Design Verification Project

SystemVerilog Based
"Counter_10000_UART" Design Verification

하만 세미콘 아카데미 2기 10조 김은성, ㅇㅇㅇ

Contents

1. Introduction

2. DUT Spec Analysis

3. TB Architecture

4. Directed Scenario based Verification Results

5. Trouble Shooting & Debugging

6. Conculsion

1. Introduction

01

Introduction

- **Project Goal**

  - UART 설계를 block-level에서 검증

  - SVA를 활용한 핵심 protocol 검증

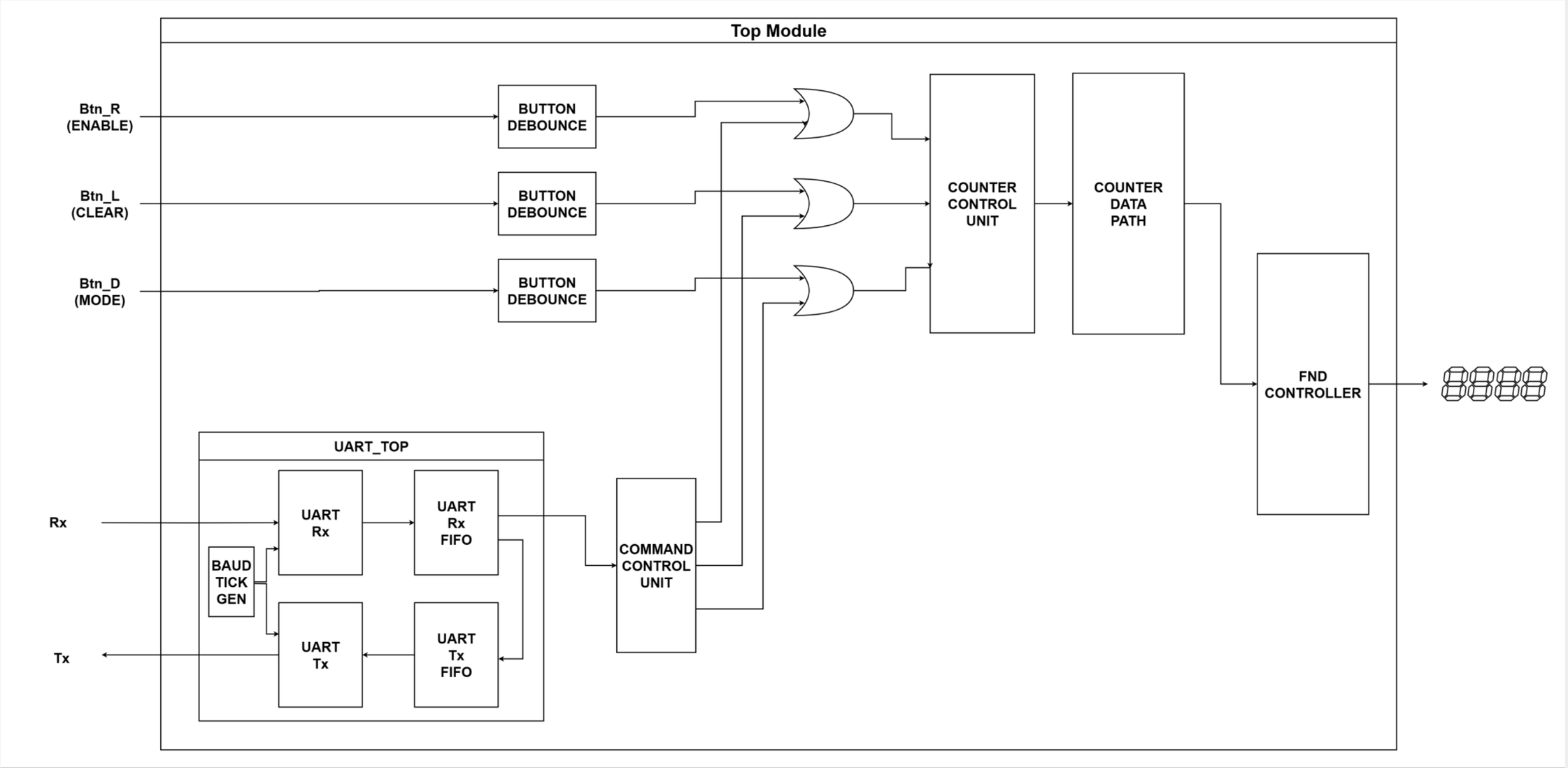
  - SV기반의 모듈화된 TB Architecture 구조를 활용해 **추상화된**, **캡슐화된**, **재사용성**이 높은 검증 환경 구축

2.
DUT Spec Analysis

02

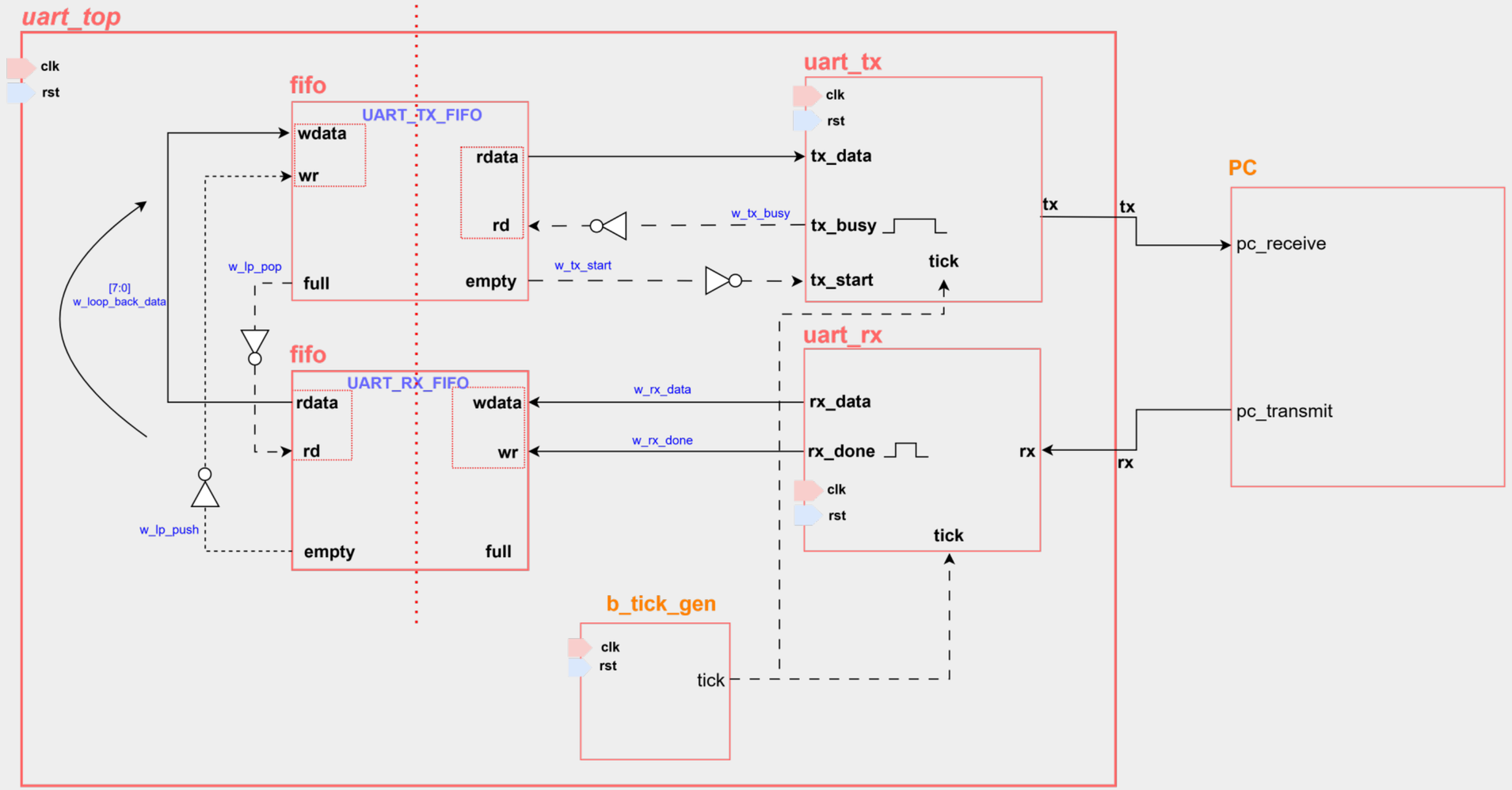
DUT Spec Analysis

- **Overall System Block Diagram**



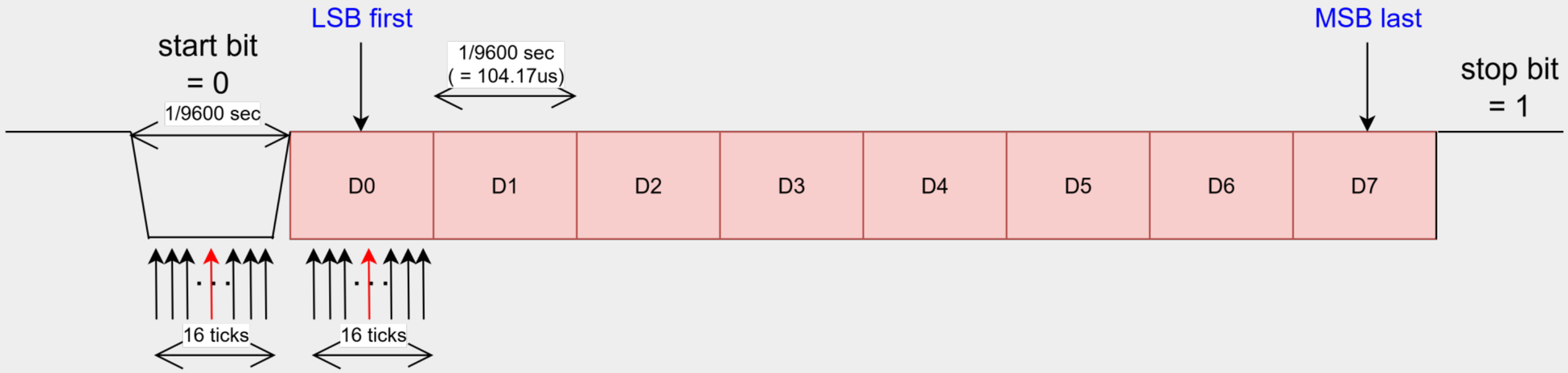
DUT Spec Analysis

- **SV based TB DUT(*UART_FIFO_LOOPBACK*) - System Block Diagram**



DUT Spec Analysis

- **UART Protocol**



UART Protocal

약속된 속도(Baud Rate)로 데이터를
한 번에 한 비트씩 직렬(Serial)로
주고받는 통신 규약

데이터 프레임 구조

Idle : 통신이 없을 때, 라인은 High(1) 상태를 유지

Start Bit: 통신 시작을 알리는 1비트의 Low(0) 신호

Data Bits: 실제 전송할 데이터(8비트), 최하위 비트(LSB)부터 전송

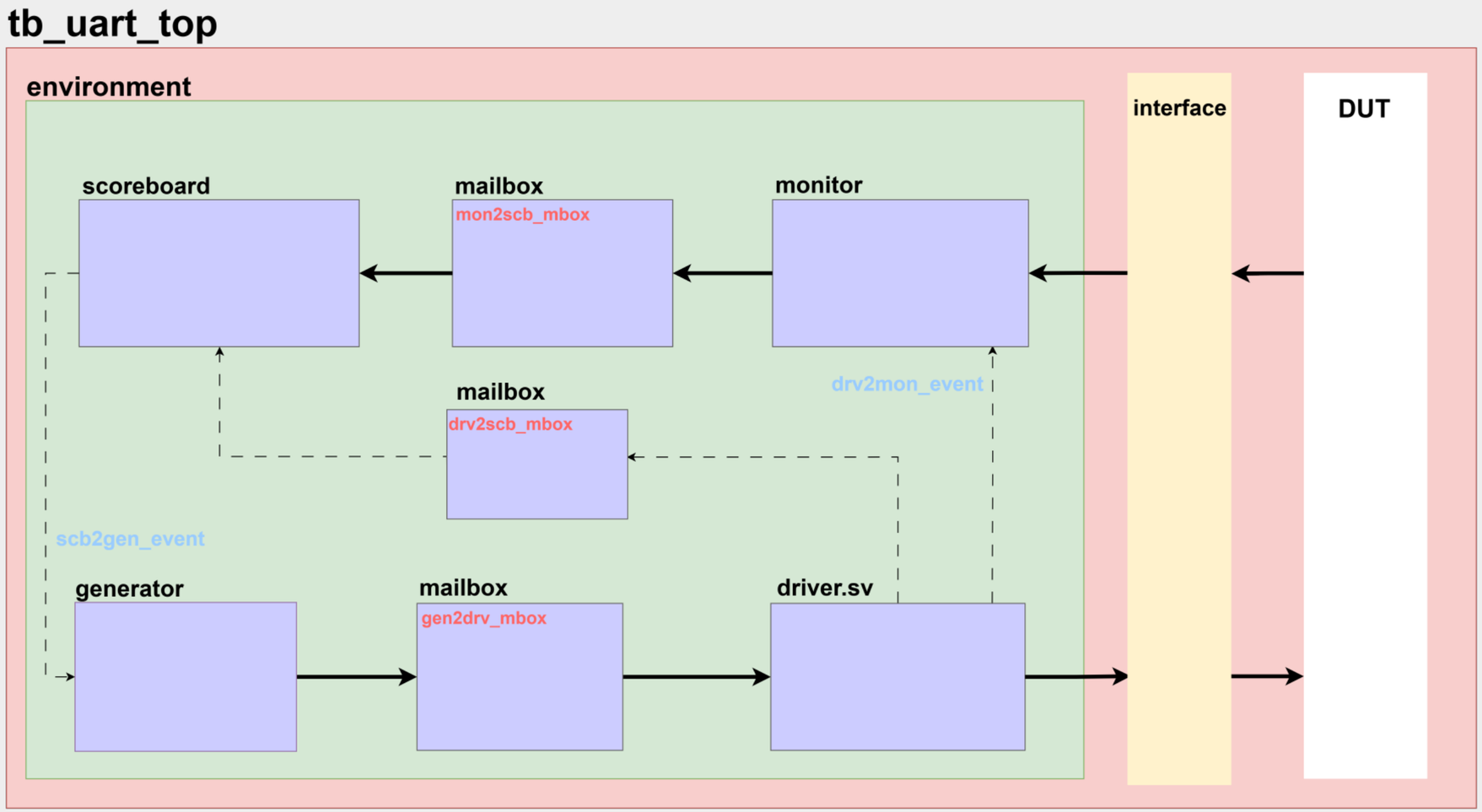
Stop Bit: 통신 종료를 알리는 1비트의 High(1) 신호

3.
TB Architecture

03

TB Architecture

- **TB Architecutre**

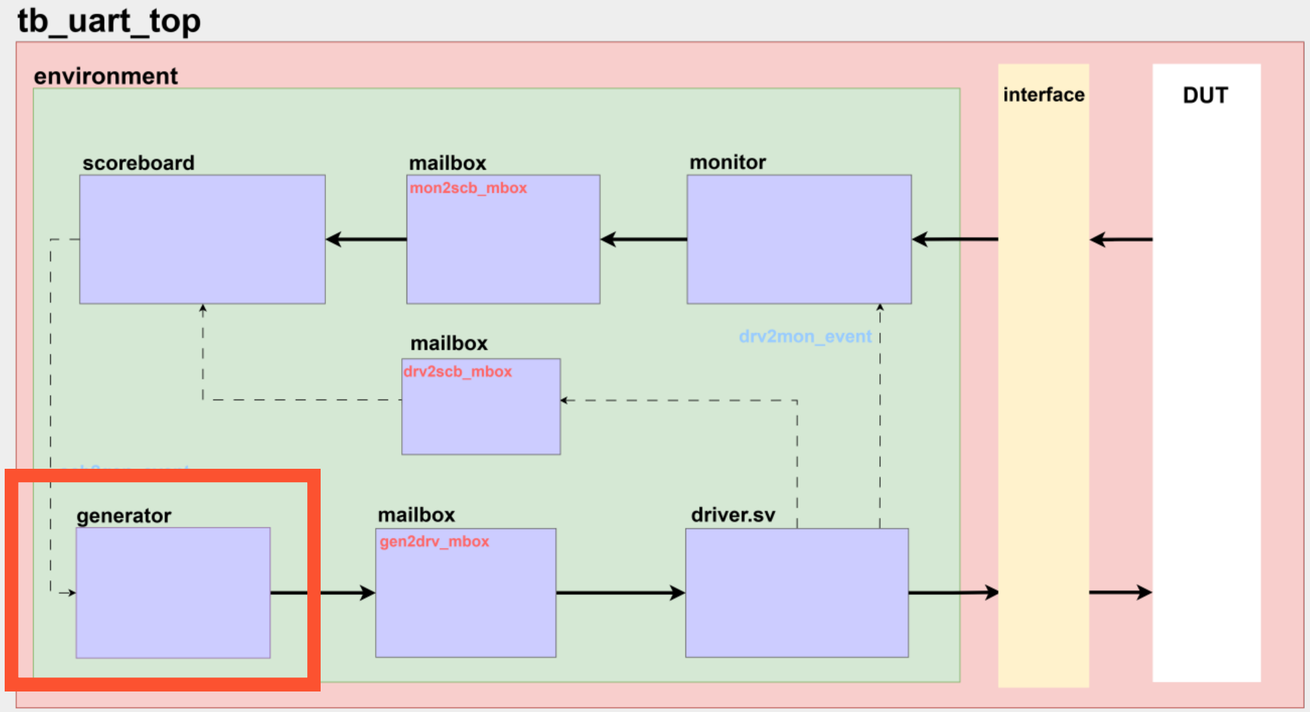


Class 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계

TB Architecture

- **TB Architecutre**

  Clss 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계



Generator: 무작위 데이터와 제어 조건이 담긴 Transaction을 생성

Driver: Transaction을 받아 DUT가 이해할 수 있는 신호로 변환 및 인가

Monitor: DUT의 출력 신호를 감시하여 결과를 Transaction 형태로 수집

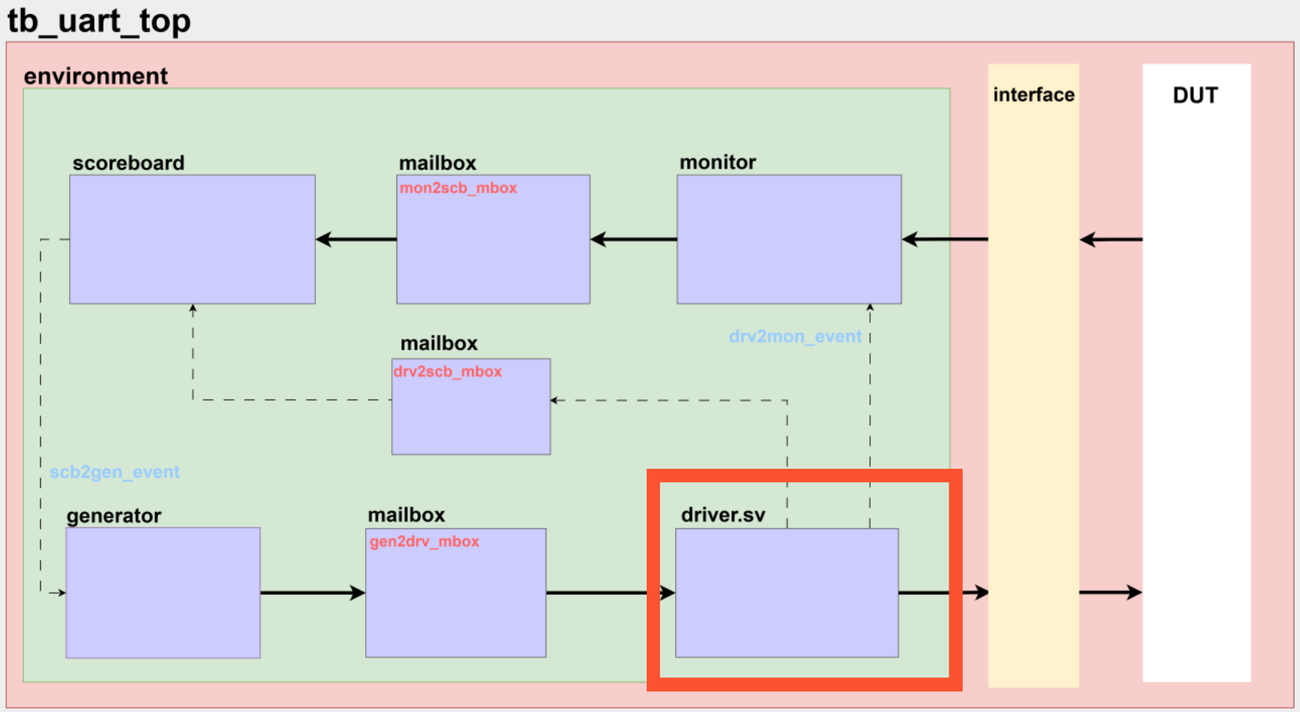
Scoreboard: Generator가 의도한 값과
Monitor가 수집한 실제 결과 값을 비교
→ 자동으로 Pass/Fail을 판정

