AXI4-Lite 인터페이스 기반 SPI / I2C 통신 프로토콜 설계

1조: 양지훈, 오고은, 유진모, 허용석
발표자: 양지훈

00.

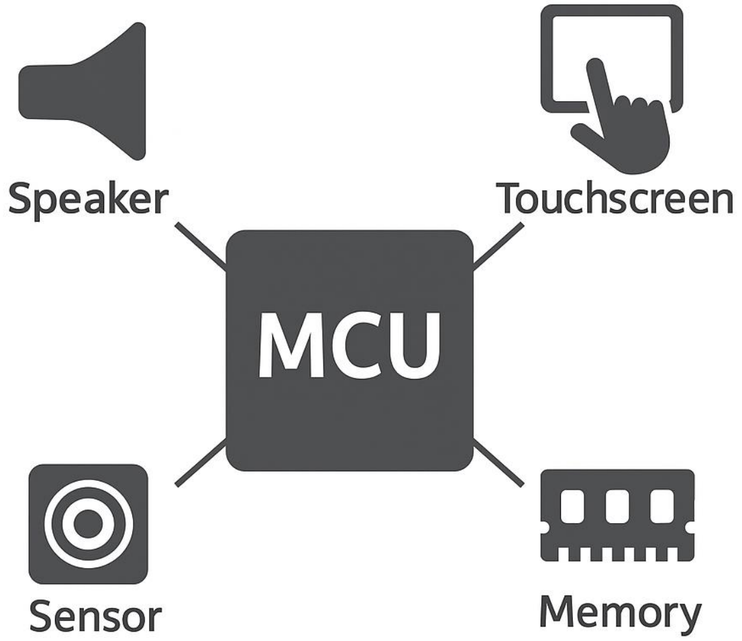
목차

01 프로젝트 개요

01.

개요

MCU는 각종 장치와의 통신을 통해 실시간 데이터를 수집 및 제어 신호를 전달



01.

개요

AXI4-Lite를 활용한 통신 프로토콜 구현

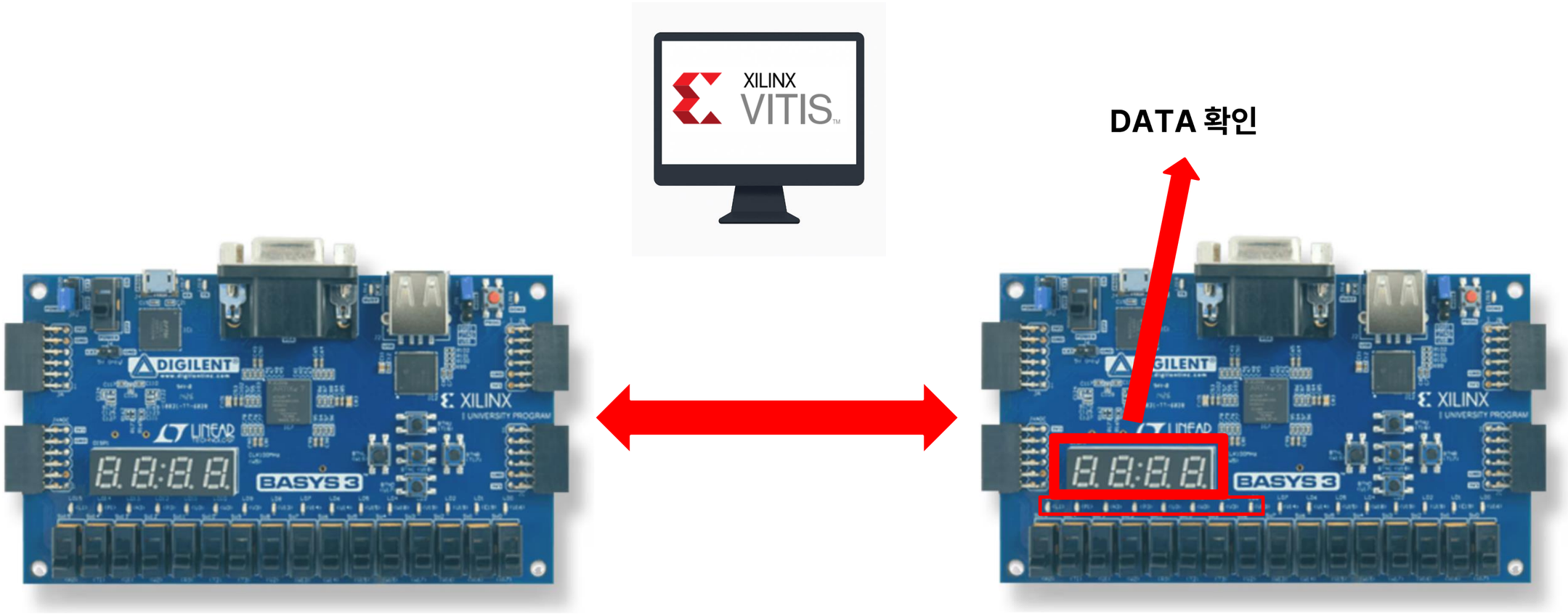
SPI 통신 구현 및 데이터 확인 – Baysis 3 보드, Verilog 및 C 언어 기반의 송/수신 데이터 검증

I2C 통신 파형 검증 – Verilog를 활용한 I2C 구현 및 시뮬레이션 검증

SPI MASTER 검증 – UVM을 활용한 검증 시스템 구축

01.

개요


XILINX
VITIS
DATA 확인
DIGILENT
XILINX
UNIVERSITY PROGRAM
LINEAR TECHNOLOGY
BASYS 3
MASTER
SLAVE

02 개발 환경

02.

개발 환경

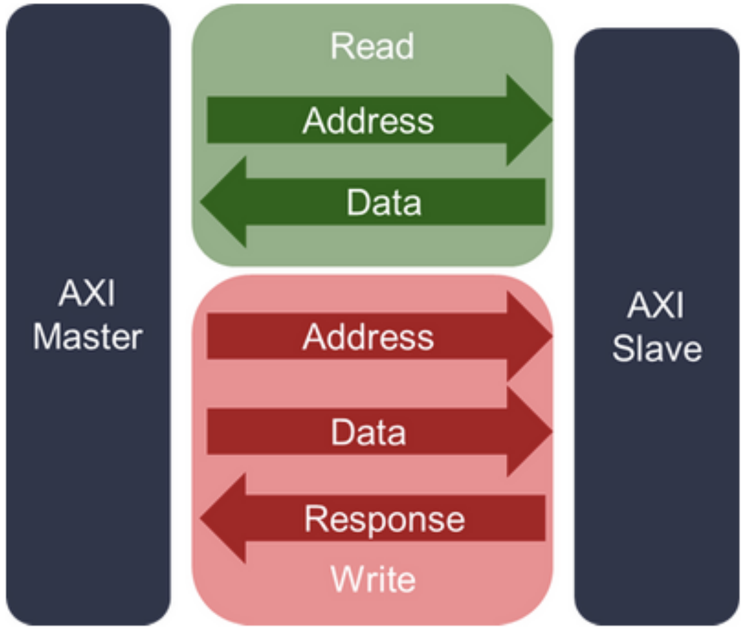
01. core	**Microblaze**
02. Hardware Board	**Basys3**
03. Tool / Language	**Vivado 2020.2, Vitis, Synopsys VCS / Verilog, C, System Verilog**
04. Protocol	**AMBA AXI4 Lite**
05. Slave	**I2C, SPI**

03 시스템 구조

03.

