FUJITSU

shaping tomorrow with you

Design Methodology for SoCs in Large Supercomputer Systems

Akira ASATO
LSI Development Division
Next Generation Technical Computing Unit
Fujitsu Limited

MPSOC2014, July 9, 2014

Constraint of the Chip Development

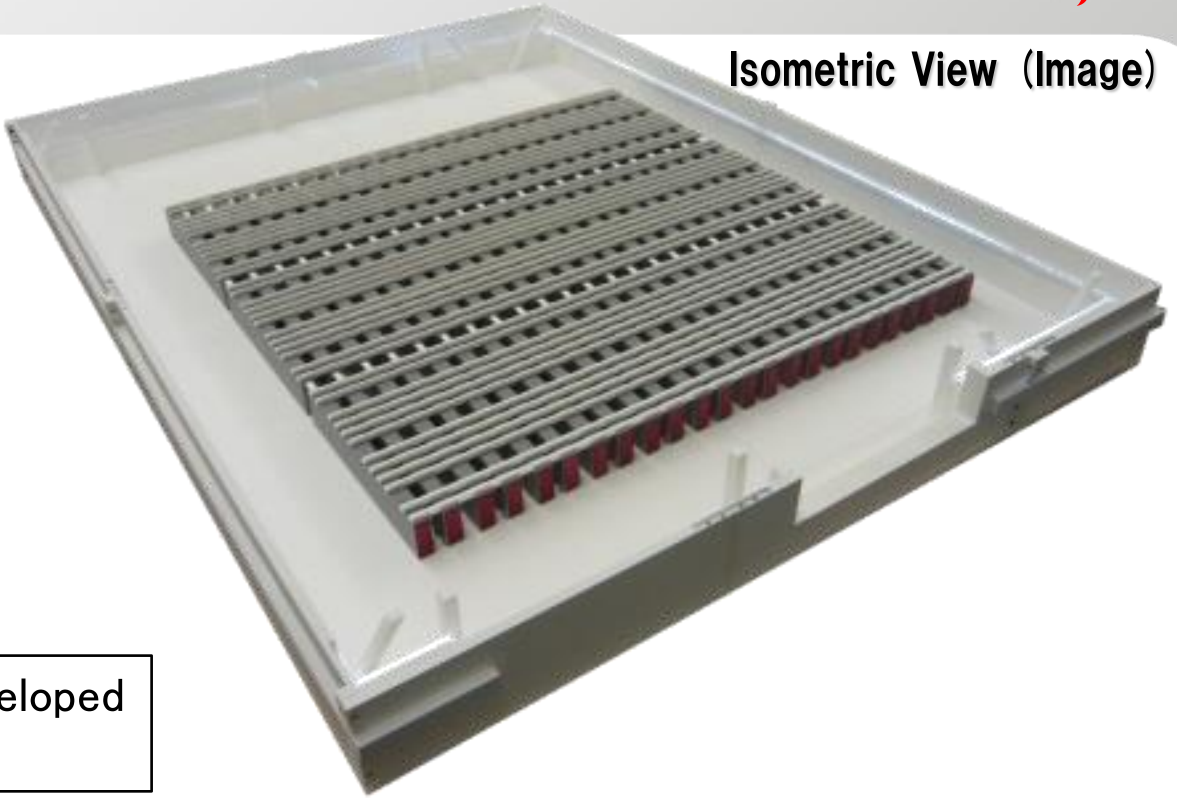
- Die Size
  - Total capacity of the logic is limited
- Power Consumption
  - Should fit within the budget
- Schedule
  - Must be completed by the due date
  - Most time-consuming process is verification, not design

<u>Design techniques for the verification are desired</u>

Presentation Topic
Based on our experience of the K computer project, design and verification techniques for a large scale system will be discussed.

