자동착자 및 검사자동화 시스템을 위한 집적회로 설계

임 태 영† · 이 천 희††

요 약

본 논문은 TV 브라운관과 컴퓨터 모니터에 사용되는 마그네트(Magnet)에 일정한 자력을 갖도록 자화 시키는 착자기를 제어하며, 검사공정을 자동화하는 제어 시스템용의 집적회로를 설계하여 개발한 것에 관한 것으로써, 착자기의 콘트롤 모듈과 프로토콜 모듈의 주변기기 제어회로 부분을 0.8um CMOS SOG 기술로 설계하여 ETRI에서 공정하여 칩(Chip)을 완성시켜 동작을 확인하였다. 본 논문에서는 개별 셀(Single cell)의 지연 예측에 사용되었던 기존의 프로파게이션/램프 지연 모델(Propagation/ramp delay model)을 분석, 문제점을 보완 수정한 LODECAP(LOgic DEsign CAPture)의 인버터 선형 지연 모델을 응용하여 타이밍 콘트롤 블록 내의 지연 체인(Delay chain)을 설계 할 수 있는 새로운 "지연 예측 수식"을 제안하였다. 본 논문은 추출된 수식에 의거하여 타이밍 콘트롤 블록을 설계, 시스템에 적용하였으며, 나머지 블록들을 설계한 기법에 대하여도 상술하였다.

VLSI Design for Automatic Magnetizing and Inspection System

Tae-Young Lim† · Cheon-Hee Yi††

ABSTRACT

In this paper a VLSI design for the automatic magnetizing and inspection system has been presented. This is a design of a peripheral controller, which magnetizes CRTs and computer monitors and controls the automatic inspection system. We implemented a programmable peripheral interface(PPI) circuit of the control and protocol module for the magnetizer controller by using a 0.8um CMOS SOG technology of ETRI. Most of the PPI functions have been confirmed. In the conventional method, the propagation/ramp delay model was used to predict the delay of cells, but used to model on only a single cell. Later, a modified "linear delay pridict model" was suggested in the LODECAP(LOgic DEsign CAPture) by adding some factors to the prior model. But this has not a full model on the delay chain. In this paper a new "delay predict equation" for the design of the timing control block in PPI system has been suggested. We have described the detail method on a design of delay chain block according to the extracted equation and applied this method to the timing control block design. And we had descriptions on the other blocks of this system.

* 본 논문은 서울대 반도체공동연구소(97-2-2036)의 지원에 의한 연구의 일부분입니다.

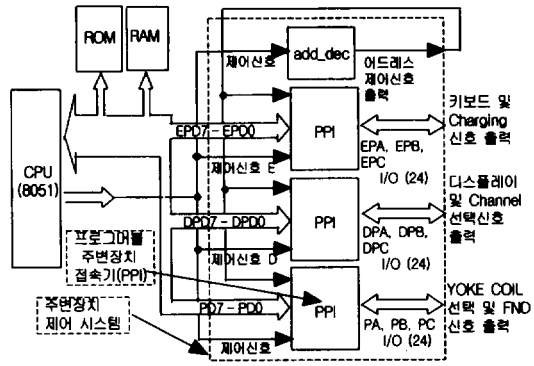
† 정 회 원 : 한국전자통신연구원 회로소자연구소
†† 종신회원 : 청주대학교 전자공학과 교수

논문접수 : 1999년 2월 9일, 심사완료 : 1999년 6월 23일

1. 서 론

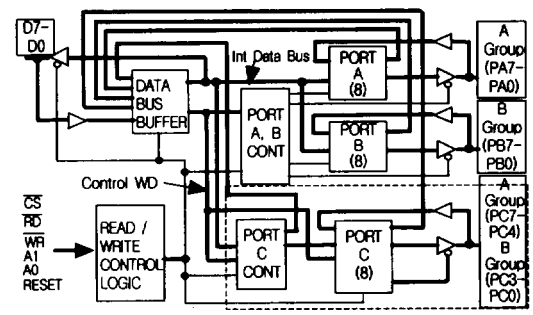
착자 자동화 제어 시스템은 착자작업을 시작하기 전에 운영자가 작업지도서에 의해 설정하는 작업설정모드로 착수되고, 지정 작업자 여부를 확인하는 작업시작모드 및 검사작업 시 가우스 미터로 마그네트의 값을 읽어서 기준치와 비교 판정하는 검사모드, 생성된 일련의 데이터를 분석하고, 출력하는 통계 및 출력모드를 구성해서 운영하는 소프트웨어 프로그램과 각종 감지 신호를 받아들이고, 각종 제어신호를 출력하는 주변장치 제어 시스템으로 구성된다[1].

또한 착자 자동화 제어 시스템은 (그림 1)에 나타낸 것처럼 CPU, RAM, ROM, PPI 및 키보드 입력부, LED 표시부, 자력 출력부, 자기 입력부 등의 주변장치로 구성 되어있다. 이러한 주변장치들을 제어하기 위해서는 72비트의 양방향 접속 포트가 필요하고, 접속된 주변장치에서 입력되는 신호들을 실시간으로 처리해야한다. 접속포트를 제어하는 일반적인 방법은 8비트 구조의 시스템 데이터를 가지고 8비트 구조의 접속장치들을 연결 시키는 포트들을 제어하게 되며, 이 때에 CPU가 포트들의 동작 상태를 수시로 점검해야 하기 때문에 시스템 소프트웨어의 추가부담이 발생하게 된다. 이것을 개선하는 방법으로써, 주변장치의 입력에 의하여 CPU를 직접 인터럽트(Interrupt) 시키는 방법이 사용되고 있다. 특히, 착자 자동화 제어 시스템처럼 12에서 24비트 구조의 주변장치들을 효과적으로 제어하기 위해서는 접속된 주변장치에서 입력되는 신호로써 CPU에 인터럽트를 요청할 수 있고, CPU의 인터럽트 응답을 받아들여서 측근의 주변장치를 직접 제어할 수 있으면서 주변장치의 접속비트들을 8~24 비트까지 가변 시킬 수 있는 제어기가 필요 하게된다. 본 논문은 주변장치 제어 시스템을 0.8um CMOS SOG 기술을 사용하여 ASIC 화 하였으며, 특히 이 ASIC에서 고속동작에 적합한 타이밍 제어 블록을 설계하기 위해서, 인버터에 대한 프로파게이션 지연 시간 산출 방법을 분석, 문제점을 보완 수정한 선형 지연 모델을 사용하여 클럭엣지 검출법으로 설계한 후 시스템에 적용 할 수 있는 기법을 상술하였다. 또한 접속된 주변장치에서 입력되는 신호로써 CPU에 인터럽트를 요청할 수 있는 포트 C 블록을 설계함으로써, 실시간으로 동작 할 수 있는 프로그래머블 주변장치 접속기(PPI)를 구현하였다[2].



(그림 1) 착자 자동화 제어 시스템의 구성도

PPI는 작동 방법을 프로그램 할 수 있는 인텔 계열의 범용 입출력 소자와 동일한 기능을 갖도록 구성하였다. 이 소자는 각각 12개로 구성된 두 그룹을 따로따로 프로그램 할 수 있는 24개의 입/출력 핀들이 있고, 세 가지의 중요한 동작 모드를 수행한다. 첫 번째 모드(모드 0)에서는 12개로 구성된 각 그룹을 4개씩으로 묶어서 입력이나 출력동작이 되도록 프로그램 할 수 있다. 두 번째 모드(모드 1)에서는 그룹당 8개를 입력이나 출력으로 프로그램 할 수 있고, 각 그룹에서 4개씩 남게되는 8개의 핀을 5개와 3개로 나누어서 핸드쉐이킹(Hand shaking)과 인터럽트 제어신호로 프로그램 할 수 있다. 세 번째 모드(모드 2)는 A 그룹 전용 모드로서, 그룹 A의 8비트를 양방향 버스로 사용하며, 나머지 4개와 B 그룹의 4개를 합쳐서 핸드쉐이킹과 인터럽트 제어신호로 프로그램 할 수 있다. (그림 2)는 PPI 1개의 블록도이며, 읽기/쓰기와 콘트롤 로직 블록, 포트 A/B 콘트롤 블록, 포트 C 콘트롤 블록, 데이터 버스 버퍼 블록, 포트 A와 B 블록, 포트 C 블록으로 구성하였고, (그림 3)에 각 포트들에 대한 동작모드를 종합 정리하였다.



