논문 2012-49SD-2-5

IEEE 802.11a OFDM 타이밍 동기화기 블록의 저면적 설계 및 구현

(Low Area Design and Implementation for IEEE 802.11a OFDM Timing Synchronization Block)

석 상 철*, 장 영 범**

(Sang-Chul Seok and Young-Beom Jang)

요 약

이 논문에서는 IEEE 802.11a OFDM MODEM SoC용 타이밍 동기화 블록에 대한 저면적 구조를 제안한다. IEEE 802.11a의 타이밍 동기화 블록은 큰 구현 면적을 필요로 한다. 제안된 자기 상관 방식의 타이밍 동기화 블록 구조는 전치 직접형 필터 구조를 사용하여 곱셈 연산을 최소화하였다. 또한 CSD(Canonic Signed Digit) 계수를 이용하는 기술과 Common Sub-expression Sharing 기술을 적용하여 곱셈연산을 저면적으로 구현하였다. 제안된 타이밍 동기화 블록 구조에 대하여 Verilog-HDL 코딩과 0.13 micron 공정을 사용하여 합성한 결과, 기존 구조와 비교하여 22.7%의 구현 면적 감소 효과를 얻을 수 있었다.

Abstract

In this paper, a low area timing synchronization structure for the IEEE 802.11a OFDM MODEM SoC is proposed. The timing synchronization block of the IEEE 802.11a OFDM MODEM SoC requires large implementation area. In the proposed timing synchronization structure, it is shown that the number of multiplication can be reduced by using the transposed direct form filter. Furthermore, implementation area of the proposed structure can be more reduced using CSD(Canonic Signed Digit) and Common Sub-expression Sharing techniques. Through Verilog-HDL coding and synthesis, it is shown that the 22.7 % of implementation area can be reduced compared with the conventional one.

Keywords : OFDM, timing synchronization, frequency offset synchronization, Canonic signed Digit(CSD)

Ⅰ. 서 론

OFDM(Orthogonal Frequency Division Multiplexing) 변조 방식은 동기 획득에 취약한 단점을 갖고 있다. 즉 송신기와 수신기의 반송파 주파수 불일치로 인한 반송파 주파수 옵셋은 반송파간 간섭(Inter-Carrier Interference)을 발생시키고, OFDM 심볼의 동기를 정확히 획득하지 못하게 하여 심볼 간의 간섭을 일으켜 시스템의 성능을 저하시킨다.

IEEE 802.11a의 OFDM 수신기에서 신호를 정확히 복조하기 위해서는 신호의 동기가 매우 중요하다. 즉 송수신단의 샘플링 주파수 차이, 도플러 효과 등으로 주파수 옵셋이 발생하므로 이를 보정하기 위한 동기화기가 필요하다. 이와 같은 동기화기는 타이밍 동기화부, 주파수 옵셋 추정부, 주파수 옵셋 보상부로 구성된다.

고속 데이터 전송은 실내 무선 환경에서 clock 주기의 수십 배가 넘는 다중 경로 지연을 발생시키는데 심볼 타이밍 옵셋은 이러한 다중 경로 지연으로 인한 심볼 도착시간의 불확실성 등으로 발생한다. 따라서 주파

* 학생회원, ** 정회원-교신저자,
상명대학교 컴퓨터정보통신공학과
(Dept. of Information Technology, SangMyung University)
접수일자: 2011년9월5일, 수정완료일: 2012년2월15일

수 옵셋을 보상하기 위해서는 정확한 타이밍 동기가 먼저 획득되어야 한다.[1~4]

OFDM 시스템의 타이밍 동기화를 위하여 차분에 기초하는 차분방식이나 상관에 기초하는 상관방식이 제안되었다.[5~9] 상관 방식의 심볼 타이밍 동기화기는 동기 획득 성능은 우수하나 많은 수의 곱셈기가 사용되므로 저면적으로 구현이 어렵다는 단점이 있다. 따라서 이 논문에서는 전치 직접형 필터 구조를 이용하여 상관방식을 사용하는 타이밍 동기화기의 저면적 구조를 제안한다. 이 논문의 Ⅱ장에서는 프리앰블을 사용하는 OFDM 주파수 옵셋 동기화기의 동작 구조에 대해 살펴보고 Ⅲ장에서는 전치 직접형 필터 구조 변환, 곱셈연산 단순화 작업, CSD(Canonic Signed Digit)와 CSS(Common Sub-expression Sharing) 기술 등을 사용하여 저면적 타이밍 동기화기의 구조를 제안한다. Ⅳ장에서는 제안된 구조에 대한 시뮬레이션 및 합성결과를 통해 구현 면적 감소 효과를 입증하고 Ⅴ장에서 결론을 맺는다.

