DBpia

Reed-Solomon 부호화/복호화를 위한 DSP 명령어 및 하드웨어 설계

정회원 이 재 성*, 선 우 명 훈**

Design of DSP Instructions and their Hardware Architecture for Reed-Solomon Codecs

Jae-Sung Lee*, Myung-Hoon Sunwoo** *Regular Members*

요 약

본 논문은 오류 정정을 위해 가장 많이 쓰이는 알고리즘 중 하나인 RS (Reed- Solomon) 부호화 및 복호화를 DSP (Digital Signal Processor) 칩에서 효율적으로 구현할 수 있는 새로운 명령어 및 하드웨어 구조를 제안한다. 제안한 구조는 원시 다항식의 변경에 따라 하드웨어를 재 설계할 필요가 없이 DSP 상에서 프로그램으로 변경이 가능하여 다양한 원시 다항식을 구현할 수 있다. 새로운 명령어 및 하드웨어 구조는 유한체 곱셈기 및 가산기를 이용하여 유한체 연산을 수행한다. 따라서, 제안한 DSP 구조는 기존 DSP 칩과 비교하여 복호화 속도를 향상시킬 수 있다. 본 하드웨어 구조는 130MHz 동작 주파수를 갖는 DSP 칩에서 228.1 Mbps의 RS 복호화 성능을 갖는다.

Key Words : RS, DSP, instruction, hardware architecture.

ABSTRACT

This paper presents new DSP (Digital Signal Processor) instructions and their hardware architecture to efficiently implement RS (Reed-Solomon) codecs, which is one of the most widely used FEC (Forward Error Control) algorithms. The proposed DSP architecture can implement various primitive polynomials by program, and thus, hardwired codecs can be replaced. The new instructions and their hardware architecture perform *GF* (Galois Field) operations using the proposed *GF* multiplier and adder. Therefore, the proposed DSP architecture can significantly reduce the number of clock cycles compared with existing DSP chips. It can perform RS decoding rate of up to 228.1 Mbps on 130MHz DSP chips.

Ⅰ. 서 론

현재의 통신기술 동향은 반도체 기술의 급격한 발달과 더불어 유선으로는 VDSL (Very-high-data-rate Digital Subscriber Line) 모뎀, 케이블 모뎀, 전력선 모뎀 등의 개발이 활발하게 진행되고, 무선으로는 WCDMA (Wideband Code Division Multiple Access)와 CDMA2000을 바탕으로 하여 IMT-2000 (International Mobile Telecommuni- cations - 2000)이라는 3세대 이동통신서비스가 등장하고 있는 등 통신 분야의 급속한 발전이 거듭되고 있다. 또한 최근 들어 전용 하드웨어로 설계하였던 통신 시스템을 소프트웨어적으로 재구성이 가능한 공통의 하드웨어를 사용하여 구성하고, 소프트웨어의 업그레이드(upgrade)만으로 다양한 통신방식에 적용할 수 있는 시스템인 SDR (Software Defined Radio)의 연구가

* 한국전자통신연구원(ETRI) 컴퓨터시스템연구부 (ljshide@etri.re.kr), ** 아주대학교 전자공학부 (sunwoo@ajou.ac.kr)

논문번호 : 020403-0911, 접수일자 : 2003년 9월 11일

※본 연구는 과학기술부에서 시행하는 국가지정연구실사업, KOSEF R01-1997-000026-0 특정기초연구 및 IDEC의 부분적인 지원을 받아 수행되었습니다.

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

진행 중이다[1]. 그러나 기존의 ASIC (Application-Specific IC) 기반의 칩들은 다양한 통신 표준에 대한 유연성 (flexibility) 부재, 개발비용 상승, 개발기간 장기화 등의 문제에 봉착하여 사용의 한계를 나타내고 있다. 이러한 제약에 따라 여러 면에서 이점을 가질 수 있는 DSP (Digital Signal Processor) 기반의 통신용 시스템 개발로 그 구현 방식이 전환되고 있다[2]. DSP칩은 기존 ASIC 칩에 비해서 업그레이드 및 구조의 변경이 용이하고, 신속한 시장 확보 (Time-to-Market)가 가능하다.

본 논문은 디지탈 통신 분야에서 채널 오류 검출과 정정을 위해 가장 많이 사용하는 알고리즘 중 하나인 RS (Reed-Solomon) 부호화 및 복호화의 효율적인 처리를 위한 DSP 명령어 및 구조를 제안한다. RS 부호화 및 복호화 과정에서 사용하는 다양한 알고리즘 블록들은 단순히 유한체 곱셈과 덧셈을 반복하는 구조를 가지고 있다. 그러나 원시 다항식에 따라 전용 유한체 연산기라는 특수 구조가 필요하므로 범용 DSP나 마이크로 프로세서에서는 구현하기가 어려워 ASIC 칩으로 설계하여 사용하였다[3-7]. 그러나 RS 전용 ASIC 칩의 경우 휴대용 단말기에서 단지 RS 부호화/복호화 만을 위해서 사용하기에는 하드웨어 크기 및 전력 소모가 크고 특히, 원시 다항식의 다양성에 유연하게 대처하지 못하므로 적용되는 표준안마다 전용 칩을 재설계해야 한다. 따라서 칩 개발비용 및 단가가 올라가며 개발기간도 오래 걸리는 문제점들이 있다.

