UART 디바이스의 VHDL 설계

김 성중, 손 승일
한신대학교 정보과학대학

A VHDL Design of UART(Universal Asynchronous Receiver Transmitter) Device

Sung Jung Kim, Seung Il Sonh
Hanshin University, College of Information Sciences
E-mail : otru2@msn.com

요 약

인터넷의 사용이 증가, 네트워크 기술이 발달하면서 컴퓨터 및 하드웨어 장비는 고속화 대용량화, 소형화 추세로 가고 있고, 기존에 외부 인터페이스와의 데이터 송수신 또한 병렬 포트를 이용한 통신이 많았으나, 외부 장비의 소형화와 고속화 그리고 휴대화가 요구되면서 차츰 직렬 포트를 이용하여 적은 전송라인을 이용한 외부 장비와의 인터페이스가 요구 되게 되었다. 본 논문에서는 내부 모듈간의 인터페이스와 외부 장치와의 데이터 송/수신이 가능한 UART 인터페이스 모듈을 하드웨어 설계언어인 VHDL 언어를 이용하여 설계하였으며, FPGA 칩인 Xilinx(SpartanII) 테스트 보드에 다운로드하여 시뮬레이션 하였다. 또한 양방향성 공통 버스로의 인터페이스 회로 설계와 다른 클럭으로 동작하는 시스템과의 비동기 회로의 동작 메커니즘을 쉽게 설계하였고, 비동기 통신 기능에 있어서 실제로 사용이 가능하도록 설계 하였다.

1. 서 론

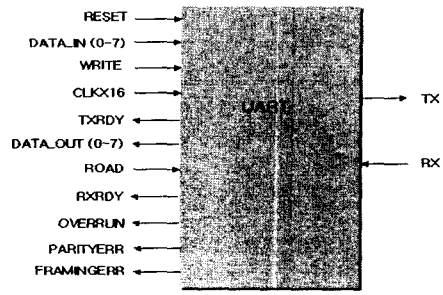
현대 사회의 컴퓨터 및 하드웨어 장비는 고속화, 대용량화, 소형화가 지속적으로 발전되면서 기존의 소프트웨어로 구성된 모듈이 하나씩 하드웨어로 제작되어 고속화, 대용량화를 꾀하고 있다. 각 컴퓨터, 하드웨어 모듈들이 소형화가 되면서 각 모듈간의 인터페이스 또한 많은 핀이 필요 하는 병렬적인 버스라인보다 많지 않는 핀으로 데이터를 효율적으로 송/수신할 수 있는 직렬 데이터 송수신을 필요로 하게 되었다. 기존에 외부 인터페이스와의 데이터 송수신 또한 병렬 포트를 이용한 통신이 많았으나, 외부 장비의 소형화와 고속화 그리고 휴대화가 요구되면서, 차츰 직렬 포트를 이용하여 적은 전송라인을 이용한 외부 장비와의 인터페이스가 요구 되게 되었다. 현재 상용화 되어있는 UART인터페이스 장비를 보면, RS-232, USB IEEE1394 등으로 기존의 병렬버스에 비교하면 적은 데이터 라인으로 고속의 데이터를 처리할 수 있는 장비들이 나오고 있다. 또한 각 장비 스펙의 버전이 올라가면서 더욱더 고속의 데이터 처리가 가능하게 설계되고 있다.

본 논문에서는 내부 모듈간의 인터페이스와 외부 장치와의 데이터 송/수신이 가능한 UART 인터페이스 모듈을 하드웨어 설계언어인 VHDL 언어를 이용하여 설계하고, FPGA 칩인 Xilinx(SpartanII) 테스트 보드에 다운로드하여 시뮬레이션 하였으며, 설계한 UART칩을 활용하여 IBM PC와 직/병렬데이터의 송수신이 원활하게 연동되도록 구현하였다[1][2].

