Compositional Dataflow Circuits

Stephen A. Edwards   Richard Townsend   Martha A. Kim

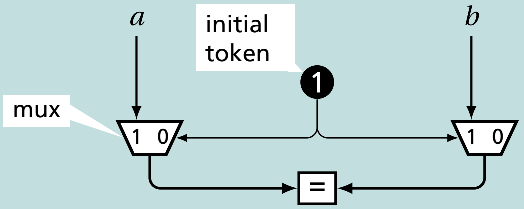
Columbia University

MEMOCODE, Vienna, Austria, October 1, 2017

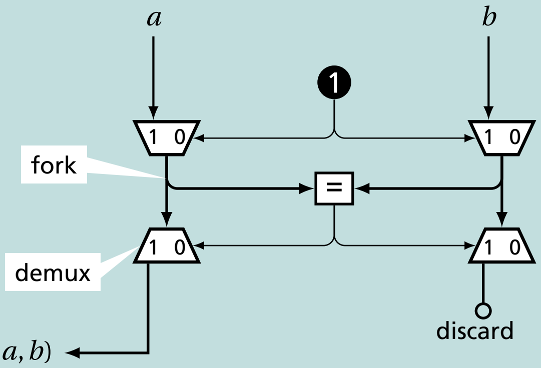
```
$\mathrm{gcd}(a, b) =$
    if $a = b$
        $a$
    else if $a < b$
        $\mathrm{gcd}(a, b - a)$
    else
        $\mathrm{gcd}(a - b, b)$
```

```
gcd($a, b$) =
    if $a = b$
        $a$
    else if $a < b$
        gcd($a, b - a$)
    else
        gcd($a - b, b$)
```

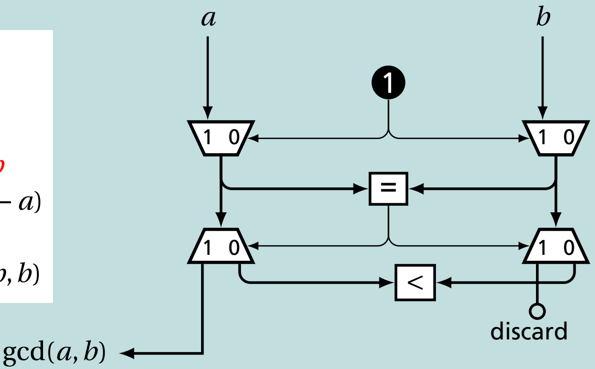
```
$\mathrm{gcd}(a, b) =$
    if $a = b$
        $a$
    else if $a < b$
        $\mathrm{gcd}(a, b - a)$
    else
        $\mathrm{gcd}(a - b, b)$
```



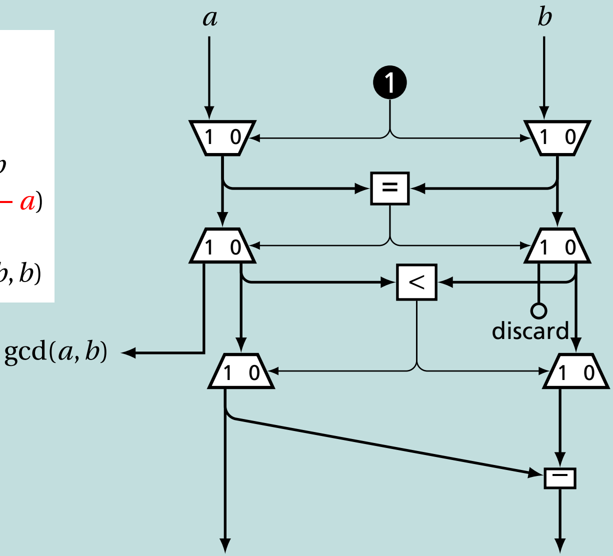
```
gcd($a, b$) =
    if $a = b$
        $a$
    else if $a < b$
        gcd($a, b - a$)
    else
        gcd($a - b, b$)
```