RS 알고리즘은 유한체 연산기를 사용하기 때문에 기존의 범용 DSP[8,9]가 보유하고 있는 일반 ALU 구조로 RS 알고리즘을 구현하기에는 싸이클 소모가 많으므로 고속 연산이 불가능하다. 유한체 연산기를 사용하지 않고 LUT (Look-Up Table)를 사용하는 방법은 많은 저장 공간이 필요하여 큰 용량의 메모리를 필요로 하므로 억세스 지연 및 파워 소모가 크다. 따라서, 범용 DSP 칩을 이용한 RS 구현은 아직까지 어려운 실정이다. 그러나 다양한 원시 다항식에 적용 가능한 유한체 연산기 회로를 갖는 DSP 칩은 RS 알고리즘의 고속 연산을 가능하게 하며 무선 통신용 단말기에서 FEC 전용 칩을 사용하지 않고도 단일 DSP에서 RS 부호화 및 복호화를 수행할 수 있다[10-11]. 또한, 단말기의 구조가 단순화되어 휴대가 용이하고 시스템 개발 비용을 대폭 줄일 수 있으며 저전력 효과를 가져올 수 있다. 따라서, RS 알고리즘을 위한 특정 명령어를 지원하는 ASDSP (Application-Specific DSP)는 다양한 통신 표준을 지원할 수 있으며, 다중 표준, 다중 밴드의 고속 데이터 전송을 위한 4세대 무선 통신의 핵심 기술인 SDR에 적합한 구조이다.

본 논문은 다음과 같이 구성된다. 2장에서는 전용 RS 프로세서 및 기존 DSP 칩[8-9]에서의 RS 알고리즘을 분석하고[3-7], 3장에서는 새로운 DSP 명령어 및 하드웨어 구조를 제안하며, 기존의 범용 DSP 들과의 성능 비교를 수행한다. 마지막으로 4장에서 결론을 맺는다.

Ⅱ. 전용 RS 프로세서 및 기존 DSP 칩에서의 RS 구현

본 장에서는 전용 RS 프로세서 및 기존의 범용 DSP에서 사용되는 RS 알고리즘 및 연산 방식을 알아보고, 하드웨어 구조를 분석한다.

2.1 전용 RS 프로세서

전용 RS 프로세서는 사용되는 분야에 따라 각각의 다른 연산 블록을 하드웨어로 구현하여 병렬로 처리하는 RS 전용 ASIC 칩이다. 따라서 유무선 통신 표준안이 요구하는 전송률보다 훨씬 빠른 전송률을 낼 수 있지만, 원시 다항식의 다양성에 유연하게 대처하지 못하므로 적용되는 표준안마다 전용 칩을 재 설계해야 하는 단점을 갖는다.

1) RS 부호기 구조

16 (2t) 개의 잉여 심볼을 첨가하는 구조를 지닌 RS의 부호기를 예로 들어 설명한다. 이를 위한 생성 다항식은 식 (1)과 같다[12-14].

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

그림 1은 생성다항식에 따라 구현한 LFSR (Linear Feedback Shift Register) 구조를 갖는 RS 부호기 블록도를 나타낸다. 각각의 레지스터는 0으로 초기화된 상태에서 시작하여 메시지 다항식 $m(x)$가 들어오고 LFSR 구조를 통해 생성 다항식 $g(x)$와 결합하여 연산을 수행한다. 메시지 다항식 $m(x)$의 입력이 끝나면 레지스터에 남아있는 값들은 패리티 심볼로 메시지 다항식 $m(x)$의 뒤에 연결되어 차례로 출력된다.

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

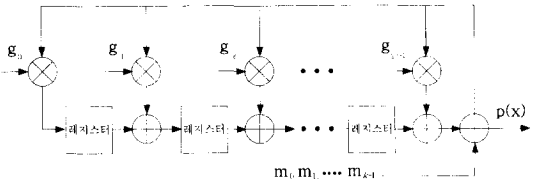


그림 1. LFSR 구조를 갖는 RS 부호기 블록도

2) RS 복호기 구조

RS 부호의 복호 순서는 오류 패턴 함수인 신드롬 (syndrome) 값을 계산하고 오류 위치 다항식 (error locator polynomial)을 결정하여 오류 위치를 찾는다. 다음으로 오류값 (error value)을 결정하여 오류를 정정한다. 이 과정에서 사용하는 오류 위치 및 오류 크기 다항식을 얻는 것은 복호 과정에서 가장 중요하며 베르캠프-메세이 (Berlekamp-Massey), 유클리드 (Euclid), 수정 유클리드 (Modified Euclid) 등의 알고리즘이 사용되고 있다. 그림 2는 전용 RS 복호기의 일반적인 블록도이다[12-16].

신드롬 값을 연산하는 블록은 그림 3에 도시하였으며 부호기에서 사용된 생성 다항식 $g(x)$의 근을 이용하여 신드롬 다항식을 계산하게 된다. 신드롬 다항식은 수신된 코드 워드의 오류 패턴 (pattern)을 나타내 주며, 이를 이용하여 오류 정정을 위한 키 해독을 수행한다.

