논문 2006-43SD-11-12

Multi-band OFDM 시스템용 고속 연판정 비터비 디코더의 효율적인 하드웨어 구조 설계에 관한 연구

(A study on the Cost-effective Architecture Design of High-speed Soft-decision Viterbi Decoder for Multi-band OFDM Systems)

이 성 주*

(Seongjoo Lee)

요 약

본 논문에서는 Multi-band OFDM(MB-OFDM) 시스템에 적합한 고속 연판정 비터비 디코더의 효율적인 하드웨어 구조에 대해서 제시한다. MB-OFDM 시스템은 최대 480Mbps의 데이터 속도를 처리해야 하고 시스템 클럭으로 528MHz가 제공되기 때문에, 설계의 신뢰도를 향상시키기 위해 병렬처리 구조를 사용한다. 따라서, 비터비 디코더도 여러 개의 데이터를 동시에 처리하는 병렬처리 구조를 지원해야 하며, 또한 고속의 데이터를 처리하기 위한 하드웨어 구조를 사용해야 한다. 본 논문에서는 4-way 병렬처리에 적합하면서도 동시에 하드웨어 부담을 최소화할 수 있는 비터비 디코더의 하드웨어 구조를 제시한다. 이를 위해, 비터비 디코더의 핵심 기능블록이라 할 수 있는 ACS의 다양한 구조를 비교 및 분석하고 하드웨어와 동작속도 측면에서 가장 적합한 구조를 찾아내도록 한다. 최적의 하드웨어 구조로 설계된 비터비 디코더는 Verilog HDL로 설계 및 검증되었으며, 하드웨어 복잡도 및 동작속도 측정을 위해 TSMC 0.13um 공정으로 합성되었다. 합성결과, 제시된 구조는 약 280K 게이트로 구성되었으며 MB-OFDM 시스템이 요구하는 동작 주파수내에서 동작함을 확인하였다.

Abstract

In this paper, we present a cost-effective architecture of high-speed soft-decision Viterbi decoder for Multi-band OFDM(MB-OFDM) systems. In the design of modem for MB-OFDM systems, a parallel processing architecture is generally used for the reliable hardware implementation, because the systems should support a very high-speed data rate of at most 480Mbps. A Viterbi decoder also should be designed by using a parallel processing structure and support a very high-speed data rate. Therefore, we present a optimized hardware architecture for 4-way parallel processing Viterbi decoder in this paper. In order to optimize the hardware of Viterbi decoder, we compare and analyze various ACS architectures and find the optimal one among them with respect to hardware complexity and operating frequency. The Viterbi decoder with a optimal hardware architecture is designed and verified by using Verilog HDL, and synthesized into gate-level circuits with TSMC 0.13um library. In the synthesis results, we find that the Viterbi decoder contains about 280K gates and works properly at the speed required in MB-OFDM systems.

Keywords: Viterbi decoder, MB-OFDM systems, parallel processing, architecture design

I. 서 론

직교 주파수분할 다중접속(OFDM, Orthogonal Frequency Division Multiplexing) 방식은 다중경로 페이딩(multi-path fading) 및 협대역 간섭(narrow-band interference)에서 우수한 성능을 보이기 때문에

* 정회원, 세종대학교 정보통신공학과
(Dept. of Information and Communication Eng. Sejong Univ.)
※ 본 연구는 21세기 프론티어 연구개발 사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크 원천기반기술개발사업의 지원에 의한 것이고, CAD Tool은 IDEC으로부터 지원 받았음.
접수일자: 2006년8월1일, 수정완료일: 2006년10월25일

통신 및 방송 시스템에서 매우 유용한 기술로 부각되고 있다[1-4]. 다중 주파수 밴드(Multi-band) OFDM (MB-OFDM) 기술은 이러한 OFDM 기술의 장점에 다중 주파수 밴드를 이용함으로써 발생하는 이점을 가미한 방식으로서, IEEE802.15.3a 초광대역(UWB, Ultra-WideBand) 통신 시스템의 표준중 하나로 제안되고 있다[5-6].

MB-OFDM 방식을 사용하는 시스템은 일반적으로 넓은 주파수 대역을 차지하면서 근거리에서 수십 Mbps에서 수백 Mbps까지의 고속 데이터 전송을 목적으로 한다. 따라서, MB-OFDM 시스템용 모뎀은 고속의 데이터 처리를 위해 매우 빠른 속도로 동작해야하기 때문에, MB-OFDM 시스템용 모뎀의 하드웨어 설계에서는 여러 가지 다양한 문제점들이 고려되어져야 한다. IEEE802.15.3a 표준으로 제안된 MB-OFDM 시스템에서는 최대 480Mbps의 데이터를 처리해야 하고 파일롯 및 가드구간(guard interval)을 고려하여 최종 디지털 모뎀의 출력이 528MHz의 속도를 가지므로, 디지털 모뎀의 클럭 주파수가 매우 높아져 신뢰성이 높은 모뎀을 설계하는 것이 매우 어렵다.