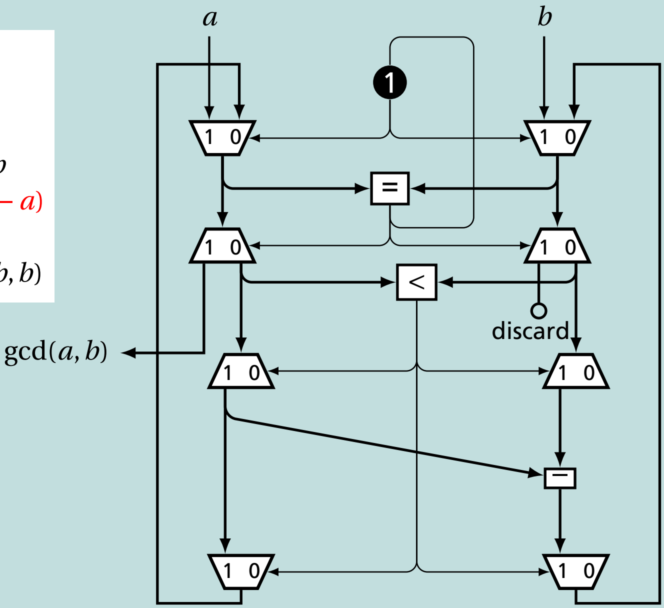
```
$\mathrm{gcd}(a, b) =$
    if $a = b$
        $a$
    else if $a < b$
        $\mathrm{gcd}(a, b - a)$
    else
        $\mathrm{gcd}(a - b, b)$
```



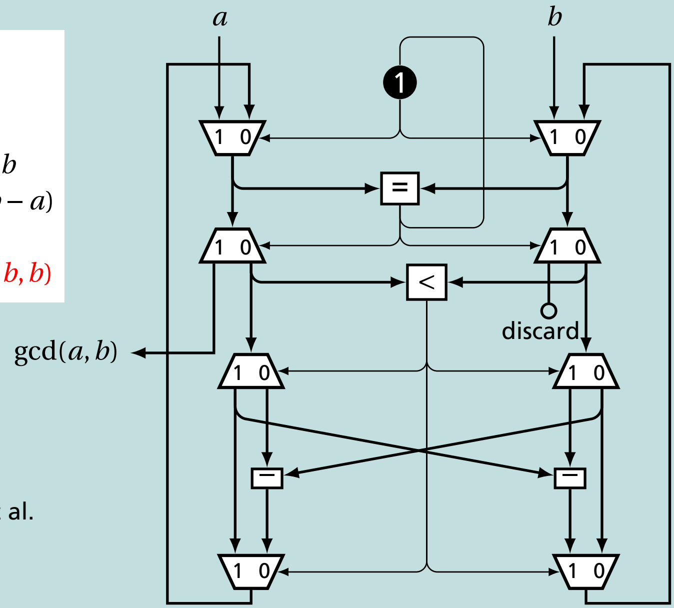
```
gcd($a, b$) =
    if $a = b$
        $a$
    else if $a < b$
        gcd($a, b - a$)
    else
        gcd($a - b, b$)
```



```
gcd($a, b$) =
    if $a = b$
        $a$
    else if $a < b$
        gcd($a, b - a$)
    else
        gcd($a - b, b$)
```



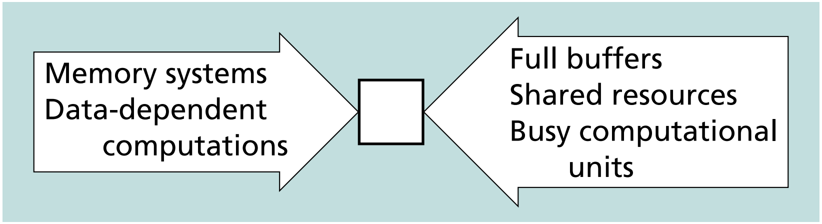
```
gcd($a, b$) =
    if $a = b$
        $a$
    else if $a < b$
        gcd($a, b - a$)
    else
        gcd($a - b, b$)
```



Townsend et al.
CC ’2017

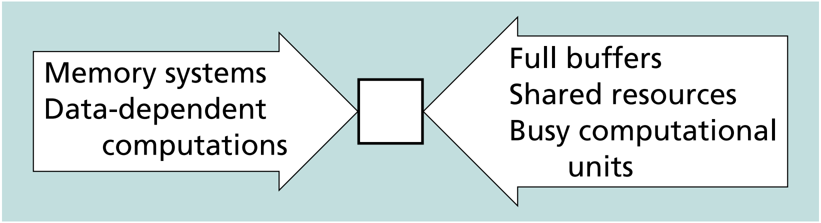
Patience Through Handshaking

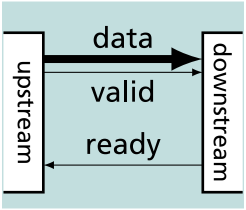
Want *patient* blocks to handle delays from



