재비교기를 이용한 PLC용 Dataflow LSP 구조에 관한 연구

박재현, 장래혁, 권욱현
서울대학교 제어계측공학과

A Study on the Architecture of Dataflow LSP using Re-matching Unit

Park Jaehyun, Chang Naehyuck, and Kwon Wook Hyun
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. And also, it has dynamic load balancing capabilites which increases the utilization of the whole system that can be hardly implemented in other multiprocessor system. The re-matching unit gets rid of unnecessary matching cycles in LSU, which increases the performance of LSU and allows the multiple input mutiple output operations.

1. 서론

PLC는 공장 자동화에서는 필수적인 기기로서 기계적인 릴레이를 대체하는데 사용이 된다. PLC의 주된 기능은 데이타를 수집하고 이를 연산하여 다시 출력하는 것이다. PLC는 처리 용량에 따라 크게 나누어 소형, 중형, 대형 PLC로 나눌 수 있으며 대형 PLC는 수천점의 입출력 접점을 고속으로 처리하여야 한다. 따라서 고성능의 대용량 PLC를 구현하려면 빠른 입출력 프로세서와 연산 프로세서를 사용하여야 한다. 빠른 입출력 프로세서는 연산 프로세서와 동시 작업을 통하여 전체시시템의 고속화를 얻을 수 있다. 그러나 결국 제어프로그램을 푸는 연산 프로세서의 성능이 전체 기기의 성능을 좌우하게 되며 따라서 고속의 연산 프로세서를 구현하기위한 많은 연구가 행해지고 있다.[1-4] PLC용 제어 언어는 크게 나누어 로직 명령과 복잡한 튜터 명령으로 나누어 생각할 수 있다. 이중 로직 명령은 각 접점 사이의 AND, OR 등의 단순한 작업을 하는 것이며 튜터 명령은 타이머, 카운터등의 동작을 표현하는 명령어 이다. 그런데 PLC용 제어 언어의 대부분은 로직 명령이 차지하고 있으므로 PLC용 로직 명령를 빠르게 처리하는 프로세서(LSP; Logic Solving Processor)의 사용은 PLC의 연산 프로세서의 성능을 증대시키느데 도움이 된다. PLC용 LSP를 구현하는데는 여러가지 종류의 멀티프로세서 구조가 사용될 수 있는데 Array Processor등을 이용한 LSP가 많이 연구 되어 왔다. 본 논문의 선행 연구에서는 PLC의 로직 명령의 특성을 이용하여 dataflow 구조를 가진 PLC용 LSP를 제안하였다.[4] 당시 논문에서 제시한 구조는 dataflow 컴퓨터 구조중 Manchester Machine에 근거한 구조로서 Manchester machine의 구조를 일부 변형하여 PLC용 로직 명령을 고속으로 처리할 수 있도록 하였다. 따라서 dataflow 구조에서 얻을 수 있는 데이타 동기화 및 로드 밸런싱의 장점을 하드웨어적으로 보장받는다. 그러나 당시의 연구는 로직 명령에 국한되어 있어 블록 형태의 명령이 추가되면 이를 범용 CPU등에서 처리함으로 속도의 지연을 많이 받게 된다. 본 논문에서 제시하는 DFLSP는 기존에 제시한 DFLSP의 기본 구조에 블록 명령을 처리할 수 있는 ASU(Arithmetic Solving Unit)을 추가하여 복잡한 블록 명령어도 고속으로 처리할 수 있을뿐더러 dataflow 컴퓨터의 특성상 일반 LSP에서 발생한는 블록 명령에 의한 시간 지연 현상을 극소화 시킬 수 있는 구조을 제시한다. 또한 재비교기(Re-matching unit)을 두어 여러개의 토큰을 처리할 수 있는 형태의 명령어(특히 OR 명령)를 처리하는데 성능향상을 가져올 수 있는 구조을 제시 하고 이에 대한 컴퓨터 시뮬레이션을 통하여 성능을 분석 하였다.

본 논문의 구성은 2 장에서 PLC용 로직 프로그램과 dataflow graph와의 상호 연관 관계에 대하여 설명을 하고 3 장에서는 dataflow LSP에 구조를 제시한다. 제 4 장에서는 3 장에서 제시한 dataflow LSP의 성능을 컴퓨터 시뮬레이션을 통하여 모의 시험하여 결과를 제시한다.

