논문 02-27-10C-14 한국통신학회논문지 '02-10 Vol.27 No.10C

DSSS 수신기에서 동기탐색을 위한 고속 정합필터

정회원 송 명 렬*

A High-Speed Matched Filter for Searching Sychronization in DSSS Receiver

Myong-Lyol Song* *Regular Member*

요 약

본 논문에서는 DSSS (Direct Sequence Spread Spectrum) 수신기에서 초기동기 탐색에 사용될 수 있는 정합필터에 대해서 연구하였다. 하드웨어기술언어 (HDL)로 정합필터를 구현하기 위한 모델이 제시되었다. 제안된 모델은 고속 처리를 위해 병렬처리와 파이프라인 구조를 기반으로 하는데 환형버퍼, 곱셈기, 덧셈기, 코드참조표 등으로 구성되어 있다. 제안된 모델에 대해 성능을 분석하였고 일반적인 DSP (Digital Signal Processor)로 구현할 경우와 비교하였다. 제안된 모델을 FPGA (Field Programmable Gate Array)상에 구현하였고 타이밍 시뮬레이션 결과를 통해서 동작을 검증하였다.

ABSTRACT

In this paper, the operation of matched filter for searching initial synchronization in direct sequence spread spectrum receiver is studied. The implementation model of the matched filter by HDL (Hardware Description Language) is proposed. The model has an architecture based on parallelism and pipeline for fast processing, which includes circular buffer, multiplier, adder, and code look-up table. The performance of the model is analyzed and compared with the implementation by a conventional digital signal processor. It is implemented on a FPGA (Field Programmable Gate Array) and its operation is validated in a timing simulation result.

I. 서 론

최근에 초고속 DSP와 고속이면서 집적도가 높은 FPGA의 등장으로 통신시스템에서 변조와 복조를 디지털 연산으로 실행하는 것이 가능하게 되었다. 따라서 기저대역의 신호뿐만 아니라 나아가 수신된 고주파 대역의 신호를 직접 A-D 변환하여 디지털로 처리하는 소프트웨어 라디오 분야가 활발히 연구되고 있다. 그리고 verilog HDL이나 VHDL 등과 같은 하드웨어기술언어 (Hardware Description Language : HDL)를 지원하는 논리회로 시스템 설계도구의 발달로 인하여 IC 제조회사의 기술과 관계없이 통신 시스템에 필요한 다양한 기능들이 프로그램 언어수준과 같은 HDL로 표현되고 있다. 또한 동작과 신뢰성이 검증된 HDL로 표현된 다양한 기능 모델들은 IP (Intellectual Property)의 형태로 마치 모듈화 된 부품처럼 시스템에 채택되어 시스템 개발시간을 현저히 단축시킬 수 있는 SOC (System-On-Chip) 기술의 기초가 되고 있다.

이러한 기술적인 배경으로 인해 DSSS 통신 시스템에서 데이터를 대역확산 및 고주파 대역으로 변조하는 송신기능과 고주파 대역의 신호의 복조 및 역확산과정으로 구성되는 수신기능의 하드웨어를 DSP로 구현하거나 HDL로 표현하여 FPGA나 ASIC (Application Specific Integrated Circuit)으로 구현하는 것이 가능하다[5][6][7][8]. 이 논문에서는 DSSS 통신 시스템에서 수신기에 수신된 대역확산 신호에 대하여 동기를 맞추기 위한 과정 중 초기동

* 호서대학교 전기정보통신공학부 (mlsong@dogsuri.hoseo.ac.kr)
논문번호 : 020111-0311, 접수일자 : 2002년 3월 11일

기 탐색방법으로서 정합필터방식을 적용하고 이를 하드웨어로 구현할 수 있는 모델에 대해서 제안하고 구현한다.

그런데 정합필터의 출력을 계산하기 위해서 필터에 입력되는 데이터의 저장, 저장된 데이터들을 이용한 곱셈, 덧셈 등과 같은 연산 및 이에 수반되는 메모리에 대한 포인터들의 갱신이 반복적으로 수행된다. 이러한 연산요소를 병렬처리가 가능한 범용의 응용을 위한 구조를 갖는 전형적인 DSP에 담당시키더라도, 하나의 명령어에 메모리 액세스와 포인터 갱신을 수행하거나 복수의 연산장치를 통해서 몇 개의 연산을 동시에 수행하는 부분적인 병렬화 수준이므로 처리속도 개선에는 한계가 있다. 따라서 본 논문에서는 FPGA나 ASIC에 구현할 목적으로 정합필터의 연산요소들을 가급적 동시에 처리할 수 있는 형태로 분해하여 HDL로 표현할 수 있는 모델에 대해서 제안하고 FPGA에 구현한다.

2장에서는 DS방식의 대역확산된 신호에 대해 초기동기를 탐색할 수 있는 정합필터의 원리에 대해서 설명하고 분석한다. 3장에서는 정합필터를 HDL로 표현하기 위한 하드웨어 구조를 제안한다. 4장에서는 제안된 정합필터의 성능을 처리시간의 관점에서 분석한다. 또한 제안된 하드웨어 구조를 FPGA에 구현하기 위해 타이밍 시뮬레이션으로 동작에 대해 검증하고 5장에서 결론을 맺는다.

Ⅱ. 초기동기 획득을 위한 정합필터의 구조 및 분석

1. 정합필터를 이용한 초기동기 획득 시스템

그림 1에 DSSS 통신 시스템의 송신기 구조를 나타냈다. 전송할 데이터 $m(t)$이 +1 또는 -1 이라 하자. 송신기는 반송파를 대역확산 코드에 의해 BPSK방식으로 변조하여 전송하고 대역확산 코드의 대역폭은 기저대역의 데이터의 대역폭에 비해 상당히 크다고 가정한다. 그러면 송신기에서 전송되는 신호는 식 (1)과 같이 표현된다.

