SC20: International Conference for High Performance Computing, Networking, Storage and Analysis | 978-1-7281-9998-6/20/$31.00 ©2020 IEEE | DOI: 10.1109/SC41405.2020.00051

Co-Design for A64FX Manycore Processor and "Fugaku"

Mitsuhisa Sato*, Yutaka Ishikawa*, Hirofumi Tomita*, Yuetsu Kodama*, Tetsuya Odajima*,
Miwako Tsuji*, Hisashi Yashiro*, Masaki Aoki†, Naoyuki Shida†, Ikuo Miyoshi†,
Kouichi Hirai†, Atsushi Furuya†, Akira Asato†, Kuniki Morita†, Toshiyuki Shimizu†
* RIKEN Center for Computational Science
Email: {msato, yutaka.ishikawa, htomita, yuetsu.kodama, tetsuya.odajima, miwako.tsuji, h.yashiro}@riken.jp
† FUJITSU LIMITED, Japan
Email: {m-aoki, shidax, miyoshi.ikuo, k-hirai, furuya.atsushi, asato, morita.kuniki, t.shimizu}@fujitsu.com

***Abstract*—We have been carrying out the FLAGSHIP 2020 Project to develop the Japanese next-generation flagship supercomputer, the Post-K, recently named "Fugaku". We have designed an original manycore processor based on Armv8 instruction sets with the Scalable Vector Extension (SVE), an A64FX processor, as well as a system including interconnect and a storage subsystem with the industry partner, Fujitsu. The "co-design" of the system and applications is a key to making it power efficient and high performance. We determined many architectural parameters by reflecting an analysis of a set of target applications provided by applications teams. In this paper, we present the pragmatic practice of our co-design effort for "Fugaku". As a result, the system has been proven to be a very power-efficient system, and it is confirmed that the performance of some target applications using the whole system is more than 100 times the performance of the K computer.**

***Index Terms*—exascale computing, co-design, manycore processor, high-performance computing**

I. Introduction

Several projects aiming to develop exascale supercomputers are underway in the US, Europe, China, and Japan. A challenge is to develop a system capable of exascale computing and operable within an affordable power budget. The system should be designed to be dramatically energy-efficient while maintaining or improving performance, programmability, and reliability [1].

As a key strategy for this challenge, co-design, has received considerable attention in the HPC community for several years [2]. The term co-design has become popular in the development of mobile phone or embedded systems, where the two perspectives of hardware and software design are brought into a co-design process. In embedded systems, co-design sometimes includes "specialization" for particular applications. The co-design process of HPC must maximize the benefits to cover as many applications as possible. Co-design has been proposed as a methodology for the scientific application, software, and hardware communities to work together for designing future HPC systems.

In this paper, we describe the pragmatic practice of our co-design effort in our project. We have been carrying out the FLAGSHIP 2020 Project [3] to develop the Japanese flagship supercomputer system following the K computer since 2014. According to the outcomes of feasibility study projects prior to the FLAGSHIP 2020 Project, we have chosen a huge-scale system with general-purpose manycore processors. With Fujitsu as a vendor partner, a new manycore processor supporting the Arm instruction set has been designed through co-design with the collaboration of architecture teams, software teams, and application teams. During the co-design process, we focused on not only energy efficiency, but also programmability for ease-of-use and efficient execution of real scientific applications with some compatibility with the K computer.

This paper focuses on what we did in the co-design process and how we made several decisions on the system design, followed by the description of the "Fugaku" system as a result of this process. The next section describes an overview of the FLAGSHIP 2020 Project with our KPIs. Section III presents our methodology for co-design. Sections IV and V describe the co-design process for the processor and the storage system, respectively. Section VI presents the specifications of the system. The co-design process for the compiler and applications is described in Section VII. Section VIII concludes this paper with some retrospective comments.

II. FLAGSHIP 2020 Project

A. Project Overview

The FLAGSHIP [1] 2020 Project was launched in April 2014. The missions were defined as follows:

- Building the Japanese national flagship supercomputer, the successor to the K computer, which was tentatively named the "Post-K".
- Developing a wide range of HPC applications that will run on the Post-K in order to solve the pressing societal and scientific issues facing our country.

The RIKEN Center for Computational Science (R-CCS) is charged with the research and development of the Post-K. For

[1]FLAGSHIP: Acronym for Future LAtency core-based General-purpose Supercomputer with HIgh Productivity

SC20, November 9-19, 2020, Is Everywhere We Are
978-1-7281-9998-6/20/$31.00 ©2020 IEEE

TABLE I
TARGET APPLICATIONS FOR CO-DESIGN WITH TARGET PERFORMANCE AND CO-DESIGN POINTS

Applications	Brief Description	Computational Method	Target Perf. Relative to K	Codesign points										
				Structured Mesh	Unstructured Mesh	Particle Comp	Integer comp.	Dense Matrix (single node)	Dense Matrix (multi-nodes)	Comm (low latency)	Comm (high bandwidth)	Neighbor Comm.	Global Comm.	Large Data I/O
GENESIS[5]	Analysis for proteins	Molecular Dynamics	125			✔						✔	✔	
Genomon[6]	Genome processing	Large data processing for Genome alignment	8				✔							✔
GAMERA[7]	Earthquake simulation	FEM on unstructured & structured grid	45		✔					✔	✔		✔	
NICAM[8] +LETKF[9]	Weather prediction system with Data assimilation	FVM, stencil on using structured grid & ensemble Kalman filter	120	✔				✔			✔	✔		✔
NTChem[10]	Molecular electronic structure calculation	High-precision MO method, sparse and dense matrix computation.	40					✔	✔		✔	✔	✔	
FFB [12]	Large Eddy Simulation	FEM on unstructured grid	35		✔							✔	✔	
RSDFT [11]	An ab-initio simulation with DFT on real space	density functional theory, dense matrix computation	30	✔				✔			✔	✔	✔	
ADVENTURE[13]	Computational mechanics simulation	FEM on unstructured grid	25		✔			✔	✔	✔		✔	✔	
LQCD [14]	Lattice QCD simulation	structured grid Monte Carlo	25	✔						✔	✔	✔	✔	

TABLE II
NINE PRIORITY ISSUES WITH LEADING INSTITUTIONS

Health and longevity	
1. Innovative computing infrastructure for drug discovery	RIKEN Quantitative Biology Center
2. Personalized and preventive medicine using big data	Institute of Medical Science, the University of Tokyo
Disaster prevention / Environment	
3. Integrated simulation systems induced by earthquake and tsunami	Earthquake Research Institute, the University of Tokyo
4. Meteorological and global environmental prediction using big data	JAMSTEC, Center for Earth Information Science and Technology of Japan
Energy issues	
5. New technologies for energy creation, conversion / storage, and use	Institute for Molecular Science, National Institute of Neural Science
6. Accelerated development of innovative clean energy systems	School of Engineering, the University of Tokyo
Industrial competitiveness enhancement	
7. Creation of new functional devices and high-performance materials	The Institute of Solid State Physics, the University of Tokyo
8.Development of innovative design and production processes	Institute of Industrial Science, the University of Tokyo
Basic science	
9. Elucidation of the fundamental laws and evolution of the universe	Center for Computational Sciences, Tsukuba University

the system development, we have been working with Fujitsu since October 2014 after the vendor's selection phase.

Our government committee organized and selected nine Priority Issues and projects tackling these issues. The nine Priority Issues with the institutions selected to lead each project are listed in Table II.

Figure 1 shows the schedule of the project. The first two years (JFY2014 and JFY2015) [2] were spent on the basic design. From JFY2016 to JFY2018, detailed design and implementation were conducted. A prototype system was built in the summer of 2018 at Fujitsu. The review committee of the Japanese government reviewed the prototype system and

[2]The Japanese fiscal year (JFY) starts from April.

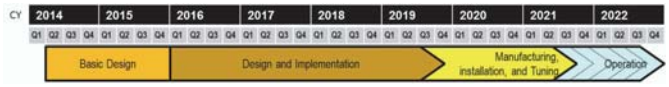


Fig. 1. Schedule of FLAGSHIP 2020 Project

decided to build the Post-K. From JFY2019 to JFY2020, the hardware is manufactured, and the system is installed and tuned.

B. Project KPIs

At the beginning of the project, we defined the following three Key Performance Indicators (KPIs):

1) **Power-efficient system:** We set the maximum electric power supply capacity of our facility between 30 and 40 MW. We should maximize the system performance within this capacity. We set more than 15 GFlops/W as the target efficiency for the dgemm kernel.
2) **Effective performance of real applications:** We focus on the performance of real applications rather than that of benchmarks. We aimed at a (maximum case) speed improvement of one hundred times over the K computer for some applications. This should be accomplished through co-design of system development and target applications for the nine Priority Issues.
3) **Ease-of-use:** Since the system will be shared by a variety of users, the system should be easy to use.

III. CO-DESIGN METHODOLOGY

A. Target Applications for Co-Design

Under the co-design concept, R-CCS, Fujitsu, and the nine selected projects have been collaborating closely to design the architecture, system software, and applications so that the societal and scientific issues can be solved effectively and early achievements can be attained. Since increasing power consumption is a critical issue in the design of the next-generation large-scale supercomputer, it is important to make

trade-offs between energy/power, cost, and performance by taking application characteristics into consideration.

