International Journal of Hybrid Information Technology
Vol.2, No.2, April, 2009

Gaining Flexibility and Performance of Computing Using Application-Specific Instructions and Reconfigurable Architecture[1]

Tian Hangpei, Gao Deyuan, and Zhu Yian
Aviation Microelectronics Center,
Northwestern Polytechnical University, Xi'an, China
dream_hunter@163.com

Abstract

As data transfer rates become higher, general processor is not competent for works with high computing intensity in wireless communication area. The other hand, existing baseband processors lack adaptive ability to support new applications and newer versions of existing application. Thus the paper puts forward a novel architecture of reconfigurable stream processor along with application-specific instruction set for software radio. The main goal of the processor is to point out an effective way to designing hardware of software radio with high computational performance and adaptive ability. We achieve this by analyzing multi-stream modalities of advanced wireless communication standards, such as 3G and WLAN. Simulation results show that the processor with powerful calculation capability can adapt to applications of wireless communication area easily.

1. Introduction

Software radio, which performs multi-communication standards on a platform, moves huge signal processing works on processors and challenges general processors' computing capabilities.[1] Although performance of general DSP is improving, it can't catch up with increasing paces of processing demands in new applications. Novel processor architectures and specific instruction set should be researched.

Some groups and individuals have researched on the subjects. Research group of Stanford University designed a super stream processor named Merrimac. The processor processes general stream signals including wireless signals with high speed [2][10][5]. [6] and [12] proposed a DSP architecture optimized for performing Viterbi calculation. [7] presented an instruction set which can accelerate some important coding and encoding algorithms. Baseband processor presented by researchers of Linkopings Universitet supports multiple communication standards via integration multiply accelerators [8][9][4]. All of above researches have some problems in different degrees. These problems include high cost, low level flexibility, only supporting certain kinds of algorithms and poor capability adapting to new applications. In order to improve the situation, this paper proposes an application-specific stream processor and application-specific instruction set. Simulation results show that storage structure of the processor can hide memory cost successfully, specific reconfigurable data path with three-level parallel speeds up executions of important algorithms effectively and configurable instruction set provides stronger adaptive capability than general baseband processor.

Our research methodology consists of following steps: (1) Analyzing stream processing of algorithms used in 3G and WLAN and extracting kernels of algorithms. (2) Designing basic reconfigurable data paths to accelerate executions of kernels. (3) Constructing configurable

[1] This paper is supported by National Natural Science Foundation of China (NSFC) (No. 60573101,60736012）

instruction set and stream processor architecture. (4) System simulation. The organization of the paper accords with the above steps.

2. Algorithms analysis

We write program of each algorithm manually and analysis these programs by GCC tools, such as Gcov and Gprof. The executing time of each basic block of a program is estimated and kernels of algorithms are extracted. At last, we construct specific computing paths to accelerate kernels' executions. The hardware cost of each algorithm is also estimated.

2.1. FFT

We use radix-4 decimation in frequency FFT. The reason why we choose radix-4 but not radix-2 is that radix-4 requires fewer stages. And also, researchers have done good job on Real FFT, which reduces the cost both in processing stages and memory and improves scalability of radix-4 FFT [3]. But radix-4 FFT still have higher data communicational overhead compared with radix-2 FFT. From experiment we found that the basic butterfly computing of radix-4 FFT illustrated in formula 1 is the kernel operations. In this data path, two ALU needed to complete plural additions and two MAC complete plural multiplications. When computing, a and c are needed at first step. Then b and d are computed by two ALU. W_N^n, W_N^{2n} and W_N^{3n} are needed to computing plural multiplications. This strategy can release pressure of bandwidth.

$$\begin{cases} A = a + b + c + d \\ B = (a - jb - c + jd)W_N^n \\ C = (a - b + c - d)W_N^{2n} \\ D = (a + jb - c - jd)W_N^{3n} \end{cases} \quad (1)$$

2.2. Rake Receiver

Rake receiver is one of the most important parts for baseband processing of DS-CDMA. It is used for de-spread and channel compensation. Figure 1 is a classic M-way Rake receiver composed by M fingers. Fingers' number in a Rake receiver is the same as the tap number of channel impulse response. When computing a figure, scrambling and de-spread can be achieved by two ALUs separately, integral is completed by a MAC and Phase estimate and figure addition need another MAC.

