemC
- Verifying SystemC/TLM
- A schedulerless semantics of SystemC/TLM
- Conclusion

TLM in electronic design flow

abstraction & speed



accuracy

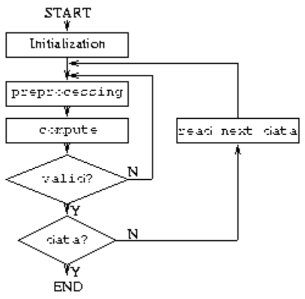
Algorithmic models

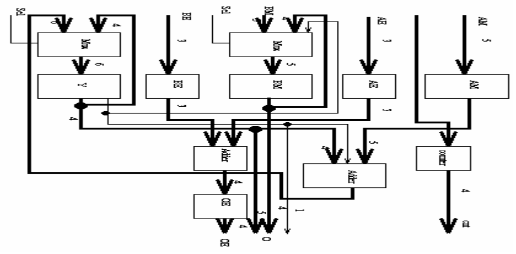
TLM { - Untimed / - Timed }

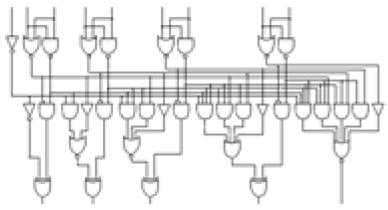
RTL models

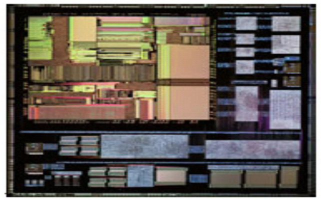
Gate level models

Real system



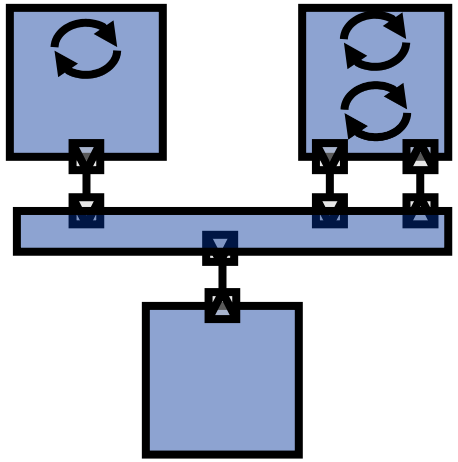






Untimed transaction level models

- Embedded software programmer’s view
  - Architecture: modules
  - Behavior: concurrent processes
  - Communication:
    - Transactions (inter-modules)
    - Synchronizations (inter-processes)
- Reference model
  - Functional verification
  - Embedded software development
  - Co-simulation



Untimed TLM model of computation

- Concurrent execution of independent processes
- System synchronization for causal dependencies between processes
- Bit-true behavior
- Bit-true communication

Outline

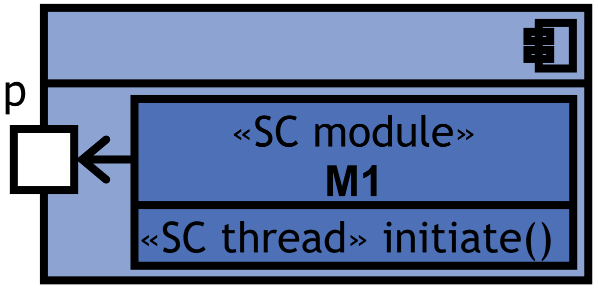
- TLM (Transaction Level Modeling)
- Untimed TLM in SystemC
- Verifying SystemC/TLM
- A schedulerless semantics of SystemC/TLM
- Conclusion

