'90 韓國自動制御學術會議論文集 1990. 10.26~27

Dataflow 구조에 기초한 PLC용 LSP 구현에 관한 연구

박 재현, 권욱현, 장 래혁

서울대학교 공과대학 제어계측공학과

A study on the implementation of dataflow LSP

Jaehyun Park, Wook Hyun Kwon, and Nae Hyuck Chang

Dept. of Control and Instrumentation Engineering

Seoul National University

ABSTRACT

In this paper, the architecture of a dataflow logic solving processor for programmable logic controller is proposed. As the proposed DFLSP(dataflow logic solving processor) is designed based on the dataflow architecture, it has inherently concurrent processing and data synchronization capabilities. The proposed DFLSP is adequate for high speed programmable logic controllers and gets rid of data synchronization problem in hardware level. The performance of the proposed DFLSP is analyzed using computer simulations and prototype hardware. With single processsing element, the logic solving time is 144 usec per 1K steps of logic program and with eight processing elements, the logic solving time is 23 usec per 1K steps of logic program with reasonable assumptions.

1. 서론

프로그래머블 로직 컨트롤러(programmable logic controller: PLC)는 공장 자동화에는 필수적인 자동화 기기로서 공장에서 사용되는 전기-기계적인 릴레이를 대체하는 기기이다. 오늘날 화학공장 및 제작공장등 거의 모든 공장에서의 자동화에는 PLC 가 사용된다. PLC는 제어 가능한 입출력 접점수에 따라서 소형, 중형, 대형등으로 분류될 수 있는데 소형 PLC인 경우는 입출력 접점수가 256 접점 이내의 것을 지칭하며 중형 PLC는 입출력 접점수가 2000 접점이내 그리고 대형은 그이상의 입출력 접점수를 가지는 것을 지칭한다. 공장이 대형화 하고 자동화 공정이 복잡해질수록 입출력 접점 수는많아지게되고 이에따라 대형 PLC가 많이 사용되게 된다. 대형 PLC는 제어가능한 입출력 접점수가 2000 접점이상 많게는 8000 접점 이상의 기능을 갖춘 기기들도 사용된다. PLC가 제기능을 발휘하기 위해서는 입력 접점으로부터의 신호에 대해 주어진 프로그램을 적용하여 출력 접점으로 데이타가 출력될 때 까지의 시간 즉 스캔타임(*scan time*)이 빨라야 하므로 스캔타임을 어느정도 수준으로 유지하면서 입출력 접점을 늘이기 위해서는 PLC가 특수한 구조로 설계되어야 하는 필요가 있다. 따라서 요즘 생산되는 대형 PLC에서는 스캔타임을 줄이기 위해서 여러가지의 특수한 구조가 사용되는데 그중 입출력만을 담당하는 IOP(Input Output Processor)와 제어 프로그램을 고속으로 처리하는 LSP의 사용이 대표적이다. IOP는 입출력 기기가 다른부분에 비하여 속도가 늦는점을 보완하기 위하여 병렬처리를 하기위하여 마련되는 프로세서이며 LSP는 PLC용 제어 프로그램이 일반적인 범용 프로그램과는 달리 단순한 기능의 동작을 반복하는 성질을 이용하여 이와같은 프로그램을 빠른 속도로 풀어나가기 위한 프로세서 이다. PLC의 제어 프로그램용 명령어는 AND, OR, STORE등의 단순 논리 명령과 COUNTER, TIMER 등의 복합 명령어로 나누어 질 수 있다. 이들 중 복합 명령어는 전체 프로그램의 25 % 이하를 차지하며 대

부분의 명령어들은 단순 논리 명령어 들이다. 따라서 단순 논리 명령어의 처리시간을 비약적으로 빨리 줄이는 LSP의 개발은 대형 PLC의 스캔타임을 줄이는데 큰 의미가 있다. 이에 대한 연구는 PLC의 제작사를 중심으로 비교적 많은 연구가 진행되어 왔다. 예로서 TI사의 PLC는 1K 개의 명령어를 처리하는데 200 usec의 시간이 걸리는 고속의 스캔타임을 가지고 있다.[1-5] 본 논문에서는 대형 PLC에 사용될 수 있는 고속의 LSP를 Dataflow 구조로 설계를 하고 이에대한 성능을 컴퓨터 시뮬레이션과 시제품 제작을 통하여 검토해 보았다. 제 2절에서는 PLC용 논리 프로그램과 Dataflow 구조의 기본 개념에 대하여 설명하고 3절에서는 DFLSP의 구조를 제안하고 4절에서 성능을 분석하였다.

