論文95-32A-9-18

이동통신 단말기용 16 비트 마이크로콘트롤러의 주변장치 개발

(Development of Peripheral Units of the 16 bit Micro-Controller for Mobile Telecommunication Terminal)

朴 性 模 *, 李 南 佶 **, 金 炯 吉 **, 金 瑞 均 **

(Seong Mo Park, Nam Gil Lee, Hyoung Gil Kim, and Seo Gyun Kim)

요 약

휴대형 정보기기의 소형, 경량, 저전력 소비화는 칩의 고집적, 저전압화를 요구하며 단말기 내의 소자수의 최소화를 요구하고 있다. 이에 따라 이제까지는 독립적인 칩으로 존재하던 소자들이 모듈화되어 보다 더 큰 칩의 일부로서 포함되고 있다. 본 논문은 이동통신 단말기용 마이크로콘트롤러에 내장되는 주변장치 개발에 관한 것이다. 주변장치는 DMA(Direct Memory Access) 제어기, 인터럽트 제어기, 타이머, Watchdog 타이머, 클럭 발생기 및 전력 제어장치로 되어있으며, EU(Execution Unit), BIU(Bus Interface Unit)와 함께 16 비트 마이크로콘트롤러를 구성하여 차세대 디지틀 이동통신 단말기용 ASIC 칩의 CORE로 사용되도록 설계되었다. 먼저 마이크로콘트롤러의 전체블럭을 VHDL behavioral 모델로 기술하여 전체적인 기능을 시뮬레이션하였다. 다음에 Watchdog 타이머와 클럭 발생기 및 전력 제어장치는 VHDL 합성기로 직접 합성하였고 나머지 주변장치들은 Compass Design Tool을 사용하여 설계하고 시뮬레이션하였다.

Abstract

The trend of compact size, light weight, low power consumption in the portable telecommunication equipments demands large scale integration and low voltage operation of chips and the minimization of the number of the components in the telecommunication terminal. According to the trend, existing chip components are modulized and are integrated as a part into a bigger chip. This paper is about the development of the peripheral units of micro-controller for mobile telecommunication terminal. Peripherals consist of DMA controller, Interrupt controller, timer, watchdog timer, clock generator, and power management unit. They are designed to be integrated with EU(Execution Unit) and BIU(Bus Interface Unit) into a 16 bit micro-controller which will be used as a core of an ASIC for next generation digital mobile telecommunication terminal. At first, whole block of the micro-controller was described by VHDL behavioral model and simulated to verify its overall operation. Then, watchdog timer, clock generator and power management unit were directly synthesized by using VHDL synthesis tool. Rest of the pheriperal units were designed and simulated by using Compass Design Tool.

* 正會員, 全南大學校 컴퓨터工學科
(Dept. of Computer Eng. Chonnam Nat'l Univ.)
** 學生會員, 全南大學校 電子工學科
(Dept. of Elec. Eng. Chonnam nat'l Univ.)

※ 본 연구는 92년도 전남대학교 학술연구비의 지원을 받아 수행되었음.
接受日字: 1995年4月2日, 수정완료일: 1995年8月28日

Ⅰ. 서 론

반도체 제조 및 설계 기술의 발달로 오늘날 집적회로의 개발 추세는 저 소비 전력화, 고집적화, 다기능 집적화를 향하여 나아가고 있다. 종전에는 독립된 칩으로 사용되던 마이크로프로세서 또한 고집적화된 칩의 일부에 포함되어 사용되고 있고, CPU에 다양한 주변장치를 집적한 단일 칩 마이크로콘트롤러들이 ASSP (Application Specific Standard Product)화 추세에 따라 개발되어 산업, 국방, 항공 공학, 로보트 공학, 화상처리 공학 등의 여러 분야에서 사용되고 있다. 지금까지는 8비트 마이크로콘트롤러를 주로 사용하였으나, 빠른 계산 능력과 큰 메모리 주소 공간의 요구, 또한 멀티미디어 응용에 따른 많은 양의 데이터에 대한 빠른 처리능력 요구 등의 이유로 16 비트 혹은 32 비트의 마이크로콘트롤러에 대한 수요가 점점 늘어가고 있는 실정이다[1].

이동통신 단말기 시스템에서 마이크로콘트롤러는 사용자와 기지국과의 호처리(call processing) 기능과 각종 사용자 편의 기능을 수행하기위해서 단말기 시스템의 모든 동작을 제어한다. 이동통신방식이 아날로그 방식에서 디지틀 방식으로 전환되어짐에따라 단말기 시스템의 하드웨어 및 소프웨어의 복잡성이 급증하여 16 비트 마이크로콘트롤러의 사용이 필요하다[2]. 이동통신 단말기의 하드웨어는 주변장치와의 인터페이스가 인터럽트로 처리되는 부분이 많아 인터럽트 제어기가 필요하고, 타이머 기능, DMA 기능 등을 많이 필요로 하며, 단말기의 중요한 과제 중의 하나인 전력 손실을 줄이기 위하여 전력 제어장치가 필요하다. 이에따라 단말기용 마이크로콘트롤러는 CPU core외에도 DMA 제어기, 인터럽트 제어기, 칩 선택 장치, 타이머, 클럭 발생기 및 전력 제어기, Watchdog 타이머 등의 내부 주변장치들로 이루어져있다. 16 비트 CPU core는 BIU(Bus Interface Unit)와 EU(Execution Unit)으로 나누어서 설계되었고[3], 내부 주변장치들은 개개의 독립된 기능블럭으로 분할하여 설계되었다. Watchdog 타이머와 클럭 발생기 및 전력 제어장치는 설계 초기에 VHDL behavioral 모델에서 직접 합성이 이루어졌기에 본 논문에서는 그 기능만 간략히 설명하였다. 나머지 주변장치들은 Compass 사의 Design Tool을 이용하여 설계되었고 개별블럭에 대한 논리시뮬레이션을 통하여 그 기능을 검증하였다.

