논문 2005-42TC-8-2

통신용 DSP를 위한 비트 조작 연산 가속기의 설계

(Design Of Bit Manipulation Accelerator for Communication DSP)

정 석 현*, 선우 명 훈**

(Sug H. Jeong and Myung H. Sunwoo)

요 약

본 논문은 스크램블링(Scrambling), 길쌈부호화(Convolutional Encoding), 펑처링(Puncturing), 인터리빙(Interleaving) 등과 같은 연산에 공통적으로 필요한 비트 조작(Bit Manipulation)을 효율적으로 지원하기 위한 비트 조작 연산 가속기를 제안한다. 기존의 DSP는 곱셈 및 가산 연산을 기본으로 연산기가 구성되어 있으며 워드 단위로 동작을 함으로 비트 조작 연산의 경우 비효율적인 연산을 수행할 수밖에 없다. 그러나 제안한 가속기는 비트 조작 연산을 다수의 데이터에 대해 병렬 쉬프트와 XOR 연산, 비트 추출 및 삽입 연산을 효율적으로 수행할 수 있다. 제안한 가속기는 VHDL로 구현 하여 삼성 0.18㎛ 표준 셀 라이브러리를 이용하여 합성하였으며 가속기의 게이트 수는 1,700개에 불과하다. 제안한 가속기를 통해 스크램블링, 길쌈부호화, 인터리빙을 수행시 기존의 DSP에 비해 40~80%의 연산 사이클의 절감이 가능하였다.

Abstract

This paper proposes a bit manipulation accelerator (BMA) having application specific instructions, which efficiently supports scrambling, convolutional encoding, puncturing, and interleaving. Conventional DSPs cannot effectively perform bit manipulation functions since they have multiply accumulate (MAC) oriented data paths and word-based functions. However, the proposed accelerator can efficiently process bit manipulation functions using parallel shift and Exclusive-OR (XOR) operations and bit insertion/extraction operations on multiple data. The proposed BMA has been modeled by VHDL and synthesized using the SEC 0.18㎛ standard cell library and the gate count of the BMA is only about 1,700 gates. Performance comparisons show that the number of clock cycles can be reduced about 40% ~ 80% for scrambling, convolutional encoding and interleaving compared with existing DSPs.

Keywords: bit manipulation, accelerator, wireless communication, ASIP, SoC design

I. 서 론

디지털 통신 기술이 급속도로 발전함에 따라 xDSL(Digital Subscriber Line), WLAN(Wireless Local Area Network), DAB(Digital Audio Broadcast), DVB (Digital Video Broadcast), IMT-2000 (International Mobile Telecommunications 2000) 등과 같은 다양한 통신 시스템들이 개발되었다. 이러한 통신 시스템들은 높은 연산 능력을 필요로 하는 반면 전력 소모는 적어야 한다. 따라서 ASIC(Application Specific Integrated Circuit)이 이러한 시스템의 구현에 광범위하게 사용되어지고 있다.

디지털 통신 시스템들은 그 표준은 다양하나 일반적으로 그림 1과 같은 유사한 기능 블록들을 가지고 있다.[1] 일반적으로 디지털 통신 시스템의 데이터 송신 과정은 다음을 거친다. 전송하고자 하는 디지털 원천 정보(source information)를 데이터 부호화기(source encoder)를 통해 SNR(Signal to Noise Ratio)을 유지하면서 압축을 한다. 부호화기를 통해 압축된 데이터는 채널 부호화기(channel encoder)를 통해 채널 특성에 적합하도록 변화된다. 그 다음으로 다중 접속(multiple access) 블록을 통해 여러 사용자들이 사용을 할 수 있도록 하며 마지막으로 변조기(modulator)를 통해 데이터를 송신 채널의 특성에 맞는 반송파(carrier)에 실어서 전송하게 된다. 데이터의 수신은 송신의 과정과 반대되는 과정을 거쳐 원천 정보를 획득할 수 있다. 그러나 통신 표준에 상관없이 이러한 블록들의 기능은 유사하나 블록의 세부 동작은 통신 표준에 따라 조금씩 차이가 있다.

