論文 86-23-6-24

Digital Signal Processor와 개발시스템의 설계 및 구현

(Design and Implementation of Digital Signal Processor and Development System)

林 光 一*, 李 宇 善*, 申 仁 澈**, 李 太 遠*

(Kwang Il Lim, Woo Sun Lee, In Chul Shin and Tae Won Rhee)

要 約

실시간 처리를 위한 microprogrammable digital signal processor를 bit-slice소자, 병렬 곱셈기, 74시리즈 TTL, MOS 메모리등을 사용하여 설계, 제작하였으며 마이크로인스트럭션을 정의하고 응용프로그램 개발을 위한 시스템을 구성하였다.

성능 평가를 위하여 디지탈 필터와 FFT를 실현시킨 결과 빠른 실행시간을 얻음으로서 신호처리를 고속으로 할 수 있음을 확인하였다.

Abstract

A real-time microprogrammable digital signal processor is designed and implemented using the bit-slice logic, a parallel multiplier, 74 series TTLs and MOS memories. A microinstruction set for the processor is defined and an application program development system is constructed. For its performance evalution, a digital filter and FFT are implemented with this digital signal processor. It is proved that this processor is faster than commercially available single chip digital signal processors such as μPD 7720, AMI 2811, enabling very high speed digital signal processing.

I. 서 론

Digital signal processor(이하 DSP)는 디지탈 필터등을 구현한 특수 프로세서로부터 프로그램 가능한 고속 FFT프로세서등 많은 종류가 있으며 신호처리에 있어 중요한 역할을 하고 있다.[1,2] DSP는 디지탈 신호처리에서 기본적으로 요구되는 고속연산기능 이외에도 유연성, 프로그래밍의 용이성, 정확성, 다른 시스템과의 인터페이싱등이 고려되어야 하며 그 DSP를 응용하기 위한 편리한 개발시스템이 구비되어야 한다.

최근 VLSI기술의 획기적인 발달과 디지탈 신호처리 분야의 연구가 활발하여짐에 따라 여러가지의 범용 DSP가 개발되었다.[3,4,5] 지금까지 개발된 DSP는 많은 종류가 있으나 대표적인 것에는 intel의 2920, AMI의 S2811, Bell 연구소의 DSP-1, NEC의 μPD7720, TI의 TMS320등이 있으며 이것들은 각각 고가의 개발시스템을 구비하고 있다.

본 논문에서는 일반 DSP의 구조적 기능을 보완하기 위하여 bit-slice 소자와 병렬 승산기, 74시리즈 TTL, MOS 메모리등을 이용하여 구조면에서 빠른 처리를 할 수 있는 real-time microprogrammable digital signal processor(이하 RMDSP)를 설계하고 이 RMDSP의 microprogram을 위한 개발 시스템에 관하여 기술하였다.

II. RMDSP의 설계

RMDSP는 인스트럭션의 흐름을 제어하는 제어부,

*正會員, 高麗大學校 電子工學科
(Dept. of Elec. Eng., Korea Univ.)
**正會員, 檀國大學校 電子工學科
(Dept. of Elec. Eng., Dan Kook Univ.)

接受日字 : 1986年 3月 31日

신호처리에 필요한 연산을 수행하는 연산부, 신호처리 중에 사용되는 계수와 중간 결과를 수록하는 메모리부, 클럭발생과 변경, interrupt, 입출력을 처리하는 부분 등으로 크게 나눌 수 있다.

이 RMDSP는 고속 신호처리를 할 수 있도록 다음과 같은 사항을 고려하여 설계하였다.

• 인스트럭션과 데이타를 별도의 메모리에 분리시키고 인스트럭션 버스와 데이타 버스를 가지는 harvard achitecture[6]로 구성하여 인스트럭션과 데이타 간에 충돌없이 동시에 access가 가능하도록 한다.

• 신호처리시 매우 중요한 overflow/underflow의 처리를 위하여 hardwired logic으로 설계된 부가회로를 두어 기존 DSP와는 달리 overflow/underflow를 검출하여 수정 처리한다.

• 데이타 전송을 위한 내부 버스와 이와는 별도의 데이타 path를 두어 concurrency를 높인다.

