Chip면적 감소를 위한 Radix-2 구조 구현

최 영 식, 한 대 현
동의대학교 전자공학과

Implementation of Radix-2 structure to reduce chip size

Young-Shig Choi, Dae Hyun Han
Dept. of Electronics, Dongeui University
E-mail : yschoi@hyomin.dongeui.ac.kr

요약

0.5 μm 공정에서는 사용된 Library Cell들의 지연 시간이 커 면적 증가라는 문제를 앉고 있지만, Radix-4 알고리즘의 아키텍쳐를 수용하여 극복할 수 있었다. 그러나 공정 기술의 발달로 인한 Library Cell 자체의 속도 증가에 따라 다시 Radix-2 알고리즘을 수용하여 속도와 면적에 관한 요구를 모두 충족할 수 있게 되었다.

Abstract

Viterbi decoder is implemented with a Radix-4 architecture at 0.5 μm process even though the delay time of standard cell is big and it causes a bigger chip size. As process develops, the delay time of standard cells is getting smaller. Therefore, the requirement of speed and chip size is satisfied by using Radix-2 algorithm to implement Viterbi decoder.

1. 서 론

일반적으로 래딕스-4 알고리즘이란 래딕스-2 알고리즘에서 발생되는 속도제한 문제를 극복하기 위한 알고리즘이지만, 거의 대부분의 속도 개선 알고리즘이 갖고 있는 문제와 마찬가지로 래딕스-4 알고리즘 또한 면적 증가라는 문제점을 갖고 온다. 0.35 μm 테크놀러지에서는 디바이스 자체 속도가 0.5 μm 테크놀러지에 비해 빨라진 0.35 μm 테크놀러지에선 과연 래딕스-4 알고리즘이 래딕스-2 알고리즘에 비해 속도와 면적 그리고 소비 전력 면에서 어느 정도의 효율을 갖게 되는 지에 대한 의문이 자연스럽게 발생하게 되었으며, 그래서 좀 더 효율적인 설계를 위한 목적으로 0.35 μm 테크놀러지에서 두 알고리즘을 각각 하드웨어 구현(합성)하여 그 속도와 면적 그리고 소비-전력 면에서의 효율성 정도를 비교, 검토 해보았다.

II. 비터비 복호기

본 논문에서 언급되는 비터비 디코더는 QPSK Demodulator의 출력 신호를 입력으로 받아 처리하는 채널 복호기를 칭한다. 앞에서 설명한 비터비 알고리즘을 구현하기 위한 비터비 복호기는 그림1과 같이 구성되어 진다. 우선 각 상태 천이 확률을 4가지 가능한 심볼과의 거리로써 계산하는데 사용되어지는 Branch-Metric-Calculation (BMC)이 있다. 이 블록에 의해서 만들어진 각 상태들의 천이 확률은 Path-Metric-Calculation (PMC)에 입력되어져 과거부터 현재까지 현 상태에 이르게 되는 천이 확률이 Path-metric으로 저장되어 진다. 즉, 천이가 가장 확실한 상태의 Path-metric은 0이고, 천이가 이루어지지 않을 부분들의 상태 값들은 매우 높은 Path-metric을 갖게 된다. 그리고 PMC는 매 BMC의 값을 입력할 때마다 각 상태들이 어느 전 상태에서 천이 되었는지를 알 수 있는 근거가 되는 Decision 값을

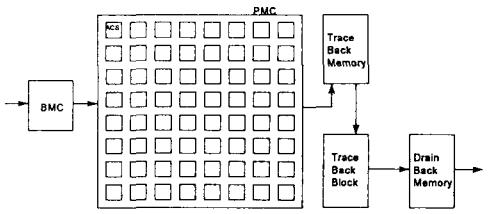

그림 1. 비터비 디코더의 구조

역추적(Trace Back) 메모리에 입력하여 이후 역추적시 활용되어진다. 역추적 블록은 메모리에 저장된 생존 경로(Survival Path)에 대한 정보로부터

역추적을 실시하며, Drain-back memory에서 데이터를 재 정렬하여 출력한다[1].

