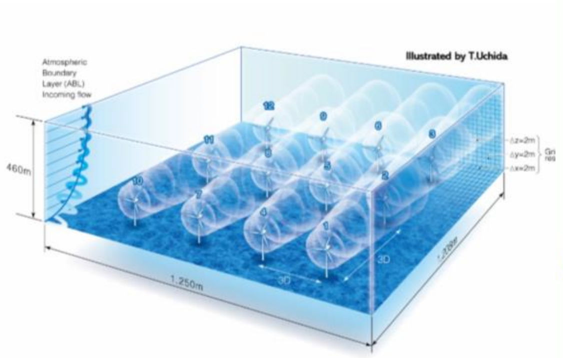
How do we debug and verify errors on 100,000-node systems?

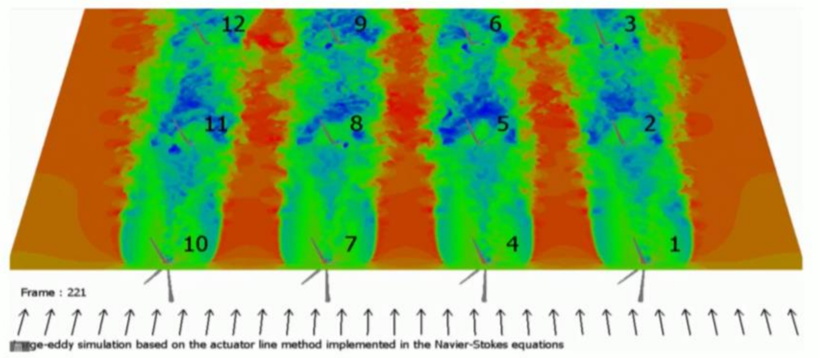
Insights through our supercomputer system projects

Fujitsu Ltd.
Takahide Yoshikawa

What is the Supercomputer System?

- Supercomputer can process huge amounts of calculations by
  - Designing dedicated acceleration modules
  - Connecting large amounts of nodes
- Applications of the systems = Computer Simulations



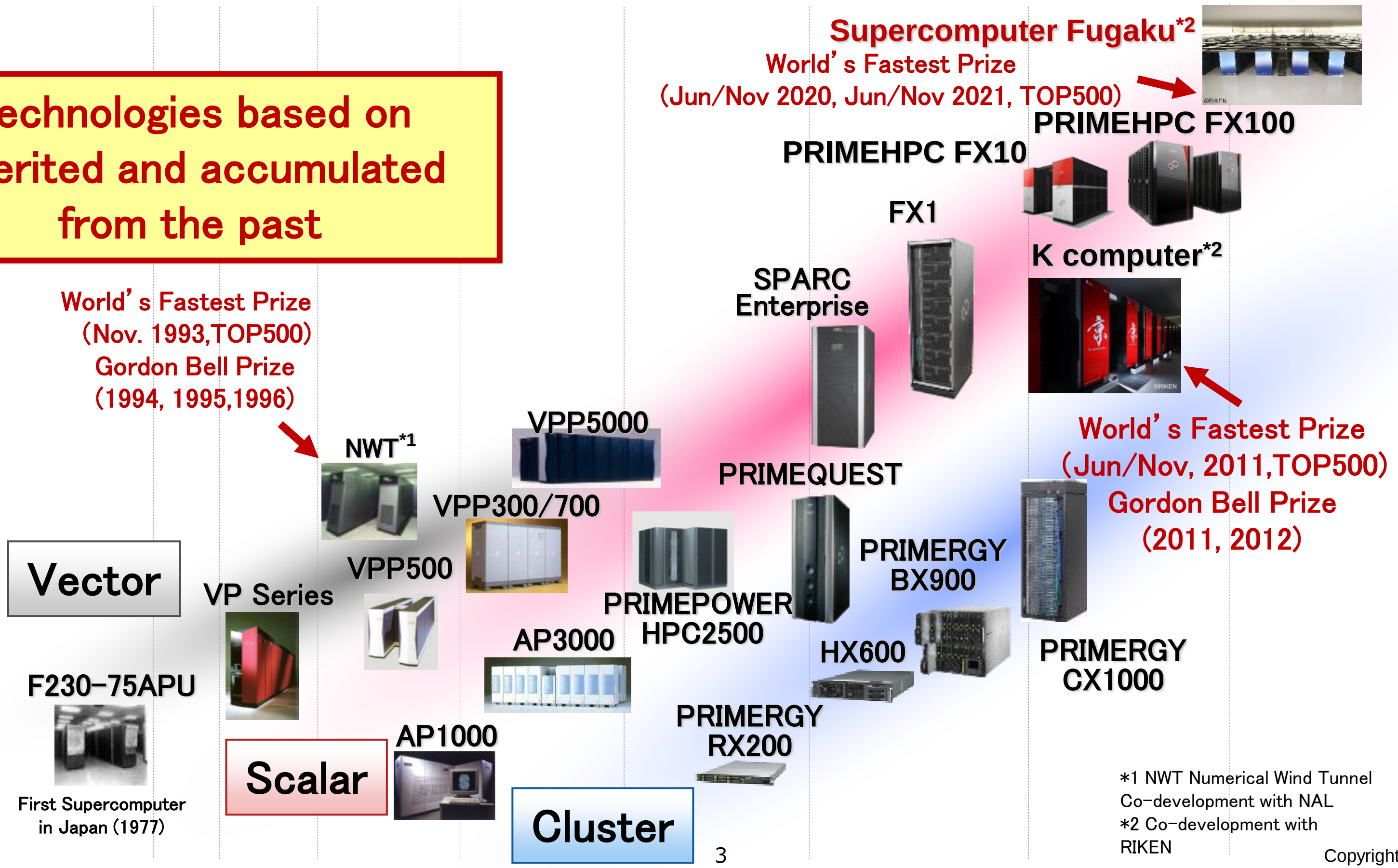


Frame : 221
Large-eddy simulation based on the actuator line method implemented in the Navier-Stokes equations

