병렬 처리 구조 터보 부호에서 라틴 방진 행렬로 구성된 충돌 방지 인터리버

준회원 김 대 선*, 오 현 영**, 종신회원 송 홍 엽*

Collision-free Interleaver Composed of a Latin Square Matrix for Parallel-architecture Turbo Codes

Dae-Son Kim*, Hyun-Young Oh** *Associate Members*,
Hong-Yeop Song* *Lifelong Member*

요 약

병렬 처리 구조 터보 부호에서 메모리 충돌을 피하기 위한 구성 인터리버 설계가 필요하다. 본 논문에서는 기존에 설계된 인터리버들과 라틴 방진 행렬로 구성된 충돌 방지 인터리버를 제안한다. 제안된 인터리버는 다양한 블록 길이와 다양한 병렬 처리 차수에 대하여 쉽게 최적화 할 수 있다. 제안된 인터리버의 성능을 컴퓨터 모의실험을 통해 검증하였다.

Key Words : Turbo codes, Interleaver, Parallel architecture, Collision-free

ABSTRACT

In the parallel-architecture turbo codes, the constituent interleaver must avoid memory collision. This paper proposes a collision-free interleaver structure composed of a Latin square matrix and pre-designed interleavers. Our proposed interleavers can be easily optimized for various information block sizes and for various degrees of parallelism. Their performances were evaluated by computer simulation.

Ⅰ. 서 론

터보 부호의 뛰어난 오류 정정능력으로 인해 그동안 많은 연구가 수행 되어왔으며 다양한 표준에 채택되어 지고 있다. 터보 부호는 쉽고 빠르게 부호화 할 수 있는 장점을 가지고 있으나, 복호기에서는 격자 (trellis) 구조에 근간하여 BCJR 알고리즘에 의해 반복 복호를 실행하기 때문에 복호 지연이라는 문제점을 가지고 있다[1],[2]. 이러한 문제점과 관련하여 병렬 처리 구조 터보 부호는 최근 채널 부호화 분야에서 가장 뜨거운 주제 중 하나이다[3]-[6].

병렬 처리 구조에서 전체 블록은 몇 개의 부-블록으로 나누어지고, 각각 부-블록들이 독립된 프로세서를 가지고 있어 부호화 및 복호화를 동시에 진행하게 된다. 순환 꼬리 물림 부호화 방법을 사용하게 되면 각 부-블록을 부호화 하는 과정에서 꼬리 비트를 필요로 하지 않는다[3],[4]. 병렬 처리 복호화 과정에서 각 프로세서는 SISO(soft-input soft-output) 장치에 해당한다. 만일 두 개 이상의 프로세서가 동시에 같은 하나의 메모리에 접근하려고 한다면 충돌이 일어나 데이터들을 한 번에 읽어 들이지 못하고 추가적인 지연이 발생하게 된다[5].

※ 본 연구는 삼성전자의 "4G wireless system에 관한 연구 개발" 과제의 지원에 의해 이루어졌음
* 연세대학교 전기전자공학과 부호 및 정보이론 연구실({ds.kim, hy.song}@yonsei.ac.kr)
** 삼성전자 (hy.oh@yonsei.ac.kr)
논문번호 : KICS2007-12-536, 접수일자 : 2007년 12월 5일 최종논문접수일자 : 2008년 1월 25일

이러한 충돌들을 해결하기 위하여 병렬 처리를 위한 충돌 방지 인터리버들이 설계 되어야 한다.

2003년 시간적 치환 (temporal permutation)과 공간적 치환 (spatial permutation)으로 이루어진 충돌 방지 인터리버가 제안되었다[6]. 본 논문에서는 이 인터리버를 2D 인터리버라 부르겠다. 2004년에는 몇 가지 변수들로 정의된 충돌 방지 ARP (almost regular permutation)가 제안되었다[7]. 최근에는 대수학적인 방법을 이용하는 QPP (quadratic permutation polynomial) 인터리버등 다양한 충돌 방지 인터리버들이 제안되어오고 있다[8].