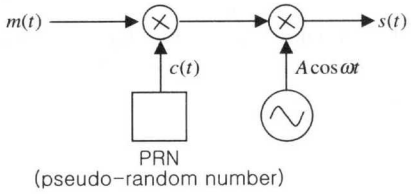


그림 1. DSSS 송신기

$$s(t) = \pm Ac(t)\cos\omega t \tag{1}$$

여기서 $c(t)$는 대역확산 코드를 의미하고 A와 ω는 각각 반송파의 진폭과 각주파수를 의미한다.

한편, 송신기와 수신기의 위치가 고정되어 있다고 가정하면 도플러효과로 인한 반송파 주파수의 변화는 없다. 따라서 신호 $s(t)$가 전송채널을 거쳐서 수신기에 도달되면 잡음, 위상지연, 감쇄가 고려되어야 하므로 수신기에 도달된 신호 $r(t)$는 식 (2)와 같이 표현될 수 있다.

$$r(t) = Pc(t - T_d)\cos(\omega t + \phi) + n(t) \tag{2}$$

여기서 P는 수신된 신호의 전력, $c(t-T_d)$는 대역확산코드 $c(t)$가 전송채널을 통해 전파되는 과정에서 T_d 만큼 지연된 것, ω와 ϕ는 각각 반송파의 주파수와 위상지연, 그리고 $n(t)$는 송신기에서 전송된 신호가 수신기에 도착할 때까지 통신채널에 유입되는 잡음을 의미한다.

본 논문에서는 식 (2)와 같이 표현되는 수신된 신호에 대해 초기동기의 탐색을 위해 일반적으로 사용되는 그림 2와 같은 구조의 정합필터방식 초기동기 탐색기를 고려한다[3][9]. 수신기의 국부발진기에서 발생된 신호 $V_m\cos\omega t$와 $V_m\sin\omega t$를 수신된 신호 $r(t)$에 곱하면 I 채널신호 $I(t)$, Q채널 신호 $Q(t)$를 얻고 이들 각각을 적분기를 통과시키면, 식 (3)과 같이 시간이 지연된 확산코드와 반송파의 위상성분만 남는다.

$$\begin{aligned} \phi_I(t) &= Ac(t - T_d)\cos\phi \\ \phi_Q(t) &= Ac(t - T_d)\sin\phi \end{aligned} \tag{3}$$

여기서 T_c는 확산코드의 1 chip에 해당하는 시간이고 반송파 주기의 정수배로 설정된다. 그리고 A는 $\frac{1}{2}P\cdot\frac{T_c}{2}$이다. 신호 $\phi_I(t)$와 $\phi_Q(t)$ 각각을 임펄스응답이 $h(t)=c(-t)$인 정합필터를 통과시키고 자승 후 서로 더하면 출력 $Y(t)$에는 식 (4)와 같이 반송파의 위상성분이 사라지게 된다.

$$Y(t) = A^2 \cdot \{c(t - T_d) * c(-t)\}^2 \tag{4}$$

따라서 출력 $Y(t)$에서는 반송파의 위상과 관계없는 수신된 확산코드 $c(t-T_d)$와 역확산에 사용되는 수신기 자체의 코드 $c(t)$만의 자기상관함수에 의해 초기동기의 탐색여부가 결정된다.

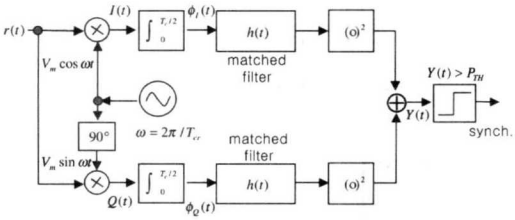


그림 2. 정합필터 방식 초기동기 탐색기

한편, 식 (4)에서 초기동기 탐색기의 출력 $Y(t)$는 수신된 확산코드의 지연이 $T_d = 0$일 경우에 최대 값을 갖는다. 따라서 적분값이 최대치에 가깝게 계산될 경우 두 코드의 위상차이가 적은 상황이므로 적절한 임계치 P_{TH}를 설정하면, $Y(t) > P_{TH}$ 인 순간 초기동기를 탐색했다고 판단하고 그 다음 과정인 동기추적 및 유지 단계로 들어간다.

2. 정합필터

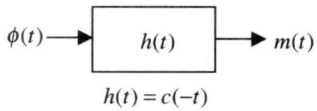


그림 3. 정합필터

그림 3에 초기동기 탐색에 사용되는 정합필터 입출력 및 임펄스응답을 표시하였다. 여기에서 $\phi(t)$와 $m(t)$은 각각 정합필터의 입력과 출력신호이고 $h(t)$는 정합필터의 임펄스응답을 의미한다. 임펄스 응답을 대역확산코드의 항으로 표현하면 정합필터의 출력은 식 (5)와 같이 컨벌루션 (convolution)으로 표현된다.

$$m(t) = \phi(t) * h(t) = \phi(t) * c(-t) \tag{5}$$

이것을 컨벌루션의 정의식으로 표현하면 식 (6)과 같이 표현된다.

$$m(t) = \int_{t_0}^{t} \phi(\tau) c(\tau - t) d\tau \tag{6}$$

식 (6)에서 $t = 0$를 대입하면,

$$m(0) = \int_{t_0}^{0} \phi(\tau) c(\tau) d\tau \tag{7}$$

가 된다. 이 식에서 $c(\tau)$는 $c(\tau - 0)$과 같은 표현이므로 코드의 위상지연이 영인 대역확산코드를 의미한다. 그런데 정합필터의 입력신호 $\phi(t)$에는 송신기의 대역확산코드의 지연된 성분 $c(t - T_d)$가 포함되어 있으므로 식 (8)과 같이 정리된다.

$$m(0) = K \int_{t_0}^{0} c(\tau - T_d) c(\tau) d\tau \tag{8}$$

여기서 K는 적분변수와 무관한 반송파의 위상성분이나 진폭과 같은 성분을 나타낸다.