Courtesy of Program for Promoting Researches on the Supercomputer Fugaku (hp220169) and RIKEN

History of our Supercomputer System

Technologies based on inherited and accumulated from the past



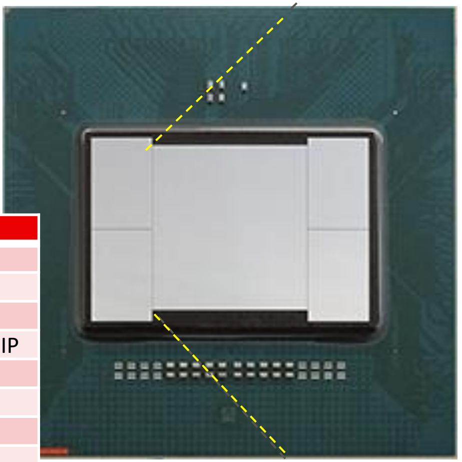
Supercomputer Fugaku*2
World’s Fastest Prize (Jun/Nov 2020, Jun/Nov 2021, TOP500)
PRIMEHPC FX100
PRIMEHPC FX10
FX1
K computer*2
SPARC Enterprise
World’s Fastest Prize (Jun/Nov, 2011,TOP500) Gordon Bell Prize (2011, 2012)
World’s Fastest Prize (Nov. 1993,TOP500) Gordon Bell Prize (1994, 1995,1996)
NWT*1
VPP5000
PRIMEQUEST
VPP300/700
PRIMERGY BX900
VPP500
Vector
VP Series
PRIMEPOWER HPC2500
AP3000
HX600
PRIMERGY CX1000
F230-75APU
PRIMERGY RX200
AP1000
Scalar
Cluster
First Supercomputer in Japan (1977)

*1 NWT Numerical Wind Tunnel Co-development with NAL
*2 Co-development with RIKEN

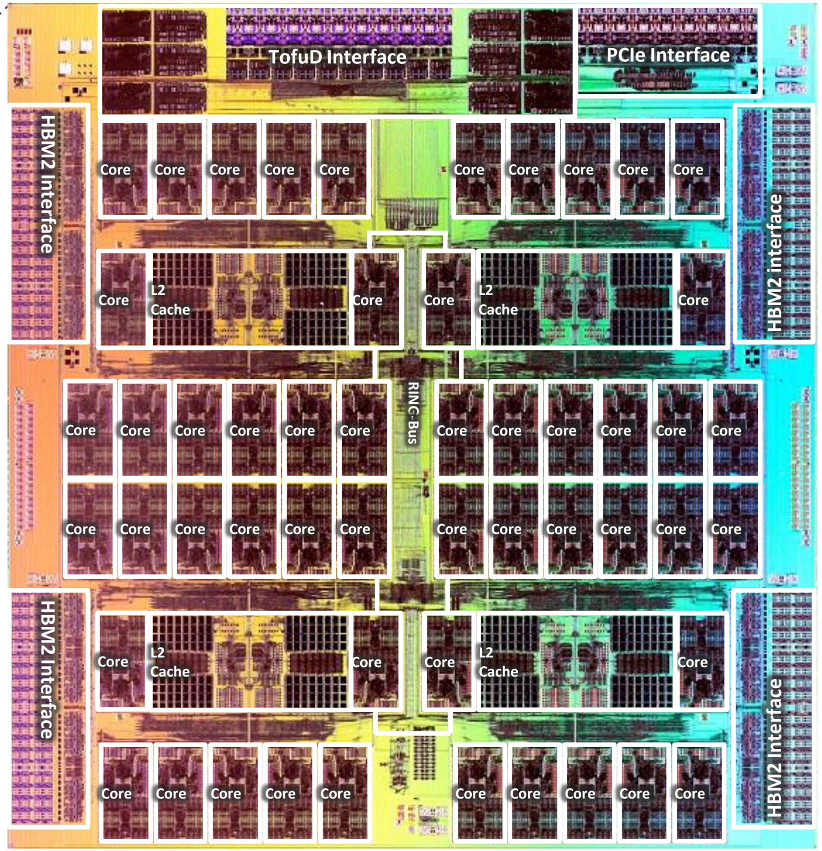
Agenda of This Presentation

A64FX CPU

- TSMC 7nm FinFET & CoWoS
  - Broadcom SerDes, HBM I/O, and SRAMs
  - 594 signal pins



CPU		SPARC64 VIIIfx	A64FX
System		K computer	Fugaku
Year		2010	2020
Package type		SCM	2.5D
Substrate		GC	Organic+SiIP
1st level	Method	C4	C4
	Material	Sn-Ag	Sn-Ag
	Melting point	221 ℃	221 ℃
2nd level	Method	BGA	BGA
	Material	Sn-Ag-Cu	Sn-Bi-Ag
	Melting point	220 ℃	138 ℃
Si-technology		45 nm	7 nm
Tr. number		760 M	8,786 M
Die size		511 mm^2	422 mm^2
PKG size		55x55 mm	60x60 mm
BGA pins		2,408	2,728

