'93 한국자동제어학술회의논문집 (1993. 10. 20~22)

PLC용 RISC 프로세서의 구조와 명령어에 관한 연구

° 구경훈, 박재현, 노갑선, 장래혁, 권욱현
서울대학교 공과대학 제어계측공학과

A Study on the Architecture and Instruction of a RISC Processor for Programmable Logic Controller

° Kyunghoon Koo, Jaehyun Park, Gab Seon Rho, Naehyuck Chang and Wook Hyun Kwon
Department of Control and Instrumentation Engineering, Seoul National University

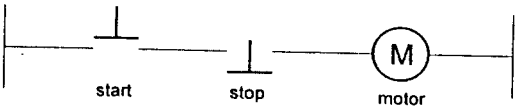
ABSTRACT

In this paper, the instruction set and the architecture of a RISC processor for programmable logic controller is suggested. From the measurement of existing programs, the characteristics of ladder instructions are analyzed. The instruction set is defined so that the existing ladder program can be reused with simple translation. Because bit instructions controls the behavior of word instructions, the processor suits for high level language like SFC. Simulations show that the PLC with the suggested processor is twenty times faster than the PLC with the multi-purpose microprocessor.

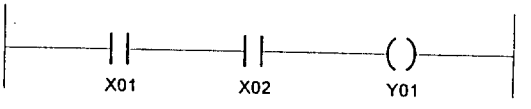
1. 서론

공장 자동화의 핵심 기기로 널리 사용되는 PLC(Programmable Logic Controller)는 과거 릴레이를 이용한 작업 기기의 순서 제어를 대체하기 위한 것이다[2]. 자동화가 각 단위 공정에서 점차 공장 전체로 발전함에 따라 PLC는 자동화 시스템의 중간 단계로 호스트 컴퓨터와 작업 기기를 연계하는 역할을 하면서 고속화, 대형화, 다기능화 추세에 놓여 있다[2][5]. 그림 1과 같이 릴레이 제어 회로를 그대로 도표화한 래더 언어를 실행하기 위해 PLC는 모든 입력 접점의 값을 읽은 후 프로그램을 수행하고 출력 접점에 값을 내보내는 과정을 반복한다. 이 과정에서 걸리는 시간을 스캔 주기라 하며 입력 접점의 변화가 출력 접점으로 전달되는 반응 시간을 결정한다. PLC의 고속화, 대형화란 많은 입출력점을 짧은 스캔 주기 내에 처리하는 것을 말한다.

PLC의 명령에는 릴레이 제어 회로를 대체하기 위한 기본 명령(비트 명령) 뿐 아니라 프로그램의 흐름을 제어하고 블록 연산을 하기 위한 응용 명령(워드 명령)이 포함된다. SFC와 같은 차세대 고급 언어를 지원하고 다양한 기능을 갖



(a) relay circuit



(b) ladder diagram

```
STR   X01        LD    X01
ANDN  X02        ANDI  X02
OUT   Y01        OUT   Y01
```

(c) instruction list

그림 1. 릴레이 제어 회로와 래더 언어

추기 위해 최근의 PLC는 100 종 이상의 다양한 응용 명령을 보유하게 되었고 프로그램에서 응용 명령이 차지하는 비중도 점차 높아지고 있다[5].

기존 PLC의 구조는 중앙 처리 장치에 상용 마이크로 프로세서를 채택하였기 때문에 범용 컴퓨터의 구조에서 벗어나지 않고 그 수행 방식은 래더 명령을 하나씩 읽고 해석하여 미리 작성된 서브루틴으로 처리하는 것이다. 이는 CISC 형태의 마이크로 프로세서가 명령어를 읽고 해석하여 마이크로프로그램으로 처리하는 것과 같다. 이러한 방식의 PLC에 있어서 특히 문제가 되는 점은 PLC에만 필요한 고유의 응용 명령의 경우 그 수행 속도가 크게 저하된다는 점이다. 그러므로 고속화, 대형화, 다기능화를 위해서는 PLC 고유의 작업 처리에 알맞는 전용 구조가 필요한 것이다.

