EURASIP Journal on Applied Signal Processing 2005:16, 2655–2663

Bit Manipulation Accelerator for Communication Systems Digital Signal Processor

Sug H. Jeong
School of Electrical and Computer Engineering, Ajou University, Suwon 443-749, Korea
Email: jshajou@nate.com

Myung H. Sunwoo
School of Electrical and Computer Engineering, Ajou University, Suwon 443-749, Korea
Email: sunwoo@ajou.ac.kr

Seong K. Oh
School of Electrical and Computer Engineering, Ajou University, Suwon 443-749, Korea
Email: oskn@ajou.ac.kr

Received 30 January 2004; Revised 14 November 2004

This paper proposes application-specific instructions and their bit manipulation unit (BMU), which efficiently support scrambling, convolutional encoding, puncturing, interleaving, and bit stream multiplexing. The proposed DSP employs the BMU supporting parallel shift and XOR (exclusive-OR) operations and bit insertion/extraction operations on multiple data. The proposed architecture has been modeled by VHDL and synthesized using the SEC 0.18 μm standard cell library and the gate count of the BMU is only about 1700 gates. Performance comparisons show that the number of clock cycles can be reduced about 40%~80% for scrambling, convolutional encoding, and interleaving compared with existing DSPs.

Keywords and phrases: bit manipulation, application-specific DSP, VLSI architecture.

1. INTRODUCTION

With the rapid progress of communication technologies, various communication systems have been developed, such as xDSL (digital subscriber line), WLAN (wireless local area network), PLC (power-line communications), DMB (digital multimedia broadcasting), and IMT-2000 (International Mobile Telecommunications-2000). These communication systems require large computational power and low power consumption. Therefore, ASIC (application-specific integrated circuit) solutions have been widely used to implement these communication systems.

However, conventional ASIC chips face several limitations such as lack of flexibility for various communication standards, high development costs, and slow time-to-market. Thus, there have been strong demands to implement communication systems using programmable processors. Recently, the concept of the software defined radio (SDR) has been promoted [1]. SDR is a flexible communication system that supports multimode and multiband using programmable processors. Thus, implementation methods are changing from ASIC solutions to DSP (digital signal processor) -based solutions that can have advantages in several aspects. Programmable DSPs are greatly improving time-to-market and allowing faster changes and upgrades than hardwired ASIC chips. Hence, the market of ASDSP (application-specific digital signal processor) compromising advantages of both ASIC and DSP is growing [2].

General communication systems consist of functional blocks, such as source coding, channel coding, modulation, synchronization, demodulation, channel decoding, and so forth. For example, Figure 1 shows the baseband processing of the WLAN OFDM (orthogonal frequency-division multiplexing) modem system [3]. The binary input data is scrambled and encoded by a standard rate 1/2 convolutional encoder. The rate may be increased to 2/3 or 3/4 by puncturing. The output of the convolutional encoder is interleaved and modulated. The interleaver size and modulation method are determined by the data rate. To facilitate coherent reception, pilot values are added, and then OFDM symbols are generated. After this step, the frequency-domain signal is transformed to the time-domain signal by applying the IFFT. The OFDM receiver basically performs the reverse operations of the transmitter.

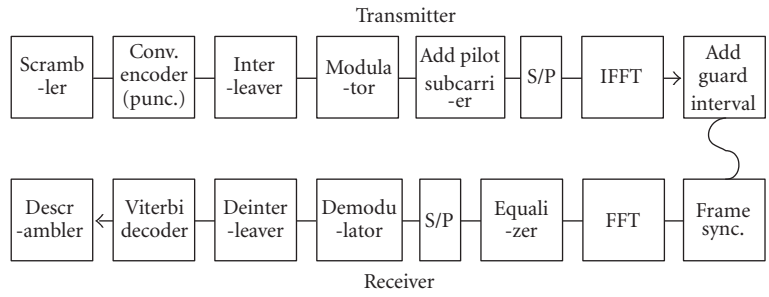

Figure 1: Transmitter and receiver of OFDM WLAN.

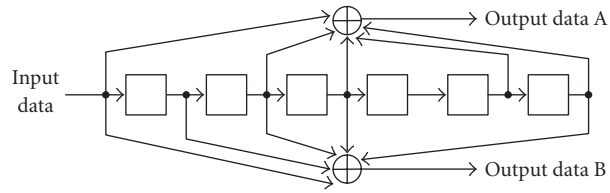

Figure 2: Convolutional encoder.