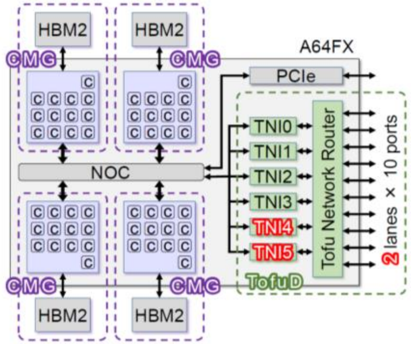


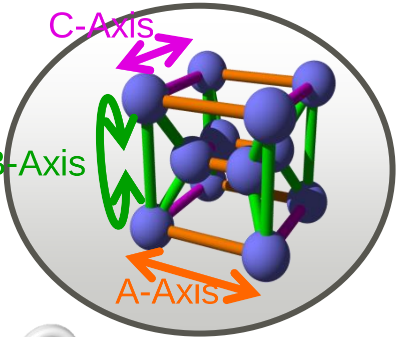
Tofu Interconnect Controller Chip (ICC)

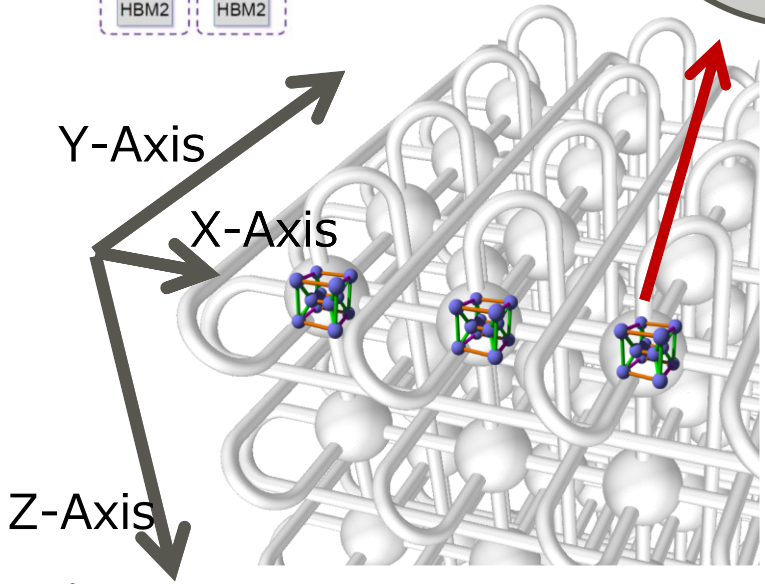
- SPECS of TofuD

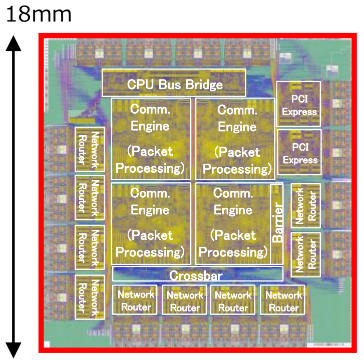
Clock Speed	425MHz
Link Bandwidth	6.8GB/s

- Features
  - 6 Packet Processing Engines
  - 28Gbps x 2lanes x 10ports
  - Tofu 6-dimensional Connections

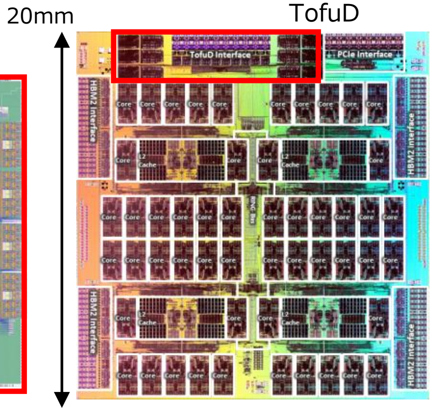






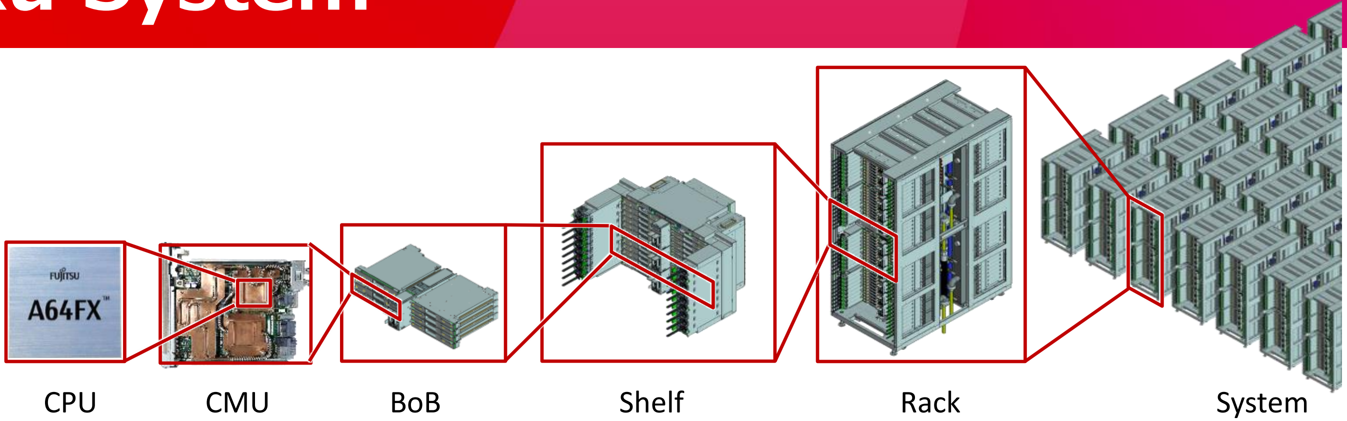


Tofu1 (45nm) for K-Computer in 2012



A64FX (7nm) for Fugaku in 2020

Fugaku System



Unit	# of nodes	Description
CPU	1	Single socket node with HBM2 & Tofu Interconnect D
CMU	2	CPU Memory Unit: 2x CPU
BoB	16	Bunch of Blades: 8x CMU
Shelf	48	3x BoB
Rack	384	8x Shelf
System	158,976	432Racks×384CPU＝165,888≠158,976 396Racks are Full, 36Racks are Half(192CPUs)

