K computer: Challenges making the Super Quality Interconnect

Takahide Yoshikawa
Design Innovation Laboratory
Fujitsu Laboratories Ltd.
4-1-1 Kamikodanaka, Nakahara-ku, Kawasaki, 211-8588
e-mail : yoshikawa.takah@jp.fujitsu.com

Abstract – The K computer, RIKEN and Fujitsu are now developing, has twice been awarded the title of the world's fastest computer by the TOP500 project. Throughout development, lots of bugs were detected, but these bugs were fixed before manufacturing. This was achieved by our advanced verification methodologies. Furthermore, the 88,128 nodes system can run 30 hours without any single fault. This was supported by our leading production test methodologies. This paper introduces these advanced verification and production methodologies.

I. Introduction

Fujitsu is now developing the K computer with RIKEN, in accordance with the High Performance Computing Infrastructure Initiative promoted by the Ministry of Education Culture Sports Science and Technology in Japan (MEXT). The K computer contains more than 80,000 SPARC64^{TM}VIIIfx [1] processors, designed and developed by Fujitsu. Each processor is connected by Tofu interconnect [2], an innovative, 6-dimensional mesh/torus network.

Figure 1 shows the system development schedule.

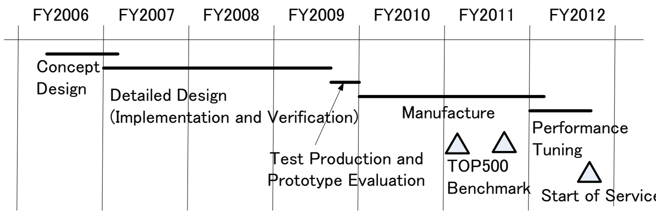


Fig. 1. System Development Schedule

This paper focused on the detailed design verification, the prototype evaluation and the manufacturing of the Tofu Interconnect Controller Chip (ICC).

The first developmental stage, concept design, began in 2006. Then, it proceeded into detailed design verification and prototype evaluation. The goal of the verification and evaluation was to fix all bugs before manufacturing. This was achieved by advanced verification strategy and management.

Next developmental stage was manufacturing. The goal of this stage was not to ship defective chips. This was supported by leading production test framework.

As a result, by May 2011, 68,544 nodes had been shipped and connected. This number had increased to 88,128 by October 2011. The 68,554 nodes system recorded 8.162 PetaFlops, and was confirmed as the world's fastest computer by the TOP500 project [4] in June 2011. In November 2011, 88,128 nodes system recorded 10.51 PetaFlops and was also awarded the first prize by the TOP500 project. Throughout the Linpack [3] benchmark measurement, both the 68,554 and 88,128 nodes system ran for 30 hours without any single fault.

Using our advanced verification methods, bugs were fixed before manufacturing. Due to our advanced production methods, the system can run for 30 hours without fault. This paper describes our leading verification and production methodologies.

II. Verification and Evaluation Planning

Figure 2 shows the development schedule of the ICC.

The preliminary concept design started in 2006. At the same time, the verification and evaluation also began by devising a strategy. The process included these five stages:

- Planning
  1. Devising verification and evaluation strategy
- Specification Verification
  2. Verifying design specification
- Simulation Verification
  3. Verifying core inside the ICC (Component Test)
  4. Verifying system (Integration/System Test)
- Evaluation
  5. Evaluating prototype

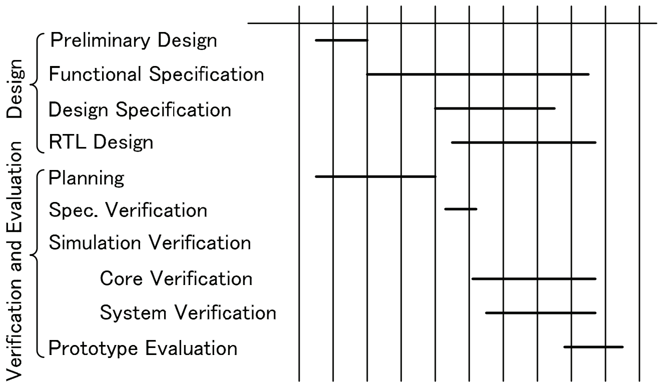


Fig. 2. Development Schedule of the ICC

The first verification stage was the most important. If the strategy was incomplete, any system bugs would not have been detected and would have remained in the products. Fujitsu Laboratories took charge of the strategy and management of ICC's verification.

