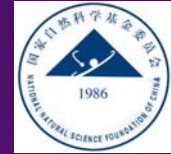


MANCHESTER  
1824

The University of Manchester

ASUNC 2015



# Deadlock Recovery in Asynchronous Networks on Chip in the Presence of Transient Faults

---

**Guangda Zhang, Jim Garside, Wei Song,  
Javier Navaridas, Zhiying Wang**

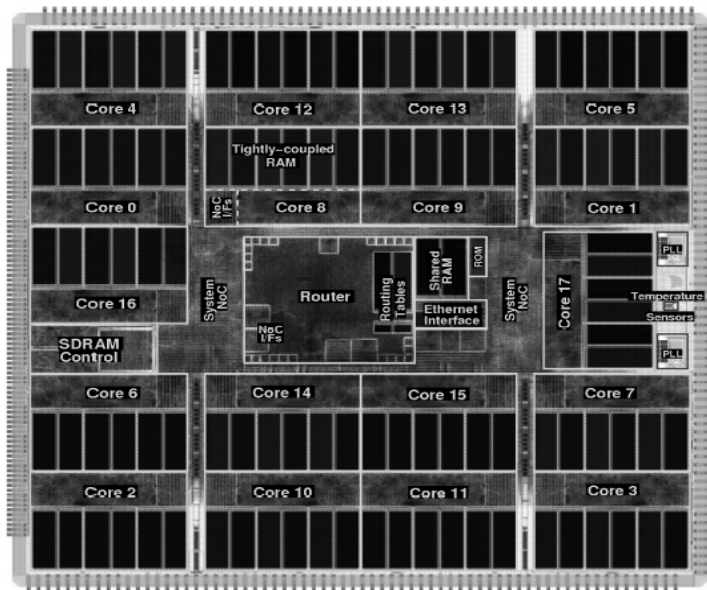
*APT Group, School of Computer Science  
University of Manchester, UK*

ASYNC 2015 – Mountain View, Silicon Valley, CA, USA, May 4-6, 2015

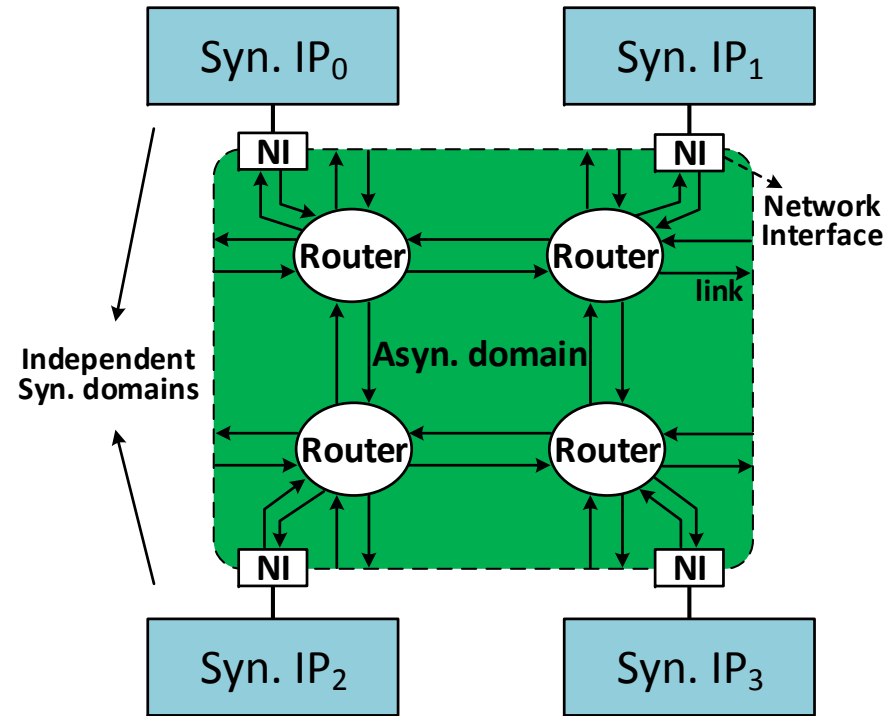
1

# Asynchronous Networks on Chip (NoCs)

1. Multi-core era --- Network-on-chip (NoC)
2. Asynchronous Networks on Chip --- GALS system



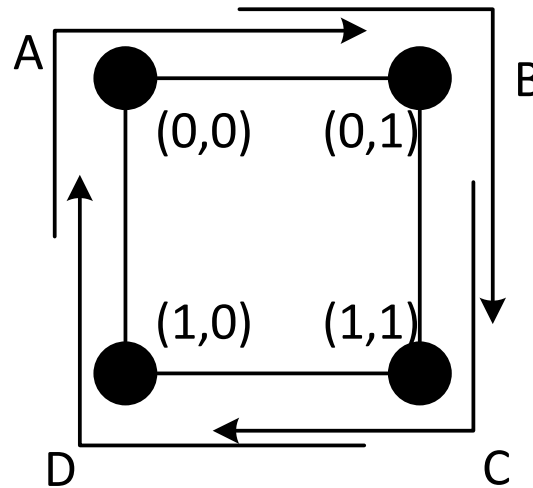
SpiNNaker MPSoC with 18 ARM cores



A GALS system constructed by an asynchronous NoC

## Deadlock in synchronous NoCs

### 1. Traditional deadlock: cyclic dependence



2. **Deadlock avoidance:** turn models, virtual channels

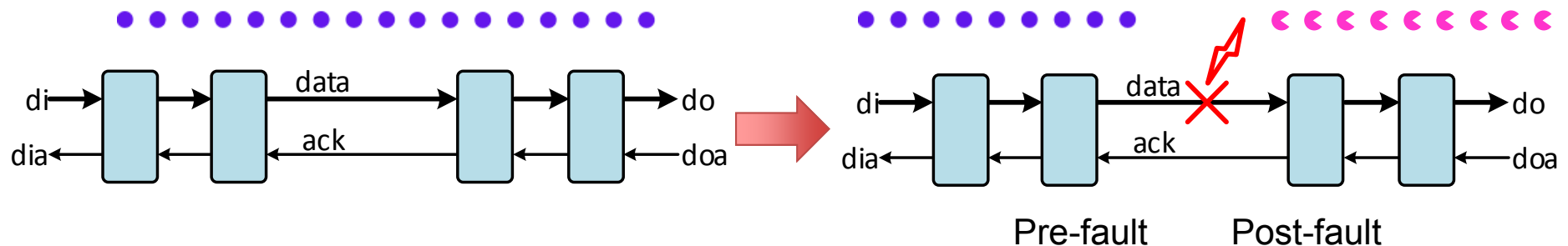
3. **Deadlock recovery:** deadlock or congestion?

# Deadlock in asynchronous NoCs

## 1. Traditional deadlock

## 2. Permanent faults\*

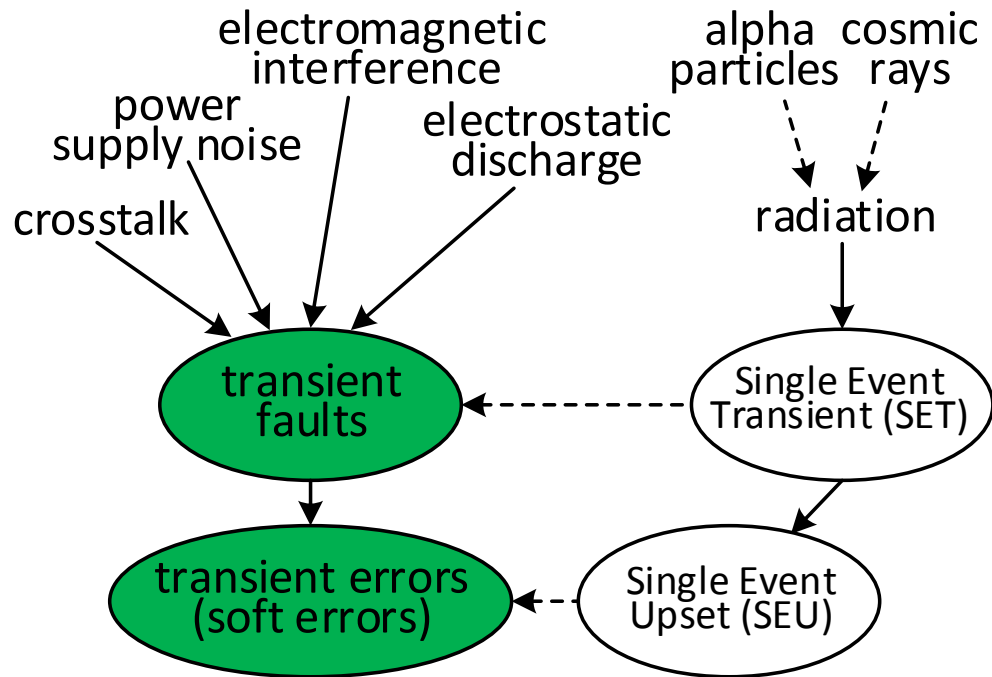
Deadlocked QDI pipeline caused by permanent faults:



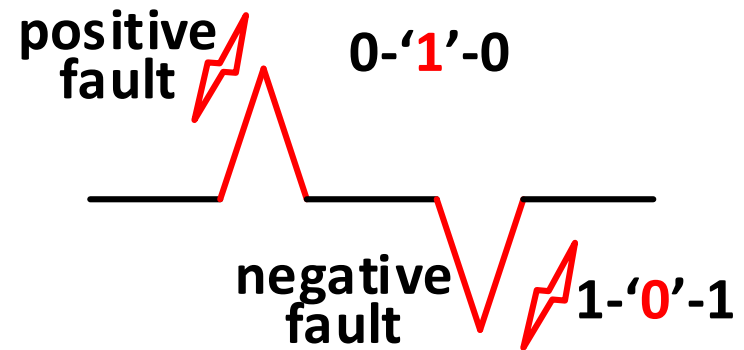
## Transient faults?