• 인스트럭션은 마이크로인스트럭션에서의 fetch, decode routine이 필요없는 마이크로프로그램 수준으로 하며, horizontal방식으로 최대한의 병렬처리를 하게한다.

• FFT를 효과적으로 처리하기 위하여 하드웨어 bit-reverse logic과 데이타 메모리 출력단에 복소수를 위한 레지스터등을 둔다.

• 제어부분과 연산부분에서 여러형태의 파이프라인으로 전달 지연시간을 감소, 실행속도를 증가시킨다.[7]

• 한 인스트럭션 사이클(T-cycle)에 곱셈과 덧셈을 동시에 수행하도록 한다.

이러한 특징들은 실시간 신호처리에서 필수적인 고속연산 능력을 가지도록 구조적으로 병렬성을 강조하고 FFT 알고리즘에 적합하도록 하기위한 것이다. RMDSP의 내부구조는 그림 1과 같으며 데이타형식은 고정 소수점 방식을 사용하고 내부 데이타의 word size는 16비트로 96dB의 dynamic range를 가진다.

표 1은 RMDSP의 제원을 나타내었다.

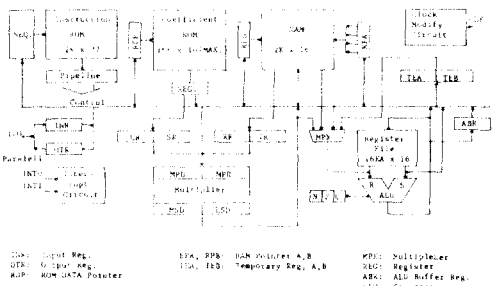

그림 1. RMDSP의 Block도.
Fig. 1. Block diagram of RMDSP.

표 1. RMDSP의 제원
Table 1. Specification of RMDSP.

Word size	16bit
Instruction ROM	77bit×2k
Coefficient ROM	16bit×16k
Data RAM	16bit×2k
Multiplier	16bit×16bit
Instruction cycle	250ns
Data format	Fixed point

1. 제어부

인스트럭션의 흐름을 제어하는 부분으로 인스트럭션 EPROM, 콘트롤러, 파이프라인 레지스터로 이루어진다. 한 인스트럭션은 77bit로 최대 2K까지 신호처리 알고리즘을 프로그램할 수 있으며 Am2910 시퀀서를 사용하여 인스트럭션 ROM을 access한다. 이 시퀀서는 16가지의 동작을 하며 5레벨의 서브루틴 스택을 가지고 있다.

파이프라인 레지스터는 인스트럭션의 처리속도를 높이기 위하여 현재의 인스트럭션을 수행하면서 동시에 다른 인스트럭션을 access하도록 한다.[8]

2. 연산부

4bit 처리능력을 가진 Am2903 RALU(registered arithmetic logic unit) 4-slice, Am2902 carry look-ahead generator, Am2904 shift & mux control logic, Am29517 병렬 승산기로 이루어져 있다.[9]

RALU 레지스터 파일은 16bit×16개의 레지스터들로 temporary 레지스터나 accumulator로 쓸 수 있다. 또한 Am2904는 Am2903과 함께 conditon처리와 shift 인스트럭션을 효과적으로 수행 할 수 있다. RALU 의 출력 port는 3-state buffer와 3-state 레지스터가 있어 출력을 내부 버스를 통해 RAM으로 전달하는 경우에는 일단 3-state 레지스터에 저장한 다음 전달을 하고, RAM이 아닌 다른 곳으로 보낼 경우에는 3-state buffer를 통해 직접 전달함으로서 전달 지연시간이 길어지지 않도록 하였다.

3. 메모리부

신호처리시 사용되는 계수를 수록하는 계수 ROM, 이 ROM의 번지를 지정하기 위한 ROM 포인터 RDP (ROM data pointer), 처리되는 데이타와 처리중의 데이타를 수록하는 RAM, 이 RAM의 번지를 지정하기 위한 포인터-RPA, RPB(RAM pointer A/B)로 구성되어있다.

