dc 분리 기반의 고속 LDPC 복호 알고리즘에 관한 연구

권해찬 · 김태훈 · 정지원*

A Study on High Speed LDPC Decoder Algorithm based on dc saperation

Hae-chan Kwon · Tae-hoon Kim · Ji-won Jung*

Department. of Radio Science Engineering, Korea Maritime University, Busan 606-791, Korea

요 약

본 논문에서는 DVB-S2 기반 고속 LDPC 복호를 위한 알고리즘을 제안하였다. 체크 노드 연산중에 비트 노드 연산을 수행하여 기존의 LDPC 복호 알고리즘에 비해 반복횟수를 줄일 수 있는 horizontal shuffle scheduling 알고리즘을 기반으로 하여 복호 속도를 보다 고속화 할 수 있는 알고리즘을 제안하였다. 기존의 체크 노드 연산은 하나의 메모리에서 값을 가져오기 때문에 체크 노드 연산과정에서 많은 지연이 발생 하는데 이를 row weight의 개수인 dc개의 병렬구조로 설계함으로써 체크 노드 연산과정의 지연을 줄일 수 있고 따라서 고속 복호가 가능하다. 이를 DVB-S2에 제시되고 있는 다양한 부호화율에서 dc개의 분리 할 수 있는 최대의 메모리를 제시하고 전송률을 제시하였다.

ABSTRACT

In this paper, we proposed high speed LDPC decoding algorithm based on DVB-S2 standard . For implementing the high speed LDPC decoder, HSS algorithm which reduce the iteration numbers without performance degradation is applied. In HSS algorithm, check node update units are update at the same time of bit node update. HSS can be accelerated to the decoding speed because it does not need to separate calculation of the bit nodes, However, check node calculation blocks need many clocks because of just one memory is used. Therefore, this paper proposed dc-split memory structure in order to reduced the delay and high speed decoder is possible. Finally, this paper presented maximum split memory and throughput for various coding rates in DVB-S2 standard.

키워드 : DVB-S2, Horizontal Shuffle Scheduling(HSS), row weight, 체크 노드 업데이트

Key word : DVB-S2, Horizontal Shuffle Scheduling(HSS), row weight, Check Node Update(CNU)

접수일자 : 2013. 05. 16 심사완료일자 : 2013. 06. 11 게재확정일자 : 2013. 06. 20

* **Corresponding Author** Ji-won Jung (E-mail:jwjung@kmou.ac.kr, Tel:+82-51-410-4920)

Department of Radio Communication Engineering, Korea Maritime and Ocean University, 727 Taejong-ro, Yeongdo-gu, Busan 606-791, Korea

Open Access <http://dx.doi.org/10.6109/jkiice.2013.17.9.2041>

print ISSN: 2234-4772 online ISSN: 2288-4165

©This is an Open Access article distributed under the terms of the Creative Commons Attribution Non-Commercial License(<http://creativecommons.org/licenses/by-nc/3.0/>) which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.
Copyright © The Korea Institute of Information and Communication Engineering.

I. 서 론

LDPC(Low Density Parity Check) 부호는 최신 위성 방송 및 무선 통신에서 적용되는 오류정정 알고리즘으로 채널 부호화 방식을 주로 적용하고 있다. 광대역 위성 방송에 적용될 수 있는 오류 정정 부호는 고속 데이터 전송에 효율적이고 성능이 우수한 부호의 적용이 필수 불가결하며, 이는 DVB-S2(Digital Video Broadcasting - Satellite - Second Generation)에서 제시된 LDPC 부호화 방식은 초고화질 다채널 실감 방송 서비스 및 멀티미디어 기반을 둔 고속 통신에 적합하다[1][2][3]. 유럽식 위성 방송 표준안인 DVB-S2에 적용되는 사논의 채널 용량 한계에 근접한 LDPC 부호는 터보 부호에 비해 부호화의 복잡도가 낮을 뿐 아니라 좋은 거리 특성으로 오류마루 현상이 나타나지 않고, 완전 병렬 처리로 고속 처리가 가능한 장점이 있다[4]. 하지만 LDPC 부호는 큰 블록 사이즈($N=64800$) 및 많은 반복 횟수를 요구하고 있다. 또한 고속화를 위한 구현상 문제점의 대부분은 부호부에 있으며, 이를 어떻게 100 Mbps급 이상의 전송률을 가지게 하는가에 있다. 따라서 부호 시 병렬 처리, 병렬 처리에 따른 메모리 설계가 중요하다.