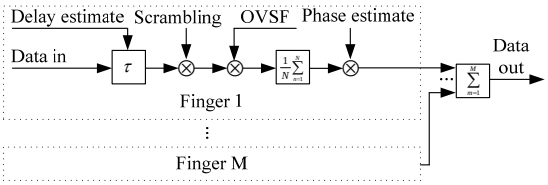


Figure 1. M way Rake receiver

2.3. Viterbi

Convolutional code can be depicted by grid. When Viterbi decodes, it measures paths in grid and chooses one which is the most similar to input data. The main process of Viterbi is reading input data, calculating local distances of possible paths and choosing right paths. By

examination we found that complicated Add-Compare-Select (cACS) computing spends most of computing resources. It's valuable to design ACS logics completing cACS with high efficiency. When computing, two ACS is needed to complete a cACS.

2.4. Turbo Coding

Turbo coding uses iterative decoding technology to reduce the decoding complexity. We choose MAX-Log-MAR which is suitable for implementing on processor. By programming, we found that main operations of the algorithms are simple Add-Compare-Select (sACS). sACS computing can be accomplished by ACS logic. Data paths of Turbo and Viterbi have same functions except that Viterbi coding is vector operation and Turbo coding is scalar operation.

3. Configurable instruction set

In previous section, we analyze basic hardware of configurable instruction. This section focuses on building a specific instruction set. Four specific configurable instructions are constructed and optimized.

In order to get appropriate computing capability and hardware overhead of each instruction, this paper defines reconfigurable instructions by setting parameters about instruction function (InsF) instruction set independence (InsSI) and Instruction relation (InsR).

InsF specifics computing function of instruction. It has two sub-parameters as basic computing path (BCP) and number of parallel basic computing path (NBCP). BCP represents specific data paths of different algorithms defined in the last sections. NBCP represents data level parallel. The two parameters are most important for instruction. Figure 2 shows that instruction computing capability and instruction hardware overhead can be adjusted mainly by BCP and NBCP. InsSI reflects relations between configurable instructions and general instructions. InsR illuminates relations between configurable instructions and showed hardware relations. It can reduce total hardware overhead.Considering four algorithms need to be accelerated, four specific instructions are present in Table 1. Each configurable instruction accelerating certain algorithm has four parameters defined above. These parameters influence instructions with different degrees.

BCP confirmed in last section determines computing function and hardware cost of instruction partly. BCP will be illustrated in terms of certain processor structure in the next paragraph. NBCP, which is the degree of data level parallel, is the hardest parameters to be confirmed. This parameter influences not only performance but also hardware cost and bandwidth of configurable instruction. We construct a matlab estimation model to measures hardware and bandwidth affected by NBCP from one to eight. We get the most suitable numbers of NBCP in Table 1. InsSI analyses relations between configurable instruction and general instruction. When considering InsR, FFT_Ins and Rake_Ins have some crossed functions and Turbo_Ins inherits functions of Viterbi_Ins. These function relations can facilitate hardware sharing and reduce total cost.

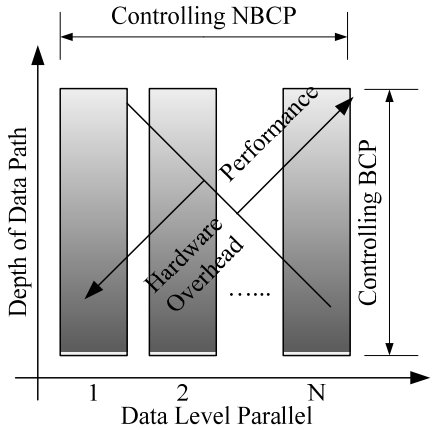


Figure 2. Functions of BCP and NBCP

Table 1. Parameters of configurable instructions

Configurable Instruction	*Algorithm accelerated*	*InsF*		*InsSI*	*InsR*
		BCP	*NBCP*		
FFT_Ins	*FFT*	*Figure4(a)*	*4*	*Executing parallel with general instruction*	*cross*
Rake_Ins	*RAKE*	*Figure4(b)*	*8*		
Viterbi_Ins	*Viterbi*	*Figure4(c)*	*8*		*inherit*
Turbo_Ins	*Turbo*	*Figure4(c)*	*8*		

