Lab 5: UART Communications

National Chiao Tung University
Chun-Jen Tsai
10/23/2015

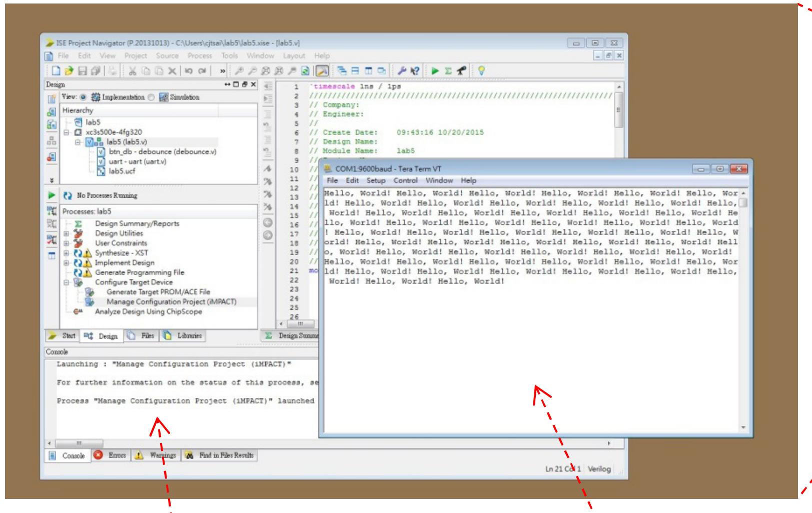
Lab 5: UART Communications

- In this lab, you will design a circuit to do the following things:
  - Read a small text file (up to 256 bytes) from the UART port
  - Convert the received text to an all upper-case text
  - Wait for the user to press the WEST button
  - Print the text through the UART to a terminal window on PC
    - Note: the text must be printed only once for each button press

- You will demo the design to your TA during the lab hours on 11/2

Setup of Lab5

- Lab 5 test the communications between the PC and the FPGA:


ISE Project Navigator (P.20131013) - C:\Users\cjtsai\lab5\lab5.xise - [lab5.v]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
lab5
xc3s500e-4fg320
lab5 (lab5.v)
btn_db - debounce (debounce.v)
uart - uart (uart.v)
lab5.ucf
No Processes Running
Processes: lab5
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Generate Target PROM/ACE File
