論 文
35~8~1

DNL 1 에서 반복루프처리장치의 설계

Implementation of Iteration Loop in DNL1

金 元 燮* · 朴 熙 淳**
(Won-Sub Kim · Hi-Soon Park)

Abstracts

We proposed a preliminary Data Flow Machine Model(DNL1) operating on the basis of Node Label. In this model, all the PMs(Processing Modules) were synchronized with the content of LC(Level Counter) and were not implemented dy the processing cability on conditional nodes.

This paper presents an architecture of a concurrent multiprocessor system which was developed from DNL1 with two additional types of memories, CF(Control Flag) and ETF (Enabled Token Flag). The CF memory holds the control condition flag ('1' or '0') to be referenced to when a node is fired and the ETF represents the firability of a certain node. Firable nodes are fetched to the PU(Processing Unit) and processed.

This Data Flow system can be extended hierarchically by a network of simple modules. The principle working elements of the machine are a set of PMs, each of which performs the execution of the data flow procedures held in a local memory, NTM(Node Token Memory) within the PM.

1. 서 론

Data Flow Machine(DFM)은 기존의 Control Driven방식을 탈피하여 instruction에 필요한 operand가 도착되면 즉시 실행되는 Data Driven방식 병렬처리 컴퓨터이다. DFM을 위한 프로그램으로서 Data Flow Graph(DFG)[1]가 사용되며 이는 instruction 또는 Function을 나타내는 Node들과 이들 사이의 데이타 흐름을 표시하는 Arc들로써 도시한다. DFM은 현재까지대략 7가지 모델[2)-8)]들이 제안 또는 연구되었고 부분 실현 및 특수 용도에 시험 적용시켜보는 단계에 있다.

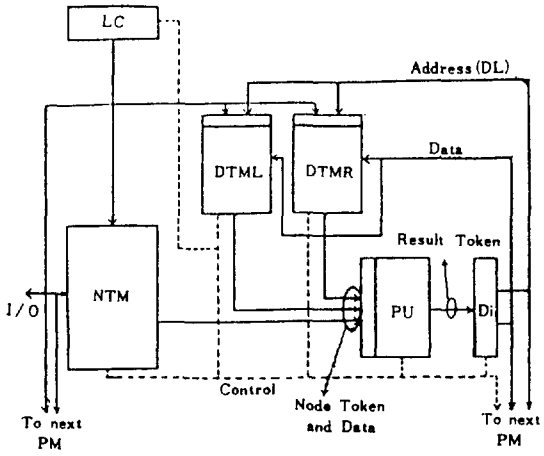
Node는 1) 그것의 모든 입력 Arc에필요한 operand가 전부 도착되었고 2) 동시에 그것의 출력 Arc에 result token이 존재하지 않을때 그 Node는 firable 또는 enable되었다고 말한다.

기본 모델 DNL 1 (그림 1)[10]에서 병렬 연결된 각 프로세서(PM)는 해당 프로세서에 분배된 명령들을 동시 처리한다. 그러나 이들 명령 처리의 시작(fetch)은 Location Counter(LC)의 Address Update후에 가능하므로 모든 명령의 완전 비동기 처리 방법이 아니다. 즉 모든 명령은 data flow 본래 개념으로부터 명령이 필요로하는 데이타가 도착되면 즉시 실행되도록 하는 것이 유리하나 DNL 1 모델에서는 데이타가 도착되었다 하더라도 시간적 처리 순위가 같은(si값이 동일한) 모든 Node의 처리가 끝난후 다음 단계 Node처리가 시작되므로 처리 시간이 짧은 명령은 처리 시간이 긴 명령의 처리 완료까지 휴지하여야 하는 단점이 있다. 즉 프로세서의 가동률이 낮다.