4. Architecture of reconfigurable stream processor

Stream processing has two outstanding aspects. One is arithmetic, another is bandwidth. Arithmetic intensity, which is the ratio of arithmetic to bandwidth, should be improved as soon as possible. One way to raise arithmetic intensity is multi-stage memory structure and specific computing path [10]. This way has two problems, its need huge amount of registers to build SRF and flexibility will be poorer while pursuing locality. This section explores not only locality, but also flexibility of processor by considering processing efficiency and general aspects of processing. We observe and summarize multi-stream processing of communication standards at first. Then a reasonable stream storage subsystem and specific data paths which can accelerate the processing are put up. Lastly we construct a processor by putting both of these subsystems into a host system.

4.1. Wireless communication stream

By observing streams of wireless communication, we summarize some basic characteristics of them. Such as streams with congruously oriented flow, real time streaming and coherent operations in one stream step. These characteristics arouse both challenge and convenience for designing processor. Because streams are congruous oriented, embranchment and feedback of data paths appear with little chance. As a result, it's easily to design pipeline or parallel hardware with simple interconnection. Because streams flow in real time, life period of data in stream cache can be predicted easily. Stream cache, without complex data replace strategy and tag for lookup, can be used to hiding memory latency by pre-accessing data from memory and adjusting original data locality into stream order. Stream order can promote parallel access and access efficiency. Data level parallel (DLP) can be accomplished

conveniently by taking advantage of coherent operations. But when execution units operate on data with same actions simultaneously, “Peak-Bottom” may arouse if concerning storage accessing. That is to say, because of identical movements, executing unit may access data from memory at one cycle which pressing the bandwidth of processor and considered as “peak”, or no one uses memory at all in other cycles which wasting the bandwidth and considered as “bottom”. All of these conveniences and challenges should be discussed when constructing specific processor structure.

4.2. Architecture overview

Considering above conveniences and challenges, we design a Reconfigurable Stream Processor (ReSP) to process wireless baseband signals with powerful computing and scalability. Figure 3 shows details of ReSP. The processor is consisted mainly by storage subsystem, configurable subsystem and host unit. We discuss their details respectively.

Storage subsystem consists of memory, stream cache, data cache and program cache. Memory is responsible for storing data and context from I/O, which are connected by DMA. Program cache stores instructions and data cache stores data. Stream cache pre-access data from memory and adjust original data locality into stream order. In stream cache, cache controller is a partially configurable block which maps data into appropriate banks and control data transfer between banks. There are eight banks in the stream cache, each of them monopolized by a configurable block which has a same serial number as banks. Although quantity of banks is large, every bank only has 2Kbyte. Eight banks can be accessed simultaneously by configurable blocks. Furthermore, Banks from zero to three or four to seven can exchange data under control of cache controller.

