다단 확산 방식을 이용한 효율적인 OVSF 코드 생성 기법

준회원 최 창 순*, 김 태 훈*, 정회원 김 영 록*, 정 화 용**

Fast OVSF Code Generation Method using Multi-Stage Spreading Scheme

Changsoon Choi*, Taehoon Kim*, *Associate Members*,
Younglok Kim*, Hwayong Joung** *Regular Members*

요 약

본 논문에서는 OVSF 코드의 단일 코드 인덱싱 기법과 이를 기반으로 하는 다단 확산 방식을 이용한 효율적인 코드 생성 기법을 제안한다. 기존의 코드 트리 구조(code-tree structure) 기반의 인덱싱 방식에서는 확산 계수와 코드 번호라는 두 개의 인덱스를 이용하여 하나의 코드워드(codeword)를 표시한 반면에 단일 코드 인덱싱 기법에서는 하나의 코드 인덱스만을 사용하여 각 층의 코드워드의 확산 계수와 코드 번호를 모두 표시할 수 있다. 단일 코드 인덱스의 이진 표현은 코드워드의 패턴을 나타내 주어 코드워드를 코드 인덱스로부터 직접 생성할 수 있게 할 뿐만 아니라, 두 개의 다른 코드워드의 직교 여부를 코드 인덱스의 비교로 결정할 수 있게 한다. 본 논문에서는 긴 코드의 확산을 여러 단계의 짧은 코드의 확산으로 나누어 실행하도록 하는 다단 확산 방식을 코드 생성에 적용하여 3GPP UMTS 시스템을 위한 빠르고 효율적인 코드 생성기를 설계하고 검증하였다.

Key Words : OVSF Code, DS-CDMA, Spreading Code

ABSTRACT

This paper proposes the fast OVSF code generation method using the multi-stage spreading scheme based on the single code indexing scheme. The conventional OVSF indexing scheme based on the code-tree structure uses two numbers as the codeword indices, the layer number and the code number of the corresponding layer. However, the single code index number implicitly includes the information of the spreading factor as well as the code number. Since the binary representation of the single code indices shows the pattern of the codeword, the orthogonality between two different codewords can be determined by comparing their code indices instead of much longer codewords. The above useful property also makes the codeword can be generated directly from its single code index. In this paper, the multi-stage spreading scheme is applied to generate the long code by spreading two shorter codewords with the appropriate code indices. The proposed fast code generation algorithm is designed for 3GPP UMTS systems and verified by the simulations.

* 서강대학교 전자공학과 (cschoi1@sogang.ac.kr, thkim7@sogang.ac.kr, ylkim@sogang.ac.kr)
** 텔슨전자 중앙연구소 (hyjoung@willcomm.com)
논문번호 : 040150-0412, 접수일자 : 2004년 4월 19일
※본 논문은 2003년도 서강대학교 교내 연구비 지원에 의하여 이루어 졌으며, 설계 Tool은 IDEC 지원을 받았습니다.

I. 서 론

OVSF(Orthogonal Variable Spreading Factor) 코드는 이동통신 표준화 기구인 3GPP(3rd Generation Project Partnership) [1]에서 UMTS(Universal Mobile Telecommunications System) 표준으로 채택된 채널 코드(channelisation code)이다. 채널 코드는 같은 셀 내의 같은 슬롯(slot)에서 다중 접속을 위해서 이용된다. 특히 OVSF 코드 기술은 서로 다른 확산 계수를 가지면서도 직교성이 유지되는 코드워드를 제공하기 때문에 3세대 무선 통신에서 요구되는 다양한 전송률을 가진 여러 유동적인 서비스들을 동시에 제공할 수 있게 한다.

Gilhousen[2]은 가변 길이의 왈쉬 코드(Walsh code)를 이용하여 다른 전송률을 지원하는 OVSF 코드를 할당하는 방법을 제안하였다. OVSF 코드를 구하기 위한 기존의 또 다른 기법인 트리 구조는 [3]에서 연구되었다. 이 기법들은 개선된 Hadamard 변환을 기반으로 착안 되었으며 어떤 특정한 OVSF 코드를 나타내기 위해서는 확산 계수와 코드 번호를 표시하는 두 개의 인덱스가 필요하다는 결점이 있다. 무선 통신 시스템의 기지국은 단말기로부터 새로운 서비스의 요청을 받으면, 기존의 코드와 직교성이 보장되는 코드를 할당해 주어야 한다. 기존의 인덱싱 기법에서는 이러한 코드 할당을 위하여 이미 할당된 코드의 목록(ASSIGNED code list)과 할당되지는 않았지만 할당된 코드와 직교하지 않아서 사용할 수 없는 코드 군의 목록(BUSY code list)이 메모리에 저장되어있어야 한다[4].