* 정회원, 대우전자
(Daewoo Electronics Corp.)
** 정회원, 아주대학교
(School of Electrical and Computer Eng., Ajou Univ.)
접수일자: 2004년11월19일, 수정완료일: 2005년8월13일

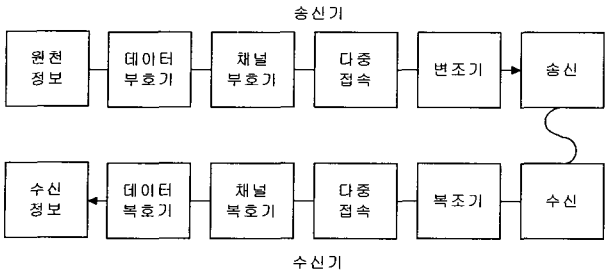

그림 1. 디지털 통신 시스템의 블록도
Fig. 1. Transmitter and receiver of digital communication system.

널리 사용되는 ASIC 칩은 고정된 기능을 고성능, 저전력으로 수행이 가능하여 통신 시스템에 적합하나 유연성이 없어 새로운 표준이 개발되면 이에 적합한 새로운 칩을 개발해야하는 문제점이 있다. ASIC 칩은 개발에 많은 비용을 필요하며 소량 생산의 경우 제작비용의 부담이 크다. 또한 설계에서 제작까지 오랜 시간을 필요로 한다. 기술의 급속한 발달에 따라 통신 시스템의 발전이 급속도로 진행되고 있는 지금 ASIC 칩의 이러한 특징은 큰 문제가 되고 있다.

이러한 이유로 최근 다양한 통신 표준을 만족시키는 프로그래밍 가능한 프로세서를 이용한 통신 시스템이 각광을 받고 있다. 프로그래밍 가능한 프로세서로는 주로 디지털 신호처리 프로세서(DSP)가 이용되는데 DSP를 사용하는 경우 ASIC 칩이 가지는 단점인 유연성의 결여, 높은 개발비용, 느린 개발 속도를 극복할 수 있다.[2]

그러나 DSP를 이용한 통신 시스템의 구현 시 데이터 부호화기에 사용되는 스크램블링, 채널 부호화기에 사용되는 길쌈부호화, 변조기에 사용되는 펑처링, 인터리빙 등의 비트 조작 연산이 비효율적으로 구현되게 된다. ASIC 설계 에서는 이러한 비트 조작 연산은 쉬프트 레지스터와 XOR로 간단하게 구현이 가능하다. 그러나 DSP는 주로 곱셈 및 가산 연산을 기본으로 연산기가 구성되어 있으며 word 단위의 처리를 기본으로 하는 메모리 구조를 가지고 있어 비트 조작 연산과 같은 연산은 비효율적으로 수행될 수밖에 없다. 또한 부호화율이 증가함에 따른 연산의 복잡도 증가가 MAC 연산에 비해 비트 조작 연산이 더 크다.[3] 따라서 비트 조작 연산을 위한 특별한 명령어와 이를 위한 특화된 가속기가 필요하다.

최근의 DSP에서는 이러한 약점을 보완하기 위해 하드웨어에 기반하고 있는 특정용도 가속기가 널리 이용되고 있다.[4]-[8] 예를 들면, TMS320C6416[9], MSC8101[10], Xtensa[11]에는 각각 채널 복호기(channel decoding), 필터링(filtering), 부동소수점 및 백터 연산을 위한 특정용도 보조프로세서가 도입되었다. 본 논문에서는 스크램블링, 길쌈부호화, 펑처링, 인터리빙과 같은 비트 조작 연산을 효과적으로 수행하기 위한 특정용도 명령어와 이를 지원하기 위한 하드웨어 구조를 제안하고 성능을 평가한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 통신 시스템에서 사용되는 비트 조작 연산의 주요 동작을 살펴보고 기존의 DSP에서 이러한 비트 조작 연산이 어떻게 이루어지 분석한다. Ⅲ장에서는 비트 조작 연산을 위한 명령어와 하드웨어를 제안한다. Ⅳ장에서 제안한 하드웨어에 대한 검증을 수행하고 기존의 DSP와의 성능을 비교한다. 마지막으로 Ⅴ장에서 결론을 맺는다.