Patience Through Handshaking

Want *patient* blocks to handle delays from





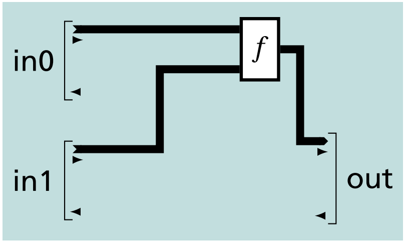
valid	ready	Meaning
1	1	Token transferred
1	0	Token valid; held
0	–	No token to transfer

Latency-insensitive Design (Carloni et al.)
Elastic Circuits (Cortadella et al.)
FIFOs with backpressure

Combinational Function Block

Strict/Unit Rate:

All input tokens required to produce an output



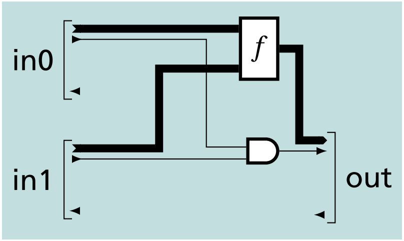
Datapath

Combinational function ignores flow control

Combinational Function Block

Strict/Unit Rate:

All input tokens required to produce an output



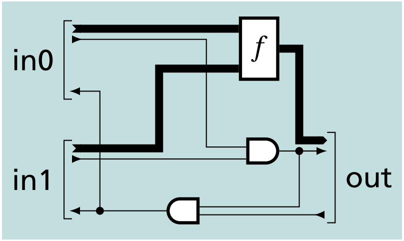
Valid network

Output valid if both inputs are valid

Combinational Function Block

Strict/Unit Rate:

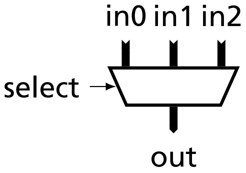
All input tokens required to produce an output

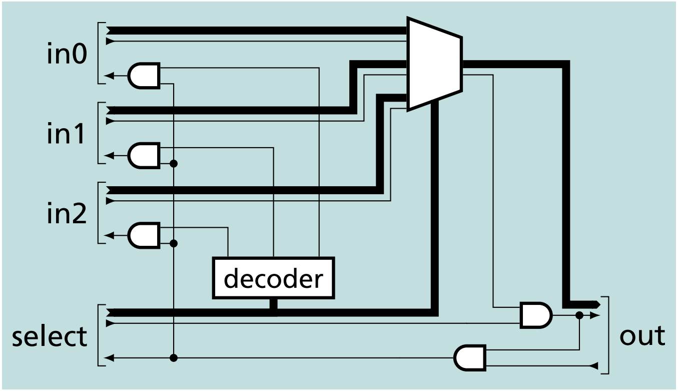


Ready network

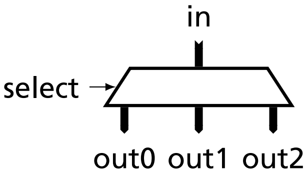
Input tokens consumed if output token is consumed
(output is valid and ready)