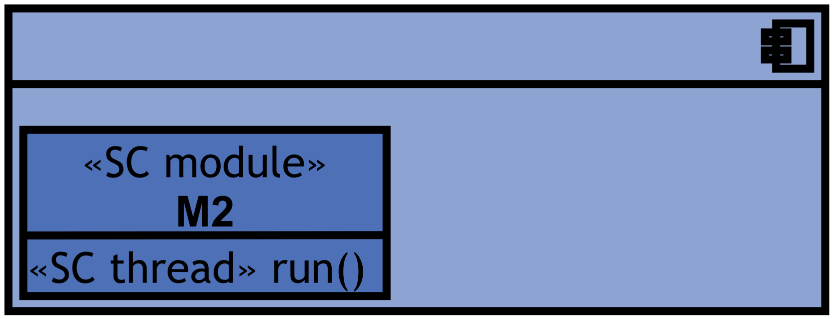
SystemC

- A C++ library
- Heterogeneous (hard/soft) system modeling
  - Classes and macros for architecture and behavior
  - Hardware convenient data types
- System simulation
  - A global scheduler
  - A simulated time

TLM architecture and behaviors in SystemC

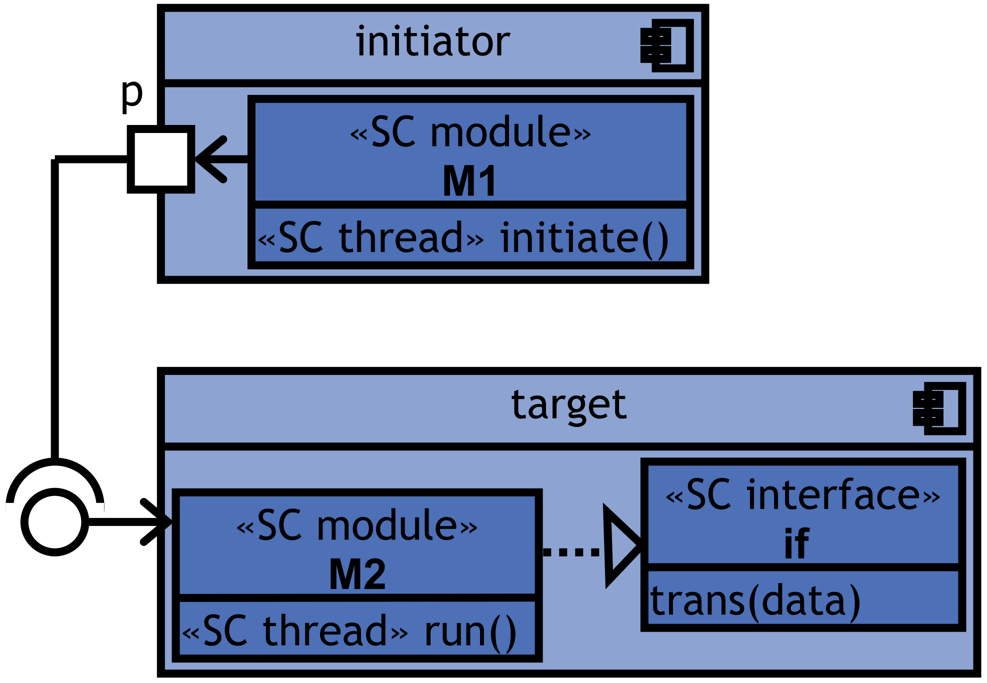
```
SC_MODULE(M1) {
  sc_port p;
  SC_CTOR(M1) {
    SC_THREAD(initiate);
  }
  void initiate() { … }
};

SC_MODULE(M2) {
  SC_CTOR(M2) {
    SC_THREAD(run);
  }
  void run() { … }
};
```





TLM transactions in SystemC

```
class if : sc_interface {
  virtual void trans(data);
};
SC_MODULE(M1) {
  sc_port<if> p;
  SC_CTOR(M1) {
    SC_THREAD(initiate);
  }
  void initiate() { p.trans(data) }
};
SC_MODULE(M2) : if {
  SC_CTOR(M2) {
    SC_THREAD(run);
  }
  void run() { … }
  void trans(data) { … }
};
```



```
int sc_main() {
  M1 initiator; M2 target;
  initiator.p.bind(target);
}
```

