고속데이터 처리를 위한 터보 인코더 설계

정덕영, 최덕영, 손승일, 이우섭
한신대학교 정보통신학과

A Design of Turbo-encoder for High-speed Applications.

Duck Young Jung, Dug Young Choi, Seung Il Sonh, Ou Seb Lee
Dept. of Information and Communication HanShin University
E-mail : coolduck@daum.net

요 약

본 연구에서는 IMT-2000방식 중 WCDMA(Wideband Code Division Multiple Access)방식의 터보 코드에 대해 연구하였으며 터보 코드에 사용하는 인터리버 중 신호대 잡음비 성능이 가장 좋은 GF 인터리버와 터보 코드의 단점 중에 하나인 처리하는 동안 걸리는 지연 시간을 보완하기 위하여 핑퐁 램을 사용하였다. 핑퐁 램을 사용함으로써 최초에 소스 데이터 입력 때 만 지연이 생기고 그 이후에는 연속적으로 데이터를 출력 할 수 있는 장점을 얻었다. 그리고 C언어를 이용한 최적화된 시뮬레이션과 터보 코드의 성능평가를 통한 최적화를 실시하였다. 그리고 최적화된 정보를 바탕으로 터보코드를 VHDL언어를 이용하여 설계하고, Xilinx Vertex 1000E(XCV1000E)-PQ240칩을 이용한 테스트 보드에 다운로드 후 PCI인터페이스를 통한 시뮬레이션을 구현하였다.

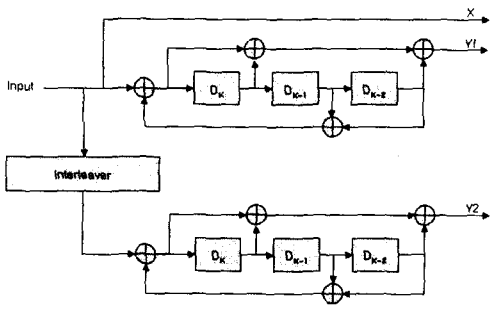
1. 서 론

요즘의 정보화 시대에 통신이 발달하고 이동통신이 보편화될수록 통신 과정에서 발생하는 에러를 최소화하고자 하는 요구는 커지고 있는 가운데, IMT-2000을 비롯한 차세대 통신시스템에서도 고속의 멀티미디어 데이터의 신뢰성 있는 전송을 요구하며 고속 데이터의 신뢰성을 높이기 위해서 강력한 채널 코딩 방식이 요구되고 있다.[1] 이런 요구로 인해 오류정정부호 중 최근 Berrou[2] 등에 의해 발표된 것으로서 비교적 간단한 구조를 가지면서도 성능이 매우 우수한 터보 부호가 제시 되었고, 인터리버의 크기가 크고 반복 복호가 충분히 수행되었을 때 비트 에러율 관점에서 샤논 한계에 근접하는 우수한 성능을 나타내어 ITU 등에서 IMT-2000 차세대 이동통신의 고속 데이터 전송용 채널 코드의 표준으로 채택되었다. 이에 따라, 국내외 많은 업체와 대학에서 사용면적이 적고 전력 소모가 작으며 처리 속도가 빠른 동시에 실시간 처리가 가능한 터보 부호기와 복호기를 구현하기 위한 연구가 진행되고 있다.[3]

본 연구에서는 IMT-2000방식 중 WCDMA (Wideband Code Division Multiple Access)방식의 터보코드에 대해 연구하였으며 터보 코드에 사용하는 인터리버 중 신호잡음비 성능이 가장 좋은 GF 인터리버와 터보 코드의 단점 중에 하나인 처리하는 동안 걸리는 지연 시간을 보안하기 위하여 핑퐁 램을 사용하였다. 그리고 C언어를 이용한 최적화된 시뮬레이션과 터보 코드의 성능평가를 통한 최적화를 실시하였다. 그리고 최적화된 정보를 바탕으로 터보코드를 VHDL언어를 이용하여 설계하고, Xilinx Vertex 1000E(XCV1000E)-PQ240칩을 이용한 테스트 보드에 다운로드 후 PCI인터페이스를 통한 시뮬레이션을 구현하였다.