통신 및 동기화: 컴포넌트 간 데이터 전달 - Mailbox /
동작 타이밍 동기화 - Event

TB Architecture

- **TB Architecutre**

  Clss 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계



Generator: 무작위 데이터와 제어 조건이 담긴 Transaction을 생성

Driver: Transaction을 받아 DUT가 이해할 수 있는 신호로 변환 및 인가

Monitor: DUT의 출력 신호를 감시하여 결과를 Transaction 형태로 수집

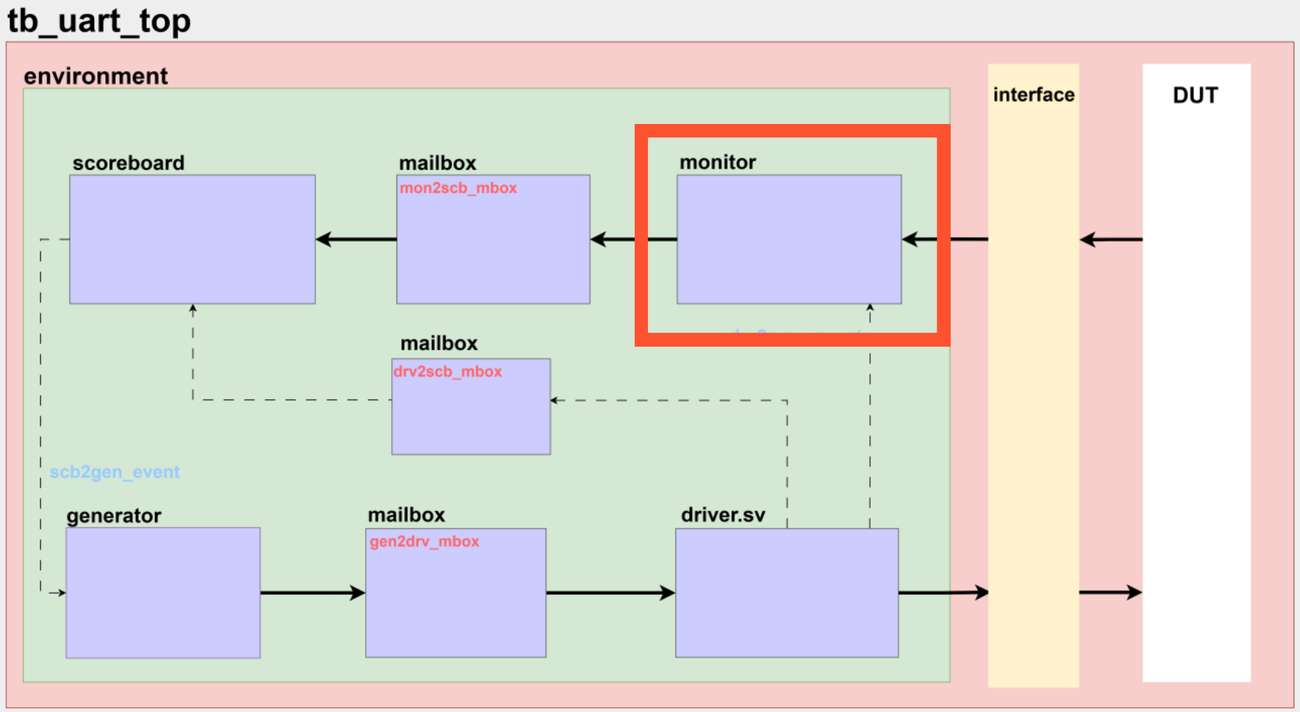
Scoreboard: Generator가 의도한 값과
Monitor가 수집한 실제 결과 값을 비교
→ 자동으로 Pass/Fail을 판정

통신 및 동기화: 컴포넌트 간 데이터 전달 - Mailbox /
동작 타이밍 동기화 - Event

TB Architecture

- **TB Architecutre**

  Clss 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계



Generator: 무작위 데이터와 제어 조건이 담긴 Transaction을 생성

Driver: Transaction을 받아 DUT가 이해할 수 있는 신호로 변환 및 인가

Monitor: DUT의 출력 신호를 감시하여 결과를 Transaction 형태로 수집

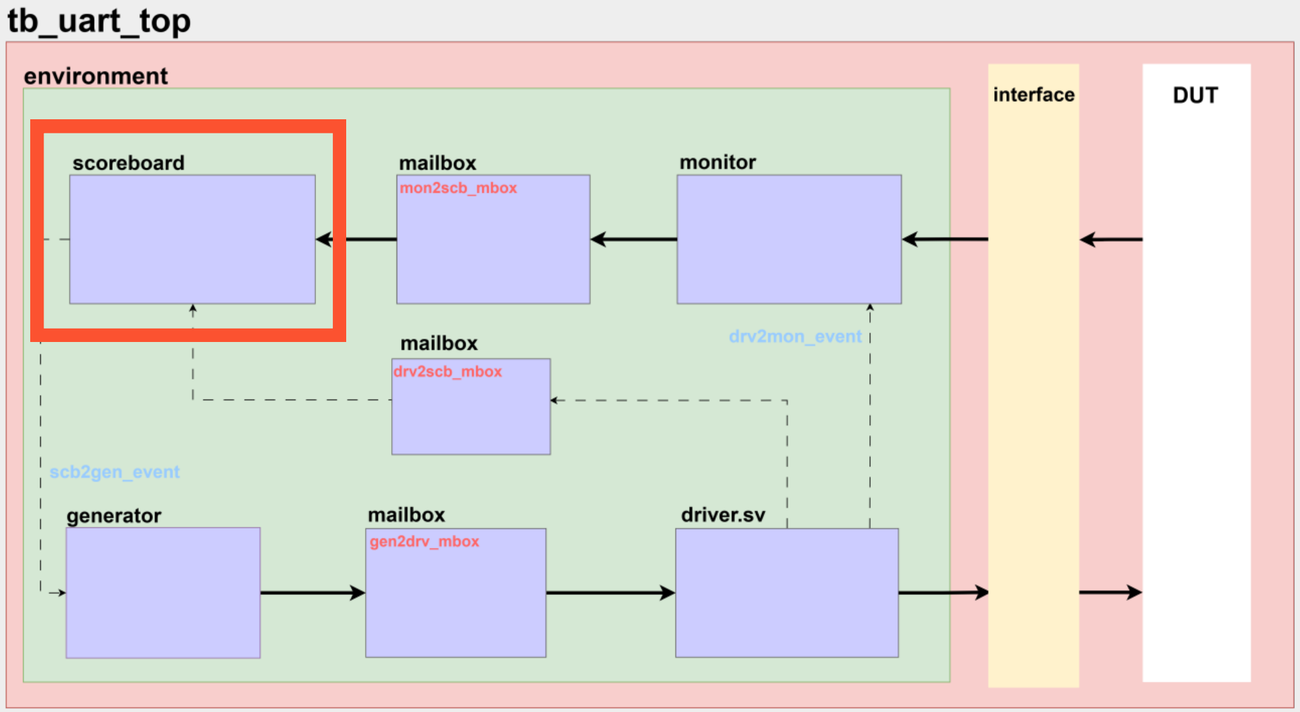
Scoreboard: Generator가 의도한 값과
Monitor가 수집한 실제 결과 값을 비교
→ 자동으로 Pass/Fail을 판정

통신 및 동기화: 컴포넌트 간 데이터 전달 - Mailbox /
동작 타이밍 동기화 - Event

TB Architecture

- **TB Architecutre**

  Clss 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계



Generator: 무작위 데이터와 제어 조건이 담긴 Transaction을 생성

Driver: Transaction을 받아 DUT가 이해할 수 있는 신호로 변환 및 인가

Monitor: DUT의 출력 신호를 감시하여 결과를 Transaction 형태로 수집

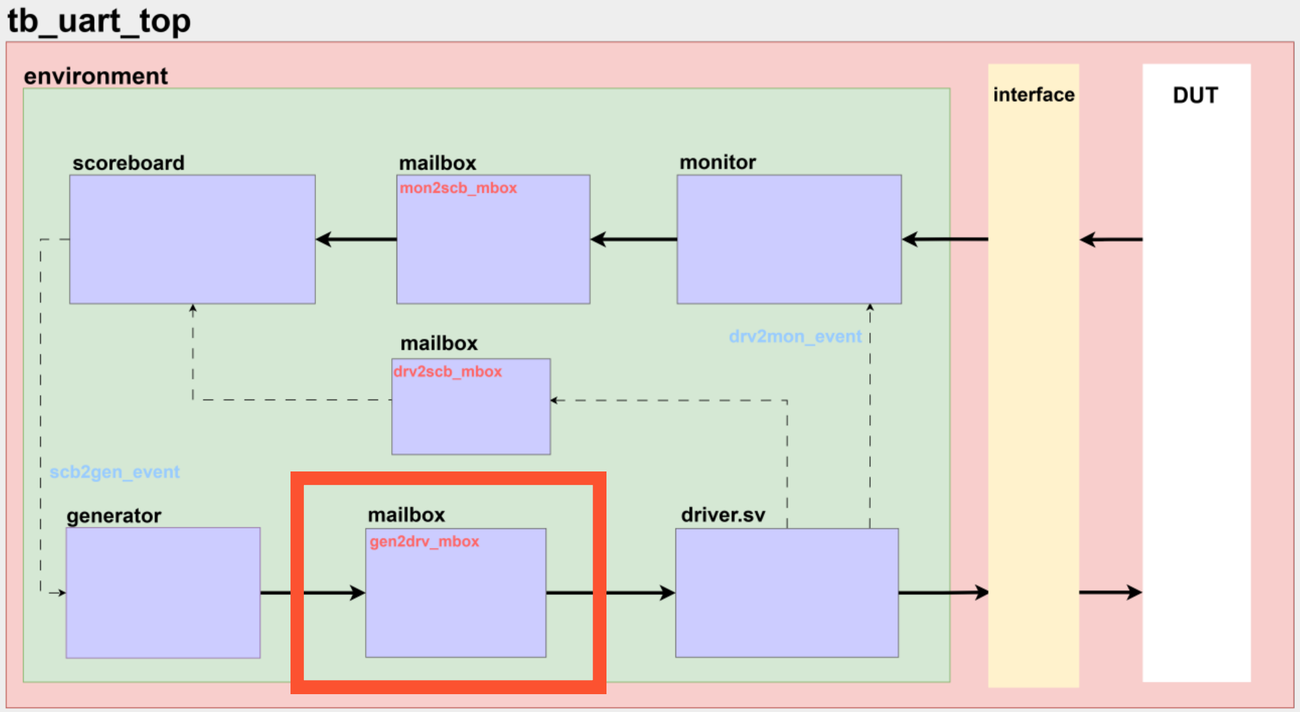
Scoreboard: Generator가 의도한 값과
Monitor가 수집한 실제 결과 값을 비교
→ 자동으로 Pass/Fail을 판정

통신 및 동기화: 컴포넌트 간 데이터 전달 - Mailbox /
동작 타이밍 동기화 - Event

TB Architecture

- **TB Architecutre**

  Clss 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계



Generator: 무작위 데이터와 제어 조건이 담긴 Transaction을 생성

Driver: Transaction을 받아 DUT가 이해할 수 있는 신호로 변환 및 인가

Monitor: DUT의 출력 신호를 감시하여 결과를 Transaction 형태로 수집

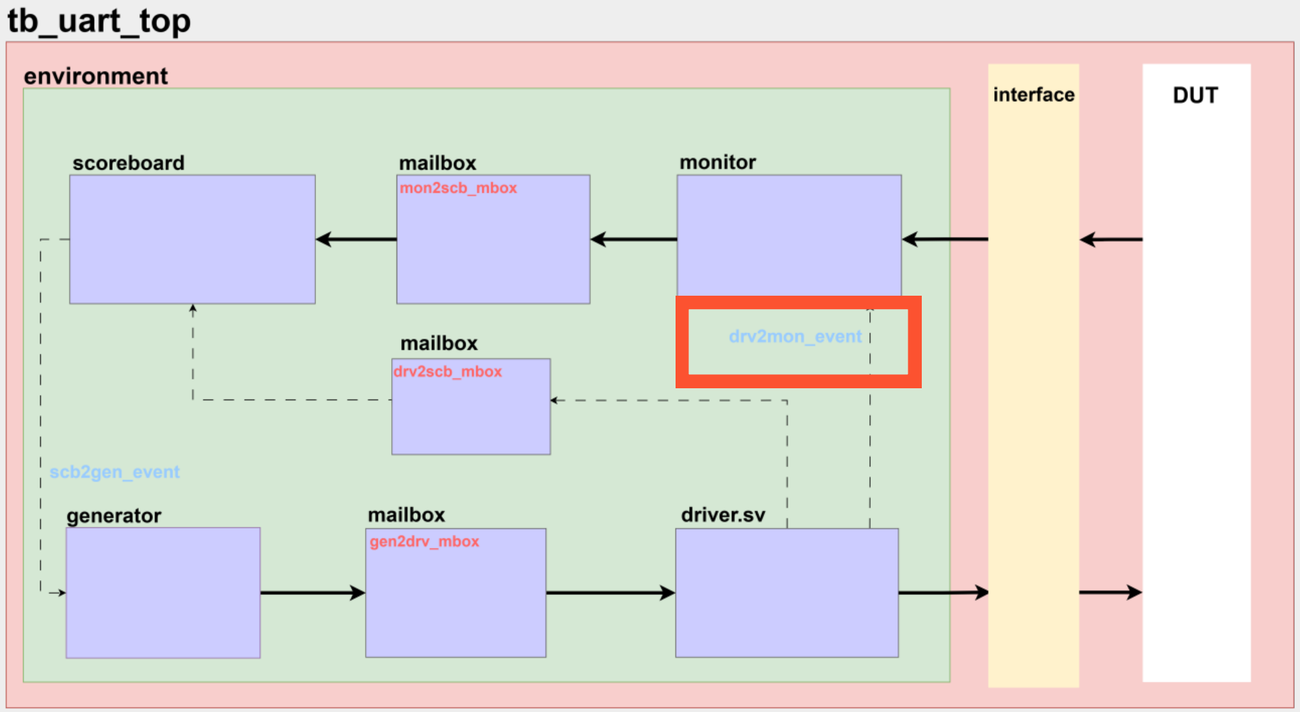
Scoreboard: Generator가 의도한 값과
Monitor가 수집한 실제 결과 값을 비교
→ 자동으로 Pass/Fail을 판정

통신 및 동기화: 컴포넌트 간 데이터 전달 - Mailbox /
동작 타이밍 동기화 - Event

TB Architecture

- **TB Architecutre**

  Clss 기반 구조: UART 검증을 위해 객체 지향 프로그래밍(OOP) 개념을 적용한 테스트벤치를 설계



Generator: 무작위 데이터와 제어 조건이 담긴 Transaction을 생성

Driver: Transaction을 받아 DUT가 이해할 수 있는 신호로 변환 및 인가

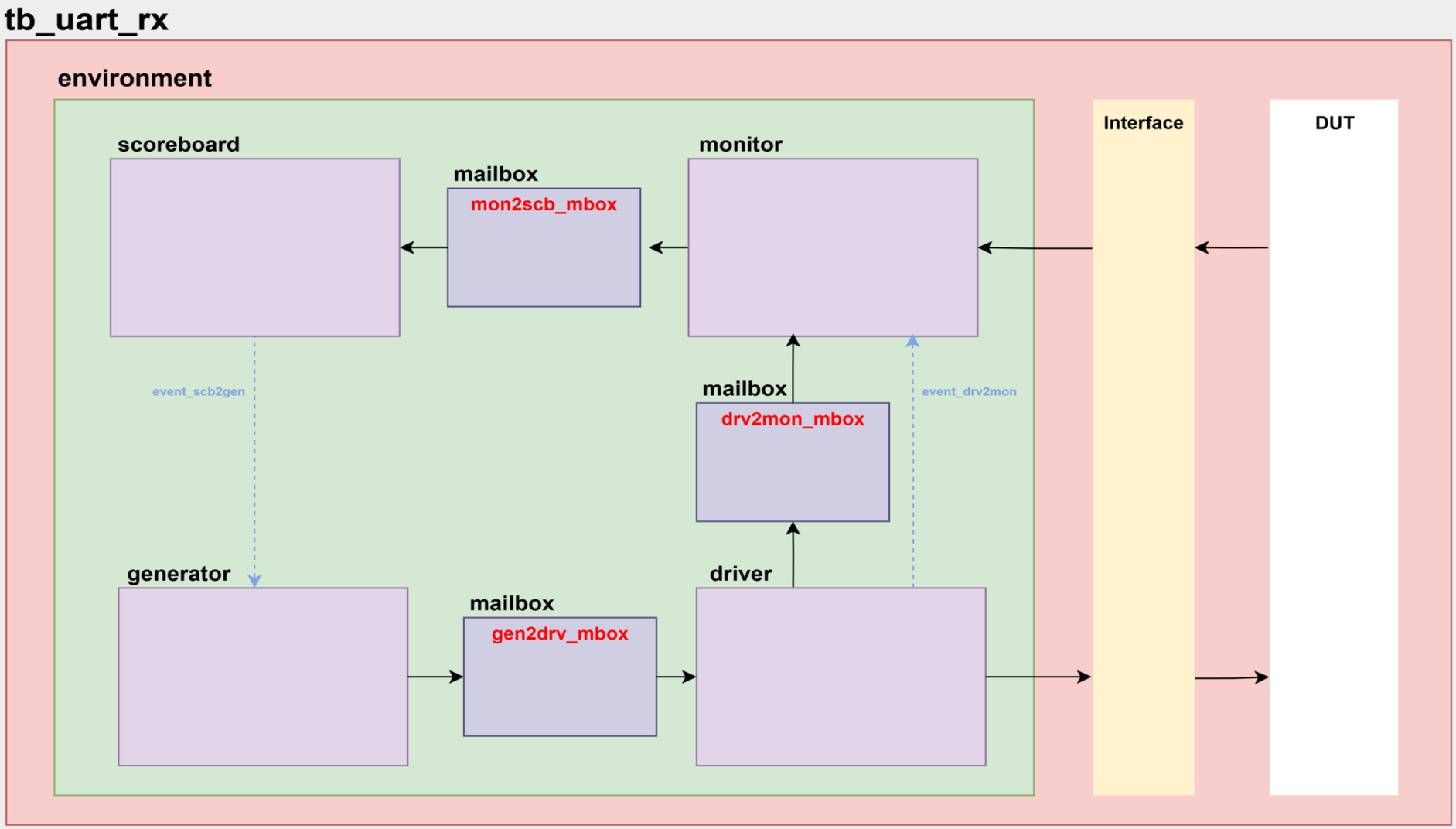
Monitor: DUT의 출력 신호를 감시하여 결과를 Transaction 형태로 수집