The goal of the verification was to fix all bugs before manufacturing. At the beginning of the verification strategy stage, the following questions were evaluated carefully.

A. What kind of items to be verified are listed?
B. Which items are verified in which stage?
C. How to verify these items?

Note: The K computer is the nickname RIKEN has been using for the supercomputer of this project since July 2010.

D. How much resources are assigned?

For the verification, problems, such as complicated functions and lack of developing resources, prevent bug convergence. Therefore, these following questions were also assessed carefully.

E. What kind of problems might occur during verification?

F. How to avoid the problems and reduce the risk?

The actual verifications were divided into four stages, design specification verification, core verification, system verification, and prototype evaluation. In the verification strategy stage, each actual verification stage was broken down.

- Design Specification Verification

  Verification engineers checked the specification sheets one by one. Usually some important information is not written on the sheets. Such a precise detail of the specification was only known by designers themselves. Therefore, the verification engineers extracted detailed information from designers and put it down on paper.

  The goals of this stage were
  - Making the specification documents consistent and complete
  - Understanding the aim of the designers

In the next three simulation verification and evaluation stages, test scenarios were run on the design (Figure 3).

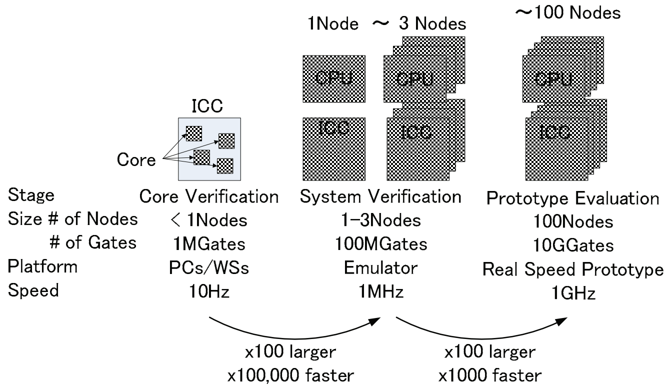


Fig. 3. Simulation Verification and Evaluation

- Core Verification

  ICC has many cores (Million-gates function modules). During this stage, cores were extracted and verified individually (Figure 4).

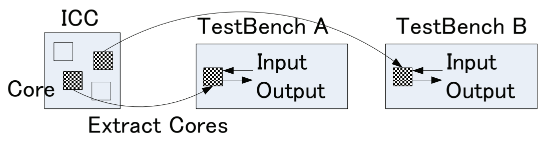


Fig. 4. Core Verification

The separated core was tested by the testbench, and the testbench generated the input signals and checked the output signals of the core. This sequence of the inputs and outputs set was called the test scenario.

Many bugs were detected and fixed in a short term during this stage. Therefore the verification environment should have a short turn-around time of simulations. For this reason, the core verification was run on conventional computers.

However, the conventional computer was not so fast. The verification speeds were limited to 10Hz, thus, test scenarios were also limited to the combination of minimum, maximum and a couple of selected values.

Each scenario was up to 1,000 clock cycles long, and 1,048,077 test scenarios were written manually and conducted.

The goals of this stage were:
- Verifying the results of each function by checking the output signals
- Varying the value of timing and parameters, including min, max and selected values

As a result, bugs converged to some degree, and then such a short turn-around time of simulations wasn't required any more. The design was able to proceed into the system verification stage.

- System Verification

  1 to 3 nodes (CPU and ICC) were connected and verified all at once. They were 100 times larger than the core verification target. Conventional computers could not handle such huge designs. Therefore, the emulator [6] was used in this stage. It could simulate the design 100,000 times faster than the conventional computers.

