

# IEEE802.11g WLAN 시스템을 위한 OFDM 알고리즘의 DSP(TM320C6416) 구현

김 세현<sup>o</sup> 조성호<sup>o</sup>

한양대학교 정보통신공학과

{shkim<sup>o</sup>, shcho<sup>o</sup>}shkim@casp.hanyang.ac.kr

Information and Communications Hanyang University

Se Hyun Kim<sup>o</sup> Sung Ho Cho<sup>o</sup>

Dept. Information and Communications Hanyang University

## 요 약

본 논문에서는 무선랜 프로토콜 IEEE802.11g에서 사용되는 변조방식인 OFDM (Orthogonal frequency division multiplexing) 방식을 TMX320C6416을 사용하여 구현하였다. 통신 시스템의 발달에 따라 고속으로 데이터를 전송하는 방식이 요구되었고, 유선망 없이 장소에 구애 받지 않고 데이터를 송수신할 수 있는 무선랜 시스템의 필요성이 증가하고 있다. 이러한 필요성에 따라 2002년 6월 현재 발표된 IEEE 802.11g 규격 Draft 2.8은 2.4GHz 주파수 대역에서 기존 IEEE 802.11b 규격과 호환성은 유지하며 전송속도를 최소 20Mbps 이상을 목표로 한다. IEEE802.11g에 정의된 규정에 따라 OFDM 방식의 기저대역 알고리즘을 TMX320C6416을 사용하여 각 블록별로 C언어로 구현하였고, TMX320C6416에 대한 리니어 어셈블리 언어로 구현하는 최적화 작업을 통해 연산속도를 향상시키는 것을 목적으로 하며, 구현한 알고리즘의 비트 오류 확률을 매트랩 시뮬레이션 결과와 비교해 보도록 한다.

## 1. 서 론

본 논문에서는 무선랜 프로토콜인 IEEE802.11g에서 사용되는 핵심적인 변조방식인 OFDM (Orthogonal frequency division multiplexing) 방식을 TMX320C6416 DSP를 이용하여 실시간 구현을 목적으로 하고, QPSK 기반의 OFDM 변조 방식에 대해 Matlab 시뮬레이션으로 성능을 검증하였다.

소프트웨어 구현은 각 블록별로 C 언어로 구현한 뒤 전체 연산 속도를 줄이기 위해 리니어 어셈블리어 프로그램으로 재구성하였으며, 파이프라인(pipeline), unrolled 프로그래밍 등의 소프트웨어 최적화 기법을 사용하였다.

OFDM 방식에 대하여 간단히 고찰해 보도록 하고, 다음은 본 논문에서 구현한 무선랜 프로토콜 IEEE802.11g 시스템의 구성에 관하여 설명하였다. 일반적인 DSP와 구현에 사용된 TI사 TMX320C6416 DSP의 구조 및 특성을 설명하였고, 실제적인 구현으로 시뮬레이션 및 간단한 C 프로그램과 리니어 어셈블리어 프로그램의 비교 예와 프로그래밍 결과를 나타내었다. 마지막으로 결론을 맺는다.

## 2. OFDM 방식

OFDM 은 다중 반송파 전송방식의 일종이며, 데이터 열을

여러 개의 비트 열로 나누고 이 비트 열이 병렬처리 되어 낮은 데이터 전송률을 갖는 반송파를 통해 전송된다. 하나의 반송파를 사용하여 데이터를 순차적으로 전송하는 경우보다 전송되는 심볼 간격이 길어져서 채널의 지연시간 영향을 덜 받게 되며 연속된 심볼간의 간섭을 줄일 수 있어 다중 경로 채널에 강한 특징을 가지고 있다 [1]

## 3. IEEE802.11g 시스템의 구성

### 3.1 IEEE 802.11g 동작 모드

IEEE 802.11g 규격에는 ERP-DSS/CCK와 ERP-OFDM의 필수 동작 모드와 Extended Rate PBCC와 CCK-OFDM의 선택적 동작 모드 네 가지의 동작 모드가 규정되어 있다.

ERP-DSS/CCK 필수 동작 모드는 IEEE 802.11b 규격에 따라 1, 2, 5.5, 11Mbps의 전송속도를 갖고서 동작되며 IEEE 802.11b 규격에서 선택사항이었던 짧은 PLCP(physical layer convergence protocol) PPDU(PLCP protocol data units) 헤더 포맷의 지원이 필수사항으로 요구된다.

