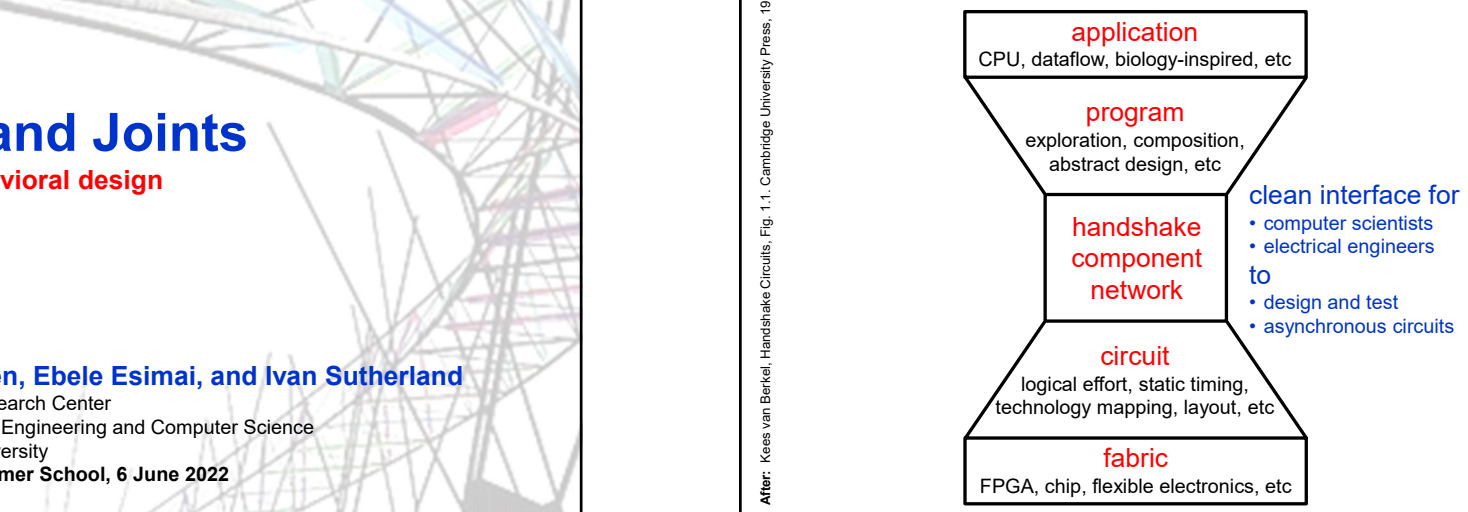
Links and Joints

PART 1: behavioral design

Marly Roncken, Ebele Esimai, and Ivan Sutherland
Asynchronous Research Center
Maseeh College of Engineering and Computer Science
Portland State University
ASYNC 2022 Summer School, 6 June 2022

1

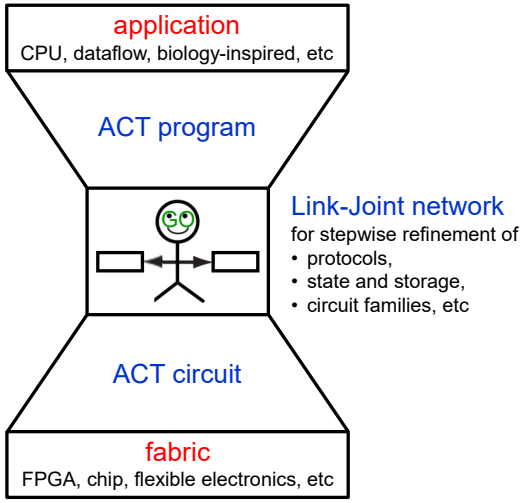
Context



After: Kees van Berkel, Handshake Circuits, Fig. 1.1. Cambridge University Press, 1993.

