Jens Sparsø

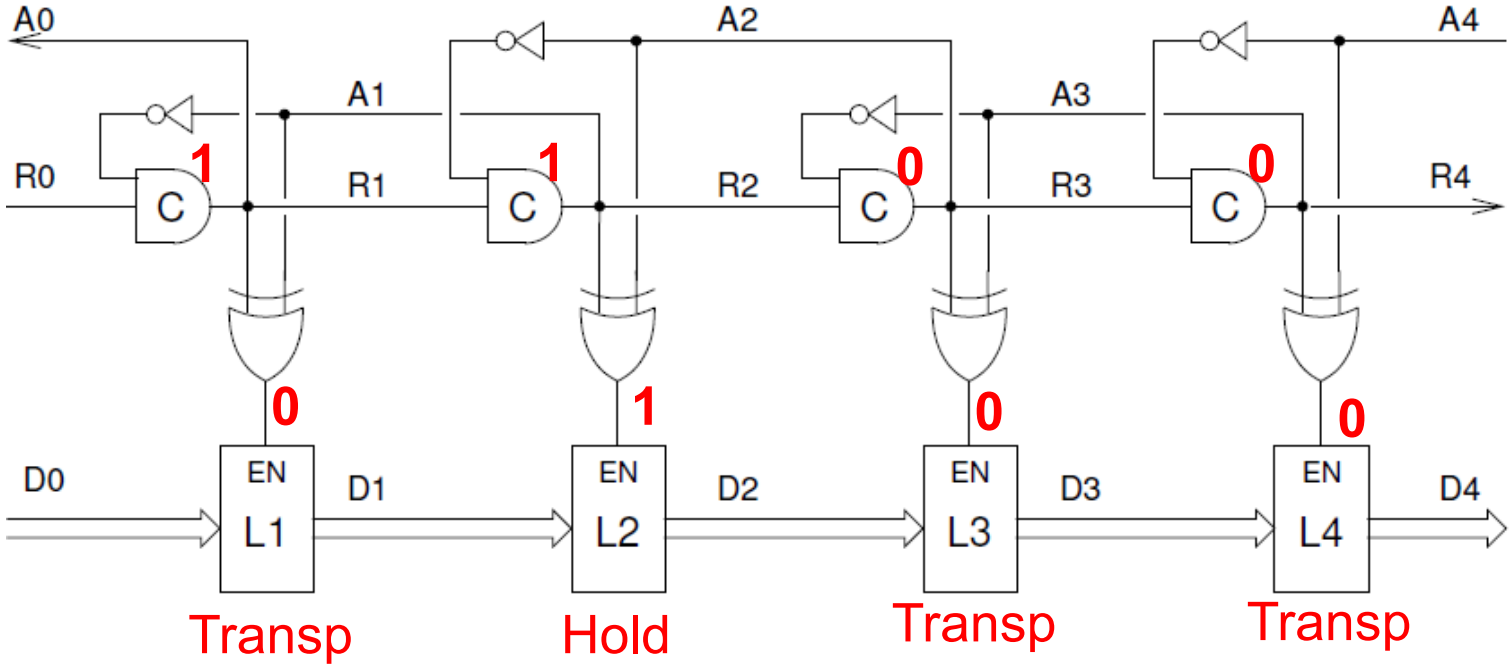
Dataflow Design

ASYNC 2026 Summer School
June 1-2, 2026 @ DTU

Dataflow graphs and Dataflow computation

- Structural method of describing computations by specifying functions and their sequence of operations together graphically
- Intuitive way to convey design intent. Similar to an RTL-schematic
- Not necessarily asynchronous, though async has some nice advantages as a natural mapping of dataflow graphs
- Internal representation used in high-level synthesis tools.
  Source code -> data-flow graph representation -> optimizations/transformations -> circuit
- Jack Dennis, MIT (dataflow pioneer); "*I formed the Computation Structures Group [within CSAIL in 1963] and focused on architectural concepts that could narrow the acknowledged gap between programming concepts and the organization of computer hardware.*"
  – Related stuff: Single assignment code. Pure functional programming.
  Hardware parallelism. Self-timed circuits.

Dataflow graphs and Data flow computation

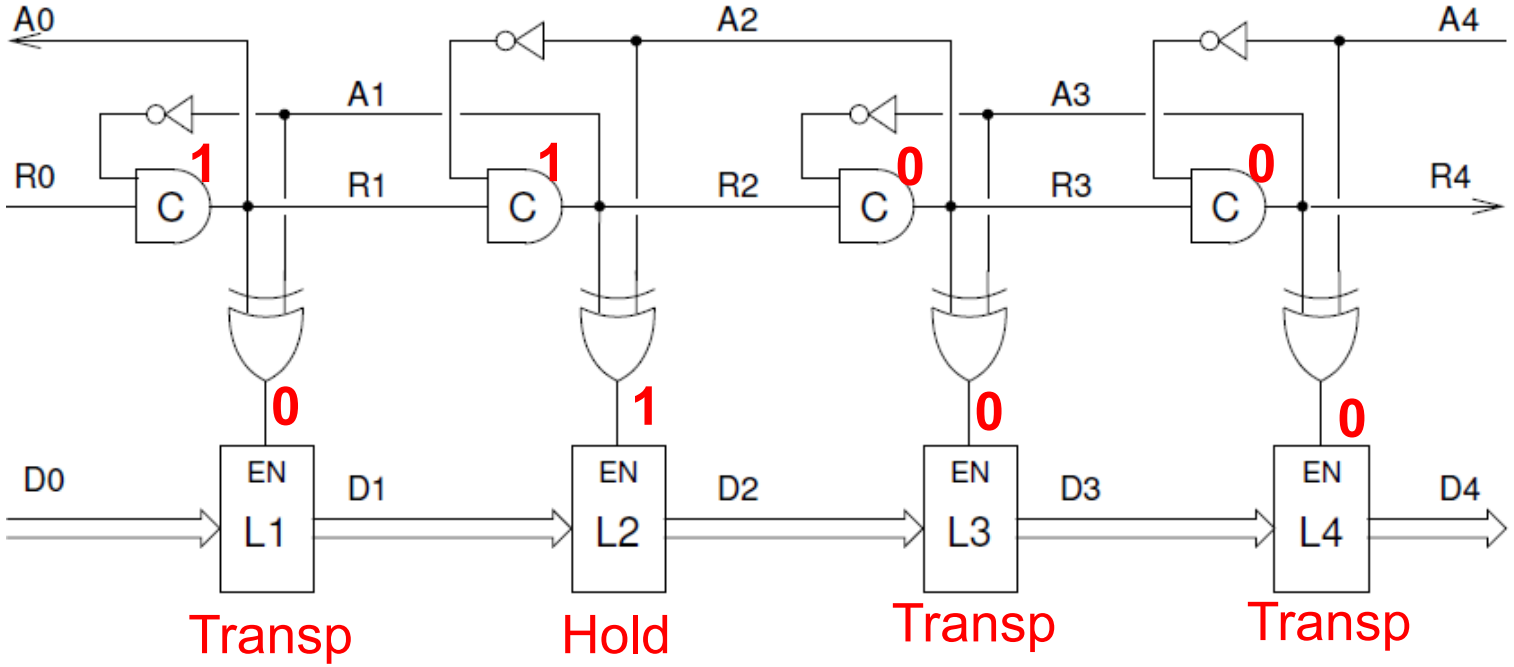


- Snapshot of 2-phase bundled-data pipeline built from transparant latches controlled by a Muller pipeline.
- How do we understand, describe, and analyze its behavior? Timing diagram, hmm ☹

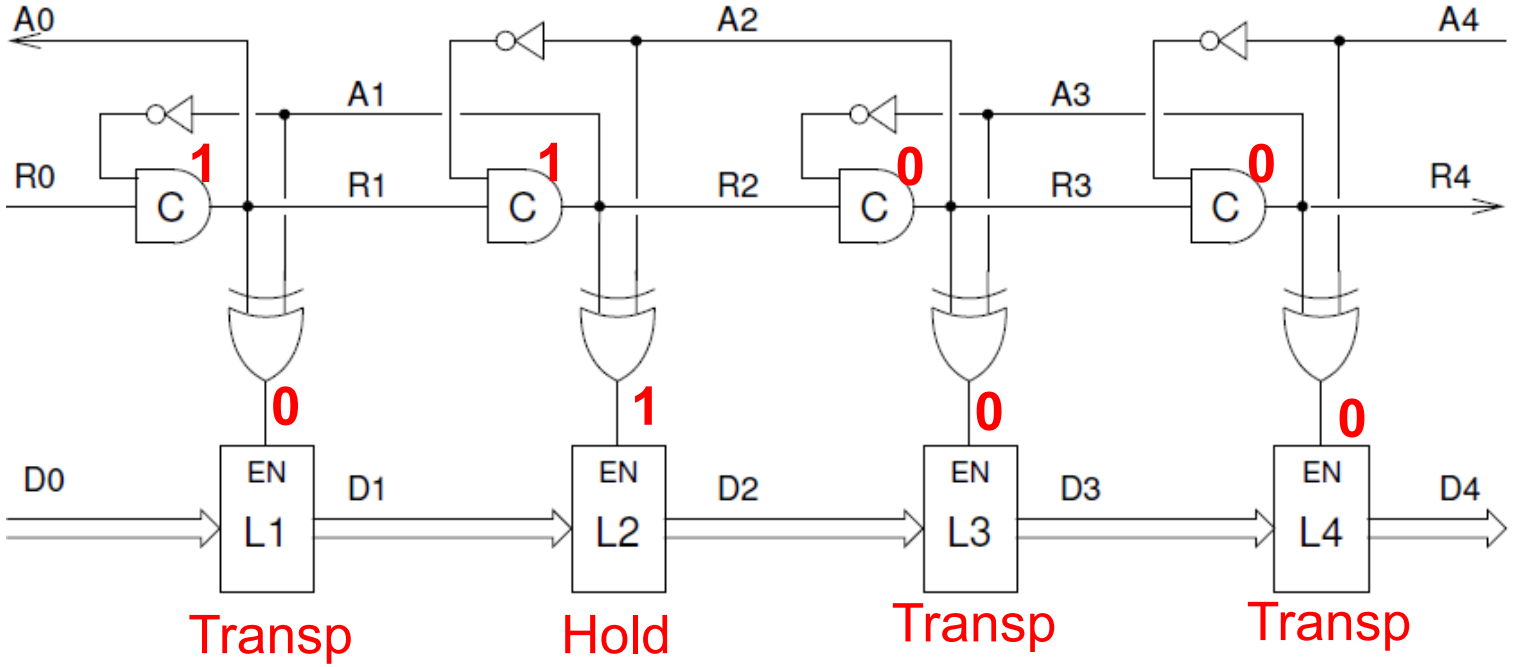
I.E. Sutherland, ”Micropipelines” Communications of the ACM 23(6), pp.720-738, 1989

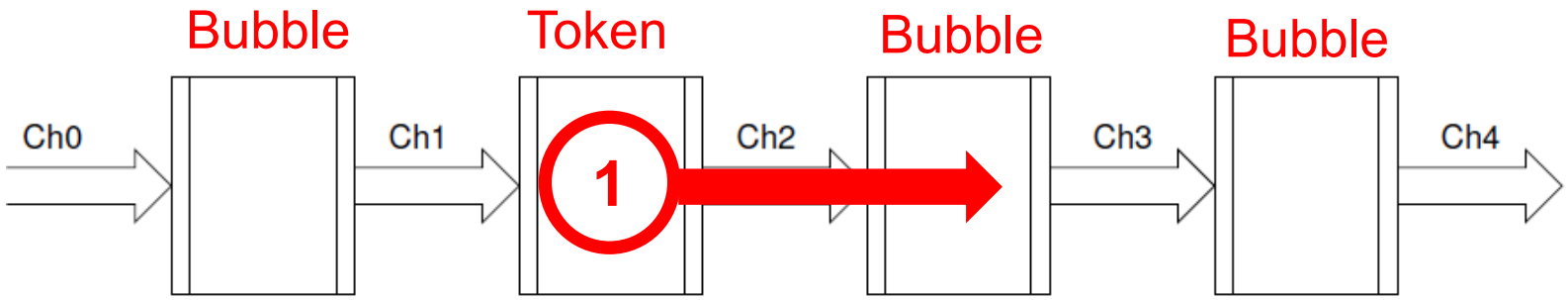
Dataflow (tokens and bubbles)

How?