2. 터보 코드 기능

터보 코드의 인코더에 기능은 두 개의 길쌈 부호기와 한 개의 인터리버를 연결한 형태로써 부호화는 하나의 systematic 정보와 두 개의 Recursive Systemaic 길쌈부호기의 출력을 통하여 부호화되며 길쌈 부호기 사이에 인터리버를 두어 서로 다른 입력 정보를 포함할 수 있도록 한다. 정보비트는 터보 부호기 구조가 systematic이므로 채널을 통하여 그대로 전송되고 또한 2개의 구성 부호기에 의해서 원래 정보가 부호화되어 첫 번째 부호기는 정보비트의 원래 입력 순서에 따라 부호화 과정을 수행하며 두 번째 부호기는 인터리버에 의하여 순서가 재배열된 정보 비트의 순서에 따라 부호화 과정을 수행한다.



[그림 1] 터보 코드 구조

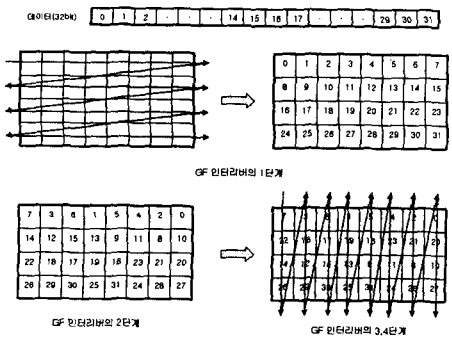
[그림 1]은 8상태(구속장 4)를 갖는 PCCC(Parallel Concatenated Convolution Code) 터보 부호기를 보여주고 있다. 그림에서 각 구성부호기의 전달함수는 [식1]과 같다.

$$G(D) = [1, \frac{g_1(D)}{g_0(D)}] \qquad [식1]$$

위 [식1]에서 g0(D) = 1 + D2 + D3 으로서 Feedback polynomial을 나타내고 있으며, $g_1(D) = 1 + D + D^3$은 Feedforward polynomial 이다. 모든 PCCC의 shift register의 초기값은 0이다. PCCC의 출력은 시스템에서 사용하고자하는 부호율에 맞추어주기 위하여 펑처링 과정을 거치게 된다. 부호율이 1/3이면 펑처링 되지 않으며 출력이 X(0), Y(0), Y'(0), X(1), Y(1), Y'(1).... 가 되고, 인코더의 부호율이 1/2일 때에는 펑처링 되어 X(0), Y(0), X(1), Y'(1), X(2), Y(2)... 가 된다.[5][6]

3. GF 인터리버

인터리버에는 블록, 랜덤, Mother, GF 등이 존재한다. 그러나, 현재 연구에서 채택한 인터리버는 WCDMA 방식에 사용되는 표준으로 논의되었던 방식중의 하나로 4×8의 크기를 가지는 GF인터리버 방식이며, 처리 방식은 [그림 2]와 같다.



[그림 2] GF 인터리버

GF 인터리버는 [그림 3]과 같이 총 4단계로 행하여 지는데 1단계는 행단위로 인터리버를 채우고, 2단계는 규칙에 따라 각 열 I (i =0, 1, 2, 3)에 대하여 순열을 [식1]과 [식2]와 같이 바꾸어 재배열한다.