\*G. Zhang, etc., "An asynchronous SDM network-on-chip tolerating permanent faults," ASYNC 2014

# Transient faults



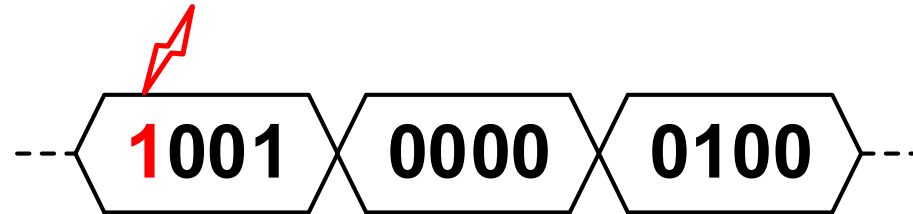
Sources of Transient faults



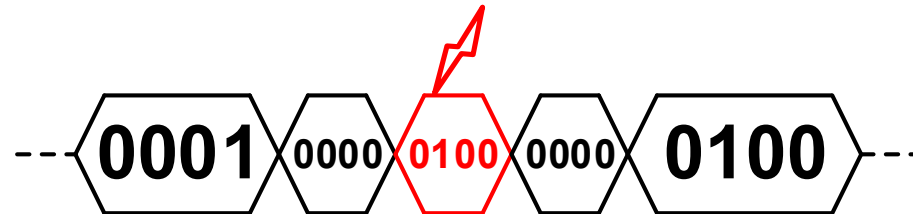
Positive/negative

## Impact of transient faults on QDI pipelines (4-phase 1-of-4 pipeline)

### 1. Symbol corruption\*



### 2. Symbol insertion\*



### 3. Deadlock ?

\*G. Zhang, etc, "Protecting QDI interconnects from transient faults using delay-insensitive redundant check codes," Microprocessors and Microsystems, 2014

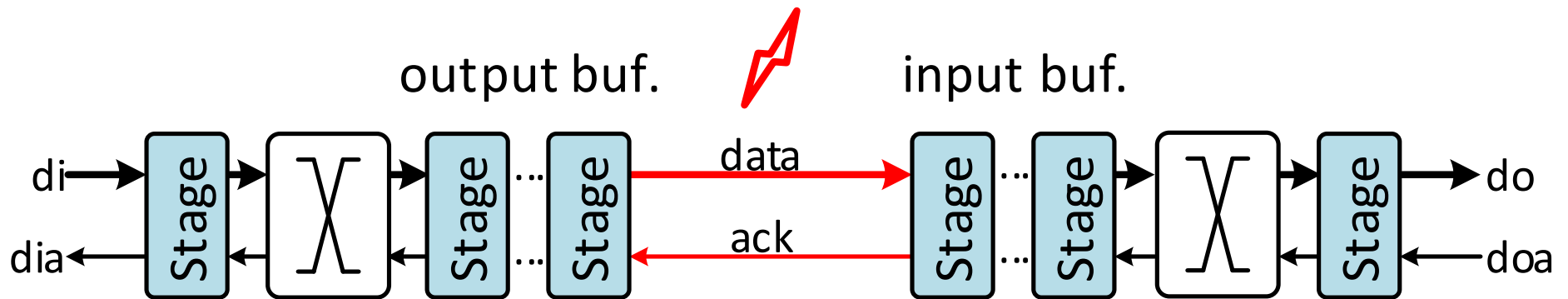
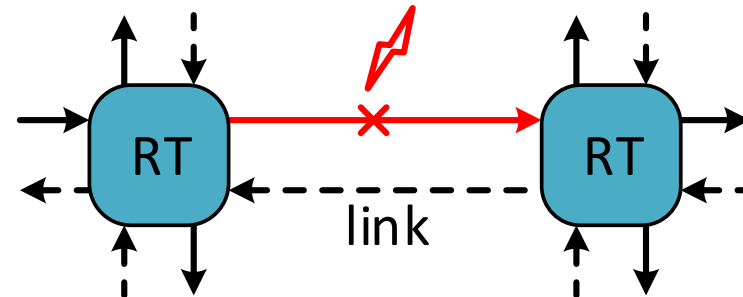
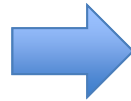
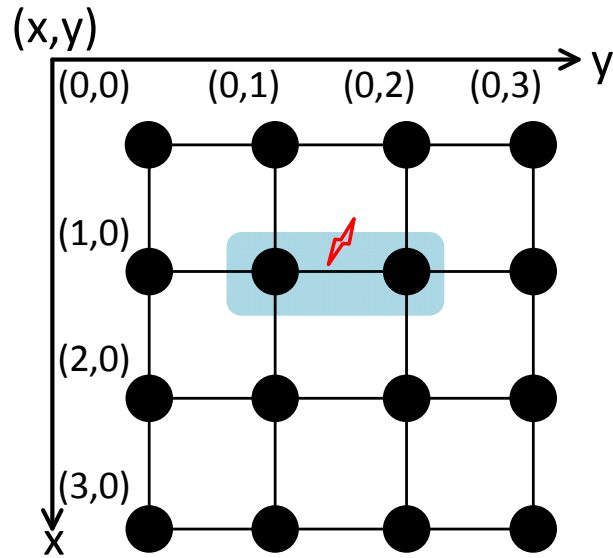
## Why this work is important?

### Deadlock ?

- Usual network deadlock (network/data link layer)
- Congestion
- Deadlock due to permanent faults (physical layer)
- Deadlock due to transient faults (physical layer)

1. Differentiate all deadlock types
2. All deadlock detection could use a common time-out mechanism
3. A fine-grained recovery mechanism  
(system reboot? NO!!!)

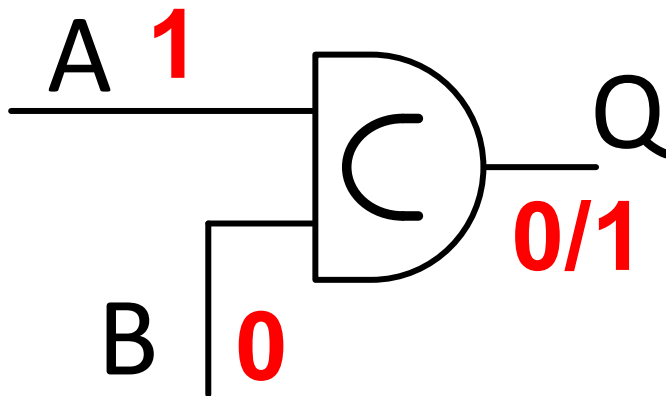
# Modelling QDI Asynchronous NoCs





## Deadlocked QDI pipeline

**Deadlock:** In a deadlocked QDI pipeline, no transitions could be fired any more and the pipeline gets stuck at a “stable” state.

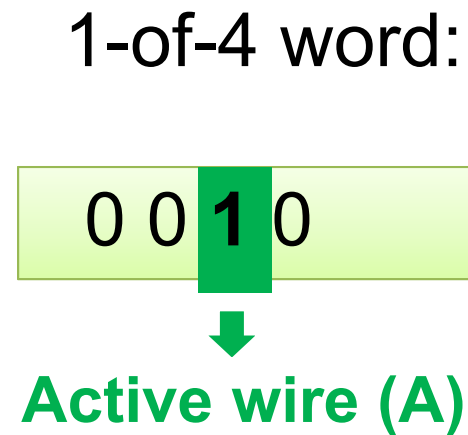
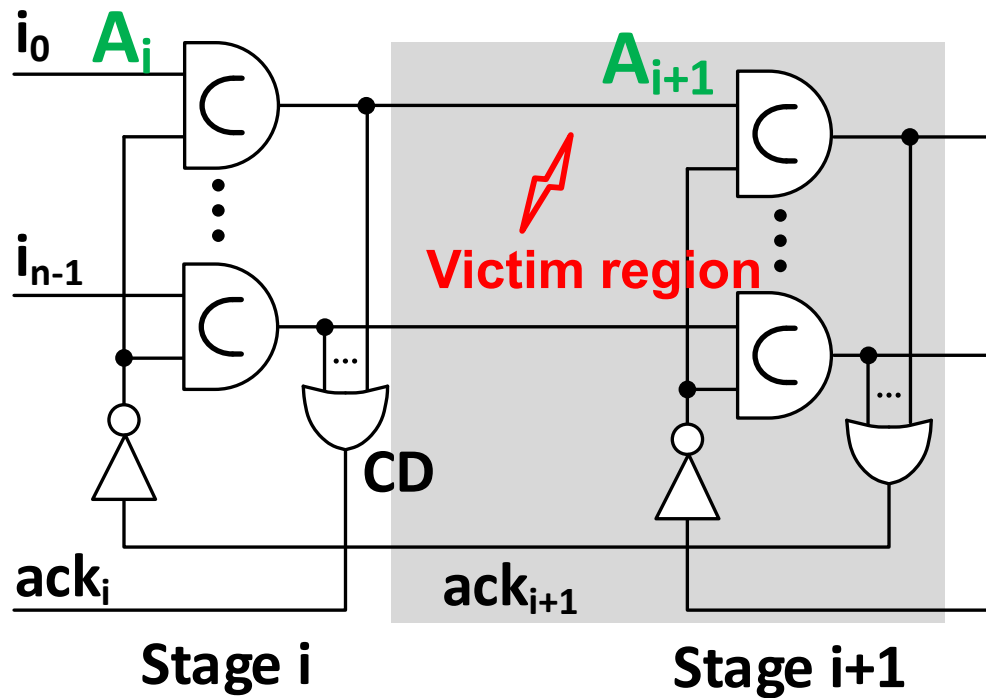