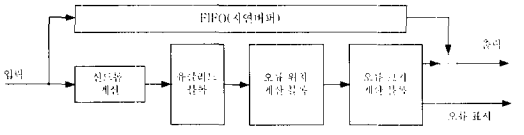


그림 2. 일반적인 RS 복호기의 블록도

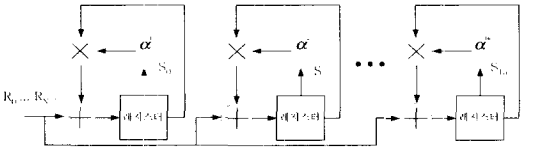


그림 3. RS 복호기의 신드롬 계산 블록

신드롬 블록의 셀 개수는 RS 코드의 오류 정정 능력에 따라 정해지게 되는데, 오류 정정 심볼 수의 두배인 2t 개의 셀이 필요하다. 즉, 생성 다항식 근의 수만큼 셀이 필요하다. 따라서, RS 복호기의 오류 정정 능력 (t)이 8 인 RS 복호기의 경우 신드롬 블록은 2t = 16개의 셀이 필요하며, 그림 3처럼 형성하고 있다.

신드롬 회로에 의해 계산된 신드롬 값들은 신드롬 다항식의 계수이며, 이 다항식을 사용하여 오류 위치 및 오류 크기 다항식을 연산한다. 오류 위치 다항식 및 오류 크기 다항식의 계산은 RS 디코딩 과정 중 가장 복잡하고, 처리 시간이 많이 소모되는 부분으로서 RS 복호기의 크기 및 동작 속도에 가장 많은 영향을 준다. 오류 위치 다항식을 계산하기 위한 알고리즘에는 베르캠프-메세이, 유클리드, 수정 유클리드 등의 알고리즘이 있다.

일반적으로 베르캠프-메세이 알고리즘의 경우 유클리드 알고리즘에 비해 단순한 구조를 이용 작은 하드웨어 면적을 갖는 RS 복호기 구현이 가능하다. 그러나 베르캠프-메세이 알고리즘은 직렬 (Serial) 구조의 경우 큰 지연 시간을 갖으며, 고속의 복호를 위한 병렬 (Parallel) 구조는 큰 하드웨어 면적을 필요로 하여 고속의 RS 복호기에 적합하지 않다. 그러나 유클리드 알고리즘 연산 회로는 베르캠프-메세이 회로 보다 하드웨어 크기는 크지만 고속 구현에 유리하며, 기본 블록의 규칙성으로 인하여 RS 복호기 구현이 쉽다. 그림 4는 수정 유클리드 알고리즘 연산기의 구조를 나타낸다[4].

수정 유클리드 알고리즘은 유클리드 알고리즘이 갖고 있는 몫을 구하는 부분에 대한 구현 어려움의 단점을 해소하고 고속 연산을 수행할 수 있도록 개선된 알고리즘이다. 따라서, 몫을 위한 LUT를 필요로 하지 않고 효율적으로 ASIC 칩의 면적을 줄일 수 있으며, RS 복호기 구현이 보다 쉽고, 빠르게 키 해독을 할 수 있다.

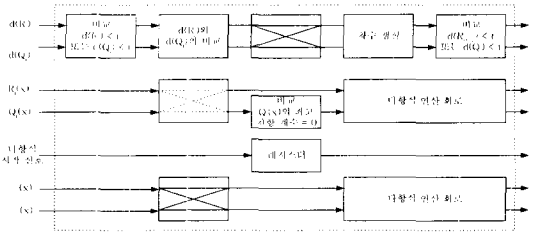


그림 4. 수정 유클리드 알고리즘 연산기 구조

유클리드 알고리즘 등을 이용하여 오류 위치 다항식과 오류 크기 다항식을 얻으면, Chien Search[7,14-15] 방법과 Forney 알고리즘[7,8]을 이용하여 오류 위치를 얻고, 그에 따른 오류 크기를 구한다. Chien Search 방법은 근을 대입하는 데 있어서 하드웨어로 구현이 가능하도록 개선한 방법으로서 그림 5는 Chien

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

Search 알고리즘을 이용하여 오류 위치 다항식의 근을 구하는 블록도이다.

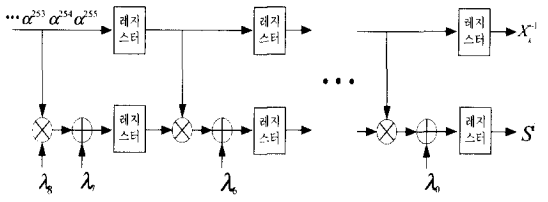


그림 5. Chien Search 알고리즘 블록도

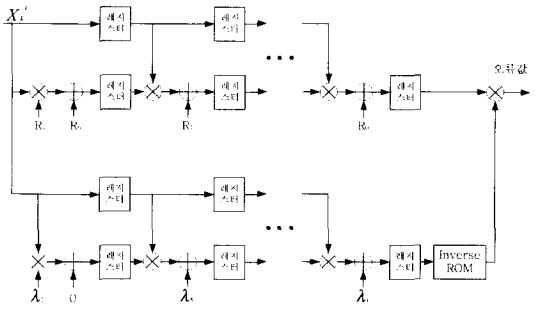


그림 6. Forney 알고리즘 블록도

그림 5에서 오류 위치 다항식의 계수 (λ_i)를 이용하여 오류 위치에 대한 근을 구하고 그림 6에서 오류 위치 다항식의 계수 (λ_i)와 오류 크기 다항식의 계수 (R_i)를 이용하여 오류 위치에 따른 오류 크기를 계산한다. 그림 5와 6에서 알 수 있듯이 오류 위치 및 크기를 계산하는 데 각각 모듈로 MAC (Multiply - Accumulate) 연산을 t, (t-1)+t/2번 필요로 한다.