In the co-design process, a set of target applications, as shown in Table I, was provided from each area of the nine Priority Issues. The reasons for choosing these applications as target applications are based on two criteria. One criterion is from the computational viewpoint, e.g., 1) the total target applications cover dense matrix operation and sparse matrix operation, and 2) they also cover the data access pattern (continuous, stripe, random). The other criterion is from the viewpoint of scientific excellence. We selected each application and its problem size, discussing with members of the project for the post-K priority issues. For example, the target application in the meteorological field is a very ambitious problem, which estimates global atmospheric conditions in very high resolution (3.5 km) using a global cloud resolving model and the data assimilation technique of the localized ensemble transformed Kalman filter based on over 1000 ensembles.

System design has been performed using these applications as a target. For each application, the target performance is set relative to the K computer when using the whole system with a power budget under approximately 30 to 40 MW. The target performance is estimated by co-design tools in the basic design phase.

The target applications are also considered representatives of almost all of our applications in terms of computational methods and communication patterns relevant to designing architectural features. The table shows the points of the co-design process for each application. These target applications are used to set the performance goals of the system and are also used to verify that these goals have been achieved.

Moreover, the typical benchmarks kernels such as dgemm, HPL, and the stream benchmark were used for performance analysis.

B. Tools for Co-Design

The following tools were used for the co-design process:

- **The performance estimation tool:** This tool, taking execution profile data of the latest Fujitsu supercomputer, FX100, as an input, enables the performance projection by a given set of architecture parameters. The performance projection is modeled according to the Fujitsu microarchitecture. This tool can also estimate the power consumption based on the architecture model. This tool has been available since the initial phase of the co-design process.
- **Fujitsu in-house processor simulator:** We used an extended FX100 (SPARC) simulator and compiler, developed by Fujitsu, for preliminary studies in the initial phase, and an Armv8+SVE simulator and compiler afterwards. It runs at a frequency of roughly 100 kHz.
- **Hardware emulator:** The hardware emulator for the processor was developed for logic design verification and accurate evaluation of performance and power consumption. It runs at a frequency of roughly 1 MHz and has been available since the later phase of the co-design process, after the circuit design was almost completed.
- **Gem5 simulator for the Post-K processor:** Gem5 [15] is an open-source system-level processor simulator. The Post-K processor simulator [16] based on gem5 has been developed by R-CCS during the co-design process for architecture verification and performance tuning of software.

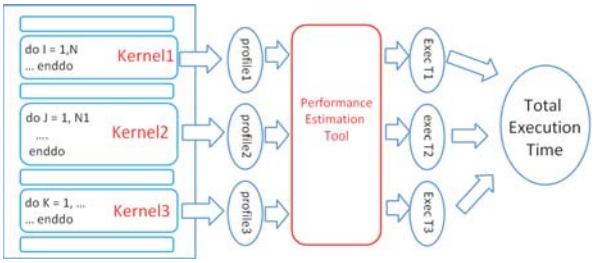


Fig. 2. Usage of the performance estimation tool

C. Co-Design Process: Design Space Exploration

As described in the next section, the basic architecture has been chosen as a manycore processor system by the feasibility studies before starting the project. Even for designing a manycore processor, there are many parameters and items to be determined in the design space exploration.

A fundamental problem is the scale of scientific applications that are expected to be run on the Post-K. Even our target applications are thousands of lines of code and are written to use complex algorithms and data structures. Although the processor simulators are capable of providing very accurate performance results at the cycle level, they are very slow and are limited to execution on a single processor without MPI communications between the nodes.

On other hand, our performance estimation tool is very useful since it enables performance analysis based on the execution profile taken from an actual run on the FX100 hardware. The FX100 hardware has a rich set of performance counters, including busy cycles for read/write memory access, busy cycles for L1/L2 cache access, busy cycles of floating-point arithmetic, and cycles for instruction commit. It enables the performance projection for a new set of hardware parameters by changing the busy cycles of functional blocks. The breakdown of the execution time (cycles) can be calculated by summing the busy cycles of each functional block in the pipeline according to the processor microarchitecture. For example, suppose the L2 cache access is a bottleneck. When doubling L2 cache bandwidth, the busy cycles of L2 cache access are halved and the execution time will be reduced if the time for L2 cache access is hidden by other cycles in the pipeline. Since the execution time is estimated by a simple formula modeling the pipeline, it can be applied to a region of uniform behavior such as a kernel loop. Otherwise, the accuracy of the estimation would worsen. Furthermore, it is hard to take the impact of the O3 (out-of-order) resources into account in the performance estimation.

The first step of performance analysis is to identify kernels in each target application and insert the library calls to get the execution profile. As shown in Figure 2, the total execution time is calculated by summing the estimated execution time of each kernel. We repeated this process changing several architecture parameters for design space exploration.

Some important kernels were extracted as independent programs. These kernels can be executed by the cycle-level processor simulators for more accurate analysis. This enables analysis for a new instruction set and the effect of changing the O3 resources. These kernels were also used for the processor emulator for logic-design verification.

We used an analytical model for simple kernels, such as the HPL and dgemm kernel, and the stream benchmark. This is sometimes useful because the execution time can be calculated by a simple mathematical formula using some hardware parameters.

In the co-design process for the interconnect, the communication patterns were extracted, and we estimated the communication performance by an analytical model, such as the LogP network performance model.

IV. Co-Design of Post-K

In this section, we describe how we made several decisions on the architecture of the Post-K through our co-design process. Most co-design for the architecture was done during the basic design phase from 2014 to 2015. Since the production of the system was scheduled for 2019, the co-design process needed to be preceded by assessing what kinds of technologies for silicon fabrication and memory, SerDes, and IO interface standards would be available at the time of the production.

A. Basic Architecture Design by Feasibility Studies

Prior to the FLAGSHIP 2020 Project, the following feasibility study projects were carried out in order to investigate the basic design from 2012 to 2013:

- System study for the Next-generation ”General-Purpose” Supercomputer, led by the University of Tokyo
- System study for Exascale Heterogeneous Systems with Accelerators, led by the University of Tsukuba
- System study for Memory-bandwidth Oriented Vector Architecture, led by Tohoku University
- Application study led by the RIKEN R-CCS

Based on these feasibility studies, the initial plan at the beginning (2014) was a combined system of a general purpose supercomputer and accelerators with a target performance goal of exaFLOPS. The development of the accelerator part was, however, canceled due to a budget problem. As a result, the basic architecture was decided as a large-scale system using only general-purpose manycore processors studied by the feasibility study project of the University of Tokyo.

The processor architecture suggested by the feasibility study was a manycore processor with 64 cores and two 512-bit-width SIMD arithmetic units for each core. It was reported that other configurations, such as wider SIMD arithmetic units or a dedicated small matrix multiply arithmetic, have been examined but not adopted due to low applicability and low efficacy for real applications. Since the evaluation was done by the configuration supporting one pipe of 1,024-bit-width SIMD and matrix multiply units together, the number of cores was reduced to 32 cores per chip, resulting in poor performance. For example, the execution time of the RS-DFT application by this configuration was increased by 28% compared to two pipes of 512-bit-width SIMD. Only 1,024-bit-width SIMD without matrix multiply units was not evaluated, but the difference from two pipes of 512-bit-width SIMD would be small by the estimation tool based on the profile.

B. Instruction Set Architecture and SIMD Instructions

The choice of the instruction set architecture was an important decision for architecture design. Instead of the SPARC instruction set [17] used for the K computer, Fujitsu offered the Armv8 instruction set with the Arm SIMD instruction set called the Scalable Vector Extension (SVE) [18] for the Post-K. They collaborated with Arm, contributing to the design of the SVE as a lead partner, and adopted the results in the processor architecture design. The processor is custom designed by Fujitsu using their microarchitecture as a backend of the processor core.

The Arm instruction-set architecture has been widely accepted by software developers and users not only for mobile processors, but also for HPC recently. For example, Cavium Thunder X2 is a processor designed for servers and HPC and used for several supercomputer systems, including Astra [19] and Isambard [20]. While the Intel x86 architecture is dominant in HPC, Arm processors are expected to be open to the possibilities and the diversity in the HPC community.

The SVE is an extended SIMD instruction set. The most significant feature of the SVE is uniform support for vector lengths from 128 bits to 2,048 bits. The SVE realizes Vector Length Agnostic (VLA) programming, as the name suggests, and does not depend on the vector length. We have decided to have two 512-bit-width SIMD arithmetic units, as suggested by the feasibility study. Our previous research [21] using the gem5 simulator and McPAT also suggests that this configuration of SIMD provides a good balance between performance and energy consumption under the current O3 resources described later. Note that our implementation, A64FX processor, is the first processor supporting the SVE.

C. Processor Chip Configuration

Fujitsu proposed the basic structure of the manycore processor architecture. Each core has an L1 cache, and a cluster of cores shares an L2 cache and memory controller. This cluster of cores is called a Core-Memory Group (CMG). In a processor chip, CMGs are connected via a network-on-chip.

