論文2001-38SD-4-4

설계영역 탐색을 이용한 최적의 비터비 복호기 자동생성기

(Automated Design of Optimal Viterbi Decoders Using Exploration of Design Space)

金 紀 甫 *, 金 鍾 兌 **

(Ki Bo Kim and Jong Tae Kim)

요 약

디지털 통신시스템의 오류정정을 위한 길쌈부호의 대표적인 복호방식인 비터비 복호기는 사용되는 시스템의 사양에 따라서 그리고 복호기의 복호 아키텍처에 따라서 다양한 방식으로 설계할 수 있다. 본 논문에서는 이러한 다양한 설계방법들 중에서 가장 효율적인 복호기의 설계구조를 결정해서 자동으로 원하는 사양에 맞는 비터비 복호기의 VHDL 모델을 생성해내는 자동생성기를 제시한다. 자동생성된 VHDL 모델을 이용하면 설계 초기단계에서 필요한 시간을 단축시킬 수 있다. 자동생성기는 설계영역 내에서 복호기의 설계크기와 복호속도를 비교해서 여러 가지 설계 아키텍처들 중에서 가장 최적인 것으로 판단되는 설계사양을 결정할 수 있다.

Abstract

Viterbi algorithm is widely used in digital communication system for FEC(forward error correction). Each communication systems based on the Viterbi algorithm use specific Viterbi decoder which has different code parameter values. Even if Viterbi decoder has the same code parameters, it can be varied by the design architecture adopted. We propose the parameterized VHDL model generator for the efficiency of the design. It makes it possible to achieve shorter design time and lower design cost. The model generator searches the design space available and finds out the optimal design point to generate a decoder model.

I. 서 론

디지털 통신시스템에서는 데이터 전송시 채널 장애로 인한 전송오류가 발생하게 되는데 이렇게 발생된 오류를 수신측에서 판단하여 올바른 데이터로 정정해 주어야 한다. 이렇게 오류정정이 가능하도록 보내고자 하는 정보를 변형시키는 과정을 오류정정 부호화(error correcting coding) 혹은 채널 코딩(Channel coding)이라고 한다. 이 오류정정 부호화에는 크게 블록 부호(block code)와 길쌈 부호(convolutional code)가 쓰인다.

길쌈 부호의 복호 방법으로는 최대 유사 복호 알고리즘(Maximum likelihood decoding algorithm)인 비터비 알고리즘[1]을 이용해서 복호를 한다.

길쌈부호는 생성다항식(generator polynomial), 구속장의 크기(constraint length), 부호율(code rate)등의 사양에 따라 달라진다. 비터비 복호기는 여러 종류의 통

* 正會員, 三成電子
(Samsung Electronics Co.)
** 正會員, 成均館大學校 電氣電子 및 컴퓨터工學部
(School of Electrical and Computer Engineering Sungkyunkwan University)
※ 이 연구는 성균관대학교의 1999학년도 성균학술연구비와 IDEC 지원으로 연구되었음.
接受日字:2000年8月21日, 수정완료일:2001年3月28日

신 시스템에서 사용되는데 각각의 시스템에서 요구하는 오류정정 능력에 따라 특정한 부호사양(구속장, 부호율, 생성다항식)을 갖는 비터비 복호기를 사용하게 된다. 따라서 해당 시스템에 맞는 비터비 복호기를 각각 설계할 필요가 있다. 부호사양이 같은 비터비 복호기라 할지라도 설계 아키텍쳐에 따라 서로 다른 설계가 가능하다. 즉 설계면적, 복호속도, 소모전력 등 관심을 두는 설계방향에 따라 해당목적에 최적화 된 아키텍쳐를 사용하여 복호기를 설계할 수 있다. 이렇게 다양한 설계들 중에서 해당되는 시스템의 부호사양에 적합하면서 설계자가 요구하는 디자인 크기와 복호속도를 만족하는 최적의 복호기를 빠른 시간내에 설계하는 것은 매우 힘든 일이다.

이제까지는 사용자의 입력값에 따라서 원하는 사양을 가진 비터비 복호기의 HDL 코드를 자동으로 생성해 내는 모델생성기에 대한 연구가 이루어져 왔다[2,3].