ERP-OFDM 필수 동작모드는 IEEE 802.11b의 채널 파라미터를 사용하면서 IEEE 802.11a 의 규격에 따라 6, 9, 12, 18, 24, 36, 48, 54Mbps의 다양한 전송속도를 갖고서

동작할 수 있다. 다양한 전송속도 중에서 6, 12, 24Mbps의 전송속도는 IEEE 802.11a에서와 마찬가지로 필수 전송속도이다

Extended Rate PBCC 선택적 동작 모드의 기본적 동작 시나리오는 5.5Mbps, 11Mbps 전송속도의 IEEE 802.11b 선택적 동작 모드인PBCC와는 동일하지만 부호화 방식과 변조 방식에 차이가 있다.

CCK-OFDM 선택적 동작모드는 IEEE 802.11b와의 호환성을 유지하기 위하여 IEEE 802.11b의 긴(long) 또는 짧은 프리앰블을 전송하고 이어서IEEE 802.11a의 긴 프리앰블과 SIGNAL 필드 그리고 데이터 순으로 전송하는 하이브리드 방식의 변조 모드이다. 이 선택적동작 모드에서의 전송 속도는 ERP-OFDM의 전송속도를 따른다.[3]

### 3.2 PPDU 포맷

IEEE 802.11g 규격은 세가지의 프리앰블과 헤더 포맷을 지원한다. 첫 번째는IEEE 802.11b의 긴 프리앰블과 헤더 포맷이다. 이 프리앰블과 헤더 포맷을 사용하는 PPDU는 1, 2, 5.5, 11Mbps의 전송속도를 사용할 때 상호운용성이 보장된다. 두 번째 지원 포맷은IEEE 802.11b의 선택사항인 짧은 프리앰블과 헤더 포맷이다. 이 포맷을 사용하는 PPDU는 2, 5.5, 11Mbps의 전송속도를 사용할 때 최대의 전송률을 얻을 수 있다. 세 번째로 지원되는 프리앰블과 헤더포맷은 IEEE 802.11a에 규정된 OFDM 프리앰블과 헤더이다.[3]

### 3.3 PLCP 송수신 절차

IEEE 802.11g 규격에서 송신절차는 요구되는 데이터 전송률과 변조 포맷에 따라서 달라지는데 1, 2, 5.5, 11Mbps의 필수 데이터 전송률은 IEEE 802.11b 규격의 전송절차를 따르며, 6, 12, 24Mbps의 필수 데이터 전송률과 9, 18, 36, 48, 54Mbps의 선택적 데이터 전송률은 IEEE 802.11a 규격의 전송절차를 따른다. 또한 IEEE 802.11g에서 새롭게 규정한 선택적인 PBCC 방식의22, 33Mbps 데이터 전송률과 선택적인 CCK-OFDM 변조 방식에서의 전송 절차는 IEEE 802.11g에 새롭게 정의하고 있다.[3]

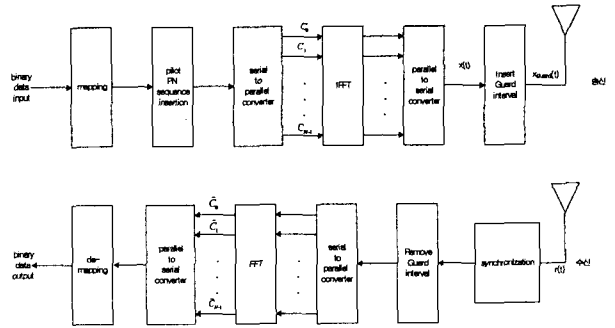


그림 1. IEEE802.11g의 기본 송수신기 구조

## 4. TMX320C6416 DSP의 구조 및 특성

본 논문에서 구현에 사용된 TMX320C6416 DSP칩은 TI(Texas Instrument)사의 고정소수점 DSP이며 빠른 성능과 향상된 VLIW (Very-Long-Instruction-Word)구조로 이루어져 있다. TMX320C6416은 600MHz의 CPU 클럭, 4800MIPS의 계산량 처리능력으로 우수한 성능을 나타낸다. [4]

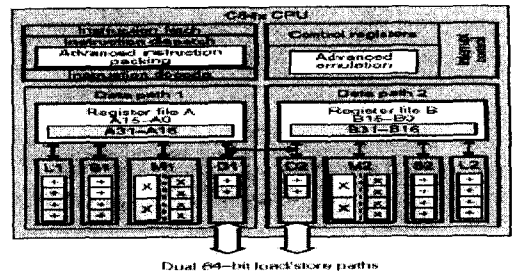


그림2. TMS320C6416 내부 블럭도

## 5. 시뮬레이션과 구현

본 논문에서는 IEEE802.11g 프로토콜의 규정에 따라 OFDM 알고리즘의 기저대역 알고리즘을 구현하였다 구현한 시스템은 그림 1이다.