Ⅱ. 비트 조작 연산 및 기존 DSP에서의 처리

본 장에서는 통신 시스템에서 사용되는 스크램블링, 길쌈부호화, 펑처링, 인터리빙, 디인터리빙과 같은 비트 조작 연산에 대해 분석하고 기존의 DSP에서 비트 조작 연산 방법에 대해 설명한다.

1. 비트 조작 연산

비트 조작 연산은 수행하는 연산의 성격에 따라 두 그룹으로 구분할 수 있다. 첫 번째 그룹은 입력 데이터를 쉬프트하고 XOR 연산을 하는 스크램블링, 길쌈부호화이다. 이러한 연산들은 입력된 데이터를 쉬프트 레지스터에 저장한 후 데이터를 쉬프트 시켜 부호화기의 구조에 따라 적절한 비트와 XOR 연산을 통해 출력이 얻어진다. 스크램블링, 디스크램블링, 길쌈부호화는 구속장, 부호화율, 생성 다항식으로 나타낼 수 있다.

두 번째 그룹은 입력된 데이터의 특정한 비트를 추출 및 삽입하는 동작을 하는 펑처링, 인터리빙, 디인터리빙(deinterleaving)이다. 펑처링은 입력 데이터에서 임의의 비트를 삭제하는 연산으로 부호율에 따라 삭제하는 유형(pattern)이 결정된다. 인터리빙, 디인터리빙은 입력비트의 순서를 바꾸어 출력하는 연산으로 인터리버(interliaver), 디인터리버(deinterleaver)의 크기(전체 비트수)와 인터리빙 방식에 따라 정해진다. 이러한 연산들도 규칙적인 유형이 존재하나 표준에 따른 다양한 변화를 수용하기 위한 구조 구현은 용이하지 않다. 그러나 임의의 위치에 비트를 삽입(insert)하거나 임의의 위치에서 비트를 추출(extract)하는 연산은 펑처링, 인터리빙, 디인터리빙의 기본적인 공통연산이라고 할 수 있다.

2. 기존 DSP에서 비트 조작 연산의 처리

그림 2는 기존 DSP의 데이터 연산기(DPU:data processing unit)를 나타낸 것이다. 데이터 연산기는 산술 연산기, 논리 연산기, 쉬프터로 구성된다. 이를 이용하여 앞에서 설명한 첫 번째 그룹의 연산을 수행하기 위해서는 반복적인 쉬프트와 XOR 연산을 이용할 수밖에 없다. 비트 조작 연산을 수행하기 위해서는 우선 레지스터에 입력 데이터를 넣고 쉬프터는 이 데이터를 쉬프트 한다. 마지막으로 논리 연산기는 XOR 연산을 수행한다. 그러나 기존의 디지털 신호처리 프로세서는 다수의 데이터에 대해 병렬 쉬프트와 XOR 연산을 지원하지 않는다.

따라서 이러한 연산을 수행하기 위해서는 많은 연산 사이클이 소모된다. 또한 앞에서 설명한 두 번째 그룹의 연산을 수행하기 위해서는 쉬프트를 사용하여 서로 반대 방향으로 번갈아 쉬프트 함으로써(왼쪽→오른쪽, 오른쪽→왼쪽) 임의의 위치의 1개의 비트 또는 연속된 비트들을 추출할 수 있다. 이를 위해서는 2회의 쉬프트 연산을 필요로 하며 임의의 위치의 연속해 있지 않은 다수의 비트를 추출하기 위해서는 더 많은 저장 공간과 쉬프트 연산을 필요로 하게 되며 또한 새로이 추출된 비트들을 통합하는 연산을 필요로 한다. 비트 삽입 논리 연산기의 OR 연산을 통해 수행이 가능하며 비트 삭제는 연산기의 AND 명령을 통해 수행이 가능하다.