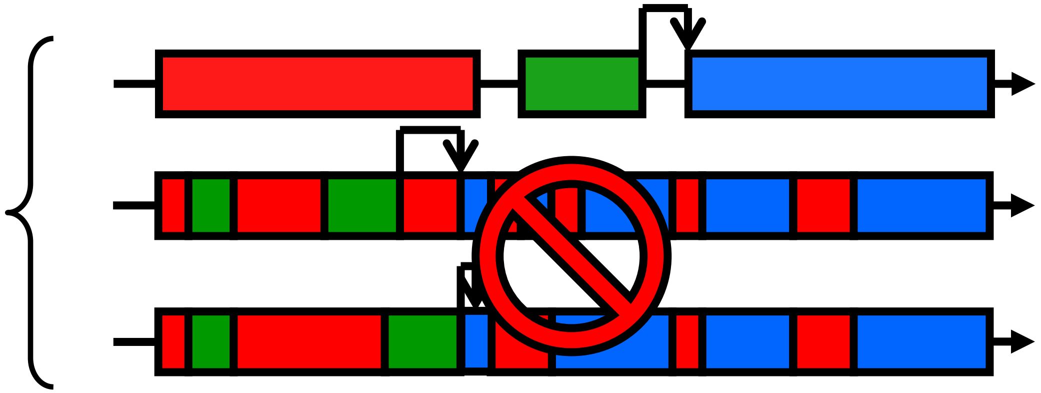
TLM synchronizations in SystemC

- Event-based mechanism between threads
  - Wait
  - Notify

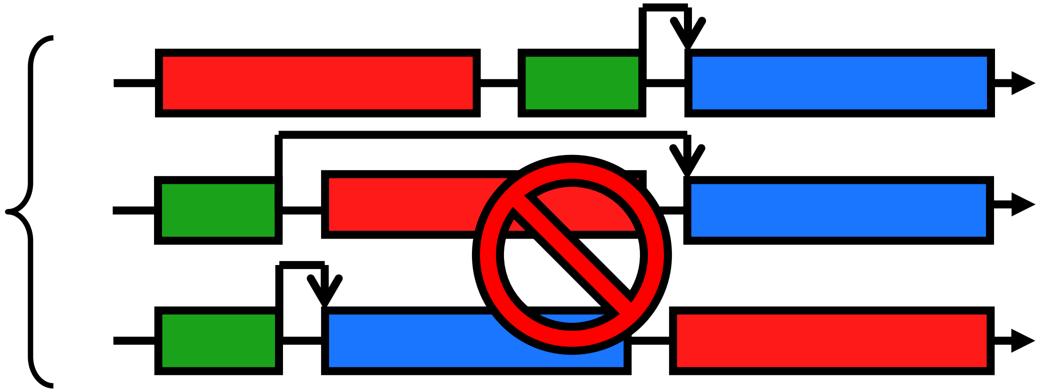
```
SC_MODULE(M2) : if {
  event e;
  SC_CTOR(M2) {
    SC_THREAD(run);
  }
  void run() { … wait(e); … }

  void trans(data) { … e.notify(); … }
};
```

