

실험 9. UART 통신 구현

Digital System Design and Experiment

Made by MMS Students

Graduate School of Convergence Science and Technology

Seoul National University

 **Mobile Multimedia Systems Group**

Pre-work: Python 환경 설정

- [illegible]

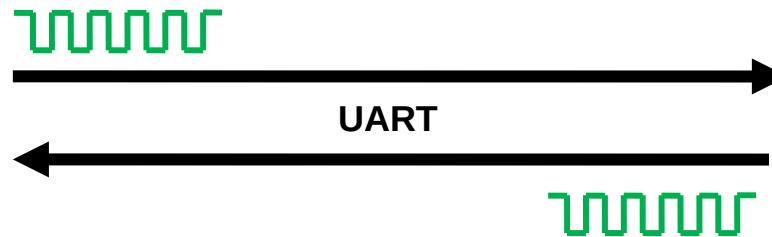
목차

- 실험 목표
- UART 통신 개요
- 실험
- 실험 결과 및 제출 방법

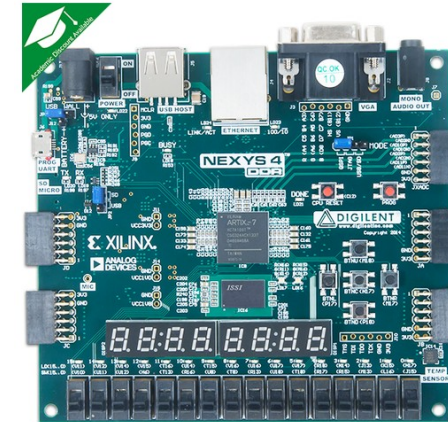
실험 목표

- UART(Universal Asynchronous Receiver-Transmitter) 를 통해 데이터 송수신하는 일부 모듈 구현
- 시뮬레이션으로 구현한 모듈 검증
- Python 을 이용하여 PC 에서 UART 를 통해 data 전송하여 FPGA 보드 상에서 구현한 동작 확인

Jupyter-notebook



Nexys 4 DDR



1.

Background

Communication Protocols

- Wired Communications

- I2C, SPI
- PCI Express
- **UART**, USART
- USB, Firewire
- Ethernet
- ...

- Wireless Communication

- Bluetooth
- Zigbee
- WiFi
- ...

UART 통신 개요

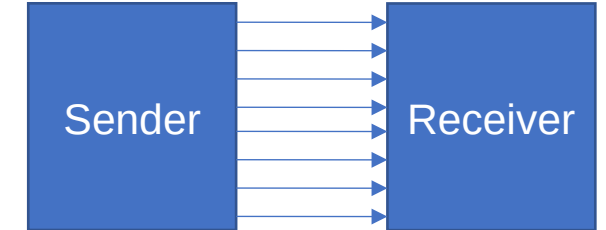
- 범용 비동기 송수신기 (Universal Asynchronous Receiver-Transmitter)

- Serial 통신의 일종

- Bit 단위의 데이터 통신
- Tx, Rx 가 구분되어 있음 (Full-duplex)



Serial Transfer



Parallel Transfer

- Baud-rate

- 데이터를 전송하는 속도 .
- BPS(Bit / Sec) 단위를 사용
- 9,600bps, 112,500bps 등의 속도를 사용
- 연결된 2 개의 Chip 의 Baud-rate 가 같아야 데이터를 올바르게 수신

- 참고

- <http://www.circuitbasics.com/basics-uart-communication/>
- <http://www.gpcet.ac.in/wp-content/uploads/2018/08/UNIT-V.pdf>

UART 동작

■ 동작

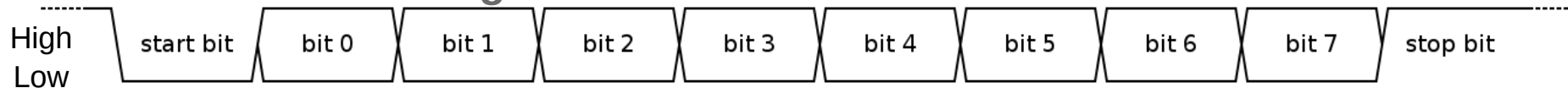
- HIGH(대기상태) -> LOW(시작) -> 데이터 전송 -> HIGH(stop & 대기 상태)

■ Data packet 구성

- Start bit : 통신의 시작 . HIGH -> LOW 로 변경하면서 시작 (1bit)
- Data frame : 전송하려는 data. (5 ~ 9bit)
- Parity bit : 오류 검증 비트 . (0 ~ 2bit)
- Stop bit : 통신 종료를 의미 . HIGH 상태로 변경하고 대기상태 진입 (1 ~ 2bit)

■ 본 실험에서의 설정

- Start bit: 1bit 이고 항상 “ low” 로 해야 함
- Data frame: 8bit 로 해야함
- Parity bit: 사용 안함
- Stop bit: 1bit 로 하고 항상 “ high” 해야함



2.

