대한상공회의소
서울기술교육센터

2025.11.17

SoC SPI/I2C 통신 인터페이스 설계

하만 세미콘 아카데미 변지인

목차

01

프로젝트 개요

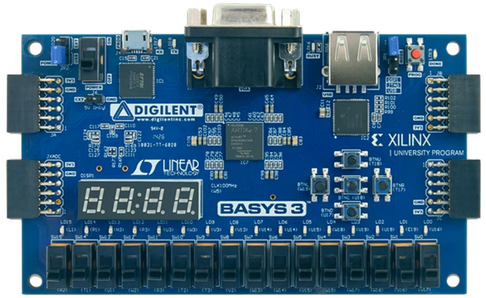
01 프로젝트 개요

1. 프로젝트 목표

- SPI 및 I2C Master/SIave 통신 모듈 설계
- MicroBIaze SoC 환경에서 AXI4-Lite 버스를 활용해 I2C Master IP 통합
- 통합된 I2C IP를 활용하여 C Application 구현

2. 개발 환경

- Basys3 FPGA, SystemVerilog, C, Vitis, Vivado



XILINX
VITIS™

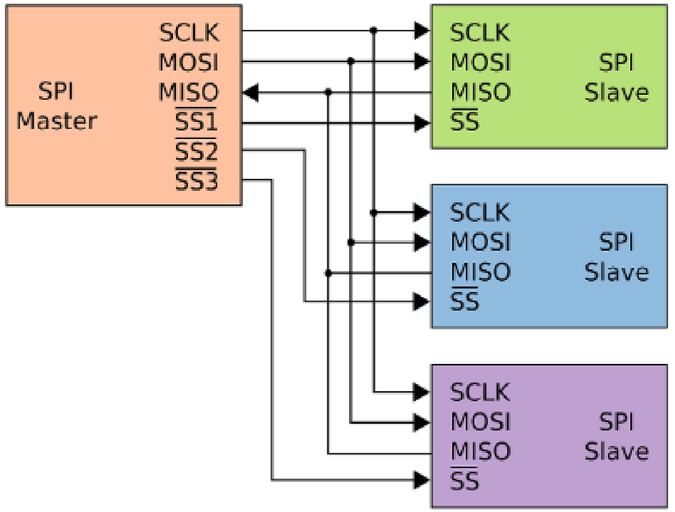
02

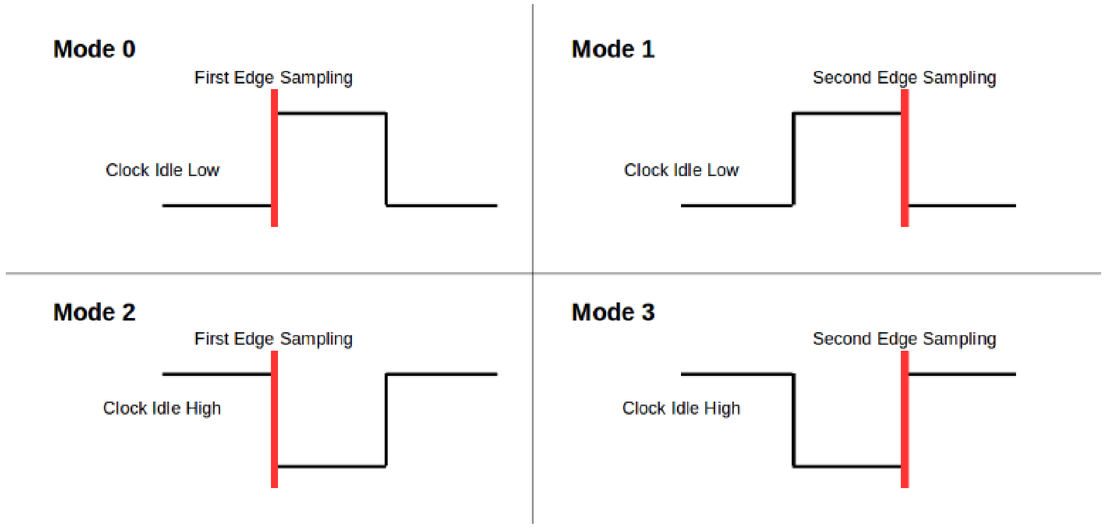
SPI

02 SPI

SPI (Serial Peripheral Interface)

- 마이크로컨트롤러와 여러 개의 주변장치의 고속 통신을 위한 동기식 직렬 통신 방식
- 4-Wire 구조: SCLK, MOSI, MISO, CS/SS 사용
- Clock Polarity(CPOL)와 Clock Phase(CPHA)에 따라 4가지 통신 모드를 가짐





02 SPI

SCLK (Serial Clock)

- Master가 생성하며, 데이터 전송 타이밍을 동기화하는 클럭 신호

MOSI (Master Out Slave In)