기존의 상용 프로세서를 사용하는 PLC의 단점을 극복하

기 위해 여러가지 새로운 구조의 PLC가 연구되고 있다. 이를 크게 두가지로 나누어 보면 그 첫번째 구조는 래더 명령 중에서 기본 명령을 처리하기 위한 전용 프로세서와 응용 명령을 처리하는 상용 프로세서를 사용하는 멀티 프로세서 구조이다[3]. 이 구조는 각각의 명령을 위해 프로세서의 구조를 최대한 적용시킬 수 있지만 기본 명령과 응용 명령의 유기적인 연결이 힘들다는 단점이 있고 응용 명령의 비중이 커지는 추세에 있어서 기본 명령만을 빠르게 하는 것은 전체 시스템의 성능 향상과는 약간의 거리가 있다. 이 구조의 대표적 연구로는 래더 언어를 사용하지 않고 페트리 넷이나 데이타 플로우 그래프와 같은 새로운 언어를 사용하는 전용 프로세서 연구이다[4][7][8][9]. 이 방법은 병렬 처리 구조로 만들기가 쉽다는 장점이 있지만 기존의 래더 언어를 변환해야 하는 문제를 가지고 있다. 두번째 구조는 기본 명령과 응용 명령을 모두 처리하는 단일 전용 프로세서 구조로 기본 명령과 응용 명령이 하나의 프로세서에서 처리되므로 보다 효율적인 구조라고 말할 수 있다. 신호 처리, 영상 처리, 그래픽 등의 여러 분야에서 전용 프로세서로 채택되어 개발되고 있는 RISC 형태의 전용 프로세서가 이 구조에서 채택되고 있다[1][6][5]. Y. Shimokawa 등이 제안한 전용 프로세서는 RISC 형태의 전용 프로세서로 PID 명령과 같은 고급 응용 명령을 포함하지만 프로그램의 흐름을 제어하는 응용 명령을 포함하지 않아 SFC와 같은 고급 언어 프로그램을 효율적으로 수행하기 어렵고 메모리 억세스가 Load/Store 방식이 아니기 때문에 다양한 응용 명령을 갖추지 못하고 있다[10].

본 논문에서는 RISC 형태의 전용 프로세서의 구조를 제시하고 그 명령어 세트를 정의하였다. 이를 위해 실제로 사용되는 래더 프로그램에서 각 명령의 사용 빈도와 오퍼랜드의 종류, 서브루틴의 깊이 등이 분석되었다. 제안된 구조는 명령과 데이타를 위한 버스가 나누어진 하바드 구조를 가지고 있으며 Load/Store 방식으로 메모리를 억세스하며 모든 명령이 4단계 파이프라인 처리가 된다. PLC가 필요로 하는 다양한 응용 명령을 포함하고 기본 명령의 계산 결과가 응용 명령의 수행을 제어하게 함으로써 차세대 PLC 언어인 SFC의 고속 처리가 가능하도록 하였다. 기존에 작성되어 사용하는 프로그램을 사용할 수 있도록 변환 프로그램이 작성되었으며 시뮬레이션을 통해 성능을 비교하였다.

2. 래더 언어와 프로그램 분석

2.1 래더 언어의 분석

래더 언어는 기본 명령과 응용 명령으로 이루어져 있다. 기본 명령은 1개의 비트 오퍼랜드를 가지고 있고 그 종류는 20종 내외이다. 기본 명령의 오퍼랜드는 입출력 접점, 내부 접점, 연산 상태, 카운터와 타이머의 상태를 나타낸다. 기본 명령의 수행 방식은 계산 결과만이 남는 스택 방식이다. 기본 명령의 계산 결과는 응용 명령의 수행 여부를 결정한다.

