FT-Matrix: A Coordination-Aware Architecture for Signal Processing

VECTOR-SIMD ARCHITECTURES OFFER HIGH PERFORMANCE IN SIGNAL-PROCESSING APPLICATIONS BUT ARE INEFFICIENT AT THE COORDINATED EXPLOITATION OF HARDWARE UNITS. THE FT-MATRIX ARCHITECTURE REFINES THE COOPERATION BETWEEN THE SCALAR AND SIMD UNIT, ENHANCES COMMUNICATION AMONG SIMD LANES, AND ACHIEVES DATA SHARING AMONG VECTOR MEMORY BANKS. EVALUATION RESULTS SHOW AN AVERAGE PERFORMANCE GAIN OF 58.5 PERCENT AGAINST VECTOR-SIMD ARCHITECTURES WITHOUT THE PROPOSED IMPROVEMENTS.

Shuming Chen
Yaohua Wang
Sheng Liu
Jianghua Wan
Haiyan Chen
Hengzhu Liu
Kai Zhang
Xiangyuan Liu
Xi Ning
National University of Defense Technology, China

Vector-SIMD architectures can exploit data-level parallelism (DLP) while maintaining real-time and low-power constraints.[1] Moreover, vector-SIMD architectures with a scalar unit (SU) and SIMD unit (SIMDU) executing in parallel can further improve overall performance. Examples include the AnySP[2] and BBE[3] architectures. The SU is responsible for overall flow control, scalar processing, and SIMDU execution. The SIMDU, which comprises multiple lanes, is responsible for computation-intensive processing.

Given their wide usage, the overall performance of vector-SIMD architectures remains limited by inefficiency in coordinating different hardware units. This inefficiency has three aspects. The first is the cooperation between the SU and SIMDU. Although both tightly and loosely coupled execution of the SU and SIMDU are required, existing vector-SIMD architectures lack efficient support for the loosely coupled mode, leading to both performance degradation and a waste of hardware resources. The second aspect is the communication among SIMD lanes. Special data-accessing patterns in signal-processing applications would usually result in area and time penalties on the interlane communication unit of vector-SIMD architectures. The third is data sharing among multiple memory banks. Existing data sharing schemes are either too simple to be time efficient or too complex to be hardware efficient.

To overcome the above inefficiencies, this article proposes FT-Matrix, a coordination-aware vector-SIMD architecture for signal processing. FT-Matrix has greatly improved the coordination of different hardware units with three features: dynamic coupling execution (DCE), matrix-style communication, and unaligned vector memory (UAVM) access. In the first feature, the SU and

Published by the IEEE Computer Society

0272-1732/14/$31.00 © 2014 IEEE

Related Work in Signal Processing

One similar architecture enhancement to the dynamic coupling execution feature lies in the SA-1500,[1] which supports both the parallel execution method and tightly coupled method between a strongARM core and an attached media coprocessor. However, the concurrent execution of two different kernels is precluded. Efficient communication and control flow handling mechanisms are also not supported in the tightly coupled mode.

Using special structures for interlane communication is not a new concept. Many previous architectures adopted the SRAM-based shuffle network and the matrix register file.[2-4] For the shuffle unit, the crossbar is broadly adopted by vector-SIMD processors. Raghavan and Munaga proposed a customized crossbar in domain-specific vector-SIMD processors.[4] Such designs yield a power-efficient solution. However, they cannot support new shuffle operations for different applications post fabrication. AnySP employs the SRAM cells to replace flip flop at the cross point.[2] However, it needs too large a memory space to represent shuffle patterns. MRF can provide efficient data accessing for a fixed matrix size.[3] In the case of multiple submatrices, MRF provides no performance gain. Compared with existing architectures, novelties of our interlane communication structure include the compression of memory space in the SRAM-based shuffle network and the flexible configuration of the multigrained matrix register file (MMRF).

Vector memory has been widely used in embedded or media processors.[5-7] In these processors, each SIMD lane can access only its private memory bank. This limitation introduces too many interlane communication operations and impacts the system performance. The GPU provides the gather-scatter instructions to share memory space among SIMD lanes. However, this adds to the hardware complexity to deal with multiple cases of bank conflicts. Unaligned vector memory (UAVM) is a middle scheme between the private and sophisticated gather and scatter methods, achieving a good tradeoff between the performance and the hardware cost. ARM NEON technology also has similar unaligned memory access support.[8] However, the NEON technology aligns memory accesses by just half of the vector line, whereas our UAVM can access the vector memory from an arbitrary byte, which can greatly improve the overall performance with reasonable hardware cost.