제안된 충돌 방지 인터리버들은 복잡한 최적화 과정을 가지고 있다. 통신 시스템에서는 일반적으로 다양한 블록 크기들을 지원하고 있으며, 구성 터보 부호의 인터리버 또한 가능한 모든 블록 크기들에 대하여 정의 되어야 한다. 또한 블록 길이에 따라 다양한 병렬 처리 수준을 바꾸어 줘야 한다. 본 논문은 다양한 블록 길이와 병렬 처리 수준에 대하여 쉽게 최적화 할 수 있는 인터리버를 제안하고자 한다.

Ⅱ. 기존의 출동 방지 인터리버

본 장에서는 참고 문헌 [6], [7] 그리고 [8]에서 제안된 충돌 방지 인터리버들에 대하여 살펴보고자 한다.

2.1 2D 인터리버

2D 인터리버는 시간적 치환과 공간적 치환으로 구성되어 있다[6]. 시간적 치환은 각각의 부-블록에 있는 데이터 심벌들을 그 부-블록 내에서 섞어주며, 공간적 치환은 부-블록 간에 데이터 심벌들을 섞게 된다. 전체 블록의 길이를 K, 부-블록들의 개수를 L 이라고 하고 각각 부-블록의 길이를 M이라 하자. 그러면 M은 K/L과 같다. 심벌 인덱스 $k\in\{0,1,...,K-1\}$는 시간적 인덱스 t와 공간적 인덱스 s를 사용하여 2차원적 배열 구조로 나타낼 수 있으며 여기서 $k=s\cdot M+t$이고, $s\in\{0,1,...,L-1\}$, $t\in\{0,1,...,M-1\}$이다. 시간적 치환을 $\pi_T(t,s)$, 공간적 치환을 $\pi_S(t,s)$라 하면 2D 인터리버는 다음과 같이 정의된다.

$$\pi(k)=\pi(t,s)=\pi_S(t,s)M+\pi_T(t,s) \qquad (1)$$

k번째로 읽혀지는 심벌은 $\pi_S(t,s)$번째 부-블록의 $\pi_T(t,s)$번째 위치에서 읽혀지게 된다. 병렬 처리 구조의 터보 부호에서 충돌을 방지하기 위해서는, 공간 치환 $\pi_S(t,s)$의 모든 $t\in\{0,1,...,M-1\}$에 대하여 $\pi_S(t,s)_{s=0,1,...,L-1}$가 부-블록들 $0,1,...,L-1$과 일대일 대응관계가 있어야 한다.

2.2 Almost Regular Permutation (ARP)

RP는 서로소 (relatively prime) 인터리버를 기반으로 하고 주기적인 변화 패턴을 추가하여 제안되었다 [7]. ARP는 다음 수학식과 같이 정의된다.

$$\pi(k)=Pk+L(\alpha(k)P+\beta(k))+\gamma\ (\mathrm{mod}K) \qquad (2)$$

여기서 P는 K와 서로소인 수이고, L은 병렬 처리의 차수를 뜻하고, $\alpha(k)$와 $\beta(k)$는 $0\le k\le K-1$에 대해 주기가 L이면서 양의 정수로 이루어진 수열이고, γ는 초기 오프셋이다. 일반적으로 $\alpha(k)$는 0 또는 1의 값을 가지며, $\beta(k)$는 0에서부터 8사이의 값을 가진다. ARP는 IEEE 802.16, DVB-RCS, 그리고 DVB-RCT에서 채널 부호화 표준 기술로 채택되어 사용되고 있다.