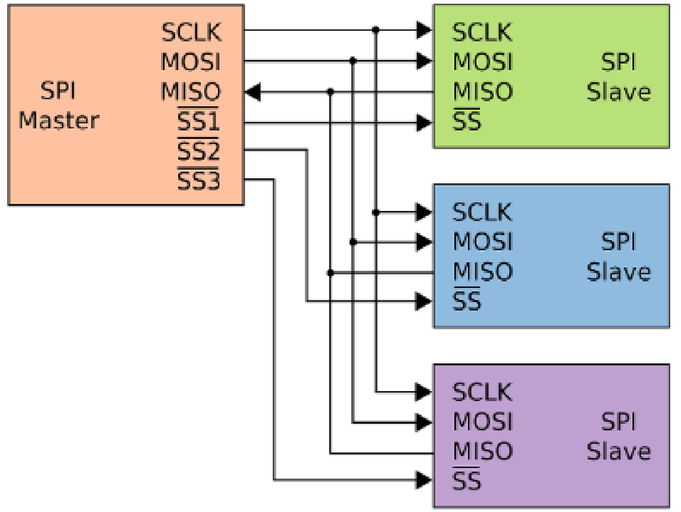
- Master가 Slave에게 데이터를 보내는 단방향 데이터 라인

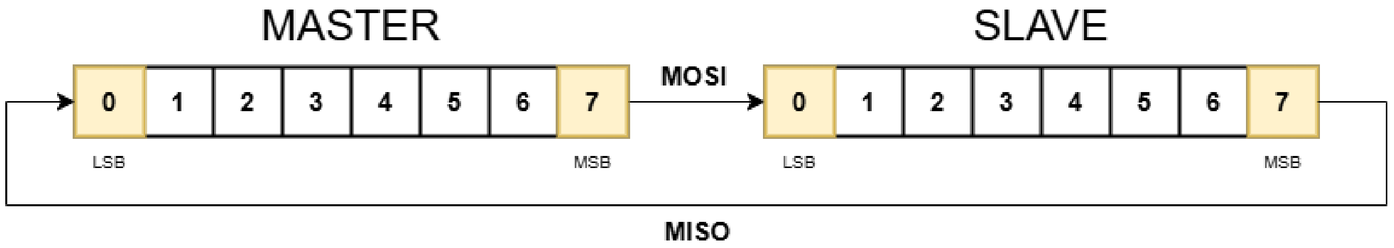
MISO (Master In Slave Out)

- Slave가 Master에게 데이터를 보내는 단방향 데이터 라인

CS (Chip Select)

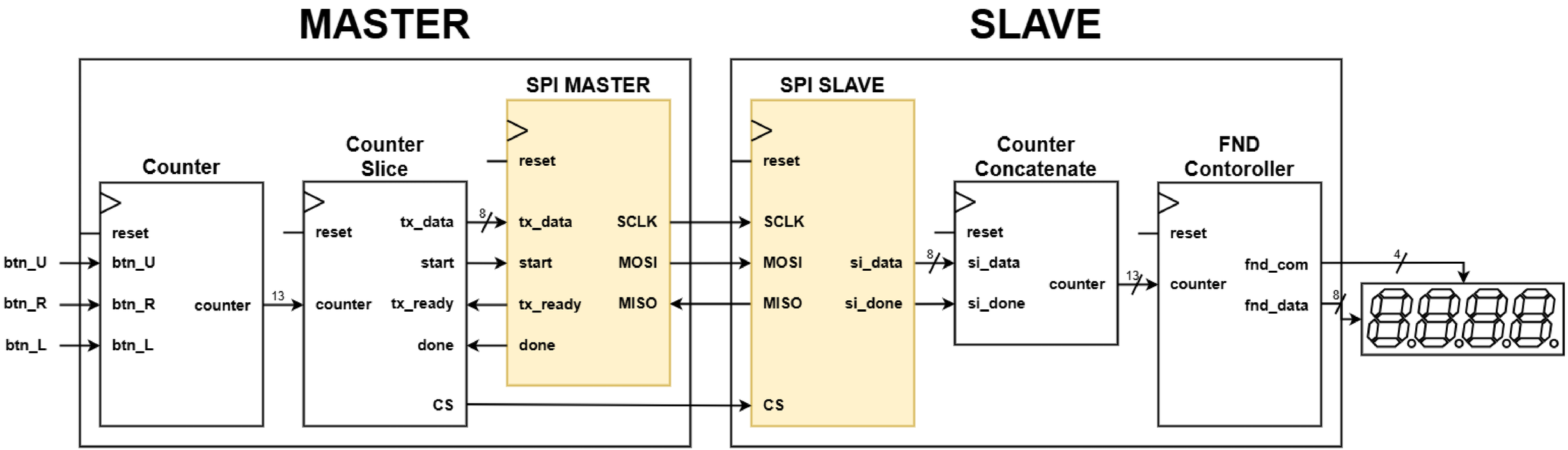
- 마스터가 통신할 Slave를 선택하고 통신을 활성화(Active Low)하는 제어 신호