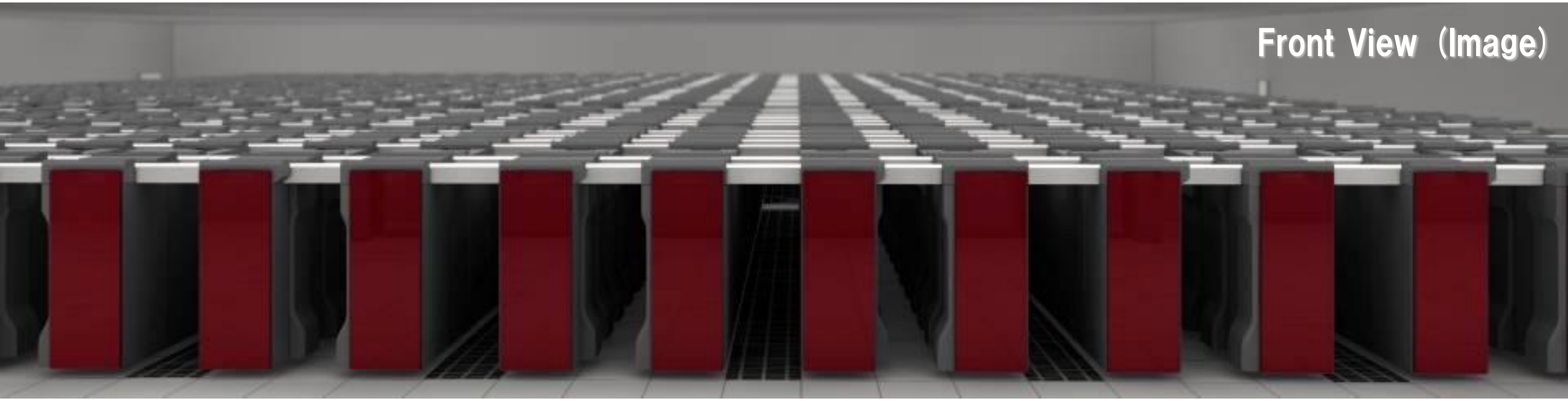
K computer

- # of Racks
  - 36 x 24 = 864 racks
- # of CPUs
  - 88,128
- Total Memory Size
  - More than 1.40PB