실험 개요

| 실험

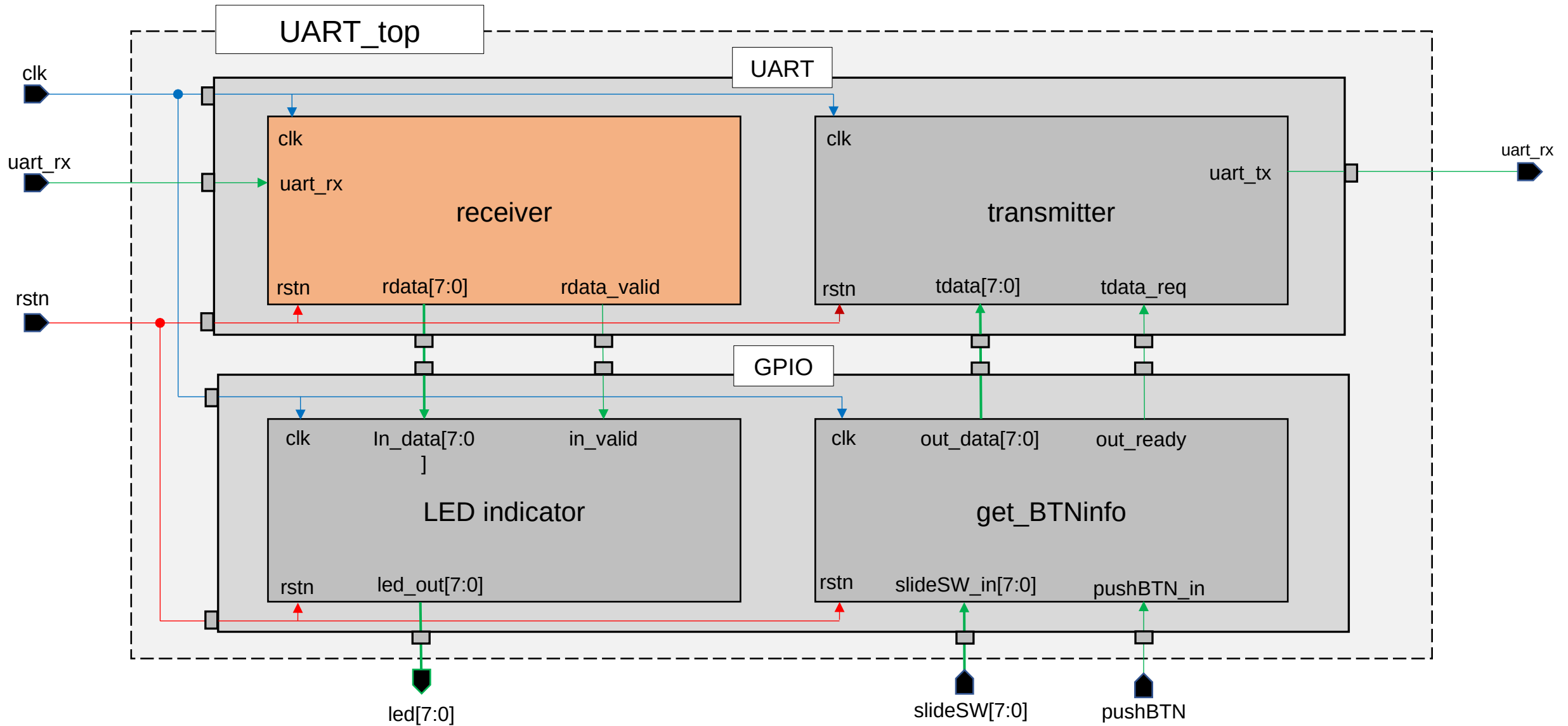
■ 실험 목표

- UART-Rx : Jupyter-notebook 을 이용하여 FPGA 에 8bit 씩 , 64bit data 를 전송
 - LED7~LED0 로 매번 입력한 data 확인 가능
- UART-Tx : Slide Switch 로 송신할 Bit 를 설정하고 , Push button 을 이용하여 송신 명령을 내린 뒤 , Jupyter-notebook 에서 수신한 값을 송신한 값과 비교하여 일치 여부 확인
- 이번 실험에서 구현한 uart.v 를 Project 에서 활용

■ 실험 환경

- Vivado 2019.1
- Jupyter notebook
- Nexys 4 DDR FPGA 보드
 - System clock 100MHz
 - **UART baud rate: 921,600 bps**

Block diagram



Metastability Problem

■ Metastability 란 ?