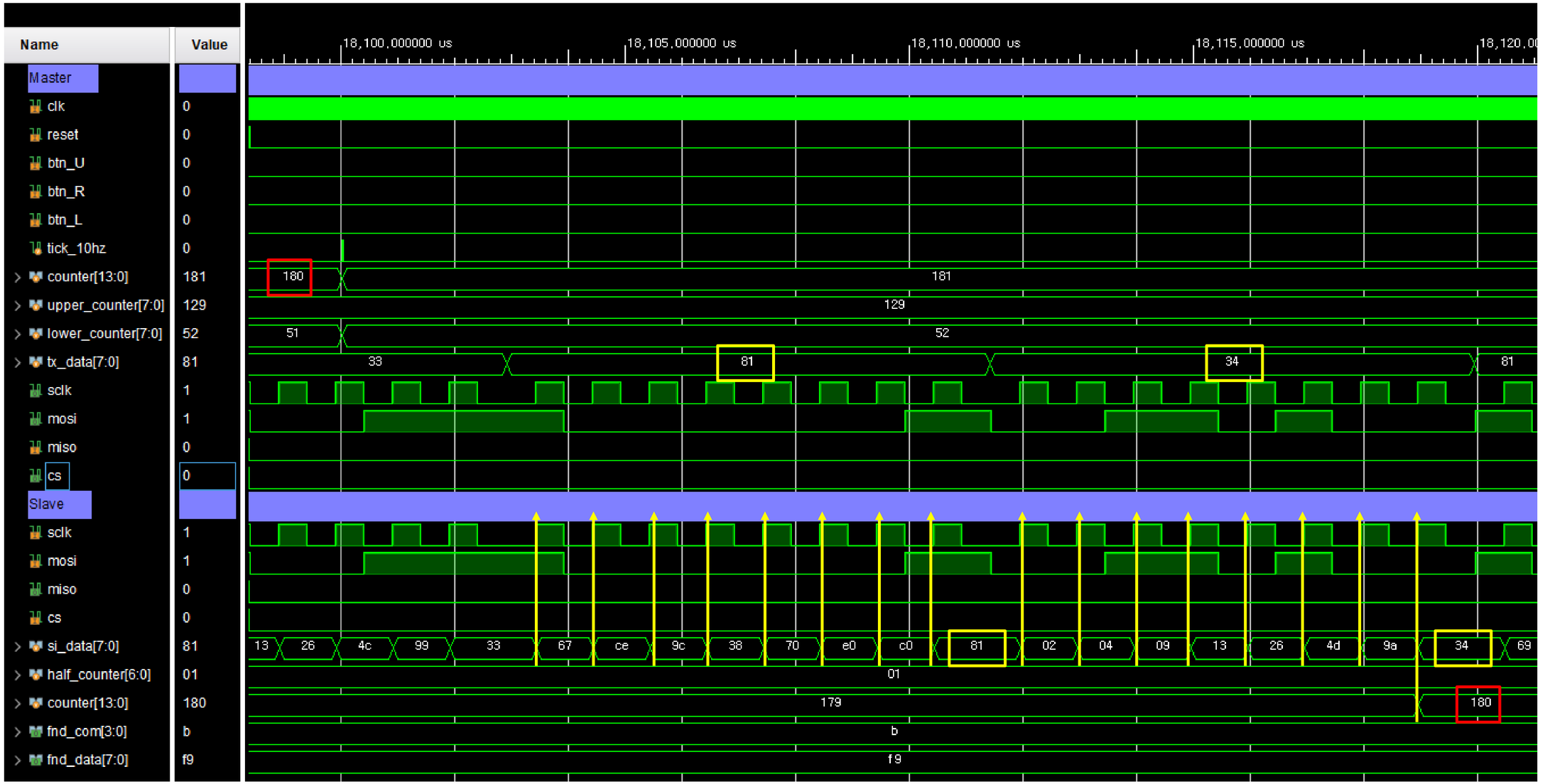
02 SPI

Block Diagram



02 SPI

Simulation



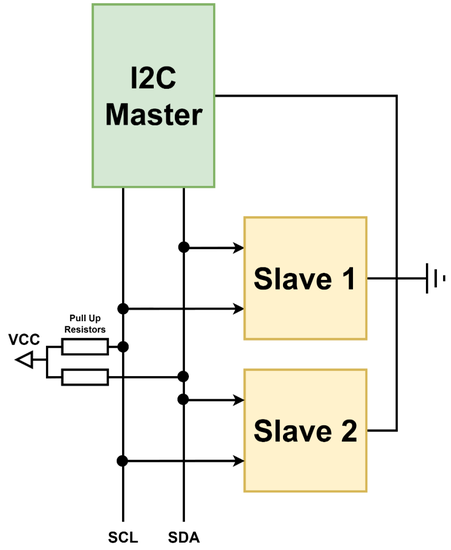
03

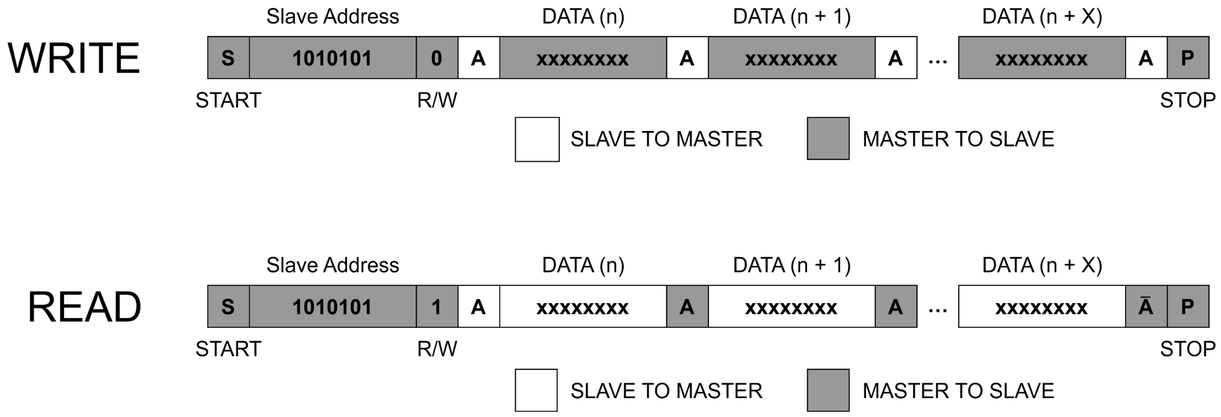
I2C

03 I2C

I2C (Inter-Integrated Circuit)

- 보드 내 칩 사이의 짧은 거리 통신을 위한 동기식 직렬 통신 방식
- SCL, SDA 두 개의 라인으로 통신