Scoreboard: Generator가 의도한 값과
Monitor가 수집한 실제 결과 값을 비교
→ 자동으로 Pass/Fail을 판정

통신 및 동기화: 컴포넌트 간 데이터 전달 - Mailbox /
동작 타이밍 동기화 - Event

TB Architecture

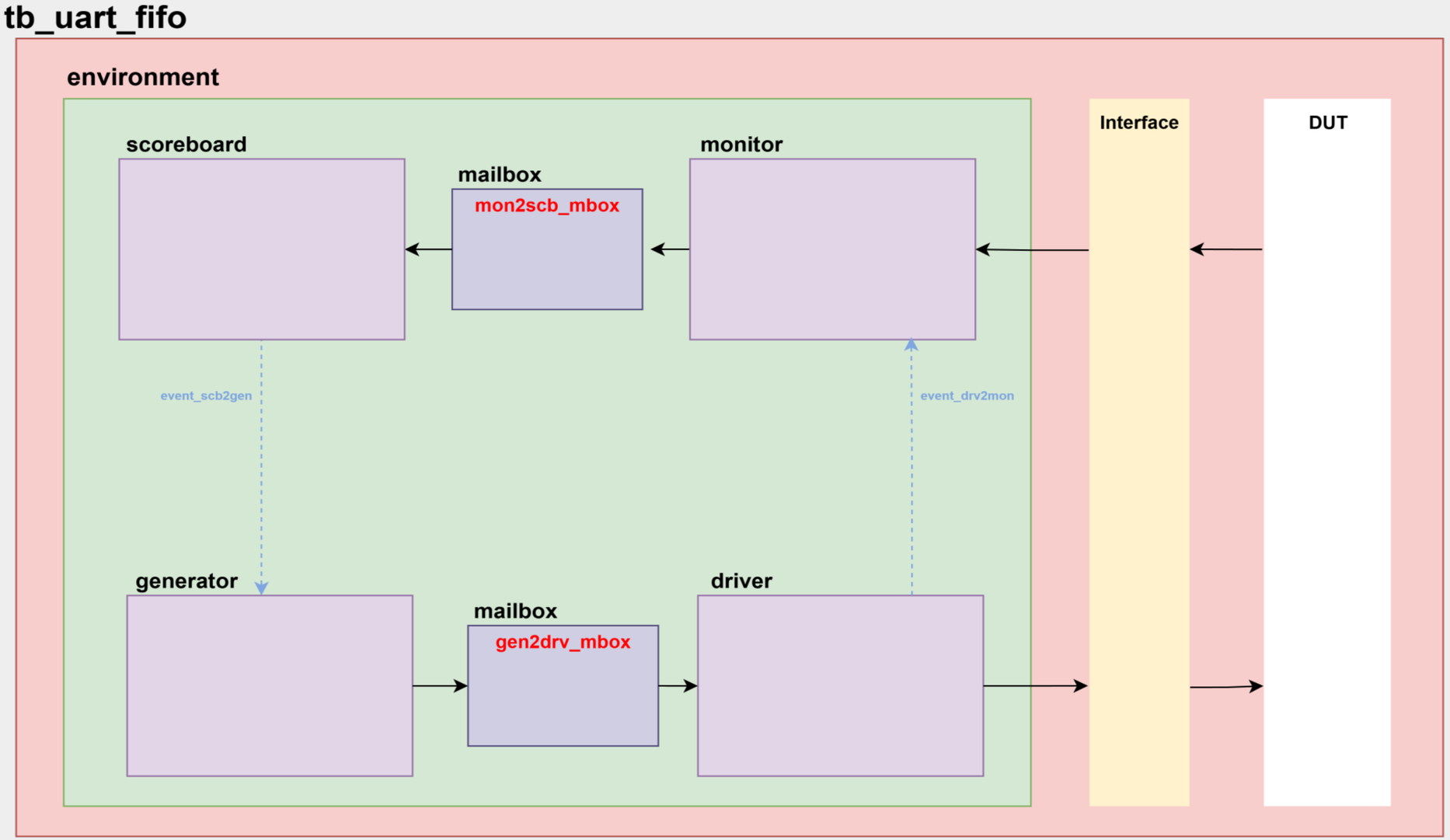
- **TB Architecutre**



UART 모듈별 검증에 각각 맞게 조금씩 변형하여 적용

TB Architecture

- **TB Architecutre**



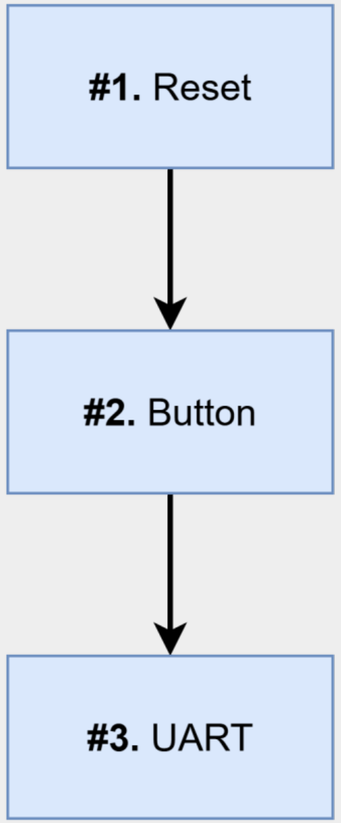
UART 모듈별 검증에 각각 맞게 조금씩 변형하여 적용

4.
Directed Scenario
&
Verification Results

04

Directed Scenario & Verification Results

1. COUNTER Top Simulation



Scenario		Base Code	Check Point

#1	Reset	`initial begin`
		`clk = 0;`
		`rst = 1;`
		`rx = 1;`
		`mode = 0;`
		`enable = 0;`
		`clear = 0;`
			Intialization

#2	BUTTON	`task press_button;`	btn_enable : counter run ↔ stop btn_mode : up mode ↔ down mode btn_clear : counter 0으로 초기화
#3	UART	`task send_uart_byte;`	uart 'r' command : counter run ↔ stop uart 'm' command : up mode ↔ down mode uart 'c' command : counter 0으로 초기화

Directed Scenario & Verification Results

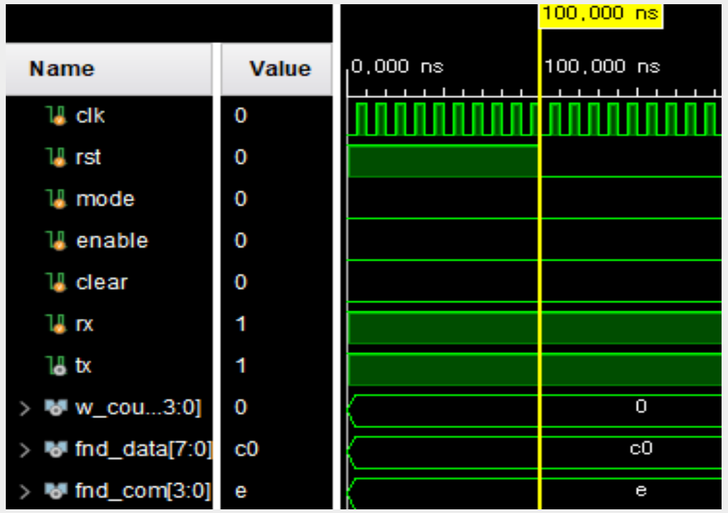
1. COUNTER Top Simulation - Waveform & Report

Log, Report

Waveform

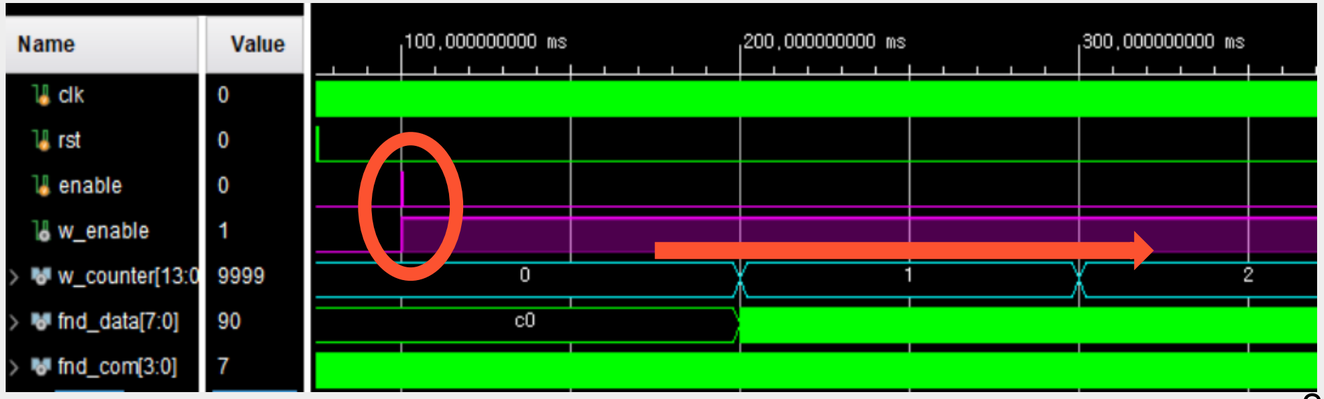
#1. Reset

```
=== Counter Top Testbench Start ===
Time:                 0 - Counter value:     0, Mode:   UP, Enable: OFF
Time:            100000 - Reset
```



#2. btn_enable (run)

```
=== Test 1: Physical Button Controls ===
Time:        100000100000 - Test ENABLE button (start)
Time:        100000135000 - Button ENABLE command detected
Time:        200000155000   Counter value:     1, Mode:   UP, Enable:  0
Time:        300000155000   Counter value:     2, Mode:   UP, Enable:  0
```



Directed Scenario & Verification Results

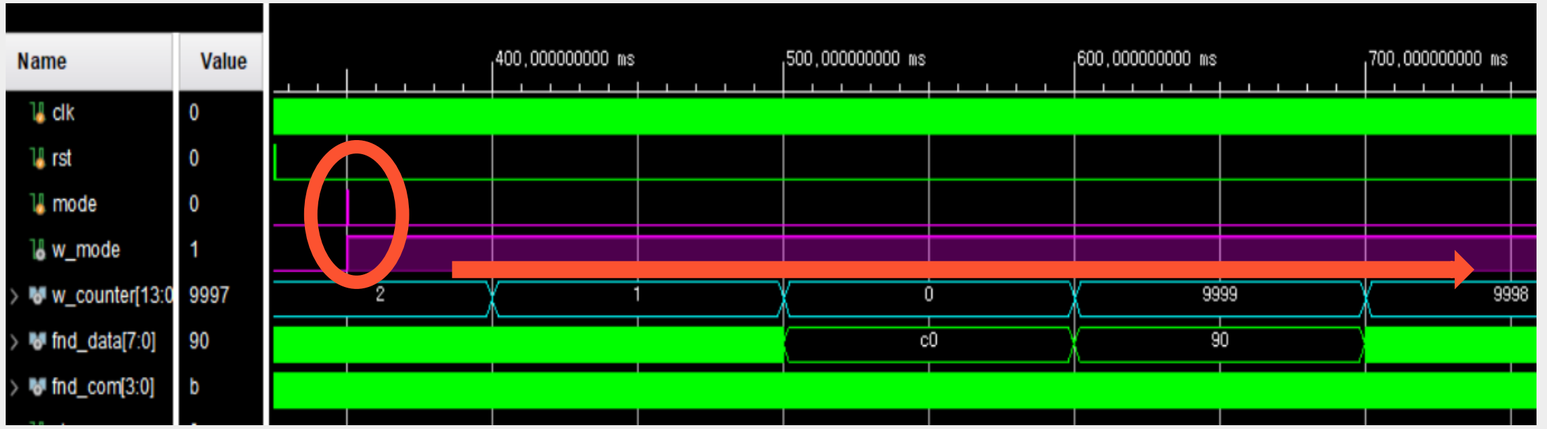
1. COUNTER Top Simulation - Waveform & Report

Log, Report

Waveform

#2. btn_mode (UP → DOWN)

```
Time:        350008100000 - Test MODE button
Time:        350008135000 - Button MODE command detected
Time:        400000155000 - Counter value:    1, Mode: DOWN, Enable:  ON
Time:        500000155000 - Counter value:    0, Mode: DOWN, Enable:  ON
Time:        600000155000 - Counter value: 9999, Mode: DOWN, Enable:  ON
Time:        700000155000 - Counter value: 9998, Mode: DOWN, Enable:  ON
Time:        800000155000 - Counter value: 9997, Mode: DOWN, Enable:  ON
```



#2. btn_enable (stop)

```
Time:        800016100000 - Test ENABLE button (stop)
Time:        800016135000 - Button ENABLE command detected
```

Directed Scenario & Verification Results

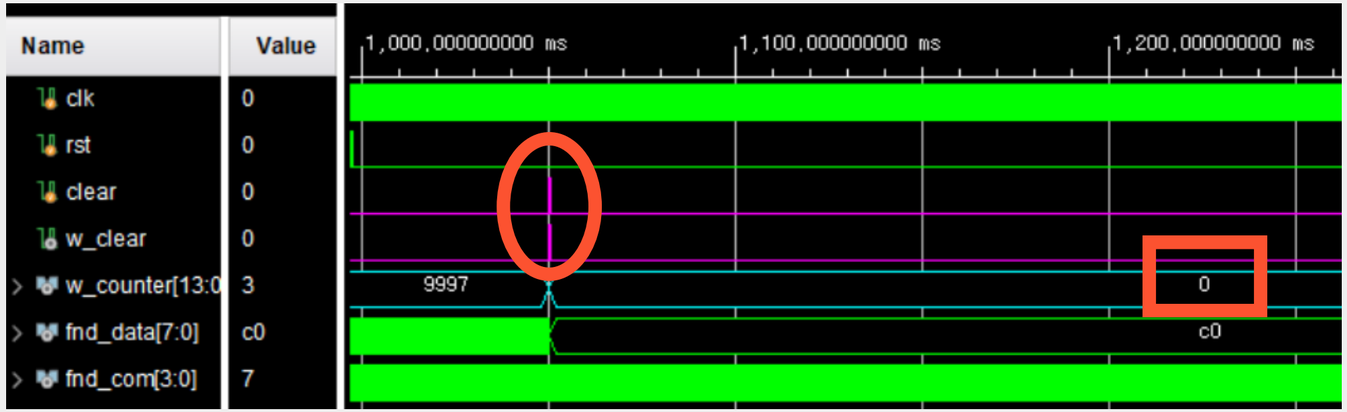
1. COUNTER Top Simulation - Waveform & Report

Log, Report

Waveform

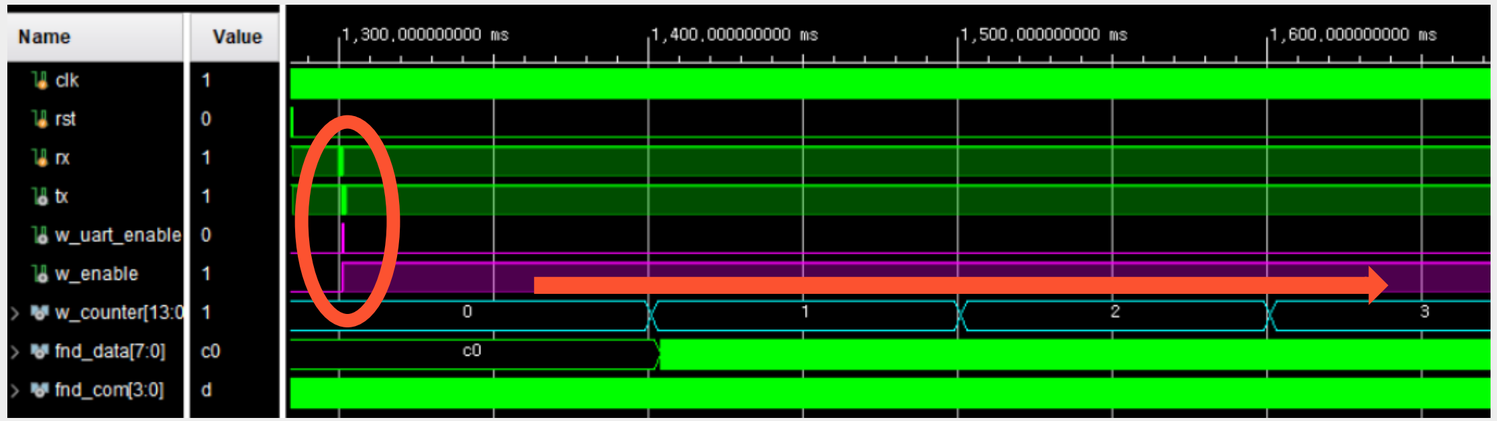
#2. btn_clear

```
Time:        1050024100000 - Test CLEAR button
Time:        1050024135000 - Button CLEAR command detected
Time:        1050024155000 - Counter value:     0,  ode: DOWN, Enable: OFF
```



#3. uart_enable (run)

```
=== Test 2: UART Command Controls ===
Time:        1300040100000 - Sending UART 'r' (start)
Time:        1301127775000 - UART RUN command detected
Time:        1401127795000 - Counter value:     1, Mode:   UP, Enable:  ON
Time:        1501127795000 - Counter value:     2, Mode:   UP, Enable:  ON
Time:        1601127795000 - Counter value:     3, Mode:   UP, Enable:  ON
```



