RV32I 기반 Multi-Cycle CPU 및 AMBA APB Peripheral 설계 검증

채민석

대한상공회의소

CONTENTS

01 개 요

02 Multi Cycle

03 AMBA APB

04 Peripheral

05 Verification

06 C Application

07 Trouble Shooting

1. 개 요

● 프로젝트 목표

- RSIC-V RV32I 기반 Muti-Cycle CPU CORE 설계
- AMBA APB 기반 Peripheral 연결
- SystemVerilog를 이용한 Peripheral 검증
- C언어를 통한 Application Code 작성 및 동작 구현

● 구현환경

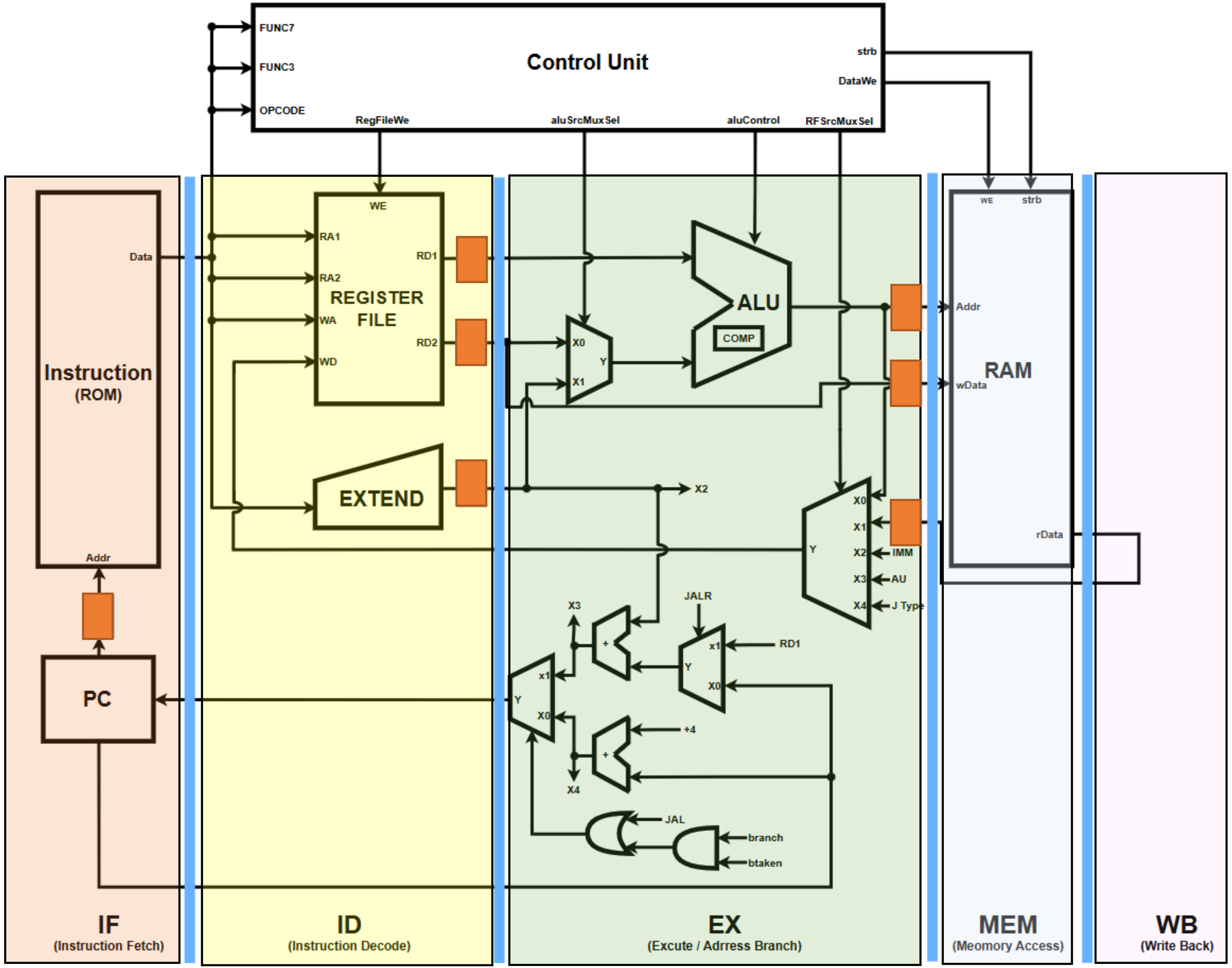
SystemVerilog

Vivado

RISC-V

AMBA

2. Multi Cycle



Single Cycle

Instruction

1 Clk Cycle

- 가장 느린 Instrcution에 따라 clk period설정

Multi Cycle

FETCH	DECODE	EXCUTE	MEMORY	WRITE BACK

1 Clk Cycle

FETCH : Instruction Memory에서 명령어 가져옴

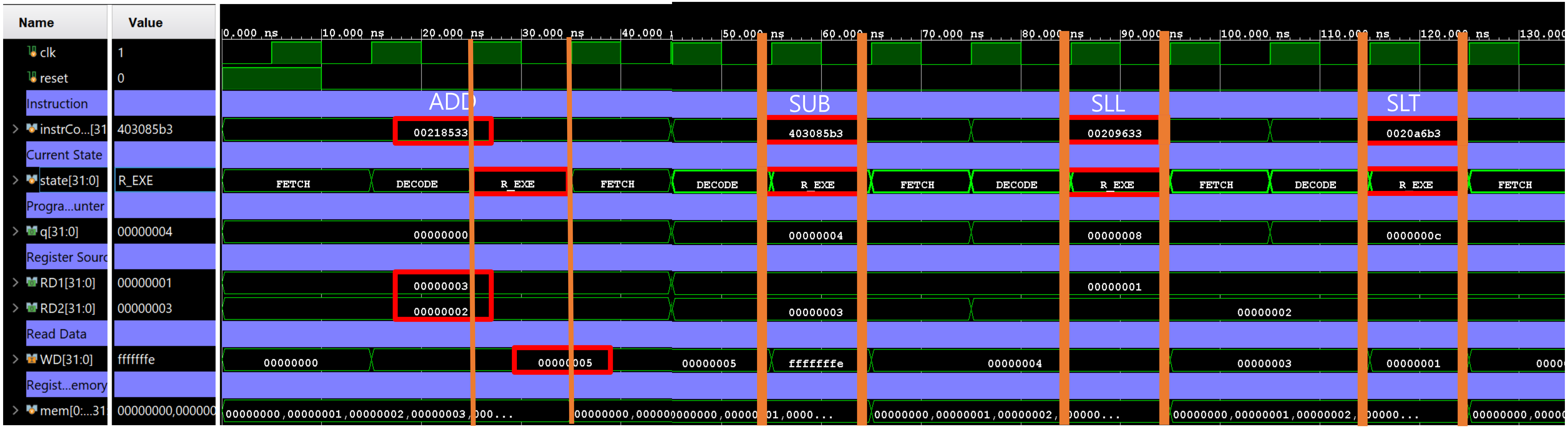
DECODE : 명령어 분석 및 Reg 읽기 / immediate 확장

EXCUTE : ALU에 연산 수행

MEMORY : L / S Type에서의 Data Memory 접근

WRITE BACK : Register에 저장

2. R-Type Simulation

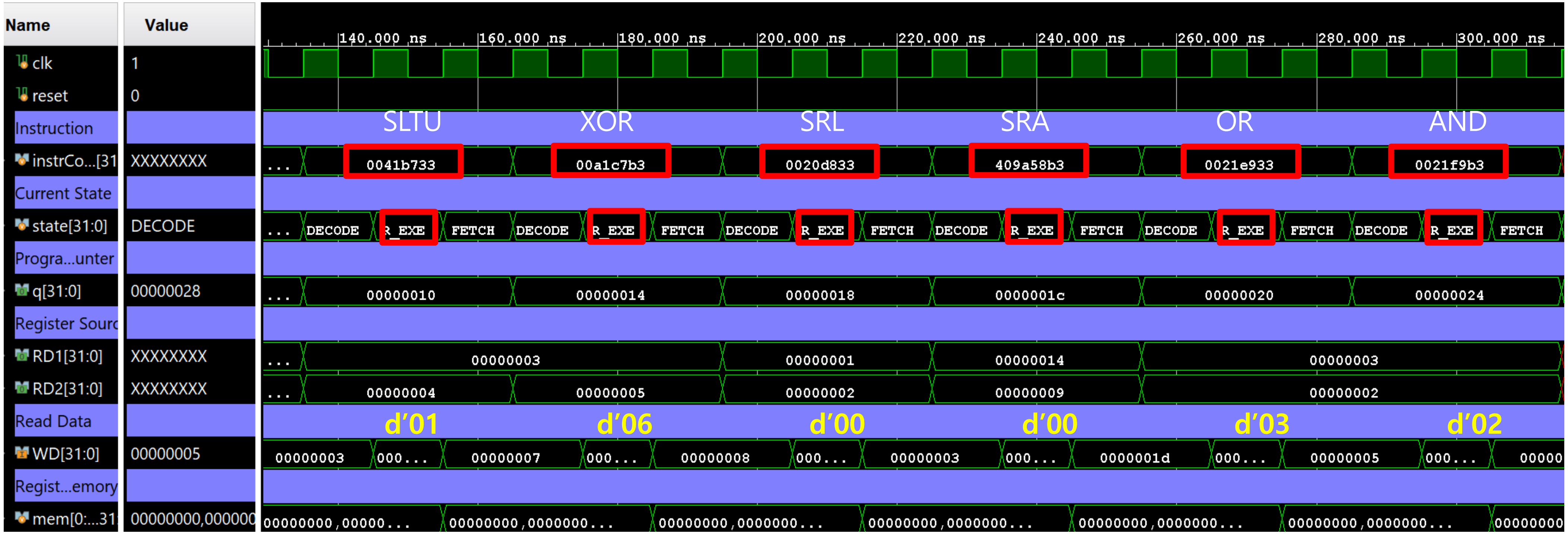


Name	Value
clk	1
reset	0
Instruction	
instrCo...[31	403085b3
Current State	
state[31:0]	R_EXE
Progra...unter	
q[31:0]	00000004
Register Sourc	
RD1[31:0]	00000001
RD2[31:0]	00000003
Read Data	
WD[31:0]	fffffffe
Regist...emory	
mem[0:...31	00000000,000000

mnemoic	Semantics	Destination	result
ADD	rd = rs1 + rs2	x10, x2 x3	x0000_0005
SUB	rd = rs1 - rs2	x11 x3 x1	xFFFF_FFFE
SLL	rd = rs1 << rs2	x12 x2 x1	x0000_0004
SLT	rd ($signed(a)<$signed(b))?1:0	x13 x2 x1	x0000_0001
SLTU	rd = (rs1 < rs2)?1:0	x14 x4 x3	x0000_0001

mnemoic	Semantics	Destination	result
XOR	rd = rs1 ^ rs2	x15, x5 x3	x0000_0006
SRL	rd = rs1 >> rs2	x16 x2 x1	x0000_0000
SRA	rd = rs1 >>> rs2	x17 x9 x20	x0000_0000
OR	rd = rs1 | rs2	x18 x2 x3	x0000_0003
AND	rd = rs1 & rs2	x19 x2 x3	x0000_0002