Ⅱ. IEEE 802.11a 주파수 옵셋 동기화기

2.1. 프리앰블의 구조

802.11a의 송신 단에서 만들어지는 패킷은 프리앰블(Preamble), 헤더(Header), 전송하고자하는 데이터인 페이로드(Payload)로 구성된다. 802.11a에서 사용되는 프리앰블의 구조는 그림 1과 같이 Short training symbol 10개와 Guard Interval 1개, Long training symbol 2개로 구성된다.[4] Short training symbol은 주로 Timing synchronization과 Coarse frequency offset estimation에 이용되고, Long training symbol은 Fine frequency offset estimation에 사용된다. Short training sequences는 s_1부터 s_{10}까지 10개의 같은 심볼로 구성되며 각각은 16개의 실수부와 16개의 허수부로 구성된다. Long training sequences는 l_1과 l_2의 두 개의 심볼로 구성되며 각각 64개의 실수부와 64개의 허수부로 구성된다.

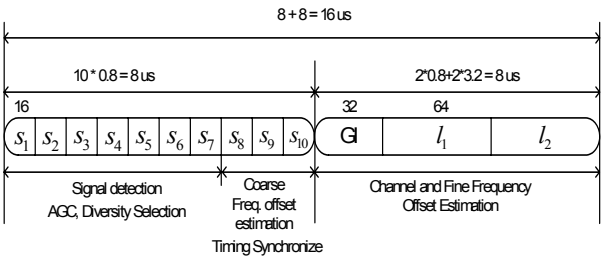

그림 1. 프리앰블의 구조
Fig. 1. Structure of Preamble.

프리앰블을 이용한 주파수 옵셋 동기화는 타이밍 동기화(timing synchronization)와 주파수 동기화(frequency synchronization)로 구성된다. 그림 1의 프리앰블에서 타이밍 동기화는 s_1부터 s_7의 7개의 심볼을 이용하여 심볼의 시작점을 정확히 찾는 작업이다.

2.2. 프리앰블을 이용한 주파수 옵셋 동기화기

802.11a OFDM의 프리앰블을 이용한 주파수 옵셋 동기화기의 블록도는 그림 2와 같다. 주파수 동기화는 주파수 옵셋 추정부와 주파수 옵셋 보상부로 구성된다.

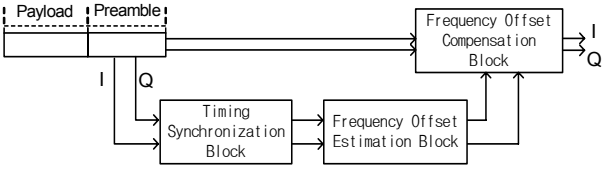

그림 2. OFDM 타이밍 동기화 및 주파수 동기화 블록도
Fig. 2. Block of Timing Synchronization and Frequency Offset Synchronization of OFDM.

이 논문에서는 그림 2의 타이밍 동기화부(timing synchronization block)를 저면적으로 구현하는 방법을 제안한다. 그림 2에서 Preamble 중 s_1부터 s_7까지의 Short training symbol이 타이밍 동기화기로 입력될 때에 타이밍 동기를 획득하여야 한다. 타이밍 동기를 찾기 위하여 타이밍 동기화부는 다음 식을 연속으로 계산하고 있다.

$$R = \left| \sum_{n=0}^{15} Y_n X_n^* \right| \tag{1}$$

이 식에서 R은 상관 값이며 X_n은 수신기가 미리 알고 있는 Short training symbol의 n번째 값이고 $*$은 conjugate를 나타내고 Y_n은 수신된 신호의 n번째 값이다. 식 (1)에서 보듯이 상호상관 값 R은 수신 받은 신호와 Short training symbol의 16개의 샘플을 복소 곱셈하여 얻은 값이다. 따라서 식 (1)을 수신 신호에 대하여 연속으로 수행하면 16개의 반복적인 실수와 허수 샘플들이 끝나는 시점에서 상관 값이 크게 증가하므로 심볼 타이밍 동기를 획득할 수 있다. 식 (1)에서 미리 알고 있는 Short training symbol과 입력신호와의 연산은 다음 식으로 표현할 수 있다.

$$\begin{aligned} R &= |YX^*| \\ &= |\sum (y_r + jy_i)(x_r - jx_i)| \\ &= |\sum (y_r x_r - jy_r x_i + jy_i x_r + y_i x_i)| \\ &= |\sum [(y_r x_r + y_i x_i) + j(y_i x_r - y_r x_i)]| \\ &= |r(n) + ji(n)| = \sqrt{r^2(n) + i^2(n)} \end{aligned} \quad (2)$$

위의 식에서 복소 신호 X는 수신기에서 미리 알고 있는 Short training symbol이며 x_r은 실수 값, x_i는 허수 값을 나타낸다. 복소 신호 Y는 수신된 신호이며 y_r은 실수 값, y_i은 허수 값을 나타낸다. 이 식을 연산하면 상관 값이 구해지며 피크가 발생한다. 이 피크를 이용하여 타이밍 동기를 얻을 수 있다.

Ⅲ. 제안된 타이밍 동기화기 구조

3.1. 기본 구조

식 (2)를 구현하는 타이밍 동기화기의 기본 구조는 그림 3과 같다. 그림 3의 타이밍 동기화기 구조에서 y_r은 수신신호의 실수, y_i는 수신신호의 허수를 나타낸다. $x_r(0)$은 알고 있는 훈련 심볼의 0번째 실수샘플 값이고 15번까지 총 16개가 있고, $x_i(0)$는 훈련 심볼의 0번째 허수샘플 값이며 15번까지 총 16개가 있다.