높은 클럭 주파수를 사용하게 되면 플립플롭(Flip-flop)과 플립플롭 사이의 타이밍 마진(timing margin)이 매우 부족해지기 때문에, 레이아웃(layout) 설계에서 타이밍 시뮬레이션(timing simulation)을 통과하기가 매우 힘들다. 따라서, 하드웨어 설계의 신뢰도를 높이기 위해서는 클럭 주파수를 낮춰야 하는데, 이를 위해서 가장 일반적으로 사용하는 방법 중의 하나가 병렬처리 구조이다. 병렬처리 구조는 하나의 데이터 경로를 여러 개의 데이터 경로로 나누어 동시에 데이터를 처리함으로써 클럭 속도를 데이터 경로의 배수만큼 줄일 수 있다. 그러나 병렬처리 방식으로 회로를 설계하는 경우 각 기능블록에 대한 제어 회로 및 데이터 경로가 매우 복잡해지기 때문에, 최적의 하드웨어 구조를 찾아내기 위해서는 매우 많은 노력과 시간을 투자해야 한다. 특히, 비터비 디코더에 병렬처리 방식이 사용되는 경우 구조에 따라 하드웨어 복잡도가 기하급수적으로 증가할 수 있기 때문에, 적절한 하드웨어를 사용하면서도 동작 속도를 만족시킬 수 있는 하드웨어 구조를 개발하는 것이 무엇보다도 중요한 문제가 된다[7-10]. 따라서 본 논문에서는 고속으로 동작하는 병렬처리 비터비 디코더를 위해 매우 복잡한 하드웨어의 사용을 방지하면서도 주어진 동작속도를 만족시킬 수 있는 최적 하드웨어 구조를 제시하고자 한다. 이를 위해, 다양한 연산처리 구조들을 주어진 설계공정(library)을 적용하여 동작 속도 및 하드웨어 복잡도 측면에서 비교 및 분석하고, 동작속도를 만족하면서도 하드웨어 복잡도가 가장 적은 최적 구조를 선택한다.

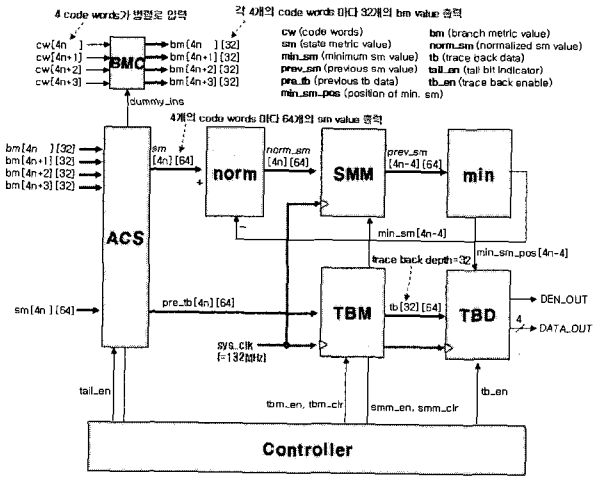

그림 1. 병렬처리 비터비 디코더의 블록도

Fig. 1. Block diagram of Parallel Processing Viterbi Decoder.

본 논문은 다음과 같이 구성된다. 먼저 Ⅱ장에서는 4-way 병렬처리 비터비 디코더의 구성에 대해서 언급하고, Ⅲ장에서는 각 핵심 기능블록에 대한 구조 설계 방법을 제시한다. Ⅳ장에서는 설계 및 검증 결과를 분석하고, 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 4-way 병렬처리 비터비 디코더

MB-OFDM용 모뎀에서는 시스템 클럭 속도를 줄이고 집적회로(IC)로 설계 시 신뢰도를 높이기 위해, 4-way 병렬처리 구조를 사용한다. 병렬처리 구조를 사용하지 않는 경우, 아날로그 회로와의 연결을 위해 528MHz의 클럭을 사용해야하지만, 4-way 병렬처리 구조를 사용하는 경우, 시스템 클럭이 132MHz로 본래 속도의 1/4로 줄어든다. 그러나, 병렬처리를 위해 데이터는 동시에 4개씩 처리가 되고, 따라서, 비터비 디코더로 입력되는 데이터는 3 비트로 연판정된 4개의 코드워드(code words)가 132MHz의 속도로 동시에 입력된다. 비터비 디코더의 입력부터 출력까지의 모든 데이터 처리는 4개 단위로 수행되며, ACS(Add, Compare, and Selection)는 물론 역추적(trace-back), 복호(decoding) 과정도 4단(step)씩 수행되어 진다. 그림 1은 4-way 병렬처리 비터비 디코더의 블록도를 보여준다.

그림 1에서 보면, 비터비 디코더는 크게 BMC (Branch Metric Calculator), ACS(Add, Compare, and Selection), SMM(State Metric Memory), TBM(Trace-back Memory), TBD(Trace-back and Decoder), 그리고 제어부(Controller)로 구성된다. BMC는 가지 메트릭 값(BM, Branch Metric)을 계산하는 블록이고, ACS는 이전 상태 메트릭 값(SM, State Metric)에서 다음 상태 메트릭 값의 최적 경로를 계산하는 모듈이다. 그리고, SMM은 SM을 저장하는 메모리 모듈이고, TBM은 역추적 경로를 저장하기 위한 메모리 모듈이다. TBD는 TBM으로부터 역추적 경로 값을 받아서 역추적을 수행하고 최종적으로 데이터를 복호하는 블록이고, 마지막으로 제어부는 각 기능블록의 타이밍 및 동작을 제어한다. 이 중에서 ACS와 역추적 회로(TBM과 TBD)가 비터비 디코더의 동작속도를 결정하면서 동시에 비터비 디코더의 하드웨어 복잡도를 결정짓는 주요 기능 블록들이다. 따라서, ACS와 역추적 회로에 대한 최적의 하드웨어 구조를 설정하는 것이 비터비 디코더의 최적화를 결정짓는 가장 중요한 핵심 설계사항이라고 볼 수 있다.