$$A \wedge B \rightarrow Q \uparrow$$

$$\neg A \wedge \neg B \rightarrow Q \downarrow$$

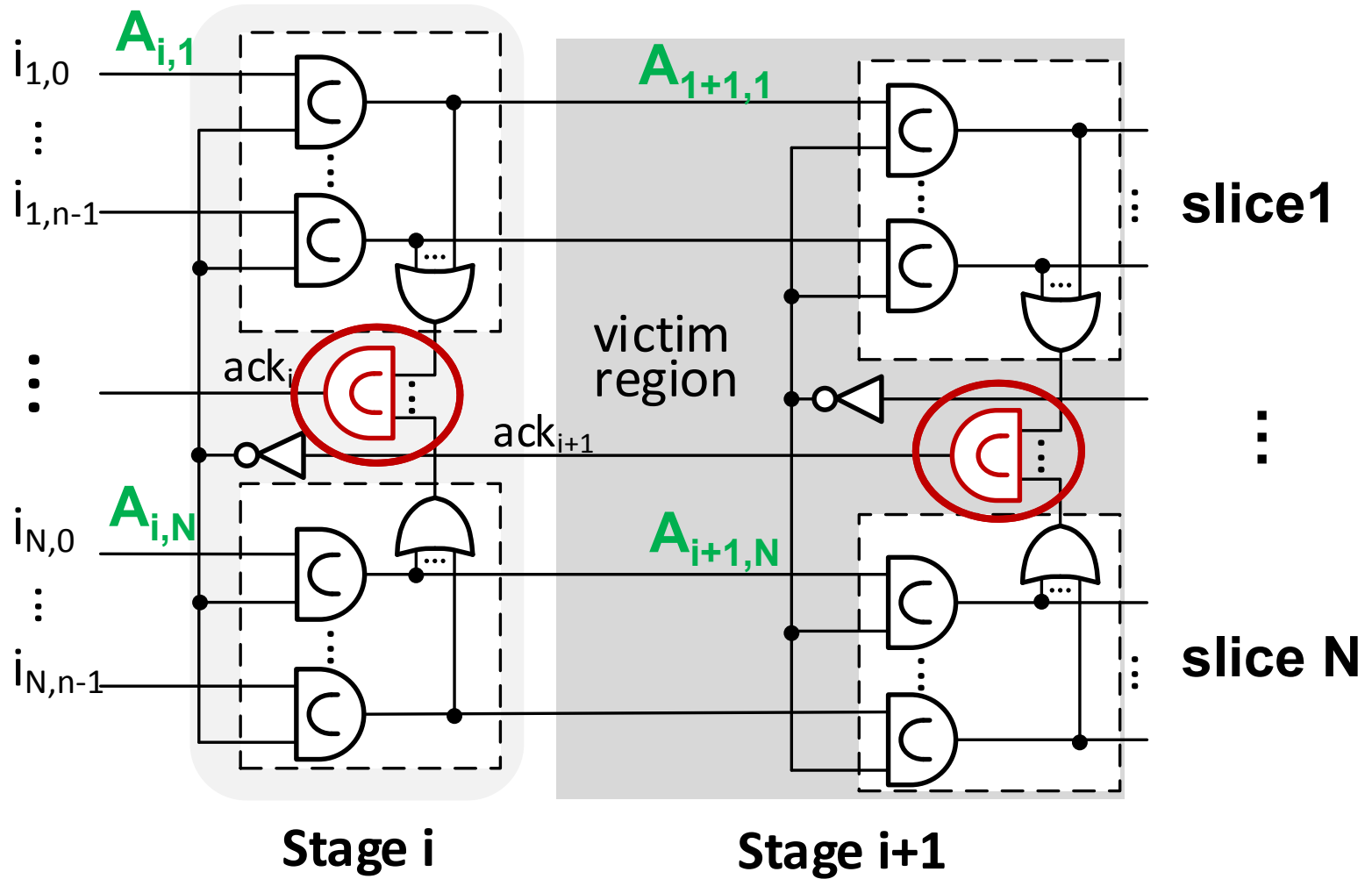
Deadlock state: (  $A==1, B==0$  ) or (  $A==0, B==1$  )

# Single-word 4-phase 1-of-n pipeline

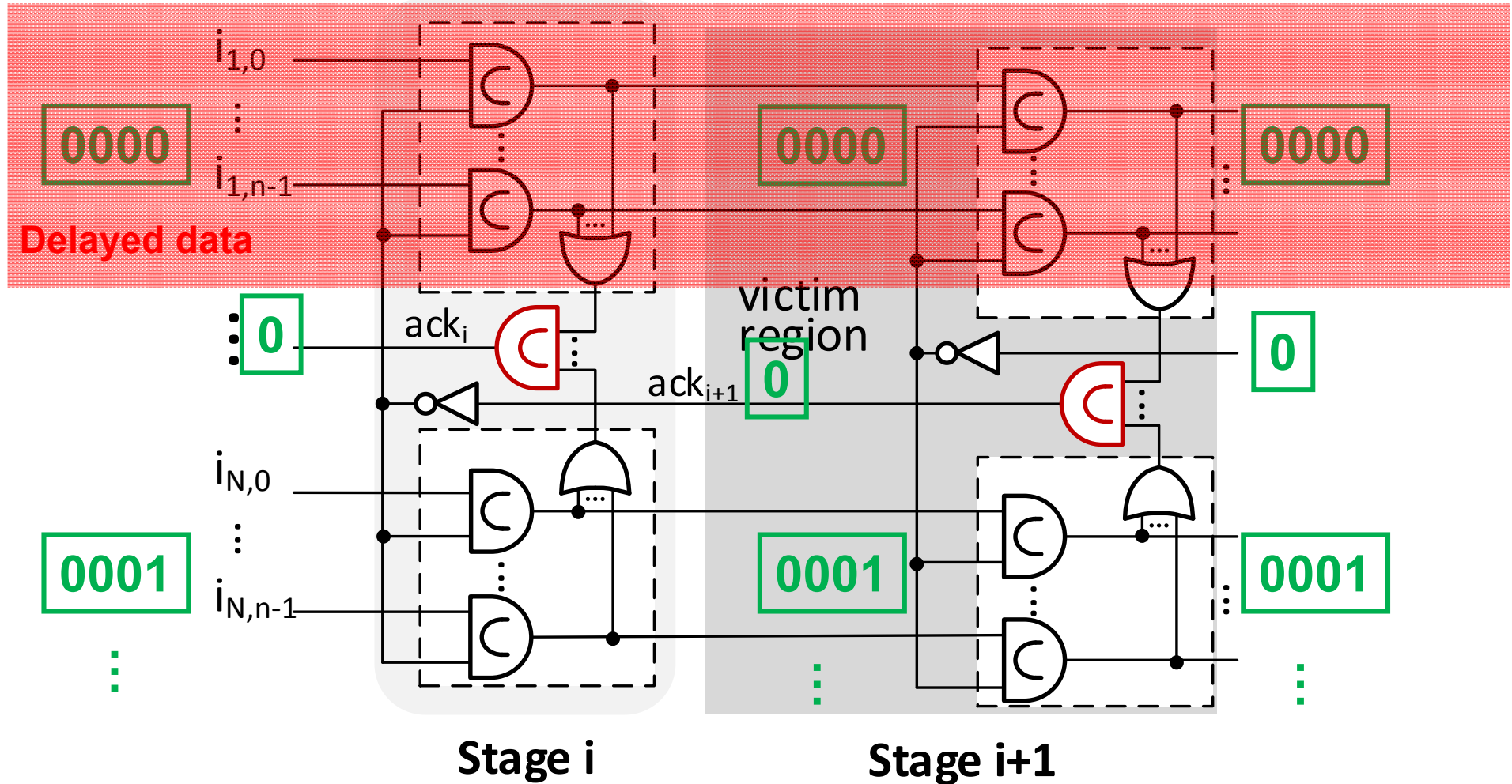


Theorem 1: A transient fault can cause symbol corruption and insertion, but **NOT** deadlock.

