論 文
53P-1-2

가변형 교통 표지판의 서브 컨트롤부 구현

Implementation of Sub Control Part for Variable Message Signboard

申 載 興* · 金 弘 烈** · 李 相 基***
(Jae-Heung Shin · Hong-Ryul Kim · Sang-Kee Lee)

Abstract - Previously, in order to send information from the local controller to the display board, the hardware or software had to be handled and run through 3-phases, which include the PC-card or PC-add Board, I/F card and Sub board. This study will attempt to design a board that handles information by connecting the USB port of the PC directly to the Sub board. In addition, an MPU will be attached to the previously complex hardware circuit to design a software drive engine module, which allows for the development of new products by modifying only the software engine and not the hardware.

Key Words : Variable message signboard, MPU, drive engine

1. 서 론

우리나라는 좁은 국토와 산악지형으로 인하여 새로운 도로를 건설하는 데는 막대한 비용이 발생한다. 또한, 기존의 도로가 계획에 의해 건설되었다기 보다는 고도성장기의 필요에 따라 건설된 관계로 각 도로 사이의 연계를 통한 교통분산이 어렵고, 급격한 경제개발과 산업화로 인하여 물동량이 엄청난 속도로 증가하고 있다. 이러한 물동량을 효율적으로 처리하기 위해서는 도로망의 확충이 필수적이지만 원활하게 이루어지지 못한 관계로 막대한 물류비용이 발생하고 있다.

한정된 도로자원과 엄청난 물동량을 효율적으로 처리하는 방안 중의 하나는 지능형 교통 시스템(ITS : Intelligent Transport System)을 적용한 도로를 구축하는 것이다. 지능형 도로를 건설하거나 기존도로에 적용하여 구축하기 위해서는 막대한 비용이 발생하지만, 우리나라와 같이 도로 자원이 부족한 나라에서는 도로자원을 효율적으로 활용할 수 있는 가장 적합한 대안이 될 수 있다. 지능형 도로는 다양한 센서와 화상 정보를 분석·처리하여 그 도로나 주변 도로를 운행하는 자동차 운전자들에게 도로와 관련된 실시간 정보를 제공함으로써 운전자가 도로의 다양한 상황에 능동적으로 대처할 수 있도록 해줄 수 있다.

지능형 교통 시스템은 정보를 수집하는 시스템, 수집된 정보를 분석·처리하는 시스템, 그리고 가공된 정보를 실수요자인 운전자에게 제공하는 시스템으로 나누어진다. 이 가운데 정보 제공 시스템은 주로 가변형교통표지판(VMS : Variable Message Signboard)을 사용한다. 가변형교통표지판은 교통정보를 제공하는 주 시스템과의 인터페이스를 수행하고 현장의 가변형교통표지판 시스템을 관리하는 현장제어기부와 실제 표출 소자인 LED(Light Emitting Diode)를 구동하는 드라이브부, 그리고 현장제어기부의 표출정보를 드라이브부에 적합한 데이터로 변환하여 적절히 분배하여 주는 서브컨터롤(sub control)부 등으로 구성된다.

오늘날 가변형교통표지판에 표출 데이터 량이 급격히 증가하고 있으며 기존의 PC-card 또는 PC-add board와 I/F card 및 서브 보드(sub board)의 3단계로 구성되는 시스템의 복잡성을 간소화하여 장애를 최소화하고, 기존의 시리얼 인터페이스 방식이 갖는 데이터 처리 속도의 제한을 극복하는 시스템 개발 필요성이 대두되고 있다.

본 논문에서는 처리속도의 제한을 극복하기 위한 USB를 이용한 인터페이스 구현, 사양의 변경에 따라 하드웨어를 다시 개발해야 하는 문제점을 해결하기 위한 소프트웨어 엔진 개발을 개발하고, CPLD(Complex Programmable Logic Device)를 이용하여 고속 메모리 스캔부 구현하는 서브컨트롤부를 개발한다.

2. 서브 컨트롤 보드의 전체 구성

서브 컨트롤 보드(sub control board)는 두 가지 방식으로 구성된다. 첫 번째 방식은 현장제어기로 주로 사용되는 PC 내부에 PCI나 ISA 버스를 통하여 현장제어기 내부에 장착되는 방식이다. 이 방식은 드라이브 부와 연결하기 위한 커넥터와 케이블이 주요한 장애의 요인으로 작용하고, 또한 서브 컨트롤 보드의 유지보수를 위해서는 현장제어기를 분해해야 하는 어려움이 있다. 두 번째 방식은 현장제어기와 RS232C/RS422 또는 USB로 연결되어 현장제어기와는 별도

* 正 會 員 : 동서울대학 컴퓨터시스템과 助敎授 · 工博
** 正 會 員 : 동서울대학 컴퓨터시스템과 助敎授 · 工博
*** 正 會 員 : 봉오전자공업(주) 연구소장, 동서울대학 컴퓨터시스템과 兼任敎授 ·
接受日字 : 2003年 10月 27日
最終完了 : 2003年 11月 25日

로 외부에 장착되는 방식이다. 이 방식은 가변교통표지판에 표출할 데이터가 많아 통신속도가 문제가 되는 극히 예외적인 경우를 제외하고는 드라이브부와 연결이나 유지보수 측면에서 훨씬 편리하다.