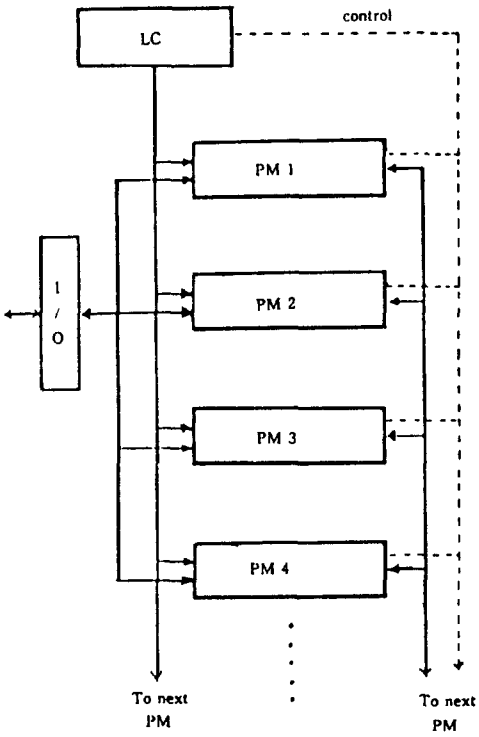
본 논문은 위의 기본 모델에서 반복 루프가 실행되게 하기 위하여 Conditional Node의 처리 방법을

*正 會 員 : 全州大 工大 電氣工學科 敎授 · 工博
**正 會 員 : 圓光大 工大 電算機工學科 助敎授 · 工博
接受日字 : 1985年 8月 2日

고찰하고 Processing Module(PM) 사이에 완전 비동기 동작이 가능하도록 하는 개선된 PM구조에 대하여 설명한다.



(a) Processing Modue(PM)의 구조



(b) PM의 연결

그림 1. 기본 모델 DNL 1의 구조
Fig. 1. Types of nodes.

2. 본　　론

2.1 반복 루프를 갖는 DFG와 Node Label

DFG에서 사용되는 기본적인 Node의 표시는 그림 2와 같고 실선은 Data Token, 점선은 Control Token('True' or 'false')을 표시한다. 루프의 반복계산과정을 설명하기 위하여 그림 3과 같은 샘플 프로그램[8]을 선택하며 이는 n! 계산에 관한 것이다. 그림에서 각 Node의 오른쪽 첨자는 Node Label(NL)[10]을 표시하고 모든 Node의 논리적 주소(Logical Address)로 사용된다.

NL은 2차원 첨자로서

si, pi　　si = 1, 2, 3, ···, N
　　pi = 1, 2, 3, ···, M

로 표시한다. si는 시간적 처리 순서를 의미하는 직렬적(Sequential) 일련번호이나 각 PM에서의 논리적 주소 또는 Address offset로 사용된다. 즉 임의의 프로그램에 Load Address가 할당되면 이값에 offset(si)를 더하거나 어떤 Algorithm에 의해 계산된 위치에 load한후 실행에 들어가도록함이 가능하다. Pi는 동시 처리 가능 Node(Parallel Node)들에 할당한 PM번호를 표시하고 최대값 M은 해당 시스템의 총 PM수를 나타낸다. Si와 Pi값은 반드시 순서적일 필요는 없으나 메모리의 이용률을 높이기 위해서 순서적으로 Labelling하는 것이 유리하다.

그림 3에서 초기치 n값은 루프를 반복할 때마다 1씩 감소하고 n(n-1)(n-2)(n-3)···(n-i) 계산을 실행하며 n-i=1 일때 계산을 끝낸후 'n'에 그 결과값

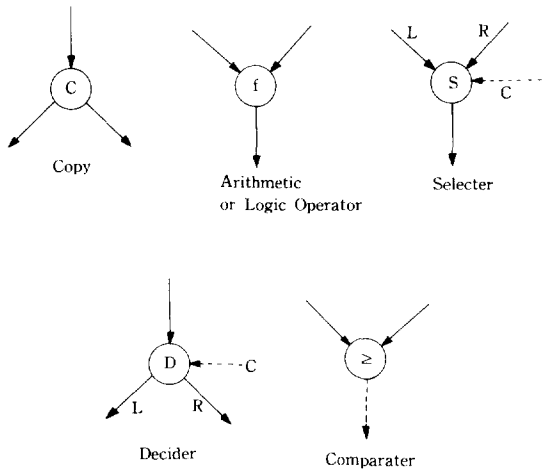


그림 2. Node의 종류
Fig. 2. Preliminary architecture of DNL 1.