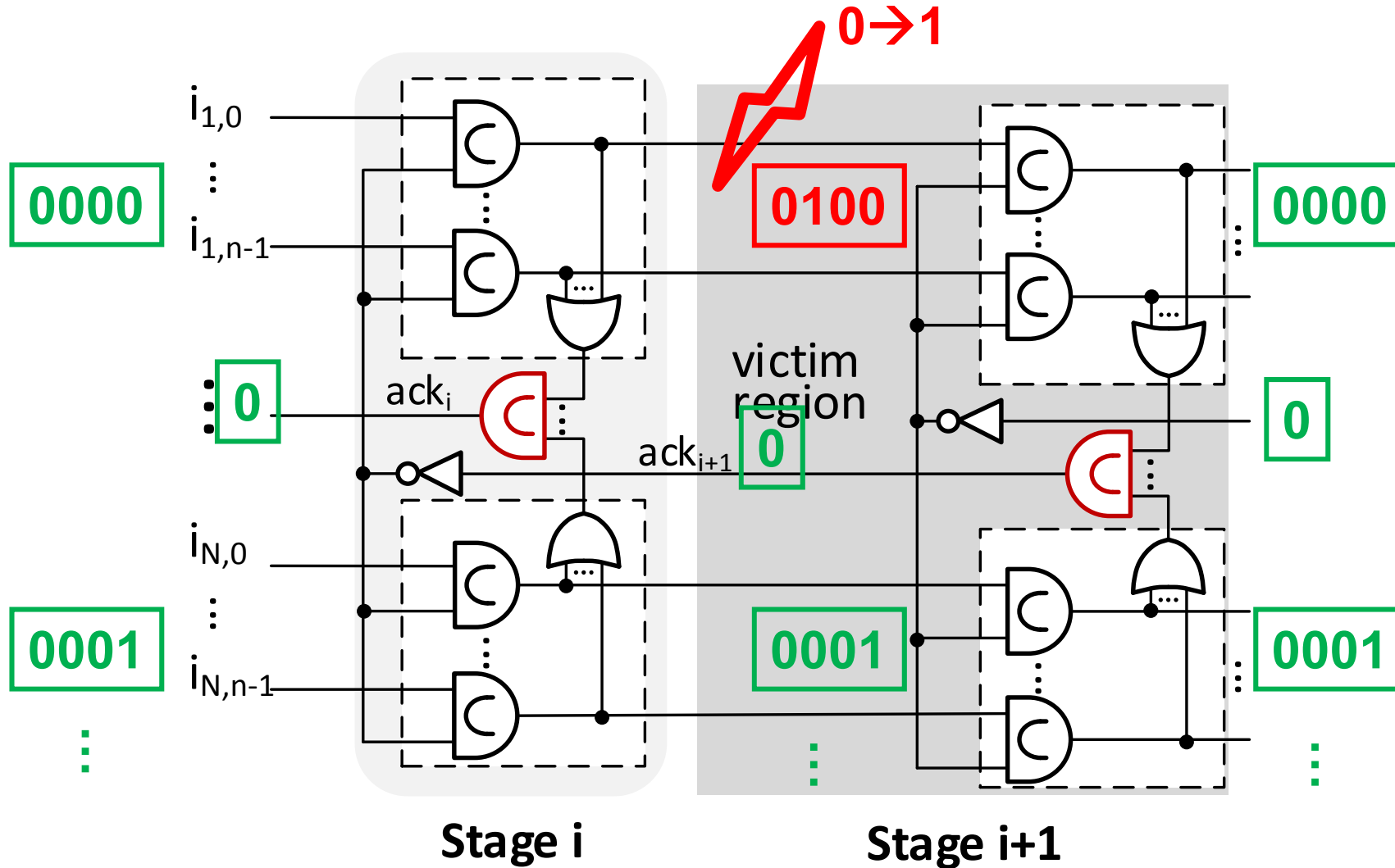
# Deadlock of a multi-word pipeline



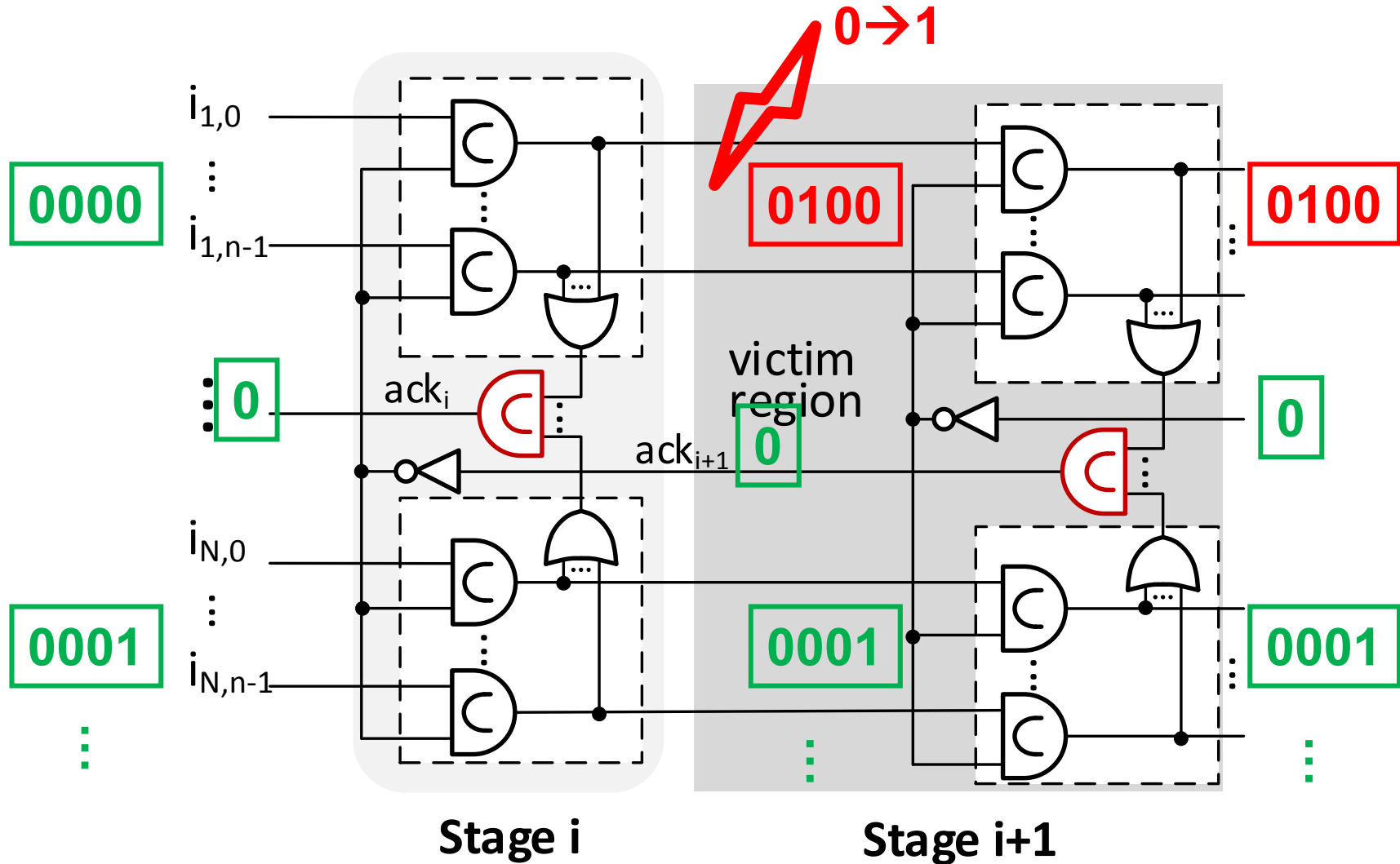
# Deadlock of a multi-word pipeline (1/7)



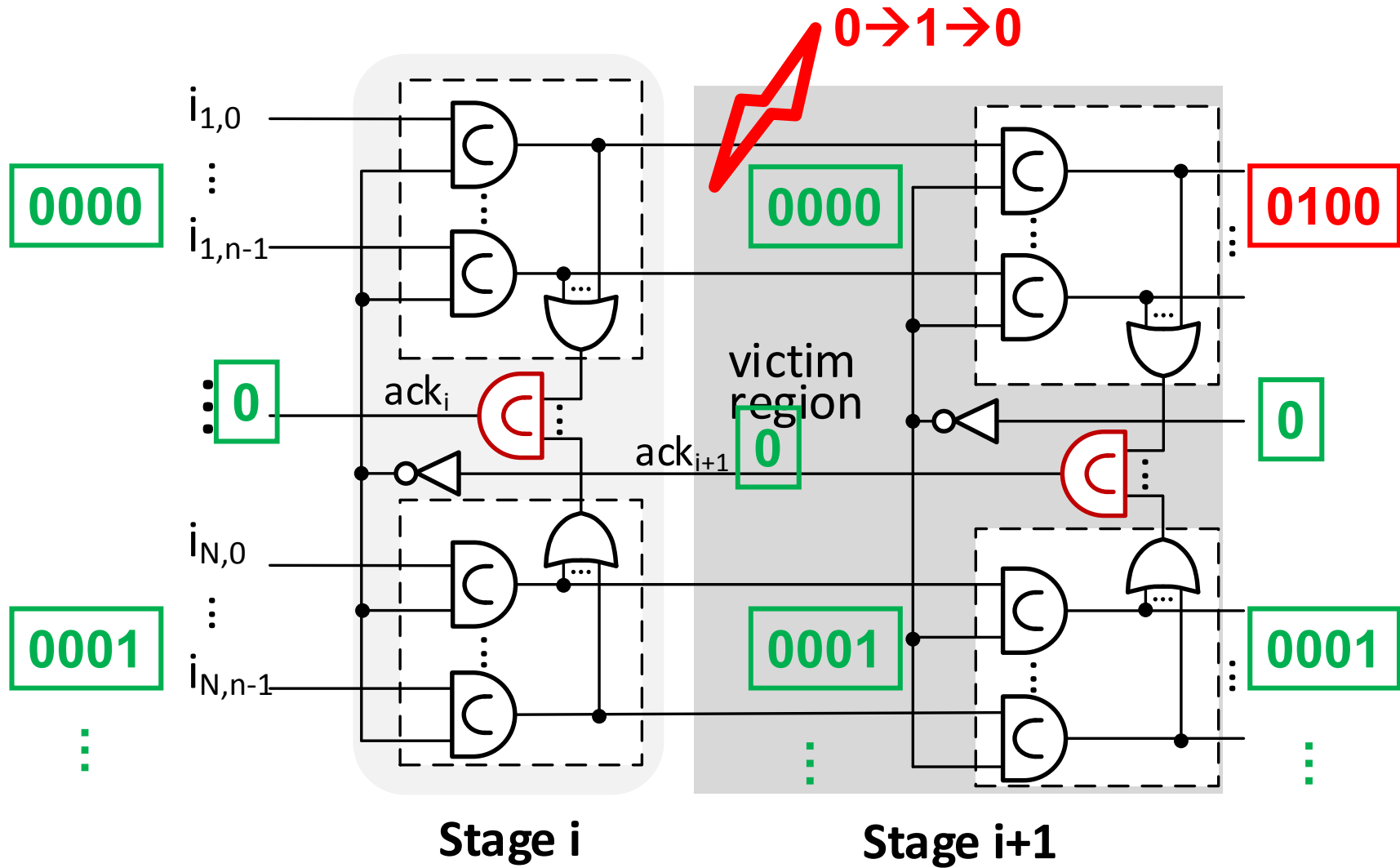
## Deadlock of a multi-word pipeline (2/7)



