# SPI & I2C

# 인터페이스 설계

발표자 : 박찬호

# 목차

# 개요

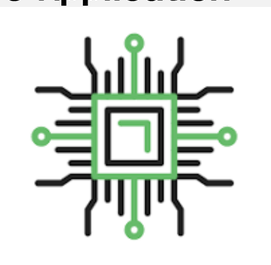
I2C 인터페이스 설계

AXI4-Lite 버스를 통해 MicroBlaze 임베디드 시스템에 통합하여 C 기반 어플리케이션 구현

SPI


I2C


C Application

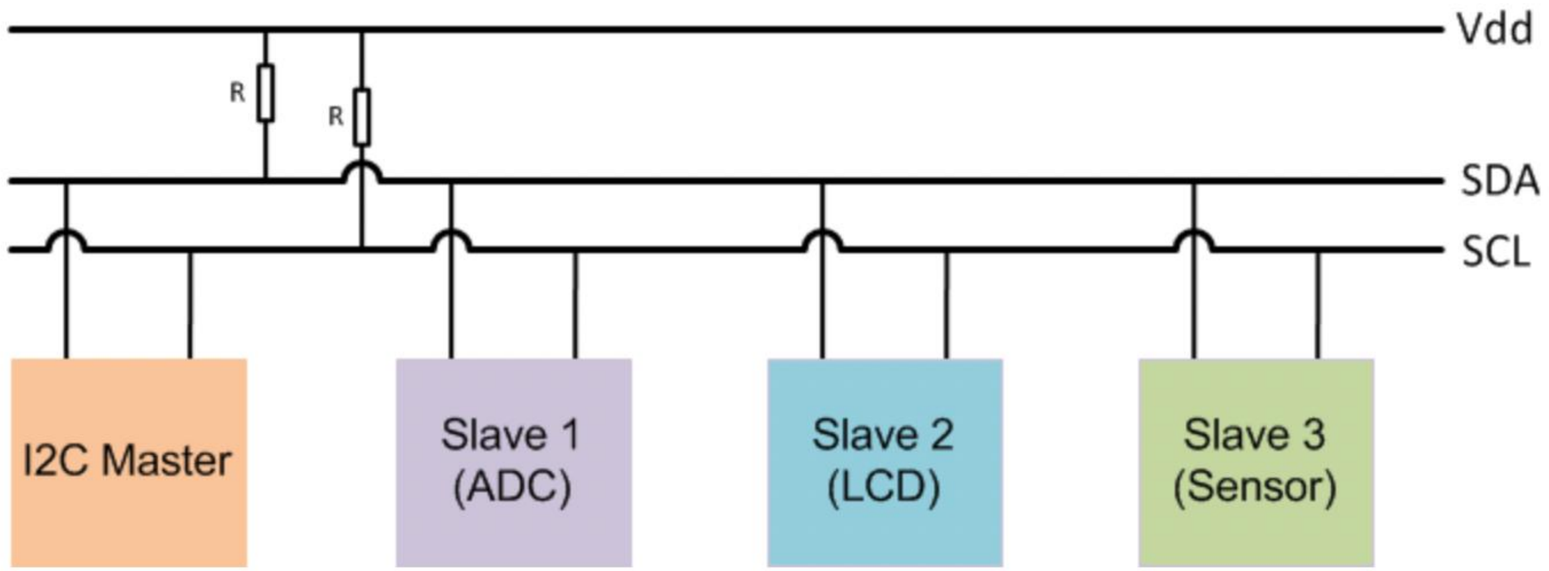


# 직렬 통신 인터페이스

| | UART | SPI | I2C |
|---|---|---|---|
| 동기 / 비동기 | 비동기 | 동기 | 동기 |
| 연결성 | 1:1 | 1:N | N:N |
| 이중 통신 | full-duplex | full-duplex | Half-duplex |
| 통신 속도 | 느리다. | 빠르다. | 느리다. |

- UART -> 비동기 1대1 통신으로 가장 단순하다.
- SPI -> 동기식, 전이중 통신이며, 빠르다.
- I2C -> 2개의 라인으로 여러 장치를 연결할 수 있다.

# I2C

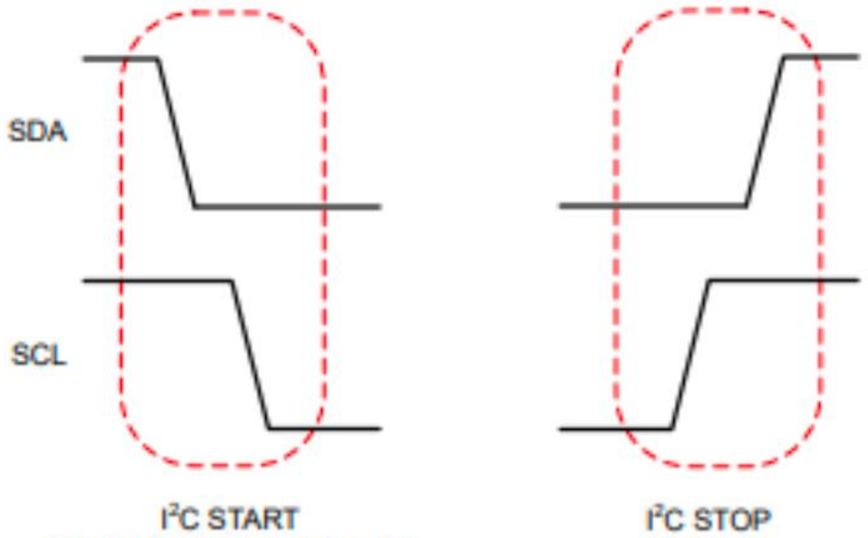



칩과 칩 사이의 짧은 거리 통신을 위한 직렬 통신 방식이다.