5

Context



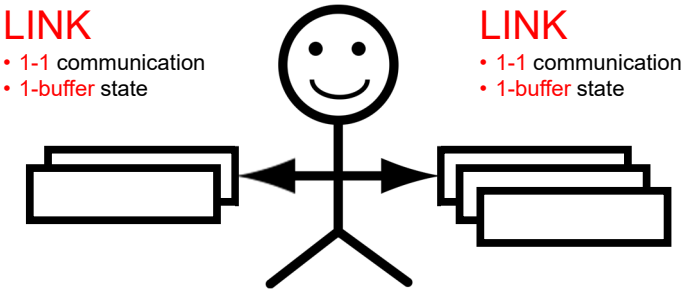
After: Kees van Berkel, Handshake Circuits, Fig. 1.1. Cambridge University Press, 1993.

6

Links and Joints

7

Links and Joints



• computation
• flow control

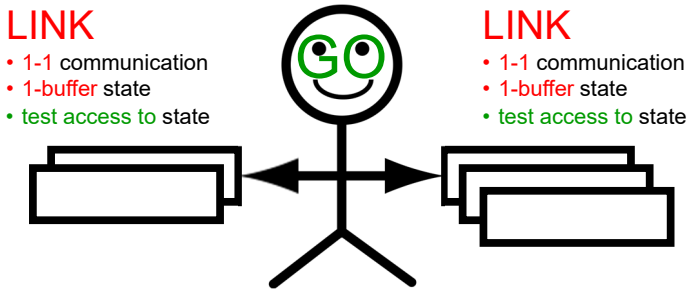
JOINT

"a place where Links meet to exchange information"

slide 9 of 38

9

Links and Joints



• Joint-specific (no)GO
• computation
• flow control

JOINT

"a place where Links meet to exchange information"

slide 10 of 38

10

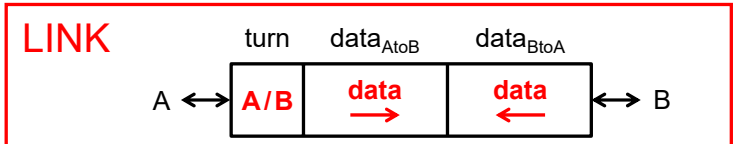
Behavioral model

[Esimai-Roncken, Flexible Active-Passive and Push-Pull Protocols, IEEE ESL, 2022]

slide 11 of 38

11

Behavioral model



• Two ports: A, B
• State with 3 variables shared between the ports
  • turn : indicates which port may change the Link state
  • $data_{AtoB}$: ≥0 data bits, for A to write, B to read
  • $data_{BtoA}$: ≥0 data bits, for B to write, A to read

• data validity:
  • $data_{AtoB}$: valid and stable when turn=B (receiver of data)
  • $data_{BtoA}$: valid and stable when turn=A (receiver of data)

slide 15 of 38

15

Behavioral model

JOINT

- guarded command specification:

  myturn(in) ∧ myturn(out) ∧ **GO**
  →
  myW(out) := myR(in)
  yourturn(in)
  yourturn(out)

- terminology:
  - myturn(p) : TRUE if p has permission to change the Link state
  - myR(p) : Link data that p can read
  - myW(p) : Link data that p can write
  - yourturn(p) : relinquish permission
  - **GO** : external signal for initialization and test

- interleaving model:
  arbitrarily select a guarded command with TRUE guard
- atomicity:
  Link states update all at once when the command terminates