프리앰블 입력 값은 복소수로 각각의 곱셈기 파트별로 실수인 y_r, 허수인 y_i가 입력된다. 입력 신호들은 쉬프트 레지스터인 매 clock마다 D0로 입력되며 그 옆의 쉬프트 레지스터로 이동된다. 쉬프트 레지스터는 D0부터 D14까지 총 15개가 있다. 각 쉬프트 레지스터에 입력된 값들은 매 clock마다 각각의 곱셈기 파트에서 x_r과 x_i를 이용해 곱셈연산을 수행한다. 그림 3의 실수 곱셈 연산 블록과 허수 곱셈 연산 블록의 연산은 각각 다음과 같다.

$$\begin{aligned} r_1(n) &= \sum y_r x_r \quad r_2(n) = \sum y_i x_i \quad r(n) = r_1(n) + r_2(n) \\ i_1(n) &= \sum y_i x_r \quad i_2(n) = \sum y_r x_i \quad i(n) = i_1(n) - i_2(n) \end{aligned} \quad (3)$$

그림 3에서 보듯이 심볼 타이밍 동기화기 기본 구조는 64개의 곱셈기를 사용하고 있으므로 그대로 구현하면 면적이 커지게 된다. 따라서 이 논문에서는 기본 구조의 곱셈기 파트를 효율적으로 설계하여 저면적으로 구현할 수 있는 구조를 제안한다. 그림 3의 기본 구조에 대한 저면적 구조는 전치직접형 구조설계 단계, 단순화 작업 단계, CSD 계수 설계 단계, CSS 구조 설계 단계의 4 단계로 이루어진다.

3.2. 전치 직접형 구조 변환

곱셈기 블록은 $r_1(n)$과 $r_2(n)$의 실수 블록 2개와 $i_1(n)$과 $i_2(n)$의 허수 블록 2개로 구성이 되며 하나의 곱셈기 블록은 16개의 곱셈기와 15개의 쉬프트 레지스터로 구

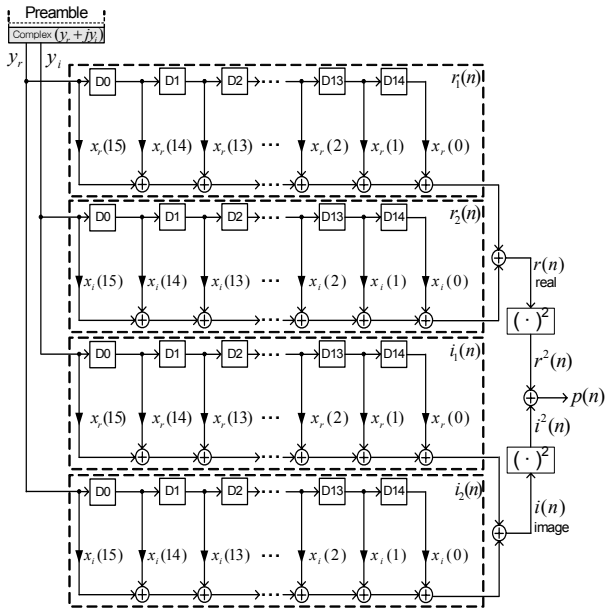

그림 3. 타이밍 동기화기 기본 구조
Fig. 3. Basic Timing Synchronization Structure.

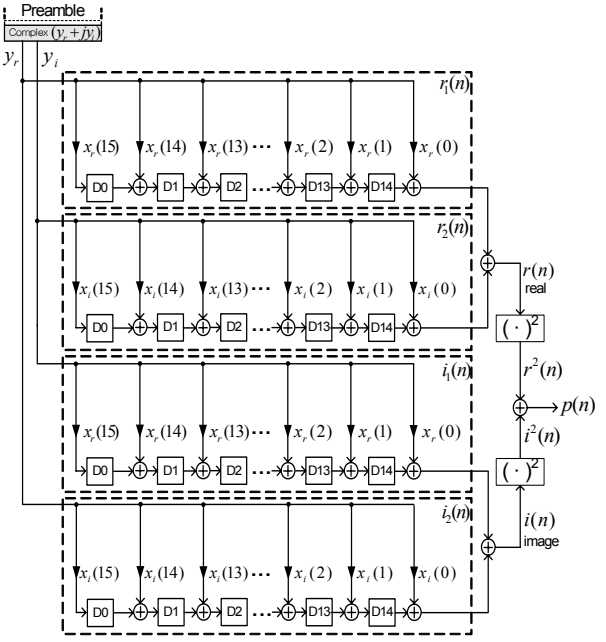

그림 4. 타이밍 동기화기 기본구조의 전치 직접형 블록도
Fig. 4. Transposed Direct Form of the Basic Timing Synchronization Structure.