SCL, SDA 두 개의 라인으로 통신한다는 것이 장점이다.

SPI와 달리 CS 신호가 없기 때문에 SDA 라인을 통해 주소를 보내 Slave를 선택한다.

# I2C 프로토콜







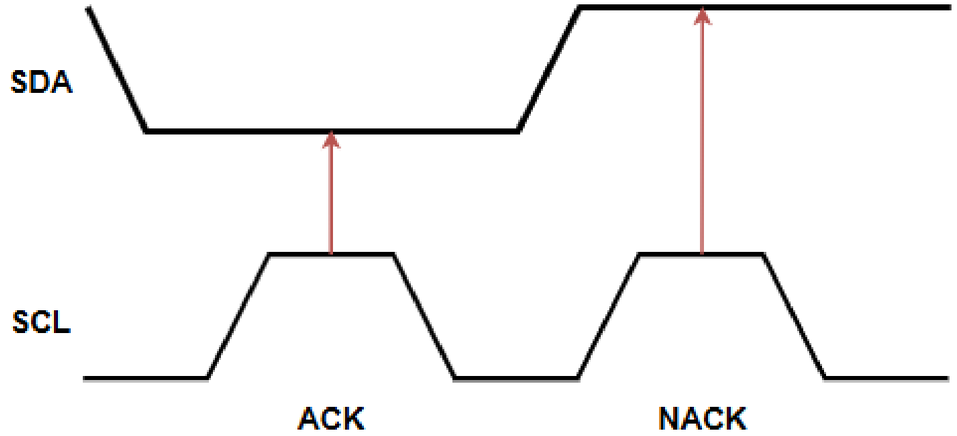
## START

SCL = 1 && SDA Falling

## STOP

SCL = 1 && SDA rising

I2C 프로토콜에서 SCL의 High 상태에서 SDA가 변할 수 있는 2가지 상태




## ACK

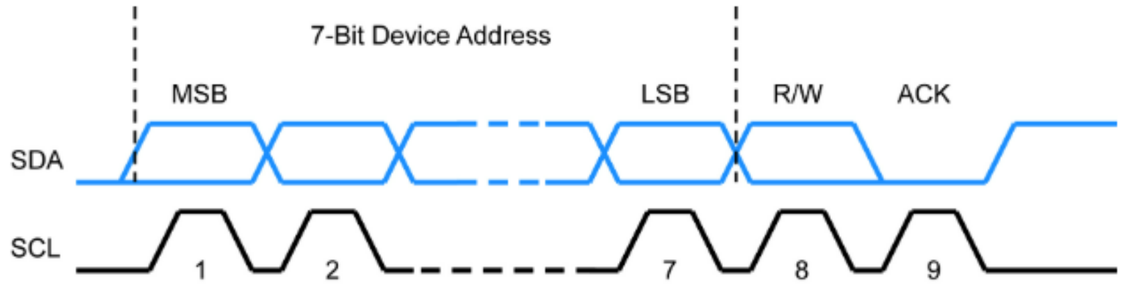
ACK : 수신 성공 신호

## NACK

NACK : 수신 거부 또는 종료 신호.