Agenda of This Presentation

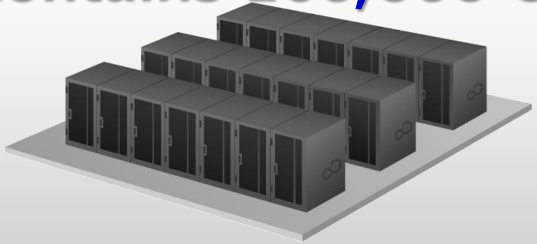
Reliability and availability

If a CPU would be broken once in 100 years,

PC(contains 1 CPU)

System was broken
once in 100 years

Super Computer
(contains 160,000 CPUs)



System was broken
once in 5 hours

In order to achieve a high-availability system,
each component should have superior reliability.

Availability of Fugaku and K-Computer

FUJITSU

- **Execution time of TOP500 (June 2024) benchmark.**
  The whole system can run hours. It means that each
  CPU be broken once in 147 - 296 years.

	Computer	Performance	Exec. Time
		PFlops	Hour
1	Frontier	1206	2.21
2	Aurora	1012	4.36
3	Eagle	561	0.54
4	Fugaku	442	4.04
	K Computer	10.51	29.47

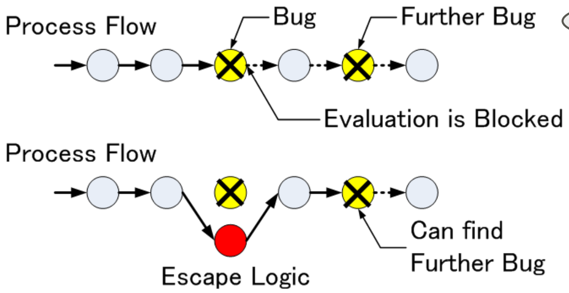
# of CPUs	Component Reliability
158,976	147 Years
88,128	296 Years

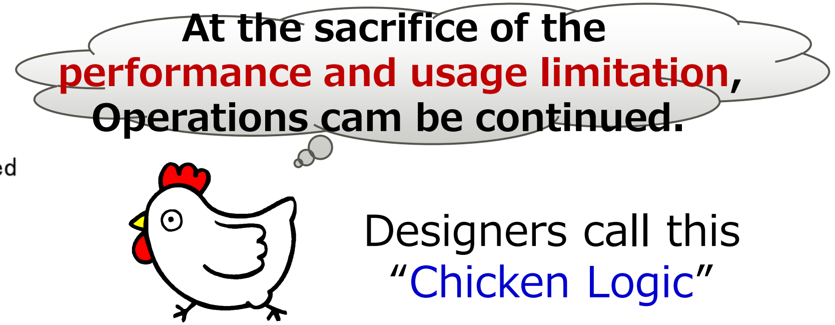
Actions to Assure the Reliability

Identifying failures and bugs in the **smallest configurations**.
Also, even if there is, system makes it **avoidable**.

Action 1: Taking a larger margin (Frequency, Voltage etc.)

Action 2: Implement logics to avoid suspected modules



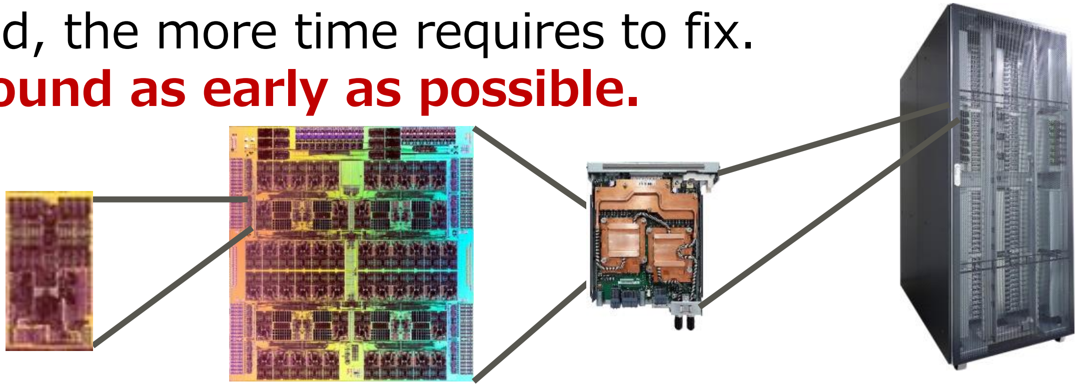

At the sacrifice of the
performance and usage limitation,
Operations cam be continued.

Designers call this
"Chicken Logic"

Action 3: Finding bugs as early as possible
→ Dedicated verification and test logics