그림 3은 상용 DSP에서 지원하는 비트 삽입 및 추출 연산 과정을 나타낸 것이다. 그림 3(a)는 스타코어(StarCore)의 SC140에서의 연산 과정을 나타낸 것으로 길이와 오프셋(offset)을 입력받아 연속된 비트들을 추출하는 연산과, 연속된 비트를 삽입하는 연산을 보이고 있다.[12] 또한 그림 3(b)는 TI(Texas Instruments)의 TMS320C6x의 연산 과정을 나타낸 것으로 오프셋을 입력받아 연속된 비트들을 추출하는 연산과 두 개의 워드(word)를 한 비트씩 번갈아 혼합 및 역혼합 하는 고정된 연산을 보이고 있다.[13] 그림 3(c)는 TI의

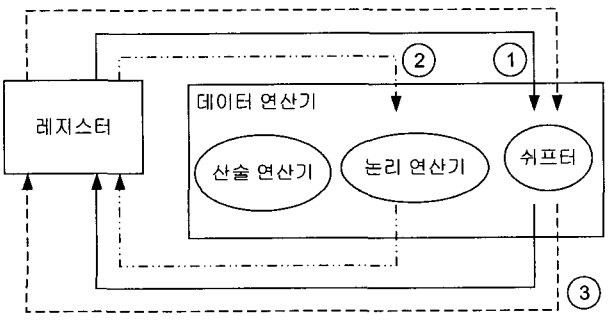

그림 2. 기존 DSP의 데이터 연산기
Fig. 2. Data processing unit of general DSPs.

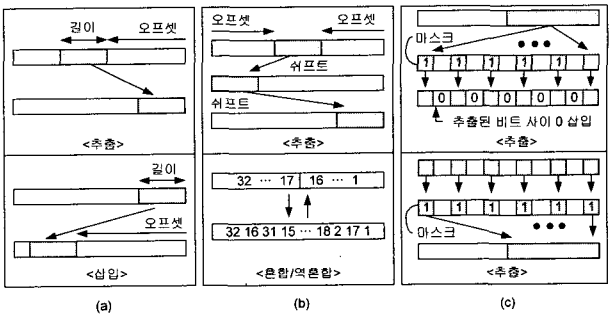

그림 3. 상용 DSP에서 지원하는 비트 삽입 및 추출 연산
Fig. 3. Bit extraction/insertion operations of existing DSPs.

TMS320C55x의 연산 과정을 나타낸 것으로 연속된 입력 비트들을 MASK의 비트들 중 세트(set)되어 있는 비트의 위치에 삽입 및 추출하는 비트 삽입 및 추출 연산을 보이고 있다. 그러나 TMS320C55x의 삽입 연산은 결과에 삽입 비트들 사이에 0이 들어가게 되며 추출 연산은 추출한 비트들이 위치하는 부분 이외에는 0으로 처리되기 때문에 주로 하나의 입력을 두개로 나누거나 두개의 입력을 하나로 합치는데 사용된다.[14]

기존의 디지털 신호처리 프로세서가 지원하는 비트 삽입 및 추출 연산은 특정한 규칙에 따라 동작하도록 되어 있다. 그러나 임의의 위치에 연속하게 있지 않는 비트들을 추출하거나 삽입하는 연산을 수행하지 못한다. 따라서 이러한 연산을 수행하기 위해서는 다수의 연산 사이클이 필요하다는 문제점이 있다.

III. 비트 조작 연산 및 기존 DSP에서의 처리

본 장에서는 비트 조작 연산을 위한 특정용도 명령어 및 이를 지원하기 위한 가속기를 제안한다. 제안하는 가속기는 N비트의 입력 데이터를 1비트 단위로 쉬프트를 수행한 N개의 데이터를 생성하고 이 중 지정한 데이터들과 원본 입력 데이터와 XOR 연산하기 위한 shift-XOR 배열 연산기, 임의의 불연속한 비트의 삽입 및 추출을 위한 비트 삽입 및 추출 연산기, 데이터를 비트 단위로 입력할 수 있는 비트 단위 입력 레지스터로 구성되어 진다. 제안하는 명령어들은 스크램블링을 위한 SCB, 길쌈부호화를 위한 CONV, 펑처링과 인터리빙 및 멀티플렉싱(multiplexing)을 위한 PUNC이다.