20

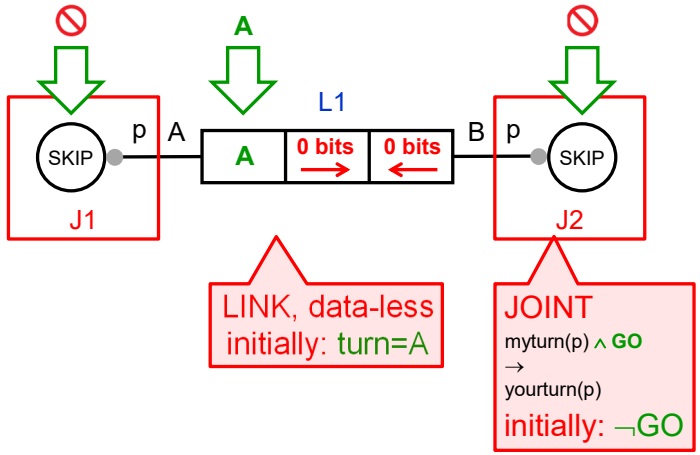
Time for 1-2 questions

21

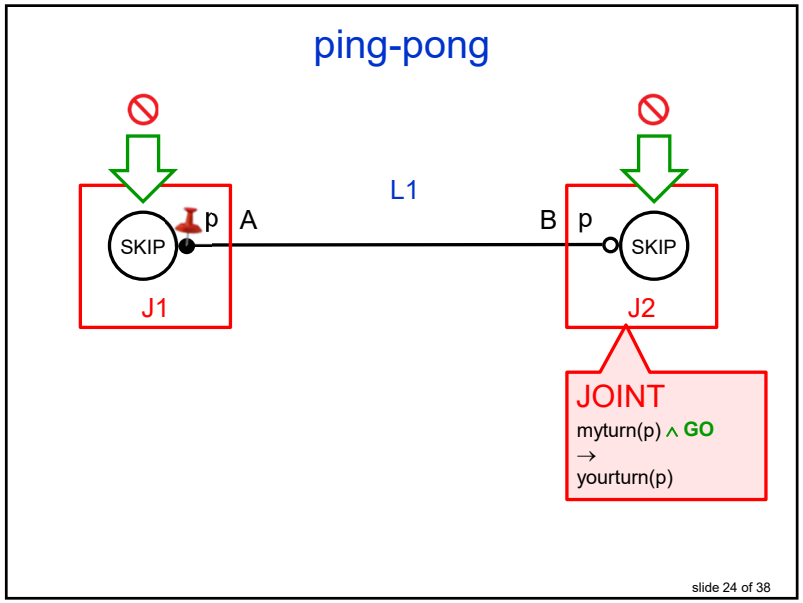
Example: ping-pong

22

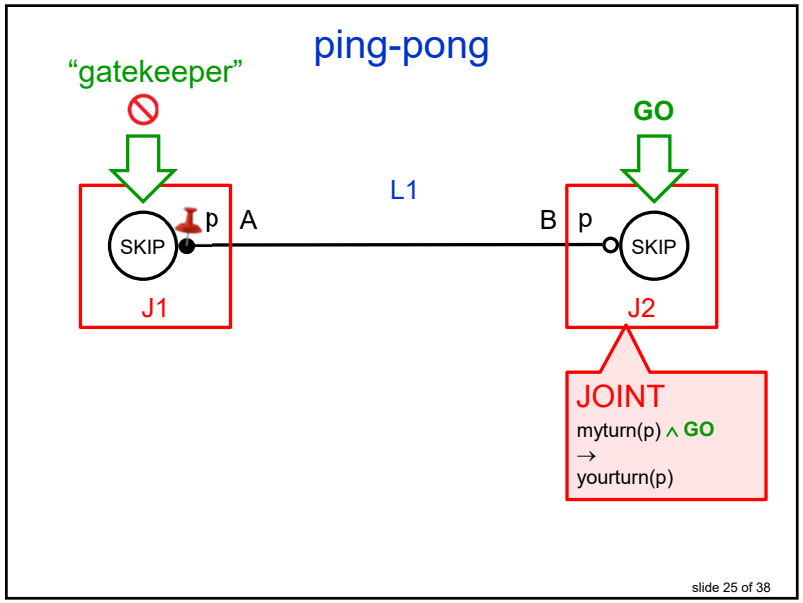
ping-pong



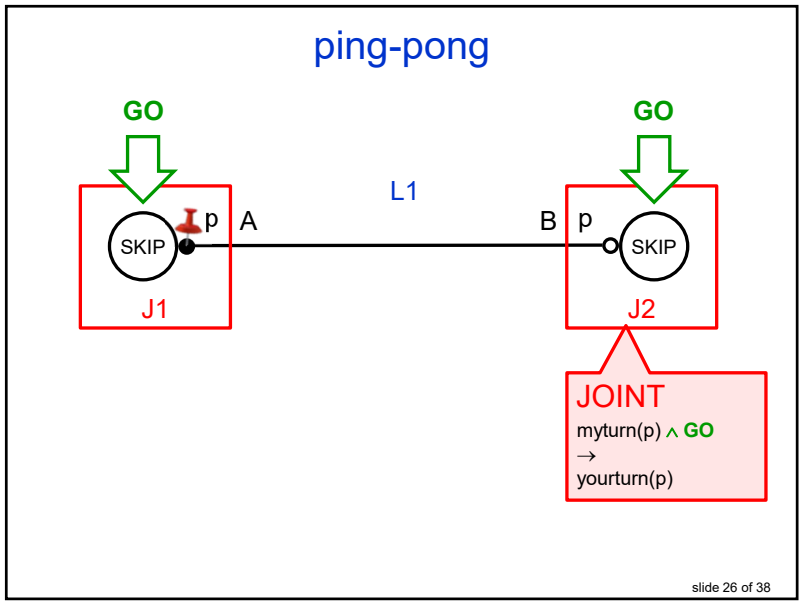
23


ping-pong
L1
p
A
B
p
SKIP
SKIP
J1
J2
JOINT
myturn(p) ∧ GO
→
yourturn(p)
slide 24 of 38