Verification and Validation of Fugaku

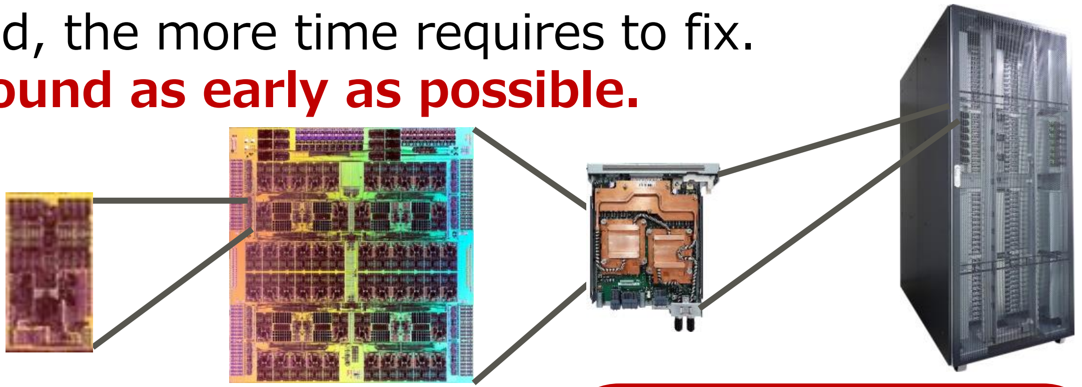
- Verifying huge system by gradually building up from small parts
- The later bug is found, the more time requires to fix.
  →**Bugs should be found as early as possible.**



	Component Verification	Integrated Verification	System Verification	Post-Silicon Validation
Target Scale	Single Module ～100kGates	Multiple Modules 1MG～100MG	1～11nodes 100MG～	>3nodes
Target Spec.	Implementation	I/F	System	System
Verification Platform/Tool	Simulator Formal Tool	Simulator Emulator(Accelerator)	Emulator	Prototype System
Times to Fix Bug	Hours	Days	Weeks	Months

Verification and Validation of Fugaku

- Verifying huge system by gradually building up from small parts
- The later bug is found, the more time requires to fix.
  →**Bugs should be found as early as possible.**



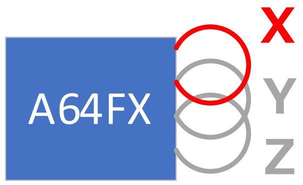
	Component Verification	Integrated Verification	System Verification	Post-Silicon Validation
Target Scale	Single Module ～100kGates	Multiple Modules 1MG～100MG	1～11nodes 100MG～	>3nodes
Target Spec.	Implementation	I/F	System	System
Verification Platform/Tool	Simulator Formal Tool	Simulator Emulator(Accelerator)	Emulator	Prototype System
Times to Fix Bug	Hours	Days	Weeks	Months

Agenda of This Presentation

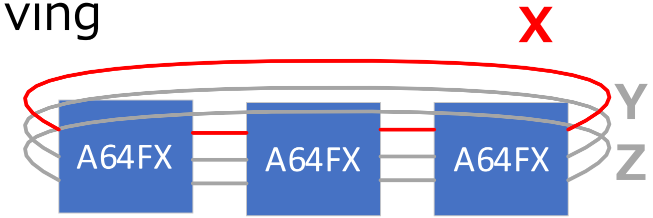
Configurations for Pre-Silicon Verification

The A64FX CPU's interconnect has 10 ports. The following three configurations were set to verify the communication functions.

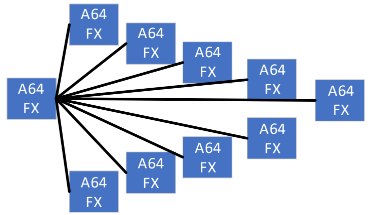
- **1 node loopback for functional verification**
  Each port has loopback logic.
  This can cover most functional verification scenarios.



- **3 nodes torus for high load verification**
  If there is a difference between sending and receiving capacity, one of them cannot make enough load.
  Thus, verification is conducted at 2 : 1.

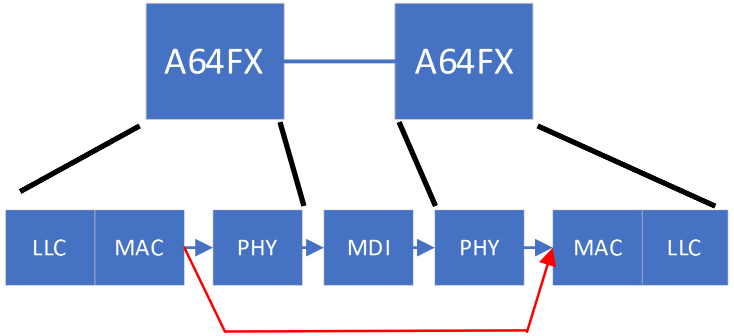


- **11 nodes star connection**
  This configuration is too time-consuming to set up.
  We did not use this setting during Fugaku development.

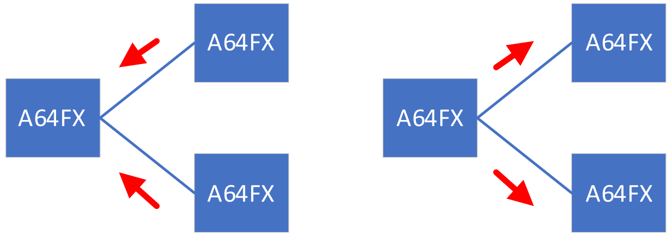


Performance Verification: Throughput

- **Base Performance**
  - These are **measured between 2 nodes**.
  - Before the prototype was available,
    we assumed that there were no bottlenecks
    in the analog circuit and network cables.
  - So, the logics are directly connected.



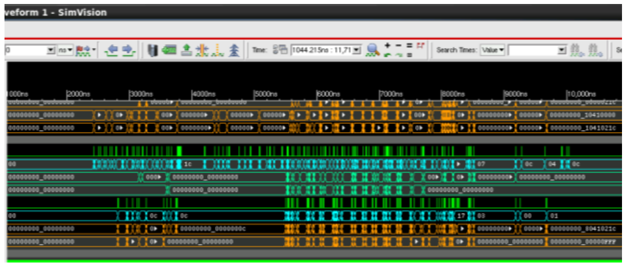
- **Send/Receive Capability Chec**
  - These are **measured among 3 nodes**.
  - Specifically,
    2 nodes sending to 1 node
    1 node sending to 2 nodes
    to capture the difference in sending and receiving performance.



