Received January 26, 2019, accepted March 8, 2019, date of publication March 18, 2019, date of current version April 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905302

A Novel DSP Architecture for Scientific Computing and Deep Learning

CHAO YANG[1], SHUMING CHEN[1,2], (Member, IEEE), JIAN ZHANG[1], ZHAO LV[1], AND ZHI WANG[1]
[1]College of Computer, National University of Defense Technology, Changsha 410073, China
[2]State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China

Corresponding authors: Shuming Chen (smchen@nudt.edu.cn) and Jian Zhang (jianzhang@nudt.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grant 61602493 and Grant 61572025.

ABSTRACT Exascale computing requires accelerators with ultrahigh power efficiency. Digital signal processors (DSPs), the most important embedded processors widely known for high power efficiency, are rarely explored in the HPC community. We propose a 64-bit general purpose DSP architecture, FT-Matrix2000, which not only integrates the main features of DSPs but also presents several novel enhancements for scientific computing. The FT-Matrix2000 architecture comprises multiple FT-Matrix2 cores and optional RISC CPU cores. The FT-Matrix2 core utilizes a VLIW+SIMD architecture, provides support for double precision operations, and optimizes both the data and control path for scientific computing. Our evaluations show that the performance and efficiency of FT-Matrix2000 are 1107GFLOPS and 92.25%. Compared with the MIC and a 40nm process GPU, FT-Matrix2000 improves the GEMM power efficiency with a factor of 1.49 and 2.68, respectively. We build up a prototype supercomputer with FT-Matrix2000/12. Its HPL efficiency achieves 62.2%, and the performance power ratio is 5.33 GFLOPS/W, which can rank the fourth in the latest Green500 list. These results validate that the FT-Matrix2000 architecture is suitable for scientific computing while maintaining the efficiency of signal processing well. Moreover, the enhancement of FT-Matrix2000 in vector and matrix related computations also enable it to efficiently support deep learning related applications. We have implemented some typical DCNN models on FT-Matrx2000, NVIDIA GPUs, and Vision P6 DSP. The experiments demonstrate that the average computation efficiency of the proposed architecture based on Matrix2000 is about 20 $\sim$ 35% and 8% higher respectively than GPUs and Cadence Vision P6 DSP.

INDEX TERMS DSP, architecture, scientific computing, HPC.

I. INTRODUCTION

Exascale computing requires designs with ultrahigh power efficiency. An efficient way to continuously improve power efficiency is deploying heterogeneous architectures with hardware accelerators. By 2016, nine of the top ten supercomputers in the green500 list [1] and three of the top ten supercomputers in the top500 list [2] utilize special accelerators, including x86 based many-core co-processors and GPGPUs. However, the most important embedded processors, digital signal processors (DSPs), are rarely explored in the high-performance computing (HPC) community, even though they are widely known for high power efficiency [3]. We carry out this study to validate whether a DSP is efficient for scientific computing.

The associate editor coordinating the review of this manuscript and approving it for publication was Songwen Pei.

Early DSPs were not considered for scientific computing, because they did not support floating point operations. Nowadays, floating point functional units are essential for high performance DSPs [3]. Also, compared with x86 based

CPU and GPGPU co-processors, the DSPs have multiple advantages for scientific computing. First, DSPs have a high power performance ratio. This is very appealing since the power consumption has already become as important as, if not more than, the performance for exascale computing. Second, several features of DSPs (such as the scratchpad memory and imprecise interrupt) ensure real-time and predictable execution and improve the computation efficiency. For example, the explicitly managed scratchpad memory enables programmers to easily hide the data movement latency behind the computation latency [3].

A preliminary exploration with the TI C6678 showed excellent promise of applying DSPs into scientific computing [4]. Yet, C6678 is mainly designed for signal processing; it lacks important features for scientific computing. For example, its peak double precision performance is only 32GFLOPs, which is quite limited for scientific computing. Also, its GEMM efficiency is only 57%, which is far less than the efficiency of other processors like CPUs or GPGPUs.

Instead of experimenting with existing DSPs, we propose a novel architecture, the FT-Matrix2000 general purpose DSP architecture, for both signal processing and scientific computing. FT-Matrix2000 preserves main features of DSPs, including low power consumption and real time processing. Also, it integrates several novel enhancements for scientific computing. FT-Matrix2000 comprises multiple FT-Matrix2 cores and optional RISC CPU cores. To provide high performance for signal processing, the FT-Matrix2 core utilizes a VLIW+SIMD architecture. It contains a scalar processing unit (SPU), a vector processing unit (VPU) with 16 vector processing elements (VPE), and the on-chip scratchpad memory with both vector memory (VM) and scalar memory (SM).

There are four key features of the FT-Matrix2000 architecture: First, both the data path and control path of FT-Matrix2 core are highly optimized for scientific computing. The widths of data path and general purpose register are increased to 64 bits to support double precision operations. Rich arithmetic types, including division, transcendental and reciprocal operations are supported. Overall, the peak double precision performance of a FT-Matrix2 core is 100FLOPs each cycle.

Second, FT-Matrix2 core provides high bandwidth and low latency memory access. The VM provides a 4096-bit wide data path for the VPU; each VPE can load or store four double precision operands simultaneously. To further improve the bandwidth, scalar broadcast is supported to broadcast data from the scalar register of SPU into the vector register of VPU.

Third, FT-Matrix2000 proposes a memory-side shared global cache (GC) to reduce off-chip DRAM access latencies. GC allows users to change the address interleaving mode to achieve configurable share/private partitions. It also allows zero-reuse data to bypass the GC to prevent cache pollution. Moreover, the GC applies a priority replacement policy to extend the stay for high-reuse data.

Fourth, FT-Matrix2000 deploys a direct memory access (DMA) controller, which can transpose matrix during data movement for efficient scientific computing. It also integrates novel designs to improve the DRAM access locality and SIMD occupancy.

To ease the use of the FT-Matrix2000 chip, we build a complete set of programming tools, including the compiler, assembler and linker, for the FT-Matrix2000. The compiler is highly optimized to efficiently support both VLIW and SIMD paradigms. Several parallel support libraries, including the BLAS, FFT and OpenGL library, are developed for high level applications.

The FT-Matrix2000 architecture can be implemented in two varieties: an accelerator and a host processor. When FT-Matrix2000 works as an accelerator, multiple FT-Matrix2 cores link external off-chip RISC CPU cores with a high speed I/O bus (i.e. PCIe). In the host variety, multiple FT-Matrix2 cores and on-chip RISC CPU cores are tightly coupled and share a single memory space, and FT-Matrix2000 can work like a computer.

We fabricated the FT-Matrix2000 architecture as a scientific computing accelerator on a 40nm process. The chip achieves 1GHz frequencies. Our experiments are conducted in two modes: in a testing board and in a prototype supercomputer with 160 Tflops peak performance. For the testing board, the performance of FFT is about 13.6 times of that of TI C6678. The performance and efficiency of GEMM are 1107 GFLOPS and 92.25% respectively. Compared with the MIC and a 40nm process GPU, FT-Matrix2000 improves the GEMM power efficiency by a factor of 1.49 and 2.68 respectively. For the prototype supercomputer, the HPL efficiency is 62.2%, and the performance power ratio is 5.33 GFLOPs/W, which can rank the fourth in the latest Green500 list. These results demonstrate that FT-Matrix2000 is an efficient architecture for both scientific computing and signal processing.

Deep convolutional neural network(DCNN) has become a successful and popular algorithm in the age of artificial intelligence [5]–[8]. DCNN algorithm could achieve state-of-the-art performance in regions such as object detection, image classification and so on [9]–[14]. The performance improvement of DCNN comes at the cost of huge amount of computations. Due to such computation pressure, it is necessary to accelerate DCNN based on hardware accelerators. And the FT-Matrix2000 is suited for accelerating DCNN because FT-Matrix2000 is efficient in vector and matrix related computations, which is the kernel computation pattern of manly deep learning applications. We tested some typical DCNN models based on FT-Matrix2000 and other hardware platforms, and the results show that FT-Matrix2000 provides high performance on DCNN models. These results demonstrate that FT-Matrix2000 is an efficient architecture for deep learning.