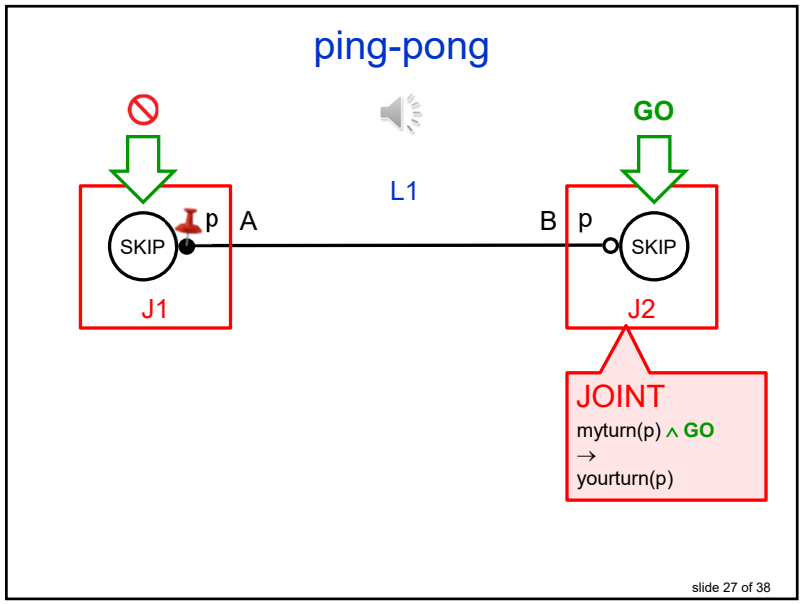
24


ping-pong
“gatekeeper”
GO
L1
p
A
B
p
SKIP
SKIP
J1
J2
JOINT
myturn(p) ∧ GO
→
yourturn(p)
slide 25 of 38

25


ping-pong
GO
GO
L1
p
A
B
p
SKIP
SKIP
J1
J2
JOINT
myturn(p) ∧ GO
→
yourturn(p)
slide 26 of 38

26


ping-pong
GO
L1
p
A
B
p
SKIP
SKIP
J1
J2
JOINT
myturn(p) ∧ GO
→
yourturn(p)
slide 27 of 38