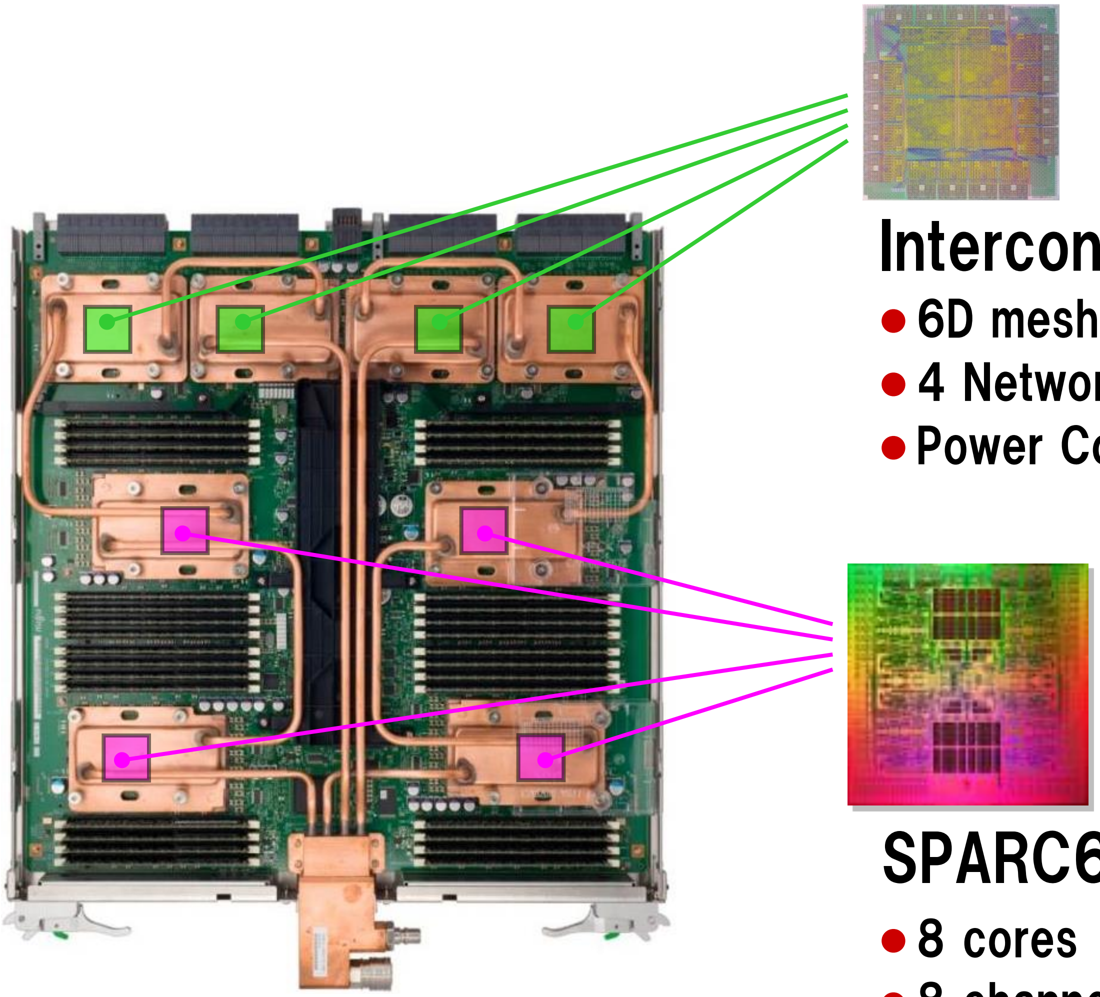
Isometric View (Image)

K computer has been jointly developed by RIKEN and Fujitsu.



Front View (Image)

Chips in the K computer



Interconnect Controller (ICC)

- 6D mesh/torus network topology
- 4 Network Interfaces
- Power Consumption 28W

SPARC64^{TM} VIIIfx (CPU)

- 8 cores
- 8 channel memory controllers
- Power Consumption 58W (Typ.)

System Board

Challenges of Verification

Verification Period ＝ （# of Items）／（Verification Speed）

Exponentially increases with design size

In order to reduce the verification period,

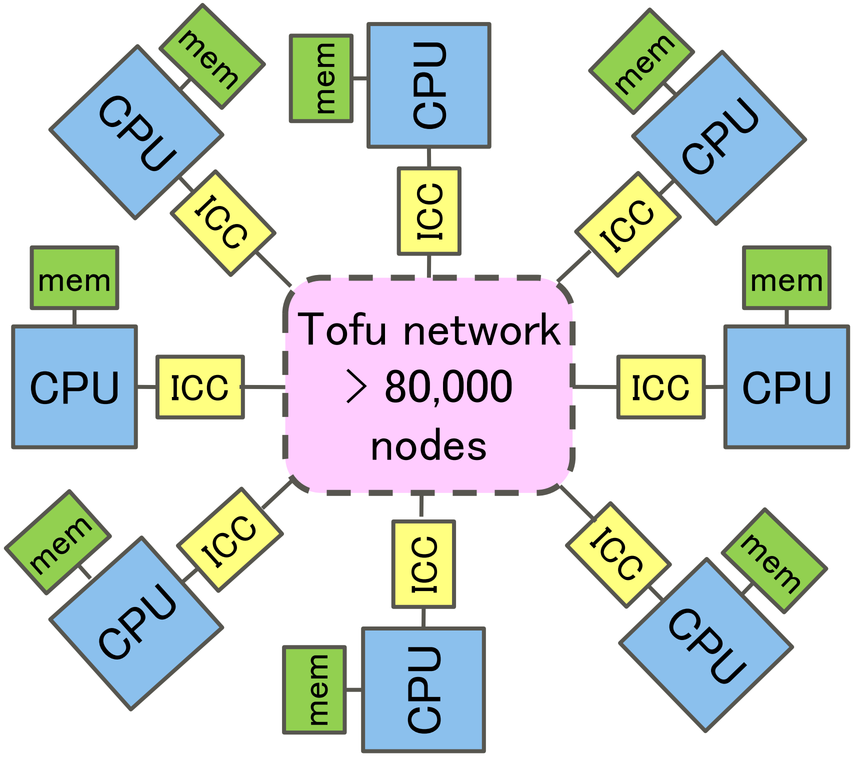
- Decrease the numerator
  - **Design for Verification**
- Increase the denominator
  - **Increasing Verification Efficiency**
  - **Avoiding Waiting Time**