Manage Configuration Project (iMPACT)
Analyze Design Using ChipScope
Start Design Files Libraries
Design Summary
Console
Console Errors Warnings Find in Files Results
Ln 21 Col 1 Verilog
COM1:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date:    09:43:16 10/20/2015
// Design Name:
// Module Name:    lab5
```

```
Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, Wor
ld! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello,
 World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! He
llo, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World
! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, W
orld! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hell
o, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World!
Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, Wor
ld! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello,
 World! Hello, World! Hello, World!
```

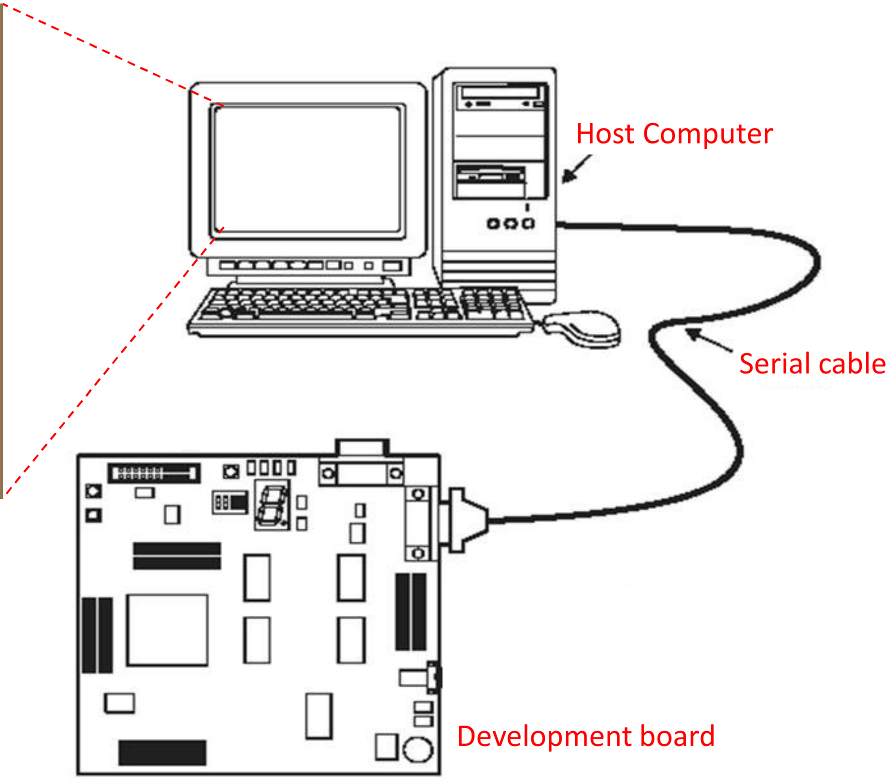
```