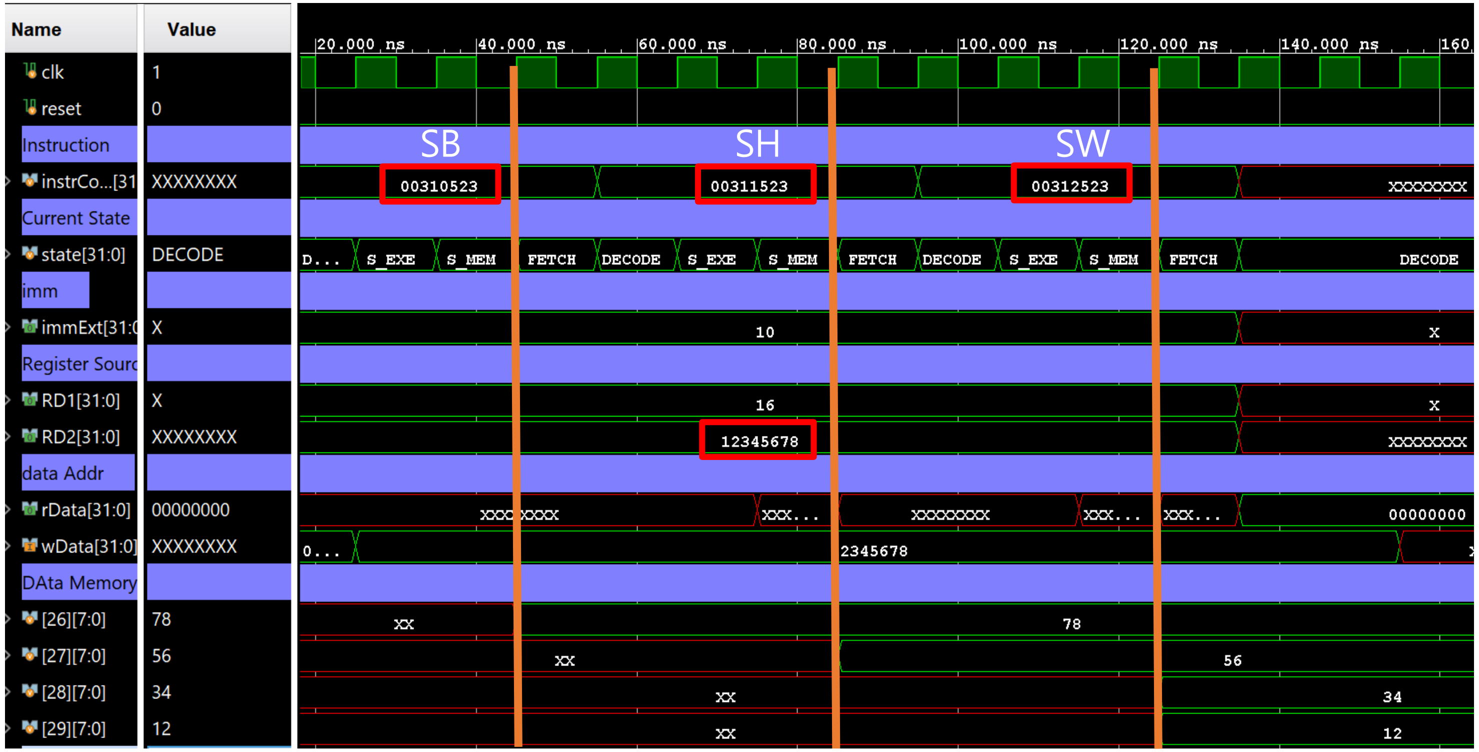
2. R-Type Simulation



mnemoic	Semantics	Destination	result
ADD	rd = rs1 + rs2	x10, x2 x3	x0000_0005
SUB	rd = rs1 - rs2	x11 x3 x1	xFFFF_FFFE
SLL	rd = rs1 << rs2	x12 x2 x1	x0000_0004
SLT	rd ($signed(a)<$signed(b))?1:0	x13 x2 x1	x0000_0001
SLTU	rd = (rs1 < rs2)?1:0	x14 x4 x3	x0000_0001

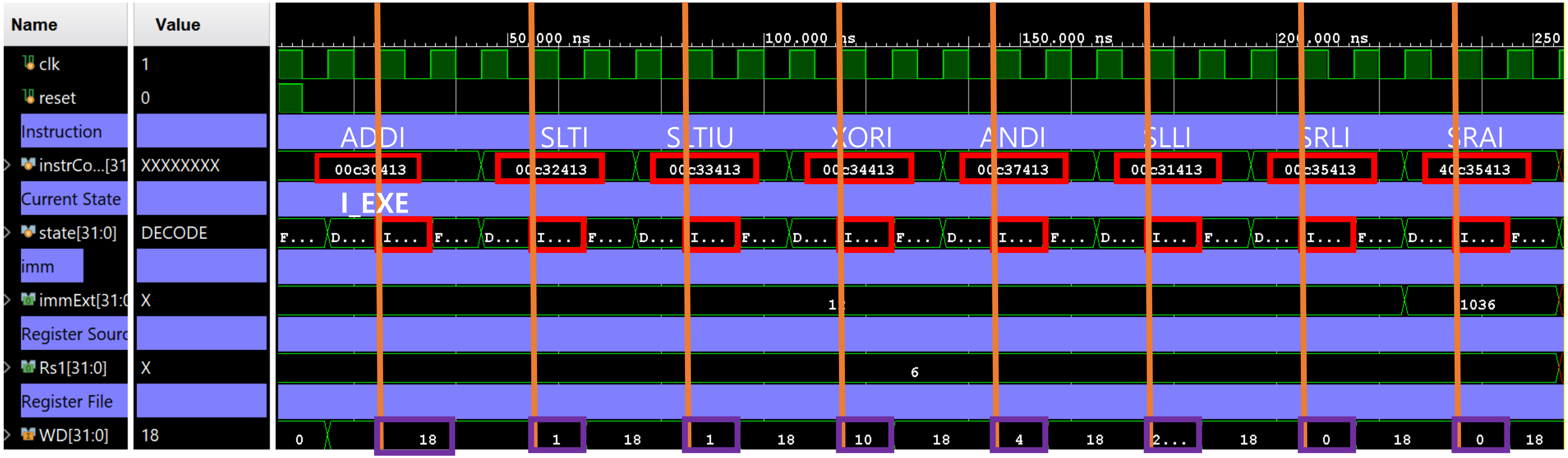
mnemoic	Semantics	Destination	result
XOR	rd = rs1 ^ rs2	x15, x5 x3	x0000_0006
SRL	rd = rs1 >> rs2	x16 x2 x1	x0000_0000
SRA	rd = rs1 >>> rs2	x17 x9 x20	x0000_0000
OR	rd = rs1 | rs2	x18 x2 x3	x0000_0003
AND	rd = rs1 & rs2	x19 x2 x3	x0000_0002

2. S-Type Simulation



mnemoic	Semantics	Destination	Result
SB	M[rs1+imm][0:7] =rs2[0:7]	x12345678, 10 x16	XXXX_0078
SH	M[rs1+imm][0:15] =rs2[0:15]	x12345678, 10 x16	XXXX_5678
SW	M[rs1+imm][0:31] =rs2[0:31]	x12345678, 10 x16	1234_5678

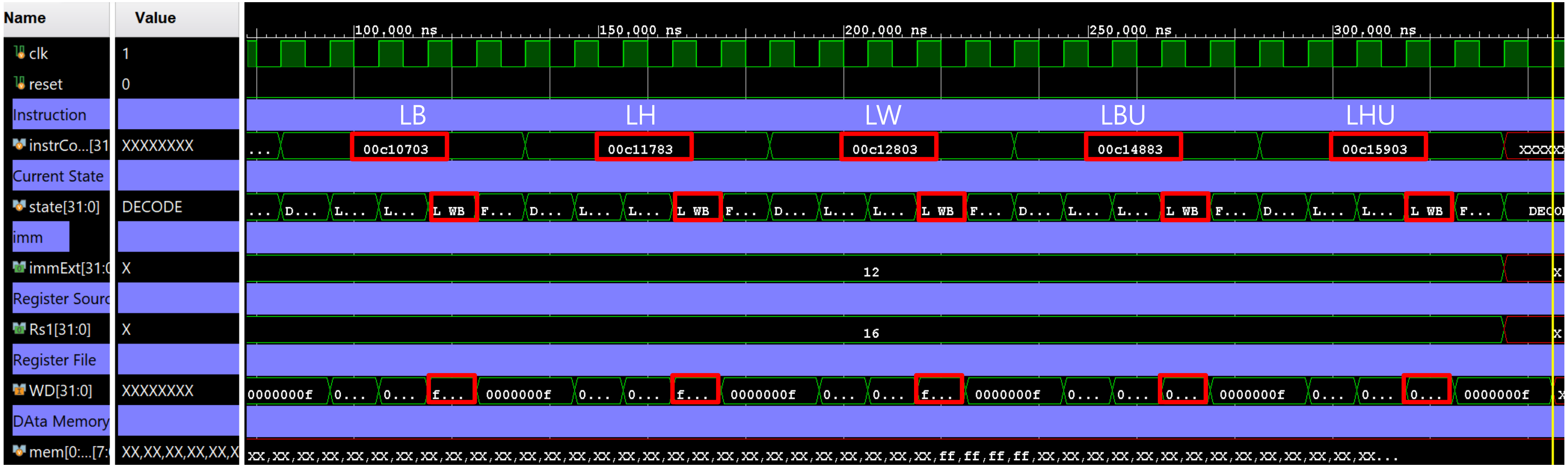
2. I-Type Simulation



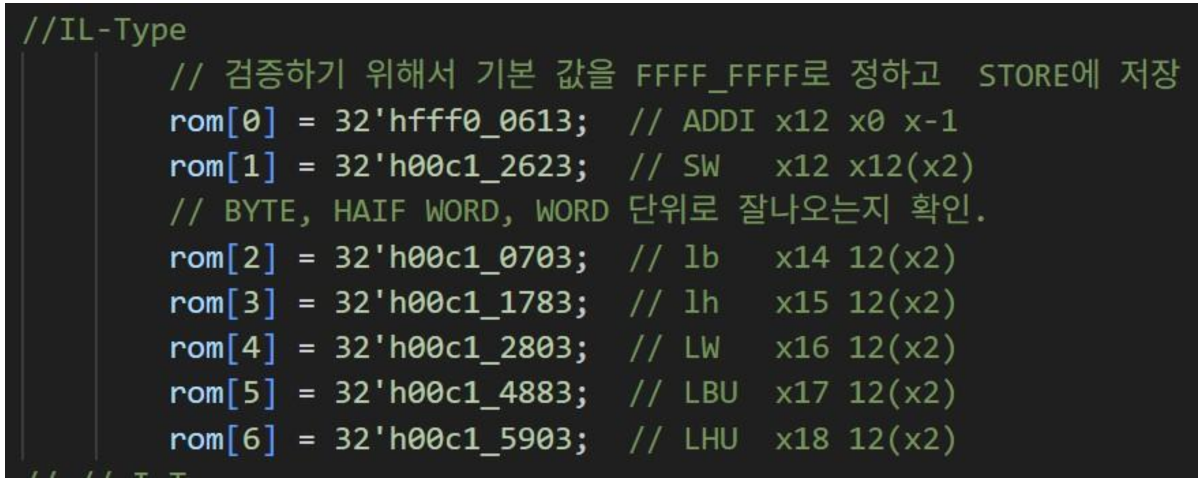
mnemoic	Semantics	Format	Result
ADDI	rd = rs1 + imm	x5, x6 x12	‘dx18
SLTI	rd = (rs1 < imm)?1:0	x5, x6 x12	‘dx1
SLTIU	rd = (rs1 < imm)?1:0	x5, x6 x12	‘dx1
XORI	rd = rs1 ˆ imm	x5, x6 x12	‘dx10

mnemoic	Semantics	Format	Result
ANDI	rd = rs1 & imm	x5, x6 x12	‘dx4
SLLI	rd = rs1 << imm[0:4]	x5, x6 x12	‘dx24576
SRLI	rd = M[rs1+imm][0:31]	x5, x6 x12	‘dx0
SRAI	rd = rs1 >> imm[0:4]	x5, x6 x12	‘dx0