시스템 구조

AXI4-Lite



AXI4-Lite는 총 5개의 채널로 구성

각 채널은 VALID / READY 신호를 이용한
Handshake 프로토콜로 동작

- AW (Address Write)
- W (Write Data)
- B (Write Response)
- AR (Address Read)
- R (Read Data)

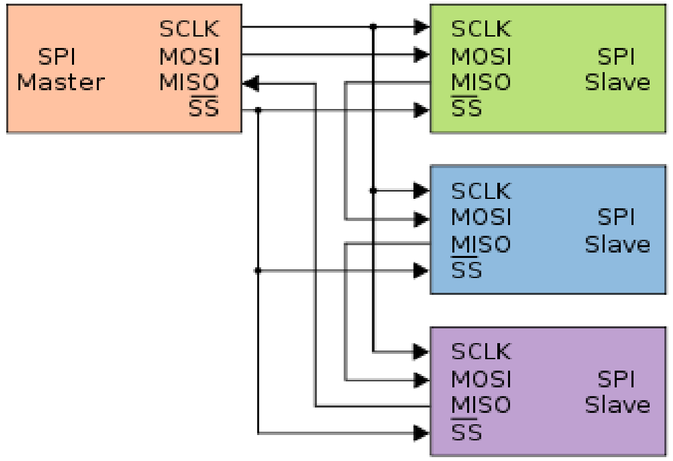
Write: AW → W → B
Read: AR → R

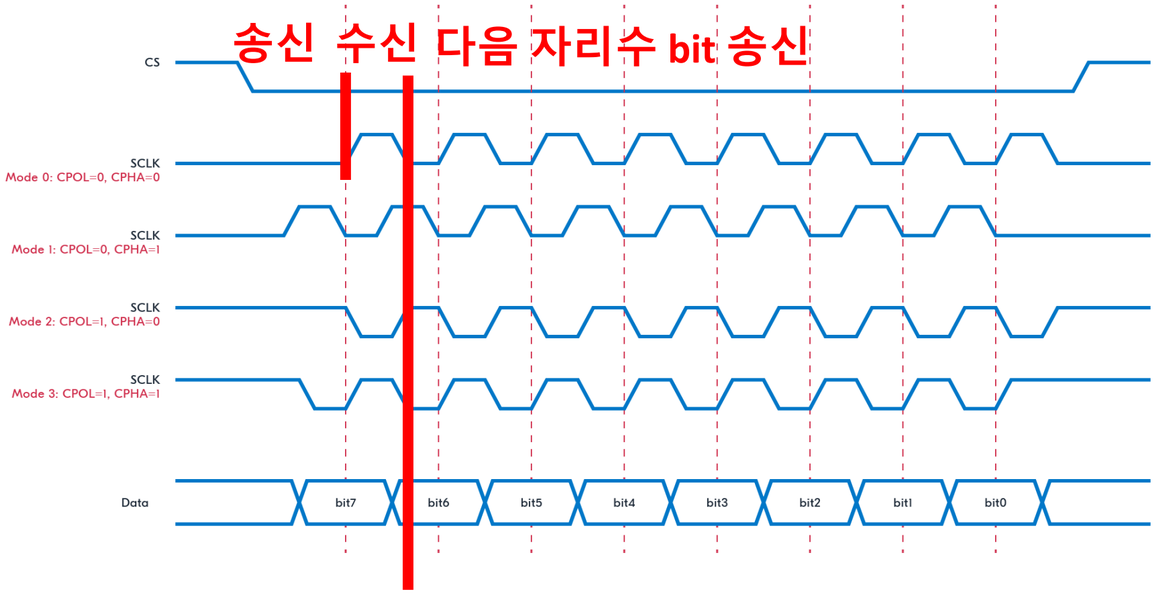
Point-to-Point 인터페이스
→ 각 Master와 Slave는 1:1 연결
→ Broadcasting ✕
→ Bus 형태 ✕

03.

시스템 구조

SPI





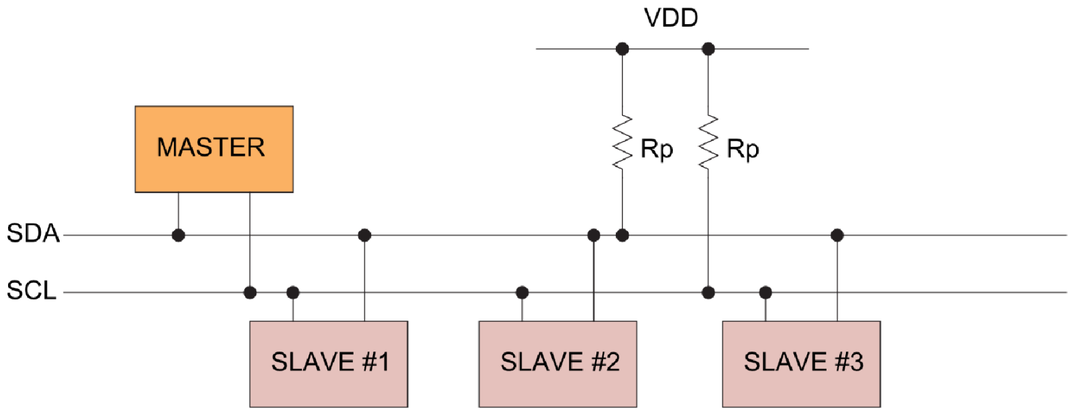
MASTER – SLVAE 구조의 동기식 통신 방식

1. 송/수신 장치 간의 동기화 필요
2. 송/수신이 동시에 가능
3. 한 개의 마스터 장치로 여러 슬레이브 제어
   (Slave Select 방식)
4. I2C에 비해 통신 속도 빠름

SCLK : Serial Clock (Master가 생성)
MOSI : Master Out Slave In
MISO : Master In Slave Out
SS : Slave Select (Active Low)

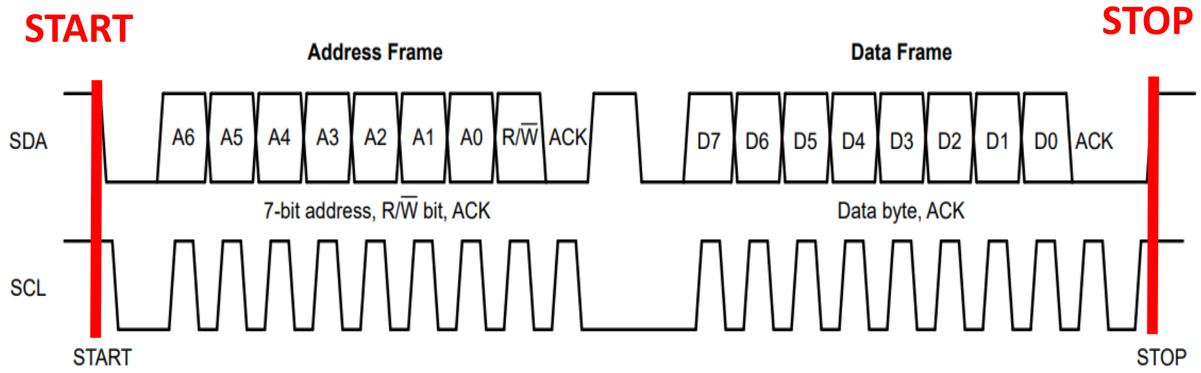
03.