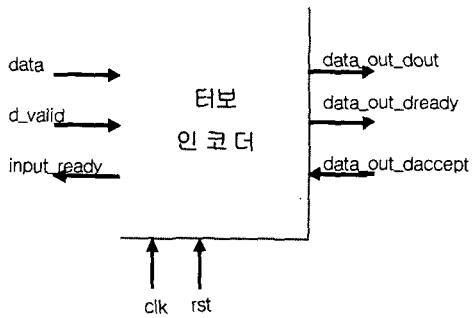
$$j \leftarrow \log a^{ib}(a^{i0} + a^{j}) \quad for \quad j = 0,1,2,3,4,5,6 \qquad [식2]$$
$$j \leftarrow \log a^{ib}(a^{i0}) \quad for \quad j = 7 \qquad [식3]$$

3단계는 행에 대한 인덱스 값 {00, 01, 10, 11 }에 대하여 비트 리버설 방법에 따라 재배치하고, 마지막으로 4단계는 인터리버 내용을 열 단위로 읽는다.[7]

4. 터보 인코더 설계

4.1 인코더의 인터페이스 신호

[그림 3]는 인코더의 인터페이스 신호이다.



[그림 3] 터보 인코더의 인터페이스 신호

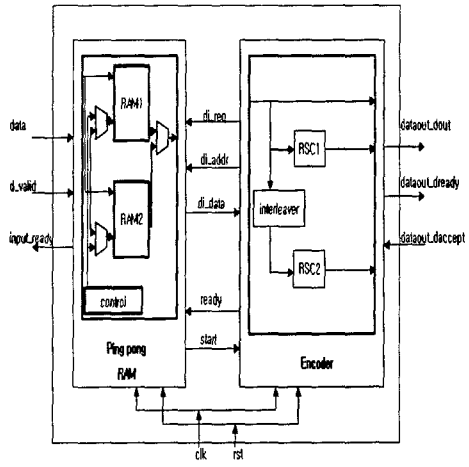
리셋이 활성화 되고 input_ready가 활성화 되면 d_valid가 활성화 되면서 32비트 크기의 데이터를 읽어 들인다. 그리고 data_out_daccept가 활성화 되면 터보 인코더에서 처리된 데어터가 data_out_dready가 활성화 되면서 1/3으로 data_out_dout로 출력된다.

여기서 input_ready는 내부 핑퐁램의 컨트롤에 의존하며 비활성화 되었을 때는 데이터를 읽어 들이지 못한다.

4.2 터보 인코더의 블록도

[그림4]는 ping-pong RAM을 이용한 터보 인코더의 블록 다이어그램이다. 처음에는 input_ready가 리셋 후 활성화 되고 data_valid가 활성화 되면서 32비트의 데이터가 핑퐁램으로 들어온다. 핑퐁램의 램 1과 2에 데이터가 모두 들어오면 input_ready 신호가 비활성화 된다. 터보코드에서는 dataout_daccept를 받은 후 ready 신호를 핑퐁램으로 보내면 핑퐁램에서 데이터 유무에 따라 start 신호를 보내고 신호를 받은 터보코드는

di_addr과 di_req신호를 보내서 32비트의 데이터를 받아와서 처리하게 된다. 터보코드가 데이터를 처리한후에 dataout_dready 신호를 활성화 하면서 처리된 데이터를 보낸다.



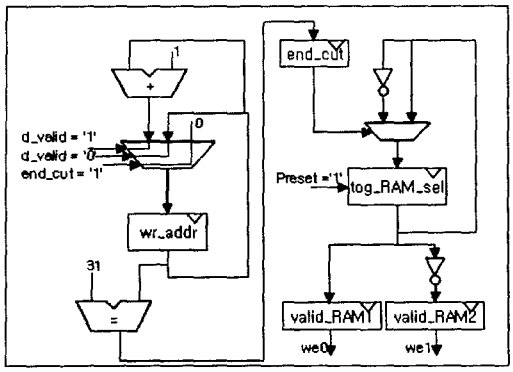
[그림 4] 터보 인코더 블록 다이어그램

4.3 핑퐁 램

터보 코드의 GF 인터리버는 데이터를 재배열 할 때 32비트의 데이터가 모두 있어야만 한다. 이와 같은 GF 인터리버의 특성상 매 데이터 처리마다 32비트씩 저장을 하여야 하기에 데이터 처리를 한 번 할 때마다 32클럭의 지연이 생긴다. 뿐만 아니라 인터리버에서 재배열된 데이터와 소스 데이터가 그대로 나가는 출력이 동시에 이루어져야 하므로 더 많은 지연이 생기게 된다. 이를 보완하기 위하여 핑퐁램을 사용하였으며 핑퐁램의 사용으로 인하여 GF 인터리버의 단점인 32비트로 데이터를 고정하는 것과 시간문제를 쉽게 해결 할 수 있다.