Each of these functions performs the similar operations in various standards [4]. However, it has different characteristics according to the standards. Hence, a flexible DSP-based solution is more attractive than an ASIC solution. However, among these functions pure software implementation onto a DSP is inefficient in some applications, such as scrambling, convolutional encoding, puncturing, interleaving, and so forth. These functions are generally not multiply-intensive functions, but bit manipulation functions. In ASIC design, these are implemented using simple components, such as shift registers and XOR gates. However, DSPs have MAC (multiply-accumulate) oriented datapaths and word-based memories. Moreover, computational complexities of bit manipulation functions are more significant than those of MAC-based functions as data rates increase [5]. Hence, special hardware and instructions are necessary for bit manipulation functions.

Hardware-based application acceleration in a DSP is widely used in recent DSPs [6, 7, 8, 9, 10, 11, 12, 13]. Moreover, recent commercial DSPs are adopting application-specific coprocessors for channel decoding in TMS320C6416 [14], filtering in MSC8101 [15], floating point/vector arithmetic in Xtensa [16], and so forth. This paper proposes the application-specific instructions and their BMU (bit manipulation unit) architecture having little extra hardware, which effectively support scrambling, convolutional encoding, puncturing, interleaving, and bit stream multiplexing. To verify the architecture, the proposed architecture has been modeled by VHDL and simulated.

This paper is organized as follows. Section 2 analyzes key operations of bit manipulations used in communication systems and architectures of existing DSPs for bit manipulations. Section 3 describes the proposed instructions and their hardware architectures. Section 4 presents implementation results and performance comparisons with existing DSPs. Finally, Section 5 contains conclusions.

2. BIT MANIPULATIONS AND EXISTING DSP ARCHITECTURES

This section analyzes bit manipulation operations used in various communication standards, and describes architectures of existing DSPs for bit manipulation operations.

2.1. *Bit manipulations*

Bit manipulations are operations including bit setting, clearing, rearranging, and so forth. These operations are accomplished by bit AND, bit OR, bit XOR, shift operations, and so forth. In communication systems, function blocks such as scrambling, convolutional encoding, puncturing, interleaving, and bit stream multiplexing use bit manipulations.

In dedicated hardware, a scrambler and a convolutional encoder are implemented using shift registers and XOR gates. For example, Figure 2 shows the convolutional encoder [17]. During the encoding process, each input bit passes to a shift register and the output of the encoder is derived by combining (XOR operations) the bits in the shift register determined by the structure of the encoder in use. Scrambling and convolutional encoding can be characterized by the constraint length, the code rate, and the generator polynomial, and they require XOR operations of the shifted data.

Next, we consider puncturing, interleaving, and bit stream multiplexing. Puncturing is a procedure for omitting some of the encoded bits according to the puncturing patterns. Figure 3 shows a puncturing example [17]. The shaded

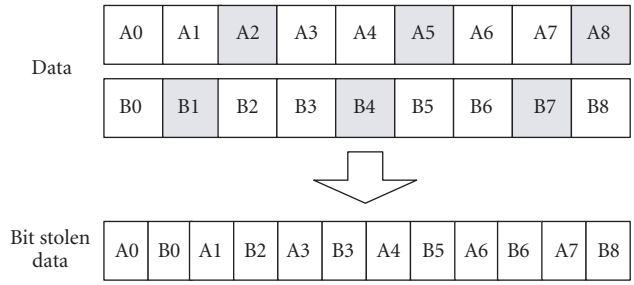

Figure 3: Puncturing.

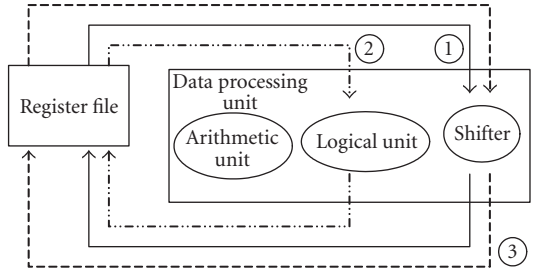

Figure 4: Computation units of a general DSP.

blocks represent the omitted bits. Interleaving is the operation of shuffling input bits and characterized by the size of the interleaver and an interleaving scheme. Bit stream multiplexing is used to merge encoded data according to the code rates in the convolutional encoder. Although the operations have regular patterns, it is not easy to implement hardware which can accommodate different characteristics of various communication standards. However, bit extraction and insertion in arbitrary bit positions are common operations for puncturing, interleaving, and bit stream multiplexing.