Design for Verification

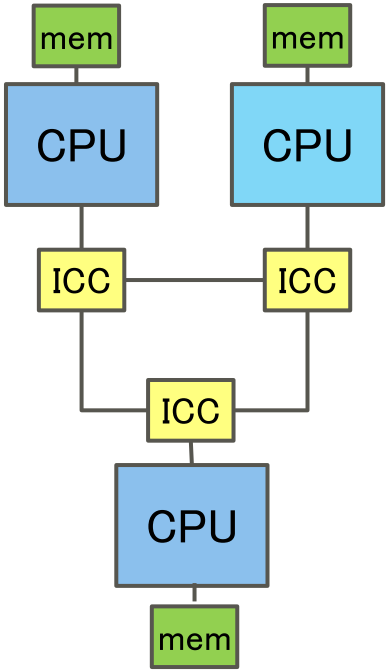
Verification items are increasing exponentially

- Challenge
  - Reduce the number of necessary items as much as possible
- Solution
  - Reusing Common modules
    - Maximize reuse of common modules to avoid duplicate verification
  - Self-Contained Block Design
    - Increase the independency between blocks
  - Verification Support Functions
    - Check the whole behavior of 80,000 nodes in K computer within a small framework
    - Insertion of verification-purpose-only function blocks

Verification Support Function



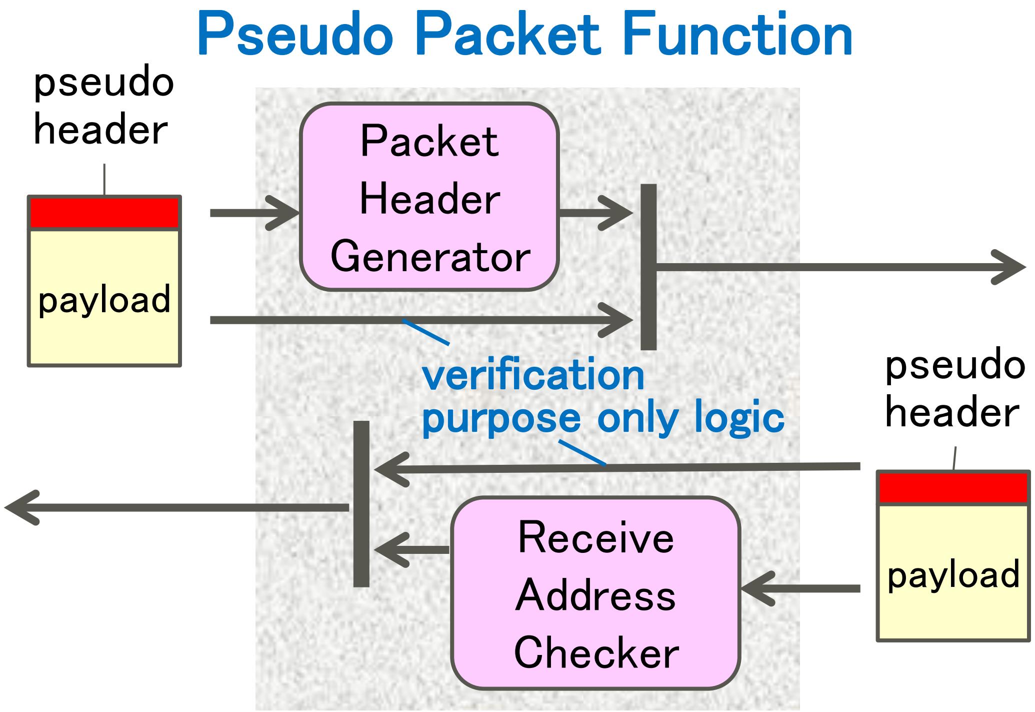
Real System

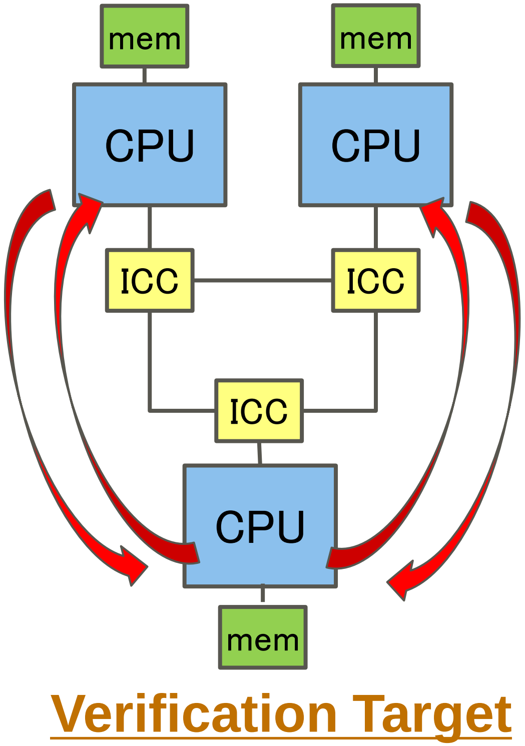


Verification Target

- Verification target is limited by resource constraints