시스템 구조



단일 데이터 라인으로 Master – Slave 양방향 통신

1. 2개의 선으로 다수의 master, slave 통신
2. Master에서 SCLK 생성
3. 하나의 하드웨어, 하나의 고유 Address

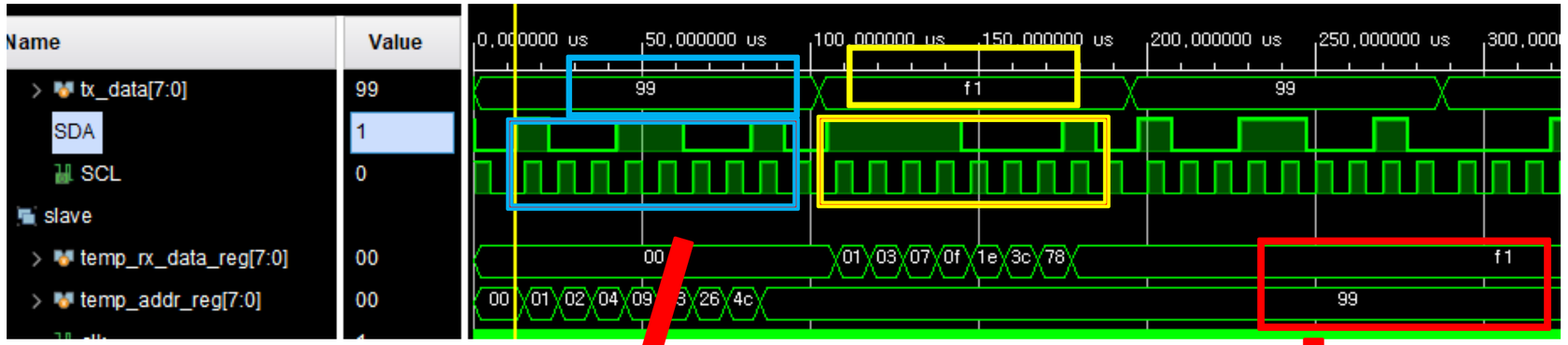


start: SCL 신호는 그대로(high), SDA 1 →0

stop: SCL 신호는 그대로(high), SDA 0 →1

SDA =0 && SCL =0 : 데이터 전송 준비

04 상세 구현



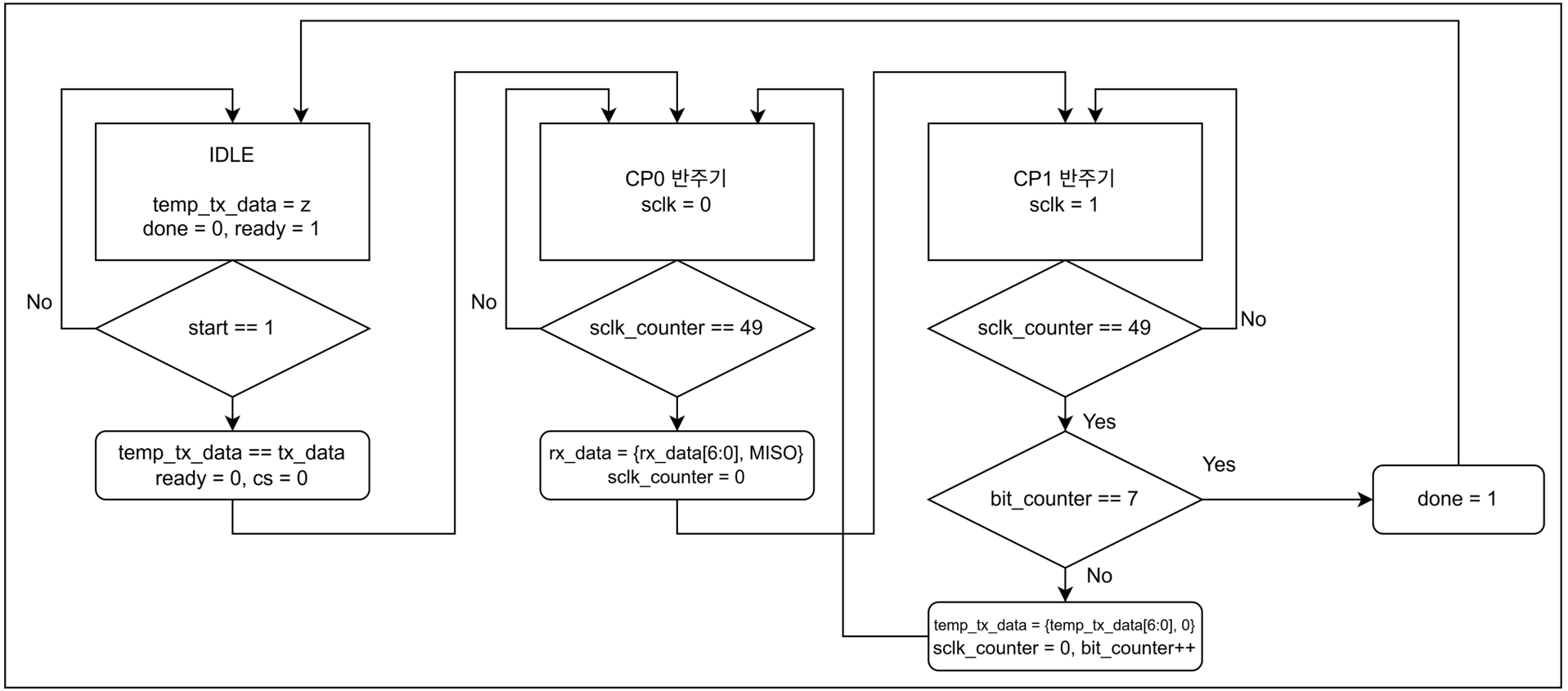
0x99에서 lsb를 뺀 7'b1001100 주소에 0xf1 값 Write

I2C Slave에 값 저장

04.

상세 구현

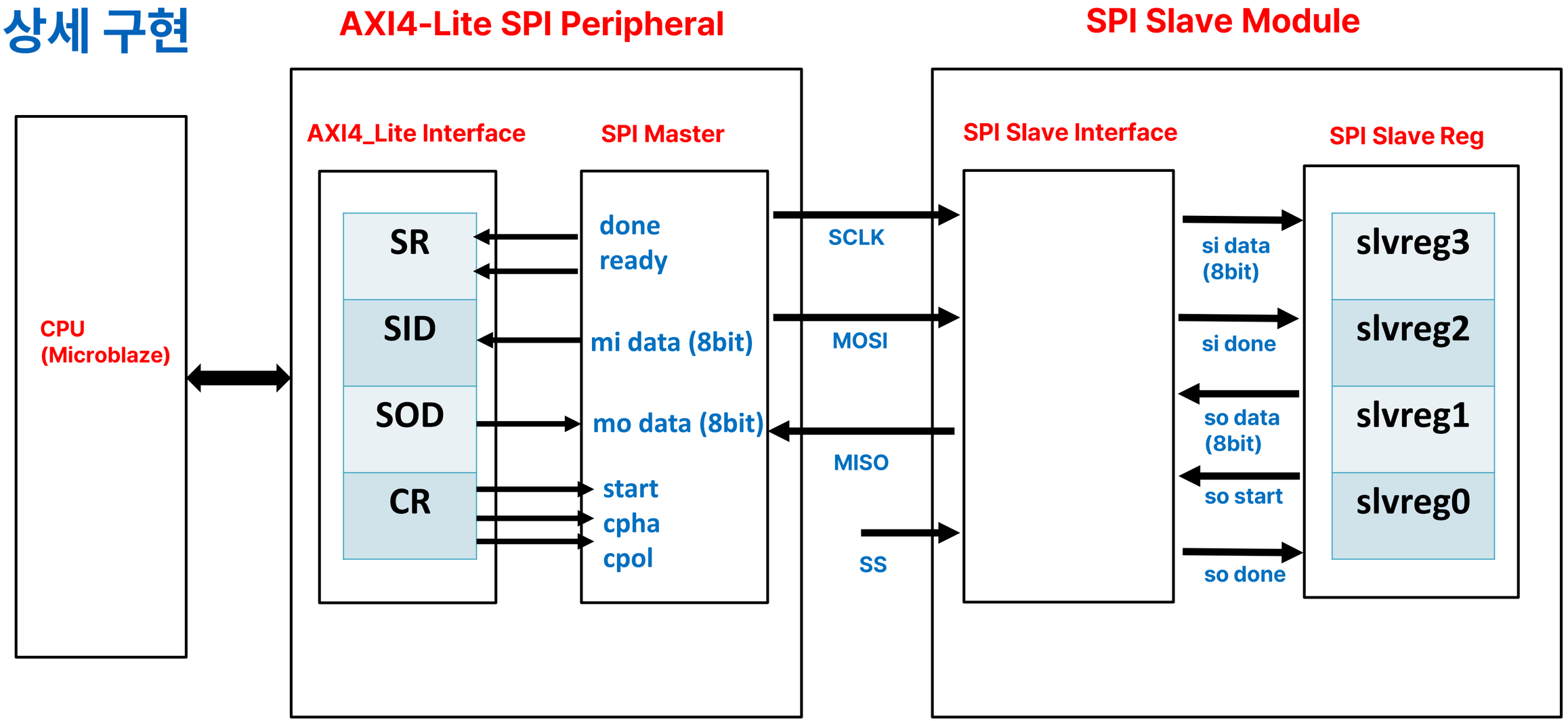
SPI


IDLE
temp_tx_data = z
done = 0, ready = 1
No
start == 1
temp_tx_data == tx_data
ready = 0, cs = 0
CP0 반주기
sclk = 0
No
sclk_counter == 49
rx_data = {rx_data[6:0], MISO}
sclk_counter = 0
CP1 반주기
sclk = 1
sclk_counter == 49
No
Yes
bit_counter == 7
Yes
done = 1
No
temp_tx_data = {temp_tx_data[6:0], 0}
sclk_counter = 0, bit_counter++

04.