이러한 목록을 이용하여 다중 코드전송(multi-code transmission)을 이용한 다중 전송률 서비스를 제공하는 단말기에서의 효과적이고 최적화된 코드 할당 기법은 [5]에서 제안되었다. 기존의 코드 인덱싱 기법을 기반으로 한 이러한 코드 생성 기법이나 코드 할당 기법은 시스템에서 요구하는 최대 확산 계수에 따라서 메모리 소요가 기하급수적으로 증가하여, 메모리 접근 시간이 길어지게 되어 코드의 할당과 생성을 느리게 하는 요인이 된다. 이를 극복하기 위해서 단일 코드 인덱스 방식이 [6]에서 제안되었다.

본 논문에서는 [6]에서 제안된 OVSF 코드를 위한 단일 코드 인덱스를 소개하고, 단일 코드 인덱스의 다양한 장점 중의 하나인 다단 확산 방식을 이용한 빠른 코드 생성 기법을 제안한다. 서론에 이어 2장에서는 기존의 트리 구조를 기반으로 한 코드의 생성과 할당에 대해서 설명하고, 3장에서는 Kronecker product를 기반으로 하는 단일 코드 인덱싱 기법에 대한 소개와 장점들을 설명한다. 단일 코드 인덱싱 기법을 이용한 OVSF 코드 생성기의 회로 설계와 검증의 결과가 4장에서 기술되며, 마지막으로 5장에서 본 논문의 결론을 맺는다.

II. 트리 구조를 기반으로 한 코드 생성

열 벡터인 $C_N(p)$가 확산 계수 $N = 2^L$ 을 가지는 OVSF 코드워드(codeword)라 정의하고, 이때 p와 L은 코드 번호와 레이어(layer) 번호를 나타낸다. 코드 트리 구조를 기반으로 한 기존의 OVSF 코드의 인덱싱 방식은 하나의 코드워드를 지정하기 위해서 확산 계수와 코드 번호라는 두 개의 인덱스가 필요하다. 지정된 확산 계수와 코드 번호에 따른 코드워드는 그림 1에서 보여주는 코드 트리로부터 반복적으로 생성된다[3]. 참고 문헌 [2]와 [5]의 정의에 따라 모코드(mother code)는 특정 코드에서 루트 코드(root code) $C_1(0)$ 까지의 경로 중에 더 낮은 단계(layer)의 코드가 되고, 자코드(descendent code)는 그 특정 코드가 모코드인 모든 코드가 된다. 예를 들면, $C_8(2)$의 모코드는 $C_4(1)$, $C_2(0)$, $C_1(0)$이고, $C_4(1)$의 자코드는 $C_8(2)$, $C_8(3)$와 이것들의 모든 자코드로 나타내어진다.

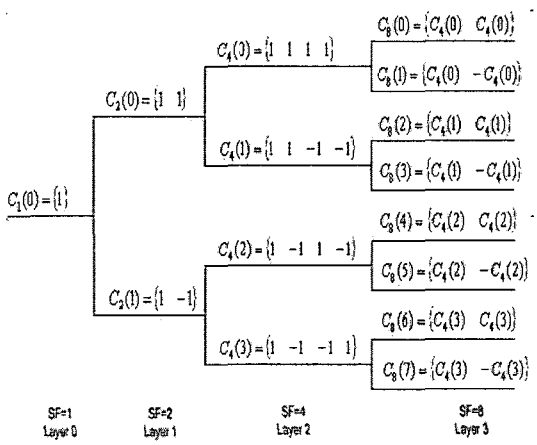


그림 1. OVSF 코드의 트리 구조

코드 트리 구조에서는 서로 다른 확산 계수를 가지는 두 개의 OVSF 코드가 같은 가지에 위치해 있다면 이것들은 서로 직교하지 않는 다는 것을 보여준다. 어떤 코드가 다른 한 코드와 모코드나 자코드의 관계에 있다면, 이 두 코드는 같은 가지에 위

치한다는 것이며, 직교하지 않는다는 것을 알 수 있다. 따라서 자코드나 모코드의 관계에 있는 두 코드는 동시에 할당되어 사용될 수 없다. 다시 말해 어떠한 두 개의 코드가 직교(orthogonal)하다면 이것들은 서로 다른 코드의 모코드도 자코드도 아니라는 것을 알 수 있다. 요청된 통신 서비스를 만족하는 전송률을 지원하는 확산 계수를 가진 코드가 필요하다면, 그 코드의 모코드나 자코드에 해당하는 모든 코드가 사용되고 있지 않아야 한다. 특정한 데이터 전송 속도의 새로운 통화가 설정되도록 요구되면, 시스템은 전송 속도에 맞는 확산 계수를 계산하여 새로운 코드를 할당한다. 이 새로운 코드와 이미 할당된 코드간의 직교성(orthogonality)을 유지하기 위해서는 새로운 코드가 할당 될 때마다 사용 가능한 코드의 목록을 생성하고 갱신해야 한다. 이 목록은 할당된 코드 자신과 그 코드의 모든 자코드와 모코드들을 제거함으로써 갱신되며, 최대 확산 계수에 따라서 필요한 메모리의 크기가 기하급수적으로 늘어나게 된다.