성된다. 따라서 총 곱셈기는 64개가 필요하며 기본 구조를 그대로 구현하면 면적이 커지게 된다.

이와 같은 큰 구현 면적을 줄이기 위하여 CSS 방식을 사용하여 덧셈기만을 사용하여 구현하는 구조를 제안한다. CSS 방식을 쓰기 위해서는 그림 3의 직접형(Direct form) 필터 구조를 그림 4와 같이 전치 직접형(Transposed direct form) 필터 구조로 바꿔야 한다.

3.3. Simplified 전치 직접형 구조 변환

그림 4의 구조에서 4개의 곱셈기 블록이 있는데, 이 중에 $r_1(n)$과 $i_1(n)$은 필터 계수가 같으므로 같은 구조의 필터이다. 또한 $r_2(n)$과 $i_2(n)$도 같은 필터이다. 따라서 $r_1(n)$과 $r_2(n)$의 필터 계수만 살펴보면 표 1과 같다.

표 1의 $r_1(n)$ 필터의 필터 계수를 보면 중복되어 사용되는 계수들이 있다. 예를 들면 $x_r(0)$과 $x_r(8)$의 값은

표 1. Short traning symbol($r_1(n)$)의 값
Table 1. Value of Short Training Symbol($r_1(n)$).

$r_1(n)$ filter coefficients		$r_2(n)$ filter coefficients	
$x_r(0)$	0.0460	$x_i(0)$	0.0460
$x_r(1)$	-0.1324	$x_i(1)$	0.0023
$x_r(2)$	-0.0135	$x_i(2)$	-0.0785
$x_r(3)$	0.1428	$x_i(3)$	-0.0127
$x_r(4)$	0.0920	$x_i(4)$	0
$x_r(5)$	0.1428	$x_i(5)$	-0.0127
$x_r(6)$	-0.0135	$x_i(6)$	-0.0785
$x_r(7)$	-0.1324	$x_i(7)$	0.0023
$x_r(8)$	0.0460	$x_i(8)$	0.0460
$x_r(9)$	0.0023	$x_i(9)$	-0.1324
$x_r(10)$	-0.0785	$x_i(10)$	-0.0135
$x_r(11)$	-0.0127	$x_i(11)$	0.1428
$x_r(12)$	0	$x_i(12)$	0.0920
$x_r(13)$	-0.0127	$x_i(13)$	0.1428
$x_r(14)$	-0.0785	$x_i(14)$	-0.0135
$x_r(15)$	0.0023	$x_i(15)$	-0.1324

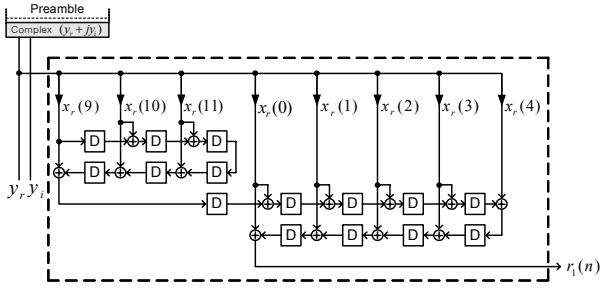

그림 5. 타이밍 동기화기 구조의 전치 직접형 블록도
Fig. 5. Transposed Direct Form Structure of Timing Synchronization.

같다. 이와 같이 $r_1(n)$ 필터에서 중복된 계수들을 하나의 곱셈기로 구현하면 그림 5와 같다.

그림 4의 $r_1(n)$ 필터는 16개의 곱셈기로 구성되었으나 그림 5의 $r_1(n)$ 필터는 8개의 곱셈기로 감소되었음을 알 수 있다. 그 외의 $r_2(n)$, $i_1(n)$, $i_2(n)$ 필터도 모두 중복된 계수를 갖고 있으므로 각각 8개의 곱셈기로 구현할 수 있다. 따라서 32개의 곱셈기로 구현될 수 있다.

3.4. CSD형 계수 필터 구조

이 절에서는 $r_1(n)$, $r_2(n)$, $i_1(n)$, $i_2(n)$의 4개의 필터에 대하여 덧셈기를 사용하여 구현한다. 4개의 필터 중에서 $r_1(n)$ 필터에 대하여 구현하며 나머지 다른 3개의 필터는 같은 방식을 적용하면 된다. 고정된 계수를 덧셈기를 사용하여 효과적으로 구현하려면 CSD형의 필터 계수를 사용하여야 한다.[7~8] 따라서 $r_1(n)$ 필터의 8개의 계수를 CSD형으로 나타내면 표 2와 같다.

표 2의 맨 오른쪽 열에는 각 필터 계수를 곱하는 데 필요한 덧셈기의 수를 나타내었다. 즉 필터 계수를 구현하는 데에 총 26개의 덧셈기가 필요하며, 지연소자 체인에 14개의 덧셈기가 필요하므로 $r_1(n)$ 필터의 구현에 총 40개의 덧셈기가 사용된다.

