Translating Hardware Process Algebra into Standard Process Algebra

Illustration with CHP and LOTOS

Gwen Salaün and Wendelin Serwe

INRIA Rhône-Alpes / VASY

655, avenue de l'Europe

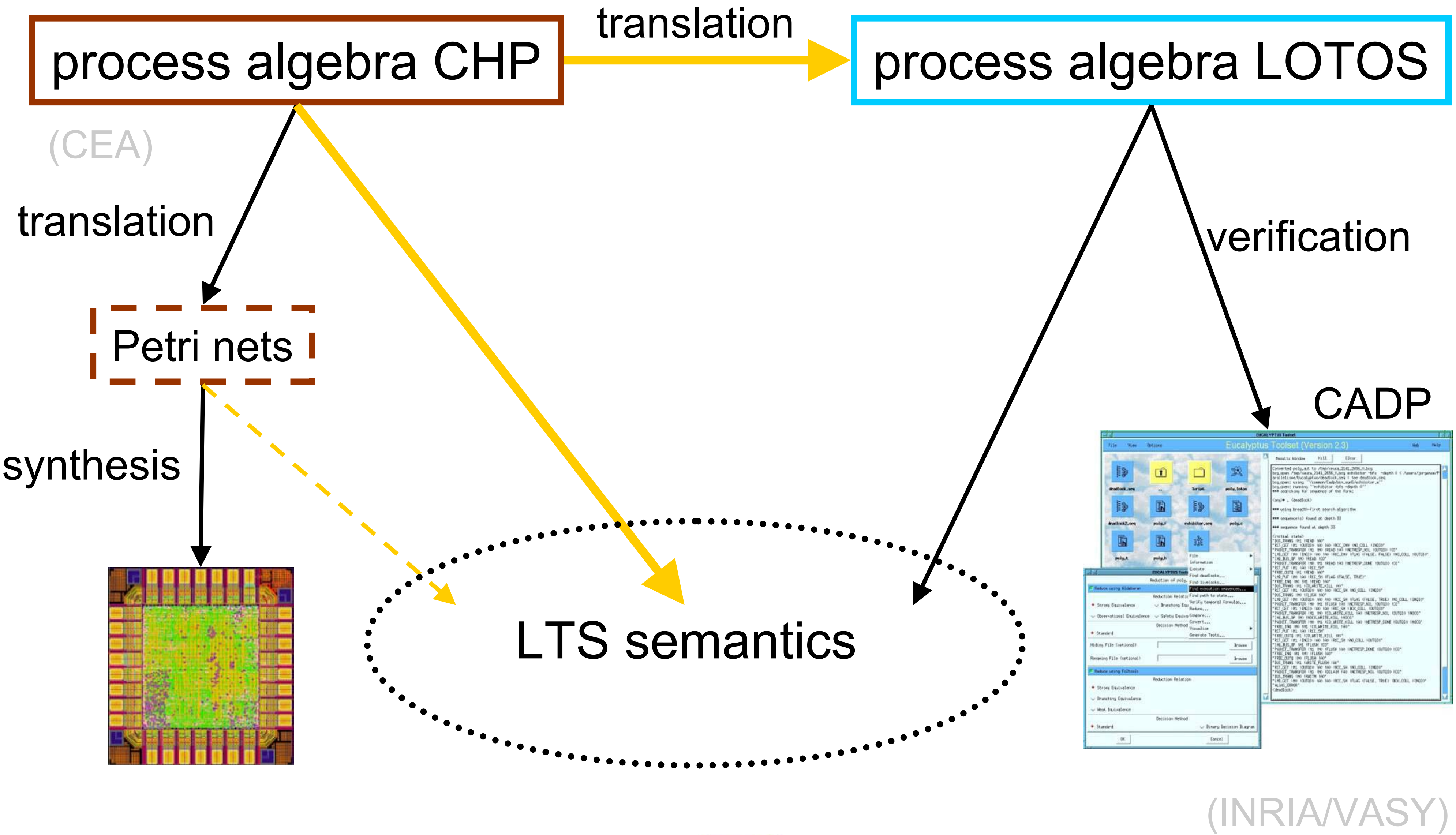
F-38330 Montbonnot Saint-Martin

http://www.inrialpes.fr/vasy

Synchronous *vs* Asynchronous Circuits

- Current standard: synchronous circuits
- Asynchronous circuits: absence of global clock
  + lower power consumption
  + higher performance
  + absence of global timing problems
- Main difficulty: complex designs
  - existing tool support: simulation and synthesis
  - verification is needed!
- High-level languages to describe processes communicating by message-passing along wires

