論文 96-21-8-21

다중의 Add-Compare-Select 모듈을 갖는 병렬 비터비 알고리즘의 메모리 관리 방법

正會員 지 현 순*, 박 동 선**, 송 상 섭***

A Memory Management Scheme for Parallel Viterbi Algorithm with Multiple Add-Compare-Select Modules

Hyun Soon Chi*, Dong-Sun Park**, Sang Seob Song*** *Regular Members*

요 약

본 논문에서는 병렬 비터비 복호기 구현을 위한 메모리 구조와 제어 방법을 제안한다. 비터비 복호기에서 사용되는 메모리는 경로값을 저장하기 위한 SMM(State Metric Memory)과 생존 경로 정보를 저장하기 위한 TBM(Traceback Memory)으로 구분되며 각 메모리의 구조는 복호기의 수행 속도와 하드웨어 복잡도에 영향을 주는 요소이다.

고속 수행을 위한 병렬 비터비 복호기는 다수의 ACS모듈로 구성되며 ACS모듈의 병렬화에 따라 TBM과 SMM은 각각 독립적인 소단위 모듈쌍으로 인터리빙된다. SMM은 병렬화된 ACS모듈이 동시에 경로값을 계산할 수 있도록 주소 발생기를 통해 제어되며 분할된 TBM은 각 모듈들에 생존 경로 정보를 동시에 저장할 수 있도록 고안된 비트 셔플러에 의해 제어된다.

ABSTRACT

In this paper, a memory organization and its control method are proposed for the implementation of parallel Viterbi decoders. The design is mainly focused on lowering the hardware complexity of a parallel Viterbi decoder which is to ruduce the decoding speed. The memories required in a Viterbi decoder are the SMM(State Metric Memory) and the TBM(Traceback Memory); the SMM for storing the path metrics of states and the TBM for storing the

*삼성 데이터 시스템
Samsung Data System
**전북대학교 정보통신공학과
Dept. of Information and Communication Engineering, Chonbuk National University
***전북대학교 전기·전자·제어공학부
School of Electrical Engineering, Chonbuk National University
論文番號:96069-0222
接受日字:1996年 2月 22日

survival path information. A general parallel Viterbi decoder for high data rate usually consists of multiple ACS (Add-Compare-Select) units and their corresponding memory modules. For parallel ACS units, SMMs and and TBMs are partitioned into smaller independent pairs of memory modules which are separately interleaved to provide the maximum processing speed. In this design SMMs are controlled with address generators which can simultaneously compute addresses of the new path metrics. A bit shuffle technique is employed to provide a parallel access to the TBMs to store the survivor path informations from multiple ACS modules.

Ⅰ. 서 론

길쌈 부호화된 신호의 복호 방식들중 1967년 비터비에 의해 제안된 비터비 복호 방식은 수신된 경로에 대한 최대우도열추정(Maximum likelihood sequence estimation) 알고리즘이다.[1] 비터비 복호 알고리즘은 초기에는 구속장이 K=6, 7 정도까지 개발되었으나 설계 기술과 반도체 기술의 발전에 힘입어 최근 구속장이 큰(K=9)것 까지도 개발 되고있다.[2] 길쌈 부호기에 비하여 복잡도가 매우 큰 비터비 복호기의 설계 기술은 이동 채널 환경등에서 고품질의 통신을 위하여 중요한 연구 분야로 인식되고 있으며 특히 고속 통신 시스템에 적용하기 위해서는 반드시 병렬 구현 방법이 고려되어야한다.

비터비 복호기는 BM(Branch Metric)부, ACS(Add-Compare-Select)부, TB(Traceback)등 주요 3개의 모듈로 구성된다.[3] ACS 모듈은 비터비 복호기의 심장부로서 많은 계산을 반복해야 하는 부분으로 비터비 복호기의 고속화에 걸림돌이 되는 소위 병목(bottleneck)에 해당된다. 이러한 병목 현상을 해결하기 위하여 다수의 ACS모듈을 채용한 병렬 경로 계산 방법들이 제안되고 있으며 병렬화의 정도는 구체적인 응용 분야에서 요구되는 동작 속도에 따라 달라진다. 예로서, Fettweis는 ACS모듈의 파이프 라인 설계를 제안하였으며 C. M. Rader는 SMM(State Metric Memory)을 단일 버퍼로 구현하고 수행 속도의 단축을 위하여 ACS모듈로 입력되는 경로값을 동시에 읽어 올 수 있도록 상태의 패리티에 따라 SMM을 분할하는 방법도 제시하였다.[4][5][6]

본 논문은 C. M. Rader에 의해 제안된 패리티에 의해 분할되는 SMM의 제어 방법을 제시하고 더 나아가서 비터비 복호기를 병렬로 구현하는 경우 SMM의 분할 방법과 그 제어 방법을 제안한다. 제안된 SMM 분할 방법은 ACS모듈의 병렬화에 따라 경로값을 동시에 처리할 수 있도록 SMM을 패리티에 따라 소단위 모듈쌍으로 인터리빙시켜 구성한다. SMM 인터리빙 방법은 구속장의 크기에 관계없이 적용될 수 있으며 분할된 메모리 모듈들의 주소를 제어하는 방법을 고안하였다.

ACS 모듈의 병렬화와 SMM의 인터리빙에 의해서 병렬로 처리되는 상태의 생존 경로 정보들은 TBM (Traceback Memory)에 동시에 저장되어야한다. 본 논문에서는 생존 경로 정보를 동시에 저장할 수 있는 TBM의 구조를 설계하였으며 ACS 모듈이 병렬화됨에 따라 처리 속도를 높이기 위한 TBM의 병렬 분할 방법을 제시한다. 분할된 TBM을 제어하기 위한 주소 발생 방법을 제시하였으며 수행 시간을 단축하기 위한 TBM 주소 발생기로 비트 셔플러를 고안하였다. 본 논문에서 제안하는 SMM과 TBM의 구조는 ACS연산중에 발생되는 메모리에 대한 제어 시간을 단축시킬 수 있도록 병렬화되었으며 제어 방법과 하드웨어 구현 방법을 통해 실제 복호기의 구현시 부가적인 하드웨어를 줄일 수 있도록 구성되었다.

Ⅱ장은 길쌈 부호화, 비터비 복호 알고리즘의 기본 이론과 용어에 대해 설명하고 Ⅲ장에서는 본 논문에서 제안하는 SMM의 병렬 구조, 제어 방법과 하드웨어 구현 방법이 설명된다. Ⅳ장에서는 TBM의 구조, 제어 방법과 하드웨어 구조에 대해 설명하며 Ⅴ장에서 본 논문의 결론을 맺는다.