표 2. $r_1(n)$ 필터 계수의 CSD형
Table 2. CSD form coefficients of the $r_1(n)$ filter.

	coefficient	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	# of adders
$x_r(0)$	0.0460	0	0	0	0	1	0	N	0	0	0	N	0	0	1	0	N	4
$x_r(1)$	-0.1324	0	0	0	N	0	0	0	N	0	0	0	1	0	0	N	0	3
$x_r(2)$	-0.0135	0	0	0	0	0	0	N	0	0	1	0	0	1	0	N	0	3
$x_r(3)$	0.1428	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	N	4
$x_r(4)$	0.0920	0	0	0	1	0	N	0	0	0	N	0	0	1	0	0	N	4
$x_r(9)$	0.0023	0	0	0	0	0	0	0	0	0	1	0	1	0	N	0	N	3
$x_r(10)$	-0.0785	0	0	0	0	N	0	N	0	0	0	0	N	0	1	0	0	3
$x_r(11)$	-0.0127	0	0	0	0	0	0	N	0	1	0	N	0	0	0	0	0	2

3.5. CSS 방식 필터 구조

$r_1(n)$ 필터의 구현에 사용되는 덧셈기의 수를 더욱 감소시키기 위하여 CSS(Common Sub-expression Sharing) 방식을 적용한다.[10~11] CSS 방식을 적용하려면 공통 패턴을 먼저 정의하여야 하므로 표 2의 CSD형에서 공통 패턴을 2중 실선으로 표기하면 표 3과 같다.

표 3에서 보듯이 먼저 공통패턴들을 2중 실선으로 표시하였다. 즉 10N, 101, 100N, 1001의 4개의 공통패턴이 있음을 알 수 있다. 여기서 10N, 101, 100N은 - 만 붙

표 3. $r_1(n)$ 필터 계수의 공통 패턴
Table 3. Common sub-expression of the $r_1(n)$ filter.

	coefficient	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	# of adders
$x_r(0)$	0.0460	0	0	0	0	1	0	N	0	0	0	N	0	0	1	0	N	2
$x_r(1)$	-0.1324	0	0	0	N	0	0	0	N	0	0	0	1	0	0	N	0	2
$x_r(2)$	-0.0135	0	0	0	0	0	0	N	0	0	1	0	0	1	0	N	0	1
$x_r(3)$	0.1428	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	N	2
$x_r(4)$	0.0920	0	0	0	1	0	N	0	0	0	N	0	0	1	0	0	N	2
$x_r(9)$	0.0023	0	0	0	0	0	0	0	0	0	1	0	1	0	N	0	N	1
$x_r(10)$	-0.0785	0	0	0	0	N	0	N	0	0	0	0	N	0	1	0	0	1
$x_r(11)$	-0.0127	0	0	0	0	0	0	N	0	1	0	N	0	0	0	0	0	1

이면 N01, N0N, N001과 같은 패턴이므로 하나의 같은 패턴으로 본다. 이 공통패턴은 a_0의 기준을 사용하여 다음 식과 같이 표기할 수 있다.

$$\begin{aligned} a_1 &= a_0 - a_0 \gg 2 \\ a_2 &= a_0 + a_0 \gg 2 \\ a_3 &= a_0 - a_0 \gg 3 \\ a_4 &= a_0 + a_0 \gg 3 \end{aligned} \tag{4}$$

이와 같이 공통 패턴을 만드는 데에 4개의 덧셈기가 사용된다. 위의 공통패턴을 이용하여 $r_1(n)$ 필터의 각각의 필터 계수를 표현하면 식 (5)와 같다.

$$\begin{aligned} x_r(0) &= a_1 \gg 4 - a_3 \gg 10 - a_0 \gg 15 \\ x_r(1) &= -a_0 \gg 3 - a_0 \gg 7 + a_3 \gg 11 \\ x_r(2) &= -a_3 \gg 6 + a_1 \gg 12 \\ x_r(3) &= a_4 \gg 3 + a_4 \gg 9 - a_0 \gg 15 \\ x_r(4) &= a_1 \gg 3 - a_3 \gg 9 - a_0 \gg 15 \\ x_r(9) &= a_2 \gg 9 - a_2 \gg 13 \\ x_r(10) &= -a_2 \gg 4 - a_1 \gg 11 = -(a_2 \gg 4 + a_1 \gg 11) \\ x_r(11) &= -a_1 \gg 6 - a_0 \gg 10 = -(a_1 \gg 6 + a_0 \gg 10) \end{aligned} \tag{5}$$

표 3의 맨 오른쪽 열에 그 필터 계수를 구현하는데 필요한 덧셈기의 수를 표기하였으며 12개가 필요하다. 따라서 공통 패턴을 만드는 데에 필요한 덧셈기 수와 합하면 총 16개의 덧셈기가 필요하다. 따라서 공통 패턴을 사용하여 $r_1(n)$ 필터를 구현하면 필터 계수용 덧셈기를 26개에서 16개로 줄일 수 있으며 그림 6과 같다.