특히 RAM 포인터 중에 RPA는 RMDSP가 처리한

수 있는 최대 1024 point complex FFT 수행시 bit reverse를 하드웨어적으로 처리할 수 있도록 하였다. 여기서 RAM이나 계수 ROM은 복소수 처리를 위하여 실수 부분과 허수 부분으로 나누지 않고 포인터 값이 짝수인 경우는 실수, 홀수인 경우는 허수로 취하는 형식이므로 bit reverse 방식이 달라지는 경우를 고려하였다. 또한 RAM에 데이타를 쓰거나 RAM으로 부터 데이타를 읽을 경우 RPB는 바로 전 인스트럭션 에서 그 번지로 값이 바뀌어져야 한다.

RMDSP에는 특수 포인터 외에도 RAM의 입출력단과 계수 ROM의 출력단에는 FFT를 위한 레지스터 를 둔다. RAM에는 실수 값을 위해 RR 레지스터, 허수 값을 위한 IR 레지스터가 있으며 계수 ROM에는 실수 값을 위하여 CR 레지스터, 허수 값을 위하여 SR레지스터가 있어 각 레지스터에 먼저 저장 함으로서 곱셈과 메모리 access를 동시에 할 수 있어 butterfly 계산시 인스트럭션 수를 줄일 수 있다.

4. 입출력과 interrupt부

입출력은 병렬 16bit port로 수행하며, 처리될 데이타를 받아들이는 입력 레지스터 INR과 처리된 결과를 외부로 보내는 출력 레지스터 OTR로 되어있으며 interrupt는 INT0와 INT1이 있다.

INT0인 경우는 인스트럭션 콘트롤러에 의해 7FFH 번지로, INT1인 경우는 7FEH번지로 점프하여 수행한다. 두 interrupt 인스트럭션 간에 priority는 없으며 한 interrupt 수행시 다른 interrupt는 disable 되도록 하였다.

5. 클럭과 overflow/underflow 처리부

외부에서 8MHz 클럭이 입력되면 내부클럭 변경회로에서 이것을 분주하여 4MHz로 시스템을 동작시킨다. 따라서 한 인스트럭션의 수행시간은 250nsec가 된다. 이 내부클럭은 overflow나 underflow 발생시 변경되어 correction operation을 수행하도록 설계되었다. 신호처리시 overflow/underflow가 발생하면 처리결과가 전혀 다른 값을 가지므로 hardwired logic으로 overflow/underflow를 검출하여 자동적으로 correction operation을 수행하여 overflow인 경우 ALU 출력으로 7FFFH가 출력되고 underflow인 경우는 8000H가 출력된다.

그림 2는 정상적인 인스트럭션과 overflow/underflow를 발생하는 인스트럭션 실행시의 타이밍도를 나타내었다.

Ⅲ. Micro-instruction set

인스트럭션은 총 77bit로 24개의 필드로 구분되며

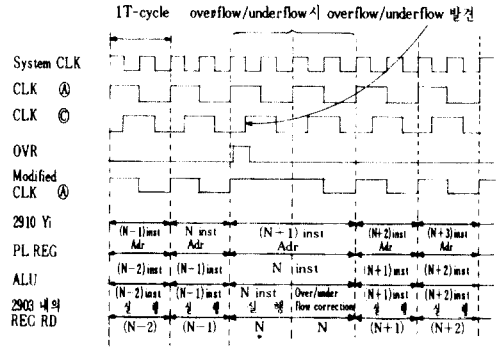

그림 2. 정상적인 인스트럭션과 overflow/underflow를 유발하는 인스트럭션 실행시 타이밍도

Fig. 2. Timing diagram for executions of normal instruction and instruction causing overflow/underflow.

그림 3에 마이크로인스트럭션 형식을 나타내었다.

이 마이크로인스트럭션 형식은 각기 하드웨어 기능별로 sequencer field, RALU field, multiplier field, internal bus field, pointer field 등으로 크게 나누어질 수 있다. 따라서 micro-instruction set 정의는 이러한 field들의 기능을 중심으로 mnemonic과 code를 구성하고 분류하였다.

표 2는 정의된 micro-instruction set이다.