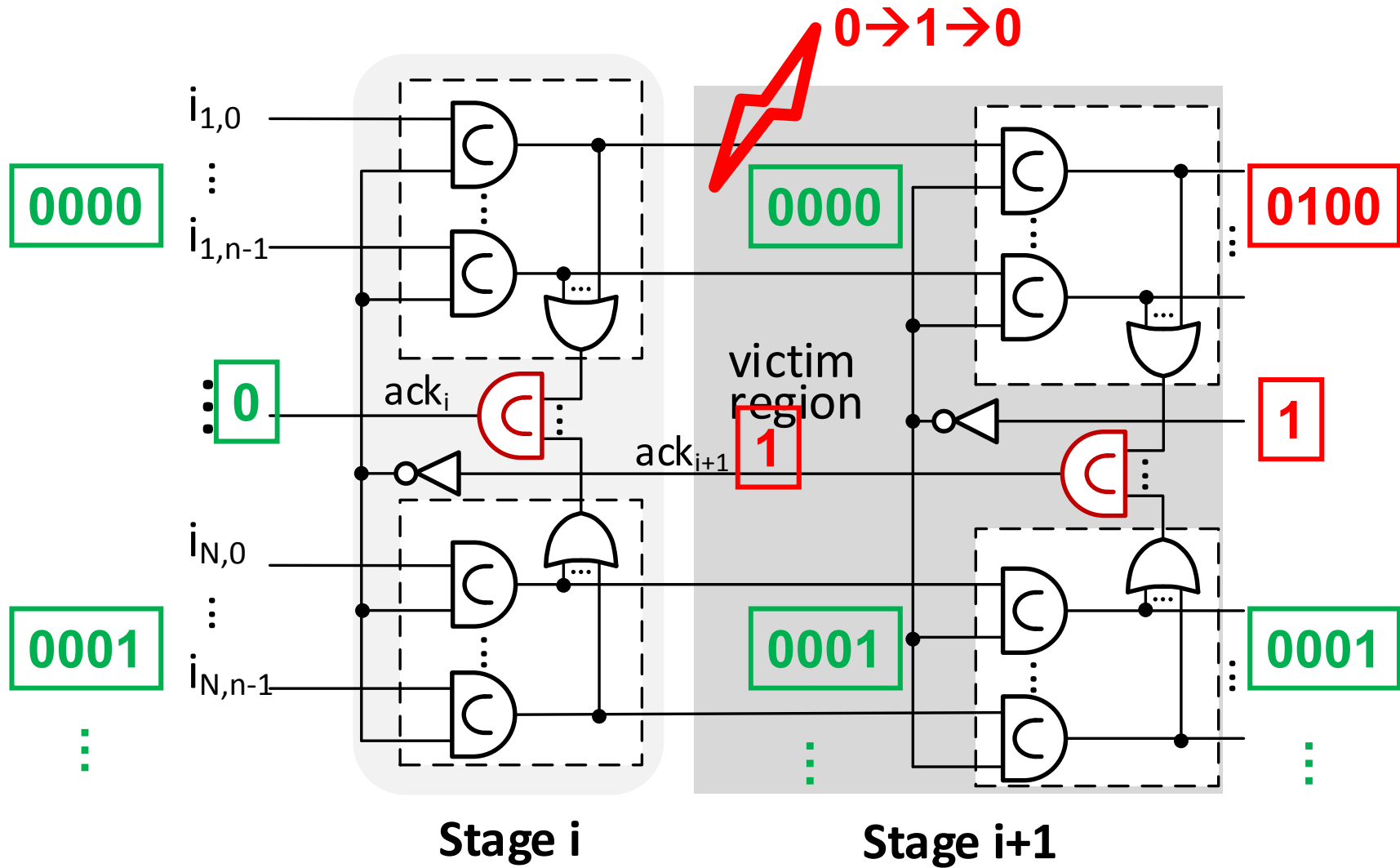
## Deadlock of a multi-word pipeline (3/7)