III. ACS 연산과 역추적 회로의 최적 하드웨어 구조 설계

1. ACS의 최적 하드웨어 구조 설계

비터비 디코더의 동작 속도를 결정짓는 가장 중요한 블록이 ACS이다. ACS는 이전 SM으로부터 다음 SM을 결정하는 연산을 수행하므로, 파이프 라인(pipeline) 기법을 사용할 수 없고 반드시 들어오는 데이터를 1번의 클럭 내에서 처리해야 하는 부담이 있다. 4-way 병렬처리 비터비 디코더는 4개의 코드워드를 매 클럭마다 동시에 입력으로 받기 때문에, 비터비에 사용되는 ACS도 4개의 코드워드를 동시에 처리할 수 있는 구조로 설계되어야 한다. ACS의 가장 기본적인 구조는 radix-2 구조로 2개의 이전 SM에 대해서 다음의 SM을 결정하는 구조이다. 그림 2는 radix-2의 ACS구조를 보여준다.

그림에서 알 수 있듯이, 각 SM으로 2개의 경로가 도달될 수 있으며, 그 중에서 메트릭이 작은 것을 선택하는 나비구조를 가진다. 그러나 이 구조는 1개의 코드워드를 처리할 수 있는 형태이므로, 4개의 코드워드가 동시에 입력되는 경우, 그림 3과 같이 radix-2^4의 구조를

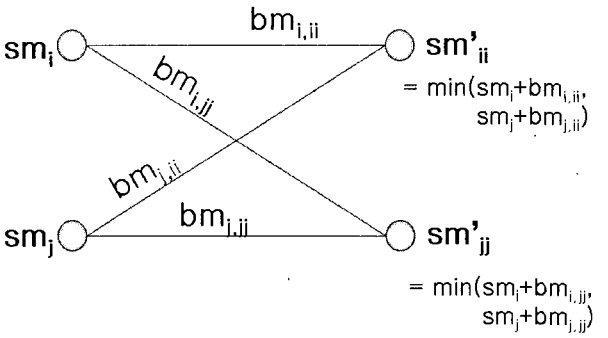

그림 2. radix-2 ACS 구조
Fig. 2. radix-2 ACS structure.

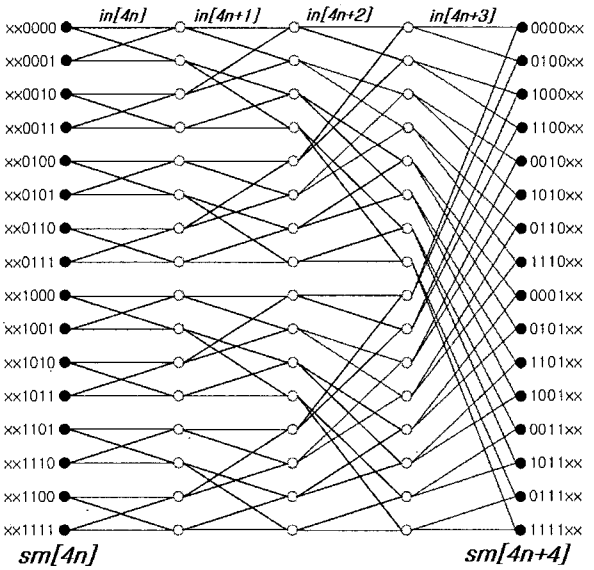

그림 3. radix-2^4 ACS 구조
Fig. 3. radix-2^4 ACS structure.

가져야 한다. MB-OFDM 시스템에서는 K=7인 길쌈부호화기(convolutional encoder)를 사용하므로, 64개의 SM 값이 존재할 수 있는데, 그림 2는 그 중에서 16개의 상태를 하나의 그룹으로 묶어서 보여준다. 따라서 이러한 ACS 그룹이 총 4개가 존재하게 된다.

그림 3을 살펴보면, radix-2^4 ACS 구조는 radix-2 ACS 4단이 직렬로 연결되어 있으며, 4단의 ACS를 통과해야 SM을 레지스터에 저장할 수 있다. radix-2^4 ACS 구조는 매우 간단한 구조로 되어 있어 하드웨어 부담이 매우 적은 모습을 하고 있다. 그러나 다음 SM을 구하기 위해서 4번의 덧셈과정과 4번의 비교 및 선택과정이 요구되므로, 다음 SM이 결정되는데 많은 시간이 필요하게 된다.

ACS 연산속도를 높이기 위해서는 덧셈 과정를 줄이는 방법이 필요한데, 그러기 위해서는 동시에 여러 개의 입력을 받아서 처리하는 radix-N (N=2^n, n은 동시

입력의 수)의 형태를 가져야 한다. radix-N의 형태를 가지게 되면, radix-2를 직렬로 연결하는 구조에 비해 덧셈 과정이 줄어들게 되지만, 하나의 SM에서 갈 수 있는 경로의 수는 N으로 증가되기 때문에, 하드웨어 부담이 커지게 된다. 본 논문에서는 4개의 입력이 동시에 들어오므로, 최대 radix-16까지 가능하지만, 이 경우, 하드웨어 부담이 급격하게 증가하여 전력소모 및 하드웨어 구현 측면에서 문제점을 일으킬 수 있다. 따라서, radix-4나 radix-8을 직렬로 연결하는 구조를 고려하는 것이 하드웨어 측면에서 더 효율적이라고 본다. 그림 4는 radix-4^2 ACS 구조를 보여준다.

