PAPER

Implementations of FFT and STBD for MIMO-OFDM on a Reconfigurable Baseband Platform

Shuang ZHAO[†], *Student Member*, **Wenqing LU**[†], **Xiaofang ZHOU**[†a)], **Dian ZHOU**[†], *and* **Gerald E. SOBELMAN**[††], *Nonmembers*

SUMMARY MIMO-OFDM systems aim to improve transmission quality and/or throughput but require significant signal processing capability and flexibility at reasonable cost. This paper proposes a reconfigurable architecture and associated algorithm optimizations for these types of systems based on the IEEE 802.11n and IEEE 802.16e standards. In particular, we describe the implementation of two key computations onto this architecture, namely Fast Fourier Transform (FFT) and Space-Time Block Decoding (STBD). The design is post-layout using a UMC 0.18 micron technology at a clock rate of 100 MHz. Performance comparisons with other optimization methods and hardware implementations are given.

key words: *MIMO-OFDM, reconfigurable architecture, FFT, STBD, algorithm implementation*

1. Introduction

MIMO techniques have received great attention and have developed significantly in recent years. By exploiting multiple antennas at both transmitter and receiver, multi-path effects can be effectively restrained, and the systems have the potential to achieve much higher bandwidth efficiencies and performance reliability [1]. Additionally, by combining MIMO systems with orthogonal frequency division multiplexing (OFDM), which is known as a MIMO-OFDM, a system can provide high spectral efficiency in a scattering environment [2] and can also mitigate the effects of inter-symbol interference (ISI). Because of these features, the MIMO-OFDM technique has been widely used in many high-throughput systems, including both the IEEE 802.11n and 802.16e standards.

High throughput means higher data bandwidth, and thus these systems require stronger signal processing capabilities. The higher the throughput requirement, the more antennas will be required, resulting in greater system complexity. Moreover, among those protocols, various sizes of channel matrix, baseband algorithms and operational modes are used. Even within the same standard, the number of antennas may be adjusted according to the existing channel quality. In order to meet such requirements, a hardware architecture having a very high degree of flexibility is needed. The challenge is to provide a flexible architecture having reasonable cost which can support multiple channel links. A coarse-grained reconfigurable architecture with heterogeneous operational units has been developed in our previous work [3] to implement WLAN OFDM systems. Considering the increased communications among the different execute units, a register-based interconnection and storage [3] architecture cannot meet the needs of MIMO systems. Instead, a statically scheduled connection solution based on a switching network and global/local memories is introduced in this paper to deal with the large quantity of data transfer and storage. The improved architecture is general enough to handle all of the baseband operations required in both ordinary OFDM systems and in MIMO-OFDM systems.

In addition to the architectural issues, the implementation method optimization and algorithm-level optimization also play an important role in improving performance. This paper will explore two algorithm implementations on the reconfigurable baseband platform widely used in MIMO-OFDM systems, namely Fast Fourier Transform (FFT) and Space Time Block Decoding (STBD), according to IEEE 802.11n and 802.16e standards. A variety of sizes of FFTs are required, such as 64-point and 128-point FFTs in 802.11n [4], and 2 K-point FFTs in 802.16 [5]. In addition, different types of modulation, such as BPSK, 64-QAM, 128-QAM, etc., will affect the operation of the STBD. In order to handle such a broad range of possibilities, flexible computational resources, data storage and data flow mechanisms are required.

The remainder of this paper is organized as follows: Sect. 2 gives an overview of the proposed baseband architecture platform. Section 3 describes the FFT and STBD algorithms and their optimizations. Section 4 shows an implementation of the proposed algorithms on improved architecture. Section 5 contains the performance results and comparisons, and Sect. 6 presents our conclusions.

2. Previous Work and Its Enhancement

2.1 Overview of Previous Work

Most of the existing reconfigurable architectures use an FPGA computational model which consists of a set of fine-grained to medium-grained homogenous modules [6], [7], or embedded reconfigurable logic cores [8]. While such techniques provide good flexibility and work well with compilation tools, they can lead to a high interconnection com-

Manuscript received July 15, 2009.
Manuscript revised November 19, 2009.

[†]The authors are with the State Key Lab of ASIC and System, Fudan University, China.
[††]The author is with the Dept. of Electrical and Computer Engineering, University of Minnesota, USA.
a) E-mail: xiaofangzhou@fudan.edu.cn

DOI: 10.1587/transinf.E93.D.811

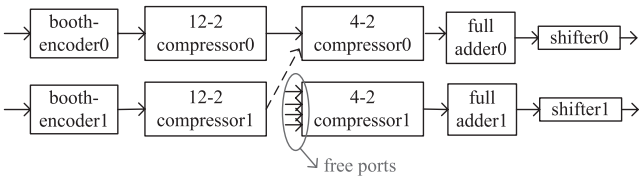


Fig. 1 Structure of an RAU slice.

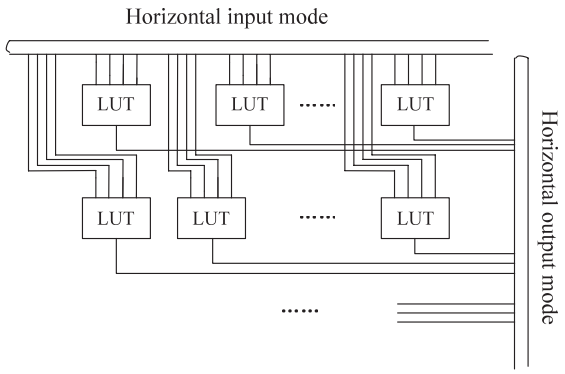


Fig. 2 Structure of RLU slice in horizontal mode.

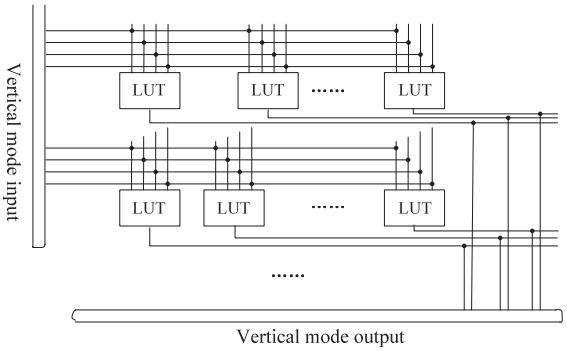


Fig. 3 Structure of RLU slice in vertical mode.

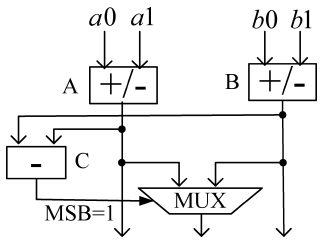


Fig. 4 Structure of an AAU slice.