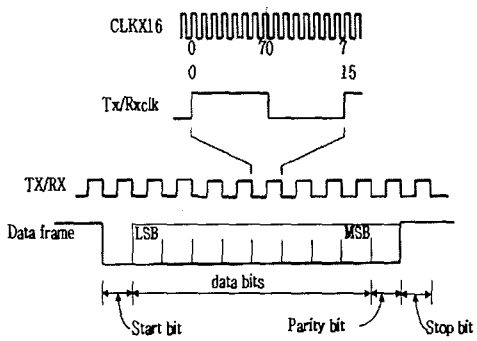
2. 본 론

2.1 UART의 개요

UART는 거의 모든 마이크로컨트롤러가 내장하고 있는 범용 비동기식 송수신장치로서, 8비트 병렬데이터를 비동기식 직렬전송을 하고, 직렬데이터를 비동기식 8비트 병렬데이터로 바꾸어 전송한다. 본 UART는 전이중 방식으로 데이터를 송수신하도록 설계하였으며, UART의 설계와 테스트 벤치의 작성, 합성, 그리고 기능 및 타이밍 시뮬레이션 등 VHDL을 이용한 설계의 전 과정에 대한 것이며 직렬 송/수신 8비트 데이터패리티 비트, 스톱비트로 고정되어 있고 패리티 에러(ParityError) 및 프레밍 에러(Framing Error) 검출 기능을 내장하고 있다. UART의 외부 연결핀 및 엔터티 선언은 그림 1과 같이 8비트 양방향성 데이터 버스 읽기 및 쓰기를 위한 제어 핀, 그리고 통신 에러 등을 나타내는 핀들로 구성하였고, 그림2와 같은 프레임 형식으로 데이터가 송수신 된다[3].



[그림 1] UART 입/출력도

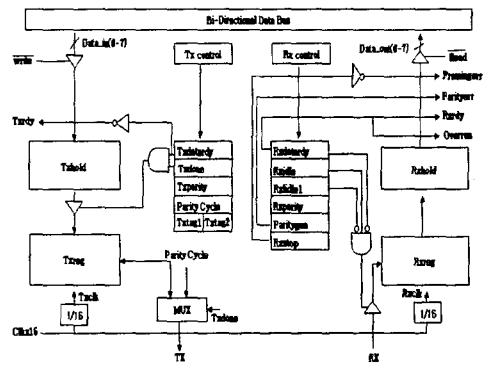


[그림 2] 직렬데이터 프레임 형식

2.2 UART모듈 설계

2.2.1 UART 내부 구성

본 UART의 내부구성은 그림3에서 보듯이 송신부와 수신부로 나뉘어 설계 하였으며, 시스템 버스의 양방향성 공통데이터 버스에 인터페이스 될 때 병렬데이터 입/출력 제어신호 관계를볼 수 있다 [4][5][6].



[그림 3] UART 내부 구성도

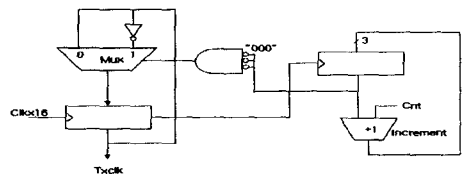
송/수신기능을 원활하게 하기 위하여 클럭은 분주회로를 통한 16분주 클럭을 사용하며 UART와 버퍼 기능을 이루는 구성요소들에 의해 사용되는 시스템 클럭이다. Txhold는 일정량의 버퍼레지스터를 제공함으로써 CPU와 직렬 장치들 간의 데이터 스트림이 대등하도록 맞추기 위함이다. Txreg는 Txhold로부터 들어오는 시리얼신호에 스타트 비트, 스탑 비트, 패리티 비트를 추가하여 단일 비트 스트림으로 변환하여 직렬장치들로 전송하는 역할을 한다. Rxreg는 직렬장치들로부터 들어오는 단일 비트 스트림에서 스타트 비트 스탑 비트, 패리티 비트 등을 분리, 검증한 후 단일 비트스트림을 Rxhold로 전달한다. Rxhold는 단일 비트스트림을 CPU에 맞는 방식으로 변환하여 전송하게 된다.

2.2.2 Tx 모듈의 설계

송신부의 모듈 설계는 통신 속도를 UART의 마스터 클럭을 16분주한 송신 클럭 발생기(Txclk)와 패리티 발생회로 및 송신 종료 검출회로를 갖춘 직렬 쉬프트 레지스터(Txreg), 외부 인터페이스를 위한 제어부(Txio)로 구성된다.