그림에서 알 수 있듯이, radix-4^2 ACS 구조는 radix-4 ACS가 직렬로 2단 연결되어 있다. radix-4 ACS는 2개의 코드워드를 받아서 처리하며, 이전 SM과 다음 SM이 각각 4개씩 쌍으로 존재한다. 각 SM에서 갈 수 있는 경로는 2개의 입력에 대해 수행하므로 총 4개가 존재하고, 전체적으로 볼 때 ACS 한 쌍에 대해 총 16개의 경로가 존재한다. 따라서, 하드웨어 부담이 radix-2에 비해 약 4배 이상 증가된다고 볼 수 있다. SM을 계산하기 위한 덧셈 과정은 radix-4^2 ACS 구조에서 총 2번으로 줄어들었지만, 비교 및 선택과정의 경우, 1번 ACS를 수행하는데 있어서 비교 대상이 되는 경로의 수가 2에서 4로 증가하였기 때문에, 단 수가 줄어들어도 radix-2^4 ACS 구조와 동일한 회수의 비교동작이 필요하다. 결과적으로 radix-4^2 ACS 구조의 경우, radix-2^4 ACS 구조보다 덧셈과정의 단축으로 인한 동작속도 향상 이득을 가질 수 있다.

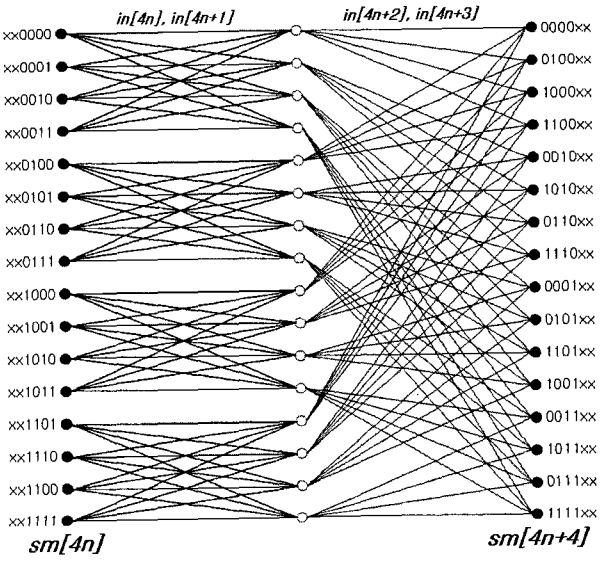

그림 4. radix-4^2 ACS 구조
Fig. 4. radix-4^2 ACS structure.

연산처리 지연시간을 더욱 줄이기 위해서는 그림 5와 같이 radix-16의 구조를 가져야 한다. 이 경우, 각 SM당 존재할 수 있는 경로의 수는 총 16개가 되고, 16개의 SM이 하나의 ACS에 존재하므로, 전체 경로의 수는 256개가 되어, radix-2에 비해 64배 증가하게 된다. 그러나 다음 SM을 계산하기 위한 덧셈과정은 radix-2^4에 비해 4번에서 1번으로 감소되어 연산처리 시간은 매우 많이 줄어든다.

각 ACS 구조에 대한 하드웨어 복잡도를 비교하기 위해, radix-N^m ACS 구조에 대한 하드웨어 복잡도를 정형화된 수식으로 표현할 필요가 있다. 우선, ACS를 구현하기 위해서는 이전 SM에서 다음 SM으로 진행되는 경로의 BM 값을 구해야 한다. BM은 ACS를 수행하기 이전에 입력되는 코드워드로부터 미리 생성해 낼 수 있으므로, ACS의 동작속도에는 영향을 주지 않는다. 그러나, 입력되는 코드워드와 각각의 경로가 가지는 코드워드 사이의 유클리디안 거리(Euclidean distance) 값에 대한 계산 및 해당 결과를 저장하기 위한 하드웨어는 BM 값을 구하는데 필요한 부분이므로 복잡도 계산에 포함되어야 한다. 우선, MB-OFDM 시스템의 경우, 1/3 코드 율을 사용하므로, 1개의 입력에 대해 총 8가지의 코드워드가 가능하게 된다. 따라서 radix-N 의 경우, BM 계산 및 저장을 위해 필요한 하드웨어 복잡도 CBM|radix-N는

$$C_{BM|radix-N} = 8^{\log_2 N}(C_{ED|radix-N} + C_{REG|radix-N}) = N^3(C_{ED|radix-N} + C_{REG|radix-N}) \quad (1)$$

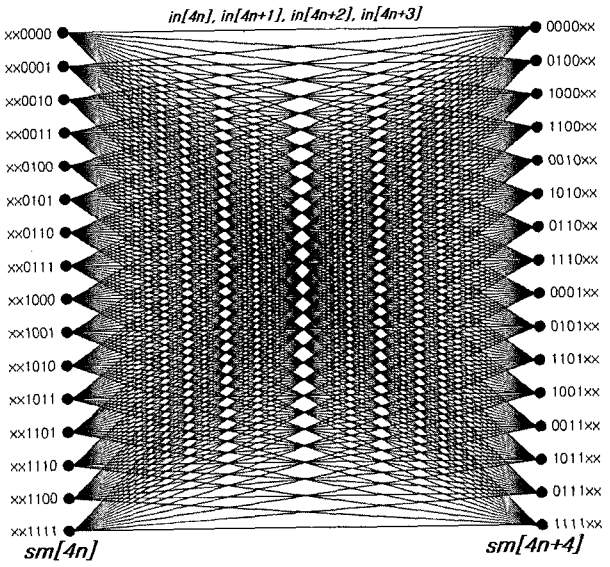

그림 5. radix-16 ACS 구조
Fig. 5. radix-16 ACS structure.