# I2C 프로토콜

**ADDRESS**



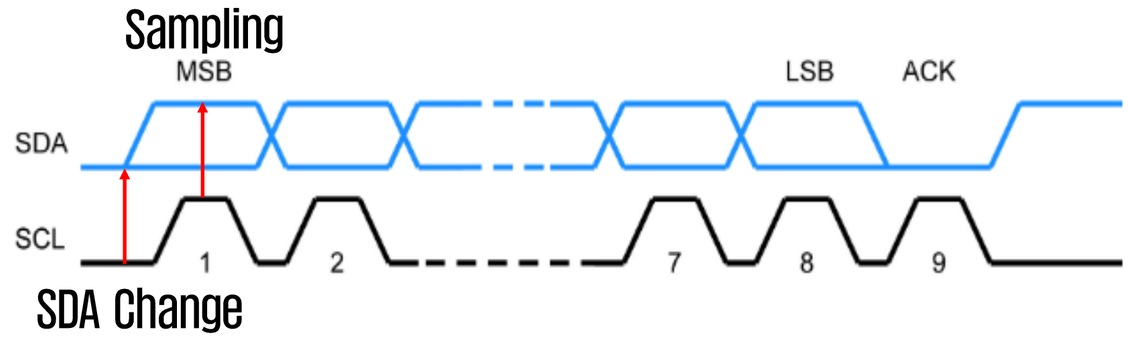

주소 프레임

주소 7 bit + R/W bit + ACK bit로 구성

1 : Read

0 : Write

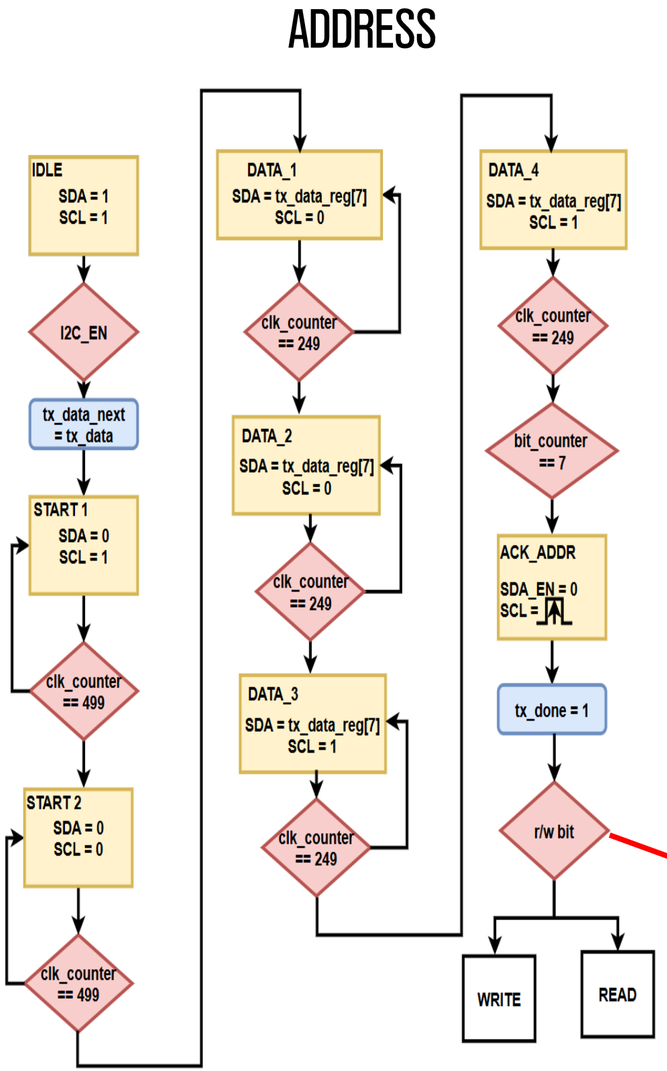
**DATA**

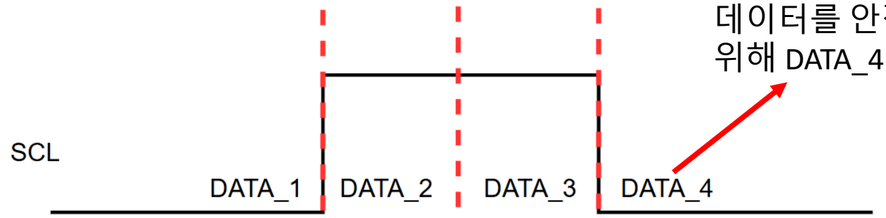



SCL LOW 구간에서 데이터를 변경하고,

SCL HIGH 구간에서 데이터를 샘플링한다.