주변장치들의 상호 작용 및 CPU core와의 인터페이스 상태에 대한 기능 검증은 Compass Tool로는 시뮬레이션할 수가 없어서 Schematic으로부터 Netlist를 추출하여 VHDL 모델로 바꾼 다음 Synopsys VHDL 시뮬레이터를 이용하여 검증하였다.

본 논문에서는 주변장치의 구조 및 기능에 대해서 설명하고, 마이크로콘트롤러 시스템안에서 주변장치가 어떻게 인터페이스 되는가를 보여주며, 각 기능블럭들의 설계와 시뮬레이션 결과에 대해서 논한다.

Ⅱ. 주변장치의 구조 및 설계

그림 1은 이동통신 단말기용 16 비트 마이크로콘트롤러의 전체 시스템 블럭도이다. 모든 내부 주변장치는 프로그램이 가능하고 소프트웨어에 의해 제어된다. 16 비트 제어레지스터의 특별한 설정은 주변장치 제어 블럭(PCB)이라 불리는 I/O나 메모리 공간의 256 바이트 블럭에서 공급된다[4].

주변장치는 개개의 기능블럭으로 분할하여 설계하였고 Compass Design Tool을 사용하여 각각의 기능블럭의 동작을 시뮬레이션하였다. 회로 설계는 Compass 사의 0.8 ㎛ CMOS CBIC 라이브러리를 이용하여 수행하였고, 전력소모를 줄이기위해 하드웨어를 최소화하도록 설계하였다.

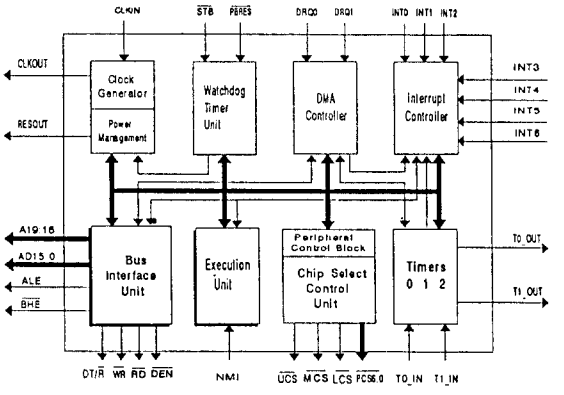


그림 1. 마이크로콘트롤러의 블럭도
Fig. 1. Block Diagram of the Microcontroller.

1. DMA 제어장치

DMA 제어장치는 2개의 독립적인 고속 데이터 전송 채널을 갖고 있으며, DRQ0, DRQ1, Timer2에 의한 DMA Request 동작, 그리고 DMA 채널 자체에 의한

동작등 네 가지 동작 모드로 동작하도록 설계되었다. 보통 DMA의 동작을 블럭 이동이나 사이클 스틸의 형태로 대별할 수 있으나, 본 시스템의 구현에 있어서는 블럭 이동의 형태에서도 필요에 따라서는 DMA 요청을 잠시 중단하였다가 다시 연속하여 실행할 수 있게 하는 기능을 두었다[5]. 메모리와 I/O 공간 사이에는 2개의 독립적인 고속 데이터 전송 DMA 채널이 제공되고, 각 DMA 채널은 매 데이터 전송 후 증가나 감소하는 20 비트 Source와 Destination Pointer를 유지한다. 각 채널은 그 동작을 정의하는 20 비트 Source Pointer 레지스터, 20 비트 Destination Pointer 레지스터, 16 비트 전송 카운트 레지스터, 그리고 16 비트 제어 레지스터등 6개 레지스터를 갖고 있다. 전송 카운트 레지스터에는 수행할 DMA 전송의 수가 명기되며, 64K 바이트나 워드까지의 전송이 가능하다. 제어 레지스터에 의해 동작 사양이 결정되며, DMA 동작중 이들 레지스터 값의 변화는 즉시 현재 수행중인 DMA 동작에 영향을 미친다.

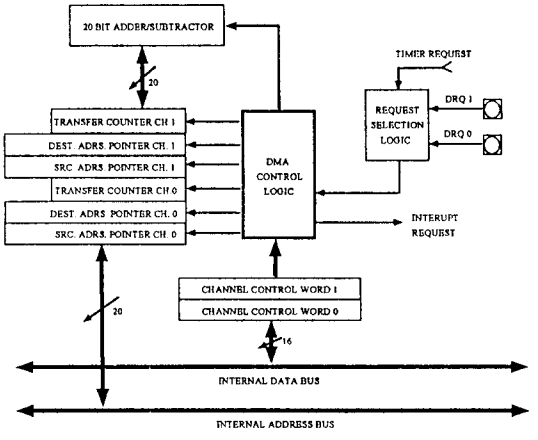


그림 2. DMA 제어장치의 블럭도
Fig. 2. Block Diagram of the DMA Controller.

DMA 제어장치의 동작모드는 외부적 요구인 DRQ0, DRQ1 입력에 의한 것과 내부적 요구인 타이머-2 나 DMA 채널 자체에 의한 것의 형태로 구분된다. 그림 2는 DMA 제어장치의 블럭도인데 각 채널에 해당하는 레지스터 그룹과 DMA 요청 신호를 받아 우선도에 따라 채널을 선택하는 Request Selection Logic, 그리고 전체 회로의 동작을 규정하는 제어 로직으로 구성되어 있다.

DMA 제어장치는 전체적인 타이밍을 제어하는 상태 머신 블럭, 주소를 통해 각 채널의 레지스터를 선택하는 Decoder 블럭, DMA의 동작 모드를 결정하는 제어 레지스터 블럭, Source와 Destination의 주소를 만들어 내보내는 주소 카운터 블럭, DMA 전송량을 카운트해서 터미널 카운트 신호를 발생하는 전송 카운터 블럭, 우선순위를 결정하는 Priority 블럭, 그리고 마지막으로 CPU에게 DMA 전송에 관한 정보를 주는 Sbar 블럭으로 이루어졌다. 이들 블럭들은 clkperi란 클럭 신호에 의해서 동작되는 상태 머신에 의해서 제어되도록 설계되었다[6]. 상태 머신은, 그 단계가 S1에서 S8까지, 그리고 Sx, S9상태의 10단계로 나뉘어져 있으며, 그림 3에 보인 상태 머신의 동작은 다음과 같다.