와 같이 주어진다. 여기서 $C_{ED}|_{\text{radix-}N}$와 $C_{REG}|_{\text{radix-}N}$는 각각 radix-$N$ 구조에서 유클리디안 거리를 계산하는데 필요한 하드웨어 복잡도와 그 값을 저장하는데 필요한 레지스터의 하드웨어 복잡도를 의미한다. 따라서, radix-N^m ACS 구조에 대한 C_{BM}은 식(2)와 같이 정의할 수 있다.

$$C_{BM} = m \times C_{BM}|_{\text{radix-}N} \tag{2}$$

다음으로 ACS 동작을 수행하는데 필요한 하드웨어 복잡도를 계산하기 위해서는 우선적으로 총 경로의 수를 산출해야 한다. S개의 SM에 대해서 radix-N을 사용하는 경우, S/N개의 ACS 쌍이 존재하게 되고, 1개의 ACS 쌍에는 N^2개의 경로가 존재하므로(1개의 SM당 N개의 경로가 존재하므로), radix-N에 대한 총 경로의 수 $N_P|_{\text{radix-}N}$는

$$N_P|_{\text{radix-}N} = S \times N \tag{3}$$

와 같이 정의될 수 있다. 따라서, radix-N^m ACS 구조의 경우, 전체 경로의 수는 식(4)와 같이 표현된다.

$$N_P = m \times N_P|_{\text{radix-}N} \tag{4}$$

각 경로당 덧셈이 필요하고, 각 경로들을 비교하기 위해 N_P-1개의 비교기가 필요하므로 ACS에 필요한 하드웨어 복잡도 C_{ACS}는

$$C_{ACS} = N_P \times C_{ADD}|_{\text{radix-}N} + (N_P - 1) \times C_{COMP}|_{\text{radix-}N} \tag{5}$$

와 같이 정의될 수 있다. 여기서, $C_{ADD}|_{\text{radix-}N}$와 $C_{COMP}|_{\text{radix-}N}$는 각각 radix-$N$ 구조에서 덧셈기와 비교기의 하드웨어 복잡도이다. 식 (1)부터 식(5)까지의 관계를 통해 radix-N^m ACS 구조에서 전체 ACS 연산에 필요한 하드웨어 복잡도는 식(6)과 같이 정의된다.

$$C = mSNC_{ADD}|_{\text{radix-}N} + (mSN-1)C_{COMP}|_{\text{radix-}N} + mN^3(C_{ED}|_{\text{radix-}N} + C_{REG}|_{\text{radix-}N}) \tag{6}$$

동작속도는 하나의 SM에서 다음 SM을 계산하는데 필요한 덧셈의 회수와 비교 회수를 고려하여 연산지연시간을 구하면 쉽게 정의할 수 있다. radix-N^m ACS 구조에서 덧셈 회수는 N과 무관하게 m이 되고, 비교 회수는 $m\log_2 N$으로 정의할 수 있으므로, 플립플롭의 출력지연시간 및 셋업시간(setup time)을 고려하면, ACS를 수행하는데 필요한 연산시간은

$$\begin{aligned} T &= T_{F/F} + mT_{ADD}|_N + mT_{COMP}|_N \log_2 N \\ &= T_{F/F} + m(T_{ADD}|_N + T_{COMP}|_N \log_2 N) \end{aligned} \tag{7}$$

와 같이 정의할 수 있다. 여기서 $T_{ADD}|_N$와 $T_{COMP}|_N$는 각각 radix-N에서 덧셈과 비교를 하는데 소요되는 연산시간을 의미하고, $T_{F/F}$는 플립플롭의 셋업시간과 출력지연시간을 합한 값이다. 따라서, 최대 동작속도는

$$F_{MAX} = 1/T \tag{8}$$

가 된다.