응용 명령은 범용 마이크로프로세서의 명령과 그 형태가 같고 프로그램의 플로우 콘트롤과 비트 접점의 블록 연산 등에 이용된다. 응용 명령의 수는 제조 회사에 따라 많이 달라지는데 대개 100종 내외이다. 이중 PLC에만 필요한 응용 명령은 약 10종 내외이다.

래더 언어에 사용되는 어드레싱 모드는 Absolute 어드레싱과 Immediate 방식만을 사용하고 있으며 응용 명령의 경우 두가지 어드레싱 방식이 혼합되어 사용되기도 한다. 루프 변수와 서브루틴 인수 등도 모두 메모리에 정의된 변수를 사용하므로 전용 프로세서로 수행시킬 경우 필요한 레지스터의 수는 매우 적다. 래더 명령은 기본 명령과 같이 한번의 메모리 억세스가 필요한 명령에서부터 블럭 전송과 같이 여러번의 메모리 억세스가 필요한 명령들을 포함한다. 그러므로 제안하는 전용 프로세서는 Load/Store 구조의 RISC 형태로 각 명령의 사용 빈도에 따라 빈도가 높은 명령들을 내장하고 빈도가 낮은 명령들은 몇개의 명령으로 대체한다.

래더 프로그램은 스캔 방식에 의해 수행된다. 그리고 한 스캔 내에서 프로그램의 대부분의 명령들은 한번씩만 수행된다. 이는 기존 상용 프로세서의 명령 캐쉬의 필요성이 없음을 뜻한다.

2.2 래더 명령의 빈도 분석

현장에서 사용되는 래더 프로그램에서 각 명령의 실행 빈도를 조사하였다. PLC가 소형인지 대형인지에 따라 응용 명령의 사용 빈도는 크게 달라지므로 본 조사에서는 중형 이상의 PLC를 사용하는 프로그램을 대상으로 하였다. 표 1에서 보는 바와 같이 래더 프로그램의 60~80 퍼센트가 20종 내외의 기본 명령이고 나머지가 응용 명령이었다. 이는 명령어 선정에 있어서 기본 명령을 최우선해야 함을 의미한다. 응용 명령 중에서는 전송 명령이 가장 많은 부분을 차지하고 그 다음이 비교 명령이었다. 특히 비교 명령은 그 결과가 기본 명령에서 사용되므로 효율적 처리가 필요한 명령이다.

명령의 종류	명령의 빈도
기본 명령	69%
타이머/카운터	1.5%
전송	12%
비교	9.3%
연산	6.1%
콘트롤	2.2%

표 1. 래더 명령의 빈도

모든 명령의 수행 시간이 같지 않으므로 조사한 명령의 빈도에 그 수행 시간을 곱한 값이 보다 정확한 빈도이다. 상용 PLC에서는 기본 명령보다 응용 명령의 수행 시간이 길므로 응용 명령의 실행 시간 빈도가 기본 명령보다 높아진다. 이는 PLC 시스템 성능 향상을 위해서는 기본 명령보다 응용 명령의 고속 처리가 더 중요함을 의미한다. 기본 명령의 연산 결과가 응용 명령의 수행 여부를 결정하므로 상용 프로세서를 사용하는 경우 흐름 제어 명령이 사용되는데 제안하는 RISC 구조에서는 추가의 흐름 제어 명령없이 기본 명령 이후의 응용 명령이 실행되도록 하였다.

2.3 오퍼랜드 빈도 분석

래더 프로그램에서 오퍼랜드의 어드레싱 모드의 종류와 빈도를 조사하였다. 표 2에서 보는 바와 같이 Absolute 어드레싱이 차지하는 빈도가 범용 컴퓨터에 비해 매우 높으므로 명령 버스와 데이타 버스가 분리된 하바드 구조가 적합함을 알 수 있다. 그리고 하나의 오퍼랜드가 참조되는 평균 회수가 두번 이하로 범용 프로세서에서 사용되는 데이타 캐쉬의 필요성이 없다.