시스템을 리셋 시키는 resint 신호에 의해 상태 머신은 초기 상태값을 갖게 되는데, 이 때 S1을 제외한 모든 상태는 low 상태를 유지한다. S1 상태에서 DMA 요청 신호가 있을 때까지 기다리며 DMA 요청이 있으면 S2로 이동한다. S2의 상승 모서리에서 bus_rqst_dma 신호를 CPU에 보내고, CPU로부터 bus_ack_dma 신호를 받아 S3 상태로 넘어간다. S3 상태는 버스 사이클의 시작인 T1에 해당하고 S4 상태를 지나 S5 상태에서 bus_done 신호를 받아 Source에서 데이터를 fetch하는 사이클을 끝내고 S6 상태로 이동한다. S6 상태에서 S7 상태를 지나 S8 상태에서 bus_done 신호를 받고 Destination에 데이터를 deposit하는 사이클이 종료된다. 1회 DMA 전송시 최소 두 버스 사이클이 필요하게 되는데, 이것은 Source에서 데이터를 Fetch(S3~S5:Fetch Cycle)하고 Destination에 저장(S6~ S8:Deposit Cycle)하는 것을 고려한 것이다. S8 상태에서 TC(Terminal Count)가 1(전송완료신호)이면 S9 상태로 들어가 전송을 끝내게 되고, 그렇지 않으면 S3 상태(블럭이동)나 Sx(Cycle Steal)를 지나 S2 상태로 가 전송이 계속해서 이루어진다.

Destination에 동기화된 전송인 경우에는 bus_rqst_dma신호를 연속하여 보내지 않고 1 버스 사이클을 쉬게 한다. 이것을 상태 머신상에서는 S8 다음 S3로 가지 않고 Sx를 거쳐 1 클럭을 기다려 S2 상태로 가게 하여 1 버스 사이클을 잃게 함으로써 Cycle Steal전송을 이루게 하였다. S9 상태에서 int_ack_dma신호를 받으면 S1 대기상태로 이동하여

DMA 제어장치는 또 다른 DMA 요청이 있을 때까지 기다리게 된다. 한편 DMA 전송중 DMA 요청 신호가 중지되면 DMA 제어장치는 현재 수행중인 DMA 전송 사이클의 S8 상태에서 S1 상태로 이동하게 되고 DMA 요청이 다시 재기될 때 연속하여 DMA 전송을 시작하게 된다.

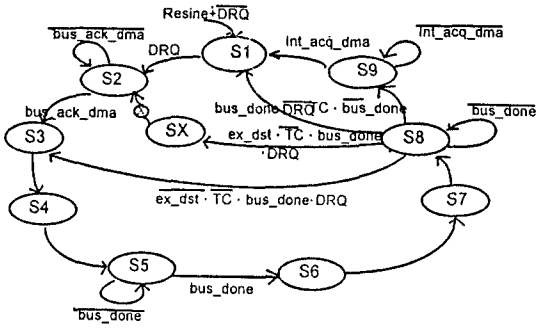


그림 3. DMA 제어장치의 상태도
Fig. 3. State Diagram of the DMA Controller.

2. 인터럽트 제어장치

인터럽트 제어장치의 주요 기능은 CPU 내부와 외부에서 발생하는 인터럽트 소스의 서비스 여부를 우선도에 준하여 판정하여 인터럽트 요청 신호를 BIU에 전달하는 기능이다. 그리고 BIU가 인터럽트를 인식하여 인터럽트 인식 신호를 보내오면 인터럽트 제어장치는 발생된 인터럽트에 대한 벡터값을 내보내게 된다. 인터럽트 서비스 루틴이 종료하면 인터럽트 제어장치는 그에 대한 적합한 조치를 하게 된다[7].

인터럽트 제어장치가 인터럽트 서비스 루틴을 수행하고 있는 중에 그 보다 더 높은 우선도를 가지는 소스가 인터럽트를 요청하면 더 높은 우선도의 인터럽트를 받아들여 현재 진행중인 서비스 루틴을 잠시 중지하고 새로운 서비스 루틴을 수행하게 된다. 또한 NMI가 발생하면 레지스터의 한 비트를 세트시켜 저장하고 DMA동작을 중지시키는 신호를 보내게 된다.

그림 4는 인터럽트 제어장치를 기능별로 8개의 블럭으로 나누어 구성한 블럭도이다.

인터럽트 제어장치의 각 블럭별 기능 설명은 다음과 같다. 우선 Priority 블럭은 외부에서 요구하는 인터럽트 소스를 받아서 서비스 여부를 판정하고 인터럽트 인식신호와 인터럽트 벡터 테이블에 의거한 벡터값을 BIU에 전달한다. Wt_rd 블럭은 주소값을 디코드하여 각 레지스터를 읽거나 쓸 수 있는 신호를 발생하는 블럭이다. Request 블럭은 CPU 내부, 외부에서 발생하는 인터럽트 소스를 저장하는 블럭이다. Insts 블럭은 NMI 발생을 기록하여 DMA 중지신호를 내보내고 타이머로 인한 인터럽트를 구분시키기 위한 블럭이다. Inserv 블럭은 현재 서비스중인 인터럽트 소스를 나타내며, EOI(End of Interrupt) 명령을 위한 정보를 포함하는 블럭이다. Poll과 Prmsk 블럭은 polling 정보를 가지고 있는 블럭과 현재 진행중인 인터럽트의 우선도 값을 가지고 있는 블럭이다. Icon 블럭은 인터럽트 소스의 우선도값을 가지고 있는 제어 레지스터와 소스의 마스크 여부를 결정하기 위한 정보가 포함된 블록이다.