이상의 연구에서 제안된 생성기는 사용자로부터 필요한 부호 사양을 입력받아 해당되는 복호기를 생성해 내는 형태였다. 하지만 이러한 방법은 생성기 자체가 특정한 설계 아키텍쳐를 염두에 두고 작성된 것이므로 생성결과 얻게 되는 복호기 모델 또한 특정 아키텍쳐에 종속적이다. 또 특정 사양에 맞는 복호기를 빠르게 얻을 수는 있지만 생성된 복호기 모델을 논리 합성했을 때 어느 정도 하드웨어 크기를 갖고있으며 복호속도는 대략 어느 정도 되는지 즉시 알 수 없다. 더구나 생성된 복호기 모델이 과연 우리의 설계 제한사항(design constraint)을 만족하는지 혹은 가능한 최적의 설계 아키텍쳐인지 확인하기 어렵다. 이를 확인하기 위해서는 여러 가지 구조로 복호기 모델을 생성하고 합성하고 비교하는 일련의 작업이 필요하다.

본 논문에서는 사용자가 원하는 가장 효율적인 비터비 복호기를 VDHL 모델생성기에서 자동 판단하여 생성시켜주는 방법을 제안한다. 사용자는 생성기에 대한 입력값으로 해당시스템의 기본적인 부호사양과 함께 설계제한사항(복호기 크기, 복호속도 등)과 설계자의 요구사항 등을 입력한다. 생성기에서는 사용자로부터 입력받은 값들로부터 부호사양을 만족하면서 설계제한사항 내에서 구현 가능한 최적의 복호기 아키텍쳐를 결정하고 여기에 따라서 복호기의 VHDL 모델을 자동적으로 생성해낸다. 뿐만아니라 이렇게 자동생성된 복호기를 논리 합성했을 때 예상되는 크기와 복호시 필요한 클럭수를 알려준다. 이러한 예측값을 통해서 설계자는 합성이나 시뮬레이션을 하지 않고도 필요한 정보를 빠르게 얻을 수 있다.

Ⅱ. 복호기 구조

비터비 복호기는 그림 1과 같이 크게 BMC(branch metric calculation), ACS(add-compare-select unit), PMM(path metric memory), SMU(state metric unit) 등 네 부분으로 구분할 수 있다.

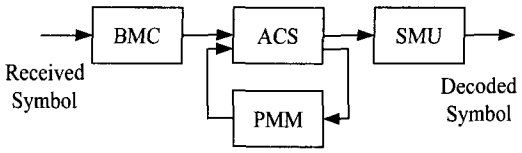

그림 1. 비터비 복호기의 블록도
Fig. 1. Block diagram of Viterbi decoder.

1. BMC부

BMC는 8단계(3비트 연판정) 혹은 16단계(4bit 연판정)등으로 양자화된 수신데이터와 격자도(trellis diagram)상에서 상태 천이시 발생하는 부호어사이의 유클리디언 거리값인 지로값(branch metric)을 구한다. 본 논문에서는 수신된 데이터를 offset binary format으로 사용하여 간단한 연산만으로 지로값을 계산했다. 부호율이 1/2일 때 발생 가능한 부호어는 모두 4가지(00, 01, 10, 11)존재하며 이들과 수신된 데이터와의 지로값도 역시 4가지가 존재한다. (X와, Y는 수신된 데이터로 각각 3비트 연판정 값이다)

$$BM00 = (X \oplus 0) + (Y \oplus 0) = X + Y$$
$$BM01 = (X \oplus 0) + (Y \oplus 1) = X + \overline{Y}$$
$$BM10 = (X \oplus 1) + (Y \oplus 0) = \overline{X} + Y$$
$$BM11 = (X \oplus 1) + (Y \oplus 1) = \overline{X} + \overline{Y}$$

이 연산을 통하여 간단한 논리회로만으로 지로값을 구할 수 있다. 그림 2에는 BMC 블록의 블록도를 보였다.

2. ACS부

ACS는 각 상태별로 BMC에서 구한 지로값과 이전 복호단계(decoding cycle)까지의 결과로 PMM에 누적된 경로값을 합하여 새로운 경로값을 구한 다음 이들 값을 비교해서 그 상태의 생존경로(Survivor Path)를 결정한다.

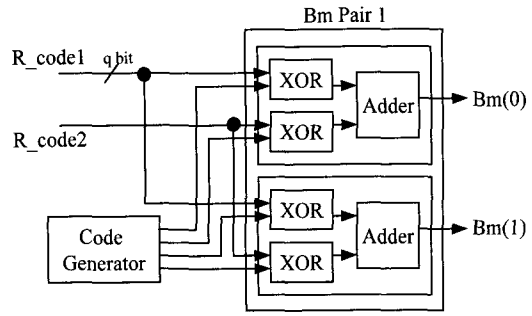

그림 2. 한 개 BM Pair의 블록다이어 그램
Fig. 2. Block diagram of 1 BM pair.

$$PM[i]_{(t+1)} = \underset{all}{MIN}\,(PM[k]_{(t)} + BM([k],[i]))$$

여기서 $PM[i]_{(t+1)}$는 (t+1) 복호단계에서 $[i]$ state가 갖는 PM값을 나타내고, $BM([k],[i])$는 $[k]$state에서 $[i]$state로 천이시 발생하는 지로값이다.