- SCL과 SDA는 Open-Drain 방식으로 모든 장치와 연결되며, 외부 풀업 저항이 필수
- 반 이중 통신(Half Duplex) : 한 번에 한 방향으로만 데이터 통신 가능





03 I2C

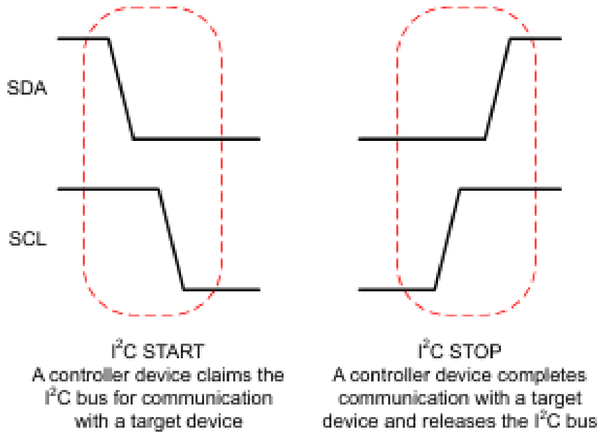
SCL (Serial Clock)

- Master가 생성하며, 데이터 전송 타이밍을 동기화하는 클럭 신호

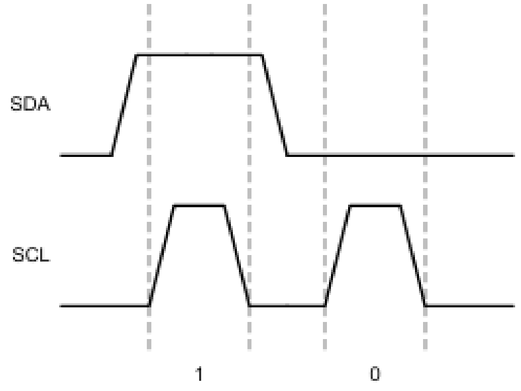
SDA (Serial Data)