오퍼랜드의 종류	오퍼랜드의 빈도
Absolute	75%
Immediate	20%
없음	5%

표 2. 오퍼랜드의 종류와 빈도

2.4 서브루틴의 깊이 분석

래더 프로그램에서 사용되는 서브루틴은 범용 언어와는 달리 사용 빈도가 매우 적으며 네스팅이 거의 없다. 즉 서브루틴의 깊이가 평균 1이하로 그 이유는 PLC가 사용되는 순서제어에서는 반복되는 과정이 별로 없기 때문이다. 그러므로 범용 RISC에서 잘 사용되는 레지스터 윈도우 등이 필요없다.

3. PLC용 RISC 프로세서의 명령어

3.1 명령어 세트

래더 언어와 프로그램의 분석에 따라 정의한 명령어들은 모두 58종으로 다음과 같다.

(1) 기본 명령

기본 명령은 모두 24종으로 이중에서 출력 명령과 DIF 명령은 각각 2개와 3개의 명령으로 치환되었다. 이는 필요한 메모리 억세스가 각각 2회와 3회이기 때문이다. 기본 명령의 수행 결과는 LRS(logical result stack)이라고 하는 쉬프트 레지스터에 저장되며 마스터 콘트롤 명령의 경우 MCS(master control stack)을 사용한다. LOOP 명령을 제외한 모든 응용 명령은 조건부 수행이 가능하며 그 수행 조건은 LRS의 MSB이다.

(2) 메모리 억세스 명령

워드 명령 중에서 메모리 억세스는 LOAD와 STORE의 두가지 명령에 의해서만 일어난다. 가능한 어드레싱 모드는 Absolute, Register Indirect, Immediate의 세가지이다.

(3) 레지스터 전송 명령

레지스터 간의 데이타 이동에는 MOVE 명령이 사용된다.

(4) 연산 명령

불럭 연산을 위한 명령으로 ADD 등 12종을 정의하였다. 이중에서 BIN, BCD 명령은 범용 마이크로프로세서에는 없는 명령이다.

(5) 쉬프트 명령

쉬프트와 로테이트 명령은 SHL 등 모두 6종이다.

(6) 비교 명령

비교 명령은 STRCGT을 포함하여 모두 6종이다. 비교의 결과는 기본 명령의 결과가 저장되는 LRS에 저장되므로 비교의 결과에 따라 다른 응용 명령의 수행이 결정된다.

(7) 콘트롤 명령

콘트롤 명령은 JMP 명령 등 모두 7종이다. CALL 명령의 경우 리턴 어드레스는 레지스터에 저장하며 리턴에는 JMP 명령을 사용한다. LOOP 명령은 지정한 레지스터의 값이 영인지를 검사하고 '0'이 아닐 경우 분기와 동시에 값을 감소시키는 동작을 하는데 대부분의 경우 분기가 일어나므로 무조건 수행이 되도록 하여 분기 페널티가 한 클록만 생기도록 하였다.

3.2 명령어 형식

제안된 명령어 세트를 위한 모든 명령어의 크기는 32비트이고 명령어와 오퍼랜드의 종류에 따라 다섯가지 형식으로 나누어진다. 기본 명령의 어드레싱 공간은 16 Mbit이고 응용 명령의 어드레싱 공간은 16 Mbyte이다. 모든 응용 명령이 조건부 수행이 가능하도록 C 필드를 마련하였다. 즉 C 값이 '1'인 경우 LRS의 MSB값에 따라 수행 여부가 결정된다. 별도의 명령이 없이 응용 명령의 조건부 수행이 가능하므로 SFC와 같은 고급 언어도 고속 처리가 가능하다. g-code는 명령의 그룹을 뜻하는 필드이고 i-code와 함께 명령의 종류를 나타낸다.