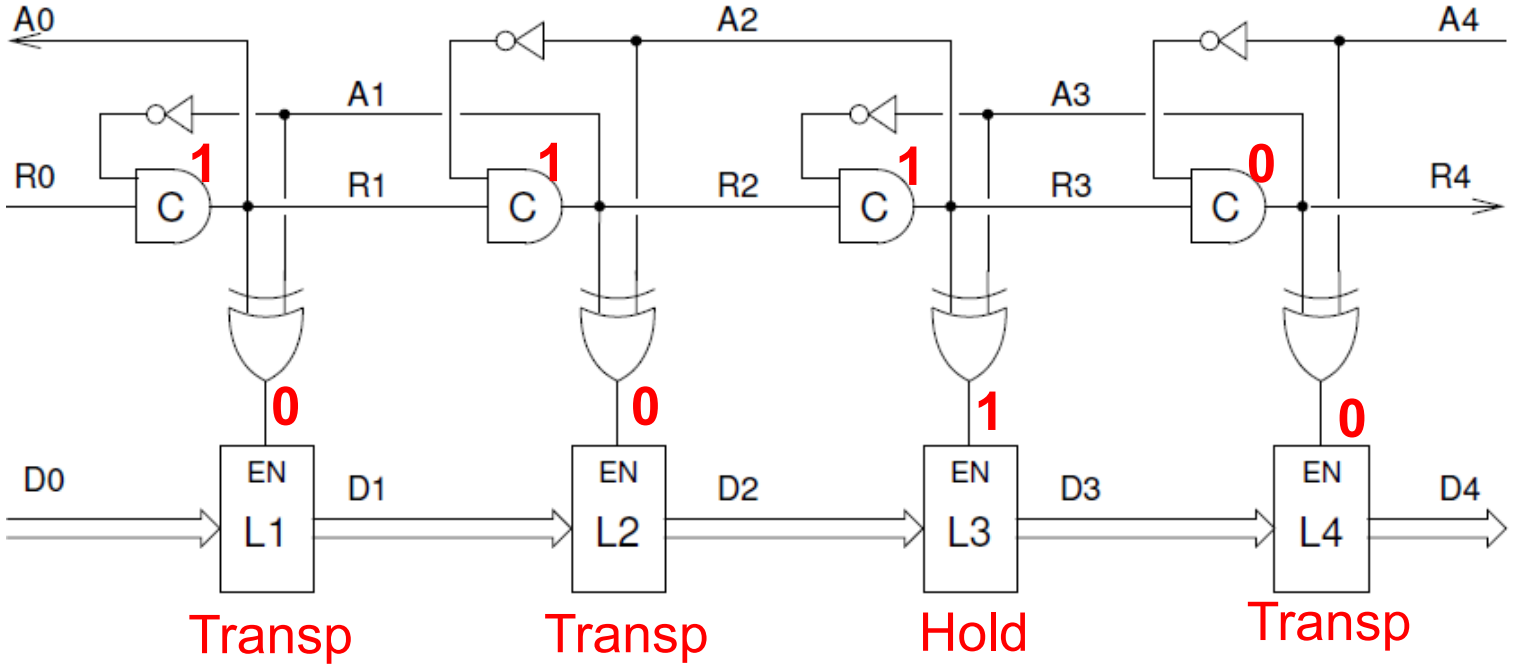


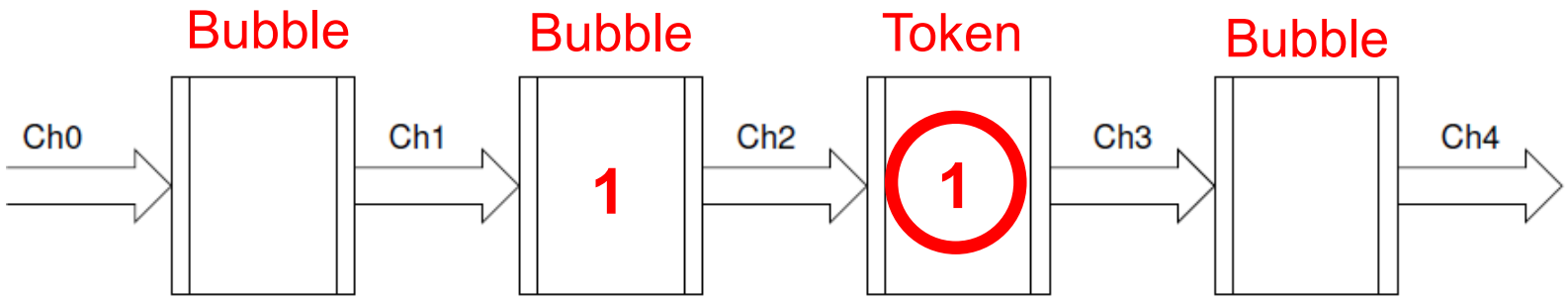
Data-flow (tokens and bubbles)





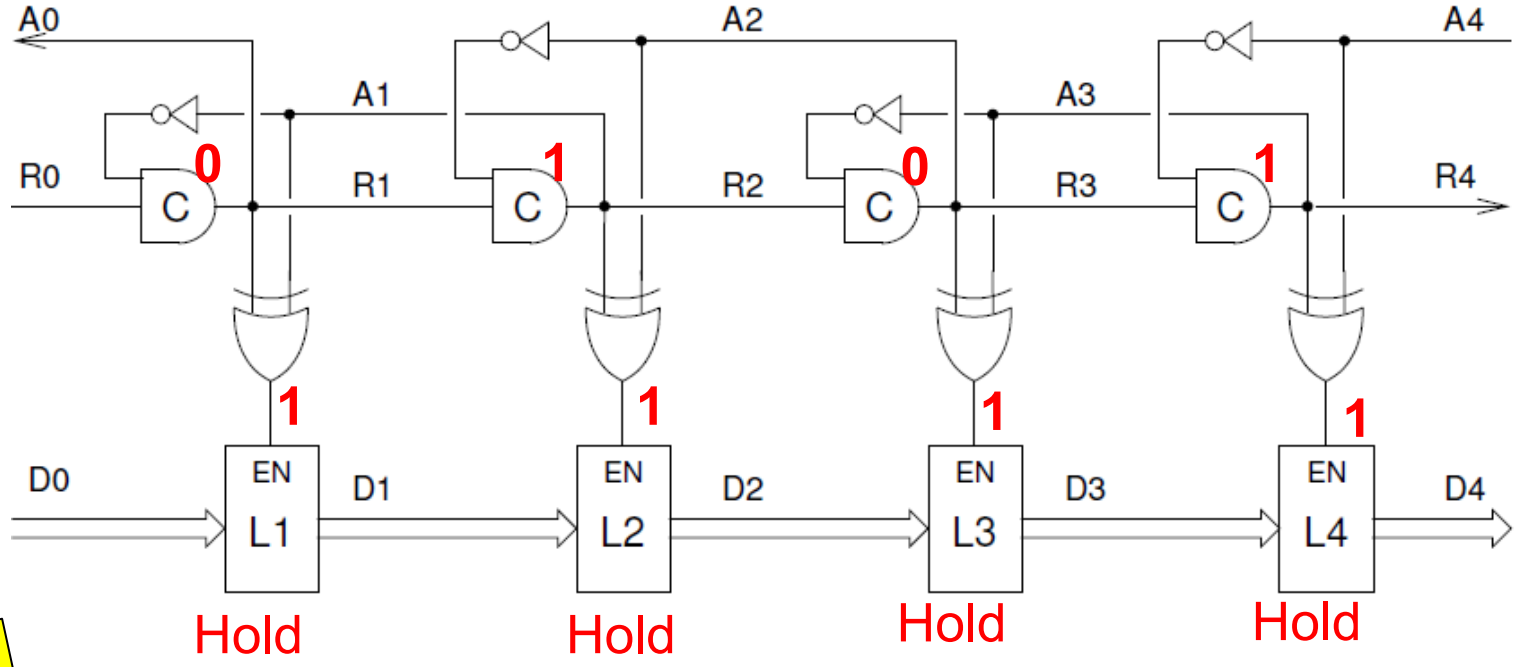
Dataflow (tokens and bubbles)



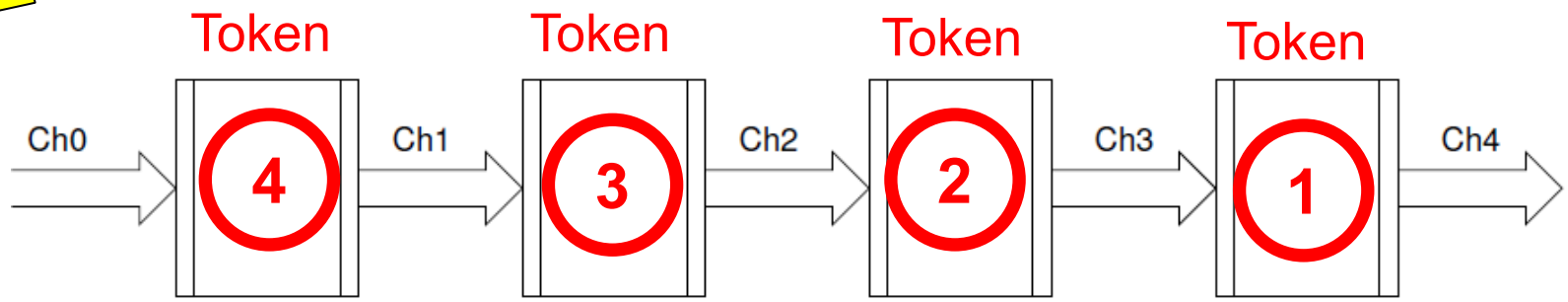


Dataflow (tokens and bubbles)