그림 6에서 보듯이 식 (4)에서 정의한 공통패턴을 왼쪽 상단부분에 구현하였다. 수신신호로부터 a_1, a_2, a_3, a_4 의 공통 패턴을 만들어서 초기 입력 값과 함께 쉬프트와 덧셈, 뺄셈연산을 통해 각각의 8개의 연산 값들을 계산한다. 각각 쉬프트 레지스터에 저장된 값들은 다음 clock에서 계산되는 연산결과와 덧셈연산이 이루어져 최종적으로 상관 값들의 합이 출력으로 나올 수

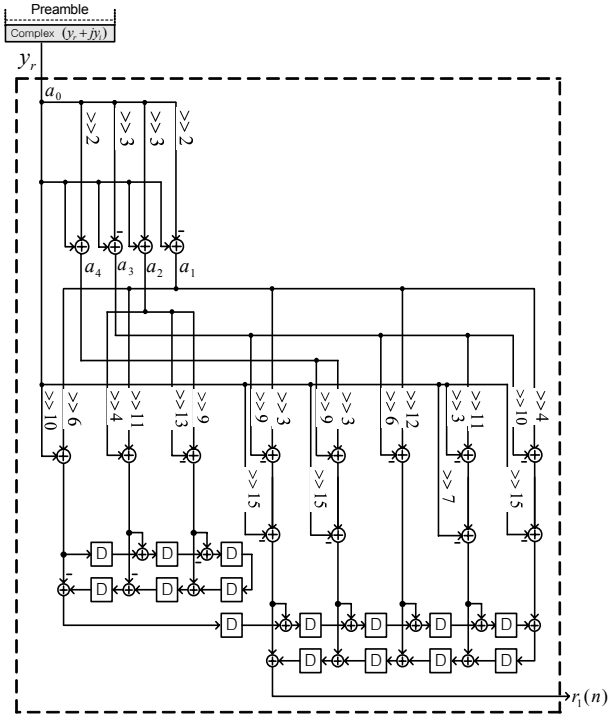

그림 6. 제안된 $r_1(n)$ 필터 구조
Fig. 6. Proposed $r_1(n)$ filter structure.

표 4. 제안된 타이밍 동기화기 구조의 덧셈기 수
Table 4. Number of Adders in the Proposed Timing Synchronization Structure.

$r_1(n)$	$r_2(n)$	$i_1(n)$	$i_2(n)$	기타	Total
30	30	30	30	3	123

있게 설계하였다. CSS 방식을 사용하여 필터 계수를 구현하는 데에 총 16개의 덧셈기가 필요하며, 지연소자 체인에 14개의 덧셈기가 필요하므로 $r_1(n)$ 필터의 구현에 총 30개의 덧셈기가 필요하다. 지금까지 기술한 바와 같이 기본 구조의 $r_1(n)$ 필터에 대하여 5 단계를 통해 효율적인 구조를 제안하였다. 나머지 $r_2(n)$, $i_1(n)$, $i_2(n)$의 필터도 같은 단계를 사용하여 구조를 설계할 수 있다.

따라서 전체 심볼 타이밍 동기화기를 제안된 방식으로 구현하면 덧셈기의 수는 표 4와 같다.

Ⅳ. 구 현

4.1. Matlab Function Simulation

이 절에서는 Matlab을 사용하여 제안된 타이밍 동기화기 구조의 function을 검증한다. Simulation에 사용된 테스트 벡터는 그림 7과 같다.

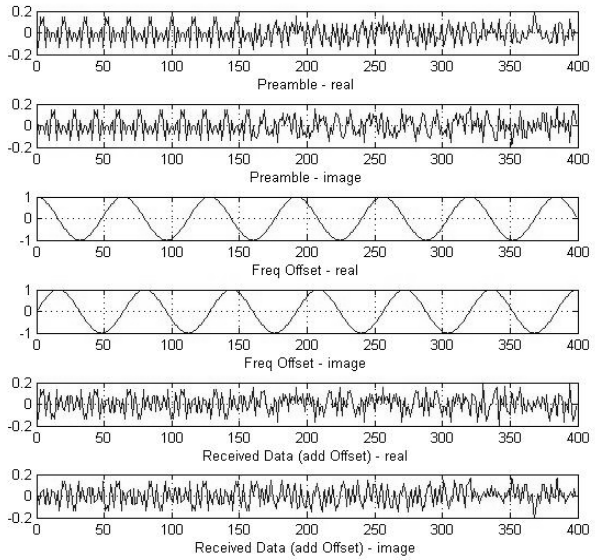

그림 7. 실험에 사용된 수신 프리앰블 신호
Fig. 7. Signal of Preamble.

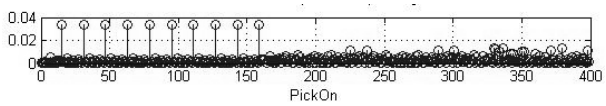

그림 8. 찾아낸 10개의 피크
Fig. 8. Cross-Correlation Peak.

표 5. 수신신호와의 상관 값
Table 5. Value of Cross-Correlation and Received Data.