4. PLC용 RISC 프로세서의 구조

4.1 파이프라인 처리

제안하는 전용 프로세서는 그림 3과 같이 4단계의 파이프라인을 사용하여 병렬 처리가 이루어진다.

(1) 명령 읽기 단계(IF)

이 단계에서는 명령 버스를 통해 메모리에서 명령을 읽어온다. 동시에 프로그램 카운터가 변화된다.

형식 1. 기본 명령

1	4	3	4	20
0	g-code	i-code	Bit Position	Absolute Address

형식 2. LOAD Immediate, Register Indirest

1	4	3	3	3	1	1	16
C	g-code	i-code	DR	SR	0	P	Immediate Data

형식 3. LOAD, STORE, LOOP Absolute

1	4	3	24
C	g-code	DR,SR	Absolute Address

형식 4. JMP, CALL Absolute

1	4	3	24
C	g-code	i-code	Absolute Address

형식 5. MOVE, 연산

1	4	3	3	3	1	7	5	5
C	g-code	SR1	DR	SR2	S	0	i-code	const

그림 2. 전용 RISC 프로세서의 명령어 형식

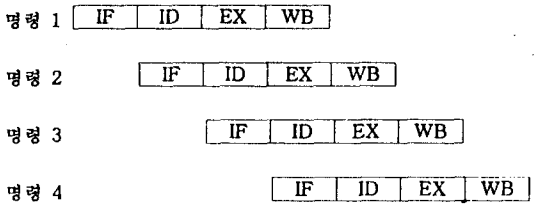


그림 3. 파이프라인 처리

(2) 명령 해석 단계(ID)

이 단계는 읽은 명령을 해석하는 단계이다. 이와 동시에 레지스터가 선택되며 Absolute 어드레스가 MAR에 읽혀진다.

(3) 명령 실행 단계(EX)

이 단계는 명령을 실행하는 단계로 LOAD와 STORE 명령의 경우 메모리를 억세스하는 단계이다. 기본 명령의 경우 오퍼랜드를 읽기와 논리 연산이 이 단계에서 이루어진다. 이는 다음 명령이 응용 명령일 경우 그 수행을 제어하기 위해서이다.

(4) 결과 저장 단계(WB)

마지막인 저장 단계에서는 기본 명령의 경우에는 아무일도 하지 않고 응용 명령일 경우에만 연산 결과가 레지스터에 저장이 된다.

데이타 의존에 의한 파이프라인 장애를 해결하기 위해 연산 유니트와 메모리 유니트에서 포워딩을 사용한다. 기본 명령의 수행이 세번째 단계에서 일어나므로 분기 명령이 성립될 때의 패널티는 두 클럭이다.

4.2 프로세서의 구조

제안하는 전용 프로세서의 구조는 그림 4와 같이 네부분으로 나누어진다. 앞에 설명한 바와 같이 데이타와 명령을 위한 각각의 버스를 가지며 4단계 파이프라인을 사용하여 매 클럭마다 명령의 수행이 이루어진다. PC 유니트에는 세개의 프로그램 카운터와 명령 레지스터 그리고 명령 해석기와 콘트롤 파트가 들어있다. 메모리 인터페이스 유니트에는 메모리 인터페이스 레지스터가 들어 있다. 레지스터 유니트에는 기본 명령을 위한 두개의 쉬프트 레지스터와 응용 명령을 위한 8개의 범용 레지스터, 상태 레지스터가 있으며 평션 유니트에는 ALU, SHIFTER, 곱셈기, 비트 연산 로직들이 있다. 정수 처리 기능 만을 가지고 있으며 실수 처리를 위해서는 보조 프로세서를 사용한다. 그림에서 보는 바와 같이 레지스터 유니트에서 나오는 버스들 외에도 평션 유니트와 메모리 인터페이스 유니트, PC 유니트 사이에 포워딩을 위한 버스가 있다. PC 유니트 내의 콘트롤 파트는 범용 RISC와 마찬가지로 하드와이어 로직으로 이루어진다.