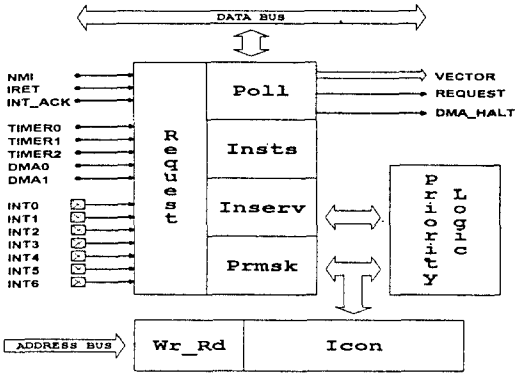


그림 4. 인터럽트 제어장치의 블럭도
Fig. 4. Block Diagram of the Interrupt Controller.

인터럽트 제어장치는 내부에서 기능별로 8개 블럭으로 구분시켜 역할을 수행하도록 설계하였다.

설계된 인터럽트 제어장치는 이동통신의 단말기 기능에 적합하도록 CPU 외부 인터럽트 소스 7개, 타이머 소스 3개, DMA 소스 2개를 가지고 있다. 인터럽트 소스가 발생하면 인터럽트 제어장치에서 처리되는 순서는 아래와 같다.

인터럽트가 발생하면 인터럽트 제어장치는 Request 레지스터의 적절한 비트를 세트하여 인터럽트 발생 사실을 기록한다. 다음으로 Mask 레지스터내의 정보에 따라 인터럽트 소스에 마스크가 걸려 있는지 아닌지를 판정한다. 마스크가 안 걸려 있으면 인터럽트 제어장치는 우선순위로직을 통해 In-Service 레지스터에 기록

된 현재 진행중인 인터럽트와 새로 들어온 인터럽트의 제어 레지스터를 서로 비교해서 새로운 인터럽트에 대한 서비스 여부를 결정한다. 새로 들어온 인터럽트의 우선순위가 높으면 인터럽트 제어장치는 BIU에 인터럽트 요청신호를 보내게 된다. 인터럽트 요청신호를 받은 BIU는 얼마 후 인터럽트 인식신호를 보내게 된다.

CPU가 직접 인터럽트 제어장치 내의 Poll 레지스터를 판독하여 인터럽트 발생 여부를 감지할 수도 있는데 이때에는 인터럽트 발생여부를 저장해둔 Request 레지스터의 해당비트가 clear되고 In-Service 레지스터의 해당 비트가 세트된다. 또한 서비스가 허락된 인터럽트의 우선도가 Priority Mask 레지스터에 저장된다.

인터럽트 서비스 루틴이 수행되고 있는 중에 그 보다 더 높은 우선도를 가지는 소스가 인터럽트를 요청하면 위의 과정을 반복하여 인터럽트 요청신호를 보낸다. 인터럽트 인식신호를 전달받으면 새로운 인터럽트를 받아들여 현재 진행중인 서비스 루틴을 잠시 중지하고 새로운 서비스 루틴을 수행하게 되므로 Nesting이 제공되어진다.

인터럽트 서비스 루틴은 본래의 프로그램으로 돌아오기전 EOI 명령을 수행하게 되는데 이 명령에 의해 인터럽트 서비스 루틴이 종료하게 된다. 소프트 웨어에 의한 EOI 명령은 EOI 레지스터의 내용을 수정시키고 그 수정된 값에 따라 In-Service 레지스터의 해당비트를 clear시킨다. 또한 이 레지스터의 15번째 비트에 따라 특정한 인터럽트를 clear할지 가장 최근에 서비스를 받은 인터럽트를 clear할지를 정하게 된다.

3. 칩 선택 장치

칩 선택 장치의 주요 기능은 20 비트의 주소를 버스 인터페이스 장치로부터 내부 버스를 통하여 전달 받아 그 값에 따라 적절한 칩 선택 핀을 활성화시켜주는 것이다.

칩 선택 핀으로는 UCS(Upper Chip Select), LCS(Lower Chip Select), 그리고 4개의 MCS (Middle range Chip Select) 및 7개의 PCS(Peripheral Chip Select)가 있다. 아울러 각 칩 선택을 조정하는 제어 레지스터로는 UMCS (Upper Memory Chip Select), LMCS(Lower Memory Chip Select), PACS(Peripheral Address Chip Select), MMCS(Middle range Memory Chip Select) 및 MPCS(Middle range Peripheral Chip Select)가 있는데 이들 레지스터에서 몇 개의 대기 상태를 버스 사이클에 삽입하여야 할지의 여부와 버스 인터페이스 장치가 Ready 로직에 필요한 신호를 생성하는데 사용하는 정보를 버스 인터페이스 장치에 전달한다[8]. UCS를 제외한 모든 칩 선택 신호는 각 칩선택 신호 발생을 조정하는 값이 사용자 소프트웨어에 의하여 최소한 한 번 이상 지정된 후에나 발생시킬 수 있도록 되어 있어, 제어 레지스터들이 사용자 프로그램에 의하여 그 값이 지정된 사실이 있는 지의 여부를 기억하는 기능도 수행한다.

그림 5는 칩 선택장치의 블럭도이다. 리셋이나 파워 다운시의 제어부와 레지스터 값을 갱신하는 로직 및 칩 선택 신호를 내보내는 로직으로 나뉘어 있다.

칩 선택 장치는 select와 sel_out의 2개 모듈로 구분되며, CPU core로부터 어드레스를 전달받아 그 값에 따라 적절한 칩 선택 핀을 활성화시켜 주도록 설계되었다. Select 로직은 제어 레지스터의 갱신과 선택을 수행하는데 CPU로부터 8비트 주소버스값과 함께 읽기 또는 쓰기 명령을 받아 UCS, LCS, MCS, PCS를 위한 제어레지스터들의 값을 갱신하고 각각 wait_cnt를 발생시킨다.