While other high-performance processors, such as those of Intel and AMD, have L1 and L2 caches in the core and share an L3 cache as a last-level cache, the core of our processor has only an L1 cache to reduce the die size for the core. In the case of the A64FX chip, the area size for the L1 cache occupies more than 10% of each core. If an additional other

TABLE III
COST OF DIE AND PACKAGE, MULTI-CHIP MODULE

#Cores x #PKG(p)/MCM(M)	#Dies	Die area	Die cost	PKG/MCM cost	Total
64 x 1p	1	1.00	0.82	0.18	1.00
32 x 2p	2	0.65	0.80	0.26	1.06
16 x 4p	4	0.38	0.73	0.30	1.03
32 x 2M	2	0.64	0.77	0.31	1.08
16 x 4M	4	0.37	0.71	0.34	1.05

level of cache was implemented and required a similar area in each core, the die size would increase by at least more than 4%, estimated from the chip photograph. This impact is so large that it would be not acceptable from a cost perspective.

Based on this basic structure, we had to decide the following parameters:

- The number of cores in a CMG
- The number of CMGs in a chip
- How to connect cores to shared L2 in a CMG
- The number of ways, the size, and throughputs of the L1 and L2 caches
- The topology of network-on-chip to connect CMGs
- The die size of the chip
- The number of chips in a node

Our technology target for silicon fabrication was 7-nm FinFET technology. The die size of the chip is the most dominant factor in terms of cost. It is known that the cost of the chip increases in proportion to the size and increases significantly beyond a certain size. Moreover, the yield of the chip becomes low as the size of the chip increases. One configuration is to use small chips and connect these chips by multi-chip module (MCM) technology. Recently, AMD has used this "chiplet" approach successfully. The advantage of this approach is that a small chip can be relatively cheap with a good yield. However, in the present case, the cost of MCM was deemed too high, and furthermore, a different kind of chip for the interconnect and IO must be made, resulting in even higher costs. Table III shows the cost of die and package, MCM, relative to the cost of 64 cores in 1 package at the time of basic design phase. The connection between chips on the MCM would also increase the power consumption.

Thus, our decision was to use a single large die containing some CMGs and the network interface for interconnect and PCIe for I/O connected by a network-on-chip. The size of the die was about $400mm^2$, which was reasonable in terms of cost for 7-nm FinFET technology.

In order to maintain a certain degree of compatibility with the K computer, we assumed that a commonly used programming model was OpenMP-MPI hybrid programming in which each MPI process multithreaded with OpenMP runs in a CMG. In order to share the L2 cache with cores in a CMG, we chose a cross-bar connection between L2 and cores.

We assumed that the total number of cores was 64 and examined the performance for 8 cores/CMG x 8 CMGs and 16 cores/CMG x 4 CMGs by running extracted kernels from the target applications. Table IV shows the performance of some kernels by the 4 CMG and 8 CMG configurations for 64 cores. For the performance evaluation, the total L2 cache size was kept the same for both cases. The results evaluated by the performance estimation were as follows:

TABLE IV
KERNEL PERFORMANCE BY THE 4 CMG AND 8 CMG CONFIGURATIONS FOR 64 CORES

kernel	4CMG 1proc/CMG 16threads/proc	4CMG 2proc/CMG 8threads/proc	8CMG 1proc/CMG 8threads/proc
Kernel (Adventure)	100%	112%	112%
Kernel (FFB)	100%	105%	110%
KernelA (NICAM)	100%	100%	100%
KernelC (NICAM)	100%	90%	90%
kernelD (NICAM)	100%	113%	122%

NOTE: The performance in this table is the average of small kernels extracted in the different ways from the kernel shown Table VI, V

- For some kernels, the performance of 8 cores/CMG was better since the latency to the small L2 cache is reduced compared to the 16 cores/CMG configuration.
- We found that a few kernels ran slower on 8 cores/CMG because the cache hit ratio of the smaller cache was low.

If the number of cores in CMGs was less than 8, then the L2 cache size was too small. The number of cores in a CMG was examined by the analytical performance model of HPL.

Our conclusion was that 8 cores/CMG was better than 16 cores/CMG. When we made a decision to use the 7-nm FinFET technology of TSMC, we decided 48 cores (plus 4 cores) and 12 cores/CMG x 4 CMGs. Four CMGs provided 4 High-Bandwidth Memory (HBM) units as a main memory, as described later.

We have designed the network-on-chip similar to a ring-topology network to support the memory coherence protocol between CMGs so that a shared memory program can be executed using multiple CMGs.

D. Memory Technologies

As the peak floating-point performance of the CPU chip was expected to reach a few TFLOPS, the memory bandwidth of DDR4 was too low compared to the floating-point performance. Thus, high-speed memory technologies, such as HBM and Hybrid Memory Cube (HMC), were examined to balance the memory bandwidth and floating-point performance.

The HMC is a technology to connect the stacked memory chips and the CPU chip via high-speed serial links. Fujitsu already had adopted the HMC 1.0 for the FX100. The link speed by HMC 2.0 was 60 GB/s per link (per direction), and up to 4 links can be used. The power consumption to drive the serial links is large.

The HBM is a stacked memory chip connected via TSV on a silicon interposer. The HBM2 provides a bandwidth of 256 GB/s per module. The capacity of HBM2 is up to 8 GiB, but the cost is high because the silicon interposer is required.

As a memory technology available around 2019, HBM2 was chosen for its power efficiency and high memory bandwidth for both read and write. We decided not to use any additional DDR memory to reduce the cost. As described in the previous section, the number of HBM2 modules attached to CMG is 4,

TABLE V
KERNEL PERFORMANCE BY CHANGING L1 CACHE LINE SIZES (NORMALIZED TO THE ESTIMATED PERFORMANCE OF 256 BYTES)

Kernel (Application)	128 Byte	256 Byte
Kernel (Adventure)	99 %	100 %
Kernel (FFB)	78 %	100 %
Kernel A (NICAM)	86 %	100 %
Kernel B (NICAM)	56 %	100 %

that is, the main memory capacity is 32 GiB. Although it seems small for certain applications, we already have many scalable applications developed for the K computer. Such scalable applications can increase the problem size by increasing the number of used nodes.

E. Cache Structure

The key to designing a cache architecture is to provide a high hit rate for many applications and to prevent a bottleneck when data is supplied with full bandwidth from memory. We examined various parameters such as the line size, the number of ways, and the capacity in order to optimize the cache performance under the constraint of the size of the area on the die and the amount of power consumption.

When the capacity of data is increased and the line size is the same, the area for tags is increased as well as the area for data. The area size of the cache for the tag and data depends on the kind of available RAM macro. When the number of ways is increased, the complexity of the data path and the amount of control logic are increased even when the size of the data remains the same. We found that changing the number of ways and the line size of the L1 cache affects the area size. When the line size is changed from 256 bytes to 128 bytes, the area size is increased by 5%.

We examined the impact of the cache configuration on the performance by running some kernels extracted from target applications on the simulator for a single CMG. As shown in Table V, when changing from 256 bytes to 128 bytes, a performance degradation was found in some kernels because the number of cache misses was increased. Although applications with fine-grain memory access would have the benefits of a small cache size, many HPC applications have a good performance with a 256-byte line size. No significant benefit was found by changing the cache line size to more than 256 bytes. Moreover, a wider cache line requires a wider data path, resulting in a larger area and longer latency. Table VI shows the performance by changing the L1/L2 cache size and number of ways. No significant performance gain is found by changing from 4 ways to 8 ways, which has a very small impact on the power consumption. As a result, we chose a four-way L1 cache with a 256-byte line size and an 8 MB L2 size.

We have designed the cache to save power for accessing data in a set associative cache. Data read from a way and tag search may be used in parallel to reduce the latency, but this may waste power because the data will not be used when the tag is not matched. In our design, data access is performed after a tag match. While it causes a long latency, there is less impact on the performance in the case of throughput-intensive HPC applications. This design is applied to the L1 cache for vector access and the L2 cache. We found that this design reduces the amount of power by 10% in HPL with almost no performance degradation.

TABLE VI
KERNEL PERFORMANCE BY CHANGING L1/L2 CACHE SIZES AND NUMBER OF WAYS (NORMALIZED TO THE ESTIMATED PERFORMANCE OF L1:64KIB4WAY AND L2:6MIB24WAY)

Kernel (Application) L1 / L2	64K4W 6MB24W	64K8W 6MB24W	64K4W 8MB16W
Kernel (Adventure)	100 %	100 %	100 %
Kernel (FFB)	100 %	101 %	101 %
Kernel B (NICAM)	100 %	**94** %	**110** %
Kernel C (NICAM)	100 %	101 %	101 %
Kernel D (NICAM)	100 %	**98** %	101 %
Kernel (Seism3D)	100 %	100 %	99 %

F. Out-of-Order (O3) Resources

The microarchitecture is an out-of-order architecture designed by Fujitsu. The parameters for O3 resources include:

- Number of entries in the reservation station
- Number of Re-Order Buffers (ROBs)
- Number of renaming registers (general-purpose registers and floating-point/SIMD registers)
- Number of entries for the load/store queue

First, we examined several combinations of these parameters by running some kernels on the processor simulator to decide the ratio of the O3 resources. Keeping the decided ratio, the amount of the O3 resources was decided by the trade-off between the performance and the impact to the die size. Figure 3 shows the impact to the area size and the performance of kernels picked from a set of measured kernels and the average by changing the amount of O3 resources. We found that the impact of O3 resources to the area size is large when increasing these O3 resources, while larger resources may improve the performance.