Ⅱ. 비터비 복호기

1. 길쌈 부호화

길쌈 부호화는 k비트의 입력 데이타를 n비트의 심볼로 부호화하는 방법이며 그 파라미터값은 (n, k, K)로 표현된다. 여기에서 K는 한 입력 데이타가 다음 심볼의 생성에 영향을 미치는 횟수로서 구속장(Constraint length)이라 부르며 길쌈부호의 성능과 비터비

복호기의 복잡도에 영향을 주는 주요 변수이다. 부호어(codeword)는 K−1크기의 쉬프트 레지스터에 저장된 이전의 데이타와 현재 입력 데이타를 사전에 정의된 생성 다항식(generator polynomial)에 의해 구성된 이진 가산기를 통해 생성된다. 쉬프트 레지스터의 값은 상태(State)로 정의되며 구속장 K에 따라 2^{K-1}가지가 존재한다.

길쌈 부호화는 시간에 따라 상태 천이를 나타내는 격자도(Trellis diagram)로 표현할 수 있다. 다음 그림 1은 (n, k, K)=(2, 1, 3)인 경우의 격자도로서 구속장이 3이며 (00, 01, 10, 11) 등의 상태가 있다. 입력 데이터는 1비트로서 이값에 따라 출력되는 2비트의 부호어가 결정된다. 예로서 입력 데이터가 1인 경우 초기 상태를 00으로 가정할 때 출력 부호어는 00이며 상태 10으로 천이한다.

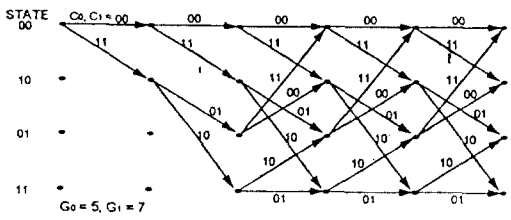


그림 1. (2, 1, 3)인 부호기의 격자도

그림 1에서 상태 천이 경로상에 표시되는 값은 입력 데이타에 따라 생성된 부호어들을 나타내며 각 상태로 입력되는 천이 경로는 두 개가 존재한다. L비트 단위의 입력 데이타를 한 프레임으로 정의하면 쉬프트 레지스터의 내용에 따라 존재하는 가능한 전송 경로 즉 상태열, 또는 부호열의 가지수는 2^L개이다. 따라서 L이 클수록 수신측에서 최적의 전송 경로를 추정하데 필요한 계산량이 기하급수적으로 늘어난다.

2. 비터비 복호 알고리즘

길쌈 부호화된 신호의 최적의 복호 알고리즘으로 알려진 비터비 알고리즘은 부호기의 입력 데이타에 따른 전송 경로와 가장 가까운 경로를 추정하여 전송 비트열을 복호하는 방법이다. 즉 에러를 포함하는 수신된 비트들을 길쌈 부호기의 출력으로 가정하고 그와 같은 출력을 발생할 확률이 가장 큰 입력 데이타

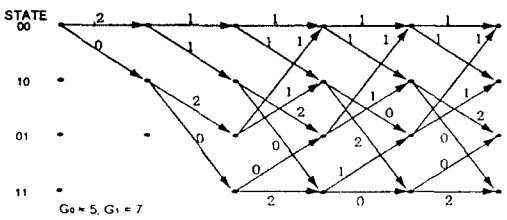


그림 2. (2, 1, 3)의 복호기의 격자도

를 복호된 비트열로 추정한다. 다음 그림 2는 구속장이 3이고 부호율이 1/2인 복호기의 격자도이다.

그림 2에서 천이 경로위의 수치는 수신된 심볼과 부호기의 격자도에서 발생되는 부호어의 거리로써 가지값(Branch Metric)으로 정의된다. 가지값은 경성 판정인 경우 해밍 거리로 구해지며 연성 판정인 경우 유클리드 거리를 계산함으로서 구해진다.[7] 전송 경로는 구속장에 의해 발생되는 상태수와 송신될 데이타양에 따라 지수적으로 증가한다.

비터비 복호 알고리즘은 수신 주기마다 가지값이 작은 경로인 생존경로를 선택하여 기본적으로 가능성이 적은 경로를 처음부터 고려 대상에서 제거한다. 그림 2의 격자도에서 같은 상태로 천이되는 경로는 각 상태마다 두개씩이 존재하며 심볼이 수신될때마다 작은 거리값을 갖는 생존 경로를 선택한다. 생존 경로의 선택은 계산해야 할 경로수를 상태수만큼으로 제한하며 비터비 복호기의 계산량과 하드웨어의 복잡도를 감소시킨다. 비터비 알고리즘은 최적의 경로를 추정하기 위해서 모든 상태에 대한 생존 경로를 계산하며 traceback depth동안 가지값을 누적시켜 경로값(State Metric)을 계산한다. 선택된 생존 경로에 대한 정보는 상태의 천이에 따라 없어지는 하위 비트이며 TBM에 저장된다. 데이타를 복호하기 위해서 비터비 알고리즘은 traceback depth후에 경로값이 가장 작은 상태를 시작으로 TBM에 저장되어 있는 정보를 이용하여 경로를 추정한다. 최종적으로 복호되는 데이타는 추정된 경로의 초기 상태의 상위 비트이며 한 비트를 복호하기 위해서 Traceback depth만큼의 지연이 필요하다.

3. 비터비 복호기 구조

비터비 알고리즘을 수행하기 위한 비터비 복호기

는 BM부, ACS부, TB부의 3개의 모듈로 구성된다.

(1) BM부

BM부는 수신된 심볼들의 전송 확률을 계산하는 모듈로 수신된 심볼과 격자도에서 발생 가능한 부호어들의 거리값을 계산한다. 길쌈 부호화에 의해 구속장이 K인 경우 2^{K-1}개의 가능한 상태가 존재하며 수신된 심볼에 대해 각 상태로 천이하는 가지값을 계산한다. 예를 들면 그림 3은 부호율이 1/3이고 구속장 9인 경우의 상태 천이에 따라 가지값을 계산하기 위한 일반화된 격자 구조를 보여주고 있다.

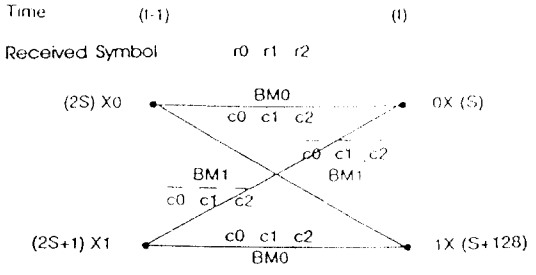


그림 3. 부호율 1/3, K=9인 격자 구조

그림 3에서 X는 시간 t-1에서 부호기의 K-1(8)개의 시프트 레지스터중 왼쪽 7비트 값을 의미하여 그 값은 0에서 127사이의 값을 갖는다. 즉 시간 t-1에서 상태값은 X0 또는 X1이고 시간 t에 들어온 입력이 1이냐 0이냐에 따라 1X, 0X로 천이한다. BM1과 BM0는 채널에 따라 t-1시간에서 t의 시간으로 상태가 변화할 때 부호어(c)와 수신된 심볼(r)과의 해밍거리 또는 유클리드 거리로서 두 개의 평행 천이시에는 (X0→1X, X1→1X)일때의 부호어가 서로 같고 대각천이 (X0→1X, X1→0X)일때의 부호어가 서로 같으므로 평행 천이시에는 BM0, 대각 천이시에는 BM1으로 구해진다.