2.3 Quadratic Permutation Polynomial (QPP) 인터리버

QPP 인터리버는 대수학적인 방법에 의해 설계된 인터리버이다. QPP 인터리버는 Takeshita에 의해 최대 충돌 방지 인터리버라는 것이 증명 되었다 [8]. 즉 설계된 QPP 인터리버는 전체 블록 길이 K를 나누는 모든 부 블록 길이 M에 대하여 충동 방지 조건을 만족한다. QPP 인터리버는 수식 (3)과 같이 정의된다.

$$\pi(k)=f_1k+f_2k^2(\mathrm{mod}K) \qquad (3)$$

여기서 f_1과 f_2는 음이 아닌 정수이며 QPP 인터리버가 되기 위한 조건은 논문 [8]에 나와 있다.

Ⅲ. 제안한 인터리버

충돌 방지 인터리버를 정의함에 있어 공간적 치환을 행렬 형태로 정의하면 수식 (4)와 같다.

$$\pi(k)=\pi(sM+t)=u_{ts}M+\pi_T(t) \qquad (4)$$

여기서 $M\times L$ 행렬 $\mathbf{U}=\{u_{ts}\}$는 부-블록 사이에서의 사상을 나타낸다. 즉 k번째로 읽혀지는 심벌은 u_{ts}번째 부-블록의 $\pi_T(t)$번째 위치에있는 심벌을 읽

어 오게 된다. 메모리의 충돌을 피하기 위해서 U의 각각의 행벡터는 서브블록들, 즉 $0,1,...,L-1$의 치환 형태여야 한다. 시간적 치환 $\pi_T(t)$로는 미리 정의된 인터리버를 사용한다. 최적화 과정은 사상 행렬인 U를 정하는 과정이다. 즉 $\{0,1,...,L-1\}$에 대한 M개의 치환을 정하는 과정이다.

인터리빙 전과 후에 같은 부-블록에 속하게 되는 데이터 심벌들의 패턴은 성능에 영향을 미치는 요인이 된다. 만약 같은 부-블록에 속하는 심벌들이 인터리빙 후에도 같은 부-블록에 속하게 된다면 그 심벌들은 낮은 무게를 가지는 부호어가 되어 성능을 열화 시킬 수 있으며, 두 개의 심벌들의 경우 그 사이클이 작다면 반복 복호 과정에서 정보 전달을 막아 성능열화를 가져 오게 된다. 따라서 같은 부-블록에 속하는 심벌들은 최대한 다른 부-블록으로 퍼트려야 한다. $L\times L$ 라틴 방진 행렬 $\mathbf{u}_L$은 서로 다른 L개의 문자로 이루어져 있으며, 모든 행과 열은 L개의 문자의 치환 형태로 이루어져 있다[10]. U 행렬을 $L\times L$ 라틴 방진 행렬의 열 방향으로의 반복된 형태로 정의하겠다. 라틴 방진 행렬로 구조화된 U행렬을 라틴 방진 인터리버라 부르겠다. 라틴 방진 인터리버는 라틴 방진 행렬의 반복되는 형태 때문에 각각의 열벡터에 중복되는 서브블록의 인덱스 분포가 균일하다. 즉 최대한 같은 블록에 있는 심벌들을 다른 블록으로 흩어뜨릴 수 있다.

3.1 라틴 방진 인터리버

짧거나 중간 크기의 블록에서는 일반적으로 4 정도의 병렬 처리 수준이 고려되어 진다. 최적화 과정의 복잡성을 줄이기 위해 축소된 (reduced) 라틴 방진 구조를 이용하면 최적화 과정이 더 간단해진다. 즉 $\mathbf{u}_4$의 첫 번째 행을 $(0,1,2,3)$으로 고정시킨다[10]. 4×4 라틴 방진의 가능한 모든 경우의 수인 576가지 있는데 축소된 라틴 방진 구조의 경우 총 24가지 밖에 존재 하지 않는다. 라틴 방진 인터리버는 라틴 방진 행렬로부터 설계 되므로 즉 조사해야할 라틴 방진 인터리버의 경우의 수는 24가지가 된다. 더구나 그 중에서 좋은 경우를 선택하게 되면 이 숫자를 더욱더 줄 일 수 있다.