III. 래딕스-2 알고리즘

래딕스-2 알고리즘이란 비터비 알고리즘의 기본 구조를 의미하며, 단지 래딕스-4나 래딕스-16과 같은 알고리즘과 구별하기 위한 명칭이다[2]. Convolutional 부호화 된 데이타를 복호시키는 알고리즘 가운데 Maximum-Likelihood 방식을 이용한 비터비 알고리즘이 일반적으로 가장 성능이 좋다고 알려져 있다.

설명을 위해 그림2과 같이 단순한 상태 4가지만 존재하는 Convolutional 부호를 가정한다. 표1에 예시된 예는 그림2와 같은 Convolutional 부호화를 전제로 시작된다. 표-1은 부호화할 때의 Data 입력과 현재상태, 다음 상태와 데이터 출력(g1,g2)의 값과 채널을 지난 후 수신 단에서 받는 입력(g1,g2)의 값을 보인다.

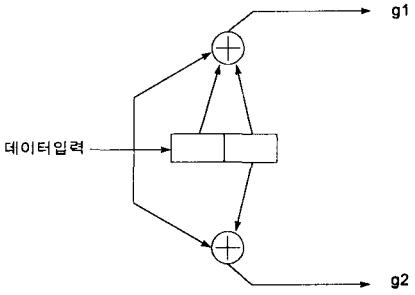

그림 2. 4가지 상태를 갖는 Convolutional 부호기

데이터 입력	현재 상태	다음 상태	송신부(g1,g2)	수신부(g1,g2)
0	00	00	00	00
1	00	10	11	11
1	10	11	01	11
0	11	01	01	01
1	01	10	00	00
0	10	01	10	10
1	01	10	00	00
1	10	11	01	11
0	11	01	01	01
0	01	00	11	11

표 1. Convolutional 부호기의 입력 및 상태 천이

표에서 보듯이 3번째와 8번째의 수신 신호는 송신 신호와 다름을 알 수 있다. 이는 채널에서 발생한 에러 때문이라고 가정한다. 이의 복호화를 설명하기 위해 우선 그림3의 트렐리스 다이어그램을 살펴본다. 그림3의 트렐리스 다이어그램은 그림1의 부호화기에서 유추해낸 것이다. 즉, 한 예로 현 상태 00은 입력이 0일 경우는 다음 상태 00으로 천이 되고 출력 (g1,g2)=(0,0)이다. 또한 입력이 1인 경우는 다음 상태10으로 천이 되고 출력 (g1,g2)=(1,1)이 된다. 그림3은 이 트랠리스도를 기본으로 하는 트랠리스도의 흐름을 제시하고 복호화의 흐름을 나타내 준다.

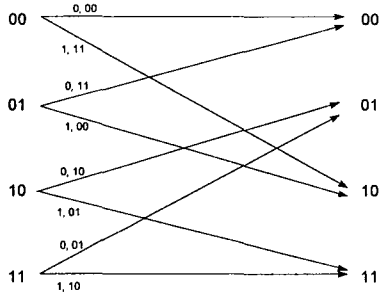

그림 3. 4가지 상태들의 천이도

IV. 래딕스-4 알고리즘

Path-Metric-Calculation(PMC) 블록에선, BMC로부터 입력 받은 Branch-Metric을 전 사이클의 Path-Metric과 더하여 다음 사이클의 Path_Metric 후보들을 만들어 낸다. 이 후보들 중 더 작은 값을 갖는 후보를 다음 사이클의 Path-Metric으로 선택하게 되는데 이러한 과정은 PMC블록의 하위 블록인 Add-Compare-Selection(ACS)블록에서 처리된다.

0.5μm 테크놀러지 공정에서 래딕스-2 알고리즘은 비터비가 60MHz에서 동작할 때, PMC가 이 속도에 맞는 정 동작을 수행하지 못 하였다. 그래서 PMC 블록 내의 속도를 전체 동작 주파수의 1/2에서 수행할 수 있도록 한 래딕스-4 알고리즘을 수용하였으며, 이로 인해 60MHz에서 정동작할 수 있는 비터비 복호기를 구현 할 수 있게 되었다. 즉, 래딕스-4 알고리즘은 래딕스-2 알고리즘이 상태 천이를 한 싸이클에 한 번씩 하는 반면 한 싸이클에 두 번의 상태 천이를 실시하므로 전체 비터비의 동작이 60MHz에서 실시될 때 PMC 내부적으로는 30MHz에서 동작하도록 하는 알고리즘이다[3]. 그림4는 한 사이클에 두 번의 상태 천이를 하는 래딕스-4 트렐리스 다이어그램을 보인다.