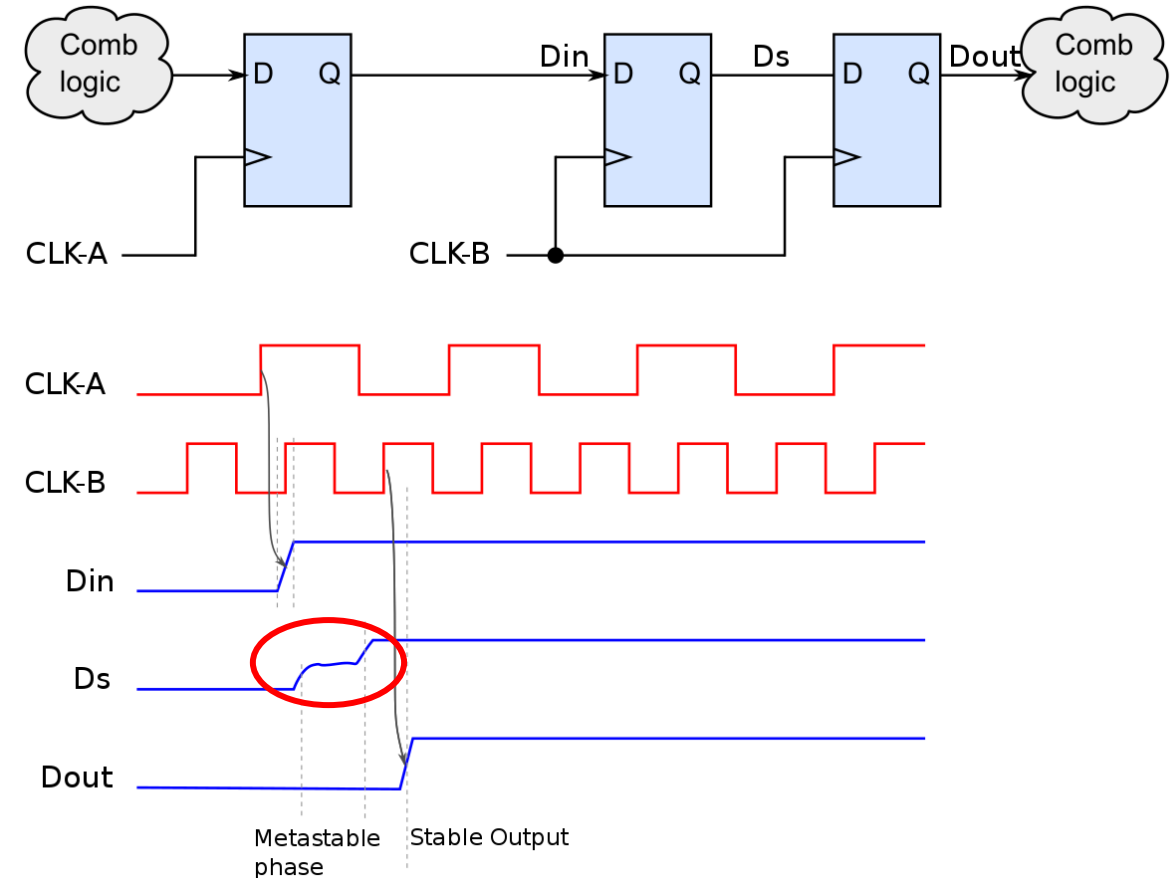
- asynchronous data transfer 시 발생하는 문제
- data 가 stable 하지 않은 상태에서 값을 capture 하면 , 간혹 0 과 1 모두 아닌 어중간한 값으로 남아있음

■ Common solution

- 여러 Flip-Flop 으로 buffering 수행
- 보통 일정 시간이 지나면 0 또는 1 로 값이 stable 해짐
- buffering 을 통해 metastability 가 발생할 확률을 낮춤

■ UART 통신 또한 asynchronous

- 따라서 receive 시 위와 같은 buffering 필요



제공된 코드파일 설명

- Verilog 파일 (____.v)
 - uart_top.v
 - 전체 top module
 - **uart.v**
 - **Receiver**
 - **수정해야 할 부분 .**
 - PC로부터 받은 serial data 를 parallel data 로 변환한 뒤 GPIO.v 의 LEDindicator 모듈로 전송함
 - Transmitter
 - GPIO.v 의 getBTNinfo 에서 전송받은 parallel data 를 serial data 로 변환한 뒤 PC 로 데이터를 전송함
- gpio.v
 - LED indicator
 - uart receive 에서 변환해 준 8-bit parallel Data 를 LED 7~0 을 이용하여 표시함
 - Get_BTNinfo
 - 8 개의 Slide Switch 를 이용해 설정한 Data 를 uart transmitter 로 보냄
 - Push button 이 눌리면 uart transmitter 로 transmit request 를 보냄 . 이 때 Push button 의 debouncing 을 수행하며 , 중복 전송을 막기 위해 1 회 송신 후 100ms 동안은 재송신을 막음

실험 설명

■ TO DO

- 본 실험은 “**uart.v**” 파일 내에서 수정해야 함
- Serial data 를 parallel data 로 변환하는 receiver module 을 구현해야 함
- 아래 설명 대로 port 에 해당 신호를 전송하면 됨
 - “rdata_valid” 는 GPIO 모듈로 전송하는 data 가 valid 할 때 “high”
 - “rdata” 는 Parallel data 로 변환된 data 이고 GPIO 모듈에 전송됨

Port 설명

- **Rule 1: receiver module 내의 input output port 들 변경 불가**
- **Rule 2: Output port 기능 대로 신호를 보낼 것 .**

Receiver module 의 Input / Output Ports

Port name	In/out	type	Description
clk	input	wire	System clock signal
rstn	input	wire	Reset signal 이고 falling edge 에서 반응함
uart_rx	input	wire	PC 로부터 받는 bit serial data
rdata_valid	output	reg	gpio 모듈로 전송하는 data 가 valid 할 때 “ high” 로 해야함
rdata	output	reg [7:0]	Parallel data 로 변환된 data 이고 gpio 모듈에 전송됨

TIP 1

- “Init count”, “Baud_rate_count”, “Receive bit” signal 을 이용하여 state 변경
- “uart_rx” port 에 들어오는 PC 에서 전송한 Data 를 저장하고 bit shift 하는 방법으로 구현
- 주어진 Internal Signal 외에 더 필요할 경우 , 추가 혹은 제거하여 사용 가능

