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
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:

[Diagram showing pre-fault and post-fault states]

Transient faults?

*G. Zhang, etc., “An asynchronous SDM network-on-chip tolerating permanent faults,” ASYNC 2014
Transient faults

Sources of Transient faults

- Electromagnetic interference
- Power supply noise
- Crosstalk
- Electrostatic discharge
- Radiation (alpha particles, cosmic rays)

Positive/negative fault transitions:
- Positive fault: 0→1→0
- Negative fault: 1→0→1

Transient errors (soft errors)

Single Event Transient (SET)

Single Event Upset (SEU)
Impact of transient faults on QDI pipelines
(4-phase 1-of-4 pipeline)

1. Symbol corruption*

   ![Symbol corruption diagram]

2. Symbol insertion*

   ![Symbol insertion diagram]

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

![Diagram of QDI Asynchronous NoCs](image)

- (x,y) coordinates
- RT: Reconfiguration Transmitter
- Link: Connection between stages
- Stage: Processing unit
- output buf.: Data buffer for output
- input buf.: Data buffer for input
- di: Data input
- dia: Data input acknowledge
- do: Data output
- doa: Data output acknowledge
- data: Data signal
- ack: Acknowledge signal
**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.

![Diagram of a QDI pipeline with inputs A and B and output Q]

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

\[A \land B \rightarrow Q \uparrow\]
\[\neg A \land \neg B \rightarrow Q \downarrow\]
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)

Delayed data

Stage i

Stage i+1
Deadlock of a multi-word pipeline (2/7)

Stage $i$

Stage $i+1$

$i_{1,0} \rightarrow i_{1,n-1} \rightarrow \ldots \rightarrow i_{N,0} \rightarrow i_{N,n-1}$

Ack$_i$\rightarrow Ack$_{i+1}$

0→1

Victim region

0000

0001

0000

0001

0000

0001
Deadlock of a multi-word pipeline (3/7)

Stage $i$

\[ i_{1,0} \quad \cdots \quad i_{1,n-1} \]

\[ \text{ack}_i \]

\[ i_{N,0} \quad \cdots \quad i_{N,n-1} \]

Stage $i+1$

\[ \text{victim region} \]

\[ \text{ack}_{i+1} \]

\[ 0 \rightarrow 1 \]

\[ 0000 \quad \cdots \quad 0100 \quad \cdots \quad 0001 \quad \cdots \quad 0000 \]

\[ 0001 \quad \cdots \quad 0100 \quad \cdots \quad 0001 \quad \cdots \quad 0001 \]
Deadlock of a multi-word pipeline (4/7)

Stage $i$

Stage $i+1$

$0 \rightarrow 1 \rightarrow 0$

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

Stage $i$

Stage $i+1$

0 → 1 → 0

victim region

$0000$

$0001$

$0100$

$0001$

$0001$

$0100$
Deadlock of a multi-word pipeline (6/7)

Stage i

Stage i+1

0 \rightarrow 1 \rightarrow 0
Deadlock of a multi-word pipeline (7/7)
Deadlock pattern comparison

**Positive transient fault:**

\[
(\{A_{i,1},\ldots, A_{i,N}\}, \text{ack}_i, \{A_{i+1,1},\ldots, A_{i+1,N}\}, \text{ack}_{i+1} ) = (\{1,\ldots,1\}, 0, \{0,1,\ldots,1\}, 1)
\]

<table>
<thead>
<tr>
<th>Pre-fault</th>
<th>Post-fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>complete</td>
<td>almost full</td>
</tr>
</tbody>
</table>

\[
\begin{array}{c}
0100 \\
\vdots \\
01_{n-1} \\
\vdots \\
0001 \\
\vdots \\
0000 \\
\vdots \\
0000 
\end{array}
\quad \quad
\begin{array}{c}
0000 \\
\vdots \\
0000 \\
\vdots \\
0000 \\
\vdots \\
1 \\
\vdots \\
0001 
\end{array}
\]

**Stage i**

**Stage i+1**
Deadlock pattern comparison
(transient or permanent)

spacer  almost_full  almost_empty  complete data

 transient
      & ack+  & ack-

 permanent
      & ack-  & ack+

\[ft\_type = \{(\text{almost\_full} \& \neg \text{ack}) \mid (\text{almost\_empty} \& \text{ack})\};\]
\[((\text{almost\_full} \& \text{ack}) \mid \text{almost\_empty} \& \neg \text{ack})\}\]

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

*G. Zhang, etc., “An asynchronous SDM network-on-chip tolerating permanent faults,” ASYNC 2014
A general deadlock detection architecture

SYNC.

Deadlock detection

ASYNC.

Stage

Deadlock detection

Stage

Stage

Stage

Stage

Pipeline piece

Pipeline piece
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. $ft_{\text{type}} = ((\text{almost\_full} \& !\text{ack}) \lor (\text{almost\_empty} \& \text{ack}) ;
   ((\text{almost\_full} \& \text{ack}) \lor \text{almost\_empty} \& !\text{ack}))$
Network configuration

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

Idle

TF_Confirm (transient)

Permanent fault?

No

Yes

DK_Confirm (permanent)

Enquiry

start

timeout

timeout

timeout

timeout

timeout
Deadlock detection

TF_Confirm (transient) → Idle → Start → DK_Confirm (permanent)

Deadlock confirmed!!
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

*G. Zhang, etc., “An asynchronous SDM network-on-chip tolerating permanent faults,” ASYNC 2014
Resume the blocked sub-link to use
Experimental Results

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

<table>
<thead>
<tr>
<th></th>
<th>Original</th>
<th>Protected</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (um2)</td>
<td>63446</td>
<td>72394</td>
<td>14.1%</td>
</tr>
<tr>
<td>Throughput (Mbyte/s/node)</td>
<td>693</td>
<td>648</td>
<td>-6.5%</td>
</tr>
<tr>
<td>Energy (pJ/Byte)</td>
<td>3.7</td>
<td>4.3</td>
<td>16%</td>
</tr>
</tbody>
</table>
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