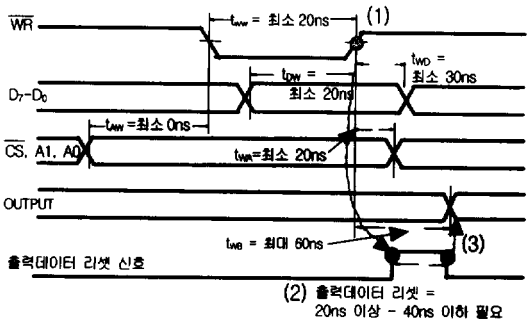
(그림 2) PPI의 블록도

	MODE 0			MODE 1			MODE 2	
		IN	OUT		IN	OUT	GROUP A ONLY	
A Group	PA_0	IN	OUT	PA_0	IN	OUT	PA_0	↔
A Group	PA_1	IN	OUT	PA_1	IN	OUT	PA_1	↔
A Group	PA_2	IN	OUT	PA_2	IN	OUT	PA_2	↔
A Group	PA_3	IN	OUT	PA_3	IN	OUT	PA_3	↔
A Group	PA_4	IN	OUT	PA_4	IN	OUT	PA_4	↔
A Group	PA_5	IN	OUT	PA_5	IN	OUT	PA_5	↔
A Group	PA_6	IN	OUT	PA_6	IN	OUT	PA_6	↔
A Group	PA_7	IN	OUT	PA_7	IN	OUT	PA_7	↔
B Group	PB_0	IN	OUT	PB_0	IN	OUT	PB_0	
B Group	PB_1	IN	OUT	PB_1	IN	OUT	PB_1	
B Group	PB_2	IN	OUT	PB_2	IN	OUT	PB_2	MODE 0
B Group	PB_3	IN	OUT	PB_3	IN	OUT	PB_3	OR
B Group	PB_4	IN	OUT	PB_4	IN	OUT	PB_4	MODE 1
B Group	PB_5	IN	OUT	PB_5	IN	OUT	PB_5	ONLY
B Group	PB_6	IN	OUT	PB_6	IN	OUT	PB_6	
B Group	PB_7	IN	OUT	PB_7	IN	OUT	PB_7	
B Group	PC_0	IN	OUT	PC_0	$\overline{INTR_B}$	$\overline{INTR_B}$	PC_0	I/O
B Group	PC_1	IN	OUT	PC_1	$\overline{IBF_B}$	$\overline{OBF_B}$	PC_1	I/O
B Group	PC_2	IN	OUT	PC_2	STB_B	ACK_B	PC_2	I/O
B Group	PC_3	IN	OUT	PC_3	$\overline{INTR_A}$	$INTR_A$	PC_3	$\overline{INTR_A}$
A Group	PC_4	IN	OUT	PC_4	STB_A	I/O	PC_4	STB_A
A Group	PC_5	IN	OUT	PC_5	IBF_{BA}	$\overline{I/O}$	PC_5	$\overline{IBF_{BA}}$
A Group	PC_6	IN	OUT	PC_6	I/O	$\overline{ACK_A}$	PC_6	$\overline{ACK_A}$
A Group	PC_7	IN	OUT	PC_7	I/O	$\overline{OBF_A}$	PC_7	$\overline{OBF_A}$

(그림 3) 동작모드의 종합 정리표

2. 엣지 검출 클럭발생기의 구성 및 설계

PPI는 CPU로부터 입력되는 읽기(RD)와 쓰기(WR) 신호의 라이징 엣지(Rising edge)에서 클럭킹(Clocking)이 되도록 설계 해야한다. 엣지 검출 클럭발생기(Edge detected clock generator)는 이 엣지 신호를 검출해서 (1)콘트롤 워드(Control word)를 레지스터에 저장하고, (2)이 후에 출력 포트들을 클리어(Clear) 시키고, (3)레지스터의 내용을 내부 데이터 버스라인을 통해 출력 할 수 있는 신호를 생성 해야한다. 이 블록은 (그림 4)와 같은 동작파형과 AC 특성이 요구된다.



(그림 4) 엣지 검출 클럭 발생기의 파형

그림에서 제일 먼저 CS, A1, A0가 인가되고, t_{AW}(써넣기 신호가 하강하기 전에 필요한 어드레스 안정 시간 : Address stable befor WR bar)를 최소 0ns 인가한 이후에, 다시 말하면 CS, A1, A0가 인가된 후 즉시, 최소한 100ns 이상의 폭을 가진 써넣기 신호(WR bar)가 하강되고, 이 써넣기 신호(WR bar)가 상승되기 이전에 최소한 100ns의 폭을 가진 t_{DW}(써넣기 신호가 상승하기 전에 필요한 데이터 셋업 시간 : Data setup time befor WR bar)를 포함한, 데이터 버스가 인가되어야 한다. 이 때부터 최소한 30ns가 지나간 후에 t_{WD} (써넣기 신호가 상승 한 이후에 필요한 데이터 홀드 시간 : Data hold time after WR bar) 시간을 포함한 데이터 버스 신호가 종료되어야한다. 써넣기 신호가 상승 한 이후부터 최소 20ns 이후에 CS, A1, A0가 종료되어야 하고, 최대 300ns 이전에 포트로 출력이 나갈 수 있어야 한다. 또한, 써넣기 신호가 올라갈 때에 인가되는 데이터 신호가 유효한 데이터이므로, 써넣기 신호가 올라갈 때(Rising edge)부터 어드레스 홀드 시간(t_{WA}) 사이에 콘트롤 워드 레지스터에 데이터를 써넣어야 하고, 이 후 먼저번 출력을 유지하고 있는 시간(t_{WB}) 사이에서 출력 데이터를 리셋 시키고, 콘트롤 워드 레지스터의 내용을 출력포트로 출력해야한다. 결국 읽기(RD)와 쓰기(WR) 신호의 라이징 엣지(Rising edge) 신호를 검출해서 20ns를 지연하고, 20ns 폭의 "출력 데이터 리셋 신호"를 엣지 검출 클럭발생기에서 생성 해야한다. 이 신호는 지연 체인을 구성 함으로써 설계가 가능하다.

설계 전단부(Front-end) 과정에서 20ns 정도의 지연 체인을 어떤 소자를 사용하여서 몇 개로 구성 할 것인가를 결정하는 일은 쉽지가 않다. 설계 전단부 과정용 시뮬레이터들은 배치배선에 대한 고려가 불충분한 지연 모델을 적용하므로 정확한 지연 예측이 곤란하다. 그러나 후단부(Back-end) 과정에서는 배치배선을 끝낸 상태이므로 포스트 시뮬레이션 이 후에 정확한 지연 값을 모델링 할 수 있다. 따라서 후단부 과정까지 진행된 다음에 정확한 지연 값을 알 수 있으므로, 원하는 값의 지연 체인을 구성하려면 그 값을 구할 때까지 전단부 과정과 후단부 과정을 여러 번 반복해야하는 불편함이 있다. 이러한 문제점을 해결하려면 설계 초기 단계에서 기초 셀에 대한 지연 값을 구 할 수 있어야한다.

게이트에 대한 지연 특성은 여러 가지 방법으로 모델링 할 수 있다. 입력파형의 기울기가 끼치는 영향 및 출력 파형과 프로파게이션 지연에 관한 분석적 표현 방법이 제안되었고[3], 입력에 인가되는 파형의 상승 및 하강시간과 출력에 연결된 부하 캐패시턴스를 변수로 하는 2차원 테이블 형태로 모델링 할 수 있다

[4]. 게이트 지연시간에 대한 배선회로의 영향은 유효 캐패시턴스 개념[5]을 통하여 표현 할 수 있고, 배선회로 지연시간에 대한 게이트의 영향은 게이트 구동 특성 모델[6]로 표현 할 수 있으며, 인버터를 구성하는 개개의 MOS에 대하여 시간에 따른 출력전압을 구함으로써 지연시간을 예측하는 방법도 제안되었다[7].