이에 본 논문에서는 비트 노드 계산을 체크 노드 계산 후에 하지 않고 체크 노드 계산 중에 수행하는 HSS(Horizontal Shuffle Scheduling) 방식[5]을 기반으로 하는 부호기 구조에서 기존의 방식 보다 고속화 할 수 있는 알고리즘을 연구 하였으며, 기존의 row weight 개수인 dc 개의 직렬 구조에서 dc 개의 병렬 구조로 메모리를 효율적으로 설계함으로써 고속화가 가능하게끔 하였다. 이를 DVB-S2에 제시되고 있는 다양한 부호화율에서 dc 개의 분리 할 수 있는 최대의 메모리를 제시 하고 전송률을 제시하였다[6].

II. HSS 기반 LDPC 부호 알고리즘

기존의 LDPC 부호기의 부호 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 부호 방식에 의해 체크 노드 업데이트 연산이 모두 끝난 후 비트 노드 업데이트를 하기 때문에 한 번의 반복에도 많은 지연이 발생하여 고속의 LDPC 부호를 할 수 없다.

이를 극복하기 위해 HSS 방식은 기존의 방식과는 달리 체크 노드 업데이트 연산을 하면서 비트 노드 업데이트 연산을 동시에 하는 것이 가능하다. HSS 부호 방식의 흐름도는 그림 1과 같다.

그림 1. HSS 부호의 흐름도
Fig. 1 HSS Decoding Flow

HSS 방식을 이용한 LDPC 부호 과정 중 각 비트 노드의 값은 다음 식에 의해 구할 수 있다.

$$S_i = LLR_i + \sum_{i=1}^{dv} u_i \quad (1)$$

여기서 S_i 는 비트 노드의 최종 값을 나타내고, LLR 은 수신 데이터를 나타낸다. i 는 비트 노드이고, dv 는 비트 노드에 연결된 edge 수이다. 그리고 u_i 는 체크 노드 업데이트를 통해 얻어진 각 edge 값이다. 체크 노드 업데이트는 식 (2)와 같다.

$$u_j = \bigoplus_{k=1, k \neq j}^{d_c} v_k \quad (2)$$

v 는 비트 노드에서 체크 노드로 향하는 엣지를 나타내고, dc 는 체크 노드에 연결된 엣지 수이다. 위 식에서 $\oplus$ 는 다음과 같이 구할 수 있다[7][8].

$$|v_i \oplus v_j| = \min(|v_i|, |v_j|) - offset \quad (3)$$

$$sign(|v_i \oplus v_j|) = sign(v_i) \times sign(v_j) \quad (4)$$

체크 노드 업데이트의 값을 이용하여, 각 비트 노드의 값은 식 (5)에 의해 구해진다.

$$v_i = S_i - v'_i \quad (5)$$

v'_i 는 이전 반복에서의 엣지 값을 나타낸다. 이와 같이 한 번의 반복이 끝나게 된다. 이러한 과정이 반복되면서 모든 반복이 끝나거나, 신뢰성 있는 데이터가 나올 때, 복호 과정은 끝나게 된다.

HSS 방식을 한 번의 반복에서 비트 노드가 row weight만큼 업데이트 되기 때문에 기존의 방식에 비해 좋은 성능을 가진다. 이는 기존의 알고리즘 보다 요구되는 반복횟수가 많이 줄어들음을 의미한다.

다음 그림 2는 HSS 방식을 적용한 LDPC 복호 구조를 나타낸다.

그림 2. HSS 방식을 적용한 LDPC 복호기 구조
Fig. 2 Structure of LDPC decoder based on HSS

HSS 방식을 적용한 복호기의 구조를 보면 크게 메모리 부분과 CNU 부분으로 나눌 수 있다. 특히 수신 데이터를 저장하는 메모리와 Sj메모리에서 데이터를 읽어올 때는 H 매트릭스에 따라 랜덤하게 메모리 액세스가

되어야 하기 때문에 이를 구현할 수 있는 여러 인덱스들이 필요하다.