ACS는 나비구조[4]를 사용하여 구현하였다. 이 방법은 PMM에서 경로값을 읽어 들이는 횟수를 감소시키고 중복되는 연산을 생략함으로서 소모전력을 줄이는 방법이다.

ACS의 구성방식에 따라 완전병렬형과 부분 병렬형으로 구분된다. 완전병렬형은 복호속도가 빠르고 복호기의 구조가 간단하지만 구속장이 커질수록 복호기의 상태의 개수가 지수적으로 증가하여(2^{K-1}) 복호기의 크기가 매우 커진다. 따라서 구속장이 클 때는 복호기의 크기가 문제가 된다. 반면 부분병렬형은 일정수의 ACS를 여러 번 공유하여 사용하므로 복호기의 크기를 그 만큼 줄일 수 있으나 복호속도가 공유 횟수만큼 느려지고 복호기의 구조가 다소 복잡해지는 단점이 있다. 일반적으로 ACS를 공유하는 횟수가 커질수록 ACS 블럭의 크기는 작아지게 되고, 복호시 필요한 클럭 수는 증가하게 된다. 따라서 최적의 속도, 크기를 갖는 ACS의 공유 횟수를 찾아낼 필요가 있다. 복호기의 상태갯수(2^{K-1})와 ACS 공유횟수($iter$), ACS pair의 개수(p) 사이에는 다음과 같은 관계가 있다.

$$p = \frac{2^{K-1}}{2 \times iter}$$

ACS부는 그림 3과 같이 크게 3가지 블록으로 구성되어 있다.

ACS BLK은 p개의 ACS Pair로 이루어져있다. 한 개의 ACS Pair는 2개 state의 연산을 나비구조로 연산한다.

Routing부는 ACS BLK에서 발생한 추적정보 d (2^{K-1}bit)를 SMU에서의 순서에 맞도록 재배열하는 역할을 한다. 재배열된 정보(그림 3에서 a)는 SMU부로 전달된다. 이 블럭은 ACS에서 나비구조를 이용한 연산을 수행하기 때문에 달라지는 순서를 재배치하기 위한 것으로 모델 생성기에서 요구하는 다양한 사양에 대해서 융통성을 가질 수 있게 한다.

Min_PM부는 ACS에서 전달받은 생존경로(Survivor path)들 중에서 가장 작은 경로값을 갖는 상태를 결정하는 부분이다. 최소 경로값을 갖는 상태 정보는 SMU부로 전달된다. 또 ACS연산 결과로 얻어진 경로값들은 다음 복호단계를 위해 PMM에 저장된다.(그림 3의 c)

3. PMM부

PMM에서는 ACS부에서 전달받은 각 상태의 경로값을 다음 복호단계에서 사용될 수 있도록 저장한다. 완전병렬형 ACS구조일 때는 ACS연산이 모든 상태에 걸쳐서 동시에 이루어지므로 PMM은 $2^{K-1} \times bit\ width_{PM}$ 크기의 메모리소자 만으로 구성이 가능하다. 하지만 부분병렬형 ACS구조를 사용할 때는 ACS연산이 몇 개의 그룹으로 나뉘어져서 이루어지므로 연산중간에 임시로

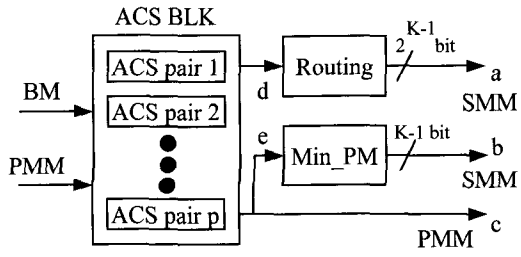

그림 3. ACS BLK의 블록도
Fig. 3. Block diagram of ACS BLK.

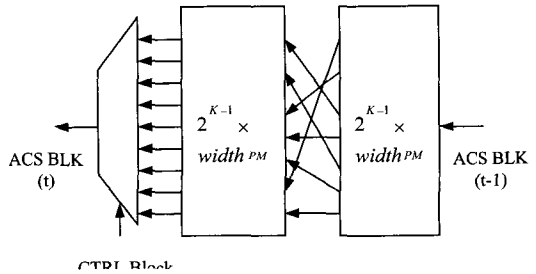

그림 4. PMM의 블록도
Fig. 4. Block diagram of PMM.