2. LSP용 로직 프로그램과 dataflow graph

원래 PLC는 공장에서 전기-기계적인 릴레이를 대체하기 위하여 사용하기 시작했기 때문에 현재 사용중인 PLC용 명령어들도 순차적인 제어에 적합하도록 설계되어 있다. 현재 사용되는 여러가지 PLC용 명령어중에서 가장 널리 사용되는 명령은 래더 명령어로 그림 1-a와 같은 형태를 가진다. 래더 명령어는 로직 명령어와 블럭 명령어로 나눌 수 있는데 로직 명령어는 AND, OR, NOT-AND, STORE 등의 명령어를 포함하고 있으며 블럭 명령어는 COUNTER, TIMER, SHIFT REGISTER, DRUM 등을 의미한다. 실제 프로그램에 있어서는 대부분의 명령어들은 로직 명령어들이며 블럭 명령어는 전체 프로그램의 20%이내를 차지하게 된다.

래더언어를 처리하는 방법에는 행위주의 방법과 열위주의 방법으로 나누어 생각할 수 있으며, 각 방법에 따라 처리 결과가 달라지는 경우가 발생할 수 있고, 원하는 결과를 얻기위해서는 사용 PLC의 처리 방법에 따라 프로그램 구성도 변경되어야 한다. 따라서 래더언어에서 각 명령어의 상대적인 위치에 상관없이 항상 같은 결과를 얻을 수 있는 프로세서의 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 예이다. Dataflow 컴퓨터에서는 모든 동작이 데이터 중심으로 처리되므로 병렬처리 컴퓨터에서 문제가 되는 각 데이타의 동기화 문제등이 자동적으로 해결되는 장점이 있다. 특히 이러한 성질을 PLC용 로직 프로그램에 응용할 경우 앞서서 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 때문에 고속의 병렬처리 LSP를 구현하기가 용이하다.

래더 프로그램은 2개의 입력을 가진 방향성 그래프(2 input directed graph)로 변환이 가능하다. 2입력 방향성 그래프란 2개의 노드가 방향성이 있는 branch로 연결이 되어 있고 각각의 노드는 2개의 입력을 가지고 있는 그래프이다. 이런 방법을 이용하면 그림 1-a의 래더 프로그램은 그림 1-b의 그림으로 변환이 가능하다. 그림 1-b와 같은 2 입력 방향성 그래프가 dataflow 그래프의 기본 명령어가 된다. dataflow 컴퓨터의 기본 명령어로서 dataflow 그래프는 link와 actor의 두가지 종류의 노드로 분류될 수 있다. link는 각 actor와 actor사이에 token의 유, 무를 표시해주는 token의 임시 저장장소이며 actor는 실제로 오퍼레이션이 행해지는 곳을 의미한다. 일반적으로 dataflow 컴퓨터에서 하나의 actor가 동작을 하기 위해서는 모든 입력 link에 token이 존재해야하며 어떠한 출력 link에도 token이 존재해서는 안되다. 이런 조건을 만족 하는 모든 actor는 동작 가능하며 이런 방법을 통하여 각 actor사이에 데이타 동기화 및 병렬처리가 보장받는다.

3. Dataflow LSP 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 처리기이다. LSP를 Dataflow 구조로 설계하면 다음과 같은 많은 장점을 얻을 수 있다. 첫째, Dataflow 구조가 가지는 병렬처리 기능을 이용하여 병렬처리 LSP를 구성하여 고속의 처리속도를 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 얻을 수 있다. 세째, 복합명령 처리기 및 입출력 제어기와의 병렬처리시 생기는 문제점을 극소화 시킬 수 있으며 효율을 높일 수 있다.

본 논문에서 제안하는 DFLSP 는 그림 2에서 볼 수 있듯이 크게 나누어 토큰큐(Token Queue), 분배기(Distribution Network), LSU(Logic solving unit), ASU(Arithemetic Solving Unit), 그리고 재비교기(Re-matching unit)의 5 개의 부분으로 나눌 수 있다.

3.1 토큰 큐(Token Queue)

DFLSP에서는 최대 8000개의 token을 가질 수 있으며 따라서 8000개의 PE가 존재하는 경우 최대의 동시성을 가질 수 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 분배기(Distribution Network)을 통하여 쉬고있는 LSU에 자동적으로 배정된다. 또한 LSU가 문제를 푸는 과정에서 상대 token이 준비되지 않아서 더이상의 문제 풀이가 불가능한 경우 token은 재비교기(Re-matching unit)을 거쳐 다시 token 큐의 마지막에 저장된다.