본 논문에서 구현하는 집적회로는 0.8um CMOS SOG 기술을 사용하여 ASIC 화하였고, SOG는 비아(Via), 콘택(Contact), 메탈(Metal) 레이어(Layer)를 제외한 나머지 레이어에 대한 웨이퍼 가공공정이 완료된 상태에서 회로설계를 시작한다. 따라서 여러 가지 지연 모델 기법에 관련된 기초 셀들의 각종 파라메타가 확정된 상태에서 착수되므로 공정회사에서 제공하는 기초 셀들의 라이브러리에 대한 분석이 필요하였다[8]. 본 논문에서는 라이브러리에 적용된, 인버터에 대한 프로파게이션 지연 시간 산출방법을 분석하여 이 지연 예측 모델을 보완 수정해서 "지연 예측 수식"을 제안한 것으로 지연 체인(Delay chain)을 설계한다.

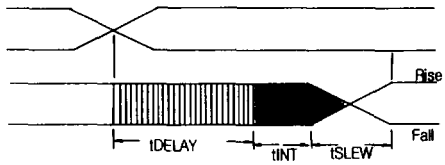
지연 체인용 논리소자를 선정할 때에 우선적으로 고려해야할 사항은, 구현하려는 디자인 룰을 확정하고(0.8um SOG), 전력소모가 적은 기초셀을 선택하여, 적은 수로도 지연 효과가 큰 기초셀을 선택해야한다. 본 논문에서는 가장 간단한 구조로 만들 수 있는 인버터를 선정하여 체인으로 연결하도록 하며, 지연 효과를 비교하기 위해 구동능력이 0.5X와 1X인 두 종류의 인버터를 선정하여 각각의 지연 시간(Delay time)을 검토하였다.

3. 지연 모델의 분석 및 지연 예측치 산출 수식

3.1 프로파게이션/램프 지연 모델의 분석

입력의 변화를 받아 출력이 변하는데 걸리데 까지 지연되는 시간을 프로파게이션 지연(Propagation delay)이라 하고, 입력의 스위칭 스레스홀드 레벨(Switching threshold level)에서 출력의 스위칭 레벨까지의 시간을 측정하는 방법이다. 0.8um SOG에서는 프로파게이션/램프 지연 모델을 적용하고 있으며, ISM(Input Slope Modeling) 기법으로 50%의 엣지 레이트(Edge rate)를 모델링하고 있다[8].

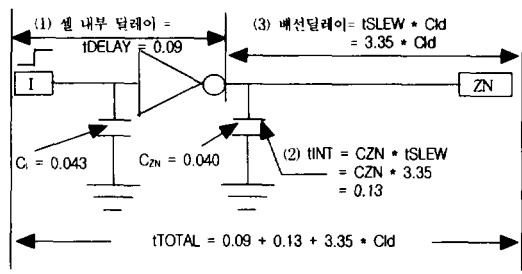
앞 뒤 연결이 없는 기초 셀 하나의 입력 핀에서부터 출력 핀까지의 총 지연(tTOTAL)는 (그림 5)와 같고 식(1)로 표현한다. 이 값은 신호가 0에서 1로 올라갈 때와 1에서 0으로 내려갈 때의 두 가지 값이 있다.



(그림 5) 기초셀의 총 딜레이(tTOTAL)

tTOTAL = tDELAY + tINT + tSLEW * Cld -> (1)

식(1)을 적용한 0.8um CMOS SOG 인버터 셀 라이브러리의 예는 데이터 북에 잘 나타나 있다[2]. IN01D0와 IN01D1는 각각 0.5X와 1X의 구동능력을 갖는 인버터 기초 셀을 의미한다. I는 입력핀의 캐패시턴스 이고, ZN은 출력 핀의 캐패시턴스로써, 각각 0.043과 0.087pF이며, 0.040과 0.043pF으로써, 이 값들이 지연를 산출하는데 중요한 수치가 된다. 또한 IN01D0의 상승에 대한 수행 방정식(Performance equation)을 보면, 0.09ns는 소자의 내부 지연(tDELAY)이고, 0.13ns는 출력 캐패시턴스 자체에 기인하는 지연(tINT)이며, 3.35ns/pF는 출력파형의 기울기(tSLEW)로써 식(1)의 형식으로 나타낸 것을 볼 수 있다. 데이터북에 있는 지연의 성분은[8] (그림 6)과 같이 모델링하여 나타낼 수 있다.



(그림 6) 딜레이 성분의 모델링

그림은 상승(Rise)입력의 예이며, 총 지연(tTOTAL)는 (1)셀 내부 지연(tDELAY)과 (2)출력 캐패시턴스 자체에 기인하는 지연(tINT) 및 (3)배선 지연[출력파형의 기울기(tSLEW)에다 각 출력의 부하 캐패시턴스(Cld)를 곱한 값]으로 구성되어 있고, 식(1)의 형식을 나타낸다. 이 때 입력 신호가 Fall 일 때의 총 지연은 tTOTAL = 0.07 + 0.06 + 1.43 * Cld로 나타낸다. 구동능력이 0.5X와 1X인 인버터에서 표준부하(Standard load)를 0.43pF로 가정하였을 때의 프로파게이션/램프 지연값은 상승과 하강에서 각각 0.51과 0.25 이므로 인버터 한쌍의

tTOTAL은 0.76ns가 된다. 이상을 토대로, 인버터 체인 지연의 연결개수에 따른 0.5와 1X에서의 프로파게이션/램프 지연값을 (그림 8)과 (그림 9)에 나타내었다.

이와 같은 프로파게이션/램프 지연은 게이트간의 연결(Connection)에 의해 지연되는 지연과 C_w(Wire Cap.)에 대한 고려가 없다. 또한 공정회사에서 제공하는 라이브러리(Library)에는 식(1)의 제일 끝에 있는, 각 출력의 부하 캐패시턴스(Cld)의 수치가 없다. 이는 공정회사의 칩 제작 공정, 게이트 어레이 원판의 종류, 배선의 길이, 팬아우트(Fanout)의 개수에 의하여 달라지기 때문에 어떤 값을 정 할 수 없고, 표준부하(Standard load)를 추정 값으로 하여 산출하게 된다. 따라서 공정회사에서 제공하는 라이브러리만 가지고는 인버터셀의 지연을 정확하게 산출 할 수 없다는 문제점이 있다[8].

3.2 공정변수와 배선 및 단자 캡(Cap)을 고려한 선형 지연 예측 모델의 분석

LODECAP(LOgic DEsign CAPture)은 모든 설계정보를 VHDL 언어로 통일하여 사용하므로 기초셀도 VHDL 선형 지연 예측 모델로 되어있다[9]. LODECAP에서는 이 예측모델을 프리시뮬레이션(Presimulation) 단계에서 적용이 가능한 특징이 있다. 본 논문에서는 이를 분석하여 "지연 예측 수식"을 제안함으로써 엣지 검출 클럭발생기에 적용하였다.

