UMTS 송수신 필터의 최적 설계 및 효율적인 구현

김춘기°, 오우진
금오공과대학교 전자공학부

Optimal Design and Efficient Implementation of UMTS Tx/Rx Filter

Chun-Gi Kim°, Woo-Jin Oh
Dept. of Electronic Eng., Kumoh National University of Tech.
wjoh@knut.kumoh.ac.kr

* 본 논문은 산학연 공동 기술개발 지역 컨소시엄 연구개발 사업의 지원 결과임

요 약

본 논문에서는 UMTS(Universal Mobile Telecommunications System)의 규격에 명시된 SRCF(Square Root Raised Cosine Filter) 와 호환성을 가지면서, 우수한 성능과 구현이 간단한 필터를 설계 및 구현하는 방법에 대하여 소개한다. 또한, VHDL을 이용하여 ALTERA FPGA에서 필터를 직접 구현하는 결과를 보이고 있다.

** 필터 설계부분은 특허관계로 생략함*

1. 서 론

디지털 통신 시스템에서는 송신단에서 심볼간의 간섭, 즉 ISI(Inter Symbol Interference)를 제거하고 전송 대역을 제한하기 위하여 송신 필터를 사용하고 있다. 일반적으로 많이 사용되는 송신 필터는 RCF (Raised Cosine Filter)로, RCF는 Nyquist Criterion을 만족하여 ISI=0인 특징을 갖고 있으며, 다음과 같은 설계 식으로 표현된다[1].

$$p(t)=\left(\frac{\sin(\pi t/T)}{\pi t/T}\right)\left(\frac{\cos(\alpha\pi t/T)}{1-(2\alpha t/T)^2}\right) \quad (1)$$

$$P(\omega)=\begin{cases} T & 0\le|\omega|\le(1-\alpha)\pi/T \\ \frac{T}{2}\left(1-\sin\left[\frac{T}{2\alpha}(|\omega|-\pi/T)\right]\right) & (1-\alpha)\pi/T\le|\omega|\le(1+\alpha)\pi/T \\ 0 & |\omega|>(1+\alpha)\pi/T \end{cases}$$

여기서 α는 roll-off factor이다.

UMTS에서는 (1)의 Root-squared형인 Root-raised Cosine을 송신 필터로 사용하며, 다음 식으로 규정하고 있다.

$$RC_0(t)=\frac{\sin\left(\pi\frac{t}{T_C}(1-\alpha)\right)+4\alpha\frac{t}{T_C}\cos\left(\pi\frac{t}{T_C}(1+\alpha)\right)}{\pi\frac{t}{T_C}\left(1-\left(4\alpha\frac{t}{T_C}\right)^2\right)} \quad (2)$$

여기서 α는 Roll-off Factor로 0.22, T_c는 칩 주기로 0.26042μs이다[2]. 이 필터는 송신단과 수신단에 공통으로 사용되어 상호 결합된 최소의 ISI를 갖는 RCF의 특성을 얻는 특징이 있다. UMTS에서 사용하는 식 (2)의 RCF형 송신 필터는 그림 1과 같이 주파수 특성이 차단주파수 (Stopband Frequency) 근처에서 작은 감쇄를, 고주파에서 큰 감쇄를 갖고 있다. 따라서 UMTS에서 요구하는 45dB 이상의 저지대역 감쇄를 얻으려면 약 177탭이상이 필요하여 실제 구현이 어려운 단점이 있다.

SRCF(Square Root RCF)와 RCF 필터는 식 (1), (2)의 Closed-form으로 표현되기 때문에 부동 소수점 계수(Floating Point Coefficients)로 간단히 설계할 수 있다. 그러나 실제 구현에 사용되는 고정 소수점 계수(Finite Precision Coefficients)로 설계하려면 부동 소수점 계수를 단순한 Rounding 이나 Simple Search와 같은 국부최적(Local Optimal)의 방법을 적용해야 하므로 부동 소수점에 비하여 성능이 저하되는 단점이 있다.