27

Time for 1-2 questions

slide 28 of 38

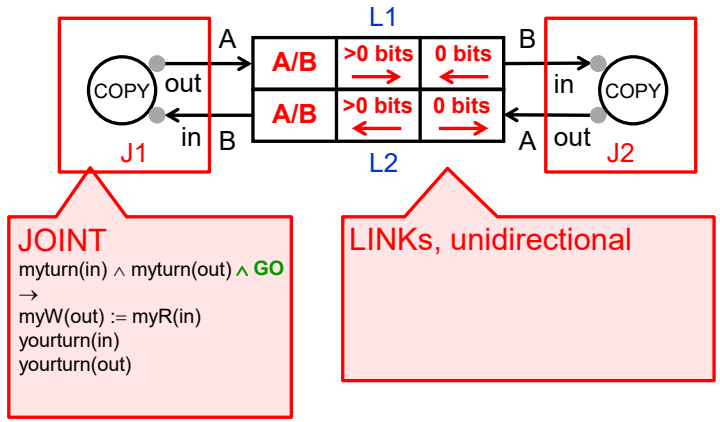
28

Example: ring-FIFO

slide 29 of 38

29

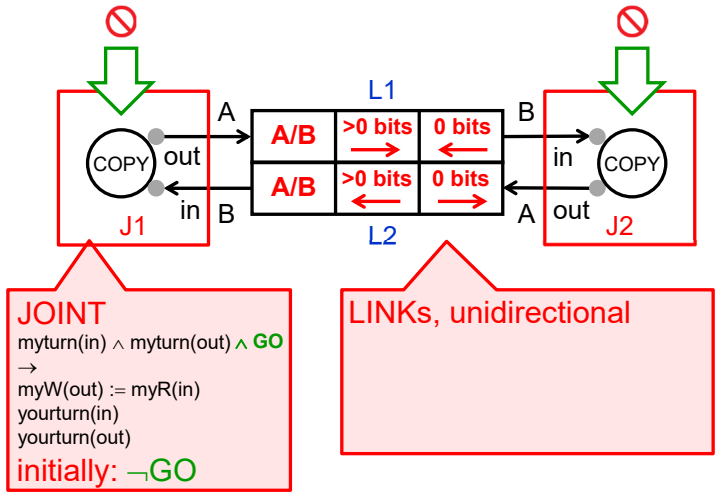
ring-FIFO



slide 31 of 38

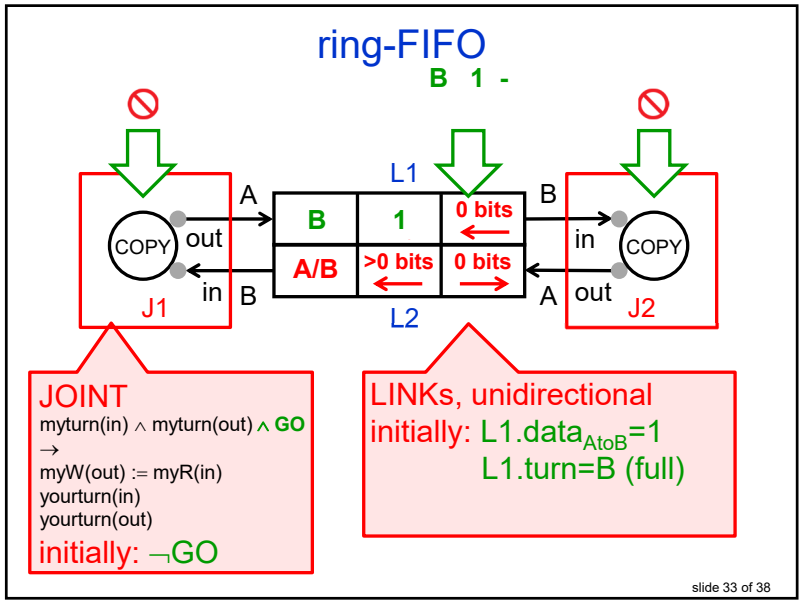
31

ring-FIFO

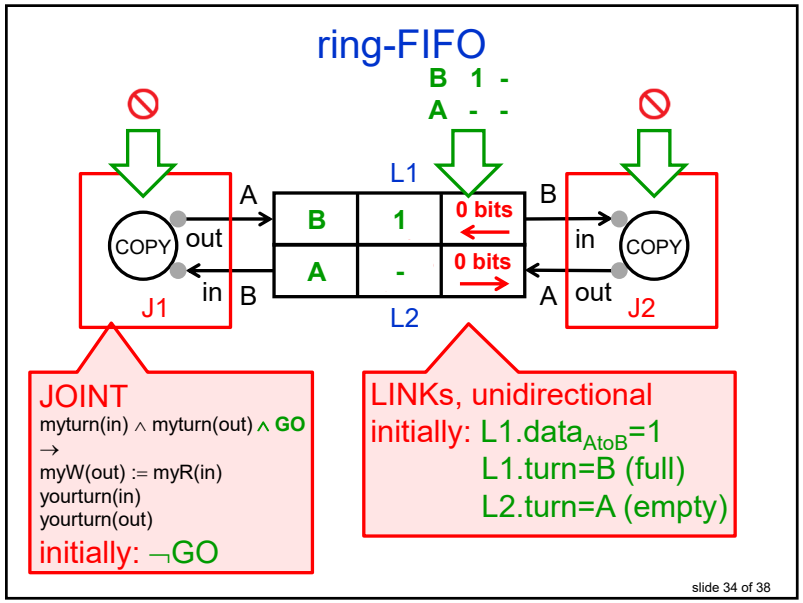


slide 32 of 38