2.2.2.1 Make_Txclk 생성

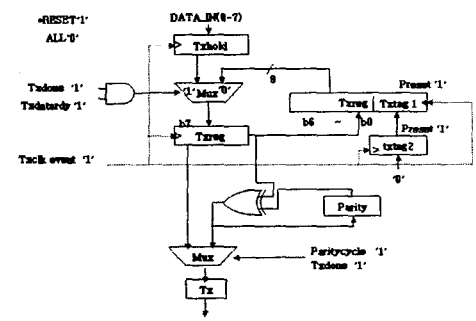
송신 클럭은 그림4와 같이 3비트 카운터를 이용하여 송신 클럭 발생회로를 토글시킴으로써 UART의 마스터 클럭을 16분주한다. 토글에 의하여 송신 클럭을 발생하기 위해서는 처음에 리셋일 때 카운터를 초기화 시켜야 한다. 송신클럭은 Txreg를 쉬프트 시킬 때 사용된다.



[그림 4] Make_Txclk 블럭도

2.2.2.2 직렬 송신 쉬프트 레지스터

그림5의 직렬 송신 쉬프트레지스터인 Txreg는 외부로부터 쓰기 신호에 의하여 입력된 8비트 데이터를 받아 래치된 Txhold데이터를 Txreg로 로드한 후 Txclk에 의하여 쉬프트 한다. 쉬프트된 데이터에는 스타트 비트 패리티 비트, 스탑 비트가 첨가되어 10비트의 데이터 형식을 갖추게 된다. Txtag1과 Txtag2는 10비트 직렬 데이터 프레임을 만들고 송신 종료를 감지하기 위하여 사용된다.

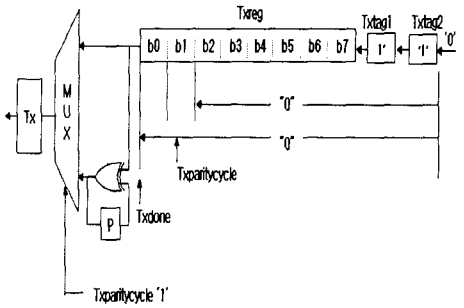


[그림 5] Txshift 레지스터 블록도

직렬 송신과정은 Txtag1과 Txtag2모두가 '1'로 초기화하고 쉬프트 레지스터 Txreg는 래치된 8비트 병렬데이터 Txhold를 로드한다. 송신할 데이터는 Txdone과 Txdatardy신호에 따라 제어된다. 송신 종료를 나타내는 Txdone의 발생은 쉬프트 할 때 Txtag2에 '0'을 채우게 되므로 Txreg와 Txtag1, Txtag2모두 '0'일 때 송신 종료이다.

2.2.2.3 Txshift 와 패리티비트 생성

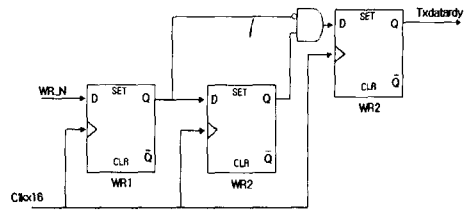
송신할 8비트 데이터가 로드 된 후 직렬전송이 시작되면서 패리티 비트의 계산이 이루어 진다. 패리티 비트 계산 과정의 구성은 그림6은 직렬 쉬프트 레지스터와 패리티 비트 생성기능의 구조를 보여주고 있다. 그림에서 보듯이 송신 종료 Txdone는 8비트데이터와 시작비트, 패리티 비트를 모두 전송하였음을 나타낸다. 따라서 패리티 비트를 송신 비트 프레임에 삽입하려면 송신 종료이전에 패리티 비트 삽입 싸이클을 감지하기 위한 회로가 필요하므로 설계하였다. 패리티 싸이클 감지는 송신 데이터가 초기화 될 때 Txtag1비트가 쉬프트 되어 쉬프트레지스터의 Txreg(0)에 도달 한때이다. Txtag1 과 Txtag2 비트에 있던 '1'의 값이 Txreg(0)과 Txreg(1)에 도달하면 패리티 계산은 종료하고 그다음 송신될 비트에 패리티 비트 값을 MUX를 통하여 출력하게 된다.