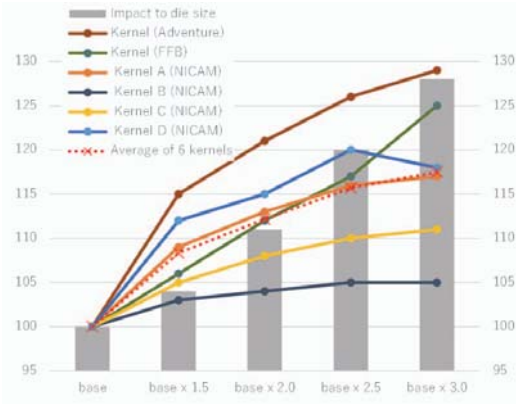
The decisions regarding the O3 resources were some of the most difficult problems in the core design because of the following reasons:

- At the basic design phase, the compiler optimization was immature to evaluate reasonable sets of the parameters.
- Fujitsu had experience in designing O3 processors, but these instruction sets were SPARC with proprietary extensions and were completely different from Armv8.

As shown in the Figure, ”base x 2.0” and ”base x 2.5” can be candidates, and ”base x 2.0” was chosen with respect to the impact to area size. The final decision was made as described in the microarchitecture document [27], considering the balance with the area size.

G. Enhancement for Target Applications

During the co-design process, we received feedback to enhance the architecture to improve the performance of the target applications. The chosen enhancements are as follows:



Base Set: #RS=40, #ROB=64, renaming regs #FP=48, #GP=32, #Fetch Port=20, #Store Port=12, #write Buffer=4

Fig. 3. Impact to area size and kernel performance by changing the amount of O3 resources (Size and performance are shown in %, relative to "Base Set")

- **Combined gather operations:** For the indirect memory access in the gather instruction, when target elements are within a 128-byte aligned block for a pair of regs, requests to cache access are combined into one request, resulting in doubling the gather (indirect) load's data throughput. We found the performance improvement of the kernels containing the indirect load: 36% in "Kernel (FFB)" and 20% in "Kernel D (NICAM)".
- **Optimization for re-use of data on L2 cache:** In some applications, we found that the same data on the L2 cache is re-used frequently and increases the traffic to replace the data on the L1 cache, resulting in a performance bottleneck. We reduce the traffic for eviction by updating only the tag when it is detected that data on the L1 cache is not modified. This optimization contributes decreasing the L1 busy rate from 84% to 78% in "Kernel A (NICAM)", making room for improvement of the cache bandwidth.

H. Interconnect between Nodes

We have selected the Tofu network topology, a six-dimensional torus network, for performance compatibility with the K computer in large-scale applications.

From the viewpoint of the technology availability and maturity, we chose 28 Gbps SerDes for the link of the interconnect. While the link speed is 6.8 GB/s per link, the injection bandwidth per node is 40.8 GB/s because the number of Tofu Network Interfaces (TNIs), which are DMA engines between the network and the memory in a chip, is six.

Communication patterns were extracted from target applications, and the communication performance was estimated by the analytical model. Many target applications have neighbor communication patterns or communicate with near nodes. Therefore, the "Tofu" network with an injection bandwidth of 40.8 GB/s was concluded to be sufficient.

For all-to-all communication used in some applications, we investigated the benefits or feasibility of an additional dedicated all-to-all network, but it was not adopted due to cost. Instead, we decided to enhance the reduction operation in the interconnect to support the reduction of three double-precision floating-point numbers for QCD applications.

The new version of the interconnect is named "TofuD". The details of TofuD are described in [22].

TABLE VII
EVALUATION OF POWER KNOBS FOR DGEMM AND STREAM

Power Knob	Default	Setting	dgemm		stream	
			Perf.	Power	Perf.	Power
Frequency	2.0GHz	1.6GHz	80%	82%	98%	89%
Inst. issuances	4 inst.	2 inst.	59%	95%	100%	98%
FPU	2 pipes	1 pipe	52%	84%	100%	100%
EXU	2 pipes	1 pipe	96%	100%	100%	100%
Memory throttling	100%	50%	100%	100%	62%	84%
Eco mode	off	on	52%	71%	100%	82%

I. Co-Design for Low Power

One of the most important challenges for an exascale system is to reduce the power consumption. We investigated several power control mechanisms for saving power. Our fundamental policy was to increase the power efficiency by reducing unnecessary power without degrading performance. To meet this goal, we prepared multiple power control mechanisms called *power knobs*, which can be turned them on and off according to the applications' characteristics.

The initial candidates for the power knobs were: the clock frequency, the SIMD width, the number of pipelines for the floating-point unit and the integer unit, the number of ways for the L2 cache, the number of instruction issuances, the O3 resources such as reservation station entries and rename registers, and memory access throttling. In order to determine which power knob should be implemented, we first estimated the power consumption by the performance estimation tool, followed by evaluation using a cycle-level simulator for more accurate analysis. We used the dgemm kernel as a compute-intensive workload and the stream benchmark as a memory-intensive workload, as well as some kernels from target applications. As a result, we decided not to implement the control related to O3 resources because they were not so effective, except for the number of instruction issuances. As described in the section on the cache design, the cache is well designed for the lower power, and the control of the number of ways in the L2 cache was found to be no longer effective in reducing power consumption. Table VII shows a part of the results of evaluation of power knobs using dgemm and stream. For example, when changing the frequency from 2.0 GHz to 1.6 GHz, the performance of dgemm is 20% reduced and power is 18% reduced, while when limiting the FPU pipeline to one pipe, the performance of dgemm is 50% reduced but power is only 16% reduced.

Memory access throttling was expected to be beneficial for compute-intensive applications. However, in the case of HBM, unlike HMC, the benefit by changing the frequency to access the memory is small since the power consumption to access

TABLE VIII
POWER MODE FOR TARGET APPLICATION (PERFORMANCE AND POWER IS RELATIVE TO NORMAL MODE (N))

Application	Pow. mode	B		N+E		B+E	
		perf.	pow.	perf.	pow.	perf.	pow.
GENESIS	B	1.09	1.20	1.00	0.80	1.09	0.96
Genomon	B	1.10	1.17	-	-	-	-
GAMERA	B+E	1.06	(1.14)	1.00	0.81	1.06	0.89
NICAM+LETKF	B+E	1.07	(1.18)	0.97	0.79	1.04	0.91
NTChem	B	1.08	1.21	0.57	0.69	0.62	0.83
ADVENTURE	N	1.07	(1.21)	0.90	0.85	0.98	1.00
RSDFT	B	1.06	1.20	0.71	0.80	0.77	0.90
FFB	B+E	1.10	(1.17)	1.00	0.80	1.10	0.94
LQCD	B+E	1.05	1.17	1.00	0.74	1.05	0.83

the HBM is small enough if the memory is not accessed frequently.

For memory-intensive applications, the control of the arithmetic pipeline is expected to be beneficial for saving power because the utilization of the arithmetic unit is low. Actually, we found that there is no reduction of power by controlling the arithmetic unit for stream as in Table VII. This was because the mechanism for stable operations against power fluctuation consumed a certain amount of power, even when the arithmetic unit was not working. We defined a new mode called *eco mode* in which only one arithmetic unit is active with the mechanism for stable operation for another unit inactive. According to the evaluation of the stream benchmark, it was confirmed that the power consumption of the stream benchmark could be reduced by 18% without any performance degradation by using the eco mode as in Table VII.

In order to improve the power efficiency, it is important to lower the power supply voltage as much as possible while increasing the frequency. Another demand is to improve the maximum performance even if the power consumption is slightly increased in order to pursue the performance. For this demand, we provide *boost mode* in which the power supply voltage is increased to increase the maximum frequency beyond that in the normal mode. As described in Section VI, the clock frequency of the boost mode is 2.2 GHz, increased from 2.0GHz of the normal mode.

Since the eco mode can also be active in the boost mode, the four modes of 'boost' (B), 'boost+eco' (B+E), 'normal' (N), and 'normal+eco' (N+E) can be selected. We have evaluated several kernels taken from target applications using a performance estimation tool, and estimated the modes that could achieve the maximum performance within the capacity of the maximum power. Table VIII shows the results. The values are the ratio based on each normal performance and power. The value in parentheses indicates that the maximum power has been exceeded. It should be noted that four applications out of nine target applications choose the 'boost eco mode' for maximum performance, where boost eco mode improves performance while reducing power.

The power knobs can be controlled within user programs using the Sandia Power API [23], [24]. Moreover, this API allows acquiring the estimated power calculated from the performance counters and the measured power of the node.

In a large-scale system, the power consumption at the idle state is also important because the number of nodes is huge because the total power consumption of the system reaches several megawatts. To save power in the idle state, we defined a new CPU power state called *node retention*, in which the power consumption is reduced to 50% of the CPU standby state.

More detailed analysis of the processor power consumption is described in [25].

V. CO-DESIGN FOR STORAGE AND FILE I/O

A. Basic Design: From K to Post-K

Our previous system, the K computer, consists of two types of nodes, compute and I/O nodes. The I/O nodes, which are connected to compute nodes by the Tofu interconnect, provide the local file system, and no applications run on these nodes. The global file system, connected to the main component by an I/O network, has limited data transfer throughput (47.3 GB/s) for writing. Thus, the K introduced a file staging method in which input files and output file are transferred between the local file system and the global file system before and after the job's execution. Since the staging operation can be overlapped with the job execution, this method mitigated the drawback of the global file system's throughput, but caused uncertain processing times on the staging-out phase if a huge number of files were copied.