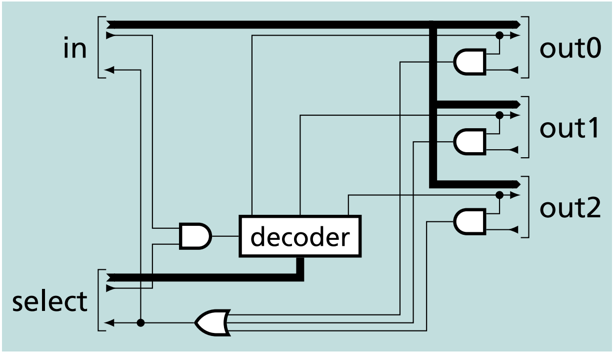
Multiplexer Block



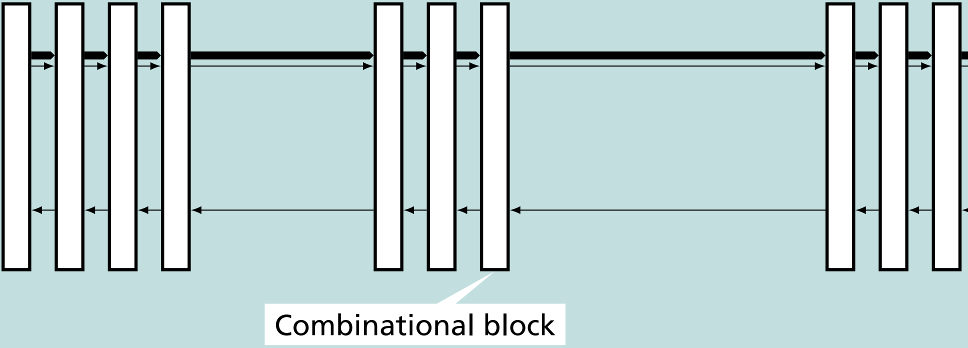


Demultiplexer Block

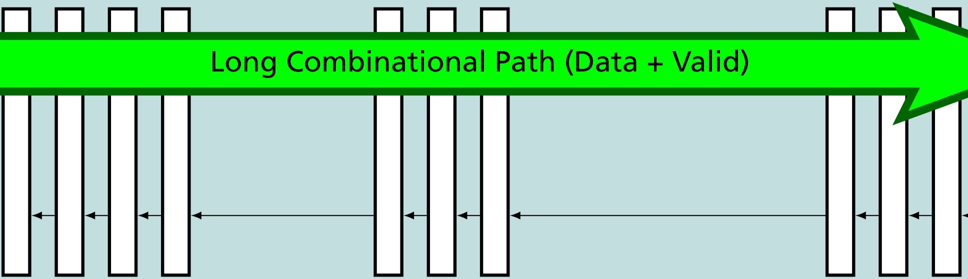




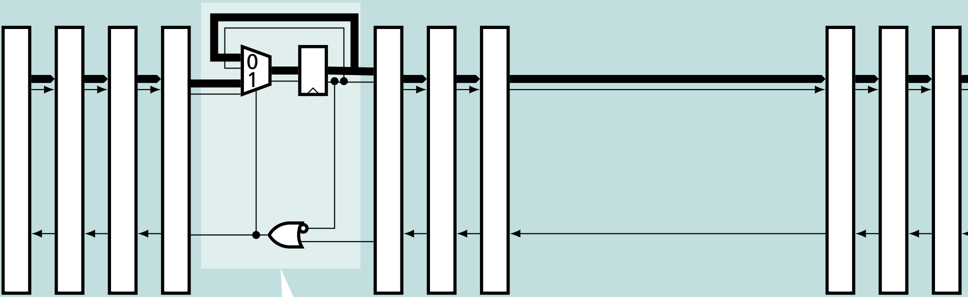
Buffering a Linear Pipeline (Point 1/4)



Buffering a Linear Pipeline (Point 1/4)

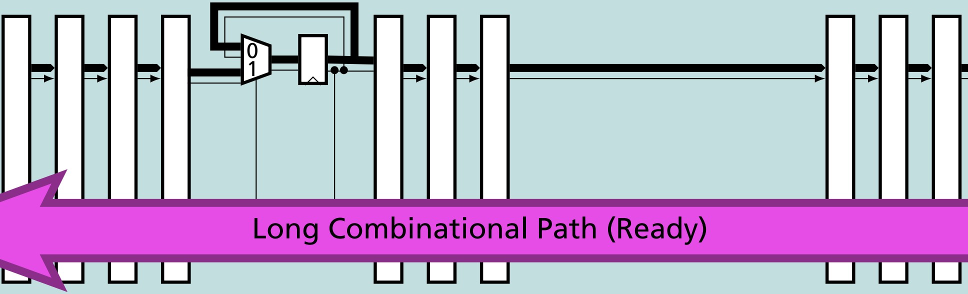


Buffering a Linear Pipeline (Point 1/4)

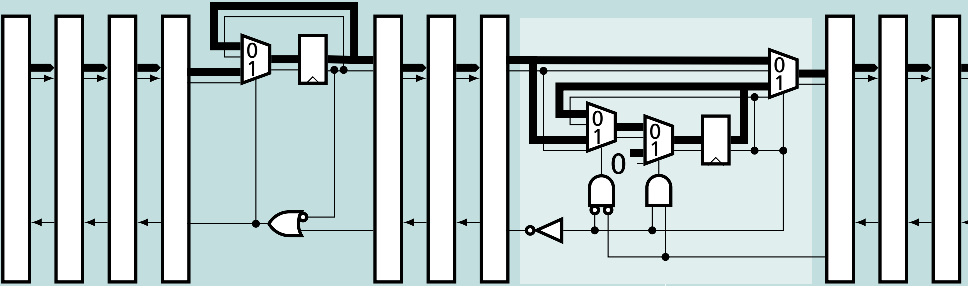


Data buffer:
Pipeline register
with valid, enable

Buffering a Linear Pipeline (Point 1/4)