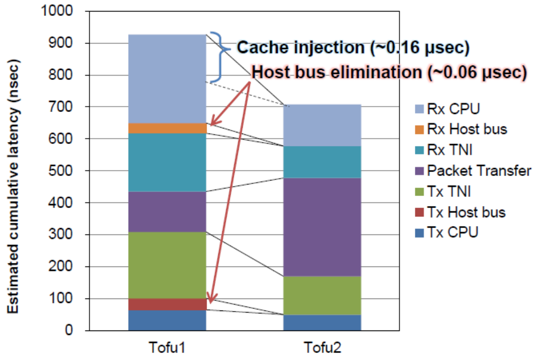
Performance Verification: Latency

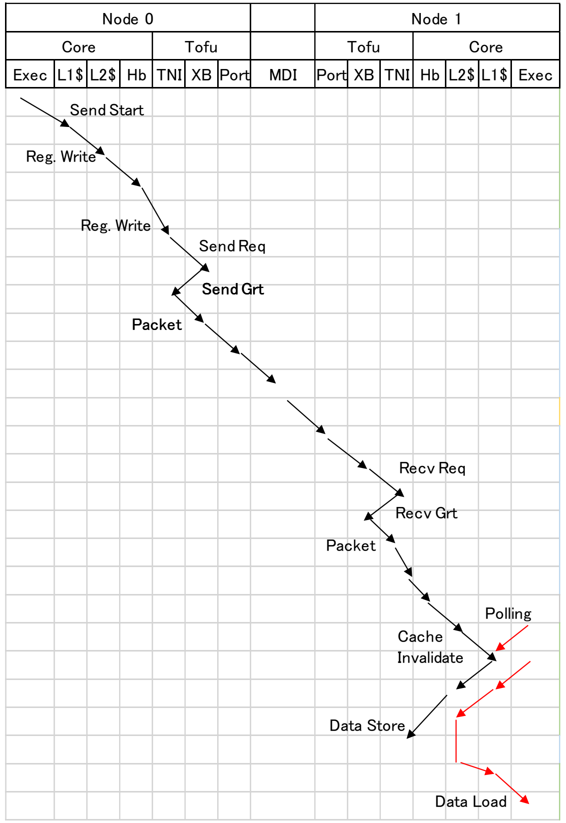
- Latency is measured between 2 nodes.
- The latency of analog modules (PHY and MDI) is replaced with fixed values obtained from the specifications.
- How we measure the latency:
  - **Identifying and agreeing on the I/F signals** between modules.

    If a clear separation is not made, misunderstandings will occur
    and the accumulated latency will not match with the end-to-end latency.
  - **Getting target latency values** from each module (L1$, TNI etc.).
  - **Tracing a single packet** in the waveform



Courtesy of Cadence (SimVision™)





41% of verification work = debug

Workflow of the Debug:

① **Check the verification env. again, again and again…**

② **Narrow down module**

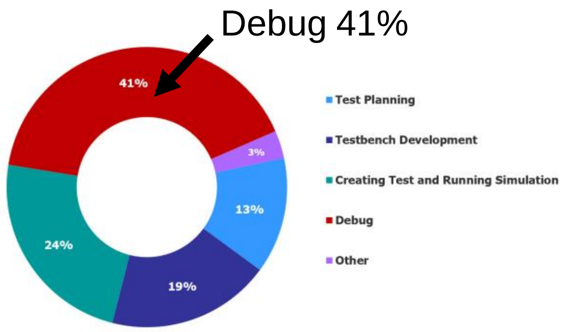
Designers typically cover their own modules.
(L1$, ALU, Memory Controller, etc.)
Unless the name of the designer is not identified,
no one will take the issue.

→ The responsibility of the verification engineer is
not simply to detect the issue, but to narrow down the suspected module.

③ **Designer analyze the trace and fix**

④ **Regression**

Once the bug is fixed, the same verification scenarios are run to confirm the fix.
Moreover, **related scenarios are run to check the fix does not affect other logic**.



Courtesy of Wilson Research Group and Siemens EDA

How to debug the hardware (Function)

Identifying suspected modules by comparing spec. and signal waves.

Verification engineers know the typical behavior of every single module.

- **Component/Integrated Verification**

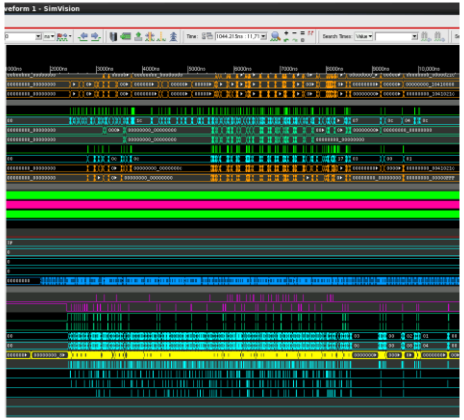
  All the sequences after reset can be observed.
  Therefore, the cause of the bug can always be found.
  (Root cause analysis by tools is also possible.)