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

III. dc개의 메모리 분리 기반의 고속 복호 알고리즘 제안

HSS를 적용한 LDPC 복호 알고리즘에서는 체크 노드 연산과정에서 edge메모리와 Sj메모리를 각각 하나씩 사용하였기 때문에 체크 노드 연산 과정에서 dc개의 데이터를 읽어오고 체크 노드 연산 후 다시 edge메모리와 Sj메모리에 저장하는데 지연이 발생하게 된다. 이러한 문제를 해결하고 고속 복호를 위해 체크 노드 연산에 필요한 dc개의 데이터를 한 번에 읽어와 연산하고 한 번에 저장할 수 있는 방안인 edge 메모리와 Sj메모리를 각각 dc개로 분리하는 알고리즘을 제안하였다. 그림 3는 edge메모리와 Sj메모리를 각각 dc개로 분리한 알고리즘을 나타낸다.

그림 3. edge 메모리와 Sj 메모리를 dc개로 분리한 복호 알고리즘
Fig. 3 Decoding algorithm of dc split dege memory and Sj memory

edge메모리와 Sj메모리를 각각 dc개로 분리하게 되면 체크 노드 연산과정에서 동시에 dc개의 데이터를 연

산할 수 있게 된다. 이렇게 되면 한 번의 클럭으로 dc 개의 데이터를 동시에 처리 가능하기 때문에 고속 복호가 가능하게 된다. 동시에 dc 개의 데이터를 읽고 저장할 수 있게 되면 체크 노드 연산을 병렬로 처리할 수 있기 때문에 이 과정에서도 소요되는 클럭을 줄일 수 있다. 이때에 필요한 클럭 수는 $2^q \geq dc$ 를 만족하는 가장 작은 q 로 나타낼 수 있다. 이에 따라 부호화율이 1/2인 경우 dc 가 7이기 때문에 3 클럭의 연산으로 체크 노드 연산을 할 수 있게 된다. 다음은 HSS 알고리즘과 제안한 알고리즘을 적용시켰을 때의 성능을 비교한 그림이다. 부호화율은 1/2이고 이때 dc 의 개수는 7이다.

그림 4. eHSS 알고리즘과 제안한 알고리즘의 성능 비교
Fig. 4 Performance comparison by HSS and proposed algorithm

edge메모리와 S_j 메모리를 dc 개로 나눈 알고리즘과 기존의 HSS 알고리즘의 성능을 비교한 결과 거의 비슷한 성능을 보임을 알 수 있다. 이는 제안한 알고리즘으로 성능의 열화 없이 고속 LDPC 복호를 구현할 수 있다는 것을 나타낸다.

IV. 부호화율에 따른 속도 및 메모리

본 절에서는 위에서 살펴본 고속 복호 알고리즘에 따른 메모리량과 속도를 비교하여 고속 복호를 위한 최적의 알고리즘을 알아보겠다.

다음 표 1은 위에서 제시한 복호 알고리즘에 따른 각각의 메모리 사용량을 비교한 표이다.

표 1. 복호 알고리즘에 따른 메모리 비교

Table. 1 Memory comparison according to decoding algorithm

	요구되는 메모리	각 메모리 수	요구되는 메모리 총 개수	메모리의 양
HSS부호화율 1/2	$dc \times 360 \times q$ 180×360	1개 1개	2개	$7 \times 360 \times 90 \times 6$ $+ 180 \times 360 \times 6$ $= 1749600$
dc 개의 S_j 메모리와 dc 개의 edge 메모리	$q \times 360$ 180×360	dc 개 dc 개	$2 \times dc$ 개	$90 \times 360 \times 7 \times 6$ $+ 180 \times 360 \times 7 \times 6$ $= 4082400$
HSS부호화율 3/4	$dc \times 360 \times q$ 180×360	1개 1개	2개	$14 \times 360 \times 45 \times 6$ $+ 180 \times 360 \times 6$ $= 1749600$
dc 개의 S_j 메모리와 edge 메모리를 공유한 알고리즘	$(q+180) \times 360$	dc 개	dc 개	$270 \times 360 \times 14 \times 6$ $= 8164800$
HSS부호화율 9/10	$dc \times 360 \times q$ 180×360	1개 1개	2개	$30 \times 360 \times 18 \times 6$ $+ 180 \times 360 \times 6$ $= 1555200$
두 개의 S_j 메모리와 edge 메모리를 공유한 알고리즘	$(2 \cdot q + 2 \cdot 180) \times 360$	$dc/2$ 개	$dc/2$ 개	$396 \times 360 \times 15 \times 6$ $= 12830400$