III. 단일 코드 인덱싱 기법

1. OVSF 코드의 Kronecker product 표현 방식

앞에서 설명한 코드 트리 구조에서의 코드워드는 다음과 같이 Right Kronecker product로 나타낼 수 있다 [6].

$$C_{2N}(p) = C_2(m) \otimes C_N(k), \qquad (1)$$

여기서

$$p = 2 \cdot k + m, \qquad (2)$$

이고 k=0, 1,…, N-1, m=0 또는 1 이며, 코드의 초기값들은 아래와 같다.

$$C_1(0) = 1,\ C_2(0) = [1, 1],\ C_2(1) = [1, -1] \qquad (3)$$

위의 표현을 확장하면 확산 계수 $N = 2^L$의 값을 가지는 코드워드를 확산 계수가 2인 L개의 분리된 코드의 Kronecker product로 다음과 같이 나타낼 수 있다.

$$C_N(p) = \underbrace{C_2(a_{L-1}) \otimes \cdots \otimes C_2(a_1) \otimes C_2(a_0)}_{L} \qquad (4)$$

여기서 코드 번호 p는 다음과 같이 표현된다.

$$p = a_0 \cdot 2^{L-1} + a_1 \cdot 2^{L-2} + \cdots + a_{L-1} = \sum_{i=0}^{L-1} (a_i \cdot 2^{(L-i-1)}) \qquad (5)$$

여기서 $a_i \in \{0, 1\}$ 이다. 이와 같은 코드의 표현은 두 코드의 관계를 알아내기 유용한 다음과 같은 특성들을 가지고 있으며, 이런 특성들은 트리 구조와 수식 (4)를 이용하여 쉽게 증명될 수가 있다.

특성 1: Kronecker product로 표현된 임의의 코드 $C_N(p) = C_2(a_{L-1}) \otimes \cdots \otimes C_2(a_1) \otimes C_2(a_0)$의 모든 모코드는 $[C_2(a_{L-m}) \otimes \cdots \otimes C_2(a_1) \otimes C_2(a_0)]$로 표현되며, 여기서 $m = \{2, 3, \cdots, L\}$ 이다.

특성 2: 임의의 코드 $C_N(p)$의 모든 자코드는 $C_M(q) \otimes C_N(p)$으로 표현될 수 있으며, 여기서 M과 q는 $M = 2^K$, $K = 1, 2, \ldots$의 임의 값이다.

특성 3: 모든 코드는 2개 이상의 코드의 Kronecker product로 표현될 수 있다.
즉, $C_N(p) = C_M(q) \otimes C_L(r)$로 표현할 수 있으며, 여기서 $N = M \cdot L$이고,

$$q = \sum_{i=0}^{\log M - 1} \left(a_i \cdot 2^{(\log M - i - 1)} \right),$$

$$r = \sum_{i=\log M}^{\log N - 1} \left(a_i \cdot 2^{(\log N - i - 1)} \right)$$

이다. 수식을 간편하게 표시하기 위해서 $log_2 L$을 $logL$로 대신하여 표기하였다.

특성 1과 2를 이용하여 코드를 생성하기 전에 코드 인덱스만을 비교해서 확산 계수가 다를 수 있는 두 코드의 직교 관계를 알 수 있으며, 특성 3은 특정 코드를 두 개 이상의 코드로 나누어서 한 코드를 다른 코드로 확산시키는 방식으로 긴 코드를 생성시킬 수 있게 한다.