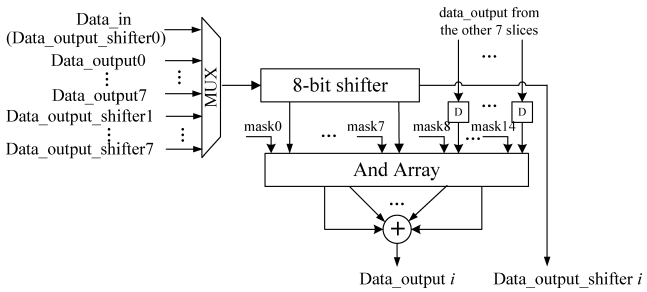


Fig. 5 Structure of a BSU slice, assuming there are 8 slices with the BSU.

plexity. Moreover, those approaches are not specifically tailored to the set of algorithms used in wireless baseband communications systems, and thus lack of a certain degree of computational efficiency for those kinds of applications. While some previous work has been done on reconfigurable baseband processor design [9]–[11], most communications applications are currently implemented as an ASIC solution. Our approach is based on analyzing the computational features which are widely used in wireless communication baseband processing flows, and thereby constructing an appropriate set of coarse-grained elements to support those computations. Four types of Execute Units (EU) are chosen: Reconfigurable Arithmetical Unit (RAU), Reconfigurable Logic Unit (RLU), Bit Serial Unit (BSU) and Adder Array Unit (AAU).

The RAU handles multiplication and addition calculations. As shown in Fig. 1, two adjacent slices can provide a combine-mode with the dashed arrow to complete one $a \times b + c \times d$ operation.

The RLU is designed as set of look-up table (LUT) elements. Each basic slice contains a 16×4 array of 4-1 LUTs and provides horizontal and vertical input/output modes. The horizontal mode, as shown in Fig. 2, imports/exports data to/from LUTs in one or more rows, whereas the vertical mode of Fig. Figure 3 provides data input/output to/from LUTs in one or more columns. Since the content of each LUT is programmable, it can support all bitwise Boolean operations by specifying a truth table.

The AAU is another module designed for arithmetic operations. Unlike the RAU, the AAU module focuses on addition-intensive operations, such as data comparison, add-select-compare, etc. Each AAU slice contains two 16-bit adders/subtracters and one 17-bit subtracter, as shown in Fig. 4. Therefore, it can complete three add/subtract operations in a single step. This circuit provides three outputs: $a0 \pm a1$, $b0 \pm b1$ and a comparison results. Note that if $a1 = 0$ and $b1 = 0$, $a0$ and $b0$ will be compared; furthermore, if $b0 = -a0$, the comparison result will be the absolute value of $a0$.

The BSU module processes bit-serial calculations used in scrambling, randomization, convolution, etc. Each BSU slice consists of an 8-bit shifter, an AND array and an XOR logic operation, as shown in Fig. 5. The MUX select and And Array mask signals are specified as part of the reconfigurable information.

The above EUs are combined with a register-based interconnection and storage mechanism to form a WLAN reconfigurable processor [3]. Each reconfigurable EU (RC EU) contains copies of these basic slices, and the number of slices can be scaled according to the needs of the application. In addition, they are independently controlled by an Engine, where all of the configurable information and con-

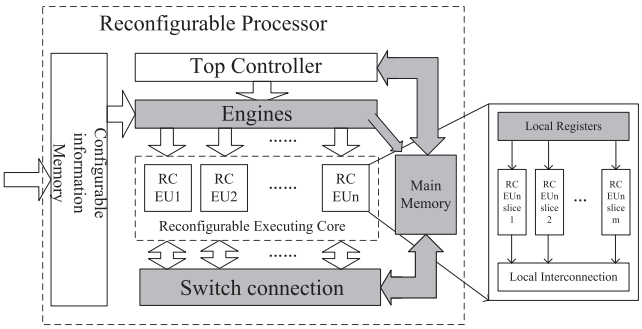


Fig. 6 The enhanced reconfigurable architecture.

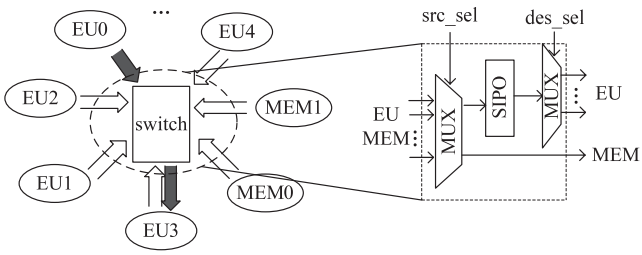


Fig. 7 The switch connection structure.

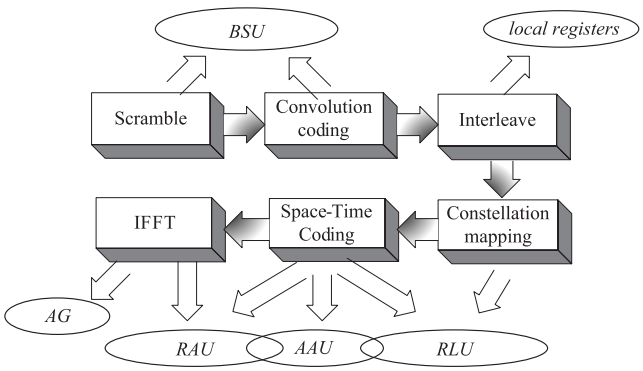


Fig. 8 The typical MIMO-OFDM system processing flow.

trol logic for the currently implemented operations are contained.

2.2 Architecture Enhancement

In order to support MIMO-OFDM systems, we add main memory and local registers to the previous platform, provide a global interconnection among EUs, and we also enhance the Engine logic, as shown in the shaded blocks of Fig. 6.

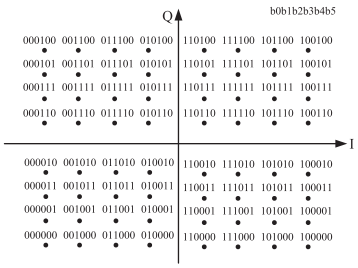
For multi-link applications, a multi-bank Main Memory is required to hold data from different channels. Each bank works independently so that multiple channels can be supported at the same time. Local registers are also necessary to reduce access frequency into the Main Memory. These local registers hold the results generated in the same EU module, or they may act as a first- in first-out (FIFO) buffer. In this way, the data to be processed in the next step can be fetched directly from local registers where the results from the last step are held, allowing the EU to operate continuously. Results may also be exported to the Main Memory for subsequent use. Memory addresses for both local and global memories are produced by address generation logic with the same structure in the Engine.

There are several possible alternatives for the Network-on-Chip (NoC) structure, such as multicasting or broadcasting, 2-D/3-D mesh, and router, but these are overly complex for our purposes. We only need to ensure that the data paths follow the required baseband processing flow. For example, data which are processed by the Forward Error Correcting (FEC) unit will be subsequently processed by constellation mapping and then the Inverse Fast Fourier Transform (IFFT) calculations. As a result, all the data traffic flows can be constrained to occur within a narrow range. In order to balance complexity and flexibility, a statically scheduled switch interconnection is used which is similar to a multiplexer-based multi-bus architecture. The multiplexer and de-multiplexer are controlled by external logic according to the requirements of the data traffic. A shifter-like sequential-in, parallel-out (SIPO) register is used to account for any mismatch between input and output ports, since the output bandwidths of each RC EU are different from their input bandwidths. Its depth L is determined by the maximum transferred data width, and the shift amount s leads to a shift delay of $\lfloor \frac{L}{s} \rfloor$. Meanwhile, the switch provides a fast link between RC EUs and global memory to improve memory access efficiency. An example of message passing is illustrated in Fig. 7 from EU0 to EU3 along with the detailed switch structure. The switch uses instruction fields src_sel, which selects the source data to be transferred, and des_sel, which determines the intended destination.

