EURASIP Journal on Applied Signal Processing 2003:13, 1346–1354

Design of Application-Specific Instructions and Hardware Accelerator for Reed-Solomon Codecs

Jung H. Lee
School of Electrical and Computer Engineering, Ajou University, San 5, Wonchun-Dong, Paldal-Gu, Suwon 442-749, Korea
Email: junghoo@ajou.ac.kr

Jaesung Lee
Computer System Department, Electronics and Telecommunications Research Institute, 161 Gajeong-Dong, Yuseong-Gu, Taejon 305-350, Korea
Email: ljshhide@etri.re.kr

Myung H. Sunwoo
School of Electrical and Computer Engineering, Ajou University, San 5, Wonchun-Dong, Paldal-Gu, Suwon 442-749, Korea
Email: sunwoo@ajou.ac.kr

Received 31 January 2003 and in revised form 6 September 2003

This paper presents new application-specific digital signal processor (ASDSP) instructions and their hardware accelerator to efficiently implement Reed-Solomon (RS) encoding and decoding, which is one of the most widely used forward error control (FEC) algorithms. The proposed ASDSP architecture can implement various programmable primitive polynomials, and thus, hardwired RS codecs can be replaced. The new instructions and their hardware accelerator perform Galois field (GF) operations using the proposed GF multiplier and adder. Therefore, the proposed digital signal processor (DSP) architecture can significantly reduce the number of clock cycles compared with existing DSP chips. The proposed GF multiplier was implemented using the Faraday 0.25 μm standard cell library and it can perform RS decoding at a rate up to 228.1 Mbps at 130 MHz.

Keywords and phrases: Reed-Solomon, application-specific DSP, GF multiplier, broadband communication, VLSI architecture.

1. INTRODUCTION

With the rapid progress of communication technologies, various broadband access systems have been developed, such as very-high-data-rate digital subscriber line (VDSL) cable modem and wireless LAN, gigabit Ethernet, 4G wireless communication, and so forth. Currently, the software defined radio (SDR) can support various communication standards since a common hardware platform can be adapted for various communication standards by means of software [1]. However, ASIC chips face several limitations such as lack of flexibility for various communication standards, high development costs, and slow time-to-market. Due to these restrictions, implementation methods have been changed to digital signal processor (DSP)-based communication systems that can have advantages in several aspects [2]. Programmable DSPs are greatly improving time-to-market and allowing faster changes and upgrades than hardwired ASIC chips. In addition, DSPs can be used for various applications as well as the Reed-Solomon (RS) decoder.

RS codes, providing the capability to efficiently correct burst errors as well as random error, have been extensively used in various communications and digital data storage systems, such as power line communications (PLC) [3], digital video broadcasting terrestrial (DVB-T) system [4], vestigial sideband (VSB) system [5], cable modem [6], satellite and mobile communications [7], magnetic recording [8], and so forth.

This paper presents new application-specific DSP (ASDSP) instructions and their hardware accelerator to efficiently implement RS codecs. Various algorithm blocks for RS codecs require Galois field (GF) multiply and add operations. Therefore, a typical RS decoder has been designed as a hardwired ASIC chip since an RS decoder needs special GF arithmetic units [9, 10, 11, 12, 13, 14, 15, 16]. Moreover, the RS decoder should be redesigned to accommodate the various primitive polynomials in recent communication systems.

Existing DSP chips [17, 18] require many clock cycles for GF multiply and add operations since they use general ALUs. The method that uses a lookup table (LUT) instead of GF operation units consumes a significant amount of power due to its large memory and large number of access delays.

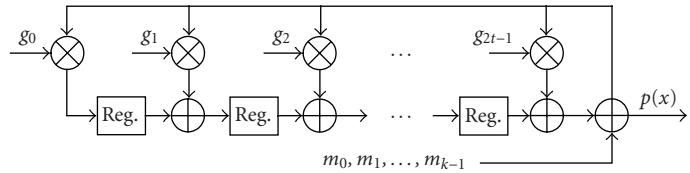

Figure 1: Typical RS encoder.

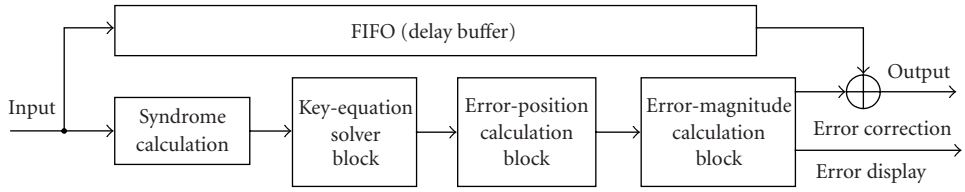

Figure 2: Typical RS decoder.

Hence, existing DSP chips have not yet satisfied the requirements of high-speed communication standards. However, if DSP chips can be made to support the special architecture for the RS algorithm, they will be able to implement RS codecs for various communication standards [19]. Thus, having application-specific instructions and their hardware accelerator for the RS algorithm, ASDSP can support various broadband communication standards.