행렬의 각 열벡터는 부-블록의 치환 패턴을 의미하며 각 열벡터에서의 패턴의 분포는 성능에 직접적인 영향을 주게 된다. U행렬 열벡터의 연속적인 2-tuple 패턴의 분포를 살펴 볼 수 있는데 예를 들어 $\mathbf{u}_4$의 순환적인 열 방향에 따라 $(0,y)$의 분포를 살펴 볼 수 있다. 여기서 $y\in\{1,2,3\}$이다. 그림 1에

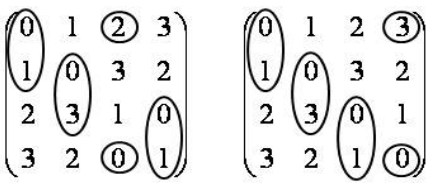

그림 1. (a) 좋은 분포와 (b) 나쁜 분포를 가지는 2-tuple 패턴의 분포 비교

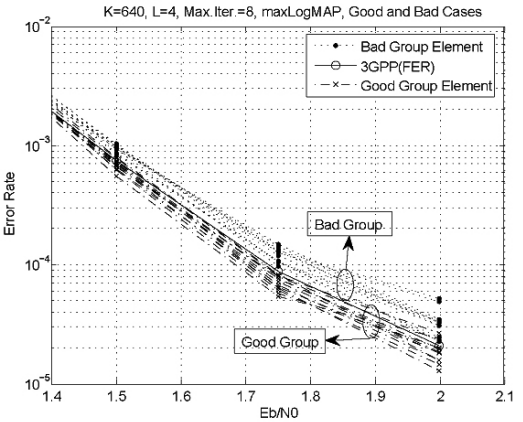

그림 2. 좋은 그룹과 나쁜 그룹의 FER 성능 비교

좋은 분포의 경우와 나쁜 분포의 경우가 나타나 있다. 그림 1. (a)은 $(0,2),(0,3)$이 한번 $(0,1)$이 두 번 존재 하지만, 그림 1. (b)는 $(0,1)$과 $(0,3)$의 패턴만이 두 번씩만 존재한다. 따라서 그림 1. (a)가 그림 1. (b)보다 더 이상적인 분포에 가깝기 때문에 더 좋은 성능을 보일 거라고 예상할 수 있다.

그림 1. (a)와 같은 분포를 가지는 라틴 방진을 좋은 그룹이라고 하고 1. (b)와 같은 분포를 가지는 라틴 방진을 나쁜 그룹이라고 하면 24가지의 축소된 라틴 방진 중에서 12개의 좋은 그룹과 12개의 나쁜 그룹으로 분리된다. 이 두 그룹으로 이루어진 라틴 방진 인터리버의 E_b/N_0에 대한 FER (frame error rate)성능 비교 곡선이 그림 2에 나타나있다. 시간적 치환으로는 길이 160의 3GPP 표준 인터리버를 사용했다. 부호화기는 3GPP 표준 부호화기로 구성 컨벌루션 부호의 발생 행렬은 $[1,(1+D+D^3)/(1+D^2+D^3)]$이며, 전체 부호율은 1/3이다. 복호 알고리듬으로 max log-MAP을 사용하였고 최대 반복복호 횟수를 최대 8번으로 하고 Genie 정지 알고리듬을 사용하였다. 즉 반복 복호 과정에서 복호된 정보비트에 오류가 없으면 복호를 중단하였다. 병렬 처리 수준은 4인 경우를 고려하였

다. 그림 2에서 좋은 그룹과 나쁜 그룹의 FER 곡선이 길이 640의 3GPP 인터리버의 FER 곡선을 기준으로 차이가 나며, 좋은 그룹의 인터리버 성능들이 더 좋은 성능을 가지는 것을 확인 할 수 있다.