정보를 저장할 메모리와 이를 처리할 router가 추가적으로 필요하다. PMM의 블록도를 그림 4에 나타내었다.

경로값은 지로값을 계속 더해주므로 복호가 진행됨에 따라서 지속적으로 증가하게된다. 따라서 한정된 하드웨어에 경로값을 저장하기 위해서는 overflow가 발생하지 않도록 정규화하는 과정이 필요하다. 정규화하는 방법에는 크게 3가지 방법이 알려져 있는데[5] 본 논문에서는 [6]에서 제안한 Modulo 2n 방법을 사용하였다. 이 방법은 연산과정에서 overflow가 발생하더라도 연산의 결과에 영향을 미치지 않도록 하여 정규화에 따르는 연산량과 하드웨어 크기를 크게 줄일 수 있다. 특히 구속장의 크기가 클수록 그 효과는 더욱 크게 나타난다.

Modulo 2n 방법을 위해서는 적정크기 이상으로 PMM 비트수를 할당해야한다. PMM의 최적 비트수는 다음 식으로부터 결정된다.[3]

$$width_{PM} = 1 + \log_2(K \times n \times (2^q - 1))$$

여기서 K는 구속장, n는 부호율의 역수, q는 연판정 비트수 이다. 예를들어 구속장이 9이고, 부호율이 1/2, 3비트 연판정일 때 8비트로 PMM을 구성하면 된다.

4. 역추적부 (SMU)

SMU는 ACS에서 결정된 생존경로를 일정 복호깊이(decoding depth)만큼 기억하고 있다가 이 경로를 역추적하여 최종 복호값을 출력하는 부분이다.

SMU의 구조는 전체 복호기의 크기와 복호속도에 큰 영향을 미치는데 Traceback, Register Exchange, Systolic array[7], One pointer Algorithm[8]등과 같은 구조가 제안되어 있다. 각각의 구조는 복호시간, 복호기의 크기, 소모전력 등에 장단점을 가지고 있다. 본 논문에서는 위의 구조들 중에서 복호속도와 복호기의 크기사이에 가장 큰 차이를 보이는 Systolic Array 역추적 방식과 Traceback 역추적 방식 중에서 설계조건에 적합한 방식을 선택할 수 있도록 하였다.

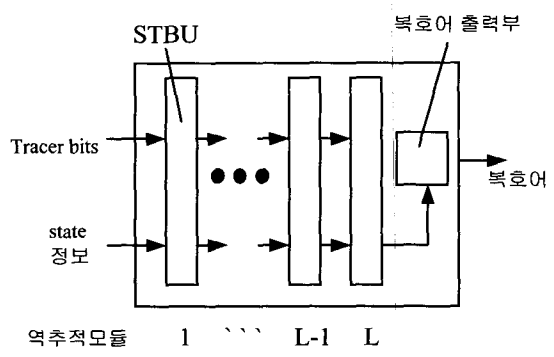

그림 5. 역추적부의 시스토릭 배열 구조
Fig. 5. Systolic array structure of Traceback unit.

1) Systolic 역추적방식

Systolic 역추적방식은 SMM부에 필요한 메모리크기가 Traceback 방식에 비해서 두배 가까이 크고 매 클럭마다 각 메모리에 저장된 값이 연결된 다음 메모리로 천이할 때 발생되는 전력소모가 크다는 단점이 있으나 복호속도가 빠르다. 복호시작 후 초기 지연시간($2L$)이 지난 후부터는 매 클럭마다 한 개의 복호어를 얻을 수 있다.

Systolic array 구조의 SMU부는 그림 5과 같이 직렬연결된 역추적 모듈(STBU)들로 구성되어 있다. 이 역추적 모듈의 연결 개수는 역추적 깊이(L)와 같은데 일반적으로 역추적 깊이의 크기가 클수록 복호성능이 좋아진다. 복호성능과 하드웨어 구현시의 효율을 동시에 고려할 때 보통 구속장의 다섯배 정도가 많이 사용된다.[9]

역추적깊이가 L일 때 역추적부의 전체 메모리 소자는 ($2L$ - 1)개가 필요하고 상태천이 메모리는 총 L개가 필요하다.

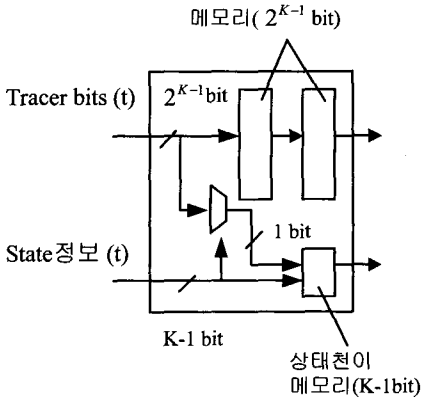

그림 6. 역추적모듈(STBU)의 블록도
Fig. 6. Block diagram of STBU.

