한국산학기술학회논문지
Vol. 12, No. 3, pp. 1387-1394, 2011

WLAN용 저면적 심볼 타이밍 옵셋 동기화기 구조

하준형[1], 장영범[1*]
[1]상명대학교 정보통신공학과

Low-Area Symbol Timing Offset Synchronization Structure for WLAN Modem

Jun-Hyung Ha[1] and Young-Beom Jang[1*]
[1]**Dept. of Information & Telecommunication Engineering, Sangmyung University**

요 약 이 논문에서는 OFDM Modem의 심볼 타이밍 옵셋 동기화 블록에 대한 저면적 구조를 제안한다. 심볼 타이밍 동기화 블록에서의 곱셈연산을 디지털 필터 구조의 개념을 도입하여 저면적 구조를 유도하였다. 즉 곱셈연산을 CSD(Canonic Signed Digit) 방식과 CSS(Common Sub-expression Sharing) 방식의 덧셈기를 사용한 구조를 제안하였다. 제안 구조에 대한 Verilog-HDL 코딩과 합성을 통하여 $0.264mm^2$로 구현하였으며, 이는 기존 구조의 $0.723mm^2$와 비교하여 63.54%의 구현 면적 감소를 달성하였다. 따라서 제안된 구조는 OFDM 시스템의 심볼 타이밍 동기화기에 효율적으로 사용 될 수 있을 것이다.

Abstract In this paper, a low-area symbol timing offset synchronization structure for WLAN Modem is proposed. Using CSD(Canonic Signed Digit) coefficients and CSS(Common Sub-expression Sharing) technique for the filter implementation, efficient structure for multiplication block can be obtained. Function simulation for proposed structure is done by using the preamble with timing offset. Through Verilog-HDL coding and synthesis, it is shown that the proposed symbol timing offset synchronization structure can be implemented with low-area semiconductor.

Key Words : OFDM, WLAN, Symbol Timing, CSD, CSS

1. 서론

OFDM(Orthogonal Frequency Division Multiplexing) 변조 방식이 다양한 유무선 데이터 전송 시스템에서 널리 사용되고 있다. OFDM 시스템은 기존의 단일 캐리어 전송방식에 비해서 주파수 이용 효율이 높고, 직교 관계를 갖는 다수의 부 반송파(Subcarriers)를 이용하여 심볼간의 간섭(Inter Symbol Interference) 문제를 해소하였으며 보호구간(Guard Interval)을 사용하여 멀티패스페이딩 환경에서도 우수한 성능을 얻을 수 있다.

OFDM 변조 방식은 동기 알고리듬에 취약한 단점을 지니고 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 심볼간의 간섭을 일으켜 시스템의 성능을 저하시킨다.

따라서 OFDM 방식의 신호를 정확히 복조하기 위해서는 신호의 심볼 타이밍 동기가 매우 중요하다. 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 불확실성과 송, 수신단 샘플링 주파수 차이 등으로 발생하

*교신저자 : 장영범(ybjang@smu.ac.kr)
접수일 10년 12월 30일 수정일 11년 03월 02일 게재확정일 11년 03월 10일

며 부반송파의 신호의 위상을 회전시키고 FFT(Fast Fourier Transform) 윈도우 위치의 동기 오류를 일으켜 인접 심볼 간 간섭의 원인이 된다.

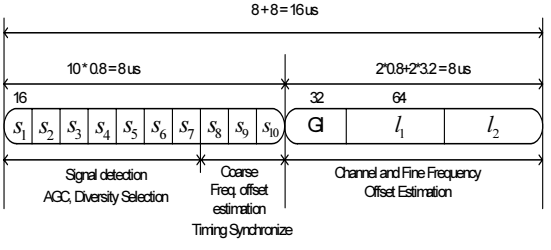
이와 같이 심볼 동기 오차는 시스템의 성능을 크게 저하시키는 이유가 되므로 OFDM 시스템의 수신 단에서는 주파수 옵셋 동기화 블록을 수행하기 이전에 송수신단의 심볼 타이밍 옵셋 동기가 선행되어야만 한다. OFDM 시스템에서 사용되는 심볼 타이밍 동기화에는 차분에 기초하는 차분방식이나 상관에 기초하는 상관방식이 제안되었다.[1-5] 상관 방식의 심볼 타이밍 동기화는 많은 수의 곱셈기가 사용되므로 저면적으로 구현이 어렵다. 따라서 이 논문에서는 상관방식을 사용하는 심볼 타이밍 동기화기 블록에 대하여 CSD(Canonic Signed Digit form)형 계수와 CSS (Common Sub-expression Sharing) 방식을 사용하여 곱셈연산을 수행하는 저면적 구조를 제안한다.