Internal Signal

Port name	type	Description
state	reg	현재 state 를 나타냄 (IDLE or RCV)
next_state	reg	다음 state 를 나타냄 (IDLE or RCV)
uart_rx_buf1,2	reg	수신한 Bit 를 buffering 함 (Metastability 를 막기 위해)
Init_count	reg	처음 Start bit 의 유효성을 확인하기 위한 counter, Low 상태로 일정 Cycle 이상 유지되면 IDLE state 에서 RCV state 로 천이한다 .
Baud_rate_count	reg	108 Cycle 마다 Data 를 취하기 위한 Counter
Receive_bit	reg	8 개의 bit 이 될 때 까지 수신한 bit 를 세어주는 Counter

UART 를 통해서 데이터는 baud rate 에 맞게 1bit 씩 들어온다 (baud rate 가 921,600bps 이므로 , 1/921600second 마다 1bit 씩 데이터가 들어온다).

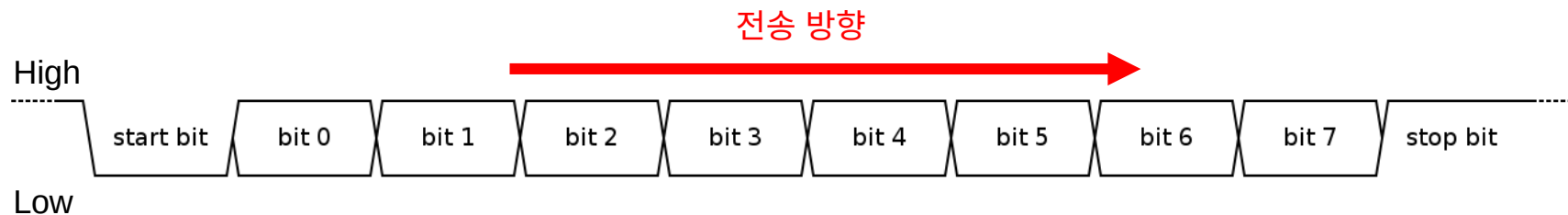
한편 , 보드에서 제공하는 system clock 은 100 MHz(1/100000000second) 로 , baud rate 보다 훨씬 빠른 속도로 작동된다 .

따라서 이를 synchronization 시켜주기 위해서 system clock 기준 108cycle (100MHz/921600) 에 한번씩 data 를 취하도록 조정해주어야 한다 .

이를 위한 것이 Baud_rate_count 이다 .

TIP 2

- Data packet 구성 (아래 그림과 같음)
 - Start bit 전에 받은 RxD 는 항상 “ high ” 임
 - Start bit 은 항상 “ low ” 임
 - Data frame 은 8bit 임
 - Parity bit 은 사용 안함
 - Stop bit 은 항상 “ high ” 임
- 본 실험에서 UART baud rate 은 921,600bps 로 하였음



TIP 3 (Ex: always block 두개로 구현하는 방법)

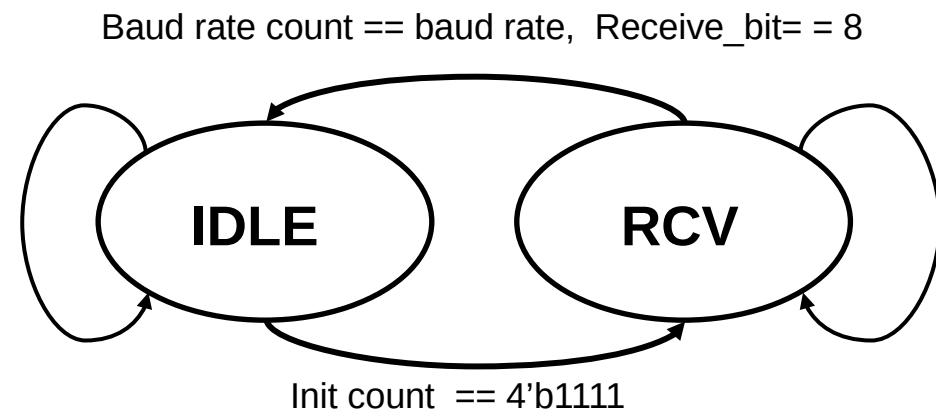
■ 1st always:

■ (IDLE state) :

Start Bit 이 일정 시간 (16 Cycle) 이상 Low 로 유지되면 (init_count 가 4'b1111 이 되면) RCV state 로 천이 , 그렇지 않으면 현 state 유지

■ (RCV state) :

8 Bit 데이터를 다 수신하였으면 IDLE 상태로 천이 , 아니면 현 state 유지



■ 2nd always:

■ Reset

■ Else



■ (Common) : 매 Clock Cycle 에서 수신한 uart_rx bit 를 shift 하면서 버퍼링 한다 .