2) Traceback 방식

Traceback 방식은 앞에서 언급한 Systolic 역추적복호 방식보다 필요한 메모리가 작다. 즉 Systolic 역추적 방식에서는 복호깊이 만큼의 상태정보를 저장하기 위

해서 $2^{K-1}\times(2L-1)$비트의 메모리가 필요했으나 Traceback 방식에서는 $2^{K-1}\times L$비트의 메모리가 필요하다. 또 매 클럭마다 메모리의 값이 차례로 다름 메모리로 전달되는 Systolic 역추적방식과는 달리 소모전력이 작다. Traceback 방식은 매 복호어를 얻기위해서 항상 L 클럭이 필요하다. 따라서 복호 속도가 Systolic 방식보다 상당히 느리다. 그림 7에 Traceback 방식 SMU의 블록도를 보였다.

Ⅲ. Modeling생성기 작성

비터비 복호기의 VHDL모델 생성기는 C언어를 이용하여 작성되었다. 생성기는 먼저 사용자로부터 입력받은 정보로부터 복호기 모델생성에 필요한 여러 세부 변수값들을 연산한 뒤 이 값들을 이용해서 Ⅱ장에서 설명된 각 블록들을 생성해낸다.

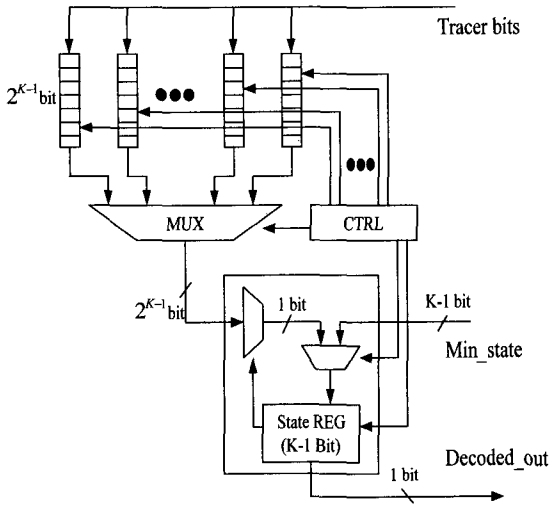

그림 7. Traceback 방식 역추적부의 블록도
Fig. 7. Block diagram of traceback block.

각 블록들은 독립적인 생성이 가능하도록 모듈화하여 작성하였다. 즉 한 블록을 생성하는 모듈은 다른 블록의 형태와 무관하게, 해당되는 입력값에 의해서만 생성작업을 수행한다. 이렇게 하면 작성된 프로그램의 검증이 쉬워지고 새로운 설계 아키텍쳐를 모듈화 하여 생성기에 추가하기가 쉬워진다.

비터비 복호기의 모델생성기는 다음과 같은 과정으로 작성되었다.

1. 가장 간단한 사양을 갖는 비터비 복호기의 VHDL 설계 및 검증

이 과정에서는 모델 생성기에 의해서 생성되길 바라는 VHDL 모델을 미리 직접 설계한다. 이 단계가 필요한 이유는 생성기의 구현이 끝난 다음 생성기에 의해서 생성된 VHDL 모델의 동작검증에 소요되는 시간을 단축시키기 위해서이다. 또, 동작검증된 복호기의 원형(prototype)을 기초로 생성기의 작성에 필요한 세부 블록의 분할작업(partitioning)을 할 수 있다. 복호기의 VHDL 설계과정을 통해서 복호기의 사양변수에 따라 달라지는 복호기 세부 구조변화를 판단해낼 수 있는데 이는 생성기의 작성을 위해 필요한 기초정보다.

2. 비터비 복호기의 사양과 구조를 결정짓는 변수의 선정

과정 1을 통해서 비터비 복호기의 설계사양을 결정짓는 변수를 찾아내고 상호연관관계를 규정지어야한다. 비터비 복호기의 사양을 결정짓는 변수들은 다음과 같다.