4.3.1 contorl 구성도

[그림 5]는 핑퐁램의 contorl 구성도를 보여준다. d_valid가 활성화 되면 wr_addr이 카운터가 되면서 처음 32비트의 데이터는 tog_RAM_sel의 초기값에 의해 we0를 활성화 시켜서 32비트의 데이터가 핑퐁램1에 들어오게 된다. 여기서 데이터가 들어오지 않을 경우에는 wr_addr은 계속 자신의 값을 가지게 되고 들어 올 경우에만 wr_addr이 1씩 증가한다. wr_addr이 31이 될 경우 end_cut가 셋팅이 되면서 핑퐁램1에 데이터가 모두 입력 되었다는 것을 알려준다. 그러면 end_cut와 현재의 tog_RAM_sel에 의해 다음 데이터를 저장 할 수 있게 핑퐁램2의 we1을 활성화 시킴으로 다음 32비트의 데이터를 저장하게 된다.

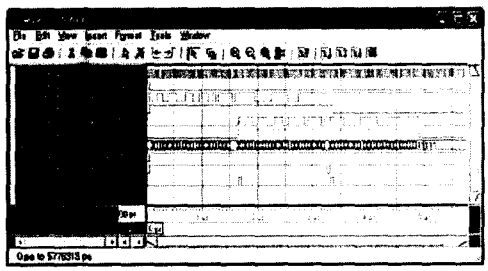


[그림 5] contorl 구성도

5. 출력파형

5.1 핑퐁램

[그림 6]은 핑퐁램의 출력파형을 보여준다. 리셋이 활성화 되고 d_valid가 활성화 되면 데이터를 읽어 들여서 핑퐁램1에 32비트의 데이터를 저장하고 다음 32비트의 데이터는 핑퐁램2에 저장되게 된다. 32비트의 데이터가 핑퐁램 한곳에 저장되고 난 후에 start가 활성화 되고 ready가 활성화 되면 인코더에서 요청신호가 활성화 될 때 저장 되었던 데이터가 출력된다. 그후에 비어있는 핑퐁램에는 다음 32비트가 저장되고 32비트의 데이터가 출력되면 다시 비어있는 핑퐁램에 데이터가 저장된다.



[그림 6] 핑퐁램

5.2 GF 인터리버

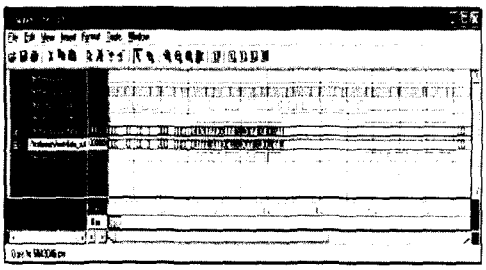
3절에서 GF 인터리버에서 설명한 것과 같이 [그림 7]는 GF 인터리버에서 처리된 파형을 보여준다.

리셋이 활성화 되고 we가 활성화 되면 한 클럭에 하나씩 데이터를 읽어 들여 32비트 크기의 데이터를 읽어 들인다. 읽어 들인 데이터는 GF 인터리버의 4단계를 통한 재배열을 한 후에 we가 활성화 되지 않을 때 한 클럭에 하나씩 재배열된 데이터가 출력으로 나가는 파형이다.