2장에서는 IEEE 802.11a 심볼 타이밍 옵셋 동기화기에 대해 살펴보고 3장에서는 제안된 CSS 방식을 사용한 심볼 타이밍 옵셋 동기화기의 구조 및 설계에 대해서 기술한다. 4장에서는 제안된 구조에 대한 시뮬레이션 및 합성 결과를 통해 성능을 입증하고 5장에서는 결론을 맺는다.

2. 심볼 타이밍 옵셋 동기화기

2.1 프리앰블의 구조

WLAN(IEEE 802.11a) OFDM 무선통신 방식의 송신단에서 만들어지는 패킷은 프리앰블(Preamble), 헤더(Header), 그리고 실제 전송하고자하는 데이터인 페이로드(Payload)로 구성된다. IEEE 802.11a에서 사용되는 프리앰블의 구조는 그림 1과 같이 Short training symbol 10개와 Long training symbol 2개로 구성된다.[5]

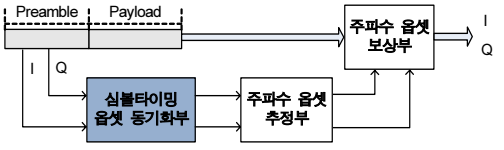


[그림 1] 프리앰블의 구조

Short training symbol은 신호 검출, AGC, Diversity selection, Timing synchronization, Coarse frequency offset estimation을 하는데 이용된다. Long training symbol은 Channel estimation과 Fine frequency offset estimation을 하는데 사용된다. Short training sequences는 s_1부터 s_{10}까지 10개의 같은 심볼로 구성되며 각각의 심볼은 16개의 실수부와 16개의 허수부로 구성되어있다. Long training sequences는 l_1과 l_2의 두 개의 심볼로 구성되며 각각의 심볼은 64개의 실수부와 64개의 허수부로 구성되어 있다.

2.2 프리앰블을 이용한 옵셋 동기화기

프리앰블을 이용한 심볼 타이밍 옵셋 동기화는 주기적인 특성을 갖는 훈련 심볼을 이용하여 타이밍을 동기화 시켜 심볼의 시작점을 정확히 찾는 작업이다. 이 시작점을 주파수 옵셋 동기화기 블록에 정확히 알려주어 FFT/IFFT 블록에 입력되는 심볼의 타이밍을 일치시킨다[6-8]. 프리앰블을 이용한 WLAN용 심볼 타이밍 옵셋 동기화기와 주파수 옵셋 동기화기의 블록도는 그림 2와 같다.



[그림 2] WLAN용 심볼 타이밍 옵셋과 주파수 옵셋 동기화기 블록도

그림 2에서 보듯이 먼저 Preamble 중 s_1부터 s_7까지의 Short training symbol의 I와 Q가 심볼 타이밍 옵셋 동기화기로 입력되어 타이밍 동기를 획득하여야 한다. 타이밍 옵셋을 찾기 위하여 다음 식을 계산한다.

$$R_n = \left| \sum_{n=0}^{15} Y_n X_{15-n}^{*} \right| \quad (1)$$

이 식에서 R_n은 상호상관 방법으로 계산된 상관 값이며 X_{15-n}은 수신기가 미리 알고 있는 Short training 심볼의 15-n번째 값이고 *은 conjugate를 의미하고 Y_n은 수신된 신호이다. 수신신호의 상호상관 값 R_n은 수신 받은 신호와 Short training 심볼의 16개의 샘플을 복소 곱셈하여 얻은 값들이다. 따라서 식 1을 수신 신호에 대하여 연속으로 수행하면 16개의 반복적인 실수와 허수 샘

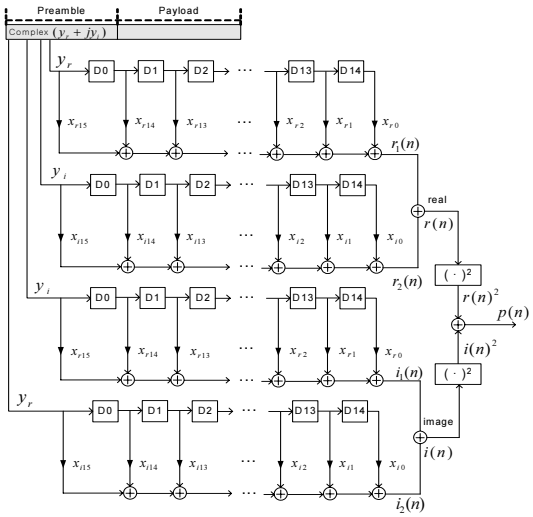
플들이 끝나는 시점에서 상관 값이 크게 증가하여 피크를 나타내므로 심볼 타이밍 동기를 정확히 얻을 수 있다. 식 1에서 미리 알고 있는 Short training symbol과 입력신호와의 연산은 다음 식으로 표현할 수 있다.