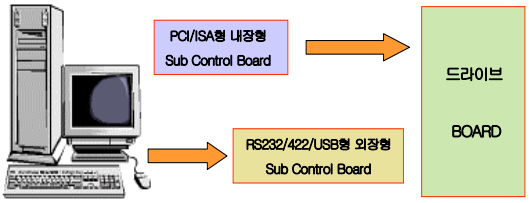


그림 1 서브 컨트롤 보드의 구성.
Fig. 1 Configuration of the sub control board.

기존의 서브 컨트롤 보드는 일반 TTL 로직을 사용하여 구성되어 있어, 드라이브부의 인터페이스 사양이 조금이라도 변경되면 다시 만들거나, 많은 점퍼 라인을 사용해야 하고, 많은 IC들로 구성되어 있기 때문에 장애 발생시 장애위치를 추적하는데 많은 시간이 소요되고, 공간을 많이 차지하는 불편함이 있다.

드라이브부의 인터페이스 사양의 변경에 대하여 능동적으로 대처할 수 있도록 기존의 하드웨어 로직으로 구성된 회로를 원칩 마이크로프로세서(one-chip microprocessor)를 이용한 소프트웨어 엔진으로 대체하고, 드라이브부와의 연결부를 일반적인 용도(general purpose)의 입출력 사양으로 설계하여, 인터페이스 사양이 변경될 때, 하드웨어를 변경하지 않고 소프트웨어 엔진의 수정으로 처리할 수 있도록 구현한다. 또한, 소프트웨어 엔진으로 구현이 어려운 그레이드 처리가 필요한 비디오 정보의 표출은 CPLD를 사용하여 고속 메모리 스캔부를 구현한다.

3. 소프트웨어 엔진의 구성 및 구현

소프트웨어 엔진을 구동하기 위해 필요한 CPU로써 기존의 Intel8051을 개선하여 머신 사이클(machine cycle)을 1/3로 줄여 같은 시스템 클럭에서 3배의 속도를 가지며, 33Mhz까지 구동 가능한 DS80C320을 사용하였다. 이것은 소프트웨어로 구현된 하드웨어 로직이 신호들 사이에 충분한 여유를 갖도록 하여 실제 하드웨어에서 타이밍에 의한 오류를 방지하고, 소프트웨어로 구현할 때, 제약을 줄이기 위하여 고속 CPU를 사용하였다.

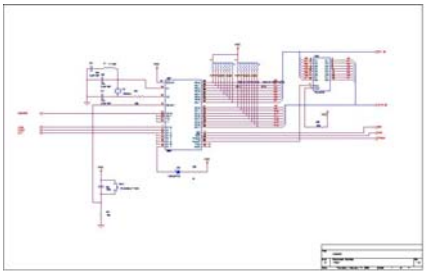


그림 2 소프트웨어 엔진 회로
Fig. 2 Software engine circuit

DS80C320은 원칩 마이크로프로세서이기 때문에 어드레스 래치를 위해 IC를 하나 더 사용하여 그림 2와 같이 간단한 회로로 구현된다.

소프트웨어 엔진을 탑재하기 위해 외부 EPROM 27256를 사용하여 32K 바이트의 코드 메모리(code memory)를 설정하고, 내부 데이터 저장 및 처리 버퍼를 위해 SRAM 62256을 사용 32K 바이트의 데이터 메모리(data memory)를 그림 3과 같이 설정했다.

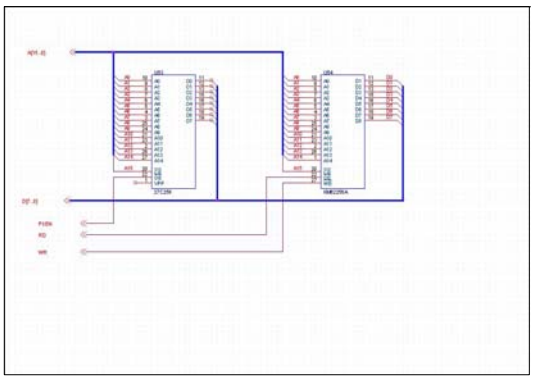


그림 3 데이터 메모리 설정
Fig. 3 Setting of the data memory

그리고 드라이브부로 신호를 전달하기 위한 신호 버퍼링(buffering)을 위해 그림 4의 회로와 같은 IC를 사용하였다. 각 IC의 입·출력은 특정 용도가 아닌 일반용도로 사용될 수 있도록 디자인되어 있어 소프트웨어 엔진에서 가장 적절한 형태로 정의하여 사용할 수 있다.