2. 단일 코드 인덱스 기법

참고 문헌 [6]에서 코드 인덱스를 다음과 같이 나타내었다.

$$(\underbrace{0, 0, \cdots 0}_{SF\ part}, 1, \underbrace{a_0, a_1, \cdots, a_{L-1}}_{code\ number\ part}) \qquad (6)$$

이러한 방식으로 확산 계수와 코드 번호를 하나의 인덱스로 나타낸다. 수식 (6)의 이진수의 전체 길이는 최대 확산 계수에 의해서 정해지며, 왼쪽부터 최초의 '1'을 만나기 전의 '0'의 개수가 지정된 코드

의 확산 계수를 나타낸다. 즉, '0'의 개수가 N이고 최대 확산 계수가 2^M 이라면 이 코드의 확산 계수는 $2^{(M-N)}$ 이 된다. 최초의 '1'이후의 이진수는 확산 계수에 따른 코드 번호를 나타낸다. 특정한 코드워드를 지정하기 위해서 동일한 길이의 이진수가 사용되었으며, 앞 절에서 보여준 바와 같이 이러한 이진수의 표현은 그 코드 인덱스 자체만으로 각 트리 레벨의 모든 가지를 나타낼 수 있다. 이것은 메모리에 저장된 사용 가능한 코드 목록의 갱신 없이 단지 현재 할당된 코드의 목록만을 가지고 요구되는 확산 계수의 새로운 코드 인덱스를 결정할 수 있다는 것을 의미한다. 표 1에서는 기존 트리 구조에서의 코드 번호와 단일 코드 인덱스의 표현을 비교하여 보여준다. 확산 계수 8까지의 OVSF 코드워드를 나타내고 있으며 이는 임의의 높은 확산 계수까지 확장될 수 있다. 첫 번째 열은 레이어 번호(또는 SF)와 확산계수에 따른 코드 번호를 나타내는 기존의 코드 인덱스이고, 두 번째 열은 이진 표현 값으로 대응된 단일 코드 인덱스를 보여준다. 그리고 세 번째 열에서는 새로운 이진 표현 값의 십진 표현 값을 보여주고 있다. 마지막 네 번째 열에서는 인덱스에 대응되는 OVSF 코드워드를 보여주고 있다.

표 1. 기존의 코드와 비교한 단일 코드 인덱스의 표현

기존의 코드 인덱스 (SF/Code 수)	이진 표현 값의 단일 코드 인덱스 (a0,a1,...)	십진 표현 값의 새로운 코드표	OVSF 코드워드 (0은 -1을 나타낸다)
1/0	0001	1	1
2/0	0010	2	11
2/1	0011	3	10
4/0	0100	4	1111
4/1	0101	5	1100
4/2	0110	6	1010
4/3	0111	7	1001
8/0	1000	8	11111111
8/1	1001	9	11110000
8/2	1010	10	11001100
8/3	1011	11	11000011
8/4	1100	12	10101010
8/5	1101	13	10100101
8/6	1110	14	10011001
8/7	1111	15	10010110

3. 단일 코드 인덱스 기법의 장점

앞에서 소개된 단일 코드 인덱싱 기법은 기존의 코드 트리 구조에 비하여 다음과 같은 장점들을 가지고 있다.

(1) 코드 인덱스를 나타내기 위한 비트 수가 줄어든다.

기존의 코드 인덱싱은 최대 확산 계수 2^L 을 나타내기 위해서 최대 코드 번호의 길이 L과 확산 계수를 지정하는 $\lceil \log_2(L)-1 \rceil$ 개의 합인 총 $L+\lceil \log_2(L)-1 \rceil$ 의 비트수가 필요한 반면, 단일 코드 인덱스 기법은 (L+1)비트만을 필요로 한다. 예를 들어 표 2와 같이 최대 확산 계수를 512라고 하면 기존의 코드 인덱스 방식에서는 확산 계수값 10개 {1, 2, 4, 8, 16, 32, 64, 128, 256, 512}를 저장하기 위해서 4비트가 필요하고 마지막 레이어에서 512개의 코드 번호를 부여하기 위해서는 9비트가 필요하여 총 13비트가 필요하다. 반면에 단일 코드 인덱싱 기법은 10비트만을 필요로 하기 때문에 3비트가 절약 된다. 총 13비트에서 3비트 감소는 메모리나 코드 인덱스를 전달하는 데 있어서 25% 정도의 절약을 가져온다. 특히 기지국에서 단말기로 할당된 코드의 목록을 전송할 때 25%의 비트 수를 절약한다는 것은 시스템 파라미터 시그널링(parameter signalling) 관점에서 매우 큰 이득이라고 할 수 있다. 결국, 트리 구조에서는 사용 중인 모든 코드 인덱스와 코드워드를 저장하기 위해서 최대 512×(13+512) = 268,800 비트의 LUT(Lookup Table)이 필요하지만, 단일 코드 인덱싱 기법은 사용 중인 코드워드의 인덱스만을 저장할 수 있는 512×10 = 5120 비트의 메모리만 필요하기 때문에 메모리 사이즈를 현격히 줄일 수 있다.

표 2. 최대 확산 계수가 512인 경우의 사용된 메모리 비교

구조	length of Index	Size for LUT	Memory for comparison
트리 구조	13 bits	268,800 bits	13,312 bits
제안된 구조	10 bits	5,120 bits	0

(2) 두 코드간의 직교 여부를 코드 인덱스만으로 판단할 수 있다.