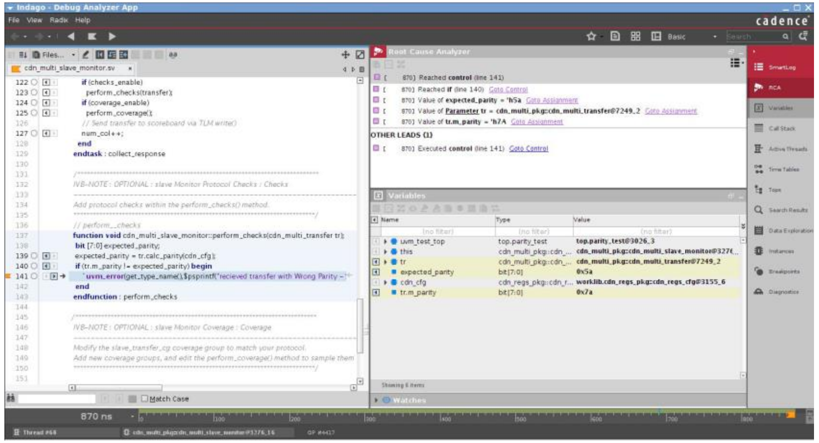
- **System Verification**

  On the other hand, if one takes the waveform of the entire CPU, it becomes **4GB/100kCycles**.

  Typical system verification scenarios run more than **1 billion cycles**, so it is impossible to get all the sequence.

  → **Waveform of a short time period and limited signals are collected many times to debug.**





```
122    if (checks_enable)
123      perform_checks(transfer);
124    if (coverage_enable)
125      perform_coverage();
126    // Send transfer to scoreboard via TLM write()
127    num_col++;
128   end
129  endtask : collect_response
130
131  /*******************************************************************
132  IVB-NOTE : OPTIONAL : slave Monitor Protocol Checks : Checks
133  -------------------------------------------------------------------
134  Add protocol checks within the perform_checks() method.
135  *******************************************************************/
136  // perform__checks
137  function void cdn_multi_slave_monitor::perform_checks(cdn_multi_transfer tr);
138   bit [7:0] expected_parity;
139   expected_parity = tr.calc_parity(cdn_cfg);
140   if (tr.m_parity != expected_parity) begin
141     `uvm_error(get_type_name(),$psprintf("recieved transfer with Wrong Parity -
142   end
143  endfunction : perform_checks
144
145  /*******************************************************************
146  IVB-NOTE : OPTIONAL : slave Monitor Coverage : Coverage
147  -------------------------------------------------------------------
148  Modify the slave_transfer_cg coverage group to match your protocol.
149  Add new coverage groups, and edit the perform_coverage() method to sample them
150  *******************************************************************/
151
```

Root Cause Analyzer

```
[     870] Reached control (line 141)
[     870] Reached if (line 140)  Goto Control
[     870] Value of expected_parity = 'h5a  Goto Assignment
[     870] Value of Parameter tr = cdn_multi_pkg::cdn_multi_transfer@7249_2  Goto Assignment
[     870] Value of tr.m_parity = 'h7A  Goto Assignment
OTHER LEADS (1)
[     870] Executed control (line 141)  Goto Control
```

Variables

Name	Type	Value
uvm_test_top	top.parity_test	top.parity_test@3026_3
this	cdn_multi_pkg::cdn_...	cdn_multi_pkg::cdn_multi_slave_monitor@327(...
tr	cdn_multi_pkg::cdn_...	cdn_multi_pkg::cdn_multi_transfer@7249_2
expected_parity	bit[7:0]	0x5a
cdn_cfg	cdn_regs_pkg::cdn_r...	worklib.cdn_regs_pkg::cdn_regs_cfg@3155_6
tr.m_parity	bit[7:0]	0x7a

Courtesy of Cadence (SimVision™ Debug and Indago™ Debug Analyzer)

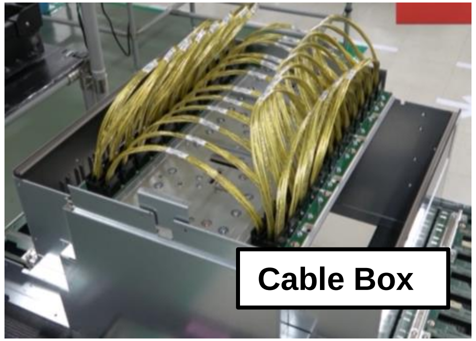
Agenda of This Presentation

1. Overview of Fugaku Supercomputer System
2. Reliability of Supercomputer Systems
3. Pre-Silicon Verification and Debug (Function and Performance)
4. **Post-Silicon Validation and Debug**
5. Manufacturing Test
6. Summary

Configuration for Post-Silicon Validation

At the Post-Silicon, the wiring inside the chassis (Cable Box) is fixed.

- **For 1 node loopback** for functional validation
  The internal test logic of the PHY module is used.
- **For 3 node torus** for high load validation
  A specially modified Cable Box is applied. Also, OS and firmware are revised for the validation environment.
- **For inter-node transmission** test
  The production cable box should also be tested.
  Thus, only the cable routing is changed for dedicated configuration.



Dedicated Equipment: Handling cables top to bottom, right to left.

Post-Silicon Validation Scenario

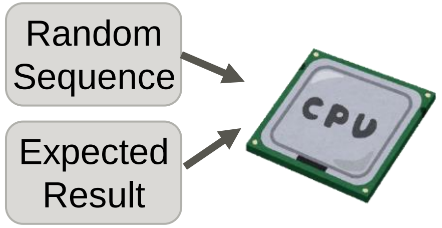
Building 1 node to 1 rack(384 nodes) system and confirming the function, power, and performance.

- Requirements to validation scenario
  - **Hand-written scenarios end up running in a few minutes.**
  - Longer scenarios should be automatically generated.