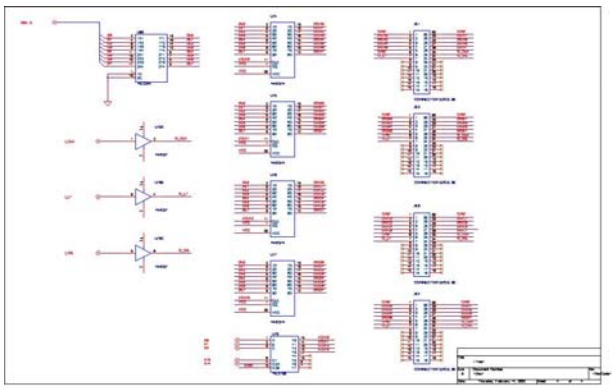


그림 4 각 IC의 내부 회로도.
Fig. 4 Internal circuit diagram using ICs.

이상과 같은 전체 구성에서 드라이브 부로의 데이터 및 제어 신호를 구성하기 위한 로직 구성부가 없고, 데이터 스캐닝에 필요한 어드레스를 구성하기 위한 카운터 회로가 없는 것은 소프트웨어 엔진, 즉, CPU에 의해 처리되기 때문이다.

DS80C320은 시리얼 통신을 위한 UART 기능이 내장되어 있어 현장제어기와의 데이터 통신에는 문제가 없지만 고속의 데이터 송·수신이 필요할 경우 한계가 있다. 따라서 선택적으로 USB를 사용할 수 있도록 아래의 회로를 추가할 수 있

도록 구현하였다. 구현된 회로는 그림 5와 같다.

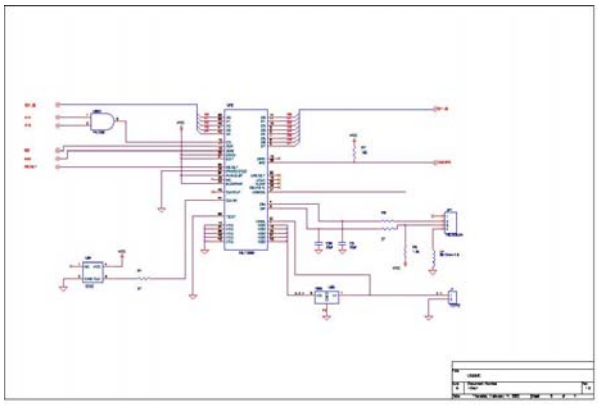


그림 5 USB 포트를 추가하여 설계한 회로도.
Fig. 5 New circuit diagram with USB port.

위의 회로에서 사용한 칩은USB1.1(12M)를 지원하는 NetChip사의 Net2890을 사용하였다. USB포트를 사용할 때, 현장제어기(USB host)와의 인터페이스를 구현하기 위해서는 소프트웨어 모듈이 추가되어야 한다.

4. CPLD 구성

그레이드가 사용된 비디오 타입의 데이터를 표출하기 위해서는 기본적인 데이터 소스(source) 자체가 256 그레이드로 구현되어야 하고, 하나의 도트(dot)에 대하여 1 바이트를 사용해야 하므로 8배의 데이터를 저장처리 해야 한다. 이러한 문제는 CPU의 클럭을 높여서 해결할 수 있지만, 저장된 데이터를 드라이브부로 보낼 때는 8 비트를 해석하여 최소 256배의 속도로 스캐닝해야 하는 문제가 있기 때문에 CPU의 클럭을 높이는 것으로 해결하는 것은 불가능하다. 드라이브부에 8 비트를 해석하여 256개의 펄스(pulse)로 변환하는 로직이 구성되어 있다면, 8배로 속도가 증가시키면 되지만 그렇지 않은 경우는 CPLD로 로직을 구현해야만 한다. CPU가 256배 이상의 처리 속도를 가진다고 가정해도 비디오 데이터는 지속적으로 스캐닝되지 않으면 화면이 부드럽게 표현되지 않고 색도 제대로 표현될 수 없다.

CPU는 내부 코드를 처리하면서 현장제어기로부터의 데이터 전송도 처리해야하므로 지속적인 스캐닝은 어렵다. 따라서 그레이드 데이터의 처리를 위해서는 아래에서 설명하는 CPLD 로직이 추가되어야만 적절하게 동작할 수 있다.