- Reproducing communication among tens of thousands of nodes in a limited framework is a critical issue.

Verification Support Function





- Functions to imitate communication between others
  - Sending packet image including pseudo header
  - Accept every packet by bypassing receive address check
- Reproducing large scale communication in a small framework

Increasing Verification Efficiency

Exceptional functions are rarely used
e.g. Retry, Buffer Full and Error Correction etc

Challenge

- Critical timings should be verified exhaustively
- However these are :
  - hard to reproduce using directed test approach
  - consuming a lot of time and effort

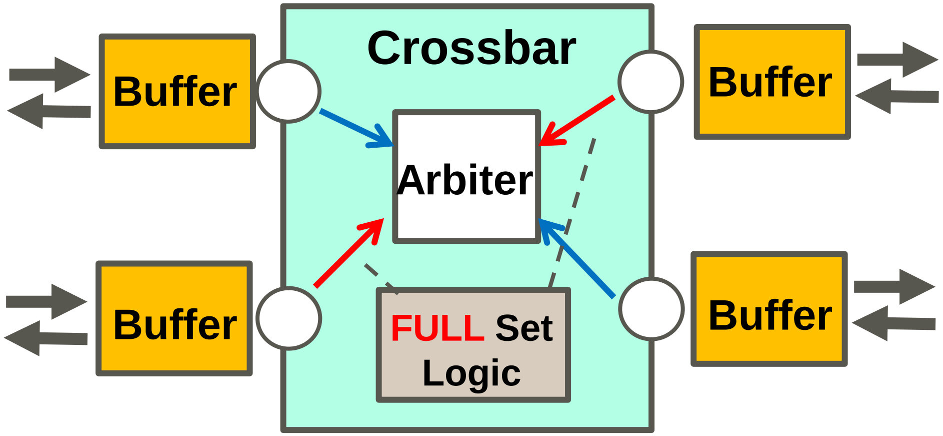
Solution

- Hardware support to accelerate these functions
- Forcibly create exceptional situations