상세 구현

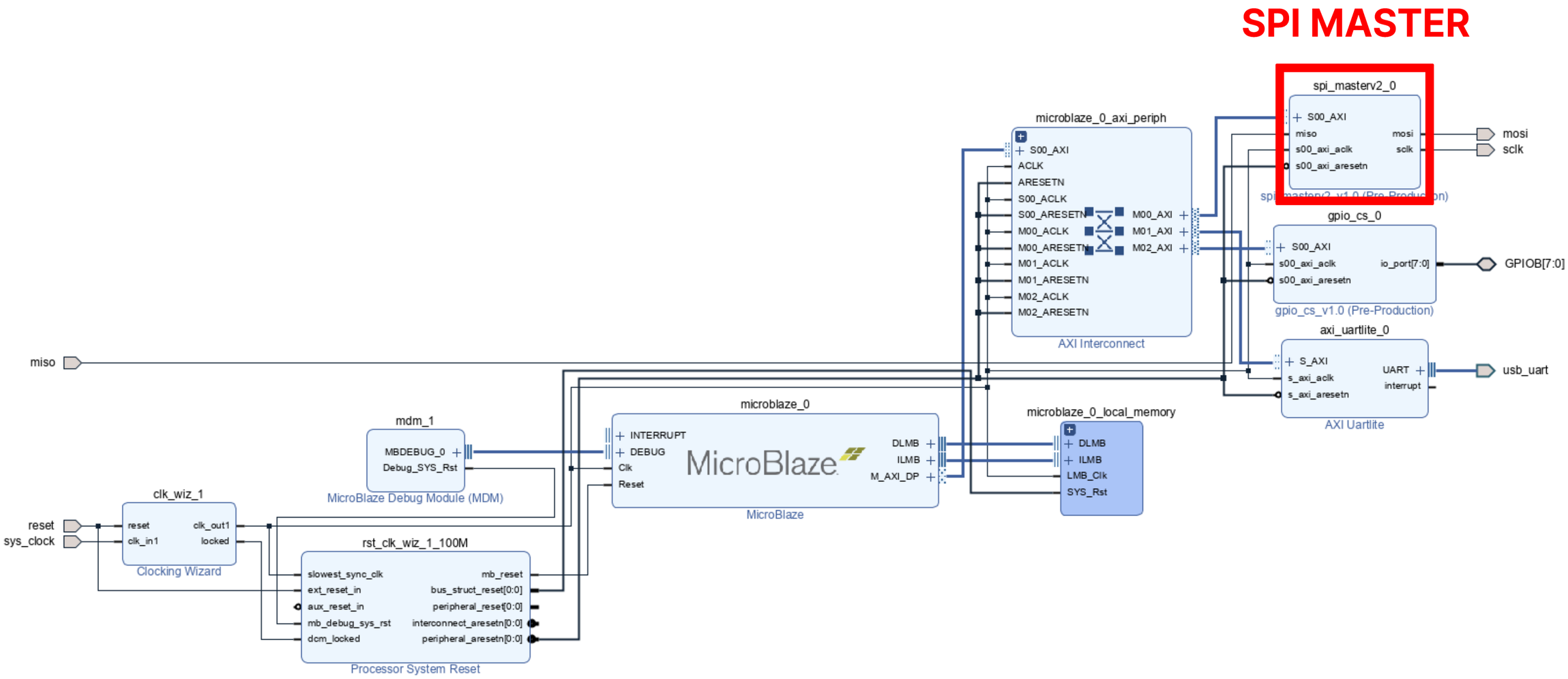
SPI


AXI4-Lite SPI Peripheral
SPI Slave Module
CPU
(Microblaze)
AXI4_Lite Interface
SPI Master
SPI Slave Interface
SPI Slave Reg
SR
SID
SOD
CR
done
ready
mi data (8bit)
mo data (8bit)
start
cpha
cpol
SCLK
MOSI
MISO
SS
si data
(8bit)
si done
so data
(8bit)
so start
so done
slvreg3
slvreg2
slvreg1
slvreg0

04.

상세 구현

SPI

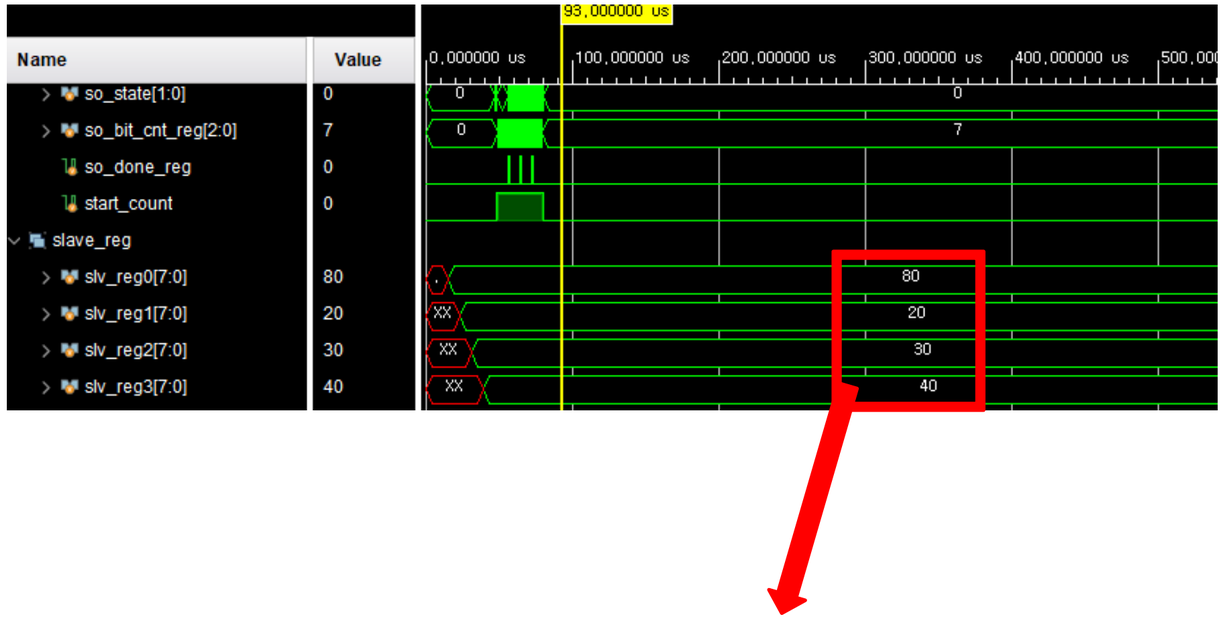

SPI MASTER
spi_masterv2_0
S00_AXI
miso
s00_axi_aclk
s00_axi_aresetn
mosi
sclk
mosi
sclk
microblaze_0_axi_periph
S00_AXI
ACLK
ARESETN
S00_ACLK
S00_ARESETN
M00_ACLK
M00_ARESETN
M01_ACLK
M01_ARESETN
M02_ACLK
M02_ARESETN
M00_AXI
M01_AXI
M02_AXI
AXI Interconnect
gpio_cs_0
S00_AXI
s00_axi_aclk
s00_axi_aresetn
io_port[7:0]
GPIOB[7:0]
gpio_cs_v1.0 (Pre-Production)
axi_uartlite_0
S_AXI
s_axi_aclk
s_axi_aresetn
UART
interrupt
usb_uart
AXI Uartlite
miso
mdm_1
MBDEBUG_0
Debug_SYS_Rst
MicroBlaze Debug Module (MDM)
microblaze_0
INTERRUPT
DEBUG
Clk
Reset
MicroBlaze
DLMB
ILMB
M_AXI_DP
microblaze_0_local_memory
DLMB
ILMB
LMB_Clk
SYS_Rst
clk_wiz_1
reset
clk_in1
clk_out1
locked
Clocking Wizard
reset
sys_clock
rst_clk_wiz_1_100M
slowest_sync_clk
ext_reset_in
aux_reset_in
mb_debug_sys_rst
dcm_locked
mb_reset
bus_struct_reset[0:0]
peripheral_reset[0:0]
interconnect_aresetn[0:0]
peripheral_aresetn[0:0]
Processor System Reset

04.

상세 구현

SPI

```
SS = 1;
@(posedge clk);
tx_data = 8'b1000_0000; start = 1; cpol = 0; cpha = 0; S
@(posedge clk);
start = 0;
wait (done == 1);
@(posedge clk);

// 0x00 address에 write data byte
@(posedge clk);
tx_data = 8'h80; start = 1; cpol = 0; cpha = 0; SS = 0;
@(posedge clk);
start = 0;
wait (done == 1);
@(posedge clk);

// 0x01 address에 write data byte
@(posedge clk);
tx_data = 8'h20; start = 1; cpol = 0; cpha = 0; SS = 0;
@(posedge clk);
start = 0;
wait (done == 1);
@(posedge clk);

// 0x02 address에 write data byte
@(posedge clk);
tx_data = 8'h30; start = 1; cpol = 0; cpha = 0; SS = 0;
@(posedge clk);
start = 0;
wait (done == 1);
@(posedge clk);

// 0x03 address에 write data byte
@(posedge clk);
tx_data = 8'h40; start = 1; cpol = 0; cpha = 0; SS = 0;
@(posedge clk);
```