2.2. *Existing DSP architectures for bit manipulations*

Figure 4 shows computation units of a general DSP. The data processing unit (DPU) of general DSPs consists of an arithmetic unit, a logical unit, and a shifter. The repetitive shift/XOR operations can be performed using the logical unit and the shifter. First, the data is read from the register. Next, the shifter shifts the data ①. Finally, the logical unit performs XOR operations ②. However, conventional DSPs do not support parallel shift and XOR operations on multiple data. In addition, a bit extraction operation is performed by a shift left followed by a shift right, and then, the field in the source is extracted ③. A bit insertion operation can also be performed using shift, AND, or OR operations.

Figure 5 illustrates bit extraction/insertion operations of commercial DSPs. StarCore SC140 shown in Figure 5a supports extract and insert instructions [18]. The extract instruction extracts a bit field from a source data register and passes to a destination data register, right-aligned and zero-extended. The insert instruction inserts a bit field from a source data register into a destination data register. The bits outside of the inserted field in the destination register are unchanged. TI (Texas Instruments) TMS320C6x shown in Figure 5b supports an extraction operation according to offsets, and shuffling/deshuffling operations of two input words [19]. TMS320C55x shown in Figure 5c supports the expand and extract instructions [20]. In the expand operation, according to the bit set to "1" in the bit-field mask, the corresponding 16 LSBs (least significant bits) of the source accumulator bits are extracted and packed toward the LSBs. The result is stored in the destination register. In the extract operation, according to the bit set to "1" in the bit-field mask, the 16 LSBs of the source accumulator bits are extracted and separated with "0" toward the MSBs (most significant bits). The result is stored in the destination register.

The conventional DSPs can carry out the bit insertion and extraction operations according to the specific rules. However, they cannot arbitrarily extract separate bits in the input data nor arbitrarily insert the extracted bits in separate positions of the output data. Therefore, many clock cycles are required to perform a series of the shift, insertion or extraction, and OR operations.

3. PROPOSED BIT MANIPULATION INSTRUCTIONS AND THEIR HARDWARE

This section presents three instructions for the bit manipulation and their hardware architecture. The proposed instructions include SCB for scrambling, CONV for convolutional encoding, and PUNC for puncturing, interleaving, and multiplexing. Figure 6 shows the proposed bit manipulation unit including a shift-XOR array, a bit extraction/insertion logic,

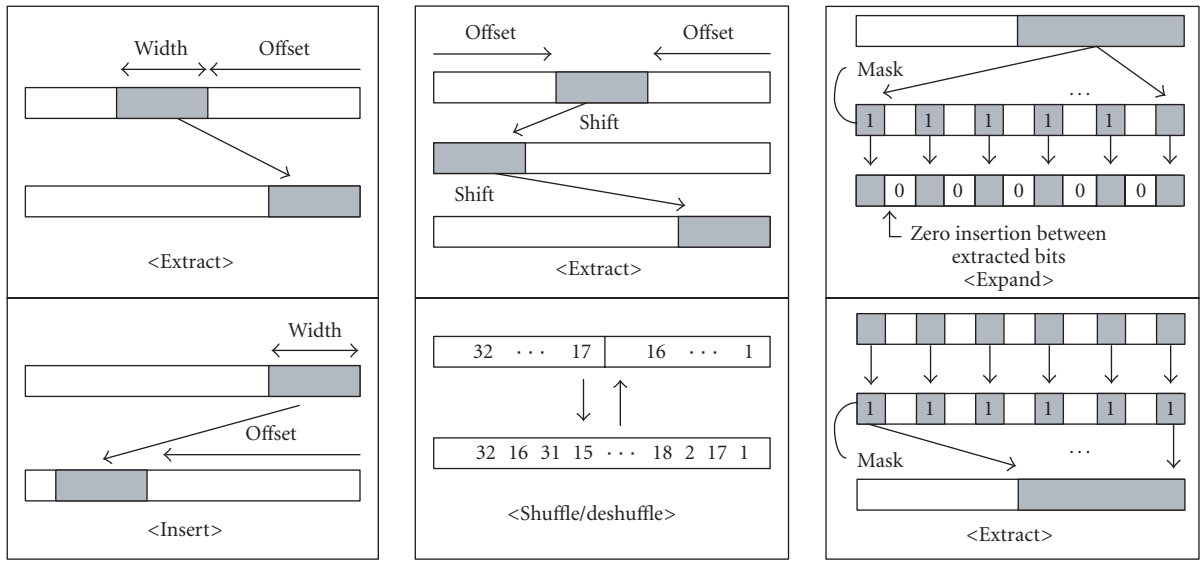

Figure 5: Bit extraction/insertion operations of commercial DSPs.