기존의 트리 구조에서는 두 코드워드를 직접 비교하는 방식으로만 직교 여부를 판단할 수가 있으며, 이는 코드워드를 생성한 후에야 가능한 작업이다. 따라서 새로운 코드를 할당하기 위해서는 각 후

보 코드워드를 생성한 후 이미 사용되고 있는 각각의 코드워드와 비교해야 하는 두 가지 작업에 시간 소요가 많다. 게다가 사용 중인 모든 코드의 모코드와 자코드의 인덱스 집합을 메모리에 저장해야 하기 때문에 $2\times(512\times13)$ = 13,312 비트의 메모리가 추가로 할당되어야 한다.

반면에 단일 코드 인덱스를 사용하면 요구된 확산 계수를 만족하는 사용 가능한 코드 인덱스를 이미 사용되고 있는 코드 인덱스와 비교하여 할당 여부를 판단할 수 있다. 예를 들어 M < N일 때, 코드 $C(a_0, a_1, \cdots, a_N)$가 이미 사용되고 있는 코드라면, 특성 2, 3에 의해서 $C(a_0, a_1, \cdots, a_N)$의 모코드는 모두 $\{C(a_0, \cdots, a_M), C(a_0, \cdots, a_{M-1}), C(a_0, \cdots, a_{M-2})\}$이고 자코드는 $\{C(x, a_0, \cdots, a_N), C(a_0, \cdots, a_{M-2}), C(x, x, a_0, \cdots, a_N), C(x, x, x, a_0, \cdots, a_N)\}$이라는 것을 쉽게 알 수 있다. 여기에서 코드 인덱스$(0,0,\cdots,0,1,a_0,\cdots,a_{L-1})$의 코드워드는 $C(a_0, \cdots, a_{L-1})$ 로 나타내었다. 따라서 두 코드의 인덱스의 비교만으로 직교 여부를 판단하여 사용 가능한 코드의 집합을 쉽게 구하여 코드 할당을 용이하게 할 수 있다.

(3) 긴 코드의 생성과 확산이 용이하다.

확산 계수가 큰 코드워드는 확산 계수가 작은 여러 코드워드의 확산으로 얻을 수 있다. 확산 계수가 작은 코드의 인덱스는 특성 3에 의해서 확산 계수가 큰 코드의 인덱스에서 바로 알아낼 수 있다.

예를 들어, $N \le M$ 이라고 하면 OVSF 코드워드 $C(a_0, a_1, \cdots, a_M)$는 $C(a_0, a_1, \cdots, a_N)$ 와 $C(a_{N+1}, a_{N+2}, \cdots, a_M)$의 Kronecker product로 표현된다. 위의 수식 (4), (5)에서 보인 바와 같이 $C(a_0, a_1, \cdots, a_M) = C(a_M) \otimes \cdots \otimes C(a_0)$ 으로 정의된다. 따라서 긴 코드는 $C(a_0, a_1, \cdots, a_N)$를 $C(a_{N+1}, a_{N+2}, \cdots, a_M)$로 확산시킴으로서 얻을 수가 있다. 또한 전송 심볼을 확산 시킬 때, $C(a_{N+1}, a_{N+2}, \cdots, a_M)$와 $C(a_0, a_1, \cdots, a_N)$ 의 두 개의 짧은 코드의 반복적인 확산을 통해서 원래의 긴 코드인 $C(a_0, a_1, \cdots, a_M)$으로 확산시킨 것과 일치하는 효과를 얻을 수가 있어, 원래의 확산 코드가 필요 없는 경우에 부가적으로 계산을 줄일 수가 있다. 이를 다단 확산 방식(multi-stage spreading)이라고 부르기로 하겠다. 이 새로운 기법은 짧은 코드의 개수나 그 코드의 확산계수를 선택하는 데에 있어서 하드웨어 복잡도와 재사용 여부, 코드 생성 속도 등을 조절할 수 있게 한다.

(4) 간단하고 빠른 동적 채널 할당이 가능하다.

단일 코드 인덱싱 기법은 AVAILABLE과 BUSY 코드 목록을 간단하고 빠르게 제공함으로써 동적 코드 할당[5,7,8]에 유리하게 사용될 수 있다. 기존의 코드 인덱싱 기법에서는 사용 중인 코드의 모코드와 자코드를 찾기 위해서 메모리에 저장된 정보를 이용해야 하며 거기에 따른 시간적 손실이 크다. 하지만, 제안된 단일 코드 인덱싱 기법에서는 메모리 액세스 시간이 걸리지가 않아 빠른 동적 코드 할당을 가능하게 한다. 또한, 인덱스에 모코드와 자코드의 정보가 모두 있으므로 코드를 재할당해야 할 경우 트리 구조 기반의 방식에 비해서 좀 더 세련된 방식의 재할당을 가능하게 하여 간단하고 빠른 동적 코드 할당을 할 수 있게 한다.