Figure 6 illustrates the entire proposed architecture. Most of the algorithm operations can be handled by one or more EUs just by changing configurable information flows which is initially stored in Config Memory. EUs are directly controlled by the Engine which receives and decodes the instructions coming from the top controller (a RISC processor), fetches the corresponding configurable information from the Config Memory, and finally gives the orders.

Figure 8 shows a typical MIMO-OFDM baseband processing flow (Tx) [4], [5], and the Rx takes the inverse operation. In some applications, not all of these algorithms are required, but the typical data flow follows the processing order shown. The number of channel links may also be different, but this can be accommodated by including additional or fewer EU slices. All of the operations can be done by the proposed hardware structure.

Scrambling and convolutional encoding operations are performed by the BSU. Two BSU slices are used to implement the scrambling operation. The first BSU slice generates a pseudo-random sequence by passing the XOR result to the input of the 8-bit shifter. The second BSU slice forms a scrambled bit by performing the XOR operation on the pseudo-random number and the input data sequence during every clock cycle. Another two BSU slices are used to im-



I/Q path input (b0b1b2/b3b4b5)	LUT output	
000	-7	1001
001	-5	1011
010	-1	1111
011	-3	1101
100	7	0111
101	5	0101
111	3	0011
110	1	0001

Fig. 9 An example of 64-QAM constellation.

plement the (2,1,7) convolutional encoding. The result after scrambling is sent to the inputs of these two slices and each slice will generate one bit according to the generator polynomials. The two output bits are then controlled by the Engine to carry out interleaving.

Symbol mapping operations are accomplished by the RLU, and may be BPSK, QPSK, 16-QAM or 64-QAM. The constellation of each can be translated into a truth table; Fig. 9 presents an example of 64-QAM constellation. Two RLU slices under the 64-QAM mapping can generate 16 mapped symbols at a time; each one slice processes I/Q path separately by the vertical mode.

Finally, the mapped symbols are sent to the Space-Time Coding (STC) and IFFT units, which are the most computationally challenging parts, and which are described in the following sections.

The Viterbi algorithm is typically used to perform convolutional decoding. It can be divided into two parts, add-select-compare and trace-back. RLU and AAU slices can compute the add-select-compare operation, and the general processor assists with the trace-back. The RLU generates the Hamming distance of the input bits and state bits via table look-up. Two distances and their corresponded metrics are added by A/B adders in one AAU slice separately, and the results are passed to the C adder, thereby generating the select the Most Significant Bit (MSB). Both the selected result and the select MSB are sent to the processor for the trace-back computation.

3. Algorithm Optimization

Implementation method and algorithm level optimization are important for the overall system process. Here, we will present the optimized Fast Fourier Transform (FFT) implementation method and the optimized reference for one of the widely used STC decoding method, Space Time Block Decode (STBD) algorithm.

3.1 FFT Optimization

FFT computation is a bottleneck in OFDM baseband processing. According to IEEE 802.11n and 802.16e, 64-point up to 2048-point FFTs are used for different data rates. Previously, various FFT architectures and algorithms have been developed, such as a radix-2 bit-reversed algorithm with a memory-based structure [12], radix-n algorithms with a pipeline structure [13], a cached-FFT architecture [14], etc.

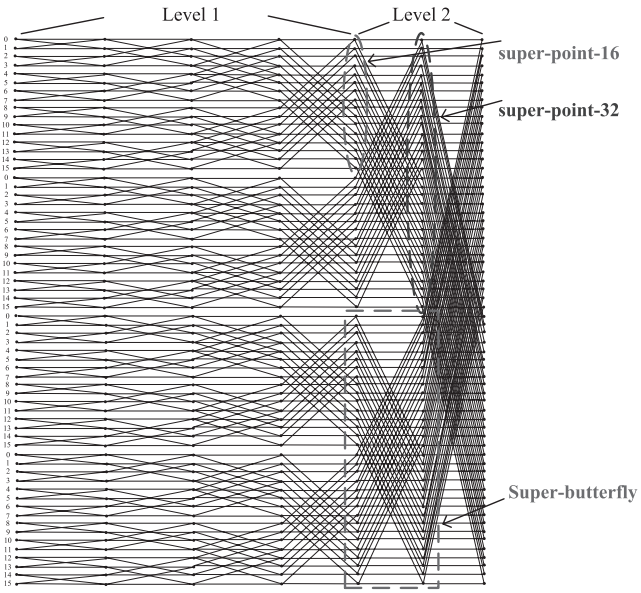


Fig. 10 64-point FFT processing flow.

The conventional radix-2 bit-reversed algorithm is best suited for small FFT sizes due to the high data access frequency and its resulting power dissipation. However, it is the easiest way to implement an FFT with the least amount of control logic. The pipeline architecture is suitable for large-size FFTs, since it partitions a large computation into several smaller sets of calculations. As a result, a large memory can be replaced by a series of small shift registers according to the given computation size. Although the pipeline structure saves on power consumption for memory accesses as well as being scalable to large-size FFTs, it does not significantly reduce the total amount of storage required and it greatly increases the control complexity. The cached-FFT structure is another approach for reducing the cost of memory accesses. It inserts small size cache between the FFT computation logic and the main memory. In this way, the cache handles most of the storage accesses, and it is also much faster than main memory. A cached-FFT structure balances processing efficiency and power dissipation.

For our hardware platform, we propose an FFT solution based on the conventional radix-2 bit-reversed principle and a cached-FFT structure to handle both small and large size FFT computations.

3.1.1 Calculation Method

A typical cached-FFT structure seeks to have most data movements occur between the processor and cache so that the average memory access time can be reduced. Based on this idea, a 64-point FFT processing flow can be partitioned as shown in Fig. 10. In the 1^{st} level, 64 points are divided into 4 groups, each consisting of 16 basic points. These 4 groups of data are calculated as four independent basic 16-

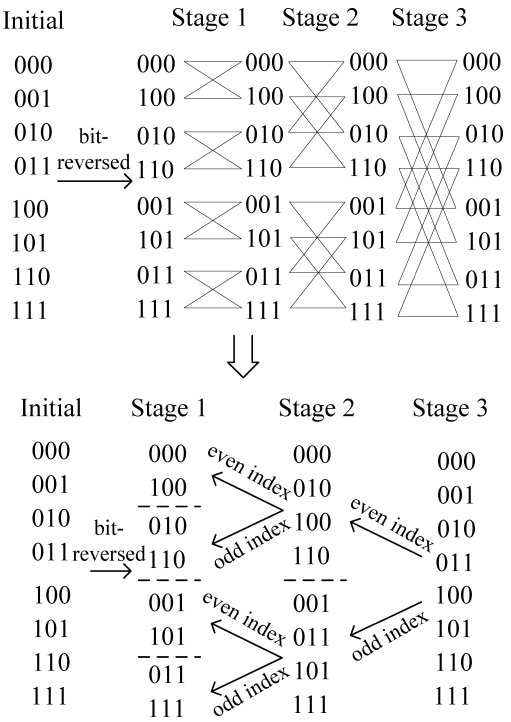


Fig. 11 Conventional 8-point FFT bit-reversed address generation procedure.

point FFTs, and four groups of results are obtained; each of these results is then defined as a super-point-16. As a result, a 4-super-point FFT computation is implemented in the 2^{nd} level. Due to symmetries in the data flow diagram, the same address generator can be used at both levels.

