하계학술회의
논문85-8-9

DNL1에서 반복 루프의 장치화
Implementation of Iteration Loop in DNL1

金 元 燮 전북대 전기공학과

朴 照 淳 원광대 전산공학과

1. 서 론

Data Flow Machine (DFM)은 기존의 Control Driven 방식을 탈피하여 instruction에 필요한 operand가 도착되면 즉시 실행되는 Data Driven방식 병렬 처리 컴퓨터이다. DFM을 위한 프로그램으로서 Data Flow Graph (DFG)[1]가 사용되며 이는 instruction 또는 fuction을 나타내는 Node들과 이들사이의 데이타 흐름을 표시하는 Arc들로써 도시한다.

Node는 1) 그것의 모든 input arc에 필요한 operand가 전부 도착되었고 2) 동시에 그것의 output arc에 result token이 존재하지 않을때 그 Node는 firable 또는 enabled되었다고 말한다.

최근에 DFM에 대한 많은 연구[2-8]들이 진행되고 있으나 아직 실용화 단계에 있지 못하고 부분실현, 부분적용 및 시험과정에 있어 앞으로 많은 연구가 필요하리라 본다.

본 논문에서는 이미 제안된 A DFM Model with Node Labeling (DNL1)[9]에서 반복 Loop가 실행되게 하기 위하여 Conditional Node의 처리를 장치화하고 Processing Module (PM)사이에 완전 비동기 동작이 가능하도록 하는 개선된 PM구조에 대하여 설명한다.

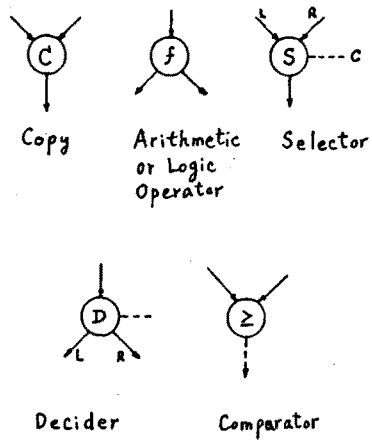

그림 1. Node의 종류

Fig. 1. Types of Nodes

2. 본 론

1) 반복루프를 갖는 DFG

DFG에서 사용되는 기본적인 Node의 표시는 그림 1과 같고 실선은 Data Token, 점선은 Control token ('True' or 'False')을 표시한다. 루프의 반복계산 과정을 설명하기 위하여 그림 2와 같은 샘플 프로그램[8]을 선택하며 이는 n! 계산에 관한 것이다.

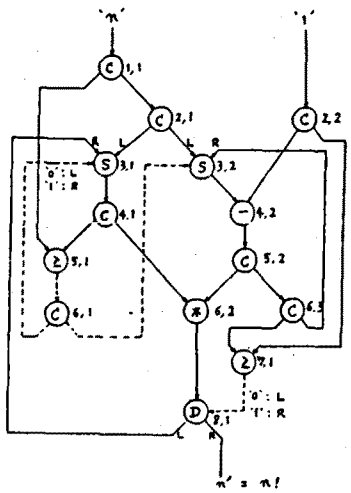

그림 2. 샘플 프로그램 (n!)

Fig. 2. Sample Program (n!)

그림에서 각 Node의 오른쪽 첨자는 Node Label[9]을 표시하고 모든 Node의 논리적주소(Logical Address)로 사용된다. 초기치 n값은 루프를 반복할때마다 1씩 감소하고 n(n-1)(n-2)…(n-i)계산을 실행하며 n-i=1일때 계산을 끝내고 n에 그 결과값을 보관한다. 여기서 S(3,1), S(3,2) 및 D(8.1)은 Coditional 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) Node Token

위의 샘플 DFG에서 Conditional Node를 제외한 모든 Node들에 대하여

NL f DNL1, [L/R/C] DNL2, [L/R/C][9]

L/R/C : Left, Right or Control Arc

와 같이 Node Token을 작성하고

Conditional Node중 Selector는

NL f DNL CF

CF : Conditional Flag ('0' or '1')

Decider는

NL f DNL1, [L/R/C] DNL2, [L/R/C] CF

와 같이 Coding한다.