Ⅳ. 다단 확산 방식을 이용한 OVSF 코드 생성기 설계

여기에서는 단일 코드 인덱싱 기법의 장점 중의 하나인 다단 확산 방식을 이용한 빠른 코드 생성기의 회로 설계와 검증 결과에 대해서 설명하기로 한다. 확산 계수가 최고 512와 16인 3GPP UMTS FDD (frequency division duplex)와 3GPP UMTS TDD (Time division duplex) 시스템의 이중 모드(dual mode)에서 사용할 수 있는 효율적인 코드 생성기 설계를 소개한다.

우선 3GPP UMTS TDD 모드에서 선택적으로 쓰일 최대 확산 계수가 16인 코드의 생성기를 설계하여 최대 512의 확산 계수를 갖는 코드의 생성에 이용한다. 확산 계수가 32 이상인 코드는 확산 계수가 16인 코드를 반복적으로 확산시켜 생성할 수 있게 함으로서 하드웨어의 복잡도를 최소화 한다. 확산 계수가 16인 코드의 생성기는 256 비트 크기의 LUT로 구현하거나, 메모리를 사용하지 않고 쉬프트 레지스터와 배타적 논리합(Exclusive-OR) 연산을 이용하여 OVSF 코드생성기를 설계할 수 있다. 여기서는 두 번째 방식으로 구현을 하였다.

우선 최고 확산 계수가 512인 코드워드를 위해서는 10비트의 코드 인덱스가 필요하다. 10비트의 코드 인덱스가 이 회로의 입력으로 들어가고 입력된 코드 인덱스에 대한 OVSF 코드워드가 출력된다.

회로의 동작을 간단하게 살펴보면, 입력으로 들어온 코드 인덱스는 10비트 쉬프트 레지스터로 구성되는 인덱스 픽스 블록(index fix block)에서 첫 번째 비트가 1이 될 때까지 오른쪽으로 쉬프트 된다. 3.2절에서 설명한 바와 같이 코드 인덱스의 0의 개수는 코드의 확산 계수를 나타내므로 쉬프트된 횟수로 코드 인덱스에 해당하는 확산 계수를 알 수 있으며, 이 블록은 10비트 카운터를 사용하여 간단하게 설계할 수 있다. 특히 3GPP UMTS TDD 모드에서 동작한다면 최대 확산 계수가 16이므로 총 10비트의 앞의 5비트는 무시하여 확산계수 16부터 시작하도록 선택할 수 있게 하면 5 클락 만큼의 시간을 절약할 수 있다.