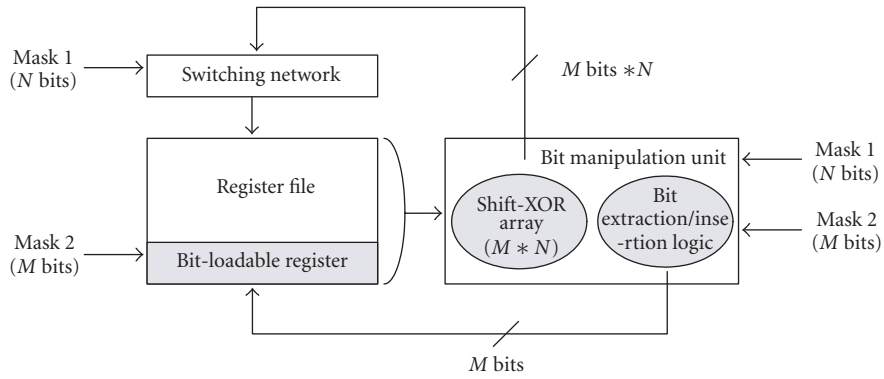

Figure 6: The proposed bit manipulation unit.

and a bit-loadable register. Mask1 and Mask2 signals are used to control the bit manipulation unit. The number of bits of Mask1, N, and that of Mask2, M, can be arbitrarily chosen.

3.1. SCB and CONV instructions for shift-XOR array

Figure 7 shows the operations of the proposed shift-XOR array in detail. First, it receives the input data of $M+N$ bits and Mask1, and generates the N shifted data that are shifted by 1 through N bits. Next, it performs parallel XOR operations of the input data and the N shifted data selected by Mask1. If the Xth bit of Mask1 is set to "1," the X-bit shifted data is XORed with the input data. Hence, the N output data are generated and transferred to the switching unit.

The switching network stores all or some of the N output data on the registers according to Mask1. Mask1 is the selection signal that enables the registers to store the only valid outputs among the N output data. The logical equations below represent operations of the shift-XOR array.

Output 1 = (Mask1(1) AND 1-bit shifted input) XOR Input,
Output 2 = (Mask1(2) AND 2-bit shifted input) XOR Output 1,
⋮
Output N = (Mask1(N) AND N-bit shifted input) XOR Output $N-1$.

The following statement shows the syntax of the SCB instruction. Mask1 and the input data (SRC) are specified in the instruction syntax. If the Xth bit of Mask1 is set, the X-bit shifted input data is XORed with the original input data, and then the Output X is stored. Depending on Mask1 values, multiple outputs can be generated. Destination registers are selected according to Mask1 values. The symbol "≫" denotes the shift-right operation, the symbol "&" represents bit concatenation, and the symbol "⇐" denotes assignment. SRC represents the source register and DST represents the destination register.