Therefore, a 64-point FFT can be simplified to consist of four basic 16-point FFTs and a 4-super-point FFT. In this way, large FFTs can be separated into several smaller-size operations. Moreover, the conventional address generation mechanism based on the bit-reversed radix-2 principle can be used, as will be illustrated later. The first level prepares data for the super-point FFT computation, and the size chosen mostly depends on the cache size. The number of the FFT groups in the first level is the size of the super-point FFT calculation. Moreover, the size of super-point FFT can be further reduced by introducing super-point-32 or more, according to cache constraints.

3.1.2 Address Generator

In this section, a novel address generator is proposed as a part of the Engine control block. Taking an basic 8-point FFT as an example, we will first review the conventional radix-2 bit-reversed algorithm.

The values shown in Fig. 11 are the addresses at which the data are stored; the column at the left of the butterfly shows the addresses the data are fetched from, and the column at the right are the addresses the results are stored in. Using bit-reversed ordering and an even-odd separation, the operands and results can share the same locations. In the

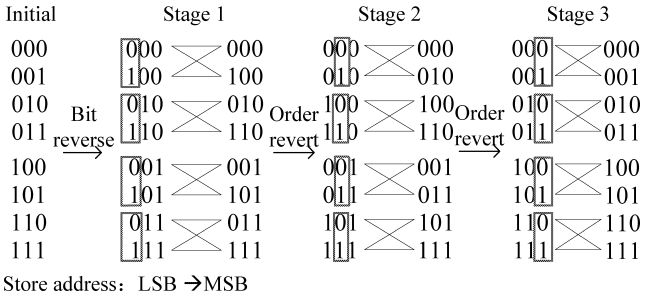


Fig. 12 Reordered 8-point FFT address generation flow.

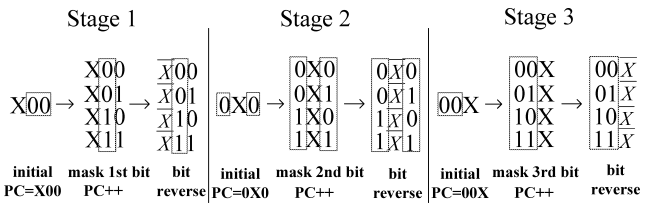


Fig. 13 Proposed address generation procedure.

conventional view, address generation for each stage is used to obtain the connections from the $(n+1)^{th}$ stage back to the n^{th} stage. However, from an implementation point of view, the only thing that needs to be considered is the relationship between the two operand addresses within the same butterfly. Therefore, given one operand address, if we find the other one, the corresponding butterfly can be completed. Based on this view, the address processing flow can be reordered, as shown in Fig. 12.

Assume that an N-point FFT ($N = 2^m$) is computed and n is any stage in the computation($1 \le n \le m$). In the n^{th} stage, the n^{th} address bit of the operands in a butterfly are opposite, as marked in Fig. 12. As shown in Fig. 13, the n^{th} bit in the n^{th} stage is masked, and a search of the remaining (m-1) bits by PC accumulation is done to list all 2^{m-1} possible binary values, from all zeros to all ones. After appending the n^{th} bit to the searched value, one operand address is obtained. Then, by reversing the same bit and keeping the others fixed, the other operand address is obtained.

In the actual hardware implementation, the masked bit X in Fig. 13 is set to 1. According to the proposed address generation algorithm, for an N-point FFT calculation, an m-bit shifter, m-bit register, m-bit counter and an m-bit bitwise XOR logic are needed, as shown in Fig. 14. The shifter is used to store the initial point counter (PC) value; in the n^{th} stage, the n^{th} bit is set to 1 and the others are set to zero. After the n^{th} stage is finished, the only "1" bit will be right-shifted for the $(n+1)^{th}$ stage computation. The m-bit register is for the PC value which is accumulated on each cycle $T_i (i = 1, 2, \cdots, 2^{m-1})$, and the counter records the PC. The value in this register is the address of one operand. After the bitwise XOR with the initial PC value, the other address is generated. When the value in the register becomes all ones, the current stage completes, and the shift command enables the next stage.

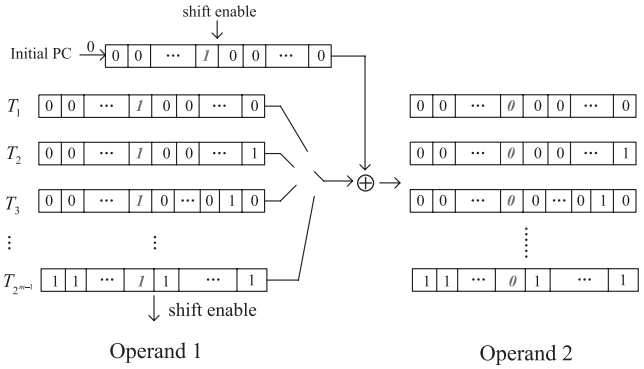


Fig. 14 Proposed address generator.

3.2 STBD Optimization

STBC (Space-Time Block Coding) is used in MIMO systems in a low SNR environment to improve communication reliability. Its inverse operation, STBD (Space-Time Block Decoding), is on the critical path in the receiver and directly affects the overall receiver efficiency. The most precise way to perform STBD is with Maximum Likelihood (ML) detection. If a 2×2 antenna matrix is used, then the generation matrix and ML detection formula can be described in the following two equations, respectively [15]:

$$G_2 = \begin{bmatrix} c_1 & c_2 \\ -c_2^* & c_1^* \end{bmatrix} \tag{1}$$

$$\begin{aligned} M_{G2}(c_1, p_k) = & \left| \left[\sum_{j=1}^{m} (r_1^j \alpha_{1,j}^* + (r_2^j)^* \alpha_{2,j}) \right] - p_k \right|^2 \\ & + \left(-1 + \sum_{j=1}^{m} \sum_{i=1}^{n} |\alpha_{i,j}|^2 \right) |p_k|^2 \end{aligned} \tag{2}$$

Here, $c_i (i = 1, 2)$ are the transmitted signals from the i^{th} transmit antenna; $r_i^j (j = 1, 2)$ are the received signals of the j^{th} receive antenna from the i^{th} transmit antenna; α_i^j is the fading factor of the channel from the i^{th} transmit antenna to the j^{th} receive antenna; $p_k (k = 1, 2, \cdots, M)$ is one of the $M - ary$ modulated symbols. The smallest metric value is the final decision, as shown in [16]:

$$\begin{aligned} c_1 = p_k \Leftrightarrow & M(c_1, p_k) \\ = & min\{M(c_1, p_1), M(c_1, p_2), \cdots, M(c_1, p_k)\} \end{aligned} \tag{3}$$

Therefore, the complexity of ML detection depends linearly on the size of the modulation constellation [15]. In order to find the best symbol, one has to search over all constellation points. A variety of algorithms have been proposed to achieve higher efficiency and lower computational cost, such as ZF (Zero-Forcing) [17], MMSE (Minimum Mean-Square-Error) [18], SIC (Successive Interference Cancellation) [19], but these achieve only limited reductions in the complexity. The sign approach [16], however, reduces the constellation size and achieves greater efficiency. It is briefly reviewed below.