CPLD의 전체 구조는 그림 6과 같다. 그림 6은 기존의 회로에 ALTERA의 CPLD 칩인 MAX7128를 추가하였고, 이 칩과 CPU의 인터페이스, 표출 그레이드 데이터 RAM과의 인터페이스를 위하여 회로를 수정하였다.

구성도를 보면 크게 3가지로 구성 되어 있다. CPU인 DS80C320 , CPLD MAX7128, 뱅크(BANK) 메모리로 구성되어 있다.

CPU는 복잡한 모듈 디스플레이 제어에 관여하여 않고 데이터 메모리로부터 데이터를 READ하여 뱅크 메모리에 WRITE하는 역할과 데이터 산술연산을 한다. CPLD는 CPU로부터 새로운 데이터가 준비되었다는 신호를 받으면, 현재 스캔하고 있는 뱅크 메모리의 데이터를 끝까지 스캔 한 후, CPU가 새로 전달한 뱅크 메모리에 저장된 데이터를 READ하여 드라이브부에 필요한 제어신호인 CLK, LATCH, OE, ADDRESS, DATA를 만들어 스캔을 시작한다.

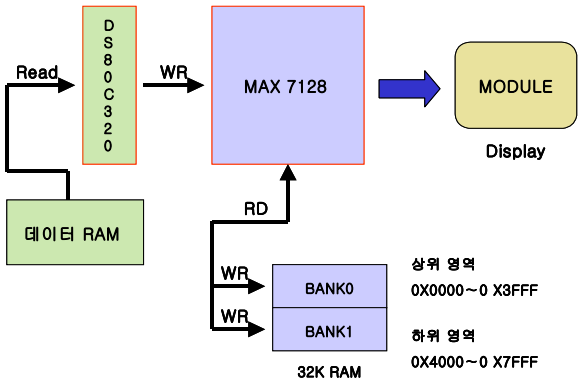


그림 6 CPLD 전체 구조도
Fig. 6 Architecture for the CPLD

CPLD의 스캔닝(뱅크 메모리의 데이터를 READ하는 동작)은 별도의 뱅크 메모리를 사용하지 않기 때문에 WRITE하는 동작과 동시에 이루어 질 수 없다, 즉, SRAM에서 동시에 READ와 WRITE동작 할 수 없다. 이 문제를 해결하기 위해 CPU가 RAM을 액세스하지 않는 동안, CPLD의 스캔닝 회로가 동작하도록 구성하였고, CPU가 ROM으로부터 실행 코드를 가져오는 머신 사이클, 외부 메모리 접근을 위해 하위 어드레스부를 래치하는(DS80C320은 데이터 버스를 어드레스의 하위부와 같이 사용하도록 디자인되어 있다.) 머신 사이클 동안에 처리를 하도록 구현하였다.

4.1 CPU 입력 클럭 생성부 구현

CPU 입력 클럭은 CPLD외부의 48MHZ 오실레이터로 클럭(범용클럭)을 입력받아 2분주하여 사용(24MHZ)하며, CPLD와 CPU의 동작의 동기를 맞춘다. 클럭 생성부는 그림 7과 같다.

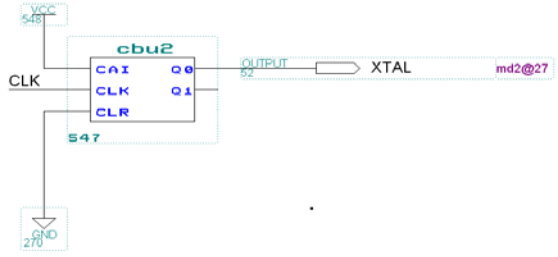


그림 7 CPU 입력 클럭 생성부 회로
Fig. 7 Circuit of the CPU input clock generation

4.2 버스 제어부 구현

CPLD는 뱅크 메모리를 하나 사용하므로 동시에 메모리를 READ하고, WRITE할 수 없기 때문에 버스제어가 필요하다.

CPU가 뱅크 메모리에 WRITE할 때, 우선순위를 주고 CPU가 뱅크 메모리에 쓰지 않을 때는 CPLD 로직이 뱅크 메모리의 데이터를 READ하여 처리하도록 구현하였다.

4.2.1 어드레스 버스 제어

CHANGE가 '1'이면 뱅크 메모리의 어드레스 버스는 CPU의 어드레스 버스와 연결되며CHANGE가 '0'이면 메모리의 어드레스 버스는 CPLD 내부의 어드레스 버스와 연결된다. 어드레스 버스 제어 회로는 그림 8과 같다.