3.2 분배기(Distribution Network)

분배기는 token 큐의 처음 위치의 token을 가져와 적당한 LSU 또는 ASU에 분배하는 역할을 맡고 있다. 제안된 DFLSP에서는 복수개의 LSU와 ASU를 가질 수 있다. 따라서 전체 시스템의 최대의 효율을 얻기 위해서는 분배기가

Token을 각각의 LSU 또는 ASU가 현재의 동작을 마치게 되면 즉시 다시 배분하여 주어야 한다.

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 Unit)의 5개 작은 부분으로 나누어질 수 있다. 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, 필요한 데이타에 토큰이 없는 경우에는 Instruction MUX로 부터 새로운 명령어를 가져오게 한다. 현재 사용중인 명령어는 재비교기(re-matching unit)으로 보내져 상대편 토큰이 형성되기를 기다린후 다시 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, STORE등의 명령어가 하드웨어적으로 처리된다. 이들 논리 명령어들중 STORE 명령은 현재 계산 결과를 데이타 메모리에 저장하는 명령어로서 다른 모든 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 수정 토큰(data memory update token)이 생성되어 다른 LSU에 보내지게 되고 다른 LSU에 의하여 수정된다.

3.4 ASU(Arithmetic Solving Unit)

PLC의 명령어중 블럭 명령어들은 LSU에서 처리되지 않고 ASU에서 처리된다. 본 DFLSP에서의 ASU는 LSU에서 사용되는 토큰에 의해 시작되는 산술 연산 프로세서이다. 산술 연산의 결과는 로컬 데이타 메모리에 저장되며 출력 토큰은 스테이터스 프래그가 된다. 논리 연산의 경우와는 다르게 산술 연산시 걸리는 시간은 메모리 참조 시간보다 월등히 오래 걸리므로 Manchester Machine의 경우와 같은 원리로 병렬처리시 속도 증가를 가져 올 수 있다. ASU는 단지 산술 연산만을 하는 프로세서 이므로 블록 명령을 수행 하려면 LSU와 함께 사용하여하 하는데 그림 3은 타이머의 경우 블럭 명령의 구현 예이다.

3.5 재비교기(Re-matching Unit)

DFLSP는 dataflow 컴퓨터를 PLC의 특성을 고려하여 변형시킴으로서 극대의 효율을 피하고 있다. 따라서 비교기(matching unit)에서 실패하는 경우를 최소한으로 하여야한다. 사실 LSU에서 로직 연산을 한번 하는 시간과 비교기에서 하나의 토큰을 비교하는데 걸리는 시간이 같기 때문에 비교기에서 실패을 하여 지연되는 시간은 전체 성능에 많은 손해를 미치게 된다. 재비교기는 각각의 LSU및 ASU로부터 실패한 토큰을 저장하고 이후 이들 명령이 사용 가능한 조건을 만족하게 되면 토큰큐로 이들 명령어를 보내어 사용이 될 수 있도록 한다. 이와 같이 재비교기를 둠으로써 LSU내의 비교기의 실패율을 현저하게 감소시켜 성능 향상을 얻을 수 있다.

4. 컴퓨터 모의 실험

DFLSP는 fine-grain MIMD기계로서 각각의 프로세싱 엘먼트인 LSU나 ASU상호 간의 통신은 토큰큐나 재비교기(Re-matching unit)를 통하여 이루어지며, 이 두 곳에서 병목 현상이 발생하게 된다. 이들 병목 현상이 없는 경우 DFLSP의 최대 성능은 보장 받으며 이경우 연산 속도는 LSU의 갯수에 비례하여 늘어나게 된다. 그리고 ASU는 LSU와 별도로 동시에 연산을 수행하므로 블럭 명령이 전체 명령의 20% 이내를 차지하고 있음을 고려해 볼때 블럭 명령을 계산하는 시간은 LSU가 로직 명령을 연산하는 시간에 가려져 전체 성능에 미치는 영향이 거의 나타나지 않을 정도로 ASU 갯수를 조정할 수 있다. 이런 바탕위에 DFLSP의 성능을 SIMAN 프로그램을 이용하여 모의실험을 하였다. DFLSP의 구조상 전체 시스템의 성능에 영향을 미치는 요소는 store 명령의 갯수로 store 명령의 갯수에 따라 x-refer 갯수와 y-refer 갯수가 변하게 된다. 이경우

store 명령 갯수 : S

X-refer 갯수 : 1/3 - S

Y-refer 갯수 : 2/3 - S/2

와 같이 계산되며 S의 값은 0 < S < 2/3의 범위내에서 변하게 된다. 이때 DFLSP의 성능을 보면 그림 4와 같다. LSU를 한개 사용한 경우 성능은 200nSec/Operation이며 4개의 LSU를 사용하는 경우 성능은 80nSec/Operation 이된다. 이 경우 병렬처리에 의한 성능향상은 2.5배이며 효율은 0.625가 된다. 그러나 ASU에의한 블럭 명령에 대한 성능향상을 고려하면 20% 정도의 효율이 향상된다.