표 1에서는 다양한 필터 길이에 대한 SRCF의 저지대역 감쇄와 두개의 SRCF를 연속으로 통과 시켰을 때의 저지대역 감쇄와 ISI를 제시하였다. 이때 각 계수는 Floating Point와 Rounding에 의한 12bit 계수 두 가지를 보였으며, 45dB 이상의 감쇄를 위해서는 12bit, 177 탭 필터를 사용해야 한다.

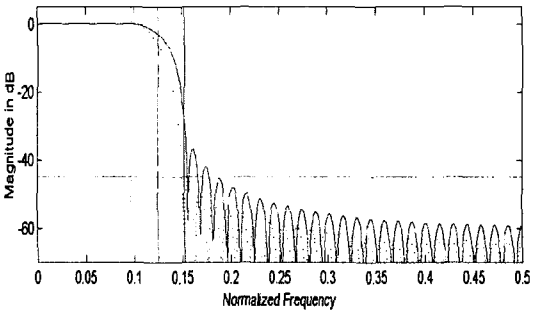

그림 1. 69탭 RCF(점선)와 71탭 SRCF(실선)의 일반적인 주파수 특성

표 1 여러 가지 길이의 필터 성능 비교
(단 12bit 계수는 부호비트를 포함하여 rounding한것임)

종 류	계수 비트수	Ripple(dB)	송수신단 전체의 ISI
71 탭 SRCF	floating	-30.38	0.00123
	12bits	-30.30	0.00164
177 탭 SRCF	floating	-44.03	0.000273
	12bits	-44.85	0.00113
71 탭 제안된 필터	floating	-49.74	0.00158
	12bits	-44.88	0.00183

2. 최적 설계

구현이 간단하고 대역제한 특성이 우수한 필터는 필터의 길이가 가장 짧으면서 저지대역 감쇄가 크다는 의미이며, 이러한 특성을 만족하기 위해서는 Equiripple형으로 설계해야 한다. Equiripple형은 Mini-max Criterion에 의해 가장 큰 Ripple을 최소화 하도록 설계하는 것으로 주어진 탭 수에서 저지대역 감쇄가 가장 큰 (Ripple이 가장 작은) 최적의 필터를 얻게 된다. 이러한 형의 필터는 Parks-McClellan Algorithm (Remez Exchange Algorithm이라고도 부름)을 많이 사용하고 있다.

제안된 필터 설계 방법은 표 1에 보인 바와 같이 선형 계획법(*Linear Programming*)을 이용하여 최적의 Ripple 특성을 가지고 있다. 또한 표 2와 같이 SRCF와 거의 호환되는 특징이 있어 제안된 필터를 송수신단 중에 이동국쪽과 같이 한쪽에만 사용할 수도 있다. 제안된 필터의 사용으로 인한 ISI의 증가 비율은 크지만, 전체적인 ISI값이 작은 값이므로 전체 성능에 미치는 영향이 적을 것으로 판단된다.

그림 2에 71 탭 제안된 필터와 71탭과 177탭 SRCF에 대한 주파수 특성을 보였다.

표 2. SRCF와 제안된 필터의 결합 성능 비교(실수형계수)

구 분		71탭 SRCF	177탭 SRCF	제안된 필터
71탭 SRCF	Ripple	-50.75	-74.41	-90.60
	ISI	0.00123	0.000725	0.00289
177탭 SRCF	Ripple	-	-88.07	-94.64
	ISI	-	0.000273	0.00267
71탭 제안된필터	Ripple	-	-	-99.49
	ISI	-	-	-0.00158

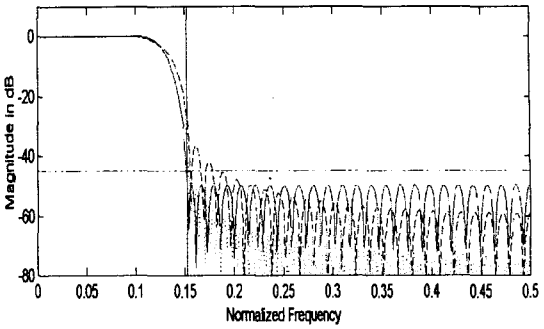

그림 2. SRCF와 제안된 필터의 주파수 특성 비교: 71 탭의 제안된 필터(실선), 71탭 SRCF(점선), 177 탭 SRCF(굵은 점선)