This paper makes the following main contributions:

- We propose FT-Matrix2000, a novel 64-bit DSP architecture with TFLOPs performance. The performance of FT-Matrix2000 can easily satisfy the requirement of scientific computing and future signal processing applications.
- We build a 160 Tflops prototype supercomputer with FT-Matrix2000, which achieves high efficiency and high performance power ratio in typical HPC benchmarks. This demonstrate that FT-Matrix2000 is an efficient architecture for scientific computing.
- We also show that FT-Matrix2000 is a promising architecture for deep learning applications like DCNN, compared with GPGPU and traditional DSPs, FT-Matrix2000 can improve the computation efficiency

of GPGPU and DSPs by 20 $\sim$ 35% and 8% respectively.

II. FROM SIGNAL PROCESSING TO SCIENTIFIC COMPUTING AND DEEP LEARNING

A. FEATURES OF A DIGITAL SIGNAL PROCESSOR

Traditionally, it is generally believed that the main architecture differences between a DSP and a general purpose CPU lie in the separate buses between the instruction and data (Harvard architecture), the hardware multiplier and the dedicated address generation unit, etc. However, with the rapid evolution of the processor architecture, most of the above differences have faded away. Modern DSP and CPU architectures have many homogeneous characteristics in the calculation and control logic. Yet, we still find out that the followings are the basic features of current DSPs, which are different from CPUs:

1) Targeting embedded applications, DSPs are more power efficient and with stronger emphasis on real-timing. DSPs adopt series of structures to reduce power and enable the timing certainty, such as static parallelism exploration, the imprecise interrupt, the scratchpad memory, software cache coherence, direct high-speed IO interface transmission, etc.
2) The machine word of DSPs is usually less than or equal to 32 bits, while 32-bit and 64-bit CPUs are both very common. Indeed some DSPs like TI TMS320C6678 [4] can support the 64-bit calculation indirectly, which is so called a pseudo 64 bits calculation since its data path and the general registers are still 32 bits.
3) DSPs typically adopt the signal processing application oriented instructions, such as the dot product, the complex multiplication, or Galois field multiplication instructions. CPUs usually do not support such operations.
4) DSPs usually have a host port, through which, a host, such as a PC, can fully control the status of a DSP, including its internal memory/register resources. However, CPUs generally do not contain the host port.

B. LIMITATIONS OF CURRENT SCIENTIFIC COMPUTING PROCESSORS

Scientific computing refers to using of computers to deal with the mathematical calculations in the scientific research and engineering technology. Currently, super computer systems usually adopt the co-processors to carry out the key operations in the scientific computing.

We believe that the typical scientific computing co-processor should have the following characteristics. 1) Its computation precision, the peek computation ability and the power efficiency should be high. Most of the current co-processors have the peak double-precision performance of 1.0+ TFLOPS and the power efficiency of 5.0+ GFLOPS/W. 2) It should have the complete data type, which means including all the data types from 8-bit to 64-bit, numerical or logical, fixed or floating point even complex data types. The data structures include a single data, a vector and matrix data, or array type of them. 3) It should have abundant operation types. In addition to the general arithmetic and logical operations, it also need to have the MAC, dot product operations or other elementary functions, such as the square root or the exponential operation, etc. 4) It should optimally organize the memory and data path to have a smooth data bandwidth supply from inner chip to outside chip.

Currently, the co-processors used in scientific computing are mainly evolved from two kinds of processors: one is the many-core based on x86, such as Knights Corner (MIC) of Xeon PhiTM from Intel, and the other is GPGPUs, such as the Tesla from NVidia, the FirePro S from AMD. Although these co-processors have brought significant computing ability improvement to the super computer systems, they still have some defects: 1) the power efficiency is relatively low. MIC is so strong in general control, and not enough in terms of the calculation ability expansion. GPGPUs have the functional redundancy in graphic processing. 2) They still cannot ensure deterministic task execution time. MIC and GPGPUs generally use the cache and hardware cache coherence. The programmers may feel convenient to use since the detail of the memory access is transparent to them. However, this brings the non-deterministic memory access, which will lead to the mapping efficiency of an algorithm can hardly be further improved.

C. USING DSPs FOR SCIENTIFIC COMPUTING AND DEEP LEARNING

A current DSP architecture has obvious defects in terms of scientific computing. 1) 32-bit width of address and data path is not wide enough, which means lack of computation accuracy and memory space. 2) The peak computation ability and data types of a current DSP is insufficient to support scientific computing. 3) For embedded applications, current DSPs mainly rely on the large on-chip memory to ease the pressure of external storage bandwidth. Its ratio of peak performance and peak external storage bandwidth is too high, which is not suitable for scientific computing. 4) Although the latest DSP has begun to consider control enhancements, current DSPs are still deficient to support the task management, process scheduling, interrupt management, and OS. 5) Current DSPs lack of a unified high-level programming model to manage different kinds of parallelism, such as multi-core, VLIW, vector and SIMD. These problems seriously limit DSPs to be a co-processor of scientific computing.

It is both innovative and challenging to use DSPs as scientific computing co-processors. This paper extends the DSP to the field of scientific computing and proposes a novel DSP through a list of architecture improvements. The proposed novel DSP architecture includes three aspects. First, it retains the basic DSP characteristics excepting the machine word length of 32 bits. Second, it extends several features in computation, memory and control structures for scientific computing. Third, it has the corresponding advanced

TABLE 1. Instruction set.

	Issue Slot	Main Functions
Scalar Instruction Issue Slot	1. Flow Control	scalar branch, vector branch, wait, nop, etc.
	2. Scalar Load/Store	scalar load/store of half/one/double/quad word with linear or circular addressing
	3. Scalar MAC1	basic operation (+/-, *, /), FMA, dot, complex multiplication, square root, elementary functions (sine/cosine/exp/log), format conversion, floating-point logic ops etc.
	4. Scalar MAC2	
	5. Scalar BP	fixed-point +/-, shift, test (==, !=, >, <, etc), logical ops, bit ops, broadcast ops, etc
Vector Instruction Issue Slot	6. Vector Load/Store 1	(16x) vector load/store of half/one/double/quad word with linear or circular addressing.
	7. Vector Load/Store 2	
	8. Vector MAC1	(16x) vector basic operation (+/-, *), FMA, dot, complex multiplication, data format conversion, floating-point logic ops etc.
	9. Vector MAC2	
	10. Vector MAC3	
	11. Vector BP	(16x) vector fixed-point +/-, shift, test, logical ops, bit ops, , shuffle, reduction, etc

programming model and software environment. These characteristics of the novel DSP architecture also make it suitable for applications beyond scientific computing, such as deep learning applications.

III. THE ARCHITECTURE OF FT-MATRIX2

As shown in Figure 1, FT-Matrix2 core deploys VLIW+SIMD architecture with tightly coupled scalar and vector processing units. The instruction dispatcher can issue 5 scalar instructions to SPU and 6 vector instructions to VPU simultaneously every cycle. Table 1 shows the brief information for each instruction issue slot. Figure 2 shows an example of memory mapping of 5 parallel instructions, which can be issued at the same cycle.

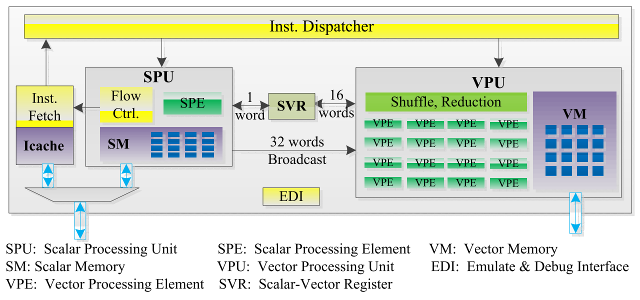
Inst. Dispatcher
SPU
Inst. Fetch
Flow Ctrl.
SPE
Icache
SM
1 word
SVR
16 words
32 words Broadcast
VPU
Shuffle, Reduction
VPE
VM
EDI
SPU: Scalar Processing Unit
SM: Scalar Memory
VPE: Vector Processing Element
SPE: Scalar Processing Element
VPU: Vector Processing Unit
SVR: Scalar-Vector Register
VM: Vector Memory
EDI: Emulate & Debug Interface

FIGURE 1. The block diagram of the FT-Matrix2 Architecture.

The SPU in FT-Matrix2 is composed of SPE, SM and flow control. SPE is in charge of scalar arithmetic and logical operations, and SM supplies data for SPE. VPU has 16 homogeneous VPEs. Shuffle network is used to exchange data among VPEs, and supports arbitrary permutation mode. The dedicate vector memory in VPU provides data for 16 VPEs. SVR is a special register group, which supports data exchange between SPU and VPU at register level. To SPU, SVR is a group of scalar registers. To VPU, SVR is a vector register. EDI is an emulate and debug interface, which provides a channel accessing internal register/memory resources for external host.