2. 역추적 회로의 최적 하드웨어 구조 설계

역추적 과정은 매 ACS 연산을 통해 계산되고 저장된 각 SM의 최적 경로 값을 메모리로부터 불러들여, 생존 경로(survival path)를 찾아내고 찾아낸 생존 경로로부터 데이터를 복호(decoding)하는 과정으로 이루어진다. 역추적을 수행하기 위해서는 SM이 진행되는 매 경로를 저장해야 하는데, 최적 경로를 어떤 방식으로 저장하고 읽어 들이냐에 따라 필요한 메모리의 크기가 달라진다. 역추적 메모리는 크게 2가지 형태로 구현될 수 있는데, 하나는 메모리를 사용한 구조이고 다른 하나는 레지스터를 사용한 구조이다. 메모리를 사용한 구조에서는 동시에 여러 경로를 읽어 들일 수 없기 때문에, 최적 경로를 여러 개의 메모리로 나누어 저장하고 역추적 과정을 여러 단계로 나누어 각 메모리가 동시에 접속(access)되지 않도록 한 방식이다[11]. 이 방식의 경우, 역추적 과정에 소모되는 클럭 지연(clock latency)이 크기 때문에, 역추적 길이(L, 여러 경쟁 경로들이 하나의 상태로 수렴할 수 있도록 충분히 길게 잡는다. L값은 사용되는 시스템과 통신환경에 따라 달라지는데, MB-OFDM에서는 42내지 48 정도를 사용한다.)보다 2배 이상의 경로를 저장해야 한다. 레지스터를 이용한 방식은 메모리 방식과 달리, 최적 경로를 레지스터에 저장하므로, 동시에 여러 경로를 읽어 들일 수 있어서 역추적 과정에서 소모되는 클럭 지연 시간이 작기 때문에, L보다 약간 긴 정도의 저장 공간을 필요로 한다. 그러나 매 경로를 레지스터로 저장하므로 단위 기억소자의 크기가 커서 메모리 방식보다 하드웨어 복잡도가 커질 수 있다[12].

메모리 방식과 레지스터 방식은 각각의 장단점을 가지고 있는데, MB-OFDM의 경우 동시에 4개의 경로가 저장되므로 매 클럭 저장되는 최적 경로 값이 256비트(=64 상태×4비트)가 되고 16 경로 단위로 메모리를 구

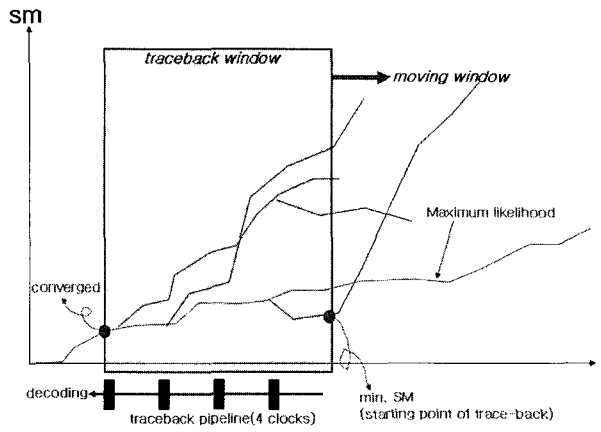

그림 6. 역추적 과정
Fig. 6. Trace-back processing.

분했을 때 행 주소(row address)가 4밖에 되지 않을 뿐만 아니라, 8개의 메모리로 분산해서 사용해야 하므로 (버스 폭이 커서 접속시간이 길어지는 경우, 더 분리될 수도 있음), 레이아웃(layout) 설계에서 레지스터 방식보다 하드웨어 복잡도 측면에서 큰 이점을 가지기가 힘들다. 특히, 역추적 과정에서 소모되는 지연시간이 레지스터보다 크기 때문에, 하드웨어 측면에서 큰 이득이 없는 경우 메모리 방식을 사용할 특별한 이유가 없다. 따라서, 본 논문에서는 매 SM 경로 값을 레지스터에 저장하고 이를 이용하여 역추적 과정을 수행하는 방법을 사용한다.

그림 6은 본 논문에서 사용한 역추적 방법을 설명한다. 우선 입력되는 코드워드를 통해 ACS를 수행하고 수행된 결과로부터 SM의 경로를 레지스터에 저장한다. $L/4$ 만큼 경로가 저장되면(경로는 한번에 4단씩 저장되므로), 가장 최소 값을 가지는 SM으로부터 역추적 과정을 수행하여, $L/4$ 만큼의 역추적 과정 이후에 데이터를 복호하게 된다. 역추적을 위해 한번에 읽어들일 수 있는 경로 값은 설계하고자 하는 공정의 속도에 좌우되는데, 본 논문에서는 0.13um 공정을 사용하였기 때문에, 최대 3단(=3×4 bit=12 depth)까지 적용한다. 이 경우, 총 4번의 역추적 과정(L이 48이므로)이 필요하고, 이 역추적 과정동안, SM이 이전의 최적 경로 값을 덮어쓰지 않도록 추가의 레지스터를 더 사용한다. 레지스터로 구성된 역추적 메모리는 $L/4$에 역추적 과정 지연을 위한 마진 4를 더해 $L/4+4$의 깊이(depth)를 가진다. 또한, 비터비 디코더 연산이 수행되는 동안 $L/4+4$의 레지스터 단을 순환하면서(cyclic) SM 경로 값을 저장하고 읽어들인다. 그림 7은 SM 경로가 어떻게 저장되고 읽혀지는가를 보여준다.

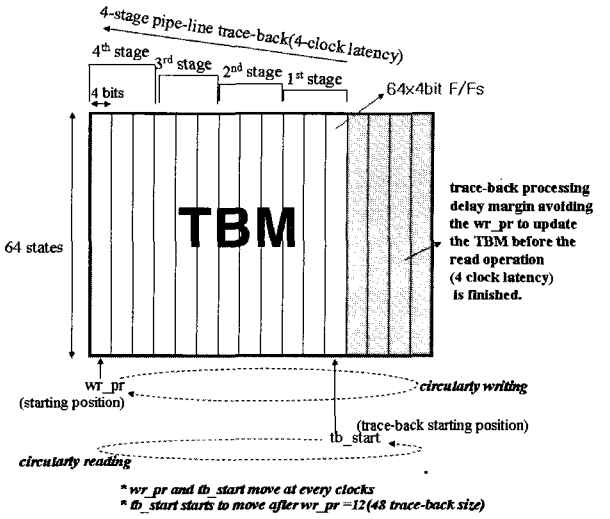

그림 7. 역추적 레지스터의 쓰기 및 읽기 전략
Fig. 7. Write and read operations during the trace-back processing.