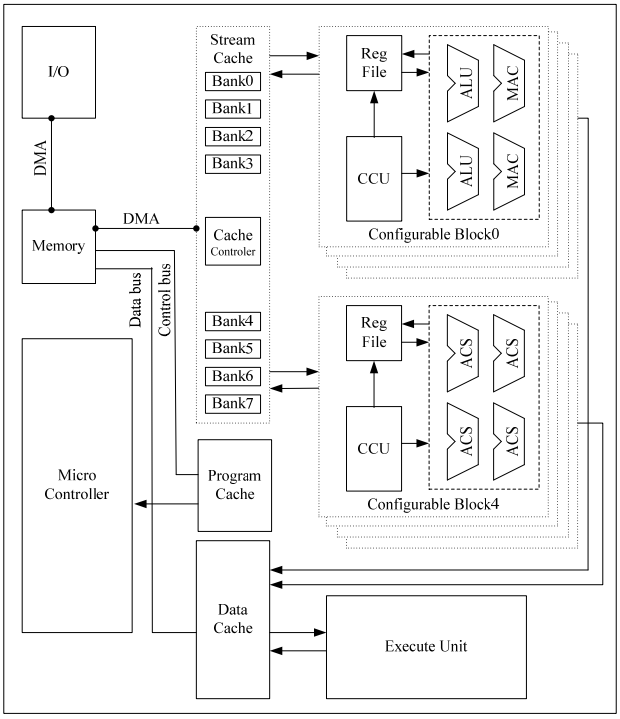


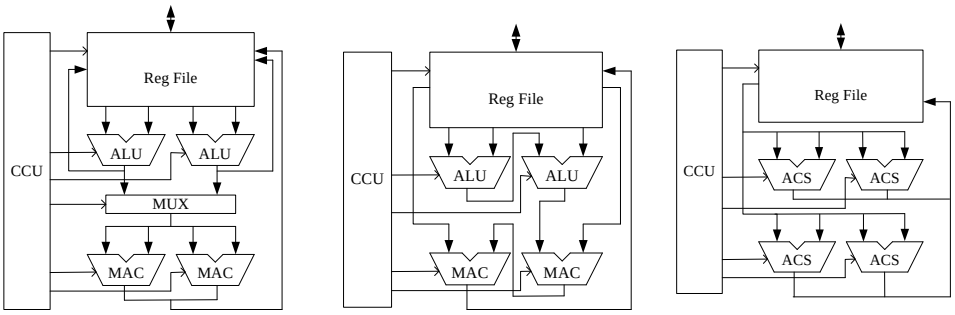
Figure 3. Architecture of ReSP

There are eight configurable blocks (CB) in the configurable subsystem. Each of CB consist of configurable control block (CCU), register file, and four calculational units (CU). The 8×32bite register file stores data from stream cache and CUs. CCU is considered as a state machine and address generator controls all actions of CB. Four CUs are consisted by ALU, MAC or ACS (Add-Compare-Select) logics compose specific data path by reconfigure interconnections. Considering concrete algorithms, Each of CUs from zero to three consist with two ALU and two MAC. Similarly, each of CUs from four to seven is composed by four ACS. Here we design ALU, MAC and ACS with sub-word calculational capability. That is to say, CU can accomplish one 32bit OP 32bit operation or two 16bit OP 16bit operations. OP means operations, such as addition, multiplication, MAC, ACS and so on. Computing results of CBs send back to memory. If CBs need to communicate with each other, they store their data into respective stream cache bank at first. Then data will be exchanged between banks in stream cache. Finally, CBs read data from their own stream cache bank.

Micro controller considered as a host processor is responsible for controlling the stream processor. It can be conducted as a RISC processor or embedded processor like ARM. Execute unit (EU) executes general instructions of processor.

4.3. Instruction of ReSP

Instructions of ReSP can be classified as general instruction and stream instruction. General instruction executed by EU includes basic DSP instructions. Stream instruction is executed by CB and stream cache. It includes all configurable instructions. Stream instruction is comprised of stream-store instruction (SSI) and stream-execution instruction (SEI). SSI and SEI execute parallel, which hides spending of memory access effectively. SEI supports SIMD structure.Four specific instructions proposed in the last section are implemented on the ReSP. Figure 4 shows the basic data path of each instruction.



(a)FFT_Ins (b) Rake_Ins (c) Viterbi_Ins/Turbo_Ins

Figure 4. Configurable data paths of algorithms

5. Results

Main goals of our processor are to provide highly powerful computing and enough flexibility. In order to evaluate these performances, we build a static-reconfigurable processor on FPGA and write assemblers manually. We keep simulation results objective by doing experiments more times. The static-reconfigurable simulator supports two configurable

modes. The first mode supports instruction of FFT_Ins. The second mode supports Rake_Ins. Both of two modes support Turbo_Ins or Viterbi_Ins.