■ (IDLE state) : Start bit 의 유효성 검증을 위해 uart_rx_buf2 값을 관찰하여 Low 이면 init_count 를 증가

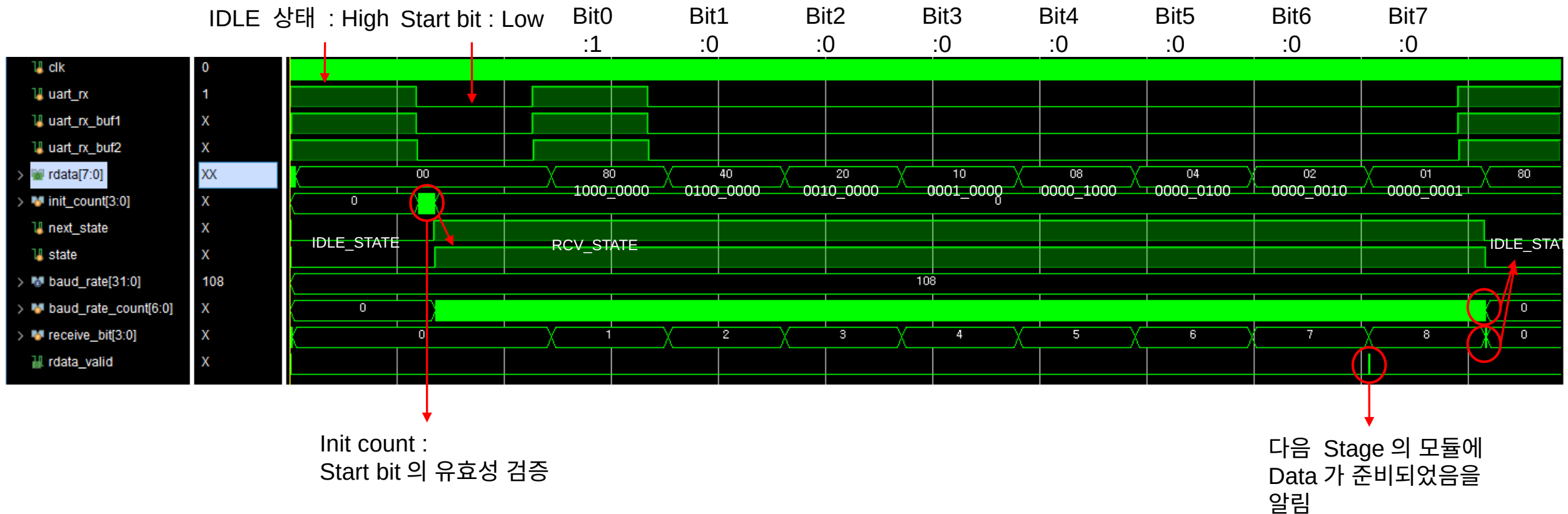
■ (RCV state) :

■ Baud_rate_count 값을 매 Clock cycle 마다 증가시킨다 .

■ Baud_rate_count 가 108 에 도달하면 rdata 값을 right shift 하고 , uart_rx_buf2 의 값을 rdata 의 MSB 자리에 넣어준다 . 그리고 Receive bit 을 1 증가시킨다 .

• + Receive bit 가 7 에 도달하면 rdata_valid 값을 1 cycle 동안 High 로 들어온다 .