선형 지연 모델은 0.65um 이상의 공정기술에서 보편적으로 사용되는 모델이다. 이 모델에서 기초 셀의 총 지연는 게이트 회로 자체에 의해 결정되는 소자의 내부 지연(D_{int}, 데이터 북에서는 tDELAY로 표기하였음)와 게이트간의 단자 및 배선 캡 값에 의해 결정되는 지연(D_{inc})의 합(식(3))에 0.0~1.0 사이의 실수 값을 갖는 부하경감요소(Derating factor)라는 변수를 곱한 결과로 결정된다(식(2)). 부하경감요소는 칩 동작 온도, 전원 전압, 칩 제작 공정의 요인들을 모델링한 변수로서 값이 작을수록 지연이 적어져 회로의 동작이 빨라지게 된다. 이 값은 전형적인 경우(Typical case) 0.5로 설정한다.

$$\text{기초셀의 총 지연} = \text{Delay} * \text{Derating\_Factor} \rightarrow (2)$$

$$\text{Delay} = D_{int} + D_{inc} = D_{int} + [C_{linear} * (\text{총단자 Cap.} + \text{총배선 Cap.})] \quad (3)$$

$$\text{총단자 Cap.} = \sum (\text{개별단자 Cap.}) \quad (4)$$

$$\text{총배선 Cap.} = C_{estimation} * \text{단자 수} + Cap0 \quad (5)$$

소자의 내부 지연(D_{int})은 셀의 레이아웃을 하면서 결정되는 고정된 지연 값인 반면에 게이트간의 연결에 의해 결정되는 지연(D_{inc})은 셀의 구동능력과 기생 캐패시턴스에 의해 결정되는 지연 값이다. 여기서 기생 캐패시턴스는 회로도의 한 네트에 연결된 모든 게이트 단자들의 캐패시턴스(총단자 Cap.)를 합한 값과 배선 자체가 갖는 캐패시턴스(총배선 Cap.)를 합한 값이다. 지연은 식(3)과 같이 기생 캐패시턴스에 관한 선형 함수로 계산되는데, 선형함수를 결정하는 기울기(C_{linear}, 데이터 북 에서는 tSLEW로 표기하였음)와 절편(D_{int}) 값은 셀마다 고유한 값으로써, 셀을 개발하는 과정에서 결정되며 기초 셀의 VHDL 프로그램안에 정의되어 있다.

총배선 캡(식(5))의 경우 네트의 연결을 구현한 폴리와 메탈의 길이와 모양, 그리고 연결점(Connection)의 개수에 의해 계산되어야 한다. 그러나 레이아웃이 아직 만들어지지 않은 논리설계 단계에서는 레이아웃에 대한 정보를 갖고 있지 않다. 이러한 문제점은 공정변수와 단자 및 배선 캡(Cap.) 값을 고려한 VHDL 선형 지연 예측 모델로써 해결 할 수 있다.

LODECAP[9]은 기초 셀의 배선 캡을 네트에 연결된 단자에 선형적으로 비례하는 배선 캡 예측 모델을 사용하며, 이 선형 모델의 기울기($C_{estimation}$)와 절편(Cap0) 값은 게이트 어레이의 원판 종류에 따라 다른 값을 사용한다.

게이트 어레이 원판 모델은 설계 흐름관리기에서 "pred***"라는 광역변수 이름(예 : pred186)으로 명명되며, LODECAP이 설치된 파일시스템의 각 라이브러리 디렉토리에 "checkRampDelay.Mcap" 이라는 이름의 파일에 정의되어 있다. 이 지연 모델은 각 기초 셀의 VHDL 모델에 정의되어 있어 VHDL 시뮬레이션에 사용되며 그밖에 램프 지연(Ramp delay) 검사나 타이밍 검증(Timing verification)에도 동일하게 사용되고 있다. 지연 체인(Delay chain)을 정확하게 설계하기 위하여 주요 구성 소자인 인버터 셀을 살펴보면, IN01D0와 IN01D1의 절편 (D_{int}) 값과 기울기(C_{linear}) 및 캐패시턴스 값을 <표 1>과 같이 정리할 수 있다.

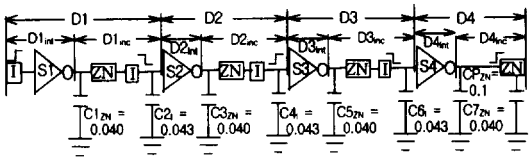
이 값은 기초 셀 단독으로 존재할 때의 값이므로 본 논문에서 구현하려는 지연 체인에는 그대로 적용하지 못하고, 게이트간의 연결에 의해 결정되는 지연에 대한 고려를 추가해야한다. 이를 위해서는 광역변수에 따른 선형모델의 기울기($C_{estimation}$)와 절편(Cap0) 값을

알아야한다.

〈표 1〉 절편(D_{int}), 기울기(C_{linear}) 및 캐패시턴스값

	Rise		Fall		캐패시턴스 값	
	절편 (D_{int})값	기울기 (C_{linear})	절편 (D_{int})값	기울기 (C_{linear})	입력 (C_I)	출력 (C_{ZN})
IN01D0	0.09ns	3.35ns/pF	0.07ns	1.43ns/pF	0.043pF	0.040pF
IN01D1	0.11ns	1.68ns/pF	0.08ns	0.71ns/pF	0.087pF	0.043pF

이 값들은 LODECAP의 "checkRampDelay.Mcap" 파일에 정의된 게이트 어레이 원판 모델중에서 pvgc450 (0.8um SOG)의 vgc400186 기판(Base array)의 값으로 구할 수 있고, 각각 0.096pF과 0.029pF이다. 이 값과 앞서 구한 절편(D_{int}) 값과 기울기(C_{linear}) 및 입력과 출력측의 캐패시턴스 값을 (그림 7)과 같은 인버터 체인 모델을 이용하여 식(6)와 식(7)을 적용하면 인버터 체인 모델의 총 지연 값을 구할 수 있다.



(그림 7) 인버터 체인 모델과 캐패시턴스 값

인버터 체인 모델의 총 지연

$$= \sum Dx * Derating\_Factor \quad (6)$$

$$Dx = Dx_{int} + Dx_{inc}$$

$$= Dx_{int} + \{C_{linear} * [\sum (\text{개별단자 Cap.}) + (C_{estimation} * \text{단자 수} + Cap0)]\} \quad (7)$$

$D1 = D1_{int} + \{Rise의\ C_{linear} * [(C1_{ZN} + C2_I) + (vgc400186의\ C_{estimation} * 단자\ 수 + Cap0)]\}$

$= 0.09ns + \{3.35ns/pF * [(0.04pF + 0.043pF) + (0.096pF * 2 + 0.029pF)]\} = 1.1084\ ns$

$D2 = D2_{int} + \{Fall의\ C_{linear} * [(C3_{ZN} + C4_I) + (vgc400186의\ C_{estimation} * 단자\ 수 + Cap0)]\}$

$= 0.07ns + \{1.43ns/pF * [(0.04pF + 0.043pF) + (0.096pF * 2 + 0.029pF)]\} = 0.5047\ ns$

D3 = D1과 동일함.

$D4 = D4_{int} + \{Fall의\ C_{linear} * [(C7_{ZN} + CP_{ZN}) + (vgc400186의\ C_{estimation} * 단자\ 수 + Cap0)]\}$

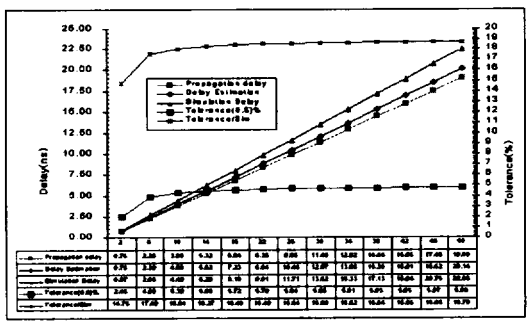
$= 0.07ns + \{1.43ns/pF * [(0.04pF + 0.1pF) + (0.096pF * 1 + 0.029pF)]\} = 0.44895\ ns$

인버터 체인 모델의 총 지연

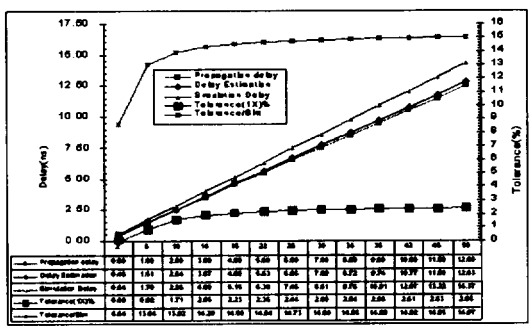
$= \sum Dx * Derating\_Factor$

$= (D1 + D2 + D3 + D4) * 0.5 = 1.585225\ ns$

이상을 토대로, 구동능력이 0.5X와 1X인 인버터 체인 지연의 연결개수에 따른 공정변수와 단자 및 배선 캡 값을 고려한 VHDL 선형 지연 예측 모델의 지연값을 (그림 8)과 (그림 9)에 나타내었고, 이를 포스트 시뮬레이션(Post simulation) 이후 의 지연 값과 비교하였다. 그림에서 프로파게이션/램프 지연 모델과 공정변수와 단자 및 배선 캡 값을 고려한 VHDL 선형 지연 모델은 포스트 시뮬레이션 이후 의 지연 값에 비해 각각 15%와 5%의 오차율이 있는 것을 알 수 있고, 1X와 비교해서 0.5X가 지연 효과가 큰 것을 알 수 있다.