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

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 수행하면 상관 값이 구해지는데 여기서 2개의 실수부 곱셈 연산 블록과 2개의 허수부 곱셈 연산 블록이 필요하므로 총 4개의 곱셈 연산 블록이 필요하다. 이와 같은 4개의 곱셈 연산 블록을 포함한 심볼 타이밍 옵셋 동기화기의 구조는 그림 3과 같다.

그림 3의 심볼타이밍 옵셋 동기화기 구조에서 y_r은 수신신호의 실수, y_i는 수신신호의 허수를 나타내고 x_{r0}은 알고 있는 훈련 심볼의 0번째 실수샘플 값이고 15번까지 총 16개가 있고, x_{i0}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 블록은 $\sum y_r x_r$의 값과 $\sum y_i x_i$ 값의 합을 연산하고, 허수 곱셈 연산 블록은 $\sum y_i x_r$의 값과 $\sum y_r x_i$ 값의 합을 연산한다. 그림 3에서 보듯이 심볼 타이밍 동기화기는 64개의 곱셈기를 사용하고 있으므로 구현 면적이 커지게 된다. 따라서 이 논문에서는 상호상관방식을 기초로 하는 심볼 타이밍 옵셋 동기화기 구조의 곱셈기 파트를 효율적으로 설계하여 저면적으로 구현할 수 있는 구조를 제안한다.



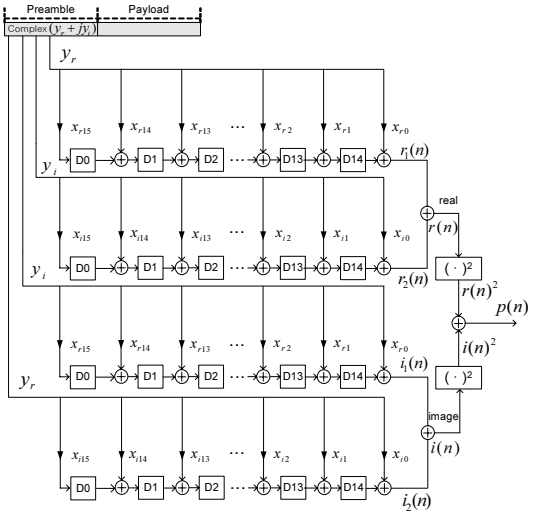
[그림 3] 심볼 타이밍 옵셋 동기화기 구조

3. 제안된 심볼 타이밍 옵셋 동기화기 구조

제안된 구조는 심볼 타이밍 옵셋 동기화기를 하나의 필터 개념으로 접근하여 설계한다. 필터 개념으로 접근하면 프리앰블의 Short training symbol의 16개 필터계수의 필터로 볼 수 있다. 그림 3의 구조에 대한 저면적 구조는 전치직접형 구조설계 단계(1단계), CSD 계수 설계 단계(2단계), CSS 구조 설계 단계(3단계)의 3 단계로 이루어진다.

3.1 전치직접형 구조 설계 단계(1단계)

15개의 곱셈기를 사용하는 심볼 타이밍 옵셋 동기화기는 매 clock 마다 복소수의 입력이 들어오며, 이는 수신기에서 미리 알고 있는 Short training symbol과 곱셈연산을 수행한다. 곱셈기 파트는 실수 파트 2개와 허수 파트 2개로 총 4개의 파트로 구성이 되며 하나의 곱셈기 파트는 15개의 곱셈기와 15개의 쉬프트 레지스터로 이루어진다. 따라서 총 곱셈기와 쉬프트 레지스터는 각각 60개가 필요하며 매 clock 마다 60번의 연산이 수행되어 구현 면적이 커지게 되는 단점이 있다. 이와 같은 단점을 CSS 방식을 사용하여 덧셈기와 쉬프터 만을 사용하여 구현한다. CSS 방식의 패턴 공유 방법을 쓰기 위해서는 그림 3의 직접형(Direct form) 필터 구조를 그림 4와 같이 전치 직접형(Transposed direct form)필터 구조로 바꿔야 한다.