2. 논리 명령과 Dataflow 구조

PLC용 Dataflow LSP의 구조에 해 논하기 전에 우선 PLC용 논리 명령과 Dataflow 구조에 대해 간략하게 설명하기로 하자. PLC는 기본적으로 기존의 기계적인 릴레이를 대체하는 목적으로 제작되었다. 따라서 PLC용 언어는 순차적 제어목적을 달성하기에 적합하게 작성되어 있으며 이들 언어중 가장 널리 사용되는 프로그램 언어는 래더언어(Ladder Diagram)이며 이는 그림 1과 같은 형태를 가진다. 래더언어를 처리하는 방법에는 행위주의 처리방법과 열위주의 방법으로 나누어 생각할 수 있으며 각 방법에 따라 처리 결과가 달라지는 경우가 발생할 수 있다. 따라서 원하는 결과를 얻기위해서는 사용하는 PLC의 처리 방법에 따라 프로그램 구성도 변경되어야 하는 경우가 발생한다. 따라서 래더언어에서 각 명령어의 상대인 위치에 상관없이 항상 같은 결과를 얻을수 있는 프로세 의 개발도 중요한 의미를 가진다. 1970 년대이후 컴퓨터의 성능을 향상시키기 위해서 여러가지 연구가 활발히 진행되어 왔으며 이런 연구는 여러가지 형태의 병렬처리 구조를 탄생시켰으며 Dataflow 구조도 이들 병렬처리 구조중 대표적인 것이다. Dataflow 구조의 컴퓨터는 MIT, Manchester 대학등에서 널리 연구되어 왔으며 기존의 컴퓨터가 제어 중심의 구조인 반면에 Dataflow 구조의 컴퓨터는 데이터 중심의 분산-병렬처리가 가능하다.[6-12] Dataflow 컴퓨터에서는 모든 동작이 데이터 중심으로 처리되므로 병렬처리 컴퓨터에서 문제가 되는 각 데이타의 동기화 문제등이 자동적으로 해결되는 장점이있다. 특히 이러한 성질을 PLC용 로직 프로그램에 응용할 경우 앞서 지적한 프로그램 처리 순서에 따른 문제점들은 자동적으로 해결할 수 있다. 이는 Dataflow 구조에서는 프로그램 처리 순서가 동작중에 자동적으로 결정되어 수행되기 때문이다. 또한 Dataflow 구조에서는 병렬처리가 근본적으로 구현되어 있기 문에 고속의 병렬처리 LSP를 구현하기가 용이하다.

3. DFLSP 의 구조

Dataflow 구조의 LSP (Dataflow LSP: DFLSP)는 Dataflow 구조의 컴퓨터가 가지는 병렬처리 기능과 데이타 동기화 기능을 살려 PLC용 논리 명령 프로그램인 래더언어를 데이타 발생 순서에 따라 고속으로 처리하는 고속 병렬처리 논리 명령 처리기 이다. LSP를 Dataflow 구조로 설계하면 다음과 같은 많은 장점을 얻을 수 있다. 첫째, Dataflow 구조가 가지는 병렬처리 기능을 이용하여 병렬처리 LSP를 구성하여 고속의 처리 속도를 얻을 수 있다.(fine grain concurrency) 둘째, 논리 명령의 처리 순서가 데이타의 발생 순서에 따라서 하드웨어에 의하여 자동적으로 결정되므로 프로그램의 작성방법에 상관없이 항상 정확한 결과를 얻을수 있다. 세째, 복합명령 처리기 및 입출력 제어기와의 병렬처리시 생기는 문제점을 극소화 시킬수 있으며 효율을 높일수 있다.