# Deadlock of a multi-word pipeline (4/7)

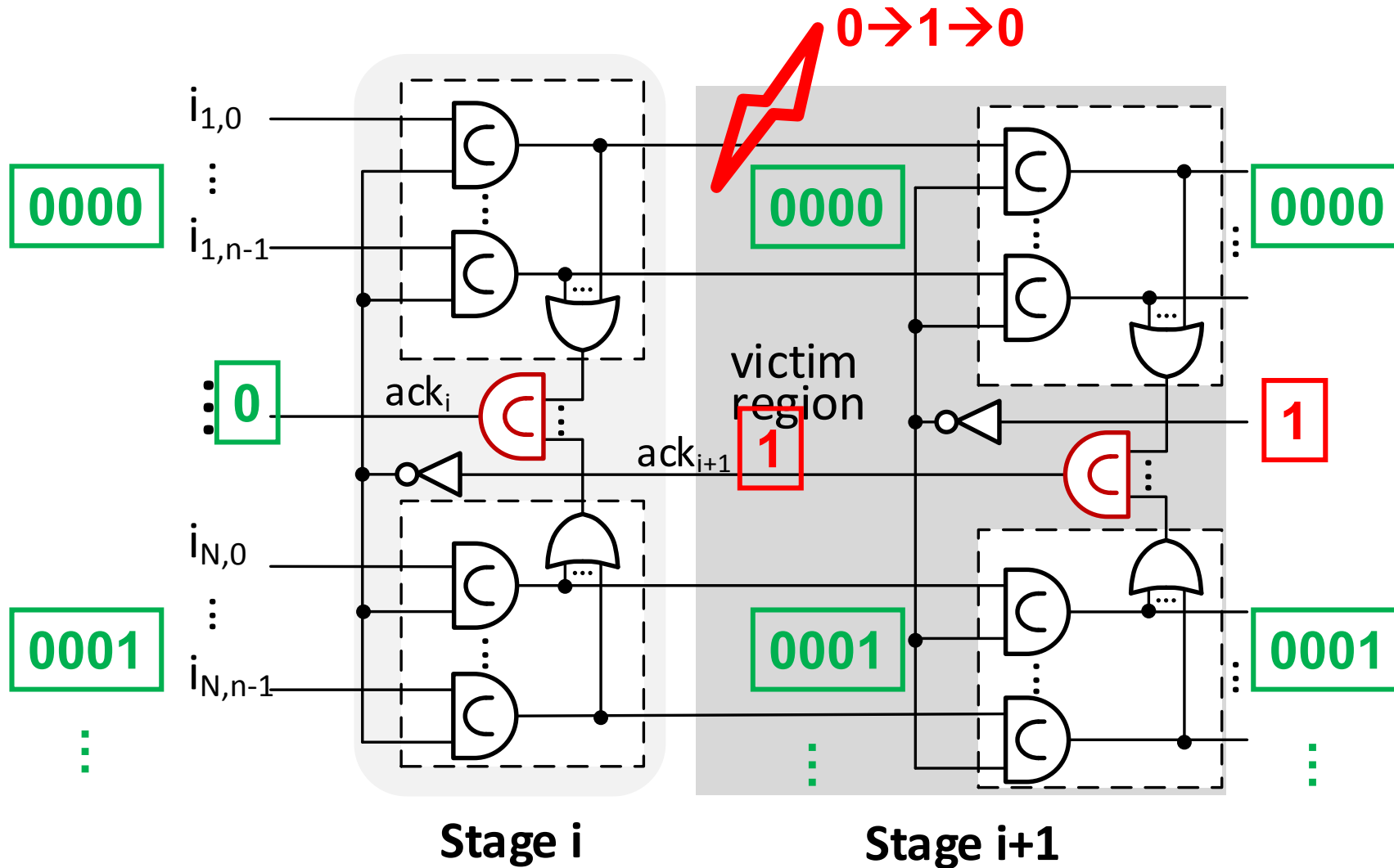


# Deadlock of a multi-word pipeline (5/7)

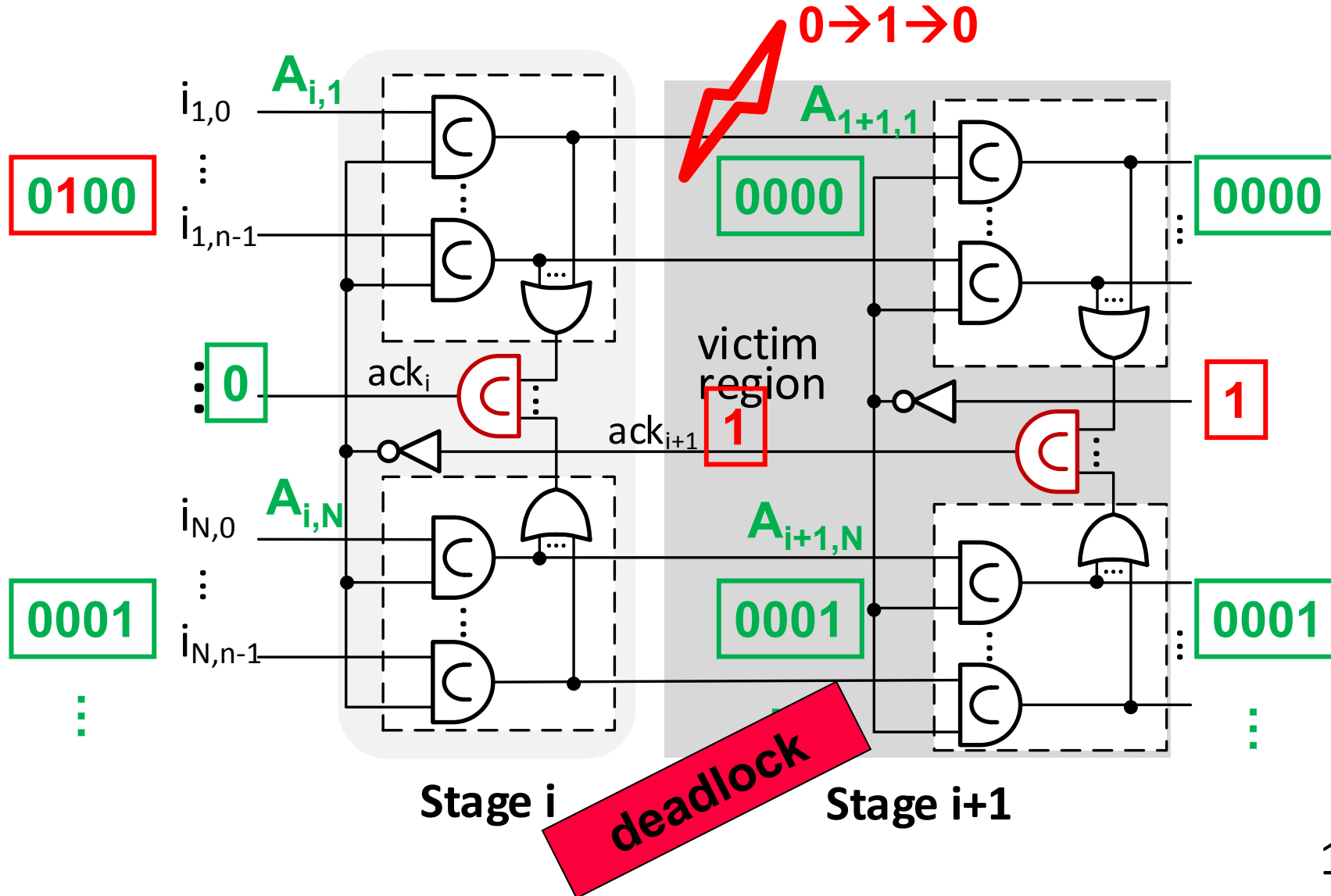




# Deadlock of a multi-word pipeline (6/7)



# Deadlock of a multi-word pipeline (7/7)



# Deadlock pattern comparison

**Positive transient fault:**

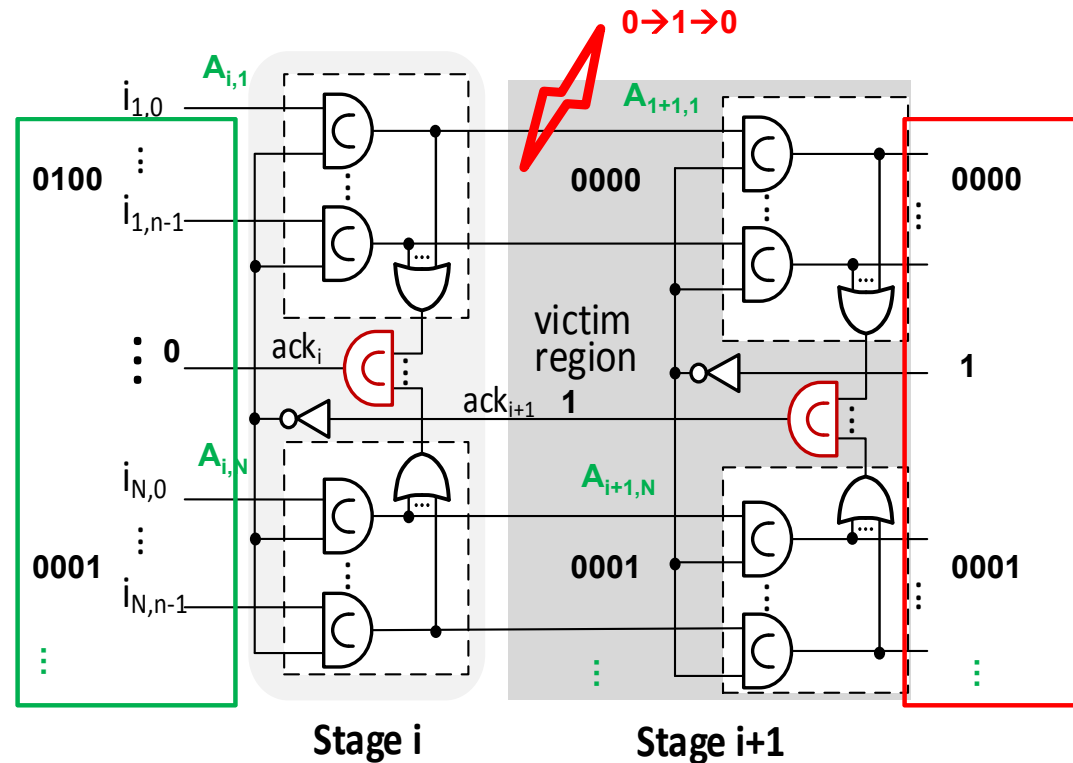
$$(\{A_{i,1}, \dots, A_{i,N}\}, \text{ack}_i, \{A_{i+1,1}, \dots, A_{i+1,N}\}, \text{ack}_{i+1}) = (\{1, \dots, 1\}, 0, \{0, 1, \dots, 1\}, 1)$$

*Pre-fault*

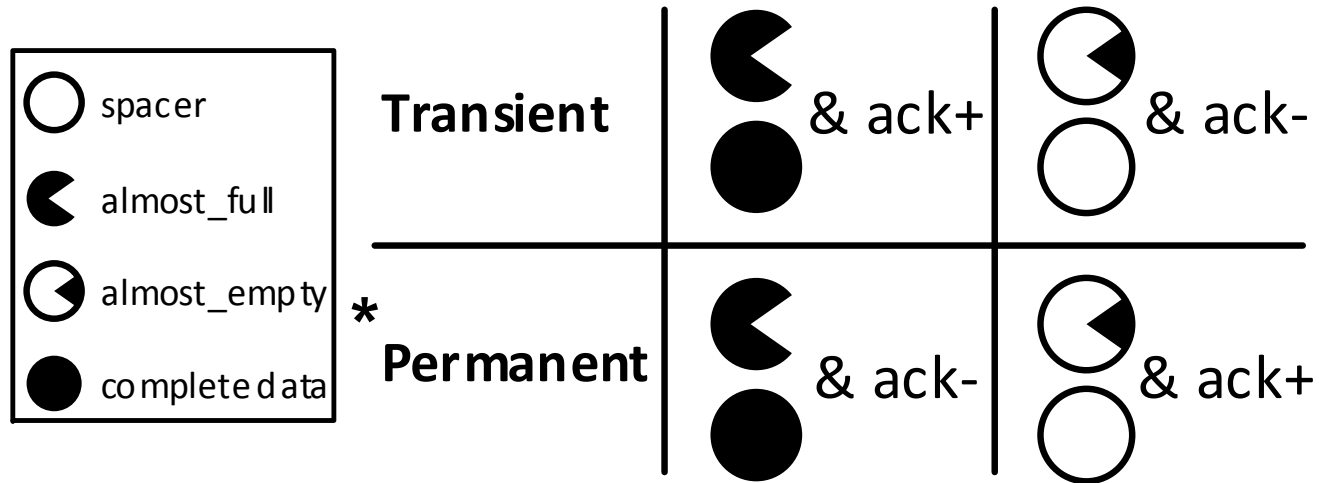
*Post-fault*

**complete**

**almost full**



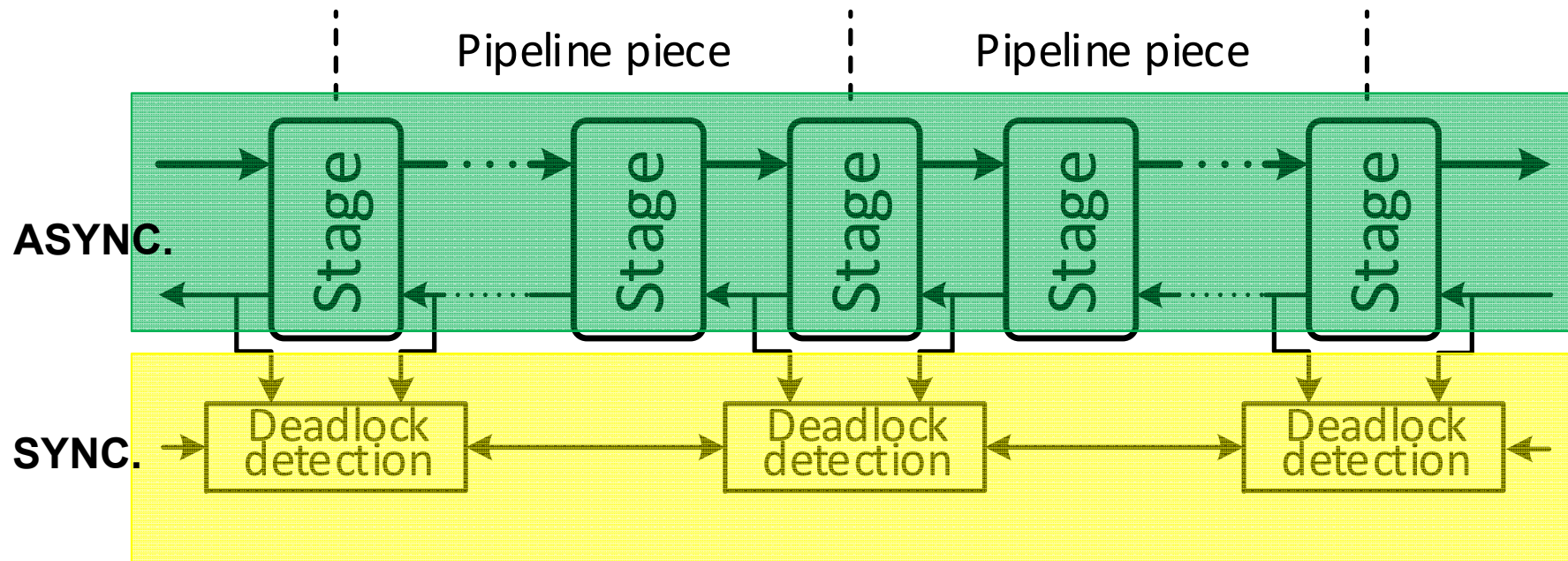
## Deadlock pattern comparison (transient or permanent)



```
ft_type = {(almost_full & !ack) | (almost_empty & ack);
           ((almost_full & ack) | almost_empty & !ack)}
```