  The emulator was so fast that the number of clock cycles had to be 100,000 times longer than the core verification scenarios. To conduct many long test scenarios, test scenario generator was developed. It selected timings and parameter values randomly and joined them together.

  Each scenario was about 1,000,000,000 clock cycles long, and 3,256 test scenarios were generated.

  In order to check some specific scenarios (corner case), unconventional connections (Figure 5) were required (for example, a loopback connection, multipath and so on).

  In a normal situation, these unconventional connections are prohibited. They cause hardware errors and software errors. These unconventional connections were supported by specific verification support logics and test scenarios (for example, masking errors, bypassing some processes and generating special packets). Some of the logics were implemented into the testbench and the others were implemented into ICC itself.

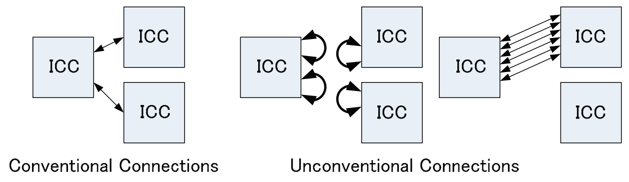


Fig. 5. Unconventional Connections

The goals of this stage were

- Checking cooperation of modules by connecting them to each other
- Covering larger numbers of selected combinations of timings and parameter values (100,000 times more than the core verification stage).

At the end of this stage, bugs had almost been converged. The design would not cause any serious problem in the prototype evaluation stage.

- Prototype Evaluation

The prototypes with up to a hundred nodes were manufactured and evaluated. They were 100 times larger and 1,000 times faster than the system verification environment.

In the prototype system, the number of nodes was strictly limited. To check some specific scenarios (corner case), unconventional connections were required, too. For these scenarios, the implemented verification support logics were used.

The goals of this stage were

- Covering lager numbers of selected combinations of timings and parameter values (1,000 times more than the system verification stage)
- Validating the verification by running many kinds of real user's applications (to look for insufficient test scenarios)

In the verification strategy stage, the assignment of the verification engineers is also very important. Each engineer has different skills and compatibilities. Assigning engineers to the correct modules is very important to the success of the verification.

Finally, many problems are expected during such a large-scale project. Foreseeable issues should be identified quickly and action taken as soon as possible. The next chapter describes how to avoid the problems.

III. Preparations for Possible Issues

Throughout development of the ICC, there were differences in difficulty and developing resources between modules. Some modules ran on schedule and achieved excellent quality, whereas others did not.

Verification engineers carefully analyzed the difficulty of the designs and estimated possible delays and number of bugs. To deal with these delays and bugs, the following methods were prepared.

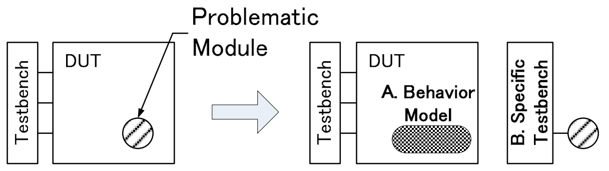
- Developing Behavior Model and Specific Testbench

Problematic modules not only affected its own verification progress but could also affect the neighbor module's verification progress.

To avoid disruption, a behavior model and specific test bench (Figure. 6) was developed for the problematic modules.

The problematic module was substituted with this behavior model so that neighbor modules could continue their verification processes.

Problematic modules usually required more detailed verification. Therefore, a specific testbench was developed to conduct this detailed verification.



A. Substitute the Problematic Module with behavior model.
B. Verify this module with the specific testbench.

Fig. 6. Behavior Model for problematic module

- Reassigning Verification Resources

Once a problematic module was detected, more verification resources were assigned to it. If the user interfaces of the verification testbenches were different between the modules, it would be hard to reassign the verification engineers. They might not be familiar with the other interfaces.

Thus, the testbench was based on the Open Verification Methodology (OVM)[5]. It standardized the verification environment. This meant both the testbench itself and the verification engineers became able to be reassigned.