2. L-Type Simulation

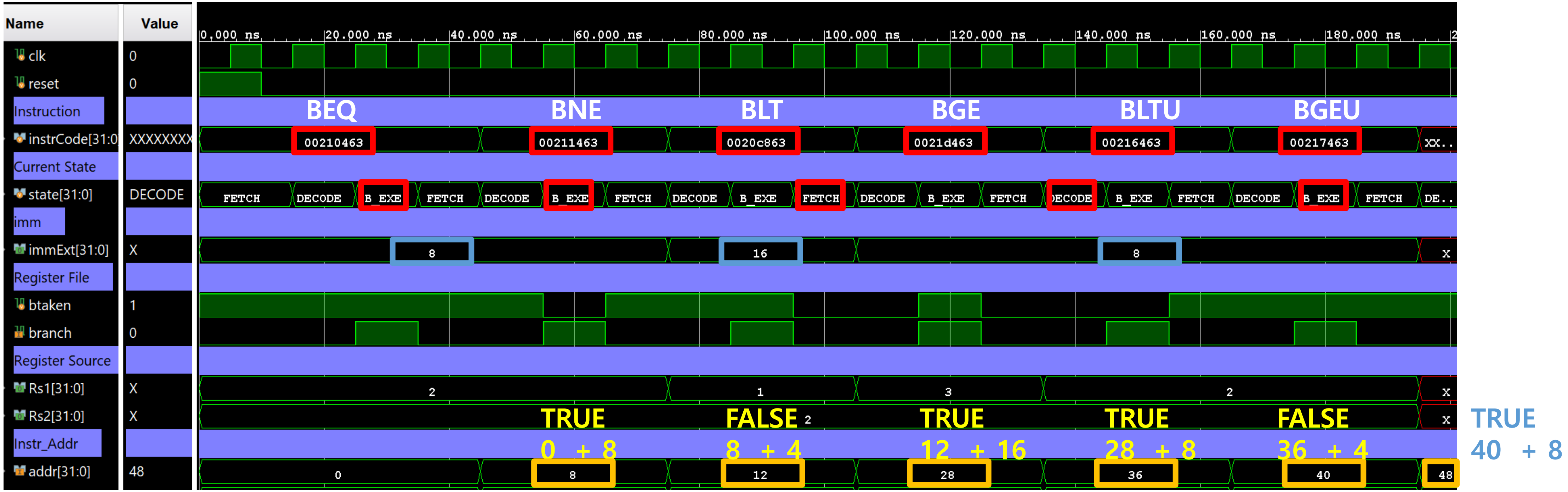


mnemoic	Semantics	Format	Result
LB	rd = M[rs1+imm][0: 7]	x5, x2 x12	FFFF_FFFF
LH	rd = M[rs1+imm][0:15]	x5, x2 x12	FFFF_FFFF
LW	rd = M[rs1+imm][0:31]	x5, x2 x12	FFFF_FFFF
LBU	rd = M[rs1+imm][0: 7] (zero)	x5, x2 x12	0000_00FF
LHU	rd = M[rs1+imm][0:15] (zero)	x5, x2 x12	0000_FFFF

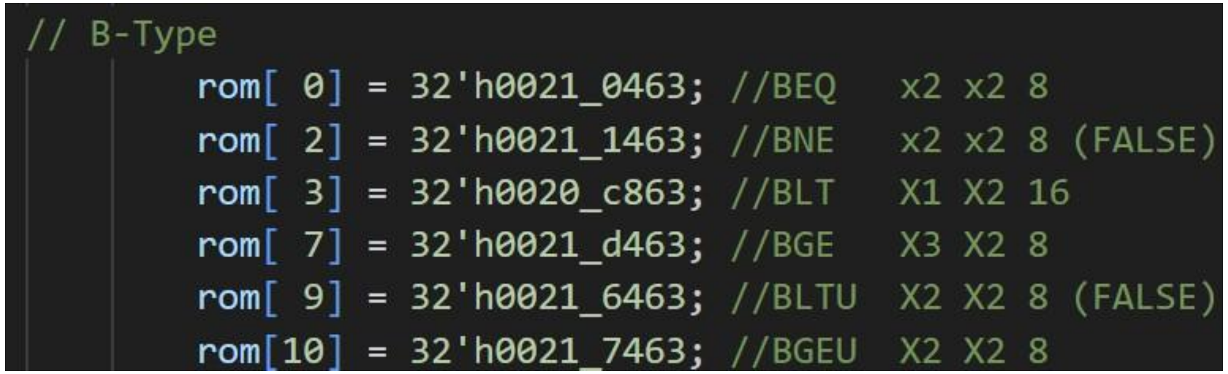


```
//IL-Type
        // 검증하기 위해서 기본 값을 FFFF_FFFF로 정하고  STORE에 저장
        rom[0] = 32'hfff0_0613;  // ADDI x12 x0 x-1
        rom[1] = 32'h00c1_2623;  // SW   x12 x12(x2)
        // BYTE, HAIF WORD, WORD 단위로 잘나오는지 확인.
        rom[2] = 32'h00c1_0703;  // lb   x14 12(x2)
        rom[3] = 32'h00c1_1783;  // lh   x15 12(x2)
        rom[4] = 32'h00c1_2803;  // LW   x16 12(x2)
        rom[5] = 32'h00c1_4883;  // LBU  x17 12(x2)
        rom[6] = 32'h00c1_5903;  // LHU  x18 12(x2)
```

2. B-Type Simulation

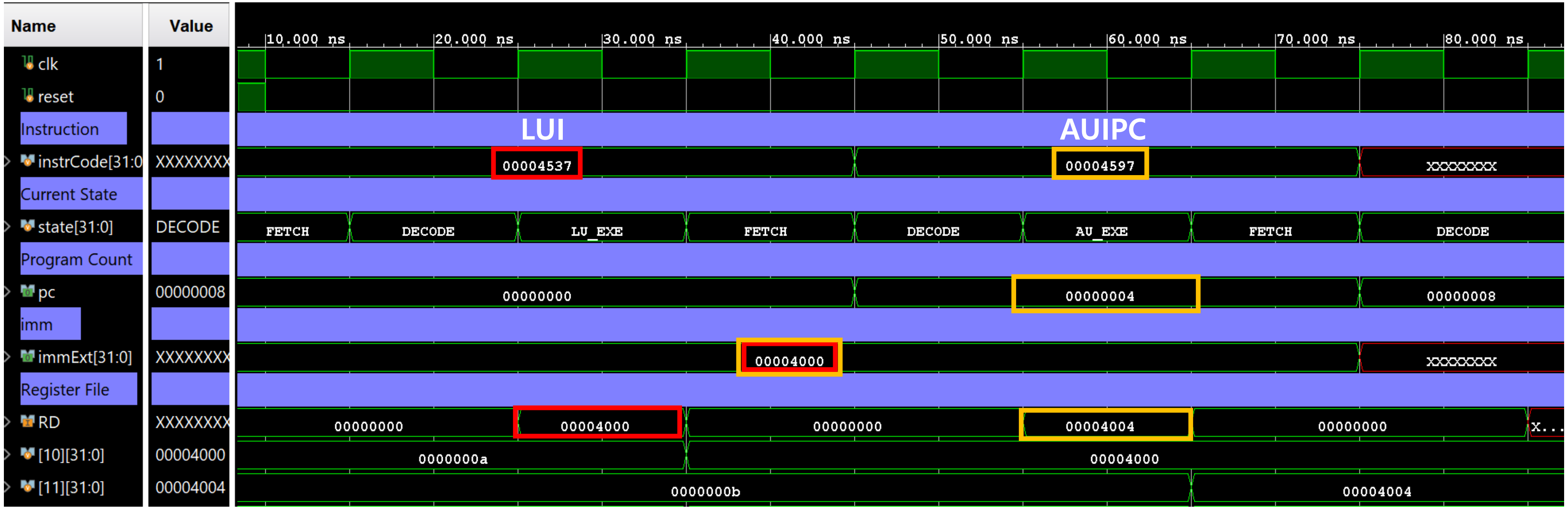


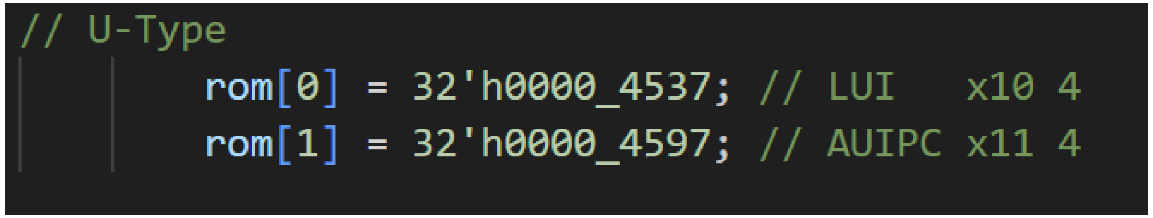
mnemoic	Semantics	Format
BEQ	if(rs1 == rs2) PC += imm	x2, x2, 8
BNE	if(rs1 != rs2) PC += imm	x2, x2, 8
BLT	if(rs1 < rs2) PC += imm	x51 x2, 16
BGE	if(rs1 >= rs2) PC += imm	x2, x2, 8
BLTU	if(rs1 < rs2) PC += imm	x2, x2, 8
BGEU	if(rs1 >= rs2) PC += imm	x2, x2, 8



```
// B-Type
        rom[ 0] = 32'h0021_0463; //BEQ   x2 x2 8
        rom[ 2] = 32'h0021_1463; //BNE   x2 x2 8 (FALSE)
        rom[ 3] = 32'h0020_c863; //BLT   X1 X2 16
        rom[ 7] = 32'h0021_d463; //BGE   X3 X2 8
        rom[ 9] = 32'h0021_6463; //BLTU  X2 X2 8 (FALSE)
        rom[10] = 32'h0021_7463; //BGEU  X2 X2 8
```

2. U-Type Simulation



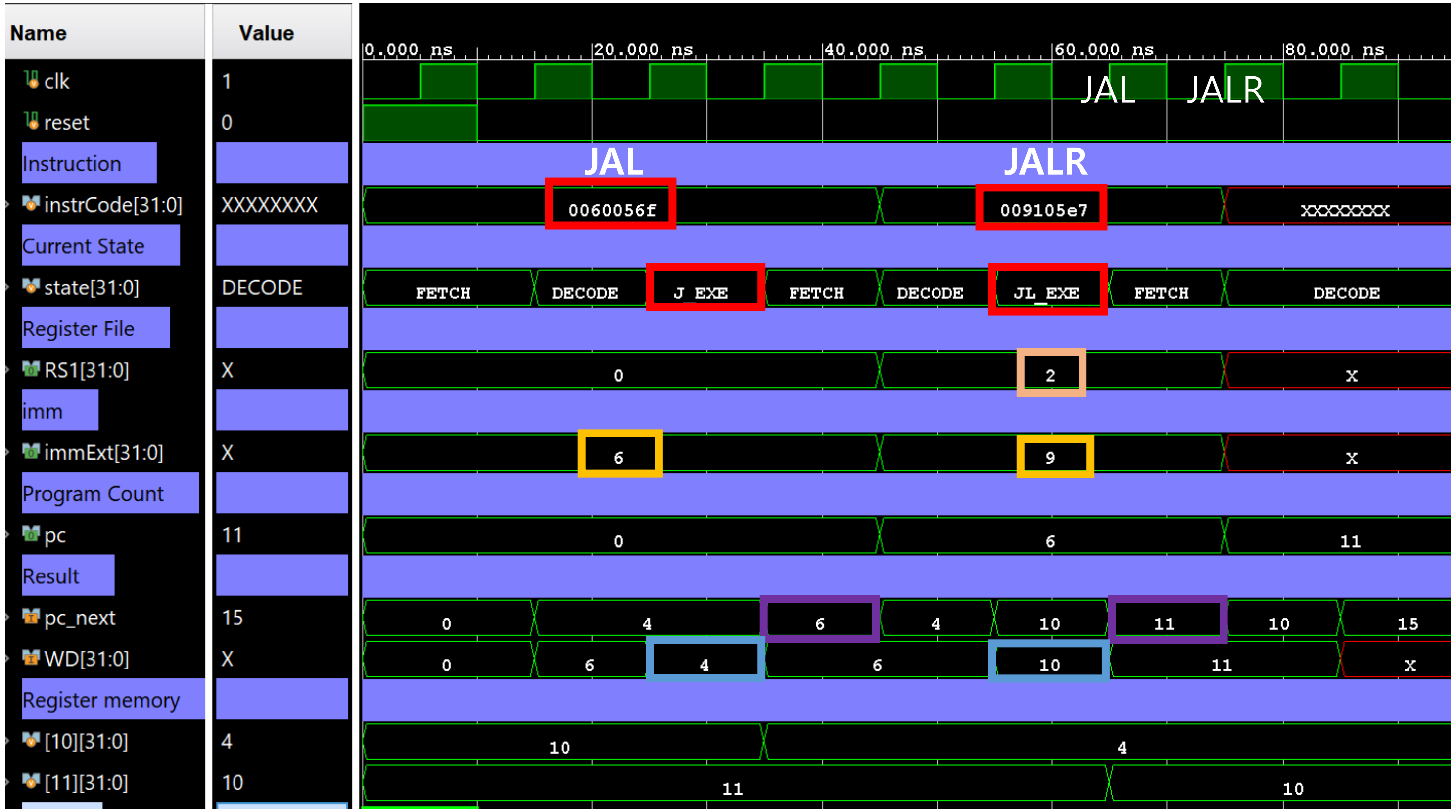


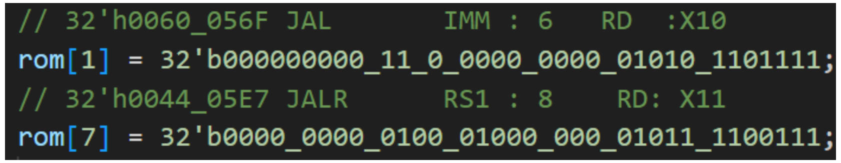
```
// U-Type
        rom[0] = 32'h0000_4537; // LUI   x10 4
        rom[1] = 32'h0000_4597; // AUIPC x11 4
```