지금까지 설명한 RS 전용 ASIC 칩은 모두 모듈로 곱셈 및 덧셈을 수행하는 유한체 연산기를 필요로 한다. 또한 칩이 사용되는 분야의 표준안이 정하는 원시 다항식에 따라 유한체 연산기의 구조가 변경되어야 하므로 사용 분야마다 칩을 재개발하여야 한다.

2.2 범용 DSP 칩에서 RS 구현

기존 범용 DSP 칩을 사용하여 RS 복호기의 구현은 가능하다. 그러나 기존 범용 DSP 칩을 사용하여 유한체 연산을 처리하면 ALU 연산 유닛을 여러 번 반복 사용하므로 많은 수행 사이클이 필요하다. 즉, 유한체 연산 부분을 처리하기 위해 ALU의 연산 유닛들을 여러 번 반복 사용하여 처리하고, 이를 서브루틴 형식으로 프로그래밍한 후 메인 프로그램이 RS 알고리즘을 수행하다가 유한체 연산부분에서 이 서브루틴을 사용하도록 하였다[12]. 식 (2)는 m이 8인 경우 모듈로 곱셈에 사용되는 곱셈식을 나타낸다. 모듈로 곱셈식은 그림 1, 3, 5, 6에서 ⊗ 기호로 표현되었던 부분들이다. 모듈로 연산에서 덧셈 및 뺄셈은 모두 XOR 논리 연산으로 표현될 수 있어 ALU의 XOR 연산을 사용하면 가능하다. 그러나, 식 (2)의 곱셈의 경우는 곱셈 결과의 비트 수가 8 비트를 초과하므로 모듈로 연산을 수행하여야 한다.

$$
\begin{aligned}
A(x) &= A_7 x^7 + A_6 x^6 + A_5 x^5 + A_4 x^4 \\
&\quad + A_3 x^3 + A_2 x^2 + A_1 x^1 + A_0 x^0 \\
B(x) &= B_7 x^7 + B_6 x^6 + B_5 x^5 + B_4 x^4 \\
&\quad + B_3 x^3 + B_2 x^2 + B_1 x^1 + B_0 x^0 \\
A(x)\times B(x) &= (A_7 \cdot B_7) x^{14} \\
&\quad + (A_7 \cdot B_6 \oplus A_6 \cdot B_7) x^{13} \\
&\quad \cdots (A_1 \cdot B_0 \oplus A_0 \cdot B_1) x^1 \\
&\quad + (A_0 \cdot B_0)
\end{aligned}
\tag{2}
$$

그림 7은 RS 복호를 지원하지 않는 일반 범용 DSP를 사용하여 모듈로 곱셈 연산을 수행하는 과정을 보여준다. 식 (2)를 구현하기 위해 우선 8 비트 데이터 (A)의 LSB부터 1 비트씩 (B)의 8비트와 AND 연산을 과정 ①에 각각 수행한다. 다음으로 과정 ②에서 과정 ①의 결과를 자리수에 맞추어 쉬프트를 취한다. 마지막으로 과정 ③에서 8개의 15 비트 데이터들을 비트 별로 XOR 연산을 수행하여 식 (2)의 세 번째 수식에 해당하는 15 비트 데이터를 얻는다. 마지막으로 15 비트 데이터를 8 비트화 하기 위해 주어진 원시 다항식에 따라 모듈로 연산을 취한다. 모듈로 연산은 XOR로 구현 가능하다. 결과적으로 ⊗ 기호의 모듈로 곱셈 연산을 위해 걸리는 DSP 연산 수행 사이클은 16 비트 DSP의 경우 약 249 명령어 사이클이 걸린다. 이러한 과정은 사용하는 DSP가 비트 및 바이트 억세스 가능하여야 하며 32 비트 DSP의 경우 병렬로 2 개의 모듈로 곱셈을 수행할 수 있고, 64 비트 DSP 라면 4 개까지 가능하다. 또한, 동시 가동할 수 있는 ALU가 여러 개라면 1/N (N은 ALU 개수) 로 사이클 수를 절감할 수 있다. 그러나, 만약 비트별 억세스가 불가능한 경우 마스킹 등에 사용되는 사이클이 추가되어 더 많은 사이클이 걸린다.