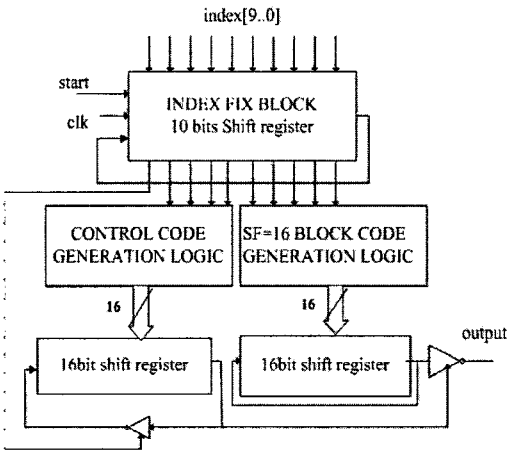


그림 2. 다단 확산 방식을 이용한 OVSF 코드 생성기의 블록 구성도

위의 과정을 거치게 되면 3.2절의 수식 (6)에서 보는 바와 같이 코드 인덱스의 첫 번째 비트가 처음 1이 나오는 비트이고 이후의 비트부터 코드 번호를 나타내는 비트가 되며 이것은 코드 생성 회로 블록의 입력으로 들어간다. 코드 생성 회로 블록(code generation logic block)은 인덱스가 0 혹은 1이냐에 따라 코드가 그대로 반복되거나 반전된 후에 반복되는 특성을 이용하여 15개의 배타적 논리게이트로 구성되고 그 과정은 다음과 같다. 인덱스의 첫 번째 비트가 SF=16인 코드의 첫 번째 비트가 된다. 코드의 첫 번째 비트와 인덱스 두 번째 비트의 배타적 논리합이 코드의 두 번째가 비트가 되고 코드의 첫 번째, 두 번째 비트와 인덱스의 세 번째 비트와의 각각의 배타적 논리합은 코드의 세 번째, 네 번째 비트가 된다. 이와 같은 방법으로 코드의 첫 번째에서 네 번째 비트와 인덱스의 네 번째 비트와의 배타적 논리합은 각각 코드의 다섯 번째 비트에서 여덟 번째 비트를 생성하고 코드의 첫 번째 비트에서 여덟 번째 비트와 인덱스의 다섯 번째 비트의 배타적 논리합으로 코드의 아홉 번째 비트에서 열여섯 번째 비트까지 생성되어 SF=16인 코드가 생성되고 16비트 쉬프트 레지스터에 로드(load)된다. 이와 같이 SF=16인 코드가 한번에 생성되기 때문에 Block code generation logic이라고 이름을 붙였다. 이와 동시에 SF=32인 컨트롤 코드가 나머지 인덱스 5비트로부터 생성된다. 마찬가지의 방법으로 SF=32인 컨트롤 코드는 인덱스의 나머지 부분의 상위비트에 1을 추가하여 6비트의 인덱스로 생성된다. 단지 차이가 있다면 실제 설계 시에는 1비트로 제어되는 인버터를 하나 추가하여 마지막 인덱스가 0 또는 1에 따라서 16비트 레지스터의 출력이 입력으로 바로 들어가던지 반전되어 들어가도록 하여 16비트 레지스터만으로 SF=32인 코드를 표현할 수 있도록 하였다. SF=32와 SF=16인 이 두 코드의 Kronecker product가 최대 SF=512인 전체 회로의 출력이 된다. Kronecker product는 OUTPUT에 연결된 1비트로 제어되는 인버터를 사용하여 수행된다. 다시 말하면, BLOCK CODE GENERATION LOGIC 으로부터 생성된 SF=16인 코드워드가 CONTROL CODE GENERATION LOGIC 으로부터 생성된 SF=32인 코드의 비트가 1 또는 0이냐에 따라서 SF=16인 코드가 그대로 출력으로 나가거나 반전되어 나가게 된다. 즉, SF=32인 코드 한 비트가 SF=16인 코드를 제어 하는 것이다.

메모리의 사용 없이 SF = 512인 코드워드를 생성하기 위해서는 최대 512의 레지스터가 필요하다. 하지만 위에서 설명한 바와 같이 단일 코드 인덱스를 이용한 OVSF 코드 생성기는 메모리를 사용하지 않고 쉬프트 레지스터의 개수를 최소화하면서 최대 SF 512까지의 코드워드를 생성할 수 있다. OVSF 코드 생성기에 사용된 총 로직은 10비트 쉬프트 레지스터, 16비트 쉬프트 레지스터 2개와 배타적 논리게이트 32개, 10비트 카운터 1개와 4비트 카운터 1개 등이다. 또한 SF 512보다 더 큰 확산 계수의 코드 생성을 위해서는 레지스터와 배타적 논리 게이트 등의 간단한 회로를 추가함으로써 쉽게 구현이 가능하다.

회로의 검증을 위하여 Xilinx ISE 5와 ModelSim XE2 v5.6e를 사용 하였다. 그림 3은 예로써

$C_{0000110110}$의 코드생성 검증 결과를 보여주고 있다. OVSF 코드 생성기의 출력이 $C_{0000110110}$의 코드와 일치함을 확인할 수 있다. 여기에서 Valid_out 신호는 코드의 길이를 의미한다. 즉, 확산 계수와 같다. Valid_out 신호가 '1'인 구간이 SF에 맞는 코드 시퀀스가 출력으로 나오게 된다. 지연(latency)은 확산 계수와 초기 로딩(loading) 지연에 의존한다. 하지만, 짧은 코드의 지연이 긴 코드보다 길기 때문에 최대 코드 생성 시간에는 영향을 주지 않는다.