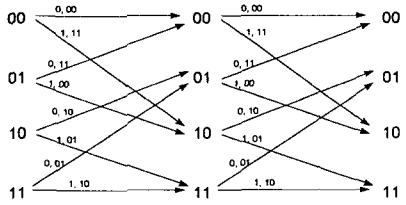

그림 4. 래딕스-4의 트렐리스 다이어그램

다음 단계의 Path-Metric은, 전 단계에서 오는 Path-Metric과 두 사이클 동안 입력 된 데이터로부터 추출된 Branch-Metric들의 합으로 생성되는 4개의 후보-Path-Metric들 중 가장 작은 값을 갖는 후보로 선택된다.

그러나 한 사이클에 두 번의 천이 과정을 처리해야 함으로써 BMC, PMC블록의 면적이 증가되었는데, 특히 ACS블록의 사이즈는 2배 이상으로 증가하였고 이는 곧 PMC블록의 면적이 2배이상 증가했음을 의미한다. 또한 BMC, PMC 그리고 역추적 블록을 구동시키는 래딕스-4 클럭이 추가되었으며, 버스 width가 증가하고 부가적인 네트들의 발생으로 라우팅 채널 면에서도 큰 증가가 발생하였다.

V. 비교 및 검토

공정 기술의 진화로 0.35μm 테크놀러지에서 비터비 복호기의 리비젼이 수행됐을 때, 과연 래딕스-4 알고리즘을 계속 사용해야 하는가, 그리고 현 공정 기술에선 어느 정도의 속도까지 래딕스-2 알고리즘을 이용할 수 있는지에 대한 의문이 발생하였다. 그래서 두 알고리즘의 비교를 위해 각각의 구조에 따른 비터비 복호기를 구현하여 그 결과를 속도, 면적 그리고 소비전력면에서 비교해 보았다.

1. 속 도

래딕스-4 알고리즘을 사용하는 주된 이유는 PMC 블럭 내에서 발생되는 Critical-path 때문이다. PMC 블록은 그림1에서 알 수 있듯이 64개의 ACS 블록의 어레이 형태이다. 0.35μm 테크놀러지에서 구현된 ACS의 최대 동작 속도를 알고리즘에 따라 각각 측정해 보면 래딕스-2는 최대 90MHz까지, 래딕스-4는 최대 60MHz까지 동작 가능했다. 즉, 래딕스-4의 경우 비터비 복호기는 120MHz까지 동작할 수 있음을 의미한다. 과거 공정(0.5μm 테크놀러지)에서는 ACS의 동작 속도가 외부 클럭속도(60MHz)에서 동작할 수 없었던 이유로 래딕스-4 알고리즘을 채용하여 원하는 결과를 얻을 수 있었다. 그러나 현 공정(0.35μm 테크놀러지)에서는 비터비 기본 구조인 래딕스-2알고리즘에서도 충분히 요구되는 동작 속도 이상을 얻을 수 있으므로 실리콘 면적의 손해를 감수해 가면서 래딕스-4구조를 채용할 필요가 없음을 알게 되었다. 물론 위 언급은 비터비 복호기의 동작 속도가 90MHz이하를 목표로 설계될 경우이며 더욱 빠른 동작을 요하는 제품의 설계에서는 래딕스-4구조가 다시 채용되어야 할 것이다.

2. 면 적

먼저 두 알고리즘을 이용한 합성 결과(레이아웃)를 각각 그림5와 그림6에 보인다[4]. 그림에서 메모리 블록과 로직 블록사이의 공간들은 라우팅 채널을 의미한다. 래딕스-2는 래딕스-4에 비해 우선 로직 블록의 면적을 볼 때 큰 감소를 얻었다는