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

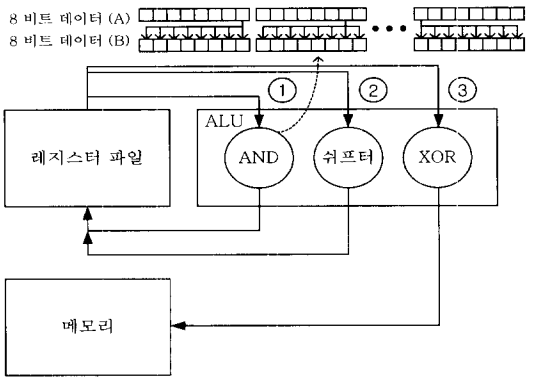


그림 7. 기존 DSP에서 모듈로 곱셈 연산 구현

따라서 기존 범용 DSP 칩의 하드웨어는 모듈로 곱셈 연산을 위해 많은 연산 사이클을 필요로 하므로 고속의 RS 복호기를 구현할 수 없었다. 최근들어 고속의 RS 복호를 지원하기 위하여 TMS320C64x는 모듈로 연산기와 명령어를 내장 하였지만 위의 칩은 모듈로 곱셈만 지원할 뿐 RS 복호에서 많이 쓰이는 모듈로 곱셈과 덧셈의 연속처리는 지원하지 않는다[8]. 또한 이 칩은 VLIW 구조이므로 하드웨어 크기가 크고 전력소모가 많은 단점이 있다. SC140는 모듈로 연산을 지원하지 않고 LUT를 사용하는 구조로써 역시 VLIW 구조로 동일한 단점을 가지고 있다[9]. 결론적으로, 기존 범용 DSP를 사용한 RS 복호기는 저속의 통신 시스템에서만 사용 가능하다.

LUT를 사용하는 구현 방법은 모듈로 연산 결과를 미리 ROM 또는 RAM과 같은 저장 장치에 저장해 놓고 억세스 하는 방법이다[18]. 이 방법은 m이 8인 경우 $2^8 \times 2^8 = 64K$ 바이트의 큰 저장 공간이 필요하므로 고집적도의 DSP 경우라도 수 Kbyte의 내부 RAM 또는 ROM을 갖고 있어 내부 메모리만을 사용해서는 구현이 어려워 외부 메모리를 사용해야 한다. LUT를 이용한 방법은 DSP의 비트수에 상관없이 한 번에 한 개의 곱셈 연산만 가능하며 내부 및 외부 메모리의 억세스 속도가 느려 수 ~ 수십 싸이클이 걸린다. 따라서, 대용량의 내부 메모리를 보유하고 고속으로 동작하는 DSP가 아니면 LUT를 사용한 구현은 거의 사용하지 않는 방법이다.

Ⅲ. 새로운 DSP 명령어 및 하드웨어 구조 제안

본 장에서는 RS용 유한체 연산을 위한 새로운 DSP 명령어 MADD (Modulo-Add), MMUL (Modulo-Multiply), MMAC (Modulo-MAC)와 이 명령어들의 구체적 수행을 위한 하드웨어 구조를 제안한다. 3.1절에서는 RS용 명령어 및 하드웨어 구조를 설계하고, 3.2절에서 기존 DSP 칩과의 성능 비교를 수행한다.

3.1 새로운 DSP 명령어 MADD, MMUL, MMAC 및 하드웨어 구조 설계

RS용 알고리즘 블록들의 구현을 위한 하드웨어 구조는 곱하고 더하는 그림 8의 단순한 구조를 계속해서 반복 사용한다. 유클리드, 수정 유클리드 알고리즘 등 최근 여러 논문에서[3-7,12] 발표되고 있는 다양한 RS용 알고리즘 블록들 역시 그림 8의 구조를 기본 요소로 가지고 있다. 실제 프로그래머블 범용 DSP 칩에서 RS 부호화/복호화기를 구현하지 못하는 이유는 바로 이 기본 구조 때문이다. 즉, 그림 8의 $\otimes$ 와 $\oplus$ 기호는 모듈로 연산으로서 일반 산술이나 논리 연산과는 전혀 다른 곱셈과 덧셈 연산을 수행한다. 그러나 일반 범용 DSP의 경우 유한체 곱셈기 및 덧셈기를 지원하지 않는다. 따라서 다양한 통신 표준에 따라서 다양한 원시 다항식을 지원 가능한 프로그램머블한 구조를 갖는 ASDSP가 필요하다.

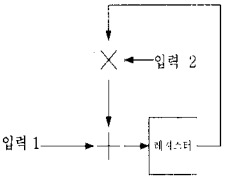


그림 8. RS 부호화기/복호화기에 반복 사용되는 셀 구조

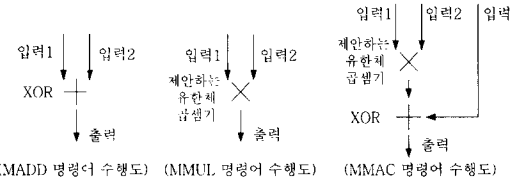


그림 9. MADD, MMUL, MMAC 연산 수행도

그림 9는 제안한 MADD, MMUL, MMAC 명령어를 나타낸다. MADD 명령어는 모듈로 덧셈 연산을 수행하며, 기존 ALU의 XOR 명령어를 사용하여 구

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

현 가능하다. 따라서 MADD 명령어의 사용에 따른 추가적인 하드웨어 부담은 필요하지 않다. MMUL 명령어는 그림 10의 제안하는 유한체 곱셈기를 사용하여 모듈로 곱셈 연산을 수행한다. 제안하는 유한체 곱셈기는 XOR 및 AND 게이트로 구성된 작은 하드웨어의 추가를 통해서 모듈로 곱셈 연산을 효율적으로 수행할 수 있다. MMAC 명령어는 MADD 및 MADD 명령어를 사용하여 구성 가능하다. MMAC 명령어는 한 사이클동안 모듈로 MAC 연산을 수행한다.