5. 결론

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.

제안된 DFLSP는 토큰큐, 분배기, LSU, ASU, 재비교기 등으로 구성되어 있으며 병렬 처리 된다. DFLSP의 성능은 컴퓨터 모의 실험을 통하여 검토해본 결과 1개의 명령어를 푸는데 걸리는 시간은 1개의 LSU를 가지는 경우 200 nSEC이며 4개의 LSU를 가지는 경우 80 nSEC이다.

참고 문헌

[1] Wook Hyun Kwon, Jong-il Kim and Jaehyun Park, "Architecture of a Ladder Solving Processor(LSP) for Programmable controllers", Proceedings of 91 IECON

[2] Jong-il Kim and Wook Hyun Kwon "The Architecture of Logic Solving Processor(LSP)," Proceedings of 88 KACC.

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

[4] 박재현, 권욱현, 장래혁, "Dataflow 구조에 기초한 PLC용 LSP 구현에 관한 연구", 1990 한국자동제어학술회의 논문집.

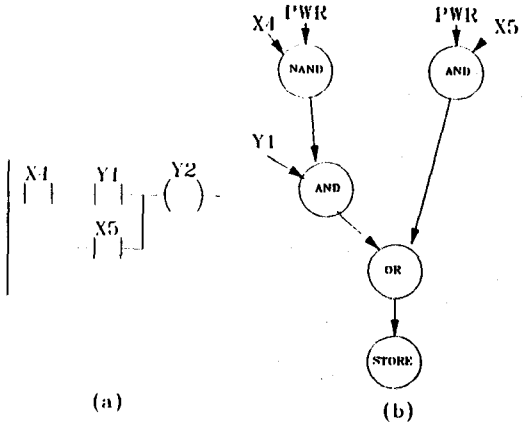

그림1. PLC용 논리 프로그램

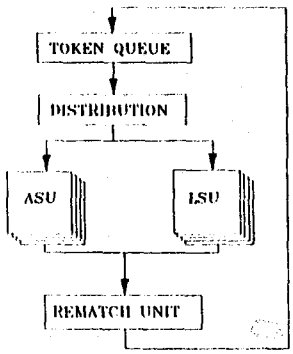

그림2. DFPLC 블럭 다이아 그램

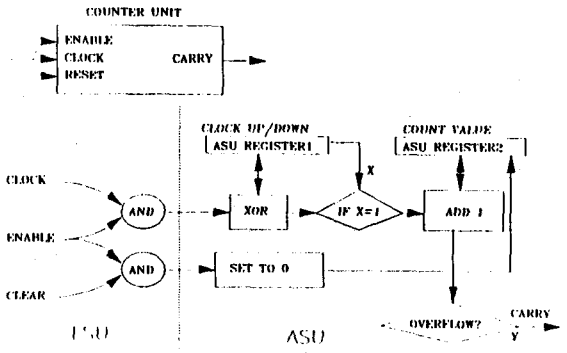

그림3. 카이머 블럭 명령

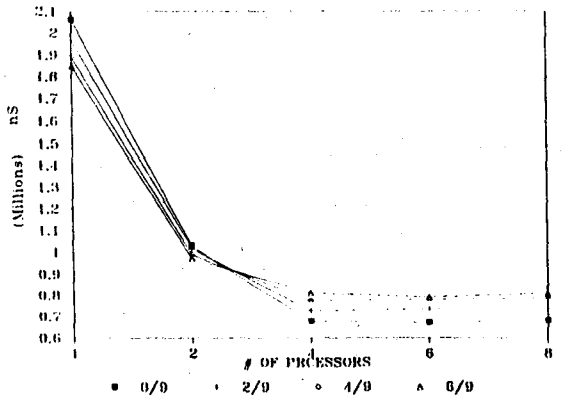

그림4. 컴퓨터 모의실험 결과