Similar to the K computer, we decided to have a 1st-layer storage system for the Post-K, but the purpose of this storage system was examined by reflecting the file I/O usages in the target applications.

The system was designed to have two types of nodes: a compute only node and a compute and I/O node, where the latter is called an SIO node. One of the 16 compute nodes is an SIO node having a 1.6-TB SSD. The SSD capacity and performance were determined based on both budget and our target application demands. The SIO node serves the 1st-layer storage system. The global file system, the 2nd-layer storage system, is a Luster-based parallel file system connected via an InfiniBand network.

In the rest of the section, after investigating the file I/O characteristics in our target applications, the design of the 1st-layer functionality is described.

B. Application I/O Characteristics

Our target applications can be categorized into three types in terms of file I/O patterns:

1) P1: RW: The P1 pattern is such that the input data is *R*ead before the main computation starts, and the output data is *W*ritten after the computation. The largest input/output file size categorized in this pattern is smaller than other categorized applications. Thus, there was no need to consider the I/O throughput required by the target applications belonging to this category.

2) P2: R[W]∗W: This is a popular file I/O pattern for time-evolved simulations, which involves three types of file I/O operations, i) *R*eading data for initialization, ii) *W*riting a simulated result every time-step, and iii) *W*riting the result

TABLE IX
SUMMARY OF I/O CHARACTERISTICS

I/O pattern	Read (GB)	Total Cyclic Write (GB)	Write (GB)	Execution Time(sec)
P2: *R[W]∗W*	67,019	109,486 (3,650 /cycle)	3,650	84,390
P3: *R[WR]∗W*	412,317	13,194,140 (3,298,535 /cycle)	3,221	77,840

at the end. Since the simulated result is written in many cycles, the output file size is huge. GAMERA, an earthquake simulator, is one of the typical cases in our target applications. In the basic design phase, it was estimated that GAMERA reads about 67 TB at the initialization time and then generates about 3.65 TB data every 2,813-second cycle. The total file size is 109.5 TB for 30 cycles (about a 24-hour execution time).

3) P3: R[WR]∗W: This is the case of an application that consists of some application codes communicating with each other via files. In our target applications, NICAM+LETKF, an application of climate ensemble simulation and data assimilation, has this file I/O pattern. NICAM reads 412 TB of initial data and then performs I/O as follows: Each 19,460-second cycle, in a typical case, NICAM generates a 2.89-PB in double precision simulated result, LETKF reads this result, and LETKF generates a 412-TB assimilated result for the input data of the next cycle of NICAM. After certain cycles are done, a 3.2-TB result is stored in the persistent storage. NICAM+LETKF generates 13.2 PB of data for 4 cycles (within a 24-hour execution).

C. Design of 1st-layer Functionality

Table IX summarizes the important file I/O characteristics for designing the storage system described in the previous section. The generated data size results from the execution. In the P2 case, although we assume 844 s (only 1% overhead of total execution time (84,390 s)) for file I/O processing is allowed, the required effective I/O throughput is:

$$213GB/s \fallingdotseq (67,019 + 109,486 + 3,650)GB/844s$$

This demand is not so high and can almost be satisfied with one volume of the 2nd-layer storage, but the requirement of periodical file accessing in P3 would be crucial if we naively estimated the required I/O throughput based on just the total output data size in P3. The required effective I/O throughput if we assume 10% overhead in 77,840 s is as follows:

$$3,443GB/s \fallingdotseq (412,317+13,194,140*2+3,221)GB/7,784s$$

This requirement is not realistic. A closer look at the file I/O behavior may be taken as follows: Each cycle involves four NICAM jobs (ensembles) and one LETKF job. Each NICAM process generates one file of simulated results, about 1,050 MiB. After finishing the four NICAM jobs, each LETKF process assimilates the results generated by those jobs (ensembles) with observed data and outputs a file for the input data, about 150 MiB, for NICAM in the next cycle. If four NICAM ensemble jobs and one LETKF job are allocated and four processes of each job run on the same node, then the size of read/write I/O operations in each node is

$$40GB \fallingdotseq ((1,050MiB + 150MiB) * 4procs) * 4jobs + (1,050MiB * 4 + 150MiB * 4) * 4procs$$

Note that the files generated by NICAM or LETKF are only read by the LETKF or NICAM processes in the same node [26], and thus a local file system was designed in the 1st-layer storage. We specified the minimum required I/O throughput of the local file system to be 49 MB/s per node, as shown in Table XI. This means that NICAM+LETKF spends 816 seconds (40 GB/49 MB) of the 19,460-second (1 cycle) execution time on file I/O processing. This is only 4% of the total execution time and is acceptable to the user.

In addition to the local file system provided in the 1st-layer storage, the following functions are provided:

- **Shared file system:** Each job may have an individual shared file system to share files across processes running on the same job. Since the lifetime of this file system is the same as the lifetime of the job, files must be copied to the global file system if needed.
- **Cache for 2nd-layer storage:** The capacity of the local and shared file systems is limited by the availability of the 1.6-TB SSD on the SIO node shared by 16 nodes. A larger data set, not fitted into the 1st-layer storage, must be on the 2nd-layer storage. The 1st-layer storage also plays the role of a cache system for the 2nd-layer storage and asynchronous data transfer is enabled.

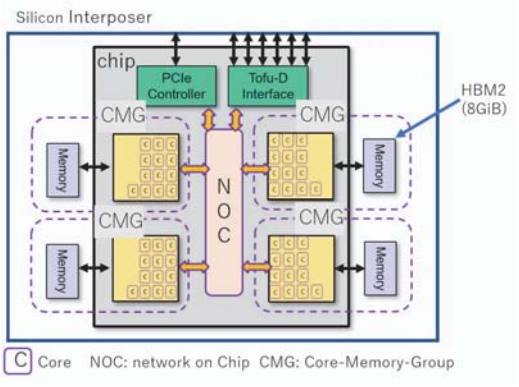


Fig. 4. Block diagram of the A64FX processor

VI. THE SUPERCOMPUTER "FUGAKU"

In 2019, the name of the system was decided as "Fugaku". The delivery of the whole system was completed in May, 2020.

A. Specifications of the A64FX Processor

The node processor is a single chip, named A64FX, which consists of 48 cores with 2 or 4 cores dedicated for OS activities, 32 GiB of HBM2 memory, TofuD Interconnect, and a PCI express controller, as shown in Table X. The diagram is illustrated in Figure 4. The details of the microarchitecture are described in [27].

B. Specifications of the System and Storage

The Fugaku system consists of 158,976 nodes in 432 racks. Each node has one A64FX chip connected with the TofuD Interconnect. As described in the section on co-design of the file

TABLE X
SPECIFICATIONS OF THE A64FX PROCESSOR

Component	Specification
Architecture	Armv8.2-A SVE (512 bits SIMD)
Core	48 cores for compute and 2/4 for OS activities
	Normal Mode: Freq: 2.0 GHz, DP: 3.072 TF, SP: 6.144 TF, HP: 12.288 TF
	Boost Mode: Freq: 2.2 GHz, DP: 3.3792TF, SP: 6.7584 TF, HP: 13.5168 TF
L1 Cache	64 KiB, 4 way, 256 GB/s(load), 128 GB/s (store) @2.0GHz
L2 Cache	8 MiB/CMG, Total 32GiB/16way, BW for Core: 128 GB/s (load), 64 GB/s (store) @ 2.0GHz
Memory	HBM2 32 GiB, BW for Chip 1024 GB/s
Interconnect	TofuD: 28 Gbps x 2 lane x 10 port, injection BW: 6.8GB/s x 6
I/O	PCIe Gen3 x 16 lane
Silicon Technology	7nm FinFET, CoWoS (Chip on Wafer on Substrate) [28] for HBM2

TABLE XI
SPECIFICATIONS OF THE 1ST- AND 2ND-LAYER STORAGES

		Minimum Throughput	Measured Throughput
1st Storage	write	49 MB/s /node	125 MB/s /node
	read	113 MB/s /node	293 MB/s /node
2nd Storage	write	200 GB/s /volume	220 GB/s /volume
	read		211 GB/s /volume

Note: The 2nd storage is formed from 6 volumes.

system, the 1st-layer storage system consists of SSDs attached to SIO nodes allocated one for every 16 compute nodes. The 2nd-layer storage system is the global file system, which is a Luster-based parallel file system, FEFS, developed by Fujitsu. The specifications of the 1st- and 2nd-layer storages are shown in Table XI.

A Linux kernel runs on each node. All system daemons run on two or four additional cores, called assistant cores. The CPU chip with two assistant cores is used on compute only nodes. The chip with four assistant cores is used on compute & IO nodes because such nodes service I/O functions requiring more CPU resources. In addition to a Linux kernel, a light-weight kernel, McKernel [29], is also supported for execution of their applications. McKernel enables customization of the runtime environment, such as memory allocation.

C. Preliminary Performance Results

We have confirmed the node performance for the basic kernels, more than 830 GB/s for the stream triad benchmark and more than 2.5 TFLOPS for the dgemm kernel with 90% efficiency. The latency and bandwidth of the TofuD Interconnect are from 0.49 to 0.54 μs and 6.35 GB/s for a 1-MB put operation.