A. ISA DESIGN FOR 64BITS OPERATION

FT-Matrix2 uses an expanded 40/80-bit instruction set to improve the processing efficiency of double-floating data. Figure 2 shows an example of instructions with different instruction lengths. For processing double-floating data efficiently, the width of scalar general purpose registers (GPRs) in SPU is set to 64-bit, and that of vector GPRs in VPU is 16$*$64-bit. Both scalar GPR count and vector GPR counts are designed as 64 to fully feed up MAC pipeline. FT-Matrix2 provides 16 base address registers and 16 address offset registers, which are used by SPU and VPU to generate scalar and vector data addresses respectively. The width of these registers is 36-bit to enable large scale data processing. FT-Matrix2 optimizes instruction issue slots, and increases instruction issue slots from 10 (FT-Matrix) to 11, to support the increased computing density.

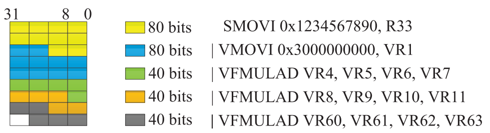
31
8
0
80 bits
SMOVI 0x1234567890, R33
80 bits
| VMOVI 0x3000000000, VR1
40 bits
| VFMULAD VR4, VR5, VR6, VR7
40 bits
| VFMULAD VR8, VR9, VR10, VR11
40 bits
| VFMULAD VR60, VR61, VR62, VR63

FIGURE 2. Instruction packet in memory.

B. ENHANCED ARITHMETIC ABILITY

FT-Matrix2 enhances the computational accuracy, operation type, and computational capability for both DSP applications and high-performance scientific applications.

1) ENHANCEMENT IN COMPUTATIONAL ACCURACY

The FT-Matrix2 instructions (except divide, square root, and basic elementary functions in SPE), with full pipeline implementation in MAC unit and Bit Processing (BP) unit, are designed to support scientific computing, especially compatible with IEEE754 double-precision floating-point standard. The bandwidths of register file and execution pipelines are all 64 bits. Meanwhile, highly hardware reuse MAC structure is designed to reduce the overhead and improve computation efficiency. The data path of double-precision floating-point operations is extended to support 64-bit fixed-point, 32-bit SIMD fixed-point and single-precision SIMD floating-point operations. A 64×64 multiplication is reused for 64-bit fixed-point MAC operations and the 53×53 mantissa multiplication for floating point MAC operations and reduced the overhead about 41%.

2) ENHANCEMENT IN OPERATION TYPE

As shown in Table 1, the FT-Matrix2 operation types are enhanced in the following aspects. First, low latency double-precision floating point FMA (fused multiply-add), multiplication, addition /subtraction with dual-path scheme [15] is presented and the execution cycles of these operations are 6, 4, 3, respectively. Second, basic elementary function unit, implementing floating-point divide and square root based on SRT-8 [16] and sine, cosine, exponential, and logarithm based on CORDIC algorithm [17], are integrated in SPE with iteration mode to efficiently calculate complex functions in scientific applications, such as Linpack and OpenGL. Finally, basic DSP operations, such as single-precision/32-bit dot and complex multiplication operations, are implemented with the extension of double-precision/ 64-bit data path to improve the performance of FT-Matrix2 for DSP application.

3) ENHANCEMENT IN COMPUTATIONAL CAPABILITY

In order to provide the highest peak performance and efficiency of computation structure for scientific applications, as many MAC units as possible should be integrated. However, the number of MAC units in each VPE is limited with the number of Read/Write ports and the size of register file. Thus, three MAC units are equipped in each VPE and execute three FMA operations in parallel at each cycle. Meanwhile, in order to improve the performance for DSP applications, the data path in double-precision/64-bit pipeline is reused to implement SIMD single-precision/32-bitoperations.Three execution pipelines, two for MAC units and one for BP unit, are included in SPE. And, four execution pipelines, three for MAC units and one for BP unit, are included in VPE. Thus, 50 MAC operations can be calculated simultaneously at each cycle in FT-Matrix2 and the peak performance of double precision and single-precision floating-point can reach 100 GFLOPS and 200 GFLOPS, respectively, running at 1 GHz.

C. OPTIMAL MEMORY ORGANIZATION

FT-Matrix2 can afford high-bandwidth memory access for scalar and vector arithmetic units. It has two kinds of memories: 1) Scalar memory (SM), which can be configured as L1 data cache or scratchpad memory. 2) Vector Memory (VM).

When SM is under L1 data cache mode, SPU can access external memory space through Global Cache (GC). The programmable L1D Cache flush registers are designed for user to maintain the cache coherence. While SM is under scratchpad memory mode, SM has its own addressing space. Except for accessing SM, the scalar load/store instruction can also access the peripheral space to configure the peripherals.

Vector memory supports four way parallel accesses, i.e. two vector load/store instructions, one DMA read channel, and one DMA write channel. It can afford 64 words/cycle data bandwidth for 16 VPEs at most.

Considering the frequent shuffle operations in the FFT algorithm with Binary-Exchange mapping method, besides supporting general aligned or unaligned continuous linear addressing modes, VM also provides six special vector access instructions with shuffle operations. These instructions can totally eliminate the shuffle operations, bring performance improvement and code size reduction for DIF/DIT FFTs. Compared the prior work [18], these instructions are suitable from radix-2 FFT to radix-2/4 FFTs.

D. IMPROVING THE CONTROL LOGIC FOR OS

As shown in Figure 1, FT-Matrix2 provides EDI to exposure all architecture memory (SM and VM) and registers (scalar/vector GPRs, states/configuration registers, etc.) to external host (a CPU on/outside chip). By EDI, the host can accomplish program/data offload, startup or pause FTMatrix2 core, etc.

Exception report/handle, privilege management and MMU are introduced in FT-Matrix2, and these mechanisms are used to support operating system. More than 10 kinds of events, such as divided by zero, addressing overstep, synchronization timeout, etc., will generate exception. These exceptions can be handled by FT-Matrix2 itself, or just report to CPUs. FT-Matrix2 supports two privilege levels: the supervisor level and the user level. In addition, FT-Matrix2 uses a segment-based MMU scheme to provide memory protection based on privilege levels. Based on this scheme, FTMatrix2 supports addressing overstep exception for program fetch, data load/store, DMA data moving. Once such violation is happened, the memory operation is canceled, and an overstep exception is reported.

E. GEMM MAPPING BASED ON FT-MATRIX2

The general matrix multiplication (GEMM) plays an important role in scientific computing. It occupies more than 95% of the execution time for the Linpack [19]. Here, we take GEMM as an example to explain how to map an algorithm on FT-Matrix2.

As shown in Figure 3, we place data of 16 × 16 matrix C, A and B on SM in SPU and VM in VPU respectively.

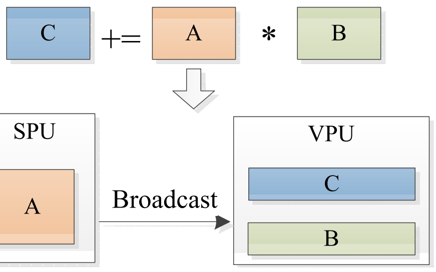

FIGURE 3. Matrix data placement on FT-Matrix2.

For simplicity, Figure 4 only shows operations on one MAC unit, but it is easy to extend the work-flow to 3 MAC units.

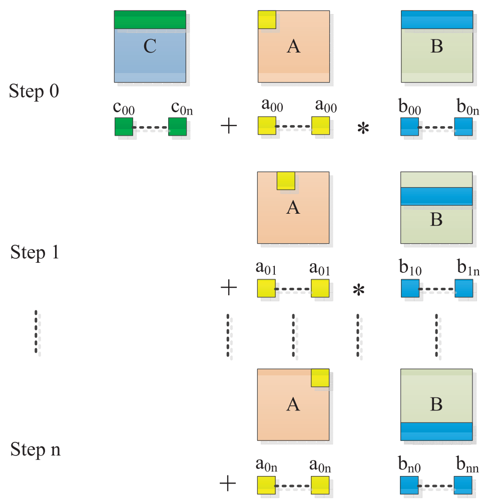

FIGURE 4. GEMM algorithm mapping on FT-Matrix2.