This paper is organized as follows. Section 2 analyzes the implementation and hardware architectures of existing DSP chips [17, 18] and custom-designed RS processors [9, 10, 11, 12, 13, 14, 15, 16]. Section 3 describes the proposed RS decoding instructions and their hardware accelerator. Section 4 presents the performance comparisons with existing DSP chips. Finally, Section 5 contains conclusions.

2. IMPLEMENTATION OF THE EXISTING DSP-BASED RS DECODERS AND HARDWIRED RS PROCESSORS

This section describes the typical RS processor to briefly review the decoding process and analyzes the existing DSP-based implementation of RS.

2.1. Typical RS processor

Depending on the application, a typical RS processor is made up of several hardware blocks for parallel processing. Such an architecture can achieve higher transmission rates than required by current communication standards; however, due to its lack of flexibility regarding the primitive polynomials in various standards, the RS processor has to be redesigned to meet these standards.

2.1.1. RS encoder architecture

The architecture of the RS processor inserts 16 ($2t$) surplus symbols when $t = 8$. The generator polynomial for this architecture is represented by (1) [19, 20, 21]:

$$\begin{aligned} g(x) &= (x+\alpha^1)(x+\alpha^2)\Lambda(x+\alpha^{2t-1})(x+\alpha^{2t}) \\ &= (x+\alpha^1)(x+\alpha^2)\Lambda(x+\alpha^{15})(x+\alpha^{16}). \end{aligned} \quad (1)$$

Figure 1 shows the typical RS encoder that has the linear feedback shift register (LFSR) structure, based on the generator polynomial. If the architecture is enabled, each register is initialized as "0." After the message polynomial $m(x)$ is inserted, the operation is executed by combining $m(x)$ and $g(x)$ through the LFSR structure. If the insertion of the message polynomial $m(x)$ is ended, the remaining values in the registers are output as parity symbols.

2.1.2. RS decoder architecture

The RS decoding process is as follows. First, the syndrome value, which is the error pattern, is calculated, and then the error-locator polynomial is calculated to find the error locations. Second, the error values are determined and corrected. Figure 2 illustrates the typical RS decoder [20, 21, 22, 23, 24].

Figure 3 shows the syndrome calculation block. The syndrome is calculated using the roots of the generator polynomial (gx), which is used in the encoder. The syndrome polynomial presents the error pattern of the received code word. By using this error pattern, the key for error correction is decoded.

The number of the cells in the syndrome block is twice the number of correctable errors. When the error correction capability (t) of the RS decoder is 8, the number of $2t = 16$ for the syndrome block is needed, as shown in Figure 3.

The error-locator and error-value polynomials are calculated using this syndrome polynomial. The calculation of the error-locator and error-value polynomials is the most complicated and time consuming process in the RS decoding. The Berlekamp-Massey [9, 10], Euclid's [11, 12], or the modified Euclid's [13, 14, 15] algorithms are used in this process. In general, the architecture of the Berlekamp-Massey algorithm is smaller than that of the Euclid's algorithm. However, the serial structure of the Berlekamp-Massey algorithm has long latency and its parallel structure requires a large gate count. Figure 4 shows the architecture of the modified Euclid's algorithm [13, 14, 15]. This architecture is more suitable for high-speed transmission systems than that of the Berlekamp-Massey algorithm. The modified Euclid's

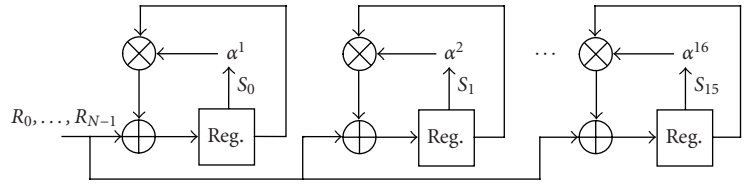

Figure 3: Syndrome calculation block.

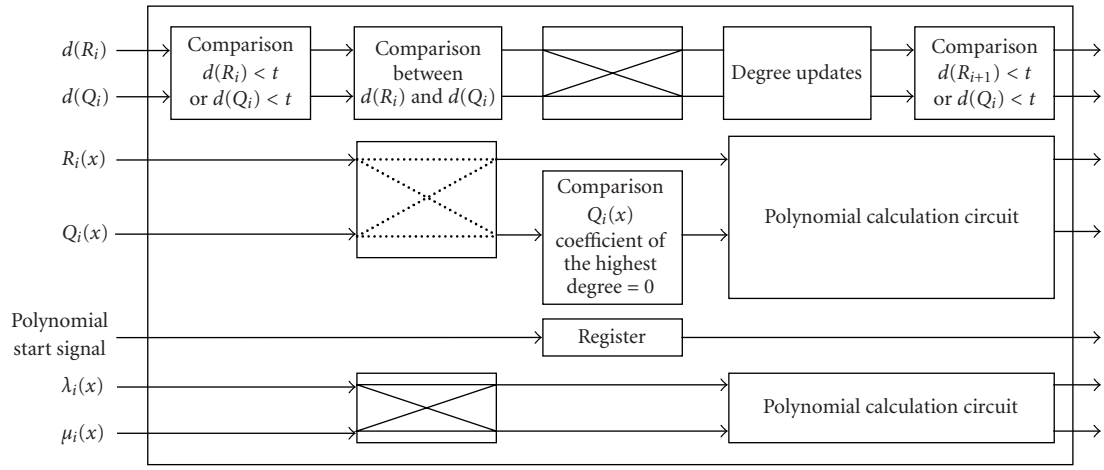

Figure 4: Architecture of the modified Euclid's algorithm.

algorithm can efficiently reduce the area since it does not require an LUT for the quotient calculation.

