DVB-S2 기반 고속 LDPC 복호를 위한 Horizontal Shuffle Scheduling 방식에 관한 연구

임병수* · 김민혁** · 정지원***

A Study on Horizontal Shuffle Scheduling for High Speed LDPC decoding in DVB-S2

Byeong-su Lim* · Min-hyuk Kim* · Ji-won Jung***

본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의
연구결과로 수행되었음 (NIPA-2012 - H0301 - 12-2 005)
본 연구는 한국해양대학교의 지원을 받아 수행된 연구강화지원사업의 연구결과입니다.

요 약

DVB-S2에 적용되는 Shannon의 채널 용량 한계에 근접한 LDPC 부호는 복호화의 낮은 복잡도와 좋은 거리 특성으로 오류마루 현상인 나타나지 않고, 완성 병렬 처리가 가능하다. 하지만 구현상에 있어서 큰 블록 사이즈 및 많은 반복 횟수 때문에 복호과정에서 고속화가 어렵다. 이에 본 논문에서는 HSS(Horizontal Shuffle Scheduling) 방식을 연구하여 최적의 반복횟수를 제시한다. 고속 복호를 위한 복호과정의 한 방법으로 HSS 방식은 체크 노드를 중심으로 체크 노드가 업데이트 되는 과정에서 비트 노드도 같이 업데이트 되기 때문에 한 번의 반복이 끝났을 때 비트노드는 여러 번 반복한 효과를 가지게 된다. 결국 기존에 제시된 반복횟수보다 HSS 방식을 적용하였을 때 더 적은 반복횟수로 동일한 성능을 얻을 수 있다. HSS 방식을 적용하여 시뮬레이션 한 결과, 각각의 부호화율에서 동일한 성능으로 최소 30% ~ 최대 50% 만큼 반복횟수를 줄일 수 있음을 확인하였다.

ABSTRACT

DVB-S2 employs LDPC codes which approach to the Shannon's limit, since it has characteristics of a good distance, error floor does not appear. Furthermore it is possible to processes full parallel processing. However, it is very difficult to high speed decoding because of a large block size and number of many iterations. This paper present HSS algorithm to reduce the iteration numbers without performance degradation. In the flooding scheme, the decoder waits until all the check-to-variable messages are updated at all parity check nodes before computing the variable metric and updating the variable-to-check messages. The HSS algorithm is to update the variable metric on a check by check basis in the same way as one code draws benefit from the other. Eventually, LDPC decoding speed based on HSS algorithm improved 30% ~50% compared to conventional one without performance degradation.

키워드

DVB-S2, Horizontal Shuffle Scheduling(HSS), 체크 노드 업데이트, 비트 노드 업데이트

Key word

DVB-S2, Horizontal Shuffle Scheduling(HSS), CNU(Check Node Update), BNU(Bit Node update)

* 준회원 : 한국해양대학교(ibs0410@hhu.ac.kr)
** 정회원 : 한국해양대학교
*** 종신회원 : 한국해양대학교

접수일자 : 2012. 04. 06
심사완료일자 : 2012. 05. 01

Open Access http://dx.doi.org/10.6109/jkiice.2012.16.10.2143

ⓒThis is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(http://creativecommons.org/li-censes/by-nc/3.0/) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

Ⅰ. 서 론

광대역 위성 방송에 적용될 수 있는 오류 정정 부호는 고속 데이터 전송에 효율적이고 성능이 우수한 복호기의 적용이 필수 불가결하며, 이는 DVB-S2에서 제시된 LDPC 부호화 방식이 초고화질 다채널 실감 방송 서비스를 전국 단위로 제공하기 위한 100Mbps급 이상의 초고속 위성 방송 전송 기술로 적합하다. 유럽식 위성 방송 표준안인 DVB-S2에 적용되는 샤논의 채널 용량 한계에 근접한 LDPC 부호는 터보 부호에 비해 복호화의 복잡도가 낮을 뿐 아니라 좋은 거리 특성으로 오류마루 현상이 나타나지 않고, 완전 병렬 처리로 고속 처리가 가능한 장점이 있다.[1][2]