(2) ACS부

ACS부는 BM부에서 계산된 가지값과 이전까지의 경로값으로 각 상태의 새로운 경로값을 계산하며 생존 경로를 결정하는 모듈이다. ACS 모듈은 새로운 경로값을 계산하기 위해서 각 상태마다 이전 시간까지의 경로값을 저장할 수 있는 메모리를 필요로한다. ACS모듈은 심볼이 수신될 때마다 모든 상태에 대한 경로값을 SMM에서 읽어와서 현재 구해진 가지값과 더하며 그 중 작은값을 갖는 경로를 선택한다. 따라서 작은값을 가지는 경로는 그 상태의 생존 경로가 되며 새로운 경로값은 SMM의 내용으로 저장된다. 생존 경로를 선택하기 위한 경로값의 계산은 다음 식 (1), (2)와 같이 표현된다.

$$SM_{0X} = Min(SM_{X0} + BM0,\ SM_{X1} + BM1) \quad (1)$$

$$SM_{1X} = Min(SM_{X0} + BM1,\ SM_{X1} + BM0) \quad (2)$$

식 (1), (2)에서 경로값 SM_{X0}, SM_{X1}는 상태 X0와 X1을 주소로 사용하여 SMM으로부터 읽어지며 상태 0X와 1X의 경로값 SM_{0X}, SM_{1X}가 새롭게 계산되어 SMM에 저장된다.

한조의 ACS모듈은 나비 구조에 의해 한번에 두개의 상태를 처리한다. 그러므로 ACS모듈의 반복 수행 횟수는 한 수신 심볼에 대해 총 상태수의 2^{K-2}이 되며 구속장이 커질수록 지수적으로 증가한다. 고품질의 통신을 요하는 시스템에서는 구속장이 커지게 되며 비터비 복호기를 고품질과 고속의 통신 시스템에 적용해야되는 경우에는 ACS모듈을 병렬화하는 구현 방법이 요구된다. 병렬 비터비 복호기는 경로값의 계산에 여러개의 ACS모듈을 할당하며 상태의 생존 경로를 동시에 계산하므로 수행 속도를 높일 수 있다. 완전 병렬 비터비 복호기는 총 상태수의 1/2조의 ACS 모듈을 사용하여 구현되는 경우이며 구속장이 커지면 하드웨어의 복잡도와 비용은 지수적으로 증가하게 된다. 그러므로 적용되는 시스템에 맞게 ACS 모듈의 병렬화를 달리하는 방법과 그에 따른 SMM과 TBM의 구조와 제어 방법이 요구된다.

(3) TB부

TB부는 ACS모듈에서 구해진 모든 상태의 생존 경로에 대한 정보를 TBM에 저장한다. TBM에 저장되는 정보는 식 (1), (2)에 따라 선택된 생존 경로의 t-1에서 이전 상태의 LSB(Least Significant Bit)이며 각 상태의 전송 경로를 역추정할 수 있는 포인터로 사용된다. 한 수신 심볼을 복호하기 위해서는 traceback depth(M)만큼의 시간이 지나야 하며 최소의 경로값

을 갖는 상태를 초기 포인터로 시작하여 TBM에 저장되어 있던 정보를 LSB로 쉬프트 시키면서 경로를 추정한다. 최종적으로 복호되는 데이타는 추정된 경로의 초기 상태중 천이에 의한 입력 비트인 MSB (Most Significant Bit)이다. TBM은 구속장에 의해 발생하는 상태수와 traceback depth의 2차원 배열 형태($2^{K-1}\times M$)로 구성되고 세로 방향은 상태로, 가로방향은 시간의 흐름에 따른 traceback depth의 카운터로 제어된다.

4. SMM과 TBM의 일반적인 구조

비터비 복호기에서 SMM과 TBM에 대한 제어는 수행 시간의 많은 부분을 차지한다. SMM은 상태의 경로값을 저장하며 구속장에 따라 크기가 결정된다. 구속장이 K이면 필요한 메모리의 양은 2^{K-1}워드이다. 생존 경로 정보를 저장하기 위한 TBM은 $2^{K-1}\times$ TB depth 만큼의 크기가 요구되며 ACS연산후에 결정되는 생존 경로 정보를 저장하기 위해 상태를 포인터로 사용하여 제어된다. 각 메모리의 구조에 따라 복호기의 수행 속도에 영향을 줄 수 있으며 본 논문에서 제안하는 메모리의 구조를 설명하기 전에 기존의 SMM과 TBM의 구조에 대해 알아본다.

(1) SMM 구조

1) 이중 버퍼링

이중 버퍼링은 상태의 경로값을 읽어오고 저장하기 위해 각각 2^{K-1}워드인 2조의 SMM을 이용하는 방법이다. 임의의 수신 시간 t에서 한쪽의 메모리가 이전의 경로값을 저장하기 위해 사용되었다면 모든 상태의 ACS연산이 끝난 t+1에서는 ACS모듈에서 경로값을 읽어가기 위한 메모리로 사용된다.

이중 버퍼링 방법은 상태를 주소로 하며 경로값을 고정된 위치에 저장하며 제어 방법이 간단하다. 그러나 SMM을 순차적으로 제어하며 한번의 ACS연산을 수행하기 위한 메모리의 제어는 각각의 경로값을 읽고 재저장하기 위해 4번의 순차적인 주소를 발생해야 한다. 또한 두 상태의 경로값을 동시에 계산할 수 없게 되어 ACS모듈의 수행 시간이 길어지며 두 조의 SMM을 사용하므로 메모리에 대한 부가적인 비용이 증가하게 된다. 다음 그림 4는 K=3인 경우 이중 버퍼링 방법으로 SMM을 구현한 경우 경로값이 저장되는 원리를 보여주고 있다.

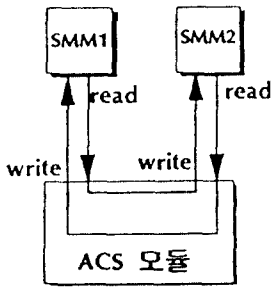


그림 4. 이중 버퍼링 구조

2) 단일 버퍼링

단일 버퍼링 방법은 하나의 메모리를 이용하여 새롭게 계산된 경로값을 읽어 온 위치에 다시 저장하는 방법이다. 단일 버퍼링은 C. M. Rader에 의해 제안된 구조이며 읽어 온 경로값의 주소에 다른 상태의 경로값이 저장되게 되어 한 스테이지의 ACS 연산마다 경로값의 저장 위치가 달라진다.[6] 단일 버퍼링은 경로값의 저장 위치가 모든 상태에 대한 경로값 계산이 끝나는 하나의 스테이지마다 변하므로 각 스테이지마다 경로값 위치를 찾을 수 있는 적절한 어드레싱이 필요하다. 단일 버퍼링 역시 한번의 ACS연산을 수행하기 위해 경로값이 순차적으로 메모리에서 읽혀지기 때문에 수행 시간이 길어진다.