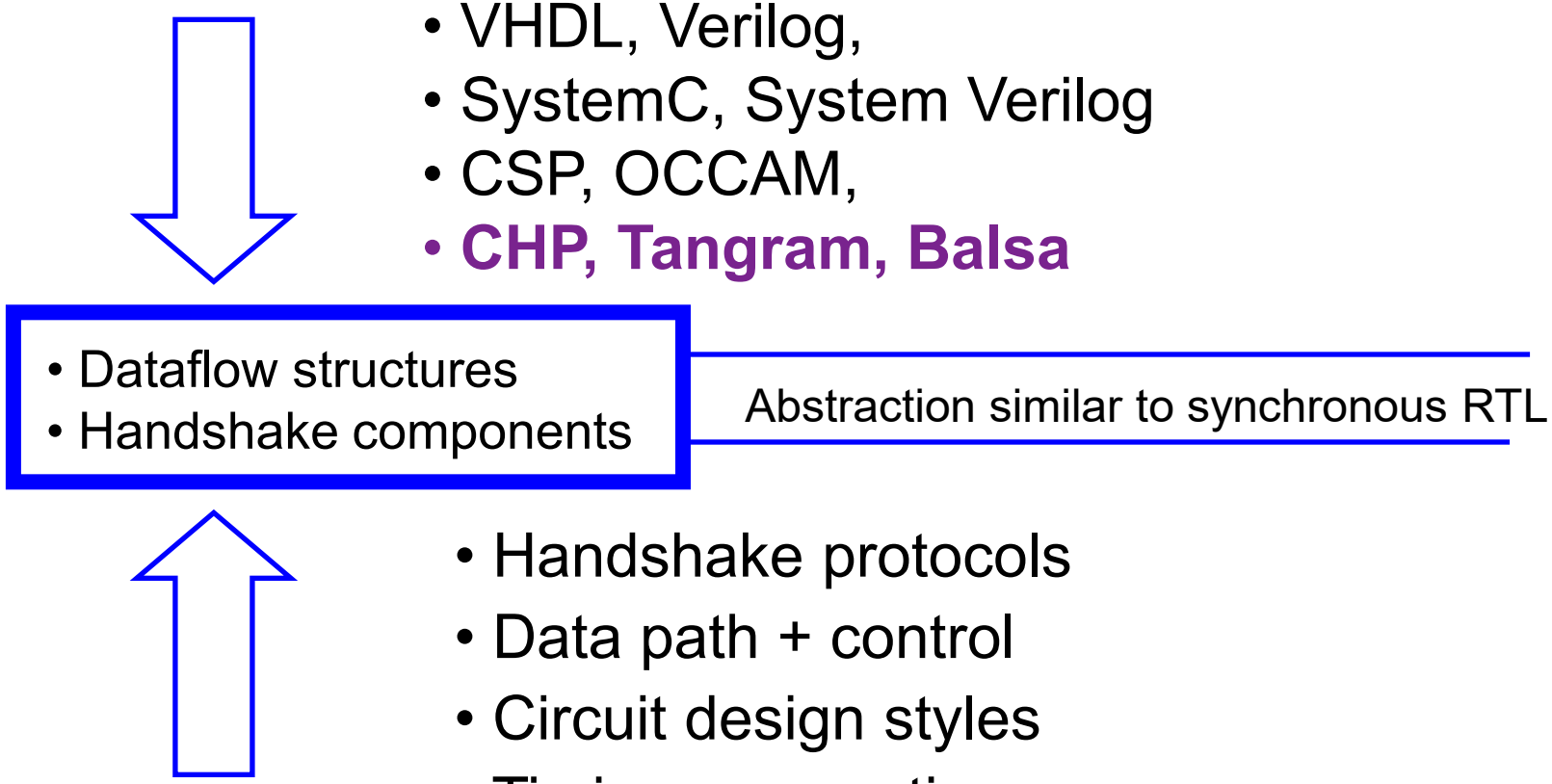
The pipeline may fill



The circuit is both a pipeline and an elastic ripple FIFO

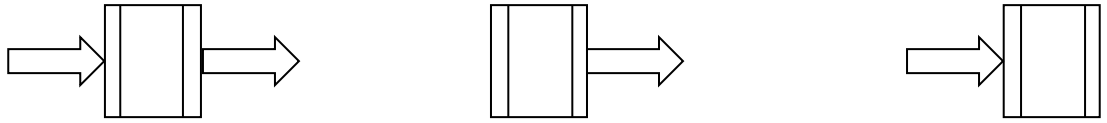


Dataflow and asynchronous circuit design



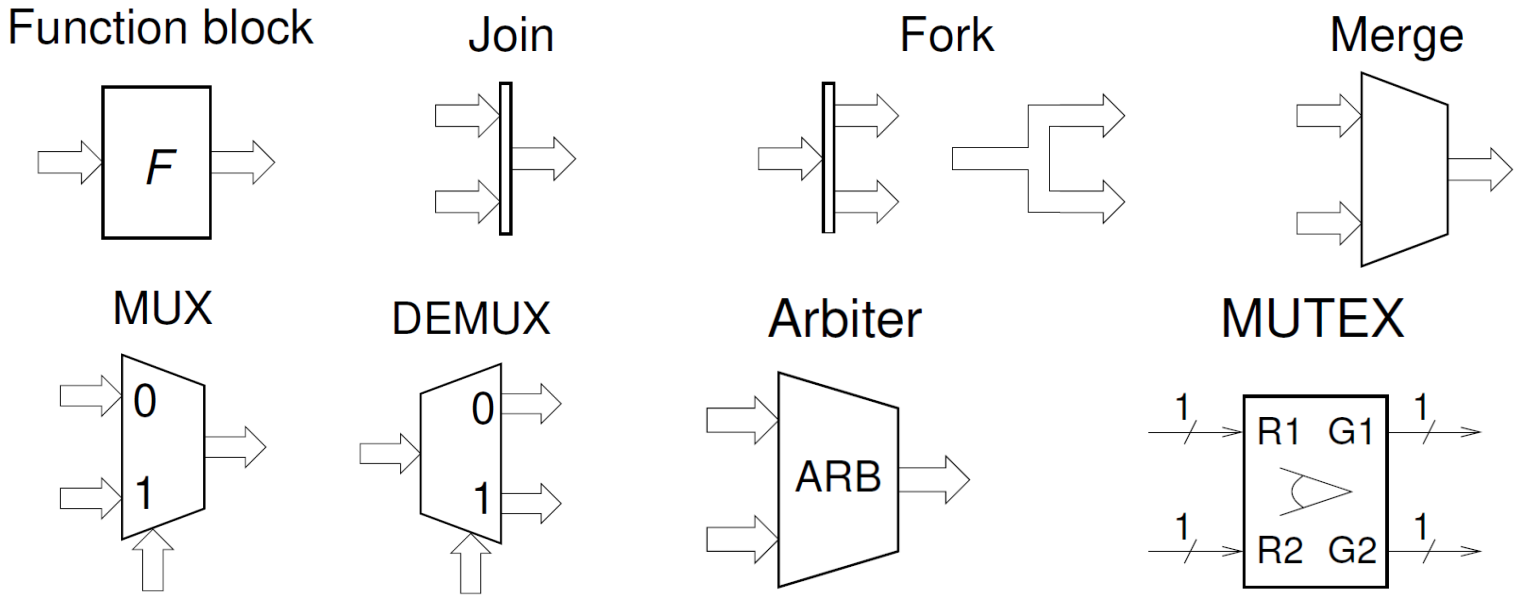
Building Blocks (data-flow components)

Active handshake components (handshake latches aka. buffers):

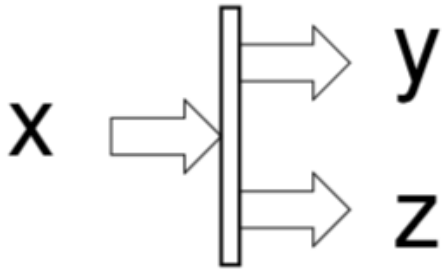


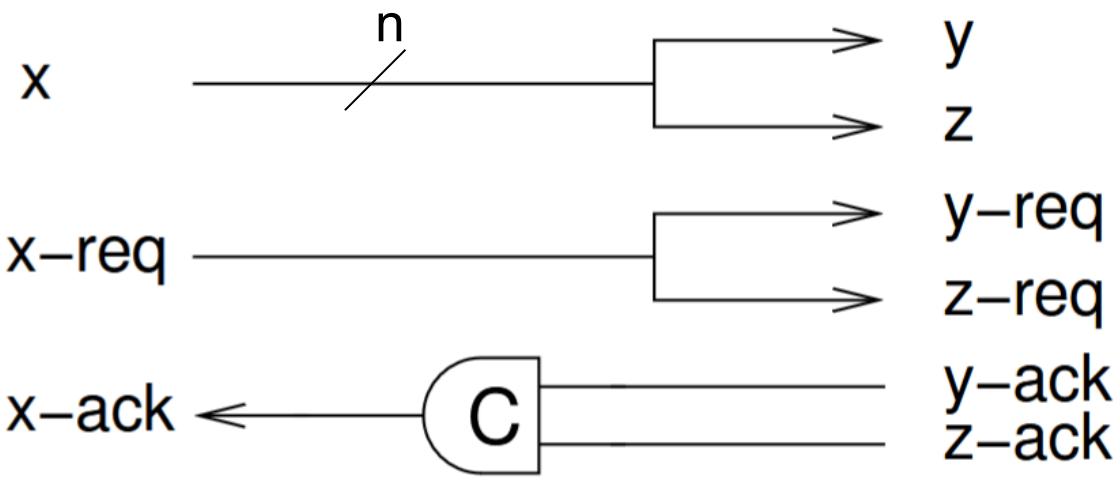
(Handshake) Latch/buffer | Source | Sink

Passive handshake components (all other). Transparent to handshaking:

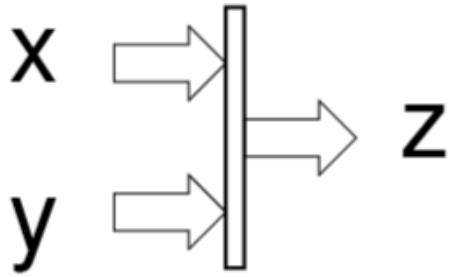


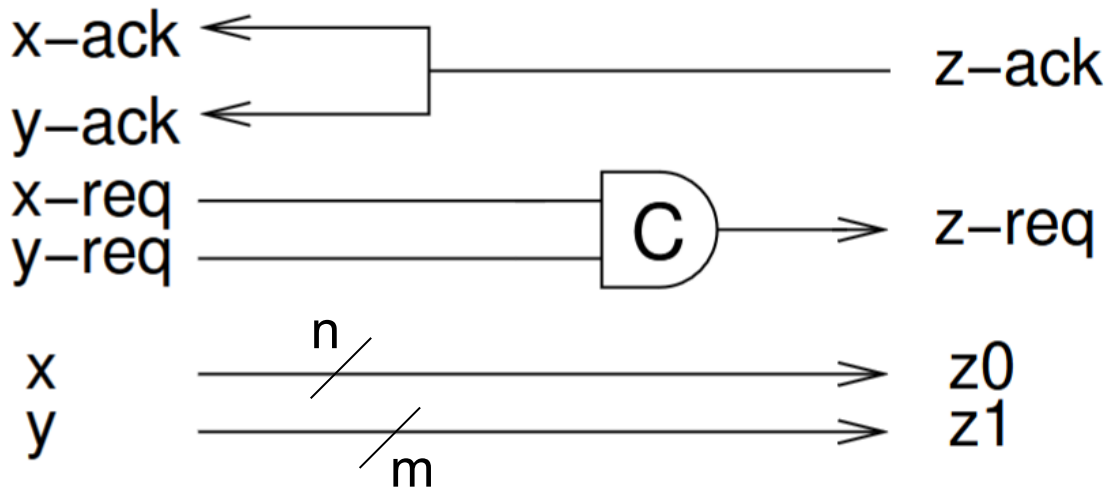
Fork


x
y
z

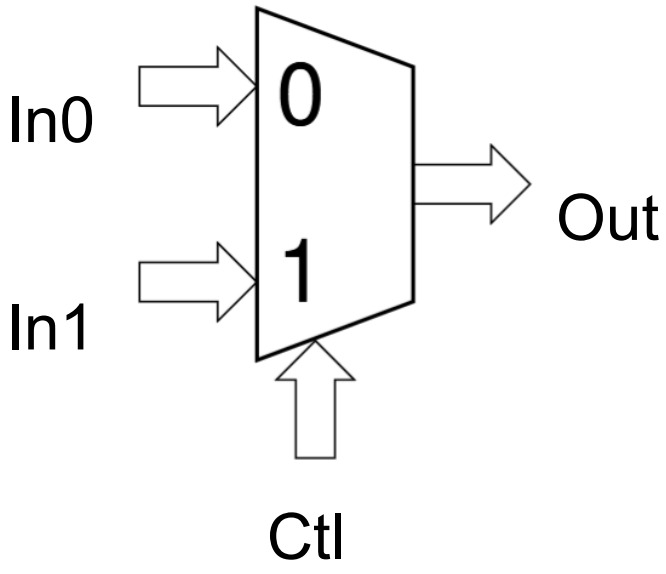

x
n
y
z
x–req
y–req
z–req
x–ack
C
y–ack
z–ack

Join


x
y
z


x–ack
y–ack
z–ack
x–req
y–req
C
z–req
x
y
n
m
z0
z1

MUX



CHP-code from the 2022 summer school

```
*[Ctl?c;
        [  c=0 -> In0?x
        [] c=1 -> In1?x];
  Out!x]
```

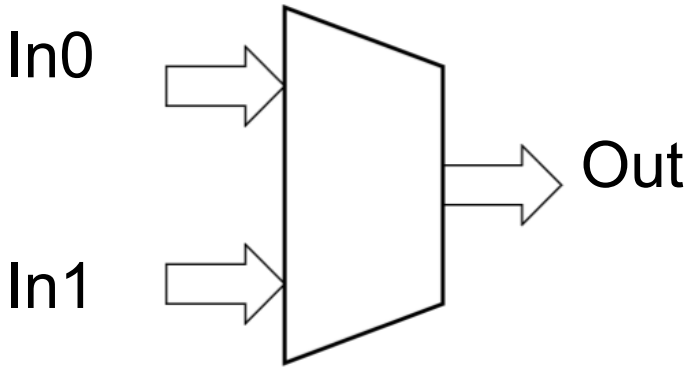
Note: does this perhaps describe a circuit with latch/buffer in Ctl and Out ports?