Verification of Asynchronous Circuits



Outline of the Talk

- Hardware process algebra – CHP
- Operational semantics for CHP
- Translation of CHP into LOTOS
- Conclusion and future work

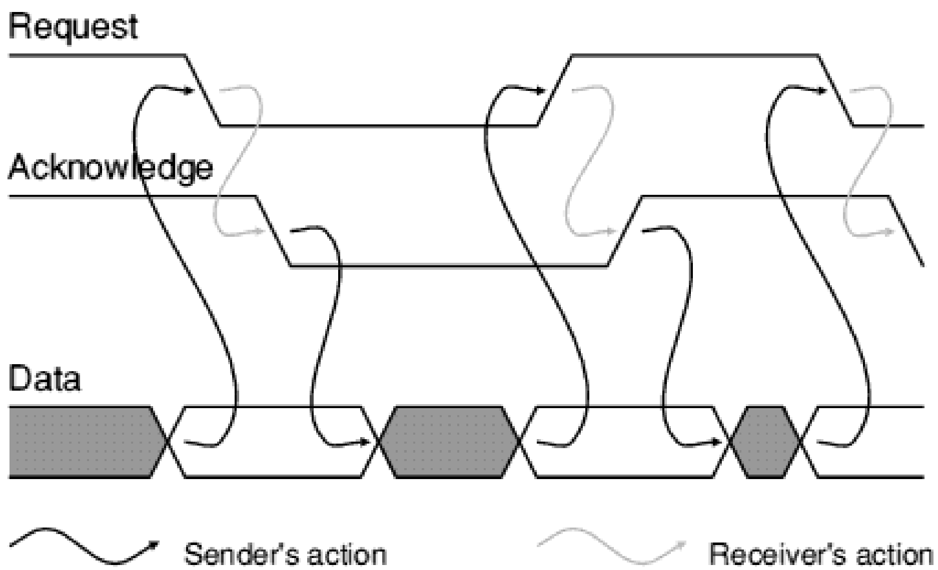
Hardware Process Algebra

- Abstract descriptions of asynchronous circuits
- Several languages: CHP, Balsa, Tangram
- CHP (Communicating Hardware Processes):
  - Compilation to VLSI circuits [Martin-86]
  - Inspired by guarded commands and CSP
  - Tool support: TAST (TIMA Lab., Grenoble)
- Specific operators to exploit low-level aspects of hardware implementation of communication channels

Channels and Handshake Protocols

- Representation of wires connecting processes
- Binary: two-point connections
- Directed: sender, receiver
- Synchronized locally by handshake protocols
- Asymmetric:
  - Active (request)
  - Passive (acknowlegde)

In this talk:
active sender
passive receiver



CHP Syntax

- Null action: skip
- Assignments to local variables: x:=v
- Communications on channels: C!v / C?x
- Sequential composition: $B_1; B_2$
- Collateral composition: B_1, B_2

no interaction
between B_1 and B_2

- Nondeterministic guarded commands:

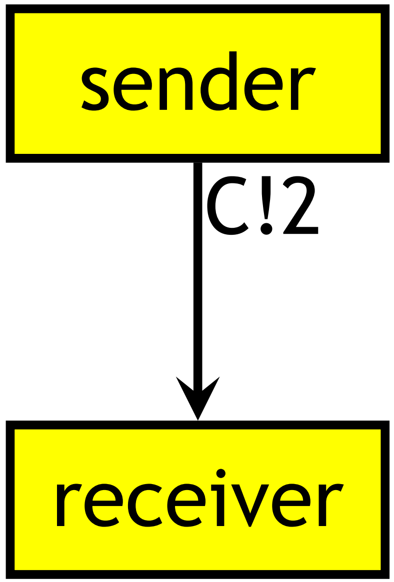
$$@[\ G_1 \Rightarrow B_1;\ end_1\ \ \ldots\ \ G_n \Rightarrow B_n;\ end_n\ ]$$