1. 비트 조작 연산 가속기의 구조

그림 4는 제안하는 비트 조작 가속기에 포함되어 있는 shift-XOR 배열 연산기, 비트 삽입 및 추출 연산기,

비트 단위 입력 레지스터를 보이고 있다. MASK1과 MASK2는 비트 조작 가속기가 원하는 동작을 하도록 제어하는데 사용된다. MASK1의 비트 수인 N과 MASK2의 비트 수인 M은 가속기가 다룰 수 있는 데이터의 크기에 맞게 임의로 선택이 가능하다.

그림 5는 제안하는 shift-XOR 배열 연산기의 동작을 자세히 나타내고 있다. 우선 M+N 비트의 입력 데이터와 N비트의 MASK1을 입력으로 받아서 각각 1에서 N까지 쉬프트 된 M 비트 길이의 N개의 출력을 생성해 낸다. MASK1에서 선택된 비트에 해당하는 쉬프트 된 데이터와 이전 출력 데이터는 XOR 연산이 병렬로 이루어진다. 즉, MASK1의 X번째 비트가 '1'로 세트되어 있다면 MASK1의 X-1 비트에 해당하는 출력 값과 XOR 연산을 거쳐 출력 되며 '0'으로 세트되어 있다면 X-1 비트에 해당하는 출력 값이 그대로 출력이 된다. 이와 같은 연산이 총 N개 병렬로 수행이 된다. 따라서 N개의 출력 데이터가 생성이 되며 이는 스위칭 네트워크(switching network)로 전달된다.

스위칭 네트워크는 받아들인 N 개의 데이터 중 MASK1에 의해 선택된 데이터를 레지스터들에 저장한다. MASK1은 레지스터의 선택 신호로 사용되며 따라서 레지스터에는 N개의 입력 신호 중 MASK1에 의해 선택된 유효한 데이터만을 선택적으로 저장하게 한다. shift-XOR 배열의 논리 연산식은 아래와 같다.

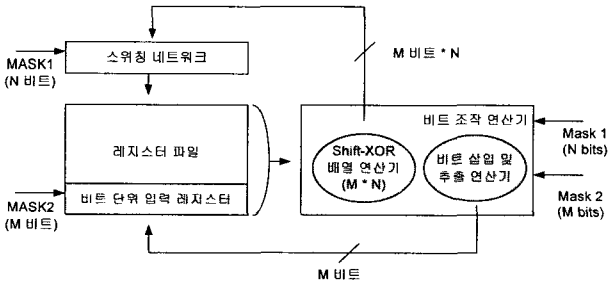

그림 4. 비트 조작 연산 가속기
Fig. 4. The proposed bit manipulation accelerator.

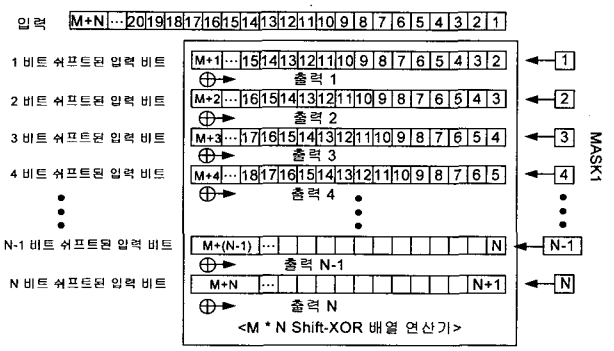

그림 5. shift-XOR 배열 연산기
Fig. 5. Logical operation of shift-XOR array.

```
Output 1 = (MASK1(1) AND input>>1) XOR Input
Output 2 = (MASK1(2) AND input>>2) XOR Output 1
...
Output N = (MASK1(N) AND input>>N) XOR Output N-1
```

이러한 XOR 연산 동작과 결과 값을 레지스터에 저장하는 동작은 한 연산 사이클에 완료되며 MASK1에 따라 다양한 연산의 수행이 가능하다. 또한 입력과 출력 비트의 크기인 M과 N은 임의로 선택할 수 있으며, 크기가 커질수록 한번에 연산할 수 있는 비트 수 및 지원 가능한 표준이 늘어나게 된다.