- When token on Ctl and selected input, propagate selected input to output
- A possible token on the other input is ignored
- Also known as: controlled merge, conditional join

A handshake-transparent implementation should:

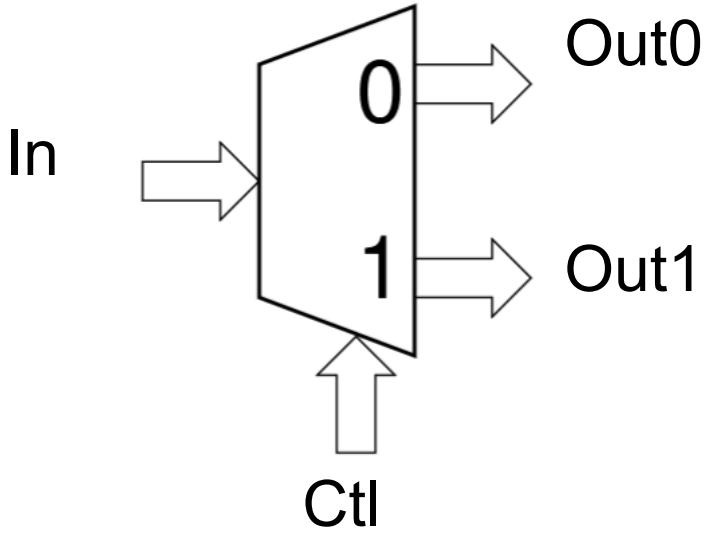
- Join Ctl-Req and the selected Inx-Req and produce Out-Req
- Relay/propagate Inx-data to Out-data
- Propagate Out-Ack to Ctl-Ack and Inx-Ack

MERGE



- Handshakes on In0 and In1 are mutually exclusive
- Active In-handshake relayed to output
- Also known as: MIXER, JOIN

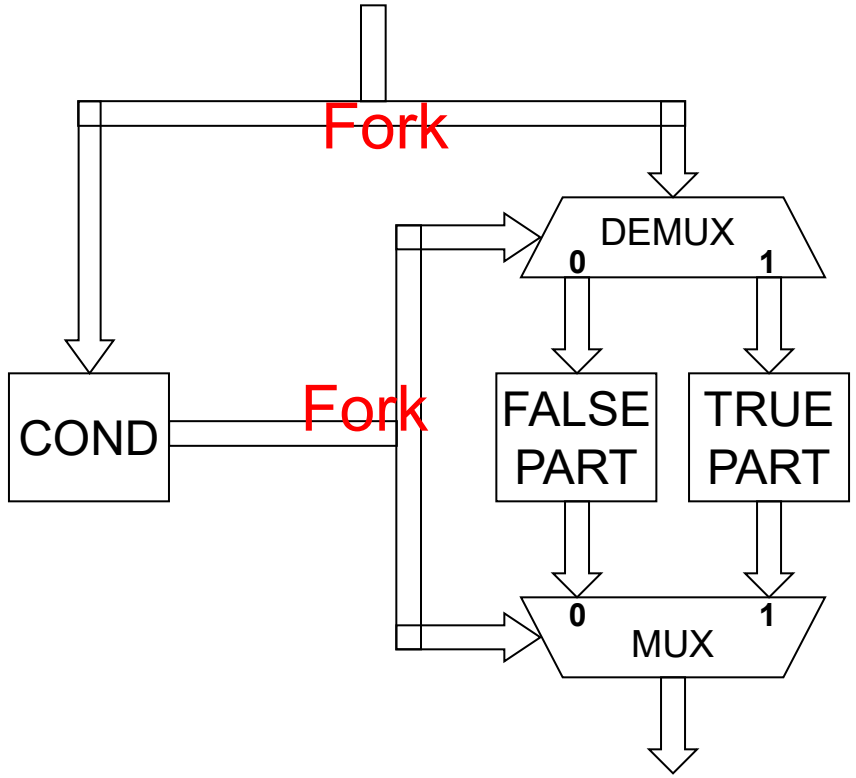
DEMUX



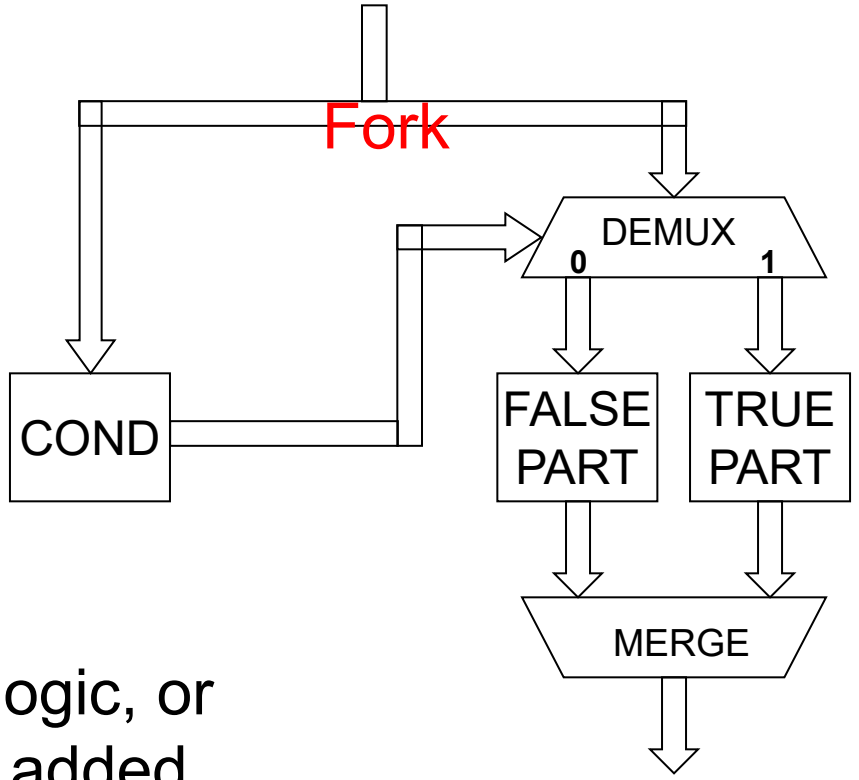
- When token on Ctl and In, propagate In-token to selected output channel
- Also known as: SPLIT