(그림 8) 0.5X 인버터 체인의 딜레이



(그림 9) 1X 인버터 체인의 딜레이

3.3 지연 예측치 산출 수식의 제안

공정변수와 단자 및 배선 캡 값을 고려한 VHDL 선형 지연 모델을 식(1)과 대비하면 식(8)로 나타낼 수 있다.

$$tTOTAL_{(Rise)} = \{tDELAY + tINT + tSLEW * [\text{ load Cap.} + (C_{estimation} * \text{Relative fanout} + Cap0)]\} * Derating\_Factor \quad (8)$$

위 식에서 디자인 룰이 확정되고(0.8u SOG), 기판(Base array)이 확정된(vgc400186) 상태이면 모든 값을 앞 절에서처럼 대입 할 수 있고, 0.5X 인버터 셀의 $tTOTAL_{(Rise)}$과 $tTOTAL_{(fall)}$은 각각 0.5542와 0.25235가 된다.

따라서 0.5X 인버터 쌍의 지연 값은 $tTOTAL_{(Rise)}$과 $tTOTAL_{(fall)}$을 합산한 값이며, 필요한 지연 예측치를 얻으려면 식(9), 식(10, 11)로 산출이 가능하다.

$$\text{지연 예측치}(D_{(X)}) = (tTOTAL_{(Rise)} + tTOTAL_{(fall)}) * \text{쌍 수}(N) \quad (9)$$

$$0.5X \text{ 인버터 쌍의 } D_{(X)} = 0.80655 * \text{쌍 수}(N) \quad (10)$$

$$1X \text{ 인버터 쌍의 } D_{(X)} = 0.514 * \text{쌍 수}(N) \quad (11)$$

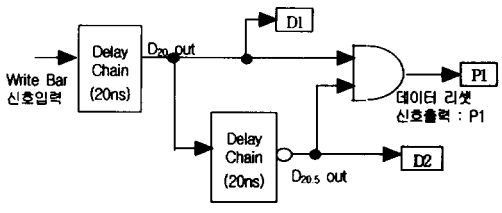
지연 체인을 설계할 때, 지연 값을 산출하지 못하는 프리씸 단계에서 이와 같은 지연 예측치 산출 수식을 적용하면 정확한 지연 체인 블록의 구현이 가능하다.

3.4 지연 체인(Delay chain) 블록의 구성 및 설계

지연 체인블록은 (1) 지연 체인 블록의 모델을 그림 10과 같이 설정하고, (2) (그림 4)에 있는 "출력 데이터 리셀 신호"의 타이밍 도를 만들어서, (3) 개별 소자가 갖는 지연 값을 산출하여 필요한 소자만큼 연쇄 연결하고, (4) S&S(Schematic and simulation) 방식으로 세부회로를 구현한다. 구체적인 방법은 다음과 같다[10].

3.4.1 지연 체인 블록의 모델을 설정

지연 체인 블록의 모델을 (그림 10)과 같이 설정한다.



(그림 10) 딜레이 체인 블록의 모델

3.4.2 지연 체인 블록의 타이밍도 설정

지연 체인 블록의 타이밍 도는 (그림 4)에 있는 "WR bar"를 입력해서 "출력 데이터 리셋 신호"를 얻어낸다.

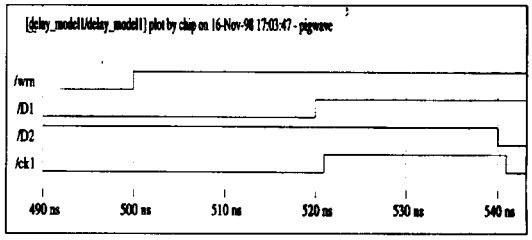
3.4.3 개별 소자가 갖는 지연 값을 산출

(그림 8)의 0.5X 인버터 체인 지연를 선정하여 지연 예측치 산출 수식을 적용하였다.

식(10)에서 원하는 지연시간[$D_{(X)}$]을 20ns라 하면 필요한 쌍 수(N)는 20 / 0.80655 = 24.797 쌍이 되므로 20ns를 얻기 위해서는 0.5X 인버터 24.5쌍을 삽입하여 D1과 D4를 포함, 인버터 50개와 51개를 연쇄 연결한다.

3.4.4 S&S 방식의 세부회로 구현

S&S(Schematic & simulation) 방식을 지연 체인 블록 모델에 적용하여 설계하였으며, 시뮬레이션 결과는 (그림 11)과 같다. 그림에서 WRN과 D2가 1ns 오차를 보이는 것은 AND 소자 1개가 추가된 결과이며, 식(10)과 시뮬레이션 결과는 일치한다.



(그림 11) 시뮬레이션 결과 파형

4. 읽기/쓰기와 콘트롤 로직 블록의 구성 및 설계

읽기/쓰기와 콘트롤 로직 블록은 타이밍 콘트롤 블록이라고도 한다. 이 블록의 기능은 데이터와 콘트롤 또는 스테터스 워드의 모든 내부와 외부 전송신호들을 이네이블 시키거나 클럭킹 시키는 신호들을 만들어낸다. 따라서 이 블록에는 입력 신호 WR, RD, A0, A1, RESET, CS를 수용하고, 이에 알맞은 출력을 낼 수 있는 읽기/쓰기 신호제어기와 앞절의 지연 체인 블록이 포함된 엣지 검출 클럭발생기를 설계하여 내부 레지스터들을 제어 할 수 있도록 하여야 한다.

PPI에 입력되는 어드레스 신호는 A0와 A1 두 개로 구성되며, 최대 4 신호까지 디코딩 할 수 있다. 따라서 2 to 4 디코더를 사용하면 입력되는 어드레스 전부를 변별할 수 있게 된다. 또한 CS 입력이 WR과 RD 중 어느 하나라도 "Low" 상태 일 때에 "Low"로 결합되면 PPI를 선택하여 정상적으로 모든 동작 모드를 수행

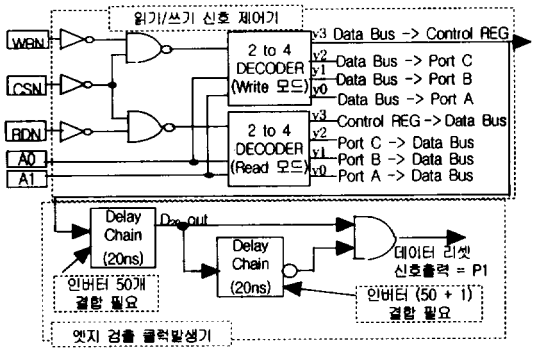
하고, "High" 일 때는 PPI를 디스에이블 시킨다. 따라서 CS는 두 개의 AND, 또는 NAND 게이트로 WR과 RD를 각각 결합하여 인가하면 된다.

또한 RESET은 이 입력이 "High"이면 콘트롤 레지스터를 클리어 시키고, 모든 포트들을 입력 모드로 만든다. 따라서 "Low" 일 때는 아무런 영향을 끼치지 않고, "High"일 때만 콘트롤 레지스터를 클리어 시키도록 AND 또는 NAND 게이트를 구성한다.

이상을 토대로 읽기/쓰기 신호제어기는 <표 2>와 같은 기능 분석표를 만들 수 있고, 지연 체인 블록으로 구성한 엣지 검출 클럭발생기를 포함하여 S&S 방법으로 (그림 12) 처럼 설계하였다.