이렇게 하여 작성된 Node Token은 표1과 같고 이는 DFG의 서술에 불과하다. 이를 다시 Parallel Level

표 1. DL별 Node Token

Table 1, Node Tokens in DL Sequence

1,1	COPY	5,1,L	2,1,L
2,1	COPY	3,1,L	3,2,L
2,2	COPY	4,2,R	7,1,R
3,1	SEL	4,1,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	3,2,C
6,2	MULT	8,1,L	
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 초기상태

Table 2. Initial state of Each PM Memory

	NTM				DTML	DTMR	CF	ETF L	ETF R	ETF C
PM1										
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
PM2										
2,2	COPY	4,2,R	7,1,R		*			0	1	1
3,2	SEL	4,2,L	0		*	*	*	0	0	0
4,2	SUBT	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
PM3										
6,3	COPY	7,1,L	3,2,R		*			0	1	1

(PL)별로 분류하고 각각의 Node Token Memory(NTM)에 Load시키면 표2와 같이 된다. 표에서 '*'표시는 해당 memory 위치에 해당 Node가 필요로하는 operand가 도착(삽입)될 memory space이고 최대 3종류의 operand 즉 Data Token Left(DTML),[8,9] Data Token Right(DTMR)[8,9]및 Control Flag(CF)가 필요하다. ETF(Enabled Token Flag)는 각각의 Node가 사용할 operand의 도착여부(present : '1', vacant : '0')를 나타내는 Flag로서 L/R/C/= '111'일때 해당Node가 Firable함을 나타낸다. 표2의 ETF값은 초기상태 값으로 Node function의 종류에 따라 각각 고유의 ETF code가 할당되어 있다.

3) PM구조

Conditional Node의 처리를 위해 개선된 PM의 구

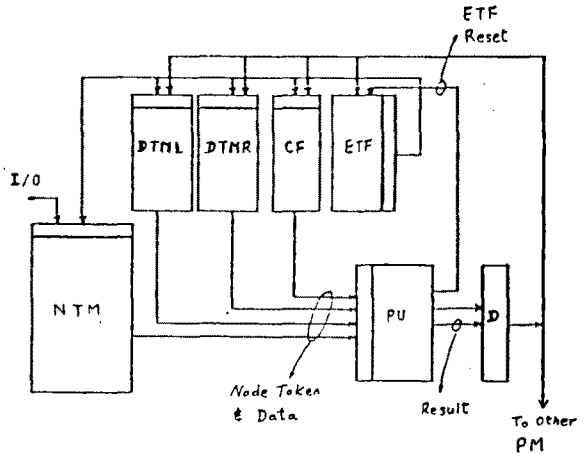

그림 3, PM의 내부 구조

Fig 3, Internal Structure of PM.

조는 그림 3과 같고 Control Flag (CF) memory와 Enabled Token Flag (ETF) Memory를 추가하여 PM간의 비동기 실행이 가능하도록 하였다. 기 발표한 DNL1에서 Location Counter (LC)를 제거하고 대신 Enabled Token의 addressing을 Associative memory인 ETF가 담당하게 한다. 즉 ETF의 LRC='111'인 Node가 발견(match)되면 해당 Node 및 데이타가 PU로 Fetch되고 실행후 결과를 해당 목적지에 Write하며 해당 ETF LRC값을 '1'로 Set한다. CF는 추가된 Control Token 보관용 memory이고 범용 RAM으로 구성한다.

4) 동작

샘플 프로그램의 실행과정은 다음과 같은 순서로 반복된다.

① 초기치 'n'과 '1'값이 해당목적지 DTML/R에 write되면서 ETF의 L/R/C Flag를 '1'로 set한다.

② 이순간 L/R/C값이 모두 '111'인 Node가 1,1 Copy와 2,2 Copy뿐이고 이들 두 Node가 enabled되었으므로 해당Node와 operand를 fetch하여 각각 PM1과 PM2의 PU에서 동시 실행된다.

③ ②의 PU에서의 실행과 동시에 fetch된 Node의 ETF값을 본래값(위의경우 '011')으로 reset시킨다. - input Arc를 vacant시켜 다음 enable이 가능하도록 함.

④ 실행된 결과(Result Token)를 Distributor가 받아 해당목적지 DTML/R 또는 CF에 Write하고 같은 위치의 LRC Flag를 set한다.

⑤ 위의 ②의 과정으로 반복한다.

3. 결 론

기 제안된 DNL1 모델중 PM간의 완전 비동기화를 가능하도록 하고 Conditional Node의 처리를 장치화(implement)하여 반복Loop를 실행하도록 하였다.

현재까지 제안된 DFM들은 대부분 Signal Processing, Weather Forecasting[10] 등 특수용도(주로 대량데이타의 고속처리)에 시험 적용시켜 보는 단계에 있어 범용 Supercomputer로서 광범위한 이용이 가능하게 되기까지는 아직 많은 연구들이 필요하리라 본다.

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] 김원섭, 박희순, "Node Label에 의한 기본적 DFM 모델", 전기학회 춘계 학술발표논문집, 1985, 5月, pp 12 - 14

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