No.	$r^2(n)$	$i^2(n)$	상관값
1	0.000036	0.000038	0.000074
2	0.000010	0.000420	0.000430
3	0.000731	0.000064	0.000795
4	0.000003	0.001574	0.001577
5	0.000496	0.000277	0.000772
6	0.000542	0.000129	0.000671
7	0.000003	0.000325	0.000328
8	0.000746	0.004460	0.005206
9	0.000209	0.000583	0.000792
10	0.000003	0.000391	0.000393
11	0.000038	0.000380	0.000418
12	0.002666	0.000091	0.002757
13	0.000015	0.000253	0.000268
14	0.000600	0.001771	0.002371
15	0.000103	0.001150	0.001253
16	0.017381	0.016283	0.033663

그림 7의 첫 번째와 두 번째 신호는 각각 Short training symbol 160개와 GI 32개, Long training symbol 128개, Header 80개의 총 400개 샘플로 구성된 프리앰블의 real과 imaginary 값이다.

이 신호에 각각 세 번째와 네 번째와 같은 옵셋을 인가하여 다섯 번째와 여섯 번째 신호와 같은 옵셋이 인가된 테스트 벡터를 생성하였다. 이 테스트 벡터 신호를 제안된 타이밍 옵셋 구조에 입력시켜 얻은 출력 신호는 그림 8과 같다.

그림 8에서 보듯이 제안 구조는 피크 값을 정확히 찾아내고 있음을 볼 수 있다.

그림 8의 상관 값 중에서 앞에서부터 16개의 값은 표 5와 같다. 표 5에서 보듯이 16번째 상관 값이 피크가 됨을 알 수 있다. 이는 수신신호와 수신기에서 알고 있는 Short training symbol의 타이밍이 일치할 때 상관 값이 가장 커진다는 것을 나타낸다. 이와 같은 피크 값을 7개 감지하면 타이밍을 찾은 것으로 간주하여 주파수 옵셋 추정부가 동작하기 시작한다.

4.2. Verilog-HDL RTL Coding

이 절에서는 제안 구조에 대하여 Verilog-HDL 코딩을 수행하였다. 이 절에서 사용한 입력 테스트 벡터는 그림 7의 다섯 번째와 여섯 번째 신호이고 출력 테스트 벡터는 그림 8의 신호이다. 즉 Verilog-HDL 코딩의 출력 값을 출력 테스트 벡터와 비교하였다. Verilog-HDL 코딩 후의 ModelSim 결과는 표 6과 같다.

표 6에서는 Verilog-HDL ModelSim 결과를 16개만 나

표 6. Matlab 값과 ModelSim 값의 비교
Table 6. Comparison Between Matlab and ModelSim Value.

	Matlab Function Simulation		ModelSim
No.	상관 값	2's Complement	2's Complement
1	0.000074	0000000000000010	0000000000000010
2	0.000429	0000000000001110	0000000000001101
3	0.000795	0000000000011010	0000000000011001
4	0.001577	0000000000110100	0000000000110011
5	0.000772	0000000000011001	0000000000011001
6	0.000670	0000000000010110	0000000000010101
7	0.000327	0000000000001011	0000000000001010
8	0.005205	0000000010101011	0000000010101010
9	0.000791	0000000000011010	0000000000011001
10	0.000393	0000000000001101	0000000000001100
11	0.000418	0000000000001110	0000000000001101
12	0.002757	0000000001011010	0000000001011010
13	0.000267	0000000000001001	0000000000001000
14	0.002371	0000000001001110	0000000001001101
15	0.001253	0000000000101001	0000000000101000
16	0.033663	0000010001001111	0000010001001011

표 7. 기존 구조와 제안 구조의 면적비교
Table 7. Comparison of Area for Proposed Structure.

	Combinational area	Noncombinational area	Net Interconnect area	Total area
Conventional structure	0.037926	0.019068	0.000344	0.057338 (100%)
Proposed structure	0.028053	0.015913	0.000316	0.044282 (77.23%)

타내었다. 표 7에서 보듯이 Verilog-HDL ModelSim 결과가 Function simulation 결과와 일치함을 볼 수 있다.

4.3. 합성

이 절에서는 제안 구조의 구현 면적을 simulation해 보기로 한다. 즉 완성된 RTL 코드는 Synopsys Design Compiler 툴을 사용하여 합성한 후 면적을 알아보았다. 제안된 구조의 면적 비교를 위하여 [12]에서 사용된 구조의 면적과 비교한다. 기존 구조와 제안 구조 모두 합성을 위해 삼성 0.13-Micron 1.2V 공정을 사용하였다. 제안 구조의 합성 후 Schematic view는 그림 9와 같다.

제안 구조의 면적을 계산하기 위하여 Synopsys Design Compiler 합성 툴을 사용한 결과는 표 7과 같다.

표 7에서 보듯이 기존구조의 면적은 0.057338 mm^2로 예상되었으며 제안구조는 0.044282 mm^2로 계산되었다. 즉 22.77%의 구현 면적 감소를 예상할 수 있다. 또한 타이밍 옵셋 동기 블록의 입력과 출력을 같도록 하드웨어를 구현한 것이기 때문에 기존 구조와 비교하여 동기 성능도 정확하게 같게 된다.