Results of these two preparations,

- Problematic modules didn't affect neighbors
- Verification resources could be relocated into the problematic modules quickly

Due to these, delay and quality problems were resolved, and then, the prototype system started to work without any serious problems.

IV. Verification Result

Figure 7 shows a bug detection percentage of each simulation verification and prototype evaluation stage.

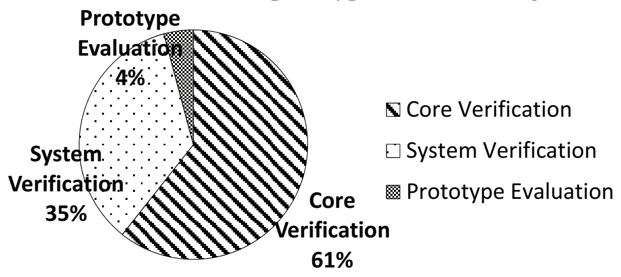


Fig. 7. Bug Detecting Percentage

There were 4% bugs detected in the prototype stage. These were undetected bugs in the simulation verification stages, core verification and system verification. These bugs can be grouped into three categories.

1. **Bugs occurred under extremely rare timing case**

Even in the prototype system, these bugs were hard to reproduce. It took more than several hours, sometimes several days. Such bugs were hard to detect in the simulation stages.

A Formal verification [7] tool was used in the core verification stage. However, it was unable to detect the aforementioned bugs. The reasons are as follows.

- Design was too large for the tool

- Signal constraints reduce the combination of timing but incorrect constraints put the bugs out of sight

Finally these bugs were reproduced on the simulation verification environment by using debug support hardware (packet capture buffer and so on), but usually it took about one month.

2. Bugs activated only under possible situations

It was quite hard to check every exceptional situation, such as unexpected operations under exceptional situation, successive errors and so on.

There was huge number of exceptional situations. Verification engineers selected as many exceptional cases as possible, but they could not cover all cases in the limited time period.

3. Bugs overlooked due to insufficient test scenarios

Many kinds of real user's applications were run on the prototype system. Design allowed variable usages, but test scenarios sometimes only chose part of them. This caused test coverage holes.

If bugs were detected in the prototype system, prototype evaluation was blocked. This meant that further bug detection was impossible. Every bug detected meant another respin.

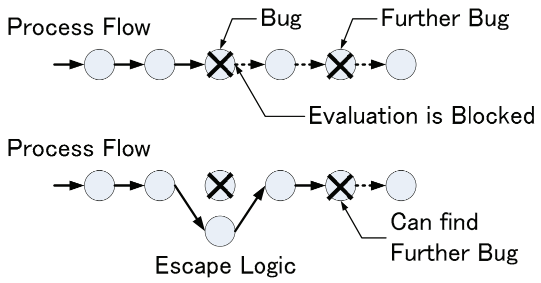


Fig. 8. Escape Logic

To prevent such series of respin, "Escape Logic" (Figure 8) was implemented inside the hardware. This logic bypassed the bug. It usually slowed the performance or limited the usage, but made it possible to continue the evaluation processes.

The 4% undetected bugs were successfully avoided by these escape logics. As a result, prototype evaluation was not blocked, and then all the bugs were fixed before manufacturing.

V. High Production Quality

Throughout the Linpack benchmark measurement, the 88,128 nodes system ran for 30 hours without any single fault. Fault means incorrect signal value or decision in a system. Fault is caused by manufacturing defect, and if the fault is actually exercised, it causes error [9].

To achieve such high production quality, a choice had to be made between the following two approaches.

1. Accepting faults
   Even though there is a defective piece of hardware and it causes a fault, user applications can continue to run using the rollback-recovery mechanism [10]. The faulty node is isolated automatically.
2. Not accepting any fault
   Never ship any defective or marginal chip. To realize this, the hardware is exposed to severe tests before shipping.

In a high performance computing system, users estimate work load carefully and tune programs to balance the load. Automatically isolating defective hardware would affect this balance. It might cause an unexpected performance slowdown.