(2) TBM 구조

ACS모듈에서 전송되는 생존 경로 정보는 traceback depth 후에 데이타를 최적의 경로를 따라 복호하기 위한 역방향 포인터로 사용된다. 일반적으로 사용되는 TBM의 구조는 단순한 2차원 배열 형태이다. 그림 5는 TBM의 구조를 나타낸다.

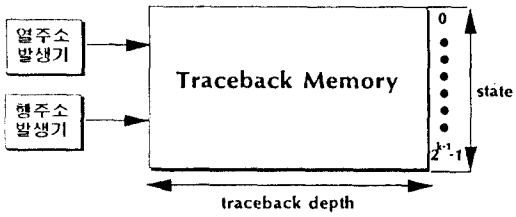


그림 5. TBM의 구조

그림 5의 TBM 구조는 메모리 자체의 단일 구조로 인해 ACS 모듈에서 두 상태의 경로값을 동시에 계산되더라도 발생되는 생존 경로 정보를 순차적으로 저장할 수 밖에 없게된다. 기존의 TBM 구조는 특별한 주소 발생이 필요하지 않지만 고속으로 동작하는 병렬 비터비 복호기의 TBM 구현에는 적합하지 않으며 TBM 자체도 병렬화되어야 한다.

Ⅲ. ACS병렬화에 의한 SMM 구조와 제어 방법

SMM의 구조는 비터비 복호기의 수행 속도에 큰 영향을 미치는 부분으로 비터비 복호기의 구현시 효율적인 SMM의 구성 방법에 대한 고려는 필수적이다. SMM을 구성함에 있어서 가장 큰 요구 사항은 ACS연산 수행시 메모리 제어에 대한 시간을 감소시키는 것이다. 기존의 이중 버퍼링 방법과 단일 버퍼링 방법은 메모리의 순차적인 제어로 인해 ACS모듈의 수행 시간을 길게 하며 나비 구조를 이루는 두 상태의 생존 경로를 동시에 처리하지 못하는 단점을 가진다. 비터비 복호기가 병렬화되어 여러개의 ACS모듈이 사용되면 새로운 SMM의 구조가 설계되어야 하며 관리 방법이 필요하다. 이 장에서는 한조의 ACS 모듈로 경로 계산을 병렬로 처리할 수 있는 SMM의 구조와 제어 방법을 설명하며 다수의 ACS모듈을 사용하여 비터비 복호기를 병렬로 구현하는 경우 SMM의 구조를 제시하고 제어 방법을 설명한다.

1. SMM의 패리티 분할 구조

경로 계산을 병렬로 수행하기 위해서는 SMM에서 두개의 경로값을 동시에 읽어 올 수 있어야 한다. 이를 위해 단일 SMM을 상태의 패리티에 의해 모듈로 분할하는 패리티 분할 구조가 제안되었으며 패리티에 의한 SMM 분할은 ACS연산시 입력되는 상태 X0, X1이 패리티에 의해 구분되는 특성을 이용한 방법이다. SMM의 패리티에 의한 분할 구조는 두 상태의 경로값을 SMM에서 동시에 읽어와서 ACS 연산을 수행할 수 있도록 한다.[6]

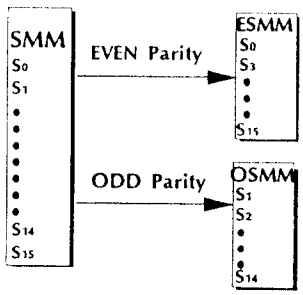


그림 6. SMM의 패리티 분할 구조

SMM은 구속장에 의해 발생될 수 있는 모든 상태가 EVEN이나 ODD패리티로 구분되는 원리를 이용하여 EVEN과 ODD 패리티 메모리의 모듈로 분할되며 상태의 경로값이 ACS모듈로 동시에 로드될 수 있도록 저장된다. 위의 그림 6은 K=5인 경우 SMM이 ESMM (Even Parity SMM)과 OSMM(Odd Parity SMM)로 인터리빙되는 원리를 보여준다. 그림 6에서 SMM에 저장될 발생 가능한 상태는 0에서 15까지이며 상태에 대한 경로값이 ESMM과 OSMM으로 구분된 메모리에 각각 저장된다.

동일한 메모리에 읽기와 쓰기를 차례로 수행하므로 경로값의 저장 위치는 한 스테이지의 ACS연산이 끝나면 변화하게 되며 다음 스테이지에서 저장 위치가 변화된 경로값을 읽어오기 위해서 SMM의 적절한 주소를 발생시켜야 한다. 상태의 생존 경로값 계산은 일정한 계산 순서와는 무관하다. 따라서 주소의 발생은 순서에 관계없이 나비 구조의 특징에 따라 스테이지마다 한 비트만 다른 두 상태의 경로값을 읽어 올 수 있도록 발생되어야 하며 스테이지는 일정한 주기를 가지고 반복된다.

2. ACS모듈의 병렬화에 따른 SMM 구조

(1) SMM 병렬화

패리티에 의한 SMM의 모듈 분할은 나비 구조에 의해서 연산이 이루어지는 두 상태의 생존 경로 계산을 동시에 처리할 수 있도록 한다. ACS 모듈이 병렬화되면 SMM 역시 ACS모듈의 병렬화 정도에 따라 병렬 구조로 바뀌어야 한다. SMM은 각각의 ACS모듈이 경로값 계산을 병렬로 처리하도록 더 작은 크기의 모듈로 인터리빙되며 high-order 인터리빙 방법이 사용된다.[8] 한조의 ACS모듈에 대해 각각 EVEN과 ODD패리티로 분할된 SMM 모듈쌍이 할당되고 병렬화되는 ACS모듈의 수를 N이라고 가정하면 SMM 모듈쌍은 N조가 된다.

그림 7은 ACS모듈의 병렬화에 따라 SMM이 high-order 인터리빙되는 원리를 보여준다. 구속장이 K일 때 상태수는 2^{K-1}이며 SMM을 제어하기 위한 주소는 k−1비트이다. ACS모듈이 N조로 병렬화되면 high-order 인터리빙 방법에 의해 SMM은 모듈쌍 N조로 분할되며 분할된 SMM을 제어하는데 필요한 주소 비트는 다음 식 (3)의해 계산된다.

$$\left(\frac{2^{K-1}}{N}\right) = \left(\frac{2^{K-1}}{2^m}\right) = 2^{K-m-1} \tag{3}$$

식 (3)에서 m비트는 high-order 인터리빙에 의해 분할된 SMM쌍을 구분하기 위한 비트이며 K−1비트의 제어 비트중 상위 비트가 할당된다. 각 SMM 모듈쌍은 내부적으로 패리티에 의해 ESMM과 OSMM 모듈로 분할되므로 실제 각 SMM 모듈을 제어하기 위한 주소 영역은 상위 m 비트와 최하위 비트를 제외한 K−m−2비트이다. 최하위 비트는 메모리 인터리빙에 의해 없어지는 상태의 패리티 비트로서 은닉(hidden)값으로 존재하며 SMM의 분할에 따라 병렬 구조로 분할되는 TBM의 주소를 발생시키는데 사용된다.