References

1. S. Santhanam et al., "A Low-Cost, 300MHz, RISC CPU with Attached Media Processor," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, 1998, pp. 1829-1839.
2. M. Who et al., "AnySP: Anytime Anywhere Anyway Signal Processing," *Proc. 36th Ann. Int'l Symp. Computer Architecture* (ISCA 09), 2009, pp. 128-139.
3. A. Shahbahrami, B. Juurlink, and S. Vassiliadis, "Versatility of Extended Subwords and the Matrix Register File," *ACM Trans. Architecture and Code Optimization*, vol. 5, no. 1, 2008, article 5.
4. P. Raghavan and S. Munaga, "A Customized Cross-Bar for Data Shuffling in Domain Specific SIMD Processors," *Proc. Architecture of Computing Systems*, 2007, pp. 57-68.
5. X. Yang et al., "A 64-Bit Stream Processor Architecture for Scientific Applications," *Proc. 34th Ann. Int'l Symp. Computer Architecture* (ISCA 07), 2007, pp. 210-219.
6. M. Wen et al., "Multiple-Morphs Adaptive Stream Architecture," *J. Computer Science and Technology*, vol. 20, 2005, pp. 635-646.
7. K. Berkel and F. Heinle, "Vector Processing as an Enabler for Software-Defined Radio in Handheld Devices," *EURASIP J. Applied Signal Processing*, vol. 16, 2005, pp. 2613-2625.
8. V.G. Reddy, *Neon Technology Introduction*, ARM Corp., 2008.

SIMDU can be dynamically configured to execute in both tightly and loosely coupled modes. This flexibility can improve the overall performance by increasing the utilization of hardware resources. In the second feature, the matrix-style communication comprises the matrix template memory-based shuffle unit and the multigrained matrix register file, providing efficient support for data communication among SIMD lanes. Finally, the local vector memory can efficiently support the unaligned memory accessing, achieving an efficient tradeoff between performance and hardware cost. (For more information on other approaches, see the "Related Work in Signal Processing" sidebar.)

The importance of coordination

To gain valuable architectural inspirations, we investigated several representative signal-processing application kernels.

Requirement of both tightly and loosely coupled execution

Vector-SIMD architectures with the SU and SIMDU executing in parallel show a tightly coupled execution of the SU and

```
STBC_encoding(**Signal,**Coeff,Coding_length)
{
  for(i = 0; i<Coding_length/SIMD_width; i++)
  {
   for(j = 0; j<Antenna_Num; j++)
     for(k = 0; k<4; k++)
       Antenna[j][i] += Signal[i][k] * Coeff[j][k])
  }
}
REM(**Antenna, Coding_length)
{
  for(j=0; j<Antenna_Num; j++)
  {
    for(i=0; i<Coding_length; i++)
    {
      Index = mapping_table[j][i]
      Mapping_result[j][Index]=Antenna[j][i]
    }
  }
}
(a)
Prologue:
STBC_encoding(**Signal,**Coeff, Block_length)
Signal+=Antenna_Num*Block_length*i
Loop Body:
for(i = 0; i<Coding_length/Block_length-1; i++)
{
  REM(**Antenna, Block_length)
  STBC_encoding(**Signal, **Coeff, Block_length)
  Antenna+=Antenna_Num*Block_length*i
  Signal+=Antenna_Num*Block_length*i
}
Epilogue:
REM(**Antenna, Block_length)
(b)
```

Figure 1. Parallelism exposed by pipeline execution. Chained SIMD and scalar kernels (a). Pipelined execution (b).

SIMDU. This can provide efficient SU assistant operations for the SIMDU. Vector kernels such as string search and image rotation can benefit from this tightly coupled structure, owing to the frequent SU assistant operations, such as data sharing and control flow handling.

However, there are also vector kernels such as fast Fourier transform (FFT), inverse discrete cosine transform (IDCT), and quantization,[4] which keep the SIMDU busy doing intensive computations, while the SU stays mostly idle, conducting only simple operations such as variable initialization and loop controlling. Performance degradation and wasted hardware resources are also seen in scalar kernels, besides vector kernels, where the SIMDU is relegated while the SU is being occupied.

From this analysis, it seems that if scalar and vector kernels could run concurrently on the SU and SIMDU, respectively, both performance and resource utilization would be improved. However, this requires exposure of the parallelism between scalar and vector kernels (PSVK). For independent scalar and vector kernels, the PSVK is obvious. Moreover, most of the scalar kernels are chained with vector kernels—such as in Resource Element Mapping Demapping with FFT, Resource Element Mapping with STBC (Space Time Block Code) encoding,[4] and the IDCT transformation and quantization kernels with reordering (IQ + R).[4] Motion estimation kernels can also be divided into chained scalar and vector kernels.