[그림 4] 심볼 타이밍 옵셋 동기화기 구조의 전치 직접형 블록도

3.2 CSD 계수 설계 단계(2단계)

그림 4의 필터 구조에 대하여 덧셈기를 사용하는 저면적 구조를 설계하기 위하여 사용되는 필터 계수를 CSD 형으로 변환한다. CSD형으로 변환하려면 훈련 심볼들을 2의 보수형으로 나타낸 후 이를 CSD형으로 변환한다. CSD형으로 구현하면 사용되는 덧셈기의 수가 적어지기 때문이다. 표 1은 $r_1(n)$ 필터에서 사용되는 계수의 실수값을 16비트의 2의 보수형으로 나타낸 표이다.

[표 1] Short training symbol(REAL)의 2의 보수형

훈련 심볼 (REAL)	2의 보수형																덧셈기수
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0.0460	0	0	0	0	0	1	0	1	1	1	1	0	0	0	1	1	6
-0.1324	1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	0	9
-0.0135	1	1	1	1	1	1	1	0	0	1	0	0	0	1	1	0	9
0.1428	0	0	0	1	0	0	1	0	0	1	0	0	0	1	1	1	5
0.0920	0	0	0	0	1	0	1	1	1	1	0	0	0	1	1	1	7
0.1428	0	0	0	1	0	0	1	0	0	1	0	0	0	1	1	1	5
-0.0135	1	1	1	1	1	1	1	0	0	1	0	0	0	1	1	0	9
-0.1324	1	1	1	0	1	1	1	1	0	0	0	0	1	1	1	0	9
0.0460	0	0	0	0	0	1	0	1	1	1	1	0	0	0	1	1	6
0.0023	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	3
-0.0785	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	0	10
-0.0127	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	8
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-0.0127	1	1	1	1	1	1	1	0	0	1	1	0	0	0	0	0	8
-0.0785	1	1	1	1	0	1	0	1	1	1	1	1	0	1	0	0	10
0.0023	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1	1	3
합계	1의 개수 : 122개, 덧셈기의 개수 : 107개																

표 1에서 보듯이 2'의 보수형 계수는 상대적으로 1의 수가 많으므로 덧셈기로 구현할 때에 구현 면적이 커진다. 따라서 덧셈기의 수를 줄이기 위하여 CSD 형의 계수로 변환하여야 한다. 즉 필터계수로 사용되는 Short training symbol의 실수 값과 허수 값을 CSD 형으로 바꿔준다. 표 2는 표 1의 2의 보수형 계수를 16비트의 CSD 형으로 나타내었다. 표 2에서는 Short training symbol의 실수 값만을 나타냈는데 허수 값은 실수 값의 8 sample 지연한 값과 같으므로 생략한다.

[표 2] Short training symbol(REAL)의 CSD형

훈련 심볼 (REAL)	CSD																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	덧셈기수
0.0460	0	0	0	0	1	0	N	0	0	0	N	0	0	1	0	N	4
-0.1324	0	0	0	N	0	0	0	N	0	0	0	1	0	0	N	0	3
-0.0135	0	0	0	0	0	0	N	0	0	1	0	0	1	0	N	0	3
0.1428	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	N	4
0.0920	0	0	0	1	0	N	0	0	0	N	0	0	1	0	0	N	4
0.1428	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	N	4
-0.0135	0	0	0	0	0	0	N	0	0	1	0	0	1	0	N	0	3
-0.1324	0	0	0	N	0	0	0	N	0	0	0	1	0	0	N	0	3
0.0460	0	0	0	0	1	0	N	0	0	0	N	0	0	1	0	N	4
0.0023	0	0	0	0	0	0	0	0	0	1	0	1	0	N	0	N	3
-0.0785	0	0	0	0	N	0	N	0	0	0	0	N	0	1	0	0	3
-0.0127	0	0	0	0	0	0	N	0	1	0	N	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-0.0127	0	0	0	0	0	0	N	0	1	0	N	0	0	0	0	0	2
-0.0785	0	0	0	0	N	0	N	0	0	0	0	N	0	1	0	0	3
0.0023	0	0	0	0	0	0	0	0	0	1	0	1	0	N	0	N	3
합계	1 또는 N의 개수 : 63개, 덧셈기의 개수 : 48개																

4개의 곱셈기 파트 중 하나의 곱셈기 파트를 살펴보면 2의 보수형 구조를 구현하기 위해서는 107개의 덧셈기가 필요하며 기타의 덧셈기를 합하면 총 122개의 덧셈기가 필요하다.