Therefore, the second approach, not accepting any fault, was mainly chosen for the K computer, and the first approach was only taken to some degree in order not to affect the application's performance.

The shipping schedule was also tight. Thus the production tests had to check hardware very efficiently. A lot of test support hardware was implemented. Due to these hardware supports, most functions could be checked in a single node system with loopback configuration.

However, some defects around the SerDes [8] could not be covered in a single node system, such as broken wires or connectors, bad solder joints, unstable PLLs and so on. In order to detect these defects, neighbor nodes were required.

Therefore, all connections were checked in a large-scale system. To handle such a large system, a distributed autonomous test framework was developed. The test program independently ran on each node and automatically cooperated with the neighbor nodes.

Using these two measures, a lot of test support hardware and the distributed autonomous test framework, the K computer acquired such very high production quality.

VI. Summary

The K computer's interconnect chip architecture was a completely new. Despite numerous delays and problems, the development and shipping schedule have not been delayed.

Delays and design quality problems were resolved by using the appropriate verification strategy and management. Therefore, the prototype was started to work without any serious problem.

Some bugs were detected in the prototype evaluation stage. However, due to the "escape logic", the detected bugs were successfully avoided and the prototype evaluation was not blocked.

Due to the efficient test frameworks, defective and marginal chips were detected and then, the 88,128 nodes system ran for 30 hours without any single fault. This high production quality made a significant contribution to achieving the world's fastest computer award.

Fujitsu holds to such advanced verification and production technologies and methodologies. Fujitsu will continue to pursue a prosperous future for Earth and its peoples through the development of supercomputers.

Acknowledgements

I would like to express my gratitude to Satoshi Hiramoto (Fujitsu CIT LTD.), Takashi Tokue, Hitoshi Shimosaka, Satoru Nakamura (Fujitsu Semiconductor LTD.), Yoshihiro Nishida, Tatsuki Kobayashi, Yoshikazu Inagaki (Fujitsu Computer Technologies LTD.), Daisuke Ninomiya, Nobuyuki Matsui, Shigeki Itou, Satoshi Hattori (Fujitsu LTD.), Akitaka Sugiyama and Satoru Yamashita (FUJITSU IT Products LTD.). They also took part in and made a great contribution to the K computer's verification and mass production.

References

[1] Takumi Maruyama, Toshio Yoshida, Ryuji Kan, Iwao Yamazaki, Shuji Yamamura, Noriyuki Takahashi, Mikio Hondou, Hiroshi Okano, "Sparc64 VIIIfx: A New-Generation Octocore Processor for Petascale Computing," IEEE Micro, vol. 30, no. 2, pp. 30-40, Mar./Apr. 2010.

[2] Yuichiro Ajima, Yuzo Takagi, Tomohiro Inoue, Shinya Hiramoto, Toshiyuki Shimizu, "The Tofu interconnect," IEEE Micro, 21 Nov. 2011.

[3] Jack Dongarra, "The LINPACK Benchmark: An Explanation", Proceedings of the 1st International Conference on Supercomputing, p.456-474, March 1988.

[4] TOP500 Project, http://www.top500.org/project

[5] UVM/OVM Verification Methodology, http://verificationacademy.com/verification-methodology

[6] Cadence Palladium, http://www.cadence.com/products/sd/palladium_series/pages/default.aspx

[7] Edmund M. Clarke, Jr. , Orna Grumberg , Doron A. Peled, "Model checking", MIT Press, Cambridge, MA, 2000.

[8] Cameron Dryden, "Survey of Design and Process Failure Modes for High-Speed SerDes in Nanometer CMOS", Proceedings of the 23rd IEEE VLSI Test Symposium (VTS'05), p.285-291, May 01-05, 2005

[9] B. Parhami, "Defect, fault, error,..., or failure?" Reliability, IEEE Transactions on, vol. 46, no. 4, pp. 450-451, Dec 1997.

[10] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in messagepassing systems", ACM Computing Surveys, 34(3):375-408, Sep 2002.