Directed Scenario & Verification Results

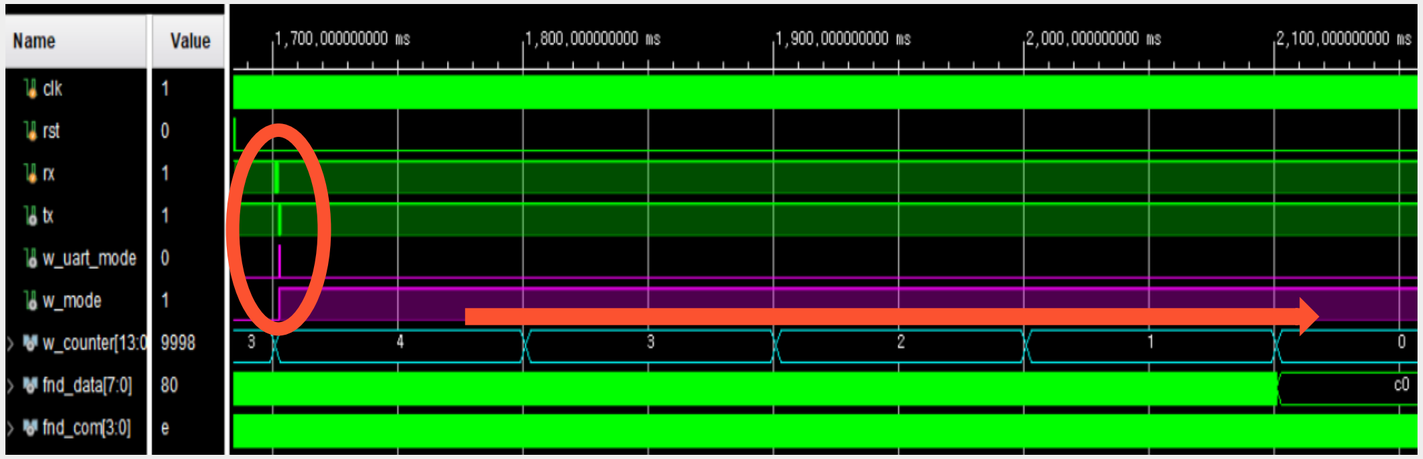
1. COUNTER Top Simulation - Waveform & Report

Log, Report

Waveform

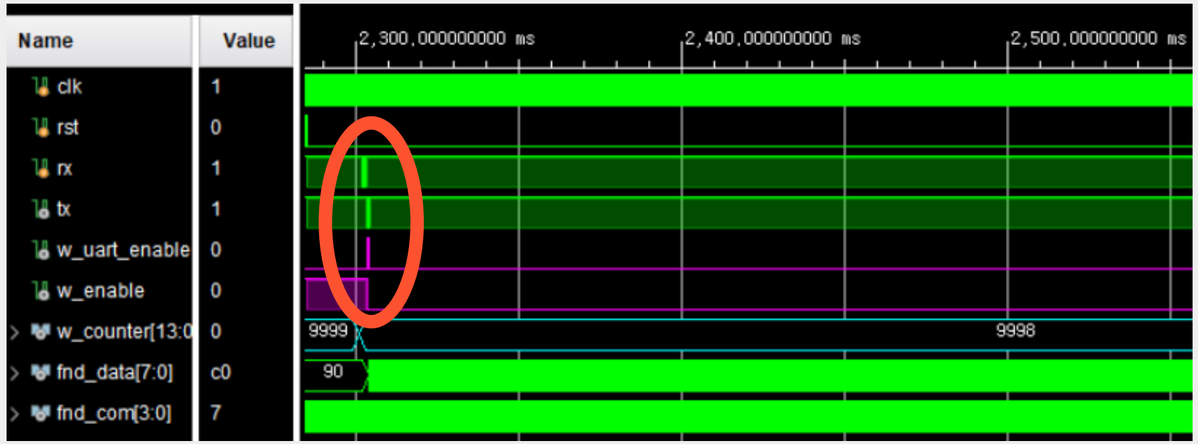
#3. uart_mode (UP → DOWN)

```
Time:        1701081760000 - Sending UART 'm'
Time:        1701127795000 - Counter value:     4, Mode:   UP, Enable:  ON
Time:        1702169815000 - UART MODE command detected
Time:        1801127795000 - Counter value:     3, Mode: DOWN, Enable:  ON
Time:        1901127795000 - Counter value:     2, Mode: DOWN, Enable:  ON
Time:        2001127795000 - Counter value:     1, Mode: DOWN, Enable:  ON
Time:        2101127795000 - Counter value:     0, Mode: DOWN, Enable:  ON
Time:        2201127795000 - Counter value:  9999, Mode: DOWN, Enable:  ON
Time:        2301127795000 - Counter value:  9998, Mode: DOWN, Enable:  ON
```



#3. uart_enable (stop)

```
Time:        2302123420000 - Sending UART 'r' (stop)
Time:        2303212075000 - UART RUN command detected
```



Directed Scenario & Verification Results

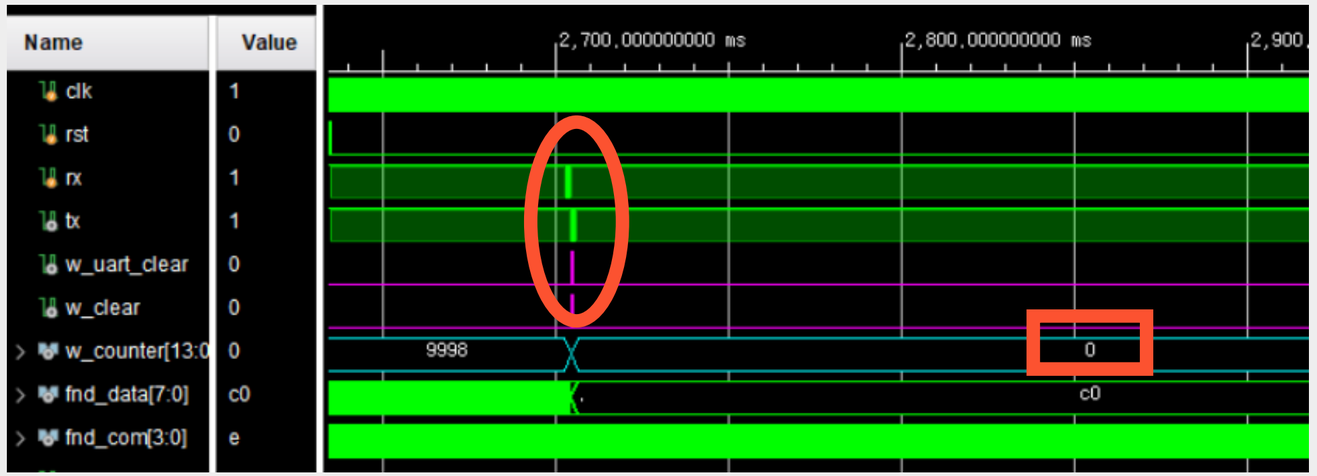
1. COUNTER Top Simulation - Waveform & Report

Log, Report

Waveform

#3. uart_clear

```
Time:        2703165080000 - Sending UART 'c'
Time:        2704254115000 - UART CLEAR command detected
Time:        2704254135000  Counter value:    0  Mode: DOWN, Enable: OFF
```

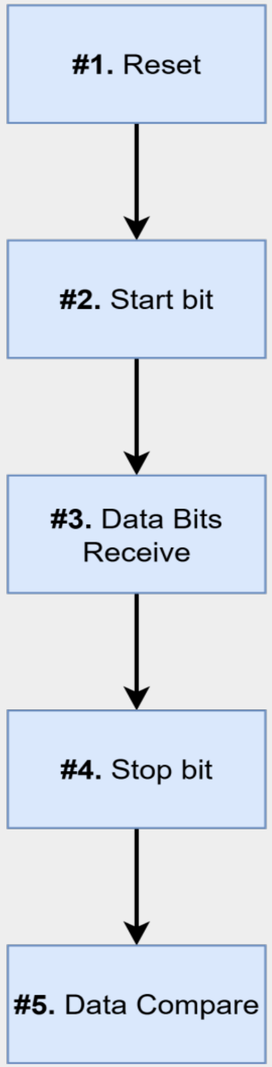


test finish

```
=== Test Complete ===
Time:        3004206740000 - Testbench finished
```

Directed Scenario & Verification Results

2. UART_RX - Test Scenario



Scenario		Base Code	Check Point
#1	Reset	`drv.reset();`	Intialization
#2	Start bit	`uart_if.rx = 1'b0;` `#(BIT_PERIOD);`	rx의 falling edge를 감지 / Start bit : 비트 주기의 중앙에서 샘플링

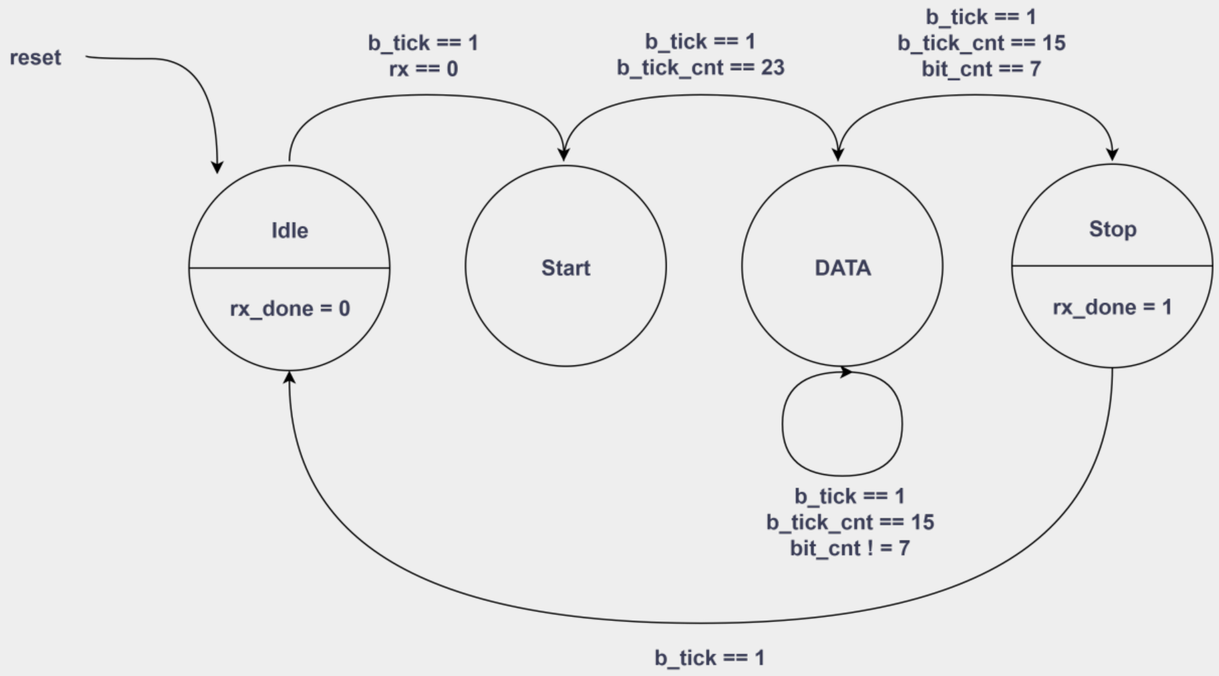
#3	Data Bits Receive	`for (i = 0; i < 8; i = i + 1) begin`
		`  uart_if.rx = expected_data[i];`
		`  #(BIT_PERIOD);`
		`end`
			8개의 데이터 비트 :
			각각의 비트 주기 중앙에서 샘플링

#4	Stop bit	`uart_if.rx = 1;` `#(BIT_PERIOD);`	Stop bit를 정상적으로 인식 / IDLE 상태로 복귀 준비
#5	Data Compare	`class scoreboard;` `if (trans.expected_data == trans.rx_data)`	expected data == trans. rx_data

Directed Scenario & Verification Results

2. UART_Rx - Test Scenario

- UART Rx FSM




```
// uart rx
// start bit
uart_if.rx = 1'b0;
#(BIT_PERIOD);

// data bits
for (i = 0; i < 8; i = i + 1) begin
    // data lsb부터  rx  보내기
    uart_if.rx = expected_data[i];
    $display("%t, Driving Data bit %0d: %b", $time, i, expected_data[i]);
    #(BIT_PERIOD);
end

// stop bit
uart_if.rx = 1;
#(BIT_PERIOD);
```

(Driver Code)
UART 통신과 동일한 Start, Data, Stop 신호를
순서대로 생성하여 DUT에 인가

Directed Scenario & Verification Results

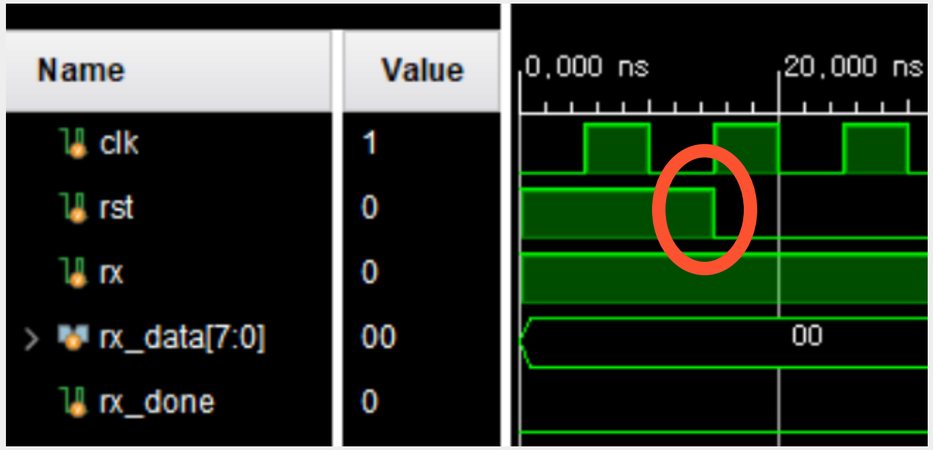
2. UART_RX - Waveform & Report

Log, Report

Waveform

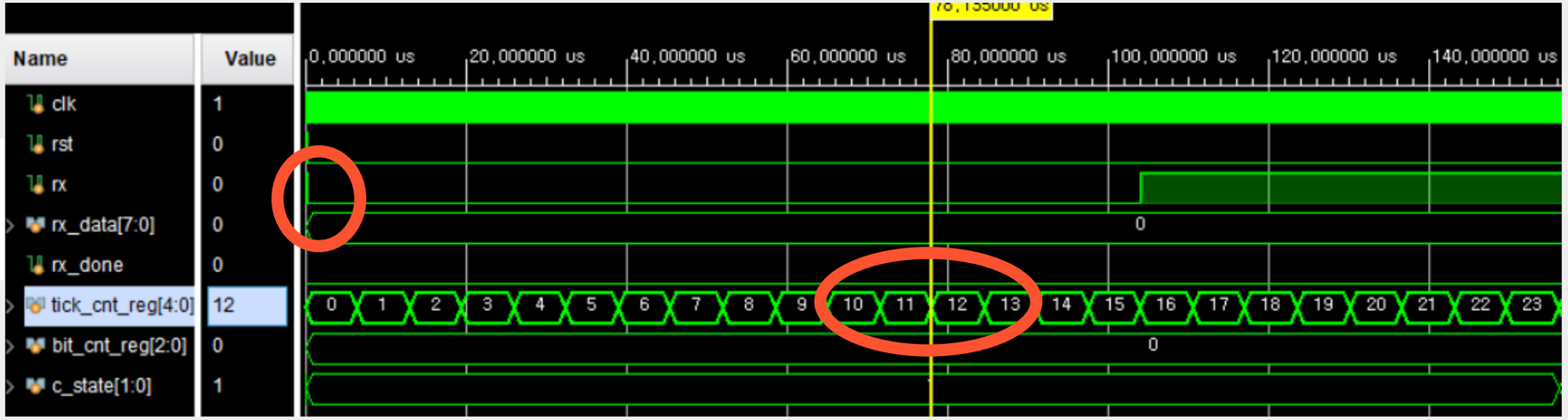
#1. Reset

```
==== UART RX Testbench Start ====
[DRV] reset done
```



#2. Start bit

```
35000, [GEN] expected_data = 213, rx_data =   x,
45000, Start bit detected correctly
```



rx의 falling edge를 감지 / Start bit : 비트 주기의 중앙에서 샘플링

Directed Scenario & Verification Results

2. UART_RX - Waveform & Report

Log, Report

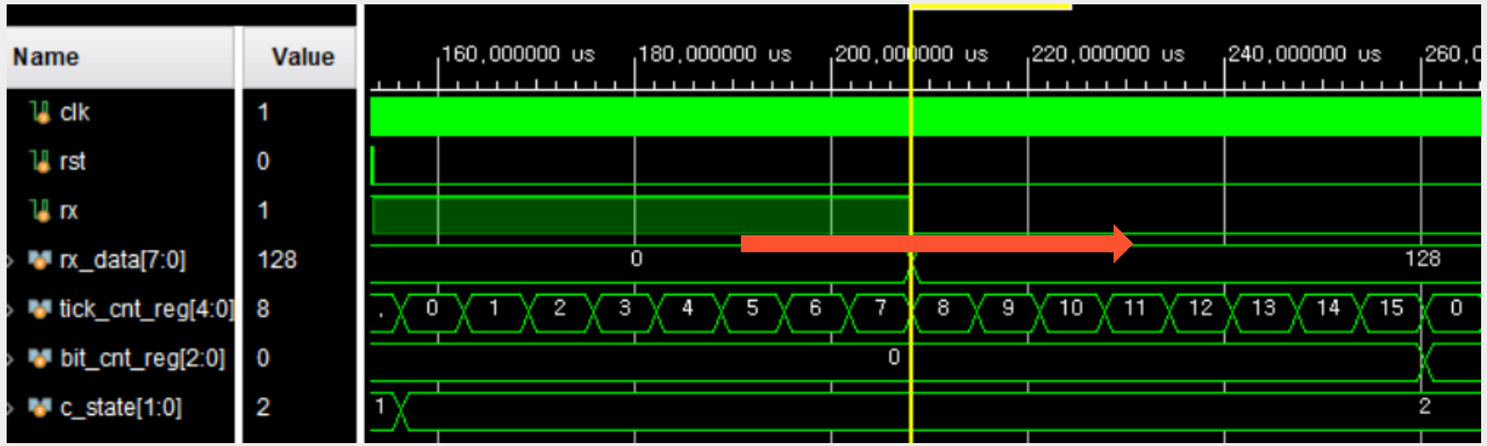
Waveform

#3. Data Bits Receive

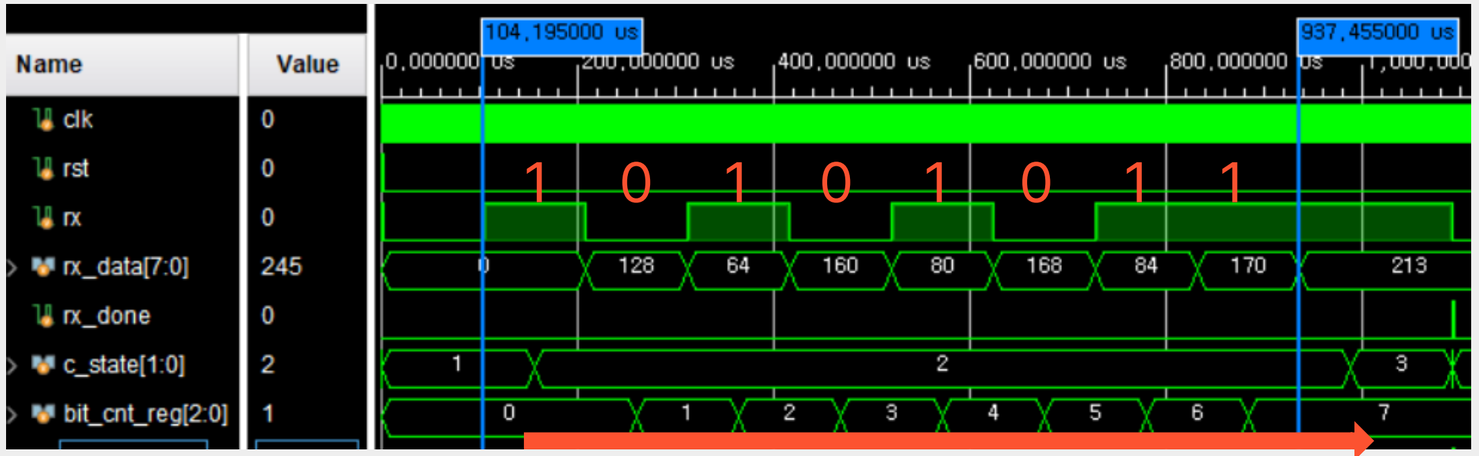
```
104195000, Driving Data bit 0: 1
208355000, Driving Data bit 1: 0
312515000, Driving Data bit 2: 1
416675000, Driving Data bit 3: 0
520835000, Driving Data bit 4: 1
624995000, Driving Data bit 5: 0
729155000, Driving Data bit 6: 1
833315000, Driving Data bit 7: 1
```