그림 6은 비트 삽입 및 추출 연산기의 동작을 나타내고 있다. 비트 삽입 및 추출 연산기는 N비트의 입력 데이터와 MASK1, M비트의 MASK2를 입력 받는다. MASK1은 입력 데이터의 비트들 중 추출을 원하는 비트들을 선택하는 기능을 가진다. 입력 데이터 중에서 MASK1에서 '1'로 세트되어 있는 비트에 해당하는 위치의 비트들이 선택되고 추출된다. MASK2는 MASK1에 의해 추출된 비트들을 출력 데이터에 삽입할 위치를 지정하는 기능을 가진다. 즉, MASK1의 비트 중 세트되어 있는 비트의 위치에 있는 입력 비트들은 MASK2의 비트 중 세트되어 있는 비트의 위치로 출력된다. 따라서 비트 삽입 및 추출 연산기는 입력 데이터의 임의의 위치에 있는 연속해 있지 않은 다수의 비트들을 추출할 수 있으며 또한 연속하지 않는 임의의 위치에 비트들을 삽입할 수 있다.

비트 삽입 및 추출 연산기에 의해 처리된 데이터는 비트 단위 입력 레지스터에 저장된다. 비트 단위 입력 레지스터는 데이터를 받아서 MASK2에 의해 지정된 위치의 데이터만을 갱신한다.

MASK1에 의해 입력 데이터 중에서 비트 단위 연산을 수행할 비트들이 정해지고 MASK2에 의해 선택된 비트들이 저장되어질 위치가 정해진다. 이러한 동작은 비트 단위 입력 레지스터 내에서 한 연산 사이클에 완료가 된다. 따라서 이러한 비트 조작 단계를 이용하면 여러 입력에서 데이터를 추출하는 경우나 추출된 데이터를 하나로 만들 경우 큰 성능 향상을 가져올 수 있다. 특히, 추출된 비트들을 하나로 만들기 위한 다수의 OR 연산을 한 연산 사이클 만에 처리가 가능한 비트 단위

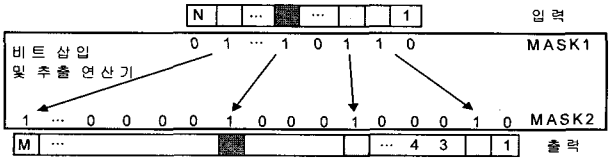

그림 6. 비트 삽입 및 추출 연산기
Fig. 6. Logical operation of bit extraction/insertion logic.

입력 레지스터로 대신할 수 있음으로 연산 속도는 더욱 빠르게 된다. 이러한 연산은 스크램블링 코드, 길쌈부호화 출력의 정렬, 펑처링, 인터리빙과 같은 연산에 효율적으로 사용될 수 있다.

2. 비트 조작 명령어

다음은 스크램블링을 위한 명령어인 SCB 명령어에 대한 문법과 설명이다. SCB 명령어는 shift-XOR 배열 연산기를 사용하며 MASK1과 입력 데이터는 명령어와 함께 지정한다. 만약 MASK1의 X번째 비트가 '1'로 세트되어 있다면 입력 데이터를 X 비트 쉬프트한 결과와 X-1번째에 해당하는 결과와 XOR 연산을 하게 된다. MASK1의 값에 따라서 다수의 출력이 병렬로 생성된다. MASK1의 값에 따라 결과가 저장될 레지스터가 지정된다. SCB 명령어의 문법과 연산식은 다음과 같다. 다음의 식에서 '>>'는 오른쪽 쉬프트를 뜻하며, '&'는 연결(concatenation)을 의미한다. 또, '<='는 값 대입을 나타낸다.

SCB MASK1, SRC(입력)
레지스터(MASK1에 의해 선택된 다수) <= SRC XOR (SRC >> (MASK1의 시프트 값))

다음은 길쌈부호화를 위한 명령어인 CONV 명령어에 대한 문법과 설명이다. CONV 명령어 역시 shift-XOR 배열 연산기를 사용하며 MASK1과 두 개의 입력 데이터, 그리고 출력 레지스터가 지정된다. 두 입력 데이터는 연결되어 shift-XOR 배열의 입력으로 사용된다. 동작은 SCB 명령어와 동일하나 SCB 명령어의 결과가 다수인 것과는 달리 최종결과만이 저장된다.