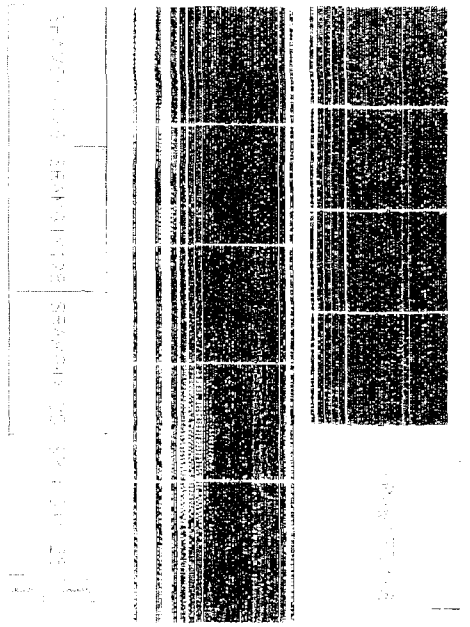

그림 5. 래딕스-4 알고리즘의 합성 결과

것을 안다. 이는 64개의 ACS어레이로 이루어진 PMC블록이 약 반 정도로 감소했기 때문이다. 또한 라우팅채 널이 비해 현저히 줄었음을 알 수 있는데 이는 우선 로직 블록의 감소, 래딕스-4용

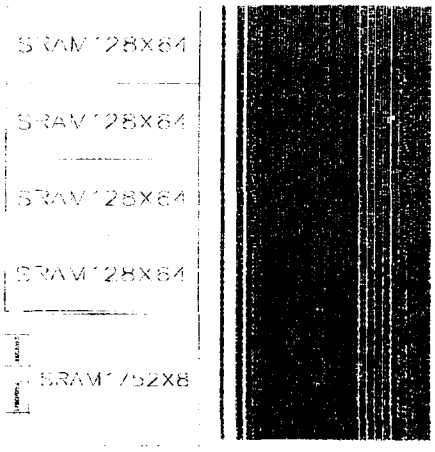

그림 6. 래딕스-2 알고리즘의 합성 결과

클럭의 제거, 그리고 Bus width의 감소로 인한 영향임을 알 수 있다. 로직 블록과 라우팅 채널에서의 면적 감소를 측정하여 표2에 보인다. 라우팅 채널이 차지하는 면적은 래딕스-4에 비해 약 54% 감소했으며, 로직 면적은 42%의 감소 효과를 얻을 수 있었다.

	Total Wire Length	Logic Area
래딕스-4	6178768 μm^2	5514678 μm^2
래딕스-2	2845838 μm^2	3201628 μm^2

표 2. 두 알고리즘의 면적 비교

3. 소비 전력

소비 전력에 관한 평가를 위해 본 연구에서는 EPIC사의 Powrmill tool을 이용하여 60MHz에서 동작할 때의 시뮬레이션 결과를 얻었다. 표3에 보인다.

	소비 전력
래딕스-4	0.264Watt = 80mA x 3.3V
래딕스-2	0.297Watt = 90mA x 3.3V

표 3. 60MHz에서의 소비 전력 비교

래딕스-4와 비교할 때 약 10% 정도의 소비전력 증가를 볼 수 있는데, 이는 래딕스-4에서 30MHz로 스위칭하던 블록(BMC, PMC, 역추적블록)들이 래딕스-2에선 60MHz로 스위칭하기 때문이다. 참고로 스위칭-게이트의 수를 비교해 보면 래딕스-2는 1791, 래딕스-4는 1951이다.

VI. 결 론

0.35μm 테크놀러지를 기반으로 비터비 디코더를 효율적으로 설계하기 위하여 래딕스-2와 래딕스-4 알고리즘을 각각 구현하여 그 결과를 비교, 분석하였다. 90MHz 이하의 속도를 요하는 제품에서는 래딕스-2 구조를 선택하는 것이 속도와 면적 그리고 소비 전력 면에서 합당하며, 90MHz 이상의 고속 비터비를 요하는 제품에서는 래딕스-4 구조를 선택해야 한다는 것을 알 수 있다.

참고 문헌

[1] T. K. Truong, Ming-Tang Shih, Irving S. Reed, E. H. Satorius, "A VLSI Design for a Trace-Back Viterbi Decoder", IEEE Trans., Commun. vol. 40, Mar. 1992, pp.616-624

[2] Shu Lin, Daniel J. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-hall, 1983

[3] Peter J. Black, "A 140-Mb/s, 32-State, Radix-4 Viterbi Decoder", IEEE Jour. of Solid-State Circuits, vol.27, No 12, pp. 1877-1885, DEC. 1992

[4] Samir. Palnitkar, "Verilog HDL : A Guide to Digital Design and Synthesis", Prentice-Hall, 1996