100Mbps급 이상의 LDPC부호를 설계하기 위해서는 부호화 관점에서는 높은 복잡도가 LDPC 부호의 중요한 문제점이었으나 최근에 삼각행렬 분해법, Linear-congruence 방법을 사용하여 부호화기를 간단하게 하였다. 특히 DVB-S2기반 부호화 알고리즘은 높은 부호화기의 복잡도를 간단하게 구현하기 위해 검사행렬에서 “1”의 위치를 주소로 생성하여 주소를 이용하여 간단하게 구현되고 있다. 구현의 문제점은 복호부에 있으며, 이를 어떻게 100Mbps급 이상의 전송률을 가지게 하는가에 있다. DVB-S2에서 제안한 알고리즘을 표준안으로 채택하고 있으며, 큰 블록 사이즈(N= 64800) 및 많은 반복 횟수를 요구하고 있다.[3] 부호화율이 1/2인 경우 60회의 반복횟수를 요구한다. 따라서 복호 시 복호 속도를 높이기 위해서는 성능은 유지하되 반복횟수를 줄일 수 있는 알고리즘이 필요하다.

본 논문에서는 고속 복호를 위한 체크 노드 관점에서 수행하는 HSS(Horizontal Shuffle Scheduling) 방식을 분석하고 구현을 위한 블록도를 제시한다. HSS방식은 체크 노드 업데이트 연산을 하면서 동시에 비트 노드 업데이트 연산을 하기 때문에 별도의 비트 노드 계산을 할 필요가 없으므로 복호과정에서 고속화가 가능하며, 반복 횟수 또한 감소시킬 수 있어 복호 throughput을 증가 시킬 수 있고 복호과정을 고속으로 처리할 수 있다.[4]

본 논문의 II절에서 HSS 방식을 적용한 LDPC 복호 알고리즘을 설명하고 III절에서는 HSS 방식의 구현 과정에서 발생하는 메모리 연결 충돌에 대해 설명하겠다. 그리고 IV절에서는 HSS 방식을 적용한 시뮬레이션 결과 분석과 구현을 위한 블록도를 제시한 후 V절에서 결론을 맺는다.

Ⅱ. HSS 알고리즘

기존의 LDPC 복호기의 복호 순서는 우선 수신데이터를 이용하여 비트 노드를 초기화 한 후 각각의 체크 노드에 연결된 비트 노드 값을 이용하여 체크 노드 업데이트를 한다. 체크 노드 업데이트 후 다시 각각의 비트 노드에 대해 업데이트를 하고 이러한 연산을 계속 반복한다. 기존의 복호 방식에 의해 체크 노드 업데이트 연산이 모두 끝난 후 비트 노드 업데이트를 하기 때문에 한 번의 반복에도 많은 지연이 발생하여 고속의 LDPC 복호를 할 수 없다. 이를 극복하기 위해 HSS 복호 방법을 제시한다. HSS 방식은 기존의 방식과는 달리 체크 노드 업데이트 연산을 하면서 비트 노드 연산을 동시에 하는 것이 가능하다. HSS 복호 방식의 흐름도는 그림 1과 같다.[4]

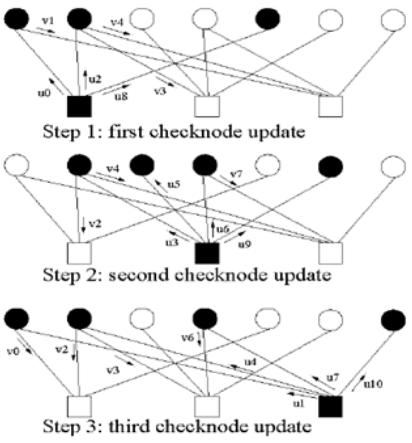

그림 1. HSS 복호 방식의 흐름도
Fig. 1 Data flow of HSS decoding

LDPC 복호 과정은 반복에 의한 복호이다. 그러므로 적어도 각 노드들이 한 번씩 업데이트가 되면 한 번의 반복이 끝나는 것이다. 그리고 전체 복호 과정은 지정된 반복 횟수만큼 반복이 되거나 그러지 않으면 충분히 신뢰성 있는 데이터를 구했을 때 끝나게 된다. 그후, 비트 노