After the error-locator and error-value polynomials are obtained using the Euclid's algorithm, the error locations are calculated using the Chien search [22, 23] and Forney algorithms [13]. Then, the error values are calculated. This algorithm for calculating the roots of the error-locator polynomial is described in Figure 5. The roots of error locations are calculated using the coefficients (λ_i) of the error-locator polynomial. The error values are computed using the coefficients (λ_i) of the error-locator polynomial and error-value polynomial coefficients (R_i) as shown in Figure 6.

Typical RS ASIC chips require the hardwired GF operation units as modulo multipliers and adders, and thus, the architecture of the GF operation units has to be redesigned based on various primitive polynomials and standards.

2.2. Existing DSP-based RS decoder

It is possible to implement the RS decoder with the existing DSP chip; however, to implement the GF operation with the existing DSP chips, a number of operations are needed to execute ALU operations repeatedly. These operations have to be programmed as a subroutine and this subroutine is called from the GF operation part of the main RS program [20].

Generally, a GF multiplication consists of two steps. In the first step, two equations are multiplied as in (2). If the least significant bit (LSB) of the multiplier is one, the multiplicand is copied down; otherwise, zeros are copied down. The partial products copied down in successive lines are shifted one position to the left from the previous partial product. The 15-bit product which is the third equation of (2) is acquired using XOR operations of all partial products. In the second step, the GF operation is executed according to the primitive polynomial to convert the 15-bit data into the 8-bit data. GF multiplications are shown as the "⊗" symbols in Figures 1, 3, 5, and 6. Additions and subtractions in GF operations can be implemented using XOR operations in the ALU:

$$\begin{aligned} A(x) &= A_7x^7 + A_6x^6 + A_5x^5 + A_4x^4 \\ &\quad + A_3x^3 + A_2x^2 + A_1x^1 + A_0x^0, \\ B(x) &= B_7x^7 + B_6x^6 + B_5x^5 + B_4x^4 \\ &\quad + B_3x^3 + B_2x^2 + B_1x^1 + B_0x^0, \\ \omega(x) &= A(x) \oplus B(x) \\ &= (A_7 \cdot B_7)x^{14} + (A_7 \cdot B_6 \oplus B_7 \cdot A_6)x^{13} \\ &\quad + \Lambda + (A_1 \cdot B_0 \oplus B_0 \cdot A_1)x^1 + (A_0 \cdot B_0) \\ &= \omega(14)x^{14} + \omega(13)x^{13} + \Lambda + \omega(1)x^1 + \omega(0)x^0. \end{aligned} \tag{2}$$

Figure 7 shows the GF multiplication flow of general DSP chips that do not support the RS decoding. To implement (2), AND operations are executed from the LSB of (A) and 8 bits of (B) to the MSB of (A) and 8 bits of (B) in cycle 1. Then, the results are shifted according to the digits in cycle 2. Eight 15-bit results are executed by XOR operations to acquire the 15-bit data that appeared in the third equation of (2). Finally, the GF operation is executed in cycle 3. The GF operation can be implemented using AND and XOR.

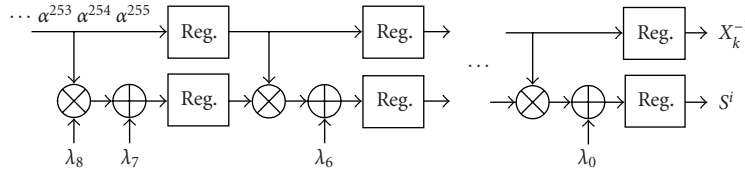

Figure 5: Chien search block.

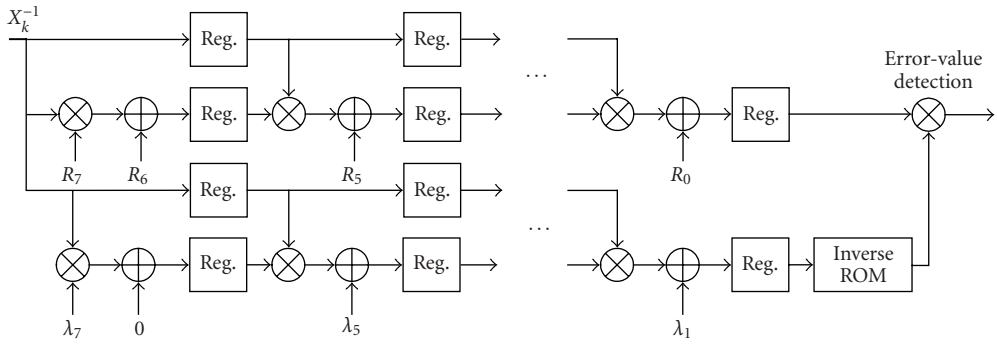

Figure 6: Forney block.