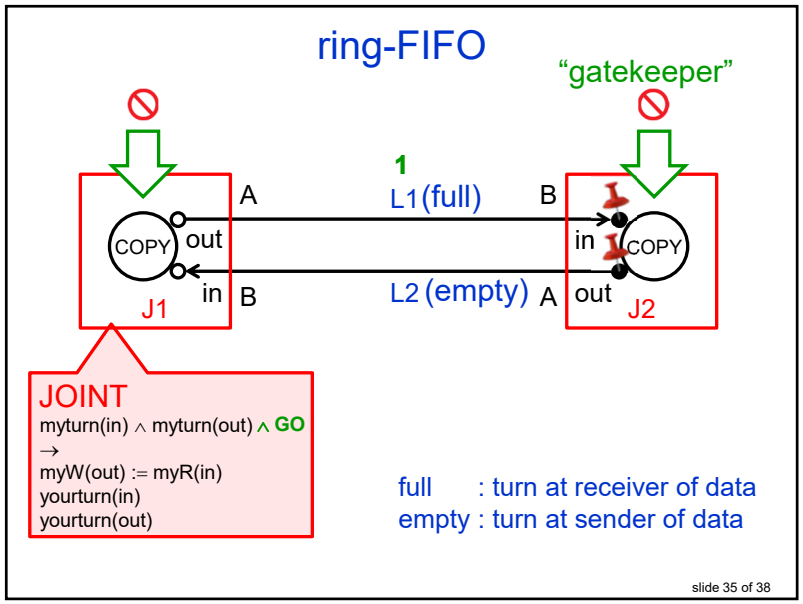
32


ring-FIFO
B 1 -
L1
A
B
1
0 bits
B
COPY
out
in
B
A/B
>0 bits
0 bits
in
COPY
A
out
J2
J1
L2
JOINT
myturn(in) ∧ myturn(out) ∧ GO
→
myW(out) := myR(in)
yourturn(in)
yourturn(out)
initially: ¬GO
LINKs, unidirectional
initially: L1.data$_{AtoB}$=1
L1.turn=B (full)
slide 33 of 38