with $end_i \in \{$ break, loop $\}$

- A system: processes interacting via channels

The Probe Construct: C#, C#v

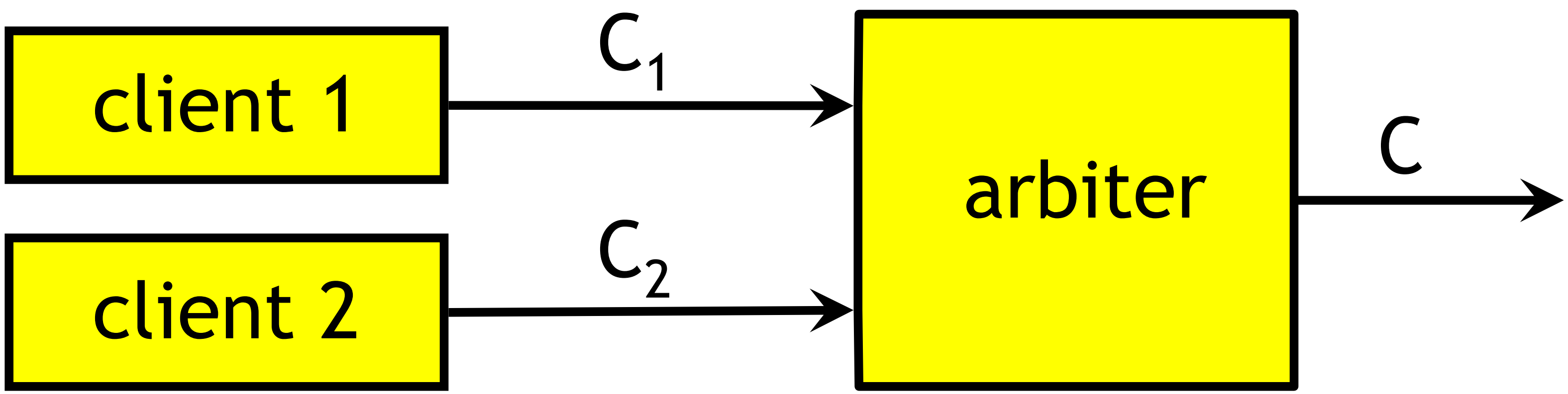
- Used in guards of the passive side only
- Checks if active side waits (for sending v) on C
- Active side is blocked in case of a probe:
  - Cannot change v before acknowledgement
  - Cannot emit another value on C
- Example:
  - sender: @[C!2; break]
  - receiver: @[C#2 ⇒ x := x+1; loop
x>10 ⇒ C?y; break]



Example: Simple Two-Way Arbiter

Three processes:

- client 1: $@[\ C_1!;\ \text{loop}\ ]$
- client 2: $@[\ C_2!;\ \text{loop}\ ]$
- arbiter: $@[\ C_1\# \Rightarrow (C_1?,\ C!1);\ \text{loop}$
  $C_2\# \Rightarrow (C_2?,\ C!2);\ \text{loop}\ ]$



Outline of the Talk

- Hardware process algebra – CHP
- Operational semantics for CHP
- Translation of CHP into LOTOS
- Conclusion and future work

Operational Semantics

- Semantics defined in SOS style
- State: $\langle$Variables, Processes$\rangle$
- Model channel C as a variable x_c
  - Modified only by active process for C
  - Read by passive process for C
- Simulate handshake protocol:
  - Request: modification of x_c
  - Acknowledgement: communication and reset of x_c

  $\Rightarrow$ Two transitions for each communication
- Probe as read access to x_c