LUI : imm = 0100_0000_0000_0000
rd= imm
hx4000 = 0100_0000_0000_0000

AUIPC : imm = 0100_0000_0000_0000 / pc = 3
rd= pc(3) + imm
hx4000 = 0100_0000_0000_0000

2. J-Type Simulation





```
// 32'h0060_056F JAL        IMM : 6    RD  :X10
rom[1] = 32'b000000000_11_0_0000_0000_01010_1101111;
// 32'h0044_05E7 JALR       RS1 : 8    RD: X11
rom[7] = 32'b0000_0000_0100_01000_000_01011_1100111;
```

JAL	rd = PC+4; PC += imm
JALR	rd = PC+4; PC = rs1 + imm

JAL : imm = 6
pc_next[6] = pc[0] + imm[6]
rd[4] = pc[0] + 4

JALR : imm = 9 , rs1 = 2
pc_next[11] = imm[9] + rs1[2]
rd[10] = pc[6] + 4

3. AMBA BUS

● AMBA (Advanced Microcontroller Bus Architecture)

- ARM에서 개발한 SoC 내부 통신을 위한 표준 protocol
- BUS 종류 : AHB, APB, AXI 등

● BUS 장점

1. 인터페이스 간소화

   - IP Block을 직접 wire로 연결하면 N×(N-1) 개의 연결선 필요

   - Bus 구조를 사용하면 각 IP는 Bus에만 연결 → wire 수 감소

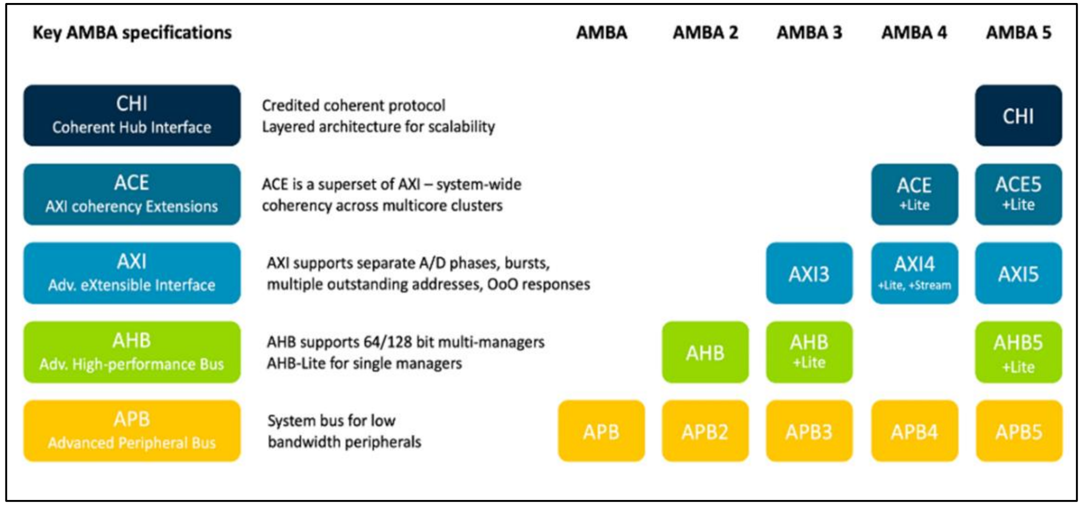
2. 쉬운 재사용과 확장

   - Bus 표준만 맞추면 다른 프로젝트의 IP Block 재사용 가능

   - 새로운 IP Block 추가 시 Bus에만 연결하면 되어 simple

3. 표준화된 프로토콜

   -ARM이 정의한 업계 표준 인터페이스, IP 간 호환 가능

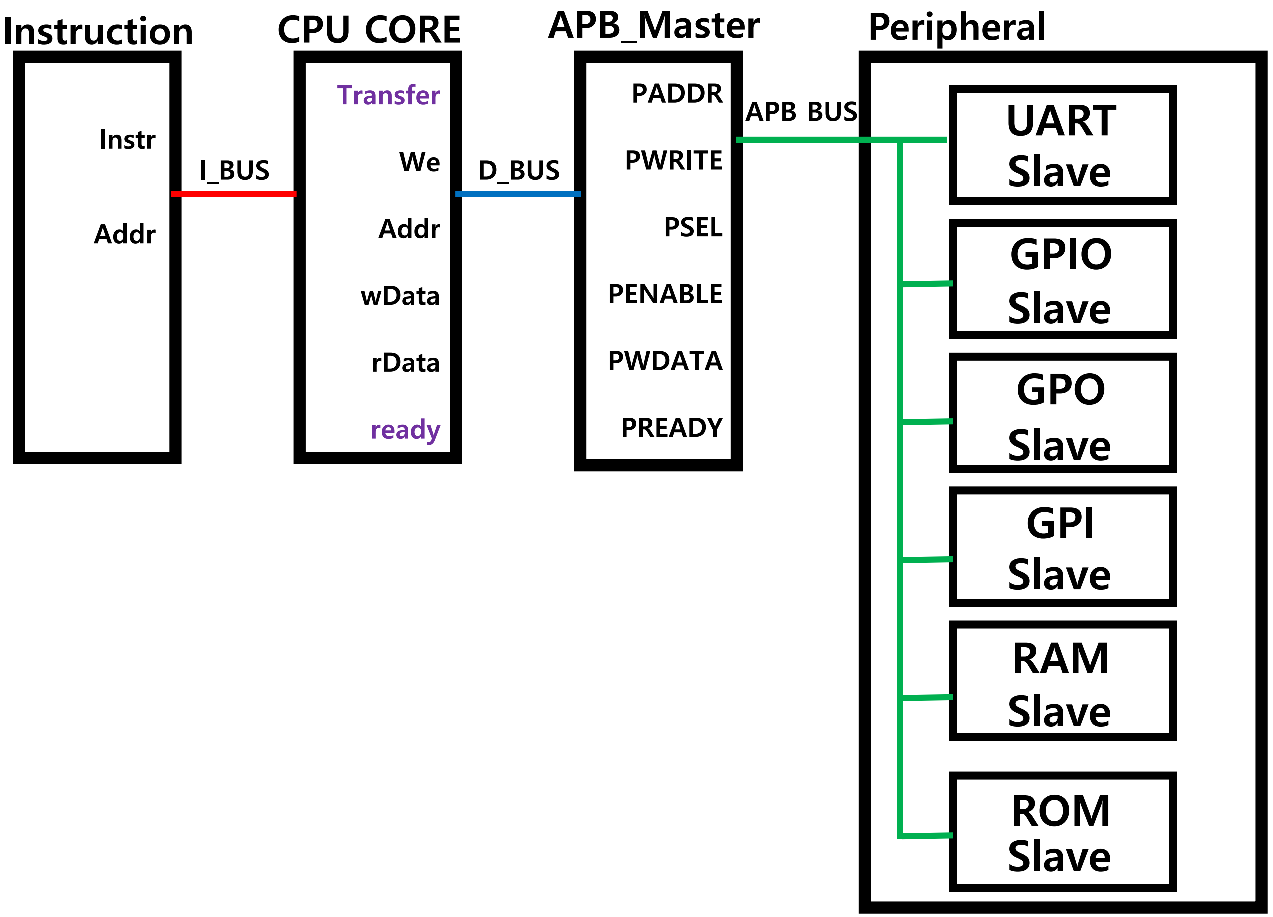


Key AMBA specifications		AMBA	AMBA 2	AMBA 3	AMBA 4	AMBA 5
CHI Coherent Hub Interface	Credited coherent protocol Layered architecture for scalability					CHI
ACE AXI coherency Extensions	ACE is a superset of AXI – system-wide coherency across multicore clusters				ACE +Lite	ACE5 +Lite
AXI Adv. eXtensible Interface	AXI supports separate A/D phases, bursts, multiple outstanding addresses, OoO responses			AXI3	AXI4 +Lite, +Stream	AXI5
AHB Adv. High-performance Bus	AHB supports 64/128 bit multi-managers AHB-Lite for single managers		AHB	AHB +Lite		AHB5 +Lite
APB Advanced Peripheral Bus	System bus for low bandwidth peripherals	APB	APB2	APB3	APB4	APB5

3. APB (Advanced Peripheral Bus)

APB Architecture

저전력·저대역폭 주변장치용 AMBA 서브버스, 동기식 non-piplined 전송



Memory Map

0x1000_4008	UART RDR
0x1000_4004	UART TDR
0x1000_4000	UART STATUS
0x1000_3008	GPIO IDR
0x1000_3004	GPIO ODR
0x1000_3000	GPIO CR
0x1000_2004	GPI IDR
0x1000_2000	GPI CR
0x1000_1004	GPO ODR
0x1000_1000	GPO CR
0x1000_0000	RAM
0x0000_0000	ROM

3. APB (Advenced Peripheral Bus)

● APB (Advenced Peripheral Bus) FSM

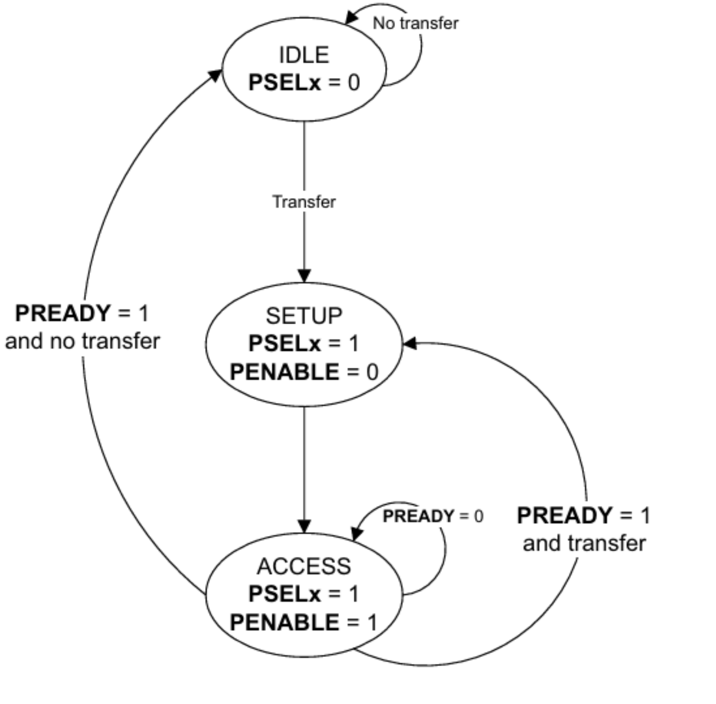
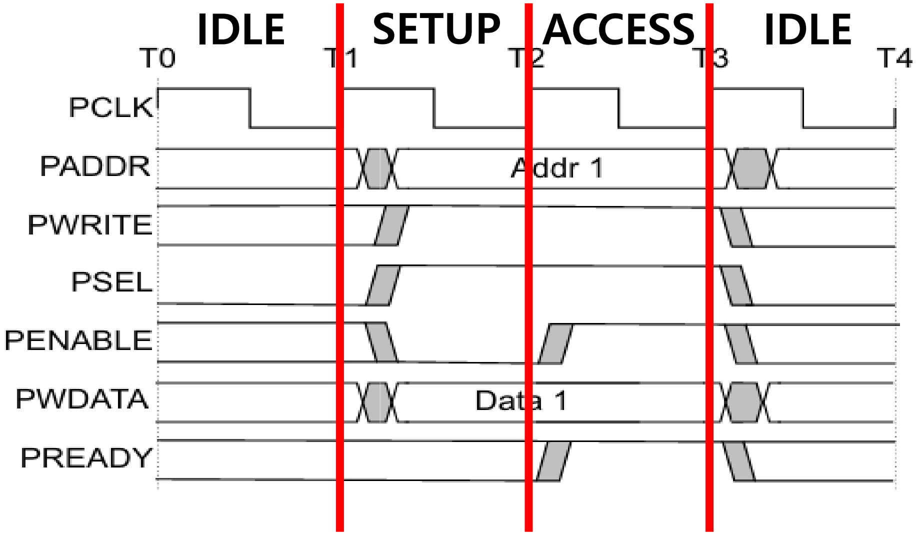
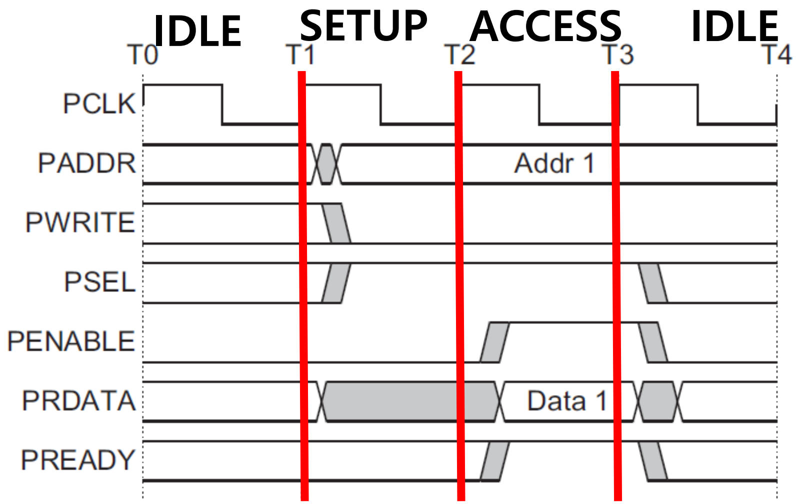