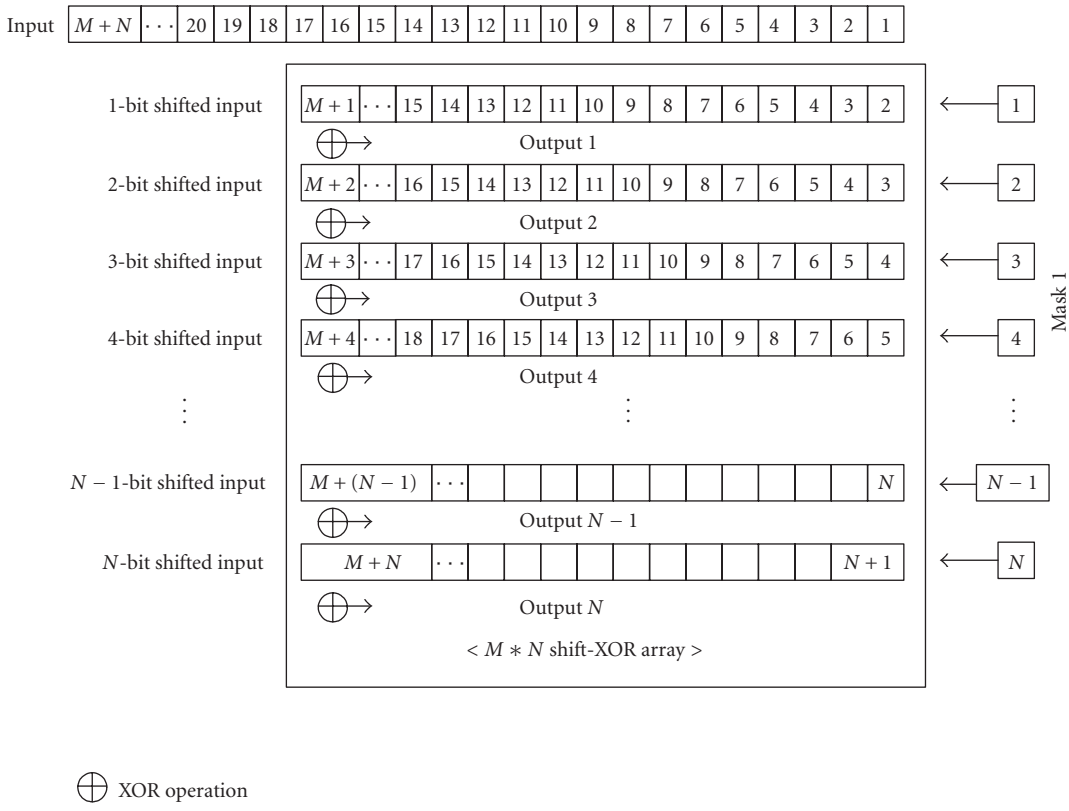

Figure 7: Proposed shift-XOR array for logical operation.

Syntax: SCB Mask1, SRC

Description: Registers (Maximally N, selected by Mask1) ⇐ SRC XOR (SRC ≫ (The shift amount determined by Mask1)).

The following statement shows the syntax of the CONV instruction. Mask1, two input data (SRC1, SRC2), and the destination register are specified. Two input data, SRC1 and SRC2, are concatenated and used as an input of the shift-XOR array. The operations are the same as the SCB instruction. However, the only Output N (last output) is stored.

Syntax: CONV Mask1, SRC1, SRC2, DST

Description: DST (Register) ⇐ (SRC1 & SRC2) XOR ((SRC1 & SRC2) ≫ (The shift amount determined by Mask1)).

Such parallel XOR operations and the storage the results in the register bank can be completed in a single clock cycle, and various operations can be performed according to Mask1. The larger the number of bits, N and M, is, the more input data bits can be processed simultaneously, which enhances the processing speed of the bit manipulations, such as scrambling and the convolutional encoding. Thus, various standards can be supported.

3.2. PUNC instruction for bit extraction/insertion logic and bit-loadable register

The bit extraction/insertion logic extracts specific bits in the N-bit input data according to Mask1 and inserts the extracted bits in the bit positions selected by Mask2. The bit extraction/insertion logic can arbitrarily extract separate bits in the input data and outputs the extracted bits to arbitrary separate bit positions in the output data. The output data manipulated by the bit extraction/insertion logic is stored in the bit-loadable register. The bit-loadable register has a function of receiving Mask2 and loading input data only for the bit positions specified by Mask2.

Figure 8 shows the operation of the bit extraction/insertion logic. Mask1 carries out the function of selectively extracting some bits of the input data. The input data bits in the bit positions where the corresponding bits in Mask1 are set to "1" are selected. Mask2 carries out the function of designating the bit positions of the output data for outputting the selected bits by Mask1. The least significant selected bit is transferred to the bit position of the output data corresponding to the least significant bit of the bits in Mask2 set to "1." The next least significant selected bit is transferred to the bit position in the output data corresponding to the next least significant bit of the bits in Mask2 set to "1," and so on.

The output data of the bit extraction/insertion logic is stored in the bit-loadable register shown in Figure 9. Hence, the bit-loadable register loads only the input data bits in the bit positions where the corresponding bits in Mask2 are set to "1" while retaining the previously stored data bits in the bit positions where the corresponding bits in Mask2 are set to "0."

The following statement shows the syntax of the PUNC instruction. Mask1, Mask2, and the input data (SRC) are specified. The bits selected by Mask1 are extracted and

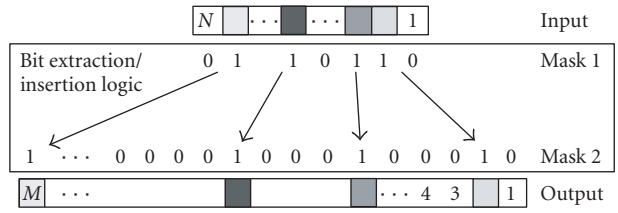

Figure 8: Logical operation of bit extraction/insertion logic.

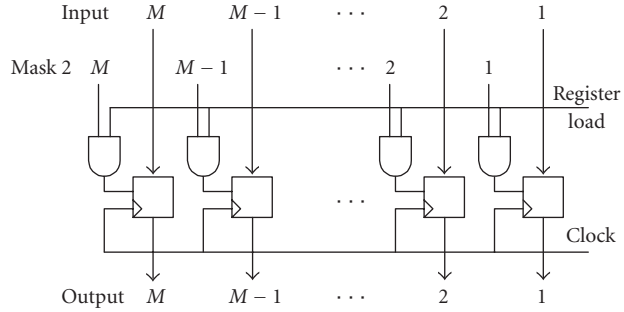

Figure 9: Bit loadable register.

inserted in the arbitrary bit positions of the bit loadable register specified by Mask2.