3. 구 현

3.1 필터 구현

디지털 필터를 하드웨어로 구현할 경우, 복잡도와 계산 속도 사이의 Trade-off를 고려하여 사용 목적에 맞게 가장 효과적인 방법으로 구현을 하여야 한다. 일반적으로 필터를 구현할 경우 곱셈기를 많이 사용하고 있으나, 계산의 속도나 하드웨어의 복잡도로 인해 shift-add방법등을 이용하기도 한다[3]. 고정된 계수를 갖는 필터를 구현할 경우 곱셈기 보다는 shift-add방법을 이용하면 효율적인 구현이 가능하다.

본 논문에서는 필터 계수가 고정된 경우로 가정하고, shift-add의 계산 방법을 적용하여 디지털 필터를 구현하는 방법을 제시하겠다.

먼저 필터의 계산량을 감소시키기 위하여 필터 계수를 살펴보면, 계수의 non-zero개수에 따라 계산량이 증가하게 됨으로 필터 계수의 non-zero개수를 최소화 시켜야 한다. 이를 위해 각 탭의 필터 계수를 CSD (Canonical Signed Digits)형식으로 바꾸어 적용함으로써, 최대 non-zero의 개수를 필터 계수 길이의 절반 이하로 감소시켰다. 디지털 필터를 하드웨어로 구현할 경우에 필터 계수를 CSD숫자로 표현하는 방법은 계산의 속도를 증가시키면서 복잡도를 감소시키는 매우 효과적인 방법으로 제안되어 있다[4][5].

그림 3은 Transposed FIR 필터의 1탭의 곱셈 계산을 shift-add방법으로 구현한 그림이다. 그림 위쪽의 점선 부분의 곱셈을 3개의 shift와 2개의 add를 이용하여 그림의 아래쪽과 같이 계산할 수 있다. 필터 계수가 고정된 경우, 그림 3의 shift-add방법의 구조는 단순한 Hard-wired shift이므로 계산을 간단하게 할 수 있다는 장점이 있다.

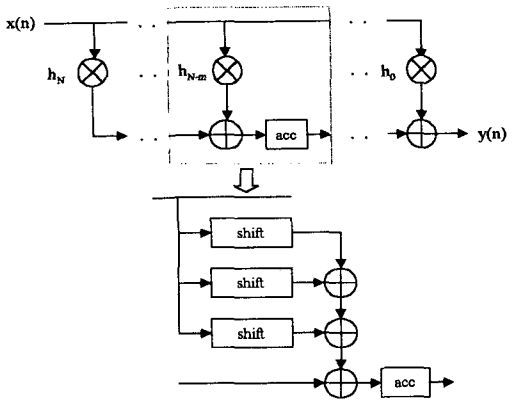

그림 3. shift-add 방법의 구조(non-zero 개수가 3인 경우)

그러나 고속의 디지털 필터를 구현할 경우, shift-add방법은 각 탭의 non-zero개수에 따라 계산 지연이 달라질 수 있다. 이 경우 필터 계수의 최대 non-zero개수에 맞추어 shift와 add부분에 Pipeline을 적용하여 계산량을 분산시키기면 고속의 계산이 가능하다[6][7]. 그림 4는 Pipeline을 고려한 shift-add방법의 필터 탭 구조를 나타낸 그림이다. 각 단계에 래치를 두어 각각의 탭을 최대 non- zero개수에 맞추어 일률적으로 계산함으로써 각 탭의 계산 지연을 동일하게 하였다.

일반적으로 VHDL로 필터를 프로그래밍 할 경우, 필터의 입력, 출력, 계수 및 내부 acc의 가변에 따라 프로그램을 매번 다시 프로그래밍을 해야하는 단점이 있다. 이를 보완하기 위하여, 구현 조건에 따라 VHDL 프로그램을 생성하는 프로그램을 Matlab를 이용하여 작성하였다.

그림 5는 Matlab 프로그램을 이용하여 생성한 제안된 방법의 VHDL프로그램 구성을 보여주고 있다.