3.2 확장 라틴 방진 인터리버

중간 크기나 긴 블록 길이에서는 4보다 큰, 예를 들어 8이나 12와 같은 병렬 처리 수준을 고려 할 수 있다. 하지만 8×8 축소된 라틴 방진 행렬의 경우 존재하는 경우의 수는 적어도 2.3×10^{10}개가 있다[11]. 그렇기 때문에 최적화를 위해서 모든 가능한 경우의 수를 고려 할 수 없다. 그 대신 $L\times L$ 라틴 방진 행렬을 $l\times l$ 라틴 방진 행렬로부터 확장하여 구할 것이다. 여기서 $L=nl$이고, $n\in\{2,3,4,...\}$이다. 확장된 $L\times L$ 라틴 방진 행렬은 다음 수식과 같다.

$$\mathbf{u}_L=(\mathbf{u}_l)^{(n)}=\begin{pmatrix}a_{11}\,a_{12}\cdots a_{1l}\\ a_{21}\,a_{22}\cdots a_{2l}\\ \vdots\ \ \vdots\ \ \ddots\ \ \vdots\\ a_{l1}\ \ a_{l2}\cdots a_{ll}\end{pmatrix}^{(n)}=\begin{pmatrix}\mathbf{a}_1\\ \mathbf{a}_2\\ \vdots\\ \mathbf{a}_l\end{pmatrix}^{(n)}$$

$$=\begin{pmatrix}\mathbf{a}_1 & \mathbf{a}_1^{(1)} & \cdots & \mathbf{a}_1^{(n-1)}\\ \mathbf{a}_2^{(n-1)} & \mathbf{a}_2 & \cdots & \mathbf{a}_2^{(n-2)}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{a}_1^{(1)} & \mathbf{a}_1^{(2)} & \cdots & \mathbf{a}_1\\ \mathbf{a}_2 & \mathbf{a}_2^{(1)} & \cdots & \mathbf{a}_2^{(n-1)}\\ \vdots & \vdots & \ddots & \vdots\\ \mathbf{a}_1^{(n-1)} & \mathbf{a}_1 & \cdots & \mathbf{a}_1^{(n-2)}\\ \mathbf{a}_2^{(n-2)} & \mathbf{a}_2^{(n-1)} & \cdots & \mathbf{a}_2^{(n-3)}\\ \vdots & \vdots & \ddots & \vdots\end{pmatrix} \quad (5)$$

여기서 $(\mathbf{u}_l)^{(n)}=\mathbf{u}_{l\cdot n}$, $\mathbf{a}_i=(a_{i1},a_{i2},...,a_{il})$, $\mathbf{a}_i^{(k)}=(a_{i1}^{(k)},a_{i2}^{(k)},...,a_{il}^{(k)})$, 그리고 $a_{i,j}^{(k)}=a_{i,j}+l\cdot k$이다. 수식 4에 의해 확장된 $L\times L$ 행렬은 각 행과 열이 서로 다른 L개의 심벌의 치환 형태가 되므로 라틴 방진 행렬이 된다. 또한 $\mathbf{u}_L$의 첫 번째 행은 $(0,1,...,L-1)$값을 가지므로 축소된 형태가 된다. $\mathbf{u}_8$은 수식 5를 통해 $\mathbf{u}_4$로부터 확장되어지며, 총 24가지 $\mathbf{u}_4$의 경우의 수가 존재 하므로 $\mathbf{u}_8$ 또한 총 24가지의 경우의 수가 존재하고, 최적화를 위해 이 경우의 수들만 조사하면 된다.

Ⅳ. 모의실험과 토의

이번 장에서는 제안한 라틴 방진 인터리버를 ARP 그리고 QPP 인터리버와 성능 비교 및 분석할 것이다.