At step 0, FT-Matrix2 executes the flowing operations:
1) vector load c_0 row of matrix C;
2) vector load b_0 row of matrix B;
3) scalar load a_{00} of matrix A, and broadcast it to VPU;
4) vector MAC of $c_0 + b_0 * a_{00}, a_{00}, \ldots, a_{00}$;
At step 1, FT-Matrix2 executes the flowing operations:
1) vector load b_1 row of matrix B;
2) scalar load a_{01} of matrix A, and broadcast it to VPU;
3) vector MAC of result of step1 $+b_1 * a_{01}, a_{01}, \ldots, a_{01}$;
At step *n*, FT-Matrix2 get partial results $(c_0 + a_0 * \text{B})$ of matrix multiplication $\text{C} + \text{A} * \text{B}$, i.e. new c_0.

IV. FT-MATRIX2000 ARCHITECTURE

As shown in Figure 5, FT-Matrix2000 is a DSP-based heterogeneous multi-core architecture, including multiple FT-Matrix2 DSP cores and multiple CPUs (optional). Two execution models are supported: accelerator mode and host mode.

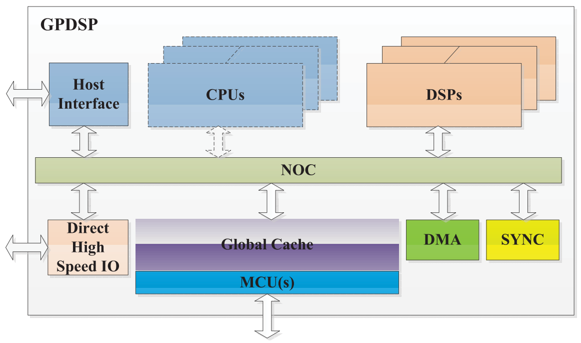

FIGURE 5. FT-Matrix2000 Multi-Core Architecture.

Under the accelerator mode, the off-chip CPUs are masters, and FT-Matrix2000 is an accelerator. CPUs can accomplish program/data offload, startup or pause each FT-Matrix2 core, etc. FT-Matrix2 cores can send interrupts to CPUs to report specific change of task states, and send exceptions, including address overstep, zero division, etc. to report any abnormal operation.

Under the host mode, CPUs and FT-Matrix2 cores are tightly coupled with fine grain control and data transmission, and all CPUs and FT-Matrix2 cores share data by global cache on chip. All CPUs and FT-Matrix2 cores can access hardware synchronization unit (SYNC), which implement read/write lock and barrier, to enable data consistence among processors. CPUs and FT-Matrix2 cores boot from external memory device and run tasks independently.

A. SUPER NODE AND INTERCONNECTION

As shown in Figure 6, the interconnection deploys n nodes to composite a ring architecture. A super node, which contains FT-Matrix2 cores (or CPUs, high-speed peripheral devices) and a sub-bank of global cache, communicates with the other nodes through a network interface (NI), where requests arbitration occurs and interconnection transactions arose.

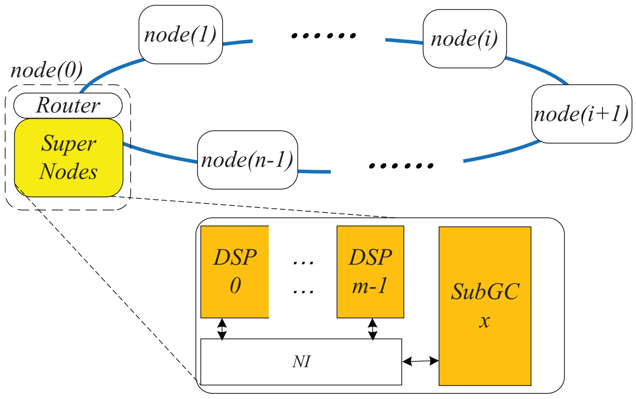

FIGURE 6. TOP Structure and ring topo logic.

1) MULTI-LAYER LINKS AND HIGH-BANDWIDTH NoC

There are multiple-layer physical links of matched width for different transactions, such as command transactions, configuration transactions and data transactions. A 512-bit link for data transactions in each direction makes the bisectional bandwidth in this design is essentially considerable, which can efficiently meet the communication and data feeding requirements of DSP nodes.

2) BIDIRECTIONAL NoC

Some links run clockwise and the others counter-clockwise for the transactions of the same function. And more than one transaction can be carried out simultaneously in the same layer if they occupy different segments of the ring links. It can efficiently reduce the average hops and improve the throughput and traffic jam.

3) BUFFERLESS ROUTERS AND EXPRESS NoC

Bufferless-router and unblocking for transactions on the ring enable the realtime transmission. A transaction requests to the router from NI can be acknowledged only when the destination is available and the next links are available.

Besides, special physic design skills and efforts are made to ensure the exact one-cycle hop-delay. All those design considerations and implementation effort came out a reward of a low-cost and traffic-efficient ring interconnection.

B. GLOBAL CACHE

GC is the Last Level Cache (LLC) of FT-Matrix2000. It locates between the Ring and MCU. GC accelerates the access of DDR for Icache, Dcache, DMA and other devices, and supports the ECC (correct 1-bit error and check 2-bit error) of the data bank. Beside, GC provides a series of flush operations for programmers to support the software cache coherence, avoiding the timing indeterminacy of hardware cache coherence. GC adopts the distributed design policy. Each MCU has two corresponding Sub of GC (SubGC). SubGC has the characteristics of high bandwidth (1024bit/cycle), high capacity (512KB).

GC adopts three innovative strategies to deal with the more complicate trend of data access since the bigger proportion of the streaming and jumble data access under the scientific computing and embedded applications.

1) CONFIGURABLE SHARE/PRIVATE CACHE PARTITION

On the basis of the shared cache, GC provides programmers with control registers to change the high-low order interleaving mode of DDR space to achieve the configurable share/private cache partition. Figure 7 takes four DDRs as an example, the DDR space is divided into two parts, the upper part is interleaved on high-order address bits, and the lower part is interleaved on low-order address bits. The entire DDR space can be divided into 8 ratios between high-order and low-order interleaving spaces. Assuming the size of the DDR space is S, then the range size of high-order interleaving space can be S, S/2, S/4, S/8, S/16, S/32, S/64, and S/128, and the remaining space is low-order interleaving pace.

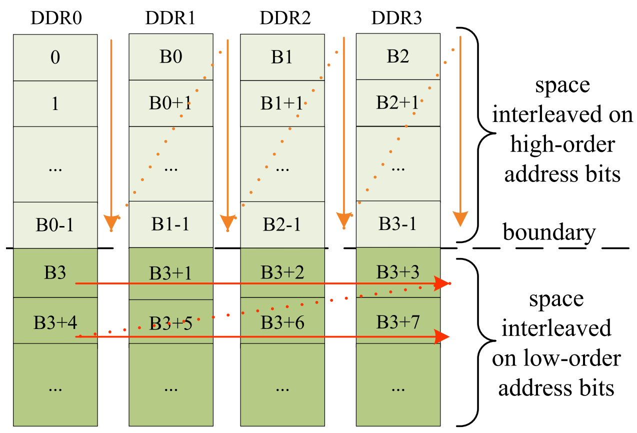

FIGURE 7. High-low order interleaving mode of the DDR space.

2) CONFIGURABLE BYPASS CACHE POLICY

Zero-reuse data in the GEMM can seriously damage the access of high-reuse data, which will cause great loss of the performance. To address this issue, we provide the configurable bypass cache policy for users to set the zero-reuse block to not be located in GC. We add additional bypass-GC page attribute and design two separated path-ways from the request sources to the data return path, which can totally isolate the bypass-GC requests and resident-GC requests.

3) PRIORITY REPLACE POLICY

Beside the zero-reuse block, the low-reuse block will also bring the jolt to the LLC. To let the high-reuse data reside in the GC as long as possible, GC adopts the priority replace policy, which includes the additional priority page attribute and hardware for tree-LRU with priorities. The data block with the high replacement priority can't be replaced by the data block with low replacement priority. Programmers can also configure the control register to clear the replacement priorities in all the GC cache lines.

The three strategies mentioned above are complementary to each other. In our evaluation, the execution time of GEMM and FFT for designs with these strategies decreased to about one-twelfth and one-fifth of the originals, respectively.

C. DMA

The Direct Memory Access (DMA) controller is the central module for data movement. Its basic function is to perform a point to point (P2P) transmission between a local memory (SM and VM) and a global memory (GC and DRAM) for a 2D data array. A 2D array comprises several frames, each of which has multiple words. The DMA controller incorporates several novel features to improve performance for scientific computing.