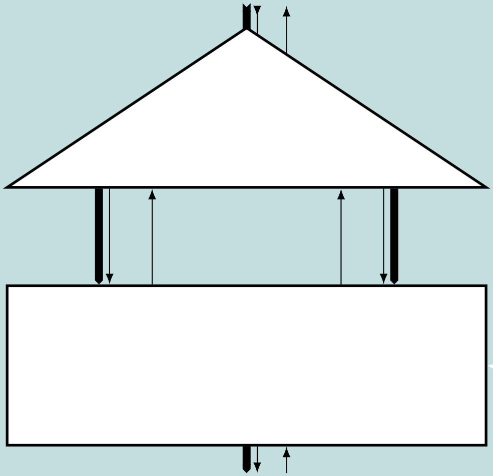
Buffering a Linear Pipeline (Point 1/4)



Control Buffer:
Register diverts token when downstream suddenly stops

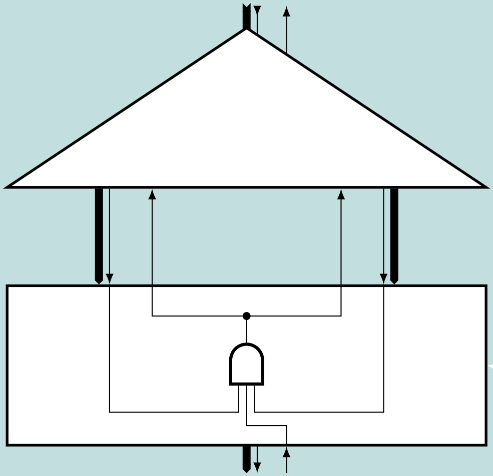
Cao et al. MEMOCODE 2015
Inspired by Carloni’s Latency Insensitive Design (e.g., MEMOCODE 2007)

The Problem with Fork



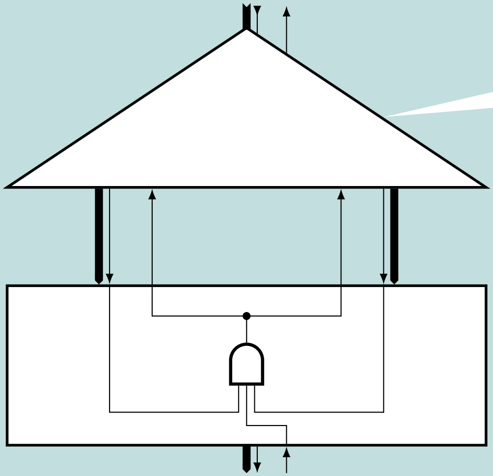
Combinational Block: inputs ready when both valid & output ready

The Problem with Fork



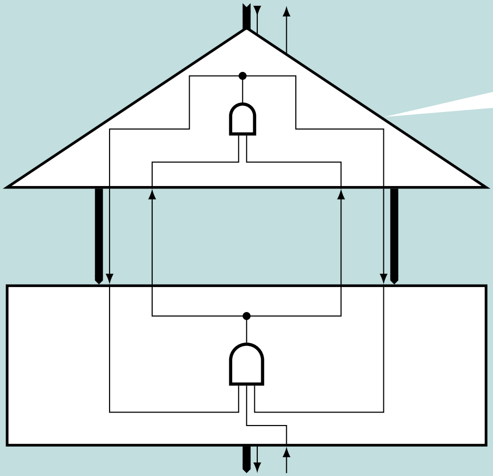
Combinational Block: inputs ready when both valid & output ready

The Problem with Fork



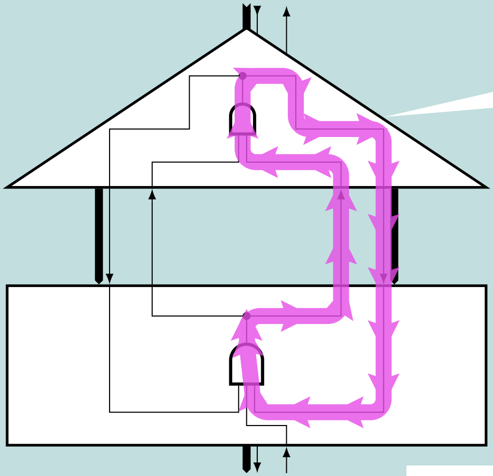
Fork:
outputs valid only
when all are ready