제안하는 명령어들은 그림 1의 RS 부호기, 그림 2의 신드롬 연산 블록, 그림 5와 6의 Chien Search 및 Forney 알고리즘 블록 등 RS 알고리즘 구현에 폭넓게 사용된다. 반면에 Texas Instruments 사의 TMS320C64x의 경우 모듈로 곱셈 연산만을 지원하고 모듈로 MAC 연산을 지원하지 않는다. 따라서 제안하는 명령어 및 하드웨어 구조는 DSP를 사용한 RS 부호기 및 복호기의 성능을 향상시킬 수 있다.

그림 10은 $GF(2^m, m = 8)$의 제안하는 프로그래머블 유한체 곱셈기 블록도이며, MMUL 및 MMAC 명령어는 제안하는 유한체 곱셈기를 사용하여 구현된다. 두 개의 8 비트 데이터 a와 b의 각각의 비트 별 곱셈 (AND)쌍들을 처리한 뒤 비트별 덧셈(XOR)을 수행하여 중간 곱셈 결과인 15 비트의 $\omega(i)$ 값을 얻는다. 이 15 비트의 $\omega(i)$ 값을 m과 원시다항식에 따라 모듈로 연산을 취하여 8 비트의 $\Omega(i)$를 얻는다. 제안하는 유한체 곱셈기는 원시 다항식 디코더를 포함하여 약 630개의 게이트를 사용하여 구현할 수 있다. 제안한유한체 곱셈기의 게이트 숫자는 전용의 RS ASIC 칩[7]이 사용하는 유한체 곱셈기의 개수 87개 보다 크다. 그러나 8개의 오류 정정이 가능한 전용 RS ASIC 칩은 신드롬 연산 블록 16개, 수정 유클리드 연산 블록 64개, Chien Search 블록 8개, Forney 알고리즘 블록에 1개의 유한체 곱셈기를 사용하므로, 전체 89개의 유한체 곱셈기를 필요로 한다. 따라서 8개의 제안하는 유한체 곱셈기를 사용하는 ASDSP의 경우 전용의 ASIC 칩에 비해서 훨씬 적은 게이트를 사용한다. 즉, ASDSP는 작은 하드웨어 추가를 통해서 보다 향상된 RS 복호 성능을 얻을 수 있다. m 값이 8보다 큰 경우 덧셈기의 경우 XOR 게이트의 추가로 구현 가능하며, 그림 10의 유한체 곱셈기 역시 AND 및 XOR 게이트의 추가로 간단히 구현 가능하다.

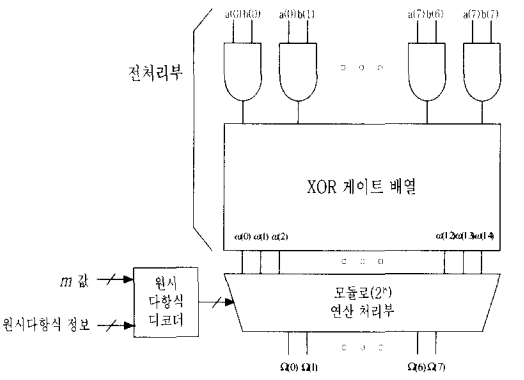


그림 10. 제안하는 유한체 곱셈기의 블록도 ($GF(2^m, m = 8)$)

그림 10의 모듈로 연산 처리부는 차수 m과 원시 다항식 정보를 디코딩하여 얻은 제어 신호를 입력받아 모듈로 연산을 수행한다. 그림 11은 AND 및 XOR 게이트로 구성된 제안하는 모듈로 연산 처리부를 나타낸다. 15 비트의 $\omega(i)$는 원시 다항식 디코더에서 출력된 제어 신호에 따라서 XOR 연산이 수행되며, 그 결과 8비트의 $\Omega(i)$값이 모듈로 연산 처리부로부터 출력된다. 식 (3)의 방정식은 원시 다항식이 $x^8+x^4+x^3+x^2+x = 0$이며 $m = 8$인 경우의 모듈로 연산 결과이다. 원시 다항식 디코더는 RS 사양 (m, 원시 다항식)들에 따라 모듈로 연산 처리부에 $\omega(i)$ 신호의 비트들을 XOR 게이트에 입력할 것인가 하지 않을 것인가에 대한 정보가 들어 있다.

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

다양한 통신 규격에서 많이 사용되는 m 및 원시 다항식을 선별하면 대략 8 가지 정도면 충분하므로 3 비트 ($8=2^3$) 입력을 받아 m = 8 일 경우 그림 11와 같이 15 × 8 = 120 비트의 제어 신호를 출력한다. 제안하는 유한체 곱셈기는 이 제어 신호를 사용하여 모듈로 연산을 수행한다. 원시 다항식 디코더는 조합 논리 회로를 사용하여 구현되었다.