TIP 4 Timing Diagram

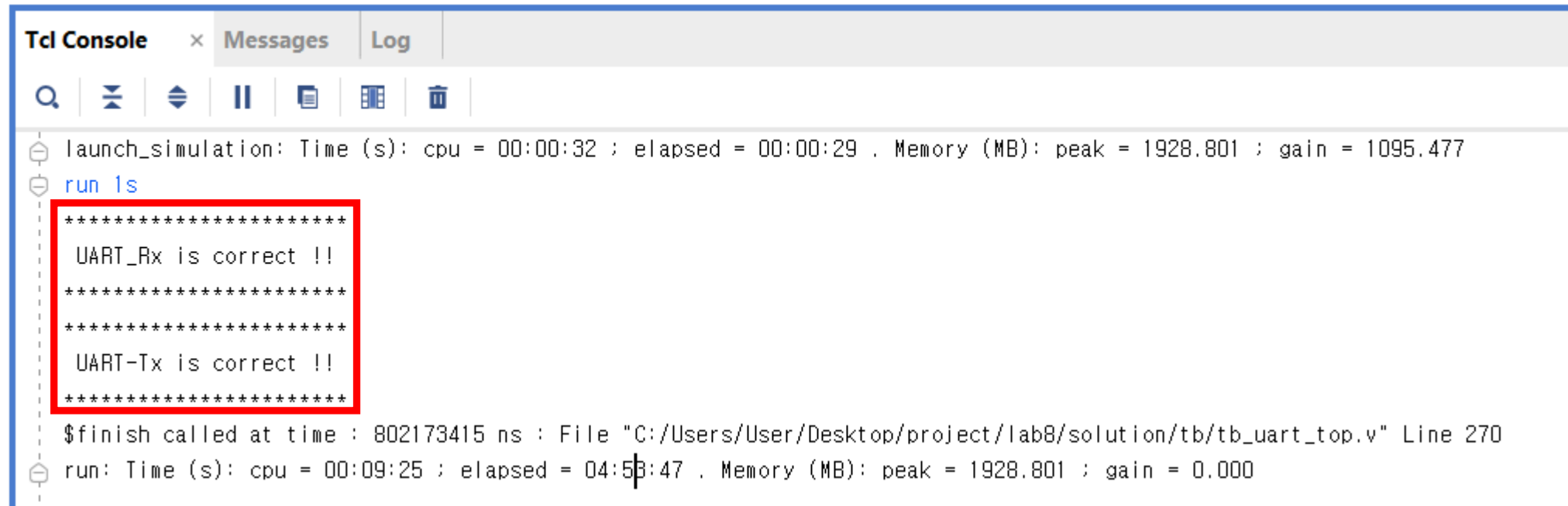


| 기타 구현 방법

- Always block 하나로도 구현 가능하므로 자유롭게 구현 하길 바람
- 다만 실험 설명 Slide 에 나온 동작 규칙을 지키길 바람

Simulation 검증

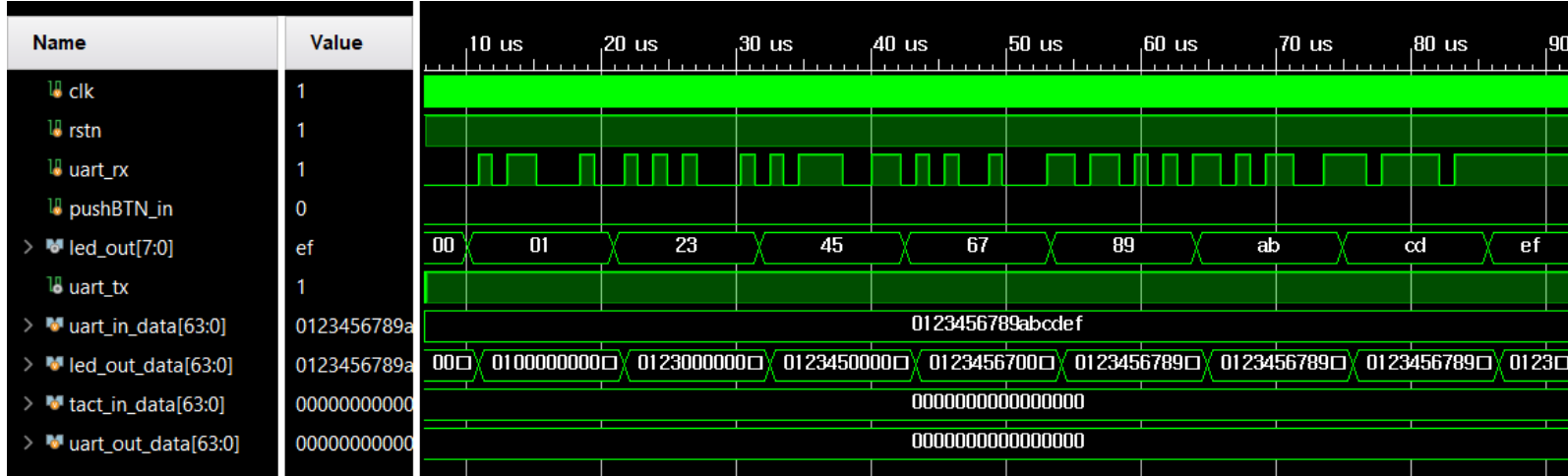
- “tb_UART_top.v” testbench 파일을 추가
- Simulation 실행 후 , Tcl console 창에 아래와 같은 결과가 나오면 보드에 올려서 검증 (Run Time : 201 ms 이상)



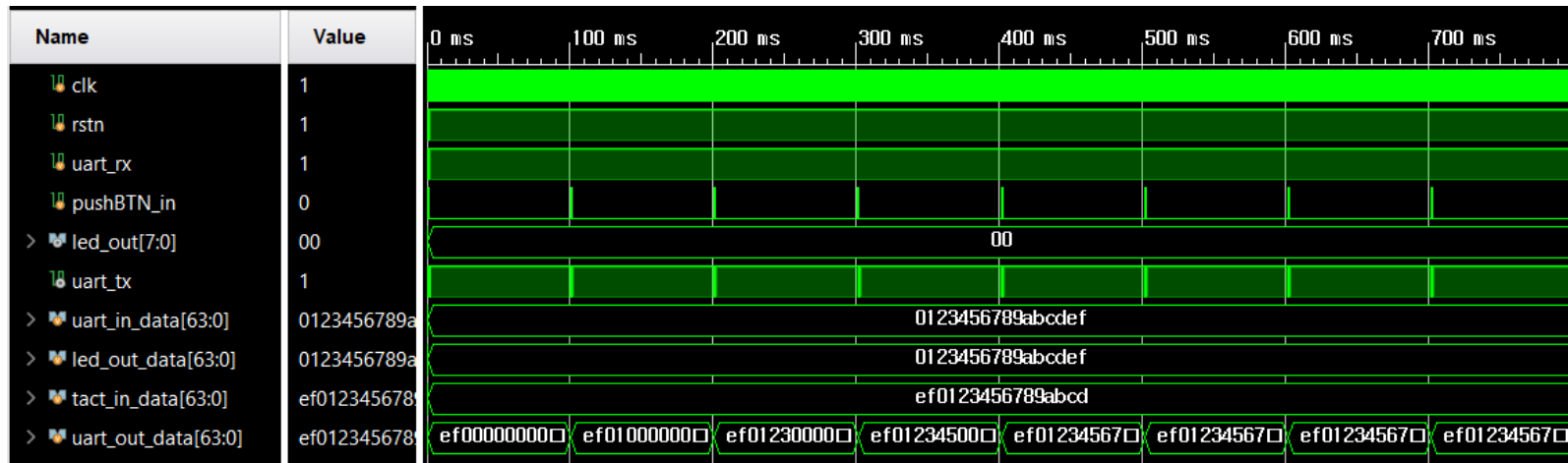
```
Tcl Console x Messages Log
[Icons]
launch_simulation: Time (s): cpu = 00:00:32 ; elapsed = 00:00:29 . Memory (MB): peak = 1928.801 ; gain = 1095.477
run ls
*****
UART_Rx is correct !!
*****
*****
UART-Tx is correct !!
*****
$finish called at time : 802173415 ns : File "C:/Users/User/Desktop/project/lab8/solution/tb/tb_uart_top.v" Line 270
run: Time (s): cpu = 00:09:25 ; elapsed = 04:53:47 . Memory (MB): peak = 1928.801 ; gain = 0.000
```