LSB

MSB



각각의 비트 주기 중앙에서 샘플링



8개 Data bits Receive

Directed Scenario & Verification Results

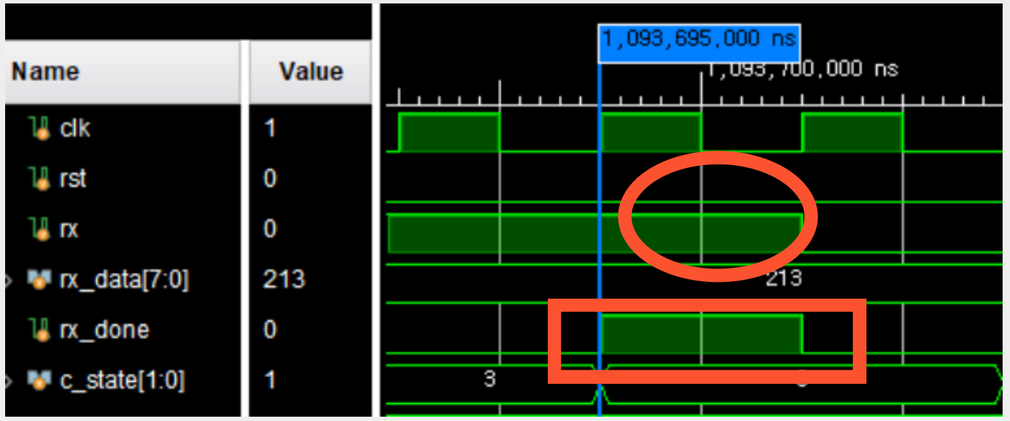
2. UART_RX - Waveform & Report

Log, Report

Waveform

#4. Stop Bit & Reception Complete

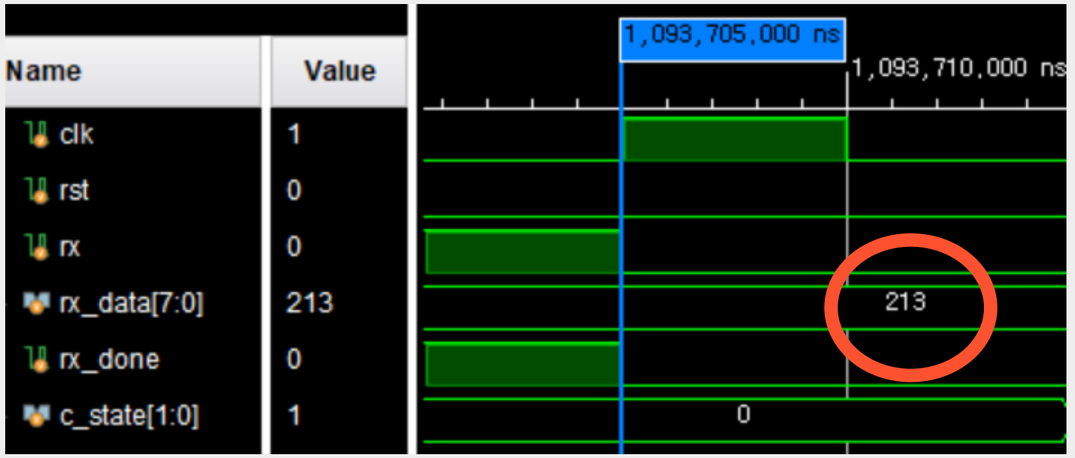
```
1093695000, [DRV] expected_data = 213, rx_data =   x, rx_done = x
1093695000, [MON] expected_data = 213, rx_data = 213, rx_done = 1
```



Stop bit 인식

#5. Data Compare

```
1093705000, [SCB] expected_data = 213, rx_data = 213, rx_done = 1
------------------------------------------------------------------
-------- Pass : expected data (213) == received data (213) ---------
------------------------------------------------------------------
```



Directed Scenario & Verification Results

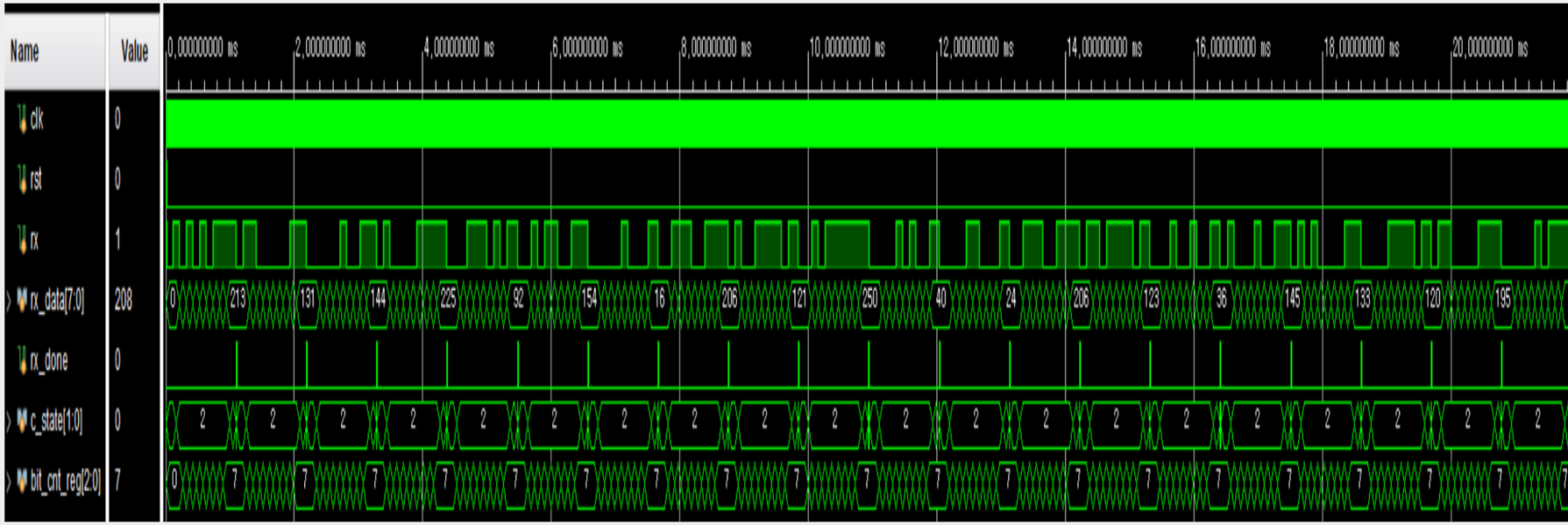
2. UART_RX - Waveform & Report

Log, Report

Waveform

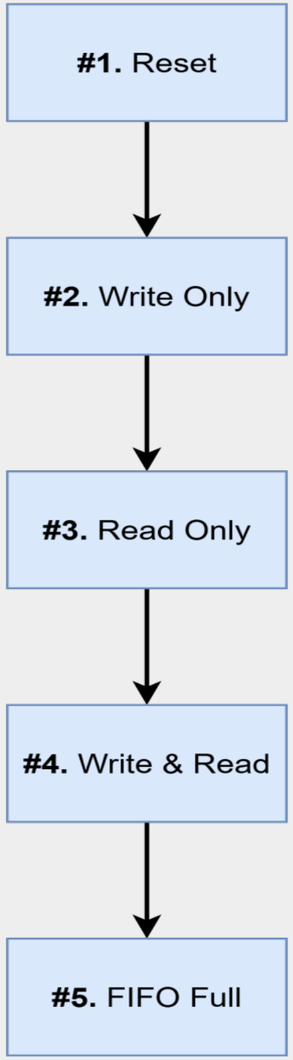
#6. Final Report

```
========================================
============test report================
========================================
========== Total test :   20 ==========
========== Pass test :   20 ==========
========== Fail test :    0 ==========
========================================
==========Test bench is finish=========
========================================
```



Directed Scenario & Verification Results

3. UART_FIFO - Test Scenario



Scenario		Base Code	Check Point
#1	Reset	`drv.reset();`	Intialization
#2	Write Only	`class scoreboard;` `if (trans.wr)`	데이터가 순차적으로 저장 / 첫 Write 이후 empty 0으로 변화
#3	Read Only	`class scoreboard;` `if (trans.rd)`	데이터 First-In, First-Out 순서로 출력

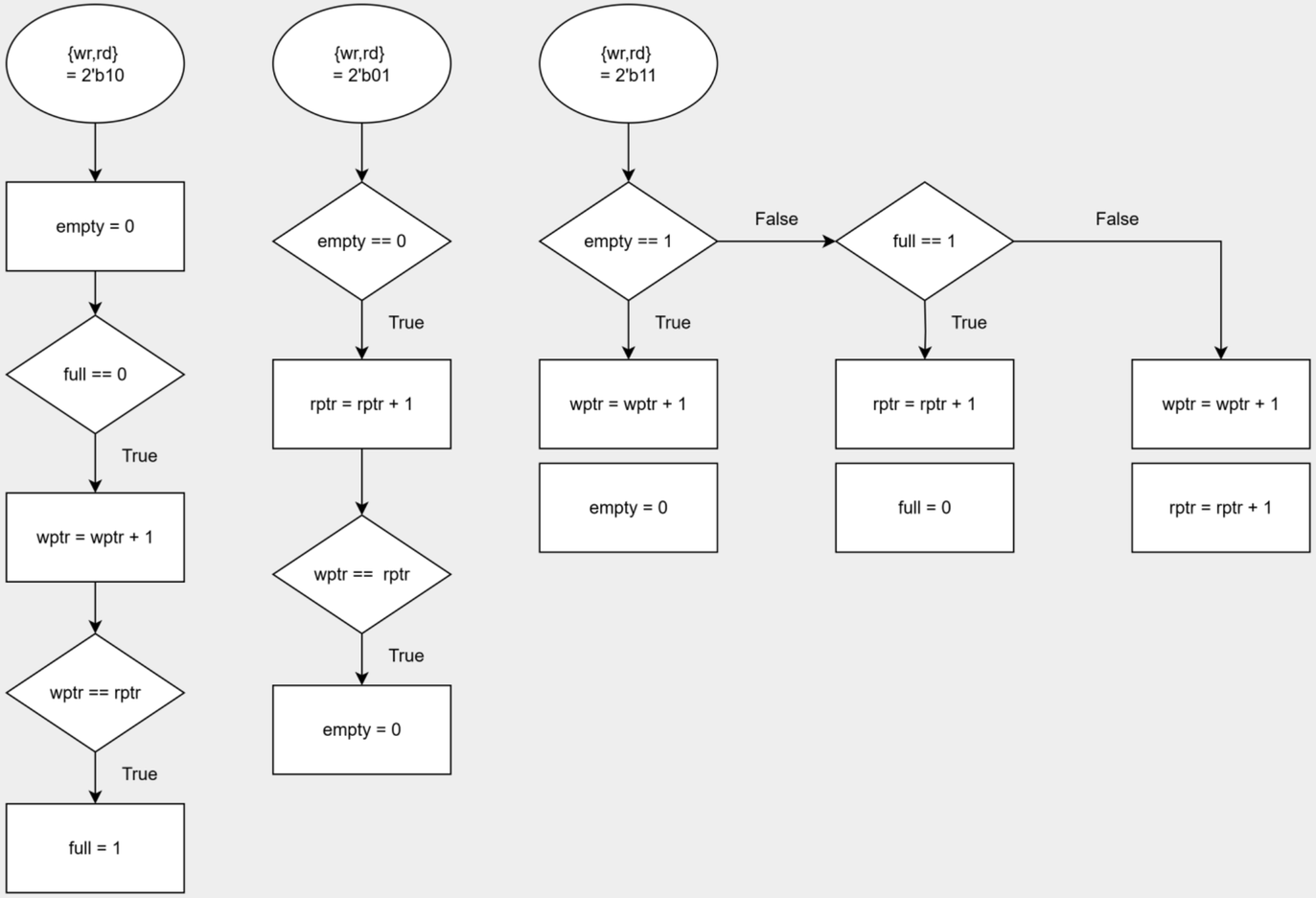
#4	Write & Read	`class scoreboard;`
		`if (trans.wr)`
		`if (trans.rd)`
			데이터 총량이 일정하게 유지 /
			Write & Read 충돌 없이 처리

#5	FIFO Full	`class scoreboard;`
		`if (trans.wr)`
		`  if (trans.full)`
			FIFO가 꽉 찼을 때 : full : 1 /
			추가 Write 시도가 무시 (Overflow 방지)

Directed Scenario & Verification Results

3. UART_FIFO - Test Scenario

- UART FIFO ASM




```
// trans.wr == 1 : push
if (trans.wr) begin
    write_count++;
    if (!trans.full) begin
        fifo_queue.push_back(trans.wdata);
        $display(
            "[SCB] : Data store in Queue : data : %d, size :  %d",
            trans.wdata, fifo_queue.size());
    end else begin
        $display("[SCB] : Queue is full : %d", fifo_queue.size());
    end
end

// trans.rd == 1 : pop
if (trans.rd) begin
    read_count++;
    if (!trans.empty) begin
        expected_data = fifo_queue.pop_front();
        if (trans.rdata == expected_data) begin
            read_pass_count++;
            $display("[SCB] : Data matched : %d", trans.rdata);
        end else begin
            read_fail_count++;
            $display("[SCB] : Data mis-matched : %d, %d",
                     trans.rdata, expected_data);
        end
    end else begin
        $display("[SCB] FIFO is Empty");
    end
end
```

(Scoreboard Code)

- Queue를 이용해 이상적인 FIFO 동작 결과 미리 예측
- DUT의 실제 출력 값과 자신의 예측 값을 비교하여 Pass / Fail을 자동 판정

Directed Scenario & Verification Results

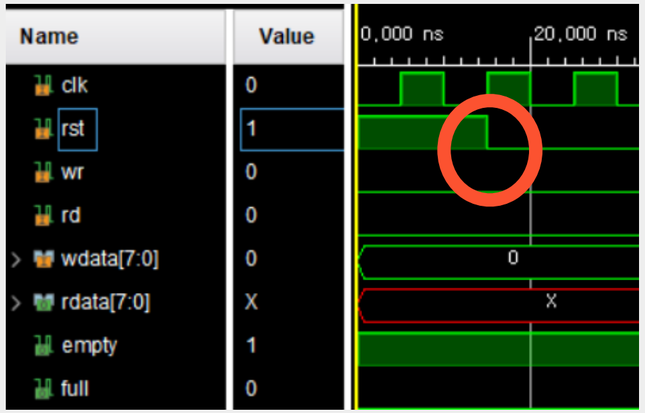
3. UART_FIFO - Waveform & Report

Log, Report

Waveform

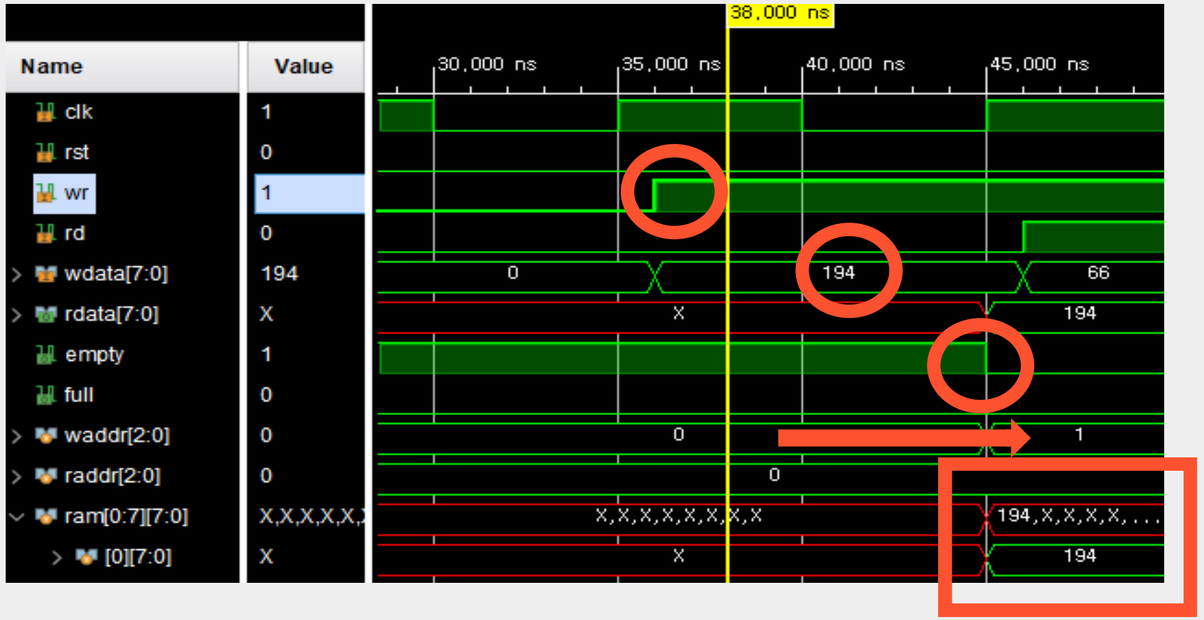
#1. Reset

```
========== UART FIFO Testbench Start ==========
DRV] reset done
```



#2. Write Only

```
35000, [GEN] wr = 1, rd = 0  wdata = 194  rdata =   x, full = x, empty = x
36000, [DRV] wr = 1, rd = 0  wdata = 194  rdata =   x, full = x, empty = x
38000, [MON] wr = 1, rd = 0  wdata = 194  rdata =   x, full = 0, empty = 1
38000, [SCB] wr = 1, rd = 0  wdata = 194  rdata =   x, full = 0, empty = 1
[SCB] : Data store in Queue : data : 194, size :           1
--------------------------------------------------------------------------------
'{194}
--------------------------------------------------------------------------------
```