Structures of handshake components

IF statement



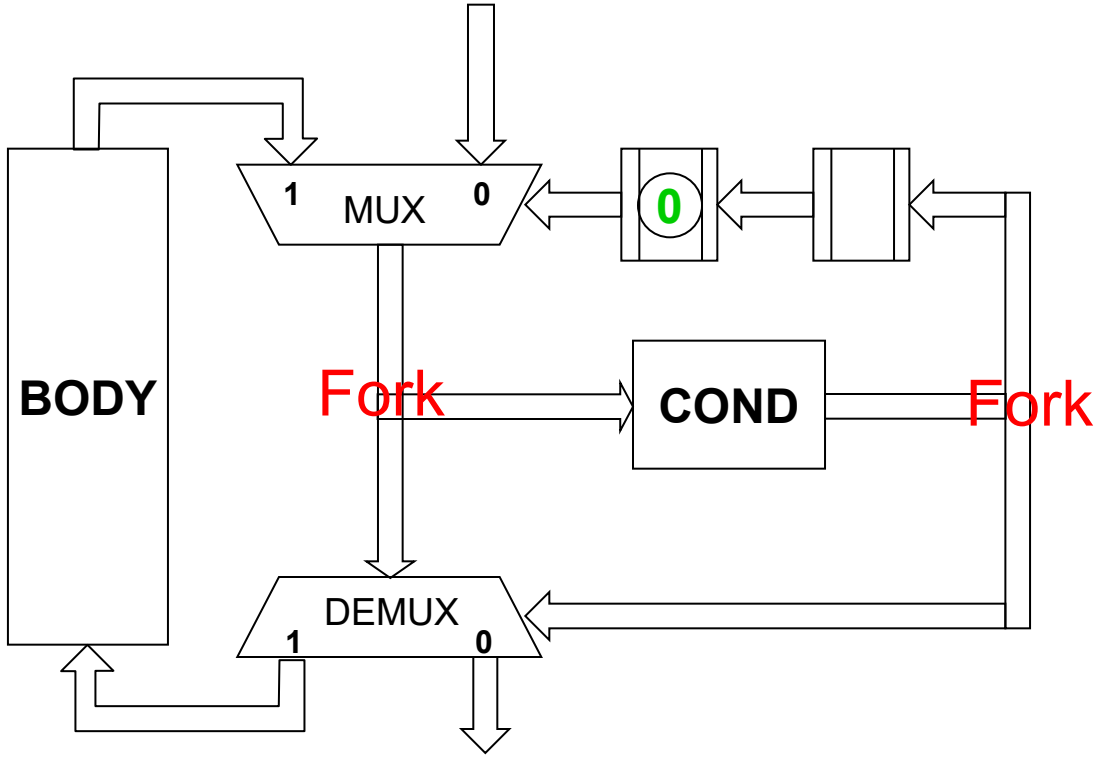
IF statement



Combinational logic, or
latches may be added

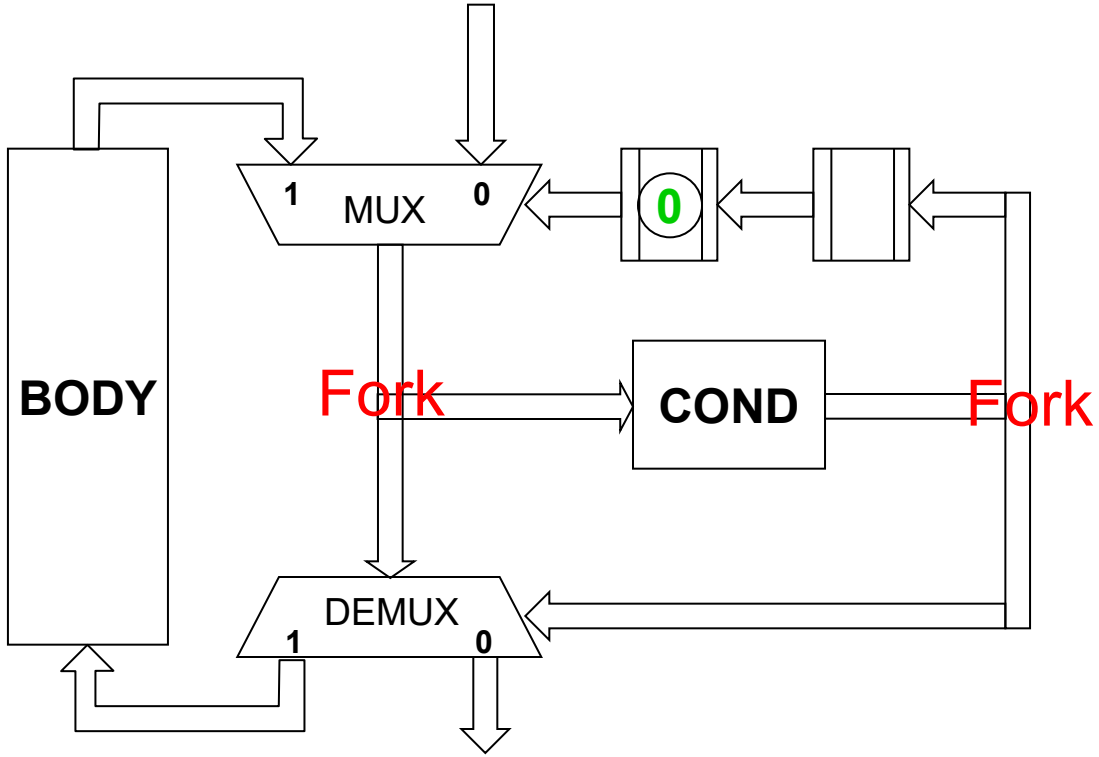
WHILE statement

`while` <COND> **`do`** <BODY>



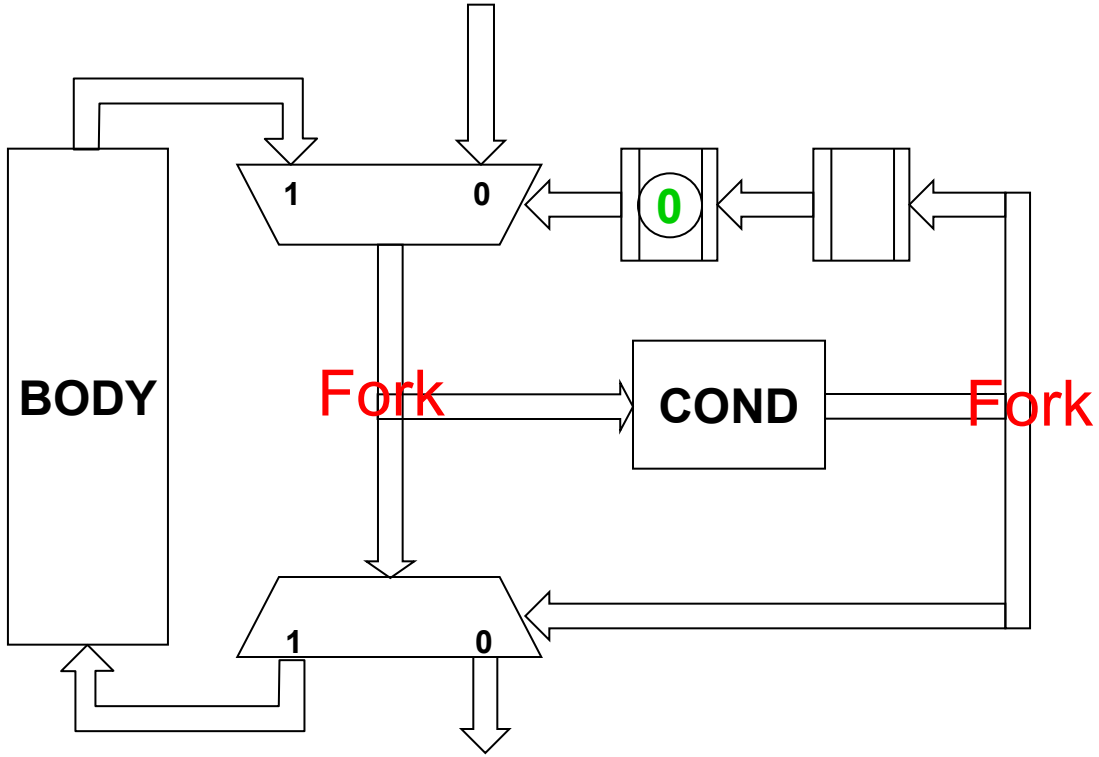
WHILE statement

`while` <COND> **`do`** <BODY>



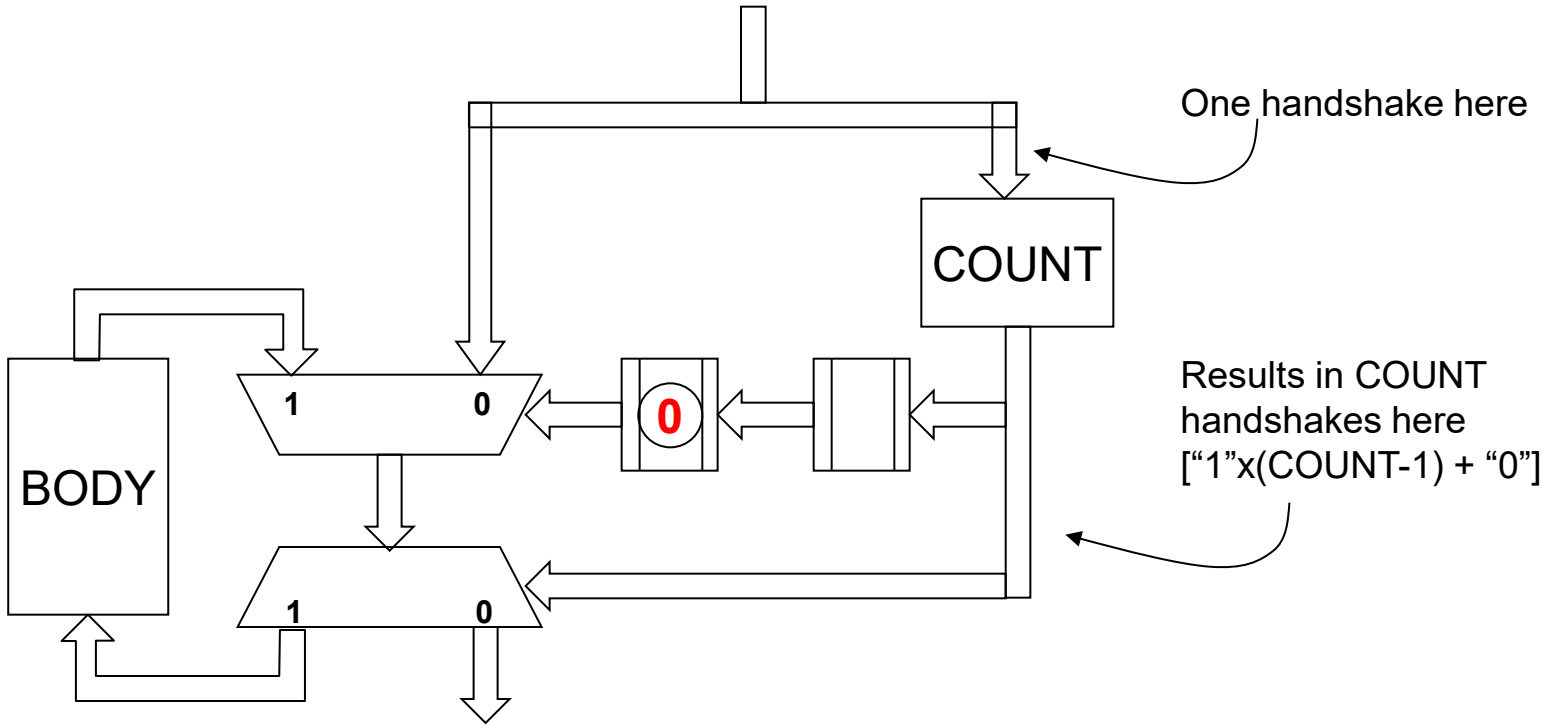
WHILE statement

`while` <COND> **`do`** <BODY>



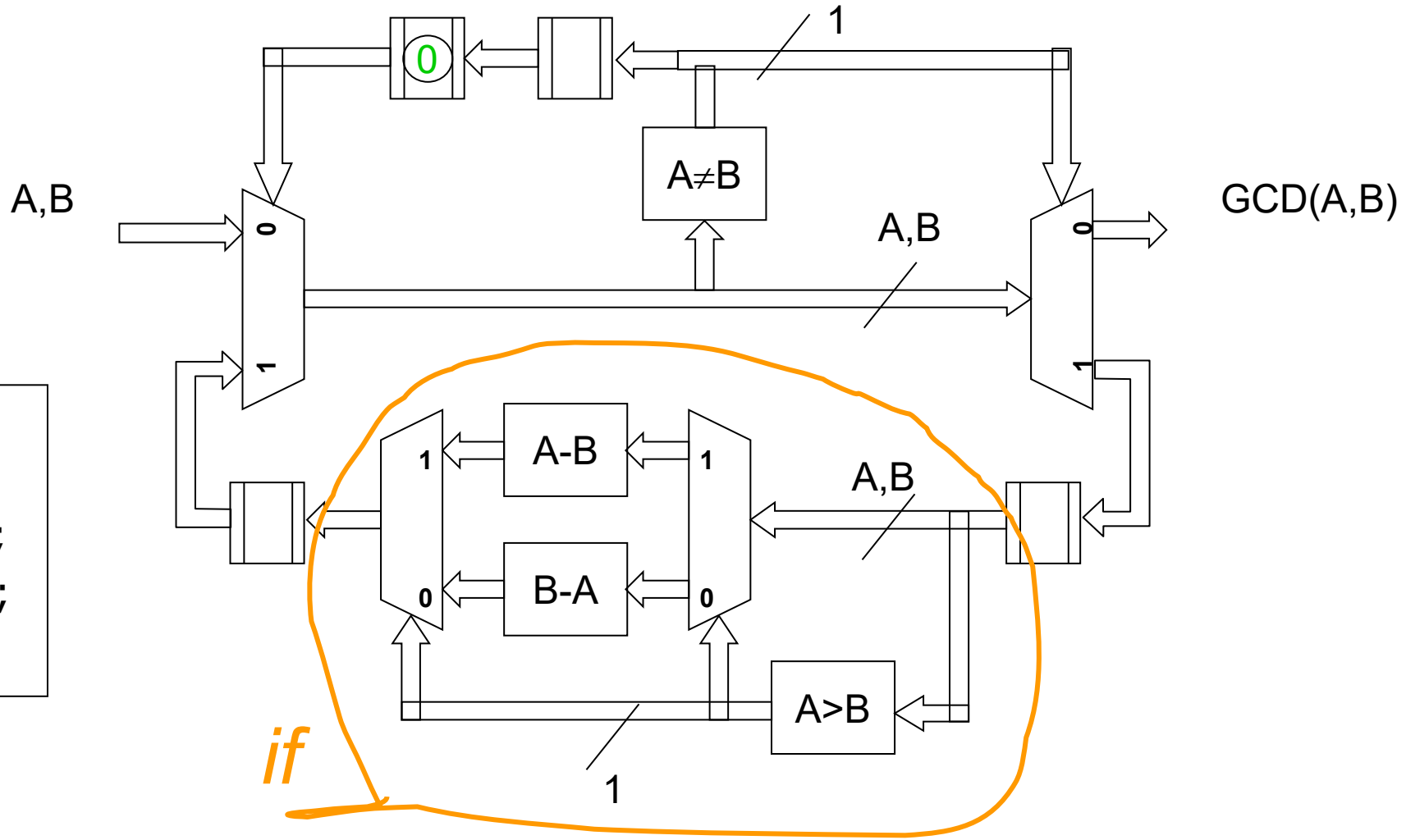
FOR statement

`for` <COUNT> **`do`** <BODY>



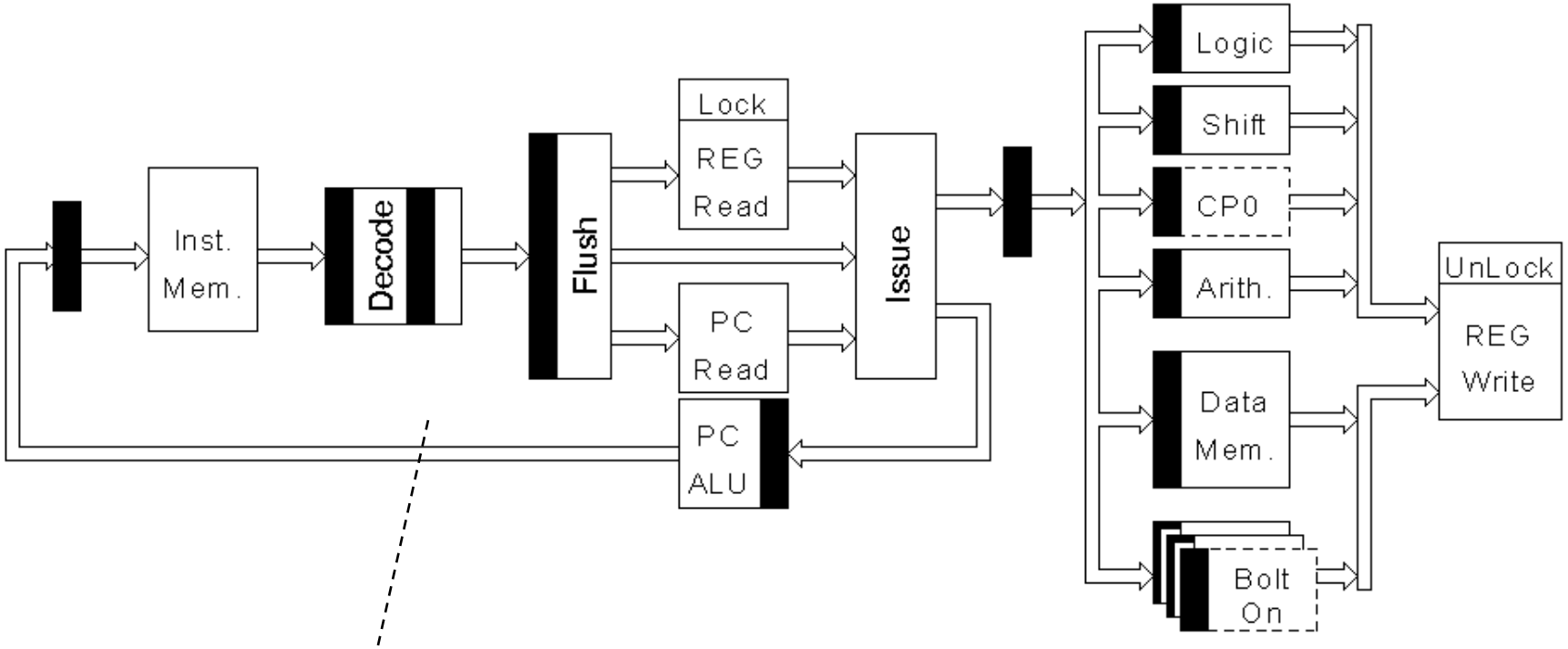
Asyncronous GCD

```
input (a, b);
while a ≠ b do
  if a > b then a ← a-b ;
           else  b ← b-a ;
output (a);
```