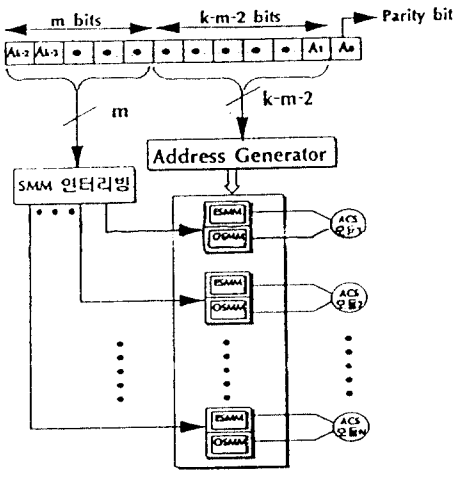


그림 7. 병렬 ACS를 위한 메모리 인터리빙

임의의 SMM모듈쌍은 일정한 범위의 상태에 대한 경로값만을 저장한다. SMM모듈쌍이 저장할 수 있는 상태의 범위(range)는 다음 식 (4), (5)에 의해 계산된다. a_j는 상위 m비트의 계수를 나타내며 0또는 1의 값을 갖는다.

$$S_i^i = \sum_{j=0}^{m-1} a_j \cdot 2^{K-2-j},\ a_j \in \{0, 1\} \tag{4}$$

$$S_i^i \le S_j < S_i^{i+1} \tag{5}$$

식 (4)와 (5)에서 S_i^i는 i번째 SMM모듈쌍에 저장되는 경로값의 상태들 중 최하위 상태를 의미하며 S_j는 임의의 모듈에 저장된 경로값의 상태를 나타낸다. 상태의 경로값 저장 위치는 식 (5)에 의해 고정된 SMM 모듈쌍으로 범위가 정해지며 경로값 저장 위치는 스테이지에 따라 같은 모듈내에서 위치만 달라지게된다.

(2) 주소 반복 주기와 SMM 제어

ACS모듈의 병렬화에 따라 SMM은 2^{K-1} 크기의 단일 버퍼가 소단위 모듈로 분할되는 병렬 구조로 구성된다. N조의 ACS모듈에서는 각각 한 비트씩 다른 상태 X0, X1의 경로값이 동시에 읽어들여지며 X0, X1의 경로값이 저장되어 있던 위치에는 계산된 상태 0X, 1X의 경로값이 동시에 저장된다. 한 ACS모듈에서 계산된 경로값은 정해진 범위의 SMM모듈쌍에 저장되어야 하므로 계산된 경로값의 상태에 따라 SMM 모듈쌍으로 교차되어 저장된다. 다음 그림 8은 ACS 모듈이 2조인 경우 스테이지마다 ACS모듈에서 계산된 경로값이 SMM모듈쌍에 연결되는 원리를 보여주고 있다.

SMM1$_{X0}$과 SMM1$_{X1}$은 SMM1모듈쌍의 ESMM이나 OSMM에 저장된 경로값의 상태를 나타내며 SMM0$_{X0}$

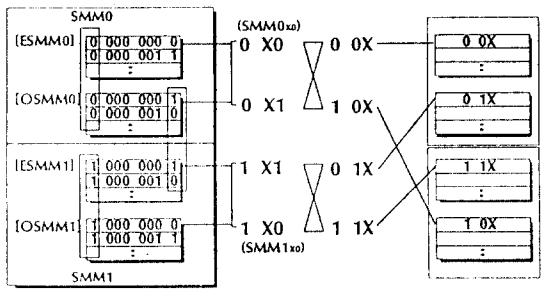


그림 8. 경로값이 SMM모듈에 연결되는 원리

과 $SMM0_{X1}$는 SMM0에 저장된 경로값의 상태를 나타낸다. ACS 모듈에서 새롭게 계산된 상태의 경로값은 X의 패리티에 따라 저장될 모듈이 결정된다. 만약 ACS모듈이 완전하게 병렬화되면 SMM은 모두 분할되어 하드웨어적인 연결만하면 제어되며 주소는 무의미하게된다.

그림 8과 같이 ACS모듈이 완전히 병렬화되지 않은 경우에는 한 스테이지의 ACS연산이 끝날때마다 각 SMM모듈에 저장되는 경로값의 저장 위치가 변하게 되며 다음 스테이지의 ACS연산을 위해 SMM의 주소를 제어해야 한다. SMM0와 SMM1에 저장되는 상태는 모듈쌍을 구분할 수 있는 최상위 비트만 제외하면 ESMM0와 OSMM1, ESMM1과 OSMM0의 나머지 비트가 동일하며 동시에 2조의 ACS연산을 위해 두 SMM모듈쌍이 같은 주소로 제어될 수 있다. SMM모듈쌍은 내부적으로 패리티에 의한 모듈로 분할되어 있으므로 어느 패리티의 모듈을 순차적인 주소로 제어할 것인지를 결정해야 한다. 순차적인 주소로 제어되는 모듈을 기준 메모리라고 정의한다.

기준 메모리는 어느 패리티 모듈로 정해져도 무관하지만 모든 SMM모듈쌍에 동일하게 적용되지 않으며 동시에 계산되는 상태 $SMM0_{0X0}$, $SMM0_{0X1}$, $SMM1_{1X0}$, $SMM1_{1X1}$이 모두 패리티 의해 구분될 수 있도록 정해져야한다. ACS연산을 통해 계산되는 경로값은 상태에 따라 범위가 고정된 SMM 모듈로 저장되며 동시에 동일한 패리티 모듈에 저장될 경로값이 존재할 경우가 발생한다. 예를들면 같은 패리티 모듈로 저장되는 상태들은 서로 2비트의 차이가 나며 $SMM0_{0X0}$과 $SMM1_{1X0}$가 같은 패리티인 경우 동시에 같은 모듈에 저장되는 상태가 발생된다. 따라서 SMM0의 기준 메모리가 ESMM0로 결정되면 SMM1의 기준 메모리는 OSMM1으로 결정되어야 데이타의 충돌이 발생하지 않는다. 실제적으로는 상위 m비트를 제외하고 같은 패리티인 모듈들이 기준 메모리로 결정된다. 식 (6)은 SMM0와 SMM1에서 동시에 읽혀지는 상태들의 관계를 나타낸다.

$$SMM1_{X0} = SMM0_{X0} + \left(\frac{2^{K-1}}{N} \right) \tag{6}$$

ACS모듈이 N조로 병렬화되면 각 ACS모듈에서 계산되는 상태의 관계는 각각이 패리티에 의해 구분이 되기 위해서 $2^{K-1}/N$ 만큼의 차이가나야 하며 기준 메모리는 식 (6)을 만족하도록 $2^{K-1}/N$의 배수를 포함하는 메모리 모듈로 설정된다.