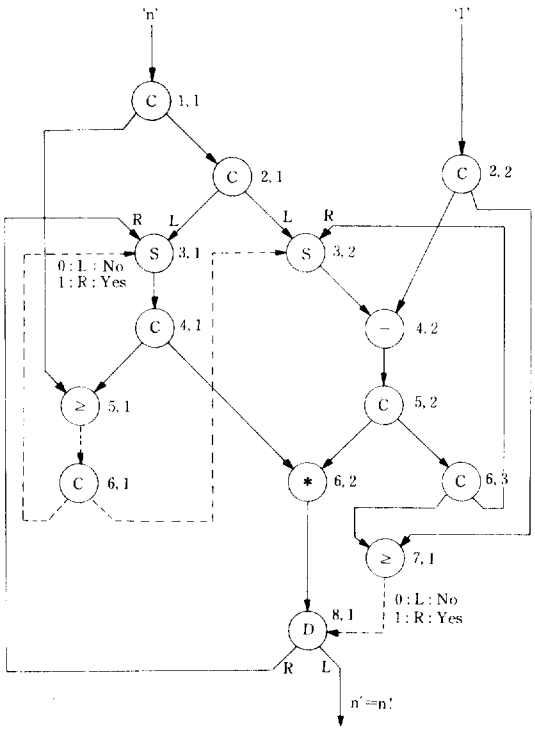


그림 3. 샘플 프로그램 n!
Fig. 3. Sample program n!.

을 보관한다. 여기서 S(3, 1), S(3, 2) 및 D(8, 1)은 Conditional Node로서 Control Token의 상태('0' or '1')에 따라 입출력 Arc중 Left 또는 Right를 결정한다. 따라서 S(3, 1)과 S(3, 2)는 start시 '0'로 reset되어 있어야 한다.

2.2 Node Token

위의 DFG의 모든 Node에 대하여 다음과 같이 코딩한다. 즉,

a) Nonconditional Node :

NL f DNL 1, [L/R/C] DNL 2, [L/R/C]

NL; Node Label

f; function(+, −, *, AND, OR등)

DNL 1 ; Destination Node Label 1

DNL 2 ; Destination Node Label 2

[L/R/C]; Left, Right or Control Arc

예 : 3, 2 COPY 4, 1, L 5, 2, R

b) Selecter :

NL f DNL CF

CF; Control Flag('0' or '1')

예 : 2, 1 SEL 2, 3, L 0

c) Decider :

NL f DNL 1, [L/R/C] DNL 2, [L/R/C] CF

예 : 4, 2 DECI 6, 2, L 8, 1, R 1

위의 예중 "3, 2 COPY 4, 1, L 5, 2, R" 명령은 다음과 같은 의미를 갖는다. 즉 DFG상에서 NL값 3, 2 위치에 존재하는 COPY명령은 입력 Arc(정해진 메모리 space)에 데이터가 도착되면 이를 그대로 복사(multiplex)하여 4, 1 Node의 Left Arc와 5, 2 Node의 Right Arc에 transfer(정해진 메모리 space에 write)한다. 또한 "2, 1 SEL 2, 3, L 0" 명령에서 NL=2, 1인 SEL Node는 CF=0이므로 2개의 입력 데이타중 왼쪽 데이타를 선택하여 2, 3 Node의 왼쪽 Arc로 보내준다. "4, 2 DECI 6, 2, L 8, 1, R 1" 에서는 Cf=1이므로 입력 데이타를 오른쪽 Arc 즉 8, 1, R로 출력시킨다.