Ⅲ. 정합필터 방식 초기동기 탐색 하드웨어 모델

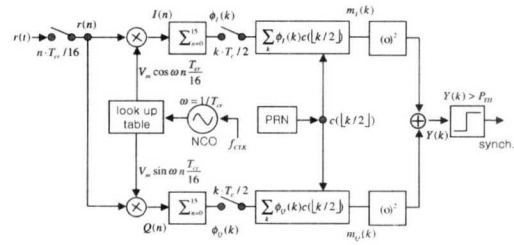


그림 4. 정합필터 방식 초기동기 탐색기의 하드웨어 구현 모델

정합필터 방식의 초기동기 탐색기능을 하드웨어로 구현하기 위한 모델을 그림 4에 나타냈다. 이 모델은 수치제어발진기 (NCO), 반송파의 위상천이를 위한 참조표 (look-up table), 곱셈기, 누산기, 정합필터, 자승기, 덧셈기 및 비교기로 구성되어 있다. 이 논문에서는 초기동기 탐색에 사용되는 정합필터의 구조에 초점을 맞추므로 다른 부분에 대해서는 설명을 생략한다.

1. 정합필터에서 연산

정합필터 방식 초기동기 탐색기에서 정합필터는 그림 4에서 I채널과 Q채널 상에서 각각 신호 $\phi_I(k)$와 $m_I(k)$ 사이에 그리고 $\phi_Q(k)$와 $m_Q(k)$ 사이에 위치한다. 정합필터의 하드웨어 모델을 표현하기 위해 2절의 식 (7)을 다시 보면,

$$m(0) = \int_{t_0}^{0} \phi(\tau) c(\tau) d\tau$$

이다. 여기서 $m(0)$는 시각 $t = 0$에서 정합필터의 출력이다. 이 식의 계산은 구간 $(t_0, 0)$동안 정합필터에 입력된 신호 $\phi(t)$와 수신기에서 자체적으로 발생되는 대역확산코드 $c(t)$의 곱을 적분하는 것이다. 그런데 대역확산코드 $c(t)$는 수신기 자체적으로 발생되므로 수신기에서 코드의 위상을 임의로 지정

할 수 있다. 따라서 $m(0)$는 위상이 0으로 설정된 수신기의 확산코드 $c(t)$를 기준으로 할 때 $\phi(t)$에 포함되어 있는 지연되어 수신된 송신기 확산코드의 지연 T_d의 정도를 나타내는 수단이 될 수 있다.

이제 식 (7)을 디지털 연산으로 수행하기 위해서는 정합필터에 입력되는 신호 $\phi(t)$를 일정한 속도로 표본화 해야한다. T_s를 정합필터에 입력되는 신호에 대한 표본화 주기라 하고, ϕ_k를 시각 $t=kT_s$에서 정합필터의 입력신호 $\phi(t)$의 표본화된 신호라 한다. 그리고 $m_k(0)$를 시각 $t=kT_s$에서 계산된 정합필터의 출력이라 하자. $m_k(0)$를 계산하기 위해서는 ϕ_k를 포함하여 과거에 입력되었던 신호들이 필요하다. 그러나 과거의 신호를 저장할 메모리의 용량과 출력 계산을 위한 연산량 증가로 인해 개수가 W로 제한된 표본 $\{\phi_{k-W+1}, \phi_{k-W+2}, \cdots, \phi_k\}$만 $m_k(0)$의 계산에 이용한다. 그리고 기준위상이 되는 수신기의 대역확산코드에 대한 표본값은 $\{c_{-W+1}, c_{-W+2}, \cdots, c_{-1}, c_0\}$을 적용한다. 여기서 $c_{-i}=c(-iT_s)$ 이다. 그러면 식 (7)에 대응하는 이산신호입력에 대한 정합필터의 출력 $m_k(0)$는 식 (9)와 같이 디지털 연산으로 처리할 수 있게 되고 Φ_k와 C의 내적이 된다.

$$m_k(0) = \sum_{i=0}^{W-1} \phi_{k-W+1+i} \cdot c_{-W+1+i} = \Phi_k \cdot C \tag{9}$$

여기서 $\Phi_k=\{\phi_{k-W+1}, \phi_{k-W+2}, \cdots, \phi_k\}$ 이고 $C=\{c_{-W+1}, c_{-W+2}, \cdots, c_{-1}, c_0\}$ 이다.

그림 5에 식 (9)의 계산과정을 나타냈다. 그림 5(a)는 ϕ_k가 정합필터에 입력되는 순간, 정합필터의 출력 $m_k(0)$은 수신기 대역확산코드 C와 정합필터의 입력신호들 Φ_k의 내적과 같은 방법으로 계산됨을 나타낸 것이다. 그리고 그림 5(b)는 ϕ_{k+1}에 대한 출력 $m_{k+1}(0)$는 $\Phi_{k+1}\cdot C$로 계산되는 것을 의미한다. 따라서 새로운 데이터가 입력될 때마다 이와 같은 과정을 반복하면, 디지털 연산에 의한 정합필터의 출력은 식 (10)에 표현된 원소와 같은 값을 차례로 갖게 된다.

$$\{\cdots, \Phi_{k-1}\cdot C, \Phi_k \cdot C, \Phi_{k+1}\cdot C, \cdots\} \tag{10}$$