- Random Scenario Generator

①Offline (pre-flight)

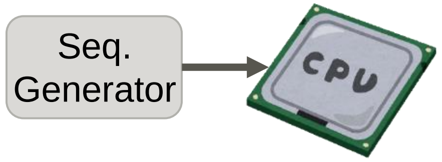
1. Generating sequences on server
2. Getting expected result by using ISS(*)
3. **Uploading sequences and expected result**
4. Running sequences and checking result



(*) ISS: Instruction Set Simulator

②Online (in-flight)

1. **Uploading random sequence generator**
2. Generating sequences on CPU under validation
   (90% of time is spent on generation.)
3. Running the same sequence twice
   (on different cores/configurations/addresses)
4. Comparing the results



© いらすとや

Debug on Post-Silicon Validation

1. Distinguishing between **Defective Hardware and Bug**
   by replacing CPUs, Memories, etc.

2. **Selecting signals** which may be connected to the failure, and
   capturing values through trace buffer.

3. Guessing the hardware behavior using the values.

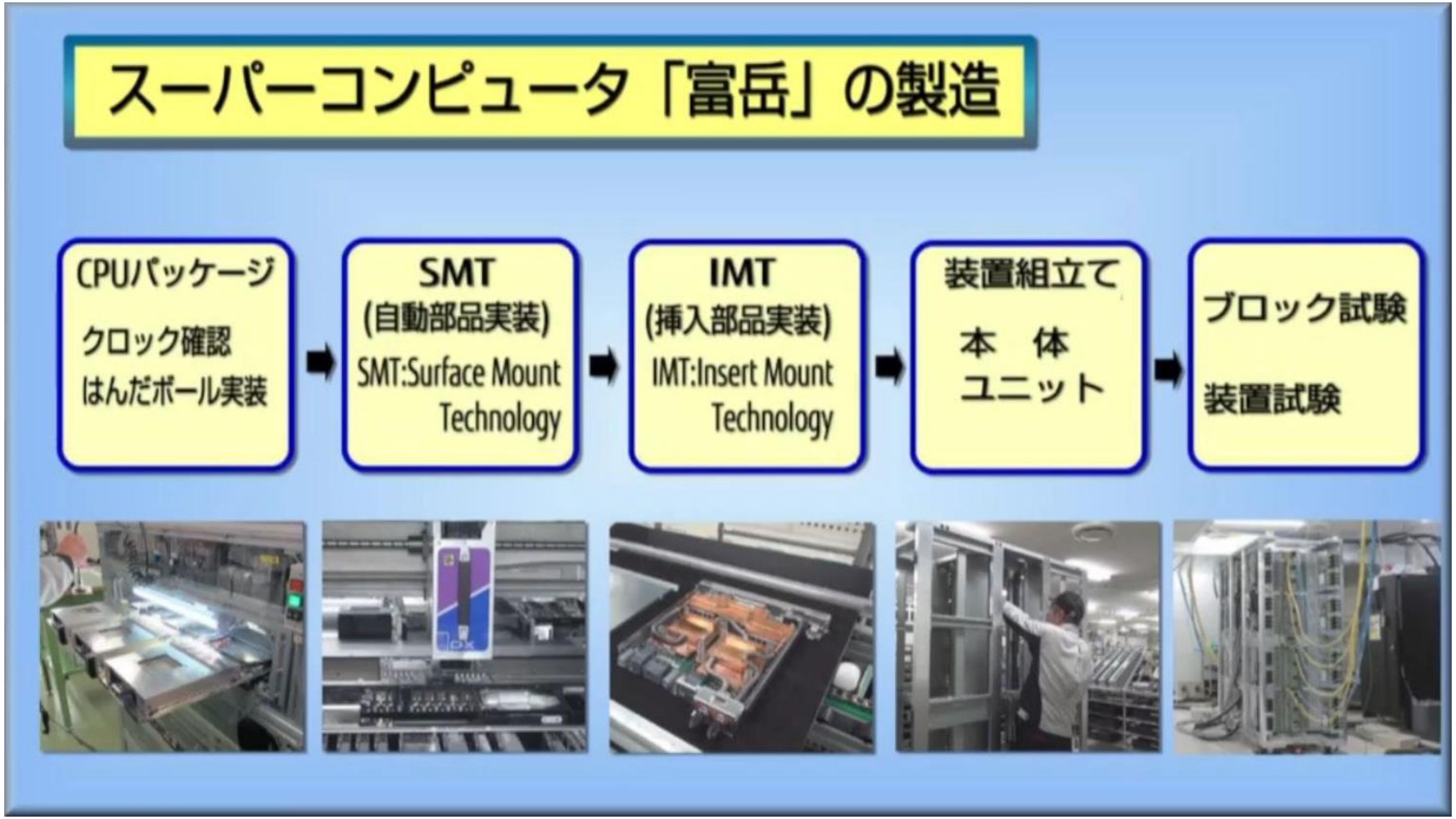
4. Trace Buffer is small (a few KB),
   values are captured over and over again
   varying signal set/timing/instruction sequences.

Is it possible to automate this? → NO!

- Hardware is operating as written in RTL.
- “Right Behavior” is described in a natural language (Spec. Sheet).
- To tell the difference between them is impossible so far.
  Incomplete automation makes debugging more difficult.

Agenda of This Presentation

Fugaku Assembly Line



Agenda of This Presentation

1. Overview of Fugaku Supercomputer System
2. Reliability of Supercomputer Systems
3. Pre-Silicon Verification and Debug (Function and Performance)
4. Post-Silicon Validation and Debug
5. Manufacturing Test
6. **Summary**

Summary of this Presentation

In this presentation, I have introduced some part of our Quality Assurance Methodologies:

- Overview of Fugaku system
- Verification, Validation and Manufacturing Test

  Smaller Environment and **Shorter Time** techniques are the key
- Debug

  Introducing the reality that verification eng. Have to narrow down bugs.

These efforts ensure the reqiability and then
enable the stable operation of Fugaku, with 158,976 CPUs.

If you have any questions please get in touch:

Takahide Yoshikawa
yoshikawa.takah@fujitsu.com