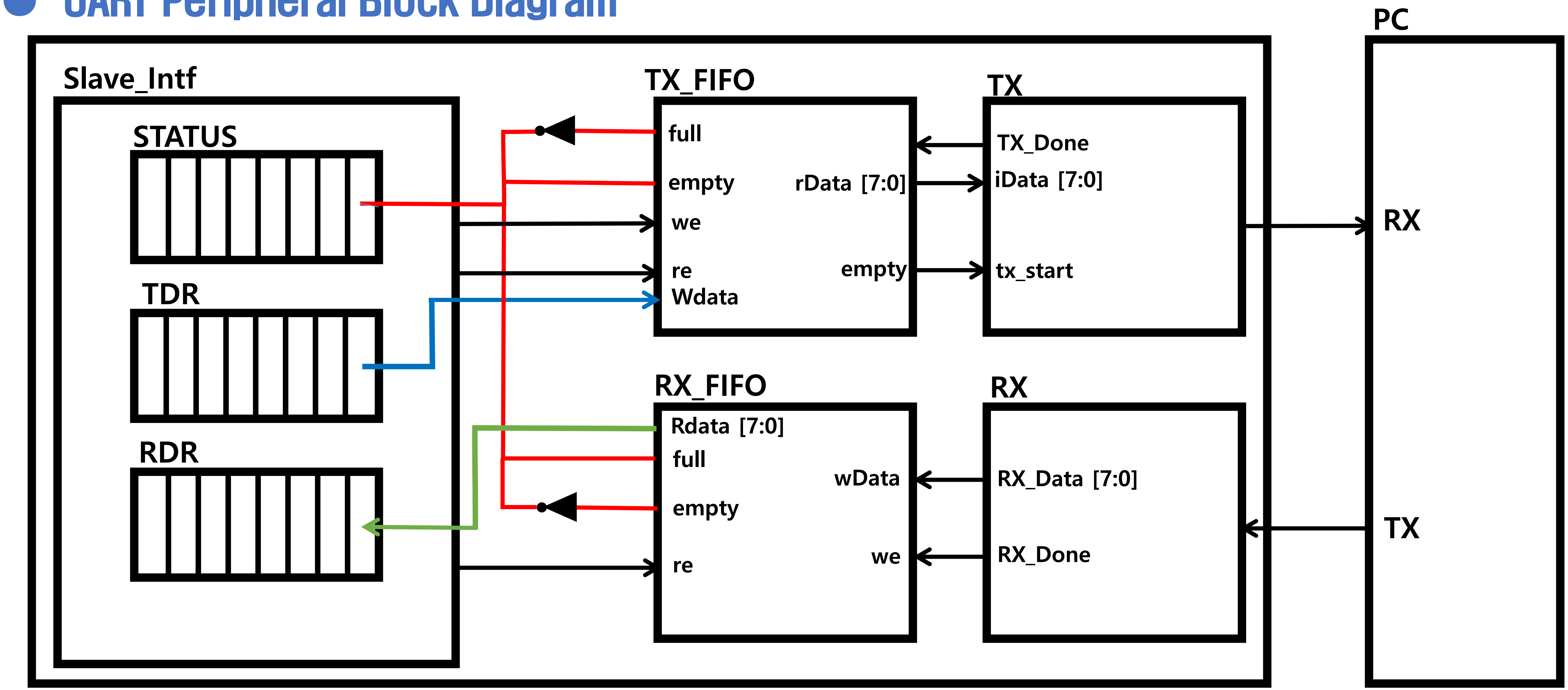
Figure 4-1 State diagram





4. UART Peripheral

UART Peripheral Block Diagram



대한상공회의소

4. UART Peripheral

● UART Peripheral

OFFSET	Register	31 ~ 8	7	6	5	4	3	2	1	0
4000	USR	Reserved					RXFF	TXFE	TXFNF	RXFNE
4004	TDR	Reserved	TDR							
4008	RDR	Reserved	RDR							

1. USR (UART Status Register) 0x4000

- [0] RXFNE (RX FIFO Not Empty)
- [1] TXFNF (TX FIFO Not Full)
- [2] TXFE (TX FIFO Empty)
- [3] RXFF (RX FIFO Full)

2. TDR (TX Data Register) 0x0008

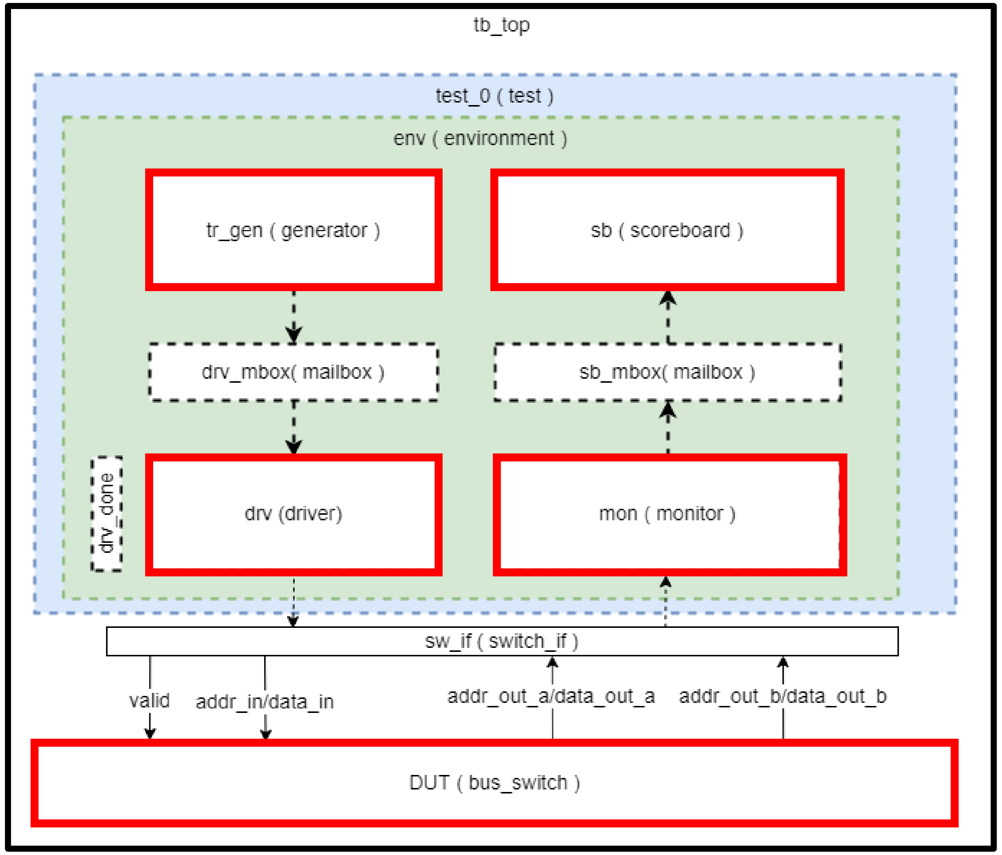
- Write : TX FIFO에 Data Write

3. RDR (RX Data Register) 0x000C

- Read : RX FIFO에서 Data Read

5. UART Peripheral Verification

● Verification 방법



RX 검증

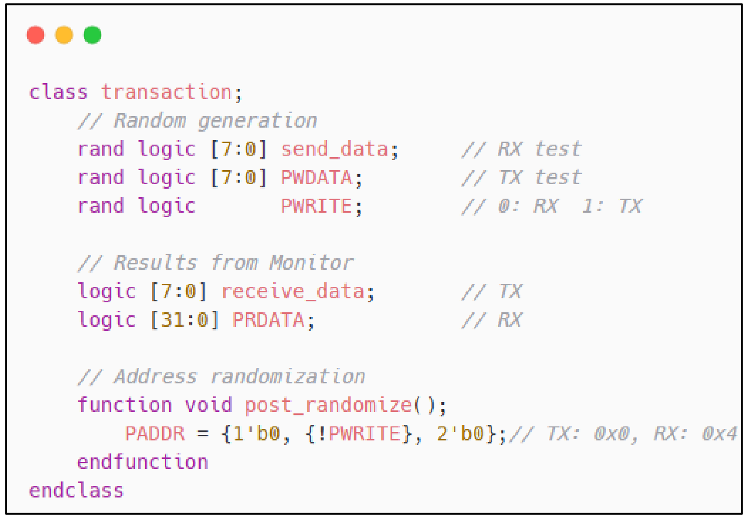
- Generator에서 send_data[7:0] 랜덤 생성
- Driver가 UART rx 핀으로 send_data 전송
- DUT의 UART_RX가 데이터 수신 → RX FIFO 저장
- Driver가 APB Read (RDR 0x4)로 데이터 읽기
- Monitor가 PRDATA 캡처
- Scoreboard에서 비교: send_data == PRDATA

TX 검증

- Generator에서 PWDATA[7:0] 랜덤 생성
- Driver가 APB Write (TDR 0x0)로 PWDATA 전송
- DUT의 TX FIFO → UART_TX로 데이터 전송
- Monitor가 tx 핀에서 직렬 데이터 샘플링
- Scoreboard에서 비교: PWDATA == receive_data