한편, 식 (9)를 보면 정합필터에 1개의 표본이 입력될 때마다 그에 대응하는 출력값을 계산하기 위해

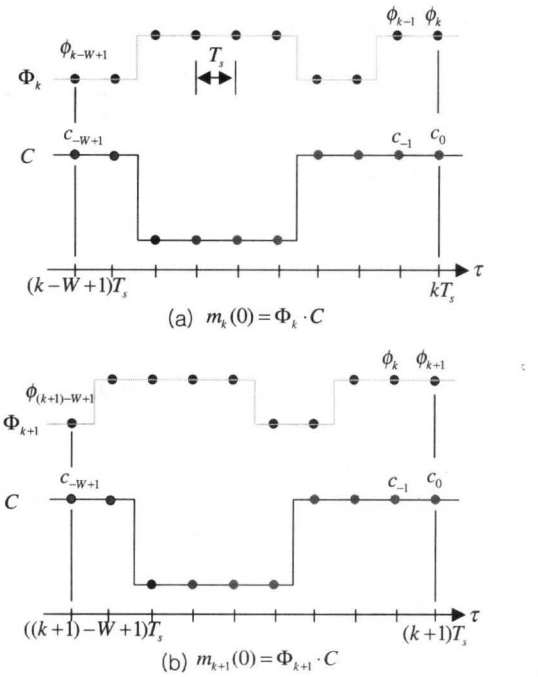


그림 5. 정합필터에서 디지털 연산 과정

곱셈 W회, 덧셈 $W-1$회의 연산이 수행되어야 한다. 그리고 정합필터에 입력되었던 과거의 데이터를 저장해 둘 메모리와 연산시 저장되었던 표본값이나 확산코드값을 읽기 위한 포인터 (pointer) 및 포인터 값 제어를 위한 기능이 필요하다.

2. 정합필터의 HDL 구현을 위한 모델

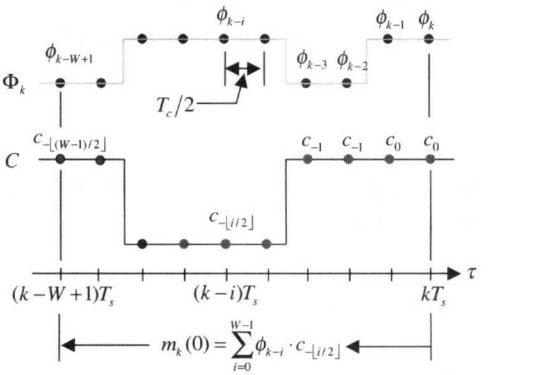


그림 6. 제안된 정합필터의 연산 방법

초기동기 탐색기는 송신기에서 전송한 파형에 포함되어 있는 대역확산코드의 위상이 수신기 자체의 대역확산코드에 대해 $T_c/2$이내에 위치하는 가를 판

정해야 한다. 따라서 정합필터에 입력되는 신호에 대한 표본화 주기 T_s는 식 (11)과 같은 조건을 만족해야 한다.

$$T_s \leq T_c/2 \tag{11}$$

한편, 식 (9)를 풀어서 쓴 후, 덧셈의 순서를 바꾸면 식 (12)와 같이 정리된다.

$$m_k(0) = \sum_{i=0}^{W-1} \phi_{k-i} \cdot c_{-i} \tag{12}$$

이제, 정합필터의 표본화 주기를 $T_s = T_c/2$라 가정하면, 확산코드 1 chip의 시간 동안 2개의 표본과 확산코드값이 필요하므로 정합필터의 출력은 식 (13)과 같이 표현될 수 있다.

$$m_k(0) = \sum_{i=0}^{W-1} \phi_{k-i} \cdot c_{-\lfloor i/2 \rfloor} \tag{13}$$

여기서 $\lfloor x \rfloor$는 x 보다 크지 않은 최대 정수를 의미한다. 그림 6에 식 (13)에 대한 계산과정을 나타냈다. 그림에 나타낸 바와 같이 정합필터의 출력 $m_k(0)$는 가장 최근에 입력된 신호 ϕ_k에서 시작하여 가장 오래 된 신호 ϕ_{k-W+1}의 순서로 계산된다.

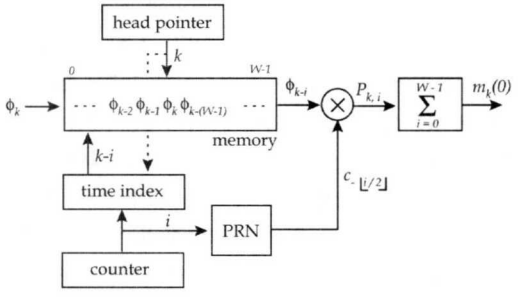


그림 7. 제안된 정합필터의 하드웨어 구현 모델

이제 식 (13)을 하드웨어로 계산하기 위해서는 정합필터에 입력되는 신호 ϕ_{k-i}를 저장할 메모리 (RAM), 대역확산 코드 $c_{-\lfloor i/2 \rfloor}$를 저장해 둘 메모리 (ROM), 저장되어 있는 ϕ_{k-i}와 $c_{-\lfloor i/2 \rfloor}$를 액세스 할 수 있는 포인터, 곱셈기, 덧셈기, 연산횟수를 제한하기 위한 논리회로 등과 같은 성분들이 필요하다. 그림 7에 식 (13)의 계산과정을 HDL로 표현하기 위한 일반적인 모델을 나타냈다.