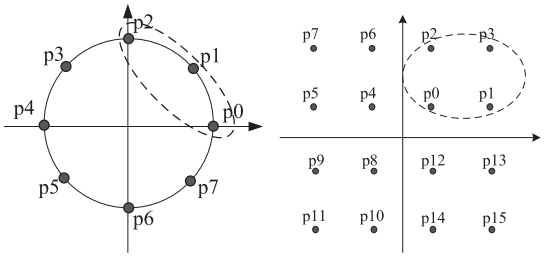


Fig. 15 8-PSK and 16-QAM constellations (after [16]).

3.2.1 Sign Approach

As described in [16], all the decision metrics can be unified as

$$M(c_n, p_k) = |(a_n + jb_n) - p_k|^2 + \beta |p_k|^2 \tag{4}$$

Taking (2) as an example:

$$a_1 + jb_1 = \sum_{j=1}^{m} (r_1^j \alpha_{1,j}^* + (r_2^j)^* \alpha_{2,j}), \beta = \left(-1 + \sum_{j=1}^{m} \sum_{i=1}^{n} |\alpha_{i,j}|^2 \right)$$

If we let $p_k = p_{kx} + jp_{ky}$, and substitute into (4), we obtain $(n = 1, 2)$:

$$\begin{aligned} M(c_n, p_k) = & a_n^2 + b_n^2 + p_{kx}^2 + p_{ky}^2 - 2(a_n p_{kx} + b_n p_{ky}) \\ & + \beta (p_{kx}^2 + p_{ky}^2) \end{aligned} \tag{5}$$

As a_n^2 and b_n^2 are common terms for all constellation point comparisons, their effects can be ignored. For equal energy modulation schemes such as M-PSK, $p_{kx}^2 + p_{ky}^2$ always equal 1, and the term -2 is common, so they can also be ignored and the metric can be simplified to that of (6) for purposes of finding the extreme values; however, for unequal energy modulation schemes, the amplitude effect can not be ignored, resulting in (7).

$$M(c_n, p_k) = a_n p_{kx} + b_n p_{ky} \tag{6}$$

$$M(c_n, p_k) = p_{kx}^2 + p_{ky}^2 - 2(a_n p_{kx} + b_n p_{ky}) + \beta (p_{kx}^2 + p_{ky}^2) \tag{7}$$

In this way, the maximal $M(c_n, p_k)$ in (6) and (7) only occur when a_n, p_{kx}, and b_n, p_{ky} have the same signs. In this way, comparison is limited to one quadrant, and thus the candidate points are reduced by nearly three quarters. Therefore, the computational efficiency can be dramatically improved.

3.2.2 Case Study

Assuming that the first quadrant is selected, for 8-PSK, the comparison points are reduced to be $p0$, $p1$ and $p2$; for a 16-QAM, they are $p0$, $p1$, $p2$, $p3$, as circled in Fig. 15. Therefore, Eqs. (6) and (7) are applied to these signals. Note that the final result c_k is the corresponding p_k in $max\{M(c_n, p_0), M(c_n, p_1), M(c_n, p_2)\}$ or $max\{M(c_n, p_0), M(c_n, p_1), M(c_n, p_2), M(c_n, p_3)\}$.

4. Algorithm Implementations

In this section the FFT and STBD algorithms will be implemented on the proposed architecture, and the performance enhancement will be demonstrated. Here, the "Memory Access Cycle (MemAC)" is the period that one RC EU transfers its local data to the main memory when the current local computation and storage is temporarily completed; the "memory access cost" means the total timing cost for memory access during certain operation; the "switch access cost" means the timing cost on switch transfer; the "timing cost" is the execution time for certain operation, including memory access cost.

4.1 FFT

According to IEEE 802.11n and 802.16e, 64-point up to 2048-point FFTs are required. Moreover, in the high-throughput mode of IEEE 802.11n, the frame time required for a 128-point FFT is 4 μsec including all the baseband operations. This is the most critical timing requirement among the supported sizes. The maximal clock frequency of our architecture is 100 MHz at the post-layout level, and 8 RAU slices can handle one butterfly per clock cycle. At least 64 slices are needed, 16 per channel link, with a total of 4 links supported. In this way, during one clock cycle, two butterflies are handled, and the 128-point FFT can be finished within 2.24 μsec, excluding memory access cost. According to the proposed method, local registers are used instead of a cache memory since the EU has such local memory, which guarantees the highest access efficiency within the EU. Considering the area cost of local registers and the main memory access frequency, 16 basic points of data per channel link can be held in RAU local registers and each point can be quantized to a maximum of 32 bits, with 16 bits each for the real and imaginary parts. In this design, two 1 $K\times$128-bit dual-port memories with 4 banks are used; one is used for processing the current frame, while the other is used for the next frame. Therefore, at the same time that the current frame is being processed, the next frame of data can be received. Additionally, the dual ports make it possible to fetch the operands for one butterfly computation within a single clock cycle.

As shown in Table 1, a 128-point FFT will be used to illustrate the procedure. It starts with a set of basic 16-point FFTs; 8 local access cycles are needed to complete this first level of processing. Each local access cycle includes 4 stages of basic 16-point FFT and each stage takes 4 clock cycles, or 16 clock cycles in total. Addresses of local registers are generated by the proposed AG circuit with 4-bit values. The results of each Basic 16-point FFT are sent to the main memory and stored successively as one MemAC; 8 instances of super-point-16 MemAC are needed. At the second level, an 8-super-point FFT starts which includes 3 stages. The basic points within one super-point-16 have the same macro address but different bias addresses; on the other hand, the two operands of a basic butterfly in the corresponding super-butterfly have the same bias address but different macro addresses, as shown in Fig. 16. Therefore, the macro address can be generated by the proposed address generator using 3 bits, and the bias address can be indicated by a point counter. If 16 basic points are accessed in one MemAC, two basic 8-point FFTs can be implemented per channel link during one MemAC; in total, 8 MemAC are needed to complete the 8-super-point FFT, as shown in Fig. 17. Results are stored back to memory using the "same-address" principle.

Table 1 128-point FFT implement flow.

	EU/ memory	operation	MemAC	clock cycles
1	16 RAU slices local register	1^{st} basic 16-point FFT	1	16
2	16 RAU slices local register	2^{nd} basic 16-point FFT	1	16
	main memory	1^{st} super-point-16 storage	1	8
...				
8	16 RAU slices local register	8^{th} basic 16-point FFT	1	16
	main memory local memory	7^{th} super-point-16 storage 8-super-point FFT fetch (PC=0,1)	2	16
9	16 RAU slices local register	1^{st} basic 16-FFT in super 8-super-point FFT (stage 1)	1	16
	main memory local memory	8^{th} super-points-16 storage 8-super-point FFT fetch (PC=2, 3)	2	16
...				