Launching : "Manage Configuration Project (iMPACT)"

For further information on the status of this process, se

Process "Manage Configuration Project (iMPACT)" launched
```

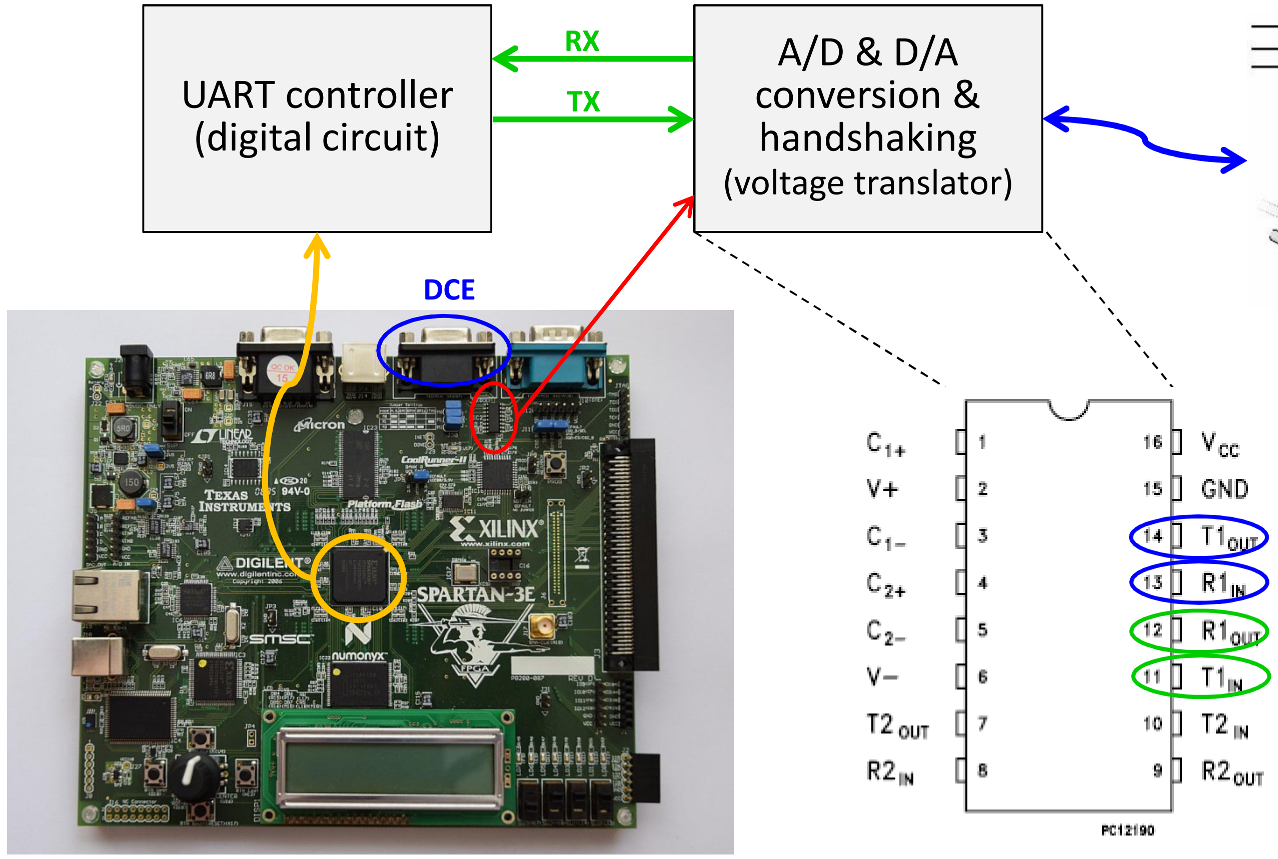
ISE Project Navigator

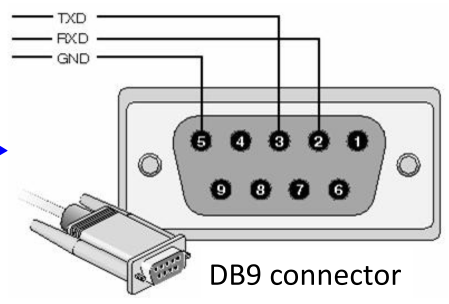
Tera Term terminal emulation program



UART Communication

- Universal Asynchronous Receiver/Transmitter (UART) is one of the most popular I/O devices in small systems





Note: in our board, it uses a simple voltage translator IC without any HW handshaking control

UART Physical Layer

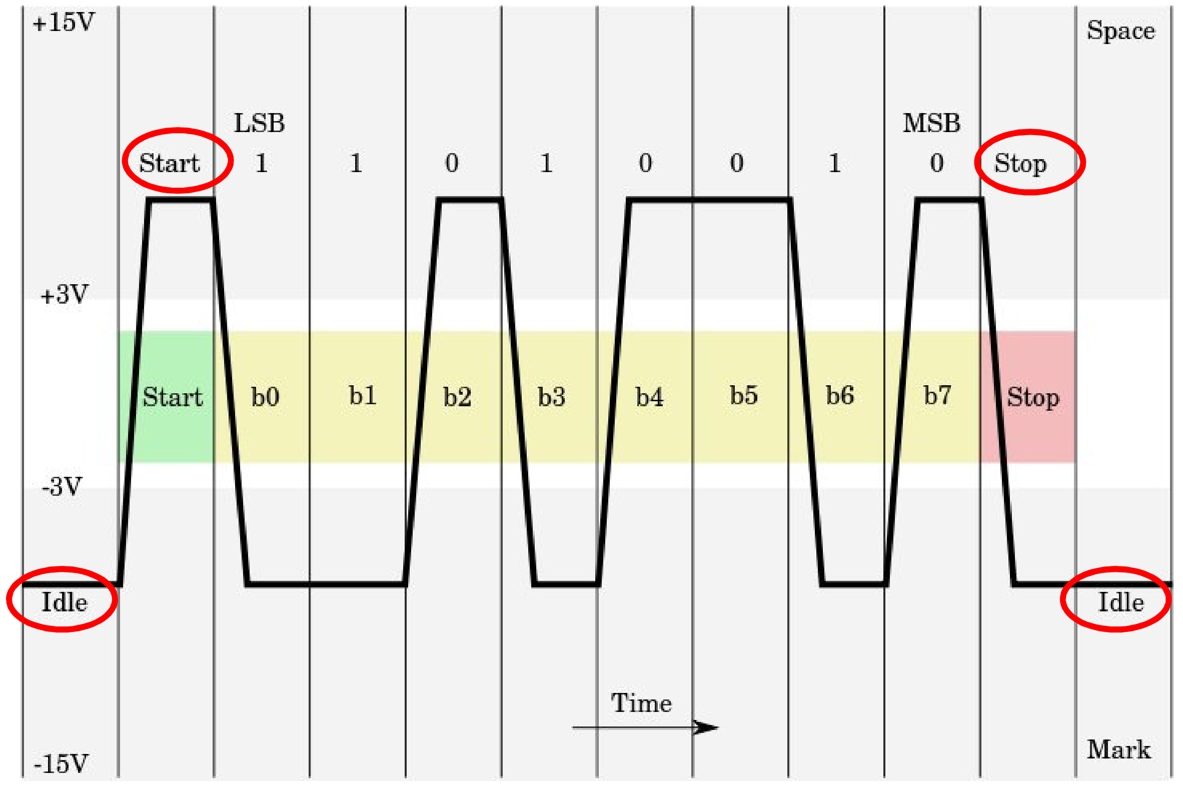
- UART is a asynchronous transmission standard, thus, there is no common clock signal for synchronization
- The most popular physical layer for UART transmission line is the RS-232 standard
  - Common baud rate for RS-232 signals ranges from 4800 bps to 115200 bps
  - RS-232 voltages are $(-15, -3)$ for ‘1’ and $(3, 15)$ for ‘0’

UART Link Layer

- The serial line is 1 when it is idle