〈표 2〉 읽기/쓰기 신호 제어기의 기능 분석표

입력신호					출력신호			
					Read 모드		Write 모드	
A1	A0	RD bar	WR bar	CS bar	신호	기능	신호	기능
0	0	0	1	0	pard	PA -> D_Bus		
0	1	0	1	0	pbrd	PB -> D_Bus		
1	0	0	1	0	pcrd	PC -> D_Bus		
1	1	0	1	0	cwdr enb	CWD ->DBus		
0	0	1	0	0			cp1	D_Bus -> PA
0	1	1	0	0			cp2	D_Bus -> PB
1	0	1	0	0			ck1	D_Bus -> PC
1	1	1	0	0			cwd clk	DBus -> CWD

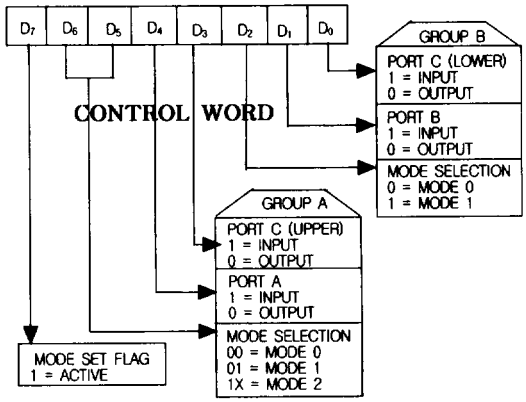


(그림 12) 읽기/쓰기와 콘트롤 로직 블록도

5. 포트 A/B 콘트롤 블록의 구성 및 설계

이 블록은 포트 A의 8비트와 포트 B의 8비트를 전담 제어하도록 구성하였다. 이 블록을 설계하기 위해서는 콘트롤 워드에 대한 분석이 필요하다.

PPI의 콘트롤 워드는 (그림 13)과 같은 형식으로 되어있고, D7이 "1"이면 "모드 셑 액티브"이며, "0"이면 "비트 셑"임을 나타낸다.



(그림 13) 콘트롤 워드의 형식

따라서, 포트 A/B에 대한 모드셑은 D7을 "1"로 고정시키고 나머지 비트들에 대해서만 고려하여 설계 하면된다.

(그림 13)의 콘트롤 워드를 분석하면, 포트 A와 포트 B의 입출력을 제어하는 비트는 D1과 D4 비트이며, 포트 C의 입출력 제어는 D0과 D3 비트가 담당한다. 또한 D2와 D5 및 D6 비트로 포트 B와 포트 A의 모드를 설정한다.

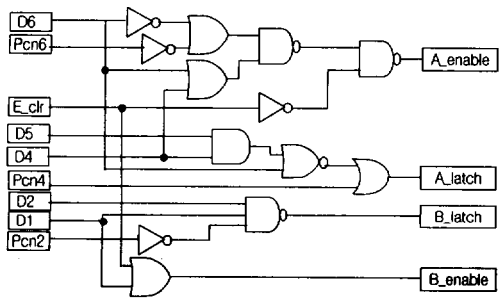
따라서, 포트 A와 포트 B는 D0, D3, D7을 제외한 나머지 비트들로 포트 A/B 콘트롤 블록을 제어하는 것이다. 이 블록에 필요한 신호들은 다음과 같다.

콘트롤 워드의 D1 비트는 포트 B를 입력 또는 출력 상태로 만들며, D2 비트는 포트 B를 모드 0 또는 모드 1이 되도록 한다. 이 블록에서 필요한 출력은 B_enable 이다. 콘트롤 워드의 D4 비트는 포트 A를 입력 또는 출력 상태로 만들며, D5 비트는 포트 A를 모드 0 또는 모드 1이 되도록 한다. 이 블록에서 필요한 출력은 A_enable 이다. 또한 D6 비트가 "1"이 되면 모드 2로 되며, 포드 A를 양방향으로 제어할 수 있도록 한다. 이 때 모드 1 에서는 포트 C의 PC2 비트 입력과 콘트롤 워드 D1, D2 비트를 조합하여 포트 B로 입력되는 신호를 "입력신호 레지스터"에 랫치 할 수 있는 출력 신호가 필요하며, 포트 C의 PC4 비트 입력과 콘트롤 워드 D4, D5 비트를 조합하여 포트 A로 입력되는 신호를 "입력신호 레지스터"에 랫치 할 수 있는 출력 신

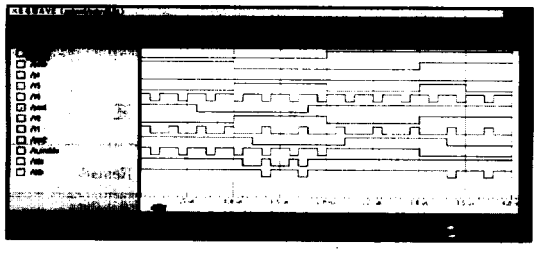
호가 필요하다. 여기에 필요한 출력은 각각 A_latch, B_latch 이다. 이 블록은 <표 3>과 같은 진리표를 만들어서, "Truth table edit"로 (그림 14)와 같이 설계하였고, 시뮬레이션 결과는 (그림 15)와 같다.

〈표 3〉 포트 A/B 콘트롤 블록의 진리치표

입력									출력 "0" Active				비고	
clr	콘트롤 워드					포트 C								
EC	D6	D5	D4	D2	D1	Pc6	Pc4	Pc2	A_enb	B_enb	A_latch	B_latch	모드	기능
0	X	X	X	0	0	X	X	X	1	0	1	1	0	PB Out
X	X	X	X	0	1	X	X	X	1	1	1	1	0	PB In
0	0	0	0	X	X	X	X	X	0	1	1	1	0	PA Out
X	0	0	1	X	X	X	X	X	1	1	1	1	0	PA In
X	X	X	X	1	0	X	X	X	1	0	1	1	1	PB Out
X	X	X	X	1	1	X	X	0	1	1	1	0	1	PB In
X	0	1	0	X	X	X	X	X	0	1	1	1	1	PA Out
X	0	1	1	X	X	X	0	X	1	1	0	1	1	PA In
X	1	X	X	X	X	X	0	X	1	1	0	1	2	PAI/O
0	1	X	X	X	X	1	X	X	0	1	1	1	2	PA Out
0	X	X	X	X	0	1	X	X	1	0	1	1	2	PB Out



(그림 14) 포트 A/B 콘트롤 블록도



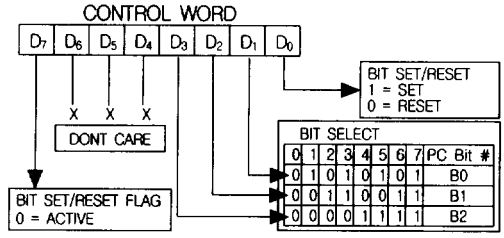
(그림 15) 포트 A/B 콘트롤 블록의 Sim 파형

6. 포트 C 콘트롤 블록의 기능

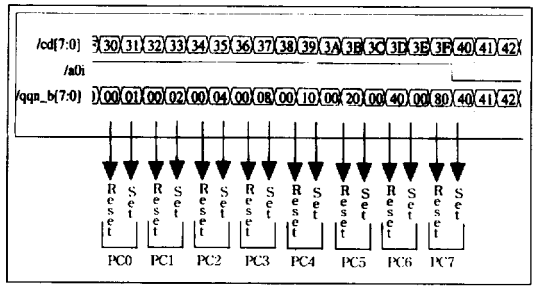
포트 C 콘트롤 블록은 포트 C의 상위 4비트와 하위 4비트를 전담 제어하면서, 콘트롤 워드로써 데이터 버스 레지스터 및 비트 셀 레지스터 기능이 호환 동작하도록 설계하여 "모드 셀"과 "비트 셀" 기능을 구비하였다.

따라서, 이 블록은 포트 C에 한정하여 모드 0에서는 포트 A/B 콘트롤 블록과 동일하게 데이터 버스 레지스터를 작동시키며, 모드 1과 모드 2에서는 비트 셀 레지스터로 작동 시켜, 포트 C의 개별 비트들을 제어할 수 있는 신호들을 생성하였다.