Syntax: PUNC Mask1, Mask2, SRC

Description: Bit-loadable register (Bit positions selected by Mask2) ⇐ SRC (Bits selected by Mask1).

The bit manipulation of selecting all or some of the input data bits according to Mask1 and storing the selected bits in the bit positions, selected by Mask2, in the bit-loadable register can be completed in a single cycle. Thus, using such bit manipulation steps, the operation speed for extracting bits from several data words and combining the extracted bits into a single data word can be remarkably improved. In particular, the bit-loadable register removes the use of multiple OR operations for combining the extracted bits, which increases the operation speed further. These efficient operations can be employed in puncturing, interleaving, and bit stream multiplexing.

3.3. Operation examples of the BMU

Operation examples of the bit manipulation unit are described below. First, scrambling can be performed using the shift-XOR array and the bit extraction/insertion logic. In scrambling having the constraint length K and described by the generator polynomial of (1), a scrambling code can be obtained by XOR operations of the $(K-1-A)$-bit shifted input data, the A-bit shifted input data, the $2A$-bit shifted input data, the $3A$-bit shifted input data, ..., the nA-bit shifted input data, where n is a natural number, and nA is smaller than or equal to $(K-1)$:

$$g(x) = x^{K-1} + x^A + x + 1. \tag{1}$$

The data shifts and XOR operations are carried out by the shift-XOR array. The bit positions corresponding to the shift amounts in Mask1 are set to "1" while the other bit positions are set to "0." After the XOR operations, the number of valid output data bits from each XOR operation is ($K-1$-shift amount) bits. The maximal number of valid output data is $K-1$ bits, and the valid output data from each XOR operation should be combined into a single word. Such a combining task can be carried out by the bit extraction/insertion logic.

Convolutional encoding can also be performed by the shift-XOR array. Assuming that the generator polynomial is described by (2), the convolutional encoding is performed by XOR operations of the $(K-1)$-bit shifted data, the A-bit shifted data, the B-bit shifted data, the C-bit shifted data, and the original input data. The $(K-1)$th, the Ath, the Bth, and the Cth bit positions from the least significant bit in Mask1 are set to "1" while the other bit positions are set to "0." When the data is encoded according to a generator polynomial, the bit extraction/insertion logic should combine the encoded data according to the code rate:

$$g(x) = x^{K-1} + x^A + x^B + x^C + 1. \tag{2}$$

Puncturing and interleaving, which are omitting and inserting some data bits in the input data word, can be performed by the bit extraction/insertion logic. The bit extraction/insertion logic receives the input data, Mask1, Mask2, and then extracts some of the input data bits according to Mask1. The extracted bits are stored in the bit-loadable register according to Mask2. Hence, only the valid output data bits are stored.

4. IMPLEMENTATION RESULTS AND PERFORMANCE COMPARISONS

Figure 10 shows the proposed DSP architecture that has one program memory, two data memories, PCU (program

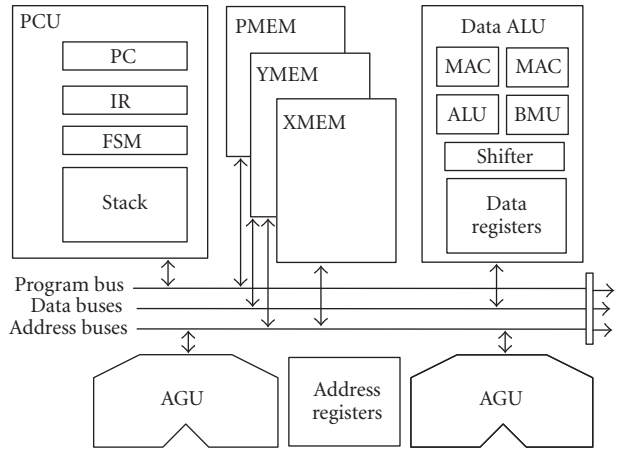

Figure 10: Proposed DSP architecture.

Table 1: The result of performance comparisons.