SystemC simulation

- Based on a global scheduler
  - Nonpreemption



  - Fixed order of execution



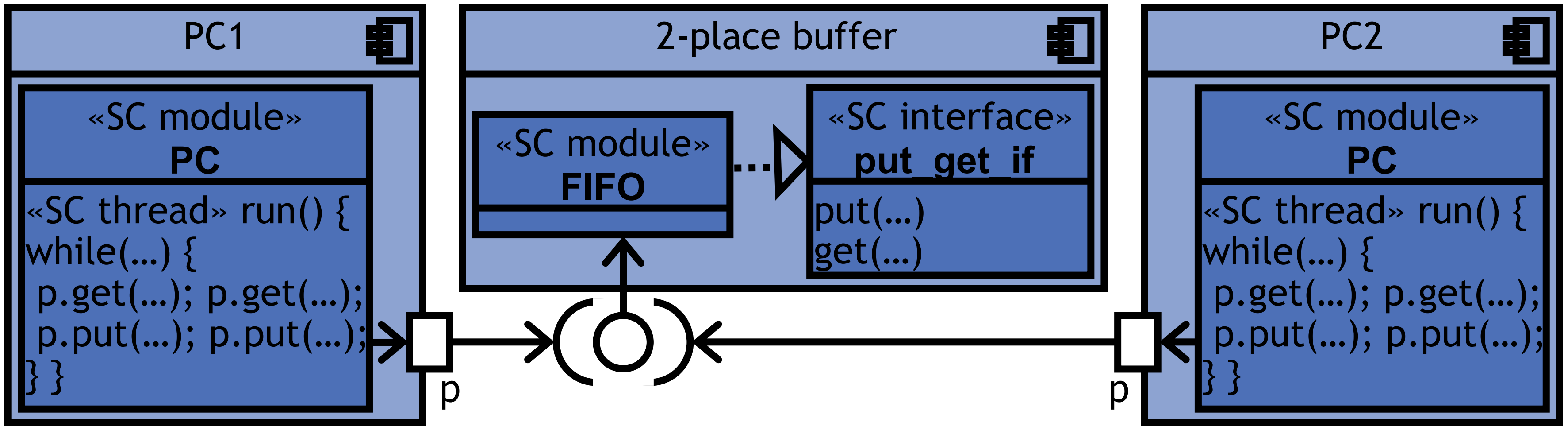
Outline

- TLM (Transaction Level Modeling)
- Untimed TLM in SystemC
- Verifying SystemC/TLM
- A schedulerless semantics of SystemC/TLM
- Conclusion

Verifying SystemC/TLM models

- Main TLM modeling challenge: find all the synchronizations between processes
- Verification needs to explore
  - Data space *and*
  - Processes interleaving space
- Methods based on the SystemC simulation semantics are limited by
  - Nonpreemption → possible interleavings not considered
  - Fixed execution order → only one interleaving explored

Issue related to nonpreemption



- In a concurrent implementation, deadlocks could occur!
- With SystemC simulation semantics, erroneous behaviors are hidden!

Outline

- TLM (Transaction Level Modeling)
- Untimed TLM in SystemC
- Verifying SystemC/TLM
- A schedulerless semantics of SystemC/TLM
- Conclusion