1. Sequencer instruction

RMDSP는 구조면에서 instruction의 decoding routine을 가지고 있지 않으므로 micro instruction의 흐름을 제어하는 기능을 가진 인스트럭션이 필요하며 sequencer field에서 이 기능을 수행한다. 이 인스트럭션들은 기능에 따라 condition 또는 branch address 또는 counter value 등을 수반할 수 있다.

RMDSP의 status flag는 carry, zero, negative, overflow가 있으며 status는 sequencer instruction의 condition으로 사용된다. 그러나, overflow는 하드웨어로 자동 처리된다.

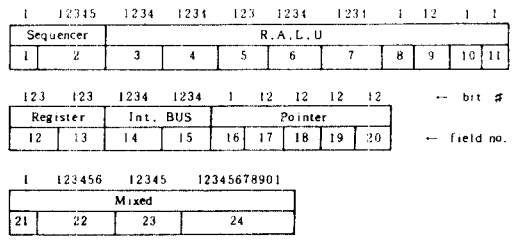

그림 3. RMDSP의 마이크로 인스트럭션 형식

Fig. 3. Micro-instruction format of RMDSP.

표 2. 마이크로 인스트럭션 set
Table 2. Micro-instruction set.

	Mnemonic	Operation		Mnemonic	Operation
1	JMP	Jump branch addr.		CMPR	Complement R
	JSR	Jump subroutine		CMPS	Complement S
	LDCT	Load counter value		TXR	Transfer R
	RPCT	Repeat until CTR=0		TXS	Transfer S
	CONT	Continue		INC	Increment
	JPZ	Jump zero		CLR	Clear
	RTN	Return from subr.		CLC	Clear carry flag
	CJMP	Conditional JMP		SEC	Set carry flag
	CJSR	Conditional JSR		SLA	Shift left Arith.
	CRTN	Conditional RTN		SLL	Shift left logical
2	SBCS	F←S-R+C-1		SRL	Shift right logical
	SBCR	F←R-S+C-1		RLC	Rotate left with C
	SUBS	F←S-R-1		ROL	Rotate left no C
	SUBR	F←R-S-1		RRC	Rotate right with C
	ADC	F←R+S+C		ROR	Rotate right no C
	ADD	F←R+S	3	LDMR	Load multiplier Reg.
	EOR	F←R EXOR S		MULT	Execute multiplier
	AND	F←R AND S	4	MOVE	Transfer data
	NOR	F←R NOR S		DATA	Immediate data
	NAND	F←R NAND S	5	SPTR	Set pointer field
	ORA	F←R OR S			

2. RALU instruction

RALU field의 동작을 결정하는 인스트럭션으로 16개의 범용 레지스터(0~15)와 ALU의 입력 및 출력의 data path와 기능을 제어한다. 이 RALU instruction은

- Arithmetic-Logic instruction
- Carry control instruction
- Shift-Rotate instruction

으로 나누어지며 이인스트럭션에 사용되는 operand는 표3의 BUS operand가 해당한다.

3. Multiplier instruction

Multiplier와 4개의 복소수 레지스터를 제어하기 위한 instruction으로 레지스터에 data를 로드시키거나 승수, 피승수를 로드하여 실행시키는 기능을 수행한다.

4. Internal BUS instruction

Internal BUS field는 프로세서내의 데이타 흐름을 제어하는 field로 ALU나 승산기가 실행하는 동안 내부버스를 사용하지 않는다면 동시에 데이타의 이동이 가능하다. 이 인스트럭션은 내부버스에서 데이타의 이동을 제어하거나 직접 데이타를 로드시키는 경우에 사용

표 3. 오퍼란드표
Table 3. Operand table.