ACS연산후에 경로값의 저장 위치는 스테이지에 따라 달라지게 되며 스테이지는 일정한 반복 주기를 가진다. 스테이지의 주기는 ACS모듈의 병렬화 정도와 구속장에 의해 계산될 수 있다. 먼저 스테이지에 따라 상태의 저장 위치는 다음 식 (7)에 의해 계산된다. m은 ACS모듈의 병렬화에 따른 SMM 인터리빙에 의한 제어 비트이며 $A_{K-m-2:1}$는 SMM모듈에 저장되는 상태의 초기 주소를 나타낸다. EP(Even Parity)또는 OP(Odd Parity)는 SMM 패리티 모듈에 따라 주소 비트와 제어 비트의 패리티를 만족하도록 구해지는 패리티 비트이다.

$$P(S)_{stage} = CSL(A_{K-m-2:1} : EP)_{stage} \tag{7}$$

P는 스테이지에 따른 상태(S)의 저장 위치를 나타내며 EP를 스테이지만큼 CSL(Circular Shift Left)시킨 레지스터의 값이다. 상태의 저장 위치는 스테이지가 K-m-1만큼 순환되면 원래의 위치로 돌아오게 되며 이에 따라 주소를 발생시키기 위한 스테이지의 주기는 식 (8)에 의해 계산된다.

$$T_{StageCycle} = K - m - 1 \tag{8}$$

3. K = 9, ACS 2조인 SMM 제어 예

본 논문에서 적용한 메모리의 인터리빙을 이용한 SMM 병렬화는 임의의 구속장의 크기와 요구되는 ACS 모듈의 병렬화 정도에 따라 적용할 수 있는 일반적인 방법이다. 구속장 K는 9로 가정하고 ACS모듈의 갯

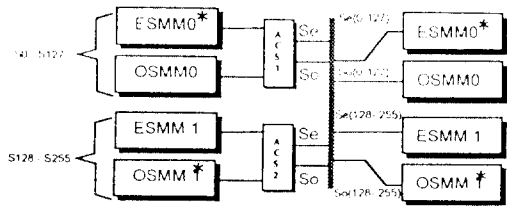


그림 9. ACS모듈 2조인 경우 SMM 구조

수는 2조로 가정한다. 구속장 K에 따라 발생되는 상태의 경우의 수는 256이며 필요한 메모리양은 256워드이다. 다음 그림 9는 위의 가정에 따른 SMM의 구조이다.

그림 9의 SMM은 ACS모듈 2조에 의해 2개의 모듈쌍으로 분할된다. 각 SMM 모듈쌍은 내부적으로 패리티에 따라 ESMM과 OSMM 모듈로 구성되며 256워드 크기의 SMM을 제어하기 위해 8비트가 필요하다. 제어 비트를 A_7-A_0라고 놓으면 최상위 비트인 A_7은 SMM 모듈쌍 1과 2를 구분하기 위한 비트로 이용된다. A_0는 각 패리티 메모리 모듈에 저장되는 상태의 패리티를 만족하는 EP 또는 OP로 임의의 순간에 ACS모듈에 입력되는 경로값의 상태를 구하기 위해 이용되며 스테이지에 따라 달라지는 각 상태의 경로값 저장 위치를 구할 수 있는 정보이다. 실제적으로 주소 발생기에서 발생되는 주소는 A_6-A_1이며 순차적인 주소와 함께 메모리 모듈을 제어하게 된다. 다음 그림 10은 ACS연산에 따른 상태의 저장 위치의 변화를 나타낸다.

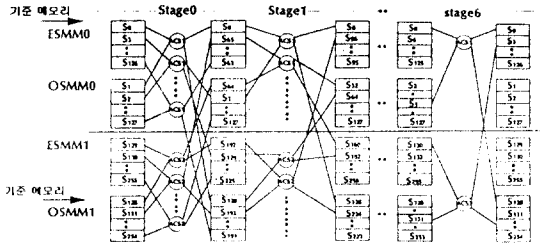


그림 10. 2조의 SMM 모듈쌍의 ACS연산 흐름

스테이지의 주기는 식 (8)에 의해 구속장이 9이고 ACS모듈이 2인 경우 7이 된다. 주소 발생기는 상태의 경로값 저장 위치가 스테이지마다 반복 주기를 가지고 변화하므로 ACS모듈에 서로 한 비트만 다른 상태의 경로값을 전송하기 위해서 7 스테이지를 주기로 메모리 모듈의 주소를 발생시킨다.

기준 메모리는 식 (6)에 의해 상태 0과 128을 포함하는 메모리 모듈로 설정된다. 위의 그림 8에서는 ESMM0와 OSMM1이 기준 메모리로 설정된다. 기준 메모리가 설정되면 주소의 발생은 두 가지로 나뉜다. 기준 메모리는 순차적인 카운터 값에 의하여 발생되고 비기준 메모리는 스테이지에 따라 기준 메모리 주소를 이용하여 발생시킨다.

4. 주소 발생기 하드웨어 구조

병렬화된 SMM은 스테이지에 따라 주소 발생기에 의해 발생되는 기준 또는 비기준 메모리의 주소에 의해 제어된다. 패리티에 의해 분할된 메모리는 동시에 제어되며 주소 발생기는 스테이지에 따라 상태 X0, X1의 경로값이 각각의 ACS모듈에 입력되도록 설계된다. 기준 메모리와 비기준 메모리의 관계는 스테이지에 따른 저장 위치의 변화에 의해 구해질 수 있다. 비기준 메모리의 주소는 기준 메모리의 주소와 스테이지의 함수로 표현되며 발생되는 기준 메모리의 주소가 스테이지에 따라 한 비트씩 토글되는 형태로 발생된다. 비기준 메모리의 주소는 스테이지에 따른 기준 메모리의 주소를 이용하여 관계식 (9)에 의해 표현된다.

$$A_i = D_{i+1} \oplus W_i \tag{9}$$

다음 그림 11은 관계식 (9)에 의하여 구현된 주소 발생기의 하드웨어 모듈이다.

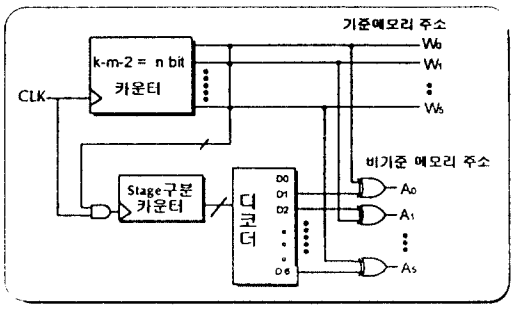


그림 11. 주소 발생기 하드웨어 구조

A_i는 발생되는 비기준 메모리 모듈의 주소이고 W_i는 순차적인 카운터에 의해 발생되는 기준 메모리 모듈의 주소 비트이다. D_i는 스테이지를 구분하기 위한 디코더의 출력이다. 비기준 메모리의 주소는 위의 관계식에 따라 기준 메모리 주소와 디코더 출력을 exclusive-

OR함으로써 구해질 수 있다. 전체적인 SMM 주소 발생기는 7비트 카운터와 디코더에 의해 구현된다.