드의 값을 이용하여 비트를 결정한다. HSS 방식을 이용한 LDPC 복호 과정 중 각 비트 노드의 값은 식 (1)에 의해 구할 수 있다.

$$S_i = LLR_i + \sum_{i=1}^{dv} u_i \tag{1}$$

여기서 S_i는 비트 노드의 최종 값을 나타내고, LLR은 수신 데이터를 나타낸다. i는 비트 노드이고, dv는 비트 노드에 연결된 엣지의 수이다. 그리고 u_i는 체크 노드 업데이트를 통해 얻어진 각 엣지 값이다.

그림 1을 통해 간단한 예를 들어 보면, 첫 번째 체크 노드 업데이트를 하기 위해 v_0, v_2, v_8 각각의 엣지 값을 가지고 체크 노드 업데이트를 한다. 이 때, 각 dpt지의 값은 수신 데이터 LLR_0, LLR_2, LLR_8이고, 체크 노드 업데이트 방법은 식 (2)에 나타내었다.

$$u_j = \bigoplus_{k=1, k\neq j}^{dc} v_k \tag{2}$$

v는 비트 노드에서 체크 노드로 향하는 엣지를 나타내고, d_c는 체크 노드에 연결된 엣지의 수이다. 위 식에서 $\oplus$는 다음과 같이 구할 수 있다.

$$|v_i| \oplus |v_j| = \min(|v_i|, |v_j|) - offset \tag{3}$$

$$sign(|v_i| \oplus |v_j|) = sign(v_i) \times sign(v_j) \tag{4}$$

체크 노드 업데이트의 값을 이용하여, 각 비트 노드의 값은 $S_0 = LLR_0 + u_0$, $S_1 = LLR_1 + u_2$, $S_4 = LLR_4 + u_8$이 된다. 그 후, 두 번째 체크 노드 업데이트를 위해 v_3, v_5, v_6, v_8을 가지고 온다. v는 v'을 이전 반복에서의 엣지 값이라 하면 다음 식에 의해 구해질 수 있다.

$$v_i = S_i - v'_i \tag{5}$$

두 번째 체크 노드의 업데이트가 끝나면 각 엣지의 값을 이용하여 다시 S_1, S_2, S_3, S_5가 구해지고, 세 번째 체크 노드에 대한 업데이트를 하여 S_0, S_1, S_3, S_7 역시 업데이트 되어 진다. 이와 같이 한 번의 반복이 끝나게 된다. 이러한 과정이 반복되면서 모든 반복이 끝나거나, 신뢰성 있는 데이터가 나올 때, 복호 과정은 끝나게 된다.

Ⅲ. HSS 구현 시 발생하는 메모리 연결 충돌

DVB-S2 규격의 LDPC 부호는 메트릭스 구조 상 부분화 시켜서 병렬연산이 가능하다. 즉, H 메트릭스에서 부분화 되는 만큼 병렬로 연산하여 속도를 증가시킬 수 있다. 하지만, 이렇게 나누게 되면 HSS 방식을 적용하여 구현 할 경우 부분화 된 집합 내에서 한 번에 두 개 이상의 비트 노드와 체크 노드가 연결이 되는 경우가 발생한다. 식 (1) 부터 식 (5) 까지를 보면 HSS 방식은 한 번에 하나씩 비트 노드와 체크 노드간에 업데이트를 실행함을 알 수 있다. 그래서 한 번에 두 개 이상의 비트 노드와 체크 노드가 연결이 되면 각 데이터 간에 메모리 연결 충돌이 발생하여 HSS 연산에서 에러가 발생할 수 있다. 그림 2는 이러한 경우를 보여준다.