C 프로그램을 최적화 시키더라도 데이터 처리속도에는 한계가 있으므로, 실시간 동작이 가능하도록 하기 위해서는 어셈블리 언어로 구현하는 것은 필수적이다. 본 논문에서는 TI사의 DSP중 6x 계열에서 지원되는 리니어 어셈블리어를 사용하여 구현하였다. 리니어 어셈블리어는 어셈블리어와 비교하여 95~100%의 효율성을 갖고, functional UNIT과 NOP (No-operation)등을 고려하지 않아도 된다.

```
void IFFT(int *Q_I, int *Q_Q, int *Ia, int *Iq, int *eI, int *eQ)
{
    int m, k, K, i, j, b;
    int cc, cr;
    b = 1;
    for(m=1; m<K; m++)
        I2(m+b);
        I2(m-b);
        Q_I[m] = Q_I[m];
        Q_Q[m] = Q_Q[m];
        Q_I[m] = eI;
        Q_Q[m] = eQ;
        m = K+m;
        while(m<K) {
            b=m-K;
            m=K+b;
        }
    while(
        m=K;
        while(m<K) {
            m=K-b;
        }
    }
    b=m-K;
}
```

그림3. 구현한 IFFT 과정(C 프로그램)

```

4          movl $4, %eax          ; tmp = 4
5          movl %eax, %ecx       ; tmp_bb = 4
6          movl $1, %eax        ; m = 1
7          movl %eax, %ecx      ; m = 1
8          movl $7, %eax        ; const = 7
9
10         movl %eax, %ecx       ; m = 1
11         movl $32, %ecx        ; k = 32
12         movl %ecx, %eax       ; k = b
13         movl %eax, %ecx       ; k = b
14         movl %ecx, %eax       ; k = b/2
15         movl %eax, %ecx       ; k = b/2
16         movl %ecx, %eax       ; k = b+k
17         movl %eax, %ecx       ; k = b+k
18         movl %eax, %ecx       ; k = b+k
19         movl %eax, %ecx       ; k = b+k
20         movl %eax, %ecx       ; k = b+k
21         movl %eax, %ecx       ; k = b+k
22         movl %eax, %ecx       ; k = b+k
23         movl %eax, %ecx       ; k = b+k
24         movl %eax, %ecx       ; k = b+k
25         movl %eax, %ecx       ; k = b+k
26         movl %eax, %ecx       ; k = b+k
27         movl %eax, %ecx       ; k = b+k
28         movl %eax, %ecx       ; k = b+k
29         movl %eax, %ecx       ; k = b+k
30         movl %eax, %ecx       ; k = b+k
31         movl %eax, %ecx       ; k = b+k
32         movl %eax, %ecx       ; k = b+k
33         movl %eax, %ecx       ; k = b+k
34         movl %eax, %ecx       ; k = b+k
35         movl %eax, %ecx       ; k = b+k
36         movl %eax, %ecx       ; k = b+k
37         movl %eax, %ecx       ; k = b+k
38         movl %eax, %ecx       ; k = b+k
39         movl %eax, %ecx       ; k = b+k
40         movl %eax, %ecx       ; k = b+k
41         movl %eax, %ecx       ; k = b+k
42         movl %eax, %ecx       ; k = b+k
43         movl %eax, %ecx       ; k = b+k
44         movl %eax, %ecx       ; k = b+k
45         movl %eax, %ecx       ; k = b+k
46         movl %eax, %ecx       ; k = b+k
47         movl %eax, %ecx       ; k = b+k
48         movl %eax, %ecx       ; k = b+k
49         movl %eax, %ecx       ; k = b+k
50         movl %eax, %ecx       ; k = b+k
51         movl %eax, %ecx       ; k = b+k
52         movl %eax, %ecx       ; k = b+k
53         movl %eax, %ecx       ; k = b+k
54         movl %eax, %ecx       ; k = b+k
55         movl %eax, %ecx       ; k = b+k
56         movl %eax, %ecx       ; k = b+k
57         movl %eax, %ecx       ; k = b+k
58         movl %eax, %ecx       ; k = b+k
59         movl %eax, %ecx       ; k = b+k
60         movl %eax, %ecx       ; k = b+k
61         movl %eax, %ecx       ; k = b+k
62         movl %eax, %ecx       ; k = b+k
63         movl %eax, %ecx       ; k = b+k
64         movl %eax, %ecx       ; k = b+k
65         movl %eax, %ecx       ; k = b+k
66         movl %eax, %ecx       ; k = b+k
67         movl %eax, %ecx       ; k = b+k
68         movl %eax, %ecx       ; k = b+k
69         movl %eax, %ecx       ; k = b+k
70         movl %eax, %ecx       ; k = b+k
71         movl %eax, %ecx       ; k = b+k
72         movl %eax, %ecx       ; k = b+k
73         movl %eax, %ecx       ; k = b+k
74         movl %eax, %ecx       ; k = b+k
75         movl %eax, %ecx       ; k = b+k
76         movl %eax, %ecx       ; k = b+k
77         movl %eax, %ecx       ; k = b+k
78         movl %eax, %ecx       ; k = b+k
79         movl %eax, %ecx       ; k = b+k
80         movl %eax, %ecx       ; k = b+k
81         movl %eax, %ecx       ; k = b+k
82         movl %eax, %ecx       ; k = b+k
83         movl %eax, %ecx       ; k = b+k
84         movl %eax, %ecx       ; k = b+k
85         movl %eax, %ecx       ; k = b+k
86         movl %eax, %ecx       ; k = b+k
87         movl %eax, %ecx       ; k = b+k
88         movl %eax, %ecx       ; k = b+k
89         movl %eax, %ecx       ; k = b+k
90         movl %eax, %ecx       ; k = b+k
91         movl %eax, %ecx       ; k = b+k
92         movl %eax, %ecx       ; k = b+k
93         movl %eax, %ecx       ; k = b+k
94         movl %eax, %ecx       ; k = b+k
95         movl %eax, %ecx       ; k = b+k
96         movl %eax, %ecx       ; k = b+k
97         movl %eax, %ecx       ; k = b+k
98         movl %eax, %ecx       ; k = b+k
99         movl %eax, %ecx       ; k = b+k
100        movl %eax, %ecx       ; k = b+k