- The transmission starts with a start bit, which is 0, followed by data bits and an optional parity bit, and ends with stop bits, which are 1
- The number of data bits can be 6, 7, or 8
- The optional parity bit is used for error detection
  - For odd parity, it is set to 0 when the data bits have an odd number of 1's
  - For even parity, it is set to 0 when the data bits have an even number of 1 's
- The number of stop bits can be 1, 1.5, or 2

RS-232 Transmission Example

- An example of the RS-232 transmission signals:



Out-of-Band Parameter Setting

- UART control parameters such as: bit-rate, #data bits, #stop bits, and types of parity-check must be set on both side of the serial transmission line before the communication begins

- Implicit clocks must be generated on both sides for correct transmission
  - Bit rate per second (bps), or baud rate, is used to imply the clock on both end of the transmission line
  - Baud rates must be two’s multiples of 1200, e.g., 2400, 4800, 9600, …, 115200, etc.
  - This clock is often called the baud rate generator

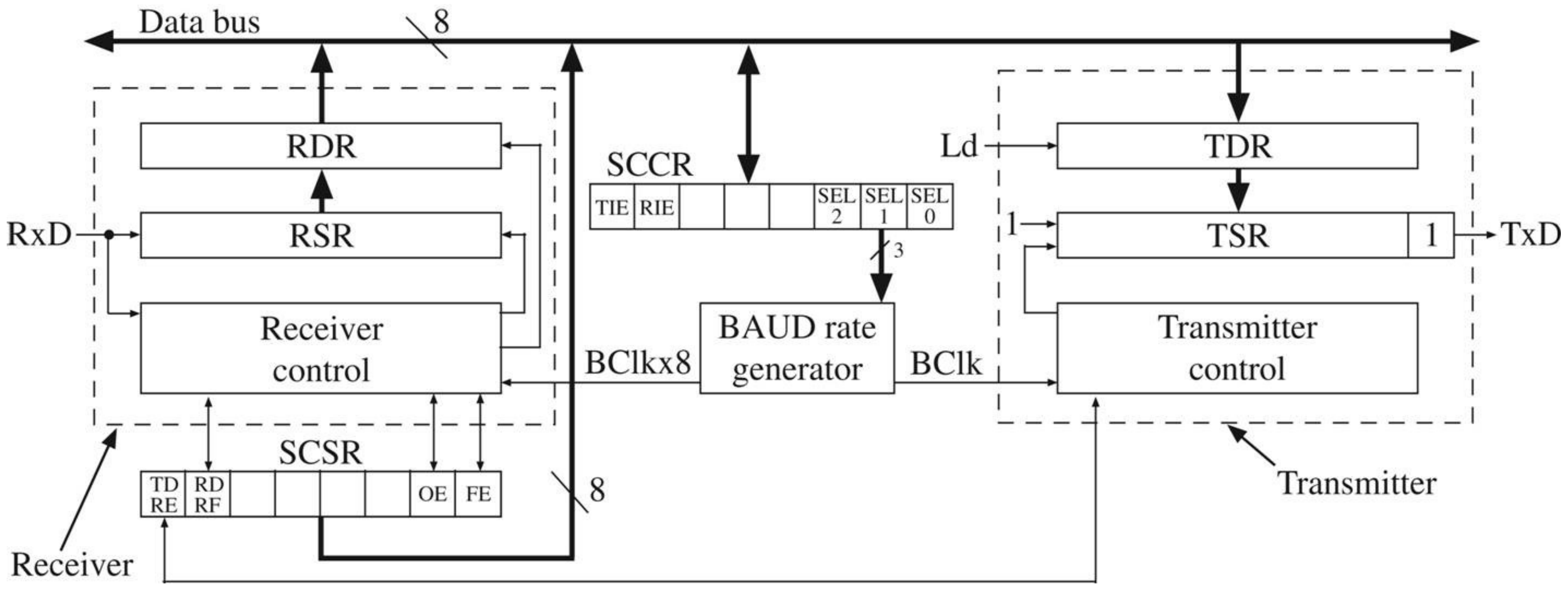
UART Controller

- A UART controller performs the following tasks
  - Convert 8-bit parallel data to a serial bit stream and vice versa
  - Insert (or remove) start bit, parity bit, and stop bit for every 8 bits of data
  - Maintain a local clock for data transmission at correct rate

- A UART controller includes a transmitter, a receiver, and a baud rate generator
  - The transmitter is essentially a special shift register that loads data in parallel and then shifts it out bit by bit at a specific rate
  - The receiver, on the other hand, shifts in data bit by bit and then reassembles the data

An Example of UART Controller

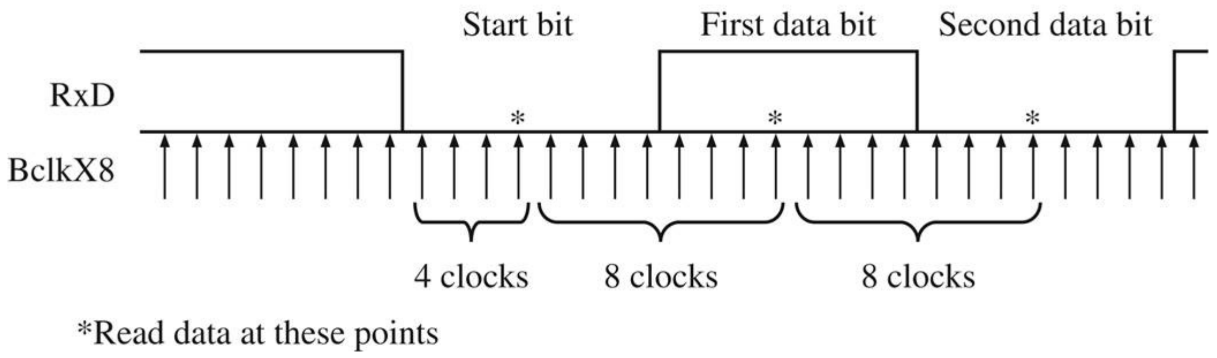
- The following diagram shows a typical UART controller:



RSR – Receive shift register
TSR – Transmit shift register
RDR – Receive data register
TDR – Transmit data register
SCCR – Serial communications control register