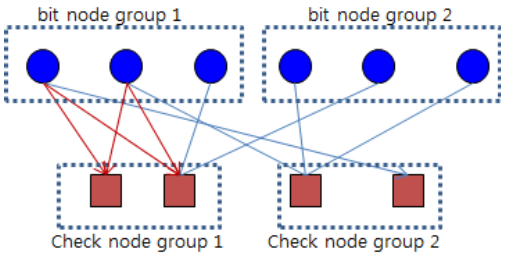

그림 2. 메모리 연결 충돌의 예
Fig. 2 Example of memory conflict

이 경우 이러한 메모리 연결 충돌을 방지하기 위하여 따로 두 개의 임시 프로세서를 사용할 수 있다. 간단한 예로 한 번에 두 개의 비트 노드와 체크 노드가 연결되었다면, 그림 2에서 첫 번째 비트 노드를 S라 하고, 이 비트 노드에서 체크 노드로 가는 세 개의 엣지 값을 각각 v_0, v_1, v_2, 체크 노드에서 비트 노드로 업데이트 되는 세 개의 엣지 값을 각각 u_0, u_1, u_2 라 하자.

$$S' = LLR + u'_0 + u'_1 + u'_2 \quad (6)$$

식 (6)에서 u'_0, u'_1, u'_2 는 이전 반복에서의 엣지 값이고, S'는 이전 반복에서의 비트 노드 값이다. 현재 반복에서의 u_2를 구하기 위해 세 번째 연결된 체크 노드로 가는 엣지 v_2를 구하기 위해서는 식 (7)이 나와야 한다.

$$S = LLR + u_0 + u_1 + u'_2 \quad (7)$$

식 (7)은 정상적인 상태에서 HSS 방식을 적용하였을 때 나올 수 있다. 하지만 그림 2에서처럼 두 개의 엣지가 한 번에 계산되어 질 때는 식 (7)처럼 나올 수가 없다. 그래서 따로 두 개의 임시 프로세서를 사용하여 임시 프로세서 두 개를 각각 S_1, S_2 라 하면 식 (8)과 같이 구할 수 있다.

$$\begin{aligned} v_0 &= S' - u'_0, \quad v_1 = S' - u'_1 \\ S_1 &= LLR + u_0 + u'_1 + u'_2 \\ S_2 &= LLR + u'_0 + u_1 + u'_2 \\ S &= S_1 + S_2 - S' \end{aligned} \quad (8)$$

Ⅳ. HSS 방식을 적용한 시뮬레이션 결과 분석 및 구현을 위한 블록도

4.1. 시뮬레이션 결과 분석

HSS 방식은 한 번의 반복에서 비트 노드가 row_weight 만큼 업데이트 되기 때문에 기존의 방식에 비해 좋은 성능을 가진다. 이는 기존의 알고리즘 보다 요구되는 반복횟수가 많이 줄어듬을 의미한다. 다음 그림 3에서 그림 11은 부호화율에 따른 기존의 알고리즘과 HSS 방식의 반복 횟수에 의한 성능을 비교한 것이다.

성능 비교에 사용된 데이터는 약 100만개이고, AWGN 환경에서 시뮬레이션을 하였다. 시뮬레이션의 모든 부호화율에서 한 블록의 크기는 64800이다. 기존의 방식은 반복 횟수를 기존방식에서의 최적의 값으로 고정하였고, HSS 방식은 반복 횟수를 10회부터 5회씩 증가시키면서 성능을 비교하였다.

그림 3은 부호화율이 1/2인 경우 기존 방식의 최적의 반복횟수 60회와 HSS 방식의 반복 횟수 30회에서 거의 동일한 성능을 보임을 알 수 있다.