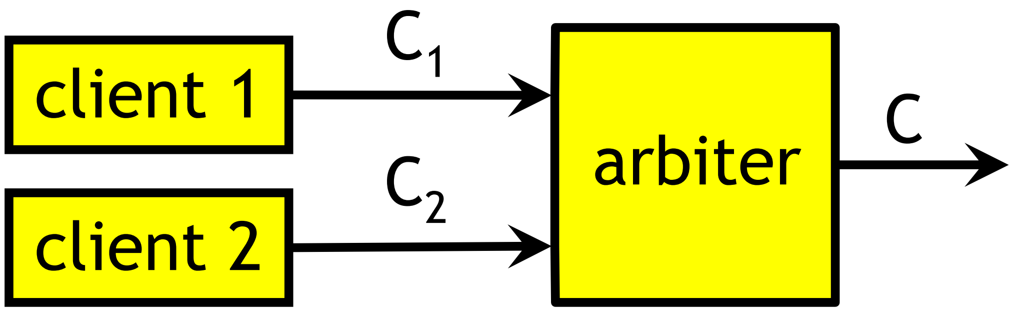
Arbiter: Operational Semantics

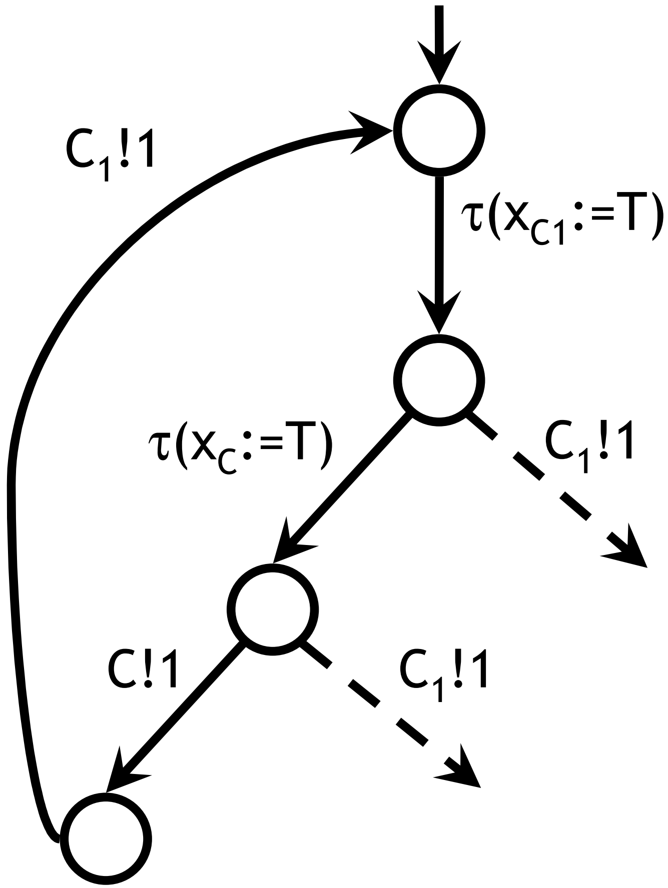
Two-way arbiter:

- client 1: @[C_1!; loop]
- client 2: @[C_2!; loop]
- arbiter:

  @[C_1# $\Rightarrow$ (C_1?, C!1); loop

  C_2# $\Rightarrow$ (C_2?, C!2); loop]





Interaction with client 1 only

Outline of the Talk

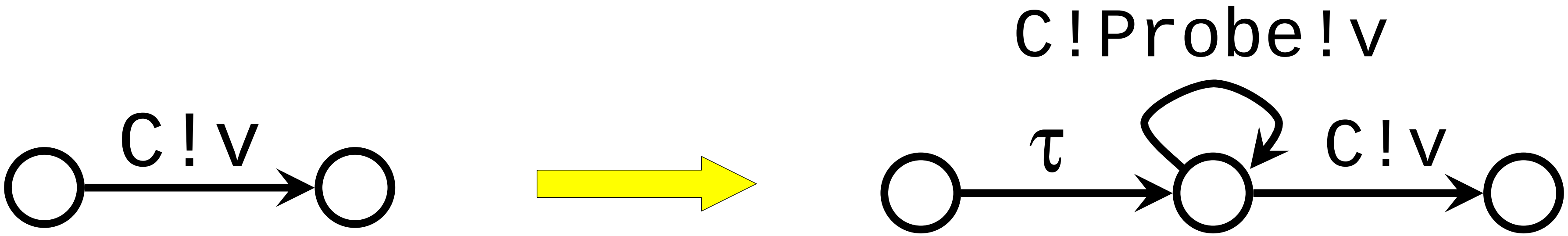
- Hardware process algebra – CHP
- Operational semantics for CHP
- Translation of CHP into LOTOS
- Conclusion and future work

CHP to LOTOS: Overview

- In this work, translation of:
  - the behaviour of processes
  - basic datatypes: Nat, Int, Bool
- Main differences between CHP and LOTOS:
  - looping guarded commands *vs* recursive processes
  - symmetrical *vs* assymmetrical sequential composition
  - implicit termination *vs* explicit termination
  - implicit *vs* explicit (exit, >>) variable passing
- Prototype chp2lotos:
  - total of 10.000 lines of SYNTAX, LOTOS NT, and C
  - validated on about 300 CHP files

CHP to LOTOS: Probe
Probe has to be a communication!