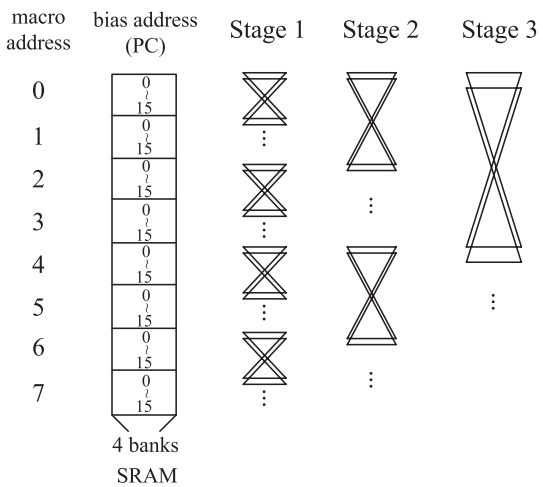


Fig. 16 Macro and bias address mechanism in the super-point FFT computation.

4.2 STBD

Consider an 8 × 4 channel matrix with a coding rate of 1/2, one receive antenna ($m = 1$) and 16-QAM modulation, and use c_1 detection as an example. The required operations of STBD include the summation of the attenuated received signals from all the transmit antennas, signs and

values comparison, etc. In this hardware architecture, the RAU is used for multiplication and summation; the RLU can handle the constellation quadrant decision; combined with an RAU slice, the AAU handles the data comparison. Moreover, the combining mode of the RAU slice releases a 4-2 compressor per every two adjacent slices, as marked in Fig. 1. Table 2 explains the detailed computation flow.

The proposed structure of execute units is well suited to performing regular, repeated computations. Therefore, it is advantageous to reduce the control logic and implement a consistent pattern of arithmetic and logic operations. From this point of view, instead of a bias value search [16], we directly use (7) to implement QAM modulation. Assuming that the channel condition is slow fading or pseudo-static, the attenuation factor α remains fixed for at least one frame. Therefore, β only needs to be computed once at the very beginning of each frame. Furthermore, the points on the constellation are fixed during the entire processing, and thus, $A(p_{kx}^2 + p_{ky}^2)$ can be calculated in advance as fixed information. As a result, the quadrant decision and the minimal/maximal data selection are the only time-critical computations for each received signal. Using an 8×4 channel matrix with one receive antenna and 16-QAM, 8 multiply-with-addition computations, 16 sign comparisons among the constellation points, and the minimum value selection among 4 data values are required. In our architecture, two RAU slices can complete one $a \times b + c \times d$ operation, using 16 bits for each operand. In order to reduce the processing time as much as possible, we introduce 17 RAU slices, 2 RLU slices and 1 AAU slice to decode one symbol. Notice that all the steps can occur simultaneously, so that only 2 clock cycles are needed to decode one symbol, excluding the pre-computed information and the switch access cost. If there are more receive antennas, more RAU slices will be used in step 1. Similarly, a larger constellation size will require additional RLU slices in step 1, RAU slices in step 2 and AAU slices in step 3. As a result, the total resource cost of STBD is determined by the size of channel matrix, the number of receive antennas and the constellation size.

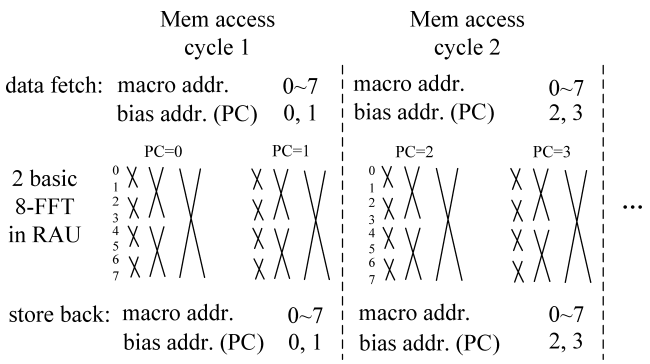


Fig. 17 8-Super-point FFT memory access cycles.

Table 2 STBD processing flow.

	EUs	slices	operations	cycles
1	RAU	9	$a_1 + jb_1 = \sum_{j=1}^{m}(r_1^j\alpha_{1,j}^* + (r_2^j)^*\alpha_{2,j} + r_3^j\alpha_{3,j}^* + (r_4^j)^*\alpha_{4,j})$	2
	RLU	2	$\{p_{k1}, \cdots, p_{k4}\} = \{sign(a_n, p_{kx}) = 1$ && $sign(b_n, p_{ky}) = 1\}$	2
2	RAU	8	$C_i = A(p_{kx}^2 + p_{ky}^2) - (a_n p_{kix} + b_n p_{kiy})$ $A = 1 + \beta, (i = 1, 2, 3, 4)$	2
3	AAU	1	$min(C_i), (i = 1, 2, 3, 4)$	2

5. Performance Analysis and Comparison

5.1 Instruction Generation

In our current system implementation, the algorithms are manually mapped onto the architecture. We generate the configurable instructions step by step as a script, and then use Perl to translate the script into a Verilog testbench file. Most of the operations require more than one clock cycle to execute. We classify the instructions into two categories, static instructions and dynamic instructions. The parameters of a static instruction remain fixed throughout its execution, while those of a dynamic instruction will change during the time that it is executing.

5.2 Performance Analysis

In this section, we will present the performance results for the enhanced hardware and the algorithm implementations. In the FFT implementing, we use 64 RAU slices for computations, local registers having a capacity of 64 points and two $1K \times 128$ dual-port SRAMs for data storage. In the case of the STBD implementation, we use 17 RAU slices, 4 RLU slices and 8 AAU slices to support at most four data sequences with 64-QAM modulation. Each execute unit as well as the global interconnection structure has been post-layout using a UMC 0.18 μm technology under the clock rate of 100 MHz, and the areas and bandwidths are listed in Table 3. Each data stream operates independently and occupies the same computational resources. If fewer than four channel links are required, the hardware can be scaled down in order to lower the power and chip size.

Table 3 Performance of execute units and switch (under 100 MHz clock frequency).

EU	Source included	P&R core area (mm^2)	Max input bandwidth (Mbps)	Max output bandwidth (Mbps)
RAU	64 RAU slices local memory I/O control	11.2	102,400	102,400
RLU	4 RLU slices I/O control	0.1	102,400	25,600
BSU	12 BSU slices I/O control	0.045	102,400	3,200
AAU	8 AAU slices I/O control	0.2	102,400	25,600
switch	4 switch	0.96	102,400	25,600
Global memory	2 128×1 K-bit dual-port SRAM	2×1.97	25,600	25,600

Table 7 Comparison of FFT implement (ONE DATA SEQUENCE).

Architecture	Tech. (μm)	FFT size	Memory size (bit)	T_FFT (μs)	register/ cache size	MemAC	Core area (mm^2)
[3]	0.18	64	0	1.92	2 K-bit	0	4.8
proposed	0.18	64	64x32	1.36	2 K-bit	8	<3.41

Table 8 Performance comparison of address generation (AG) circuit.

Hardware cost ($N = 2^m$)		
D.Cohen [23]	Yutai Ma [24]	Proposed
4 barrel shifters of N-by-m-bit	2 barrel shifters of N-by-m-bit	1 barrel shifter of m-bit
2 (N-1)-bit butterfly counters	2 (N-1)-bit butterfly counters	1 m-bit butterfly counter
2 (N-1)-bit XOR trees		1 m-bit XOR
4 w-bit MUX 4 (N-1)-bit MUX	2 w-bit MUX	
2 m-bit pass counters	2 m-bit pass counters	

Table 4 FFT performance for different numbers of points.

size	Timing cost (μs)	memory access cycle
64	1.36	8
128	2.64	16
256	5.2	32
512	11.6	72
1024	23.12	192
2048	56.32	384

Table 5 STBD performance for different modulations.