본 연구에서 개발된 DFLSP 는 크게 나누어 토큰큐(Token Queue), 프리매치 유닛(Pre-match Unit), BSU(Block solving unit), LSU(Logic solving unit), 그리고 메모리 업데이트 유닛(Memory update unit)의 5 개의 부분으로 나눌 수 있다.(그림 2 참조)

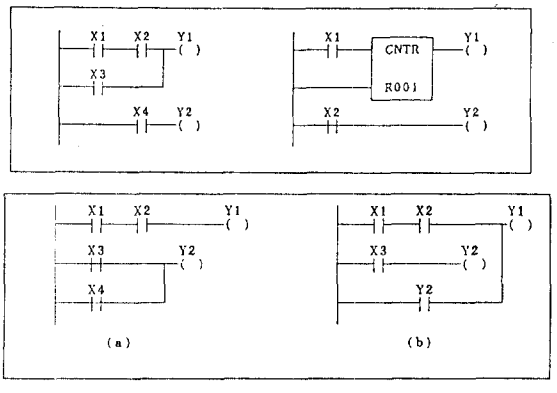

그림1 래더언어

3.1 토큰 큐(Token Queue)

DFLSP에서는 최대 8000개의 token을 가질 수 있으며 따라서 8000개의 PE가 존재하는 경우 최대의 동시성을 가질수 있다. 그러나 현실적으로 8000개의 PE를 구현하는 것을 불가능하므로 토큰 큐를 마련하여 PE가 배정되지 않은 token을 저장해 둔다. DFLSP 에서의 Token은 1 비트의 데이타(논리적 상태)와 16 비트의 NIP(Next Instruction Pointer)로 구성된다. DFLSP의 Token 큐는 최대 8000 개의 token을 저장할 수 있으며 FIFO의 구조를 가진다. Token 큐에 저장된 token은 프리매치유닛(pre-match unit)과 분배기(Distribution Network)을 통하여 쉬고있는 LSU에 자동적으로 배정된다. 또한 LSU가 문제를 푸는 과정에서 상대 token이 준비되지 않아서 더이상의 문제 풀이가 불가능한 경우 token은 다시 token 큐의 마지막에 저장된다.

3.2 프리매치 유닛(Pre-match Unit)

프리매치 유닛은 token 큐의 처음 위치의 token으로부터 DFLSP용 명령어를 만드는 기능을 수행한다. DFLSP의 명령어란 기본 토큰구조에 논리 연산 명령과 상대 토큰의 포인터등이 결합된 형태의 구조를 의미하며 이 형태의 명령어가 DFLSP의 LSU에서 기본 토큰대신 사용된다. 여기서 논리 연산 명령은 3비트로 구성되어 있으며 상대 토큰의 포인터는 15비트로 구성된다. 프리매치 유닛에서 형성된 명령어는 분배기(distribution network)을 통하여 동작을 마치고 대기상태에 있는 LSU에 분배된다. 프리매치 유닛과 분배기는 두단계의 파이프라인 구조로 구성되어 있으며 각각의 파이프라인은 비동기적으로 동작을 한다.