- Master와 Slave가 데이터를 주고 받는 양방향 데이터 라인

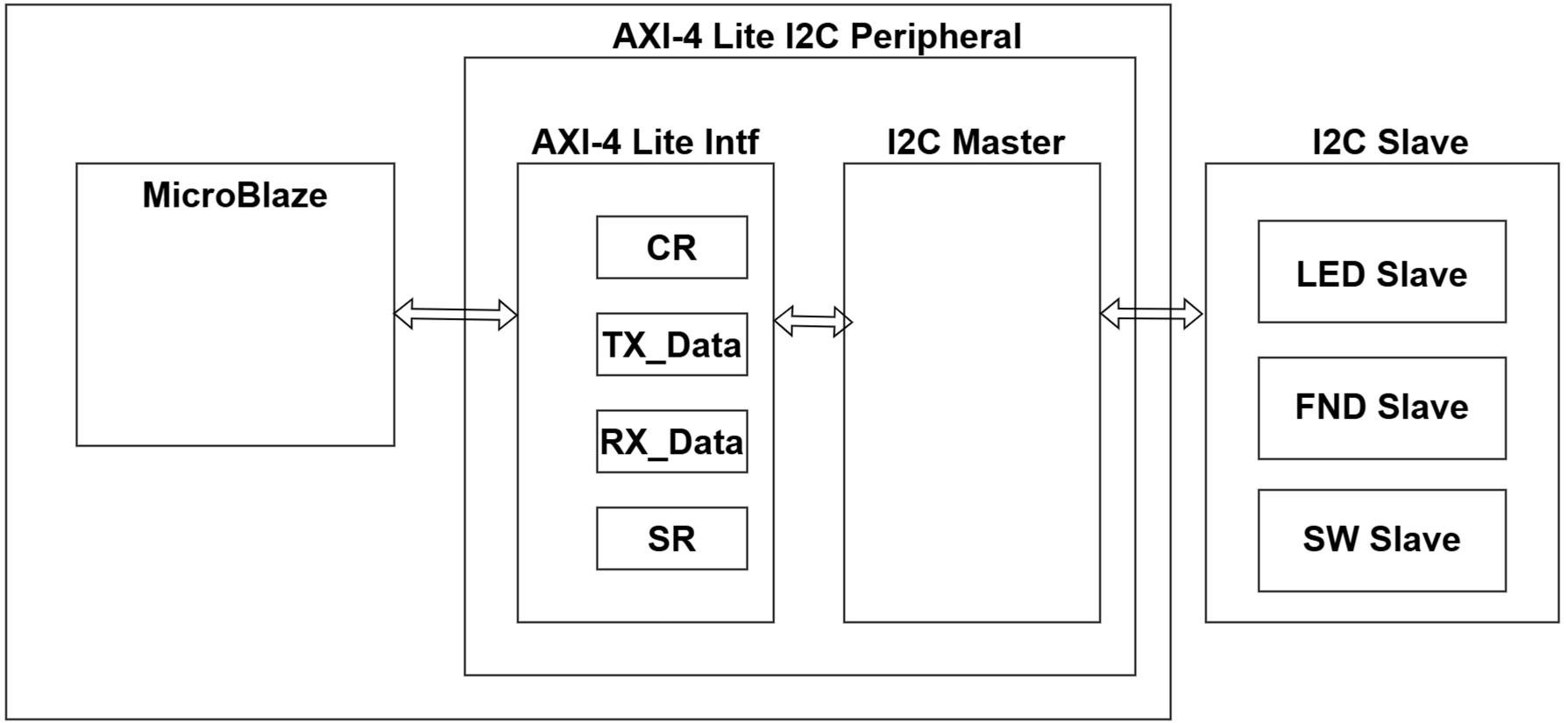
I2C START and STOP



SDA Representations



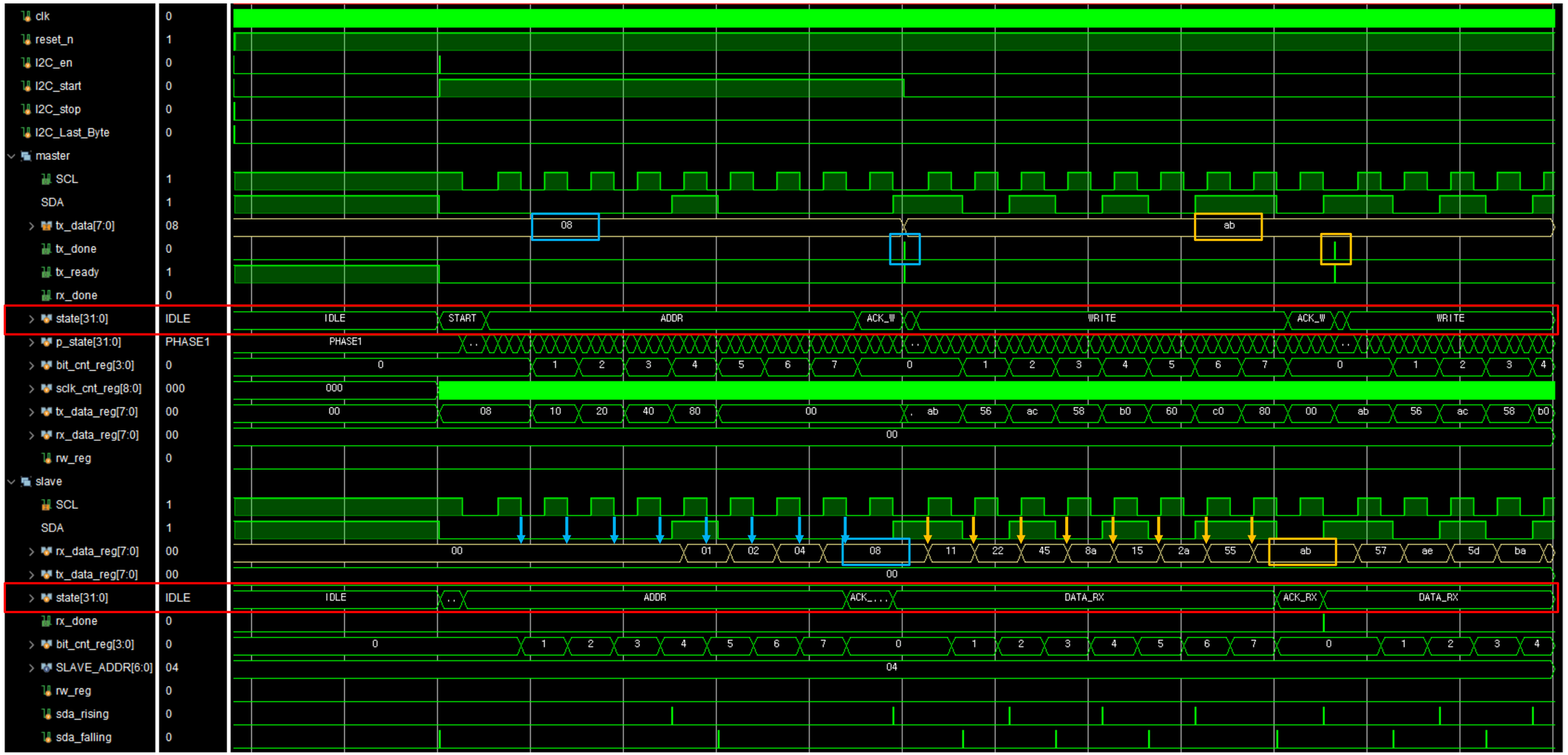
System Architecture



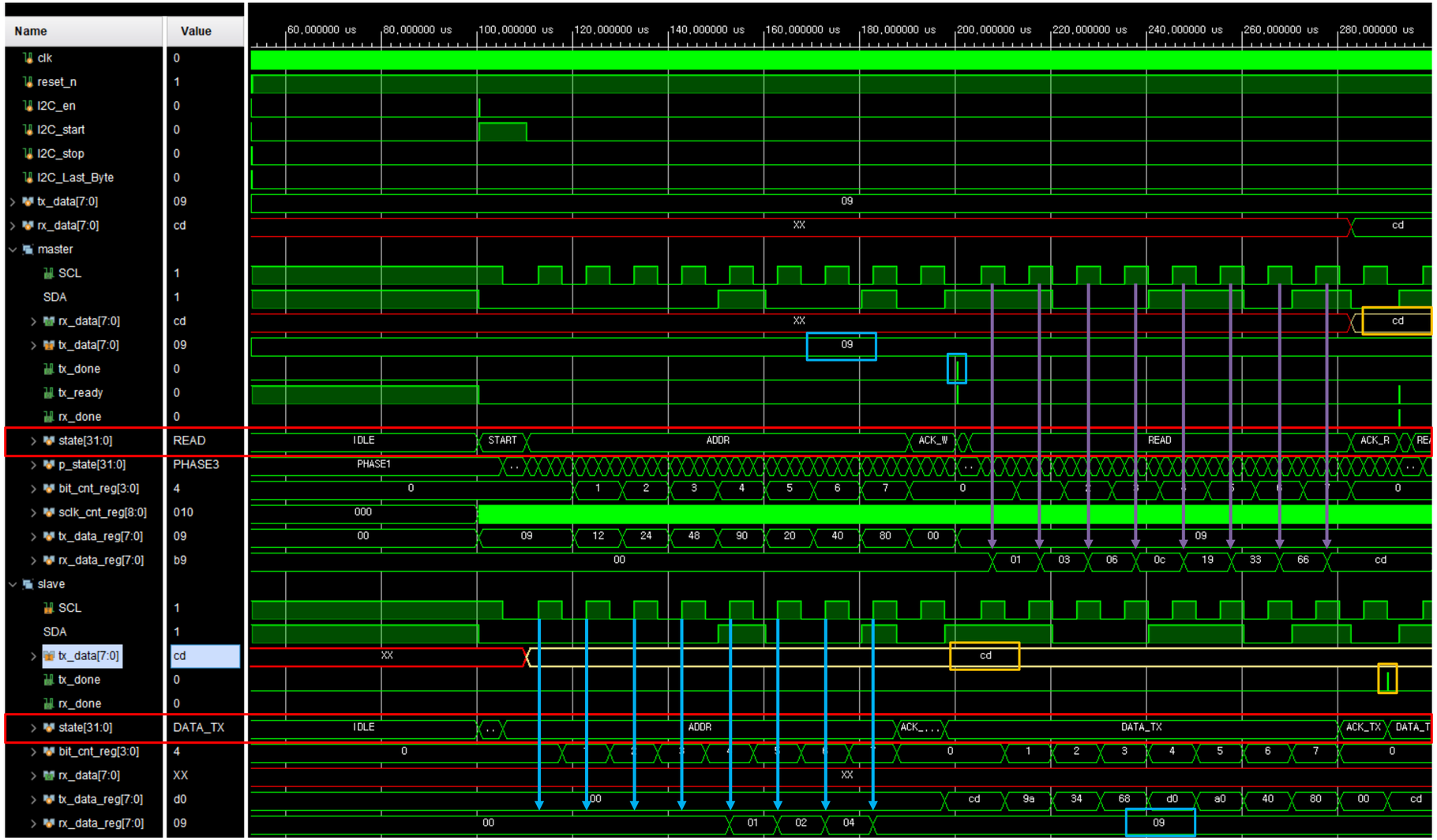
offset	Register	31…8	7	6	5	4	3	2	1	0
0000	CR	Reserved					I2C_Last_Byte	I2C_stop	I2C_start	I2C_en
0004	ODR	Reserved	tx_data							
0008	IDR	Reserved	rx_data							
000C	SR	Reserved						rx_done	tx_ready	tx_done