The successful achievement of our co-design effort is such that the prototype systems of A64FX processors took the 1st place at Green 500 [4] in November 2019. This performance measurement demonstrated that the A64FX processor has the highest energy efficiency, achieving 1.9995 PFLOPS for HPL compared to 2.36 PFLOPS in peak performance and 16.87 GFLOPS/W in performance per 1 watt of power consumption, exceeding the efficiency of GPUs. This corresponds to the first item in our KPIs, that is, a power-efficient system.

It has been already measured that two target applications, GENESIS and NICAM+LETKF, achieve a performance more

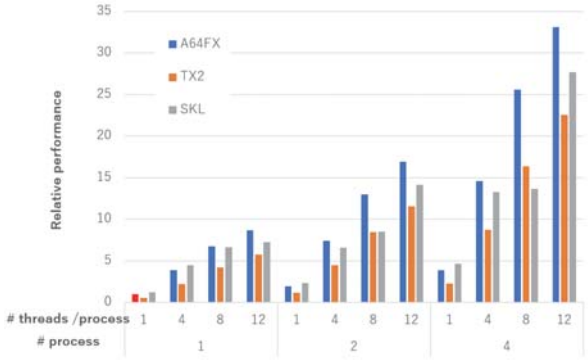


Fig. 5. Performance of CloverLeaf (results are normalized to a single core of A64FX)

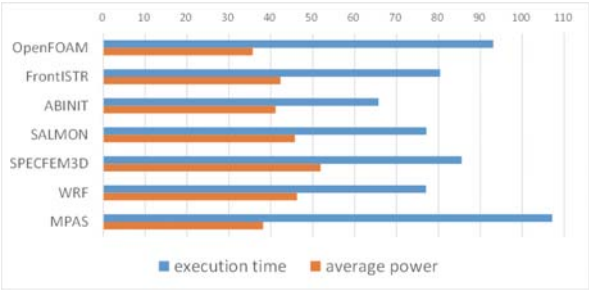


Fig. 6. Performance and power efficiency of open-source applications (results are shown in %, relative to Intel Xeon (dual sockets))

than 100 times faster than that of the K computer. Other target applications will also be able to achieve the expected performances, as indicated in Table I.

The results we shall present below focuses on a single node in comparison to other state-of-the-art processors found in the majority of supercomputers today. These results will be extended to evaluate the whole Fugaku system for a future study of an at-scale production run.

Figure 5 shows the performance of A64FX (2.0 GHz, single socket) for CloverLeaf [30] taken from UK Mini-apps in comparison with the dual sockets of Xeon Gold 6126 (Skylake, 2.6 GHz, 12 cores/socket) and Cavium Thunder X2 (2.0 GHz, 28 cores/sockets). The CloverLeaf hydrodynamics mini-app solves Euler's equations of compressible fluid dynamics, under a Lagrangian-Eulerian scheme, on a two-dimensional spatial regular structured grid. The benchmark is an example of a stencil code and is classed as memory bandwidth-bound. The performance is normalized to a single core of A64FX, while changing the number of MPI processes and the number of cores for OpenMP in each MPI process. The performance of A64FX is better than those other processors thanks to the large memory bandwidth of HBM2.

Other UK Mini-apps results are presented in [31].

Figure 6 shows the execution time and the average power of A64FX (2.2 GHz, single socket) relative to dual sockets of Xeon Platinum 8268 (Cascadelake, 2.90 GHz, 24 cores/socket) for several open-source scientific applications, described in Table XII. As the results demonstrate, the power consumption of A64FX is about half that of the Intel Xeon's in most of

TABLE XII
DESCRIPTIONS AND PARAMETERS OF OPEN-SOURCE APPLICATIONS

Applications	Field	Ver.	Size	Model	Solver	MPI/OpenMP	Measured Section
OpenFOAM [32]	CFD	1812	14 million meshes	Motor Bike	PCG	FlatMPI (48 procs)	Time integration loop
FrontISTR [33]	Structure Analysis	5	0.4 million meshes	Hinge	SSOR+CG	FlatMPI (48 procs)	Analysis section
ABINIT [34]	Material	8.10.2	26244 FFT meshes	tmbt_3	GW/spectral method	FlatMPI (48 procs)	Analysis section
SALMON [35]	Ab-initio Light-Matter	1.2.1	1.0 million meshes	exercise_07_classicEM_lr	FDTD	FlatMPI (48 procs)	Analysis section
SPECFEM3D [36]	Seismic	7.0.2	64 x 64	s362ani	SEM	Hybrid (4 procs x 12 threads)	Iteration loop
WRF [37]	weather	3.8.1	425 x 300 x 35	Conus 12km		Hybrid (4 procs x 12 threads)	Domain integration loop
MPAS [38]	weather	6.2	40962 meshes	The Jablonowski and Williamson baroclinic wave		Hybrid (24 procs x 2 threads)	Time integration loop

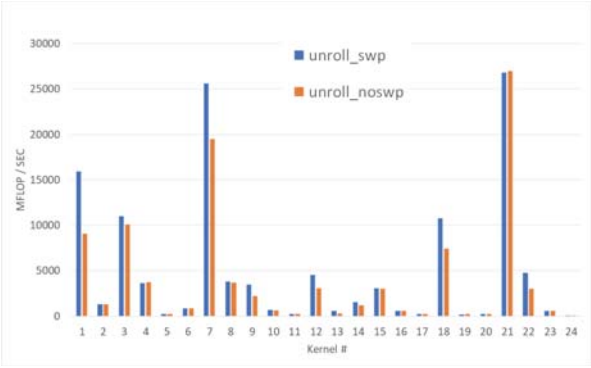


Fig. 7. Performance improvement of Livermore Loops by software pipelining optimization

these applications, while the performance of A64FX is better.

VII. CO-DESIGN OF COMPILER AND APPLICATIONS

Since October 2019, the test system of Fugaku has been available for users. As well as the performance evaluation of several programs, including the target applications and benchmark programs, the co-design effort must move in the direction from architecture to software, including compiler optimization and performance turning of application programs.

A. Compiler Optimization for A64FX Processor

Through the co-design process for the A64FX processor, the architecture has been designed as an HPC-oriented processor. This means that the latencies to execute floating-point instructions are relatively long because many kinds of floating-point arithmetic operations are executed in nine cycles by using a Fused Multiply Add (FMA) arithmetic unit. This unit contributes to reducing the die size, but may cause a long latency. The cache structure having only an L1 cache within each core may increase the latency to access the data, while other server-class processors have L1 and L2 caches inside the core. Many HPC applications can be executed efficiently with throughput-oriented operations, such as loops for vector operations, but this may cause a problem if there are a lot of operations affected by long latency. This latency problem may cause another problem whereby the O3 resources tend to be depleted, resulting in a performance degradation.

To mitigate the problem, we found that the software pipelining technique is effective to improve the performance of some types of loops. This is because the software pipelining optimization may help to make the live period of values in O3 execution short so that it enables O3 resources to be used efficiently. Figure 7 shows the performance improvement of each loop in the Livermore benchmark [39]. Note that each loop is executed by a single core.

Another optimization is loop fission. As the amount of O3 resources is small compared to the latency of floating-point instructions and load/store instructions, a loop with a long body may cause a shortage of O3 resources. In addition, such a loop causes a lot of register spills, resulting in low performance. The Fujitsu compiler supports the function of automatic loop fission. Moreover, this compiler facilitates software pipelining for split loops and expects overlap in operations.

B. Co-Design of Applications: NICAM Case

In parallel with the co-design process for the architecture, co-design on applications was carried out in the nine priority projects. We describe some co-design efforts in NICAM [8] global cloud-resolving weather application as a case study.

Coding to use structure load/store instructions: The SVE supports SIMD structure load/store instructions. When the dimension size of the array is unknown at the compilation time, the compiler may generate a code using the indirect-load/store instruction. We found that if the distance to the next element is known at compile time by passing the dimension size as an argument to the subroutine, the structure load/store instruction, which is more efficient than an indirect load/store, may be generated, resulting in improved performance. This optimization is effective when a three-dimensional array is stored as Array of Structures (AoS) data in the NICAM code.

Loop fission for large loop body: In NICAM, the part computing the physical processes has loops containing a large loop body. We found that the performance is improved by dividing the loop body into smaller loops because the optimization of software pipelining may be applied, and some register spills are removed. We found that it runs from 1.3 to two times faster in some kernels by loop fission.

Mixed-precision computation: We applied mixed-precision computation by using single-precision floating-point numbers for the variables in the solver of the fluid dynamics and some parts of the physical process. This is a very promising optimization for memory-intensive computation because the demand for memory bandwidth is decreased. This optimization is found to be effective so that we successfully make the entire execution about 1.6 times faster. As this optimization may cause a loss of accuracy, we must carefully verify the accuracy of the simulation results.

VIII. Conclusion and Retrospective Comments

In 2020, the installation was completed and the system already partially serves the early-access program including projects to fight against the COVID-19 pandemic.

In June 2020, the Fugaku achieved HPL performance of 415.53 PFLOPS, using 396 racks (152,064 nodes, approximately 95.6% of the entire system) with a computing efficiency ratio of 80.87%, and ranked as the 1st position of the Top 500 list. It also took the first place in the ranking of HPCG, achieving 13,400 TFLOPS (360 racks, 138,240 nodes, approximately 87% of the entire system), while claiming the first position in the HPL-AI ranking with 1.421 EFLOPS using 330 racks (126,720 nodes, approximately 79.7% of the entire system) HPL-AI is a new benchmark that takes into account the capabilities of single-precision and half-precision arithmetic logic units used in artificial intelligence. It has taken the top spot on the Graph500 list, a ranking of the world's fastest supercomputers on data-intensive workloads, with the performance of a breadth-first search of a large scale graph 70,980 GTEPS, using 92,160 nodes (approximately 58% of the entire system), The achievement of remarkable records in these rankings demonstrates the overall high performance of Fugaku for a wide range of workloads.