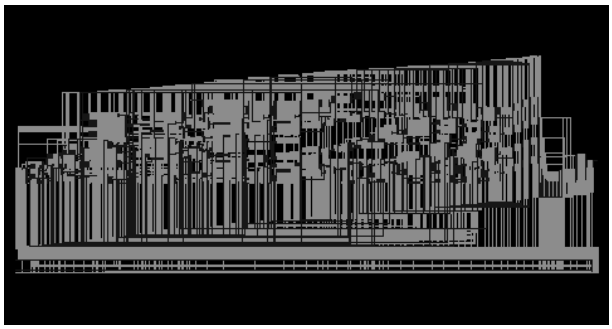

그림 9. 제안 구조의 Schematic view
Fig. 9. Schematic View of the Proposed Structure.

V. 결 론

이 논문에서는 IEEE 802.11a OFDM 통신 방식의 주파수 옵셋 동기화기에서 타이밍 동기화기 블록에 대한 저면적 구조를 제안하였다. 제안 구조에서는 전치 직접형 구조를 사용하여 곱셈연산의 수를 최소화하였으며 CSD 방식과 CSS 방식을 사용하여 저면적으로 곱셈 연산을 구현하였다. Function simulation을 통하여 제안 구조의 동작을 검증 하였으며, Verilog-HDL 코딩과 합성을 통하여 22.77%의 면적 감소 효과를 확인하였다.

참 고 문 헌

[1] C. L. Wang and H. C. Wang, “Optimized Joint Fine Timing Synchronization and Channel Estimation for MIMO Systems”, IEEE Trans. on Commun., vol. 59, no. 4, pp. 1089-1098, April, 2011.

[2] Y. Yuzhe, D. Xiaodai and N. Tin “Design and Analysis of Timing Synchronization in Block Transmission UWB Systems”, IEEE Trans. on Commun., vol. 59, no. 6, pp. 1686-1696, Jun. 2011.

[3] T. Liu and X. Zhou, “Joint blind estimation of symbol timing offset and carrier frequency offset for OFDM systems with I/Q imbalance”, IEICE Electronics Express, vol. 6, no. 8, pp. 443-448, Aug. 2009.

[4] IEEE 802.11, “Wireless MAC and PHY Specifications: High Speed Physical Layer in the 5 GHz Band”, P802.11a.D7.0, July, 1999.

[5] Zhongshan Zhang, Keping Long, Ming Zhao, Yuannan Liu. “Joint Frame Synchronization and Frequency Offset Estimation in OFDM Systems”. IEEE Trans. VOL. 51, No. 3, Sep. 2005.

[6] A. R. S. Bahai, B. R. Saltzberg, and M. Ergen, “Multi-Carrier Digital Communications: Theory and Applications of OFDM”, Springer, New York, 2004.

[7] J. J. van de Beek, P. O. Borjesson, M. L. Boucheret, D. Landstrom, J. M. Arenas, P. Odling, C. Ostberg, M. Wahlqvist, and S. K. Wilson, “A time and frequency synchronization scheme for multiuser OFDM,” IEEE J. Select. Areas Commun., vol. 17, pp. 1900-1914, Nov. 1999.

[8] T. M. Schmidl and D. C. Cox, “Robust frequency and timing synchronization in OFDM”, IEEE Trans. on Commun., vol. 45, pp. 1613-1621, Dec. 1997.

[9] Schmidl Timothy M, Cox Donald C. “Robust Frequency and Timing Synchronization for OFDM”. IEEE Trans. Commun, 45(12), pp. 1613-1621, 1997.

[10] R. I. Hartley, “Subexpression sharing in filters using canonic signed digit multipliers”, IEEE Trans. Circuits and Systems-II: Analog and Digital Signal Processing, vol. 43, No.10, pp. 677-688, Oct. 1996.

[11] Y. Jang, and S. Yang, “Low-power CSD linear phase FIR filter structure using vertical common

sub-expression" IEE Electronics Letters, vol. 38, No. 15, pp. 777-779, Jul. 2002.
[12] 하준형, 장영범, "WLAN용 저면적 심볼 타이밍 옵셋 동기화기 구조" 한국산학기술학회논문지, 제12권 제3호, pp. 1387-1394, 2011년 3월.

저 자 소 개

석 상 철(학생회원)
2010년 상명대학교 정보통신공학과 학사 졸업.
2012년~현재 상명대학교 대학원 컴퓨터 정보통신공학과 석사과정.
<주관심분야 : 반도체, 통신, 신호처리>

장 영 범(정회원)-교신저자
1981년 연세대학교 전기공학과 학사 졸업.
1990년 Polytechnic University 대학원 석사 졸업.
1994년 Polytechnic University 대학원 박사 졸업.
1981년~1999년 삼성전자 System LSI 사업부 수석연구원.
2000년~2002년 이화여자대학교 정보통신학과 연구교수.
2002년~현재 상명대학교 정보통신공학과 교수.
<주관심분야 : 통신신호처리, 비디오신호처리, SoC 설계>