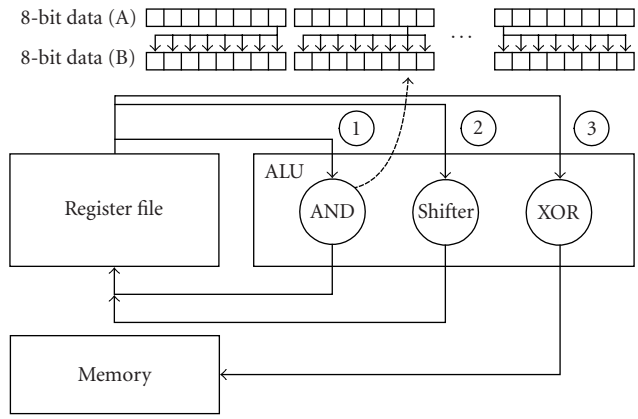

Figure 7: GF multiplication flow of existing DSPs.

To implement this procedure, general purpose DSP chips require quite a number of clock cycles. The DSP used here should be accessible by a bit as well as a byte. If the DSP is a 32-bit machine, it can compute two GF multiply operations. If the DSP is a 64-bit machine, it can compute four GF multiply operations simultaneously. If N ALUs can be operated at the same time, $1/N$ cycles are taken to compute the GF multiplication. However, if the DSP cannot be accessed by a byte, a number of additional cycles is required.

Hence, we cannot get a fast RS decoding rate since the hardware architecture and instructions are not supported for the GF multiplication on existing DSP chips. Therefore, for the RS decoding, the existing DSP chips can be used only in slow-speed data communication. Recently, TMS320C64x has 8 GF multipliers and the GMPY4 instructions can perform four GF multiplications of two integers, each of which contains 4 packed bytes. Two GMPY4 instructions can be executed in parallel; hence the 8 GF multiplications can be performed in a single cycle. However, it supports only the GF multiply operation [19] and does not support the GF multiply and add operations. Moreover, it has a large hardware size and high power consumption due to its VLIW architecture.

SC140 does not support GF operations and is also a VLIW architecture having similar disadvantages. In addition, it consumes more power and needs larger memory since it uses the LUT method [25]. In the implementation using an LUT, the results of GF operations have been stored in ROM or RAM, and they are accessed when they are needed [25]. When m is equal to 8, a $2^8 \times 2^8 \simeq 64$ Kbytes storage device is needed. Even in the highly integrated DSP, it is hard to use on-chip memory only for storing these values. Regardless of the data width of DSP, only one GF operation at a time is

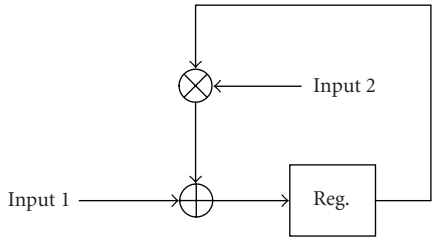

Figure 8: Repetitive multiply and add operations for the RS codec.

possible. Moreover, additional cycles are needed to access the on-chip and off-chip memories. Hence, most DSPs implement the RS decoding without using an LUT.

3. NEW INSTRUCTIONS AND THEIR ARCHITECTURE

This section presents three instructions for the RS decoder implementation and the proposed operation flows, and their new architecture. The proposed instructions include modulo-add (MADD), modulo-multiply (MMUL), and modulo-MAC (MMAC).

Various algorithm blocks for RS codecs require repetitive multiply and add operations, as shown in Figure 8. The Berlekamp-Massey [9, 10] algorithm, the Euclid [11, 12] algorithm, and the modified Euclid [13, 14, 15] algorithm also use the circuit shown in Figure 8 [9, 10, 11, 12, 13, 14, 15, 19] to implement the RS decoding. The multiplier and adder used for RS have the same circuit shown in Figure 8 regardless of various algorithms or primitive polynomials. The architecture of the hardwired RS codec is redesigned based on the primitive polynomial. In general, implementing the RS decoder on an existing DSP chip is not effective since the instructions of DSP chips do not support GF multiply and add operations. The GF multiply and add operations, shown in Figure 8, are different from general multiply and add operations. Hence, we need an ASDSP chip that has a programmable architecture to support various primitive polynomials according to various communication standards.

Figure 9 represents the proposed MADD, MMUL, and MMAC instructions. The MADD instruction performs the modulo (GF) add operation and can be implemented with an XOR operation of an existing ALU; thus, we do not need additional hardware for the MADD instruction. The MMUL instruction can implement the GF multiply operation for error-value detection with the proposed GF multiplier shown in Figure 10. The proposed GF multiplier can perform successive GF multiply operations by adding a small amount of extra hardware, consisting of XOR gates and AND gates. The MMAC instruction can perform successive operations of the MADD and MMUL instructions. The MMAC instruction takes one cycle to execute the general modulo MAC instruction.