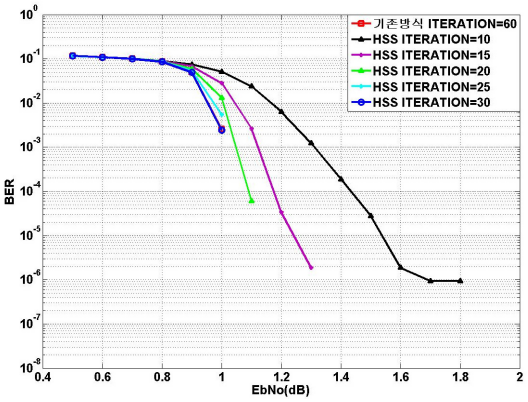

그림 3. 부호화율 1/2에서 기존의 알고리즘과 HSS의 반복횟수에 따른 성능 비교
Fig. 3 Performance between conventional method and the number of iteration in HSS from coding rate 1/2

그림 4는 부호화율 1/3에서의 성능을 나타내고 있다. 부호화율 1/3에서의 최적의 반복횟수 40회와 HSS 방식의 반복횟수 20회에서 거의 동일한 성능을 보임을 알 수 있다.

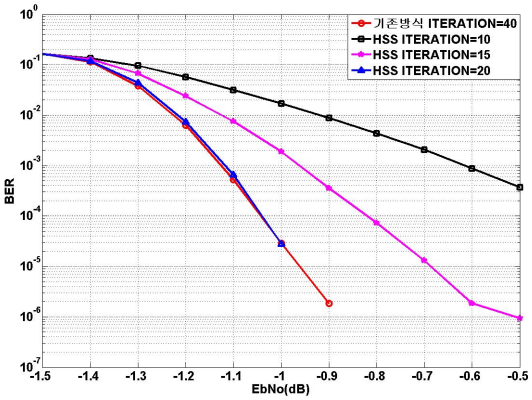

그림 4. 부호화율 1/3에서 기존의 알고리즘과 HSS의 반복횟수에 따른 성능 비교
Fig. 4 Performance between conventional method and the number of iteration in HSS from coding rate 1/3

그림 5는 부호화율 1/4에서의 성능을 나타낸다. 역시 기존 방식의 최적의 반복횟수 40회와 HSS 방식의 반복횟수 20회에서 거의 동일한 성능을 나타냄을 알 수 있다.

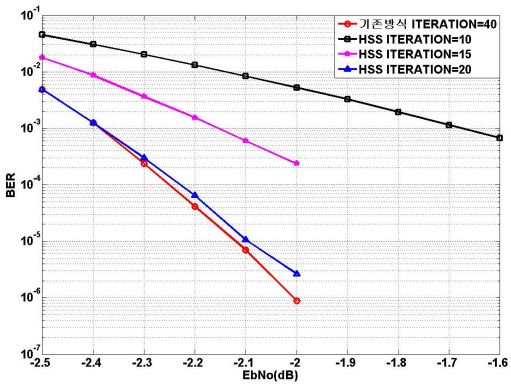

그림 5. 부호화율 1/4에서 기존의 알고리즘과 HSS의 반복횟수에 따른 성능 비교
Fig. 5 Performance between conventional method and the number of iteration in HSS from coding rate 1/4

그림 6은 부호화율 2/3에서의 성능을 나타낸다. 기존 방식의 최적의 반복횟수 50회와 HSS 방식의 반복횟수 25회에서 거의 비슷한 성능을 나타냄을 알 수 있다.

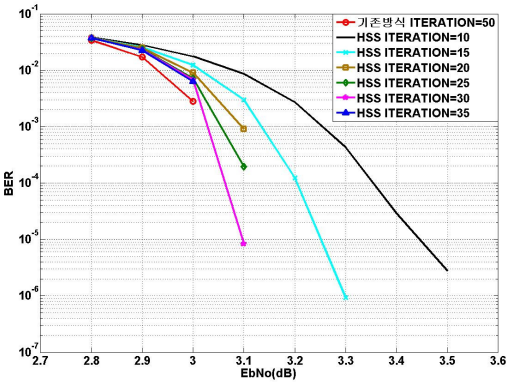

그림 6. 부호화율 2/3에서 기존의 알고리즘과 HSS의 반복횟수에 따른 성능 비교
Fig. 6 Performance between conventional method and the number of iteration in HSS from coding rate 2/3

그 밖의 부호화율에서도 기존의 최적의 반복횟수에 따른 성능과 HSS 방식에서 거의 동일한 성능을 보이는 반복횟수는 기존의 반복횟수에 비해 약 30%~50% 감소됨을 확인할 수 있었다.