(for one symbol decoding)	clock cycle	RAU slice	RLU slice	AAU slice
BPSK (2×2)	2	2	1	0
QPSK (4×4)	2	8	1	0
8 PSK (4×4)	2	8	1	1
16 QAM (4×4)	2	17	2	1
64 QAM (4×4)	3	17	4	8

Table 4 illustrates the timing cost and memory access cycle for calculating different sizes of FFTs. With the same circuit area, they increase linearly with the size of the calculation. Table 5 gives the STBD operation time and hardware cost for different modulation schemes. Since the quadrant decision for BPSK and QPSK directly gives the minimum value result, the AAU structure is not needed. Therefore, the computation primarily depends on the channel matrix size and the number of receive antennas. The processing time stays within a narrow range, while the hardware cost increases linearly with the constellation size. In addition, the simulated timing results in Table 4 and Table 5 easily meet the timing requirements defined in IEEE 802.11n and 802.16e.

5.3 Comparison

There are other types of reconfigurable architectures that have been developed for baseband processing of MIMO-OFDM or OFDM systems, such as [9], [20], [21], and [10]. [10] uses a GPP/DSP core with accelerators, and [8] uses an FPGA core. [9], [20], [21] are homogenous coarse-grained reconfigurable architectures with specifically designed modules. A homogeneous array can provide a regular instruction flow and interconnection structure, but it is less efficient for the specific set of computations needed. Moreover, a regular interconnection structure limits the achievable communication bandwidth; for example, the throughput of our implementation is 102.4 Gbps, which is almost twice that of [21].

Table 6 Performance of interconnect structure comparison (under 100 MHz clock frequency).

Structure	Bandwidth (Mbps)	average area (μm^2)	average latency (cycle)
2D NoC Mesh	102,400	1,657,745	92
CDMA 4-bit [22]	102,400	1,647,982	16
CDMA 8-bit [22]	102,400	2,504,533	16
ship based [3]	25,600	2,153,797	4
Proposed	102,400	1,722,407	4

Table 6 shows the average area and latency of different interconnect architectures with different bandwidths, using the same clock frequency and the same CMOS process technology (UMC 0.18 μm). The meaning of average area here is the area consumption of the global interconnection logic which can provide the transmission bandwidth for one channel link. The average latency is packet transmission delay based on clock cycle from data receiving to transmitting of one interconnection logic. CDMA-based NoC mesh topology is proposed in [22], which is a more general architecture especially for multi-casting and with lower area cost, but requires longer transmission latency. The ship-based interconnection is used in our previous work. Although it provides a very high flexibility to interconnect all the other RC EUs, it costs more resources, especially for larger data flows.

Table 7 presents a comparison of the hardware cost, execution time and memory dissipation for the FFT processing on the previous and the enhanced architectures. With the hierarchical memory structure and the super-point mechanism, the execution time is lower than before. The equivalent area for one data sequence is also decreased to 3.41 mm^2 including the control logic for data selection of multiple channels. Therefore, the actual area for single channel processing is less than 3.41 mm^2.

The proposed super-point mechanism also makes it convenient to use the same address generator structure for both local registers/cache and main memory. Moreover, the

Table 9 Comparison of STBD implementations on ASIC and reconfigurable architectures.

	precision	multiplier	adder	cycle /symbol
[16]	9 bits	14	19	2
This paper	16 bits	16	18	3

hardware cost in our scheme is quite low as shown in Table 8. As described in Sect. 3, it can be easily scaled for various sizes of FFTs.

In terms of STBD, E. Canvus [16] describes a custom-designed ASIC architecture for at most four receive antennas. Table 9 shows the resource costs and performance comparison for reconfigurable and ASIC structures, both of which are based on 16-QAM modulation. The ASIC solution aims at architecture simplification and resource economization. On the other hand, the goal of the reconfigurable architecture is to improve computational flexibility. As a result, sufficient hardware resources must be allocated to handle the maximal set of requirements, and this leads to its larger number of bits of precision.

6. Conclusion

This paper has mapped the FFT and STBD algorithms onto an enhanced reconfigurable baseband platform according to IEEE 802.11n and 802.16e. The FFT implementation uses an optimized implementing method proposed in this paper, which takes advantage of the features of the platform's architecture. As part of the FFT optimization, we utilize local registers within the RAU execute unit to serve as a cache. In addition, the memory access mechanism and address generator can make use of the radix-2 bit-reversed technique. The STBD processing is optimized through the use of precomputed values. For both algorithms, the implementing processes are carefully designed to fully utilize the available hardware resources and to reduce the required number of clock cycles. A comparison with other optimization methods and implementations show that a good balance between performance and hardware cost is achieved, while still providing adequate flexibility to support other portions of the baseband processing flow.

Acknowledgement

This work is supported by China NSF 60876016, State Key Lab of ASIC and Systems ZD20080103 and GF20080306.

References

[1] H. Yu, M.S. Kim, E.Y. Choi, T. Jeon, and S.K. Lee, "Design and prototype development of mimo-ofdm for next generation wireless lan," IEEE Trans. Consum. Electron., vol.51, no.4, pp.1134–1142, Nov. 2005.

[2] N. Huaning and N. Chiu, "Diversity and multiplexing switching in 802.11n mimo systems," Proc. Asilomar conference on signals, systems and computers (SSC'06), pp.1644–1648, Pacific Grove CA, USA, Nov. 2006.

[3] W. Lu, S. Zhao, C. Lu, X. Zhou, D. Zhou, and G.E. Sobelman, "A heterogeneous reconfigurable baseband architecture for wireless lan transceivers," Proc. IEEE International Conference on Electro/Information Technology (EIT'08), Ames IA, USA, pp.284–288, May 2008.

[4] "Wireless LAN medium access control (MAC) and physical layer (PHY) specification amendments 4: Enhancements for higher throughput," 2007.

[5] "Air interface for fixed and mobile broadband wireless access system amendments 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands," 2005.

[6] Y. Satou, M. Amagasaki, H. Miura, K. Matsuyama, R. Yamaguchi, M. Iida, and T. Sueyoshi, "An embedded reconfigurable logic core based on variable grain logic cell architecture," Proc. IEEE International Conference on Field-Programmable Technology (ICFPT'09), Kitakyusyu, Japan, Dec. 2007.

[7] M. Myjak and J. Delgado-Frias, "A medium-grain reconfigurable architecture for dsp: Vlsi design, benchmark mapping, and performance," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.16, no.1, pp.14–23, Jan. 2008.

[8] H. Wang, P. Leray, and J. Palicot, "An efficient mimo v-blast decoder based on a dynamically reconfigurable fpga including its reconfiguration management," Proc. IEEE International Conference on Communications (ICC'08), Beijing, China, May 2008.

[9] A. Poon, "An energy-efficient reconfigurable baseband processor for wireless communications," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.15, no.3, pp.319–327, March 2007.