Simulation 검증

■ UART-Rx part Simulation Waveform



- UART-Tx part Simulation Waveform



보드 검증

- Step 1: Bitstream 을 생성하고 보드에 탑재하는 과정은 Lab.8 을 참고바람
- Step 2: Jupyter-notebook 을 이용하여 “ Lab09_UART_Tutorial.ipynb ” 파일을 실행
- Step 3: “shift+enter” 혹은 메뉴 상단 “ Run ” 클릭하여 실행
- Step 4: Random 하게 생성된 8bit 을 8 회 전송하여 , 보드의 LED 에 나타난 값과 PC(Jupyter-notebook) 에서 송신한 값이 같으면 UART-Rx 검증 완료
- Step 5: 보드에서 Slide Switch 를 이용하여 임의의 8bit 을 세팅하고 , Push button 을 눌러 PC 로 데이터를 전송한 뒤 , 두 값을 비교하여 같으면 UART-Tx 검증 완료

실험 결과 및 제출 방법

■ 제출 방법

1) 제출 파일

- 구현한 “**uart.v**” 파일
- simulation 화면과 Tcl Console 창의 UART_Rx_correct, UART_Tx_Correct 부분을 같이 Capture 한 이미지 파일
- 동영상 파일 : Jupyter Notebook 에서 전송한 Data 가 보드의 LED 에서 동일하게 표현되는 것과 , 보드에서 전송한 데이터가 Jupyter Notebook 에서 동일하게 나타나는 것을 녹화

2) 파일 이름 : “ 학번 _ 이름 _lab9.zip”

Ex) 2022-12345_ 홍길동 _lab9.zip

3) 제출 기한

11 월 13 일 (일요일) 23:59 까지

3.

Jupyter 사용법

Backup slide

■ Jupyter 사용방법

- Lab09_UART_Tutorial.ipynb 파일과 scale_uart.py 파일을 한 폴더에 다운 받는다 .
- Jupyter 를 실행하여 Jupyter web browse 가 열리면 “ Lab09_UART_Tutorial.ipynb” 를 찾아 클릭



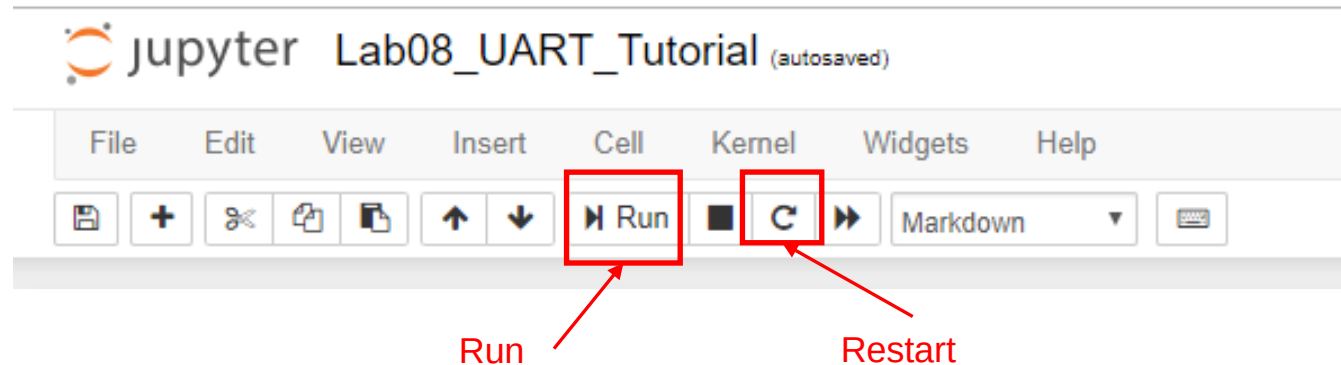
The screenshot shows the Jupyter web interface. At the top, there is a header with the Jupyter logo and buttons for 'Quit' and 'Logout'. Below the header, there are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, and it displays a file browser view of the /DSD directory. The interface includes a search bar, a 'Select items to perform actions on them.' prompt, and buttons for 'Upload', 'New', and a refresh icon. The file list is organized into columns: Name, Last Modified, and File size. The files are listed as follows:

Name	Last Modified	File size
..	몇 초 전	
parameter	6일 전	
Lab09_UART_Tutorial.ipynb	18시간 전	8.5 kB
sw_modeling_mnist.ipynb	14일 전	61.4 kB
module.py	14일 전	1.39 kB
README.txt	14일 전	540 B
scale_uart.py	일 년 전	991 B
setup_mnist.py	14일 전	2.78 kB

Backup slide

■ Jupyter 사용방법

- 해당 block 을 선택하여 “ Run” 혹은 : “shift+enter” 클릭 (다시 시작할 경우 “ Restart” 클릭)
- 이번 Lab 에서는 상단 block 부터 순차적으로 실행한다 .