	Mnemonic	Meaning		Mnemonic	Meaning
1	MEM	RAM data		AOB	ALU output buffer
	RPA	RAM pointer A		IDB	Internal BUS
	RPB	RAM pointer B		0~15	Reg. file 0~15
	MSD	MULT output MSB	2	CLRA	Clear RPA
	LSD	MULT output LSB		DECA	Decrement RPA
	MPD	Multiplicand		INCA	Increment RPA
	MPR	Multiplier		CLRB	Clear RPB
	ROM	Coeff. ROM data		DECB	Decrement RPB
	RDP	ROM data pointer		INCB	Increment RPB
	INR	Input register		CLRD	Clear RDP
	OTR	Output register		DECD	Decrement RDP
	TEA	Temporary Reg. A		INCD	Increment RDP
	TEB	Temporary Reg. B		BRRN	Bit-Rev. real number
	ABR	ALU buffer Reg.		BRCR	Bit-Rev. complex Re.
	CTR	Sequencer CNTR.		BRCI	Bit-Rev. complex Im.

* 1. BUS operand 2. pointer operand

되도록 정의하였다.

5. Pointer instruction

내부에 있는 각 메모리의 번지를 지정하는 pointer를 제어하기 위한 인스트럭션으로 여기에서 사용되는 operand는 신호처리의 알고리즘의 일반적 특징인 데이타의 규칙적인 access를 위하여 각 포인터의 증가, 감소 기능을 나타내도록 표3에 정의하였다.

Ⅳ. 개발 시스템

RMDSP를 위한 개발 시스템은 host computer 와 RMDSP를 연결시키는 hardware support와 microprogram 개발을 위한 software support로 구성하였다. 이 시스템의 목적은 복잡한 하드웨어를 가진 RMDSP를 일반프로세서 수준으로 프로그래밍할 수 있도록 하고, 응용 program의 개발과정을 간편하도록 하기 위한 것이다.

1. Hardware support

Hardware의 기본구성은 host computer와 RMDSP의 instruction메모리 사이의 interface가 주요장치로 되어있다. 이 interface는 host에서 생성된 RMDSP의 인스트럭션 코드를 host computer로 부터 전송 받아 RMDSP의 instruction memory로 로드시키는 역할을 수행한다. 이러한 구성을 위해서는 instruction EPROM을 static RAM으로 교체하여야 하고 interface에서 instruction memory formation에 맞는 적절한 code로의 변환이 필요하다. 그림 4는 interface 회

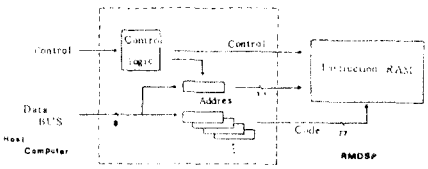

그림 4. Interface회로의 블럭도.
Fig. 4. Block diagram of interface circuit.

로의 블럭도이다.

2. Software support

RMDSP의 microprogram개발을 위하여 host computer에서 일괄적으로 자동처리 할 수 있는 microprogram development software(MDS) package를 개발하였다. 이 MDS는 editor, micro-assembler, loader/utility로 구성하였으며, 6502 assembly language로 작성하였다.

• Editor ——— microinstruction set에서 정의된 mnemonic으로 source program을 작성하기 위하여 text의 생성, 삭제, 검색, 출력 등의 편집기능을 수행한다.

• Micro-assembler ——— micro-assembler는 앞에서 정의된 micro-assembly language로 작성된 source program을 micro-instruction의 bit pattern으로 출력하는 2-pass assembler이다. 이 assembler의 source program 형식은 multistatement-per-instruction format[10]을 변형시켜 기능에 적합하도록 하였으며, 각 statement는 text number, symbol address(optional), instruction, operand, comment(optional)의 5개 field로 구성하였다. 다음의 예는 RMDSP에서 한 instruction에서 가능한 최대의 operation의 source program이다.

```
0001 *PARALELL OPERATION EXAMPLE
0002 PARAL JMP   START      'NEXT ADDR. START'
0003       ADC   0, TEA, I   'TEA (TEA)→REG'
0004       MULT  RR, CR      'RR*CR→MSD'
0005       MOVE  RPA, INR    'INR→RPA'
0006       LDMR  IR, SR      'RAM→IR, ROM→SR'
0007       SPTR  DECB, CLRD, TEC RPB, 'CLEAR RDP'
```

• Loader/utility ——— loader는 RMDSP의 instruction memory로 전송시키는 program으로 downloading 기능을 수행한다. utility는 응용 program 개발에서 필요한 부수적인 작업을 지원하기 위한 프로그램으로 file의 조작, EPROM 기록, 메모리관리 등의 기능을 수행한다. 따라서 MDS를 사용한 응용 program의 개발의 flow는 그림 5와 같이 할 수 있다.

