계수 초기화 방식의 128-Tap FIR 필터 설계

이 근 택, 이 찬 호, 송 인 채
숭실대학교 전자공학과
전화 : (02) 816-6073 / 팩스 : (02) 821-7653

Design of a Coefficient-Loadable 128-Tap FIR Filter

Geun-Taek Lee, Chanho Lee, Inchae Song
Department of Electronics Engineering, Soongsil University
E-mail : leda@hanul.soongsil.ac.kr

Abstract

We designed a 128-tap FIR filter for a modem which complies with ITU-T V.32. We adopted pipeline technique and realized delay-taps with two ring-buffers. The multiplier in this filter carries out 2's complement fixed-point multiplication of 14bit × 16bit. The designed filter is expected to operate at 50MHz.

Ⅰ. 서 론

일반 전화회선을 이용하는 디지털 통신기인 모뎀에는 반송파(carrier)를 걸러내는 필터가 사용되는데 이는 FIR 필터로 만들 수 있다. 여기서는 ITU-T V.32(International Telecommunication Unit Telecommunication Sector)[1] 규격의 모뎀에 기초하여 최대 동작 주파수가 50 MHz인 FIR 필터를 설계하였다. 동작 속도를 높이기 위해서 파이프라인(pipeline) 구조를 사용하였고, 계수의 좌우대칭 성질을 이용하여 곱셈연산 횟수를 줄였다. 0.65μm CMOS 기술을 사용하였고, 반도체설계교육센터(IDEC)에서 지원해 준 Cadence tool과 Hspice를 이용하여 simulation을 수행하였다.

Ⅱ. FIR 필터의 구조와 동작

필터 delay-tap의 총 수는 128개이며 계수(coefficient)는 좌우 대칭이다. 이 경우를 수식으로 표현하면 식 (1)과 같이 표현된다.

$$\begin{aligned} y[n] = \; & x[n]c[0]+x[n-1]c[1]+\cdots \\ & +x[n-\tfrac{k}{2}+1]c[\tfrac{k}{2}-1] \\ & +x[n-\tfrac{k}{2}]c[\tfrac{k}{2}-1]+\cdots \\ & +x[n-k+1]c[0] \end{aligned} \quad (1)$$

식 (1)에서 k는 필터의 delay-tap의 수이다. 식 (1)에서도 보는 바와 같이 각 delay-tap의 중앙을 기준으로 서로 대칭인 delay-tap에 저장된 입력 값들은 같은 계수를 곱하게 된다. 따라서 같은 계수를 곱하는 두 개의 입력을 미리 합한 뒤 계수를 곱할 수 있다. 이를 다시 수식으로 표현하면 식 (2)와 같이 표현할 수 있다.

$$y[n]=\sum_{i=0}^{\frac{k}{2}-1}(x[n-i]+x[n-k+i+1])c[i] \quad (2)$$

따라서 계수의 반만을 필터 동작 처음에 외부에서 읽어들여 메모리에 저장한 뒤 사용한다.

모든 계산은 고정 소수점방식(fixed point number)을 이용하였으며 이 방식은 2의 보수 이진 연산과 자리 옮김(shift)동작으로 쉽게 구현 할 수 있다. 입력 값과 계수 값의 범위는 -1 보다 크고 1보다 작은 값이 된다.

필터는 각 연산에 따라 세 단계로 구분되며 각 단계는 파이프라인(pipeline)구조를 이용하여 처리속도를 향상시킨다. 첫 번째는 delay-tap의 두 값을 읽어내는 동작이고, 두 번째는 계수를 읽어내는 동작과 첫 번째 동작에서 얻은 값을 더하는 동작이다. 그리고 세 번째 동작은 계수를 곱하여 accumulator에 저장하는 동작이다. 각 단계는

한 클럭 사이클(clock cycle)동안에 동시에 동작한다. 그림 1은 FIR 필터의 블록 다이어그램이다.