5. UART Peripheral Verification

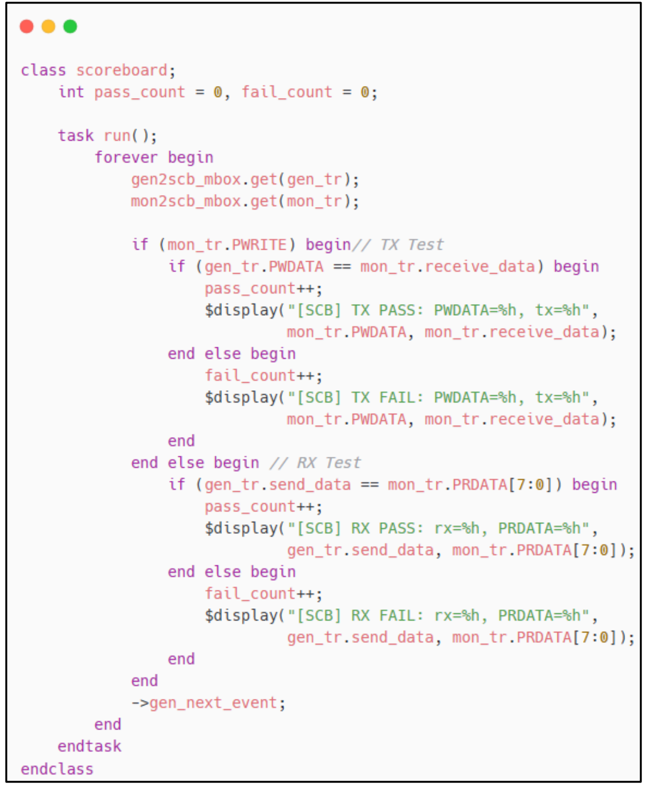
● Verification code



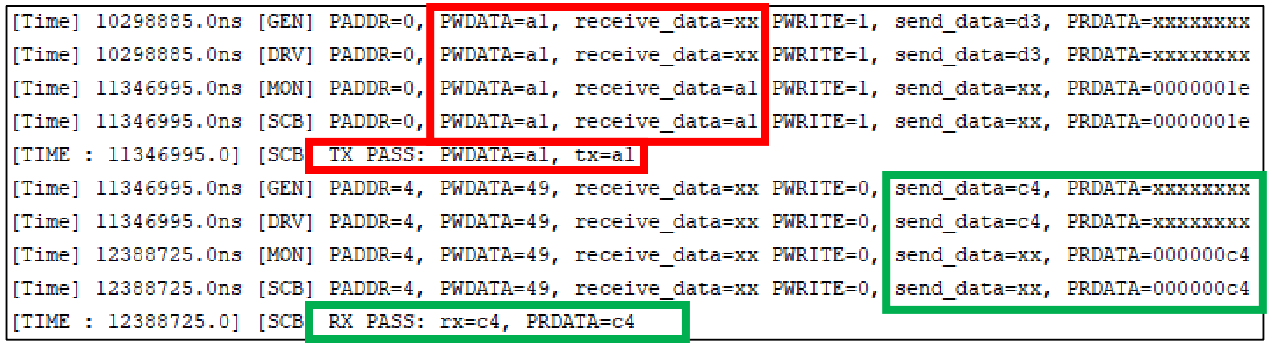
```
class transaction;
    // Random generation
    rand logic [7:0] send_data;     // RX test
    rand logic [7:0] PWDATA;        // TX test
    rand logic       PWRITE;        // 0: RX  1: TX

    // Results from Monitor
    logic [7:0] receive_data;       // TX
    logic [31:0] PRDATA;            // RX

    // Address randomization
    function void post_randomize();
        PADDR = {1'b0, {!PWRITE}, 2'b0};// TX: 0x0, RX: 0x4
    endfunction
endclass
```

Transaction 생성

- send_data와 PWDATA random 생성
- PWRITE 값에 따라 RX, TX 구분



```
class scoreboard;
    int pass_count = 0, fail_count = 0;

    task run();
        forever begin
            gen2scb_mbox.get(gen_tr);
            mon2scb_mbox.get(mon_tr);

            if (mon_tr.PWRITE) begin// TX Test
                if (gen_tr.PWDATA == mon_tr.receive_data) begin
                    pass_count++;
                    $display("[SCB] TX PASS: PWDATA=%h, tx=%h",
                             mon_tr.PWDATA, mon_tr.receive_data);
                end else begin
                    fail_count++;
                    $display("[SCB] TX FAIL: PWDATA=%h, tx=%h",
                             mon_tr.PWDATA, mon_tr.receive_data);
                end
            end else begin // RX Test
                if (gen_tr.send_data == mon_tr.PRDATA[7:0]) begin
                    pass_count++;
                    $display("[SCB] RX PASS: rx=%h, PRDATA=%h",
                             gen_tr.send_data, mon_tr.PRDATA[7:0]);
                end else begin
                    fail_count++;
                    $display("[SCB] RX FAIL: rx=%h, PRDATA=%h",
                             gen_tr.send_data, mon_tr.PRDATA[7:0]);
                end
            end
            ->gen_next_event;
        end
    endtask
endclass
```



```
[Time] 10298885.0ns [GEN] PADDR=0, PWDATA=a1, receive_data=xx PWRITE=1, send_data=d3, PRDATA=xxxxxxxx
[Time] 10298885.0ns [DRV] PADDR=0, PWDATA=a1, receive_data=xx PWRITE=1, send_data=d3, PRDATA=xxxxxxxx
[Time] 11346995.0ns [MON] PADDR=0, PWDATA=a1, receive_data=a1 PWRITE=1, send_data=xx, PRDATA=0000001e
[Time] 11346995.0ns [SCB] PADDR=0, PWDATA=a1, receive_data=a1 PWRITE=1, send_data=xx, PRDATA=0000001e
[TIME : 11346995.0] [SCB  TX PASS: PWDATA=a1, tx=a1
[Time] 11346995.0ns [GEN] PADDR=4, PWDATA=49, receive_data=xx PWRITE=0, send_data=c4, PRDATA=xxxxxxxx
[Time] 11346995.0ns [DRV] PADDR=4, PWDATA=49, receive_data=xx PWRITE=0, send_data=c4, PRDATA=xxxxxxxx
[Time] 12388725.0ns [MON] PADDR=4, PWDATA=49, receive_data=xx PWRITE=0, send_data=xx, PRDATA=000000c4
[Time] 12388725.0ns [SCB] PADDR=4, PWDATA=49, receive_data=xx PWRITE=0, send_data=xx, PRDATA=000000c4
[TIME : 12388725.0] [SCB  RX PASS: rx=c4, PRDATA=c4
```

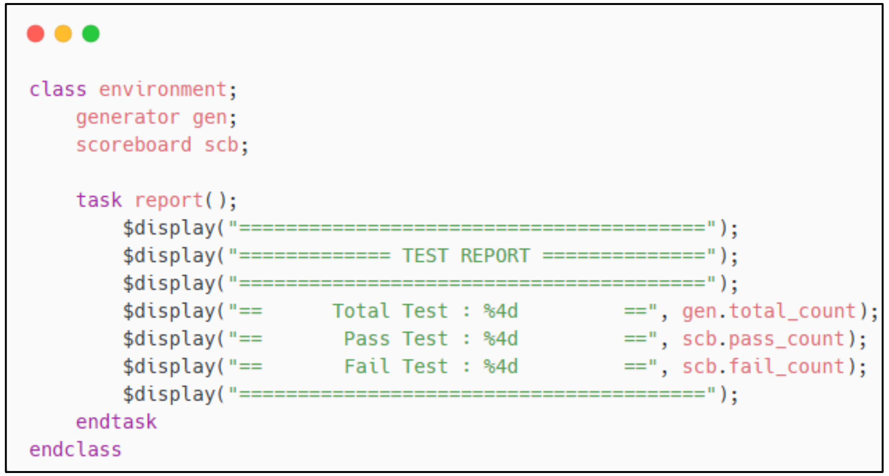
ScoreBoard 값 비교

-TX : PWDATA == RECEIVE DATA

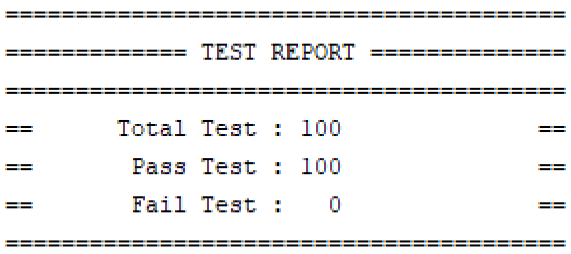
-RX : PRDATA == SEND_DATA

5. UART Peripheral Verification

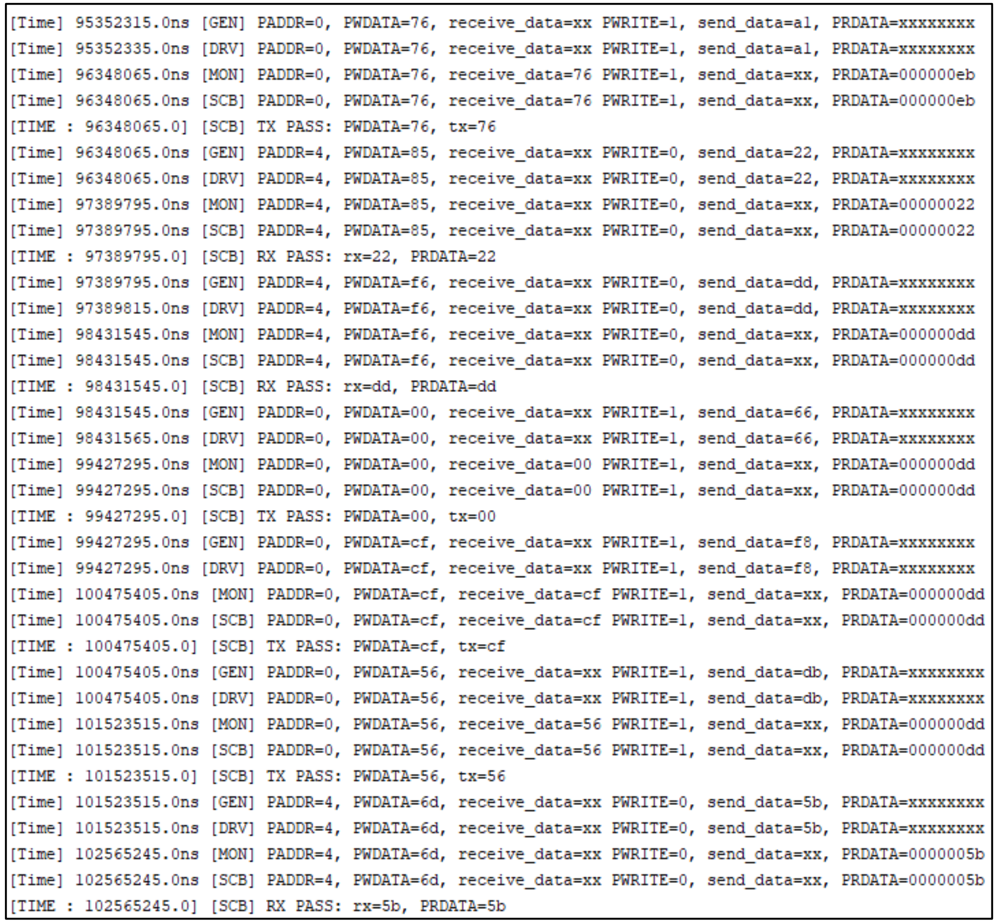
● Verification



```
class environment;
    generator gen;
    scoreboard scb;

    task report();
        $display("========================================");
        $display("============= TEST REPORT ==============");
        $display("========================================");
        $display("==      Total Test : %4d         ==", gen.total_count);
        $display("==       Pass Test : %4d         ==", scb.pass_count);
        $display("==       Fail Test : %4d         ==", scb.fail_count);
        $display("========================================");
    endtask
endclass
```



```
========================================
============= TEST REPORT ==============
========================================
==      Total Test : 100              ==
==       Pass Test : 100              ==
==       Fail Test :   0              ==
========================================
```



```
[Time] 95352315.0ns [GEN] PADDR=0, PWDATA=76, receive_data=xx PWRITE=1, send_data=a1, PRDATA=xxxxxxxx
[Time] 95352335.0ns [DRV] PADDR=0, PWDATA=76, receive_data=xx PWRITE=1, send_data=a1, PRDATA=xxxxxxxx
[Time] 96348065.0ns [MON] PADDR=0, PWDATA=76, receive_data=76 PWRITE=1, send_data=xx, PRDATA=000000eb
[Time] 96348065.0ns [SCB] PADDR=0, PWDATA=76, receive_data=76 PWRITE=1, send_data=xx, PRDATA=000000eb
[TIME : 96348065.0] [SCB] TX PASS: PWDATA=76, tx=76
[Time] 96348065.0ns [GEN] PADDR=4, PWDATA=85, receive_data=xx PWRITE=0, send_data=22, PRDATA=xxxxxxxx
[Time] 96348065.0ns [DRV] PADDR=4, PWDATA=85, receive_data=xx PWRITE=0, send_data=22, PRDATA=xxxxxxxx
[Time] 97389795.0ns [MON] PADDR=4, PWDATA=85, receive_data=xx PWRITE=0, send_data=xx, PRDATA=00000022
[Time] 97389795.0ns [SCB] PADDR=4, PWDATA=85, receive_data=xx PWRITE=0, send_data=xx, PRDATA=00000022
[TIME : 97389795.0] [SCB] RX PASS: rx=22, PRDATA=22
[Time] 97389795.0ns [GEN] PADDR=4, PWDATA=f6, receive_data=xx PWRITE=0, send_data=dd, PRDATA=xxxxxxxx
[Time] 97389815.0ns [DRV] PADDR=4, PWDATA=f6, receive_data=xx PWRITE=0, send_data=dd, PRDATA=xxxxxxxx
[Time] 98431545.0ns [MON] PADDR=4, PWDATA=f6, receive_data=xx PWRITE=0, send_data=xx, PRDATA=000000dd
[Time] 98431545.0ns [SCB] PADDR=4, PWDATA=f6, receive_data=xx PWRITE=0, send_data=xx, PRDATA=000000dd
[TIME : 98431545.0] [SCB] RX PASS: rx=dd, PRDATA=dd
[Time] 98431545.0ns [GEN] PADDR=0, PWDATA=00, receive_data=xx PWRITE=1, send_data=66, PRDATA=xxxxxxxx
[Time] 98431565.0ns [DRV] PADDR=0, PWDATA=00, receive_data=xx PWRITE=1, send_data=66, PRDATA=xxxxxxxx
[Time] 99427295.0ns [MON] PADDR=0, PWDATA=00, receive_data=00 PWRITE=1, send_data=xx, PRDATA=000000dd
[Time] 99427295.0ns [SCB] PADDR=0, PWDATA=00, receive_data=00 PWRITE=1, send_data=xx, PRDATA=000000dd
[TIME : 99427295.0] [SCB] TX PASS: PWDATA=00, tx=00
[Time] 99427295.0ns [GEN] PADDR=0, PWDATA=cf, receive_data=xx PWRITE=1, send_data=f8, PRDATA=xxxxxxxx
[Time] 99427295.0ns [DRV] PADDR=0, PWDATA=cf, receive_data=xx PWRITE=1, send_data=f8, PRDATA=xxxxxxxx
[Time] 100475405.0ns [MON] PADDR=0, PWDATA=cf, receive_data=cf PWRITE=1, send_data=xx, PRDATA=000000dd
[Time] 100475405.0ns [SCB] PADDR=0, PWDATA=cf, receive_data=cf PWRITE=1, send_data=xx, PRDATA=000000dd
[TIME : 100475405.0] [SCB] TX PASS: PWDATA=cf, tx=cf
[Time] 100475405.0ns [GEN] PADDR=0, PWDATA=56, receive_data=xx PWRITE=1, send_data=db, PRDATA=xxxxxxxx
[Time] 100475405.0ns [DRV] PADDR=0, PWDATA=56, receive_data=xx PWRITE=1, send_data=db, PRDATA=xxxxxxxx
[Time] 101523515.0ns [MON] PADDR=0, PWDATA=56, receive_data=56 PWRITE=1, send_data=xx, PRDATA=000000dd
[Time] 101523515.0ns [SCB] PADDR=0, PWDATA=56, receive_data=56 PWRITE=1, send_data=xx, PRDATA=000000dd
[TIME : 101523515.0] [SCB] TX PASS: PWDATA=56, tx=56
[Time] 101523515.0ns [GEN] PADDR=4, PWDATA=6d, receive_data=xx PWRITE=0, send_data=5b, PRDATA=xxxxxxxx
[Time] 101523515.0ns [DRV] PADDR=4, PWDATA=6d, receive_data=xx PWRITE=0, send_data=5b, PRDATA=xxxxxxxx
[Time] 102565245.0ns [MON] PADDR=4, PWDATA=6d, receive_data=xx PWRITE=0, send_data=xx, PRDATA=0000005b
[Time] 102565245.0ns [SCB] PADDR=4, PWDATA=6d, receive_data=xx PWRITE=0, send_data=xx, PRDATA=0000005b
[TIME : 102565245.0] [SCB] RX PASS: rx=5b, PRDATA=5b
```