```

그림4. 구현한 IFFT 과정(ASM 프로그램)

예시된 프로그램은 decimation-in-time방식의 FFT 과정 중 비트 전환 연산의 일부분이다.

구현한 알고리즘에서 송신단의 경우, BPSK로 매핑된 경우 입력되는 데이터율은 12Mbps이므로 외부메모리로 입력되는 데이터를 48비트씩 읽어들이며 하나의 OFDM 심볼을 처리하는 사이클 수로부터 12Mbps의 데이터를 처리하는 총 사이클 수를 구하였다. 수신단에서는 I채널, Q채널 각각 80개의 데이터를 읽어들이며 하나의 심볼을 처리하는 사이클 수로부터 20MSPS의 전송률을 갖는 수신 데이터를 처리하는 총 사이클 수를 구하였다.

	송신단(*10 <sup>6</sup> )	수신단(*10 <sup>6</sup> )
c언어 프로그램	10110.3	11600.9
리니어어셈블리	1505.3	1953.1
파이프라인기법	871.2	920.5
Unrolled프로그램	720.9	785.4

표 1. 구현한 알고리즘 사이클 수

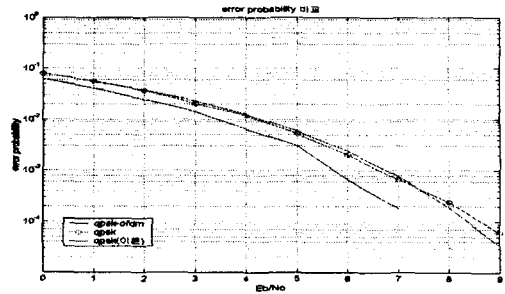


그림5. 구현한 OFDM 비트 오류 확률

### 6. 결론

IEEE802.11a 프로토콜에서 지원하는 다양한 데이터 전송률에 대해 DSP TMX320C6416에서 실시간으로 데이터 처리가 가능하도록 소프트웨어 최적화 작업을 하였으며, 이론치와 비트 오류 확률을 비교하였다.

TMX320C6416의 전체 CPU 클럭 수의 약 60% ~ 97%가량의 사이클 수를 사용하여 실시간으로 데이터 처리가 가능하였고, Q format으로 인한 데이터의 손실로 약간의 오차는 있지만 비교적 우수한 성능을 가지므로, 향후 무선랜 시스템과 OFDM 변복조 방식을 사용하는 시스템에서 기저대역 모델로 사용이 가능하다.

그러나 본 논문에서는 IEEE802.11g 프로토콜의 모든 기능을 구현한 것은 아니다. 2.4G 주파수 대역을 사용하는 IEEE802.11b의 프로토콜의 기능을 전송속도 측면에서 IEEE802.11a의 기능에 견줄 수 있게 하는데 의의를 두고자 한다.

### 7.참고문헌

- [1] 조용수 역, 무선 멀티미디어 통신을 위한 OFDM 기초, 대영사, 2000.
- [2] IEEE 802.11a Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications, High-speed Physical Layer in the 5GHz Band, 1999
- [3] DRAFT Supplement to STANDARD [for] Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specification, 2002