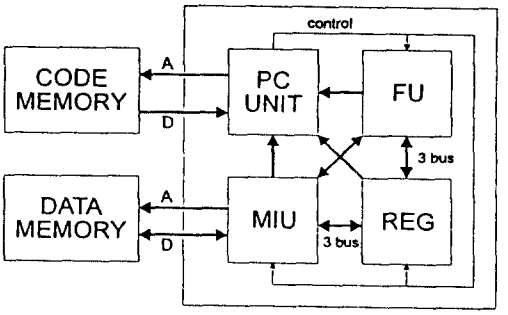


그림 4 전용 프로세서의 구조

5. PLC용 RISC 프로세서의 성능 평가

5.1 기존 래더 프로그램의 변환

제안된 전용 프로세서에서는 기존 PLC에서 사용되던 프로그램을 그대로 사용할 수 있다. 그러나 래더 명령어를 전용 프로세서의 명령어로 대체하는 과정이 필요하고 이를 위해 IBM PC에서 래더 프로그램을 자동으로 변환하는 프로그램을 작성하였다. 변환 비율은 평균 1:2.3 정도로 이는 모든 래더 명령어를 수행할 수 있는 CISC 형의 전용 프로세서보다 1.74배 빠름을 의미한다.

5.2 시뮬레이션과 성능 비교

상용 PLC나 기존의 연구에서 PLC의 성능은 가장 단순한 기본 명령의 수행 시간으로 계산되었다. 이것은 마치 범용

프로세서에서 사용되던 MIPS와 비슷한 개념으로 실제의 성능을 잘 나타내지 못한다. 그러므로 본 연구에서는 실제의 성능과 근접하기 위하여 두가지 방법으로 성능을 계산하였다. 그 첫번째 방법은 2장의 통계 분석의 결과로 얻은 기본 명령과 응용 명령의 비율로 래더 명령이 혼합된 프로그램의 실행 시간을 측정하는 것이다. 두번째 방법은 실제 사용되고 있는 래더 프로그램의 실행 시간을 측정하는 것이다. 래더 프로그램을 전용 프로세서의 명령으로 변환한 후 시뮬레이션하는

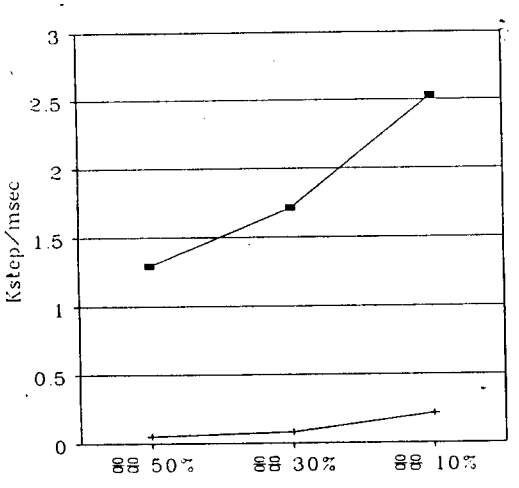


그림 5. 성능 비교 1(혼합된 명령)

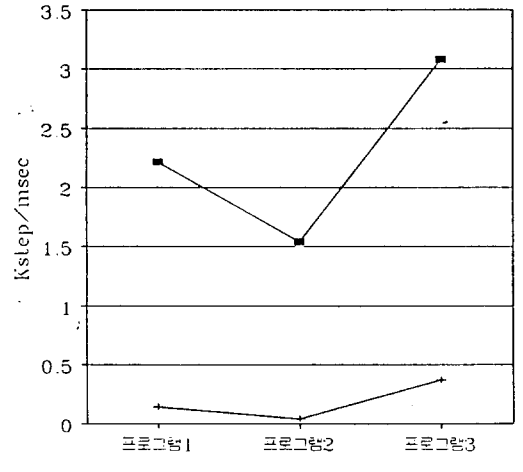


그림 6. 성능 비교 2(실제 프로그램)