6. C Application – UART 기반 LED 제어

● UART 기반 LED 제어

UART와 Switch 입력으로 LED 제어하는 Bare-Matal C 어플리케이션

● 핵심 기능

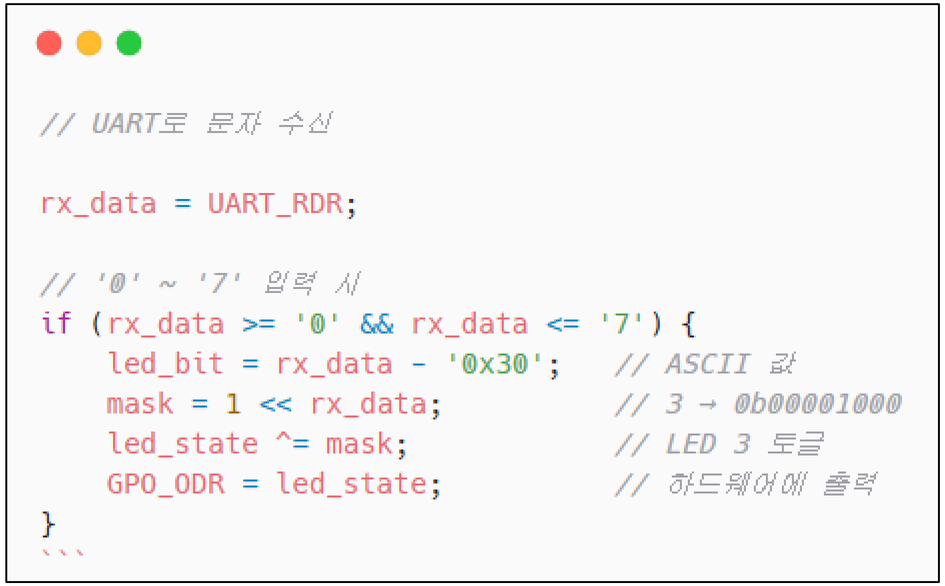
1. UART 명령 처리 (‘0’ ~ ‘7’, ‘A’)
2. Switch 상태 모니터링
3. LED 제어 및 FeedBack

● 기능 구현

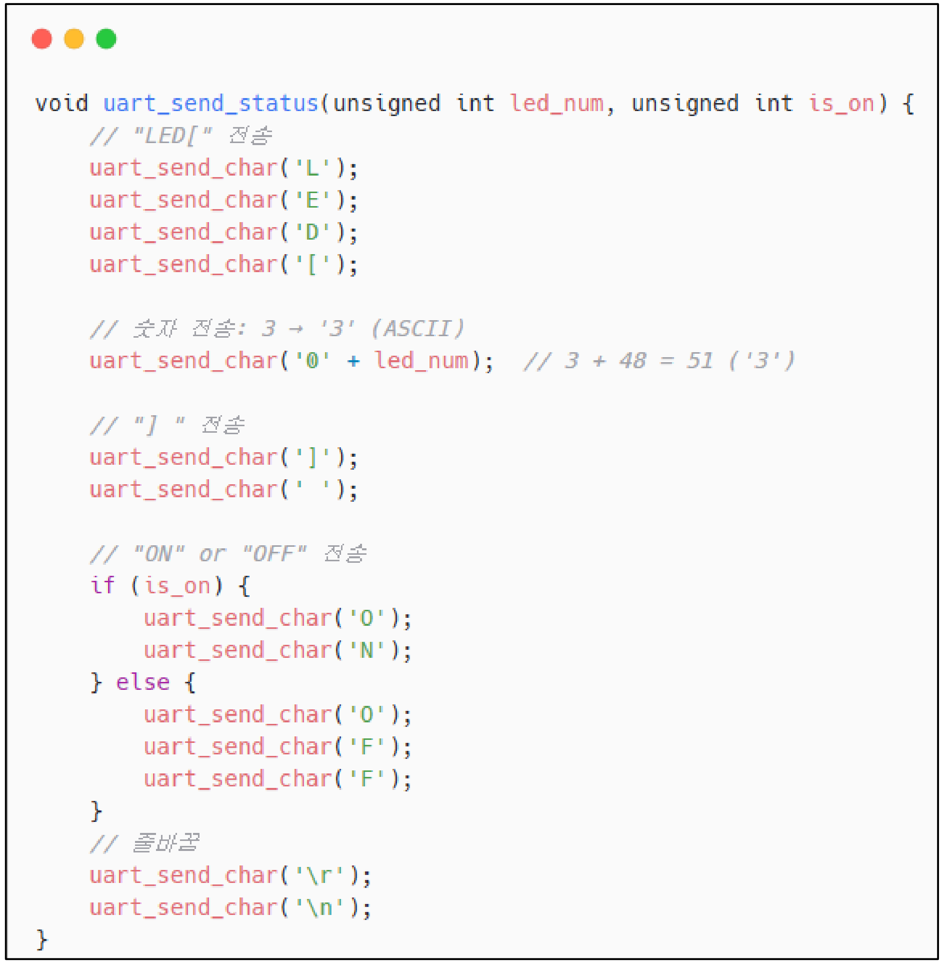
1. 0 ~ 7 입력시 해당 숫자의 LED 토글, ‘A’ 입력시 초기화
2. Switch ON / OFF 시 해당 LED 연동
3. LED 입력시 UART FeedBack   ex) LED[3] ON

6. C Application – UART 기반 LED 제어

● 숫자에 맞는 LED ON/OFF



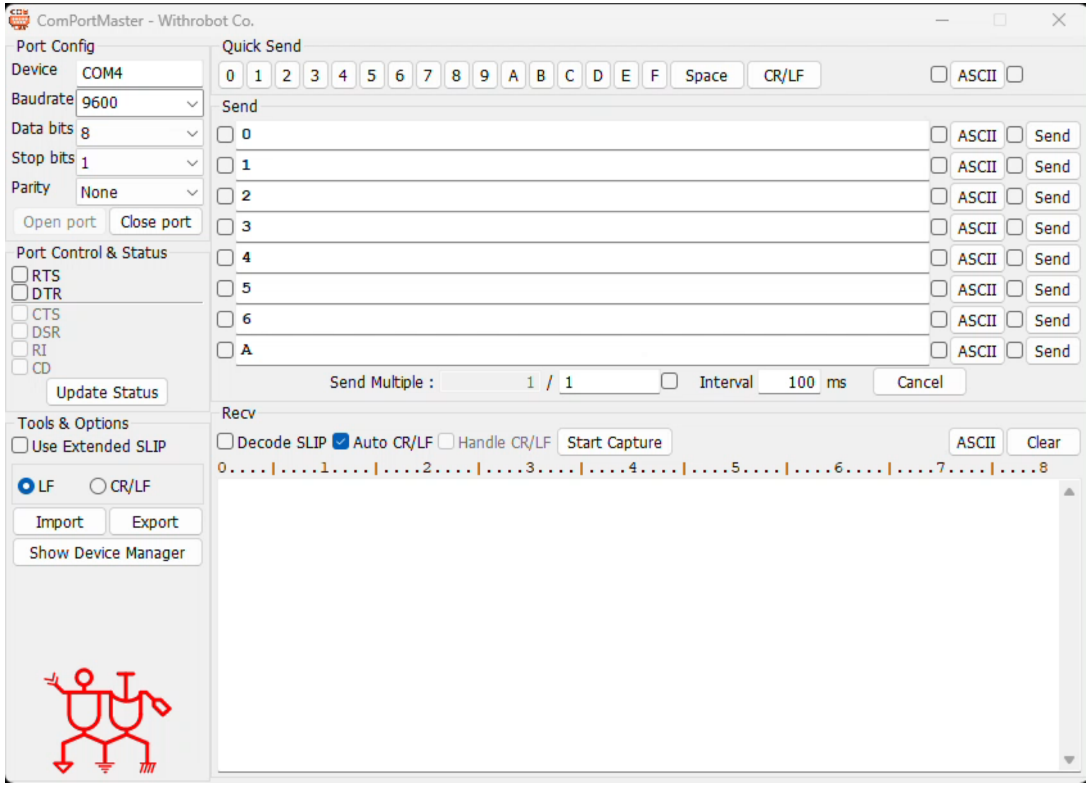
```c
// UART로 문자 수신

rx_data = UART_RDR;

// '0' ~ '7' 입력 시
if (rx_data >= '0' && rx_data <= '7') {
    led_bit = rx_data - '0x30';   // ASCII 값
    mask = 1 << rx_data;          // 3 → 0b00001000
    led_state ^= mask;            // LED 3 토글
    GPO_ODR = led_state;          // 하드웨어에 출력
}
```



```c
void uart_send_status(unsigned int led_num, unsigned int is_on) {
    // "LED[" 전송
    uart_send_char('L');
    uart_send_char('E');
    uart_send_char('D');
    uart_send_char('[');

    // 숫자 전송: 3 → '3' (ASCII)
    uart_send_char('0' + led_num);  // 3 + 48 = 51 ('3')

    // "] " 전송
    uart_send_char(']');
    uart_send_char(' ');

    // "ON" or "OFF" 전송
    if (is_on) {
        uart_send_char('O');
        uart_send_char('N');
    } else {
        uart_send_char('O');
        uart_send_char('F');
        uart_send_char('F');
    }
    // 줄바꿈
    uart_send_char('\r');
    uart_send_char('\n');
}
```

6. C Application – UART 기반 LED 제어

● 동작 영상



ComPortMaster - Withrobot Co.