Ⅳ. TBM의 병렬화와 주소 제어 방법

ACS연산후에 계산된 상태의 생존 경로 정보는 TBM에 저장된다. 기존의 TBM은 ACS연산에 의해 동시에 입력되는 생존 경로 정보를 순차적으로 저장하는 구조로 복호기의 수행 속도를 느리게 한다. 본 논문에서 제안하는 TBM은 SMM과 마찬가지로 ACS모듈에서 병렬로 계산되는 두 상태의 생존 경로의 정보를 동시에 저장하기 위한 구조이다. TBM에는 ACS연산 후에 각 상태의 생존 경로에 따라 1비트씩의 정보 비트가 저장되고 SMM과는 달리 스테이지마다 저장 위치가 변하지 않는다. 그러나 SMM의 단일 버퍼링에 의하여 스테이지마다 상태들의 ACS연산의 순서가 달라지게 되므로 상태에 의해 제어되는 TBM의 열에 대한 주소를 발생시켜야 한다. 본 장에서는 TBM 병렬 구조와 제어 방법을 설명하고 제안된 TBM을 제어하기 위해 고안된 비트 서플러의 구현 원리를 설명한다.

1. TBM의 병렬화

ACS모듈에서는 나비 구조에 의해 상태 0X, 1X의 경로값을 동시에 계산한다. 계산된 상태의 생존 경로 정보는 TBM으로 전송되며 ACS모듈이 병렬화되더라도 TB모듈에서 처리할 수 없으면 복호기의 수행 속도가 저하된다. 생존 경로 정보들을 동시에 저장하기 위하여 TBM은 SMM과 마찬가지로 소단위 모듈로 분할된 병렬 구조로 구성된다. 다음 그림 12는 K=9인 경우 TBM의 분할 구조를 보여준다.

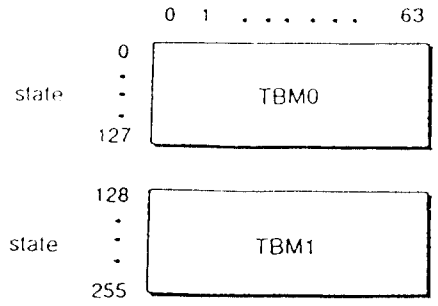


그림 12. TBM 분할 구조

TBM을 분할시키는 경우 ACS모듈의 연산 순서가 순차적이지 않고 스테이지마다 변화되므로 현재 계산되어 입력되는 생존 경로 정보가 어느 상태의 정보인지 알아야 한다. TBM의 열에 대한 주소는 ACS연산 후에 계산되는 상태 0X와 1X에 의해 제어되므로 TBM의 주소 발생기는 ACS연산이 수행중인 상태를 구하는 동작을 수행한다. TBM 역시 ACS모듈의 분할에 따라 더 작은 소단위 모듈로 분할되며 다음 그림 13은 TBM이 병렬화되는 원리와 격자도에 의해 ACS연산 후에 천이되는 상태를 나타낸다.

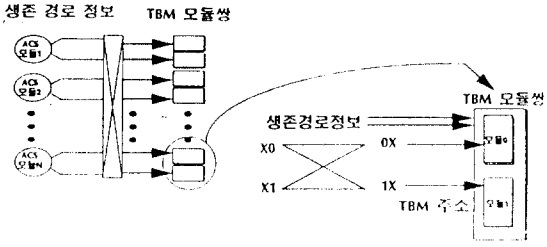


그림 13. TBM 병렬 분할과 천이 상태에 의한 생존 경로 정보 저장

2. TBM 제어 방법

병렬화된 SMM구조에 의하여 ACS모듈에서 계산되는 상태의 순서는 스테이지에 따라 달라진다. 각 상태의 생존 경로 정보를 저장하는 TBM은 행의 주소가 상태에 의해 제어되며 TBM 제어기는 ACS모듈에 의해 계산되는 상태 X0, X1을 구해야한다. 상태를 구하기 위하여 사용할 수 있는 정보는 현재 어느 스테이지의 ACS연산인지와 발생되는 각 SMM 모듈의 주소이다. 이 정보를 이용하여 TBM의 행의 주소가 되는 상태를 구하게 되며 다음 식 (4)와 같이 SMM의 주소와 스테이지의 함수로 표현될 수 있다.

$$TB_{Address}(S_{K-m-1:0}) = f(SMM_{Address} \cdot stage) \qquad (10)$$

TBM의 행의 주소가 되는 상태를 표현하기 위해서 K-1비트가 필요하며 K-1비트는 SMM의 주소, 메

모리 인터리빙으로 모듈쌍을 구분하기 위한 비트와 은닉으로 되는 패리티 비트로 구성된다. 상태를 구하기 위해 K−1비트의 제어 비트의 포맷을 써보면 다음 그림 14와 같다.

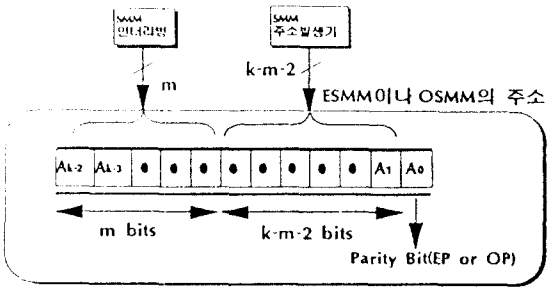


그림 14. 제어 비트 포맷

그림 14에서 상위 비트들은 SMM 모듈쌍의 구분 비트이며 A_{k-m-2}−A_1은 그 순간에 발생되는 ESMM이나 OSMM의 발생 주소이다.

먼저 상태를 구하기 위해 최하위 비트인 패리티 비트 A_0를 구한다. A_0는 어느 SMM 모듈의 주소를 인수로 사용하느냐에 따라 두 가지 방법으로 구해질 수 있다. ESMM의 주소를 사용하는 경우 A_0는 상위 비트들이 EVEN 패리티(EP)가 되도록 정해지며 OSMM의 주소를 이용하는 경우 ODD 패리티(OP)가 되도록 구해진다. 구해진 K−m−2비트들을 그림 15에서 보여지는 것처럼 스테이지만큼 오른쪽으로 순환 쉬프트(CSR)시키면 원하는 상태를 구할 수 있다. 다음 그림 15는 그림 10의 상태 흐름도에서 스테이지 1인 경우 ESMM1의 주소1에 저장되어 있는 상태 65를 구하는 경우이다.