CONV MASK1, SRC1(입력1), SRC2(입력2), DST(출력)
DST(레지스터) <= (SRC1&SRC2) XOR ((SRC1 & SRC2) >> (MASK1의 쉬프트 값))

다음은 펑처링을 위한 명령어인 PUNC 명령어에 대한 문법과 설명이다. PUNC 명령어는 비트 추출 및 삽입 연산기를 사용하며 MASK1과 MASK2, 그리고 입력 데이터가 지정된다. MASK1에 의해 입력 데이터 중에서 추출할 비트가 선택이 되며 MASK2에 의해 비트 단위 입력 레지스터에 추출된 비트가 삽입될 위치를 지정한다.

PUNC MASK1, MASK2, SRC(입력)
비트 단위 입력 레지스터(MASK2에 의해 선택된 비트 위치) <= SRC(MASK1에 의해 선택된 비트)

Ⅳ. 시뮬레이션 및 성능 평가

제안한 비트 조작 가속기는 제안하는 DSP 중 DPU의 일부분으로 유용하게 사용되었다. 제안한 DSP는 하나의 프로그램 메모리, 두 개의 데이터 메모리, PCU(Program Control Unit:프로그램 제어 장치), DPU, 그리고 AGU(Address Generation Unit:주소 생성 장치)로 구성되어 있다. 제안하는 비트 조작 가속기는 DPU의 일부분이다. 제안한 DSP는 16비트 데이터 처리를 기본으로 하며 명령어 파이프라인은 6단계로 이루어져 있다. 제안한 구조는 VHDL 언어를 이용하여 구현하였으며 삼성의 0.18㎛ 표준 셀(standard-cell) 라이브러리를 이용하여 합성하였다. 제안한 DSP의 전체 게이트 수는 약, 80,000개이며 비트 조작 가속기(N=8, M=16)의 게이트 수는 1,700개에 불과하다. 합성 결과 제안한 DSP의 최대 동작 속도는 약 280 MHz로 나타났다.

표 1은 기존의 프로그래밍 가능한 프로세서와 제안한 프로세서와의 비교를 보이고 있다. 기존의 DSP들이 4개의 쉬프터와 4개의 논리 연산 장치를 갖는 VLIW(Very Long Instruction Word) 구조를 채택하고 있음에도 불구하고 제안한 DSP가 스크램블링, 길쌈 부호화, 인터리빙에 더 효율적임을 볼 수 있다.

스타코어 SC140과 비교해 보면, 제안한 구조는 길쌈부호화시 약 67%, 블록 인터리빙시 78%의 연산 사이클을 감소시킬 수 있다. TI의 62x와 비교해 보면, 제안한 구조는 스크램블링시 48%, 길쌈부호화시 84%의 연산 사이클을 감소시킬 수 있다.

표 1. 성능 비교
Table 1. The result of performance comparisons.

	StarCore SC140[15]	TI 62x[16]	제안하는 DSP
연산기	4 Shifters, 4 ALUs	4 Shifters, 4 ALUs	BMA
길쌈부호화 (Cycle) (IS-95, K=9, R=1/2, 192 bits)	463	-	152
블록 인터리빙 (Cycle) (16 × 6 bits)	414	-	91
스크램블링 (MIPS) (802.11a, 12 Mbps)	-	39×10^6	20×10^6
길쌈부호화 (MIPS) (802.11a, 12 Mbps)	-	77×10^6	12×10^6

Ⅴ. 결 론

본 논문에서는 스크램블링, 길쌈 부호화, 펑처링, 인터리빙과 같은 연산을 효과적으로 지원하기 위한 특정

용도 명령어와 이를 지원하기 위한 비트 조작 가속기를 제안하였다. 제안한 비트 조작 가속기는 병렬 쉬프트, XOR 연산과 비트 추출 및 삽입 연산을 지원한다. 제안한 비트 조작 가속기는 VHDL 언어를 이용하여 구현하였으며 삼성의 0.18㎛ 표준 셀 라이브러리를 이용하여 합성하였다. 성능은 기존의 DSP와 비교해 40% ~ 80%의 연산 사이클을 감소시킬 수 있으며 게이트 수는 1700개에 불과하다. 제안한 가속기는 다양한 통신 표준에 비트 조작 연산을 위해 쉽게 구현될 수 있다.