먼저 정보 블록 길이 320과 640에 대해서 병렬 처리 수준 L이 4인 경우, 라틴 방진 인터리버와 ARP를 비교하였다. 제안된 인터리버에서 시간적 치환으로 각각 길이 80, 160의 3GPP 표준의 인터리버를 사용하였고, 라틴 방진 인터리버에 사용된 라틴 방진 행렬은 다음 수식 (6)와 같다.

$$\mathbf{u}_{4,1}=\begin{pmatrix}0\,1\,2\,3\\ 1\,0\,3\,2\\ 3\,2\,0\,1\\ 2\,3\,1\,0\end{pmatrix},\quad \mathbf{u}_{4,2}=\begin{pmatrix}0\,1\,2\,3\\ 2\,3\,0\,1\\ 3\,0\,1\,2\\ 1\,2\,3\,0\end{pmatrix} \quad (6)$$

12개의 좋은 그룹의 후보들 중에서 $K=320$일 때는 $\mathbf{u}_{4,1}$가 $K=640$일 때는 $\mathbf{u}_{4,2}$가 가장 좋은 성능을 보여주었다. ARP의 경우 3GPP2 표준화 작업에 제안된 적이 있으며 그 때의 인터리버 파라미터들은 다음과 같다[9].

- $P=197, L=4, \alpha=(0,0,1,1), \beta=(0,2,5,3), \gamma=3$ for $K=320$
- $P=201, L=4, \alpha=(0,0,1,1), \beta=(0,6,3,1), \gamma=3$ for $K=640$

그림 3은 블록 길이가 $K=320, 640$일 때 E_b/N_0에 대한 FER 성능 곡선이다. 다른 모의실험 환경은 3장에서와 동일하다. $K=320$일 때 제안된 라틴 방진 인터리버는 ARP와 거의 동일한 성능을 보여주고 있다. $K=640$일 때는 라틴 방진 인터리버와 ARP가 약 1.25dB까지는 비슷한 성능을 보여주다가 낙수 지역(waterfall region)에서는 ARP가 약간 더 좋은 성능을 보여 주고 있다. 하지만 ARP는 오류 마루 (error floor) 현상이 빨리 일어났으며 약 2.25dB 이후로는 라틴 방진 인터리버가 더 좋은 성능을 보여 주고 있으며 오류 마루 현상도 거의 일어나지 않고 있다.

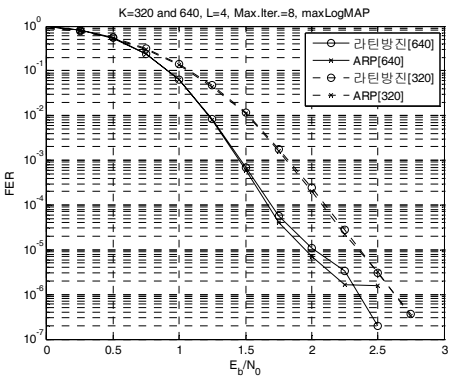

그림 3. 라틴 방진 인터리버와 ARP 인터리버의 성능 비교

정보 블록 길이 1024에 대해서 병렬 처리 수준 L이 8인 경우, 라틴 방진 인터리버와 QPP 인터리버를 비교하였다. 라틴 방진 인터리버의 시간적 치환으로는 길이 128의 3GPP 표준 인터리버를 사용하였고, 라틴 방진 인터리버에 사용된 라틴 방진 행렬은 수식 (7)과 같다.

$$\mathbf{u}_8 = \begin{pmatrix} 0123 \\ 1230 \\ 3012 \\ 2301 \end{pmatrix}^{(2)} = \begin{pmatrix} 01234567 \\ 56741230 \\ 30127456 \\ 67452301 \\ 45670123 \\ 12305674 \\ 74563012 \\ 23016745 \end{pmatrix} \quad (7)$$

$\mathbf{u}_8$은 24개의 후보들 중에서 가장 좋은 성능을 보여주었다. QPP 인터리버의 파라미터로 f_1은 31이고 f_2는 64이다. 그림 4는 그림 3의 $K=640$경우와 비슷한 결과를 보여 주고 있다. QPP 인터리버가 낙수 지역에서 약간 더 좋은 성능을 보여 주고 있지만 오류 마루 현상으로 인해 성능 역전이 일어난다. 제안된 인터리버는 오류 마루 현상을 잘 억제 할 수 있음을 보여주고 있다.