The chained structure exhibits abundant PSVK with the help of the software pipelined scheme. As Figure 1a shows, Resource Element Mapping (REM in the figure) takes the result of STBC as input. If we divide the input data stream into a group of data blocks, we can pipeline these two kernels as shown in Figure 1b. Thus, a large amount of PSVK is exposed.

With abundant PSVK exposed, a loosely coupled execution of the SU and SIMDU should also be supported, so that both scalar and vector kernels can be executed concurrently on corresponding units.

Communication among SIMD lanes

Interlane communication operations include register-level shuffle operations[2] and matrix-oriented data accessing patterns.[5]

Shuffle operations are usually conducted on the shuffle unit. In traditional vector-SIMD processors, shuffle units must preload the shuffle patterns of applications into general registers. This can consume a large amount of register space. AnySP adopts an SRAM-based crossbar to eliminate this problem.[2] However, the SRAM-based method can cause unacceptable area overhead by

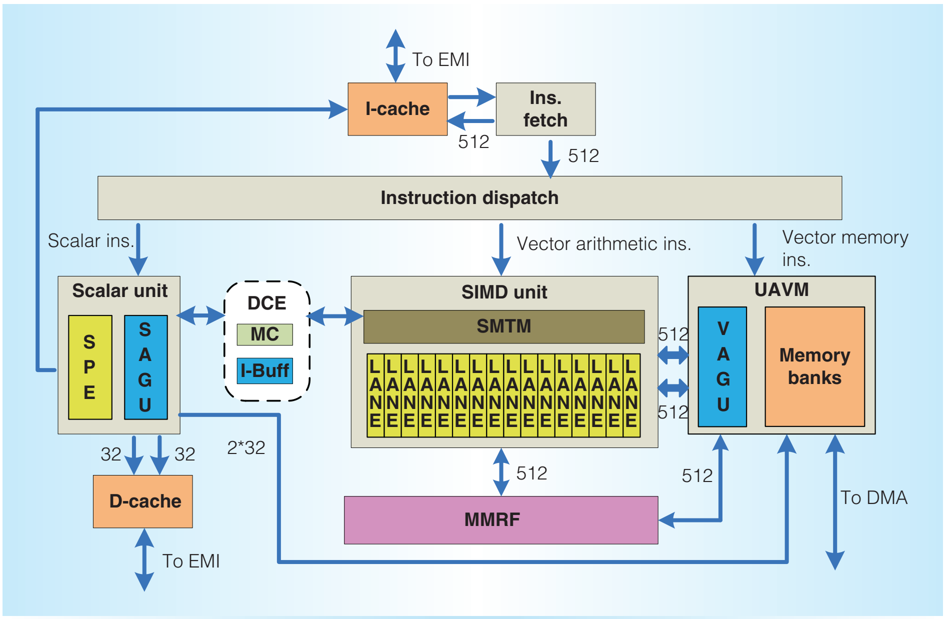

Figure 2. FT-Matrix architecture block diagram. The instruction fetch and dispatch units supply instructions to the scalar unit (SU), SIMD unit (SIMDU), and vector memory. (DCE: dynamic coupling execution; DMA: direct memory access; I-Buff: instruction buffer; Ins.: Instruction; MC: micro control unit; MMRF: multigrained matrix register file; SAGU: scalar address generation unit; SMTM: shuffle unit with matrix template memory; SPE: scalar processing element; UAVM: unaligned vector memory; VAGU: vector address generation unit.)

increasing the SIMD width. An ideal shuffle unit should eliminate the consumption of registers with reduced area penalty.

Signal-processing applications contain a large amount of matrix operations with different matrix sizes, such as the 2×2 and 4×4 matrices in wireless communication domain and the 8×8 matrices in video processing applications. Matrix operations cause a large amount of data rearrange overhead resulting from the column-wise data-accessing pattern. A matrix register file (MRF),[5] which supports both row-wise and column-wise data accessing, can be a better solution for efficient matrix operations. However, an MRF can provide efficient support only for fixed-sized matrices. For different-size matrices, MRF is used as general registers. Because signal-processing applications have different matrix sizes, they should support multiple-sized-matrix data accessing.

Unaligned vector memory access

Vector-SIMD architectures always adopt a multibanked memory structure (vector memory) for a large memory bandwidth. However, the irregular memory access seriously restricts the available bandwidth. Although direct memory access and MRF can regulate some irregular memory accesses into regular ones, irregular memory accesses remain that limit the available memory bandwidth. After an investigation of the signal-processing kernels FIR (finite impulse response), Autocor, and SAD (sum of absolute difference), we discovered that the UAVM access, which lets the vector load and store instructions to start accessing the vector memory from any byte address, is the most typical irregular access mode. In the FIR, Autocor, and SAD kernels, the unaligned memory access occupies about 45 to 80 percent of the irregular memory accesses. Thus, it is necessary to efficiently support the unaligned vector access.