CSD 구조를 구현하기 위해서는 덧셈기는 48개가 필요하며 기타의 덧셈기를 합하면 총 63개의 덧셈기가 필요하다. 이와 같이 CSD 형의 필터계수를 사용하면 덧셈기의 수는 2의 보수형 구조보다 약 48.63%의 덧셈기를 감소시킬 수 있다.

3.3 CSS 구조 설계 단계(3단계)

마지막으로 3단계는 덧셈기의 수를 더욱 감소시키기 위하여 CSS 방식을사용하는 구조를 유도한다. CSD 형의 계수에서 표 3의 2중 실선과 같이 공통 패턴을 정의하여야 한다.

표 3에서 CSS 방식을 적용하기 위하여 먼저 공통패턴들을 2중실선으로 표시하였다. 표에서 보면 10N, 101, 100N, 1001의 4개의 공통패턴이 있음을 알 수 있다. 여기서 10N, 101, 100N은 - 만 붙이면 N01, N0N, N001과 같은 패턴이므로 하나의 패턴으로 본다. 표의 공통패턴은 다음 식과 같이 표기할 수 있다.

[표 3] 16비트 훈련 심볼의 CSD 계수와 CSS

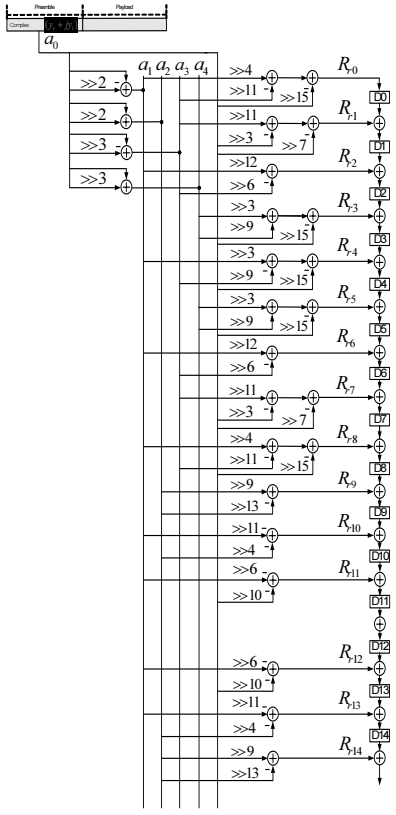
훈련심볼	CSD															
(REAL)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0.0460	0	0	0	0	1	0	N	0	0	0	N	0	0	1	0	N
-0.1324	0	0	0	N	0	0	0	N	0	0	0	1	0	0	N	0
-0.0135	0	0	0	0	0	0	N	0	0	1	0	0	1	0	N	0
0.1428	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	N
0.0920	0	0	0	1	0	N	0	0	0	N	0	0	1	0	0	N
0.1428	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	N
-0.0135	0	0	0	0	0	0	N	0	0	1	0	0	1	0	N	0
-0.1324	0	0	0	N	0	0	0	N	0	0	0	1	0	0	N	0
0.0460	0	0	0	0	1	0	N	0	0	0	N	0	0	1	0	N
0.0023	0	0	0	0	0	0	0	0	0	1	0	1	0	N	0	N
-0.0785	0	0	0	0	N	0	N	0	0	0	0	0	N	0	1	0	0
-0.0127	0	0	0	0	0	0	N	0	1	0	N	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-0.0127	0	0	0	0	0	0	N	0	1	0	N	0	0	0	0	0
-0.0785	0	0	0	0	N	0	N	0	0	0	0	N	0	1	0	0
0.0023	0	0	0	0	0	0	0	0	0	1	0	1	0	N	0	N

$$
\begin{aligned}
a_{1} &= a_0 - a_0 \gg 2 \\
a_{2} &= a_0 + a_0 \gg 2 \\
a_{3} &= a_0 - a_0 \gg 3 \\
a_{4} &= a_0 + a_0 \gg 3
\end{aligned} \tag{3}
$$

위의 공통패턴을 이용하여 각각의 훈련 심볼을 표현하면 식 (4)와 같다. 표 3의 순서대로 값들을 x_{r0}부터 x_{r15}까지로 정의하여 표현한다. 아래첨자 r은 실수를 의미한다.