1) FEATURES TO IMPROVE THE MATRIX TRANSPOSE EFFICIENCY

The matrix transpose is one of the most important kernels of several scientific and engineering applications [19]–[21], [24]. A traditional design first moves the source matrix into the local memory, and then uses the load/store and functional units to fulfill the transpose operation. This not only increases instruction overheads, but also consumes a significant amount of memory bandwidth. To overcome these limitations, the DMA controller deploys the matrix transpose (MT) transmission to transpose the matrix while moving the data.

The key component for MT transmission is an 8×8 matrix transpose register group (MTRG). As shown in Figure 8, when all 8 rows are written, the DMA controller read the MTRG column by column. This transposes an 8×8 matrix. Transposing a large matrix is performed by dividing it into multiple 8×8 blocks and then transposing these small blocks. We use the double buffering mechanism to further improve performance with two MTRGs. When one MTRG conducts write operations, the other one conducts read operations, and vice versa. Compared with the traditional design, the MT transmission achieves a 7.53 times speedup for transposing a 256KB matrix.

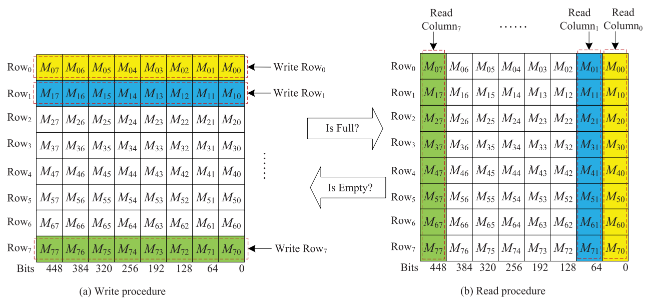

FIGURE 8. The DMA controller writes the MTRG in rows and reads the MTRF in columns.

2) FEATURES TO IMPROVE THE GEMM EFFICIENCY

The DMA controller optimizes the data movement for the widely used GotoBLAS GEMM implementation, which divides a matrix into several partitions, and distributes them to different cores for parallelization [22], [23]. Figure 9 illustrates an example layout of 12 continuous partitions for 12 cores. Each partition comprises 512 48-word frames. The same frames of 12 partitions are in the one page, while different frames are in different pages. We proposed optimizations to improve the page access locality. The DMA controller moves data in a frame-major manner. Since the same frames of different partitions are in the same page, they achieve high page access locality.

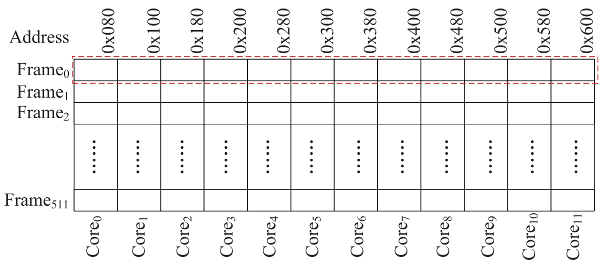
Address
0x080 0x100 0x180 0x200 0x280 0x300 0x380 0x400 0x480 0x500 0x580 0x600
Frame0
Frame1
Frame2
Frame511
Core0 Core1 Core2 Core3 Core4 Core5 Core6 Core7 Core8 Core9 Core10 Core11

FIGURE 9. An example layout of 12 continuous partitions.

D. IO INTERFACE AND SYNCHRONIZATION

FT-Matrix2000 uses host interface to enable accelerator mode, under which one or multiple external hosts manage all resources within a processor of FT-Matrix2000 architecture. Host interface should be fast enough to overlap data transmission time and data processing time. Currently, PCIe is very suitable to be a host interface. In fact, we develop a dedicate PCIe driver under Linux operating system. The driver enables the host to control the whole workflow of the accelerator, such as program and data offload, start up or pause the execution of each core. In addition, FTMatrix2000 uses direct high speed IO to enable direct data transmission among multiple processors, which can greatly reduce the communication delay between hosts and accelerators. FT-Matrix2000 uses a hardware synchronization unit with hardware locks and barriers to support the synchronization efficiently among multiple cores.

V. SOFTWARE AND PROGRAMMING ENVIRONMENT

A. COMPILER FOR A SINGLE CORE

To ease the use of FT-Matrix2000, we provide a three level programming model shown in Figure 10. The top level supports users to program in standard C, which can reduce application's development period. FT-Matrix2000 C Compiler can dig the vector data and produce vector instructions automatically. To improve performance, the middle level also provides Vector C programming. Users can use Vector C provided by FT-Matrix2000 compiler to gain more vector performance. Vector C is implemented in FT-Matrix2000 compiler based on intrinsic technology. If users want to gain higher performance, they can also program in FT-Matrix2000 assemble language, as shown in Figure 10.

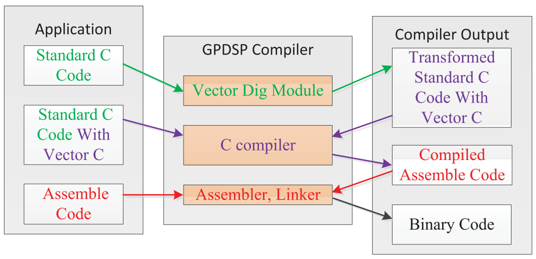

FIGURE 10. FT-Matrix2000's three-level programming model.

B. MPI PROGRAMMING ENVIRONMENT

With increasing core numbers, it is very urgent to support multi-core application developments for FT-Matrix2000. FTMatrix2000 MPI is ported on MPICH2.0. We choose MPI as multi-core FT-Matrix2000 application development environment, because:

- MPI is a mature stand: MPI stand is widely accepted in super computer;
- MPI demands less hardware support: MPI does compel FT-Matrix2000 to support Cache coherence;
- It is very simple for users to learn;
- MPI is suit for bigger granularity parallel applications.

Users can configure different MPI running models according to applications. In host mode, the usual models are as Figure 11(A). CPU offloads applications on DSP, and different DSPs communicate by MPI. The mutex of multi-threads is implemented on barrier and lock provided by FT-Matrix2000, and the data share is achieved by global memory.

In accelerator model, CPU nodes in FT-Matrix can be configured as a MPI node as shown in Figure 11(B) CPU nodes is in the same level of DSP nodes.

C. MATHEMATICS LIBRARY

Mathematical libraries on top of the FT-Matrix2 runtime include Basic Linear Algebra Subprograms (BLAS) libraries and Fast Fourier Transform (FFT) libraries. The BLAS libraries are collections of functions with standard interfaces

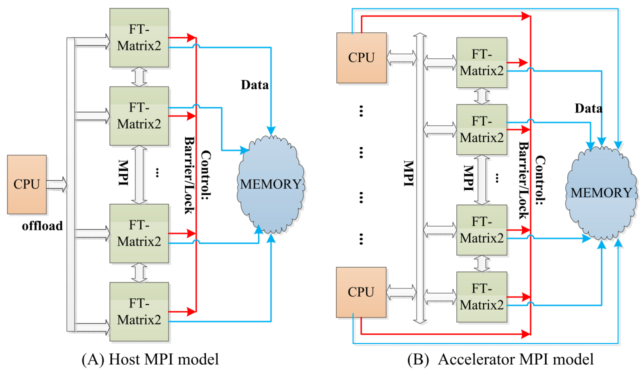

FIGURE 11. Host MPI model and accelerator MPI model.

that are used for basic linear algebra operations on vectors and matrices. They are frequently used in high-performance computing, especially when a large variety of vector and matrix operations are required. The FFT libraries include Radix-2,4 Decimation-In-Time FFT and Radix-2,4 Decimation in Frequency FFT, that are algorithms widely used today in science, engineering and digital signal processing.

OpenGL2.0 (Open Graphics Library) is a programmable graphics standard supported by FT-Matrix2. FT-Matrix's architecture is very suitable for OpenGL. OpenGL2.0 libraries are implemented based on vector instructions and multi-cores to achieve high performance.

D. OS

uClinux [32] has many merits in embedded field including highly optimized performance, highly compressed code and good portability. We ported uClinux 2.6.34 stably on DSP core FT-Matrix2. The binary code compiled by the compiler is 1MB. In the same time, we also port UCOS [25] which supports multi-process management based on time slide rotating. UCOS is a lightweight operation system, which is suitable for embedded system. Users can achieve multiprocess management based on UCOS.

VI. CHIP DESIGN AND PERFORMANCE TEST

A. CHIP DESIGN

Based on FT-Matrix2000 architecture, we implement FTMatrix2000 at 40nm CMOS process, which has 12 FT-Matrix2 cores, 1GHz clock frequency, and achieving 1.2 TFlops double-precision peak performance.