[그림 6] 패리티 비트 생성 블럭도

2.2.2.4 시스템 인터페이스(Txio)

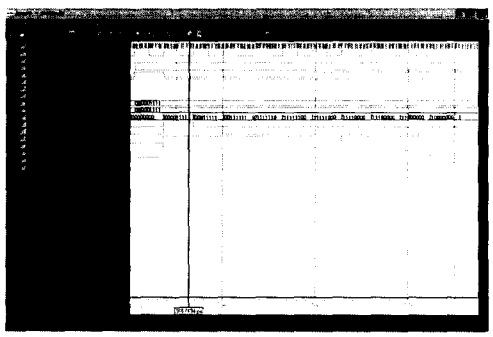
그림7은 시스템 인터페이스(Txio)를 설계한 것이다. Txreg에 전송할 데이터를 초기화 할 때 사용했던 Txdatardy는 외부로부터 입력되는 8비트 데이터가 준비되었음을 알려주는데 사용된다. 외부 시스템으로부터 송신 데이터의 준비는 Wr_n신호와 함께 이루어진다. 이때 외부 시스템과의 인터페이스이므로 Txclk와는 무관하게 작동되며 2비트 레지스터를 이용하여 외부 제어신호 Wr_n를 감지한 후 Clkx16의 상승 에지에 맞춰 내부 제어신호인 Txdatardy를 생성하는 회로이다.



[그림 7] txio 블럭도

2.2.2.5 Tx모듈의 파형 분석

Tx모듈의 파형은 그림8과 같다. 전반적인 파형 분석은 다음과 같다. Txdone신호와 Txrdy가 '1'일 때 외부로부터의 데이터를 받아들 수 있게 된다. 외부로부터 Data가 준비되면 Wr_n이 '1'이 된다. 이때 Data는 Txhold에 래치 되고 Wr_n의 '1'이 Wr1과 Wr2로 쉬프트 되면서Write의 Falling Edge가 검출되면 Txdatardy가 Set된다. Txdone이 '1'이고 Txdatardy가 '1'인 상태에서 송신 레지스터가 Txtag1과 Txtag2가 Set으로 초기화 되고 초기화 과정에 의하여 Txdone이 '0', Txdone1이 '1'일때 Txdatardy가 '0'으로 래치 되어 Txdatardy가 '0'인 구간에서 직렬전송이 시작된다. 파형 분석하는 과정에서 데이터 인터페이스와 각 제어 신호들의 제어 동작 타이밍을 주의 깊게 보아야 하며, 특히 비동기 신호의 에지를 검출하기 위한 제어신호 생성과 타이밍을 주의 깊게 보아야 한다.

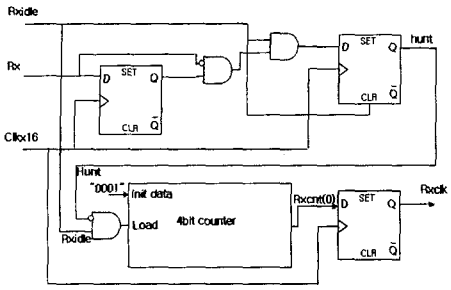


[그림 8] Tx 모듈의 파형

2.2.3 Rx 모듈 설계

UART의 수신 모듈의 주요 기능은 수신클럭 발생기(Make_Rxclk)와 직렬데이터의 수신(Rx_Proc) 시스템 인터페이스(Rxio)로 나누어 볼 수 있다. 직렬데이터 수신은 송신보다 좀 더 복잡한 구조이다. 대기 상태(idle)에서 직렬데이터 프레임의 시작 비트를 검출하고 데이터

수신이 완료되면 패리티 비트, 프레임, 오버런 등의 에러를 감지하여야 한다. 우선 수신 클럭 생성을 보면 그림 9와 같다.