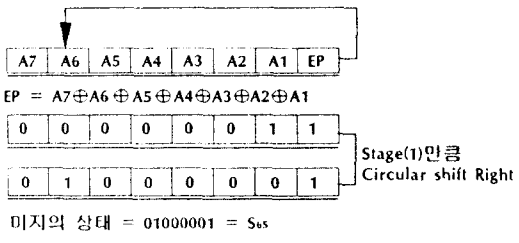


그림 15. CSR에 의한 상태 구하기

3. 비트 셔플러

K−1비트를 스테이지만큼 오른쪽으로 순환 쉬프트 시키는 방법은 한 비트 쉬프트당 그만큼의 클럭이 소모된다. 비트 셔플러는 이러한 문제점을 해결하여 한 클럭동안 쉬프트된 결과가 나올 수 있도록 고안되었다. 비트 셔플러의 구현 원리를 설명하기 위하여 함수 CSR을 식 (11)과 같이 정의한다.

$$CSR(A_{K-m-2:1}:EP)_i = (A_{K-m-2}, A_{K-m-3}, \ldots, A_1, EP)_{i\text{-}CircularShifiRight} \quad 0 \le i \le (K-m-2) \tag{11}$$

식 (11)에서 i는 구속장과 ACS모듈의 병렬화에 따라 구해지는 스테이지의 주기 범위이며 CSR은 상태를 구하기 위한 제어 비트와 EP를 인수로 하여 EP를 LSB로 i번 오른쪽으로 순환 쉬프트 시키는 결과이다. 스테이지에 따라 구하는 상태를 $S[K-m-1:0]$라 하면 구하는 상태는 다음 식 (12)에 의해 표현된다.

$$S[K-m-2:0]_{stage} = CSR(A_{K-m-2:1}:EP)_{stage} \tag{12}$$

식 (12)를 수행하는 CSR은 K−m−1개의 MUX를 이용하여 구현된다. $S[K-m-2:0]$의 각각의 비트는 MUX의 출력으로 이루어지며 상태의 각 비트를 표현하는 MUX를 첨자로 구분하면 MUX_0에서 MUX_{K-m-2}로 나타낼 수 있다. 각 MUX로 연결되는 입력은 첨자에 따라 다르게 연결되며 상태의 임의의 비트를 표현하는 MUX_j의 입력을 $I[K-m-1:0]_j$로 표현하면 식 (13)으로 나타낼 수 있다.

$$I[K-m-2:0]_j = CSR(A_{K-m-2:1}:EP)_j \tag{13}$$

다음 그림 16은 식 (13)에 의하여 구현된 비트 셔플러의 하드웨어 구조이다.

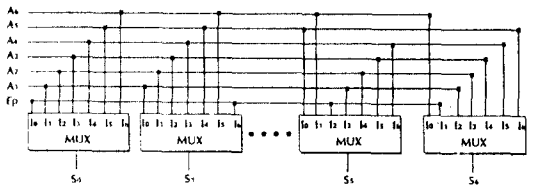


그림 16. 비트 셔플러

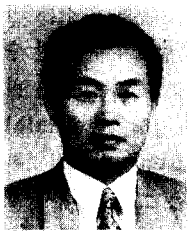
V. 결 론

길쌈 부호화와 비터비 복호 알고리즘은 디지탈 통신 시스템에서 신뢰성있는 데이타의 전송을 위해 사용되는 오류 정정 기술이다. 본 논문에서는 비터비 복호기 처리 속도의 병목이 되는 ACS모듈의 수행 속도를 높이기 위한 방법으로 SMM과 TBM의 구조와 주소 제어 방법을 제안하였다. 제안하는 SMM의 구조는 임의의 구속장에 대해서 적용할 수 있으며 ACS 병렬화의 정도에 따라 모듈로 분할되는 병렬 메모리 구조이다. SMM은 구속장에 따라 크기가 결정되며 ACS모듈의 병렬화 정도에 따라 소단위의 모듈로 인터리빙된다. ACS모듈 한조당 EVEN과 ODD패리티 상태의 경로값을 저장하는 메모리 모듈쌍이 기본으로 구성되며 각각의 메모리는 주소 발생기를 통해 제어된다.

각 상태의 생존경로 정보를 저장하는 TBM은 ACS 모듈의 병렬화에 따라 동시에 입력되는 정보를 저장할 수 있도록 병렬화된다. TBM은 SMM과 마찬가지로 ACS모듈 한조당 두개의 모듈로 분할되며 ACS모듈에서 동시에 계산되는 생존 경로를 정보를 병렬로 처리할 수 있다. TBM에 저장되는 상태의 경로 정보는 저장 위치가 변하지 않지만 전송되는 경로 정보를 올바른 위치에 저장하기 위해서는 스테이지에 따라 상태 발생기가 필요하다. 본 논문에서는 병렬화된 TBM을 제어할 수 있는 방법을 제시하였으며 그 하드웨어로 비트 셔플러를 고안하였다.

참 고 문 헌

1. Andrew J. Viterbi, Jim K. Omura, Principles of Digital Communication and Coding, McGraw-Hill Inc., 1979.
2. Jerrold A. Heller. Irwin Mark Jacobs. "Viterbi Decoding for Satellite and Space Communication", IEEE Trans. on Comm. Technology. Vol. COM-19, No. 5, pp. 835-848, October 1971.
3. G. Fettweis, H. Meyr, "A 100 Mbit/s Viterbi Decoder chip: Novel Architecture and its realization," IEEE International Conference on Communications, Atlanta, No. 307. 4, pp. 463-467, April 1990.
4. G. Fettweis, Student Members, "Parallel Viterbi Algorithm Implementation: Breaking the ACS-Bottleneck", IEEE Trans. on Comm., Vol. COM-37, No. 8, pp. 785-790 Aug. 1989.
5. P. G. Gulak and T. Kailath, "Locally connected VLSI architectures for the Viterbi algorithm", IEEE J. Select. Areas Comm., Vol. SAC-6, pp. 527-537, 1988.
6. C. M. Rader, "Memory Management in a Viterbi Decoder", IEEE Trans. on Comm., Vol. COM-29, No. 9, pp. 1399-1401, Sept. 1981.
7. Bernard Sklar, Digital Communications Fundamental and Application, Prentice-Hall, 1988.
8. K. Hwang Computer Architecture and Parallel Processing, McGraw-Hill Inc., 1984.



박 동 선(Dong-Sun Park) 정회원

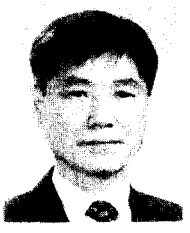
1979년 2월 : 고려대학교 전자공학과(학사)

1984년 12월 : 미국 미주리 주립대 전기 및 컴퓨터공학과(공학석사)

1990년 12월 : 미국 미주리 주립대 전기 및 컴퓨터공학과(공학박사)

1991년 3월~현재 : 전북대학교 정보통신공학과 조교수

※주관심분야 : 컴퓨터시각, 멀티미디어통신, 패턴인식



송 상 섭(Sang Seob Song) 정회원

1978년 2월 : 전북대학교 전기공학과(학사)

1980년 2월 : 한국과학기술원 전기 및 전자공학과(공학석사)

1990년 8월 : 캐나다 마니토바 대학 전기 및 컴퓨터공학과(공학박사)

1981년 4월~현재 : 전북대학교 전기·전자·제어공학부 교수

※주관심분야 : 채널부호이론, 고속전송기술, ATM ASIC, 무선 멀티미디어 전송기술

池 玄 順(Hyun Soon Chi) 정회원
1994년 2월 : 전북대학교 정보통신 공학과(학사)
1996년 2월 : 전북대학교 정보통신 공학과(공학석사)
1996년 1월~현재 : 삼성 데이터 시스템 중앙 개발 정보 시스템실
※주관심분야 : 통신 소프트웨어, 멀티미디어 전송 기술