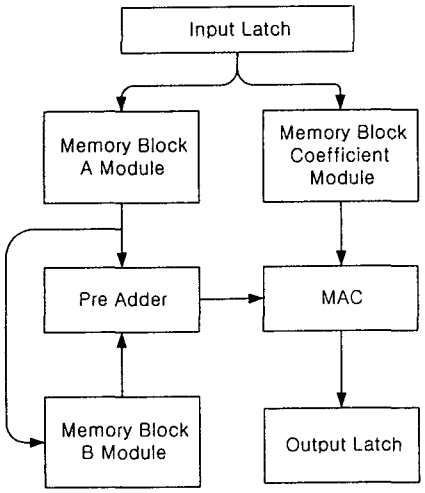

그림 1 FIR 필터의 전체 구조

Ⅲ. FIR 필터의 설계

3-1. 메모리 모듈의 설계

Delay-tap은 SRAM을 이용하여 환형 버퍼(ring buffer)를 만들어 사용한다[2]. 연산 시간을 줄이기 위해서는 좌우대칭의 입력 값 중 같은 계수를 곱하는 입력 값들을 동시에 꺼내야 한다. 하지만 두 개의 입력 값들의 delay 순서가 반대이므로 환형 버퍼를 하나로 구성하지 않고 둘로 나누어 방향을 다르게 한다. 그러면 두 개의 환형 버퍼에서 값을 읽어내는 방향이 다르더라도 독립적으로 동작하므로 쉽게 제어할 수 있다.

SRAM에서 값을 읽어내는 주소(address)나 저장하는 주소는 shift register를 이용하여 만들어 낸다. 일반적인 램(RAM)의 address decoder대신에 shift register를 사용하면 별도의 주소 생성기(address generator)를 사용하지 않고도 쉽게 주소를 만들어 낼 수 있다. 그림 2는 주소를 만들어내는 shift register의 구조를 보여주고 있다. Shift register는 SRAM의 row address decoder와 column address decoder를 대신하기 위하여 두 개의 작은 shift register로 나뉘어진다. 이 두 개를 각각 row shift register, column shift register라 하고, row shift register는 column shift register의 값이 완전히 한 바퀴를 돌면 한번 이동한다. Shift register를 조절(control)하기 위해서 외부에 shift 신호(signal)를 두었다.

그림 3은 SRAM을 이용한 환형 버퍼의 전체 구조이다. 램(RAM) cell을 세로로 16줄, 가로로 56줄(14bits × 4words)을 배열하므로 column shift register는 4비트로 구성되고, row shift register는 16비트로 구성된다.

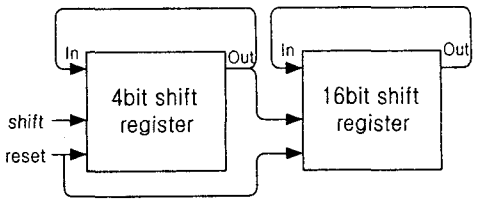

그림 2 Shift register

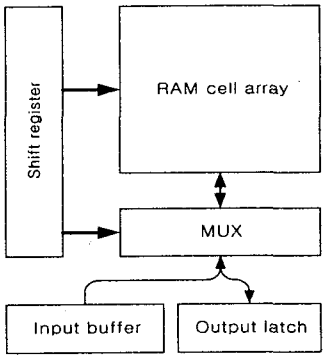

그림 3 Memory module

필터에는 세 개의 메모리 모듈이 사용된다. 두 개는 입력 값을 저장하는 모듈이고, 나머지 하나는 계수 값을 저장하는 모듈이다. 두 가지의 차이점은 word의 크기뿐이다. 입력을 저장하는 모듈은 14bit를, 계수를 저장하는 모듈은 16bit를 사용한다.

표 1은 입력을 저장하는 모듈의 동작을 표로 나타낸 것이다. A 모듈은 첫 번째 클럭에서 외부의 입력 값을 받아 저장하고 두 번째 클럭부터 저장되어 있는 값을 순차적으로 매 클럭에 하나씩 읽어 낸다. 따라서 외부에서 입력을 받아들이는데 한 클럭을 사용하고 모든 값을 읽어내어 계산하는데 64 클럭을 사용하므로 한 주기는 65 클럭이 된다. A 모듈은 제일 오래된 값을 지우고 새로운 값을 저장한 뒤 새로운 값부터 읽어 내기 때문에 표 1에서 보는 것과 같이 최종 클럭과 첫 번째 클럭에서 주소를 변경하지 않는다.

표 1 A, B 모듈의 동작

clock	State	A module	B module
last clock	rising	start read	start read
	high	reading	reading
	falling	data latch	data latch, shift
	low		
1st clock	rising	start write	start write
	high	writing	writing
	falling	write end	write end, shift
	low		
2nd clock	rising	start read	start read
	high	reading	reading
	falling	data latch, shift	data latch, shift
	low		
3rd clock ~ 64th clock : 2nd clock과 동일			

B 모듈은 가장 오래된 입력 값을 지우고 그 자리에 A 모듈에서 가장 오래된 입력 값을 받아 저장한다. B 모듈은 A 모듈과 달리 가장 오래된 값부터 읽어내므로 매 클럭 마다 주소를 변경한다.