기존 방식에 비해 성능 열화가 거의 발생하지 않으면서 반복횟수를 약 30%~50%만큼 줄일 수 있다는 것은 체크 노드 연산과 비트 노드 연산을 동시에 하는 HSS 방식을 사용함으로써 복호과정에서 감소된 반복횟수만큼 계산량을 줄일 수 있다는 것이고 이에 따라 복호속도가 증가한다는 것을 의미한다.

표 1은 각 부호화 방식에서 HSS 알고리즘 적용 시 요구되는 반복횟수를 나타낸다.

표 1. HSS 알고리즘 적용 시 요구되는 반복횟수
Table. 1 The number of iteration of HSS algorithm

부호화율	기존방식 (회)	HSS (회)	감소량 (%)
1/4	40	20	50
1/3	40	20	50
1/2	60	30	50
2/3	50	25	50
2/5	40	25	38
3/4	60	30	50
3/5	40	30	25
4/5	40	25	38
5/6	50	30	40
8/9	40	25	38

4.2. 구현을 위한 블록도

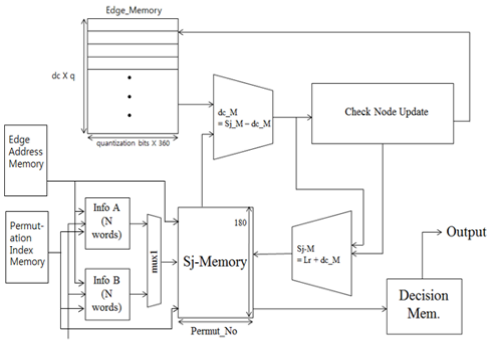

그림 7. HSS 방식을 적용한 LDPC 복호기의 블록도
Fig. 7 Structure of LDPC decoder based on HSS

HSS 방식을 적용한 LDPC 복호기는 BNU(Bit Node Update) 계산을 위한 블록을 따로 만들지 않는다. 그 이

유는 CNU(Check Node Update) 블록에서 나온 출력 값들을 바로 Sj 메모리에 업데이트 시키기 때문에 BNU를 위한 연산만을 따로 하지 않기 때문이다. 즉, LDPC 복호를 위한 연산 블록으로는 CNU 만을 필요로 한다. CNU를 계산하기 위해서는 Edge address 메모리와 Permutation index 메모리가 필요하다. Edge address 메모리는 CNU를 계산하기 위해 몇 번째 Edge 메모리 주소를 불러와야 하는가를 정하기 위해 필요하다. 1/2 같은 경우 360개씩 데이터를 분할하면 CNU 계산에 필요한 같은 컬럼에 위치한 데이터들이 90개의 블록 중 몇 번째 블록에 있는지 그 위치를 나타내 주는 것이다. 360개씩 데이터를 분할하면 각 블록은 우 순환 소 정방 행렬의 형태를 가진다. 이 때, 얼마만큼 순환 이동이 되었는지를 나타내는 것이 Permutation index 메모리이다. 즉, Permutation index 메모리는 그 블록에서 몇 번째부터 읽어야 하는지를 지시한다.

V. 결 론

본 논문에서는 DVB-S2 기반 LDPC 복호과정에서 고속 복호를 위한 HSS 방식을 분석하였다. DVB-S2에서 제안한 알고리즘은 큰 블록 사이즈(N=64800) 및 많은 반복 횟수를 요구하고 있다. 고속 복호를 위해서는 기존 알고리즘의 많은 반복 횟수를 줄일 수 있는 알고리즘의 필요성이 대두되고 있다. 이에 HSS 방식은 체크노드를 중심으로 체크노드가 업데이트 될 때 그에 연결되어 있는 비트노드도 같이 업데이트가 된다. 이는 한 번의 반복을 거치면서 비트노드는 여러 번 반복한 효과를 받기 때문에 복호과정에서 많은 반복횟수를 줄여 고속화가 가능하게 되는 것이다.