이와 같은 방법으로 작성된 샘플 프로그램의 Node Token은 표 1과 같고 이는 DFG의 서술에 불과하다. 이를 다시 Parallel Level(Pi)별로 분류하고 각각의 Node Token Memory(NTM)에 load시키면 표 2와 같다. 표에서 '*' 표시는 해당 메모리위치에 해당 Node가 필요로 하는 operand가 도착될 메모리 space이고 최대 3종류 즉, Data Token Memory Left(DTML)[8], Data Token Memory Right(DTMR)[8] 및 Control Flag Memory(CF)가 필요하다.

ETF(Enabled Token Flag)는 각각의 Node가 사용할 operand의 도착 여부(present : '1', vacant: '0')를 나타내는 Flag로서 L/R/C='111'일때 해당 Node가 Firable을 함을 나타낸다. 표 2의 ETF값은 초기상태 값으로 function(opcode)의 종류에 따라 각각 고유의 code가 할당되어 있다.

표 1. DL별 node token
Table 1. Node tokens in DL sequence.

1, 1	COPY	5, 1, L	2, 1, L
2, 1	COP	3, 1, L	3, 2, L
2, 2	COPY	4, 2, R	7, 1, R
3, 1	SEL	4, , L	0
3, 2	SEL	4, 2, L	0
4, 1	COPY	5, 1, R	6, 2, L
4, 2	SUBT	5, 2, L	
5, 1	GE	6, 1, L	
5, 2	COPY	6, 2, R	6, 3, L
6, 1	COPY	3, 1, C	3, 3, C
6, 3	COPY	7, 1, L	3, 2, R
7, 1	GE	8, 1, C	
8, 1	DECI	3, 1, R	n' 1

표 2. 각 PM 메모리의 초기 상태
Table2. Initial state of PM memories.

PM 1	NTM	DTML	DTMR	CF	ETF L R C
1, 1	COPY 5, 1, L 2, 1, L	*			0 1 1
2, 1	COPY 3, 1, L 3, 2, L	*			0 1 1
3, 1	SEL 4, 1, L 0	*	*	*	0 0 0
4, 1	COPY 5, 1, R 6, 2, L	*			0 1 1
5, 1	GE 6, 1, L	*	*		0 0 1
6, 1	COPY 3, 1, C 3, 2, C	*			0 1 1
7, 1	GE 8, 1, C	*	*		0 0 1
8, 1	DECI 3, 1, R n′ 1	*		*	0 1 0
PM 2					
2, 2	COPY 4, 2, R 7, 1, R	*			0 1 1
3, 2	SEL 4, 2, L 0	*	*	*	0 0 0
4, 2	SUBY 5, 2, L	*	*		0 0 1
5, 2	COPY 6, 2, R 6, 3, L	*			0 1 1
6, 2	MULT 8, 1, L	*	*		0 0 1
PM 3					
6, 3	COPY 7, 1, L 3, 2, R	*			0 1 1

표 3. 수정된 DFG에 대한 메모리 내용
Table3. Memory contents of the Revised DFG.

PM 1	NTM	DTML	DTMR	CF	ETF
1, 1	COPY 1, 2, L 2, 4, L	*			0 1 1
2, 1	SEL 2, 3, L 0, ,	*	*	*	0 0 0
3, 1	COPY 3, 3, R 3, 4, L	*			0 1 1
4, 1	GE 4, 2, C	*	*		0 0 1
PM 2					
1, 2	COPY 1, 4, L 2, 1, L	*			0 1 1
2, 2	COPY 2, 4, R 3, 3, L	*			0 1 1
3, 2	COPY 1, 4, C 2, 1, C	*			0 1 1
4, 2	DECI N′ 1, 4, R	*		*	0 1 0
PM 3					
1, 3	COPY 2, 3, R 4, 1, R	*			0 1 1
2, 3	SUBT 3, 1, L	*	*		0 0 1
3, 3	MULT 4, 2, L	*	*		0 0 1
PM 4					
1, 4	SEL 2, 2, L 0	*	*	*	0 0 0
2, 4	GE 3, 2, L	*	*		0 0 1
3, 4	COPY 4, 1, L	*			0 1 1