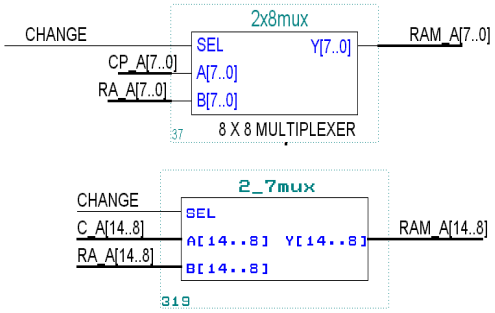


그림 8 어드레스 버스 제어
Fig. 8 Control of the address bus

4.2.2 데이터 버스 제어

CHANGE 신호가 '1'이면 CPU_D[7..0]이 BI_D[7..0]으로 출력된다. CHANGE 신호가 '0'이면 BI_D[0..7]은 CPLD 로직 데이터 입력 핀이 된다. 그리고 BB_D[7..0]은 BI_D[7..0]와 연결되어 입력 핀으로 작동된다. 즉, 버스 제어가 CPU에게 있으면, 양방향 핀인 BI_D[7..0]은 CPU의 데이터 버스선과 연결되어 출력 핀이 된다. 반대로, 버스 제어가 CPLD가 만든 로직에 있으면, CHANGE 신호에 의해 CPU_D[7..0]은 BI_D[7..0]과 연결이 DISABLE이 되고 RAM의 데이터 버스와 연결 되어 BI_D[7..0]이 입력 핀이 되어 데이터를 읽어 들인다. 데이터 버스 제어 회로는 그림 9와 같다.

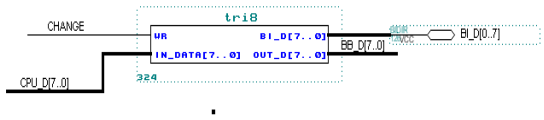


그림 9 데이터 버스 제어
Fig. 9 Control of the data bus

4.2.3 버스 제어 신호

버스 제어에서 중요한 것은 뱅크 RAM이 하나이므로 RAM의 RD 및 WR가 충돌이 나지 않도록 제어하는 것이다. CPU가 RAM에 데이터를 WRITE할 때, CPLD 로직이 동시에 RAM의 데이터를 READ하면 데이터를 가공·처리하지 못한다. 따라서 CPU가 데이터를 WRITE할 때는 버스 제어를 CPU에 주고, CPLD 로직이 RAM 데이터를 READ하지 않도록 그림 10과 같이 버스 제어 신호를 만들었다.

그림 10 버스 제어 신호
Fig. 10 Bus control signal

CPU는 앞에서 언급한 것처럼 뱅크 메모리로 8000H이상의 번지를 사용하기 때문에 어드레스 A15를 사용한다. 그리고 CPU가 데이터를 WRITE할 때는 PSEN신호가 항상 '1'이 되어 있으므로 이 두 신호를 이용하여 CPU가 WRITE할 때의 신호를 만들어준다.

4.3 CPU RAM READ부 구현

그림 11은 CPLD 로직회로가 RAM 데이터를 READ할 때의 타이밍도이다. ADR_CLK는 뱅크 메모리를 읽는 클럭이다. ADR_CLK가 첫번째 클럭에서(1부분) HIGH 에지 트리거(edge trigger)가 되면 RAM 데이터 1 바이트가 RED DISPLAY 데이터 1 바이트로 래치되고 LOW 에지 트리거가 되면, RAM 어드레스가 하나 증가한다.

다음 두 번째 클럭에(2부분) 의해 마찬가지로 HIGH 에지 트리거에서 GREEN DISPLAY 데이터가 1 바이트 래치되고 LOW 에지 트리거에서 어드레스가 하나 증가한다. 그리고 3부분에서 래치되어 있는 데이터로 DISPLAY 데이터와 클럭을 만들어 출력한다.

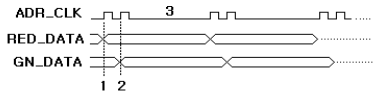


그림 11 RAM 데이터 READ 타이밍도
Fig. 11 Timing diagram of the RAM data READ

4.3.1 어드레스 생성

ADR_CLK를 입력 클럭으로 사용하여 LOW 에지에서 카운트하여 RAM 데이터 READ ADDRESS 버스를 만든다. 이것은 0000H~0FFFH까지 카운트되므로 A12가 '1'이면 클리어되어 처음부터 다시 카운트한다.

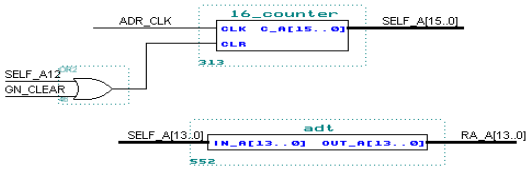


그림 12 어드레스 생성 회로
Fig. 12 Circuit of the address generation

4.3.2 RAM_RD 구현

CPLD 로직의 RAM RD 신호 ACCESS는 CPU가 WRITE할 때가 아니므로 그림 13와 같이 구현하였다.