위의 표를 보게 되면 부호화율이 1/2인 HSS 알고리즘의 경우 edge 메모리와 S_j 메모리가 1개씩 총 2개가 필요하고 필요한 총 메모리의 사용량은 1,749,600 bit이다. dc 개로 분리하여 edge 메모리와 S_j 메모리를 각각 나누는 구조의 경우는 edge메모리와 S_j 메모리가 각각 dc 개만큼 필요하기 때문에 전체적인 메모리의 사용량은 4,082,400 bit이다.

부호화율이 3/4일 때 dc 개로 나누는 edge 메모리와 S_j 메모리를 공유한 알고리즘의 전체적인 사용량은 8164800 bit이고, 부호화율이 9/10일 때 두 개의 S_j 메모리와 edge 메모리를 공유한 알고리즘의 경우 총 메모리 사용량은 12,830,400 bit이다. 메모리 사용량에서는 HSS 방식에 비해 훨씬 많은 양의 메모리를 사용함을 알 수 있다. 하지만 dc 개로 나누는 edge 메모리와 S_j 메모리를 공유한 알고리즘이 dc 개로 분리하여 edge 메모리와 S_j 메모리를 각각 나누는 알고리즘보다 메모리의 수가 더 작고 두 개의 S_j 메모리와 edge 메모리를 공유한 알고리즘이 dc 개로 나누는 edge 메모리와 S_j 메모리를 공유한 알고리즘보다 메모리 수가 더 작음을 알 수 있다. 그렇기 때문에 부호화율에 따른 dc 의 개수에 따라 적합한 알고리즘을 선택하여 사용할 수 있다.

복호 throughput을 계산하기 위해서는 1회 반복에서 각각의 알고리즘에서 요구되는 클럭 수를 계산하여야 한다. 다음 식 (6)은 부호화율에 따른 기존 HSS 방식의 클럭 수를 계산하는 식이다.

$$N_{tc} = (q \times dc) + N_c + (dc \times 2) \times 2 \quad (6)$$

위 식에서 N_{tc} 는 1회 반복에서 요구되는 총 클럭 수를 나타내고, N_c 는 체크노드 연산에 필요한 클럭 수를 나타낸다. q 는 체크 노드 그룹 수를 나타내고 dc 는 Row weight의 수를 나타낸다. 이 부분은 체크 노드 연산을 위해 Sj메모리와 edge메모리에서 각각 dc 만큼의 데이터를 읽어오고 체크 노드 업데이트가 끝난 후 다시 edge 메모리와 Sj메모리에 값이 저장되기 때문이다. 다음 식 (7)는 3장에서 제안한 방식들의 클럭 수를 계산하는 식이다.

$$N_{tc} = \{(N_c + 2) + (q - 1)\} \times N_m \quad (7)$$

위의 식에서 N_m 은 하나의 메모리에서 가져오는 메모리의 수를 나타낸다.

다음 표 2는 식 (6)을 이용하여 부호화율 1/2에서 기존 HSS 방식과 dc 개의 Sj메모리와 dc 개의 edge 메모리를 각각 나눈 알고리즘의 속도를 비교한 표이다.

표 2. 기존 HSS 방식과 Sj메모리와 edge메모리를 각각 dc 개로 나눈 알고리즘의 속도비교

Table. 2 Throughput comparison by HSS method and algorithm of dc split edge memory and sj memory

	요구되는 clock (1회 반복)	HSS 적용 가능 여부	최대 반복 시 요구 되는 clock	10ns clock에서 throughput
HSS($r=1/2, q=90, dc=7$)	662 clock	가능	19860 clock (30회)	326 Mbit/s
dc 개의 Sj 메모리와 dc 개의 edge 메모리 ($N_m=1$)	94 clock	가능	2820 clock (30회)	2.29 Gbit/s
HSS($r=3/4, q=45, dc=14$)	698 clock	가능	20970 (30회)	309 Mbit/s
dc 개의 Sj 메모리와 edge 메모리를 공유한 구조 ($N_m=2$)	100 clock	가능	3000 clock (30회)	2.16 Gbit/s
HSS($r=9/10, q=18, dc=30$)	688 clock	가능	20670 (30회)	313 Mbit/s
두 개의 Sj 메모리와 edge 메모리를 공유한 구조 ($N_m=4$)	96 clock	가능	2880 clock (30회)	2.25 Gbit/s