그림에서 ϕ_k는 현재시각 ($t=0$)에서 정합필터에 입력되는 이산신호이고 메모리에 저장된 후 일정시간 즉, $W \cdot T_s$ 시간동안 머물면서 필터의 출력계산에 사용된 후 폐기된다. ϕ_{k-1}, ϕ_{i-2}, $\cdots$, $\phi_{k-(W-1)}$은 현재시각 이전에 필터에 입력되었던 이산신호들이고 출력의 계산을 위해 저장되었던 것들이다. 한편, 정합필터에 입력되는 이산신호들은 환형버퍼 구조를 갖는 메모리에 저장되어야 한다. 만일 일반적인 구조의 메모리를 사용하면, 새로운 출력을 계산할 때마다 저장된 입력신호들에 대해 $W-1$회 반복해서 복사 (메모리 읽기, 쓰기 동작) 하여 위치를 이동시켜야 하므로 연산에 소요되는 데이터 량에 비례하여 연산시간이 증가한다. 따라서 환형버퍼의 구조를 위해서 새로 입력되는 데이터를 저장할 위치를 나타내는 포인터가 필요하다. 그러므로 입력신호 ϕ_k는 그림에서 'head pointer'의 값이 지시하는 메모리 주소 k의 위치에 저장된다. 또한 'head pointer'는 상향 카운터가 사용되기 때문에 새로운 입력 데이터를 메모리에 기록할 때마다 다음에 입력되는 데이터의 저장을 위해 1씩 증가하여 $\{0,1,\cdots,W-1\}$의 값들을 차례로 순환한다. 한편, 포인터 'time index'의 값 $k-i$는 정합필터의 출력을 계산하기 위해 메모리에 저장되어 있던 데이터들 중 ϕ_{k-i}를 읽어내기 위한 것으로서 새로운 데이터 ϕ_k가 저장될 때마다 'header pointer'의 값 k로 초기화되고 출력계산을 위해 메모리로부터 데이터를 읽을 때마다 1씩 감소된다. 그리고 'counter'의 값 i는 대역확산코드의 참조를 위한 포인터 기능 및 출력의 계산을 위해 진행된 연산횟수를 나타낸다. 이 값은 ϕ_k가 입력될 때마다 0으로 설정되고 메모리로부터 ϕ_{k-i}를 읽을 때마다 1씩 증가한다.

확산코드는 PRN에서 발생되는데, 일반적으로 exclusive-OR 게이트와 플립플롭들로 구성되는 LFSR (Linear Feedback Shift Register) 구조로 구현된다[1][2]. 그러나 본 논문에서 제안된 정합필터에서는 대역확산코드의 한 주기에 해당하는 코드열 전체가 필요하지 않고 일부분 만으로도 초기동기탐색이 가능하다. 따라서 쉬프트 레지스터 구조 보다는 탐색에 필요한 코드열이 저장되어 있는 표 1과 같은 코드참조표 (look-up table)를 이용한다.

표 1. 코드참조표

i	0	1	2	3	4	⋯
$c_{(0-\lfloor i/2 \rfloor)}$	c_0	c_0	c_{-1}	c_{-1}	c_{-2}	⋯

곱셈기는 ϕ_{k-i}와 $c_{(0-\lfloor i/2 \rfloor)}$의 곱기능을 수행하

는데, $c_{(0-\lfloor i/2 \rfloor)}$가 -1 또는 1이므로 결과 P_i는 $-\phi_{k-i}$와 ϕ_{k-i} 중 어느 한 값을 갖게 된다. 그러므로 멀티플렉서와 2의 보수 논리회로를 이용하여 구현 가능하다. 누산기는 각 i에 대해 ϕ_{k-i}와 $c_{(0-\lfloor i/2 \rfloor)}$의 곱의 결과 P_i를 더하는 기능을 수행한다.

Ⅳ. 성능분석 및 실험결과 고찰

1. 성능분석

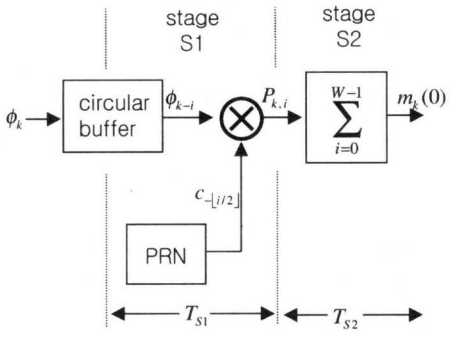
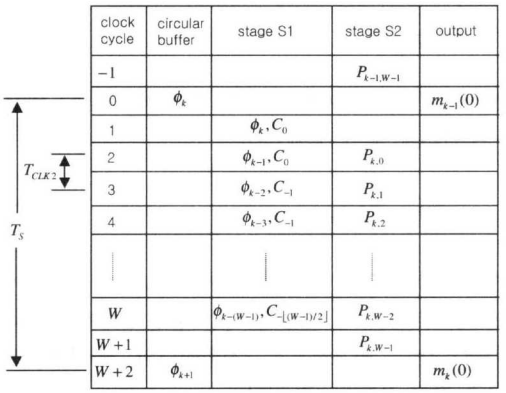


그림 8. 정합필터에서 데이터의 이동단계

이 절에서는 정합필터에 대해 제안된 하드웨어에서 데이터가 이동되는 경로를 분석한다. 그림 8에 나타낸 바와 같이 표본화된 이산신호는 먼저 환형버퍼(circular buffer)에 저장된다. 그리고 환형버퍼에 저장되었던 이산신호와 코드참조표로부터 읽어낸 코드는 곱셈기에 전달되어 서로 곱해진다. 이 곱셈결과는 다시 누산기로 이동되어 누산기의 현재값에 더해진다. 한편, 누산기에서 더해지는 것과 동시에 환형버퍼와 코드참조표로부터 다음 차례의 데이터가 읽혀져 곱셈기에 전달되어 곱셈이 수행된다. 따라서 파이프라인 방식과 같이 곱셈과 덧셈이 동시에 수행된다.