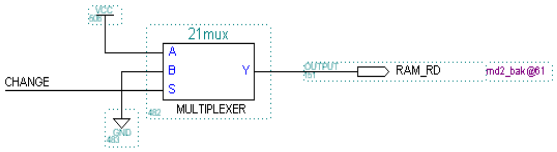


그림 13 RAM_RD 신호 생성 회로
Fig. 13 Circuit of the RAM_RD signal generation

CHANGE는 CPU가 WRITE할 때, 1이 되므로 이때만 RAM_RD신호를 DISABLE로 하면 된다.

4.3.3 데이터 래치 회로 구현

데이터 래치 회로는 ADR_CLK의 첫번째 HIGH 에지 트리거에 의해 RED_DATA가 래치되고 두 번째 HIGH 에지 트리거에 의해 GN_DATA가 래치되도록 그림 14와 같이 구현하였다.

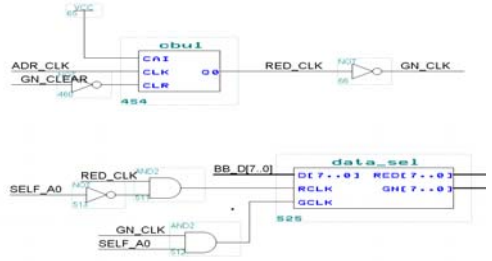


그림 14 데이터 래치 회로
Fig. 14 Circuit of the data latch

4.4 CPU RAM WRITE부 구현

4.4.1 타이밍도

그림 15는 CPU가 RAM으로 데이터를 WRITE할 때 타이밍 차트를 나타낸 것이다. 그림 15에서 A15와 ALE, PSEN그리고 WR신호를 이용하여 WRITE 타이밍을 설계하였다.

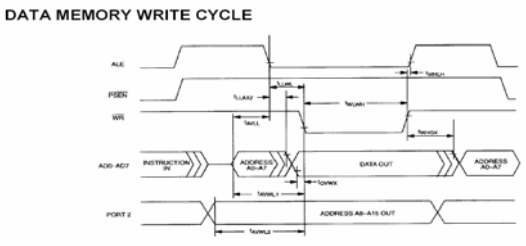


그림 15 데이터 메모리 WRITE 타이밍도
Fig. 15 Timing diagram of the data memory WRITE

4.4.2 RAM_WR 생성 회로

메모리 8000H~FFFFH영역을 DISPLAY RAM 뱅크로 사용하기 때문에 이 메모리 영역을 나타내는 A15와 CPU의 WR신호를 이용하여 RAM WR신호를 만들었다. 그리고 CPU가 메모리 WRITE할 때, 타이밍도에서 나타낸 것처럼 PSEN를 이용하여 버스제어 신호인 CHANGE 신호를 만들었다. RAM_RD 신호 생성 회로는 그림 16과 같다.

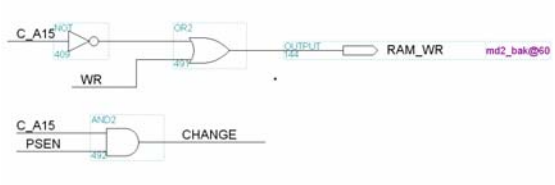


그림 16 RAM_RD 신호 생성 회로
Fig. 16 Circuit of the RAM_RD signal generation

4.4.3 CPU_A[7-0] 생성 회로

CPU의 하위 어드레스 버스는 데이터 버스와 같이 사용하므로 ALE를 이용하여 8BIT D 플립-플롭으로 어드레스 버스를 분리한다. CPLD의 입력 핀 수를 줄이기 위해 사용한 로직이다. CPU_A[7-0] 생성 회로는 그림 17과 같다.

그림 17 CPU_A[7-0] 생성 회로.
Fig. 17 Circuit of the CPU_A[7-0] generation.

4.5 스캐닝 신호 생성부 구현

4.5.1 LED 모듈 타이밍

그림 18은 표출 모듈에 대한 타이밍도로 한 줄에 대한 RED, GREEN 데이터를 클럭으로 실어 보낸 후, 래치와 OE를 주고 그 라인에 대한 어드레스 정보를 주면 DISPLAY된다.

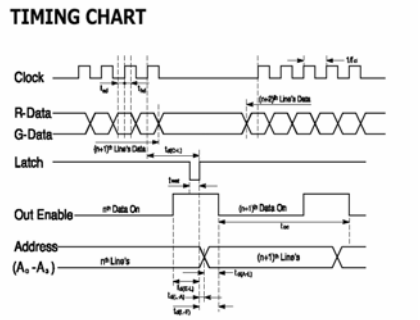


그림 18 LED 모듈 타이밍도.
Fig. 18 Timing diagram of the LED module.