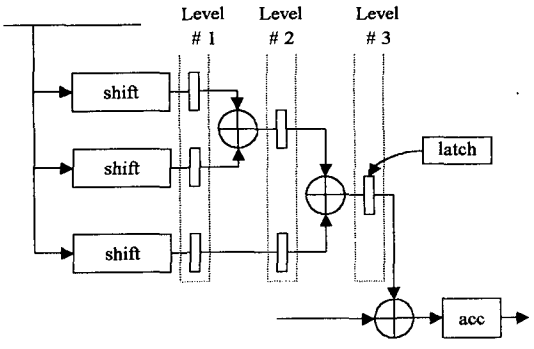

그림 4. Pipeline을 갖는 shift-add방식의 탭 구조 (Non-Zero 개수가 3개인 경우)

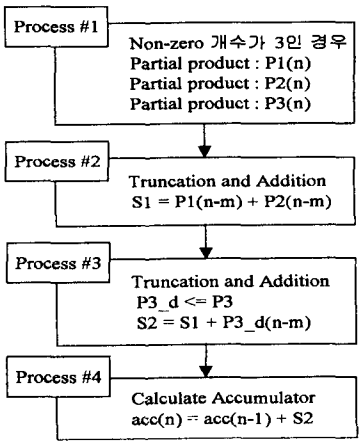

그림 5. VHDL 프로그램의 구성 (3 level Pipeline의 경우)

Process #1에서는 필터 계수의 non-zero개수만큼 Partial Products를 구한다. Process #2에서는 Partial Products한 P1과 P2를 유효 비트의 범위에 따라 불필요한 부분을 잘라내고 덧셈을 한다. Process #3에서는 Pipeline 적용 순서를 같게 하기 위하여 P3을 한번 지연시킨 후, 유효 비트 범위로 제한하여 Process #2의 결과와 더한다. 마지막으로 Process #4에서는 S2를 이전 탭의 acc에 더하여 현재 acc의 값을 계산한다.

3.2 성능 비교

본 장에서는 위에서 제안한 디지털 필터 구현 방법의 성능을 ALTERA사의 FIR Compiler MegaCore Function[8]과 비교하여 분석하였다. VHDL(Very High Speed Intergrated Circuits HDL)로 설계된 필터를 MAX+PLUS II 9.3 소프트웨어로 동일한 조건에서 컴파일을 하였으며, 필터 구현에 사용된 LC(Logic Cell)의 개수로 복잡도를 산출하였고 타이밍 시뮬레이션상의 최대 속도를 계산하여 비교하였다.

표 3, 4의 성능 비교에 사용된 필터는 공통적으로 16탭이며 8비트 계수와 8비트 입력의 경우에 대하여 복잡도와 계산 속도를 비교하였다. 표 3은 Non-pipeline의 경우로, 일반적인 16비트 출력을 갖는 I 번의 경우 제안된 방법의 LC사용 개수는 388개로 FIR Compiler의 537개에 비하여 28% 감소하였다. 출력을 12 비트로 제한한 II의 경우에는 FIR Compiler보다 제안된 방법의 복잡도가 39%까지 감소함을 알 수 있다. 속도에서도 I,II의 제안된 방법이 85(MSPS) 정도로 FIR Compiler의 50(MSPS)보다 약 70% 개선되었다.

표 3. Non-pipeline의 경우 필터 성능 비교

		acc 비트수	출력 비트수	복잡도 (LC개수)	Speed (MSPS)
I	FIR Compiler	-	16	537	50
	제안된 필터	16	16	388	83
II	FIR Compiler	-	12	528	50
	제안된 필터	16	12	320	85

표 4. Pipeline의 경우 필터 성능 비교

		acc 비트수	출력 비트수	복잡도 (LC개수)	Speed (MSPS)
III	FIR Compiler	-	16	659	111
	제안된 필터	16	16	476	151
IV	FIR Compiler	-	12	643	125
	제안된 필터	16	12	470	151
		12	12	382	156
V	FIR Compiler	-	8	628	119
	제안된 필터	8	8	378	167

표 4는 Pipeline을 적용하여 고속의 동작이 가능한 경우로, 일반적인 16비트 출력을 갖는 III의 경우 제안된 방법의 LC사용 개수는 476개로 FIR Compiler의 659개에 비하여 28% 감소하였고, 속도는 제안된 방법이 151(MSPS)로 FIR Compiler의 111(MSPS)보다 36% 증가하였다. 출력을 12비트와 8비트로 제한한 IV, V의 경우는 FIR Compiler보다 제안된 방법의 복잡도가 약 40%까지 감소하였고, 속도에서도 제안된 방법이 IV의 경우 25%, V의 경우 40% 개선되었다.