The proposed instructions are used extensively in RS algorithm blocks, such as the encoder, the syndrome computation block, the modified Euclid's algorithm block, the Chien search block, and the Forney algorithm block, as shown in Figures 1, 3, 5, and 6. In contrast, TMS320C64x supports the modulo MUL operation but does not support the modulo MAC operation. Hence, the proposed architecture can improve the performance of the RS codec.

Figure 10 shows the proposed GF multiplier block used for the MMUL and MMAC instructions in GF (2^m, $m = 8$). The required number of AND operations shown in the upper side of Figure 10 is the same as the value of m. In Figure 10, after two 8-bit data a and b are multiplied, the 15-bit $\omega(i)$, which is the third equation in (2), is obtained through the modulo add operation of the multiplication results. Then the 8-bit result $\Omega(i)$ can be obtained from GF multiply operations of 15-bit $\omega(i)$.

The proposed GF multiplier uses about 630 gates including the primitive polynomial decoder. The gate count of the proposed GF multiplier is larger than that of a GF multiplier of the hardwired RS ASIC chip (about 261 gates). However, the hardwired RS ASIC chip uses about 89 GF multipliers for $t = 8$ [13], 16 GF multipliers for the syndrome calculation block, 64 GF multipliers for the modified Euclid's algorithm block, 8 GF multipliers for the Chien search block, and one GF multiplier for the Forney algorithm. The proposed ASDSP uses only 8 proposed GF multipliers, and thus, requires a much lower gate count than does the hardwired RS ASIC chip. Therefore, the ASDSP has little extra hardware. When m is greater than 8, the adder can be implemented with additional XOR gates, and the GF multiplier shown in Figure 10 can also be implemented with additional AND and XOR gates.

The modulo operation unit shown in Figure 10 executes GF operations with control signals according to the value of m and the primitive polynomial. Figure 11 shows the proposed modulo operation unit that is designed with AND and XOR gates. The 15-bit $\omega(12)$ is performed by the XOR operation after it is enabled or disabled according to control signals, and then, the 8-bit $\Omega(i)$ value can be obtained from the proposed modulo operation unit. Equations (3) are the result value of the GF operation when the primitive polynomial is $x^8 + x^4 + x^3 + x^2 + x^1$ and $m = 8$:

$$\begin{aligned}
\Omega(0) &= \omega(0) \oplus \omega(8) \oplus \omega(12) \oplus \omega(13) \oplus \omega(14);\\
\Omega(1) &= \omega(1) \oplus \omega(9) \oplus \omega(13) \oplus \omega(14);\\
\Omega(2) &= \omega(2) \oplus \omega(8) \oplus \omega(10) \oplus \omega(12) \oplus \omega(13);\\
\Omega(3) &= \omega(3) \oplus \omega(8) \oplus \omega(9) \oplus \omega(11) \oplus \omega(12);\\
\Omega(4) &= \omega(4) \oplus \omega(8) \oplus \omega(9) \oplus \omega(10) \oplus \omega(14);\\
\Omega(5) &= \omega(5) \oplus \omega(9) \oplus \omega(10) \oplus \omega(11);\\
\Omega(6) &= \omega(6) \oplus \omega(10) \oplus \omega(11) \oplus \omega(12);\\
\Omega(7) &= \omega(7) \oplus \omega(11) \oplus \omega(12) \oplus \omega(13).
\end{aligned} \tag{3}$$

The primitive polynomial decoder of the proposed GF multiplier has the information whether the $\omega(i)$ is enabled or disabled. About 8 cases according to m values and the primitive polynomials are used in various communication standards. Hence, the decoder receives 3 bits ($8 = 2^3$) and outputs $15 \times 8 = 120$-bit control signals, as shown in Figure 11. The proposed GF multiplier performs the GF operation with

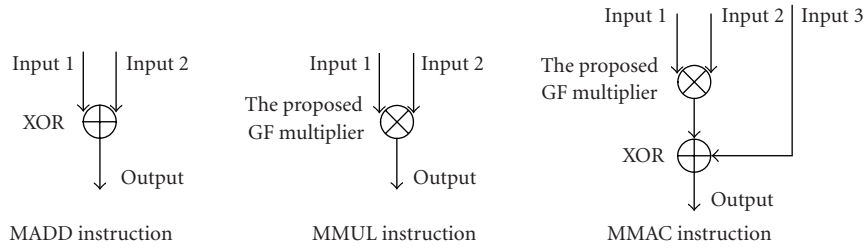

Figure 9: The proposed MADD, MMUL, and MMAC instructions.