Our evaluation informed a coordination-aware architectural analysis, where the coordination lies between the SU and SIMDU, and among SIMD lanes and memory banks.

The FT-Matrix architecture

Figure 2 shows the FT-Matrix architecture. The instruction fetch and dispatch units, under the SU's control, supply instructions to the SU, SIMDU, and vector memory. The SU contains a scalar processing element and a scalar address-generation unit. The SIMDU consists of 16 lanes. The function units in each lane are organized in a very long instruction-word manner. The vector memory contains multiple memory banks and a vector address-generation unit.

Dynamic coupling execution

We introduce a DCE scheme that can dynamically switch the execution of the SU and SIMDU between tightly and loosely coupled modes.

As Figure 2 shows, the cornerstone of DCE is a micro control unit and an instruction buffer (I-Buff). The micro control unit is responsible for SU assistant functions and conducting instruction saving and issuing of the I-Buff. The I-Buff is used to save vector kernel instructions.

With these components, the FT-Matrix architecture can be dynamically switched to a loosely coupled mode in which the SU runs as a single scalar core while the SIMDU fetches instructions from I-Buff under the micro control unit's control. Vector kernel instructions are prefetched into I-Buff. The scalar and vector kernels can be processed concurrently on corresponding units.

To support smooth switching between the two modes, we introduce two special instructions: Start_L and Wait. The default execution mode is the tightly coupled one. After vector kernel instructions are prefetched into the I-Buff, the Start_L instruction instructs the SIMDU to fetch instructions from I-Buff under the guide of the micro control unit, and then the entire architecture turns into the loosely coupled mode. The Wait instruction forces the SU to keep querying the SIMDU until all the lanes are idle. Then the execution mode is turned back into the tightly coupled one. Programmers can also change the execution mode from loosely coupled to tightly coupled by using interrupts.

Matrix-style communication

The matrix-style communication comprises a shuffle unit with matrix template memory (SMTM) and a multigrained matrix register file (MMRF).

Shuffle unit with matrix template memory. The SMTM (see Figure 2) comprises the matrix template memory, the transform logic, and the crossbar. An application's compressed shuffle patterns (CSPs) are prestored in the template memory and selected by the template memory index. The transform logic decompresses a CSP into an extended shuffle pattern, which drives the crossbar to shuffle source data elements.

To improve the memory efficiency of shuffle patterns, we adopt the binary coding compression scheme, which compresses the selection vector of each output port to its corresponding binary code. Besides, shuffle patterns with different shuffle grains are compressed separately. For a shuffle operation with a word grain, the needed bits will be compressed from $2N^2$ to $N\left(\log_2^N - 1\right)/4$. This scheme leads to high memory efficiency.

Multigrained matrix register file. The MMRF supports both row-wise and column-wise accesses of multigrained matrices. The MMRF consists of 16×16 32-bit registers. It includes 16 row-vector registers and 16 column-vector registers. The MMRF can be dynamically configured into one 16×16 matrix, four 8×8 matrices, and 16 4×4 matrices. As Figure 3 shows, when dealing with 4×4 sized matrices, four elements of independent result matrices can be generated concurrently. Additional data alignment operations, which are needed in traditional vector-SIMD architectures, are eliminated. The MMRF is implemented with a hierarchical, fully customized design method.

Unaligned vector memory

The vector memory comprises several vector banks. With the help of UAVM, vector data accesses can be started from an arbitrary byte address rather than from the start of a vector line.

To support UAVM, the possible addresses for each vector bank can be in the current line or the next line. Thus, we introduce additional selection logic to make decisions between two addresses. This simple selection logic does not delay the vector memory's