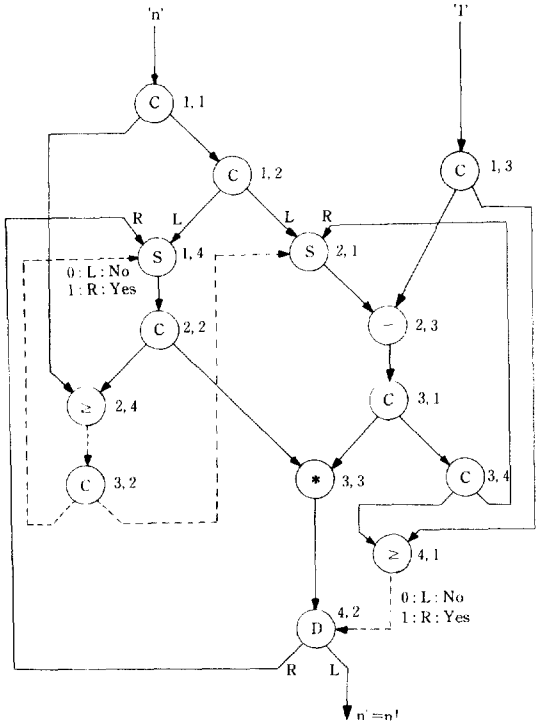


그림 4. 수정된 NL에 의한 DFG
Fig. 4. DFG with the revised NL.

그림 4 는 샘플 프로그램에 대한 Labelling을 최대 Enabled Token의 addressing을 ETF가 담당하게 하였다. 즉 ETF의 LRC = '111'인 Node가 발견(match)되면 해당 Node 및 데이타가 Processing Unit PM수(예로서 4 개)에 맞도록 변경한 것으로 Pi값을 1 ~ 4 까지 할당하고 Pi값이 한번 변화할 때마다 Si값을 증가시킨다. 즉

(1, 1) (1, 2) (1, 3) (1, 4) (2, 1) (2, 2) … (N, M)

와 같은 첨자열을 할당한 것이다. 이들 수열의 변화는 매우 단순하므로 PM수가 다른 시스템에 적용될 경우 그 수정 변경이 어렵지 않다. 이렇게 변경된 NL에 의해 위의 방법으로 코딩하고 분류하면 표 3 과 같다. 프로그램이 각 NTM에 고르게 분포되어 있음을 알 수 있다.

2.3 PM 구조

Conditional Node의 처리를 위해 개선된 PM구조는 그림 5 와 같고 Conditional Flag(CF) Memory와 Enabled Token Flag(ETF) Memory가 추가되었다. 이미 발표된 DNL 1 에서 PM간의 비동기 실행이 가능하도록 LC(Location Counter)를 제거하고 대신 (PU)로 fetch되고 실행후의 결과를 해당 목적지에