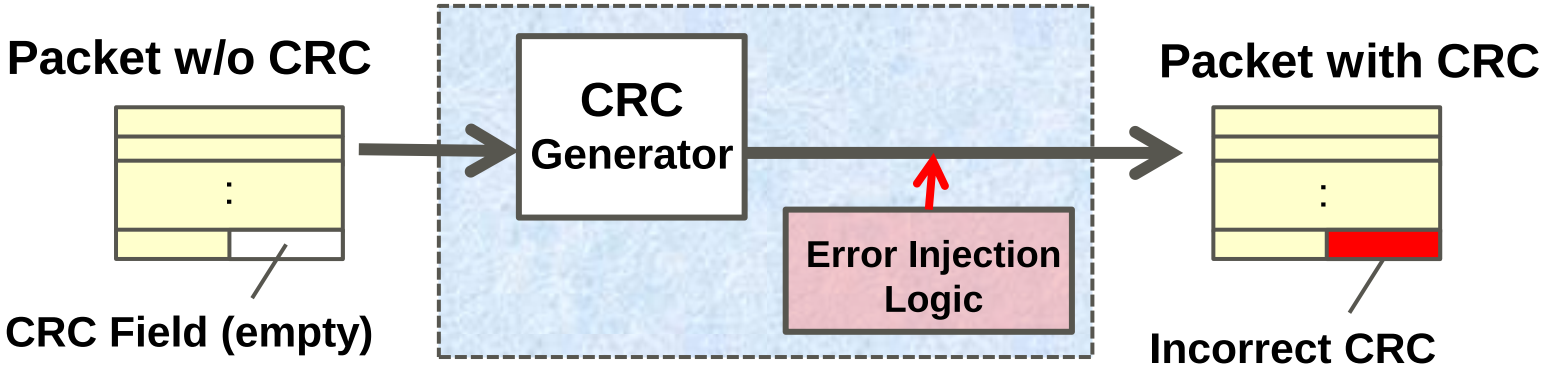
Acceleration and Error Injection

- Forcibly Setting **FULL** Signal
  - Network Congestion is generated by setting **FULL**
  - Burst Traffic is generated by releasing **FULL**

⇨ Various timings and sequences can be reproduced



- Error Injection
  - Forcibly insert a hardware error to check error recovery functions