Through our co-design effort, we have almost confirmed the KPIs set at the beginning of the project. Fugaku has been proved to be the most power-efficient system, as the Green 500 result indicated, and it is confirmed that the target applications can achieve the target performance as set in the project. Since our system is a general-purpose manycore-based system without any accelerators, standard OpenMP-MPI hybrid programming can be used so that the system has some degree of compatibility with the K computer.

Since the development of Fugaku has almost reached its end, we would make retrospective comments as follows:

- **Support for AI/ML workload:** Although AI/ML topics were not included as a target application, the SVE has a SIMD instruction for half-precision floating-point operations, which is expected to contribute to the acceleration of AI/ML workloads. If AI/ML topics were set as a target at the beginning, a specialized function for AI/ML would be investigated.
- **Small HBM memory:** We have decided to have only HBM memory because the additional DDR memory would cost too much for the huge number of nodes in our system. If the number of nodes were not so large, then the additional DDR memory could be attached as in Intel KNL.
- **Decision of O3 resources:** It was difficult to decide the O3 resources because the compiler was immature for a new Arm instruction set. Although long latency of instructions may cause a shortage of O3 resources, compiler optimization, such as software pipelining, could mitigate this problem to improve the performance. Deeper interaction of architecture and compiler at an early phase in the co-design process would be desirable. For another point of view, it should be noted that the limited amount of O3 resources in our design contributes to reducing the power consumption and the area size for HPC workloads.
- **Future "disruptive" architecture:** We have decided not to take an accelerator-based approach at the beginning of the co-design process. For future systems beyond exascale, a more disruptive architecture, such as accelerators and specialized hardware, would be required. We also believe that there will still be room to exploit more power efficiency in the approach of a general-purpose processor architecture, as our design demonstrated.

The Fugaku is scheduled to be put into operation for public service around 2021.

Acknowledgment

We would like to thank all members of the FLAGSHIP 2020 project, especially members of the architecture design working group and researchers working for the target applications for their co-design effort.

This work is funded by the Japanese Ministry of Education, Culture, Sports, Science and Technology (MEXT) program for the Development and Improvement for the Next Generation Ultra High-Speed Computer System, under its Subsidies for Operating the Specific Advanced Large Research Facilities.

References

[1] R. Stevens, A. White, S. Dosanjh, et al. "Scientific Grand Challenges: Architectures and Technology for Extreme-scale Computing Report.", Technical report, 2011.

[2] Richard F. Barrett, Shekhar Borkar, Sudip S. Dosanjh, Simon D. Hammond, Michael A. Heroux, X. Sharon Hu, Justin Luitjens, Steven G. Parker, John Shalf, Li Tang, "On the Role of Co-design in High Performance Computing", Volume 24: Transition of HPC Towards Exascale Computing, Advances in Parallel Computing, IOS Press, DOI: 10.3233/978-1-61499-324-7-141, pp. 141 - 155, 2013.

[3] https://www.r-ccs.riken.jp/en/overview/exascalepj *and* https://www.r-ccs.riken.jp/en/postk/project, *FLAGSHIP 2020 Project*

[4] https://www.top500.org/green500/, *The Green500 Web site*

[5] Jung, J., Mori, T., Kobayashi, C., Matsunaga, Y., Yoda, T., Feig, M., and Sugita, Y., "GENESIS: a hybrid-parallel and multi-scale molecular dynamics simulator with enhanced sampling algorithms for biomolecular and cellular simulations," Wiley Interdisciplinary Reviews: Computational Molecular Science, 2015, 5, pp. 310-323

[6] Yoshida, K., Sanada, M., Shiraishi, Y. et al. "Frequent pathway mutations of splicing machinery in myelodysplasia." Nature 478, 6469 (2011). https://doi.org/10.1038/nature10496. See also http://genomon.hgc.jp/

[7] T. Ichimura et al., "Physics-Based Urban Earthquake Simulation Enhanced by 10.7 BlnDOF × 30 K Time-Step Unstructured FE Non-Linear Seismic Wave Simulation," SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, New Orleans, LA, 2014, pp. 15-26.

[8] M. Satoh et al., "The non-hydrostatic Icosahedral Atmospheric Model: description and development," Prog. Earth Planet. Sci., vol. 1, no. 1, 2014, doi: 10.1186/s40645-014-0018-1.

[9] T. Miyoshi and S. Yamane, "Local Ensemble Transform Kalman Filtering with an AGCM at a T159/L48 Resolution," Mon. Weather Rev., vol. 135, no. 11, pp. 38413861, 2007, doi: 10.1175/2007mwr1873.1.

[10] https://www.r-ccs.riken.jp/software_center/software/ntchem/overview/, *NTChem Overview*.

[11] Yukihiro Hasegawa, Jun-Ichi Iwata, Miwako Tsuji, Daisuke Takahashi, Atsushi Oshiyama, Kazuo Minami, Taisuke Boku, Hikaru Inoue, Yoshito Kitazawa, Ikuo Miyoshi, Mitsuo Yokokawa, "Performace evaluation of ultra-large-scale first-principles electronic structure calculation code on the K computer," Internationa Journal of High Performance Computing Applications 28, pp335-355, (2014). See also https://github.com/j-iwata/RSDFT

[12] http://www.ciss.iis.u-tokyo.ac.jp/english/dl/index.php, *FrontFlow/blue*

[13] https://adventure.sys.t.u-tokyo.ac.jp/, *Adventure Proejct home page*

[14] Yoshifumi Nakamura, Yuta Mukai, Ken-Ichi Ishikawa, Issaku Kanamori, "Lattice quantum chromodynamics simulation library for Fugaku and computers with wide SIMD", available at https://github.com/RIKEN-LQCD/qws/.

[15] N. Binkert, B. Beckmann, G. Black, S.K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D.R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M.D. Hill and D.A. Wood. *The gem5 Simulator*, ACM SIGARCH Computer Architecture News, Vol.29, Issue 2, May 2011.

[16] Y. Kodama, T. Odajima, A. Asato and M. Sato, "Accuracy improvement of memory system simulation for modern shared memory processor," Proc. of the HPCAsia2020, pp.142-149, Jan. 2020.

[17] http://www.fujitsu.com/downloads/JP/archive/imgjp/jhpc/sparc64viiifx-extensions.pdf, SPARC64 VIIIfx Extensions

[18] Nigel Stephens et al. "The ARM scalable vector extension." In: IEEE Micro 37.2 (2017), pp. 26–39. doi: 10.1109/MM.2017.35.

[19] https://vanguard.sandia.gov/astra/, *VanGuard Astra - Sandia National Laboratories*

[20] Simon McIntosh-Smith, James Price, Tom Deakin and Andrei Poenaru. "A performance analysis of the First generation of HPC-optimized Arm processors." In: Concurrency and Computation: Practice and Experience 31.16 (2019), e5110. doi: 10.1002/cpe.5110.

[21] Tetsuya Odajima, Yuetsu Kodama, Mitsuhisa Sato, "Power performance analysis of ARM scalable vector extension." COOL CHIPS 2018: pp. 1-3, 2018.

[22] Y. Ajima, T. Kawashima, T. Okamoto, N. Shida, K. Hirai, T. Shimizu, S. Hiramoto, Y. Ikeda, T. Yoshikawa, K. Uchida and T. Inoue, "The Tofu Interconnect D," Proc. of the IEEE Cluster 2018, pp.646-654, Sep. 2018.

[23] R. E. Grant, M. Levenhagen, S. L. Olivier, D. DeBonis, K. T. Pedretti and J. H. Laros III, "Standardizing Power Monitoring and Control at Exascale," Computer, vol. 49, no. 10, pp. 38-46, Oct. 2016.

[24] http://powerapi.sandia.gov/, *Sandia Power API*

[25] Yuetsu Kodama, Tetsuya Odajima, Eishi Arima and Mitsuhisa Sato, "Evaluation of Power Controls on Supercomputer Fugaku", Proceeding of Energy Efficient HPC State of the Practice Workshop (EE HPC SOP 2020) in conjunction with IEEE Cluster 2020, Sep. 2020.

[26] H. Yashiro, K. Terasaki, T. Miyoshi, and H. Tomita, "Performance evaluation of a throughput-aware framework for ensemble data assimilation: The case of NICAM-LETKF," Geosci. Model Dev., vol. 9, no. 7, 2016, doi: 10.5194/gmd-9-2293-2016.

[27] https://github.com/fujitsu/A64FX/blob/master/doc/A64FX_Microarchitecture_Manual_en_1.0.pdf, *A64FX Microarchtecture Manual*

[28] https://www.tsmc.com/english/dedicatedFoundry/services/cowos.htm, *CoWoS (Chip-on-Wafer-on-Substrate) Services*

[29] Balazs Gerofi, Masamichi Takagi, Gou Nakamura, Tomoki Shirasawa, Atsushi Hori and Yutaka Ishikawa "On the Scalability, Performance Isolation and Device Driver Transparency of the IHK/McKernel Hybrid Lightweight Kernel," IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016, Chicago, US. also in https://www.sys.r-ccs.riken.jp/ResearchTopics/os/mckernel/.