[10] E. Tell, A. Nilsson, and D. Liu, "A low area and low power programmable baseband processor architecture," Proc. IEEE International workshop on System-on-Chip for Real-Time Applications (IWSOC'05), Banff, Canada, July 2005.

[11] C. Ebeling, C. Fisher, G. Xing, M. Shen, and H. Liu, "Implementing an ofdm receiver on the rapid reconfigurable architecture," IEEE Trans. Comput., vol.53, no.11, pp.1436–1448, Nov. 2004.

[12] S. Magar, S. Shen, G. Luikuo, M. Fleming, and R. Aguilar, "An application specific dsp chip set for 100 mhz data rates," Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'98), pp.1989–1992, April 1998.

[13] L.Y.-W. and L.C.-Y, "Design of an fft/ifft processor for mimo ofdm systems," IEEE Trans. Circuits Syst. I, Fundam. Theory Appl., vol.54, no.4, pp.807–815, April 2007.

[14] Y.H. Lee, T.H. Yu, K.K. Huang, and A.Y. Wu, "Rapid ip design of variable-length cached-fft processor for ofdm-based communication systems," Proc. IEEE International Workshop on Signal Processing Systems Design and Implementation (SPSDI'97), pp.62–65, Oct. 2006.

[15] V. Tarokh, H. Jafarkhani, and A.R. Calderbank, "Space-time block coding for wireless communications: performance results," IEEE J. Sel. Areas Commun., vol.17, no.3, pp.451–460, March 1999.

[16] E. Cavus and B. Daneshrad, "A very low-complexity space-time block decoder (STBD) ASIC for wireless systems," IEEE Trans. Circuits Syst. I, vol.53, no.1, pp.60–69, Jan. 2006.

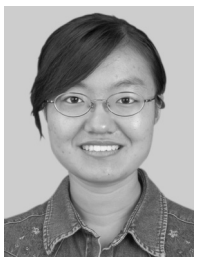
[17] M. Vehkapera and M. Juntti, "Analysis of space-time coded and spatially multiplexed mimo systems with zf receivers," Proc. IEEE International Conference on Communications (ICC'07), pp.738–743, June 2007.

[18] Y. Zhang and D. Li, "Mmse linear detector for space-time transmit diversity over fast fading channels," Proc. IEEE Personal, Indoor and Mobile Radio Communications (PIMRC '03), pp.2388–2392, Sept. 2003.

[19] J.H. Lee and S.C. Kim, "Efficient isi cancellation for stbc ofdm systems using successive interference cancellation," IEEE Commun. Lett., vol.10, no.8, pp.629–631, Aug. 2006.

[20] F. Thoma, M. Kuhnle, P. Bonnot, E. Panainte, K. Bertels, S. Goller, A. Schneider, S. Guyetant, E. Schuler, K. Muller-Glaser, and J. Becker, "Morpheus: Heterogeneous reconfigurable computing," Proc. IEEE International Conference on Field Programmable Logic

and Applications (FPL'07), Amsterdam, Netherlands, Aug. 2007.
[21] C. Liang and X. Huang, "Mapping parallel fft algorithm onto smart-cell coarse-grained reconfigurable architecture," Proc. IEEE International Conference on Application-specific Systems, Architectures and Processors (ASAP'09), Boston, USA, July 2009.
[22] W. Lee and G.E. Sobelman, "Mesh-star hybrid noc architecture with cdma switch," Proc. IEEE International Symposium on Circuits and Systems (ISCAS'09), Taiwan, China, May 2009.
[23] D. Cohen, "Simplified control of fft hardware," IEEE Trans. Acoust. Speech Signal Process., vol.24, no.6, pp.577–579, Dec. 1976.
[24] Y. Ma and L. Wanhammar, "A hardware efficient control of memory addressing for high-performance fft processors," IEEE Trans. Signal Process., vol.48, no.3, pp.917–921, March 2000.

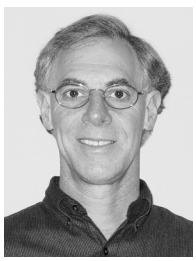


Shuang Zhao received the B.S. degree in electrical engineering from China's University of Mining and Technology, in 2005. Since 2005, she has been studying on microelectronics for her Ph.D. degree in the State Key Lab of ASIC and System, Fudan University. She worked mainly on video and communication digital processing. Now her research focuses on reconfigurable architecture for communication system.

Wenqing Lu received the B.S. degree in electrical engineering from Shanghai Jiaotong University, Shanghai, China in 2006. Since 2006, she has been studying microelectronics in the State Key Lab of ASIC and System, Fudan University, Shanghai, China. Her main research interests are communication baseband signal processing and IC design.

Xiaofang Zhou received the B.S, M.S, and Ph.D. from Fudan University, Shanghai, China in 1992, 1995, and 1998, respectively. He was with Bell Laboratories, Lucent Technologies (China) as member of technical staff, from 1998 to 2001. He was with Shanghai Fudan high-technology corporation as senior engineer, from 2001 to 2002. He joined Fudan University, Shanghai, in July 2002 as faculty member. His current research interests include digital SoC, reconfigurable architecture and communication.

Dian Zhou received the B.S. degree in physics and the M.S. degree in electrical engineering from Fudan University, Shanghai, China, in 1982 and 1985, respectively. He received the Ph.D. degree in electrical and computer engineering from University of Illinois at Urbana-Champaign, in 1990. He has been a full professor in Electrical and Computer Engineering Department, University of Texas at Dallas, since 1999. His research interests include high-speed VLSI systems, and CAD tools and algorithms. He currently is a Changjiang Honored Professor at Fudan University, where he also serves the dean of school of microelectronics, Fudan University. Dr. Zhou received IEEE Circuits and Systems Society Darlington Award in 1993, and NSF Young Investigator Award in 1994. He was an Associate Editor for IEEE Transactions on Circuits and Systems.



Gerald E. Sobelman received a B.S. degree in physics from the University of California, Los Angeles. He was awarded M.S. and Ph.D. degrees in physics from Harvard University. He was a postdoctoral researcher at The Rockefeller University, and he has held senior engineering positions at Sperry Corporation and Control Data Corporation. He is currently a faculty member in the Department of Electrical and Computer Engineering at the University of Minnesota. He also serves as the Director of Graduate Studies for the Graduate Program in Computer Engineering at the University of Minnesota. Prof. Sobelman is a Senior Member of IEEE and serves on the technical program committees for IEEE ISCAS, IEEE SOCC and IEEE ICCSC. He is currently Chair of the Technical Committee on Circuits and Systems for Communications (CASCOM) of the IEEE Circuits and Systems Society. He has also served as an Associate Editor of IEEE Signal Processing Letters. He was Local Arrangements Chair for the 1993 IEEE International Conference on Acoustics, Speech and Signal Processing. In addition, he has chaired many sessions at international conferences in the areas of communications and VLSI design, and he is a Distinguished Lecturer of the IEEE Circuits and Systems Society for 2008–2009. He has developed and presented short courses on digital VLSI design at several industrial sites. He has also given invited lectures at many universities, and he has been a consultant to a number of companies. He current research interests are in the areas of digital VLSI circuit and system design for applications in communications and signal processing. He has authored or co-authored more than 100 technical papers and 1 book, and he holds 11 U.S. patents.