00: default; 01:transient; 10: permanent; 11: invalid

## A general deadlock detection architecture



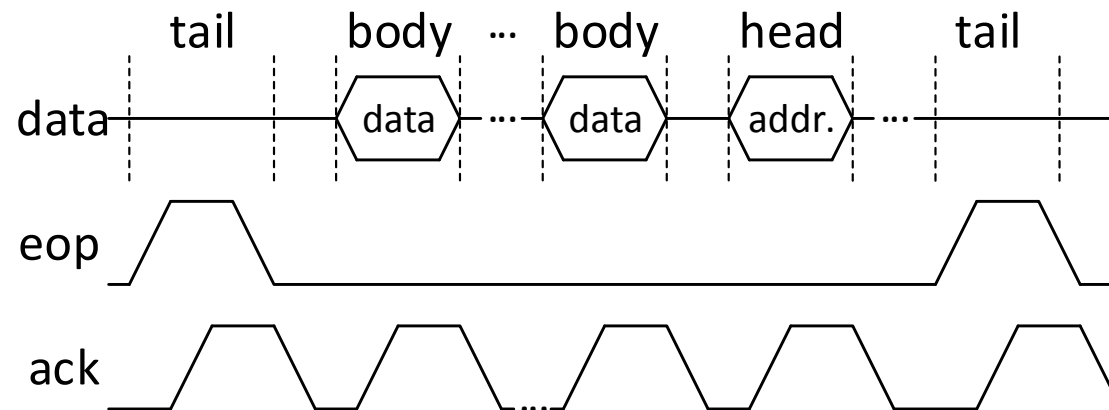
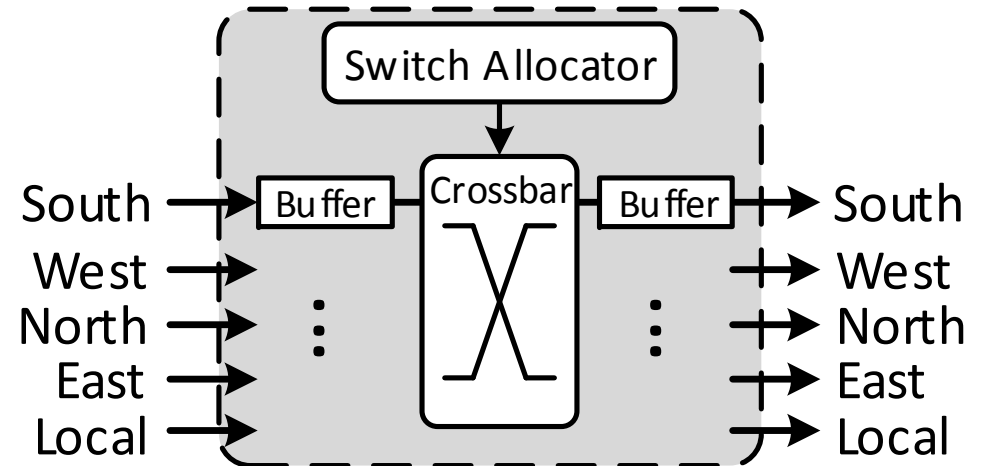
## Deadlock detection

- Usual network deadlock/congestion
- **Deadlock due to permanent/transient faults**

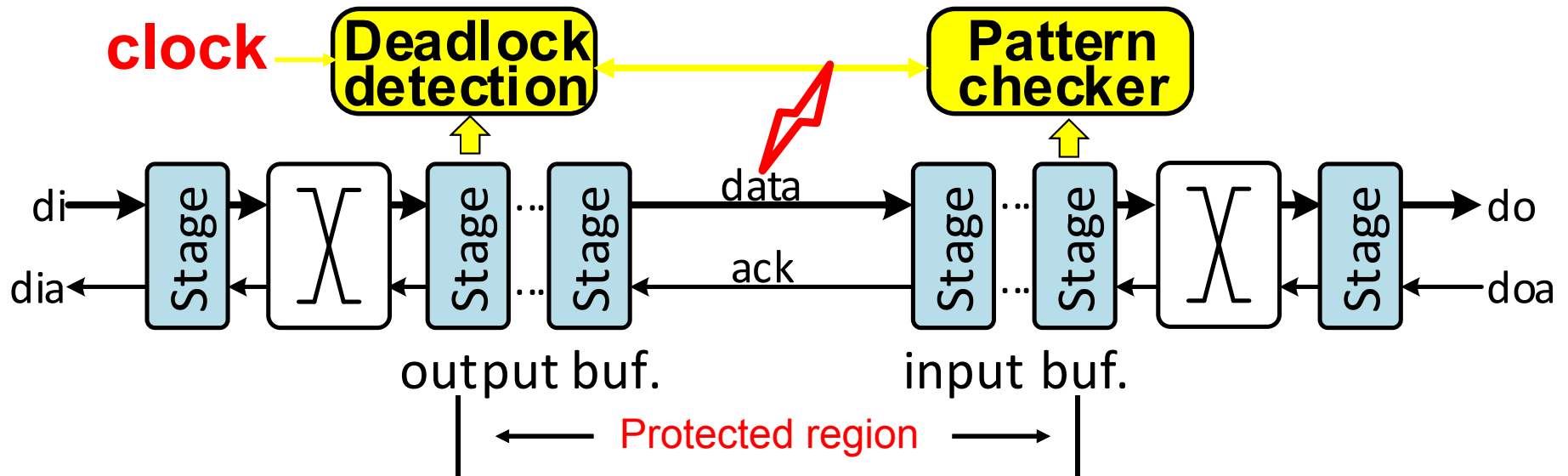
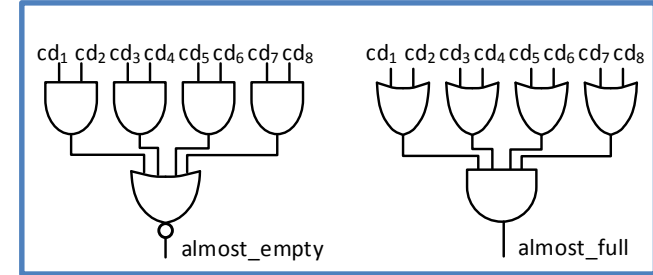
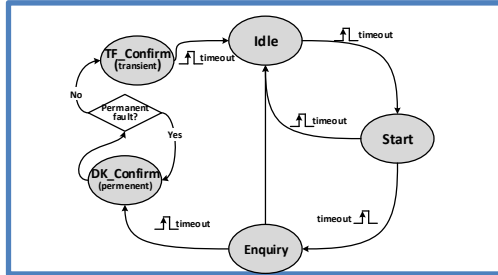
1. No transitions (a time-out)
2. All pipeline stages downstream of the fault have the same *ack* while the *ack* signals in stages upstream of fault are alternately valued (**deadlock by faults**)
3.  $ft\_type = \{(almost\_full \ \& \ !ack) \mid (almost\_empty \ \& \ ack);$   
 $((almost\_full \ \& \ ack) \mid almost\_empty \ \& \ !ack)\}$