본 논문에서 분석한 HSS 방식을 DVB-S2 LDPC 복호기에 적용시켜 기존의 알고리즘과 성능을 비교해보았다. 부보화율 1/2에서 기존에 60회 반복을 권장했으나 HSS 방식을 적용하여 시뮬레이션 한 결과, 반복횟수 30회에서 기존의 성능과 비슷한 결과를 얻었다. 그 밖의 부호화율에서도 동일한 성능에서 약 30%~50% 반복횟수를 감소시킬 수 있음을 확인할 수 있었다.

또한, HSS 방식을 적용한 구현을 위한 블록도를 제시하였다. HSS 방식은 CNU 계산을 위한 Edge address 메모리와 Permutaton_index 메모리가 필요하다. 그러나 BNU 계산을 위한 메모리는 따로 생성하지 않는데, 이는 CNU 계산에서 나오는 출력 값을 Sj 메모리에 바로 업데이트 시키기 때문이다. 그러므로 구현을 할 때 BNU 계산을 위한 과정이 생략되기 때문에 고속 복호 구현을 할 수 있다. 본 논문에서 논한 HSS방식을 이용한 고속 LDPC 복호는 초고속 위성 방송을 위한 복호의 고속화에 유용한 자료가 되리라 사료된다.

감사의 글

"본 연구는 지식경제부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음"(NIPA-2012 - H0301 - 12-2 005)

"본 연구는 한국해양대학교의 지원을 받아 수행된 연구강화지원사업의 연구결과입니다."

참고문헌

[1] R. G. Gallager, "Low-Density Parity-Check Codes," IRE trans. information theory, vol.8, PP.21-28,1962.

[2] D. J. C. Mackay and R. M. Neal, "Near Shannon Limit Performance of Low-Density Parity-Check Codes," Electron. Letter, Vol.32, PP. 1645-1646,Aug.1996.

[3] Digital Video Broadcasting(DVB). "Second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2)." European Standard (Telecommunications series) ETSI EN 302 307 V1.2.1(2009-08),2009.

[4] Arthur S., Francois V., David D., Pascal U.″ DVB-S2 compliant LDPC decoder integrating Horizontal Shuffle Scheduling,″, ISPACS2006, pp. 1013-1016.

[5] M. Gomes, G. Falcao, V. Silva, V. Ferreira, A. Sengo, and M. Falcao. "Flexible parallel architecture for DVB-S2 LDPC decoders." In Global Telecommunications Conference, 2007. GLOBECOM'07. IEEE, pages 3265-3269, Washington, USA, November 2007.

저자소개

임병수(Byeong-Su Lim)

2011년 2월: 한국해양대학교 전파공학과 (공학사)
2011년 3월～현재: 한국해양대학교 전파공학과 석사과정

※관심분야 : 위성 통신, 이동 통신, 변・복조 기술, 채널 코딩, FPGA 기술 등

김민혁(Min-Hyuk Kim)

2006년 2월: 한국해양대학교 전파공학과 (공학사)
2008년 2월: 한국해양대학교 전파공학과 (공학석사)
2008년 3월～현재: 한국해양대학교 전파공학과 박사과정

※관심분야 : 위성 통신, 이동 통신, 변・복조 기술, 채널 코딩, FPGA 기술 등

정지원(Ji-Won Jung)

1989년 2월 : 성균관대학교 전자공학과(공학사)
1991년 2월 : 성균관대학교 전자공학과(공학석사)
1995년 2월 :성균관대학교 정보공학과(공학박사)
1991년 1월～1992년 2월 : LG 정보통신연구소 연구원
1995년 9월～1996년 8월 : 한국통신 위성통신연구실 선임연구원
1997년 3월～1998년12월 : 한국전자통신연구원 초빙 연구원
1996년 9월～현재: 한국해양대학교 전파공학과 정교수
2001년 8월～2002년 8월 : 캐나다 NSERC Fellowship (Communication Research Center 근무)

※관심분야 : 위성 통신, 이동 통신, 변・복조 기술, 채널 코딩, FPGA 기술 등