Port Config

Device: COM4

Baudrate: 9600

Data bits: 8

Stop bits: 1

Parity: None

Open port | Close port

Port Control & Status

□ RTS
□ DTR
□ CTS
□ DSR
□ RI
□ CD

Update Status

Tools & Options

□ Use Extended SLIP

● LF ○ CR/LF

Import | Export

Show Device Manager

Quick Send

0 1 2 3 4 5 6 7 8 9 A B C D E F Space CR/LF □ ASCII □

Send

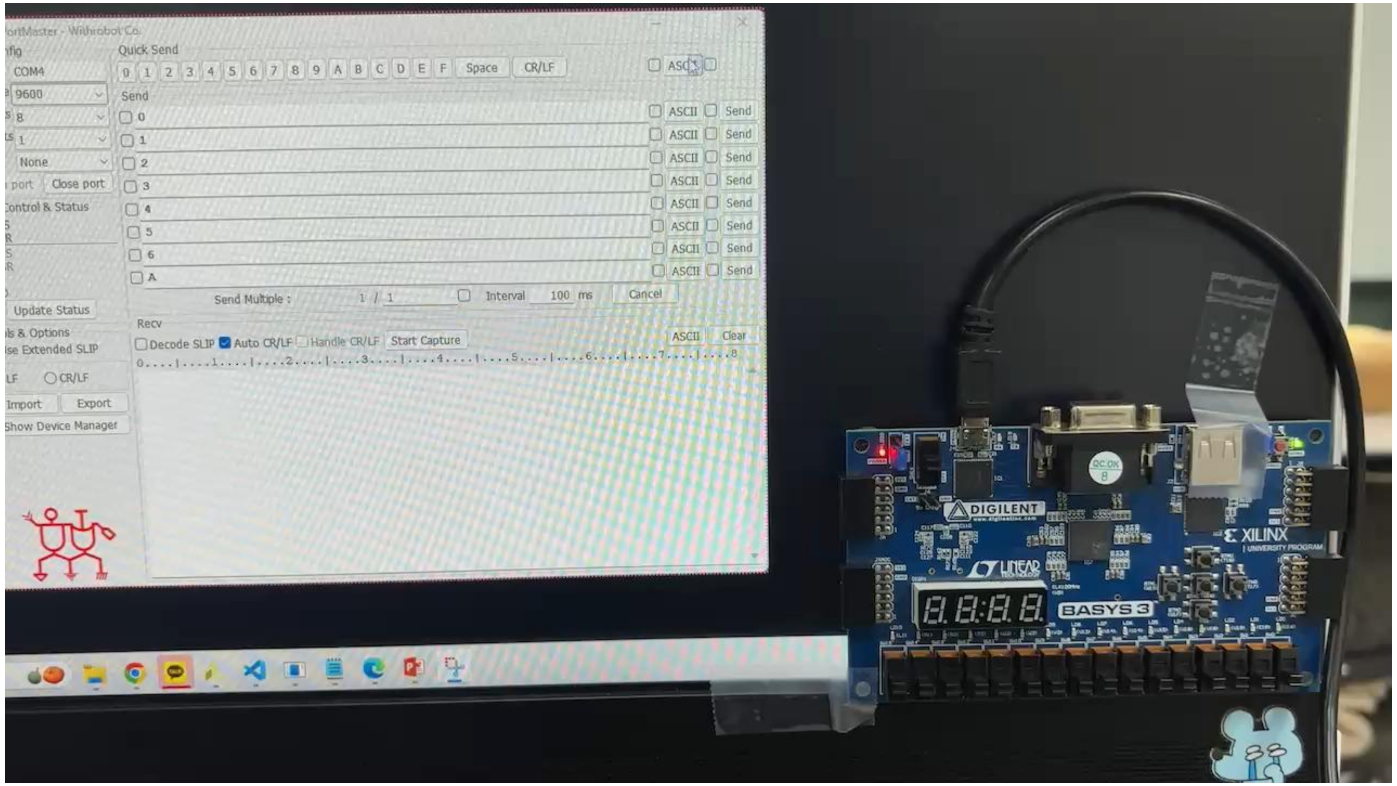
□ 0 □ ASCII □ Send
□ 1 □ ASCII □ Send
□ 2 □ ASCII □ Send
□ 3 □ ASCII □ Send
□ 4 □ ASCII □ Send
□ 5 □ ASCII □ Send
□ 6 □ ASCII □ Send
□ A □ ASCII □ Send

Send Multiple : 1 / 1 □ Interval 100 ms Cancel

Recv

□ Decode SLIP ☑ Auto CR/LF □ Handle CR/LF Start Capture ASCII Clear

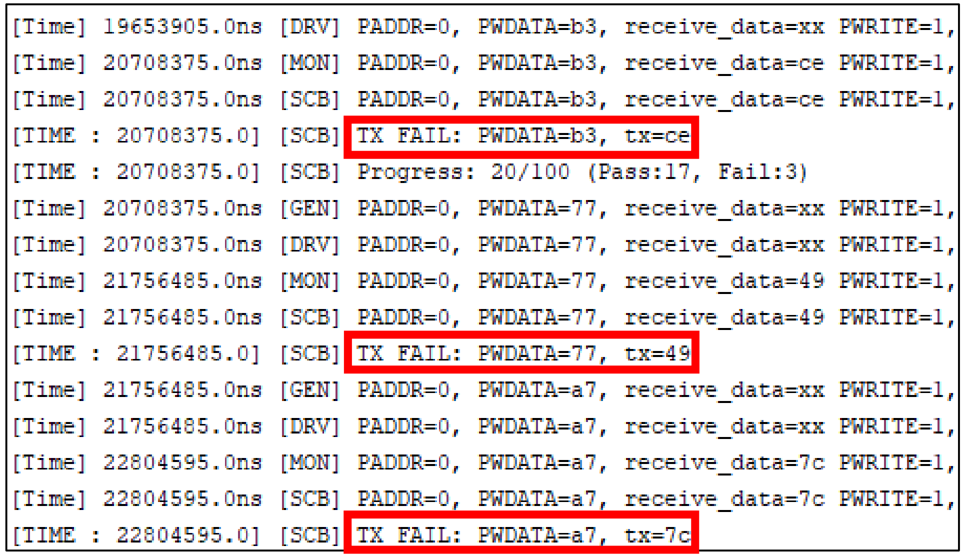
```
0....|....1....|....2....|....3....|....4....|....5....|....6....|....7....|....8
```



7. Trouble Shooting

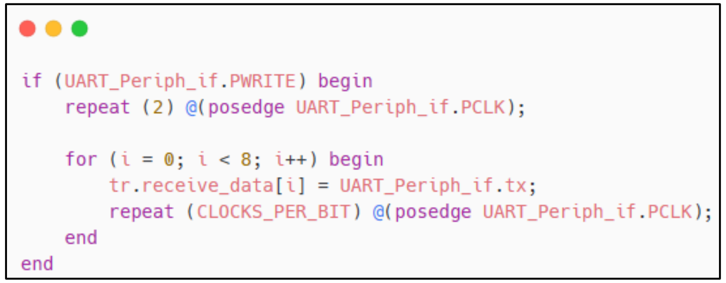
● Verification 시, TX Data Timing 불일치

TX의 PWDATA와 TX 값 불일치

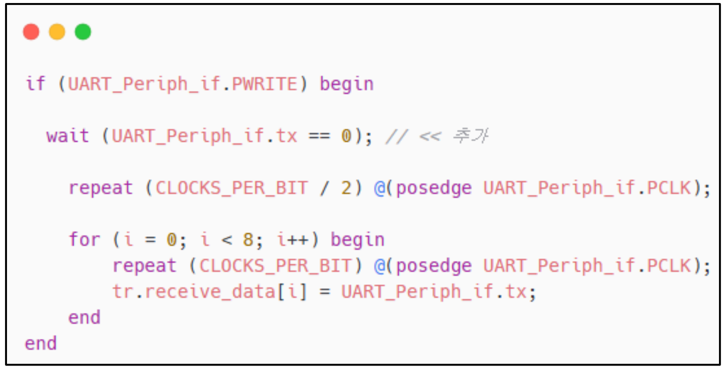


```
[Time] 19653905.0ns [DRV] PADDR=0, PWDATA=b3, receive_data=xx PWRITE=1,
[Time] 20708375.0ns [MON] PADDR=0, PWDATA=b3, receive_data=ce PWRITE=1,
[Time] 20708375.0ns [SCB] PADDR=0, PWDATA=b3, receive_data=ce PWRITE=1,
[TIME : 20708375.0] [SCB] TX FAIL: PWDATA=b3, tx=ce
[TIME : 20708375.0] [SCB] Progress: 20/100 (Pass:17, Fail:3)
[Time] 20708375.0ns [GEN] PADDR=0, PWDATA=77, receive_data=xx PWRITE=1,
[Time] 20708375.0ns [DRV] PADDR=0, PWDATA=77, receive_data=xx PWRITE=1,
[Time] 21756485.0ns [MON] PADDR=0, PWDATA=77, receive_data=49 PWRITE=1,
[Time] 21756485.0ns [SCB] PADDR=0, PWDATA=77, receive_data=49 PWRITE=1,
[TIME : 21756485.0] [SCB] TX FAIL: PWDATA=77, tx=49
[Time] 21756485.0ns [GEN] PADDR=0, PWDATA=a7, receive_data=xx PWRITE=1,
[Time] 21756485.0ns [DRV] PADDR=0, PWDATA=a7, receive_data=xx PWRITE=1,
[Time] 22804595.0ns [MON] PADDR=0, PWDATA=a7, receive_data=7c PWRITE=1,
[Time] 22804595.0ns [SCB] PADDR=0, PWDATA=a7, receive_data=7c PWRITE=1,
[TIME : 22804595.0] [SCB] TX FAIL: PWDATA=a7, tx=7c
```

Wait 조건 추가를 통한 샘플링 정확도 향상

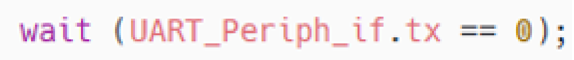


```
if (UART_Periph_if.PWRITE) begin
    repeat (2) @(posedge UART_Periph_if.PCLK);

    for (i = 0; i < 8; i++) begin
        tr.receive_data[i] = UART_Periph_if.tx;
        repeat (CLOCKS_PER_BIT) @(posedge UART_Periph_if.PCLK);
    end
end
```



```
if (UART_Periph_if.PWRITE) begin

  wait (UART_Periph_if.tx == 0); // << 추가

    repeat (CLOCKS_PER_BIT / 2) @(posedge UART_Periph_if.PCLK);

    for (i = 0; i < 8; i++) begin
        repeat (CLOCKS_PER_BIT) @(posedge UART_Periph_if.PCLK);
        tr.receive_data[i] = UART_Periph_if.tx;
    end
end
```

기존 Monitor에선 2clk 대기 후 샘플링



```
wait (UART_Periph_if.tx == 0);
```

추가 후 tx 신호를 실제로 감지 하는 event로 변경