Asynchronous processor (DTU's ARISC)

LSI Logic TinyRISC (MIPS + MIPS16 isa):

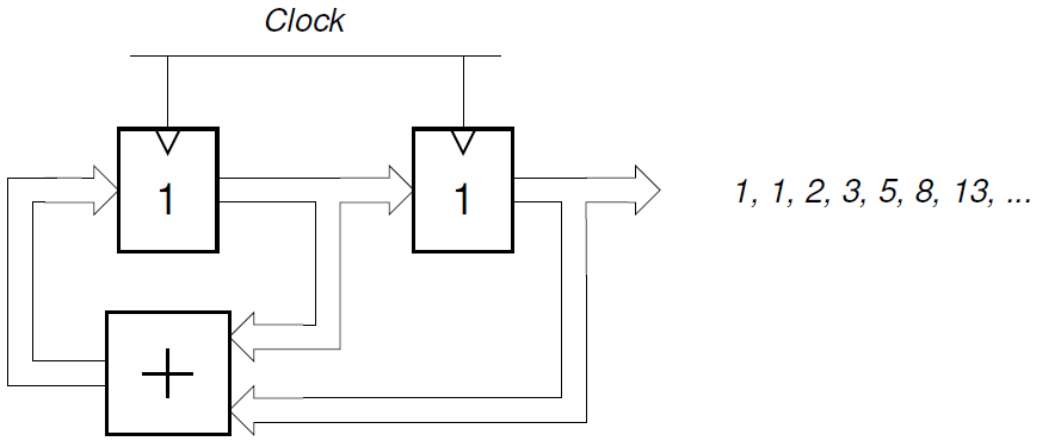


Fetch-decode-issue ring
with 2 tokens (=prefetch depth)

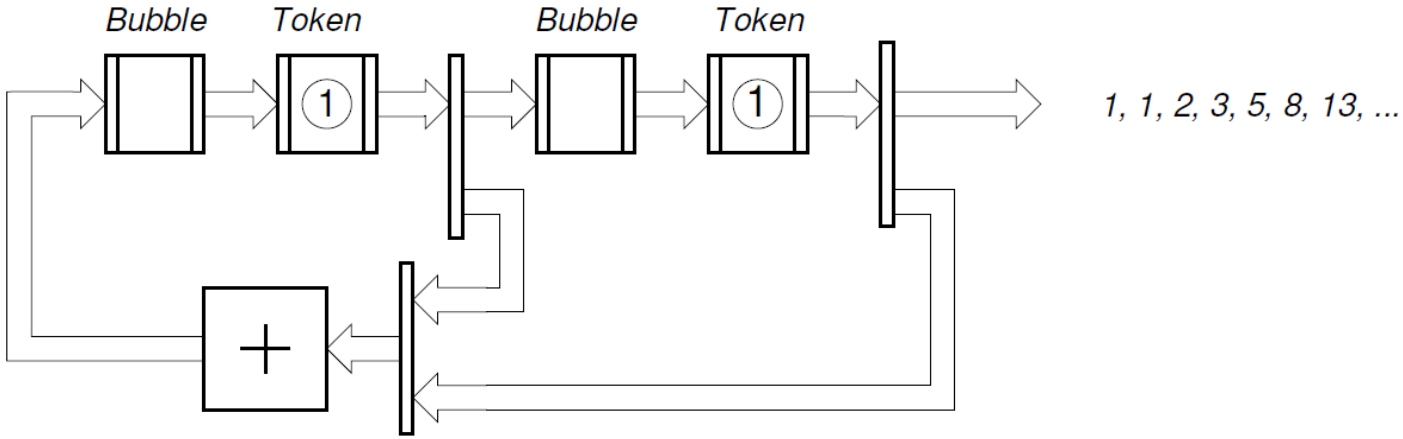
K. Christensen, P. Jensen, P. Korger and J. Sparsø,
"The design of an asynchronous TinyRISC microprocessor core,"
Proc. ASYNC'98, IEEE, 1998, pp. 108-119.

Fibonacci circuit

Synchronous

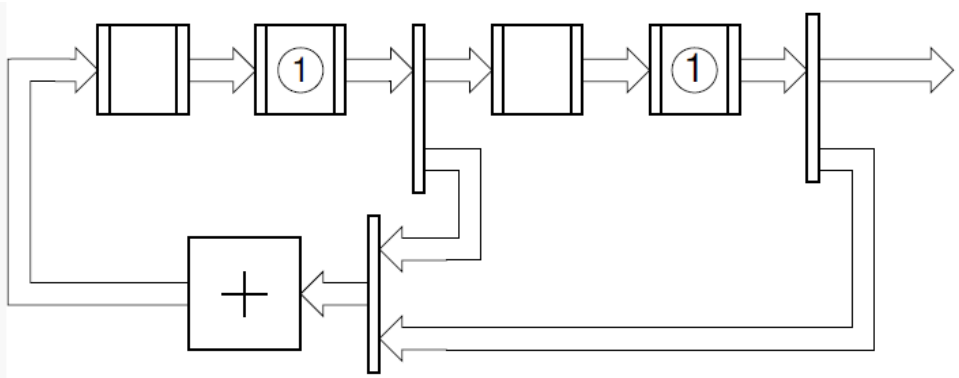


Asynchronous

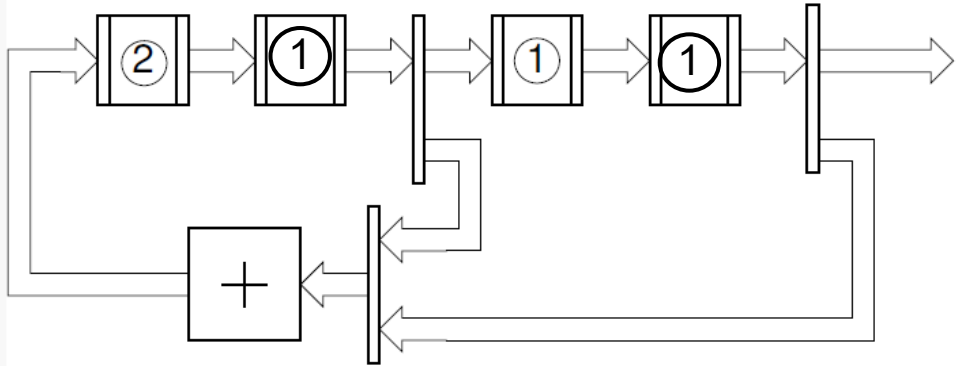


Asynchronous Fibonacci circuit

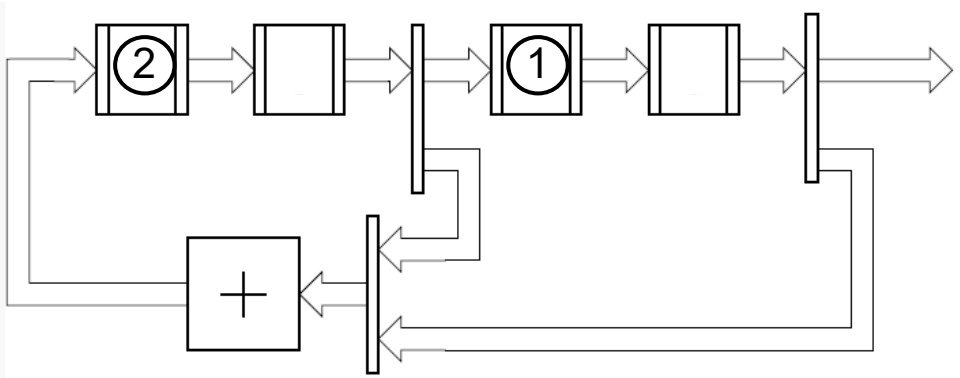
1)



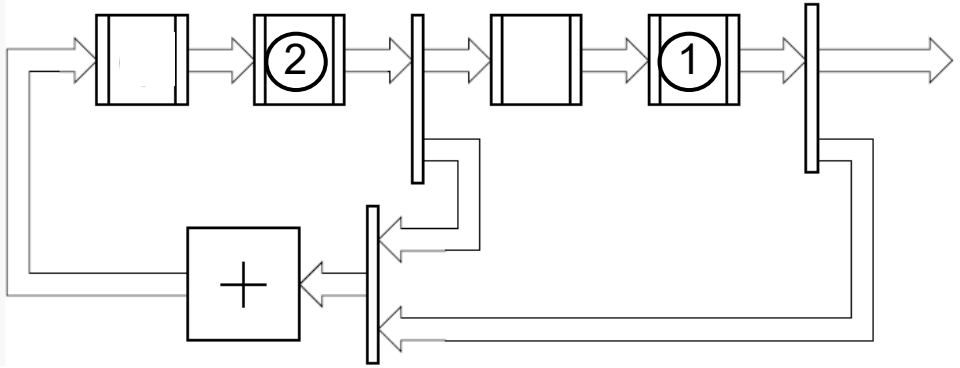
2)



3)



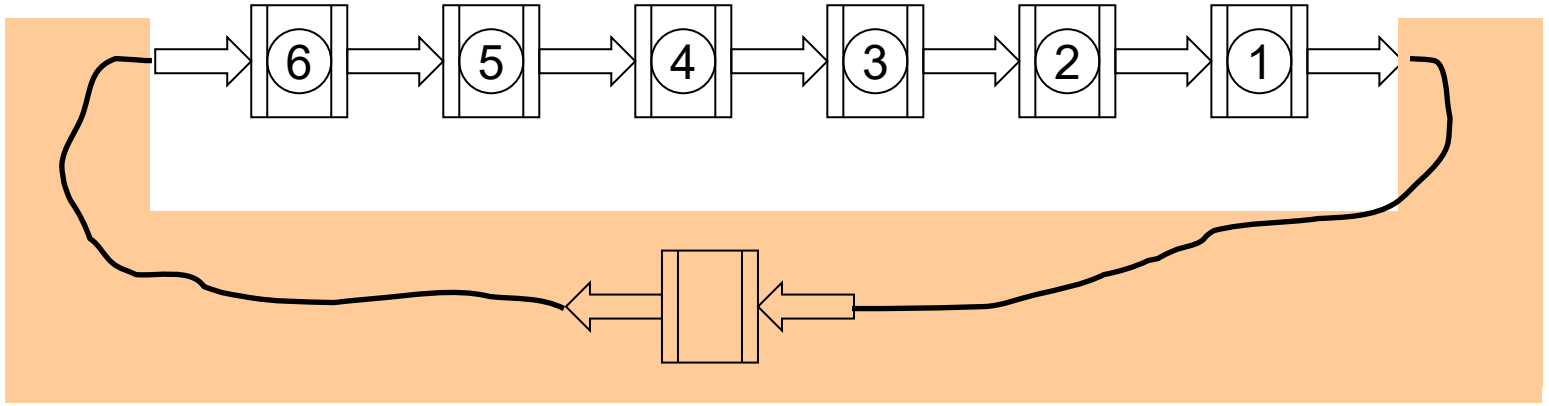
4)