4.5.2 CPLD GENERATE CLK, TCH, OE

CLK 클럭을 가공하여 MODULE CLK, OE, LATCH 신호를 만들어낸다. ALE와 CHANGE 신호를 이용하여 RAM READ에 필요한 ADR_CLK를 만들고, CLEAR은 초기화 신호이다. 또한, OVER는 1 라인의 끝을 나타내는 신호이다.

ADR_CLK은 출력 핀으로 RAM READ 클럭이고, M_CLK은 모듈 CLK, MODULE_OE는 모듈 OUT ENABLE 신호를 각각 나타내며, LATCH는 모듈 래치 신호를 나타낸다.

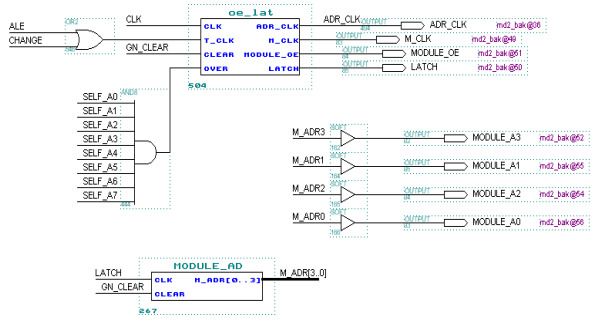


그림 19 CPLD 신호 생성 회로
Fig. 19 Circuit of the CPLD signal generation

그림 20은 위의 그림 19의 oe-lat의 내부 구현회로를 나타낸 것이다.

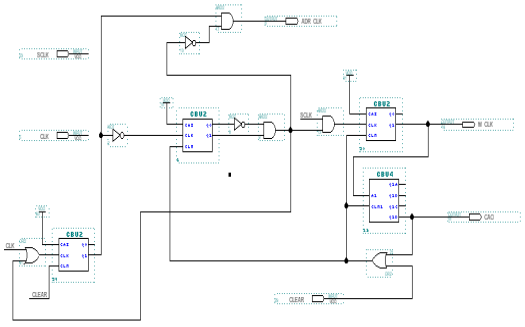


그림 20 oe-lat의 내부 회로.
Fig. 20 Internal circuit of the oe-lat.

그림 21은 ADR_CLK와 MODULE_CLK의 파형도를 나타낸 것이다.

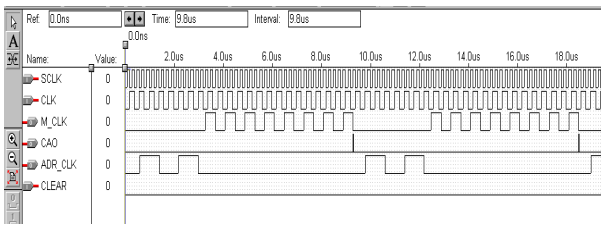


그림 21 ADR_CLK와 MODULE_CLK의 파형도.
Fig. 21 Waveform of the ADR_CLK and MODULE_CLK.

ADR_CLK에서 읽어 들인 RED, GREEN 데이터인 1 바이트를 비트 단위로 가공하여 모듈 클럭과 함께 내보낸다.
그림 22은 MODULE_OE와 LATCH의 파형을 나타낸 것이다.

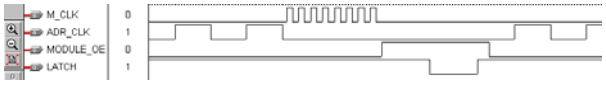


그림 22 MODULE_OE와 LATCH의 파형
Fig. 22 Waveform of the MODULE_OE and LATCH

다음 그림 23은 바이트를 비트로 출력하는 모듈 데이터 변환 회로를 나타낸 것이다.

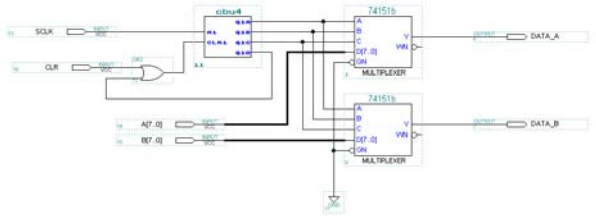
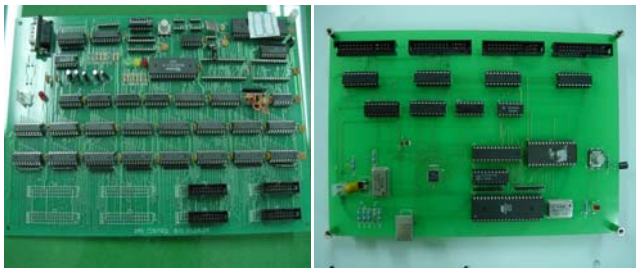


그림 23 모듈 데이터 변환 회로
Fig. 23 Conversion circuit for the module data

5. Sub Control Board 개발 결과

앞에서의 구현 회로를 이용하여 Sub Control Board를 제작하였다. 아래 그림 24의 (a)는 기존에 사용하던 Sub Control Board이고 그림 24의 (b)는 이번 연구를 통하여 제작된 Sub Control Board이다.