데이터가 순차적으로 저장 /
첫 Write 이후 empty 0으로 변화

Directed Scenario & Verification Results

3. UART_FIFO - Waveform & Report

Log, Report

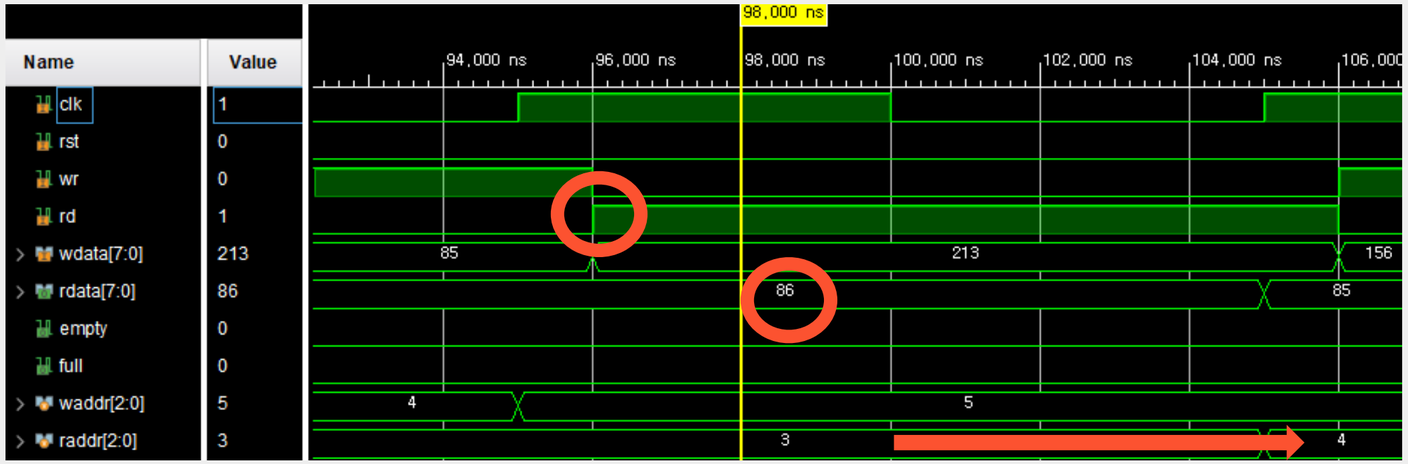
#3. Read Only

```
'{86,85}
--------------------------------------------------------------------------------
 88000, [GEN] wr = 0, rd = 1, wdata = 213, rdata =   x, full = x, empty = x
 96000, [DRV] wr = 0, rd = 1, wdata = 213, rdata =   x, full = x, empty = x
 98000, [MON] wr = 0, rd = 1, wdata = 213, rdata =  86, full = 0, empty = 0
 98000, [SCB] wr = 0, rd = 1, wdata = 213, rdata =  86, full = 0, empty = 0
[SCB] : Data matched :  86
--------------------------------------------------------------------------------
'{85}
--------------------------------------------------------------------------------
```

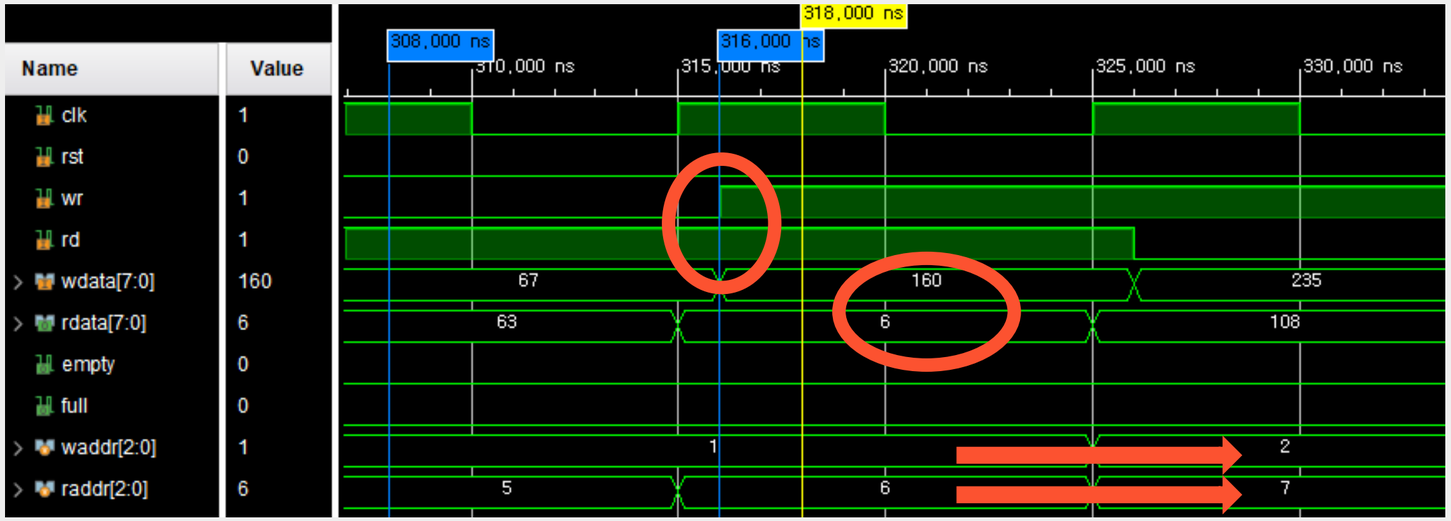
#4. Write & Read

```
'{6,108,68}
--------------------------------------------------------------------------------
308000, [GEN] wr = 1, rd = 1, wdata = 160, rdata =   x, full = x, empty = x
316000, [DRV] wr = 1, rd = 1, wdata = 160, rdata =   x, full = x, empty = x
318000, [MON] wr = 1, rd = 1, wdata = 160, rdata =   6, full = 0, empty = 0
318000, [SCB] wr = 1, rd = 1, wdata = 160, rdata =   6, full = 0, empty = 0
[SCB] : Data store in Queue : data : 160, size :           4
[SCB] : Data matched :   6
--------------------------------------------------------------------------------
{108,68,160}
--------------------------------------------------------------------------------
```

Waveform



데이터 First-In, First-Out 순서로 출력



데이터 총량이 일정하게 유지 / Write & Read 충돌 없이 처리

Directed Scenario & Verification Results

3. UART_FIFO - Waveform & Report

Log, Report

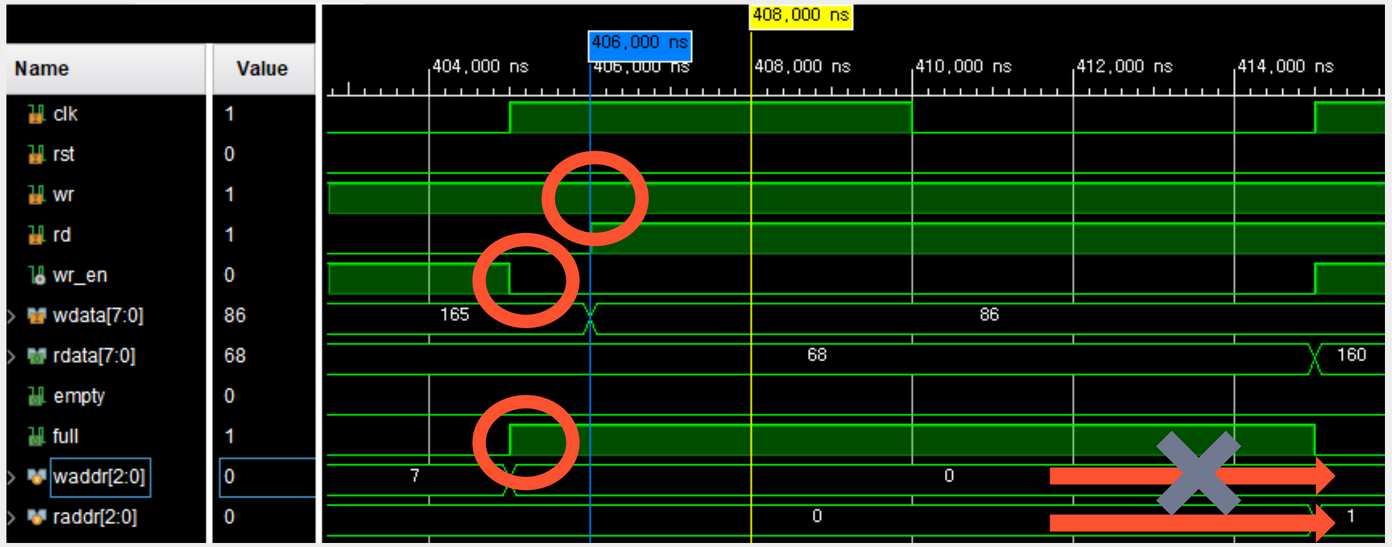
#5. FIFO Full

```
'{68,160,235,179,28,217,108,165}
------------------------------------------------------------------------------
398000, [GEN] wr = 1, rd = 1, wdata =  86, rdata =   x, full = x, empty = x
406000, [DRV] wr = 1, rd = 1, wdata =  86, rdata =   x, full = x, empty = x
408000, [MON] wr = 1, rd = 1, wdata =  86, rdata =  68, full = 1, empty = 0
408000, [SCB] wr = 1, rd = 1, wdata =  86, rdata =  68, full = 1, empty = 0
[SCB] : Queue is full :          8
 [SCB] : Data matched :  68
------------------------------------------------------------------------------
'{160,235,179,28,217,108,165}
------------------------------------------------------------------------------
```

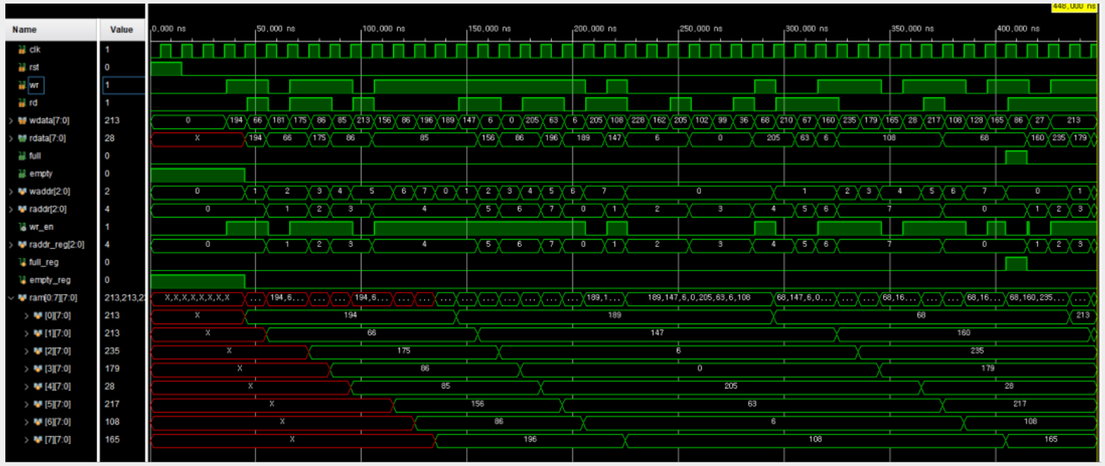
#6. Final Report

```
================================================
==================Test Report===================
================================================
===========     Write Count    :   26 ===========
===========     Read Count     :   19 ===========
===========  Pass test (Read)  :   19 ===========
===========  Fail test (Read)  :    0 ===========
================================================
================= TEST PASSED  =================
================================================
================ Test bench Finish =============
================================================
```

Waveform

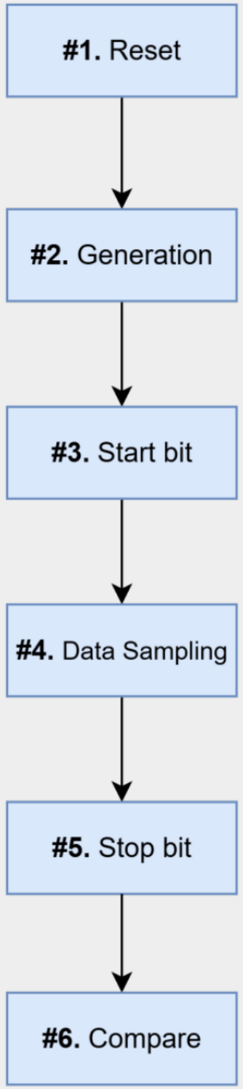


FIFO가 꽉 찼을 때 : full : 1 / 추가 Write 시도가 무시 (Overflow 방지)



Directed Scenario & Verification Results

4. UART_TX - Test Scenario



Scenario		Base Code	Check Point
#1	Reset	`driver.reset()`	Intialization

#2	Generation	`generator.body()`
		`scoreboard.add_expected_data()`
		`driver.run()`
			1. Randomized tx_data를 생성
			2. Scoreboard의 *expected data*에 random tx_data 입력
			3. tx_data를 DUT로 drive

#3	Start bit	`monitor.uart_receive()`	**Start bit 값이 샘플링 되는 시점** ***(중앙에서 샘플링)***
#4	Data Sampling	`monitor.run()`	각 bit의 값을 LSB부터 sampling
#5	Stop bit	`monitor.uart_receive()`	1. Stop bit 감지 2. tx signal 샘플링 후 *received_data*에 저장
#6	Receive & Compare	`scoreboard.run()` `scoreboard.report()`	1. Sampling된 data와 expected data값 비교 2. Report 생성

Directed Scenario & Verification Results

4. UART_TX - Test Scenario

Log, Report

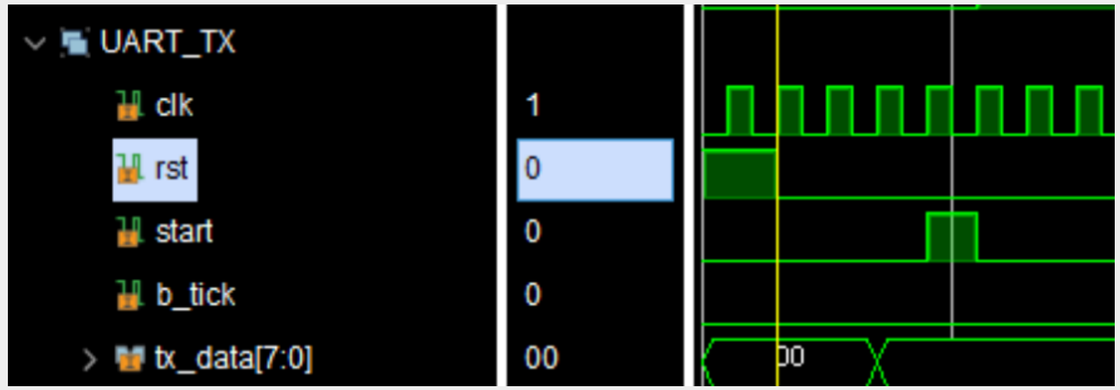
#1. Reset

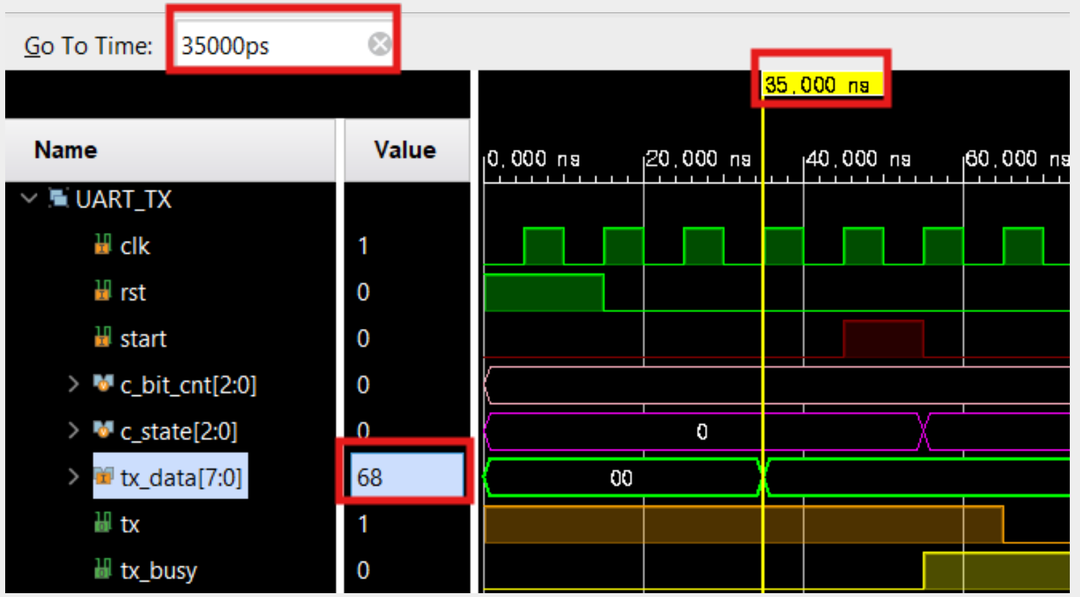
```
Reset Done
35000: [Generator] | tx_data = 0x68 (104)
[Scoreboard] Expected data added: 0x68
45000: [Driver] | tx_data = 0x68 (104)
```

#2. Generation

```
Reset Done
35000: [Generator] | tx_data = 0x68 (104)
[Scoreboard] Expected data added: 0x68
45000: [Driver] | tx_data = 0x68 (104)
```

Waveform





Directed Scenario & Verification Results

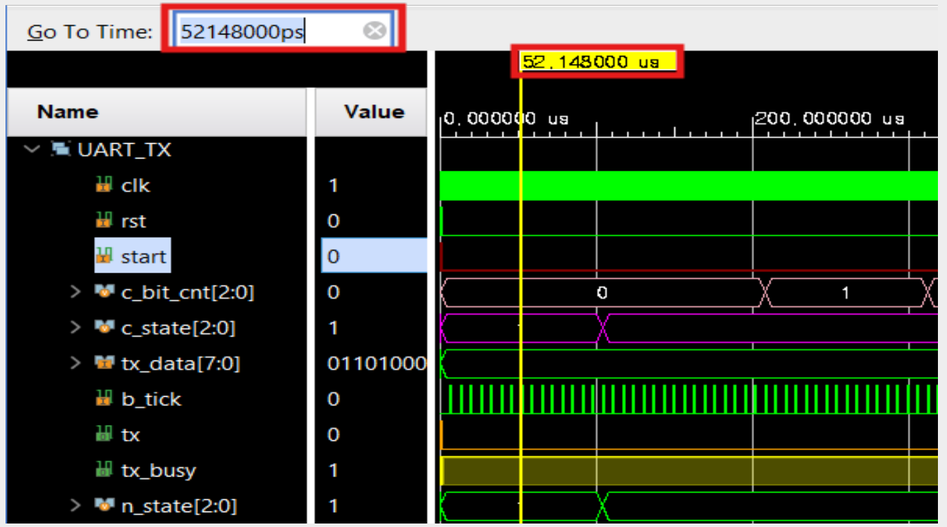
4. UART_TX - Test Scenario

Sim Log

Waveform

#3. Start bit

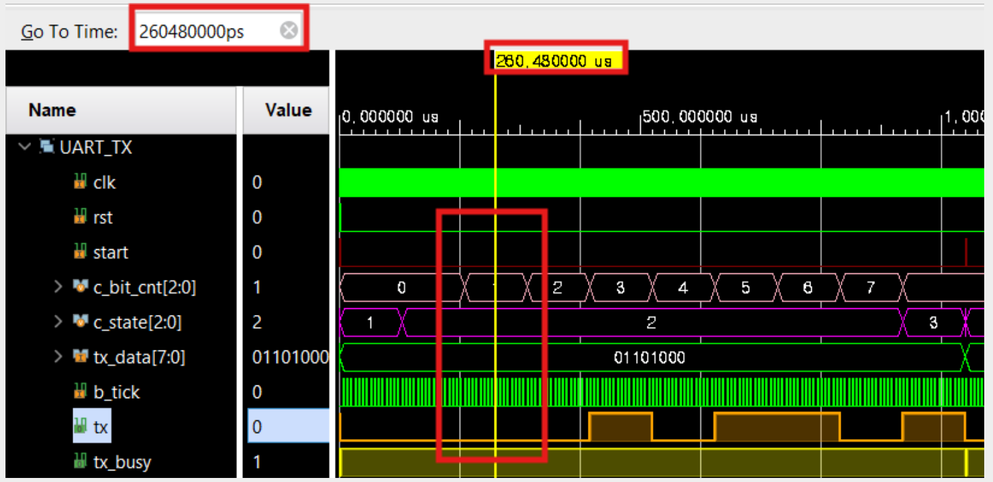
```
[Monitor] 52148000 : Start bit detected correctly
[Monitor] 156314000: Data bit 0: 0
[Monitor] 260480000: Data bit 1: 0
[Monitor] 364646000: Data bit 2: 0
[Monitor] 468812000: Data bit 3: 1
[Monitor] 572978000: Data bit 4: 0
[Monitor] 677144000: Data bit 5: 1
[Monitor] 781310000: Data bit 6: 1
[Monitor] 885476000: Data bit 7: 0
```