# I2C Master ASM

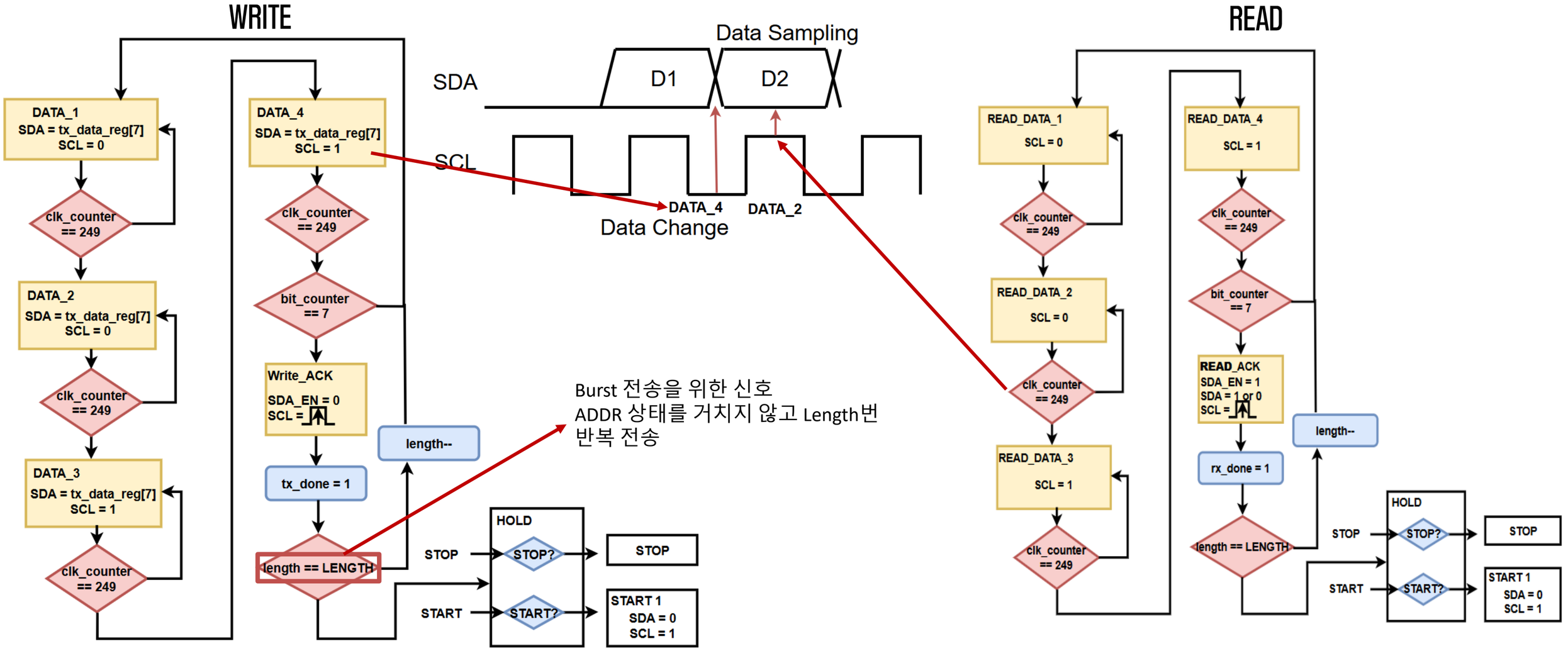



정확한 타이밍 제어를 위해 한 비트를 4개 상태로 분할

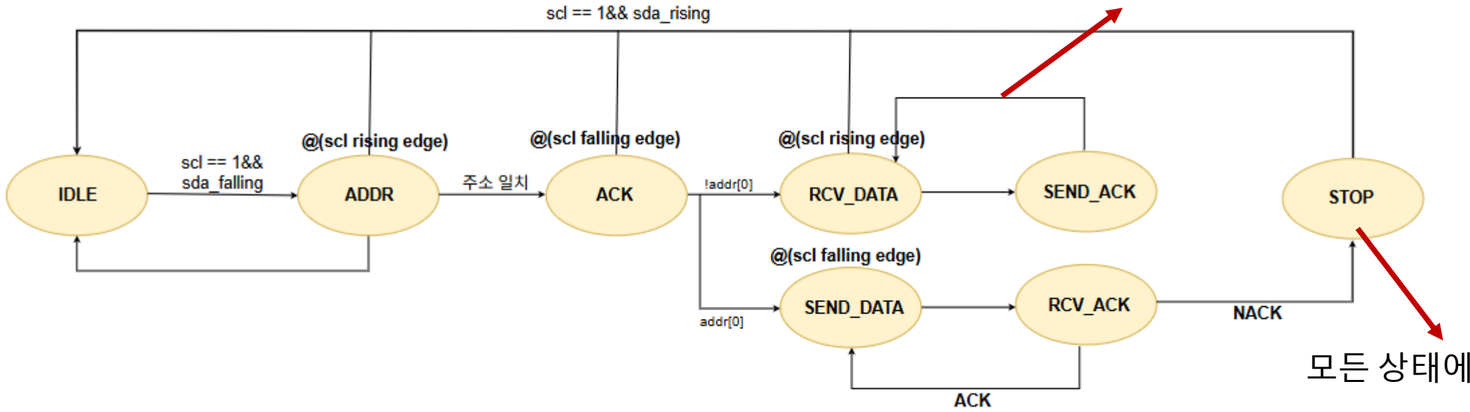
R/W bit에 따라 WRITE, READ 결정

# I2C Master FSM

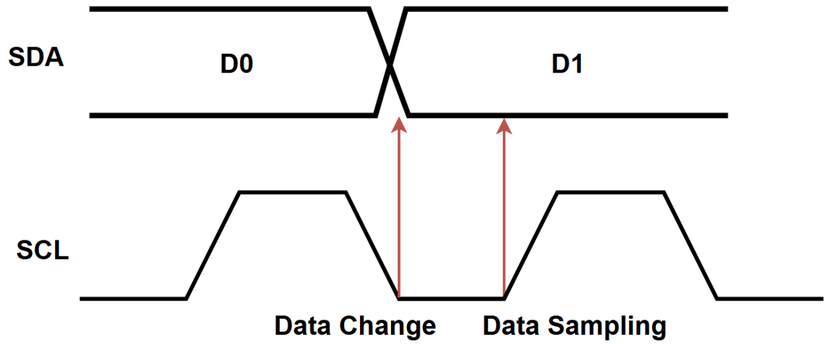


# I2C Slave FSM

Burst 전송을 위해 ACK 상태 이후 STOP이 아닌 DATA 전송 상태로 전이



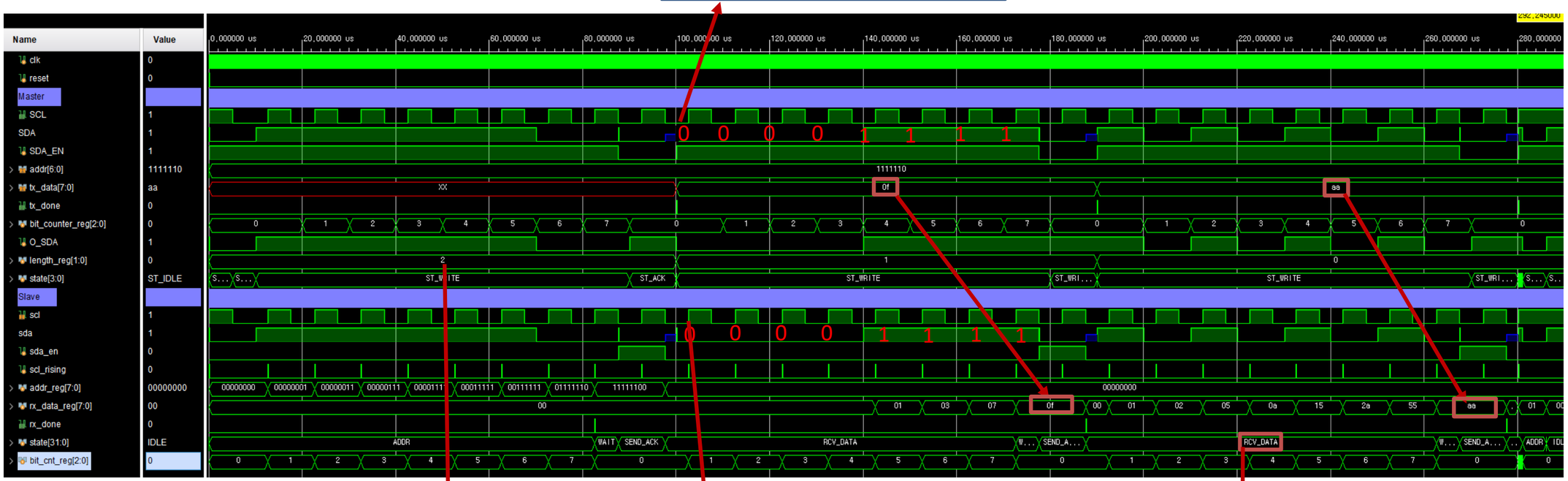

모든 상태에서 STOP 조건을 판별할 수 있도록 설계




SLAVE는 SCL의 rising edge에서 데이터를 Sampling하고 falling edge에서 데이터를 변경하도록 설계.