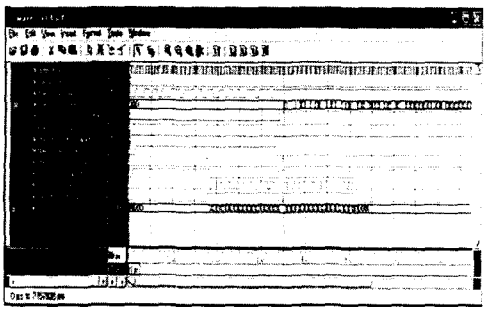
5.3 시뮬레이션 파형

[그림8]은 핑퐁 램을 사용한 터보 인코더의 파형을 보여준다. 리셋이 되고 난후에 input_ready가

활성화가 되고 d_valid가 활성화 될 때 32비트 크기의 데이터를 읽어서 핑퐁램1에 저장하고 다음 데이터는 핑퐁램2에 저장시킨다. 핑퐁램에 저장이 모두 되면 start를 활성화 시키고 data_out_dready 신호가 활성화 되지 않으면 준비 됐다는 신호와 데이터를 보내라는 요청신호를 보내면 핑퐁램에서 데이터를 보내게 되고, 인코더에서 데이터를 모두 처리하면 data_out_dready가 활성화 되고 input _ready가 활성화 되지 않을때 1/3으로 데이터를 처리하여 출력하게 된다. 32비트의 데이터가 출력되면 핑퐁램1과 핑퐁램2중 비어있는 한 곳에 다음 32비트의 데이터 저장되고 다시 32비트의 데이터가 출력되면 비어있는 핑퐁램에 저장된다.



[그림 7] GF 인터리버 출력 파형



[그림 8] 핑퐁램을 사용한 터보 인코더

7. 결 론

터보인코더는 두 개의 길쌈 부호기와 한 개의 인터리버를 연결한 형태로써 부호화는 하나의 systematic 정보와 두 개의 Recursive Systemaic 길쌈 부호기의 출력을 통하여 부호화되며 길쌈 부호기 사이에 인터리버를 두어 서로 다른 입력 정보를 포함할 수 있도록 한다. 정보비트는 터보 부호기 구조가 systematic이므로 채널을 통하여 그대로 전송되고 또한 2개의 구성 부호기에 의해서 원래 정보가 부호화되어 첫 번째 부호기는 정보 정보비트의 원래 입력 순서에 따라 부호화 과정을 수행하며 두 번째 부호기는 인터리버에 의하여 순서가 재배열된 정보 비트의 순서에 따라 부호화 과정을 수행한다.

본 논문에서는 신호 대 잡음비 성능이 가장 좋은 GF 인터리버와 터보인코더의 단점인 처리속도를 높이기 위해서 핑퐁램을 적용시켜 구현함으로써 최초에 소스 데이터 입력 때 만 지연이 생기고 그 이후에는 연속적으로 데이터를 출력 할 수 있는 장점을 얻었다.

향후 연구 방향은 4*8의 크기에만 국한 하지 않고 8*16, 8*32부터 32*128의 일반적인 크기에 적용시키겠다.

참고문헌

[1] D.Divsalar and F.Pollara, "Turbo Codes for PCS Applications", Proc. ICC'95, June. 1995.

[2] C. Berrou, A. Glavieux, and P. Thitimas jshima, "Near Shannon limit error-correcting coding and decoding:Turbo codes (1)," Proc. IEEE Int. Conf. Commun., pp. 1064-70, May 1993.

[3] Claude Berrou, Alain Glavieux, "Near Optimum Error Correcting Coding And Decoding: Turbo-Codes", IEEE Trans. On Communications, vol.44, no.10, pp.1261 -1271, Oct. 1996.

[4] http://www.xilinx.com/ipcenter/turbo_convo_lounge/index.htm#purchase

[5] 진익수외 3인 " 터보 부호의 인터리버 분석" ETRI 주간 기술동향, 통권 905호, 1999.7

[6] Qualcomm, "A New Low-Complexity Turbo Code Interleaver Employing Linear Congruential Sequences," TR45.5.3.1 1999 .01.

[7] 이문호 · 김순영, 오류정정이론: 터보코드의 기본원리와 응용, 도서출판 영일 2001