'88 韓國自動制御學術會議論文集 1988. 10. 21~22

시퀀스 명령 고속 처리 회로의 Gate Array

○
유 지 훈 , 양 오 , 신 영 민 , 안 재 봉 , 이 종 두
금 성 산 전 주 식 회 사 연 구 소

Gate Array(Custom IC) of High Speed Processing Circuit for Sequence Instruction

J.H. YOO, O. YANG, Y.M. SHIN, J.B. ANN, J.D. LEE
Goldstar Industrial Systems Co., Ltd. R&D Center

ABSTRACT

Recently PLC pursues faster scanning time, circuit confidence, reliability improvement, and smaller size. To obtain above all merit, custom IC(Gate Array) is developed. Custom IC includes 5 main blocks and 2 auxiliary blocks. The 5 main blocks process faster sequential instruction execution by only logic gate using hexa instruction code system. And the 2 auxiliary blocks generate baud rate clock (153.6 KHz, 76.8KHz) to communicate between PLC and computer or programmer.

1. 서 론

프로그래머블 로직 콘트롤러(이하 PLC)의 고속화와 고기능화의 요구 및 신뢰성, 소형화가 대두되고 있는 현재의 PLC에서는 단지 소프트웨어에 의한 처리에서 벗어나 TTL과 EPROM에 의한 하드웨어 처리로 빠른 기본 명령을 수행하고 있다. 그러나 이러한 하드웨어 로직은 전 블록이 서로 상관되어 있고 많은 IC로 구성되어 있으며 특히 EPROM의 Access Time에 의해 더 빠른 처리를 수행할 수 없는 단점을 지니고 있다. 여기서 EPROM에 의한 처리 방법을 각 명령어의 Hex Code를 전체 처리에 맞도록 작성하여 EPROM Access Time(약 120-150nsec) 보다 빠른 TTL Gate(약20-30nsec)로 처리하여 모든 Block의 회로를 하나의 Custom IC로 제작하여 신뢰성 및 회로의 기밀성을 유지 시키고 회로와 Size의 Compact화가 이루어 지도록 하였다.

2. 명령어 Code

PLC에는 기본 명령(LOAD, AND, OR, OUT)에 각 접점의 용도에 따른 다양한 Operand가 구비되어 있으며 명령과 Operand의 조합에 따라 실제 명령어는 보통 1 Step당 2Byte로 구성되어 있다. 여기서는 Gate Array에 채택된 Operand와 Code에 대해서 설명하기로 한다.

표1. Operand 종류와 Code 값

종류	용 도	점 수	CODE 값
M	내부 메모리 접점	1024	00 ~ 03
P	I/O 접점	512	04 ~ 05
K	정전 보지 메모리 접점	512	06 ~ 07
L	링크용 접점	256	08
F	특수 접점	256	09
T	타이머 접점	128	0A
C	카운터 접점	128	0A

표 1과 같은 Operand를 갖는 기본 명령어의 Code 값은 표 2와 같다.

표2. 기본 명령어 종류와 Code 값

명령어	CODE 값	명 령 어	CODE 값
LOAD	00	LOAD NOT	40
AND	10	AND NOT	50
OR	20	OR NOT	60
SET	30	RST	70
OUT	80		

표1과 표2의 명령어와 Operand를 종합하면 각각의 Code를 구성할 수 있으며 예로 LOAD P20의 경우는 LOAD의 "00"과 P의 "04"를 더한 "04"와 번지를 나타내는 20의 Hex 값인 "14"의 2 Byte, 즉 04, 14로 형성된다.

3. 명령어 수행 System

각 Operand의 내용을 보존하는 RAM에서/으로 Read/Write 작업이 쉽도록 8Bit의 Data Line중 D0 1 Bit 만을 사용하여 한 접점당 1 Byte를 차지하도록 하였다. 또한 표 2에서 보는 바와 같이 데이터 메모리에서의 Read/Write 신호는 상위 Byte B7, B5, B4의 조합으로 이루어지도록 하였다.

(1) Timming 발생부

CPU의 Start 지령에 따라 Timing 발생부를 동작시키면 기본 Clock 10MHz를 받아들여 1 Status 당 4개의 주기(400nsec)를 갖는 4개의 Status (1.6usec)로 한 명령을 처리하는 프로그램 카운터부와 시프트 레지스터(이하 S.R.)부에 반복적인 기본 Clock을 발생시킨다.

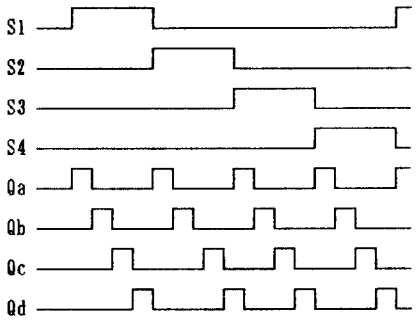

그림1. Timming Chart

S1 Status : 프로그램 메모리에서 상위 Code를 Access하여 데이터 메모리 번지의 상위 Byte를 생성하며 프로그램 메모리의 어드레스를 +1 증가시킨다.