- Active side:
  - allows an arbitrary number of probes
    $\Rightarrow$ loop until acknowledgment

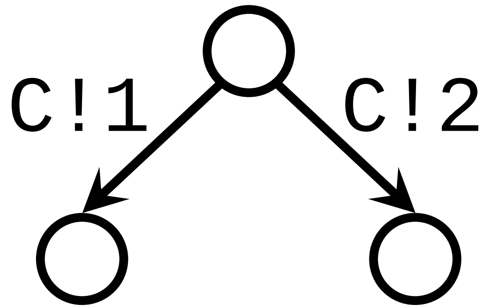


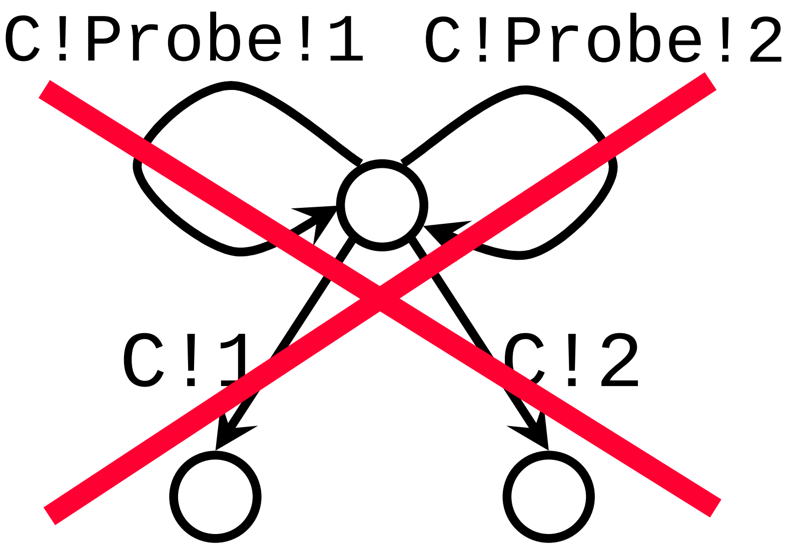
- Passive side:
  - a reception `C?x` is translated as is
  - a probe `C#v` is a value matching on `v`: `C!Probe!v`

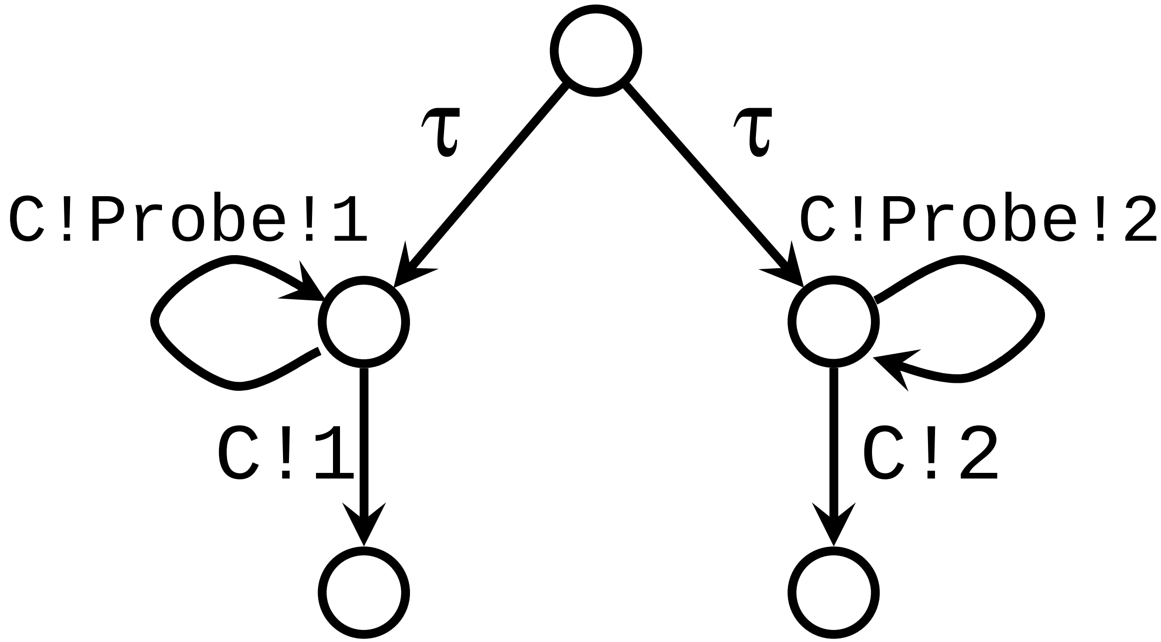
CHP to LOTOS: Probe

Why do we need a τ prefix?