한편, 그림 8에서 단계 S1과 단계 S2는 각각 곱셈기와 누산기를 포함하고 있고 T_{S1}, T_{S2}는 데이터가 이들을 통과하는데 소요되는 시간이다. 그리고 곱셈기와 누산기를 구성하는 게이트 수와 구조가 다르기 때문에 하드웨어로 연산하는데 소요되는 시간 T_{S1}, T_{S2}도 서로 다르게 된다. 그러나 제안된 정합필터를 동기식 회로로 적절히 설계하면 곱셈기와 누산기의 연산을 각각 1 클럭주기 이내의 시간에 수행시킬 수 있다. 따라서 이 절에서는 T_{CLK2}를 정합필터 하드웨어에 공급되는 클럭의 주기라 할 때, $T_{S1}=T_{S2}=T_{CLK2}$로 가정하고 정합필터의 성능을 분석한다.



clock cycle	circular buffer	stage S1	stage S2	output
−1			$P_{k-1,W-1}$	
0	ϕ_k			$m_{k-1}(0)$
1		ϕ_k, C_0		
2		ϕ_{k-1}, C_0	$P_{k,0}$	
3		ϕ_{k-2}, C_{-1}	$P_{k,1}$	
4		ϕ_{k-3}, C_{-1}	$P_{k,2}$	
⋮		⋮	⋮	
W		$\phi_{k-(W-1)}, C_{-\lfloor (W-1)/2 \rfloor}$	$P_{k,W-2}$	
$W+1$			$P_{k,W-1}$	
$W+2$	ϕ_{k+1}			$m_k(0)$

그림 9. 사이클마다 연산요소에서 처리되는 데이터

이제 정합필터의 출력계산을 위해 환형버퍼에 저장된 W개의 이산신호들이 이러한 방식으로 연산요소들을 경유하는 과정을 그림 9에 나타냈다. 그림에서 각 항들은 제안된 하드웨어의 각 단계에 나타나는 데이터를 의미한다. 설명을 위해, 표본화된 신호 ϕ_k가 환형버퍼의 입력포트에 나타나고 있는 시점(클럭사이클 0)을 기준시점으로 정하고 이 데이터는 클럭사이클 1이 되기 전에 환형버퍼에 저장된다. 항 ϕ_k, c_0는 환형버퍼에 기억되어 있던 ϕ_k와 코드참조표에 기록되어 있던 c_0가 클럭사이클 1에서 읽혀져서 단계 S1 (곱셈기)의 입력에 나타나는 것을 의미한다. ϕ_k와 c_0의 곱 $P_{k,0}$는 클럭사이클 2에서 단계 S2 (누산기)의 입력포트에 나타난다. 그리고 이와 동시에 ϕ_{k-1}, c_0가 읽혀져서 단계 S1의 입력포트에 나타난다. 이와 같은 동작을 반복하면 항 $\phi_{k-(W-1)}, c_{-\lfloor (W-1)/2 \rfloor}$는 클럭사이클 W에서 단계 S1에 입력되고 이들의 곱셈결과 $P_{k,W-1}$은 클럭사이클 $W+1$에 단계 S2에 입력된다. 따라서 입력 데이터 ϕ_k에 대한 정합필터의 출력 $m_k(0)$은 클럭사이클 $W+2$부터 출력에 나타나게 된다. 그러므로 환형버퍼에 어떤 입력데이터가 저장된 후, 정합필터의 출력계산에 있어서 순수 연산에만 소요되는 시간 T_{PROC}은 식 (14)와 같이 $W+1$개의 클럭사이클이 필요하다.

$$T_{PROC}=(W+1)T_{CLK2} \tag{14}$$

그리고 정합필터에 입력되는 데이터가 환형버퍼에

저장되는 시간을 포함하면, 입력신호의 표본화 주기 T_S와 정합필터 클럭 T_{CLK2}의 관계는 식 (15)와 같이 표현된다.

$$T_S = T_{CLK2} + T_{PROC} = (W+2)\,T_{CLK2} \qquad (15)$$

한편, 제안된 정합필터 알고리즘을 일반적인 DSP로 구현할 경우에는 메모리 읽기 2회 (환형버퍼 1회, 코드참조표 1회), 포인터의 갱신 2회 (time index 1회, counter 1회), 곱 1회, 합 1회, 반복적인 루프를 종료할 것인가에 대한 흐름제어 1회 등과 같은 연산요소들이 순차적으로 반복 수행된다. 그리고 각 연산 장치로 데이터를 이동시키기 위한 명령도 고려해야 한다. 또한 메모리 읽기와 포인터 갱신은 하나의 명령에 동시에 수행할 수 있는 구조이면서 모든 명령을 1 클럭주기에 수행하는 DSP로 처리한다고 가정하더라도, 정합필터에 표본화된 신호가 입력된 후 출력을 얻기까지의 처리시간은 환형버퍼에 저장된 데이터 1개당 최소 5회의 클럭주기가 필요하다. 그러므로 일반적인 DSP로 정합필터의 출력을 얻기 위해 필요한 시간 T_{DSP}는 식 (16)과 같이 표현할 수 있다.

$$T_{DSP} > 5W\,T_{CLK2} \qquad (16)$$

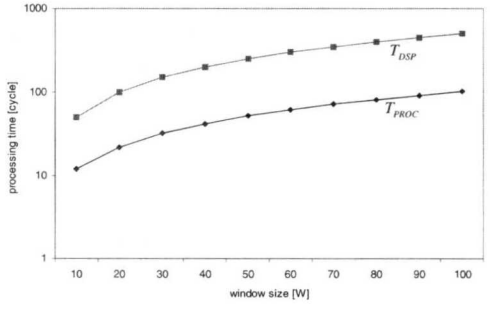


그림 10. 정합필터의 출력 계산시간 비교

제안된 정합필터의 성능과 비교를 위해 이 식에서 W와 T_{CLK2}는 각각 표본수와 클럭주기라 한다. 그림 10에 정합필터의 출력계산시간을 제안된 하드웨어 구조로 처리할 경우와 일반적인 DSP로 처리할 경우를 비교하여 나타냈다. 그림에서 보는 바와 같이 정합필터 출력 계산에 사용되는 표본수 W가 클수록 처리시간의 차이가 점점 커진다. 그러므로 입력되는 신호의 표본화율이 높아야 하는 경우에 제안된 하드웨어 구조를 적용할 수 있다.