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

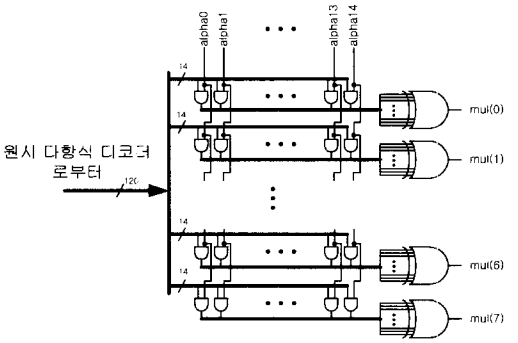


그림 11. 제안하는 모듈로 연산 처리부 ($GF(2^m, m=8)$인 경우)

3.2 제안한 구조의 성능 평가

MMUL 및 MMAC 명령어의 수행을 위해 제안한 유한체 곱셈기는 조합 논리 회로를 사용하여 구현하였으며, 고속의 모듈로 곱셈 연산을 수행한다. 그러나 기존 범용 DSP 칩의 ALU는 모듈로 곱셈 연산을 위해서 그림 7과 같이 AND, SHIFT, XOR 연산을 반복적으로 수행해야 한다. 따라서 범용 DSP의 경우 모듈로 곱셈 연산을 위해서 약 249 사이클이 필요하다. 표 1은 제안한 그림 10의 유한체 곱셈기를 8개 내장한 ASDSP와 기존 범용 DSP[8-9,17-18]의 성능을 비교한 것이다. 상용 DSP 칩의 성능은 자사의 매뉴얼[8-9]에서 인용하였다.

제안한 구조는 MMAC 명령어의 연산시 한 사이클이 걸리므로, (204,188) RS 코드의 복호를 위해서 신드롬 연산 블록 470사이클, 수정 유클리드 연산 블록 87 사이클, Chien Search 블록 211 사이클, Forney 알고리즘 블록 96 사이클이 필요하다. 따라서 ASDSP는 RS 복호를 위해서 총 930 사이클이 필요하며, 8개의 심볼 오류를 정정할 수 있다.

SC140의 경우 ROM 테이블 2.714K 바이트를 사용하고, 6개의 산술 연산 유닛을 사용 쉬프트 연산 등을 이용하여 처리 할 때 최선의 경우 819 싸이클, 최악의 경우 1,115 싸이클이 걸린다[9]. 그러나 SC140은 t = 2일 경우 소요되는 사이클 수를 계산한 것으로 t = 8일 경우 소요 사이클 수는 2배 이상 증가될 것으로 예상된다. 또한, Texas Instruments (TI)사의 TMS320C6400은 모듈로 곱셈 연산을 지원하지만 모듈로 MAC 연산을 지원하지 않는다. 따라서 ASDSP는 TMS320C6400에 비해 RS 복호를 위해 적은 사이클을 필요로 한다. 또한, TI 칩은 ASDSP에 비해서 훨씬 큰 하드웨어 구조를 갖는다. 따라서, 제안하는 유한체 곱셈기를 내장한 ASDSP는 표 1의 다른 DSP 칩에 비해서 보다 우수한 성능을 갖는다.

표 1. DSP 상에서의 RS 복호기 구현 성능 비교
(단위 : 클록 사이클 & Mbps)

	오류 정정 능력 (t)	처리 알고리즘당 소요 사이클수	전체소요 사이클수	Data rate
TMS320C6400 family[8]	t = 8	신드롬연산(470)-Berlekamp-Massey(246)+Chien Search(318)+Forney(146)	1,184	137.8
STARCORE (SC140)[9]	t = 2	Not specified in the application note	819 ~1,115	146.4 ~200.9
제안하는 유한체 곱셈기를 내장한 ASDSP	t = 8	신드롬연산(408)+수정 유클리드(215)+Chien Search(211)+Forney(96)	**930**	**228.1**

제안한 유한체 곱셈기는 Faraday 0.25 μm 공정을 사용하여 구현하였으며 7.34 ns 내에 하나의 유한체 곱셈을 수행 할 수 있다. 따라서 제안한 칩은 130 MHz로 동작할 수 있으며 t=8인 경우 228.1 Mbps ((204 symbol × 8 bit/930 clock) × 130 MHz)의 RS 복호화 성능을 갖는다.

Ⅳ. 결 론

본 논문은 고속의 RS 복호화를 위해서 새로운 DSP 명령어 및 하드웨어 구조를 제안한다. RS 복호화를 효율적으로 실행하기 위해 새로운 명령어 MMAD, MMUL, MMAC 명령어를 제안하였으며, 이들 명령어의 처리를 위한 하드웨어 구조를 제안하였다. 제안한 유한체 곱셈기는 작은 하드웨어 추가를 통해서 기존 범용 DSP가 ALU를 사용하여 모듈로 연산을 수행하는 것에 비해서 우수한 성능을 갖는다. 제안하는 유한체 곱셈기를 갖는 ASDSP는 8개의 심볼 오류 정정이 가능한 RS 코드에 대해서 100MHz 동작 주파수에서 약 228.1Mbps의 RS 복호화 성능을 갖는다. 본 논문에서 제안한 DSP를 사용하면 RS 복호를 위한 전체 사이클 수를 TI 사의 TMS320C6400 칩에 비해 약 40% 줄일 수 있어 고속의 RS 복호기 구현이 가능하다. 또한, ASDSP는

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

프로그램에 의해서 다양한 통신 표준을 지원할 수 있으며, SDR에 사용 가능하다.