계수를 저장하는 모듈은 필터 동작의 초기에 외부에서 계수 값을 받아 저장하고 그 후에는 저장된 값을 읽어 내기만 한다. 따라서 처음 한 주기동안만 쓰기 동작이 행해지고 필터가 다시 리셋(reset)될 때까지 읽기 동작만을 한다. 표 2는 계수를 저장하는 모듈의 동작을 보여주고 있다.

표 2 계수저장 모듈과 MAC의 동작

clock	State	Coefficient module	MAC
last clock	rising	data latch, shift	결과 저장
	high		multiply
	falling	start read	
	low	reading	
1st clock	rising	data latch	결과 저장
	high		multiply
	falling	start read	
	low	reading	
2nd clock	rising	data latch, shift	결과 저장
	high		
	falling	start read	
	low	reading	latch clear
3rd clock	rising	data latch, shift	결과 저장
	high		multiply
	falling	start read	
	low	reading	
4th clock ~ last clock : 3rd clock과 동일			

3-2. MAC(Multiplier accumulator)의 설계

곱셈기(multiplier)는 14bit × 16bit의 연산을 수행하게 된다. Booth 알고리즘(algorithm)을 사용하여 14bits 입력을 booth encoder에 넣고 16bits 계수를 partial product에 넣어 더할 값을 얻는다. 덧셈은 Wallace adder tree를 이용한다. 그림 4는 MAC의 구조를 보여주고 있다[3].

Wallace adder tree는 총 4단이 사용되었는데 위의 두 단은 partial product에서 나온 값들을 일차적으로 더하는 기능을 한다. 이때 Booth encoder에 들어가는 입력 값이 14bit이기 때문에 두 번째 Wallace adder tree는 3:2 compress를 사용하고 있다. 첫 번째와 두 번째 Wallace adder tree에서 더해진 값들은 세 번째 Wallace adder tree에서 더해져 실질적인 곱셈이 끝난다. 하지만 이전 클럭의 결과 값을 다시 더해야 하므로 다시 Wallace adder tree가 사용된다. 이때 사용되는 Wallace adder tree도 3:2 compress를 사용한다. 이렇게 나온 값을 CLA(Carry Lookahead Adder)를 이용하여 최종 계산한다.

표 2에서 MAC의 동작을 볼 수 있다. MAC에는

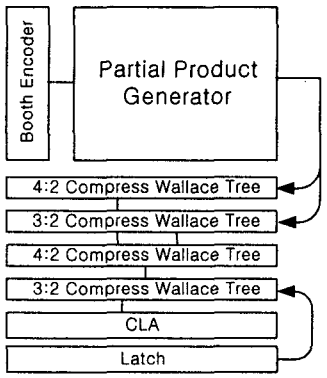

그림 4 MAC의 구조

accumulator에 저장된 값을 지우는 동작과 연산한 결과를 저장하는 동작으로 이루어져 있다.

3-3. 덧셈기(adder)와 입출력회로

덧셈기는 두 개의 입력 값을 곱셈기에 보내기 전에 미리 더해서 곱셈의 횟수를 줄여주는 부분이다. 여기에는 14bit CLA가 사용되었다[4]. 덧셈기는 입력 값이 읽혀진 직후 곱셈이 시작되기 전인 매 클럭의 로우 상태(low state)에서 동작한다.

외부 입출력에는 래치(latch)가 사용된다. 입력 래치는 계수도 받아들여야 하므로 16bit 폭을 가지며 출력은 14bit 폭을 가진다. 입력 래치는 계수를 읽어들일 때에는 매 클럭마다 동작해야 하고, 계수 초기화가 끝나면 첫 번째 클럭에서만 동작해야 한다. 출력 래치는 곱한 값들의 누적 덧셈이 끝나는 두 번째 클럭에서 동작해야 한다.

3-4. 제어기의 설계

제어기는 각 부분이 정해진 시점에서 동작 할 수 있도록 각종 신호를 보내준다. 모든 신호의 기준이 되는 클럭(clock)은 외부에서 들어온 클럭을 ½로 나누어 듀티비(duty rate)가 50%가 되도록 한다. 리셋(reset)신호는 외부에서 들어오는 리셋신호를 내부 클럭으로 동기 시켜 사용한다.

각 모듈이 원하는 클럭에서 동작하도록 하기 위해서 한 주기 중 현재의 위치를 나타내는 신호를 이진 계수기(binary counter)와 3bits shift register를 이용하여 만들어 낸다. 여기서는 last clock, 1st clock, 2nd clock, 3rd clock을 나타내는 신호가 출력되고, 리셋으로 시작할 때 2nd clock의 high 상태에서 시작할 수 있도록 한다.