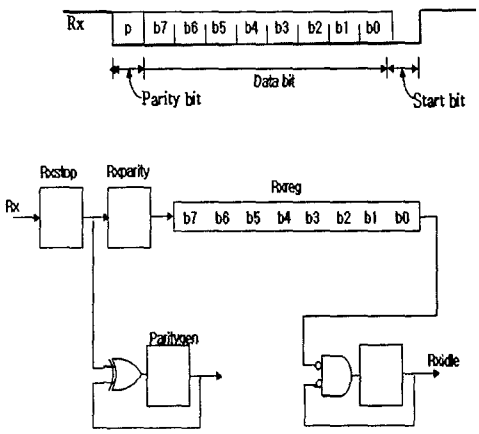
[그림 9] make_rxclk 블록도

2.2.3.1 Make_Rxclk 생성

수신 클럭을 발생시키는 분주회로는 송신클럭과는 달리 4비트 카운터를 이용하여 데이터 프레임의 MSB를 클럭 신호로 사용한다. 직렬 수신에서 시작비트가 검출되지 않은 대기 상태에 서는 분주용 카운터가 정지 상태에 있게 되며 카운터는 초기값을 "0001"로 갖는다. 이것은 시작 비트의 검출을 위해서 수신입력에 1비트 레지스터를 이용하였기 때문이다. 시작비트가 검출된 이후에는 4비트 카운터를 이용하여 Clkx16을 16분주한 수신 클럭 Rxclk이 연속적으로 발생된다. 수신 클럭은 10비트 직렬데이터 프레임이 모두 수신되었을 때 Rxidle에 의하여 정지된다.

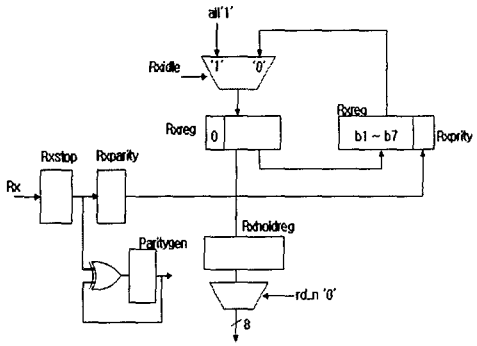
2.2.3.2 직렬데이터 수신(Rx_Proc)

수신부 제어의 시작은 대기상태를 나타내는 Rxidle이다. Rxidle은 10비트 직렬데이터 프레임이 모두 수신된 것을 감지한 후 직렬수신의 완료를 표시하게 된다. 비동기 수신 제어신호의 Rxidle 발생시키는 회로는 그림10, 11과 같이 구현 할 수 있다. 수신부의 쉬프트 레지스터는 초기화 될 때 '1'이며 Rx를 통해서 시작비트가 수신 쉬프트 LSB인 "b0"에 도달하면 10비트 프레임이 모두 수신된 것과 같다.



[그림 10] Rxidle 생성회로

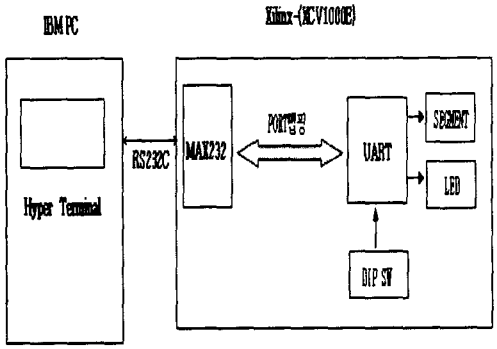
이때부터 직렬수신은 대기 상태가 되며 수신된 데이터를 외부 시스템에 출력할 수 있다. 직렬데이터의 수신이 완료 되 면 대기 신호 Rxidle이 상승하는 것을 감지하여 수신 레지스터의 8비트 값을 시스템에 출력할 수 있도록 수신데이터 준비 신호 Rxdatardy를 출력한다. 외부 시스템에서 읽기 스트로브 신호인 Rd_n가 주어지면 비로소 외부 버스에 수신한 8비트의 데이터를 출력하고 아울러 에러 신호 등도 함께 출력한다.



[그림 11] Rxshift 레지스터 블럭도

2.3 UART 설계 모듈 검증

VHDL를 이용하여 설계한 UART를 FPGA로 합성하여 보고 타이밍 시뮬레이션을 함으로써 완전한 설계가 되도록 검증하였다. 테스트벤치를 작성하여 UART모듈의 송수신을 루프 하여 시뮬레이션하는 방법도 있으나 그림12과 같이 구성을 하여 IBM PC와 연동하여 모듈의 검증하였다. 우선 IBM PC와 UART설계 모듈이 장착된 Xilinx-(Spartan II)와의 인터페이스를 위하여 데이터 통신용인터페이스를 구성하고 표1,2와 같이 입/출력장치와 데이터 통신용 IC와의 핀 연결 관계를 정의 하여 IBM PC의 하이퍼 터미널을 이용하여 데이터가 송/수신되는지를 검증하였다[7].