#4. Data Sampling (0110_1000 case)

```
[Monitor] 52148000 : Start bit detected correctly
[Monitor] 156314000: Data bit 0: 0
[Monitor] 260480000: Data bit 1: 0
[Monitor] 364646000: Data bit 2: 0
[Monitor] 468812000: Data bit 3: 1
[Monitor] 572978000: Data bit 4: 0
[Monitor] 677144000: Data bit 5: 1
[Monitor] 781310000: Data bit 6: 1
[Monitor] 885476000: Data bit 7: 0
[Monitor] 989642000:Stop bit detected correctly
```

received_data 에 저장



Directed Scenario & Verification Results

4. UART_TX - Test Scenario

Log, Report

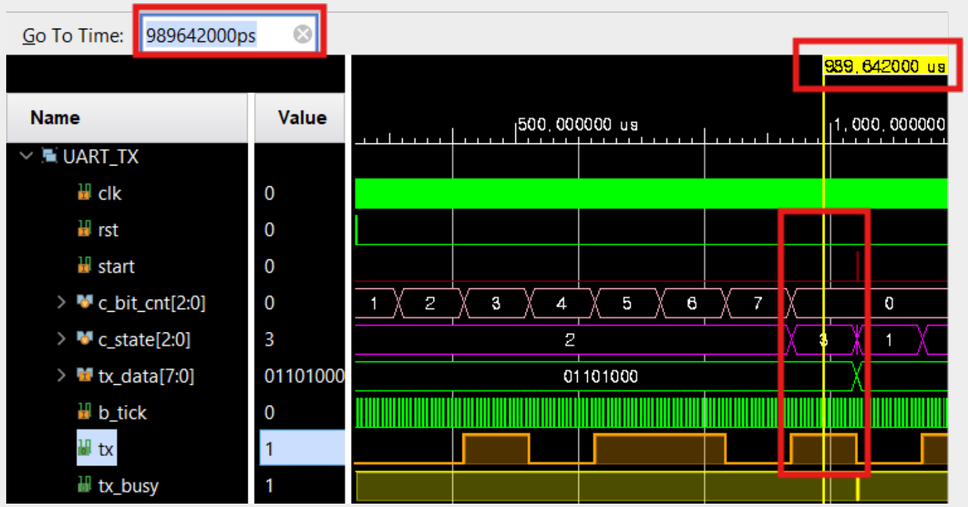
#5. Stop bit

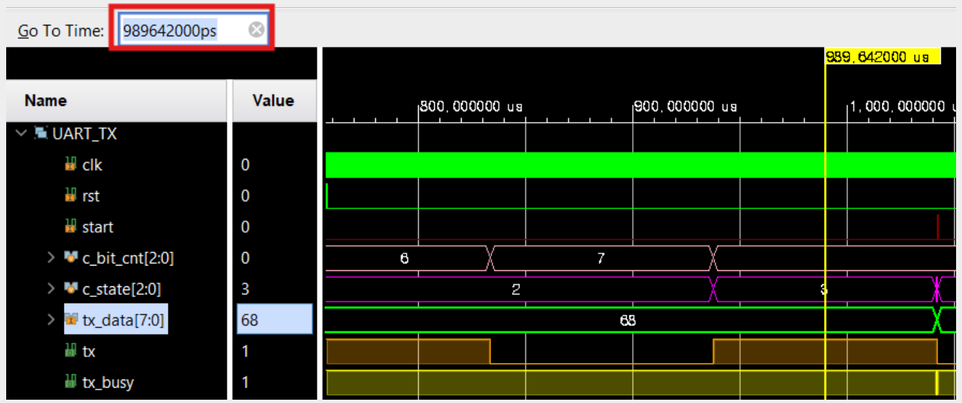
```
[Monitor] 52148000 : Start bit detected correctly
[Monitor] 156314000: Data bit 0: 0
[Monitor] 260480000: Data bit 1: 0
[Monitor] 364646000: Data bit 2: 0
[Monitor] 468812000: Data bit 3: 1
[Monitor] 572978000: Data bit 4: 0
[Monitor] 677144000: Data bit 5: 1
[Monitor] 781310000: Data bit 6: 1
[Monitor] 885476000: Data bit 7: 0
[Monitor] 989642000: Stop bit detected correctly
```

#6. Compare (Report)

```
989642000: [Monitor] | tx_data = 0x68 (104)
989642000: [Scoreboard] | tx_data = 0x68 (104)
[Scoreboard] V PASS: Expected=0x68, Received=0x68
```

Waveform





Directed Scenario & Verification Results

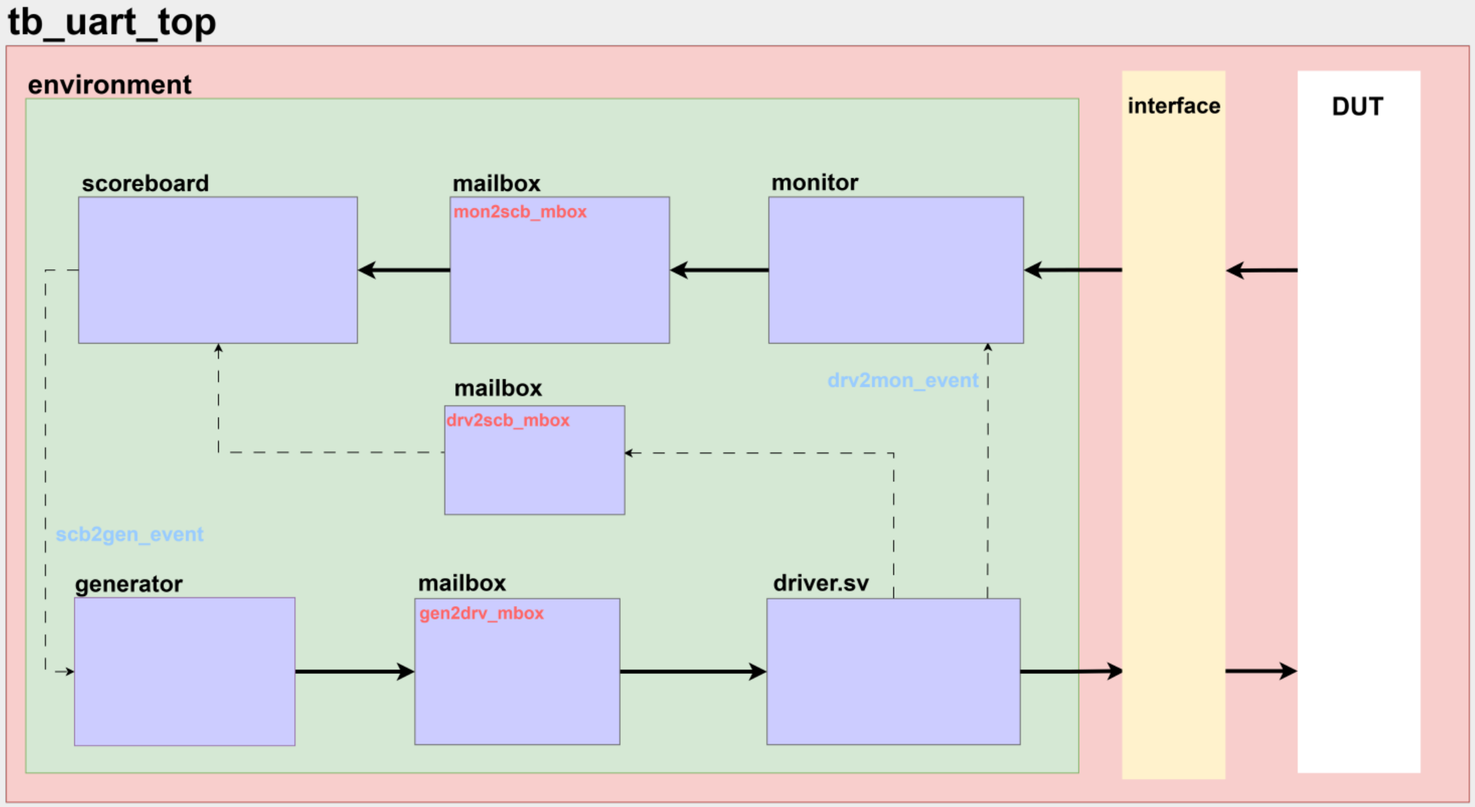
5. UART_top - Test Scenario

◐ Test Scenario

Test Scenario:

- 1. Basic Cases
  - 1.1 Reset Test
  - 1.2 Single Byte Loopback
- 2. Functional Cases
  - 2.1 Sequential Data Flow
  - 2.2 Random Data Flow

◐ TB Architecture



Directed Scenario & Verification Results

5. UART_top - #1. Basic Case

#1-1. Reset

Base Code

```
class environment;
  ...
//run()task
    task run();
        drv.reset();

        // DUT 내부 안정화를 위한 추가 대기 시간
        $display("%0t: Waiting for DUT stabilization...", $time);
        repeat (100) @(posedge vif.clk);  // 충분한 안정화 시간
        $display("%0t: DUT stabilization complete. Starting test...", $time);

        fork
            gen.body(10);
            drv.run();
            mon.run();
            scb.run();
        join_any
      ...
    endtask
  ...
endclass
```

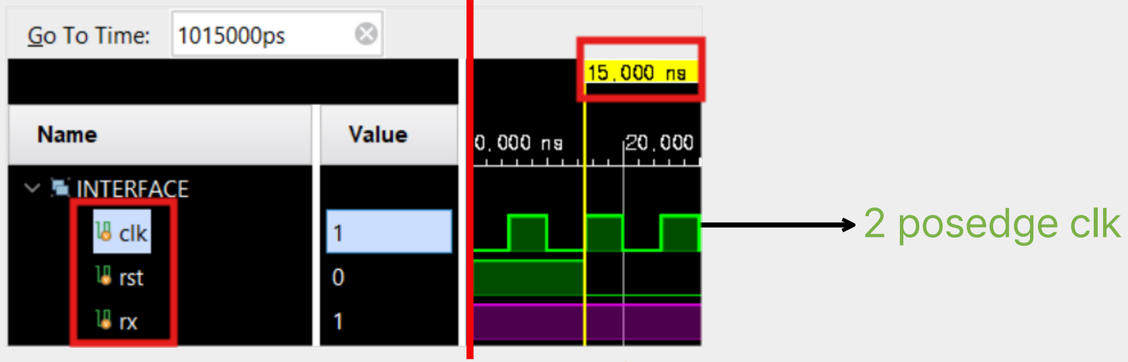
Sim Result

100 clk
cycle gap!

```
15000: Reset Done
15000: Waiting for DUT stabilization...
relaunch_sim: Time (s): cpu = 00:00:00 ; elapsed = 00:00:10 .
run all
1015000: DUT stabilization complete. Starting test...
```

```
class driver;
  ...
//reset()task
    task reset();
        vif.clk = 0;
        vif.rst = 1;
        vif.rx  = 1;  // UART idle state is high
        repeat (2) @(posedge vif.clk);
        vif.rst = 0;
        $display("%0t: Reset Done", $time);
    endtask
  ...

endclass
```



Directed Scenario & Verification Results

5. UART_top - #1. Basic Case

#1-2. Single Byte Loopback

Base Code

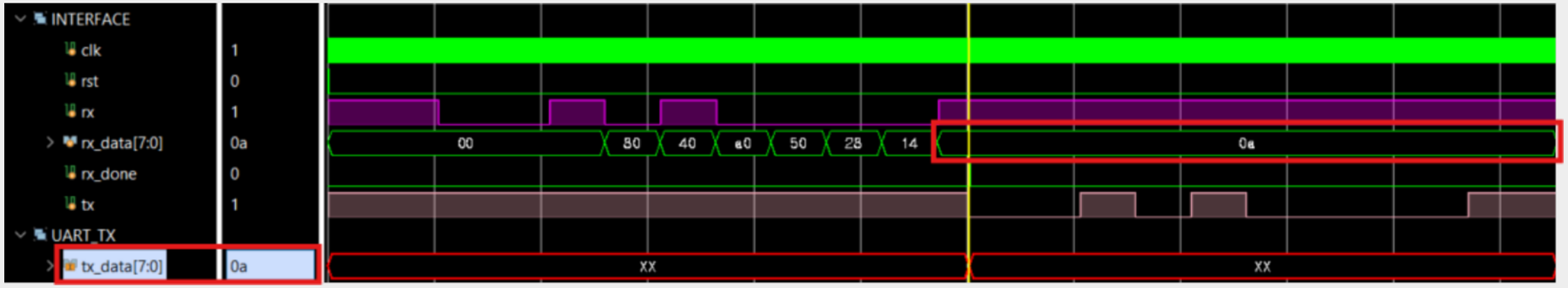
```
class environment;
  ,,,
//run()task
    task run();
        drv.reset();

        // DUT 내부 안정화를 위한 추가 대기 시간
        $display("%0t: Waiting for DUT stabilization...", $time);
        repeat (100) @(posedge vif.clk);  // 충분한 안정화 시간
        $display("%0t: DUT stabilization complete. Starting test...", $time);

        fork
            gen.body(1); //Single Byte
            drv.run();
            mon.run();
            scb.run();
        join_any
      ,,,
    endtask
  ,,,
endclass
```

Sim Result

```
15000: Reset Done
15000: Waiting for DUT stabilization...
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:12 . Memory (MB): peak = 2051.629 ; gain = 0.000
run all
1015000: DUT stabilization complete. Starting test...
1015000: [Generator : Randomized Send Data Generated] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data = 0xxx, TB_RX_Done = x
1015000: First transmission - additional stabilization wait
1015000: Scoreboard : DRV will send: 0x0a, and SCB would saved it to Queue
209335000: [Driver] : Sending data to DUT: 0x0a
1256495000: Monitor : TX Start bit(0) detected. Sampling data initiated...
1303015000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data = 0xxx, TB_RX_Done = x
---------------RX sequence complete---------------
---------------TX sequence Start---------------
2298095000: Monitor : Sampled frame: 1000010100
2298095000: Monitor : TB received TX data: 0x0a (frame valid)
2298095000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x0a, TB_RX_Done = 1
2298095000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x0a, TB_RX_Done = 1
2298095000: LOOPBACK PASS : TB_Sent(by Drv): 0x0a , TB_Received(by Mon): 0x0a
==========================================
2298095000: Generator - All 1 transactions completed
Simulation Finished
Test Report:
Total Transactions: 1
PASS Count: 1
FAIL Count: 0
All tests PASSED!
```



Directed Scenario & Verification Results

5. UART_top - #2. Functional Case

#2-1. Sequential Data Flow

Base Code

```
class generator;
 ...
// body() task
    task body(int count);
        repeat(count) begin  // 사용자 정의 횟수만큼 transaction 생성
            tr = new();

            // Sequential 테스트용 (5개 데이터일 때)
            if (count == 5) begin
                static int seq_counter = 1;
                tr.send_data = seq_counter;
                seq_counter++;
                if (seq_counter > 5) seq_counter = 1; // 1-5 순환
            end

            tr.display("Generator : Sequential/Random Send Data Generated");
            gen2drv_mbox.put(tr);
            @(scb2gen_event);  // scoreboard의 이벤트를 기다림
        end
        $display("%0t: Generator - All %0d transactions completed", $time,
                 count);
    endtask
  ...
endclass
```

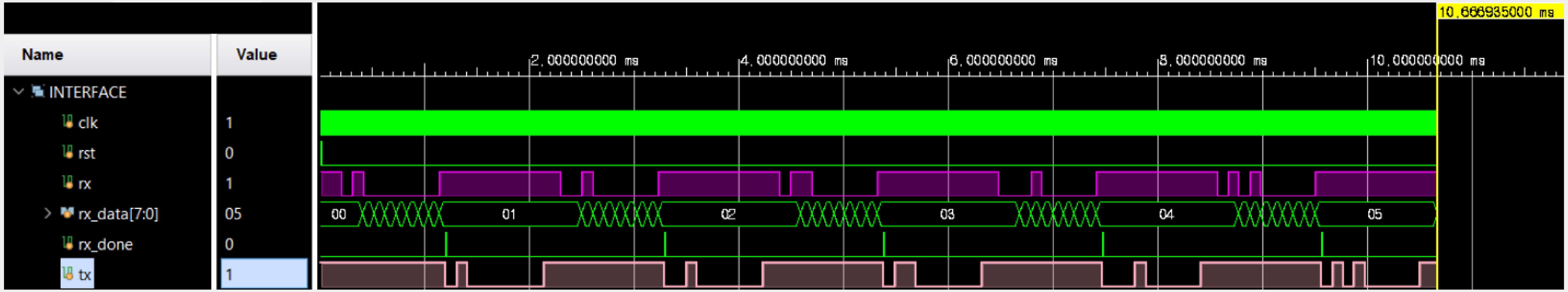
gen.body() : 1 ~ 5의 값을 가지는 sequential data를 생성

Directed Scenario & Verification Results

5. UART_top - #2. Functional Case

#2-1. Sequential Data Flow

Sim Result




```
15000: Reset Done
15000: Waiting for DUT stabilization...
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:11 . Memory (MB): peak = 2592.770 ; gain = 0.000
run all
1015000: DUT stabilization complete. Starting test...
1015000: [Generator : Sequential/Random Send Data Generated] | Send_Data = 0x01  DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_D
1015000: First transmission - additional stabilization wait
1015000: Scoreboard : DRV will send: 0x01, and SCB would saved it to Queue
209335000: [Driver] : Sending data to DUT: 0x01
1256495000: Monitor : TX Start bit(0) detected. Sampling data initiated...
1303015000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x01, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_
---------------RX sequence complete---------------
---------------TX sequence Start---------------
2298095000: Monitor : Sampled frame: 1000000010
2298095000: Monitor : TB received TX data: 0x01 (frame valid)
2298095000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x01, TB_RX_Done = 1
2298095000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x01, TB_RX_Done = 1
2298095000: LOOPBACK PASS : TB_Sent(by Drv): 0x01 , TB_Received(by Mon): 0x01
2298095000: SEQUENCE PASS : Expected[0]: 0x01, Received: 0x01
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
```