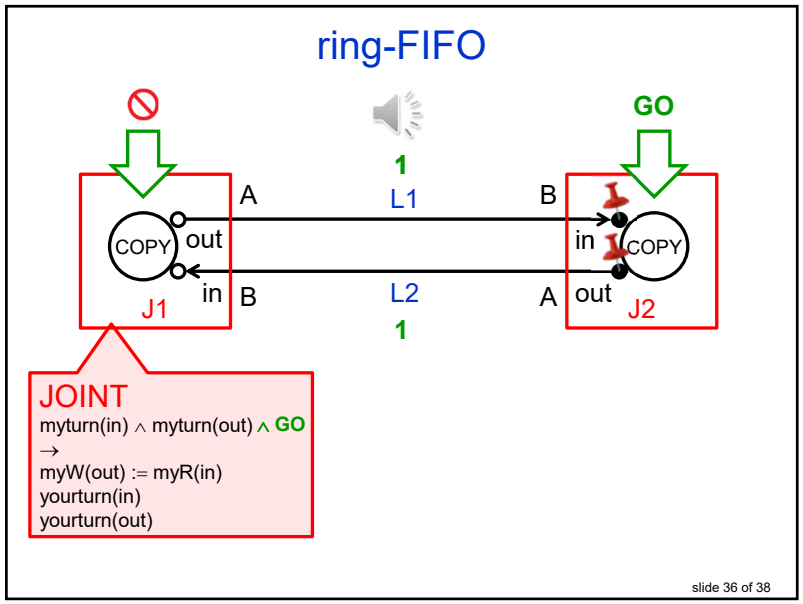
33


ring-FIFO
B 1 -
A - -
L1
A
B
1
0 bits
B
COPY
out
in
B
A
-
0 bits
in
COPY
A
out
J2
J1
L2
JOINT
myturn(in) ∧ myturn(out) ∧ GO
→
myW(out) := myR(in)
yourturn(in)
yourturn(out)
initially: ¬GO
LINKs, unidirectional
initially: L1.data$_{AtoB}$=1
L1.turn=B (full)
L2.turn=A (empty)
slide 34 of 38

34

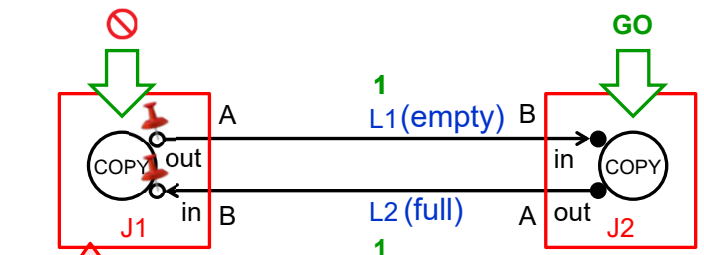

ring-FIFO
"gatekeeper"
1
L1 (full)
A
B
COPY
out
in
B
in
COPY
L2 (empty)
A
out
J2
J1
JOINT
myturn(in) ∧ myturn(out) ∧ GO
→
myW(out) := myR(in)
yourturn(in)
yourturn(out)
full : turn at receiver of data
empty : turn at sender of data
slide 35 of 38

35


ring-FIFO
GO
1
L1
A
B
COPY
out
in
B
in
COPY
L2
A
out
J2
J1
1
JOINT
myturn(in) ∧ myturn(out) ∧ GO
→
myW(out) := myR(in)
yourturn(in)
yourturn(out)
slide 36 of 38

36

ring-FIFO



JOINT
myturn(in) ∧ myturn(out) ∧ **GO**
→
myW(out) := myR(in)
yourturn(in)
yourturn(out)

full : turn at receiver of data
empty : turn at sender of data

slide 37 of 38

37

38

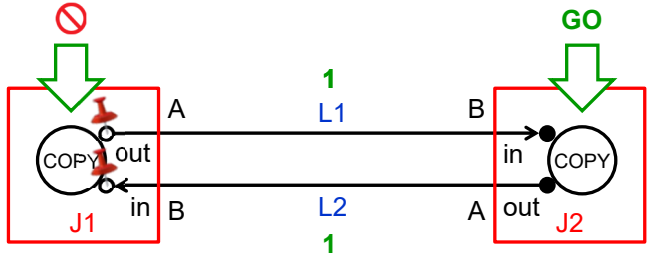
Exercise

[Roncken et al., Naturalized Communication and Testing, ASYNC 2015]