부호화율 1/2에서 HSS 알고리즘은 1회 반복에 662 클럭이 필요하며 전체 반복횟수를 기존 LDPC에 비해 절반으로 줄일 수 있기 때문에 최대 반복 시 요구되는 클럭은 19860 클럭이 소요된다. 이에 비해 edge 메모리와 Sj 메모리를 각각 dc 개로 나눈 경우에는 1회 반복

에 94 클럭이 필요하고 최대 반복 시 2820 클럭이 소요된다. 이에 따라 10ns에서 throughput을 계산하였을 때 기존의 HSS 방식은 326 Mbit/s, dc 개의 Sj메모리와 dc 개의 edge메모리를 각각 나눈 알고리즘에서는 2.29 Gbit/s로 기존 HSS 방식에 비해 약 7배의 throughput을 얻었다.

부호화율 3/4일 때 HSS 알고리즘은 1회 반복에 698 클럭이 필요하고 최대 반복 시 20970 클럭이 소요된다. 이에 비해 dc 개로 나눈 edge 메모리와 Sj 메모리를 공유한 경우에는 1회 반복에 100 클럭이 필요하고 최대 반복 시 3000 클럭이 소요된다. 또한 최대 반복에서 요구되는 클럭 수로 10ns 클럭에서 복호 throughput을 계산하였을 때, 부호화율이 3/4인 기존 HSS 방식에서는 309 Mbit/s, dc 개의 Sj 메모리와 edge 메모리를 공유한 알고리즘에서는 2.16 Gbit/s로 기존에 비해 약 7배 향상된 복호 throughput을 얻었다.

부호화율이 9/10에서 HSS 알고리즘은 1회 반복에서 688 클럭, 최대 반복 시 20670 클럭이 필요하다. 이에 비해 두 개의 Sj 메모리와 edge 메모리를 공유한 알고리즘에서는 1회 반복에서 96 클럭, 최대 반복 시 2880 클럭이 소요된다. 최대 반복에서 요구되는 클럭 수에 따라 10ns 클럭에서 복호 throughput을 계산할 수 있다. 부호화율 9/10에서 기존 HSS 방식은 313 Mbit/s, 두 개의 Sj메모리와 edge메모리를 공유한 알고리즘에서는 2.25 Gbit/s의 복호 throughput을 얻었다. 이는 기존의 HSS 방식에 비해 약 7배 향상된 복호 throughput을 얻을 수 있었다.

위의 표 2에서 보듯이 edge 메모리와 Sj 메모리를 dc 개로 나눈 알고리즘은 1회 반복에서 상대적으로 HSS 알고리즘 보다 적은 클럭이 필요하고 1회 반복에서 적은 클럭이 필요한 만큼 고속 복호를 할 수 있다. 그 이유는 체크 노드 연산과정에서 dc 개 만큼 분리되어 한번에 데이터를 읽고 저장하는 것이 가능하고 또한 체크 노드 연산을 병렬로 처리 가능하기 때문에 HSS 알고리즘보다 적은 클럭이 필요하게 되고 고속 복호가 가능하다.

V. 결 론

본 논문에서는 DVB-S2 전송 방식 중 LDPC 복호에 비중을 두어 연구를 하였으며, 복호기 구조에서 HSS 방

식을 기반으로 복호과정을 고속화할 수 있는 알고리즘을 제안하였다. HSS 알고리즘은 비트 노드 연산을 체크 노드 연산 후에 수행하지 않고 체크 노드 연산 중에 수행하기 때문에 기존의 LDPC 복호 방식에 비해 반복횟수를 줄일 수 있고 고속 복호가 가능하다. 이에 HSS 알고리즘을 기반으로 복호 속도를 더욱 높일 수 있는 알고리즘으로 기존의 row weight 개수인 dc 개의 직렬 구조에서 dc 개의 병렬 구조로 비트 노드 메모리와 edge메모리를 나누어 체크 노드 연산을 고속으로 할 수 있는 알고리즘을 제안하였다. edge메모리와 비트 노드 메모리를 dc 개로 나눈 제안한 알고리즘은 1회 반복에서 상대적으로 HSS 알고리즘 보다 적은 클럭이 필요하다. 그 이유는 체크 노드 연산과정에서 dc 개 만큼 분리되어 한번에 데이터를 읽고 저장하는 것이 가능하고 또한 체크 노드 연산을 병렬로 처리 가능하기 때문에 HSS 알고리즘보다 적은 클럭이 필요하게 되어 고속 복호가 가능하다.