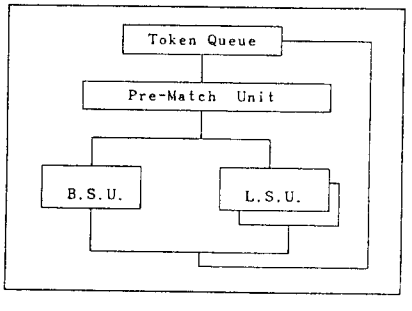

그림2 DFLSP 구조

3.3 LSU(Logic Solving Unit)

LSU는 DFLSP의 중심되는 부분으로 논리 명령을 푸는 역할을 한다. DFLSP에는 여러개의 LSU가 존재하여 병렬 처리가 가능하다. 각각의 LSU는 Intruction MUX, 프로그램 메모리, 데이타 메모리, 비교기(Matching Unit), FU(Functional nit)의 5개 작은 부분으로 나누어질 수 있다.(그림 3참조) Instruction MUX는 명령어의 경로를 분배기와 LSU자체 경로를 분리하여 적절한 명령어가 LSU에서 사용될 수 있게 해준다. LSU의 명령어는 token 큐에서나온 token이 프리매치유닛을 거쳐 형성된 명령어와 직전에 사용된 명령어가 풀리는 동안 그 다음 명령어가 만들어져 형성된 명령어의 두가지가 있다. PLC의 논리 명령어의 특성상 많은 명령어들이 래더 다이어그램의 순서에 따라 순차적으로 풀릴 수있기 때문에 이런 반복적 방법이 전체적인 성능을 높이는데 크게 기여한다. 대부분의 경우 FU로부터의 명령어가 사용되나 지정된 데이타(입출력 접점)에 토큰이 없는 경우나 래더 프로그램의 연속된 하나의 트리가 끝나는 경우 토큰큐로부터의 새로운 명령어가 사용된다. Instruction MUX로 부터의 새 명령어는 데이타 메모리를 참조하여 사용되는 입출력 접점에 토큰이 있는지 검사를 하여 비교기에 넘긴다. 비교기(Matching Unit)은 Instruction MUX로부터의 명령어가 수행되는데 필요한 데이타에 token이 있는지에 대한 검사를 하고 만일 token이 있는 경우에는 모든 데이타를 FU에 넘겨 계산을 진행하며 만일 필요한 데이타에 토큰이 없는 경우에는 Instruction MUX로 부터 새로운 명령어를 가져오게 하며 현재 사용중인 명령어는 토큰큐의 마지막 부분에 써넣게 된다. 데이타 메모리는 데이타 비트와 토큰유무를 지시하는 비트가 있으며 프로그램 메모리는 명령어, 데이타 포인트, 다음 명령 포인터등으로 구성된다. FU는 비교기로부터 비교되어 수행 가능한 명령어를 하드웨어적으로 푸는 곳으로 PLC에서 사용되는 논리 명령어인 AND, OR, NOT-AND, STORE 등의 명령어가 하드웨어적으로 처리된다. 이들 논리 명령어들 중 STORE 명령은 현재 계산 결과를 데이타 메모리에 저장하는 명령어로서 다른 모든 논리 명령어에 비하여 데이타 메모리를 수정해 주는 단계가 추가 된다. 데이타 메모리는 각각의 LSU내의 메모리와 다른 LSU내의 메모리가 있으므로 자체 데이타 메모리는 즉시 수정되며 다른 LSU의 메모리는 데이타 메모리 수정기(Memory update unit)에 의하여 수정된다.

3.4 메모리 수정기(Memory Update Unit)

DFLSP에는 여러개의 LSU가 동시에 사용될 수 있으며 각각은 각자 로컬메모리(Local Memory)를 가지고 있다. 데이타 메모리가 수정되는 경우는 DFLSP의 IOP가 입력 접점을 스캔하여 데이타 메모리에 저장하는 경우와 동작중에 STORE 명령을 수행하는 경우가 있다. 이중 입력접점을 스캔하여 저장하는 경우는 FU, 비교기(Matching unit)등이 동작을 하지 않으므로 문제가 발생하지 않으나 동작중에 메모리를 수정하는 경우는 메모리 사용에 시간 분할 방법을 사용하여야 한다. STORE 명령은 FU에 의하여 로컬 메모리는 즉시 수정이되며 동시에 메모리 수정기에 수정되어야할 메모리의 포인터와 데이타가 등록된다. 메모리 수정기는 등록된 메모리 수정요구를 각 LSU의 동작을 중간에서 가로채어 수정한다. 따라서 이경우 각 LSU는 한번의 메모리 싸이클에 해당하는 시간만큼 지연된다. 또한 여러개의 LSU가 동시에 메모리 수정을 요구하고 또한 메모리 수정 요구가 연속되는 경우 메모리 수정기는 메모리 수정을 차례대로 하기 때문에 더이상의 메모리 수정 요구를 받지않고 지연시키는 경우가 있다. 이 경우에는 수정요구를 원하는 LSU는 메모리 수정기의 앞선 요구가 해제될 때까지 지연된다.