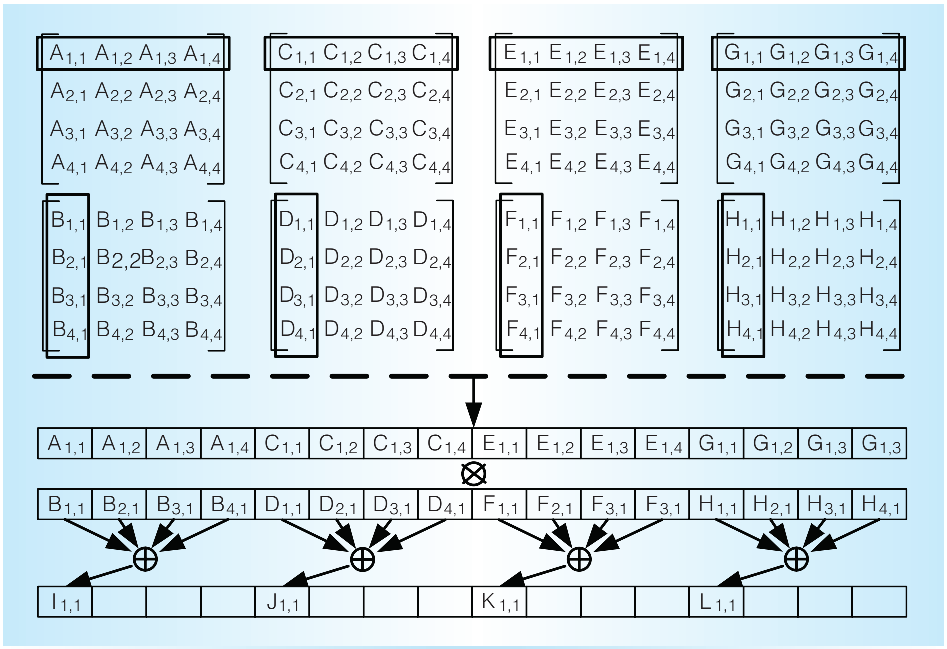
Figure 3. Part of matrix multiplication with the MMRF. Four elements of result matrices are generated concurrently, eliminating unnecessary data alignment operations.

cycle time. We also need an input data reorder unit (IDRU) and an output data reorder unit (ODRU), constructed with *N*-barrel shifters (where *N* is the SIMD width), to circular-shift vector data elements. For the read operation, the IRU right-circular shifts the vector data elements to the corresponding SIMD lanes; for the write operation, the vector data elements are left-circular shifted and written to vector banks.

Chip implementation

We completed the FT-QMBase chip (see Figure 4) with four FT-Matrix cores. FT-QMBase was implemented in the 65 nm technology, working at 500 MHz. The intercore communication mechanism was based on our previous design.[6] It implemented peripheral equipment, including a DDR3 controller and serial rapid I/Os. Each FT-Matrix core's area was 25.67 mm^2. The proposed new features consumed a total area of 4.05 percent of each FT-Matrix core.

FT-Matrix programs were written in a C-based formation with special pragma, intrinsic functions, and libraries provided by our in-house programming model. The corresponding compiler was built on the GCC compiler, which automatically compiles programs into SU and SIMDU instructions, correspondingly.

Performance evaluation

We developed a cycle-accurate microarchitectural simulator for the FT-Matrix architecture, and we validated the simulator with the real FT-Matrix core.

We compared the FT-Matrix architecture's performance with that of the baseline vector-SIMD architecture, which did not support new features proposed in the FT-Matrix. To provide a reasonable comparison with current vector-SIMD DSPs, we also built three key features of the AnySP[2] into the baseline architecture, forming an AnySP-like architecture. These features included flexible function units, a swizzle network, and a multiple output adder tree. Other features of AnySP were not included because they were orthogonal to the new FT-Matrix features. We selected several representative application kernels from the wireless communication[4] and video processing[7] domains. Table 1 lists the detailed parameters.

Figure 5 illustrates the FT-Matrix architecture's overall performance. Compared with

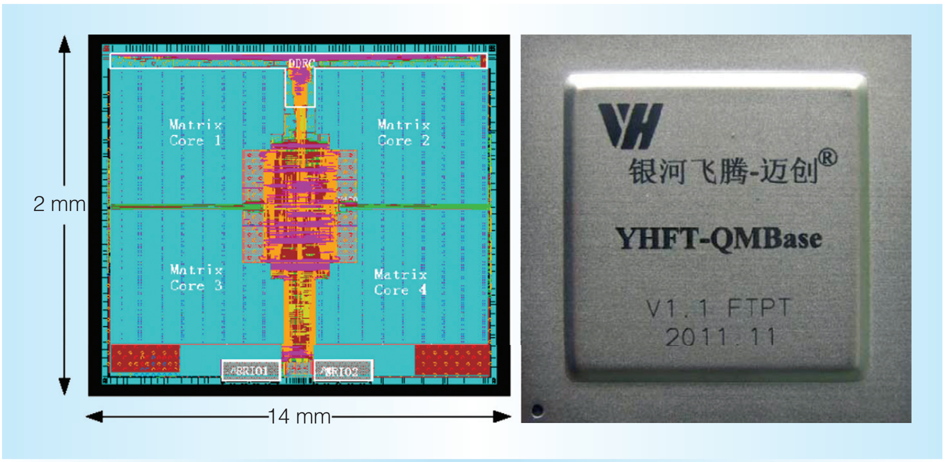

Figure 4. The FT-QMBase chip. The chip's layout (a). A photograph that shows FT-QMBase is implemented on a 14 × 12 = 168 mm^2 die in a 65-nm process (b) is the chip package.