$$
\begin{aligned}
x_{r0} &= a_1 \gg 4 - a_3 \gg 11 - a_0 \gg 15 \\
x_{r1} &= -a_0 \gg 3 - a_0 \gg 7 + a_3 \gg 11 \\
x_{r2} &= -a_3 \gg 6 + a_1 \gg 12 \\
x_{r3} &= a_4 \gg 3 + a_4 \gg 9 - a_0 \gg 15 \\
x_{r4} &= a_1 \gg 3 - a_3 \gg 9 - a_0 \gg 15 \\
x_{r5} &= a_4 \gg 3 + a_4 \gg 9 - a_0 \gg 15 \\
x_{r6} &= -a_3 \gg 6 + a_1 \gg 12 \\
x_{r7} &= -a_0 \gg 3 - a_0 \gg 7 + a_3 \gg 11 \\
x_{r8} &= a_1 \gg 4 - a_3 \gg 11 - a_0 \gg 15 \\
x_{r9} &= a_2 \gg 9 - a_2 \gg 13 \\
x_{r10} &= -a_2 \gg 4 - a_1 \gg 11 \\
x_{r11} &= -a_1 \gg 6 - a_0 \gg 10 \\
x_{r13} &= -a_1 \gg 6 - a_0 \gg 10 \\
x_{r14} &= -a_2 \gg 4 - a_1 \gg 11 \\
x_{r15} &= a_2 \gg 9 - a_2 \gg 13
\end{aligned} \tag{4}
$$

그림 4의 16탭 필터를 위에서 정의한 공통패턴을 사용하여 설계한 구조는 그림 5와 같다.



[그림 5] 제안된 심볼 타이밍 동기화기 구조

그림 5를 살펴보면 식 3에서 정의한 공통패턴을 왼쪽 상단부분에 구현하였다. 수신신호로부터의 입력은 공통패턴을 계산하여 a_1, a_2, a_3, a_4 신호를 만들어서 초기 입력 값과 함께 쉬프트와 덧셈, 뺄셈연산을 통해 각각의 15개의 연산 값들을 계산한다. 그 계산 값들은 R_{r0}부터 R_{r14} 라고 하며 여기서 아래 첨가 r은 실수 값을 의미한다. 각각 쉬프트 레지스터에 저장된 값들은 다음 clock에서 계산되는 R_{r0}부터 R_{r14} 연산결과와 덧셈연산이 이루어져 최종적으로 상관 값들의 합이 출력으로 나올 수 있게 설계하였다. 그림에서 확인 할 수 있듯이 공통 패턴을 공유하였으므로 48개를 사용했던 CSD 형보다 적은 26개의 덧셈기만으로 연산부를 설계하였다. 따라서 기타 덧셈기를 합한 총 덧셈기는 41개가 되며 이는 2의 보수형 구조보다 약 66.4%의 덧셈기 감소를 나타낸다. 표 4는 각 필터계수의 형에 따른 덧셈기의 수를 비교한 것이다.

[표 4] 계수의 형에 따른 덧셈기 수 비교

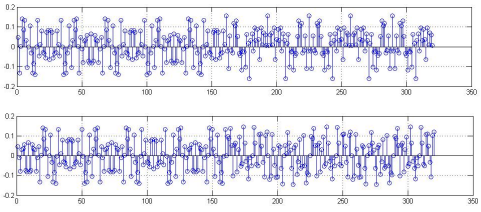
필터계수의 형	덧셈기의 수(비율)
2의 보수형	122(100%)
CSD	63(51.64%)
CSS	41(33.60%)

표 4를 살펴보면 2의 보수형의 덧셈기 수를를 100%로 정하였을 때 CSD 형 구조의 덧셈기 수는 51.64%이고 제안된 CSS형 구조의 덧셈기 수는 33.60%인 것을 확인 할 수 있었다.

4. 구현

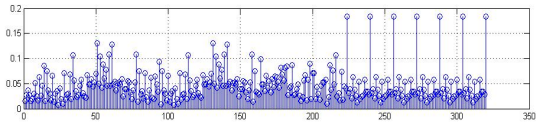
4.1 Matlab Simulation

제안구조를 검증하기 위하여 Matlab으로 그림 1의 프리앰블을 만들어 테스트 벡터를 생성하였다. Verilog-HDL로 코딩하여 출력 값을 테스트 벡터와 비교하였으며 완성된 RTL 코드는 Synopsys Design Complier로 합성하여 제안 구조에 대한 구현 면적을 계산하였다. 먼저 Matlab으로 Short training symbol 샘플 160개와 32개의 GI, Long training symbol 샘플 128개로 총 320개의 신호를 만든 후에, 옵셋을 인가하여 만들어진 프리앰블 신호 신호는 그림 6과 같다.