참 고 문 헌

[1] R. Machauer, A. Wiesler, and F. Jondral, "Comparison of UTRA-FDD and cdma2000 with Intra- and Intercell Interface," *IEEE Sixth International Symposium on Spread Spectrum Techniques and Applications*, vol. 2, pp. 652-656, Sept. 2000.

[2] J. Glossner, J. Moreno, M. Moudgill, J. Derby, E. Hokenek, D. Meltzer, U. Shvadron, and M. Ware, "Trends in compilable DSP Architecture," in *Proc. Workshop on SiGNAL Processing Systems (SiPS)*, 2000, pp. 181-199.

[3] H. M. Shao and I. S. Reed, "On the VLSI Design of a Pipeline Reed-Solomon Decoder Using Systolic Arrays," *IEEE Trans. Comput.*, vol. 37, no. 10, pp. 1273-1280, Oct. 1988.

[4] K. Iwamura, Y. Dohi, and H. Imai, "A Design of Reed-Solomon Decoder with Systolic-Array Structure," *IEEE Trans. Comput.*, vol. 44, no. 1, pp. 118-122, Jan. 1995.

[5] 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. Comput.*, vol. C-34, no. 5, pp. 393-403, May. 1985.

[6] 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. Circuit Syst. Video Technol.*, vol. 7, no. 6, pp. 864-871, Dec. 1997.

[7] H. H. Lee, M. L. Yu and L. Song, "VLSI Architecture for Decoding of Reed-Solomon Decoder Architectures," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS' 2000)*, vol. 5, May 2000, pp. 705-708.

[8] J. Sankaran, "Reed Solomon Decoder: TMS320C64x Implementation," Texas Instruments Inc., Dallas, TX, *Appl. Rep.* SPRA686, Dec. 2000.

[9] D. Taipale, I. E. Scheiwe and T. M. Redheendran, "Reed-Solomon Decoding on the StarCore Processor," Motorola Semiconductors Inc., Denver, CO, *Appl. Rep.* AN1841/D, May. 2000.

[10] Jae S. Lee and Myung H. Sunwoo, "Design of DSP instructions and their hardware architecture for a Reed-Solomon CODEC, " to appear in *Proc. IEEE Workshop Signal Processing Systems*, Sep. 2002

[11] 선우 명훈, 이 재성, "리드-솔로몬 부호화 및 복호화를 위한 프로그램머블 프로세서의 유한체 연산기 회로 및 연산방법" 특허 출원 번호 제 10-2001-0022427, 2001년 4월

[12] I. S. Reed and X. Chen, *Error-control Coding for Data Networks*, Norwell, Mass: Kluwer Academic Publishers Group, 1999.

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

[14] M. Bossert, *Channel Coding for Telecommunications.* New York, NY: John Wiley & Sons, Ltd., 1999.

[15] S. B. Wicker and V. K. Bhargava, *Reed-Solomon Codes and Their Applications.* New York, NY: IEEE Press, 1994.

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

[17] SC140 DSP Core Reference Manual, Motorola Semiconductors Inc., Denver, CO, 2000.

[18] Texas Instruments, Inc. (2000) C6000 Benchmark [Online]. Available: http:// dspvillage.ti.com

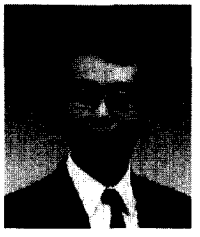
Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr

이 재 성(Jae-Sung Lee) 정회원

1999년 2월 : 아주대학교 전자공학부 학사
2001년 2월 : 아주대학교 전자공학과 석사
2001년 2월 ~ 현재 : 한국전자통신연구원(ETRI) 연구원

<주관심분야> VLSI Architecutre, 병렬처리 프로세서 및 DSP 칩 설계, Protocol Processing

선 우 명 훈(Myung-Hoon Sunwoo) 정회원



1980년 2월 : 서강대학교 전자공학
1982년 2월 : 한국과학기술원 전기 및 전자공학 석사
1982년 ~ 1985년 : 한국전자통신연구원(ETRI) 연구원
1985년 ~ 1990년 : Univ. of Texas at Austin 전기 및 컴퓨터 공학 박사
1990년 ~ 1992년 : 미국 Motorola, DSP Chip Division
2001년 ~ 현재 : IEEE Senior Member
2003년: IEEE workshop on signal Processing Systems, Technical Chair
1992년 ~ 현재 : 아주대학교 전자공학부 교수

<주관심분야> VLSI 및 SoC Architecture, 멀티미디어 통신용 DSP 칩 및 ASIC 설계

Copyright (C) 2002 Nuri Media Co., Ltd. www.dbpia.co.kr