필터를 하드웨어로 구현할 경우 입력과 필터 계수에 따라 출력의 크기를 제한하는 것은 하드웨어의 자원을 절약한다는 점에서 매우 중요하다. 일반적으로 입력과 필터 계수가 각각 8비트인 경우 출력으로 16비트가 필요하지만, 대부분의 필터 구현에서 16비트를 그대로 사용하는 경우는 적다. 이 경우 16비트의 출력을 12비트나 8비트로 제한하여 많이 사용하지만 출력 값의 정확도 때문에 문제가 될 수 있다. 하지만 출력을 제한한 결과가 필터의 사용 용도에 크게 문제되지 않는다면, 필터 구현에 있어서 출력의 크기를 제한하는 것이 하드웨어의 복잡도를 감소시키는 이점이 있으므로 더욱 효율적일 것이다[9].

그림 6은 위에서 비교에 사용한 16탭의 Altera FIR Compiler MegaCore Function과 제안된 방법의 필터

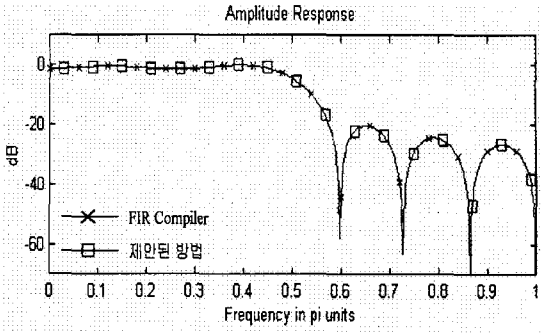

그림 6. Altera FIR Compiler와 제안된 방법의 Amplitude Response

출력 특성을 나타내고 있다. 그림에서 보듯이 두 방법의 필터 특성은 동일한 결과를 보여주고 있다.

4. 향후계획

현재 시분할 방법과 Polyphase기법 등을 이용하여 UMTS 송수신 필터를 구현 중에 있으며, 보다 더 효율적인 구현을 위하여 I/Q 채널의 혼합 구현에 대한 연구가 진행중이다.

참고문헌

[1] E. A. Lee and D. G. Messerschmitt, Digital communication, 2nd MA: Kluwer Academic Publishers. 1994.

[2] TS25.101 3GPP TSG RAN WG4 UE Radio Transmission and Reception, 3GPP, 1999.

[3] Altera Application Notes 73 Implementing FIR Filters in FLEX Devices, http//:www.altera.com, Feb. 1998.

[4] H. Kim, "Computer Simulation Results and Analysis for a Root-Raised Cosine Filter Design Using Canonical Signed Digits", NASA TM-107327, 1996.

[5] K. Hwang, Computer Arithmetic Principles, Architecture and Design, John Wiley&sons Inc., 1979.

[6] D. Li, "Minimum Number of Adders for Implementing a Multiplier and Its Application to the Design of Multiplierless Digital Filters", *IEEE Trans. CAS*, V.42 N.7, July 1995.

[7] H. Kropp, C. Reuter, T.-T. Do, P. Pirsch, "A Generator for Pipelined Multipliers on FPGAs", *Proceedings of the International Conference on Signal Processing Applications & Technology*, Vol. 1, 669-673, Sep. 1998.

[8] Altera FIR Compiler MegaCore Function User Guide, Sep. 1999.

[9] J. Valls, M. M. Peiro, T. Sansaloni, E. Boemo, "Design and FPGA implementation of Digit-Serial FIR filters", *Proc. of the 1998 IEEE ICECS*, Vol. 2, 191-194, Sep. 1998.