Sel_out 모듈은 리셋 혹은 파워 다운 모드에서의 리셋과 모듈에서 생성된 내부 칩선택 신호를 외부로 내보내는 기능을 수행한다. 이 장치에 있는 3개의 핀들은 마이크로콘트롤러에서 다른 외부 메모리를 선택하는데 쓰인다. UCSbar 핀은 메모리 공간의 상위에 위치한 메모리 장치를 구동시킨다. 이 핀은 프로그램에 의해서 정의된 크기의 메모리의 사용을 허용하고 또 필요한 대기 상태의 수를 결정한다. LCSbar 핀은 메모리 주소 00000H에서 시작하는 메모리 장치를 선택한다. UCSbar 핀과 마찬가지로, 메모리 크기와 대기상태의 수는 프로그램 가능하다. MCSbar 핀은 시작 번지와 메모리 크기가 함께 프로그램 되어진다. UCS, MCS, LCS의 최대영역은 각각 512K, 256K, 256K이다.

또한 주변장치 칩 선택 기능을 가지는데 PCSbar6 - PCSbar0으로 7개의 외부 주변장치를 선택한다. 기본 I/O 주소는 128바이트의 포트 주소 블럭 크기를 가지고 1K 바이트의 간격으로 프로그램된다.

제어 레지스터로는 UMCS, LMCS, PACS, MMCS, MPCS가 있는데 이들중 UMCS와 LMCS는

각각 UCS가 구동되는 시작번지와 LCS가 구동하는 영역의 끝번지를 결정한다. 이 레지스터의 최대 영역은 512K이다.

PACS는 PCS0가 활성화되는 시작번지를 결정하여 7개의 PCS가 이 시작번지로부터 각각 128바이트의 영역에서 활성화되도록 한다. 아울러 MMCS는 MCS가 활성화되는 시작번지를 결정하며, MCS가 활성화되는 영역의 크기는 MPCS에 의하여 결정된다. 리셋 발생시 UMCS는 FFFFH의 초기값을 갖게 되어 있다.

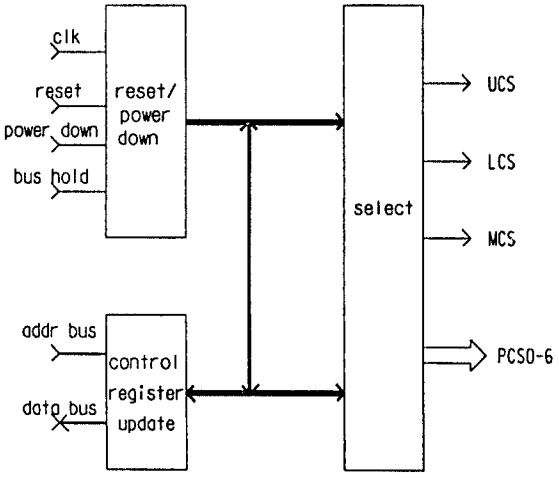


그림 5. 칩 선택 블럭도
Fig. 5. Block Diagram of the Chip Select.

4. 타이머 제어장치

타이머 제어장치는 3개의 16비트 프로그램 가능한 타이머를 제공하며, 이중 타이머 0과 1은 채널당 2개, 모두 4개의 외부 핀에 연결된다. 이 두개의 타이머는 외부 사건을 카운트하고 시간을 기록하는 역할을 수행한다. 외부 핀에 연결되어 있지 않은 타이머2는 실시간 코딩과 시간 지연 응용, 그리고 타이머0과 1에 대한 prescaler, DMA 요구 소스로서의 역할을 수행한다[9]. 각 타이머는 매 4번째 CPU 클럭 사이클에서 서비스되며, 따라서 내부 클럭주파수의 1/4의 속도로 동작한다.

타이머의 프로그램 가능한 선택 사항은 3가지가 있는데, 첫째로 3개의 모든 타이머가 터미널 카운트에서 정지하거나 계속 동작할 수 있다. 다음은 타이머0과 1은 내부와 외부 클럭중 어느 것을 사용할 수 있으며, MAX COUNT 레지스터를 alternate 모드로 사용할 수 있고, 외부 사건에 리트리거하기 위해 설정될 수도 있다. 또한 터미널 카운트에 인터럽트를 발생하도록 프로그램될 수 있다. 그림 6는 타이머 제어장치의 블럭도이다. 3개의 타이머는 각각 인터럽트를 발생시키고 타이머 0, 1은 외부 신호를 위한 IN, OUT 단자가 있다.

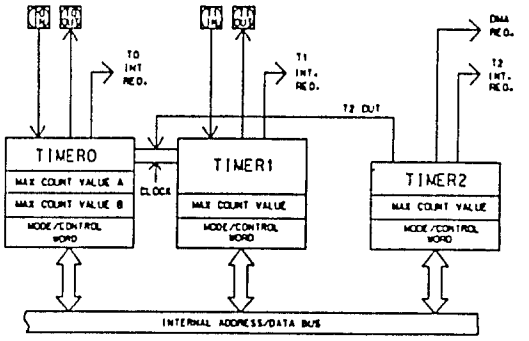


그림 6. 타이머 블럭도
Fig. 6. Block Diagram of the Timer.

타이머 제어장치의 설계는 각 타이머의 제어 신호를 발생시키는 입력 로직과 클럭의 1/4분주를 발생시키는 로직, 그리고 3개의 타이머로 구성되어 있다.