2. 실험 및 결과 고찰

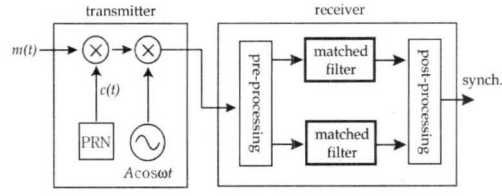


그림 11. 제안된 정합필터의 동작실험을 위한 환경

정합필터의 하드웨어 구현을 위해 앞에서 제안된 모델에서는 필터의 출력 계산에 사용되는 표본화된 신호의 갯수를 나타내는 윈도우 크기, 데이터 경로의 넓이 등과 같은 요소를 일반적으로 표현하였다. 이 절에서는 제안된 모델에 대해 제한된 조건들을 적용하여 설계 및 구현된 하드웨어가 앞에서 분석한 바와 같이 올바르게 동작하는지를 검증한다.

그림 11에 제안된 정합필터의 동작을 검증하기 위한 시스템의 구조를 나타냈다. 송신기에서 전송되는 데이터를 대역확산시키기 위해 $1+X^1+X^6$과 같은 다항식으로 표현되는 코드발생기의 구조를 이용한다[2][4]. 그러므로 다항식의 차수에 의해 코드의 최대길이는 2^6-1이 된다. 그리고 이 코드 발생기는 주기가 1280 ns인 클럭에 의해 구동된다고 가정한다.

한편, 수신기에 도달된 신호는 전처리부에서 I채널 Q채널 성분으로 분해되어 반송파 성분이 제거된 후 제안된 정합필터에 입력된다. 후처리부에서는 정합필터의 출력을 이용하여 동기탐색여부를 결정하는 일을 담당한다. 제안된 정합필터에 입력되는 신호 $\phi_I(t)$, $\phi_Q(t)$는 11 비트 정수 (signed integer) 형식으로 표현되는데, 이 신호들에는 대역확산코드 성분이 포함되어 있으므로, 초기동기 탐색을 위해 정합필터에 입력되는 신호는 확산코드 비트의 절반인 640 ns마다 표본화된다. 또한 제안된 정합필터의 출력계산을 위해 사용되는 데이터에 대한 윈도우 크기는 $W=14$로 정한다. 그러므로 식 (15)에 의해, 제안된 정합필터를 동작시키기 위한 클럭의 주기는 $T_{CLK2}=40$ns으로 설정한다.

그림 7에 나타낸 정합필터의 하드웨어 모델을 Verilog HDL을 이용하여 표현하고 Synopsys사의 FPGA Express 프로그램으로 논리회로를 합성하였다. 합성된 결과를 Altera사의 FPGA디바이스 ACEX(EP1K100)에 맞추었다[1]. 그림 12에 Altera사의 FPGA 상에 구현된 제안된 정합필터에 대한

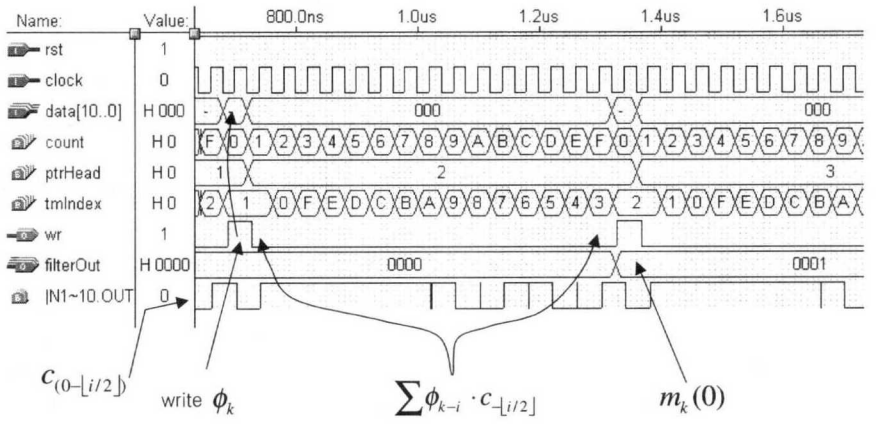


그림 12. 논리합성된 정합필터에 대한 타이밍 시뮬레이션

타이밍 시뮬레이션 결과를 나타냈다.

신호 'wr'는 메모리의 동작을 제어하는 신호로서 표본화된 신호를 저장하거나 읽기 위해 사용된다. 이 신호의 값이 1인 경우 메모리에 저장을 의미하고 0인 경우에는 메모리로부터 읽는 것을 의미한다. 그림의 파형에서는 신호 'count'의 값이 0이 될 때마다 신호 'wr'가 1로 활성화되고 이 때 신호 'data[10..0]'에 나타난 값은 정합필터에 입력되어 메모리에 저장되는 표본화된 신호 ϕ_k를 의미한다. 그리고 이 값은 신호 'ptrHead'가 지시하는 주소의 메모리에 저장된다. 한편, 신호 'count'의 값이 0이 아닌 다른 경우에는 신호 'wr'값이 0이므로 신호 'tmIndex'가 지시하는 주소의 메모리에 저장되었던 데이터들이 읽혀져 신호 'data[10..0]'에 나타나는데 이것은 곱셈기에 전달될 신호 ϕ_{k-i}를 의미한다.

그림 12에서 count = 0일 때, 정합필터의 입력데이터 ϕ_k가 메모리에 기록된다. 메모리의 데이터 포트 파형을 행 data[10..0]에 나타냈다. 신호 'count' 값이 1에서 E까지는 과거에 입력되어 메모리에 저장되어 있던 값과 확산코드의 곱과 합을 계산하는 과정이다. 그리고 신호 'count' 값이 F인 동안 정합필터의 계산결과 $m_k(0)$가 출력신호 'filterOut'에 전달되어 나타나고 이 값은 새로운 계산결과로 갱신될 때까지 유지된다. 따라서 타이밍 시뮬레이션 결과는 제안된 정합필터를 HDL로 구현한 것이 앞 절에서 분석한 바와 같이 동작되는 것을 보여주고 있다.