Performance; mostly qualitative

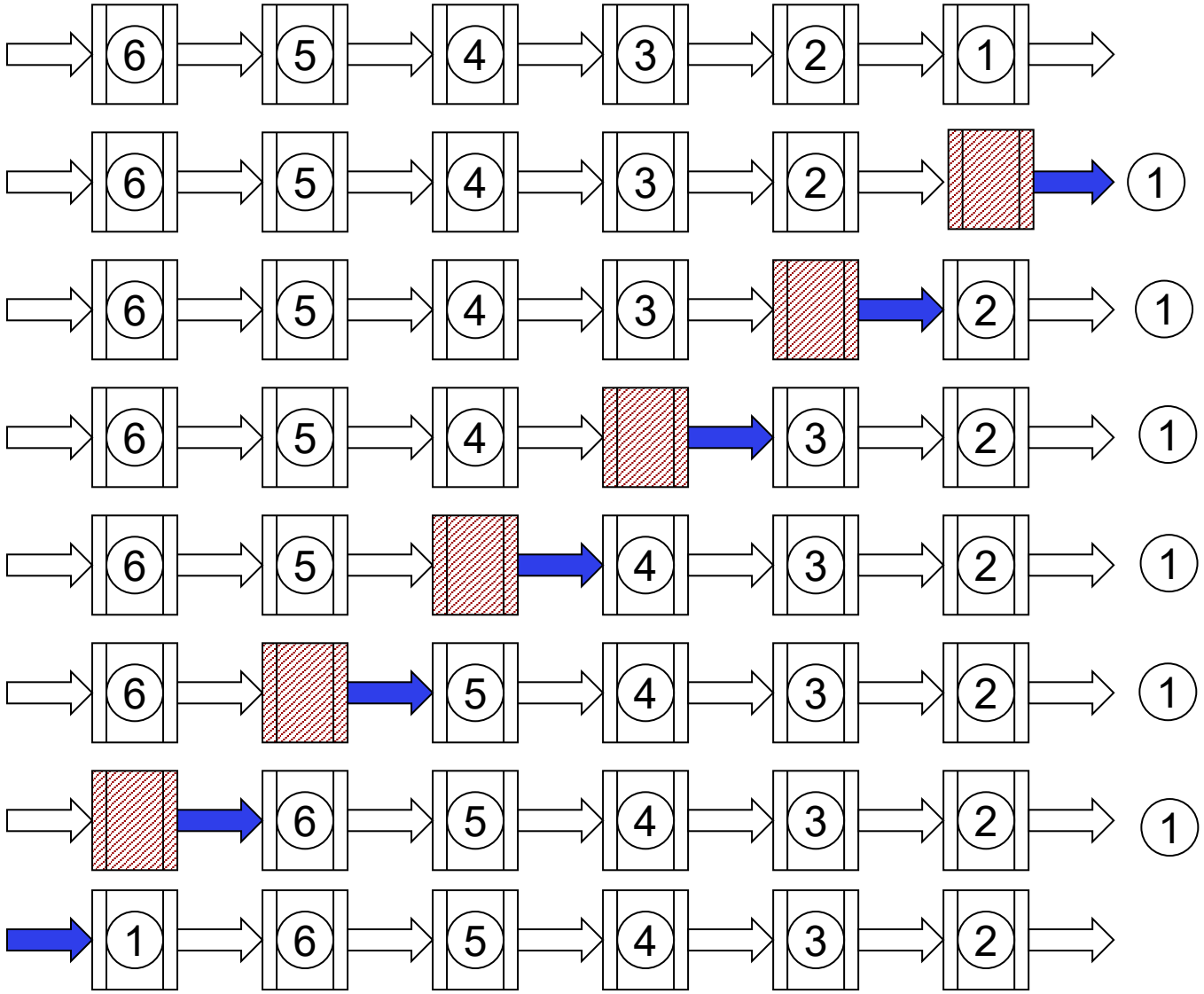
- Dynamic behavior
  - Example 1: A shift register ≈ a ring
  - Example 2: A FIFO
- Performance parameters
  - dynamic wavelength (dynamic wavelength forward latency, reveverse latency, and cycle time).

Example: A shift register



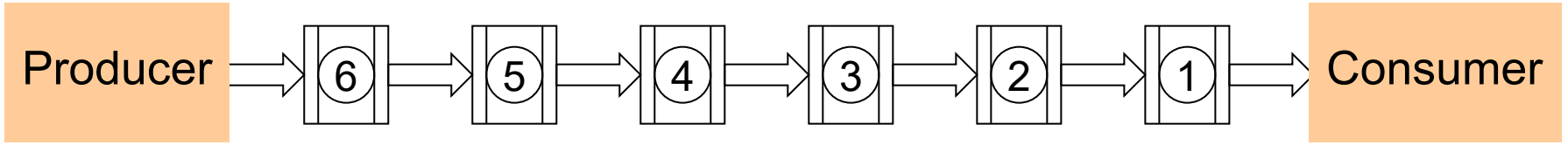
- An N stage FIFO or pipeline whose environment alternately accepts and produces tokens
- Assume the FIFO is full
- Same as N+1 stage ring with N tokens and 1 bubble
- After initialization, the number of tokens is *invariant*

Shift register (ring) performance



Only one bubble
Ripples backward

Example: Pipeline / FIFO



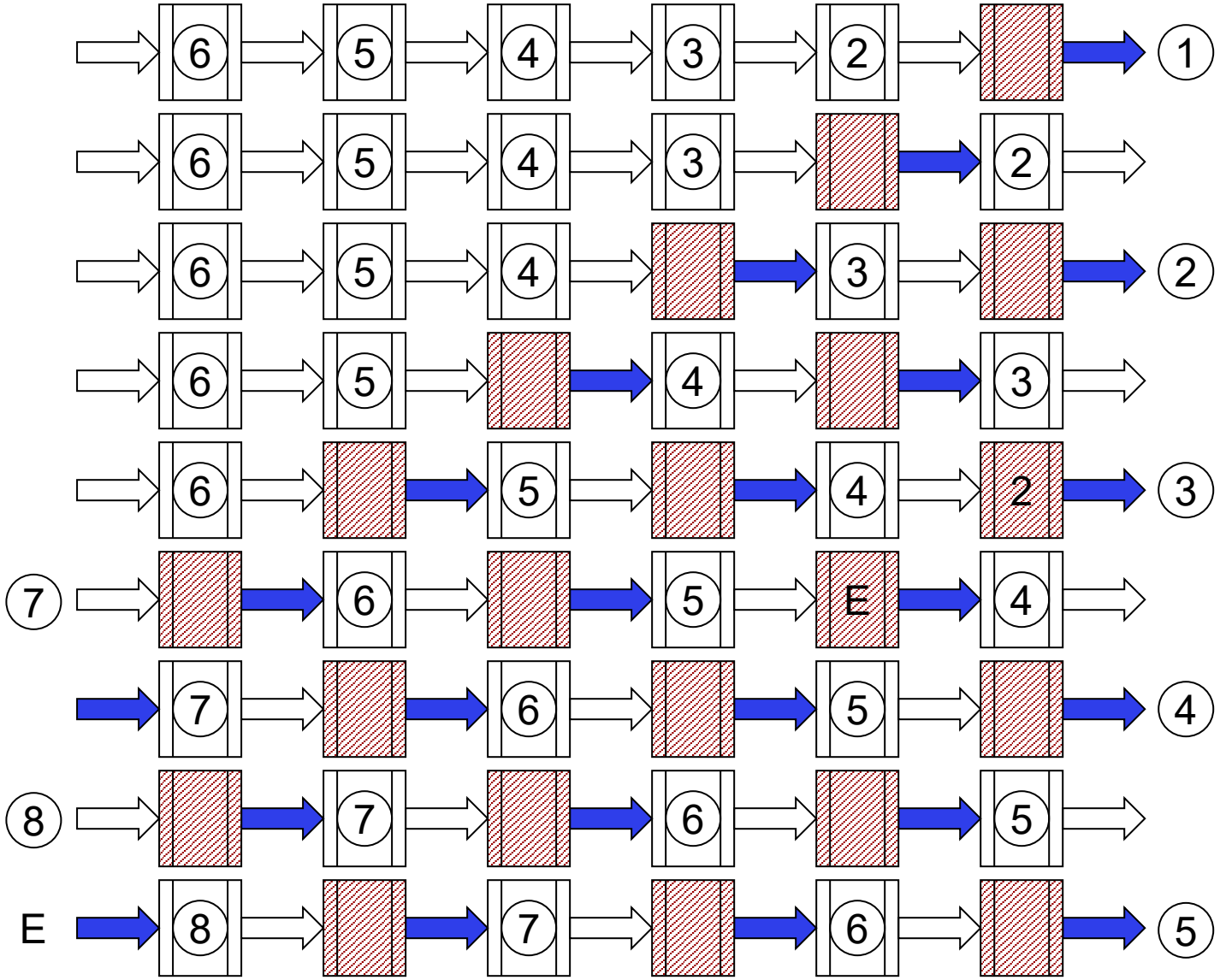
- Same circuit and same initialization (N stages, N tokens, no bubbles)
- Fast and independent producer and consumer.
- Start the circuit and observe its behavior

FIFO / Pipeline performance

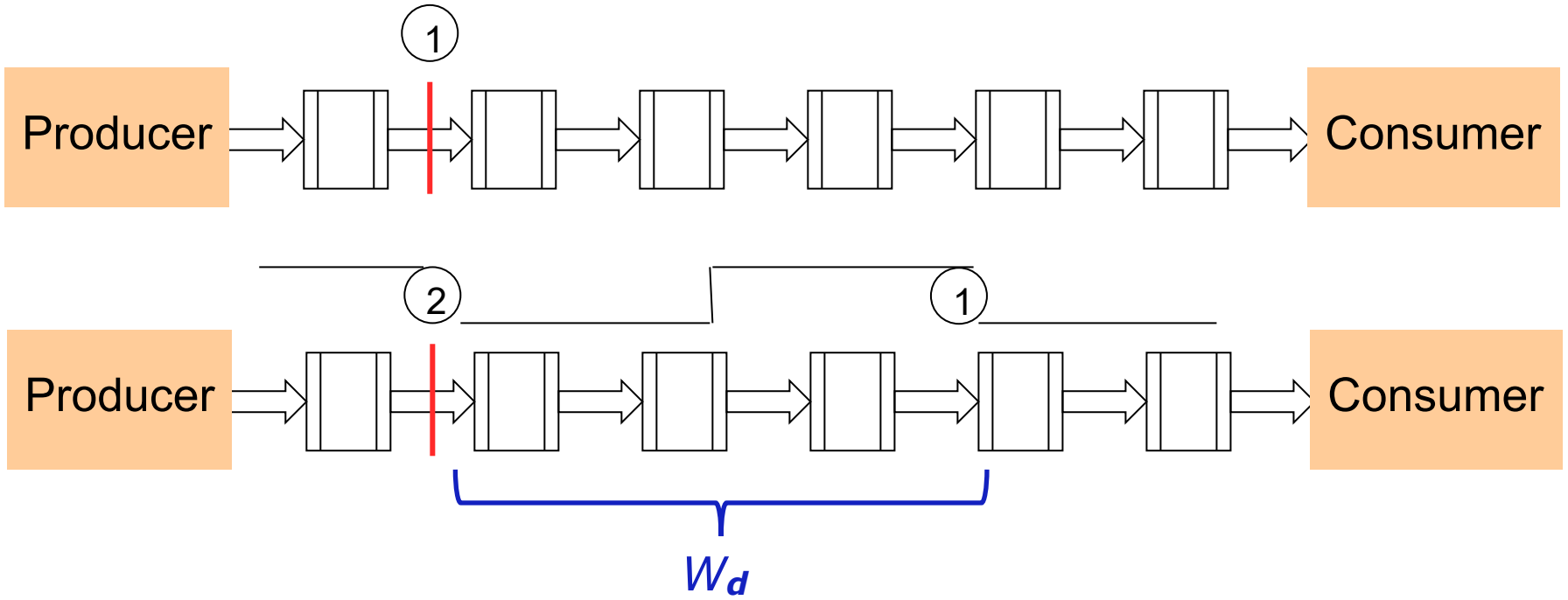
Reaches a steady-state where the time separation between successive tokens passing through a channel is minimum

The distance between successive tokens (measured as a number of latch stages) is the **dynamic Wavelength**

Here $W_d = 2$, but in an actual circuit Implementation it can be any rational number.