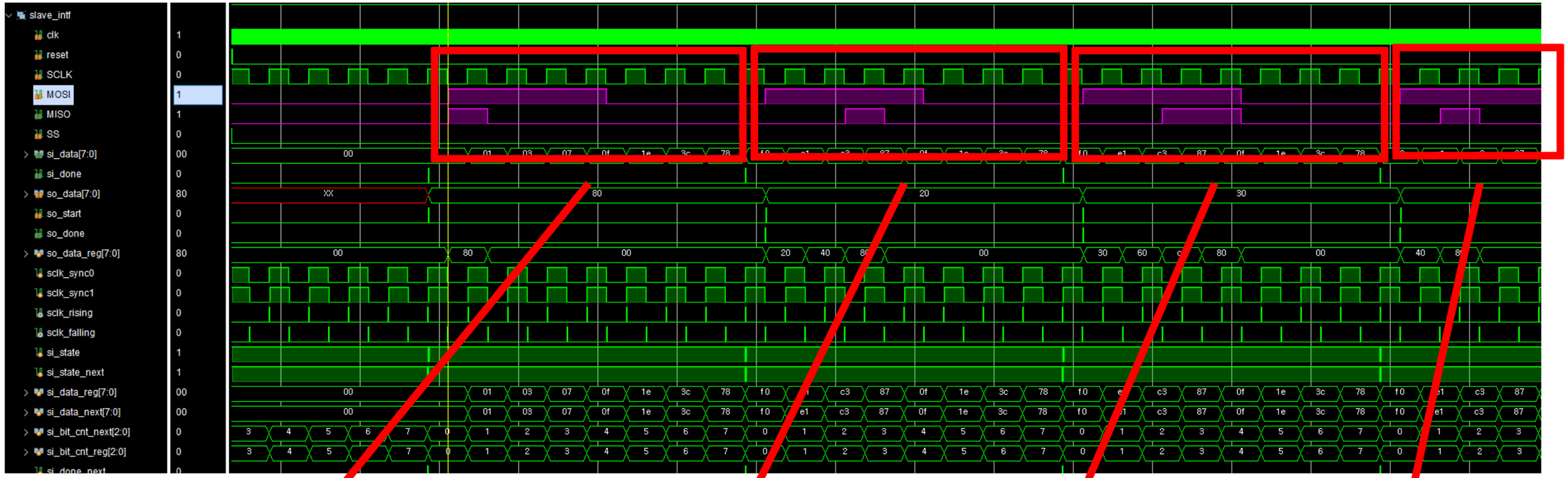
SPI Slave reg에 값 Write

slv_reg0 = 0x80
slv_reg1 = 0x20
slv_reg2 = 0x30
slv_reg3 = 0x40

04.

상세 구현

SPI



Mosi 8번 => 0xf0 (dummy)
Miso 8번 => 0x80

Mosi 8번 => 0xf0
Miso 8번 => 0x20

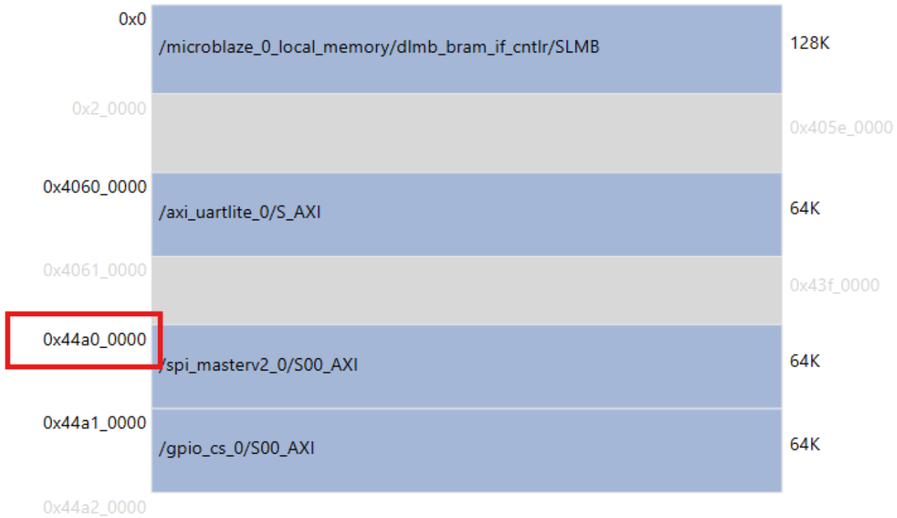
Mosi 8번 => 0xf0
Miso 8번 => 0x30

Mosi 8번 => 0xf0
Miso 8번 => 0x40

04.

상세 구현

SPI



SPI Master Memory Mapping
: 0x44a0_0000

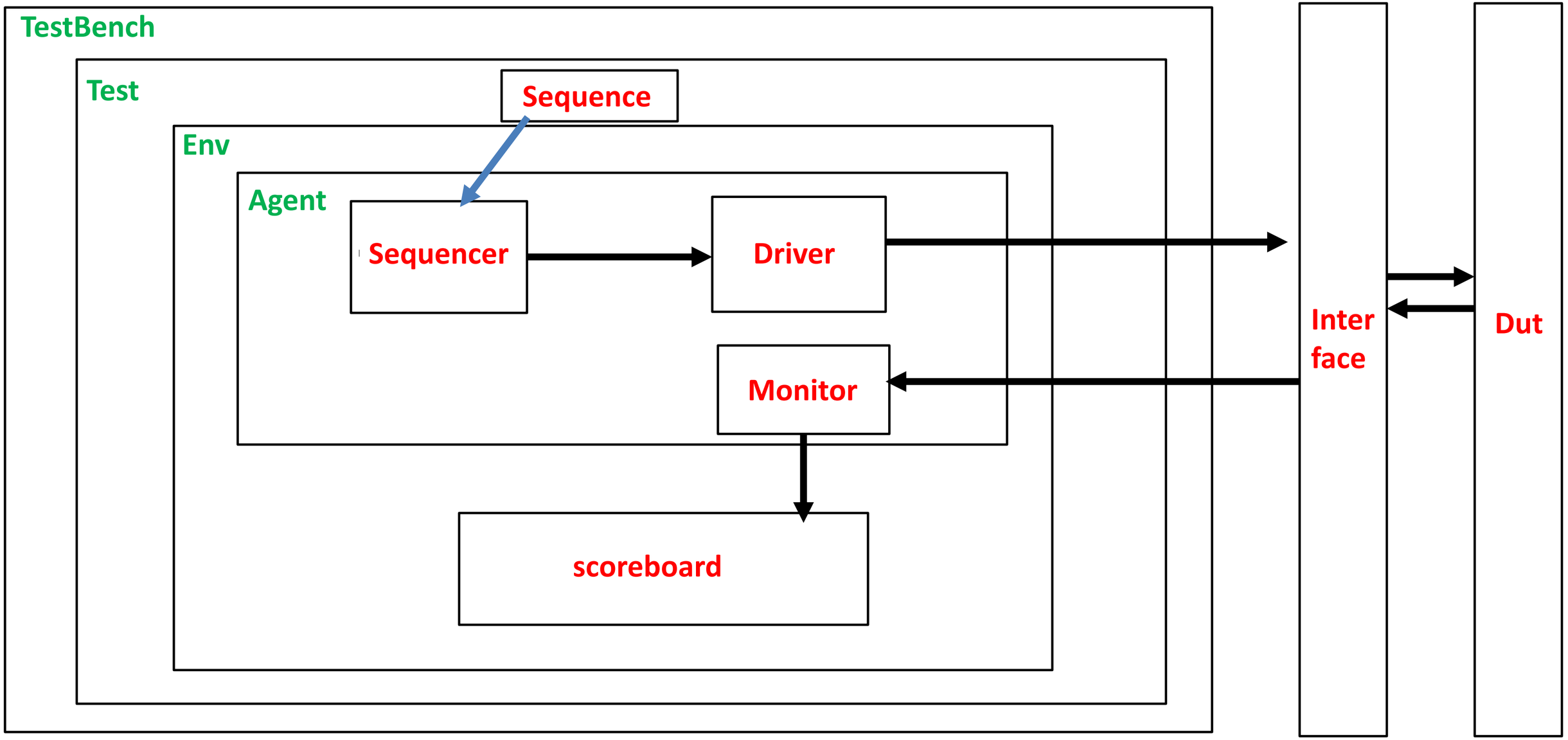
```c
void spi_send(SPI_MASTER_TypeDef *SPIx, uint32_t data) {
    SPIx->SOD = data;
    SPIx->CR = 0x04;
    while (!(SPIx->SR & (1 << 1))); // wait done
    SPIx->CR = 0x00;
    while (!(SPIx->SR & (1 << 0))); // wait ready
}

uint8_t spi_recv(SPI_MASTER_TypeDef *SPIx, uint32_t dummy) {
    SPIx->SOD = dummy;
    SPIx->CR = 0x04;
    while (!(SPIx->SR & (1 << 1)));
    uint32_t val = SPIx->SID;
    SPIx->CR = 0x00;
    return val;
}
```

AXI interface의 register를 활용

done, ready 신호로 유효한 데이터 송/수신을 대기하고,
이에 맞춰 start 신호 조절로 통신 타이밍 제어

05 검증

UVM


TestBench
Test
Sequence
Env
Agent
Sequencer
Driver
Monitor
scoreboard
Inter face
Dut

05.