03 I2C

Write Simulation



Read Simulation



동작 시나리오

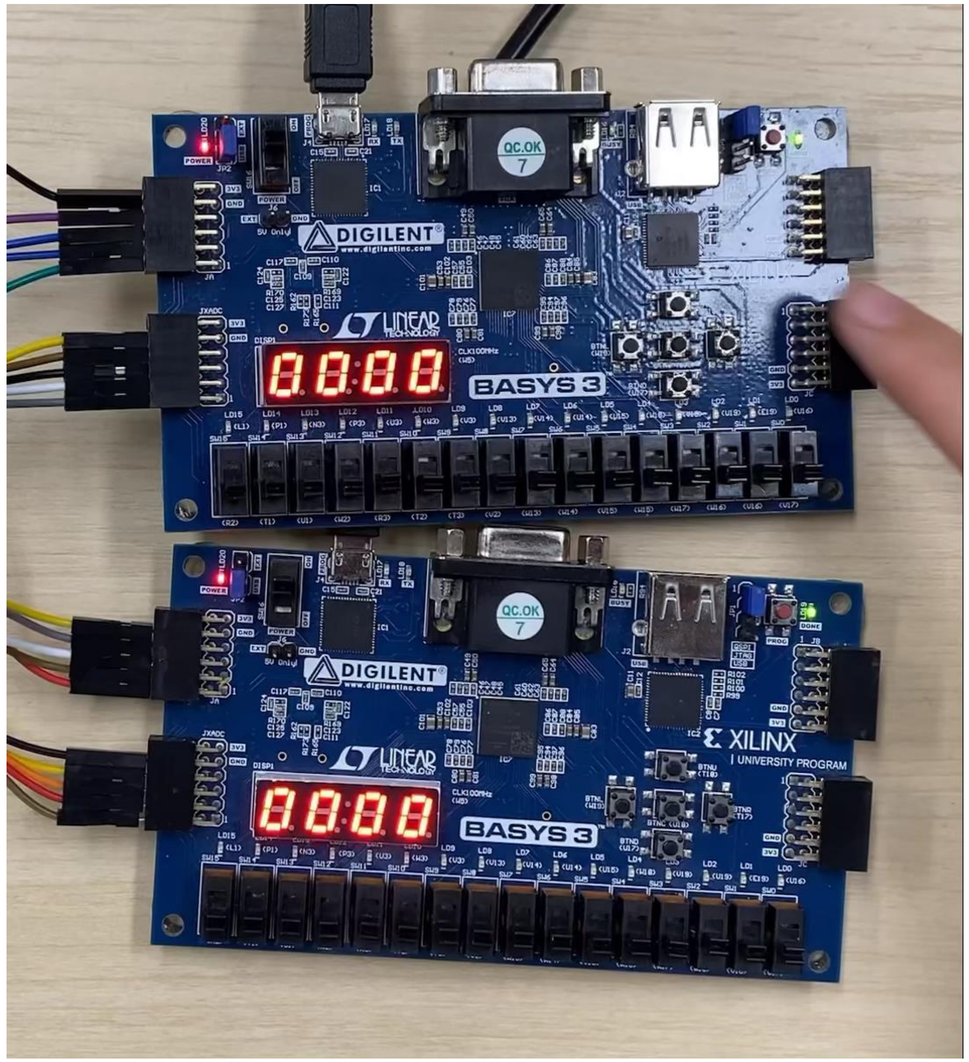
1. **시작 입력 대기 :** UART 입력으로 1이 들어올 때까지 대기
2. **랜덤 10진수 생성 :** MicroBlaze C 코드가 0~255(8 bit) 사이의 랜덤 10진수를 생성
3. **FND 출력 :** I2C Master가 랜덤 생성된 10진수를 FND Slave로 전송하여 표시
4. **SW 입력 :** LED가 순차적으로 다 꺼질 때까지 10진수의 2진수 변환 값을 스위치(8 bit)로 입력
5. **결과 비교 :** MicroBlaze가 SW Slave의 값을 읽어와 랜덤 값과 비교
6. **FND 표시 :** 결과에 따라 AAAA(정답) 또는 FFFF(오답)를 FND에 출력