표 1은 제안된 라틴 방진 인터리버와 다른 인터

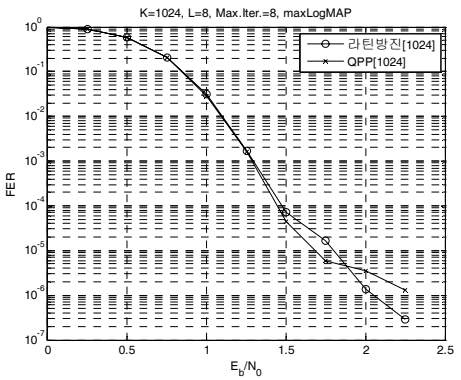

그림 4. 라틴 방진 인터리버와 QPP 인터리버의 성능 비교

표 1. 여러 인터리버의 최적화 복잡도의 비교

	$K=320, L=4$	$K=640, L=4$	$K=1024, L=8$
라틴방진	12	12	24
ARP	40960	81920	1073741824
QPP	15200	55360	261632

리버들의 최적화를 위한 복잡도의 비교를 하고 있다. ARP의 경우 α와 γ가 고정되어 있다고 가정하였고, $k=0(\mathrm{mod}L)$에 대하여 $\beta(k)$가 0값을 가지고 다른 $k\neq 0(\mathrm{mod}L)$에 대하여는 1부터 8 사이의 값을 가진다고 가정하였다[7]. 병렬 처리의 차수가 L인 시스템에 최적화된 ARP를 설계하기 위해서 고려해야 하는 경우의 수는 $|P|\cdot 8^{L-1}$이 되며, 여기서 $|P|$는 P의 cardinality이다. QPP 인터리버에서는 전체 블록 길이 K가 4로 나눠지는 경우, f_1은 K와 서로소인 양의 정수 값이며, f_2는 K를 소수의 곱의 형태로 표시 할 경우 그 소수들을 원소로 가지는 양의 정수 값들이다[8]. 그러므로 총 $|f_1|\cdot|f_2|$ 가지의 경우의 수가 존재 한다. 제안된 라틴 방진 인터리버는 L이 4일 때 12가지 경우의 수를, 8일 때는 24가지의 경우의 수만 고려하면 된다. 즉 블록 길이가 증가하거나 병렬 처리의 차수가 증가하더라고 최적화를 위한 복잡도가 크지 않으면서도, 성능 또한 좋은 것을 알 수 있다. 제안된 방법에서 시간적 치환으로 어떤 종류의 미리 정의된 인터리버라도 사용 가능하다.

Ⅴ. 결 론

본 논문은 병렬 처리 구조의 터보 부호에서 충돌 방지 인터리버를 제안하였다. 기존에 제안된 인터리버를 시간적 치환으로 사용하고 사상행렬 U를 정의함으로써 다양한 길이의 충돌 방지 인터리버를 만들어 낼 수 있다. 라틴 방진 행렬 구조의 사상 행렬을 사용하면 ARP나 QPP 인터리버 보다 더 간단한 최적화 과정을 갖는다. 제안한 라틴 방진 행렬은 최적화를 위하여 블록 길이에 상관없이 병렬 처리의 차수가 4인 경우 12가지의 경우를 고려하면 되고, 차수가 8인 경우 24가지의 경우를 고려하면 된다. 더욱이 ARP나 QPP 인터리버와 거의 같은 성능을 보여주고 있으며 오류 마루 현상이 억제가 되어 높은 신호 대 잡음비에서 좋은 성능을 보여주었다.