(a) (b)

그림 24 서브 제어 보드.
(a) 기존의 보드, (b) 새로 개발된 보드
Fig. 24 Sub Control Board.
(a) Conventional board, (b) New board.

그림 24에서 볼 수 있는 것처럼 기존의 Board에 비하여 칩의 수도 많이 줄어든 관계로 보드 자체의 안정성이 크게 향상되었으며, 유지보수의 편리성이 크게 향상되었다. 또한, 소프트웨어 엔진을 사용하여 시스템의 사양 변화에도 유연하게 대처할 수 있게 되었다.

6. 결 론

ITS시스템의 교통정보 전달을 위한 장치로써의 VMS의 역할은 더욱더 확장되어 가고 있는 추세이며, 한국도로공사가 주관하는 고속도로용 VMS, 건설기술연구원에서 주관하는 국도용 VMS, 서울시의 외곽순환도로 VMS, 각 지방 자치단체 VMS 등, 이전의 단순한 문구와 일반적인 교통안내문의 전송에서 이제는 각종 그래픽을 사용한 보다 세밀하고 구체적인 교통안내 문구를 표출하고 있어 이전의 단지 참조만을 위한 시스템에서 필수적인 시스템으로 변화되어 가고 있어 시스템의 안정성과 고품질화가 절실히 요구되고 있다.
이전에는 50mm 픽셀을 사용하여 정보를 표출하였으나, 이

제는 30mm, 25mm 픽셀을 사용하고 있고, 최근에는 보다 더 정밀한 해상도를 가지는 16×16, 16×8 등으로 모듈화된 240mm, 300mm 모듈이 사용되고 있는 추세이다.

본 논문에서는 처리속도의 제한을 극복하기 위한 USB를 이용한 인터페이스 구현, 사양의 변경에 따라 하드웨어를 다시 개발해야 하는 문제점을 해결하기 위한 소프트웨어 엔진 개발, CPLD를 이용하여 고속 메모리 스캔부 구현을 통하여 서브컨트롤 부를 개발하였다.

참 고 문 헌

[1] 박상철, 이승무, 강무성, “Level Up OrCAD,″ 성안당, 2001.

[2] 조용범, 김종오, 김훈학, 이승한, “실무와 예제를 중심으로 한 OrCAD,″ 북두출판사, 2002.

[3] NetChip Manual, ″NET2888 Evaluation Kit,″ http://www.netchip.com/product/2890eb.html, 2001.

[4] 이승호. 박용수. 박군종. 이주현, “ALTERA MAX+II를 사용한 디지털 논리회로 설계의 기초와 활용,”, 북두출판사, 1999.

[5] 편집부, “화상처리 회로기술의 모든 것,” 도서출판 세운, 1988.

[6] NetChip, “NET2890 Programming Flowchart,” NetChip Technology, Inc., 1999.

[7] ALTERA, “MAX7000 Programmable Logic Device Family Data Sheet,” ALTERA, 2001.

저 자 소 개

신 재 흥 (申 載 興)

1986년 한양대 전자공학과 졸업, 1991년 동대학원 전자공학과 졸업(공학석사), 1997년 동대학원 전자공학과 졸업(공학박사), 1997년 3월 ~ 현재 동서울대학 컴퓨터시스템과 조교수.

Tel : 031) 720 - 2175

Fax : 031) 720 - 2294

E-mail : jhshin@haksan.dsc.ac.kr

김 홍 렬 (金 弘 烈)

1986년 한양대 전자공학과 졸업, 1988년 동대학원 전자공학과 졸업(공학석사), 1997년 동대학원 전자공학과 졸업(공학박사), 2000년 3월 ~ 현재 동서울대학 컴퓨터정보과 조교수.

Tel : 031) 720 - 2153

Fax : 031) 720 - 2292

E-mail : rhkim@haksan.dsc.ac.kr

이 상 기 (李 相 基)

1987년 한양대 전자공학과 졸업, 1991년 ~ 현재 봉오전자공업(주) 연구소장, 1999년 3월 ~ 현재 동서울대학 컴퓨터시스템과 겸임교수.

Tel : 031) 421 - 6007

Fax : 031) 421 - 8622

E-mail : lseed82@hotmail.com