# I2C Write Simulation

SCL LOW 중간에서 데이터 변경



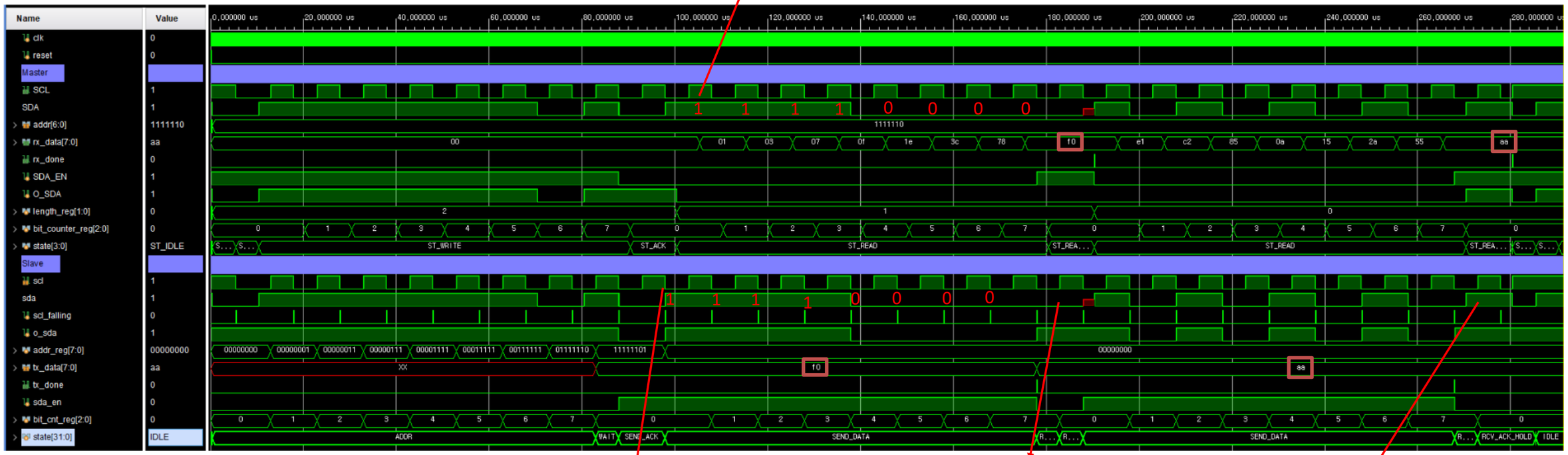

Length가 0이 될 때까지 연속적으로 데이터 전송

SCL rising edge에서 데이터 Sampling

ADDRESS 상태를 거치지 않고 다시 DATA 상태로 넘어가는 것을 볼 수 있다.