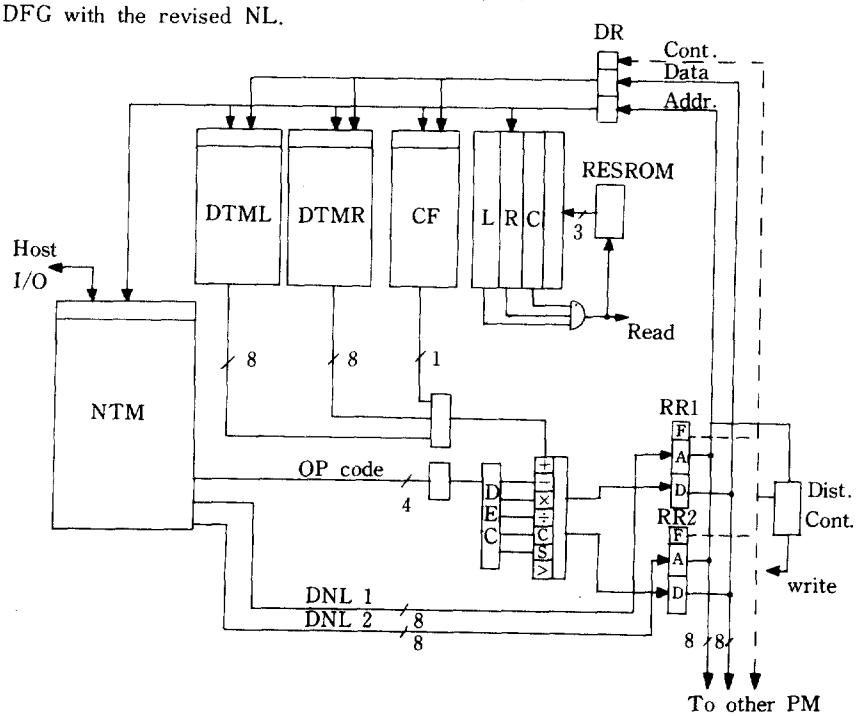


그림 5. 개선된 PM의 내부 구조
Fig. 5. Internal structure of PM.

write하며 해당 위치의 LRC값을 '1'로 set한다. CF는 추가된 Control Token 보관용 메모리이고 범용 RAM으로 구성한다.

RESROM (Reset ROM)은 각 function의 순수(고유) LRC값을 보관한다. 즉 operand가 전혀 도착하지 않았을때의 ETF값을 기억하는 ROM으로 임의 Node 및 데이타가 fetch되어 PU에서 실행됨과 동시에 동일 위치의 LRC값을 본래 상태로 vacant시키기 위한 것이다.

2.4 동 작

샘플 프로그램의 실행 과정은 다음과 같은 순서로 반복된다.

a) 초기치 'n'과 '1'값이 해당 목적지 DTML/R에 write되면서 ETF의 LRC Flag를 '1'로 set한다.

b) 이때 LRC값이 모두 '111'인 Node는 1, 1 Copy 2, 2Copy 뿐이고 이들 두 Node가 enabled되었으므로 해당 Node와 operand를 fetch하여 각각 PM 1과 PM 2에서 동시에 실행된다.

c) PU에서의 실행과 동시에 fetch된 Node의 ETF값을 본래값(이경우 '011')으로 reset시킨다. input Arc를 vacant시켜 다음 enable이 가능하도록함.

d) 실행된 결과를 Distributer가 받아 해당 목적지 DTML/R 또는 CF에 write하고 같은 위치의 LRC Flag를 set한다.

e) 위의 b)의 과정으로 반복한다.

위의 동작 과정을 간단히 도시하면 그림 6과 같고 모든 Node의 처리시간이 대략 동일한 것으로 간주하였다.

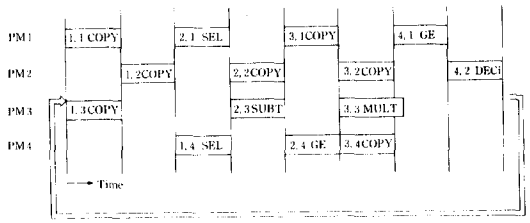
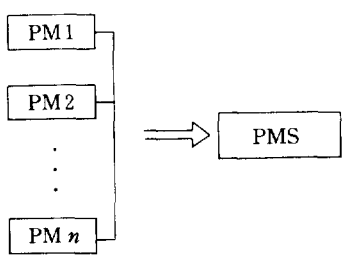
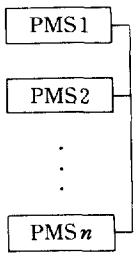


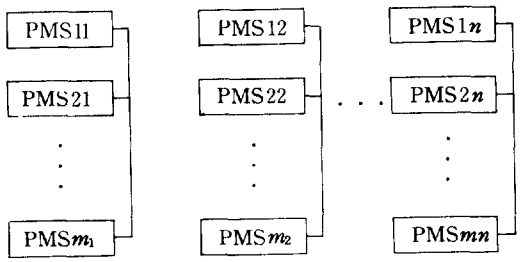
그림 6. 반복 루프의 병렬 처리(PM이 4개인 경우)
Fig. 6. Parallel processing of loop iteration (In the case of 4 PMs).