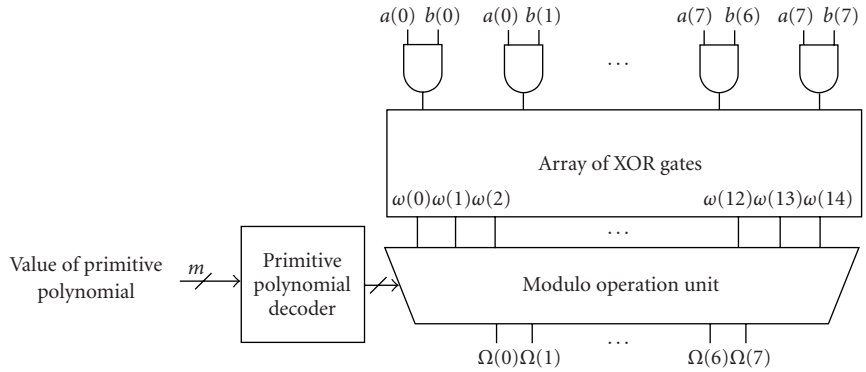

Figure 10: Proposed GF multiplier block.

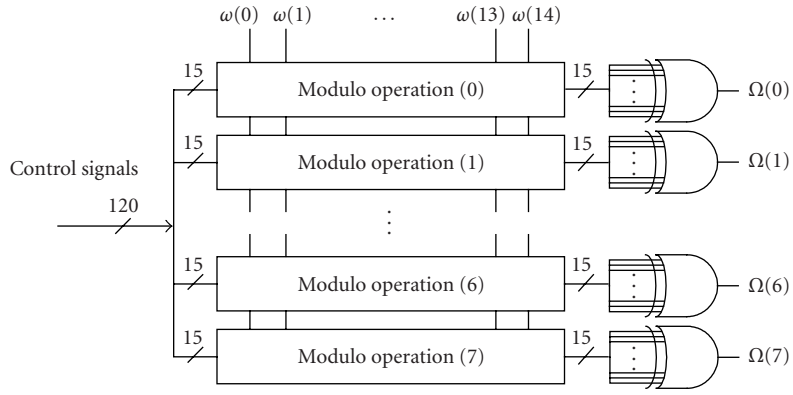

Figure 11: Proposed modulo operation unit.

these control signals. The primitive polynomial decoder is designed with combinational circuits. To implement 8 different combinations using ASIC chips, 8 different hardware implementations are required. However, the proposed ASDSP can efficiently implement these combinations.

Figure 12 shows the overall architecture of the proposed ASDSP, based on the modified Harvard architecture. Two 16-bit data memories can be accessed in a single clock cycle since the address generation unit (AGU) generates two addresses. The data processing unit (DPU) consists of two MACs, two ALUs, and one barrel shifter to efficiently support RS. The 8 GF multipliers are also included in DPU. The proposed ASDSP employs 7 pipeline stages: prefetch, fetch, decode, execute1, execute2, execute3, and write back. Every instruction, including program control instructions, is executed in a single cycle. The DO instruction, one of the most frequently used instructions, can also be executed in a cycle.

4. PERFORMANCE COMPARISONS

The proposed GF multiplier used for the MMUL and MMAC instructions is implemented with the combinational circuit and can perform high-speed GF multiplication. However, the general ALU of existing DSP chips takes quite a number of

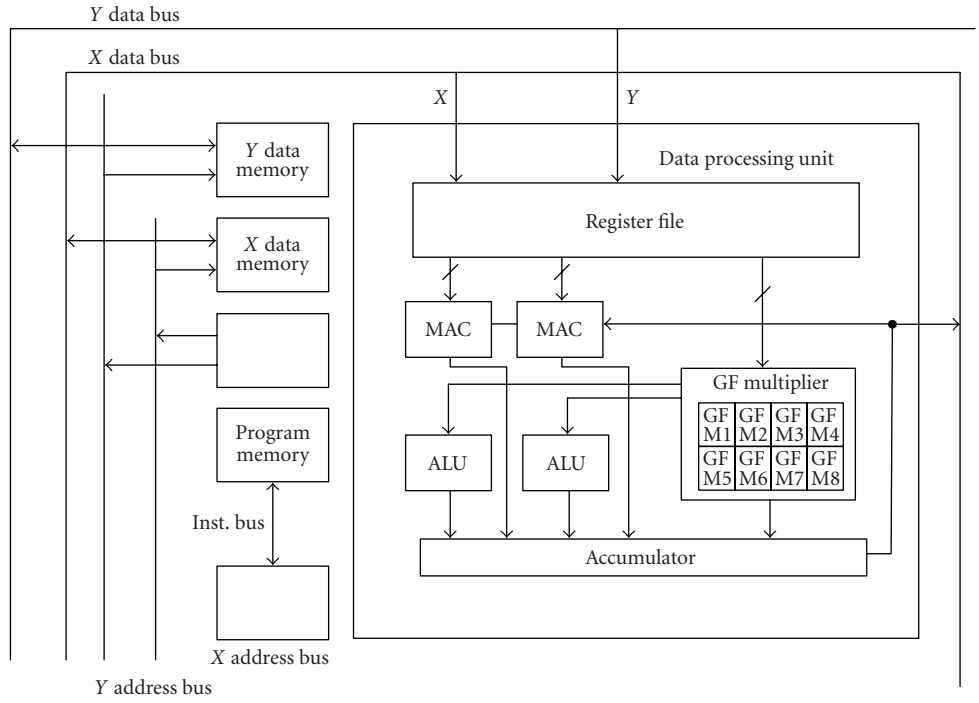

Figure 12: Overall architecture of the proposed ASDSP.