[그림 12] IBM PC와의 연동 구성도

Signal Name	Uart Port_Name	핀 이름	핀 번호	Signal Name	Uart Port_Name	핀 이름	핀 번호
Seg0	Data_Out(0)	Jp3-10	114	Digit6	Sel_Digit(5)	Jp3-9	113
Seg1	Data_Out(1)	Jp3-11	115	Expb17	Hold_Dat(0)	Jp4-41	199
Seg2	Data_Out(2	Jp3-15	119	Expb19	Hold_Dat(1)	Jp4-43	201
Seg3	Data_Out(3)	Jp3-16	120	Expb21	Hold_Dat(2)	Jp4-45	203
Seg4	Data_Out(4)	Jp3-17	121	Expb23	Hold_Dat(3)	Jp4-47	205
Seg5	Data_Out(5)	Jp3-18	122	Expb13	Clk_16X	Jp4-34	192
Seg6	Data_Out(6)	Jp3-19	123	Expb15	Clk_1X	Jp4-36	194
Seg7	Data_Out(7)	Jp3-20	125	Led7	Rxrdy	Jp2-48	102
Digit1	Sel_Digit(0)	Jp3-4	108	Led6	Txrdy	Jp2-47	101
Digit2	Sel_Digit(1)	Jp3-5	109	Led2	Parityerr	Jp2-43	97
Digit3	Sel_Digit(2)	Jp3-6	110	Led0	Framingerr	Jp2-41	95
Digit4	Sel_Digit(3)	Jp3-7	111	Expb9	Wr_En_Tmp	Jp4-30	187
Digit5	Sel_Digit(4)	Jp3-8	112	Led1	Overrun	Jp2-42	96

[표1] 출력장치와 통신용IC핀 연결 관계

Signal Name	Uart Port_Name	핀 이름	핀 번호	Signal Name	Uart Port_Name	핀 이름	핀 번호
Fpga_Clk	Clkx16	Jp2-26	80	Dipsw7	Data_In(6)	Jp2-9	63
Dipsw1	Data_In(0)	Jp2-3	57	Dipsw8	Data_In(7)	Jp2-13	67
Dipsw2	Data_In(1)	Jp2-4	58	Push1	Read	Jp2-14	68
Dipsw3	Data_In(2)	Jp2-5	59	Push2	Write	Jp2-15	69
Dipsw4	Data_In(3)	Jp2-6	60	Push4	Reset	Jp2-17	71
Dipsw5	Data_In(4)	Jp2-7	61	Frx	Rx	Jp2-19	73
Dipsw6	Data_In(5)	Jp2-8	62	Ftx	Tx	Jp2-20	74

[표2] 입력장치와 통신용IC핀 연결 관계

3. 결 론

UART는 핵심제어기 이전에는 외장형 부품정도로 사용해 오다가 점차로 반도체 내부에 필요한 기능만을 골라서 집적하거나 응용하여 사용해 나가는 추세이다. 현재 거의 모든 마이크로컨트롤러가 내장하고 있는 범용비동기식 송수신장치로서, 8비트 병렬데이터를 비동기식 직렬전송을 하고, 직렬데이터를 비동기식 8비트 병렬데이터로 바꾸어 전송한다.

본 논문에서는 실제 사용 가능한 모듈을 설계하였으며, UART를 분석을 통해 양방향성 공통 데이터 버스로의 인터페이스 회로 설계와 시스 템과의 다른 클럭으로 작동하는 비동기회로의 설계 등을 살펴 볼 수 있다. 또한 VHDL로 설계했기 때문에 32Mbps까지의 전송속도 지원하 며, 총 게이트 사용수는 789개가 사용되었다.

참고문헌

[1] 이 상구 역 "8051 마이크로 컨트롤러" 1996.
[2] 황 상미 편저 "Serial Communication" 1992.
[3] 박 현철 저 " HDL을 이용한 회로 설계"2001.
[4] Ben. Cohen, "VHDL coding styles and methodologies" 1997.
[5] 차 영배 저 "VHDL을 이용한 CPLD/FPGA 설계" 2003.
[6] 진 달복 저 "AVR과 그 응용" 2002.
[7] Carton "serial UART" 2003.