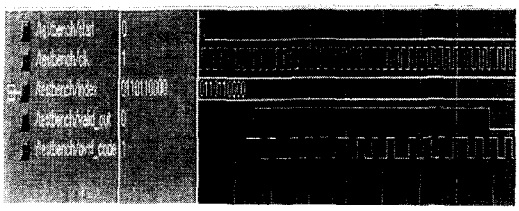


그림 3. 다단 확산 방식을 이용한 OVSF 코드 생성기의 검증 결과

V. 결 론

기존의 트리 구조 방식에서는 512의 OVSF 코드의 인덱스를 표현하기 위해 13 비트가 필요하고 268,800 비트의 LUT, 그리고 사용되고 있는 코드의 모코드와 자코드의 인덱스를 저장하기 위한 13,312 비트의 메모리가 필요하다. 반면, 단일 코드 인덱스 기법에서는 인덱스가 10 비트로 표현되어 파라미터 시그날링에서 25%의 절약을 가져오며 할당된 코드 인덱스를 저장하기 위해서 필요한 메모리의 사이즈가 5,120 비트이다. 여기서는 단일 코드 인덱스 방식의 장점 중에 하나인 다단 확산 방식을 이용하여 3GPP UMTS TDD와 FDD의 이중 모드에서 적용될 수 있는 빠르고 효율적인 OVSF 코드 생성기를 설계하고 검증하였다. 제안된 OVSF 생성기는 메모리가 추가로 필요하지 않고 42개의 쉬프트 레지스터와 32개의 배타적 논리 게이트, 10비트 카운터와 4비트 카운터 등으로 구현할 수 있다. 제안된 OVSF생성기는 기존의 방식에 비하여 회로가 간단하여 확장성이 좋고 메모리를 액세스(access)하지 않아 코드 생성 속도가 빠르며 power나 area 측면에서도 효율적이라고 할 수 있다. 검증 결과 짧은 확산 계수를 가지는 코드의 생성 시간이 긴 확산 계수를 가지는 코드보다 길지만 3GPP UMTS TDD 모드에서 인덱스 5비트는 선택적으로 무시할 수 있으므로 임의의 확산 계수에서도 비슷한 생성 속도를 보장한다. 512 보다 더 큰 확산 계수를 갖는 OVSF 코드의 생성기의 설계를 위해서는 레지스터와 배타적 논리 게이트를 몇 개 추가함으로써 추후 쉽게 확장할 수가 있다.

참 고 문 헌

[1] http://www.3gpp.org.

[2] K. S. Gilhousen, "System and method for orthogonal spread spectrum sequence generation in variable data rate systems," U.S. Patent Number 5,751,761, May 1998

[3] F. Adachi, M. Sawahashi and K. Okawa, "Tree-structured generation of orthogonal spreading codes with different length for forward link of DS-CDMA mobile radio," Electronics letters, vol. 33, no. 1, Jan. 1997

[4] E. H. Dinan and B. Jabbari, "Spreading codes for Direct sequence CDMA and wideband CDMA cellular networks," IEEE communication magazine, Sept. 1998.

[5] T. Minn and K. Y. Siu, "Dynamic Assignment of Orthogonal variable spreading factor codes in W-CDMA," IEEE Journal on selected areas in communications, vol. 18, no. 8, August 2000.

[6] Younglok Kim, Jung-Lin Pan, "OVSF code system and method", U.S Patent Number 6,552,996, April 2003.

[7] R. G. Cheng and P. Lin, "OVSF code channel assignment for IMT-2000," Vehicular Technology Conference, Spring 2000.

[8] Yu-Chee Tseng, Chih-Min Chao, "Code placement and replacement strategies for wideband CDMA OVSF code tree management," IEEE trans. on mobile computing, Vol. 1, No. 4, October-December 2002.

최 창 순 (Changsoon Choi) 준회원

2003년 2월 : 목원대학교 전자공학과 학사
2003년~현재 : 서강대학교 전자공학과 석사과정

<관심분야> VLSI 설계를 위한 DSP 알고리즘

김 태 훈 (Taehoon Kim) 준회원

2003년 8월 : 서강대학교 전자공학과 학사
2003년~현재 : 서강대학교 전자공학과 석사과정

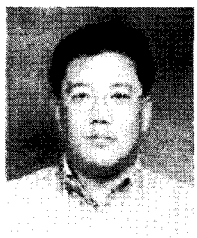
<관심분야> VLSI 설계를 위한 DSP 알고리즘 이동통신 신호 처리 및 시스템 설계

김 영 록 (Younglok Kim) 정회원

1991년 2월 : 서강대학교 전자공학과 학사
1991년~1998년 : Polytechnic Univ.(미국 뉴욕 폴리테크닉 대학) 공학석사, 공학박사
1998년~1999년 : AT&T연구소 연구원
1999년~2003년 : InterDigital Comm. Crop., (Melville,NY) 연구원
2003년~현재 : 서강대학교 전자공학과 조교수

<관심분야> VLSI 설계를 위한 DSP 알고리즘, 레이다 및 무선 통신 신호처리

정 화 용 (Hwayong Joung) 정회원



1991년 2월 : 성균관대학교 전기공학과 학사
1993년~1998년 : Polytechnic Univ.(미국 뉴욕 폴리테크닉 대학) 공학석사, 공학박사
1998년~1999년 : Polytechnic Univ. Computer Science Post-Doctorate Fellow
1999년~2001년 : 삼성전자 통신연구소 책임연구원
2001년~현재 : 텔슨전자 중앙연구소 책임연구원

<관심분야> 통신모뎀 알고리즘 및 하드웨어 디자인