V. 실험 및 검토

RMDSP의 성능을 평가하기 위하여 개발 시스템을

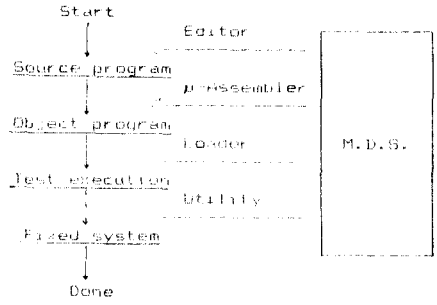

그림 5. RMDSP에 대한 프로그램 개발 흐름
Fig. 5. Program development flow for RMDSP.

이용하여 2차 IIR filter와 FFT등 실현시켜 다른 DSP로 실현했을 때와 비교하였다.

1. 2차 IIR filter의 실현

2차 IIR filter의 차분 방정식은

$W_n = X_n + A_1 * W_{n-1} + A_2 * W_{n-2}$

$Y_n = W_n + B_1 * W_{n-1} + B_2 * W_{n-2}$ 으로 표현된다.

이 2차 IIR filter와 3차 이상 IIR필터 실현시 RMDSP와 기존의 다른 DSP와 비교한 결과는 표 4이다. 표에서와 같이 2차 IIR filter실현시 μPD 7720보다 2 T-cycle 적 2개의 instruction이 줄어들며 HSP 와는 같다. 또 2차 IIR filter를 cascade로 연결하여 고차의 IIR 필터를 실현 할 수 있는데 이 경우 프로그램에서 루우프를 형성하여 실현한다. 따라서 루우프에 포함되는 T-cycle을 보면 μPD7720보다 훨 작으며 HSP보다는 하나가 많다. 그러나 RMDSP경우 overflow/underfow에 대한 대책이 마련되어 있어 유리하다.

표 4. 2차 IIR 필터 실현시 다른 DSP와의 비교
Table 4. Comparison RMDSP with other DSP on implementation of biquad IIR filter.

	2차 IIR필터 실현시 필요한 T-cycle수	3차 이상의 IIR필터 실현시 loop 포함되는 T-cycle수
실현된 RMDSP	9	7
NEC의 μPD 7720	11	9
Hitachi의 HSP	9	6

2. FFT의 실현

FFT에서는 butterfly의 계산이 가장 많이 걸리는 부분이므로 butterfly의 계산을 효율적으로 할 수 있으면 FFT를 빠른 시간에 할 수 있다. 앞에서 설명한 바와 같이 RMDSP는 butterfly를 효과적으로 하기위하여 메모리 루우프에 상수나 함수를 위한 RR, IR, CR, SR

레지스터를 두었다.

Butterfly는 그림 6 과 같이 4 번의 곱셈과 8번의 덧셈/뺄셈, 8 번의 메모리 access가 필요하다. RMD SP는 파이프라인과 동시 실행으로 butterfly를 8T-cycle에 수행한다. 표 5는 butterfly 수행시 메모리access, 곱셈, 덧셈/뺄셈의 수행 상태를 나타낸 것이다. 표 6은 FFT실현시 RMDSP와 다른 DSP를 비교한 것이다. 각 DSP들은 서로다른 클럭 주파수를 사용하므로 구조적인 측면에서 비교하기 위해서는 T-cycle로 비교하는 것이 타당하다.

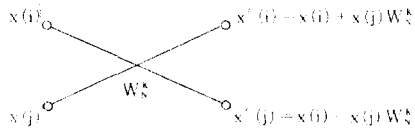

그림 6. Butterfly의 흐름선도
Fig. 6. Flowgraph of butterfly.

표 5. Butterfly의 실행
Table 5. Execution of butterfly.