slide 39 of 38

39

Exercise 1: test a Joint's data operation



- Starting from the above state (the end state of the previous ring-FIFO run)
- re-initialize the design to test if Joint J2 also copies
  - data value 0
- using external access to any
  - Joint.GO
  - Link.turn
  - Link.data$_{AtoB}$
  - Link.data$_{BtoA}$

slide 40 of 38

40

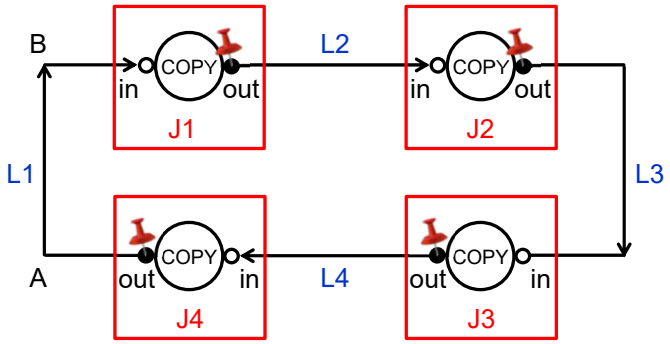
Exercise

[Roncken et al., Naturalized Communication and Testing, ASYNC 2015]

slide 41 of 38

41

Exercise 2: initialize with PARTIAL state access

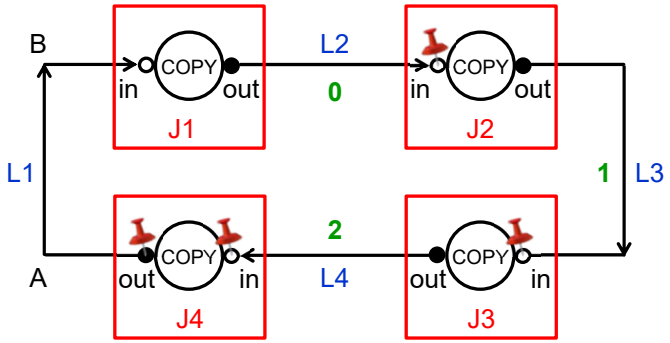


- Starting from the above initial state with all Links empty
- obtain the state in the next slide

slide 42 of 38

42

Exercise 2: continued ...



- Use external access to any Joint.GO and Link.turn but to ONLY L1.data$_{AtoB}$
- to get the pins and the data in the positions indicated above
- starting from the state in the previous slide

slide 43 of 38

43

Link-Joint References

see also: https://arc.cecs.pdx.edu/publications

- Ebele Esimai and Marly Roncken,
  Flexible Active-Passive and Push-Pull Protocols,
  IEEE Embedded Systems Letters (ESL), 2022.
- Marly Roncken and Ivan Sutherland,
  Design and test of high-speed asynchronous circuits,
  Jia Di and Scott C. Smith (Eds.): Asynchronous Circuit Applications, Chapter 7,
  The Institute of Engineering and Technology (IET), London, UK, pages 113-171, 2020.
- Marly Roncken, Ivan Sutherland, Chris Chen, Yong Hei, Warren Hunt Jr., and Cuong Chau,
  with Swetha Mettala Gilla, Hoon Park, Xiaoyu Song, Anping He, and Hong Chen,
  How to Think about Self-Timed Systems,
  In Proc. Asilomar Conference on Signals, Systems, and Computers, pages 1597-1604, 2017.
- Marly Roncken, Swetha Mettala Gilla, Hoon Park, Navaneeth Jamadagni,
  Chris Cowan, and Ivan Sutherland,
  Naturalized Communication and Testing,
  In Proc. IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC),
  pages 77-84, 2015.

slide 44 of 38

44