Table 1. Kernel parameters from our performance comparison of the FT-Matrix and the baseline vector-SIMD architecture.

Application	Description
Fast Fourier transform and resource element demap (FFT + RED)	FFT: 2,048-point Radix-2 1,200 subcarriers, 12 symbols
Space Time Block Code encoding and resource element map (STBC + REM)	STBC: 4T4R antennae 1,200 subcarriers, 12 symbols
Motion estimation	1 ref-frame with full search Block size: 16 × 16
Inverse discrete cosine transform, quantilization, and reorder (IQ + R)	4 × 4 macro block
Multiple input, multiple output decoding and deinterleaving (MIMO + DeInt)	4T4R, 14,400 resource elements Lookup-table-based mapping
Intraprediction (Intra)	9 models with 4 × 4 luma block
Subpixel interpolation (SPI)	4 × 4 macro-block 1/2- or 1/4-pixel interpolation

the baseline and AnySP-like architecture, the average performance gain is approximately 58.5 and 30.6 percent, respectively.

Comparison with the AnySP-like architecture

As Figure 5 shows, FT-Matrix performs better than the AnySP-like architecture. This is mainly because of the DCE feature, which can hide the execution time of scalar kernels. Features such as MMRF and UAVM further improve FT-Matrix's performance by reducing memory access and shuffle operations. Additionally, because most of the fused operations in signal-processing applications are multiply-add, the MAC (multiply-accumulate operation) unit in each SIMD lane of FT-Matrix can well support the multiply-add operations and compensate for the lack of flexible function units. As for SMTM, it is functionally equal to the swizzle network in the AnySP-like architecture. One exception is the intraprediction (Intra) kernel, in which the fused operations such as shuffle-add and add-shift can well accelerate the overall performance, whereas FT-Matrix supports only the multiply-add operation. FT-Matrix does