[30] A. Mallinson, D. Beckingsale, W. Gaudin, J. Herdman, J. Levesque, and S. Jarvis, "Cloverleaf: Preparing hydrodynamics codes for exascale," in The Cray User Group, May2013

[31] Tetsuya Odajima, Yuetsu Kodama, Miwako Tsuji, Motohiko Matsuda, Yutaka Maruyama, and Mitsuhisa Sato, "Preliminary Performance Evaluation of the Fujitsu A64FX Using HPC Applications", in Embracing Arm: a journey of porting and optimization to the latest Arm-based processors (EAHPC-2020), Sep. 2020.

[32] H. Jasak, A. Jemcov, Z. Tukovic et al., "OpenFOAM:A C++ library for complex physics simulations," International workshop on coupled methods in numerical dynamics, vol. 1000. IUC Dubrovnik, Croatia, pp. 120, 2007.

[33] https://gitlab.com/FrontISTR-Commons/FrontISTR, *FrontISTR : Open-Source Large-Scale Parallel FEM Program for Nonlinear Structural Analysis*

[34] X. Gonze, B. Amadon, P.-M. Anglade, J.-M. Beuken, et. al., "ABINIT: First-principles approach to material and nanosystem properties," Computer Physics Communications 180, 2582 - 2615 (2009). Abinit Project home page: https://www.abinit.org/

[35] https://salmon-tddft.jp/, *Scalable Ab-initio Light-Matter simulator for Optics and Nanoscience*

[36] https://geodynamics.org/cig/software/specfem3d_globe/, *SPECFEM3D GLOBE*

[37] https://www.mmm.ucar.edu/weather-research-and-forecasting-model, *The Weather Research and Forecasting (WRF) Model*

[38] https://mpas-dev.github.io/, *The Model for Prediction Across Scales (MPAS)*

[39] F. H. McMahon. "Livermore fortran kernels: A computer test of numerical performance range." Technical Report UCRL-53745, Lawrence Livermore National Laboratory, Livermore, CA, December 1986. https://www.netlib.org/benchmark/livermore, *Livermore Fortran Kernels*.

Appendix: Artifact Description/Artifact Evaluation

SUMMARY OF THE EXPERIMENTS REPORTED

We ran several open-source benchmarks including Cloverleaf, OpenFORM, FrontISTR, ABINIT, SALMON, SPECFEM3D, WRF, MPAS, on the Fujitsu A64FX processor. We ran these benchmarks also on Cavium Thunder X2, Intel Xeon.

For Cloverleaf (shown in Fig. 4), the details about the platforms are as follows:

- A64FX test chip
  - 48Core/1Node @ 2.0GHz
  - Fujitsu Compiler 201912 (-Kfast,openmp)
  - Paging policy == demand:demand:demand (access to local memory)
- ThunderX2 (TX2) @ Apollo70
  - 28Core/2Socket @ 2.0GHz
  - Arm HPC compiler 20.0 (-Ofast -march=armv8.1-a)
- Skylake (Xeon Gold 6126) @ Cygnus, Univ. of Tsukuba
  - 12Core/2Socket @ 2.6GHz
  - Intel compiler 19.0.3.199 (-O3 -qopenmp -march=native)

For OpenFORM, FrontISTR, ABINIT, SALMON, SPECFEM3D, WRF, MPAS shown in Figure 5, the details about the platforms are as follows:

- A64FX
  - CPU:A64FX 2.2GHzx1
- Intel Xeon Cascadelake(Xeon 8268):
  - CPU:Xeon Platinum 8268 (2.90GHz x24core)x2
  - memory:32GB(2933MHz 16GB x2)
  - OS:RHEL7.6 BIOS:R1.15.0
  - SNC:Enabled
  - Hyper-Threading:Disabled
  - TurboBoost: Enabled

For Livermore Kernels shown in Fig 6, we used only A64FX (2.0GHz)

BRIEF DESCRIPTION OF KERNELS IN THE TABLE II AND III:

- Kernel (Adventure): Kernel taken from Adventure. Block Matrix-Matrix computation for preconditioning of finite element matrices using localSchur complements.
- Kernel (FFB): Tuned kernel taken from FFB. Sparse-matrix vector computation using cube data.
- Kernel A (NICAM): Kernel taken from 2-d horizontal divergence operator in NICAM
- Kernel B (NICAM): Tuned kernel taken from 2-d horizontal divergence operator in NICAM
- Kernel C (NICAM): Kernel taken from 2-d horizontal diffusion operator in NICAM
- Kernel D (NICAM): Kernel taken from 3-d divergence damping operator in NICAM
- Kernel (Seism3D): Kernel taken from sdiffy3 in Seism3D

(Unfortunately, the source codes of the kernels cannot be disclosed due to the problem of intellectual property of Fujitsu.)

BRIEF DESCRIPTION OF APPLICATIONS

- GENESIS: "GENeralized-Ensemble SImulation System"is a molecular dynamics and modeling software for bimolecular systems such as proteins, lipids, nucleic acids, glycans, and their complexes. for proteins. https://www.r-ccs.riken.jp/labs/cbrt/
- Genomon: A pipeline software comprising a suite of bioinformatics tools and optimally designed for exome sequencing data analysis. http://genomon.hgc.jp/
- GAMERA: an unstructured 3-D finite-element-based MPI-OpenMP hybrid seismic wave amplification simulation code Earthquake simulator. (FEM in unstructured and structured grid)
- NICAM: NICAM is a Nonhydrostatic ICosahedral Atmospheric Model (NICAM), used as a Global Cloud Resolving Model (GCRM). http://nicam.jp/hiki/
- LETFK: Local Ensemble Transform Kalman Filter
- NTChem: Comprehensive software for ab initio quantum chemistry calculations of large and complicated molecular systems. https://www.r-ccs.riken.jp/software_center/software/ntchem/overview/
- RS-DFT: Real-space finite-difference pseudopotential density-functional-theory code for first-principles material simulations on massively-parallel computers https://github.com/j-iwata/RSDFT
- FFB: FrontFlow/Blue is a general purpose FEM program that calculate incompressible unsteady flows in arbitrarily-shared geometries that may involve a moving boundary surface. http://www.ciss.iis.u-tokyo.ac.jp/english/dl/index.php
- Adventure: Computational Mechanics System for Large Scale Analysis and Design (unstructured grid). https://adventure.sys.t.u-tokyo.ac.jp/
- LQCD: Lattice QCD simulation (structured grid Monte Carlo), https://github.com/RIKEN-LQCD/qws/
- Cloverleaf: The CloverLeaf hydrodynamics mini-app solves Euler's equations of compressible fluid dynamics, under a Lagrangian-Eulerian scheme, on a two-dimensional spatial regular structured grid. The benchmark is an example of a stencil code and is classed as memory bandwidth-bound. Taken from UK-benchmark.
- OpenFOAM: OpenFOAM is a modular C++ framework aiming to simplify writing custom computational fluid dynamics (CFD) solvers
- FrontISTR: Open-Source Large-Scale Parallel FEM Program for Nonlinear Structural Analysis https://gitlab.com/FrontISTR-Commons/FrontISTR
- ABINIT: a software suite to calculate the optical, mechanical, vibrational, and other observable properties of materials. Starting from the quantum equations of density functional theory. https://www.abinit.org/
- SALMON: SALMON (Scalable Ab-initio Light-Matter simulator for Optics and Nanoscience) is an open-source computer

program for ab-initio quantum-mechanical calculations of electron dynamics at the nanoscale that takes place in various situations of light-matter interactions. https://salmon-tddft.jp/
- SEPCFEM3D: a 3-D spectral-element solver for the Earth. It uses a mesh generated by program meshfem3D. https://geodynamics.org/cig/software/specfem3d_globe/
- WRF: The Weather Research and Forecasting (WRF) Model is a next-generation mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications. https://www.mmm.ucar.edu/weather-research-and-forecasting-model
- MPAS: The Model for Prediction Across Scales (MPAS) is a coupled Earth system modeling package that integrates atmospheric, oceanographic and cryospheric modeling on a variety of scales from the planetary to regional and mesoscale/microscale. https://mpas-dev.github.io/
- Livermore Fortran Kernels: A set of loops which extracted from several numerical program, mainly used to evaluate the vector supercomputer. https://www.netlib.org/benchmark/livermore

ARTIFACT AVAILABILITY

Software Artifact Availability: There are no author-created software artifacts.

Hardware Artifact Availability: Some author-created hardware artifacts are NOT maintained in a public repository or are NOT available under an OSI-approved license.

Data Artifact Availability: There are no author-created data artifacts.

Proprietary Artifacts: There are associated proprietary artifacts that are not created by the authors. Some author-created artifacts are proprietary.

BASELINE EXPERIMENTAL SETUP, AND MODIFICATIONS MADE FOR THE PAPER

Relevant hardware details: A64FX, Intel Xeon, Cavium TX2

Operating systems and versions: Linux

Compilers and versions: Fujitsu compiler and Intel Compiler and LLVM/gcc for Arm

Applications and versions: described in our paper

Libraries and versions: described in our paper

Input datasets and versions: none