Ⅳ. Simulation

파이프라인 구조를 이루고 있기 때문에 전체 동작속도

는 각 모듈 중 가장 지연속도(delay)가 큰 것이 전체 동작 속도를 좌우하게 된다. 그림 5는 곱셈기의 simulation 결과이다. 입력이 low에서 high로 변하여 출력이 high에서 low로 바뀌게 된다. 이 때 multiplier의 지연시간은 14ns정도가 된다.

그림 6은 메모리의 읽기 지연시간을 보여주고 있으며

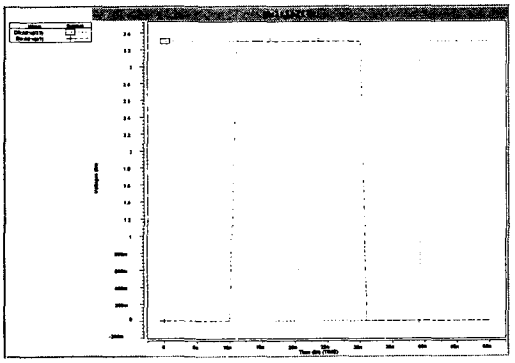

그림 5 Multiplier의 지연시간

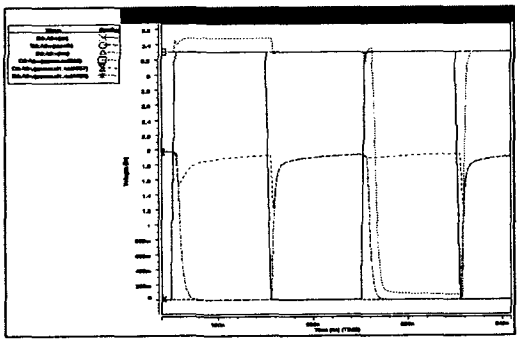

그림 5 메모리 읽기 지연시간

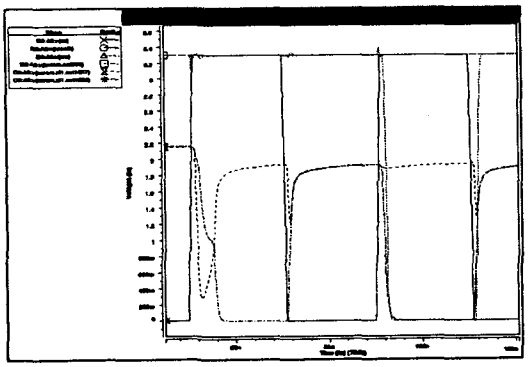

그림 7 메모리 쓰기 지연시간

그림 7은 메모리의 쓰기 지연시간을 보여주고 있다. 읽기 지연시간은 6ns, 쓰기 지연시간은 8ns가 소모되고, bit line을 precharging하기 위해서 약 4ns가 소모된다. 따라서 메모리가 동작하기 위해서는 12ns이상의 시간이 필요하다. Simulation에서 보듯이 MAC의 최대 지연 시간이 가장 길며 그 시간이 14ns이므로 필터의 내부 동작속도를 50MHz로 동작시킬 수 있다.

곱하기 연산의 경우 고정 소수점(fixed point number) 연산 방식을 사용하므로 자리 옮김 동작을 마지막에 행한 뒤 연산결과를 출력하게 된다. 최종 출력은 자리 옮김 동작을 하고 상위 14bit만을 사용하게 된다. 따라서 오차가 발생하게 되며 MAC에 의해 발생하는 오차는 최대 2^{-14}이 된다.

V. 결 론

ITU에서 규격으로 정하는 모뎀에서 사용할 수 있는 128-tap FIR 필터를 설계하였다. 동작 속도를 향상시키기 위해서 파이프라인구조를 사용하였고, 계수의 좌우대칭 성질을 이용하여 곱셈의 횟수를 반으로 줄였다. 계수는 필터의 동작 초기에 외부에서 받아들일 수 있게 했다. Delay tap은 SRAM을 변형한 환형 버퍼를 이용하였다. 최대 동작 속도는 50MHz이며, 이 경우 초당 약 770k개의 sample을 처리 할 수 있다.

참고문헌

[1] ITU-T V.32(International Telecommunication Unit Telecommunication Sector), ITU, 1993.

[2] Masataka Matsui *et al.*, "A 25-ns 1-Mbit CMOS SRAM with Loading-Free Bit Lines", *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 5, pp.733-740, Oct. 1987.

[3] Norio Ohkubo *et al.*, "A 4.4 ns CMOS 54×54-b Multiplier Using Pass-Transistor Multiplexer", *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 251-257, Mar. 1995.

[4] Neil H. E. Weste and Kamran Eshraghian, *Principles of CMOS VLSI Design*, Addison Wesley, 1994.