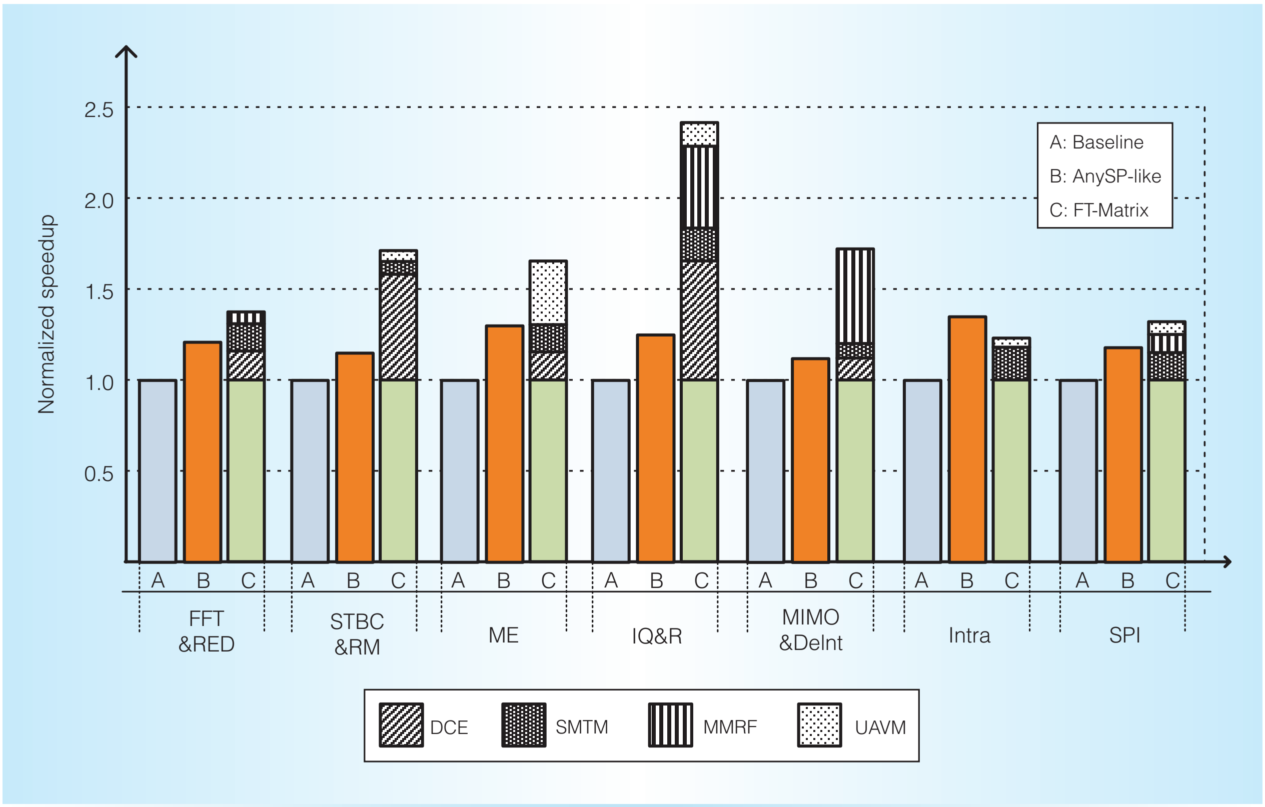

Figure 5. Overall performance of the FT-Matrix architecture. The average performance gain was approximately 58.5 percent compared with the baseline and 30.6 percent compared to the AnySP-like architecture.

not support the flexible function unit because of its complex compiler support.

Performance gain of architecture novelties

In this section, we describe performance gains of architecture novelties.

The DCE. The DCE scheme can well exploit the parallelism between scalar and vector kernels, while maintaining the efficiency of the traditional tightly coupled execution. The performance gain of the loosely coupled mode is affected by the overlap ratio of scalar and vector kernels, because scalar and vector kernels can execute only in a sequential manner in the baseline architecture. Thus, the equality of execution time for scalar and vector kernels is important to the final performance gain; the closer they are, the higher the speedup that can be achieved. As Figure 5 shows, the execution time of the REM and R is about 75 and 80 percent of STBC and IQ, respectively, leading to high performance gain.

The SMTM. The performance gain of SMTM is mainly affected by the amount of shuffle operations needed in application kernels. Compared with FFT, kernels such as STBC and multiple input, multiple output (MIMO) need only a few shuffle operations, thanks to MMRF's help, and the speedup by SMTM is a little smaller. Video applications such as SPI, IQ + R, motion estimation, and intraprediction (Intra-Pre) have a large amount of irregular data accesses and multiple data grains, leading to a large amount of shuffle operations. Therefore, SMTM can give high performance gain. In our experiment, the speedup of motion estimation and Intra is about 15 and 18 percent, respectively.

The MMRF. MMRF can exhibit high performance gain for applications with both column-wise and row-wise accesses, such as the IQ + R, MIMO + DeInt, and SPI kernels. IQ + R and MIMO + DeInt have a larger performance gain than SPI because of the diverse matrix sizes. For other applications, the relatively low performance gain is due to their lack of column-wise accesses.

The UAVM. The advantage of UAVM is affected by the amount of unaligned memory accesses. Motion estimation achieves a performance gain of 25 percent because of the massive sliding accesses to block data elements. In other kernels, such as Intra, SPI, and IQ + R, the proportion of unaligned memory accesses is relatively smaller. Their speedups by UAVM are between 5 and 13 percent. In wireless communication applications, the STBC kernel must rearrange the data elements to keep each antenna's data stream continuous. The UAVM can exhibit high performance gain. The memory access type in FFT and MIMO kernels are aligned. Thus, the role of UAVM is weak for these two kernels.

Interferences among architecture novelties

Because the DCE feature aims at the cooperation between the SU and SIMDU, it is orthogonal with features such as SMTM, MMRF, and UAVM. However, the MMRF and UAVM can greatly reduce the number of shuffle operations, which will otherwise be done on the SMTM. This interference is profitable. As for the MIMO kernel, to fulfill the column-wise data accesses, it needs 13 register accesses and 20 shuffle operations. With the help of MMRF, only eight register access operations are enough. An additional performance gain of 25 percent can be obtained, compared with barely the support of SMTM. A similar benefit can be obtained with the help of UAVM. As for IQ + R, the additional performance gain by using UAVM with SMTM is 10 percent larger than using only the SMTM feature.

Our evaluation results show that the FT-Matrix architecture greatly improves the performance of traditional vector-SIMD architectures. In future work, we will extend the coordination-aware research among multicores, which includes efficient intercore communication and work distribution. MICRO

Acknowledgments

We thank Haiyan Sun, Zhengtao Li, Yuanxi Peng, Shuwei Sun, Shenggang Chen, Zhong Liu, and Liu Yang for their efforts on this work. This work is supported by the Core Electronic Devices, High-End General Purpose Processor, and Fundamental System Software of China (no. 2009ZX01034-001-001-006).