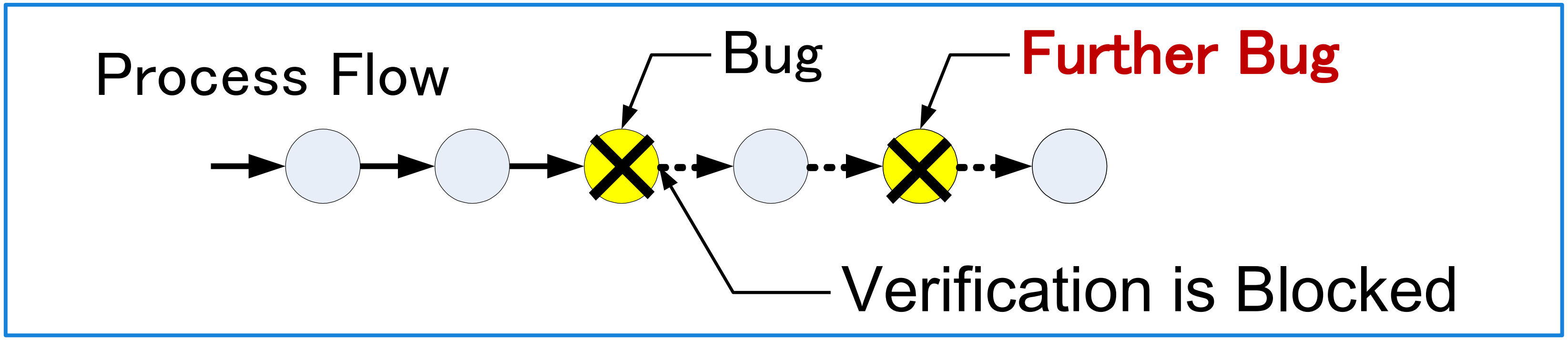
Avoiding Waiting Time

FUJITSU

- Average Bug Fix Time

  System Level Emulation: A Few Days

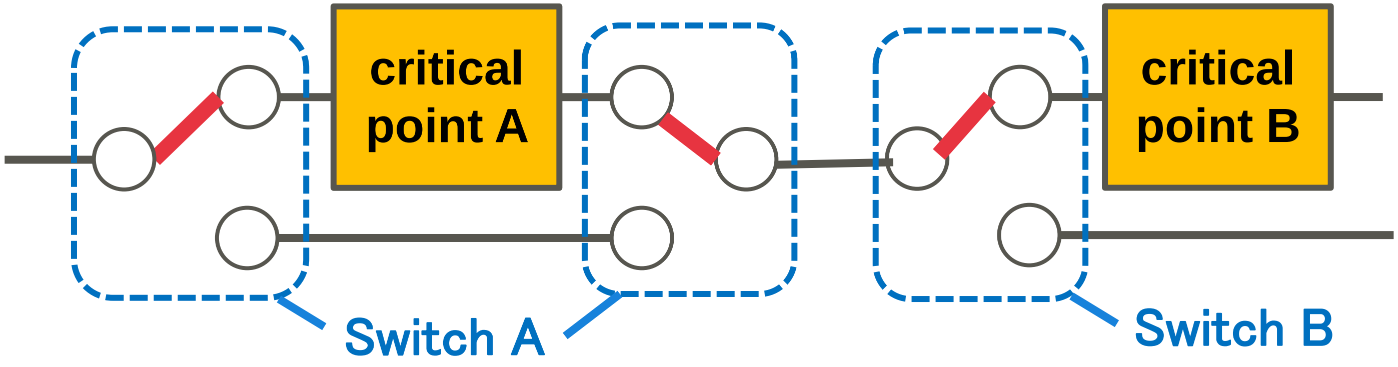
- Challenge
  - Verification is blocked by bug
  - How to detect further bug before the bug fix?