FT-Matrix2000 comprises 6 DSP nodes and 2 IO nodes which are connect by a non-blocking bi-direction ring interconnection. This interconnection consists of 8 routers, each of which interfaces the nodes, global cache or the IO peripheral equipment.

FT-Matrix2000 has 4 64-bit memory interfaces with 8-bit ECC and provides 128GB storage space. The DDR3 space is shared among all the masters, and accessed by the mechanism as shown in Figure 7.

FT-Matrix2000 takes one PCIe 2.0 16x interface as host interface, which provides 80Gbps peek bandwidth. We develop a dedicate PCIe driver under Linux operating system. By this driver, the host can control the whole work flow of the accelerator, such as program and data offload, start up or pause the execution of each core. In addition, FT-Matrix2000 has two SRIO1.4 4x interfaces, which are used for peer-to-peer communication among multiple chips. Each SRIO interface provides 12.5Gbps bandwidth.

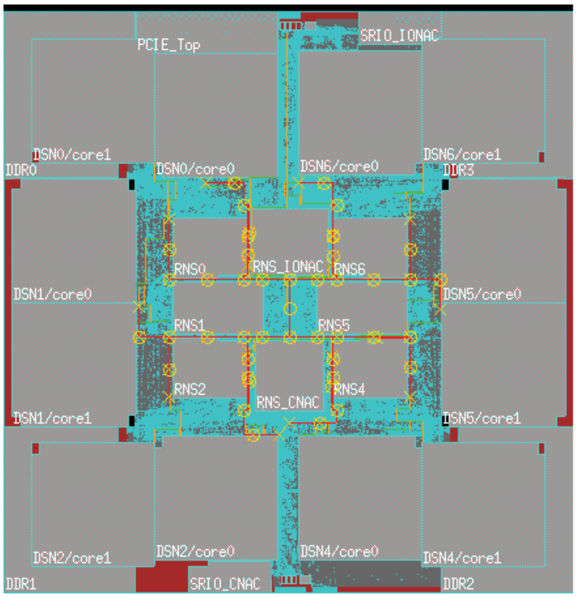

FIGURE 12. Floorplan of FT-Matrix2000 chip.

FT-Matrix2000 is designed at 40nm CMOS technology. There are more than 80M standard cells, 116 Mb on-chip memories, and 804 signal IOs. Figure 12 is the final floorplan. 8 ring nodes locate at the center of the chip. 12 DSP cores are placed around the ring nodes. The spaces between ring nodes and DSP cores are used for top level interconnect. The longest distance between DSP core and ring node is 7000 um. Because there are more than 20000 interconnect between FT-Matrix2 cores and the ring nodes, the timing between them is very critical. Bus routing techniques are used to solve these interconnect timing closure problems.

There are more than 29,000 interconnects between VPE, VM, and SC in each FT-Matrtix2 core. So the biggest challenge is to decrease the length of global interconnects and avoid routing congestion. The size is 5.17mm $\times$ 5.13mm. VPEs, SC, and VM are placed near the four sides.

Each VPE has 64 64b general purpose registers, which are divided into two register files with same structure. Each register file has 32 registers, 8 write port and 13 read ports. Figure 13 (a) is the layout of the register, with area of 402um $\times$ 174um. Figure 13 (b) is the circuit of write cell with 8 write ports. Figure 13 (c) shows the circuit of read cell. All circuits are designed using static CMOS logic to reduce power. The register file supports 13 read ports to access the same register simultaneously. Under typical corners, the read delay is 360ps, and the power of full load is 41mW@1GHz.

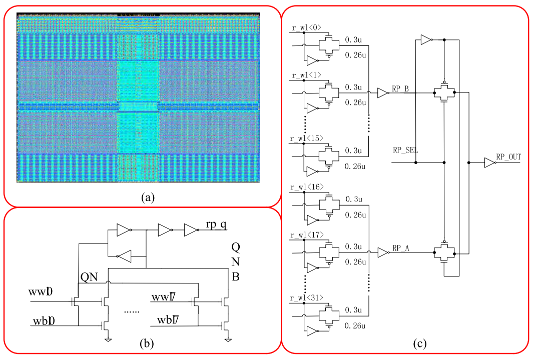

FIGURE 13. Circuit and layout of register file.

The chip size is 23.9mm × 24.1mm. Nine levels of metals are used. Clock frequency is 1GHz. At typical corner, the chip peak power is 150.4W@1GHz.

B. PERFORMANCE TESTING OF SCIENTIFIC COMPUTING AND SIGNAL PROCESSING

To test the performance of FT-Matrix2000 chip as an accelerator, we build a testing board (shown in Figure 14) with 16 GB DDR3 DRAM modules. The FT-Matrix2000 chip is connected to the host PC, running Linux OS, via a 16x PCIe 2.0 interface. To facilitate the communication between host PC and our FT-Matrix2000 chip, we implement the PCIe driver, which provides two main functions: configuration and data block transfer.

FIGURE 14. The testing board for CPU accelerator.

TABLE 2. Performance of 1-D complex and 2-D FFT algorithms, where S-FFT and D-FFT refer to single-precision and double-precision complex FFT, respectively, and the unit is milliseconds (ms).

	512K 1-D S-FFT	512K 2-D S-FFT	256K 1-D D-FFT	256K 2-D D-FFT
TI C6678 1-core	18.669	N/A	N/A	N/A
FT-Matrix2000	0.228	0.177	0.188	0.1599
TI C6678 8-core	3.103	N/A	N/A	N/A

We use 1-D (single dimension)/2-D (two dimension) complex FFT and GEMM (dense general matrix-matrix multiplication) algorithm to test the performance of FT-Matrix2000 chip running at 1 GHz. Table 2 compares the performance of complex FFT on FT-Matrix2000 chip and TI C6678 DSP chip. We implement four types of FFTs: 512K 1-D single precision float point complex FFT, 256K 1-D double precision float-point complex FFT, 512K 2-D single precision float point complex FFT, and 256K 2-D double precision float point complex FFT. Due to the intensive floating point computing power and efficient data shuffle support in the FT-Matrix2000 chip, the performance of FFT exhibits that FT-Matrix2000 chip well maintains DSP's ability in traditional signal processing applications and achieves performance improvement by a factor of 13.6, compared with parallel implementation on TI C6678 DSP with 8 cores.

TABLE 3. The test result of double-precision GEMM.

Dimensions of matrix C	Execution Time (s)	Performance (GFLOPS)	Efficiency (%)
512*576*300	0.103	1096.8	91.40
512*576*500	0.171	1102.8	91.90
512*576*600	0.238	1107.0	92.25

Table 3 illustrates the performance and efficiency of FT-Matrix2000 chip for double-precision GEMM. GEMM implements the computation of $C = C - A * B$, in which A;B and C are matrixes, representing a typical kernel in high performance computing. We have tested different computation scales with the dimensions of matrix C increased from $512*576*300$ to $512*576*700$. As shown in Table 3, with the increase of computation scale, the performance and computation efficiency of FT-Matrix2000 chip for GEMM increasing and reach up to 1107GFLOPS and 92.25%, respectively. Overall, the computation efficiency is above 90%. This is due to large computation power of FT-Matrix2000 chip accompanied with high inside memory bandwidth. So the parallelism of GEMM can be well exploited. This feature is critical in the super-computing environment.

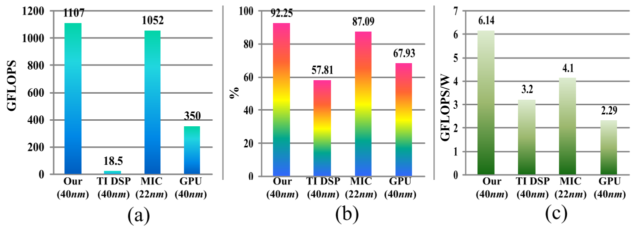

FIGURE 15. Performance, computation efficiency, and power efficiency of double-precision GEMM algorithm between FT-Matrix2000 chip and other processors.

Figure 15 compares the performance, computation efficiency, and power efficiency of double-precision GEMM algorithm between FT-Matrix2000 chip and other co-processors, where TI DSP is TI TMS320C6678 DSP [26], MIC is Intel Xeon phi co-processor (Knights Corner) [27] and GPU is NVidia Tesla M2050 [28]. Due to the extension and optimization of FT-Matrix2000 for scientific applications, our 12 cores chip can obtain the performance of 1107 GFLOPS for double-precision GEMM

TABLE 4. Comparison of experimental result.