The Problem with Fork



Fork:
outputs valid only
when all are ready

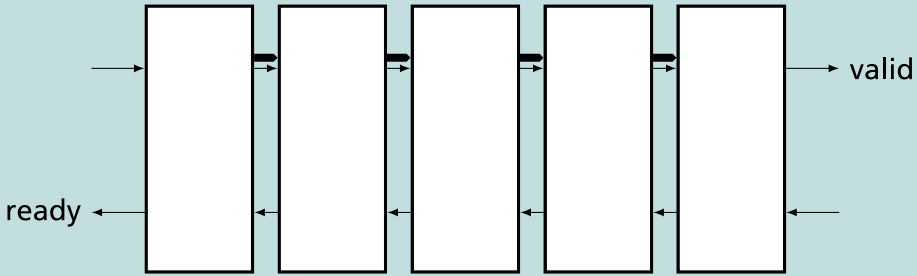
The Problem with Fork



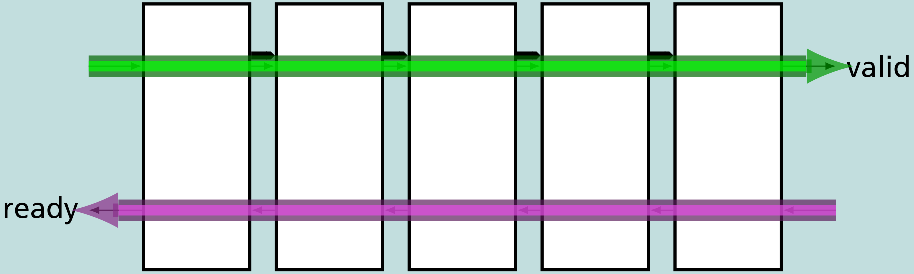
Fork:
outputs valid only
when all are ready

Oops: Combinational Cycle
This is *not* compositional

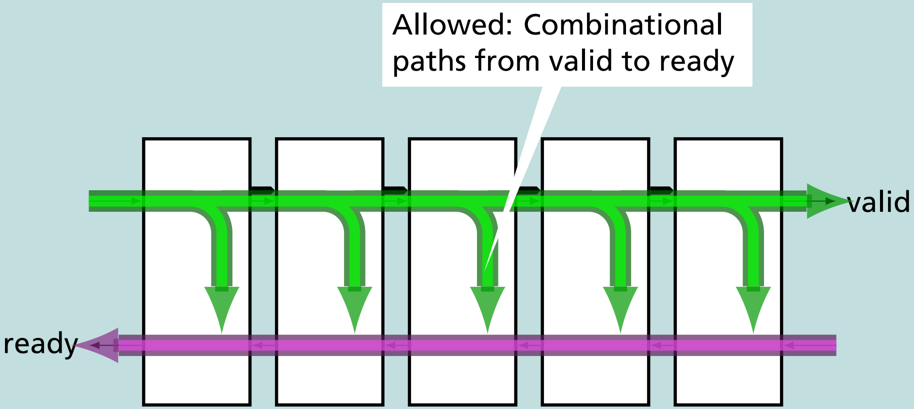
The Solution to Combinational Loops (Point 2/4)



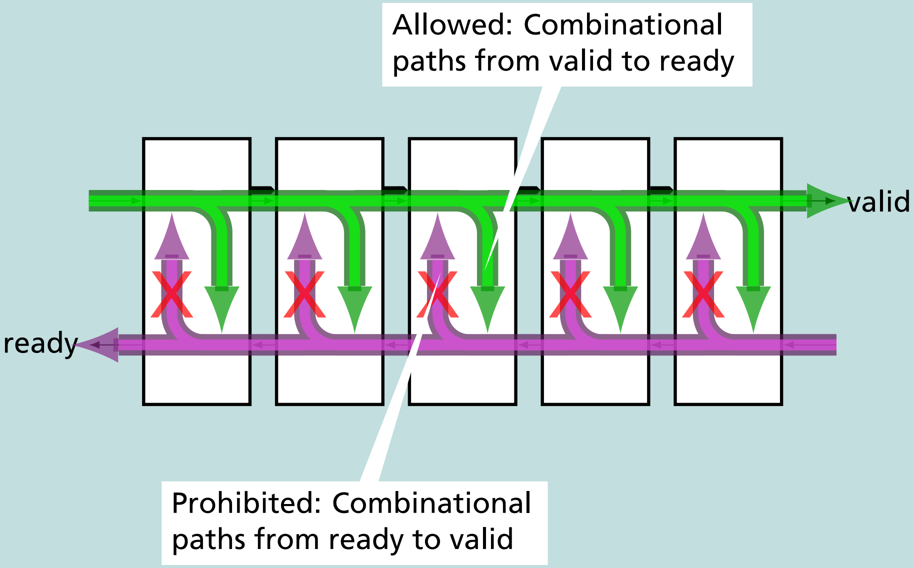
The Solution to Combinational Loops (Point 2/4)