	Memory Access	곱 셈	덧셈/뺄셈
7	$R_e x_k' \leftarrow 3R$	$R_e x_j * R_e W_N^k$	$3R \leftarrow TEB - ACCB$
8	$I_m x_k' \leftarrow 3R$	$I_m x_j * I_m W_N^k$	$ACCA \leftarrow R_e x_i * R_e W_N^k$
1	$TEA \leftarrow R_e x_i$	$I_m x_j * R_e W_N^k$	$ACCA \leftarrow R_e x_j * R_e W_N^k - I_m x_j * I_m W_N^k$
2	$TEB \leftarrow I_m x_i$	$R_e x_j * I_m W_N^k$	$ACCB \leftarrow I_m x_i * R_e W_N^k$
3	$C \leftarrow R_e W_N^k, R \leftarrow R_e x_i$		$ACCB \leftarrow I_m x_i * R_e W_N^k + R_e x_j * I_m W_N^k$
4	$S \leftarrow I_m W_N^k, I \leftarrow I_m x_i$		$3R \leftarrow TEA + ACCA$
5	$R_e x_j' \leftarrow 3R$		$3R \leftarrow TEB + ACCB$
6	$I_m x_j' \leftarrow 3R$		$3R \leftarrow TEA - ACCA$
7	$R_e x_j' \leftarrow 3R$	$R_e x_e * R_e W_N^{k'}$	$3R \leftarrow TEB - ACCB$
8	$I_m x_j' \leftarrow 3R$	$I_m x_e * I_m W_N^{k'}$	$ACCA \leftarrow R_e x_e * R_e W_N^{k'}$

표 6. FFT실현시 다른 DSP와의 비교
Table 6. Comparison RMDSP with other DSP on implementation of FFT.

	필요한 T-cycle 수				
	32 Complex	64 Complex	128 Complex	256 Complex	1024 Complex
설계된 RMDSP	984	2232	4984	11000	51192
NEC의 μPD7720	2800	6400	×	×	×
AMI의 2811	1333	×	×	×	×
Fast Bit Slice Computer for Real-Time Signal Processing	1745	3997	8883	19036	×

Ⅵ. 결 론

본 논문은 bit-slice 소자와 병렬승산기, 74시리즈 TTL, MOS메모리를 이용하여 실시간 처리용 micro-programable DSP를 설계 제작하고, 일반 마이크로프로세서 수준의 micro-instruction set을 정의 하였으며 응용 program 개발을 위한 개발 시스템을 구성하였다.

이 RMDSP에 개발시스템을 이용하여 디지탈 필터와 FFT를 실현한 결과 고속처리 능력을 확인하였다.

최근 급격히 발전하는 VLSI기술을 사용하여 이와 같은 구조를 갖는 DSP를 설계하면 보다 빠르고 단순한 instruction을 가진 단일 chip으로의 구성이 가능하리라 사료된다. 또한 simulator등의 더많은 system program 개발은 더욱 편리한 개발 시스템으로의 운용이 가능할 것이다.

參 考 文 献

[1] Lawarance R. Rabiner, Bernard Gold, *Theory and Application of Digital Signal Processing,* Prentice Hall, pp. 627-657, 1979.

[2] Andress C. Salazar, "Digital signal computers & processor," *IEEE Press,* pp. 3-71, 1977.

[3] J.R. Boddie, G.T. Daryanant, S.M. Walters "Digital signal processor: architecture and performance," *The Bell System Technical Journal* vol. 60, no. 7, Sep. 1981.

[4] Fred Mintzer, Ken Davis, Abraham Peled "The real-time signal processor," *IEEE Transaction on ASSP* vol. ASSP-31, no. 1, Feb. 1983.

[5] Yoshimune Hagiwara, Yuzo Kita, Hideo Hara "A single chip digital signal processor & It's application to real-time speech analysis," *IEEE Transactions on ASSP* vol. ASSP-31, no. 1, Feb. 1983.

[6] Kraft and Toy, *Mini/Microcomputer Hardware Design,"* Prentice Hall, pp. 307-344, 1979.

[7] Peter M. Kogge, *The Architecture of Pipelined Computer,* McGraw Hill.

[8] Jan Zeman, H. Troy Nagle JR, "A high speed microprogrammale digital signal processor employing distributed arithmetic," *IEEE Transactions on Computer* vol-29, Feb. 1980. *