5.1. Arithmetic intensity and Hardware usage

Improving arithmetic intensity is one of the most critical tasks for ReSP which has high level data parallel. High arithmetic intensity means high computational performance and low bandwidth consumption. Two strategies are used to increasing arithmetic intensity. In one hand, stream cache with multiply banks and register files in CBs are designed to improve data locality. In another hand, specific data paths configured by CBs accomplish abundantly consecutive operations. Both of two strategies decrease bandwidth. Specific data paths also improve computational performance. By comparing arithmetic intensity between ReSP and general DSP, we find that arithmetic intensities are improved dramatically. We also find that arithmetic intensity improving of Rake receiver is lower than other algorithms. This is because the algorithm is a function of time. As a SIMD processor, ReSP gets data level parallel by using CB parallel, EU parallel and sub-word parallel, all of which are supported by hardware parallel. The three-hierarchical data parallel provides high level DLP. Hardware parallel of the second mode is lower than the first mode. This is because that computing of Rake receiver is constrained by time which disperses operations of hardware. In the first mode, parallel data paths and data locality of FFT are explored, which promote hardware parallel effectively.

5.2. Flexibility and adaptive computing

As completing high computing works by fixed accelerators, general baseband processor is hard to extend functions. In order to get better flexibility, ReSP equipped with configurable logics which can be reconfigured as suitable data paths. Further, generalities of different algorithms are extracted and application-specific instruction set with general aspects is presented. The instruction set can adapt to multiply standards easily. As a result, programmers can write parametric programs comfortably for existing wireless standards and even new standards. Table 2 is the comparison of flexibility involving general baseband processor, general DSP and ReSP. In the table, we can see baseband processor has the poorest flexibility. Although general DSP can't extend instructions, it can support new functions by reprogramming. ReSP can change instruction set to adapt to various applications. The table also shows that length of program running on ReSP is shorter than TMS320C64. This means that ReSP is easier to programming and has higher effect.

Table 2. Flexibility Comparison

		Specific Processor	General DSP	Baseband Processor
Processor		*ReSP*	*TMS320C64*	*[9]*
Instruction		*Specific Ins. General Ins.*	*General Ins.*	*Accelerator*
Parametric Programming		*Yes*	*Yes*	*No*
Instruction Extension		*Yes*	*No*	*No*
Application Extension		*Yes*	*Yes*	*No*
Flexibility	FFT	*28*	*73*	--
	Rake	*23*	*57*	--
	Viterbi	*65*	*230*	--
	Turbo	*77*	*563*	--

5.3. Computational performance

In order to evaluate performance of ReSP when processing algorithms in real environments, we set a benchmark illustrated in Table 3. The benchmark simulates conditions of real time processing. Figure 5 is the simulation results. Although ReSP has lower instruction level parallel compared with MIMD processor, it has powerful performance by equipping high degree DLP. Figure 5 shows speedup of ReSP compared with general DSP. The figure shows that MIPS costs of ReSP reduce at least 7.5 times compared with general DPS. Figure 6 shows clocks of executing some algorithms. We can see that our processor is faster than general DSP when executing FFT and Viterbi.

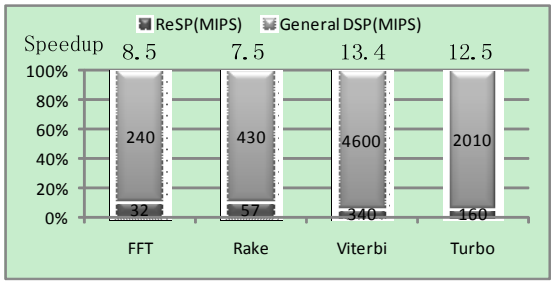


Figure 5. Performance comparison for FFT, Rake, Viterbi and Turbo

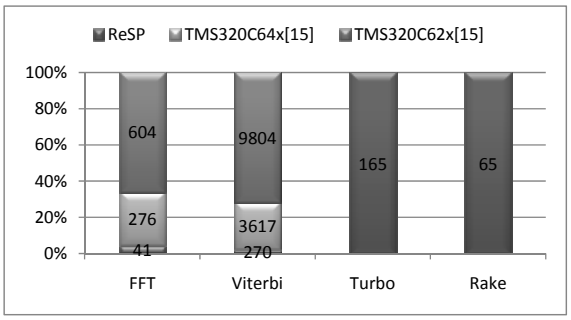


Figure 6. Clocks of Executing FFT, Rake, Viterbi and Turbo

Table 3. Benchmark

Algorithm	Benchmark
FFT	40Hz,64point radix-4 FFT
Rake	3.84Mcps, 8 finger Rake Receiver
Viterbi	constraint length 9 and coding rates 1/2
Turbo	parallel concatenated convolutional code, constraint length 3 and coding rates 1/3

6. Conclusions

New technologies and new standards of wireless communication area take big challenges for general processors. This article focuses on the problem and constructs a reconfigurable stream processor by analyzing representative communication standards. Stream cache with multi-bank and tailored reconfigurable logics is implemented. Configurable instruction set for Software radio is presented too. The paper provides a novel way of designing hardware for software radio.