참 고 문 헌

〔1〕 C. Berrou, A. Glavieux and P. Thitimajshima, “Near Shannon limit errorcorrecting coding and decoding: turbo-codes,” *Proc. of IEEE ICC’’93*, Geneva, pp.1064-1070, May 1993.

〔2〕 L. Bahl, J. Cocke, F. Jelinek, and J. Raviv,

"Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, Vol.20, pp.284-287, Mar. 1974.

〔3〕 J. B. Anderson and M. Hladik, "Tailbiting MAP decoders," *IEEE J. Select. Areas Commun.*, Vol.16, pp.297-302, Feb. 1998.

〔4〕 C. Weiß, C. Bettstetter, and S. Riedel, "Code Construction and Decoding of Parallel Concatenated Tail-Biting Codes," *IEEE Trans. Inf. Theory*, Vol.47, pp.368-388, Jan. 2001

〔5〕 A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo and LDPC decoder architectures," *IEEE Trans. Inf. Theory*, Vol.50, No.9, pp.2002~2009, Sep. 2004.

〔6〕 D. Gnaedig, E. Boutillon, M. J´ez´equel, V. Gaudet, and P. Gulak, "On multiple slice turbo codes," *in Proc. 3rd Int. Symp. on Turbo Codes and Related Topics*, Brest, France, pp.343-346, Sep. 2003.

〔7〕 C. Berrou, S. Kerouedan Y. Saouter, C. Douillard, and M. J´ez´equel, "Designing good permutations for turbo codes: towards a single model," *in Proc. Int. Conf. Commun.*, Paris, France, Vol.1, pp.341-345, Jun. 2004.

〔8〕 O. Y. Takeshita, "On Maximum Contention-free Interleavers and Permutation Polynomials Over Integer Rings," I*EEE Trans. Inf. Theory*, Vol.52, No.3, pp.1249-1253, Mar. 2006.

〔9〕 3GPP TSG RAN WG1-43, "Enhancement of Rel. 6 Turbo Code," Nov. 2005.

〔10〕 Charles J. Colbourn and Jeffrey H. Dinitz, *The CRC Handbook of Combinatorial Designs*, 2nd edition, CRC Press, pp.97-110, 1996

〔11〕 Jr. Marshall Hall, *Combinatorial Theory*, 2nd edition, John Wiley & Sons, 1996.

김 대 선 (Dae-Son Kim) 준회원

2001년 2월 건국대학교 전자공학과 졸업 (공학사)
2003년 2월 연세대학교 대학원 전기전자공학과 졸업 (공학석사)
2006년 현재 연세대학교 대학원 전기전자공학과 박사과정
<관심분야> Error Correcting Codes, Turbo code, LDPC, MC-CDMA, Spread Spectrum Communication Systems

오 현 영 (Hyun-Young Oh) 준회원

2005년 2월 연세대학교 전기전자공학과 졸업 (공학사)
2007년 2월 연세대학교 대학원 전기전자공학과 졸업 (공학석사)
2006년 현재 삼성전자 연구원
<관심분야> Error Correcting Codes, Turbo code, LDPC

송 홍 엽 (Hong-Yeop Song) 종신회원

1984년 2월 연세대학교 전자공학과 졸업 (공학사)
1986년 5월 USC 대학원 전자공학과 졸업 (공학석사)
1991년 12월 USC 대학원 전자공학과 졸업 (공학박사)
1992년~1993년 Post Doc., USC 전자공학과
1994년~1995년 8월 Qualcomm Inc., 선임연구원
2002년 3월~2003년 2월 University of Waterloo, Canada, 방문연구교수
1995년 9월~현재 연세대학교 전기전자공학부 교수
<관심분야> PN Sequences, Error Correcting Codes, Spread Spectrum Communication Systems, Steam Cipher Systems