입력 로직은 조합논리회로로서 각 레지스터의 읽기 쓰기를 위한 신호를 발생시킨다. Clk_qtr 로직은 CPU 클럭주파수의 1/4에 해당하는 주파수를 갖는 클럭을 발생시킨다. 이 로직은 타이머가 작동할 때만 클럭을 발생시키므로 절전효과를 충분히 거둘 수 있다. Tmr0, Tmr1 로직은 매 4번째 클럭 사이클에서 동작된다. 이 두개의 타이머는 터미널 카운트에서 정지하거나 동작할 수 있으며, 내부 또는 외부 클럭을 프로그램에 의하여 선택적으로 사용할 수 있다. 또 외부 신호에 의해서 리트리거가 되도록 설정할 수 있다. 이들은 타이머 2에 의해서 발생된 클럭을 사용할 수 있다. 이중 타이머 0은 MAX COUNT 레지스터 A와 B를 사용하여 Alternate 모드로 사용할 수 있다. 동작모드에서는 각각 one shot 모드와 연속 모드로 분류되어 동작한다. MAX COUNT에 도달한 경우 INT 비트에 의해 인터럽트 여부를 결정하여 TOUT0와 TOUT1으로 신호를 내 보낸다.

Tmr2 로직은 타이머 0과 1에 대한 prescaler로 작용하며, DMA 요구 소스로서의 역할을 수행한다. 타이머 2는 단일 카운트 모드로만 동작하며, one shot

모드와 연속 모드로 나뉘어 동작한다. MAX COUNT에 도달한 경우 INT 비트에 의해 인터럽트 여부를 결정하여 DMA 요구 신호를 발생시킨다.

타이머는 PCB에 위치한 11개의 16 비트 레지스터에 의해 제어되며, 이 중 Count 레지스터는 타이머의 현재 값을 지니고 있어 타이머의 동작 여부에 관계없이 읽기나 쓰기가 가능하고 매 타이머 발생때마다 그 값을 증가시킨다. 타이머가 카운트 할 수 있는 최대치를 정의하는 MAX COUNT 레지스터는 타이머 카운트가 최대치에 도달한 경우, 그 동일 클럭 내에서 0으로 리셋된다. 타이머 0과 1의 경우엔 2번째의 MAX COUNT 레지스터를 가지고 있어, 두 MAX COUNT 값 사이의 alternate counting 을 가능하게 한다. 이 경우엔 제어 레지스터의 RIU 비트에 의해 어느 MAX COUNT의 값이 카운터의 값보다 작게 설정되었을 경우의 동작을 정의하며, $0FFFFH까지 카운트한 후 0으로 리셋되고 나서, 다시 MAX COUNT 값까지 카운트한다.

5. 주변장치의 인터페이스

그림 7은 본 논문에서 기술된 주변장치와 CPU core 부분과의 인터페이스를 보여준다.

클럭 발생기 및 전력 제어기는 내부 클럭신호(clkcpu)와 내부적으로 동기된 리셋신호(resint)를 보내 CPU core를 동작시킨다. 마이크로콘트롤러가 파워 다운 모드에서 동작중일때는 전력제어기가 pwrdn 신호를 CPU core에 보내고, 이 신호를 받은 CPU core는 버스 사이클에 필요한 CPU 외부로 나가는 신호를 비활성화시킨다.

Watchdog 타이머는 전력 인가시의 리셋 신호 생성 및 시스템 감시 기능을 수행하기위한 것으로서 시스템이 주기적으로 신호를 보내오지 않을 경우 시스템의 동작이 부적절하다고 판단하여 클럭 발생기 및 전력 제어장치에 리셋 신호(resinbar)를 보낸다.

칩 선택 장치는 버스 사이클의 T1 타이밍에 CPU core로부터 io(칩 선택 요청을 받은 영역이 I/O임을 알리는 신호), cs_rqst(CPU core가 칩 선택을 요청하는 신호), addrphys_cs [19:0] (칩 선택 핀 생성용 20 비트 어드레스)를 받아서 CPU 외부의 주변 칩을 활성화시킨다. T2 사이클에 addrphys를 하이 임피던스 상태로 바꾸고 T3 사이클에 cs_rqst를 비활성화시킨다. T1 사이클에 io, cs_rqst, addrphys를 받은 칩 선택 장치는 곧바로 wait_cnt [2:0] (선택된 영역의 대기 사이클 수)를 CPU core에 보내고 CPU core는 이를 이용하여 대기 사이클(Tw) 삽입 여부를 결정한다.

각 타이머는 외부 또는 내부 클럭킹에 의해서 주기적으로 타이머 인터럽트 요구를 발생시키거나 DMA에 dma_rqst_tmr2를 보낸다.

인터럽트 제어기에서 CPU core에 int_rqst를 보내면 CPU core는 버스 사이클을 통하여 Flag(CPU의 각종 상태 비트들의 집합), CS(Code Segment), IP(Instruction Pointer)를 차례로 스택에 저장하고 int_ack를 high로 한다. int_ack 신호를 받은 인터럽트 제어기는 int_rqst 신호를 low로 하고 int_vector [7:0]를 CPU core에 보낸다. int_rqst신호가 low가 되면 CPU core는 int_ack신호를 low로 떨군다. 그리고 버스 사이클을 통하여 명령어 값에 해당하는 메모리 번지의 내용을 읽어들여 CS, IP 레지스터에 저장하고 새로운 번지에서 명령어를 가져오게 된다.

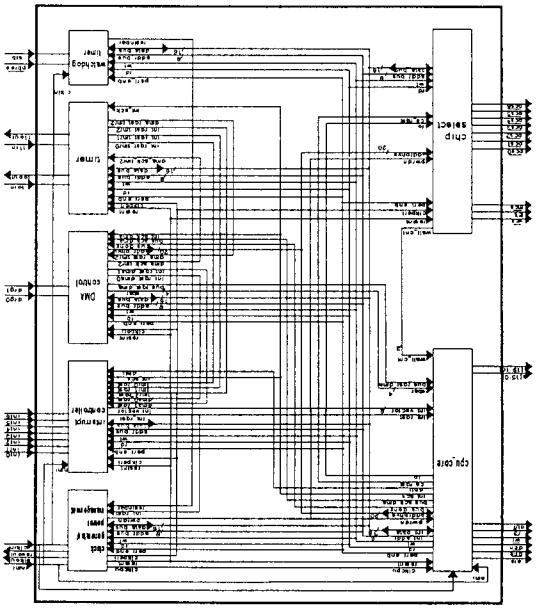


그림 7. 주변장치 인터페이스
Fig. 7. Interface of the Peripheral Units.