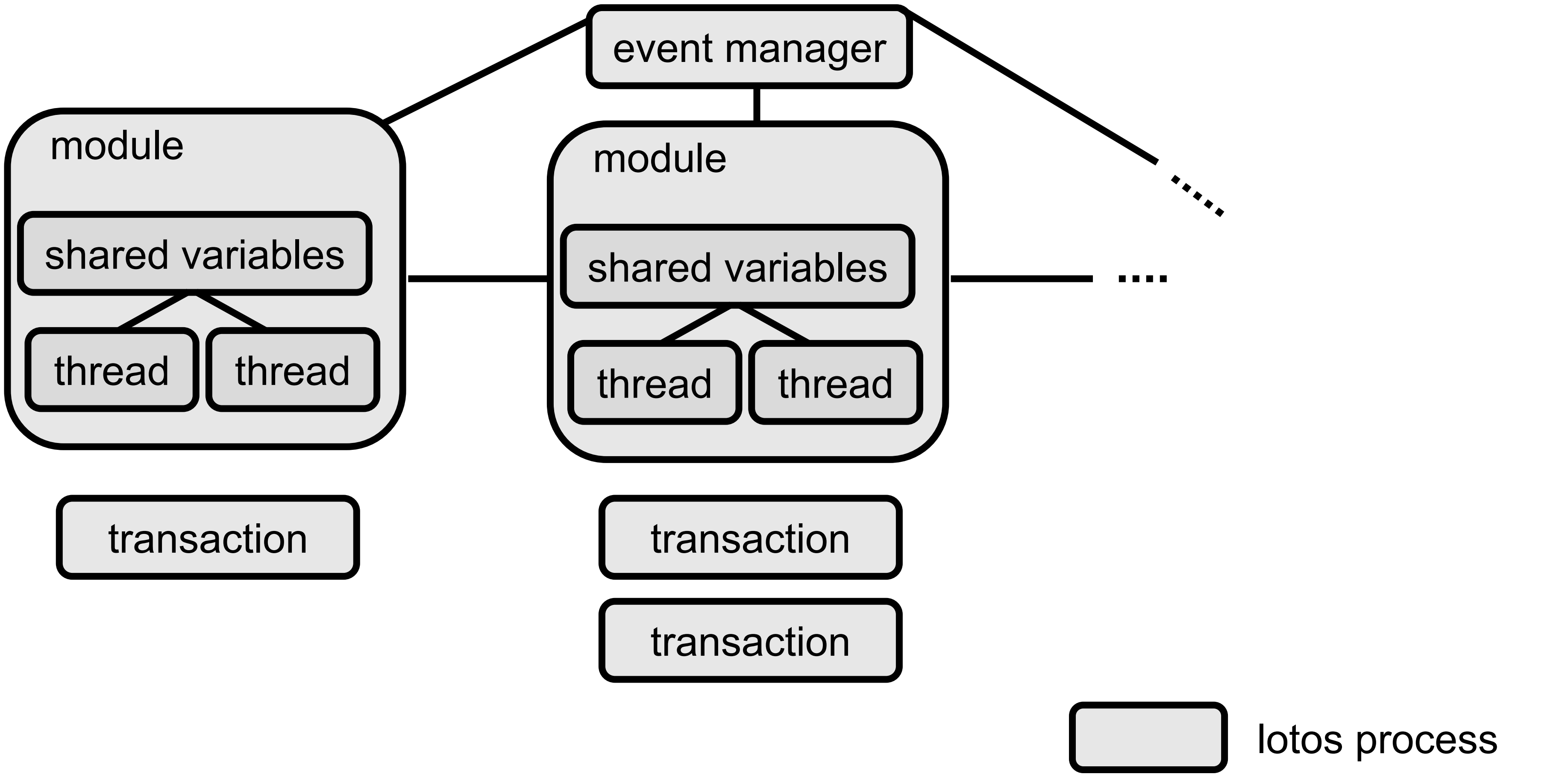
A schedulerless semantics

- Goal: “verifying the system” instead of “verifying the simulation of the system”
- What: a formal semantics with a more general concurrency model
  - Generalizing the SystemC simulation semantics
  - Closer to TLM model of computation and real system
  - Connected to established verification tools
- How: defining the translation from a TLM-subset of SystemC into Lotos formalism

Lotos and CADP

- Lotos
  - Standard process algebra
  - Formal semantics
- Lotos fits well with TLM model of computation
  - Asynchronous concurrent processes
  - Synchronization and communication by *rendezvous*
- Lotos is an input language for CADP
  - μ-calculus model-checking
  - Equivalence checking
  - Compositional verification
  - Etc.

Encoding overview



Encoding transactions

- Transactions are processes
  - Outside target modules

```
process trans[<RV_trans>](<REQ>)  : exit(<RSP>) :=
  …;   exit(<rsp>)
enproc
```

  - Instantiated by initiator processes

```
process initiator[<RV>] : noexit :=
  …; trans[<RV_trans>](<req>)
        >> accept <RSP> in …
enproc
```