S2 Status : 프로그램 메모리에서 하위 Code를 Access하여 데이터 메모리 번지의 하위 Byte를 생성한다.

S3 Status : 프로그램 메모리의 어드레스를 +1 증가시키고 Access된 Code로 하드웨어 처리가능 여부를 판단한뒤 가능한 경우 S.R.부의 동작을 수행한다.

S4 Status : 지정된 데이터 메모리의 접점 내용과 S.R.부의 ARG와의 연산결과를 Loading 또는 데이터 메모리에 Write 한다.

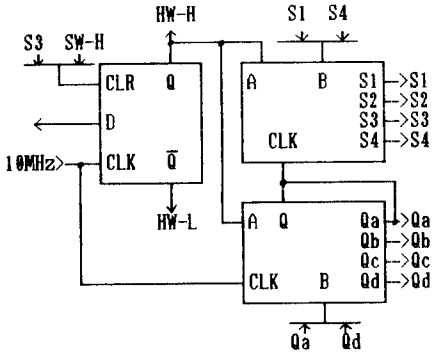

그림2. Time 발생부 Block 도

HW-H : 하드웨어 처리동안 High 신호

HW-L : 하드웨어 처리동안 Low 신호

SH-H : 하드웨어 처리불능 Code일때 High신호

(2) 프로그램 카운터부

프로그램 메모리에 저장되어 있는 명령어를 꺼내기 위한 어드레스를 공급한다. S1 Status에서 상위 Code를 Access할수 있는 우수 번지를 공급한뒤 +1 증가시켜 S2 Status에서 하위 Code를 Access할 수 있는 기수 번지를 공급한다.

또한 S3 Status에서 +1 증가되어 다음 Step의 어드레스를 지정하여 나간다. 그리고 S3와 S4 Status에서는 데이터 메모리를 지정하는 어드레스와 충돌이 되지 않도록 해당 버퍼를 Disable 시킨다.

(3) Code Latch와 Address 발생부

S1 Status에서 상위 Code를 Latch 시켜 8 Bit의 내용(B8-B15)을 Code 판독부와 신호 발생부에 넘겨줌과 동시에 상위 Nibble이 "C"가 되도록 하여 접점 상태를 저장하는 데이터 메모리의 절대 번지를 가리킬수 있도록 상위 Byte를 생성한다.

S2 Status에서 하위 Code를 Latch하여 데이터 메모리의 하위 Byte를 생성시킨뒤 S3와 S4 Status 동안 버퍼를 Enable 시켜 데이터 메모리에 어드레스를 공급한다.

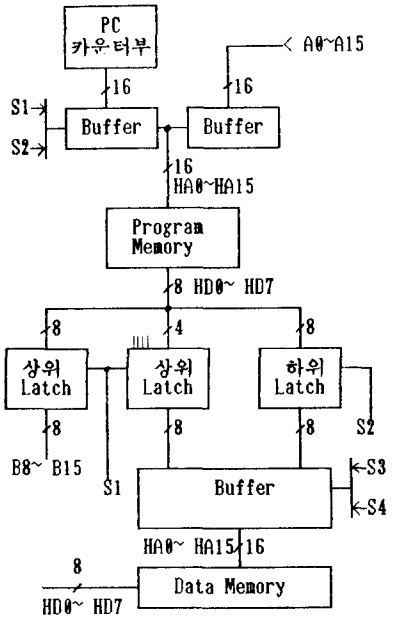

그림3. Address 발생부 Block 도

(4) Code 판독부와 신호 발생부

S1 Status에서 읽혀진 상위 8 Bit로 하드웨어 처리가 가능한지를 판별하여 불가능한 경우는 SW-H 신호를 High 상태로 전환시켜 Timing 발생부를 해제시킴과 동시에 CPU에의한 처리가 되도록 모든 Buffer를 Disable 시킨다. 또한, 하드웨어 처리 동안 프로그램/데이터 메모리에의 Read/Write 동작이 이루어지도록 R/W 신호를 구성하고 B8-B15의 값에 따른 S.R.부의 처리가 이루어 지도록 LS198의 L0, L1의 상태를 구성한다.

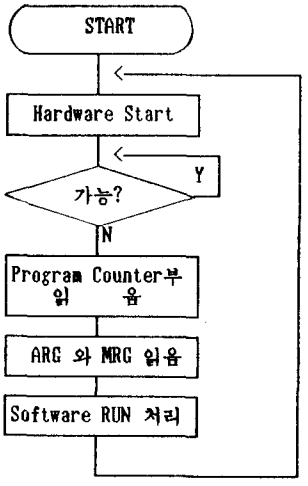

그림4. CPU의 처리 Flow

(5) 시프트 레지스터부

Code 판독부에서 하드웨어 처리가 가능한 경우 시프트 레지스터 처리가 행하여 진다.