DMA 제어장치에서 버스를 요구하는 bus_rqst_dma 신호를 보내면 CPU core에서는 버스 사용을 허락하는 bus_ack_dma 신호를 high로 하고, 2 버스 사이클 동안 high 상태를 유지하는 bus_ack_dma 신호를 받은 DMA 제어기는 DMA Source에 해당하는 DMA 전송이 읽기인지 써넣기인지 여부와 그 단위가

워드인지 바이트인지를 알리는 정보(sbar [5:0]와 addrphys [19:0] 신호)를 보낸다. DMA Destination 동작이 끝나지 않았다면 bus_rqst_dma 신호는 high로 있고 CPU core로부터 다시 bus_ack_dma 신호를 받으면 DMA 제어장치에서는 Destination에 해당하는 sbar, addrphys를 내보낸다. DMA는 bus_rqst_dma 신호를 low로 떨구고 아직 전송할 데이터가 남아 있다면 두 clkperi(주변장치를 동작시키는 클럭)후 bus_rqst_dma 신호를 high로 하여 버스 사용을 요구한다.

Ⅲ. 시뮬레이션

시뮬레이션은 먼저 Compass Tool을 사용하여 각 주변장치별로 이루어졌다. 다음에 주변장치 전체를 시스템과 연계하여 각 기능블럭이 시스템 내에 집적되었을 경우 정상동작하는 지의 여부를 확인하는 것으로 이루어졌다. 본 논문에서는 DMA 제어장치, 인터럽트 제어장치, 칩 선택 장치, 그리고 타이머에 대한 전체 시뮬레이션 과정을 설명하고 부분별 시뮬레이션 결과를 소개한다.

DMA 전송 요구는 외부적으로 DRQ 입력을 통해, 내부적으로 Timer-2가 최대 카운트에 이르렀을 때, 내부적으로 DMA 자체에 의한 요구, 세 가지중 하나이다. DMA 제어장치 시뮬레이션에서는 위에 열거한 DMA 동작 형태를 채널을 바꿔가며 연속하는 것으로서 그 기능을 확인하였다. 이 중 내부적으로 타이머 2가 최대치에 이르렀을 때 발생하는 DMA 요청 신호로서 Channel-0을 사용한 DMA 전송과정을 그림 8에 보였다. 동작 모드상 연속모드의 형태이나 타이머에 의한 요청은 적정 시간에만 순간적으로 일어나므로 실제적인 DMA 동작은 사이클 스틸의 형태를 취하고 있고, 최종적으로 TC(terminal count)에 의해 종료된다.

인터럽트 제어장치는 주요 동작을 세 부분으로 나누어 시뮬레이션해보았다. 우선 인터럽트 제어장치를 초기화시킬 때 각 제어 레지스터에 정확한 초기 값이 설정되는지를 확인하였다. 둘째, CPU의 Write 신호에 의해 데이터 버스를 통해 인터럽트 제어장치의 여러 레지스터에 원하는 값이 정확히 저장되는가 확인하였다. 셋째, 인터럽트 제어장치의 몇 가지 중요 기능 동작을 확인하는 것으로 나누어 시뮬레이션하였다. 중요 기능 동작은 다음과 같고 시뮬레이션 결과를 그림 9에 보였다. 인터럽트 소스를 받아들이는데 level trigger와 edge trigger의 차이를 두어 인식하였다. 우선순위가 같은 두 인터럽트 소스에 대해 동일한 시점 또는 순차적으로 인터럽트를 요청할 때 defalut 우선도를 기초로 하여 정확한 우선도 판정을 하여 CPU에 정확한 인터럽트 요청 신호 송출하였다. 인터럽트 인식신호 도달후 각 레지스터에 원하는 값이 설정되었다. 원하는 인터럽트가 인식되어 정확한 벡터값을 내보냈음을 알 수 있다. poll 레지스터를 읽었을 때 인터럽트 인식신호가 동일하게 역할을 수행하는지를 확인하였다. 높은 우선도를 가진 인터럽트 소스에 의한 인터럽트 서비스 루틴이 Nesting이 되었고, 마지막으로 NMI가 발생하였을 때 적절한 동작과 함께 DMA를 중지시키는 신호를 내보냈다.

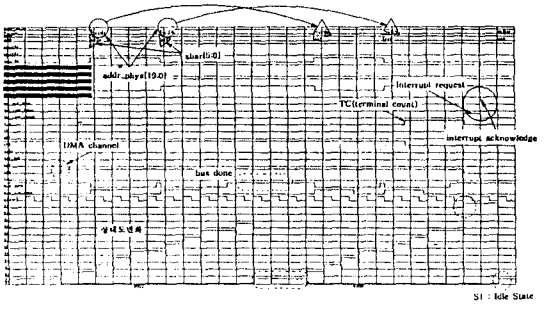


그림 8. 타이머 2 요청에 의한 DMA 전송과정
Fig. 8. DMA Transfer by the Timer 2 Request.

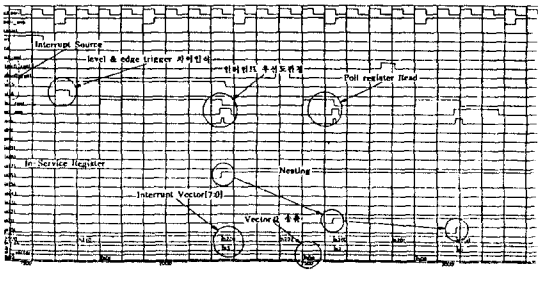


그림 9. 인터럽트 시뮬레이션 결과
Fig. 9. Simulation Result of Interrupt.