References

[1] Lin, Y. Lee, H.; Woh, M. et al. SODA: A High-Performance DSP Architecture for Software-Defined Radio, IEEE Micro, 27: 114 - 123, 2007

[2] Mattan Erez. Merrimac : high-performance and highly-efficient scientific computing with streams. Ph.D. dissertation, November 2006, Stanford University, Stanford, California

[3] Hsiang-Feng Chi and Zhao-Hong Lai. A cost-effective memory-based real-valued FFT and Hermitian symmetric IFFT processor for DMT-based wire-line transmission systems. In Proc. IEEE International Symposium on Circuits and Systems, 6:6006-6009, 2005

[4] Eric Tell, Anders Nilsson and Dake Liu. A Programmable DSP core for Baseband Processing. In Proc.IEEE Northeast Workshop on Circuits and Systems (NEWCAS), Quebec City, Canada, 403-406, 2005

[5] Mattan Erez, Nuwan Jayasena and Timothy J. Knightet al. Fault Tolerance Techniques for the Merrimac Streaming Supercomputer. In Proc. the ACM/IEEE SC Conference, 29-26, 2005

[6] Jeong H. L., Weon H. P. and Jong H. M. et al. Efficient DSP architecture for Viterbi decoding with small trace back latency. In Proc.The 2004 IEEE Asia-Pacific Conference on Circuits and Systems, 1:129-132, 2004

[7] Suman M. and Emly R. B. et al. Instruction Set Extensions for Software Defined Radio on a Multithreaded Processor, In Proc.International Conference on Compilers, Architecture and Synthesis for Embedded Systems, 24-27, 2005, San Francisco, California, USA

[8] Anders Nilsson, Eric Tell, and Dake Liu. A fully programmable Rake-receiver architecture for multi-standard baseband processors. In Proc.Networks and Communcation Systems, Krabi, Thailand, 2005

[9] Anders Nilsson, Eric Tell and Dake Liu. An accelerator architecture for programmable multi-standard baseband processors. In Proc.WNET2004, Banff, AB, Canada, 2004

[10] William J. D., Patrick H. and Mattan E. et al. Merrimac: Supercomputing with Streams. In Proc.SC'03 ACM, 2003

[11] Jon W. Mark and Weihua Zhuang. Wireless Communications and Networking. Prentice Hall. 2003

[12] Jung H. L., Jae S. L. and Sunwoo et al. Design of new DSP instructions and their hardware architecture for the Viterbi decoding algorithm. In Proc. IEEE International Symposium on Circuits and Systems, 5:561-564, 2002

[13] Texas Instruments. TMS320C64x DSP Viterbi-Decoder Coprocessor (VCP) Reference Guide, 2004

[14] Texas Instruments. TMS320C64x DSP Turbo-Decoder Coprocessor (TCP) Reference Guide, 2004

[15] Benchmarks, Texas Instruments Inc. http://www.ti.com/sc/docs/products/dsp/c6000/benchmarks/ 62x(or 64X).htm#fft

Authors

Tian Hangpei received the BSc degree in computer science and technology in 2005 from Northwestern Ploytechnical University (NPU) of China. Now he is a PhD candidate of NPU. His main research areas include high-performance processor architecture, stream processing, dynamical reconfiguration, on-chip memory hierarchy and related topics.

Gao Deyuan received the BSc degree in 1973 from Northwestern Ploytechnical University of China. He worked as a research scientist for University of Southern California of USA, where he conducted the research of VLSI during the period 1985-1987. Now he is the professor of NPU. His main research areas include high-performance processor architecture, VLSI and system

Zhu Yian received the PhD degree in 1991 from Northwestern Ploytechnical University of China. Now he is the professor of NPU. His main research areas include system performance evaluation, parallel computing and network.