# I2C Read Simulation

SCL High 중간에서 Sampling

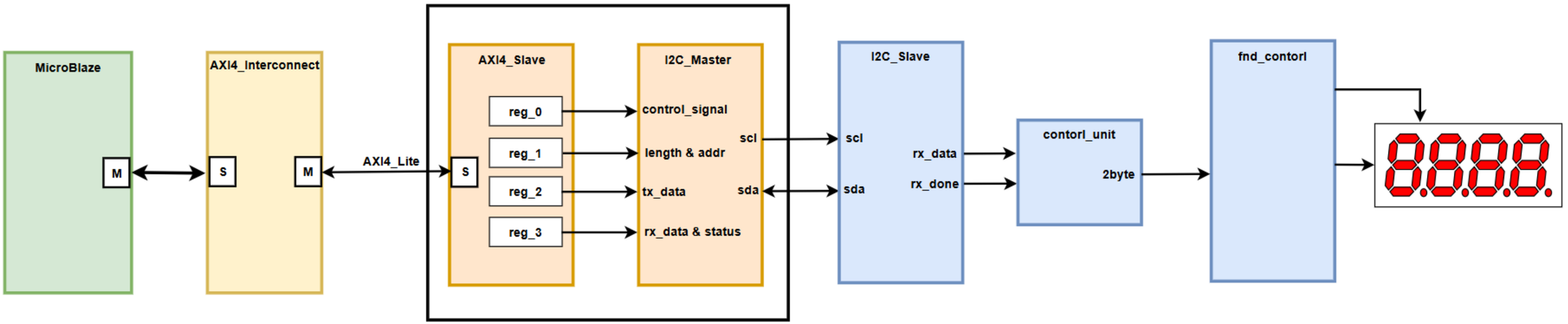



SCL falling edge에서 데이터 전송

ACK을 받았을 경우 다시 데이터 전송

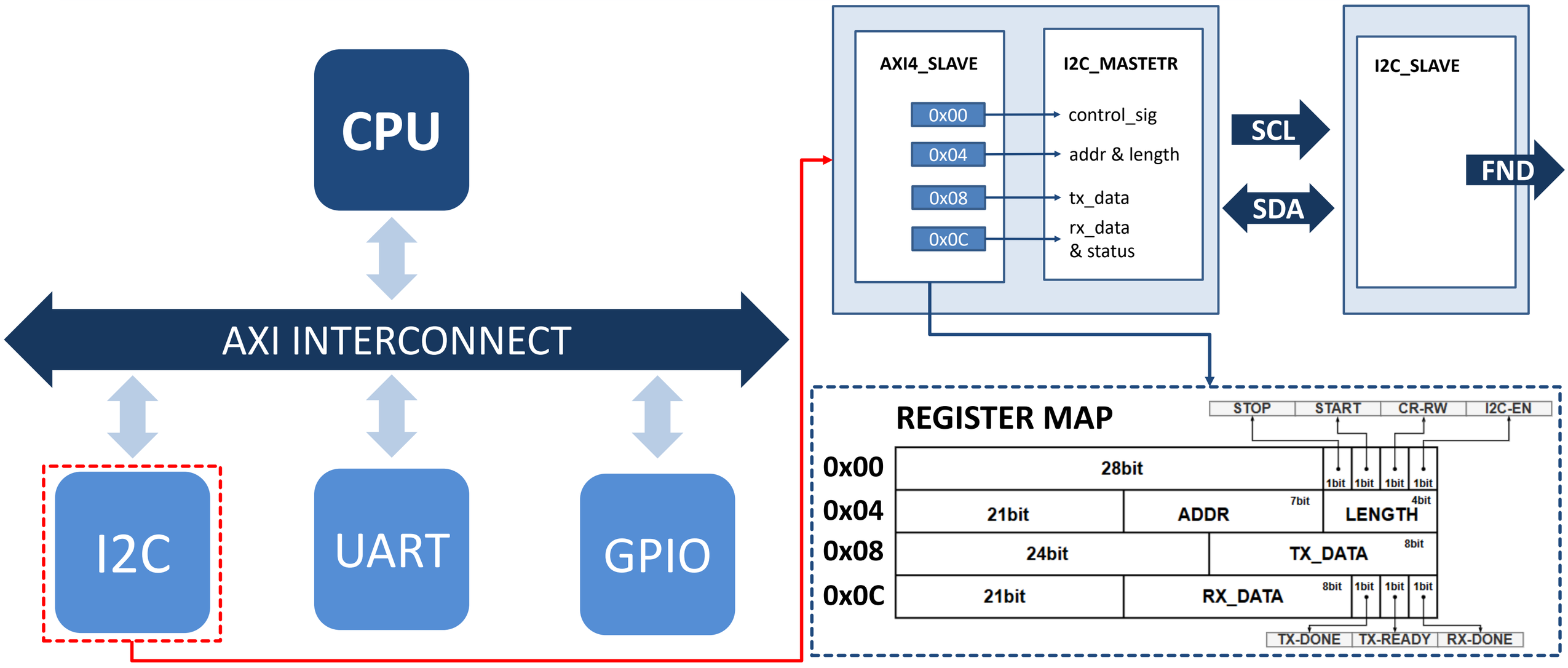
NACK을 받았을 경우 종료

# MicroBlaze 기반 I2C 통신 시스템 구조

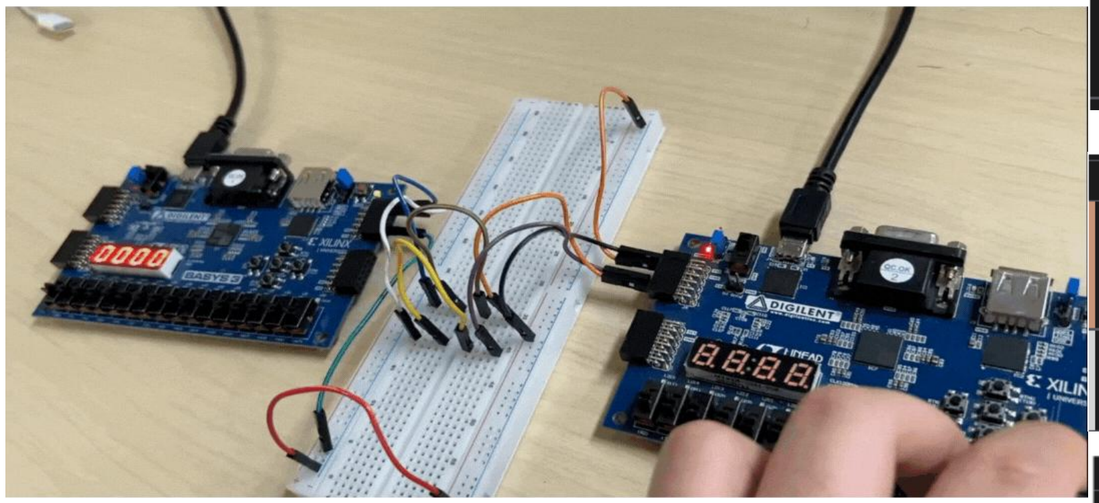


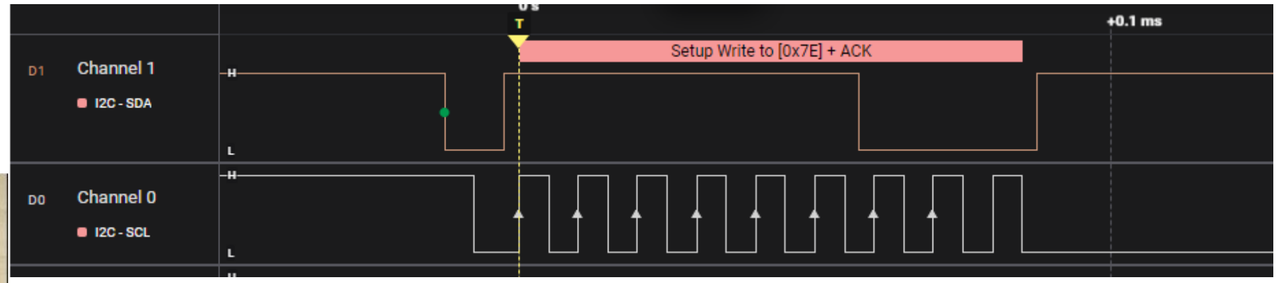

MicroBlaze에서 C 프로그램을 실행하여 업 카운터를 구현하고 I2C 통신으로 다른 보드의 FND를 제어한다.