칩 선택 장치의 시뮬레이션은 rd와 wt의 원활한 동작과 각각의 선택 핀들에 대한 동작, pwrdn 모드의 동작에 대해 행하였다. 그림 10은 rd, wt의 시뮬레이션과 함께 각 핀의 선택을 수행한 것인데, 각각에 대하

여 wait_cnt의 시뮬레이션을 행하여 정확한 출력을 얻어내었다. 리셋 동작은 pwrdn이나 T1 상승시, 또는 Resint발생시에 모두 동작함을 알 수 있었다.

타이머 장치의 시뮬레이션은 타이머 0, 1, 2의 초기화 동작 및 각각의 모드에서의 정확한 동작을 실험하여 만족할 만한 결과를 얻어내었다. 또한 파워다운 모드에서 동작이 요구되는 경우도 시뮬레이션을 행하였다.

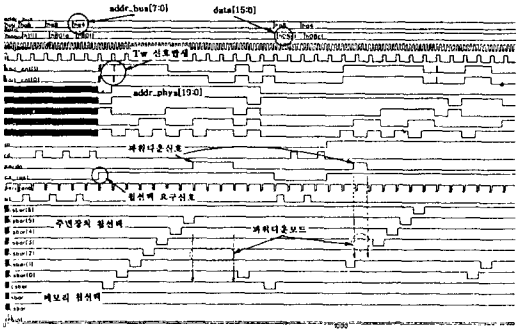


그림 10. 칩 선택 요구에 의한 주변장치 선택
Fig. 10. Chip Slection of Peripherals by the Chip Select Request.

Ⅳ. 결 론

본 논문에서는 이동통신 단말기용 마이크로콘트롤러에 내장되는 주변장치의 개발에 대하여 기술하였다. 이 주변장치는 COMPASS Design Tool을 사용하여 설계하고 검증하였으며, 설계된 Schematic으로부터 netlist를 얻고 VHDL 모델로 바꾼 다음 마이크로콘트롤러의 CPU core 부분과 결합하여 VHDL로시뮬레이션을 함으로써 그 기능이 올바로 수행됨을 확인하였다[9].

이동통신 단말기용 16 비트 마이크로콘트롤러에 내장할 목적으로 개발된 본 주변장치는 무선 전화용으로 개발된 인텔사의 80186EA에 내장되어있는 프로그램 가능한 주변장치들과 호환성을 갖고 있다. 각 주변장치들은 개별적인 모듈로 설계되어졌으므로 여러 용도의 ASIC 개발에 유용하게 사용될 것이다.

참 고 문 헌

[1] 유진영, "이동통신 단말기용 16 비트 CPU 개발", 전남대학교 석사학위청구논문, 1995

[2] 박성모, "이동통신 단말기 CPU 프로그램 개발에 관한연구", 공학논문집, 제36집, 전남대학교 공업기술연구소, pp. 115-126, 1994

[3] 유진영, 박성모, 남형진, "16 비트 CPU core 설계", 전자공학회 추계학술대회 논문집, 제17권, 제2호, pp. 1033-1036, 1994

[4] Microprocessor and Peripheral Handbook, Intel, 1987.

[5] John Uffenbeck, The 80806/8088 Family Design, Programming, Interfacing, Prentice-Hall, 1987.

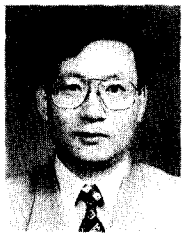
[6] David A.Patterson & John L.Hennessy, Computer Organization & Design:The Hardware/Software Interface, Morgan Kaufmann, 1994.

[7] Barry B. Brey, The Intel Microprocessors 8086/8088, 80186, 80286, 80386, and 80486 Architecture, Programming, and interfacing, Third Edition, Merrill, 1994.

[8] Yu-Cheng Liu and Glenn A. Gibson, Microcomputer System: The 8086/8088 Family, Architecure, Programming, and Design, Second Edition, Prentice-Hall, 1986.

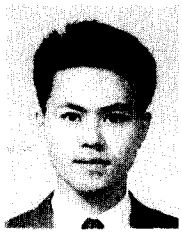
[9] 남형진, 장경희, 박경룡, 김재석, "CDMA 단말기 시스템의 VHDL 시뮬레이션", 대한전자공학회 추계학술대회 논문집, 제17권, 제2호, pp. 1350-1353, 1994

저 자 소 개



朴 性 模(正會員)

1977년 서울대학교 전자공학과 공학사. 1979년 한국과학기술원 전기 및 전자공학과 공학석사. 1988년 노스캐롤라이나 주립대학 전기 및 컴퓨터공학과 공학박사. 1979년~1984년 한국전자통신연구소 반도체단 설계개발부 연구원. 1988년~1992년 올드도미니언 대학교 전기 및 컴퓨터공학과 조교수. 1992년~현재 전남대학교 컴퓨터공학과 조교수. 현재 학과장 겸임. 1992년~1994년 전남대학교 전자계산소 연구개발부장 역임. 주관심분야는 마이크로프로세서, VLSI 시스템 설계, 신호처리용 ASIC 설계 등



金 炯 吉(學生會員)

1994년 2월 목포대학교 전자공학과 졸업(공학사). 현재 전남대학교 전자공학과 석사과정 재학중. 주관심분야는 마이크로프로세서, VLSI 시스템설계

李 南 佶(學生會員)

1992년 2월 전남대학교 전자공학과 졸업(공학사). 현재 전남대학교 전자공학과 석사과정 재학중. 주관심분야는 마이크로프로세서, VLSI 시스템설계

金 瑞 均(學生會員)

1992年 2月 전남대학교 전자공학과 졸업 (공학사). 1993年 3月~현재 (주)Pen & Brains Systems SI 사업부 실장. 현재 전남대학교 전자공학과 석사과정 재학중. 주관심분야는 VLSI 시스템설계, 멀티미디어 네트워크