검증

UVM 검증

SPI Master의 tx_data 랜덤 생성, reference model의 tx data와 DUT의 rx data를 비교 및 검증

```
virtual task run_phase(uvm_phase phase);
    forever begin
        seq_item_port.get_next_item(item);

        wait (spi_vif.ready == 1);

        @(posedge spi_vif.clk);
        spi_vif.tx_data <= item.tx_data;
        spi_vif.start   <= 1;

        @(posedge spi_vif.clk);
        spi_vif.start <= 0;

        wait (spi_vif.done == 1);
        @(posedge spi_vif.clk);
       // item.rx_data = spi_vif.rx_data;

        `uvm_info("DRV", $sformatf("TX: %0h", item.tx_data), UVM_LOW)
        seq_item_port.item_done();
    end
endtask
```

<driver>

```
virtual task run_phase(uvm_phase phase);
    forever begin
        @(posedge spi_vif.done);
        #1;
        @(posedge spi_vif.clk)
        item = spi_seq_item::type_id::create("SPI_ITEM", this);
        item.tx_data = spi_vif.tx_data;
        item.rx_data = spi_vif.rx_data;

        `uvm_info("MON", $sformatf("TX: %0x, RX: %0x", item.tx_da
        ap.write(item);
    end
endtask
```

<monitor>

done과 ready 신호를 활용한 데이터 타이밍 조절

05.

검증

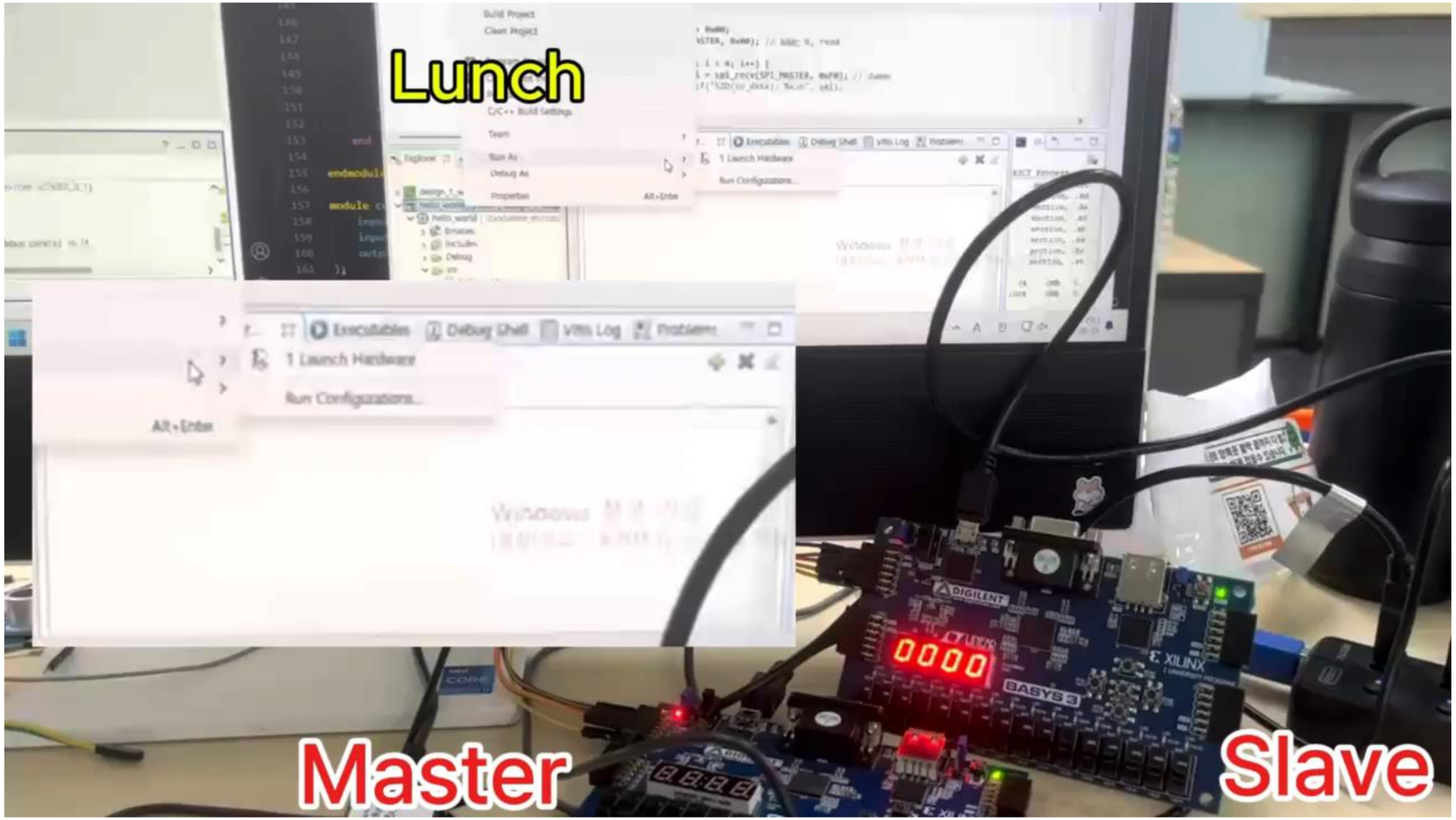
UVM 검증

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :  505
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[DRV]   100
[MON]   100
[RNTST]     1
[SCO]   200
[SEQ]   100
[TEST_DONE]     1
[UVM/RELNOTES]     1
[UVM/REPORT/CATCHER]     1
[UVMTOP]     1
```