1) 길쌈부호기의 사양에 따른 변수
   - 구속장의 크기 : K
   - 생성다항식(Generator Polynomial) : G
   - 부호율 : $1/n$
2) 복호기의 성능을 결정 짓는 변수
   - 연판정 비트수 : q
   - 복호깊이(decoding depth) : L
   - ACS쌍의 공유횟수 : $iter$
   - ACS쌍의 수 : p
   - 복호 아키텍쳐의 선택 : $archi$
3) 변수간의 관계에 의해서 결정되는 값
   - PMM메모리의 bit수 : $width_{PM}$
   - 전체상태의 개수 : 2^{K-1}

3. 각 모듈생성기 작성

Ⅱ에서 설명한 복호기의 각 블록들을 생성하는 모듈을 작성한다. 즉, BMC, ACS, PMM, SMM(Systolic), SMM(Traceback), CTRL(Control unit)모듈 등과 이 모듈들을 통합하는 Top모듈 그리고 VHDL 내에서 자료형을 정의하는 Package문을 만들어주는 모듈등을 작성한다.

4. 작성된 모델생성기의 검증

각 모듈들을 통합하여 하나의 완전한 모델생성기를 작

표 1. 입력 변수
Table 1. Input parameter.

입력	의 미
max_area	허용 가능한 최대크기(gate count)
max_clock	허용 가능한 심벌당 최대복호클럭수
AT_ratio	Area와 Time간의 비용비율

성한 다음 이 모델생성기를 통해서 생성된 비터비 복호기의 동작을 검증한다. 가장 많은 시간을 요구하는 과정이다. 생성기는 총 9개의 C 프로그램파일과 1개의 헤더파일로 작성되었으며 ANSI C로 작성하였기 때문에 운영체제환경에 무관하게 컴파일과 실행이 가능하다.

모델생성기는 비터비 복호기의 모델과 함께 이를 Functional Level에서 검증할 수 있도록 테스트 벡터를 제공한다. 사용자가 전송할 정보와 채널환경을 지정해 주면 모델생성기에서는 해당되는 테스트 벡터를 자동으로 생성해서 생성된 비터비 복호기의 동작을 검증할 수 있다.

Ⅳ. 최적 설계 구조 결정

Ⅲ에서 작성된 생성기는 사용자가 사전에 이미 결정해서 입력한 설계 사양 그대로 복호기 모델을 생성한다. 사용자가 이미 복호기의 성능과 구현에 있어서 필요한 여러 가지 사항에 대해 충분히 고려를 해서 결정한 입력 값이라면 문제가 없지만 설계자가 복호기의 세부사항에 대해서 잘 알지 못하는 경우에는 -예를 들어 Soft IP로 복호기 블럭을 구입해서 사용하고자 하는 경우- 생성기로 얻은 복호기 모델이 최적의 설계를 갖는다고 보장할 수 없다.

본 연구에서 작성된 모델생성기는 사용자가 원하는 사양을 직접 입력할 수도 있고 복호기가 최적의 사양을 결정하게 할 수도 있다. 이를 위해서 Ⅲ에서 작성된 생성기에 최적의 복호기 설계 구조를 판단하게 하는 연산모듈을 추가해야 한다. 또 사용자로부터 표 1와 같은 추가 정보를 입력받아야 한다.

생성기는 목적하는 시스템에 따라 이미 결정되어 있는 부호사양(구속장, 부호율, 생성다항식등)을 입력받고 반드시 지켜져야 할 설계제한사항 (max_area, max_clock) 그리고 복호기크기와 복호속도간 비용비율(AT_ratio)등을 사용자로부터 입력받는다.

여기서 AT_ratio는 복호기 크기의 감소에 따른 이익과 복호시간 단축에 따른 이익 사이의 비율이다. 즉 AT_ratio가 크다는 것은 크기의 감소에서 얻은 이득이 복호속도의 저하에 따른 손해보다 더 크다는 것을 나타낸다. 따라서 AT_ratio는 사용자가 설계시 복호속도와 복호기의 크기 사이에 어느 정도의 비중을 두고 설계하고자 하는지 나타내는 지표이다.

표 1와 같은 설계제한사항을 입력받은 후 생성기는 생성기로 만들어 낼 수 있는 모든 설계 영역, 설계 아키텍쳐에 대해서 복호 속도와 복호기의 크기를 예측해 낸다. 그리고 예측해낸 값들이 설계 제한사항을 만족하는지 확인하여 설계 제한사항을 만족하지 못하는 설계점들은 설계 영역에서 제거한다. 마지막으로 설계제한사항을 만족하는 설계점들 모두의 복호속도, 크기를 비교해서 이들 중에서 가능 최적의 설계점을 찾아낸다.

이렇게 찾은 최적 설계점의 설계사양과 아키텍쳐 정보는 Ⅲ장에서 설명한 생성기로 전달되어 이 정보에 맞는 복호기의 모델을 생성한다.