Table 1: Performance comparisons of the RS decoding for (204 188 8) RS code in various DSP chips.

The structure of DSP	The error correction capability (t)	Estimation	Overall latency (clock cycles)
TMS320C64x family [25]	$t = 8$	Syndrome computation (470) + Berlekamp-Massey (246) + Chien search (318) + Forney (146)	1,184
STARCORE SC140 [24]	$t = 2$	—	819~1,115
Hardwired ASIC chip [16]	$t = 8$	Syndrome computation (204) + modified Euclid's algorithm (17) + Chien search (8) + Forney (8)	237
The ASDSP having the proposed GF multiplier	$t = 8$	Syndrome computation (408) + modified Euclid's algorithm (215) + Chien search (211) + Forney (96)	930

clock cycles just for a GF multiplication, since it has to repeat the AND, SHIFT, and XOR instructions shown in Figure 7. Table 1 shows the performance comparisons of RS decoding between the ASDSP having 8 proposed GF multipliers shown in Figure 10 and the existing DSP chips [17, 18, 25]. Note that the performance figures of commercial DSP chips are given by their datasheets or references [17, 18]. The hardwired RS ASIC takes about 237 cycles for $t = 8$ [16], that is, 204 cycles for the syndrome calculation block, 17 cycles for the modified Euclid's algorithm block, 8 cycles for the Chien search block, and 8 cycles for the Forney algorithm.

The proposed architecture takes one clock cycle per MMAC instruction, therefore, 470 clock cycles for the syndrome computation, 85 clock cycles for the modified Euclid's algorithm, 211 clock cycles for the Chien search, and 96 clock cycles for the Forney algorithm are needed for the RS decoding. Hence, The ASDSP takes 930 clock cycles for the RS decoding and it can correct up to 8 symbol errors.

The overall latency of the SC140 takes between 819 clock cycles and 1115 clock cycles for $t = 2$. However, it has less error correction capability ($t = 2$) than the ASDSP ($t = 8$). The overall latency of the SC140 becomes more than double for $t = 8$. In addition, the proposed ASDSP reduces the overall latency by 25% compared with TMS320C64x, supporting only the GF multiplication but not the modulo MAC operation. Moreover, these VLIW DSPs have much larger hardware size and higher power consumption than the proposed one has. Thus, the ASDSP having the proposed GF multiplier shows better performance than the other DSP chips in Table 1.

5. CONCLUSIONS

This paper proposed new ASDSP instructions and their hardware accelerator for high-speed RS decoding. First, we proposed MMAD, MMUL, and MMAC instructions that are necessary to perform the RS decoding and proposed architecture to support these instructions. The proposed GF multiplier, having little extra hardware overhead, can perform the GF multiplication faster than the general ALU of existing DSP chips in terms of execution cycles. Hence, the proposed ASDSP having the proposed GF multiplier can support an RS decoding rate up to 228.1 Mbps at a 130 MHz operating frequency even with the 0.25 μm technology. In addition, the ASDSP can be adapted to various communication standards and can support SDR because of programmability. In the near future, all of these features will be implemented on an ASDSP chip.

ACKNOWLEDGMENTS

This work was supported in part by the National Research Laboratory (NRL) Program of Ministry of Science & Technology (MOST), in part by the HY-SDR Research Center under the ITRC Program of MIC, and in part by IC Design Education Center (IDEC).

REFERENCES

[1] R. Machauer, A. Wiesler, and F. Jondral, "Comparison of UTRA-FDD and CDMA200 with intra- and intercell interface," in *Proc. IEEE 6th International Symposium on Spread Spectrum Techniques and Applications (ISSSTA '00)*, vol. 2, pp. 652–656, NJ, USA, September 2000.

[2] J. Glosser, J. Moreno, M. Mudsill, et al., "Trends in compilable DSP architecture," in *Proc. Workshop on Signal Processing Systems (SiPS '00)*, pp. 181–199, IEEE Press, Lafayette, Ind, USA, October 2000.

[3] HomePlug Powerline Alliance, "Medium Interface Specification. Release 0.5," November 2000.

[4] DVB, "Framing structure, channel coding and modulation for digital terrestrial television," ETSI EN 300 744, vol. 4.1, January 2001.

[5] ATSC, "ATSC Digital Television Standard, ATSC standard A/53B," August 2001.

[6] DAVIC 1.4 Specification. Part 8, "Lower Layer Protocols and Physical Interface," 1998.

[7] A. M. Michelson and A. H. Levesque, *Error-Control Techniques for Digital Communication*, John Wiley & Sons, NY, USA, 1985.