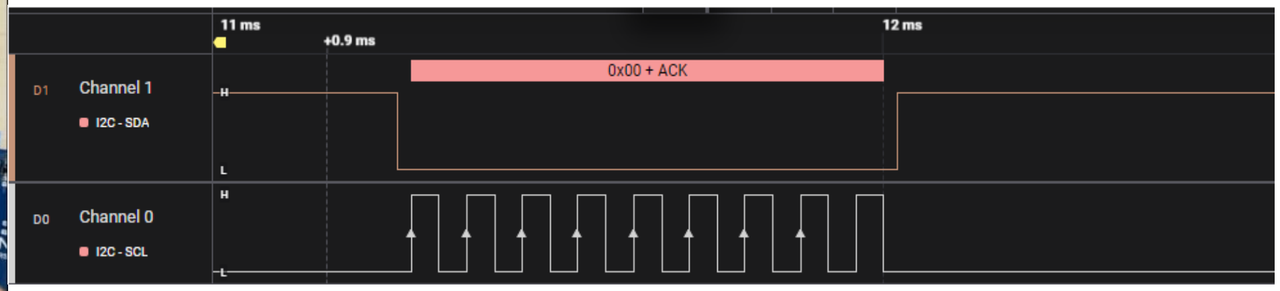
# Register Map

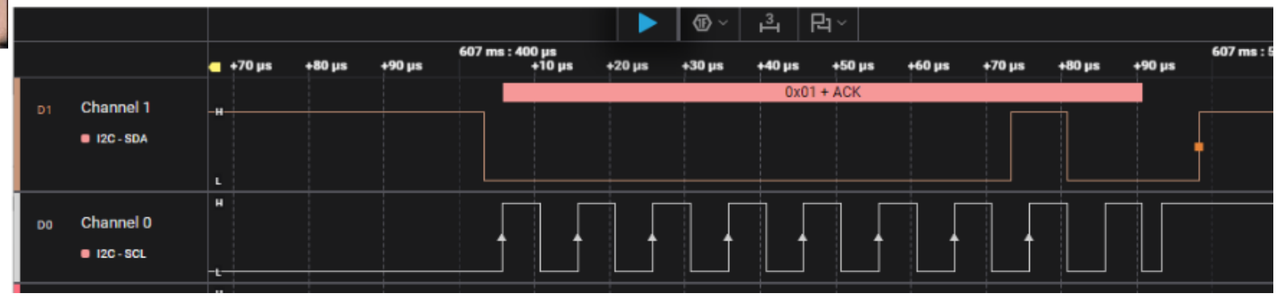


# 동작 영상



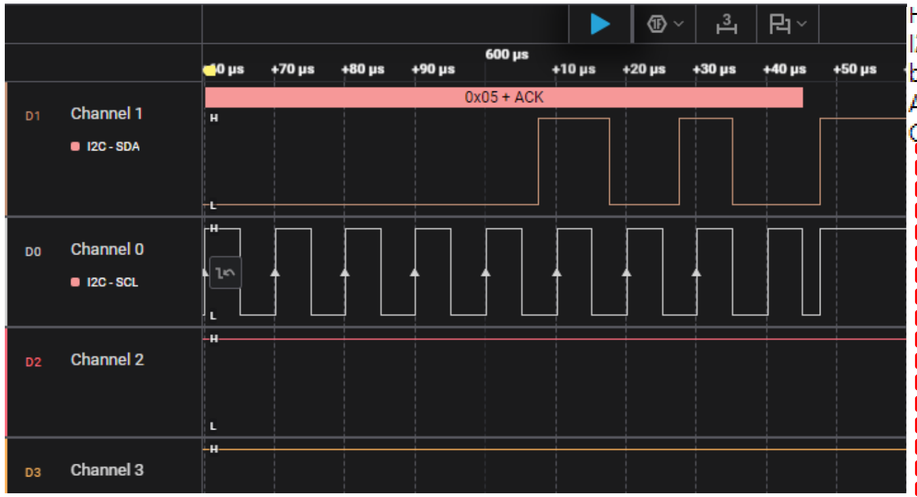

+0.1 ms
Setup Write to [0x7E] + ACK
D1
Channel 1
I2C - SDA
D0
Channel 0
I2C - SCL


11 ms
+0.9 ms
12 ms
0x00 + ACK
D1
Channel 1
I2C - SDA
D0
Channel 0
I2C - SCL


607 ms : 400 µs
+70 µs
+80 µs
+90 µs
+10 µs
+20 µs
+30 µs
+40 µs
+50 µs
+60 µs
+70 µs
+80 µs
+90 µs
0x01 + ACK
D1
Channel 1
I2C - SDA
D0
Channel 0
I2C - SCL

# TroubleShooting

- Rogic analyzer에서 ACK가 뜬 것을 확인. 즉, done을 띄운 것을 확인했지만, VITIS UART로 확인한 결과 TX_DONE을 제대로 읽지 못함.