Reset 이후, first data

```
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
8567225000: [Generator : Sequential/Random Send Data Generated] | Send_Data = 0x05, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_
8567225000: [Driver] : Sending data to DUT: 0x05
8567225000: Scoreboard : DRV will send: 0x05, and SCB would saved it to Queue
8619305000: Monitor : RX Start bit(0) detected
9563216000: [Monitor: RX_Done] | Send_Data = 0x00, DUT_RX_Data = 0x05, DUT_RX_Done = 1, DUT_TX = 1, TB_RX_Data = 0xxx, TB_RX_Done = x
9615335000: Monitor : TX Start bit(0) detected. Sampling data initiated...
9660905000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x05, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX
---------------RX sequence complete---------------
---------------TX sequence Start---------------
10656935000: Monitor : Sampled frame: 1000001010
10656935000: Monitor : TB received TX data: 0x05 (frame valid)
10656935000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x05, TB_RX_Done = 1
10656935000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x05, TB_RX_Done = 1
10656935000: LOOPBACK PASS : TB_Sent(by Drv): 0x05 , TB_Received(by Mon): 0x05
10656935000: SEQUENCE PASS : Expected[4]: 0x05, Received: 0x05
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
10656935000: Generator - All 5 transactions completed
Simulation Finished
Test Report:
Total Transactions: 5
PASS Count: 5
FAIL Count: 0
All tests PASSED!
```

Last data(5th), 최종 report

Directed Scenario & Verification Results

5. UART_top - #2. Functional Case

#2-2. Random Data Flow

Base Code

```
class transaction;
    // TB에서 DUT로 전송할 데이터
    // Gen -> Driver -> DUT 경로에서 사용될 데이터
    rand
    bit [7:0] send_data;  // TB가 전송할 데이터 (DUT의 RX로 입력됨)
  ...

endclass
```

transaction : "send_data" field 선언

```
class generator;
 ...
// body() task - 사용자 정의 횟수만큼 UART 데이터 생성
    task body(int count);
        repeat(count) begin  // 사용자 정의 횟수만큼 transaction 생성
            tr = new();
            assert (tr.randomize())
            else $error("Randomization failed");
            tr.display("Generator : Randomized Send Data Generated");
            gen2drv_mbox.put(tr);
            @(scb2gen_event);
        end
        $display("%0t: Generator - All %0d transactions completed", $time,
                 count);
    endtask
  ...

endclass
```

gen.body() : 입력한 횟수만큼, randomized send_data 생성

Directed Scenario & Verification Results

5. UART_top - #2. Functional Case

#2-2. Random Data Flow

Base Code

```
class driver;
 ...
//task uart_sender() - TB가 UART 신호를 DUT의 RX로 전송.
    task uart_sender(input [7:0] send_data);
        logic [9:0] uart_frame;
        $display("%0t: [Driver] : Sending data to DUT: 0x%02h", $time,
                 send_data);

        // UART frame 구성: Start bit(0) + 8 data bits(LSB first) + Stop bit(1)
        uart_frame = {
            1'b1, send_data, 1'b0
        };  // MSB: stop bit, data[7:0], LSB: start bit

        // 각 비트를 순서대로 전송 (LSB부터)
        for (int i = 0; i < 10; i++) begin
            vif.rx = uart_frame[i];
            repeat (BIT_PERIOD)
            @(posedge vif.clk);  // 비트 주기만큼 대기
        end

        // Idle 상태로 복귀
        vif.rx = 1'b1;
        repeat (BIT_PERIOD / 2) @(posedge vif.clk);
    endtask

  ...
endclass
```

driver.uart_sender() : radomized된 send_data를 start bit, stop bit를 포함하는 10bit짜리 하나의 data frame으로 변환하여 driving

driver.run() : data driving 및 scoreboard의 expected data에 전달

```
class environment;
 ...
//run()task
    task run();
        drv.reset();
       ...
        fork
            gen.body(10);
            drv.run();
            mon.run();
            scb.run();
        join_any
  ...
endclass
```

env.run() : ***gen.body()*** 실행 횟수 지정

Directed Scenario & Verification Results

5. UART_top - #2. Functional Case

#2-2. Random Data Flow

Base Code

```
class scoreboard;
 ...
//run()task - 올바른 루프백 검증
    task run();
        transaction mon_tr;  // Monitor용 transaction
        transaction drv_tr;  // Driver용 transaction
        forever begin
            fork
                // Monitor로부터 TB가 수신한 TX 데이터 처리
                begin
                    mon_tr = new();
                    mon2scb_mbox.get(mon_tr);

                    // TB가 TX 데이터를 수신했을 때만 검증
                    if (mon_tr.TB_rx_done) begin
                        total_transactions++;
                        mon_tr.display("Scoreboard : TB_RX");

                        // 루프백 검증: TB가 보낸 데이터와 TB가 받은 데이터 비교
                        if (sent_data_queue.size() > 0) begin
                            bit [7:0] expected_data = sent_data_queue.pop_front();

                            if (mon_tr.TB_received_data == expected_data) begin
                                pass_count++;
                                $display(
                                    "%0t: LOOPBACK PASS : TB_Sent(by Drv): 0x%02h , TB_Received(by Mon): 0x%02h",
                                    $time, expected_data,
                                    mon_tr.TB_received_data);
                                $display(
                                    "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
                            end else begin
                                fail_count++;
                                $display(
                                    "%0t: LOOPBACK FAIL : TB_Sent(by Drv): 0x%02h != TB_Received(by Mon): 0x%02h",
                                    $time, expected_data,
                                    mon_tr.TB_received_data);
                                $display(
                                    "=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=");
                            end
                        end

                        ->scb2gen_event;  // generator에게 이벤트 발생
                    end
                end

                // Driver로부터 전송 데이터 저장
                begin
                    drv_tr = new();
                    drv2scb_mbox.get(drv_tr);
                    sent_data_queue.push_back(drv_tr.send_data);
                    $display(
                        "%0t: Scoreboard : DRV will send: 0x%02h, and SCB would saved it to Queue",
                        $time, drv_tr.send_data);
                end
            join_any
  ...
endclass
```

monitor.uart_tx_decode() : tx signal을 8bit짜리 tx_data로 decoding후, received_data에 저장

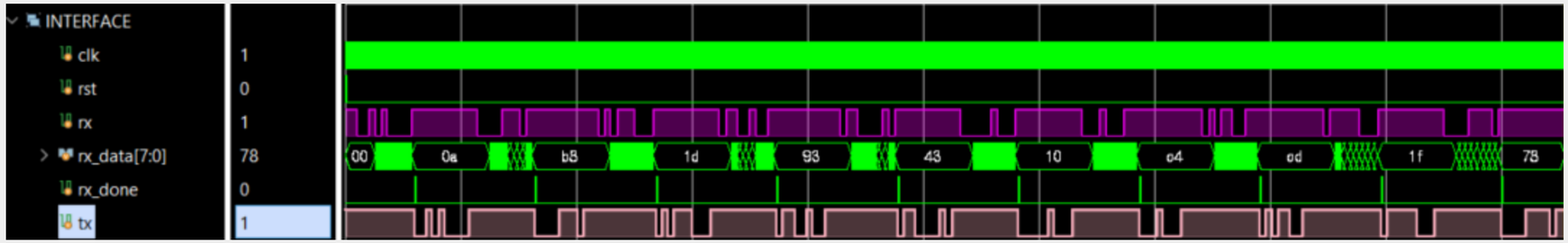
scoreboard.run() : expected_data와, received_data를 비교

Directed Scenario & Verification Results

5. UART_top - #2. Functional Case

#2-2. Random Data Flow

Simulation Result




```
15000: Reset Done
15000: Waiting for DUT stabilization...
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:11 . Memory (MB): peak = 2051.629 ; gain = 0.000
run all
1015000: DUT stabilization complete. Starting test...
1015000: [Generator : Randomized Send Data Generated] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data = 0xxx, TB_RX_Done = x
1015000: First transmission - additional stabilization wait
1015000: Scoreboard : DRV will send: 0x0a, and SCB would saved it to Queue
209335000: [Driver] : Sending data to DUT: 0x0a
1256495000: Monitor : TX Start bit(0) detected. Sampling data initiated...
1303015000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data = 0xxx, TB_RX_Done = x
---------------RX sequence complete---------------
---------------TX sequence Start---------------
2298095000: Monitor : Sampled frame: 1000010100
2298095000: Monitor : TB received TX data: 0x0a (frame valid)
2298095000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x0a, TB_RX_Done = 1
2298095000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x0a, TB_RX_Done = 1
2298095000: LOOPBACK PASS : TB_Sent(by Drv): 0x0a , TB_Received(by Mon): 0x0a
```

Reset 이후, first data

```
19015775000: [Generator : Randomized Send Data Generated] | Send_Data = 0x78, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data =
19015775000: [Driver] : Sending data to DUT: 0x78
19015775000: Scoreboard : DRV will send: 0x78, and SCB would saved it to Queue
19067855000: Monitor : RX Start bit(0) detected
20011766000: [Monitor: RX_Done] | Send_Data = 0x00, DUT_RX_Data = 0x78, DUT_RX_Done = 1, DUT_TX = 1, TB_RX_Data = 0xxx, TB_RX_Done = x
20063885000: Monitor : TX Start bit(0) detected. Sampling data initiated...
20109455000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x78, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_R
---------------RX sequence complete---------------
---------------TX sequence Start---------------
21105485000: Monitor : Sampled frame: 101111000
21105485000: Monitor : TB received TX data: 0x78 (frame valid)
21105485000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x78, TB_RX_Done = 1
21105485000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x78, TB_RX_Done = 1
21105485000: LOOPBACK PASS : TB_Sent(by Drv): 0x78 , TB_Received(by Mon): 0x78
==========
21105485000: Generator - All 10 transactions completed
Simulation Finished
Test Report:
Total Transactions: 10
PASS Count: 10
FAIL Count: 0
All tests PASSED!
```

Last data, 최종 report

5.
Trouble Shooting
&
Debugging

05

Trouble Shooting & Debugging

0. Verification Strategy

Bottom-Up 검증 전략

- Sub 모듈 : Protocol-Level Verification
  - **UART_TX**
  - **UART_RX**
  - **FIFO**
- Top 모듈 : Feature-Level Verification
  - **UART_TOP**

→ Sub module : 기본 동작 보장 & High Module : integration시의 이슈에만 집중

→ 병렬적인 검증 process 구축 가능

Trouble Shooting & Debugging

1. TB Issues

- tb_uart_top : **loopback fail 오류**
  - **발견된 Issue :**
    - Simulation이 실행됐을때, **첫번째 data에 대한 Loopback test만 실패**하고, 나머지 loopback test는 정상적으로 수행되는 현상이 발생

```
15000: Reset Done
15000: [Generator : Randomized Send Data Generated] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, PC_RX_Data
15000: [Driver] : Sending data to DUT: 0x0a
15000: Scoreboard - TB will send: 0x0a (Queue size: 1)
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:11 . Memory (MB): peak = 1512.633 ; gain = 0.621
run all
996056000: [Monitor-RX_Done] | Send_Data = 0x00, DUT_RX_Data = 0x85, DUT_RX_Done = 1, DUT_TX = 1, PC_RX_Data = 0xxx, PC_RX_Done = x
996095000: Monitor - Start bit detected
1048175000: Monitor - Start bit confirmed, sampling data...
1093695000: [[Driver] : Send Data is Transmitted to rx] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, PC_RX_D
2089775000: Monitor - Sampled frame: 1100001010
2089775000: Monitor - PC received TX data: 0x85 (frame valid)
2089775000: [Monitor-TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, PC_RX_Data = 0x85, PC_RX_Done
2089775000: [Scoreboard-PC_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, PC_RX_Data = 0x85, PC_RX_Done =
2089775000: LOOPBACK FAIL - TB_Sent: 0x0a 놂 PC_Received: 0x85
```

Reset 후, 첫번째 data

```
5273165000: [[Driver] : Send Data is Transmitted to rx] | Send_Data = 0x0
6269195000: Monitor - Sampled frame: 1000011000
6269195000: Monitor - PC received TX data: 0x0c (frame valid)
6269195000: [Monitor-TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx,
6269195000: [Scoreboard-PC_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DU
6269195000: LOOPBACK PASS - TB_Sent: 0x0c 놂 PC_Received: 0x0c
----------------------------------------
6269195000: Generator - All 3 transactions completed
Simulation Finished
Test Report:
Total Transactions: 3
PASS Count: 2
FAIL Count: 1
Some tests FAILED!
```

마지막 data 샘플링 후, report 생성

Trouble Shooting & Debugging

1. TB Issues

- tb_uart_top : **loopback fail 오류**
  - **원인 분석 :**
    - Simulation Sequence :

```
15000: Reset Done
15000: [Generator : Randomized Send Data Generated] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data = 0xxx, TB_RX_Done = x
15000: [Driver] : Sending data to DUT: 0x0a
15000: Scoreboard : DRV will send: 0x0a, and SCB would saved it to Queue
INFO: [USF-XSim-96] XSim completed. Design snapshot 'tb_uart_fifo_loopback_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:07 ; elapsed = 00:00:14 . Memory (MB): peak = 1489.516 ; gain = 24.828
```

(re)launch_sim

```
run all
1048175000: Monitor : TX Start bit(0) detected. Sampling data initiated...
1093695000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x0a, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = x, TB_RX_Data = 0xxx, TB_RX_Done = x
----------RX sequence complete----------
----------TX sequence Start----------
2089775000: Monitor : Sampled frame: 1100001010
2089775000: Monitor : TB received TX data: 0x85 (frame valid)
2089775000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x85, TB_RX_Done = 1
2089775000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_Done = x, DUT_TX = 1, TB_RX_Data = 0x85, TB_RX_Done = 1
                                                0000_1010                    1000_0101
2089775000: LOOPBACK FAIL : TB_Sent(by Drv): 0x0a != TB_Received(by Mon): 0x85
```

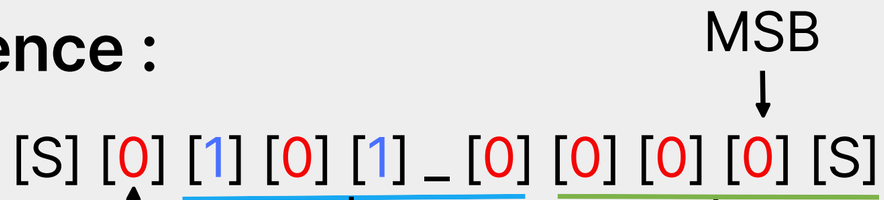
run all

Trouble Shooting & Debugging

1. TB Issues

- tb_uart_top : **loopback fail 오류**
  - **원인 분석 :**
    - **Simulation Sequence :**

원본 data 0×0a(0000_1010)의
UART 전송 sequence:



Monitor에서 샘플링한 첫번째 data
0×85(1000_0101)의 샘플링 sequence:

→ Reset 후, 첫번째 data 전달시, 충분한 시간이 확보되지 않아, DUT가 불안정해짐
→ Monitor의 TX start bit 감지가 1 clock cycle 지연
→ 이로 인해, **전체 data frame이 1bit씩 shift.**
→ 첫번째 data 전송시에만 reset timing 오류로 인해 발생함!!

Trouble Shooting & Debugging

1. TB Issues

- tb_uart_top : **loopback fail 오류**
  - 해결 방법 도출 : reset 후, 충분한 stabilization time을 인가

```
class environment;
//run task()
    task run();
        drv.reset();

        fork
            gen.body(10);
            drv.run();
            mon.run();
            scb.run();
        join_any
       ,,,
    endtask
  ,,,

enclass
```

기존 코드

```
class environment
//run()task
    task run();
        drv.reset();

        // DUT 내부 안정화를 위한 추가 대기 시간
        $display("%0t: Waiting for DUT stabilization...", $time);
        repeat (100) @(posedge vif.clk);  // 충분한 안정화 시간
        $display("%0t: DUT stabilization complete. Starting test...", $time);

        fork
            gen.body(10);
            drv.run();
            mon.run();
            scb.run();
        join_any
       ,,,
    endtask
  ,,,

enclass
```

수정된 코드

Trouble Shooting & Debugging

1. TB Issues

- tb_uart_top : **loopback fail 오류**
  - ■ 적용 결과 :

Reset 후,
run all을 실행전에 data drive가 실행되어 불안정한 DUT 상태

```
15000: Reset Done
15000: [Generator : Randomized Send Data Generated] | Send_Data = 0x0a, DUT_RX
15000: [Driver] : Sending data to DUT: 0x0a
15000: Scoreboard - TB will send: 0x0a (Queue size: 1)
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:14 . Memory (MB): pea
run all
996056000: [Monitor-RX_Done] | Send_Data = 0x00, DUT_RX_Data = 0x85, DUT_RX_Do
996095000: Monitor - Start bit detected
1048175000: Monitor - Start bit confirmed, sampling data...
1093695000: [[Driver] : Send Data is Transmitted to rx] | Send_Data = 0x0a, DU
2089775000: Monitor - Sampled frame: 1100001010
2089775000: Monitor - PC received TX data: 0x85 (frame valid)
2089775000: [Monitor-TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_R
2089775000: [Scoreboard-PC_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_
2089775000: LOOPBACK FAIL - TB_Sent: 0x0a 둔 PC_Received: 0x85
----------------------------------------
========================================
20897165000: Generator - All 10 transactions completed
Simulation Finished
Test Report:
Total Transactions: 10
PASS Count: 9
FAIL Count: 1
Some tests FAILED!
```

Reset 후,
run all을 실행시키기 전까지 충분한 시간 확보

```
15000: Reset Done
15000: Waiting for DUT stabilization...
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:12 . Memory (MB): peak
run all
1015000: DUT stabilization complete. Starting test...
1015000: [Generator : Randomized Send Data Generated] | Send_Data = 0x0a, DUT_RX_
1015000: First transmission - additional stabilization wait
1015000: Scoreboard : DRV will send: 0x0a, and SCB would saved it to Queue
209335000: [Driver] : Sending data to DUT: 0x0a
1256495000: Monitor : TX Start bit(0) detected. Sampling data initiated...
1303015000: [Driver: Send Data is Transmitted to UART_RX module] | Send_Data = 0x
----------RX sequence complete----------
----------TX sequence Start----------
2298095000: Monitor : Sampled frame: 1000010100
2298095000: Monitor : TB received TX data: 0x0a (frame valid)
2298095000: [Monitor: TX_Decoded] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_
2298095000: [Scoreboard : TB_RX] | Send_Data = 0x00, DUT_RX_Data = 0xxx, DUT_RX_D
2298095000: LOOPBACK PASS : TB_Sent(by Drv): 0x0a , TB_Received(by Mon): 0x0a
----------------------------------------
========================================
21105485000: Generator - All 10 transactions completed
Simulation Finished
Test Report:
Total Transactions: 10
PASS Count: 10
FAIL Count: 0
All tests PASSED!
```

6.
Conculsion

06

Conclusion

- **Key Achivevements**

☑ SV를 기반으로 한 **표준 TB Architecture**를 구성

☑ **Randomized된 다수의 data**를 인가함으로써 효율적인 검증 process 구축

☑ 팀 단위 작업 수행시, 더욱 효율적으로 Verification환경을 구성할 수 있는 전략수립 및 적용

Conclusion

- **Future Plan**

💡 Coverage Implementation

CDV(Coverage Driven Verification) 방법론을 활용한 정량적 평가지표 생성

- 주요 기능별로 coverpoint 정의
- Test Scenario 보완을 통해 coverage closure 달성할 예정(목표치 : > 90%)

💡 SVA(SystemVerilog Assertion)

Assertion을 활용한 핵심 protocol 및 기능 검증

- 주요 컴포넌트 혹은 모듈들에 assertion을 삽입
- 효율적인 verification closure 달성할 예정

💡 Constraint Randomized Approach

Constraint를 적용한 directed test 기반 검증 수행

- corner case, 디버깅에 있어서 적

- The End -