PPI의 콘트롤 워드를 분석하면, 비트 셀/리셀 형식은 (그림 16)과 같은 형식으로 되어있고, D7이 "0"이면 "비트 셀" 임을 나타낸다. (그림 17)은 이의 시뮬레이션 결과이며, CD(7 : 0)의 하위 4비트에 의해 포트 C의 각 비트를 셀 시키거나 리셀(00) 시킴을 알 수 있다.



(그림 16) 비트 셀/리셀 형식



(그림 17) 비트 셀/리셀 레지스터 블록의 Sim 파형

7. 데이터 버스 버퍼 및 포트 A/B 블록의 설계

7.1 데이터 버스 버퍼 블록의 구성 및 설계

이 블록은 외부에서 입력되는 데이터 버스를 받아들여 콘트롤 워드로 만들거나 버퍼링 한 후에 내부의 시스템용 데이터 버스, 또는 비트 셀/리셀 버스로 만드는 기능을 갖도록 설계한다. 또한 포트 A, B, C의 입력들을 버퍼링하여 내부의 버스 라인에 인가하는 기능을 갖도록 설계한다. 따라서 포트 A, B, C의 입력 버스와 데이터 버스 및 비트 셀/리셀 역할을 하는 콘트롤 워드 버스

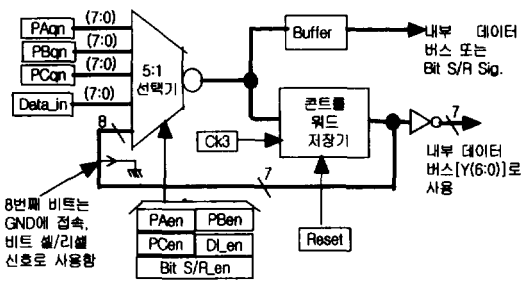
들 중의 한 버스를 선택하는 5 : 1 버스 신호 선택기와 데이터 버스가 콘트롤 워드로 입력되었을 때, 이 콘트롤 워드를 저장하는 콘트롤 워드 저장기를 설계하였다.

7.2 5 : 1 버스 신호 선택기의 세부 구성

이 선택기에 입력되는 버스는 전부 5종으로써 8비트의 버스로 인가된다. 이 중에 한 버스만 이네이블 시켜서 내부의 버스 라인으로 출력한다. 여기에 사용하는 이네이블 신호들은 읽기/쓰기와 콘트롤 로직 블록에서 생성한 것으로써, paen, pben, pcen, data_inen, cont-wden을 사용하고, 선택된 버스를 버퍼링한 후 내부 데이터 버스로 출력하며, 콘트롤 워드 저장기에 입력한다.

7.3 콘트롤 워드 저장기의 세부 구성 및 설계

이 저장기에 입력되는 버스는 콘트롤 워드용 클럭 신호가 라이싱 될 때 저장되며, 이 의 출력은 시스템 내부용 콘트롤 워드로 출력하고, 5 : 1 버스 신호 선택기에 휘드백 시켜서 비트 셀/리셀 버스 역할을 할 수 있도록 구성 및 설계를 하였다. (그림 18)에 데이터 버스 버퍼 블록의 구성도를 나타냈다.



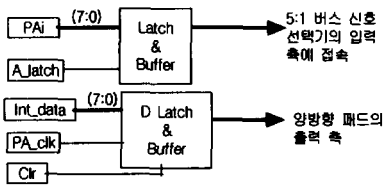
(그림 18) 데이터 버스 버퍼 블록 구성도

그림에서 콘트롤 워드 저장기의 출력이 5 : 1 버스 신호 선택기에 휘드백 할 때, 8번째 비트를 GND에 접속함으로써 비트 셀/리셀 신호로써 내부 데이터 버스에 선택될 수 있도록 설계하였다.

7.4 포트 A와 B 블록의 구성 및 설계

이 블록은 외부에서 인가하는 8-비트의 데이터 입력을 랫치한 후 데이터 버스 버퍼 블록으로 보내고, 내부의 시스템 데이터 버스를 받아들여 외부로 출력하는 기능을 한다.

이 블록에서는 입력과 출력을 별개로 구성하며, Top 블록에서 양방향 패드를 적용하여 함께 묶어서 8 핀으로 구성하였다. (그림 19)에 포트 A와 B 블록의 구성도를 나타냈다.



(그림 19) 포트 A/B 블록 구성도

8. 포트 C 및 어드레스 디코더/패드 블록의 설계

8.1 포트 C 블록의 구성 및 설계

이 블록은 외부에서 인가하는 8-비트의 입력을 버퍼링하여 5 : 1 버스 신호 선택기의 입력측에 접속하고, 포트 C 콘트롤 블록에서 인가되는 신호를 받아들여 버퍼링 한 후 양방향 포트 C 패드의 출력측에 내 보내며, 콘트롤 워드 레지스터의 내용을 읽어들여서 포트 C가 출력 모드로 될 때를 선별, 이네이블 신호를 생성시켜서 양방향 포트 C 패드의 이네이블 단자에 출력하는 기능을 갖도록 구성하였다.

8.2 이네이블 신호 생성기의 세부 설계

콘트롤 워드는 30개가 있으며, 콘트롤 워드에 의하여 모드 설정이 되고, 포트들의 입출력을 정의한다. 각 콘트롤 워드에 따른 포트 C의 출력 이네이블 논리와 모드 설정을 <표 4>에 나타내었다. 표의 출력 이네이블에서 논리 "0"은 액티브 임을 의미한다.

〈표 4〉 콘트롤 워드의 진리치 표

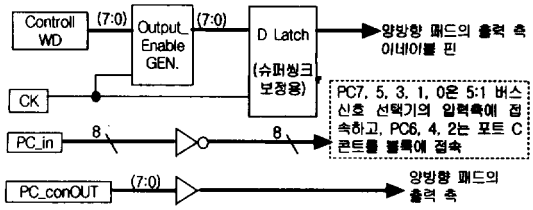
Controll word(Y)							Out bit define for PC							Remark
6	5	4	3	2	1	0	C7	6	5	4	3	2	1	C0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	M0, PC7-0 = Out
0	0	0	0	0	0	1	0	0	0	0	1	1	1	1	M0, PC7-4 = O, PC3-0 = I
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	M0, PC7-0 = Out
0	0	0	0	0	1	1	0	0	0	0	1	1	1	1	M0, PC7-4 = O, PC3-0 = I
0	0	0	1	0	0	0	1	1	1	1	0	0	0	0	M0, PC7-4 = I, PC3-0 = O
0	0	0	1	0	0	1	1	1	1	1	1	1	1	1	M0, PC7-0 = In
0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	M0, PC7-4 = I, PC3-0 = O
0	0	0	1	0	1	1	1	1	1	1	1	1	1	1	M0, PC7-0 = In
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	M0, PC7-0 = Out
0	0	1	0	0	0	1	0	0	0	0	1	1	1	1	M0, PC7-4 = O, PC3-0 = I
0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	M0, PC7-0 = Out
0	0	1	0	0	1	1	0	0	0	0	1	1	1	1	M0, PC7-4 = O, PC3-0 = I
0	0	1	1	0	0	0	1	1	1	1	0	0	0	0	M0, PC7-4 = I, PC3-0 = O