	StarCore SC140 [22]	TI 62X [23]	Proposed DSP
Computation units	4 shifters, 4 ALUs	4 shifters, 4 ALUs	BMU
Convolutional encoding (cycles) (IS-95, $K = 9$, $R = 1/2$, 192 bits)	463	N.A.	152
Block interleaving (cycles) (16∗6 bits)	414	N.A.	91
Scrambling (MIPS) (802.11a, 12 Mbps)	N.A.	39×10^6	20×10^6
Convolutional encoding (MIPS) (802.11a, 12 Mbps)	N.A.	77×10^6	12×10^6

control unit), DALU (data arithmetic & logical unit), and two AGUs (address generation units) [21]. The DALU consists of the proposed BMU, two MACs, one shifter, one ALU, and a register file. Each of the internal word lengths is 16 bit. The instruction pipeline consists of six stages. The proposed architecture has been modeled by VHDL and logic synthesis has been performed using the SEC 0.18 μm technology. The total gate count of the ASDSP is about 80 000 gates. The maximum delay is about 3.54 nanoseconds and thus, the maximum operating frequency is about 280 MHz. The proposed DSP has been verified using the iPROVE FPGA board having Xilinx Virtex-II. Moreover, we used assembly language and a cycle accurate simulator.

The gate count of the proposed BMU is only 1 700 ($N = 8$, $M = 16$). The critical path of the accelerator is 2.01 nanoseconds with the 0.18 μm technology.

Table 1 shows the performance comparisons between the proposed DSP and the conventional DSPs. Even though the conventional DSPs adopt the VLIW (very long instruction word) architecture having four shifters and four logical units, the proposed DSP is found to be more efficient than the conventional DSPs for scrambling, convolutional encoding, and interleaving.

Comparing with StarCore SC140, the proposed architecture can reduce the clock cycles about 67% for convolutional encoding, and about 78% for block interleaving. Comparing with TI 62x, the proposed architecture can reduce the clock cycles about 48% for scrambling, and by about 84% for convolutional encoding.

5. CONCLUSIONS

This paper proposed the application-specific instructions and their bit manipulation unit, which efficiently support scrambling, convolutional encoding, puncturing, and interleaving. The proposed BMU supports the parallel shift/XOR operations and the bit extraction/insertion. The BMU architecture has been modeled by VHDL and synthesized using the SEC 0.18 μm standard cell library. Performance comparisons show that the number of clock cycles can be reduced about 40% ~ 80% compared with the existing DSPs, and the gate count of the bit manipulation unit is only

about 1700 gates. The proposed architecture can easily implement various communication standards, and thus, it can be utilized for the next-generation communication platforms.

ACKNOWLEDGMENTS

This work was supported in part by the National Research Laboratory Program of MOST, in part by HY-SDR Research Center under the ITRC Program of MIC, in part by the SystemIC2010 Program, and in part by IDEC.