		GPU			DSP	
Device		Pascal Titan X	GTX 1080	GTX 1080Ti	Vision P6	FT-Matrix2000
Technology(nm)		16	16	16	16	40
Precision		32 bits float	32 bit float	32 bits float	8 bits fixed-point	32 bits float
Frequency(GHz)		1.41	1.6	1.5	1	1
Throughput		10.16 TFLOPS	8.87 TFLOPS	10.6 TFLOPS	256 GMAC/S	2.4 TFLOPS
Utilization of Computing Resources	AlexNet	0.45	0.37	0.51	0.57	0.65
	Inception V1	0.52	0.45	0.53	-	0.68
	ResNet-18	0.62	0.49	0.58	-	0.70

and computation efficiency of 92.25%. The performance and computation efficiency on FT-Matrix2000 chip for double-precision GEMM exceed high-performance and low power of multi-core DSPs (TI TMS320C6678 DSP) running at 1 GHz. The performance of FT-Matrix2000 chip is higher than TI C6678 by a factor of 63.2. The performance and computation efficiency of FT-Matrix2000 chip is little higher than these of Intel Xeon phi co-processor (Knights Corner), which is made at a 22nm process size and achieves 1052 GFLOPS of performance and 87.09% of computation efficiency for double-precision GEMM. Compared with NVidia Tesla M2050 GPGPU, with the same design process (40nm), our architecture can achieve higher power efficient with a factor of 2.68.

C. PERFORMANCE TESTING OF DEEP LEARNING APPLICATIONS

In recent years, deep convolution neural network(DCNN) has been widely used on image classification, semantic segmentation and so on. And the broader use of DCNN comes at the cost of huge computational complexity.

As FT-Matrix2000 is efficient in vector and matrix related computation, it could also be used for accelerating deep learning algorithms. We compared the utilization of computing resource on GPUs and other DSP, as shown in Table 4.

We have implemented some typical DCNN models on a FT-Matrix2000 testing board in Figure 14 with a batch size of 16 and compared it with the NVIDIA GPUs with the same batch. Efland implemented DCNN on vector-SIMD DSP Vision P6 [29]. He only tested the computation efficiency of the AlexNet, and its computation efficiency is 57%.

We measured the performance of GPUs adopting Caffe framework with the latest CuDNN v7. The power on GPUs is obtained employing NVIDIA profiling tools. We tested the power of the testing board integrating FT-Matrix2000 chip and DDR in Figure 14 through a power test instrument. From Table 4, it can be seen that the average computation efficiency of FT-Matrix2000 is about 20 $\sim$ 35% higher than GPUs, and 8% higher than DSP Vision P6.

D. APPLICATION TESTING OF A 160TFLOPS PROTOTYPE COMPUTER

Based on FT-Matrix2000 chip, we build a prototype super computer of 160TFLOPS. This super computer has 16 nodes. Each node is composed with 2 computing blade boards and 2 accelerating blade boards by a NIC (Network Interface Card) chip to the NRM (Network Routing Mainboard) board for interconnect. A computing blade board is composed with 4 CPUs and an accelerating blade board is composed with 4 FT-Matrix2000 chips. The communication of two type blade boards is based on a PCIe2.0 x16. The performance of each node is 10TFLOPS. The whole super computer system has 128 CPU chips and 128 FT-Matrix2000 chips.

The YH-Kylin OS is operated in the super computer for node management and load balance. The application tasks are assigned to each CPU in the computing node with high speed network. The CPU loads the program, and input data to the FT-Matrix2000 chip, and fetches the outcome data from FT-Matrix2000 chip when finishing the computing.

TABLE 5. The test results of double-precision HPL and FFT, where the perk performance of prototype supercomputer is 180 TFLOPS and A-Perf, RAtoP, Power, and RPtoP refer to the actual performance, ratio of actual to peak performance, system power, and ratio of performance to power, respectively.

	Test scale	A-Perf (Tflops)	RAtoP (%)	Power (KW)	RPtoP (Gflops/W)
HPL	640000	112.1	62.2	21	5.33
FFTW	64G	2.28	/	/	/

Some benchmarks are tested in the super computer system, including double-precision HPL (High Performance Linpack) and FFTW. The results are shown by Table 5. The FT-Matrix2000 chip of 1.2TFLOPS contributes most of the computing performance of the super computer. With the help FT-Matrix2000 chip. The ratio of actual to peak performance is 62.2% for HPL and the ratio of power to performance is 5.33, which can be ranked fourth in the last Green500 [1]. Compared with supercomputer with CPU and GPGPU, our super computer prototype can achieve more ratio of peak performance, such as works in a part of "Chundoong" supercomputer with Intel Xeon E5-2650 and AMD Radeon HD 7970 [30], where performance and ratio of peak performance are 93.69 TFLOPS and 46%, respectively.

VII. RELATED WORK

Vector processing is massively adopted by state-of-art processors/cores, such as CPUs, GPUs, DSPs, etc. FT-Matrix [31], CEVA and Tensilica [29]. DSP IPs are traditional 32-bit embedded DSP or DSP core, and CPUs adopt superscalar architecture, while GPUs are typical stream processors with multi-threading support. Both CPUs and GPUs cannot ensure execution time certainty. FT-Matrix2000 is the first 64-bit DSP architecture.

Although TI has mapped the GEMM algorithm on its most advanced multi-core DSP TMS320C6678 [4], the efficiency of running GEMM (single precision) is less than 60%, which is far from the efficiency of other accelerators.

The Vector Memory can efficiently accelerate the data intensive applications with regular memory access modes, such as aligned and contiguous accesses. But for more complicated irregular access modes, the general VM is short of flexibility and confronted with memory conflicts and additional shuffles needs which lead to low bandwidth efficiency [33]. There was some research in the fused instructions of vector memory access and shuffle operation. Thomas [34] provided VHALFUP and VHALFDN instructions in VIRAM. But these instructions would bring an additional MOVE operation to buffer the temp result. Zhang *et al.* [35] put forward the VEXC instruction, which reduced the additional MOVE operation, but added an additional write-port to the register file. Both of them were only suited for radix-2 FFTs, and ours is suited for Radix-2/4 FFTs, and the improvement of ours proposed instruction is more obvious than them.

Dybdahl and Stenstrom [36] provided an adaptive shared/private partition scheme, which can adjust the ratio of shared cache to private cache by the application character. Ranjith and Yannis [37] allowed Cache adaptively changed the replaced policy following the behavior of workloads. However, the hardware cost of them too large, and our configurable share/private cache partition and priority replace policy in GC are different from them. Besides, our configurable bypass cache policy in GC drawed lesson from paper [38].

Architectures with scratchpad memories usually deploy DMA controllers to move data between off-chip memory and on-chip scratchpad memories. For example, the DMA controller is the only way for moving data between the local store and the off-chip DRAM in the Cell processor [39]. Also, the TI C6678 integrates two kinds of DMA: a global DMA controller shared by all 8 cores moves data between on-chip memory and off-chip memory, and a private DMA controller moves data between L1 memory and L2 memory inside the core [4]. Recent research proposes to incorporate either software DMA controller [40] or hardware DMA controller into the GPGPUs [41].

VIII. CONCLUSION AND FUTURE WORK

This paper proposes a novel DSP architecture: FT-Matrix2000, and validates that it achieve high efficiency for scientific computing and deep learning applications. FT-Matrix2000 preserves main features of DSPs, and incorporates several novel optimizations for scientific computing. We design an accelerator FT-Matrix2000. Although the manufacture process is not very advanced, the evaluation results show that the FT-Matrix2000 architecture achieve both high performance and high efficiency. Its FFT performance is about 13.6 times of that of TI C6678. Compared with the MIC and a 40nm process GPU, FT-Matrix2000 improves the GEMM power efficiency with a factor of 1.49 and 2.68 respectively. The performance power ratio in a 160 Tflops prototype supercomputer with FT-Matrix2000 is 5.33 GFLOPS/W, which can rank the fourth in the latest Green500 list. Compared with other GPUs and DSP, FT-Matrix2000 improves the computation efficiency by about 20 $\sim$ 35% and 8% respectively for deep learning applications.

We hope that our novel DSP architecture open a new way for exascale computing and high performance servers. Our future work mainly focuses on following aspects: testing more scientific benchmarks and applications; providing more libraries for the FT-Matrix2000 architecture, such as the OpenCV library; optimizing the FT-Matrix2000 architecture for artificial intelligence applications; improving the deployment of the host processor of FT-Matrix2000 architecture.