역추적에 필요한 하드웨어 복잡도는 다음과 같이 결정된다. 우선, ACS를 위해 64개의 SM 값을 저장해야 하고, 역추적 과정을 위해 $L/4+4$ 단의 레지스터 파일이 필요하므로 역추적을 위해 사용된 하드웨어 복잡도는 식(9)와 같이 구할 수 있다.

$$C_{TB} = 64\times C_{SM} + (L/4+4)\times 64\times 4\times C_{F/F} = 64\times(SN + L + 16)\times C_{F/F} \quad (9)$$

식(9)에서 SN과 $C_{F/F}$은 각각 SM을 표현하기 위한 비트수와 1비트 플립플롭의 하드웨어 복잡도를 의미한다.

Ⅳ. HDL 설계 및 성능 비교

비터비 디코더의 성능은 하드웨어 설계가 복호 알고리듬을 변형시키지 않고 이루어지기 때문에, BER 성능보다는 하드웨어 복잡도와 동작 속도 측면에서 비교되어 진다. 따라서, 어떤 구조가 최소한의 하드웨어 복잡도를 사용하면서도 원하는 동작 속도를 만족시키는 가가고속 비터비 디코더의 설계 이슈가 된다.

본 논문에는 앞 장에서 정형화 시킨 하드웨어 복잡도 및 동작속도를 이용하여 최적의 ACS 구조를 선택하는 방식을 사용한다. 이것은 향후, 각 구조에 대한 성능 비교를 위해 불필요한 HDL 설계 시간을 줄이고 적절한 하드웨어 구조를 선택하는데 용이하도록 하기 위함이다. 이와 더불어, 추정치로 근사화 된 하드웨어 복잡도와 동작속도가 실제 합성결과와 얼마나 유사한지를 비교하기 위해, Ⅲ장에서 제시된 모든 ACS 구조들을

표 1. 하드웨어 복잡도 및 동작속도 추정을 위한 파라미터 값

Table 1. Parameters for estimation of hardware complexity and speed of Viterbi Decoder.

파라미터	radix-2^4	radix-4^2	radix-16
S	64	64	64
N	2	4	16
m	4	2	1
C_{ADD}	200	300	400
C_{COMP}	80	120	160
C_{ED}	70	100	200
C_{REG}	30	40	60
S_N	8	8	8
L	12	12	12
$C_{F/F}$	10	10	10
T_{ADD}	2.0	2.8	3.5
T_{COMP}	0.2	0.25	0.3
$T_{F/F}$	0.7	0.7	0.7

HDL로 설계한 후 게이트 수준으로 합성하였다. 우선 본 논문에서 정의한 수식을 이용하여 각 ACS 구조에 대한 하드웨어 복잡도 및 동작속도를 계산하기 위해, 표 1과 같이 파라미터를 설정하였다. 표 1에서 설정된 값들은 단위 소자들을 0.13um 공정으로 합성한 결과를 이용하여 얻어진 결과들이다.

그림 8은 각 ACS 구조에 대해서 표 1을 적용하여 구한 추정치와 실제로 TSMC 0.13um 공정을 이용하여 합성한 결과로 얻어진 실측치를 비교한 그림이다. 그림 8에서 알 수 있듯이, radix-2^4에서 radix-16까지 비교했을 때, 하드웨어 복잡도는 radix-2^4가 가장 우수하고, 동작속도는 radix-16이 가장 좋은 성능을 보였다. 그런데, radix-2^4는 하드웨어 부담이 가장 적지만 MB-OFDM 시스템에서의 요구 동작 주파수 132MHz를 만족하지 못하므로 사용할 수 없고, radix-16은 요구 동작 주파수를 만족하지만 하드웨어 복잡도가 다른 구조와 달리 하드웨어로 구현하기에 적절하지 못할 만큼 커지는 문제가 있다. 그러나 radix-4^2의 경우, MB-OFDM 시스템에서의 요구 동작 주파수를 만족하면서도(157MHz) radix-2^4에 비해 하드웨어 복잡도가 심각하게 증가하지 않기 때문에(280K 게이트) MB-OFDM 시스템에 매우 적합한 구조라고 볼 수 있다. 또한, 그림 8에서 추정된 값과 실측치를 비교해보면, 하드웨어 복잡도의 경우 radix-2^4와 radix-4^2은 유사한 형태를 보였지만, radix-16에서는 차이가 매우 심하게 벌어짐을 알 수 있다. 이것은 하드웨어 복잡도가 200만 게이트 이상으로 커지는 경우 최적화 과정에서 최적화를 위해 고려해야할 변수가 막대하게 증가함으로 인해, CAD 툴의 합성 성능이 떨어져서 발생한 것으로 판단된다. 또한, 논리합성 당시 합성에 대한 제약조건(constraint)을 어떻게 주었느냐에 따라 그 결과가 매우 다르게 나올 수 있다. 따라서, 본 논문에서 얻어진 결과는 하드웨어 구조에 따른 속도 및 면적의 경향을 파악하는데 의미를 부여할 수 있고, 앞에서 도출한 추정식에 대한 신뢰성을 판단하는 참고자료로 사용하는 것이 바람직하다고 판단된다.