프로그램을 IBM PC에서 작성하여 그림 5와 6에 그 결과를 나타내었다. 시뮬레이션에서는 100 nsec 억세스 타임의 메모리와 8 MHz의 시스템 클럭을 가정하였고 이를 같은 시스템 클럭의 상용 PLC와 비교하였다. 그림 5는 기본 명령과 응용 명령의 비율을 변화시켰을 때의 성능을 나타내고 그림 6은 대표적 프로그램들을 실행했을 때의 성능을 나타내는 것이다. 두가지 시뮬레이션의 결과 제안된 전용 프로세서의 성능은 평균 1.71 Kstep/msec로 이는 상용 PLC의 약 20배의 성능에 해당한다.

6. 결론

본 논문에서는 RISC 형태의 전용 프로세서의 명령어 세트와 구조를 정의하였다. 실제로 사용되는 래더 프로그램에서 각 명령과 오퍼랜드, 서브루틴을 분석한 결과 PLC가 필요로 하는 다양한 응용 명령을 포함하고 기본 명령의 계산 결과가 응용 명령의 수행을 제어하게 함으로써 차세대 PLC 언어인 SFC의 고속 처리가 가능하도록 하였다. 제안된 구조는 명령과 데이타를 위한 버스가 나누어진 하바드 구조를 가지고 있으며 Load/Store 방식으로 메모리를 억세스하며 모든 명령이 4단계 파이프라인 처리가 된다. 기존에 작성되어 사용하는 프로그램을 사용할 수 있도록 변환 프로그램이 작성되었으며 시뮬레이션을 통해 성능을 비교한 결과 상용 PLC보다 약 20배의 성능을 가짐을 보였다.

REFERENCES

[1] W. Stallings, "Reduced Instruction Set Computer Architecture," *Proceedings of IEEE*, Jan. 1988.

[2] I. Warnock, *Programmable controllers - operation and application*, Prentice Hall, 1988.

[3] J. Kim, J. Park, W. H. Kwon, "Architecture of a ladder solving processor for programmable controllers," *Microprocessor and Microsystems*, Vol. 16, No. 7, 1992.

[4] J. Park, N. Chang, G. S. Rho, W. H. Kwon, "Implementation of a Parallel Algorithm for Event Driven Programmable Controllers," *Control Eng. Prctice*, Vol. 1, No. 4, August 1993.

[5] Y. Shimokawa, T. Matsushita, H. Furuno, Y. Shimanuki, "A High-Performance VLSI Chip of Programmable Controller and its Language for Instrumentation and Electrical Control," *Proceedings of '91 IECON*, 1991.

[6] M. L. Anido, D. J. Allerton, "RISC Design for Computer Image Generation," *Microprocessor and Microsystems*, Vol. 14, No. 6, 1990.

[7] T. Murata, N. Kormoda, K. Matsumoto, K. Haruna, "A Petri net based controller for flexible and maintainable sequence control and its applications in factory automation," *IEEE Trans. Industrial Electronics*, Feb.

1986.

[8] H. Murakoshi, M. Sugiyama, G. Ding, T. Oumi, T. Sekiguchi, Y. Dohi, "A high speed programmable controller based on Petri net," *Proceedings of '91 IECON*, 1991.

[9] P. C. Baracos, R. D. Hudson, L. J. Vroomen, P. J. A. Zsombor-murray, "Advances in Binary Decision Based Programmable Controllers," *IEEE Trans. Industrial Electronics*, Aug. 1988.

[10] International Electrotechnical Commission, Preparation of Function Charts for Control Systems, *IEC Publication 848*, 1988