dc 개의 직렬구조에서 dc 개의 병렬 구조로 메모리 구조를 변경함으로써 복호 속도를 개선할 수 있으나 이는 메모리의 낭비를 초래하며, 메모리 효율성 면에서는 다소 떨어진다 할 수 있다. 하지만 메모리 효율성의 측면을 감안하더라도 고속화를 극대화 할 수 있는 알고리즘으로 기존의 방식보다 약 7배 이상의 고속 복호가 가능하리라 사료된다.

감사의 글

본 연구는 교육과학기술부와 한국연구재단의 지역혁신인력양성사업으로 수행된 연구결과임.

“본 연구는 미래창조과학부 및 정보통신산업진흥원의 대학 IT연구센터 지원사업의 연구결과로 수행되었음.” (NIPA-2013-H0301-13-2005)

REFERENCES

- [1] R. G. Gallager, *Low-density parity-check codes*, IRE trans. inform. theory, vol. 8, no.1, pp. 21-28, Jan. 1962.
- [2] “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 Telecommun. Standard Inst.* (ETSI) En 302 307 V1.2.1, Apr. 2009.
- [3] J. Dielissen, A. Hekstra, and V. Berg. “Low cost LDPC decoder for DVB-S2,” in *Proc. 2006 Design, Automation Test in Europe (DATE'06)*, vol. 2, pp. 1-6, Munich, Germany, Mar. 2006.
- [4] D. J. C. Mackay and R. M. Neal, “Near Shannon limit performance of low-density parity-check codes,” *Electron. Lett.*, vol. 32, no. 18, pp. 1645-1646, Aug. 1996.
- [5] A. Segard, F. Verdier, D. Declercq, P. Urard. “DVB-S2 compliant LDPC decoder integrating Horizontal Shuffle Scheduling,” *Int. Symp. Intelligent Signal Process. Commun. (ISPACS 2006)*, pp. 1013-1016, Dec. 2006.
- [6] S. M. Kim, C. S. Park, and S. Y. Hwang, “A Novel Partial Parallel Architecture for High-throughput LDPC Decoder for DVB-S2,” *IEEE Trans. Consum. Electron.*, vol. 56 no. 2, pp. 820-825, May 2010.
- [7] V. Savin. “Self-corrected min-sum decoding of LDPC codes,” in *Proc. IEEE Int. Symp. Inform. Theory (IEEE ISIT)*, pp. 146-150, July 2008.
- [8] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, “Efficient implementations of the sum-product algorithm for decoding LDPC codes,” in *Proc. IEEE Globecom*, pp.1036-1036E, San Antonio, TX, Nov. 2001.

권해찬(Hae-Chan Kwon)

2013년 2월 : 한국해양대학교 전파공학과(공학사)

2013년 3월 ~ 현재 : 한국해양대학교 전파공학과 석사과정

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

김태훈(Tae-Hoon Kim)

2013년 2월 : 한국해양대학교 전파공학과(공학사)
2013년 3월 ~ 현재 : 한국해양대학교 전파공학과 석사과정
※관심분야 : 위성 통신, 이동 통신, 수중 통신, 변 · 복조 기술, 채널코딩, 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 정보통신 연구소 연구원
1995년 9월 ~ 1996년 8월 : 한국통신 위성통신연구실 선임연구원
1997년 3월 ~ 1998년 12월 : 한국전자통신연구원 초빙연구원
1996년 9월 ~ 현재 : 한국해양대학교 전파공학과 정교수
2001년 8월 ~ 2002년 8월 : 캐나다 NSERC Fellowship(Communication Research Center 근무)
※관심분야 : 위성 통신, 이동 통신, 수중 통신, 변 · 복조 기술, 채널코딩, FPGA 기술 등