REFERENCES

[1] (Dec. 2016). *Green500*. [Online]. Available: http://www.top500.org/green500

[2] (Dec. 2016). *Top500*. [Online]. Available: http://www.top500.org

[3] D. Liu, *Embedded DSP Processor Design: Application Specific Instruction Set Processors*, 1st ed. San Mateo, CA, USA: Morgan Kaufmann, 2008.

[4] F. D. Igual, M. Ali, A. Friedmann, E. Stotzer, T. Wentz, and R. A. van de Geijn, "Unleashing the high-performance and low-power of multi-core DSPs for general-purpose HPC," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2012, pp. 1–11.

[5] X. Zhao, X. Wang, G. Zong, and H. Li, "Fuzzy-approximation-based adaptive output-feedback control for uncertain nonsmooth nonlinear systems," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 6, pp. 3847–3859, Dec. 2018.

[6] X. Zhao, P. Shi, and X. Zheng, "Fuzzy adaptive control design and discretization for a class of nonlinear uncertain systems," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1476–1483, Jun. 2016.

[7] H. Wang, P. X. Liu, S. Li, and D. Wang, "Adaptive neural output-feedback control for a class of nonlower triangular nonlinear systems with unmodeled dynamics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3658–3668, Aug. 2018.

[8] H. Wang, P. X. Liu, and B. Niu, "Robust fuzzy adaptive tracking control for nonaffine stochastic nonlinear switching systems," *IEEE Trans. Cybern.*, vol. 48, no. 8, pp. 2462–2471, Aug. 2018.

[9] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017.

[10] Y. Bazi and F. Melgani, "Convolutional SVM networks for object detection in UAV imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 56, no. 6, pp. 3017–3118, Jun. 2018.

[11] G. Lin, A. Milan, C. Shen, and I. Reid. (2016). "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation." [Online]. Available: https://arxiv.org/abs/1611.06612

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[13] F. Li, C. Wang, X. Liu, Y. Peng, and S. Jin, "A composite model of wound segmentation based on traditional methods and deep neural networks," *Comput. Intell. Neurosci.*, vol. 2018, pp. 1–12, Sep. 2018.

[14] X. Guo, E. Zhu, and J. Yin, "A fast and accurate method for detecting fingerprint reference point," *Neural Comput. Appl.*, vol. 29, no. 1, pp. 21–31, Jan. 2018.

[15] T. Lang and J. D. Bruguera, "Floating-point multiply-add-fused with reduced latency," *IEEE Trans. Comput.*, vol. 53, no. 8, pp. 988–1003, Aug. 2004.

[16] P. Soderquist and M. Leeser, "Division and square root: Choosing the right implementation," *IEEE Micro*, vol. 17, no. 4, pp. 56–66, Jul. 1997.

[17] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. AFIPS Conf*, vol. 38, May 1971, pp. 379–385.

[18] S. Liu, H. Chen, J. Wan, and Y. Wang, "Mod (2P-1) shuffle memory-access instructions for FFTs on vector SIMD DSPs," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2016, pp. 426–430.

[19] E. Anderson *et al.*, *Lapack Users, Guide*. Philadelphia, PA, USA: SIAM, 1999.

[20] W. M. Brown, "Synthetic aperture radar," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-3, no. 2, pp. 217–229, Sep. 1967.

[21] M. Coskun, K. Donglok, and K. Yongmin, "Efficient 2d fft implementation on mediaprocessors," *Parallel Comput.*, vol. 29, no. 6, pp. 691–709, Jun. 2003.

[22] K. Goto and R. van de Geijn, "Anatomy of high-performance matrix multiplication," *ACM Trans. Math. Softw.*, vol. 34, no. 3, p. 12, May 2008.

[23] T. M. Smith, R. van de Geijn, M. Smelyanskiy, J. R. Hammond, and F. G. V. Zee, "Anatomy of high-performance many-threaded matrix multiplication," in *Proc. IPDPS*, May 2014, pp. 1049–1059.

[24] S. Rixner1, W. J. Dally, U. J. Kapasi, P. Mattson, and J. D. Owens, "Memory access scheduling," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2000, pp. 128–138.

[25] (2016). *Micrium*. [Online]. Available: http://www.micrium.com

[26] T. I. (TI), (2013). *Tms320c6678 Multicore Fixed and Floating-Point Digital Signal Processor, Data Manual*. [Online]. Available: http://www.ti.com/lit/ds/symlink/tms320c6678.pdf

[27] A. Heinecke *et al.*, "Design and implementation of the linpack benchmark for single and multi-node systems based on intel xeon phi coprocessor," in *Proc. IEEE 27th Int. Symp. Parallel Distrib. Process. (IPDPS)*, May 2013, pp. 126–137.

[28] E. Philips, "Cuda accelerated linpack on clusters," in *Proc. GPU Technol. Conf.*, Sep. 2010, pp. 1–6.

[29] G. Efland, S. Parikh, H. Sanghavi, and A. Farooqui, "High performance DSP for vision, imaging and neural networks," in *Proc. Hot Chips Symp.*, Aug. 2016, pp. 1–30.

[30] G. Jo, J. Nah, J. Lee, J. Kim, and J. Lee, "Accelerating LINPACK with MPI-OpenCL on clusters of multi-GPU nodes," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 7, pp. 1814–1825, Jul. 2015.

[31] S. Chen *et al.*, "FT-matrix: A coordination-aware architecture for signal processing," *IEEE Micro*, vol. 34, no. 6, pp. 64–73, Nov./Dec. 2014.

[32] (Dec. 2016). *Uclinux*. [Online]. Available: http://www.uclinux.net

[33] J. W. Park, "Multiaccess memory system for attached SIMD computer," *IEEE Trans. Comput.*, vol. 53, no. 4, pp. 439–452, Apr. 2004.

[34] R. Thomas, "An architectural performance study of the fast fourier transform on vector iram," Dept Elect. Eng. Comput. Sci., California Univ. Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/CSD-00-1106m, 2000.

[35] K. Zhang, S. Chen, and S. Liu, "Accelerating the data shuffle operations for FFT algorithms on SIMD DSPs," in *Proc. IEEE Conf. ASIC*, Oct. 2011, pp. 1–10.

[36] H. Dybdahl and P. Stenstrom, "An adaptive shared/private nuca cache partitioning scheme for chip multiprocessors," in *Proc. HPCA*, 2007, pp. 2–12.

[37] S. Ranjith and S. Yannis, "Adaptive caches: Effective shaping of cache behavior to workloads," in *Proc. 39th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Sep. 2006, pp. 258–268.

[38] X. Lingxiang, C. Tianzhou, Q. Shi, and H. Wei, "Less reused filter: Improving l2 cache performance via filtering less reused lines," in *Proc. 23rd ACM Int. Conf. Supercomput.*, Jun. 2006, pp. 1–15.

[39] M. Kistler, M. Perrone, and F. Petrini, "Cell multiprocessor communication network: Built for speed," *IEEE MICRO*, vol. 26, no. 3, pp. 10–23, Sep. 2006.

[40] M. Bauer, H. Cook, and B. Khailany, "Cudadma: Optimizing gpu memory bandwidth via warp specialization," in *Proc. Int. Conf. High Perform. Comput., Netw.*, 2011, p. 12.

[41] D. Anoushe Jamshidi, M. Samadi, and S. Mahlke, "D2ma: Accelerating coarse-grained data transfer for gpus," in *Proc. PACT*, Sep. 2014, pp. 56–61.

CHAO YANG received the master's degree in electronic engineering from the National University of Defense Technology, in 2015, where he is currently pursuing the Ph.D. degree. His research interests include high-performance computing and deep learning.

SHUMING CHEN (M'01) received the Ph.D. degree in computer science from the National University of Defense Technology, China, where he is currently a Professor. He has co-authored more than 200 publications in international journals and conferences. His research interests include high-performance computing and VLSI design, and co-design of software and hardware.

JIAN ZHANG received the Ph.D. degree in electronic engineering from the National University of Defense Technology, in 2018, where he is currently a Research Associate with the College of Computer. His research interests include high performance computer architecture and VLSI design.

ZHAO LV received the master's degree in electronic engineering from the National University of Defense Technology, in 2014, where he is currently pursuing the Ph.D. degree. His research interests include high performance computer architecture and VLSI design.

ZHI WANG received the master's degree in electronic engineering from the National University of Defense Technology, in 2016, where he is currently pursuing the Ph.D. degree. His research interests include high-performance computing and VLSI design.