REFERENCES

[1] SDR Forum [Online], available: http://www.sdrforum.org.
[2] T. Kumura, D. Ishii, M. Ikekawa, I. Kuroda, and M. Yoshida, "A low-power programmable DSP core architecture for 3G mobile terminals," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing (ICASSP '01)*, vol. 2, pp. 1017–1020, Salt Lake City, Utah, USA, May 2001.
[3] M. F. Tariq, Y. Baltaci, T. Horseman, and M. N. A. Butler, "Development of an OFDM based high speed wireless LAN platform using the TI C6x DSP," in *Proc. IEEE International Conference on Communications (ICC '02)*, vol. 1, pp. 522–526, New York, NY, USA, April 2002.
[4] S. H. Jeong, M. H. Sunwoo, and S. K. Oh, "Reconfigurable hardware structures for spreading and scrambling operations," *Journal of Semiconductor Technology and Science*, vol. 3, no. 4, pp. 199–204, 2003.
[5] K. Masselos, S. Blionas, and T. Rautio, "Reconfigurability requirements of wireless communication systems," in *Proc. IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip*, Hamburg, Germany, April 2002.
[6] J. H. Lee, S. H. Jeong, and M. H. Sunwoo, "Application-specific DSP architecture for OFDM modem systems," in *Proc. IEEE Workshop on Signal Processing Systems (SIPS '03)*, Seoul, Korea, August 2003.
[7] U. Walther and G. P. Fettweis, "PN-generators embedded in high performance signal processors," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '01)*, pp. 45–48, Sydney, Australia, May 2001.
[8] J. H. Lee, J. S. Lee, M. H. Sunwoo, and K. H. Kim, "Design of new DSP instructions and their hardware architecture for the Viterbi decoding algorithm," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '02)*, pp. 561–564, Scottsdale, Ariz, USA, May 2002.
[9] C.-K. Chen, P.-C. Tseng, Y.-C. Chang, and L.-G. Chen, "A digital signal processor with programmable correlator array architecture for third generation wireless communication system," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 12, pp. 1110–1120, 2001.
[10] J. S. Lee and M. H. Sunwoo, "Design of new DSP instructions and their hardware architecture for high-speed FFT," *Journal of VLSI Signal Processing*, vol. 33, no. 3, pp. 247–254, 2003.
[11] K. L. Heo, S. M. Cho, J. H. Lee, and M. H. Sunwoo, "Application-specific DSP architecture for fast Fourier transform," in *Proc. IEEE 14th International Conference on Application-Specific Systems, Architectures and Processors (ASAP '03)*, pp. 369–377, Galveston, Tex, USA, June 2003.
[12] K. L. Heo, M. H. Sunwoo, and S. K. Oh, "Implementation of a wireless multimedia DSP chip for mobile applications," in *Proc. IEEE Workshop on Signal Processing Systems (SIPS '03)*, pp. 51–56, Seoul, Korea, August 2003.
[13] J. H. Lee, J. S. Lee, and M. H. Sunwoo, "Design of application-specific instructions and hardware accelerator for Reed-Solomon codecs," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 13, pp. 1346–1354, 2003.
[14] Texas Instruments, Inc. [Online], available: http://www.ti.com.
[15] Motolora, Inc. [Online], available: http://www.motorala.com.
[16] Tensilica, Inc. [Online], available: http://www.tensilica.com.
[17] IEEE, *802.11a Wireless LAN Medium Access Control and Physical Layer Specifications*, September 1999.
[18] Motorola Semiconductors Inc., *SC140 DSP Core Reference Manual*, Denver, Colo, USA, 2001.
[19] Texas Instruments Inc., *TMS320C62xx User's Manual*, Dallas, Tex, USA, 2000.
[20] Texas Instruments Inc., *TMS320C55x User's Manual*, Dallas, Tex, USA, 2001.
[21] J. H. Lee, J. H. Moon, K. L. Heo, M. H. Sunwoo, S. K. Oh, and I. H. Kim, "Implementation of application-specific DSP for OFDM systems," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '04)*, vol. 3, pp. 665–668, Vancouver, British Columbia, Canada, May 2004.
[22] Motorola Inc., *SC140 Functional Libraries*, [Online], available: http://www.motorola.com.
[23] E. Sereni, S. Culicchi, V. Vinti, E. Luchetti, S. Ottaviani, and M. Salvi, A Software RADIO OFDM Transceiver for WLAN Applications, Electronic and Information Engineering Department, University of Perugia, Italy, 2001.

Sug H. Jeong received the B.S. degree and the M.S. degree in electronic engineering from Ajou University, Suwon, Korea, in 2002 and 2004. He is currently with Daewoo Electronics Corp., Korea, Seoul. His main research interests include SOC design, DSP core design, and software-defined radio.

Myung H. Sunwoo received the B.S. degree in electronic engineering from the Sogang University in 1980, the M.S. degree in electrical and electronics engineering from the Korea Advanced Institute of Science and Technology in 1982, and the Ph.D. degree in electrical and computer engineering from the University of Texas at Austin in 1990. He worked for Electronics and Telecommunications Research Institute (ETRI) in Daejeon, Korea, from 1982 to 1985, and Digital Signal Processor Operations, Motorola, Austin, Tex, from 1990 to 1992. Since 1992, he has been a Professor with the School of Electrical and Computer Engineering, Ajou University in Suwon, Korea. He is the Director of the National Research Laboratory sponsored by the Ministry of Science and Technology. His research interests include VLSI architectures, SOC design for multimedia and communications, and application-specific DSP architectures. Dr. Sunwoo has published more than 120 papers in international transactions/journals and conferences and also has 31 patents. He served as an Associate Editor for the IEEE Transactions on Very Large Scale Integration (VLSI) Systems from 2002 to 2003 and as a Guest Editor for the Journal of VLSI Signal Processing (Kluwer, 2004). Currently, he is a Senior Member of the IEEE and a Chair of the IEEE CAS Society of the Seoul Chapter.

Seong K. Oh received the B.S. degree in electronics engineering from Kyungpook National University, Taegu, Korea, in 1983, and the M.S. and Ph.D. degrees in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Taejon, Korea, in 1985 and 1990, respectively. From 1988 to 1993, he was with Transmission Systems Lab, Samsung Electronics Inc., Seoul, Korea, as a Senior Researcher. Since 1993, he has been with Ajou University, Suwon, Korea, where he is currently a Full Professor at the School of Electronics Engineering, leading the Communication Systems Research Group. During 1996–1997, he was a Visiting Professor at Simon Fraser University, Burnaby, British Columbia, Canada. His research interests include smart antennas, space-time coding, MIMO systems, OFDM, and digital transmission technologies.