〈표 4〉 계속

Controll word(Y)							Out bit define for PC							Remark
6	5	4	3	2	1	0	C7	6	5	4	3	2	1	C0	
0	0	1	1	0	0	1	1	1	1	1	1	1	1	1	M0. PC7-0 = In
0	0	1	1	0	1	0	1	1	1	1	0	0	0	0	M0. PC7-4 = I. PC3-0 = O
0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	M0. PC7-0 = In
0	1	0	0	1	0	X	0	1	0	0	0	1	0	0	M1. PB = Out
0	1	0	0	1	1	X	0	1	0	0	0	1	0	0	M1. PA = Out. PB = In
0	1	0	1	1	0	X	0	1	1	1	0	1	0	0	M1. PB = Out
0	1	0	1	1	1	X	0	1	1	1	0	1	0	0	M1. PA = Out. PB = In
0	1	1	0	1	0	X	0	0	0	1	0	1	0	0	M1. PA = In. PB = Out
0	1	1	0	1	1	X	0	0	0	1	0	1	0	0	M1. PB = In
0	1	1	1	1	0	X	1	1	0	1	0	1	0	0	M1. PA = In. PB = Out
0	1	1	1	1	1	X	1	1	0	1	0	1	0	0	M1. PB = In
1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	M2 & 0. PA = Bir. PB = O
1	0	0	0	0	0	1	0	1	0	1	0	1	1	1	M2 & 0. PA = Bir. PB = O
1	0	0	0	0	1	0	0	1	0	1	0	0	0	0	M2 & 0. PA = B. PB = In
1	0	0	0	0	1	1	0	1	0	1	0	1	1	1	M2 & 0. PA = B. PB = In
1	0	0	0	1	0	X	0	1	0	1	0	1	0	0	M2 & 1. PA = Bir. PB = O
1	0	0	0	1	1	X	0	1	0	1	0	1	0	0	M2 & 1. PA = B. PB = In

8.3 포트 C 블록의 세부 설계

이네이블 신호 생성기는 "진리표 합성기"로 설계하였으며, 이 의 출력들을 수정 없이 그대로 적용한다면, 각 비트별로 프로파게이션 지연가 다르게 나타나므로 Top 회로를 구성 한 후 시뮬레이션 과정에서 슈퍼씽크 체크에서 에러가 발생하게 된다. 따라서 포트 C 블록에서 이를 보정 해야하며, 이는 이네이블 출력을 랫치 시키는 것으로써 해결 할 수 있다. (그림 20)에 설계한 구성도를 나타냈다.



(그림 20) 포트 C 블록 구성도

8.4 어드레스 디코더 블록의 설계

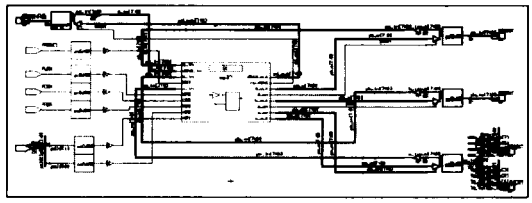
이 블록은 CPU의 어드레스 7개를 받아들여서 5가지의 경우로 PPI 3개의 a0와 a1 어드레스를 선택한다. 이 블록은 "Equation editor"로 (그림 21)과 같은 방정식을 수립하여 설계하였다.

```
INORDER = a9 a10 a11 a12 a13 a14 a15;
OUTORDER = s0 s1 s2 s3 s4 s5 s6;
s0 = ! a13;
s1 = ! a14;
s2 = ! a15;
s3 = ! a15;
s4 = a15 * a14 * a13 * a12 * a11 * ( ! a10 ) * ( ! a9 );
s5 = a15 * a14 * a13 * a12 * a11 * ( ! a10 ) * a9;
s6 = a15 * a14 * a13 * a12 * a11 * a10 * ( ! a9 );
```

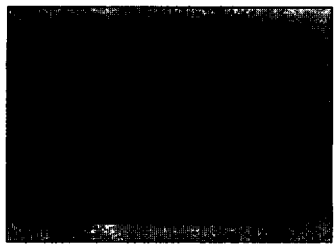
(그림 21) 어드레스 디코더용 방정식

8.5 패드 블록의 구성

PPI 1개에는 8비트의 양방향 데이터 버스, 포트 A/B/C 버스를 사용하여 24개의 양방향 패드를 채택하였고, 포트 C에는 8 비트의 양방향 패드를 채택하면서 비트별로 출력측을 이네이블 시켜서 개별 제어가 가능하도록 (그림 22)와 같이 설계하였고, (그림 23)에 개발 완료한 ASIC의 사진을 나타내었다.



(그림 22) 패드 블록 구성도



(그림 23) ASIC 사진

9. 결 론

착자 자동화 시스템의 입출력장치 제어를 위한 프로그램어블 주변장치 접속회로 제어기용 ASIC의 설계 및 시뮬레이션을 완료하고 ETRI의 반도체 공정라인을 이용하여 웨이퍼 가공 공정을 완료하였다. 본 연구에서 개발한 ASIC은 매 블록별로 LODECAP의 도면 편집기(Schematic Editor)를 이용하여 회로도 입력과 블록별 프리 시뮬레이션을 확인, 최종 회로도를 완성하였다.

개발된 ASIC은 세 개의 프로그램어블 주변장치 접속회로 제어기와 한 개의 어드레스 디코딩용 PAL을 1개의 칩에 집적시킨 것으로써, 50MHz에서 작동하고,

총 6,000 게이트로 ASIC화 되었다.

VHDL 선형 배선 캡(Cap.) 예측 모델을 사용하여 설계한 지연 체인 블록의 최종 지연시간과 시뮬레이션 결과는 일치하였다. 본 연구 결과로 구현한 ASIC은 착자기 공장 자동화 시스템의 입출력장치 제어기에 탑재하여 실용화 할 예정이다.

본 연구의 최종결과 칩을 제작하여 주신 ETRI의 관련 부서에 감사드립니다.

참 고 문 헌

[1] 이천희 외 1명, "착자자동화 시스템 및 통계분석 툴 개발", 한국정보처리학회 제3권 4호, pp.1014-1025, 1996년 7월.

[2] Pic 16/17 Micro controller data book, Microchip Technology Inc, 1995.

[3] A.I. Kayssi, K.A. Sakallah, T.M. Burks, "Analytical transient response of CMOS inverters." IEEE Trans. Circuit and Systems-I, Vol.39, pp. 42-45, Jan. 1992.

[4] E.Y. Chung, B.H. Joo, Y.K. Lee, K.H. Kim and S.H. Lee, "Advanced Delay Analysis Method for Submicron ASIC Technology." Proc. of IEEE ASIC Seminar, pp.471-474, 1992.

[5] J. Qian, S. Pullela and L.T. Pillage, "Modeling the Effective Capacitance for the RC Interconnect of CMOS Gates." IEEE Transactions on Computer-Aided Design. 13(12) : pp.1526-1535, Dec. 1994.

[6] N. Menezes, S.Pullela and L.T. Pillage, "Simultaneous Gate and Interconnect Sizing for Circuit-Level Delay Optimization." Proc. of 32nd ACM /IEEE Design Automation Conference. pp.690-695, Jun. 1995.

[7] 김동욱, 최태용, 정병권, "CMOS 인버터의 지연시간 모델", 전자공학회 논문지 제34권 C편 제6호, pp.357-367, 1997년 6월.

[8] 0.8-Micron Gate Array Library, VLSI Technology, INC, April. 1992.

[9] 배영환 외 10명, "수동 및 자동설계를 지원하는 VHDL 통합 설계 CAD 시스템", 1996 세계 한민족 과학자 종합학술대회, 1996년 7월.

[10] Neil H. E. Weste, Kamran Eshraghian, "Principle of CMOS VLSI DESIGN," Addison-Wesley, 1993.

임 태 영

e-mail : tylim@etri.re.kr

1971년 한양대학교 공업경영학과 졸업(학사)

1989년 한양대학교 산업대학원(전자공학 석사)

1994년 9월~현재 청주대학교 전자공학과 박사과정

1981년~현재 한국전자통신연구원 재직중

관심분야 : VLSI 설계, ASIC, Power IC, IP 개발 등

이 천 희

e-mail : yicheon@chongju.ac.kr

1968년 한양대학교 전자공학과 졸업, 동대학원 졸업

1975년 성균관대학교 대학원 전자자료처리학과 졸업

1986년 성균관대학교 전자공학과 공학박사학위 취득

1979년~현재 청주대학교 전자공학과 교수

1983년~1985년 미국 캘리포니아 산호세 주립대학 객원교수

관심분야 : VLSI Layout, ASIC, DRAM, CAD Tool 개발 등