[그림 6] 실험에 사용된 수신 프리앰블 신호

그림 6에서 위의 그림은 주파수 옵셋이 섞인 프리앰블 실수 신호이고 밑의 그림은 허수 신호이다. 정확한 타이밍을 찾기 위해서는 수신기가 미리 알고 있는 Short training symbol과 수신신호와의 Cross-Correlation을 수행하여 peak를 7개 찾아내야 된다. MatLab을 사용하여 peak를 찾은 결과는 그림 7과 같다.



[그림 7] Cross-Correlation peak

그림 7은 알고 있는 16개의 Short training symbol과 그림 6의 신호에 대하여 Cross-correlation을 수행한 결과로서 7개의 peak를 찾아내었다. 표 5는 처음 peak부터 첫번째 peak 점까지의 상관 값들을 나타냈다.

[표 5] 수신신호와의 상관 값

	$r(n)^2$	$i(n)^2$	상관값
1	0.000036	0.000038	0.000074
2	0.000010	0.000420	0.000430
3	0.000732	0.000064	0.000795
4	0.000003	0.001575	0.001577
5	0.000496	0.000277	0.000772
6	0.000542	0.000129	0.000670
7	0.000003	0.000325	0.000327
8	0.000746	0.004459	0.005205
9	0.000209	0.000583	0.000791
10	0.000002	0.000391	0.000393
11	0.000038	0.000380	0.000418
12	0.002667	0.000091	0.002757
13	0.000015	0.000253	0.000268
14	0.000600	0.001773	0.002374
15	0.000102	0.001150	0.001251
16	0.017382	0.016284	0.033666

표 5에서 보듯이 첫 번째 peak 점까지의 실수 상관 값을 보면 16번째 값이 가장 큰 것을 볼 수 있고 16 샘플마다 peak가 나타고 있음을 확인 할 수 있다. 이는 수신신호와 수신기에서 알고 있는 Short training symbol이 일치할 때 상관 값이 가장 커진다는 것을 나타낸다. 따라서 Short training의 symbol의 수는 10개이기 때문에 상관 값 peak 10개를 찾으면 신호의 시작지점을 정확히 확인 할 수 있다.

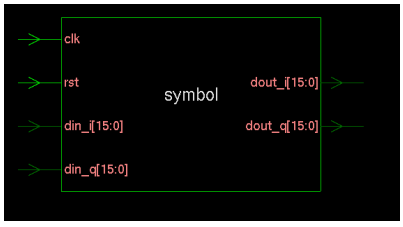
4.2 Verilog-HDL Function Simulation 및 합성

이 절에서는 제안 구조에 대하여 Verilog-HDL로 코딩하여 출력 값을 Matlab에서 만든 테스트 벡터와 비교하였다. 검증을 위한 입력신호로는 Matlab으로 만든 테스트 벡터를 사용하였으며 Verilog 코딩 후의 ModelSim 결과는 표 6과 같다.

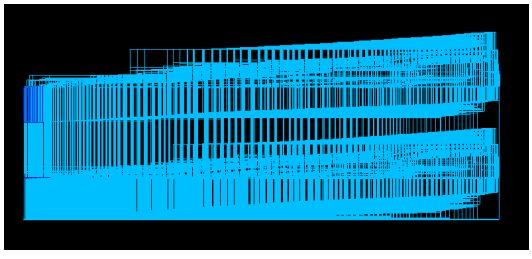
표 6에서 보듯이 ModelSim으로 확인한 값은 Matlab Simulation 값과 거의 일치하는 것을 알 수 있다. 완성된 RTL 코드는 Synopsis Design Compiler 툴을 사용하여 합성한 후 면적을 알아보았다. 합성을 위해 매그나칩 0.25-Micron 2.5V 공정을 사용하였다. 제안 구조의 합성 후 Schematic과 Symbol view는 각각 그림 8과 9와 같다.