- Solution
  - Alternative hardware functions to bypass the bug

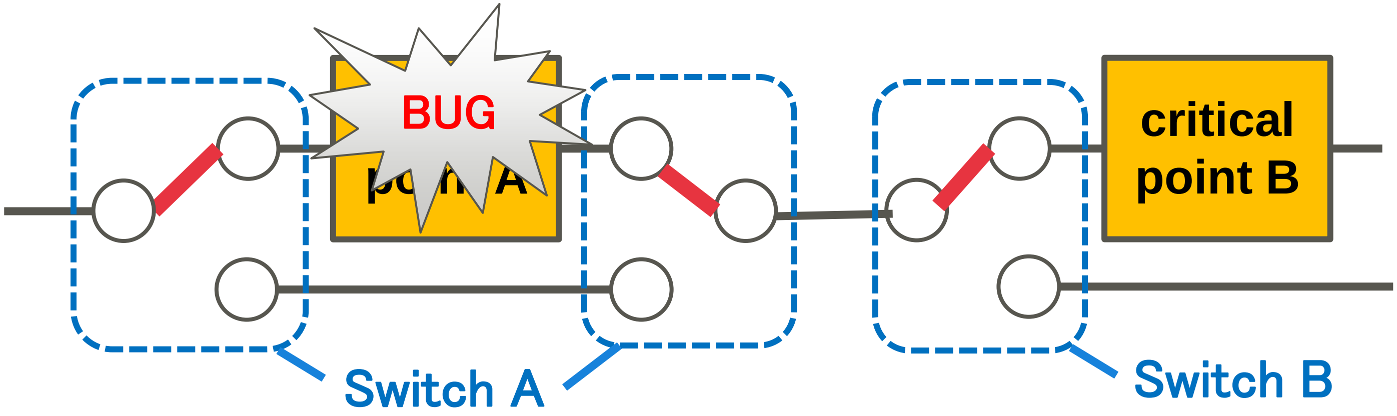
Bypass Logic

- From their experience, designers choose critical points
- They build bypass logics into these points together with switches to control them
- Examples
  - Out-of-order processing ⇒ In-order
  - Limiting concurrency (number of outstanding requests)



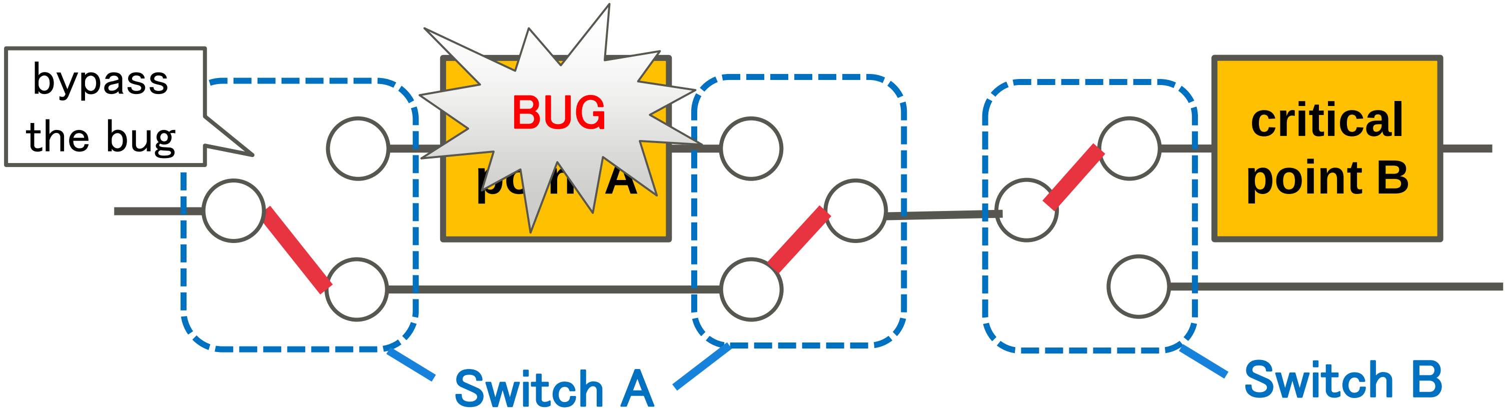
Bypass Logic

- From their experience, designers choose critical points
- They build bypass logics into these points together with switches to control them
- Examples
  - Out-of-order processing ⇒ In-order
  - Limiting concurrency (number of outstanding requests)



Bypass Logic

- From their experience, designers choose critical points
- They build bypass logics into these points together with switches to control them
- Examples
  - Out-of-order processing ⇒ In-order
  - Limiting concurrency (number of outstanding requests)



- Hunting further bugs by bypassing previous bug before fix
- ICC has 2000 bypass logic switches

  ⇒ Many bugs and defects can be successfully avoided by them

Summary

- Verification is the most time consuming process
  - Especially for chips in large computer systems
  - Complicated logic and a lot of exceptional functions
- Design and verification techniques for chips in the K computer
  - Design for Verification
  - Increasing Verification Efficiency
  - Avoiding Waiting Time
- These would be also useful in other large-scale SoCs

FUJITSU

shaping tomorrow with you