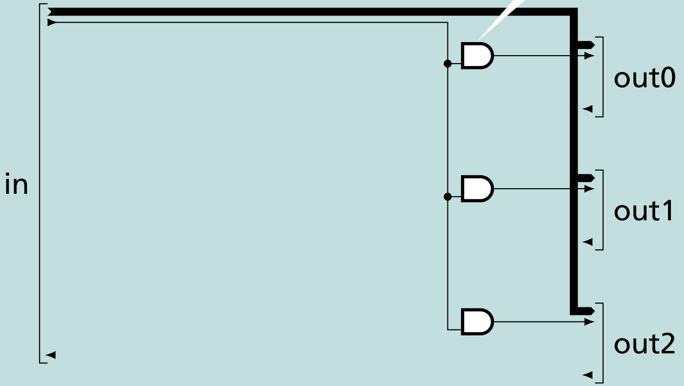
The Solution to Combinational Loops (Point 2/4)



The Solution to Combinational Loops (Point 2/4)


Allowed: Combinational paths from valid to ready
valid
ready
Prohibited: Combinational paths from ready to valid

The Solution to Fork: A Little State (Point 3/4)

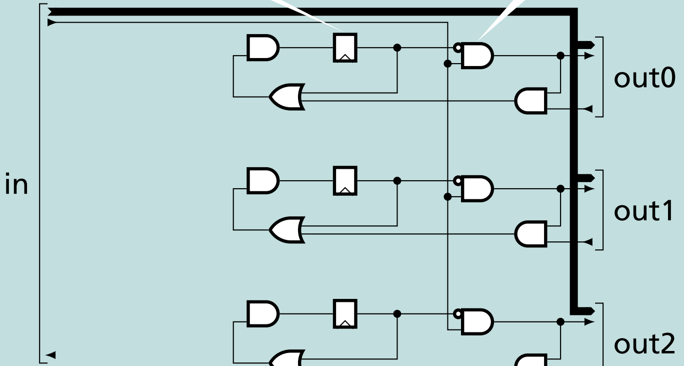


Valid out ignores ready of other outputs

The Solution to Fork: A Little State (Point 3/4)

Flip-flop set after token sent suppresses duplicates

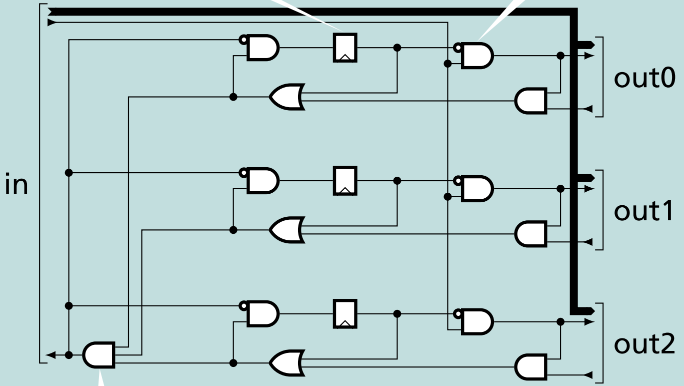
Valid out ignores ready of other outputs



The Solution to Fork: A Little State (Point 3/4)