[표 6] Matlab 값과 ModelSim 값의 비교

	상관 값	Matlab	ModelSim
1	0.000074	0000000000000010	0000000000000001
2	0.000430	0000000000001110	0000000000001001
3	0.000795	0000000000011010	0000000000010110
4	0.001577	0000000000110100	0000000000110011
5	0.000772	0000000000011001	0000000000011010
6	0.000670	0000000000010110	0000000000010110
7	0.000327	0000000000001011	0000000000001100
8	0.005205	0000000010101011	0000000010110100
9	0.000791	0000000000011010	0000000000011100
10	0.000393	0000000000001101	0000000000001100
11	0.000418	0000000000001110	0000000000001011
12	0.002757	0000000001011010	0000000001011010
13	0.000268	0000000000001001	0000000000001010
14	0.002374	0000000001001110	0000000001001010
15	0.001251	0000000000101001	0000000000101010
16	0.033666	0000010001001111	0000010001001101
17	0.000993	0000000000100001	0000000000101001
18	0.001531	0000000000110010	0000000000110110
19	0.000446	0000000000001111	0000000000001111
20	0.004015	0000000010000100	0000000010010001
21	0.001457	0000000000110000	0000000000110101
22	0.000659	0000000000010110	0000000000010110
23	0.000154	0000000000000101	0000000000000100
24	0.002860	0000000001011110	0000000001011010
25	0.000348	0000000000001011	0000000000001000
26	0.000609	0000000000010100	0000000000010011
27	0.003391	0000000000010110	0000000000010100
28	0.001171	0000000001101111	0000000001101101
29	0.000099	0000000000100110	0000000000101100
30	0.001161	0000000000100110	0000000000101001
31	0.000814	0000000000011011	0000000000011011
32	0.033666	0000010001001111	0000010001011010



[그림 8] 제안 구조의 Schematic



[그림 9] 제안 구조의 Symbol view

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

[표 7] 기존구조와 제안구조의 면적 비교

	Combinational area	Noncombinational area	Total cell area
기존구조 (그림 3)	0.688	0.0348	0.723(100%)
제안구조	0.194	0.0692	0.264(36.46%)

표 7을 보면 기존구조의 면적은 0.723 mm^2가 나왔고 제안구조는 0.264 mm^2로 계산되었다. 60개의 곱셈기를 사용한 구조를 100%라고 했을 경우 제안구조의 면적은 36.46%로 감소됨을 알 수 있다. 또한 타이밍 옵셋 동기 블록의 입력과 출력을 같도록 하드웨어를 구현한 것이기 때문에 기존 구조와 비교하여 동기 성능도 정확하게 같게 된다.

5. 결론

이 논문에서는 OFDM 통신 방식인 IEEE 802.11a WLAN의 심볼 타이밍 옵셋 동기화기 블록에 대한 저면적 구조를 제안하였다. 심볼 타이밍 옵셋 동기화기의 곱셈기를 필터 개념으로 설계를 하여 공통패턴을 찾아내 덧셈기와 쉬프트를 이용한 CSS 방식의 저면적 구조를 제안하였다. Verilog 코딩과 합성을 통하여 63%의 면적 감소 효과를 확인하였다. 따라서 제안된 심볼 타이밍 옵셋 동기화기 구조는 IEEE 802.11a WLAN용 OFDM Modem의 타이밍 옵셋 동기화기 블록에 적용함으로써 Modem SoC의 구현 면적을 감소시킬 수 있을 것이다.

참고문헌

[1] Zhongshan Zhang, Keping Long, Ming Zhao, Yuannan Liu. "Joint Frame Synchronization and Frequency Offset Estimation in OFDM Systems". IEEE Trans. Broadcasting, Vol. 51, No. 3, pp. 389-394, Sep. 2005.

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

[3] 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.
[4] 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.
[5] Schmidl Timothy M, Cox Donald C. "Robust Frequency and Timing Synchronization for OFDM". IEEE Trans. Commun, 45(12), pp. 1613-1621, 1997.
[6] IEEE 802.11, "Wireless MAC and PHY Specifications: High Speed Physical Layer in the 5 GHz Band", P802.11a.D7.0, July, 1999.
[7] 장영범, "필터 뱅크를 사용한 저전력 short-length running convolution 필터 설계 및 구현", 한국산학기술학회논문지, 제7권 제4호, pp. 625-634, 2006년 8월.
[8] 장영범, "SoC용 디지털 필터 및 TRANSFORM", 홍릉과학출판사, 2010.

하 준 형(Jun-Hyung Ha) **[정회원]**

- 2009년 2월 : 상명대학교 정보통신 공학사 졸업(공학사)
- 2011년 2월 : 상명대학교 컴퓨터정보통신공학과 대학원 졸업(공학석사)
- 2011년 1월 ~ 현재 : 이미지스 테크놀로지 주임연구원

<관심분야>
통신신호처리, SoC 설계

장 영 범(Young-Beom Jang) **[정회원]**

- 1981년 2월 : 연세대학교 전기공학과 졸업(공학사)
- 1990년 1월 : Polytechnic University 대학원졸업(공학석사)
- 1994년 1월 : Polytechnic University 대학원졸업(공학박사)
- 1981년 7월 ~ 1999년 12월 : 삼성전자 System LSI 사업부 수석연구원
- 2002년 9월 ~ 현재 : 상명대학교 정보통신공학과 교수

<관심분야>
통신신호처리, 비디오신호처리, SoC 설계