[8] T. R. N. Rao and E. Fujiwara, *Error Control Coding for Computer Systems*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1989.

[9] J.-M. Hsu and C.-L. Wang, "An area-efficient pipelined VLSI architecture for decoding of Reed-Solomon codes based on a time-domain algorithm," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no. 6, pp. 864–871, 1997.

[10] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Trans. on VLSI Systems*, vol. 9, pp. 641–655, October 2001.

[11] M. A. A. Ali, A. Abou-El-Azm, and M. F. Marie, "Error rates for non-coherent demodulation FCMA with Reed-Solomon codes in fading satellite channel," in *Proc. IEEE Vehicular Techn. Conf. (VTC '99)*, vol. 1, pp. 92–96, Amsterdam, The Netherlands, September 1999.

[12] T. K. Matsushima, T. Matsushima, and S. Hirasawa, "Parallel architecture for high-speed Reed-Solomon codec," in *Proc. IEEE Int. Telecommun. Symp. (ITS '98)*, vol. 2, pp. 468–473, São Paulo, Brazil, 1998.

[13] H. M. Shao, T. K. Truong, L. J. Deutsch, J. H. Yuen, and I. S. Reed, "A VLSI design of a pipeline Reed-Solomon decoder," *IEEE Trans. on Computers*, vol. 34, no. 5, pp. 393–403, 1985.

[14] H. M. Shao and I. S. Reed, "On the VLSI design of a pipeline Reed-Solomon decoder using systolic arrays," *IEEE Trans. on Computers*, vol. 37, no. 10, pp. 1273–1280, 1988.

[15] H. H. Lee, M. L. Yu, and L. Song, "VLSI design of Reed-Solomon decoder architectures," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '00)*, vol. 5, pp. 705–708, Geneva, Switzerland, May 2000.

[16] J. H. Baek, J. Y. Kang, and M. H. Sunwoo, "Design of a high-speed Reed-Solomon decoder," in *Proc. IEEE Int. Symp. Circuits and Systems (ISCAS '02)*, pp. 793–796, Scottsdale, Ariz, USA, May 2002.

[17] J. Sankaran, "Reed Solomon decoder: TMS320C64x Implementation," Tech. Rep. SPRA686, Texas Instruments, Dallas, Tex, USA, December 2000.

[18] D. Taipale, I. E. Scheiwe, and T. M. Redheendran, "Reed-Solomon Decoding on the StarCore Processor," Tech. Rep. AN1841/D, Motorola Semiconductors, Denver, Colo, USA, May 2000.

[19] M. H. Sunwoo and J. S. Lee, "The circuits for modulo operation and operation method of programmable processor for Reed-Solomon encoding and decoding," Korea Patent Application No. 10-2001-0022427, 2001.

[20] I. S. Reed and X. Chen, *Error-Control Coding for Data Networks*, Kluwer Academic, Norwell, Mass, USA, 1999.

[21] S. Lin and D. J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1983.

[22] M. Bossert, *Channel Coding for Telecommunications*, John Wiley & Sons, NY, USA, 1999.

[23] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications*, IEEE Press, NY, USA, 1994.

[24] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1995.

[25] Motorola Semiconductors, "SC140 DSP core reference manual," Denver, Colo, USA, 2000.

Jung H. Lee received the B.S. degree in electronic engineering from Ajou University, Suwon, Korea in 2002. He is currently working toward the Ph.D. degree in the School of Electrical and Computer Engineering, Ajou University. His main research interests include SOC design and application-specific DSP chip design.

Jaesung Lee received the B.S. and M.S. degrees in electronic engineering from Ajou University, Suwon, Korea in 1999 and 2001, respectively. He is currently working in the Electronics and Telecommunications Research Institute (ETRI) in Taejon, Korea. His research interests include VLSI architectures, design of parallel processors, DSP chips, and protocol processing.

Myung H. Sunwoo received the B.S. degree in electronic engineering from Sogang University in 1980, the M.S. degree in electrical and electronics engineering from Korea Advanced Institute of Science and Technology in 1982, and the Ph.D. in electrical and computer engineering from The University of Texas at Austin in 1990. He worked for Electronics and Telecommunications Research Institute (ETRI) in Taejon, Korea from 1982 to 1985 and Digital Signal Processor Operations Division, Motorola, USA from 1990 to 1992. Since 1992, he has been a Professor with School of Electrical and Computer Engineering, Ajou University, Suwon, Korea. His research interests include VLSI architectures, SOC design for multimedia and communications, and application-specific DSP chip design. He is the author of more than 110 journal and conference papers. He has served as a Technical Program Chair of the IEEE Workshop on Signal Processing Systems (SIPS) in 2003, as a member of Technical Committee of the IEEE Circuit and Systems VSATC since 1996, and as a member of Program Committee of the IEEE Workshop on SIPS and the IEEE International SOC Conference. He serves as an Associate Editor for the IEEE Transactions on Very Large Scale Interation Systems from 2001. He is a Senior Member of IEEE.