No change of value during communication







Arbiter in LOTOS – Clients

```
process client1[C1]: noexit :=
 τ; probed_C1[C1](1) >> client1[C1]
endproc

process probed_C1[C1](x): exit :=
    C1!x; exit
    []
    C1!Probe!x; probed_C1[C1](x)
endproc
```

Arbiter in LOTOS – Arbiter

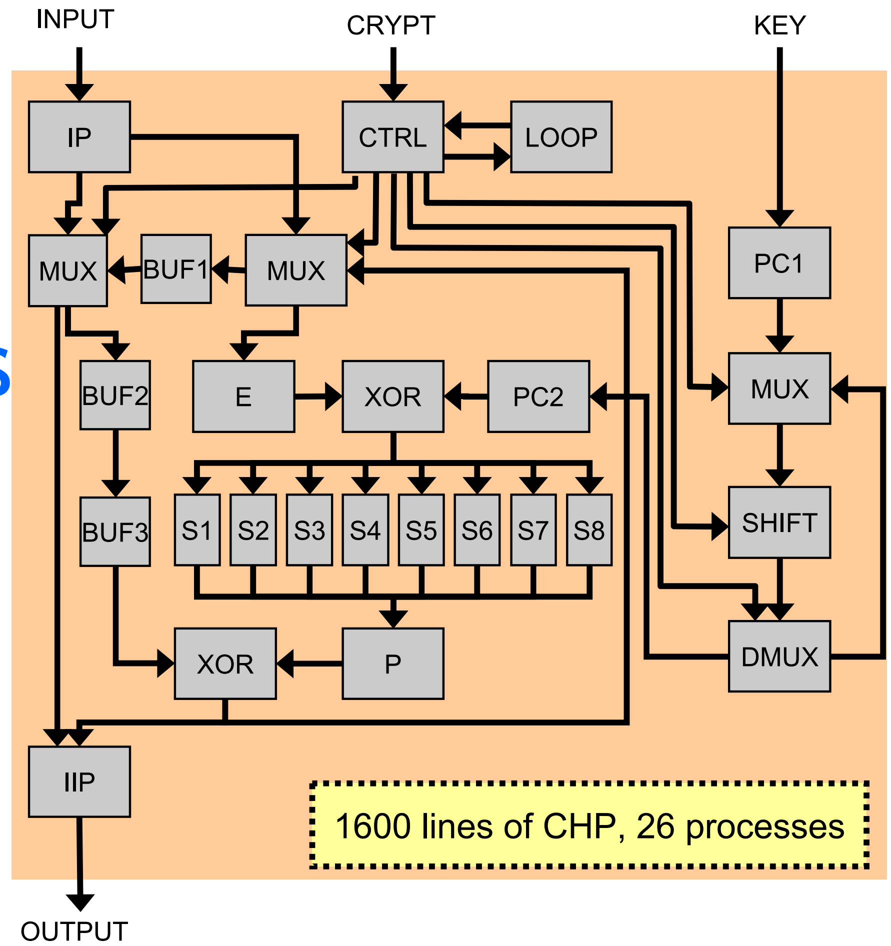
```
process arbiter[C,C1,C2]: noexit :=
   (
      C1!Probe!1;
         (
            C1?x; exit
            |||
           τ; probed_C[C](1)
         ) >> arbiter[C,C1,C2]
   )
   [] …    (* similar for client 2 *)
endproc
```

Arbiter in LOTOS – System

```
process main[C]: noexit :=
 hide C1, C2 in
   client1[C1]
   |[C1]|
   (
      client2[C2]
      |[C2]|
      arbiter[C,C1,C2]
   )
endproc
```

Case Study: Asynchronous DES

- DES (Data Encryption Standard)
- Asynchronous circuit designed by CEA
- Control-flow only
- Huge state space ($>10^8$ states)
- Compositional techniques (of CADP) successful
  - within 10 minutes
  - 16.910 st., 85.840 trans.
  - model-checking of control properties



Outline of the Talk

- Hardware process algebra – CHP
- Operational semantics for CHP
- Translation of CHP into LOTOS
- Conclusion and future work

Conclusion

- Operational Semantics for CHP:

  $\Rightarrow$ Generalizes existing semantics via Petri nets
- High-level translation of CHP into LOTOS

Future Work

- Translation of advanced features:
  - full support of CHP's predefined datatypes
  - multi-probes such as `@[ C`$_1$`#v`$_1$` or C`$_2$`#v`$_2$` => … ]`
- $LTS_{CHP} \cong LTS_{LOTOS}$: which equivalence?
- Application: verification of a NoC circuit