참 고 문 헌

[1] Sug H. Jeong, Myung H. Sunwoo, and Seong K. Oh, "Flexible Hardware Structures for CDMA Systems," in Proc. International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), July 2003.

[2] J. Glossner, J. Moreno, M. Moudgill, J. Derby, E. Hokenek and D. Meltzer et al., "Trends in compilable DSP architecture," in Proc. IEEE Workshop on Signal Processing Syst. (SIPS'2000), Oct. 2000, pp. 181-199.

[3] K. Masselos, S. Blionas, and T. Rautio, "Reconfigurability requirements of wireless communication systems," in Proc. IEEE Workshop on Heterogeneous Reconfigurable Systems on Chip, April 2002.

[4] Jeong H. Lee, Sug H. Jeong, and Myung H. Sunwoo, "Application-specific DSP architecture for OFDM modem systems," in Proc. IEEE Workshop on Signal Processing Syst. (SIPS'2003), Aug. 2003.

[5] U. Walther and G. P. Ferrweis, "PN-generators embedded in high performance signal processors," in Proc. IEEE Int. Symp. Circuits Syst.(ISCAS'2001), May 2001, pp. 45-48.

[6] Chi-Kuang Chen, Po-Chih Tseng, Yung-Chil Chang, and Liang-Gee Chen, "A digital signal processor with programmable correlator array architecture for third generation wireless communication system," IEEE Trans. Circuit Syst. II, vol. 48, pp. 1110-1120, Dec. 2001.

[7] Jae S. Lee and Myung H. Sunwoo, "Design of new DSP instructions and their hardware architecture for high-speed FFT," Journal of VLSI Signal Processing, Kluwer Academic Publishers, vol. 33, pp. 247-254, Mar. 2003.

[8] Jae S. Lee, Myung H. Sunwoo, and Seong K. Oh, "Design of DSP instructions and their hardware architecture for a Reed-Solomon codec," in Proc. IEEE Workshop on Signal Processing Syst. (SIPS'2002), Oct. 2002, pp. 103-108.

[9] Texas Instruments, Inc [Online]. Available: http://www.ti.com

[10] Motorola, Inc [Online]. Available: http://www.motorola.com

[11] Tensilica, Inc [Online]. Available: http://www.tensilica.com

[12] SC140 DSP Core Reference Manual, Motorola Semiconductors Inc., Denver, CO, 2001.

[13] TMS320C62xx User's Manual, Texas Instruments Inc., Dallas, TX, 2000.

[14] TMS320C55x User's Manual, Texas Instruments Inc., Dallas, TX, 2001.

[15] SC140 Functional Libraries, Motorola Inc. [Online]. Available: http://www.motorola.com

[16] Euro Sereni, Silvia Culicchi, Vanni Vinti, Enrica Luchetti, Simone Ottaviani, and Michele Salvi, "A Software RADIO OFDM Transceiver for WLAN applications," Electronic and Information Engineering Department, University of Perugia, Italy, 2001.

저 자 소 개

정 석 현(정회원)
2002년 아주대학교 전자공학과 학사 졸업.
2004년 아주대학교 전자공학과 석사 졸업.
2004년~현재 대우 전자 연구원.
<주관심분야 : SoC 설계, DSP 코어 설계, Software Defined R

선우명훈(정회원)
1980년 2월 서강대학교 전자공학 학사.
1982년 2월 한국과학기술원 전자 공학 석사.
1982년 3월~1985년 8월 한국전자통신연구소(ETRI) 연구원.
1985년 9월~1990년 8월 Univ. of Texas at Austin 전자공학 박사.
1990년 8월~1992년 8월 Motorola, DSP Chip Division (미국).
1992년 8월~1996년 10월 아주대학교 전기전자공학부 조교수.
1996년10월~2001년 9월 아주대학교 전자공학부 부교수.
2001년 10월~현재 아주대학교 전자공학부 교수.
<주관심분야 : VLSI 및 Parallel Architecture, 통신 멀티미디어용 DSP 칩 및 ASIC 설계>adio>