한편 타이밍 시뮬레이션 파형에는 나타나지 않지만 하드웨어로 구현할 때 FPGA의 속도에 제한이 있으므로, 제안된 정합필터에 대해서는 윈도우 크기를 W라 할 때, 식 (17)과 같은 조건을 만족하도록 설계해야 한다.

$$t_m + W(t_m + t_p + t_s) + t_r < T_c/2 \tag{17}$$

여기서 t_m, t_p, t_s, t_r는 각각 메모리 액세스 시간, 곱셈기, 누산기에서 지연시간, 레지스터 전달시간을 의미한다. 또한 정합필터에 입력되는 데이터의 해상도를 높여야 하는 경우에는 처리속도의 한계 및 구현되는 회로의 크기에 관한 문제를 더 고려해야 한다. 특히 처리속도에 관해서는, 해상도를 높이기 위해서 데이터 경로의 크기를 증가시켜야 한다. 그런데 데이터 경로의 크기가 커지면 데이터를 구성하는 비트들에 해당되는 신호가 모두 안정된 상태에 이르는데 시간이 더 걸리므로 처리시간에 대한 여유는 그만큼 줄어들게 된다. 따라서 서로 인접한 기능블럭을 연결하는 데이터 경로의 길이를 가급적 짧고 균일하게 될 수 있도록 설계해야 한다. 그리고 제안된 정합필터에서 누산기에서는 레지스터에 피드백 (feedback) 형태의 경로가 존재하므로 누산기에 입력된 데이터가 반영되어 누산기의 출력에서 안정될 때까지 가장 많은 시간이 필요하다. 따라서 제안된 정합필터의 처리속도를 더 높일 경우에 병목으로서 작용할 수도 있다.

V. 결 론

본 논문에서는 정합필터방식 초기동기 탐색기에 대해서 분석하였다. 정합필터의 동작을 디지털 연산으로 처리하기 위한 식을 정리하였다. 정합필터에 구현에 포함되는 디지털 연산을 HDL로 구현할 수

있는 모델을 제시하였다. 그리고 제안된 정합필터 모델을 verilog HDL로 표현하고 논리회로로 합성하여 FPGA에 맞춘 후 타이밍 시뮬레이션으로 구현된 정합필터가 올바르게 동작하는 것을 확인하였다.

제안된 하드웨어 모델은, 정합필터의 출력을 얻는데 필요한 데이터의 처리시간을 단축하기 위해, 파이프라인과 병렬처리 구조를 토대로 설계된 것으로서 전형적인 DSP를 이용하여 정합필터를 구현하는 경우에 비해서 최소 5배의 처리시간 단축 효과를 얻을 수 있다. 따라서 더 높은 속도를 갖는 대역확산코드의 통기탐색에 제안된 구조에 적용할 수 있다. 또한 제안된 모델은 대역확산코드 발생기의 구현에 참조표 방식을 이용하므로 참조표의 내용만 수정하면 대역확산코드의 변경이 가능한 구조이다.

그러므로 향후에 시스템 IC 설계시 제안된 구조가 적용될 수 있도록, 본 논문에서 제안된 모델을 기초로 하여 정합필터의 출력 계산에 필요한 데이터 윈도우의 크기, 표본화율, 데이터 경로의 크기, 코드 참조표 등과 같은 요소를 임의로 지정할 수 있는 HDL 구현모델에 대한 연구와 구현모델의 신뢰성을 확보하는 과정이 필요하다.

참 고 문 헌

[1] Altera, ACEX 1K Programmable Logic Device Family Data Sheet, Ver. 3.3, Sep. 2001.

[2] Robert C. Dixon, *Spread Spectrum Systems with Commercial Applications*, 3rd Edition, John Wiley & Sons, 1994.

[3] Vladan M. Jovanovic and Elvino S. Sousa, "Analysis of Non-Coherent Correlation in DS/BPSK Spread Spectrum Acquisition", *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, Part: 1, pp. 565-573, Feb. 1995.

[4] Roger L. Peterson, Rodger E. Ziemer, and David E. Borth, *Introduction to Spread Spectrum Communications*, Prentice-Hall, 1995.

[5] 김건, 조중휘, "직접확산통신을 위한 기적 대역 MODEM의 VLSI 구현", *전자공학회 논문지*, 제 33권, C편, 제 8호, pp. 541-547, 1997.

[6] 김진천, 박홍준, 임형수, 전경훈, "대역 제한된 직접 시퀀스 CDMA 확산 대역 신호를 위한 전 디지털 부호 획득 및 추적 루우프의 FPGA 구현", *전자공학회 논문지*, 제 33권, A편, 제 5호, pp. 893-899, 1996.

[7] 송명렬, "DSSS 수신기에서 동기탐색을 위한 정합필터의 HDL 구현 모델", *한국통신학회 추계종합학술대회 논문집*, Vol. 24, [17-15], 2001.

[8] 심복태, 박종현, 이홍식, 김제우, 김관옥, "Spread Spectrum 방식을 이용한 무선 LAN MODEM의 구현", *전자공학회 논문지*, 제 32권, A편, 제 1호, pp. 1-13, 1995.

[9] 이정훈, 이충웅, "비동기식 디지털 상관기를 이용한 직접부호계열 확산신호의 초기동기에 관한 연구", *전자공학회 논문지*, 제 24권, 제 1호, pp. 1-9, 1987.

송 명 렬(Myong-Lyol Song) 정회원

1985년 2월 : 연세대학교 전자공학과 학사
1988년 2월 : 연세대학교 전자공학과 석사
1996년 2월 : 연세대학교 전자공학과 박사

1984년 12월~1986년 2월 : 삼성전자 연구원
1988년 12월~1991년 7월 : 육군사관학교 전자과 전임강사
1996년 3월~현재 : 호서대학교 전기정보통신공학부 조교수

<주관심 분야> 컴퓨터통신, 통신시스템, 디지털시스템 설계, 임베디드 시스템