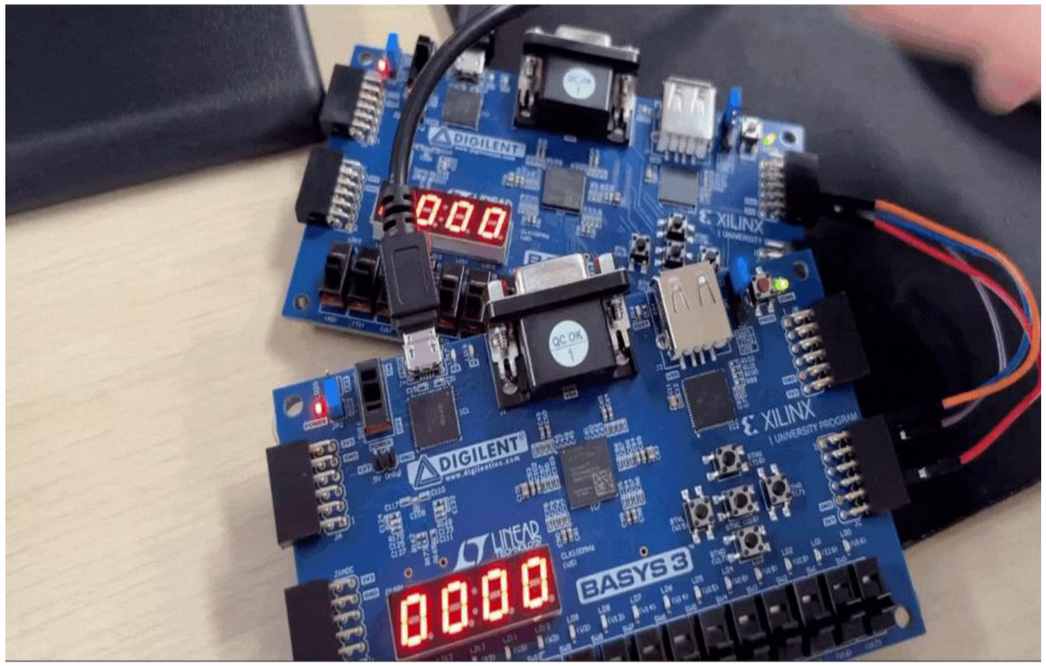

```
FND = 174
I2C FND up-counting test start
before FND_ShowNumber, count=0
ADDR REG = 0x000007E1
CR   REG = 0x00000001
[DBG] enter WaitTxDone, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
[DBG] waiting TX_DONE, status = 0x00000000
```

계속 TX_DONE을 기다림.

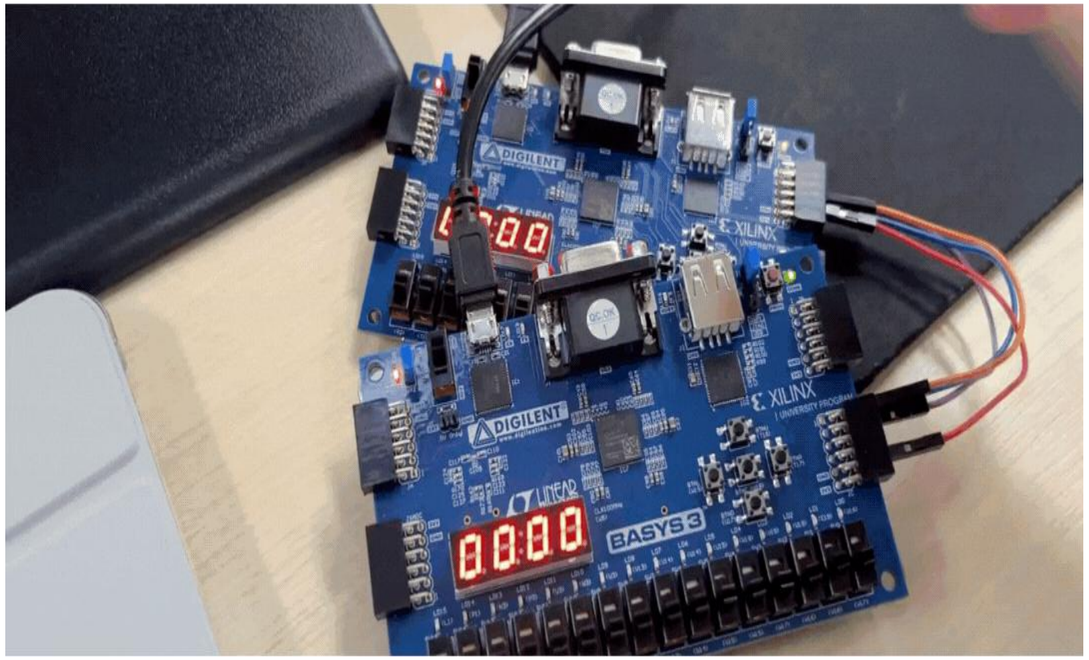
```
////////////////////////////////////////////////////////////////////
// tx_done 펄스를 sticky 플래그로 변환
always @(posedge S_AXI_ACLK) begin
    if (S_AXI_ARESETN == 1'b0) begin
        tx_done_flag <= 1'b0;
    end else begin
        // I2C_MASTER에서 1클럭짜리 tx_done 펄스 들어오면 1로 set
        if (tx_done) begin
            tx_done_flag <= 1'b1;
        end

        // CPU가 슬레이브 레지스터 3에 bit2 = 1을 쓰면 clear (write-to-clear)
        if (slv_reg_wren &&
            (axi_awaddr[ADDR_LSB+OPT_MEM_ADDR_BITS:ADDR_LSB] == 2'h3) &&
            S_AXI_WDATA[2]) begin
            tx_done_flag <= 1'b0;
        end
    end
end
////////////////////////////////////////////////////////////////////
```

펄스를 유지하기 위해 래치 회로 형태의 플래그 저장 레지스터를 추가했다.

# TroubleShooting2




Syncronizor X




Syncronizor O

클럭 주파수가 같지만 각각 독립적인 클럭 소스이므로 CDC 문제가 발생했습니다.

CDC 문제를 Syncronizor를 활용하여 해결했습니다.

# 고찰

처음에는 프로토콜만 보고 설계하는 것이 막막했는데 기본 동작 FSM부터 시작하여 기본 동작을 설계하고 Waveform을 Debugging하여 state를 수정하고 추가하면서 프로토콜을 만족시켰습니다.
기본 동작을 구현하고보니 Burst 전송이나, Repeat start 같은 확장 기능까지 구현할 수 있게 됐습니다.
다음에는 UVM을 활용하여 SPI, I2C를 검증하는 것을 목표로 하겠습니다.

# 감사합니다