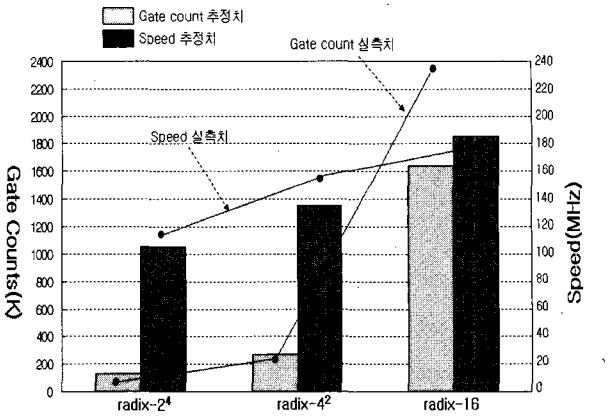

그림 8. ACS 구조에 따른 비터비 디코더의 하드웨어 복잡도 및 속도 비교

Fig. 8. Comparison of ACS structures with respect to hardware costs and speed.

V. 결 론

본 논문에서는 MB-OFDM 시스템용 고속 연판정 비터비 디코더에 적합한 최적의 하드웨어 구조에 대해서 제시하였다. 비터비 디코더에서 동작속도와 하드웨어 복잡도를 결정짓는 핵심 블록인 ACS의 최적 구조를 찾아내기 위해, 본 논문에서는 여러 가지 ACS 구조를 하드웨어 복잡도 및 동작속도 측면에서 비교 분석하였으며, 이를 정형화된 수식으로 표현하였다. 각 ACS 구조를 가지는 비터비 디코더들은 모두 HDL로 설계 및 TSMC 0.13um 공정으로 합성되었다. 성능 비교 결과, 측정치와 추정치 모두에서 radix-4^2 ACS 구조가 원하는 동작주파수를 만족하면서도(157MHz) 하드웨어 부담을 최소화할 수 있는 구조(280K 게이트)로 분석되었다. 따라서, radix-4^2 ACS 구조는 하드웨어 부담이 무엇보다도 중요한 문제로 부각되는 MB-OFDM 시스템에서 하드웨어 부담을 크게 증가시

키지 않으면서도 고속의 데이터 처리를 할 수 있는 유용한 구조로 판단된다.

참 고 문 헌

[1] Marc Engles, "Wireless OFDM Systems," Kluwer Academic Publishers, 2002.

[2] J. Heiskala and J. Terry, "OFDM Wireless LANS: A Theoretical and Practical Guide," Sams Publishing, 2001.

[3] O. Edfors, M. Sandell, J. J. Van De Beek, S. K. Wilson and P. O. Borjession, "OFDM channel estimation by Singular Value Decomposition," pp.923-927, in Proc. of IEEE Vehicular Technology Conference, Altlanta, USA, April 1996.

[4] S. Coleri, M. Ergen, A. Puri and A. Bahai, "A Study of Channel Estimation in OFDM Systems," pp.894-898, in Proc. of IEEE Vehicular Technology Conference, Vancouver, Canada, September 2002.

[5] A. Batra et al., "Multi-band OFDM Physical Layer Proposal for IEEE 802.15 Task Group 3a", IEEE P802.15-04/0493r1, Sept. 2004

[6] A. Batra, J. Balakrishnan, G.R. Aiello, J.R. Foerster, A. Dabak, "Design of a Multiband OFDM System for Realistic UWB Channel Environments", IEEE Transactions on Microwave Theory and Techniques, vol. 52, issue 9, part 1, pp.2123-2138, Sept. 2004

[7] Peter J. Black and Teresa H. Meng, "A 140-Mb/s, 32-State, Radix-4 Viterbi Decoder," IEEE Journal of Solid-State Circuits, Vol.27, No.12, pp.1877-1885, Dec. 1992.

[8] Fei Sun and Tong Zhang, "Parallel High-Throughput Limited Search Trellis Decoder VLSI Design," IEEE Transaction on Very Large Scale Integration Systems, Vol.13, No.9, pp.1013-1022, Sept. 2005.

[9] AnhDinhand Xiao Hu, "A Hardware-Efficient Technique to Implement a Trellis Code Modulation Decodr," IEEE Transaction on Very Large Scale Integeration Systems, Vol.13, No.6, pp745-750, June 2005.

[10] Mark Anders, Sanu Mathew, Ram Krishnamurthy, Shekhar Borkar, "A 64-state 2GHz 500Mbps 40mW Viterbi Accelerator in 90nm CMOS," pp.174-175, in Proc. of Symposium on VLSI Circuits, Honolulu, USA, June 2004.

[11] Gennady Feygin and P. G. Gulak, "Architectural Tradeoffs for Survivor Sequence Memory Management in Viterbi Decoder," IEEE Transactions on Communcations, Vol.41, No.3, pp.425-429, March 1993.

[12] Peter J. Black and Teresa H.Y.Meng, "Hybrid Survivor Path Architecture for Viterbi Decoders," pp.433-436, in Porc. of IEEE International Conference on Acoustics, Speech, and Signal Processing, Minneapolis, USA, April 1993.

저 자 소 개

이 성 주(정회원)

대한전자공학회 논문지
제43권 SD편 11호 참조