## Network configuration

- **2D mesh QDI NoC**
- **4-phase 1-of-4**
- **Wormhole & XY-DOR**

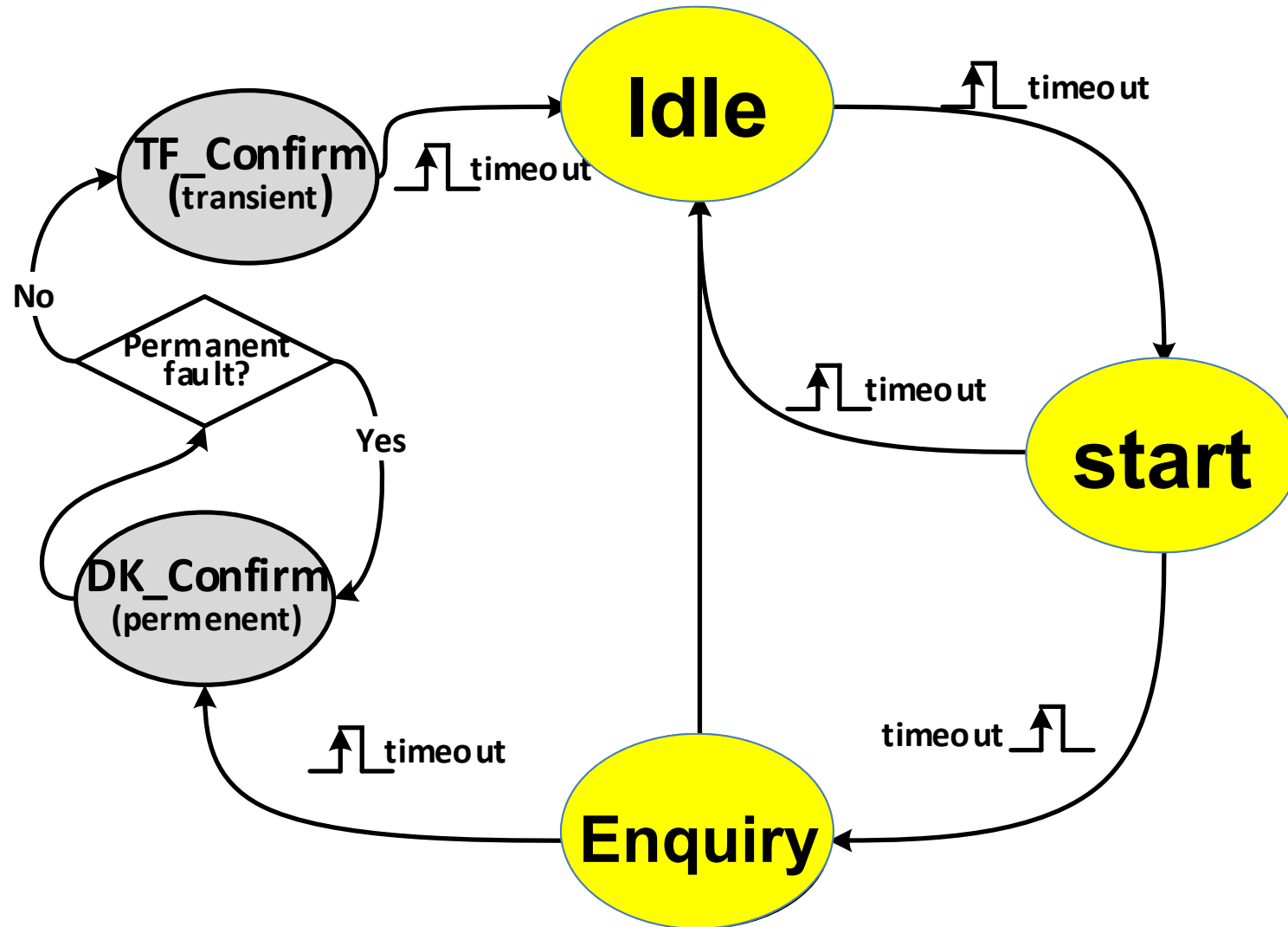


# Deadlock detection

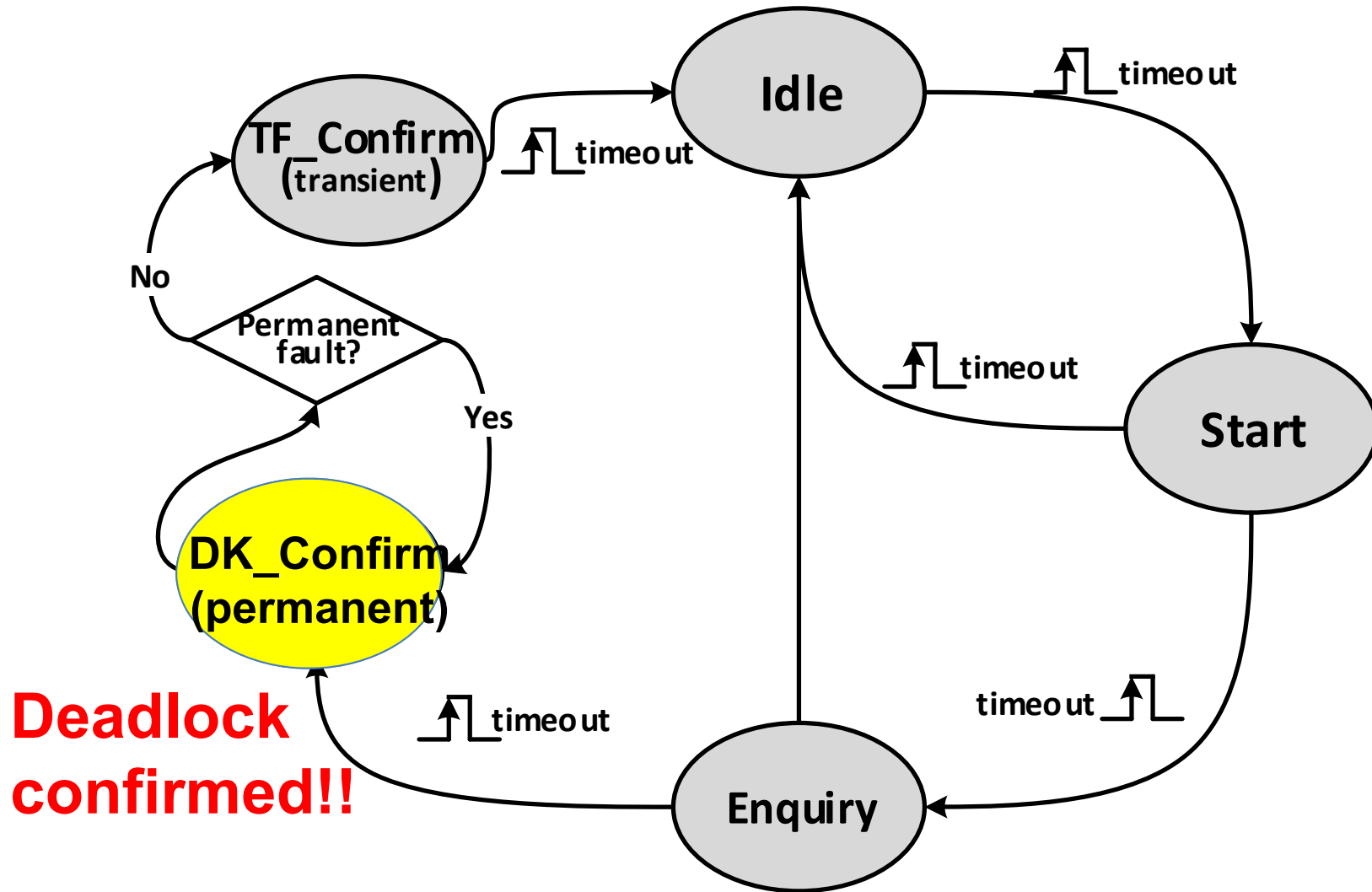




# Deadlock detection



# Deadlock detection



## Deadlock recovery

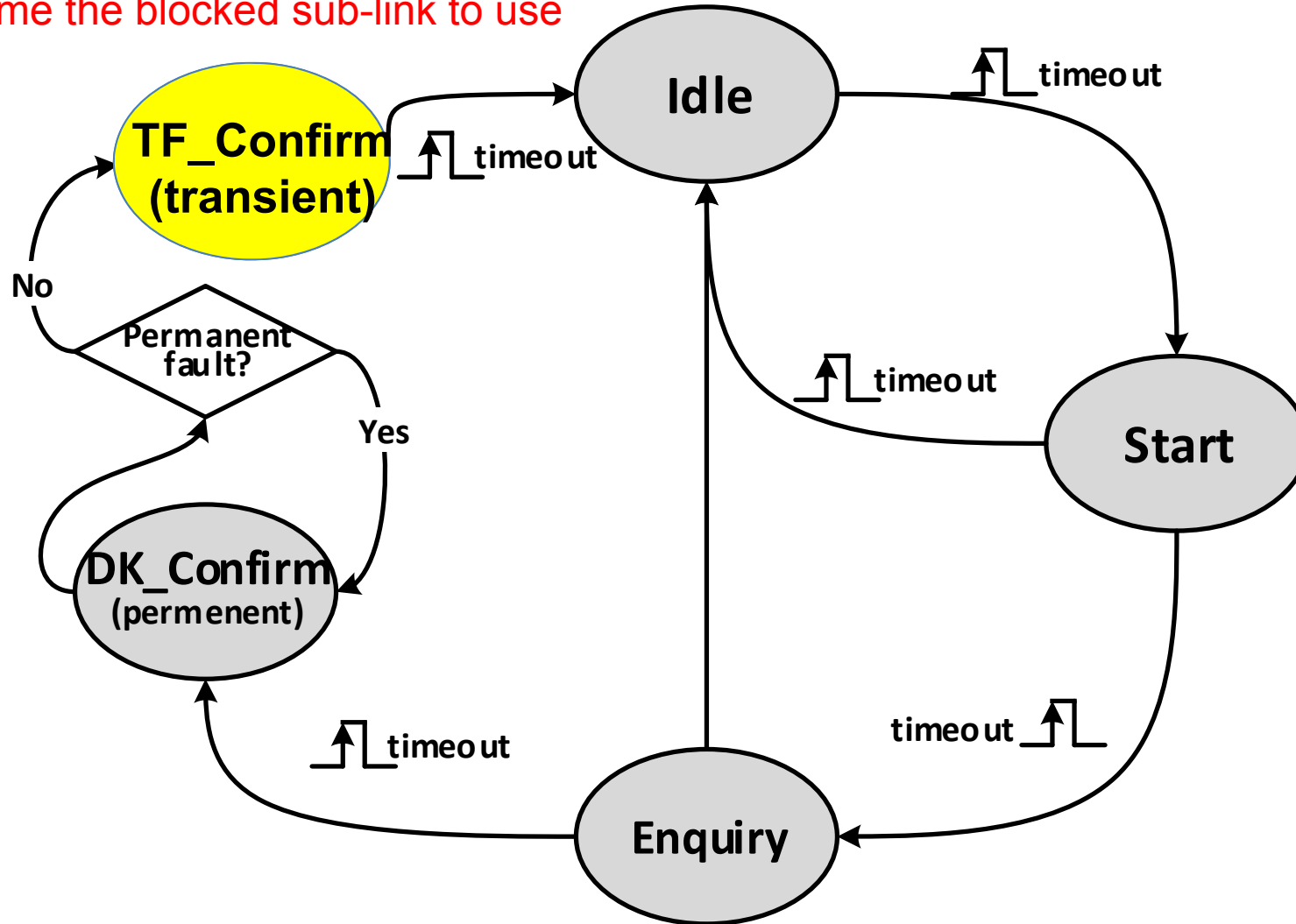
### 1. Permanent fault?

**Spatial division Multiplexing (SDM)** to divide each link into physically separated sub-links

- Block the defective sub-link
- Reconfigure the switch allocator
- Drain the flits polluted by the faults from the network

# Deadlock recovery

Resume the blocked sub-link to use



## Experimental Results

- UMC 130nm standard cell library
- Synchronous IP cores (SystemC) + post-synthesis routers
- 4 × 4 mesh 4-phase 1-of-4 SDM NoC
- packet size: 64 bytes; Local clock: 100MHz; Time-out: 1.5MHz

	Original	Protected	Overhead
<b>Area (um<sup>2</sup>)</b>	<b>63446</b>	<b>72394</b>	<b>14.1%</b>
<b>Throughput (Mbyte/s/node)</b>	<b>693</b>	<b>648</b>	<b>-6.5%</b>
<b>Energy (pJ/Byte)</b>	<b>3.7</b>	<b>4.3</b>	<b>16%</b>

## Conclusion

- **If the time difference between two slowest sub-pipelines are longer than the loop latency, a transient fault at the slowest sub-pipeline could cause deadlock.**
- **The patterns of the deadlock caused by transient faults, congestion and the usual deadlock are different, which can be used to detect the deadlock and tell its kind.**
- **For deadlock caused by transient faults, a fine-grained recovery mechanism is proposed to recovery the network to avoid expensive system reboot.**

# Thanks for your listening

## Questions?

Guangda Zhang  
University of Manchester