```
UVM_INFO ./tb/tb_spi_master.sv(53) @ 737835000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: fa
UVM_INFO ./tb/tb_spi_master.sv(96) @ 745855000: uvm_test_top.ENV.AGT.DRV [DRV] TX: fa
UVM_INFO ./tb/tb_spi_master.sv(136) @ 745855000: uvm_test_top.ENV.AGT.MON [MON] TX: fa, RX: fa
UVM_INFO ./tb/tb_spi_master.sv(162) @ 745855000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: fa -> RX: fa
UVM_INFO ./tb/tb_spi_master.sv(165) @ 745855000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 745855000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: ab
UVM_INFO ./tb/tb_spi_master.sv(96) @ 753875000: uvm_test_top.ENV.AGT.DRV [DRV] TX: ab
UVM_INFO ./tb/tb_spi_master.sv(136) @ 753875000: uvm_test_top.ENV.AGT.MON [MON] TX: ab, RX: ab
UVM_INFO ./tb/tb_spi_master.sv(162) @ 753875000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: ab -> RX: ab
UVM_INFO ./tb/tb_spi_master.sv(165) @ 753875000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 753875000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: 9d
UVM_INFO ./tb/tb_spi_master.sv(96) @ 761895000: uvm_test_top.ENV.AGT.DRV [DRV] TX: 9d
UVM_INFO ./tb/tb_spi_master.sv(136) @ 761895000: uvm_test_top.ENV.AGT.MON [MON] TX: 9d, RX: 9d
UVM_INFO ./tb/tb_spi_master.sv(162) @ 761895000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: 9d -> RX: 9d
UVM_INFO ./tb/tb_spi_master.sv(165) @ 761895000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 761895000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: 55
UVM_INFO ./tb/tb_spi_master.sv(96) @ 769915000: uvm_test_top.ENV.AGT.DRV [DRV] TX: 55
UVM_INFO ./tb/tb_spi_master.sv(136) @ 769915000: uvm_test_top.ENV.AGT.MON [MON] TX: 55, RX: 55
UVM_INFO ./tb/tb_spi_master.sv(162) @ 769915000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: 55 -> RX: 55
UVM_INFO ./tb/tb_spi_master.sv(165) @ 769915000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 769915000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: 65
UVM_INFO ./tb/tb_spi_master.sv(96) @ 777935000: uvm_test_top.ENV.AGT.DRV [DRV] TX: 65
UVM_INFO ./tb/tb_spi_master.sv(136) @ 777935000: uvm_test_top.ENV.AGT.MON [MON] TX: 65, RX: 65
UVM_INFO ./tb/tb_spi_master.sv(162) @ 777935000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: 65 -> RX: 65
UVM_INFO ./tb/tb_spi_master.sv(165) @ 777935000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 777935000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: bc
UVM_INFO ./tb/tb_spi_master.sv(96) @ 785955000: uvm_test_top.ENV.AGT.DRV [DRV] TX: bc
UVM_INFO ./tb/tb_spi_master.sv(136) @ 785955000: uvm_test_top.ENV.AGT.MON [MON] TX: bc, RX: bc
UVM_INFO ./tb/tb_spi_master.sv(162) @ 785955000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: bc -> RX: bc
UVM_INFO ./tb/tb_spi_master.sv(165) @ 785955000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 785955000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: c6
UVM_INFO ./tb/tb_spi_master.sv(96) @ 793975000: uvm_test_top.ENV.AGT.DRV [DRV] TX: c6
UVM_INFO ./tb/tb_spi_master.sv(136) @ 793975000: uvm_test_top.ENV.AGT.MON [MON] TX: c6, RX: c6
UVM_INFO ./tb/tb_spi_master.sv(162) @ 793975000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: c6 -> RX: c6
UVM_INFO ./tb/tb_spi_master.sv(165) @ 793975000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***
```

100회의 검증 시뮬레이션 모두 PASS

구현 영상

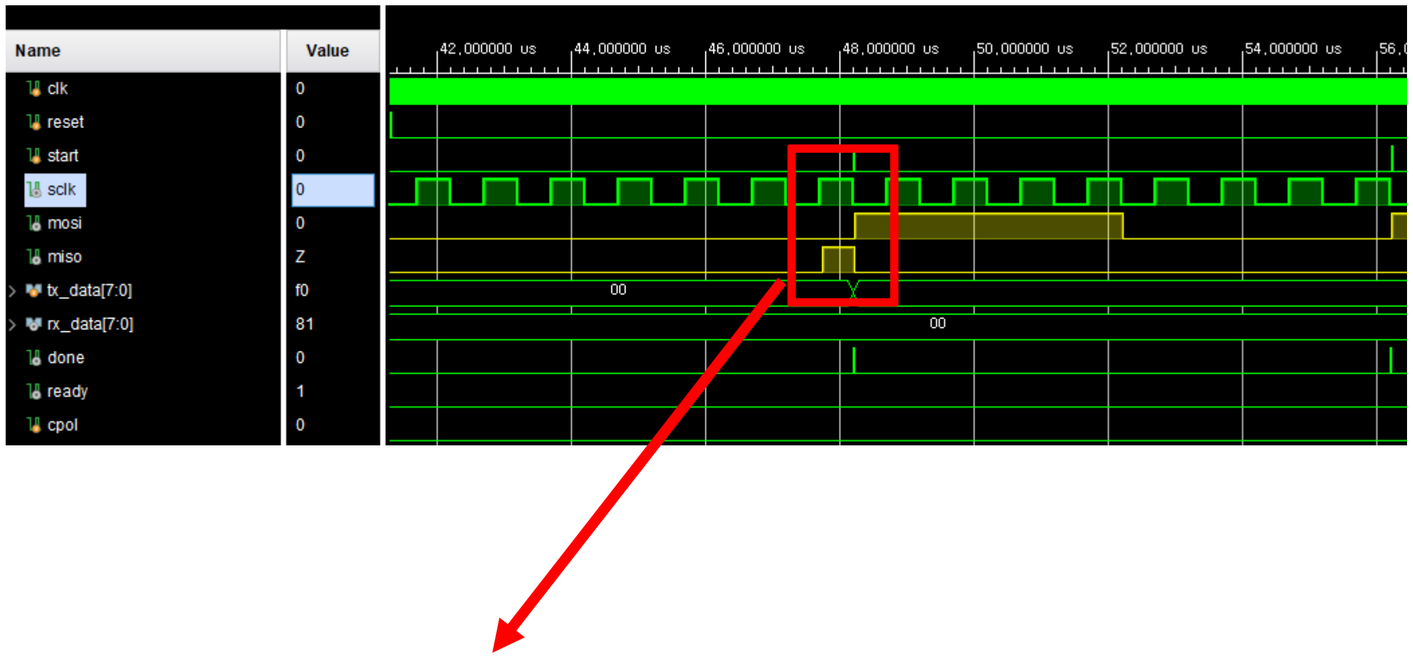


06 문제 해결

06.

문제 해결

1. MISO data가 밀리는 문제



Miso 8번으로 1000_0000이 나가야 하는 상황

첫 시작에 SCLK falling edge에서 데이터가 나가지 않고 rising edge에서부터 반 주기만 나감
=> DATA가 1개씩 밀림

06.

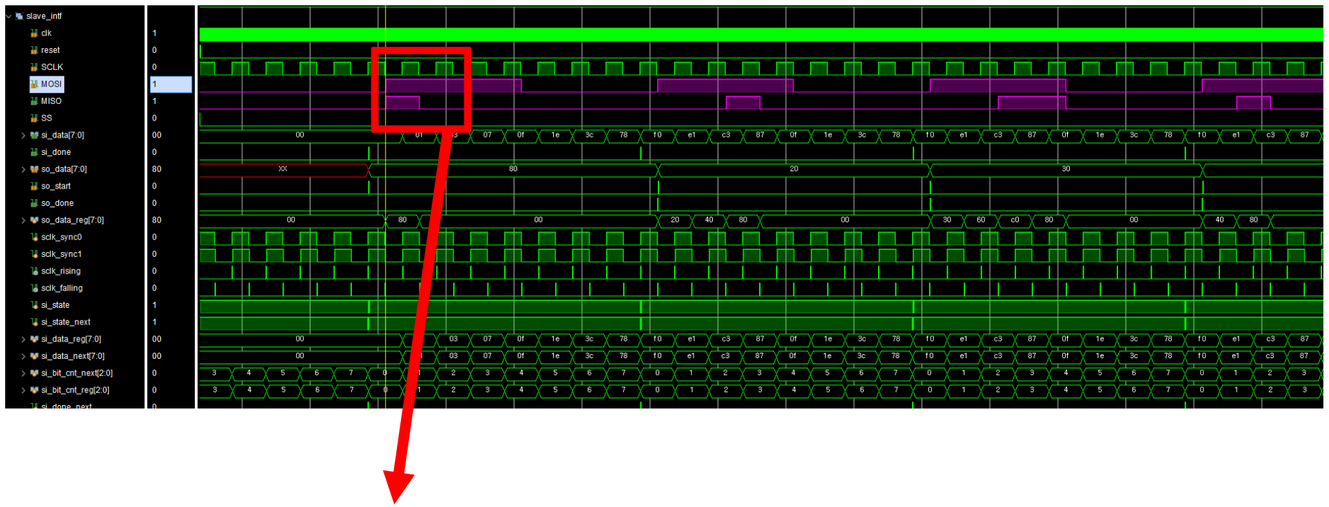
문제 해결

1. MISO data가 밀리는 문제

```
    SO_IDLE: begin
        so_done_next = 1'b0;
        if (!SS && so_start) begin
            if (start_count == 0) so_state_next = SO_WAIT;
            else begin
                so_bit_cnt_next = 0;
                so_data_next = so_data;
                so_state_next   = SO_PHASE;
            end
        end
    end

    SO_WAIT: begin
        if (sclk_falling) begin
            so_state_next = SO_PHASE;
            so_data_next = so_data;
            start_count_next = 1;
        end
    end