Backup slide

- Jupyter 사용방법
 - Port 확인

How to use the UART communications

Explanation of UART: Please refer the following site.

https://en.wikipedia.org/wiki/Universal_asynchronous_receiver-transmitter

Then, how to use UART in python?

```
def port_list():
    os_name = platform.system()
    if "Windows" in os_name:
        print("Current OS: Windows")
        ports = ['COM%s' % (i+1) for i in range(256)]
    elif "Linux" in os_name:
        print("Current OS: Linux")
        ports = glob.glob('/dev/tty[A-Za-z]*')
    elif "Darwin" in os_name:
        print("Current OS: Mac")
        ports = glob.glob('/dev/tty.*')
    result = []
    for p in ports:
        try:
            s = serial.Serial(p)
            s.close()
            result.append(p)
        except (OSError, serial.SerialException):
            pass
    print(result)
    return result
```

Port 확인

```
plist = port_list()
Current OS: Windows
['COM13']
```

And please refer the following site for more detailed information about pyserial Python-serial communication API.

https://pyserial.readthedocs.io/en/latest/pyserial_api.html

```
# First, Set the connection configuration and Port
# PORT-name may be vary depending on your systems.
# USE USB serial port.

PORT_NAME = 'COM13'
for pname in plist:
    try:
        SU = Scale_UART(str(pname))
        print("%s port connected!" % (pname))
        break
    except serial.SerialException:
        print("%s port cannot be connected." % (pname))
```

Port 연결 성공확인

COM13 port connected!

Backup slide

- Jupyter 사용방법
 - Test 할 Random data 생성

Write data

```
In [7]: packet = []  
        for i in range(8):  
            packet.append(np.random.randint(256))  
        print(packet)  
[26, 220, 78, 195, 13, 110, 188, 17]
```

Backup slide

■ Jupyter 사용방법

- 8bit data 를 각각 FPGA 에 전송
- Binary bit 으로 보드 상에서 LED 로 정확하게 입력 되었는지 확인 가능

```
: SU.snd_byte(packet[0])  
print("Packet[0]:\tDecimal value: %d, Hex val: %s" % (packet[0], f'{packet[0]:08b}'))
```

Packet[0]: Decimal value: 26, Hex val: 00011010

```
: SU.snd_byte(packet[1])  
print("Packet[1]:\tDecimal value: %d, Hex val: %s" % (packet[1], f'{packet[1]:08b}'))
```

Packet[1]: Decimal value: 220, Hex val: 11011100

```
: SU.snd_byte(packet[2])  
print("Packet[2]:\tDecimal value: %d, Hex val: %s" % (packet[2], f'{packet[2]:08b}'))
```

Packet[2]: Decimal value: 78, Hex val: 01001110

```
: SU.snd_byte(packet[3])  
print("Packet[3]:\tDecimal value: %d, Hex val: %s" % (packet[3], f'{packet[3]:08b}'))
```

Packet[3]: Decimal value: 195, Hex val: 11000011

```
: SU.snd_byte(packet[4])  
print("Packet[4]:\tDecimal value: %d, Hex val: %s" % (packet[4], f'{packet[4]:08b}'))
```

Packet[4]: Decimal value: 13, Hex val: 00001101

```
: SU.snd_byte(packet[5])  
print("Packet[5]:\tDecimal value: %d, Hex val: %s" % (packet[5], f'{packet[5]:08b}'))
```

Packet[5]: Decimal value: 110, Hex val: 01101110

```
: SU.snd_byte(packet[6])  
print("Packet[6]:\tDecimal value: %d, Hex val: %s" % (packet[6], f'{packet[6]:08b}'))
```

Packet[6]: Decimal value: 188, Hex val: 10111100

```
: SU.snd_byte(packet[7])  
print("Packet[7]:\tDecimal value: %d, Hex val: %s" % (packet[7], f'{packet[7]:08b}'))
```

Packet[7]: Decimal value: 17, Hex val: 00010001

Backup slide

■ Jupyter 사용방법

- Slide switch 를 이용하여 임의의 8bit Data 를 만든 후 Push button 을 누르면 데이터가 PC 로 전송됨
- PC 에서 수신한 값과 보드에서 세팅한 Data 가 동일한지 확인

Read data

```
return_val = SU.rcv_packet(8)
```

```
26  
220  
78  
195  
13  
110  
188  
17
```

Error

- Pyserial 이 설치되지 않은 경우 : 자료를 참조하여 설치

```
import glob
import platform
import numpy as np
import time
import serial
from scale_uart import *
```

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-1-4235acf4b755> in <module>
      3 import numpy as np
      4 import time
----> 5 import serial
      6 from scale_uart import *

ModuleNotFoundError: No module named 'serial'
```

- 경로에 scale_uart.py 이 없는 경우 : scale_uart.py 를 Lab09_UART_Tutorial 과 동일 폴더에 위치

```
import glob
import platform
import numpy as np
import time
import serial
from scale_uart import *
```

```
-----
ModuleNotFoundError                                Traceback (most recent call last)
<ipython-input-2-4235acf4b755> in <module>
      4 import time
      5 import serial
----> 6 from scale_uart import *

ModuleNotFoundError: No module named 'scale_uart'
```