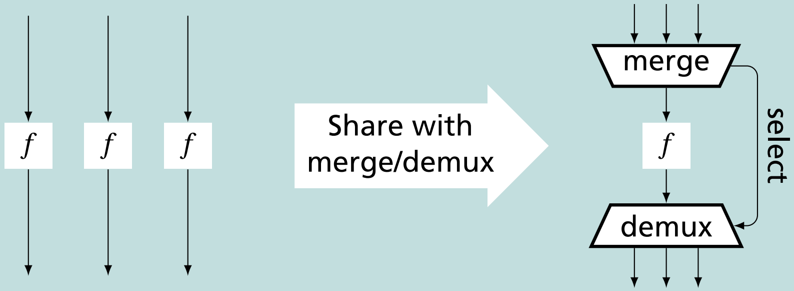
Flip-flop set after token sent suppresses duplicates

Valid out ignores ready of other outputs

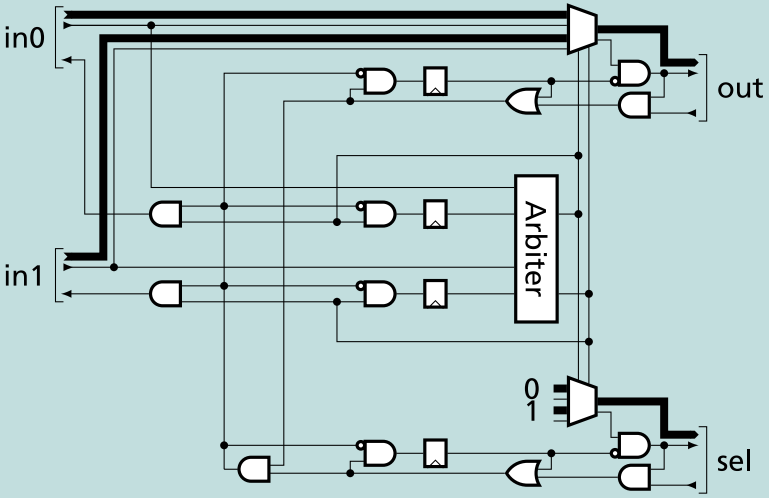


Input consumed once one token sent on every output

Nondeterministic Merge (Point 4/4)

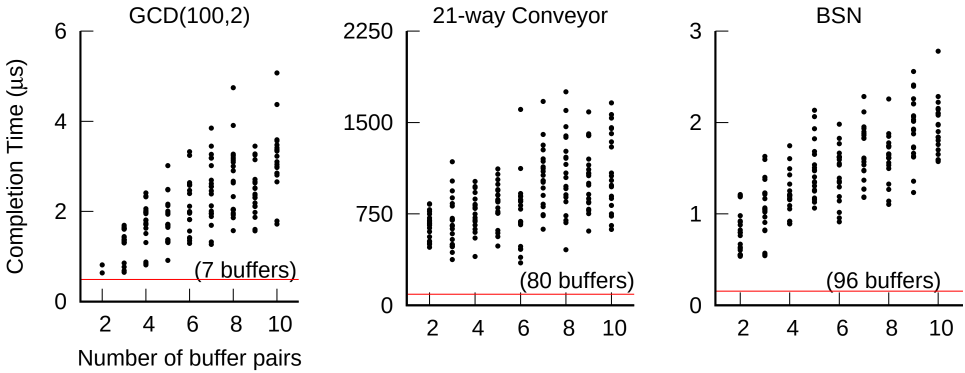


Two-Way Nondeterministic Merge Block w/ Select



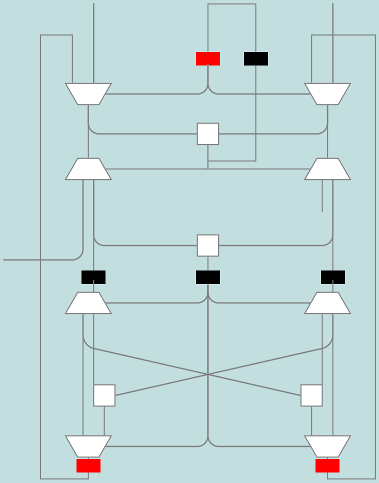
“Two-way fork with multiplexed output selected by an arbiter”

Experiments: Random Buffer Placement



Best Buffering for GCD (Manually Obtained)

Each loop has one of each buffer



⬛ Data Buffer

🟥 Control Buffer

Summary

Compositional Dataflow Networks as an IR

Patient dataflow blocks with valid/ready handshaking

1. Break downstream, upstream paths w/ two buffer types
2. Avoid comb. cycles: prohibit ready-to-valid paths
3. Add one state bit per output so forks may “race ahead”
4. Tame nondeterministic merge with a select output

Random buffer placement experiments show it works