4. DFLSP의 성능 평가

DFLSP가 논리 명령 연산기로서의 성능을 평가하기 위하여 성능평가의 척도로 논리 명령 연산 속도를 생각해 보자. DFLSP의 성능을 분석하기 위하여 우선 대상 PLC 프로그램에 관한 일반적인 가정을 하기로 하자. DFLSP가 적용될 수 있는 PLC는 대형 PLC를 가정하며 이경우 입출력 접점수는 각각 8000 접점, 그리고 최대 프로그램 크기는 64000 스텝으로 가정하자. 이경우 STORE 명령이 차지할 수 있는 최대 비율은 12.5%가 된다. 또한 하나의 출력 접점을 약 3.5회 정도 다시 참조한다면 전체 프로그램의 약 43.8%의 논리 명령이 출력 접점을 참조한다는 결론이 된다. 그리고 이들 43.8%의 논리 명령이 참조하는 출력 접점중 50%의 출력 접점이 참조되기전에 이미 준비되었다고 가정하면 최대 21.9%의 명령에 대하여 비교기에서 대기 상태가 발생한다.

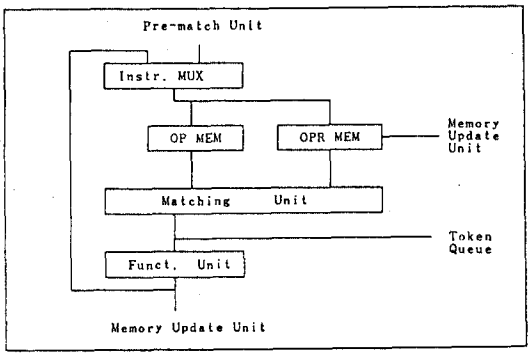

그림3 LSP 내부구조

이상의 가정을 바탕으로 컴퓨터 시뮬레이션을 통하여 DFLSP의 성능을 분석해 보면 하나의 LSU를 가진 DFLSP는 STORE 명령의 포함 비율에 따라 107 nSEC에서 175 nSEC의 성능을 가진다. 그리고 8개의 LSU를 가지는 경우 STORE 명령의 비율에 따라 13.48 nSEC 에서 33.52 nSEC의 성능을 보인다. 이것은 하나의 LSU를 이용하는 경우 6.94 MIPS, 그리고 8개의 LSU를 이용하는 경우는 43.47 MIPS의 성능과 같다. 컴퓨터 시뮬레이션 결과는 그림 4의 그래프와 같다.

5. DFSLP의 시제품 하드웨어 구현

DFLSP의 시제품을 PGA(Programmable Gate Array)를 이용하여 구현해 보았다. 시제품은 모토롤라 68030 보드를 주 제어 장치로 하고 입출력 장치, DFLSP의 3부분으로 구성되어있다. 시제품에 구현된 LSP의 내부에는 2개의 LSU가 있다. 전체는 6 단계의 파이프라인 구조로 구성되어 있는데 토큰 큐 관리와 프리매치 유닛에 2단계, LSU 내부에 3단계, 그리고 메모리 수정기(Memory Update Unit)에 1 단계의 파이프라인이 구성되어 있다. 각각의 파이프라인 단계는 동기식이 아닌 비동기식으로 동작된다. 시제품에는 느린 속도의 메모리(120 nSEC)를 사용하여 전체적인 속도는 1개의 LSU가 1개의 명령을 푸는데 걸리는 시간이 180 nSEC가 걸리며 따라서 전체적인 속도는 90 - 100 nSEC 가 걸린다. 시제품의 사진은 그림 5와 같다.

6. 결 론

본 논문에서는 dataflow 구조에 기반을 둔 PLC(Programmable Logic Controller)에 사용되는 LSP(Logic

Solving Processor)를 제안 하였다. DFLSP(Dataflow Logic Solving Processor)는 dataflow의 기본 구조를 가지고 있으므로 자체적으로 병렬 처리 기능을 가지고 있으며 PLC용 프로그램인 래더 프로그램의 입출력 사이의 데이타 동기화 문제가 자동으로 해결되는 장점등을 가지고 있다.