이 같이 설계 예측값에 의한 비교방법으로 최적의 설계점을 찾기 위해서는 생성기는 설계 가능한 영역 내에서 생성될 수 있는 모든 복호기의 크기와 속도에 대한 정보를 미리 가지고 있어야 한다. 즉 미리 발생 가능한 모든 경우의 디자인을 생성한 다음 이를 논리합성해서 database로 구축해 놓아야 한다. 하지만 모든 설계 영역에 대해서 논리합성을 수행하는 것은 많은 시간과 노력을 필요로 하는 일이다. 따라서 우리는 다음과 같은 방법으로 이러한 문제를 간략화 하였다. 먼저 복호기의 각 블록들을 두 부류로 분류한다. 즉 입력값의 변화에 따라서 (a) 같은 크기의 블록이 단지 사용되는 갯수가 변화하는 부류와 (b) 블록 수는 같지만 블록의 크기만 입력값에 비례해서 변화하는 부류로 나눈다.

(a)에 해당하는 블록들로는 ACS나, PMM, SMU 등을 예로 들 수 있으며 (b)에 해당하는 블록들은 BMC, MIN_PM, Router, CTRL, MUX 등을 예로 들 수 있다. (a) 에 해당되는 블록들은 최소 단위의 블록만 합성해서 구한 크기 값에 블록이 사용되는 횟수를 곱해서 대략적인 전체 크기를 예측할 수 있다. 한편 (b)에 해당되는 블록은 몇 가지의 입력변수에 대해서 각각 합성된 값을 구한 뒤 나머지의 입력변수 값에 대해서는 보간법과 같은 수치 해석적인 방법으로 추정할 수 있다. 최종적으로 이렇게 구한 모든 블록의 크기를 합하면

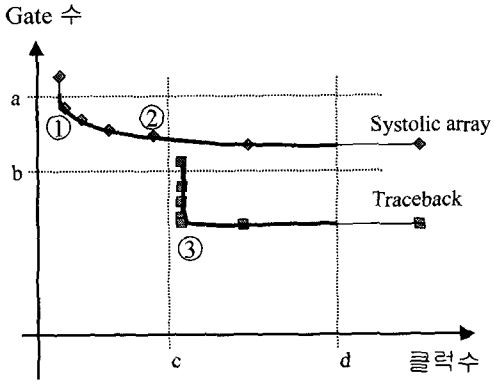

그림 8. 구속장 9인 비터비 복호기의 설계영역
Fig. 8. *Design space of Viterbi decoder with constraint length 9.*

예상되는 디자인의 전체 크기를 예측해 낼 수 있다. 표 2에는 몇 가지 설계 점들에 대해서 생성기가 예측한 값과 합성기로 실제 합성해서 산출해 낸 결과를 비교해서 보였다.

구속장이(K)=9, 부호율=1/2, 복호깊이(L)=45일 때 생성기로 설계 가능한 모든 설계영역과 복호속도, 복호기의 크기의 예측값를 표 3에서 비교해 보였다.

그림 8에는 표 3에서 보인 모든 설계 영역을 나타내었다. 여기서 원점에 가까운 점들일 수록 복호속도가 빠르고 크기가 작으므로 더 좋은 설계점이며 원점에서 멀리 떨어질수록 반대의 경우로 피해야할 설계점이다.

사용자로부터 받은 설계제한사항이 그림 8의 a, c일 때를 살펴보자. 이때는 복호 속도가 느린 Traceback 방식은 설계영역에서 제외되고 Systolic 방식만이 선택된다. 선택된 설계영역내에서 사용자가 보다 빠른 복호속도를 원하였다면 설계점 ①이 선택되며 설계영역내에서 보다 작은 크기를 원하였다면 설계점 ②가 선택될 것이다.

사용자의 설계제한사항의 그림 8의 b, d였을 경우는 반대로 크기가 상대적으로 큰 systolic array 아키텍처는 설계영역에서 제외된다. 그리고 Traceback 아키텍처중에서 크기가 가장 작으면서 속도도 가장빠른 설계점 ③이 선택된다.

설계제한사항이 그림 8의 a, d였을 경우를 생각해보자. 이때는 설계영역내에 두 가지 설계 아키텍처가 모두 선택 가능하다. 이때도 마찬가지로 사용자가 복호속도에 관심을 둔다면 설계점 ①이 최적 설계구조로 선택될 것이며 복호기의 크기를 최소화하기 원한다면 설계점 ③이 선택될 것이다.