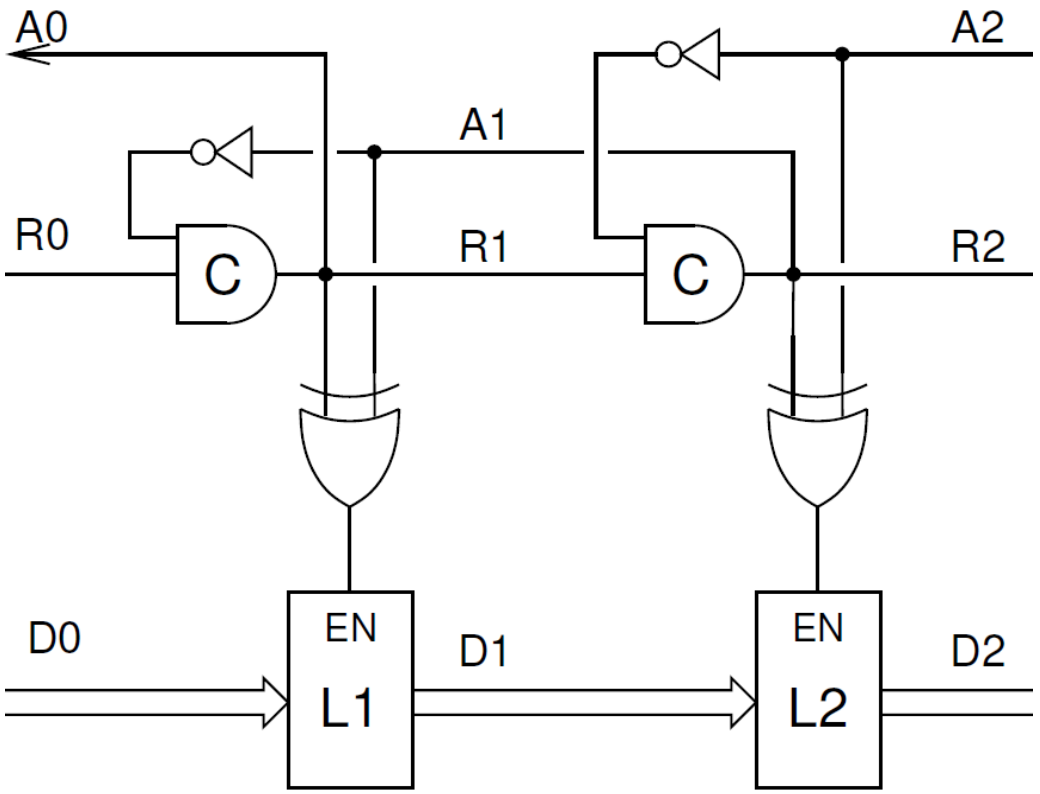


Dynamic wavelength



- T_{cycle} is time from one token to the next
- During T_{cycle} the first token propagates through W_d stages (Think of a token as the front of a wave)

Dynamic wavelength



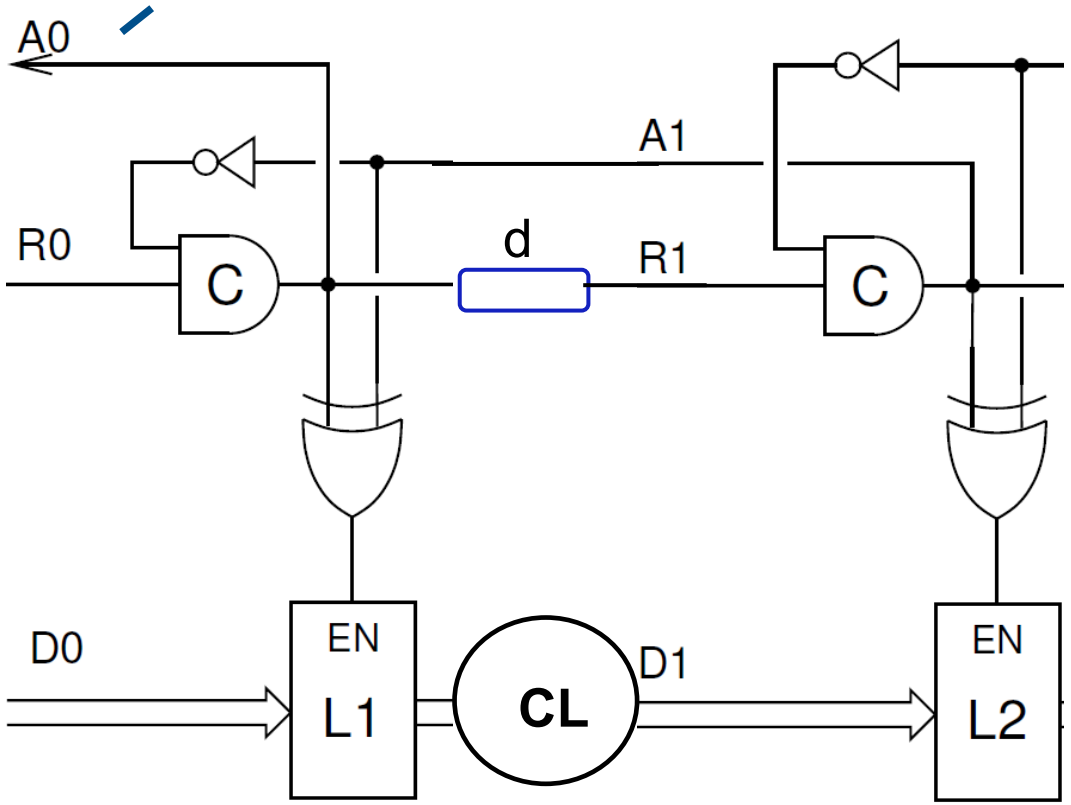
Time for a handshake on a given channel

$$T_{cycle} \geq L_f + L_r$$

During time T_{cycle}, a token propagates forward through W_d stages

Best performance if W_d stages per token

Dynamic wavelength



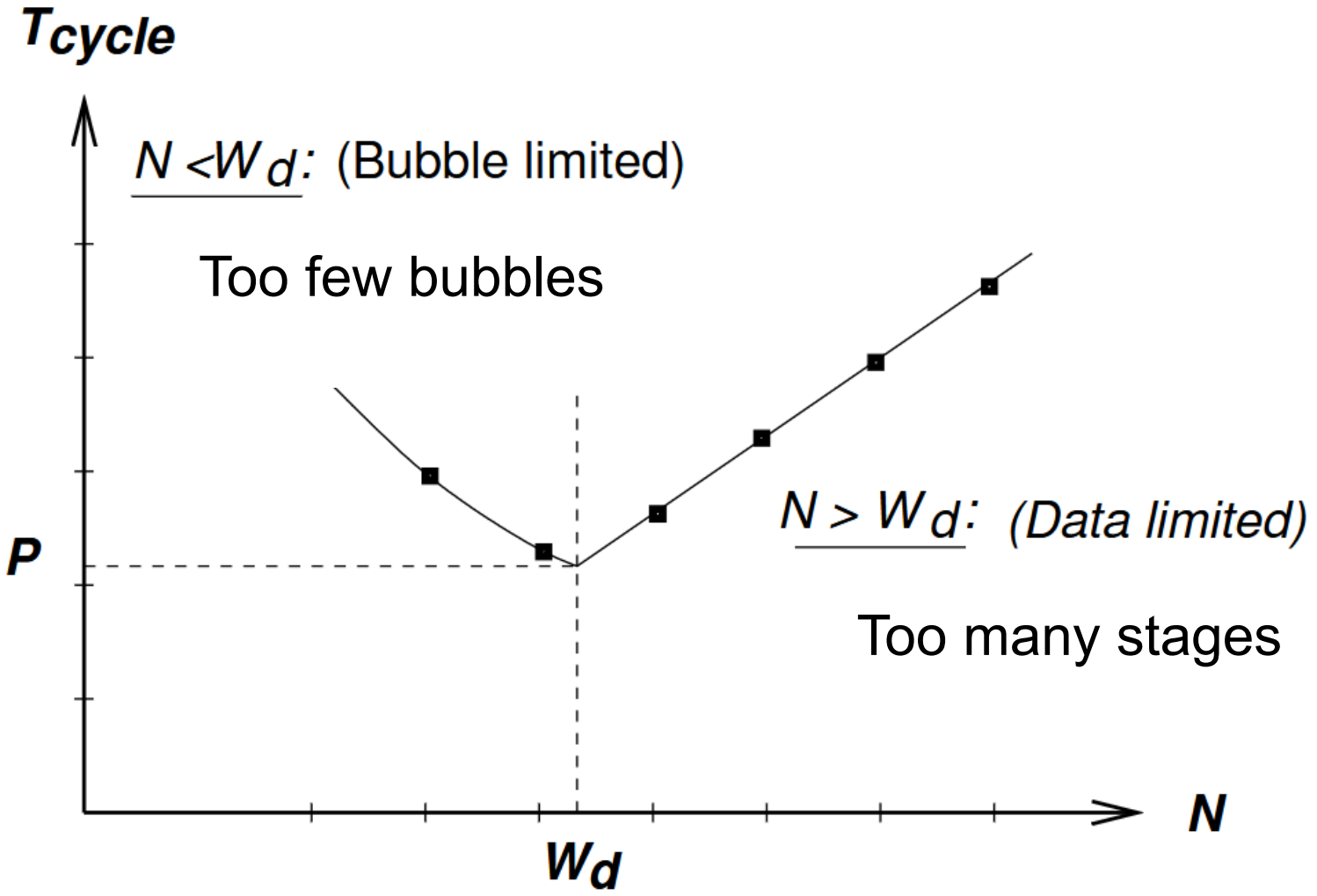
Time for a handshake on a given channel

$$T_{cycle} \geq L_f + L_r$$

During time T_{cycle}, a token propagates forward through W_d stages

Best performance if W_d stages per token

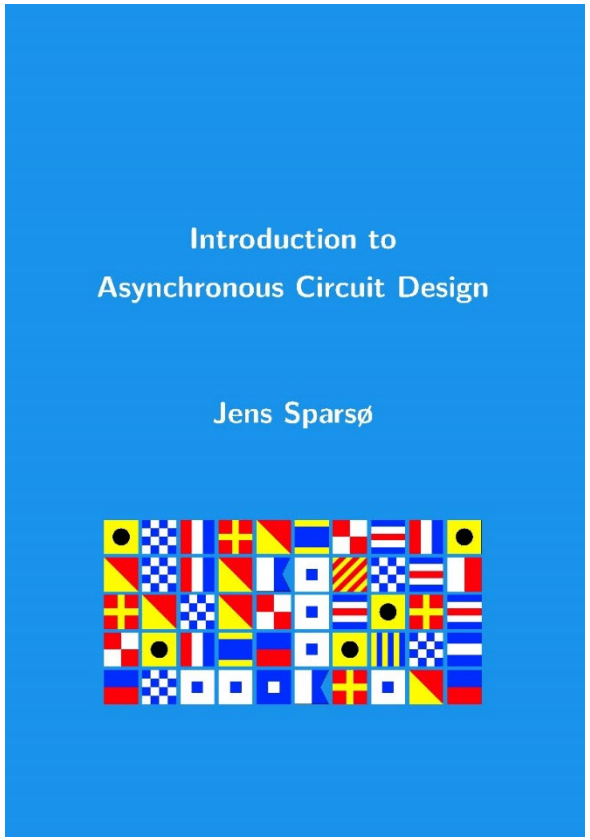
Cycle time for N-stage ring with 1 token (time for the token to make one roundtrip)



Conclusion

- Dataflow is an intuitive paradigm for describing asynchronous circuits.
- Handshake components:
  - Actively handshaking: Source, sink, (handshake) latch/buffer
  - Transparent to handshaking: fork, join, mux, demux. Merge, arbiter
  - (Data) tokens and bubbles
- Flow of tokens determines the function of the circuit,
- Adding or removing handshake latches initialized to bubbles affects performance, but does not change functionality
- Some example circuits (if-then-else, while, for, FIB, GCD, processor)
- A qualitative view on the performance of pipelines and rings
  - Balancing the number of tokens and bubbles (aka. Slack matching)
  - Dynamic wavelength.

Reading



- Jens Sparsø,
  Introduction to Asynchronous Circuit Design
  DTU Compute, Technical University of Denmark
  2020. (273 pages)
- Download free version by following the link given on my homepage https://people.compute.dtu.dk/jspa/
- Or buy the paperback book from Amazon for just 11 USD