    SO_PHASE: begin
        if (!SS) begin
            if (sclk_falling) begin  // 송신
                so_data_next = {so_data_reg[6:0], 1'b0};
                if (so_bit_cnt_reg == 7) begin
                    so_bit_cnt_next = 0;
                    so_done_next    = 1'b1; // 8bit 송신 끝 신호
                    so_state_next   = SO_IDLE;
                end else begin
                    so_bit_cnt_next = so_bit_cnt_reg + 1;
                end
            end
        end else begin
            start_count_next = 0;
            so_state_next = SO_IDLE;
        end
    end
endcase
```

처음 시작에서 falling edge가 아니라 start 신호만을 받는 것이 문제

⇒ **State 1개 추가 및 start 신호 최적화**로 첫 시작에 falling edge 조건을 충족해야 data 송신할 수 있게 구성

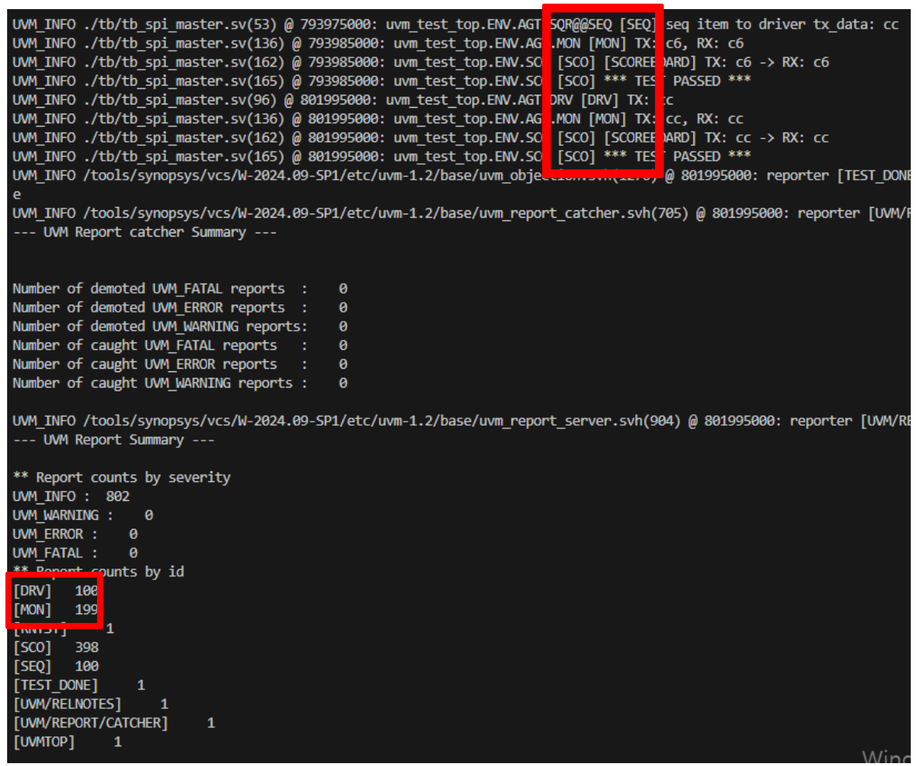


MOSI data가 밀리지 않고, SCLK falling edge에 맞춰 송신

06.

문제 해결

2. done 신호 대기 문제



100번의 검증 시뮬레이션에서 Monitor가 약 2배 많이 찍히는 문제 및 실행 순서 엉키는 문제 발생

06.

문제 해결

2. done 신호 대기 문제

```
virtual task run_phase(uvm_phase phase);
    forever begin
        wait (spi_vif.done);
        #1;
        @(posedge spi_vif.clk)
        item = spi_seq_item::type_id::create("SPI_ITEM", this);
        item.tx_data = spi_vif.tx_data;
        item.rx_data = spi_vif.rx_data;

        `uvm_info("MON", $sformatf("TX: %0x, RX: %0x", item.tx_da
        ap.write(item);
    end
endtask
```

```
virtual task run_phase(uvm_phase phase);
    forever begin
        @(posedge spi_vif.done);
        #1;
        @(posedge spi_vif.clk)
        item = spi_seq_item::type_id::create("SPI_ITEM", this);
        item.tx_data = spi_vif.tx_data;
        item.rx_data = spi_vif.rx_data;

        `uvm_info("MON", $sformatf("TX: %0x, RX: %0x", item.tx_data, iter
        ap.write(item);
    end
endtask
```

단순히 done이 1 일 때만 감지하면, Monitor에서 원하지 않는 데이터 송/수신이 진행됨

-> done의 rising edge를 감지하여 done을 한 번만 감지하게 함

06.

문제 해결

2. done 신호 대기 문제

```
--- UVM Report Summary ---

** Report counts by severity
UVM_INFO :  505
UVM_WARNING :    0
UVM_ERROR :    0
UVM_FATAL :    0
** Report counts by id
[DRV]   100
[MON]   100
[RNTST]     1
[SCO]   200
[SEQ]   100
[TEST_DONE]     1
[UVM/RELNOTES]     1
[UVM/REPORT/CATCHER]     1
[UVMTOP]     1
```

```
UVM_INFO ./tb/tb_spi_master.sv(53) @ 737835000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: fa
UVM_INFO ./tb/tb_spi_master.sv(96) @ 745855000: uvm_test_top.ENV.AGT.DRV [DRV] TX: fa
UVM_INFO ./tb/tb_spi_master.sv(136) @ 745855000: uvm_test_top.ENV.AGT.MON [MON] TX: fa, RX: fa
UVM_INFO ./tb/tb_spi_master.sv(162) @ 745855000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: fa -> RX: fa
UVM_INFO ./tb/tb_spi_master.sv(165) @ 745855000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 745855000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: ab
UVM_INFO ./tb/tb_spi_master.sv(96) @ 753875000: uvm_test_top.ENV.AGT.DRV [DRV] TX: ab
UVM_INFO ./tb/tb_spi_master.sv(136) @ 753875000: uvm_test_top.ENV.AGT.MON [MON] TX: ab, RX: ab
UVM_INFO ./tb/tb_spi_master.sv(162) @ 753875000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: ab -> RX: ab
UVM_INFO ./tb/tb_spi_master.sv(165) @ 753875000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 753875000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: 9d
UVM_INFO ./tb/tb_spi_master.sv(96) @ 761895000: uvm_test_top.ENV.AGT.DRV [DRV] TX: 9d
UVM_INFO ./tb/tb_spi_master.sv(136) @ 761895000: uvm_test_top.ENV.AGT.MON [MON] TX: 9d, RX: 9d
UVM_INFO ./tb/tb_spi_master.sv(162) @ 761895000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: 9d -> RX: 9d
UVM_INFO ./tb/tb_spi_master.sv(165) @ 761895000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 761895000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: 55
UVM_INFO ./tb/tb_spi_master.sv(96) @ 769915000: uvm_test_top.ENV.AGT.DRV [DRV] TX: 55
UVM_INFO ./tb/tb_spi_master.sv(136) @ 769915000: uvm_test_top.ENV.AGT.MON [MON] TX: 55, RX: 55
UVM_INFO ./tb/tb_spi_master.sv(162) @ 769915000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: 55 -> RX: 55
UVM_INFO ./tb/tb_spi_master.sv(165) @ 769915000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 769915000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: 65
UVM_INFO ./tb/tb_spi_master.sv(96) @ 777935000: uvm_test_top.ENV.AGT.DRV [DRV] TX: 65
UVM_INFO ./tb/tb_spi_master.sv(136) @ 777935000: uvm_test_top.ENV.AGT.MON [MON] TX: 65, RX: 65
UVM_INFO ./tb/tb_spi_master.sv(162) @ 777935000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: 65 -> RX: 65
UVM_INFO ./tb/tb_spi_master.sv(165) @ 777935000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 777935000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: bc
UVM_INFO ./tb/tb_spi_master.sv(96) @ 785955000: uvm_test_top.ENV.AGT.DRV [DRV] TX: bc
UVM_INFO ./tb/tb_spi_master.sv(136) @ 785955000: uvm_test_top.ENV.AGT.MON [MON] TX: bc, RX: bc
UVM_INFO ./tb/tb_spi_master.sv(162) @ 785955000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: bc -> RX: bc
UVM_INFO ./tb/tb_spi_master.sv(165) @ 785955000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***

UVM_INFO ./tb/tb_spi_master.sv(53) @ 785955000: uvm_test_top.ENV.AGT.SQR@@SEQ [SEQ] seq item to driver tx_data: c6
UVM_INFO ./tb/tb_spi_master.sv(96) @ 793975000: uvm_test_top.ENV.AGT.DRV [DRV] TX: c6
UVM_INFO ./tb/tb_spi_master.sv(136) @ 793975000: uvm_test_top.ENV.AGT.MON [MON] TX: c6, RX: c6
UVM_INFO ./tb/tb_spi_master.sv(162) @ 793975000: uvm_test_top.ENV.SCO [SCO] [SCOREBOARD] TX: c6 -> RX: c6
UVM_INFO ./tb/tb_spi_master.sv(165) @ 793975000: uvm_test_top.ENV.SCO [SCO] *** TEST PASSED ***
```

100회의 검증 시뮬레이션 모두 PASS

07.

고찰

통신 프로토콜 구현 프로젝트를 진행하며, 소프트웨어의 논리적 오류가 없다고 생각한 작은 차이도 하드웨어 동작 상에서는 큰 타이밍 차이로 반영된다는 것을 깨달았습니다.

프로젝트를 진행하면 중간에 문제가 생기는 경우가 많은데, 단순히 머리로만 생각할 것이 아니라 output port 등을 유동적으로 활용하여 디버깅 하는 경험이 필요하다고 생각했습니다.