가) 입력명령(LOAD, AND, OR)

S3 Status에서 표3과 같이 S.R.부 처리가 이루어진뒤 데이터 메모리에서 접점의 상태를 Access 하여 LS198의 ARG(Qa)와 AND, OR Gate를 통한 연산 결과를 S4 Status에서 LS198에 재 Loading 시켜 한 Cycle을 완료한다.

나) 출력 명령 (OUT, SET, RST)

S3 Status에서 S.R.부의 동작처리 없이 S4 Status에서 LS198의 ARG(Qa) 상태를 데이터 메모리에 Write하여 한 Cycle을 완료한다.

다) NOT, NOP

S4 Status에서 LS198 상태를 반전 또는 그대로 재 Loading하여 한 Cycle을 완료한다.

라) AND/OR LOAD

S3 Status에서 표3과 같이 ARG의 상태를 Latch 시키고 우 시프트 처리한뒤 새로운 ARG(전의 Qb 상태)와 Latch된 ARG(전의 Qa 상태)을 AND, OR 시킨 연산결과를 S4 Status에서 재 Loading 시켜 한 Cycle을 완료한다.

표3. S.R. 부 처리표

상태 / 명령어	S3			S4		
	LS198 L0	L1	처 리	L0	L1	처리
LOAD (NOT)	1	0	좌시프트	1	1	로딩
AND (NOT)	0	0	-	1	1	로딩
OR (NOT)	0	0	-	1	1	로딩
OUT, SET, RST	0	0	-	1	1	로딩
NOP	0	0	-	1	1	로딩
NOT	0	0	-	1	1	로딩
AND LOAD	0	1	우시프트	1	1	로딩
OR LOAD	0	1	우시프트	1	1	로딩

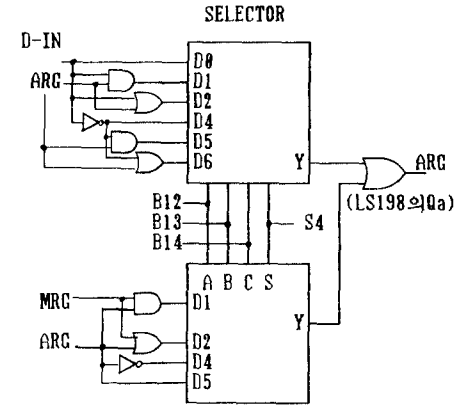

그림5. 연산 로직 Blcok 도

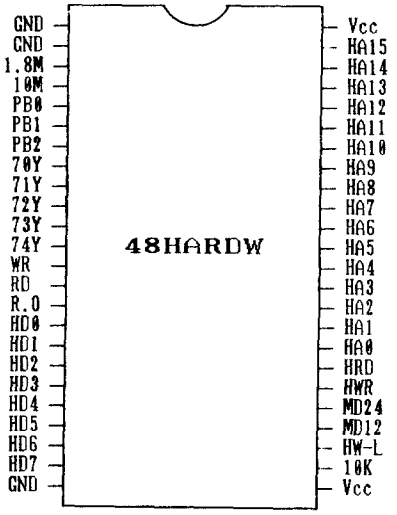

그림6. Gate Array의 Pin 배치도

4. 결 론

현재는 RAM의 Access Time 때문에 10MHz 기본 Clock(100nsec)를 사용하였지만 계속 개발되고 있는 RAM(최근 20nsec)을 사용하면 0.6usec 이하의 고속처리도 가능해진다.

또한 Gate Array의 제작시는 기존의 TTL Gate로 실현이 어렵거나 가능하더라도 복잡한 경우의 어떤 회로도 실현 가능하며 PCB상의 Pattern에 의한 간섭이 없어지므로 신뢰성이 향상되고 회로의 기밀성이 유지된다. 그밖에 변함이 없는 회로를 일부 삽입하므로 해서 전체 System의 안전 및 Size의 Compact화도 추구할 수 있는 장점을 가지고 있다.

향후에는 기본 명령뿐 아니라 일반적인 연산 명령까지 가능한 Gate Array Logic화 함으로써 Scanning Time을 빨리 할 수 있는 PLC의 발전이 기대된다.

Reference

1. 최호현, 안재봉, 지동근, 유지훈, 이영준, "Programmable Controller의 명령어 수행 System", 한국자동제어 학술회의, 1987
2. 권욱현, 김종일, 최한홍, 김덕우, 홍진우외, "대형 프로그래머블 콘트롤러의 개발 : Part I (H/W)", 한국자동제어 학술회의, 1987
3. Louis Nashelsky, "Introduction to Digital Computer Technology", 탑출판사, 1987
4. GSS, "CMOS Macrocell Manual", 1987
5. NEC, "One Chip Microcomputer User's Manual", 1988
6. J.A.Siebel and C.L.Arosor, "Programmable Controller", Automation, 1984
7. James W. Coffron and William E.Long, "Practical Interfacing Techniques for Microprocessor System", Prentice Hall, INC., 1983