References

1. Y. Lee et al., "Exploring the Tradeoffs Between Programmability and Efficiency in Data-Parallel Accelerators," *Proc. 38th Ann. Int'l Symp. Computer Architecture* (ISCA 11), 2011, pp. 129-140.
2. M. Who et al., "AnySP: Anytime Anywhere Anyway Signal Processing," *Proc. 36th Ann. Int'l Symp. Computer Architecture* (ISCA 09), 2009, pp. 128-139.
3. C. Rowen et al., "The World's Fastest DSP Core: Breaking the 100 GMAC/s Barrier," *Proc. 23rd Hot Chips Conf.*, 2011.
4. *Physical Channels and Modulation, 3GPP TS 36.211*, European Telecommunications Standards Institute, www.3gpp.org/ftp/specs/archive/36_series/36.211.
5. A. Shahbahrami, B. Juurlink, and S. Vassiliadis, "Versatility of Extended Subwords and the Matrix Register File," *ACM Trans. Architecture and Code Optimization*, vol. 5, no. 1, 2008, article 5.
6. S. Chen and J. Wan, "YHFT-QDSP: High-Performance Heterogeneous Multi-Core DSP," *J. Computer Science and Technology*, vol. 25, 2010, pp. 214-224.
7. T. Wiegard et al., "Overview of the H.264/AVC Video Coding Standard," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 13, no. 7, 2003, pp. 560-576.

Shuming Chen is a professor at the National University of Defense Technology, China. His research interests focus on VLSI design and the effects of radiation on integrated circuits. Chen has a PhD in computer science from the National University of Defense Technology, China.

Yaohua Wang is an associated professor at the National University of Defense Technology, China. His research interests include microarchitecture and signal processing. Wang has a PhD in electronic science and technology from the National University of Defense Technology, China.

Sheng Liu is an associated professor at the National University of Defense Technology, China. His research focuses on memory systems. Liu has a PhD in electronic science and technology from the National University of Defense Technology, China.

Jianghua Wan is an associated professor at the National University of Defense Technology, China. His research focuses on VLSI design. Wan has a PhD in computer science from the National University of Defense Technology, China.

Haiyan Chen is a professor at the National University of Defense Technology, China. Her research focuses on memory design. Chen has an MS in computer science from the National University of Defense Technology, China.

Hengzhu Liu is a professor at the National University of Defense Technology, China. His research interests include VLSI design and built-in self-test. Liu has a PhD in computer science from the National University of Defense Technology, China.

Kai Zhang is a PhD student at the National University of Defense Technology, China. His research focuses on VLSI design. Zhang has a PhD in electronic science and technology from the National University of Defense Technology, China.

Xiangyuan Liu is an associated professor at the National University of Defense Technology, China. His research focuses on circuit design. Liu has a PhD in electronic science and technology from the National University of Defense Technology, China.

Xi Ning is a PhD student at the National University of Defense Technology, China. His research focuses on microwave solid-circuit design. Ning has an MS in electronic science and technology from the National University of Defense Technology, China.

Direct questions and comments about this article to Shuming Chen, College of Computer Science and Technology on Parallel and Distributed Processing Laboratory, National University of Defense Technology, Changsha, Hunan, P.R. China, 410073; smchen@nudt.edu.cn.

cn *Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.*

ADVERTISER SALES INFORMATION

Advertising Personnel
Marian Anderson
Sr. Advertising Coordinator
Email: manderson@computer.org
Phone: +1 714 816 2139
Fax: +1 714 821 4010

Sandy Brown
Sr. Business Development Mgr.
Email: sbrown@computer.org
Phone: +1 714 816 2144
Fax: +1 714 821 4010

Advertising Sales Representatives (display)
Central, Northwest, Far East:
Eric Kincaid
Email: e.kincaid@computer.org
Phone: +1 214 673 3742
Fax: +1 888 886 8599

Northeast, Midwest, Europe, Middle East:
Ann & David Schissler
Email: a.schissler@computer.org, d.schissler@computer.org
Phone: +1 508 394 4026
Fax: +1 508 394 1707

Southwest, California:
Mike Hughes
Email: mikehughes@computer.org
Phone: +1 805 529 6790

Southeast:
Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 973 585 7070
Fax: +1 973 585 7071

Advertising Sales Representative (Classified Line)
Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 973 304 4123
Fax: +1 973 585 7071

Advertising Sales Representative (Jobs Board)
Heather Buonadies
Email: h.buonadies@computer.org
Phone: +1 973 304 4123
Fax: +1 973 585 7071