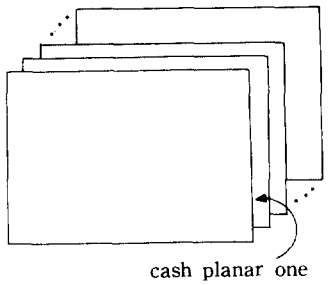
a) Linear Configuration of PM→PMS



b) Linear Configuration of PMS



c) Planar Configuration of PMS



d) Cubic Configuration

그림 7. PM의 확장
Fig. 7. Varied configurations of extended PMs.

2.5 PM의 확장[11]

NTM과 DTM을 범용 static RAM (Access time 150~200ns)으로 사용할 경우 각 PM에서의 instruction cycle time은 600ns (= Write access+Read access+Processing+Distribute) 정도에 가능하나 PM의 수가 너무 많으면 Distributer access time70 ns * 8 = 560ns로 한정함이 타당하다. 그림 7 은 단위수량(n개)의 PM들을 PMS (Processing Module Site)라 칭하고 다시 이 PMS를 단위 Block으로 사용하여 구성할 수 있는 3 가지 방법을 구상한 것이다.

3. 결 론

본 시스템은 Dennis 및 Manchester 모델의 Arbitor[2),6)]를 사용하지 않고 모든 PM들은 그것이 처리하여야할 Node Token들을 Local Memory인 NTM과 DTM에 보관한다. 따라서 각 module의 hardware 구조가 단순하여 지고 Arbitor에 의한 contention 및 transition delay time이 감소한다. 그러나 NL값의 할당과 조정 작업으로 인한 System software (Compiler 등)의 부담이 그만큼 증가한다.

현재까지 제안된 DFM[2)~8)]들은 대부분 신호처리, Weather Forecasting[9)] 등 특수용도(주로 대량 데이타의 고속처리)에 시험 적용시켜 보는 단계에 있다. 본 논문에서는 이미 제안한 DNL 1 모델에서 PM간의 비동기 실행이 가능하도록 하고 Conditional Node의 처리 장치를 설계하여 반복 루프를 실행하도록 하였으며 DFM의 1 모델로서 그 기본적 구조와 동작 원리, machine code의 생성 및 실행 과정 등을 검토하였다.

참 고 문 헌

1) A. L. Davis, "Data Flow Program Graphs", Computer, Vol. 15, Feb, 1982, pp. 26-41

2) J. B. Dennis, "Data Flow Computers", Computers, 13(11), 1980, pp. 48-56

3) A. L. Davis, "The Architecture and System Method of DDM1 : A Recursively Structured Data Driven Machine", ACM Proc. of 5th Ann. Symp. on Computer Architecture, 1978

4) A. Plas et al., "LAU System Architecture : A Parallel Data Driven Processor Based on Single Assignment", IEEE Proc., 1976, IC on parallel processing

5) J. E. Rumbaugh, "A Data Flow Multiprocessor", IEEE Trans.on Computers, 1977, C-26 (2), pp. 138-146

6) Watson, Gurd, "A Practical Data Flow Computer", Computer, Vol. 15, Feb, 1982, pp. 51-57

7) Vegdahl, "A Survey of Proposed Architectures for the Execution of Functional Languages", IEEE Trans. on Computers, C-33(12), Dec., 84, pp. 1064-1067

8) Sowa, Maruta, "A Data Flow Computer Architecture with Program and Token Memories", IEEE Trans, on Computer, Sept., C-31 (9) 1982, pp. 820-824

9) Dennis, "Modeling the weather with a Data Flow Super Computer", IEEE Trans . on Computers, C-33(7), Jul, 1984, pp. 592-603

10) 김원섭, 박희순, "Node Label에 의한 기본적 Data Flow Machine Model", 전기학회논문집 제34권 8호, 1985년 8월

11) A. L. Davis "A Data Flow Evaluation System Based on the Concept of Recursive Locality", AFIPS Proc. 1979 pp. 1076-1086