DFLSP는 토큰큐, 프리매치 유닛, BSU, LSU, MUU등으로 구성되어 있으며 각각은 비동기적으로 병렬처리 된다. DFLSP의 성능은 컴퓨터 모의 실험 및 시제품 제작을 통하여 검토해본 결과 1000개의 명령어를 푸는데 걸리는 시간은 1개의 LSU를 가지는 경우 144 uSEC이며 8개의 LSU를 가지는 경우 23 uSEC이다. 이것은 1개의 LSU인 경우 6.9 MIPS의 성능이며 8개의 LSU인 경우 43.5 MIPS의 성능을 발휘한다고 말할 수 있다.

REFERENCES

[1] TI Inc., Texas Instruments Industrial Control Products: Model 560/565 Product Profile, Texas Instrument.

[2] GOULD Inc., The MODICON 584 PC User's Manual, Gould, Jan. 1980.

[3] Wook Hyun Kwon, Jong-il Kim and Jaehyun Park, "Architecture of a Ladder Solving Processor(LSP) for Programmable controllers", Submitted to IE.

[4] Jong-il Kim, "The Architecture of Logic Solving Processor(LSP)," Proceedings of 88 KACC.

[5] Jong-il Kim, Wook Hyun Kim, and Jaehyun Park, "Architecture of a Logic Solving Processor for Programmable Controllers", Proceedings of 88 SICE.

[6] Kung-Song Weng, "An Abstract Implementation for a Generalized Data Flow Language," Ph.D thesis, MIT University, 1979.

[7] William B. Ackerman and Jack B. Dennis, "VAL -- A Value-Oriented Algorithmic Language: Preliminary reference manual," June, 1979.

[8] J. R. Gurd, C. C. Kirkham, and I. Watson, "The Manchester Prototype Dataflow Computer," Communications of the ACM, Jan. 1985.

[9] Hiroaki Nishikawa and Hiroaki Terada, "Architecture of a One-chip Data-driven Processor : Q-p," Proceedings of Internaitonal Conference on Parallel Processing, 1987.

[10] Shinji Kormori, Kenji Shima, souichi Miyata, and Hiroaki terada, "The Data-Driven Microprocessor," IEEE Micro, June, 1989.

[11] James Rumbaugh, "A dataflow Multiprocessor," IEEE Tr. on Computers, Vol. C-26, No. 2, Feb. 1977.

[12] Kim P. Gostelow and Robert E. Thomas, "Performance of a simulated dataflow computer," IEEE Tr. on computers, Vol. C-29, No. 10, Oct. 1980.

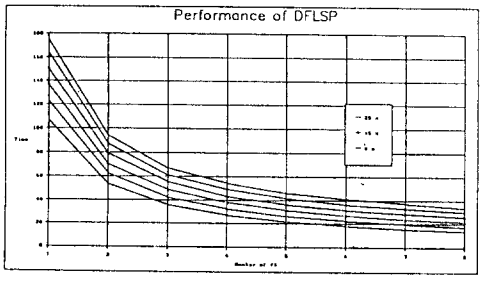

그림4 컴퓨터 모의실험결과

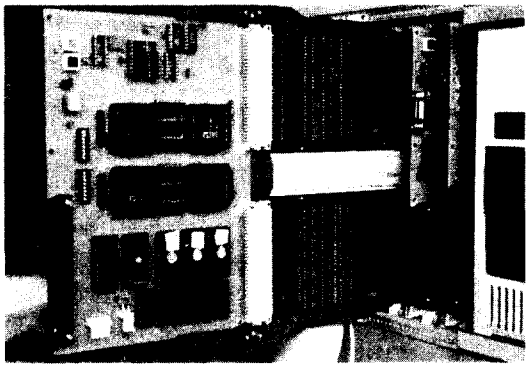

그림5 시제품 테스트 사진