Encoding event communication

- Publish-subscribe pattern
- An event manager process

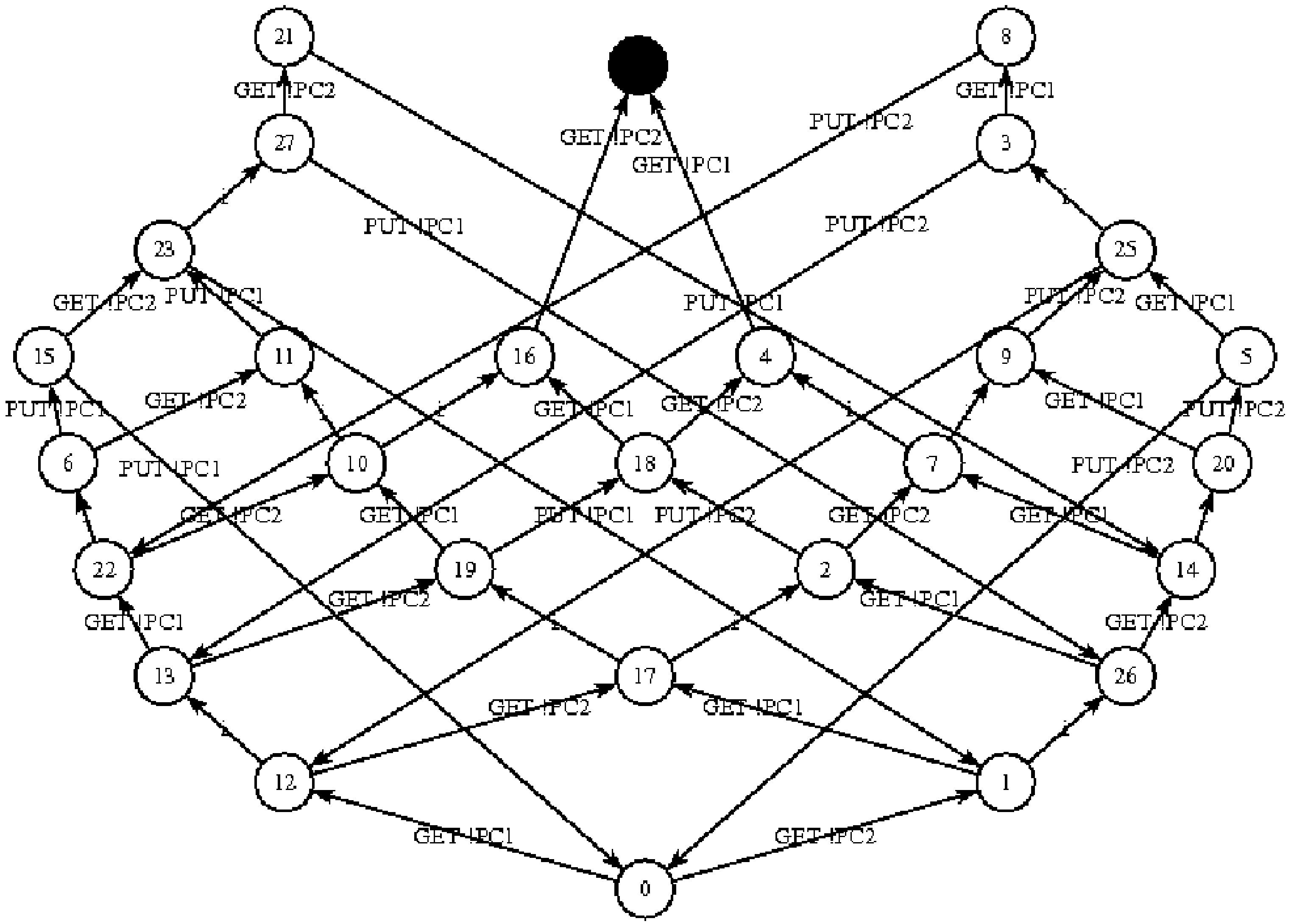
```
process event_manager[notify,suspend,resume](id_proc_evt:Bool) : noexit :=
  suspend !id_proc !evt; event_manager[notify,suspend,resume](true)
 [ ]
  notify !evt;
  ( [id_proc_evt]-> resume !id_proc; event_manager[notify,suspend,resume](false)
   [ ]
    [not(id_proc_evt)]-> event_manager[notify,suspend,resume](id_proc_event) )
endproc
```

- 3 rendezvous: notify, suspend, resume

Subscriber:
  wait(evt) ≡ { suspend !id_subscriber !evt; resume !id_subscriber

Publisher:
  evt.notify() ≡ notify !evt

Back to the issue example



Conclusion

- SystemC/TLM models should be verified for correct synchronizations
- SystemC simulation semantics is not sufficient for verification
- Our schedulerless Lotos semantics
  - Generalizes the SystemC simulation semantics
  - Is closer to real hardware and TLM
  - Is connected with formal verification tools (CADP)

Perspectives

- A first necessary step
  - To reason about SystemC/TLM models
  - To automate the translation
- On-going work on an industrial case-study (approx. 26 000 lines of code)
- Evaluation of encoding variants as regards verification performance
- Inverse translation: Lotos into SystemC/TLM

A schedulerless semantics of TLM models written in SystemC via translation into Lotos

Olivier Ponsini and Wendelin Serwe

INRIA / VASY

http://www.inrialpes.fr/vasy