04

동작영상

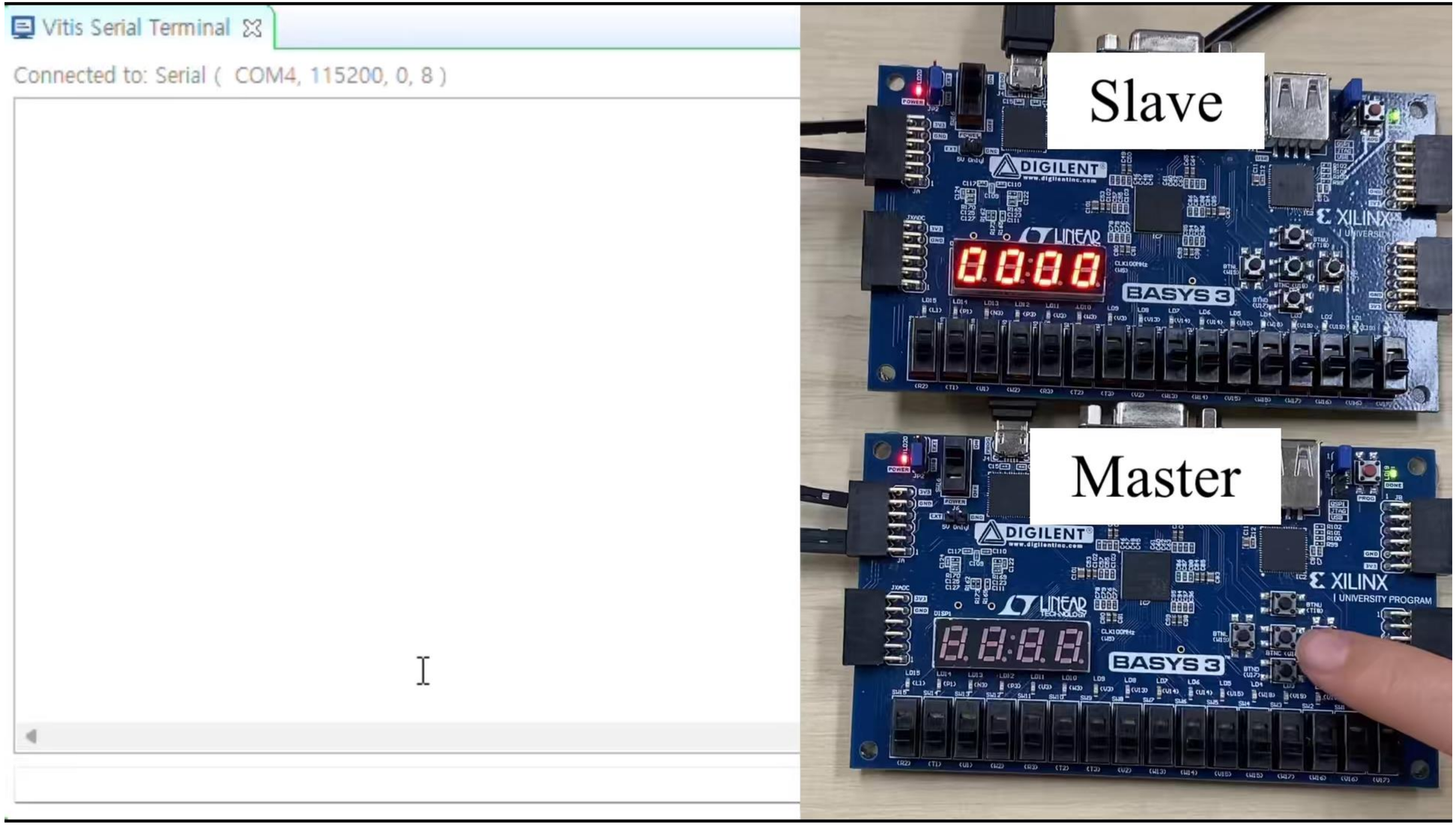
04 동작 영상

SPI 동작영상



04 동작 영상

I2C 동작영상



05

고찰

05 고찰

Trouble Shooting 1

1. 문제 현상

- 14 bit 카운터 데이터를 8 bit 단위로 단순 분할하여 전송 시, 타이밍이 조금만 어긋나도 데이터 수신 순서가 꼬이거나, 유효성 검사가 불가능

2. 해결 과정

- Master와 Slave에 Slice, Concatenate 모듈을 추가하여 카운터를 7 bit 단위로 나누고, tx_data MSB에 Upper/Lower 판별 비트를 추가하여 Slave가 데이터의 순서와 유효성을 인식하도록 설계

Counter Slice

```
if (tick_10hz) begin
    upper_counter <= {1'b1, counter[13:7]};
    lower_counter <= {1'b0, counter[6:0]};
end
```

Counter Concatenate

```
if(si_done) begin
    if (si_data[7]) half_counter <= si_data[6:0];
    else counter_reg <= {half_counter, si_data[6:0]};
end
```

05 고찰

Trouble Shooting 2

1. 문제 현상

- I2C Read일 때 Restart 조건 후 통신이 제대로 이어지지 않음
- SDA 제어를 output enable 방식으로 코딩하여 Master/Slave가 동시에 라인을 구동하는 충돌 구간이 발생

2. 해결 과정

- 현재 로직에서 HOLD 상태에 SCL 펄스를 추가 소모하는 방식으로 해결을 시도했으나, 프로토콜 표준과 벗어남
- Open-Drain 통신에 맞춰 output enable 방식으로 코딩하는 것이 아니라, SDA를 high에서 0으로만 강제하도록 코딩


```
logic pul_sda;
assign SDA = (pul_sda) ? 1'bz : 1'b0;
```

05 고찰

느낀점

이 프로젝트를 통해 단순한 통신 기능을 구현하는 것을 넘어, SoC 환경에서의 하드웨어-소프트웨어 통합이라는 복합적인 과제를 수행할 수 있었습니다.

I2C Read 시 Restart 문제를 겪으면서, 외부 풀업 저항과 같은 물리적 제약 조건이 HDL 로직 설계에 미치는 영향을 이해하며, 설계자가 외부 하드웨어의 물리적 제약 조건(저항, 커패시턴스)까지 고려해야 함을 깨닫는 중요한 경험이었습니다.

MicroBlaze를 통해 I2C IP를 AXI-Lite 레지스터 맵으로 연결하고 C 펌웨어를 통해 제어하는 과정을 통해, SoC 환경에서 하드웨어와 소프트웨어가 유기적으로 상호작용하여 하드웨어 IP가 어떻게 SoC의 주변장치로 기능하는지 체득할 수 있었습니다.

향후 UVM 환경을 구축하여 설계된 I2C/SPI IP에 대한 체계적이고 자동화된 검증을 수행하고 싶습니다.

감사합니다.