표 2. 생성기가 예측한 크기와 실제 합성결과의 비교 (구속장 : 9)
Table 2. Decoder size by estimation of generator and by synthesis (K : 9)
(단위 : 게이트 수)

구분			생성기 예측	합성 결과
ACS pair 갯수	복호 클럭수	SMU구조		
4	45	traceback	228,193	209,977
8	45	traceback	209,480	218,537
4	32	systolic	354,995	327,096

V. 결 론

본 연구에서는 비터비 복호기의 자동생성기를 제안하였다. 본 연구에서 제안된 자동생성기는 사용자로부터 복호기의 사양을 입력받아서 해당되는 비터비 복호기의 VHDL 모델을 자동으로 생성시켜준다.

표 3. 구속장 9일 때 가능한 설계영역비교
Table 3. Possible design space of Viterbi decoder with constraint length 9.
(단위 : 게이트 수)

ACS pair	clock		size(gate)	
	systolic	trackback	systolic	traceback
1	128	128	319807	183005
2	64	64	318774	181972
4	32	45	321386	184584
8	16	45	327186	190384
16	8	45	346282	209480
32	4	45	364995	228193
64	2	45	411514	274712

생성기는 고려할 필요가 있는 설계지표(하드웨어의 크기, 복호기의 복호속도)들간의 Tradeoffs를 비교하여 사용자가 최선이라고 판단하는 복호기의 설계 아키텍처를 자동으로 선택하여 모델을 생성한다. 이러한 방법은 반복적인 설계와 비교과정에 필요한 시간을 단축시킬 수 있고 좀더 효과적인 모델을 얻을 수 있다는 잇점이 있다. 이상과 같은 자동화 설계기법을 좀더 확장시킨다면 본 연구에서 제시한 두 가지 설계 아키텍처

(Systolic, Traceback) 뿐만 아니라 현재까지 발표된 모든 가능한 설계 기법들 중에서 최적인 모델을 얻을 수 있을 것이다. 또 사용자가 원하는 설계지표를 좀더 확장한다면 복호기의 크기와, 동작속도 뿐만 아니라 전력 소모량과 같은 보다 다양한 관점에서 설계를 비교, 선택할 수 있을 것이다.

참 고 문 헌

[1] G. D. Forney, "The Viterbi Algorithm", Proc. of the IEEE, Vol. 61, no.3, pp. 268~278, March 1973.

[2] 공명석, 배성일, 김재석 "사양변수를 이용한 비터비 복호기의 자동설계", 전자공학회논문집 제36권 6편제1호, pp. 1~10, 1999년 1월호.

[3] R. Burger, G. Cesana, M. Paolini, M. Turolla, S. Vercelli, "A Fully Synthesizable Parameterized Viterbi Decoder", in IEEE Custom Integrated Circuits Conference, pp. 27~30. 1999.

[4] I. Kang and A. N. Willson, Jr., "Low-Power Viterbi Decoder for CDMA Mobil Terminals", IEEE Journal of Solid-State Circuits, Vol. 33, No. 3, pp. 473~482 March 1998.

[5] I. M. Onyszchuk, K. M. Cheung, O. Collins, "Quantization Loss in Convolutional Decoding", in IEEE. Trans. on COMM. Vol. 41, No. 2, pp. 261~265, Feb. 1993.

[6] A. P. Hekstra, "An Alternative to Metric Rescaling in Viterbi Decoders", IEEE Trans. on Comm., Vol. 37, No. 11, November. 1989.

[7] T.K. Truong, M.T. Shih, I.S. Reed, E.H. Satorius, "A VLSI Design for a Trace-Back Viterbi Decoder", IEEE Trans. Comm., vol. 40, No. 3 pp. 616~624, March 1992.

[8] Gennady Feygin, P. G. Gulak "Architectural Tradeoffs for Survivor Sequence Memory Management in Viterbi Decoders", in IEEE Trans. Comm., Vol. 41, No. 3, pp. 425~429, March, 1993.

[9] F. Hemmati and D. Costello, "Truncation Error Probability in Viterbi Decoding", IEEE Trans. on Comm., pp. 530~532, May 1977.

저 자 소

金 紀 甫(正會員)

1999년 성균관대학교 전기공학과 학사. 2001년 성균관대학교 전기전자 및 컴퓨터공학과 석사. 현재 삼성전자 디지털 미디어 연구소 재직. 주 관심분야는 디지털 이동통신시템 및 방송기술, VLSI 및 SoC 설계

金 鍾 兌(正會員)

1982년 성균관대학교 전자공학과 학사. 1992년 University of Calif, Irvine 전기 및 컴퓨터공학과 박사. 1991년~1993년 The Aerospace Corp. 연구원. 1993년~1995년 전북대학교 컴퓨터공학과 교수. 1995년~현재 성균관대학교 전기전자 및 컴퓨터공학부 교수. 주 관심분야는 VLSI CAD 및 ASIC 설계, 컴퓨터 구조 및 Embedded System설계