SCSR – Serial communications status register

Clock Synchronization Problem

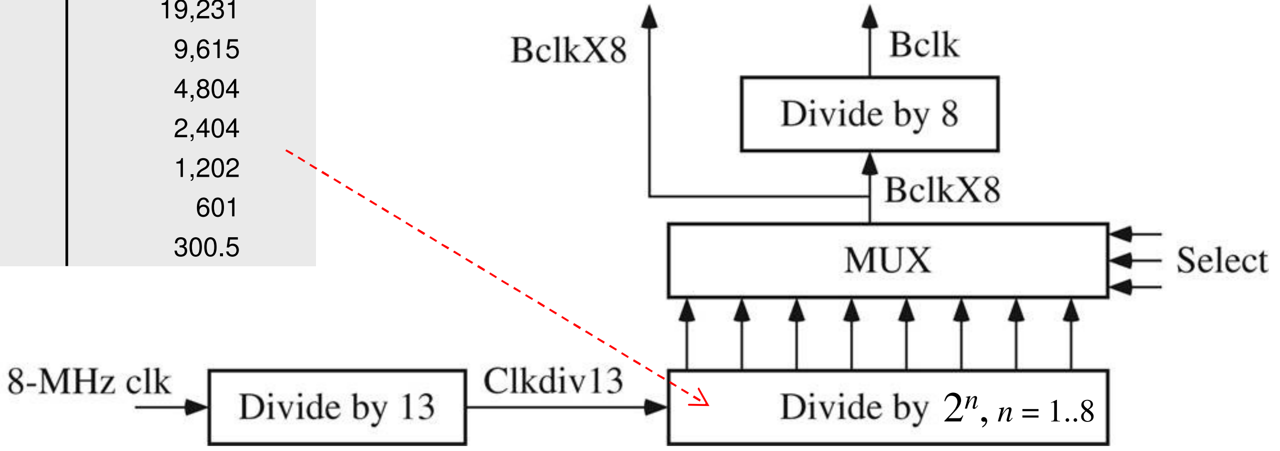
- Since there is no explicit clock signal between the transmitter and the receiver, the receiver can not simply read incoming bits based on its system clock
- To solve this problem, we sample the incoming data multiple times per baud rate clock cycle
  - Typical up-sampling rate: 8 or 16
- Take 8× sampling for example: the fourth sample of each bit time will be read as a data bit



Baud Rate Generator

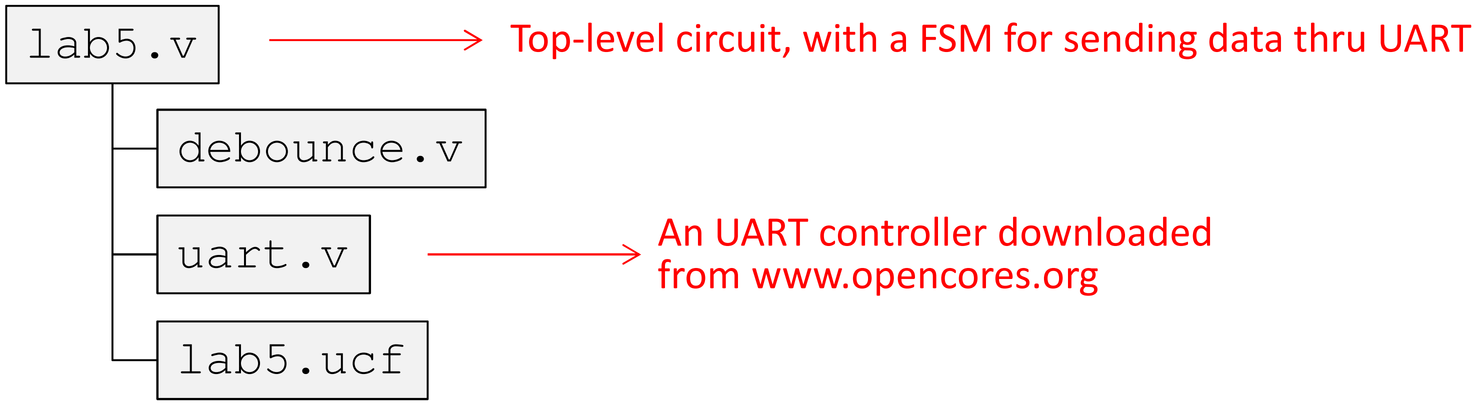
- We must generate baud rate clock (`bclk`) and 8- or 16-times baud rate clock from system clock, `clk`
  - If highest baud rate = 38400, 38400×8 = 307200
  - If the system clock is 8MHz, the divider is 8M / 307200 = 26.04
  - If a divider of 26 is used, the actual baud rate is as follows

Select Bits of clock divider	Baud Rate (Bclk)
000	38,462
001	19,231
010	9,615
011	4,804
100	2,404
101	1,202
110	601
111	300.5



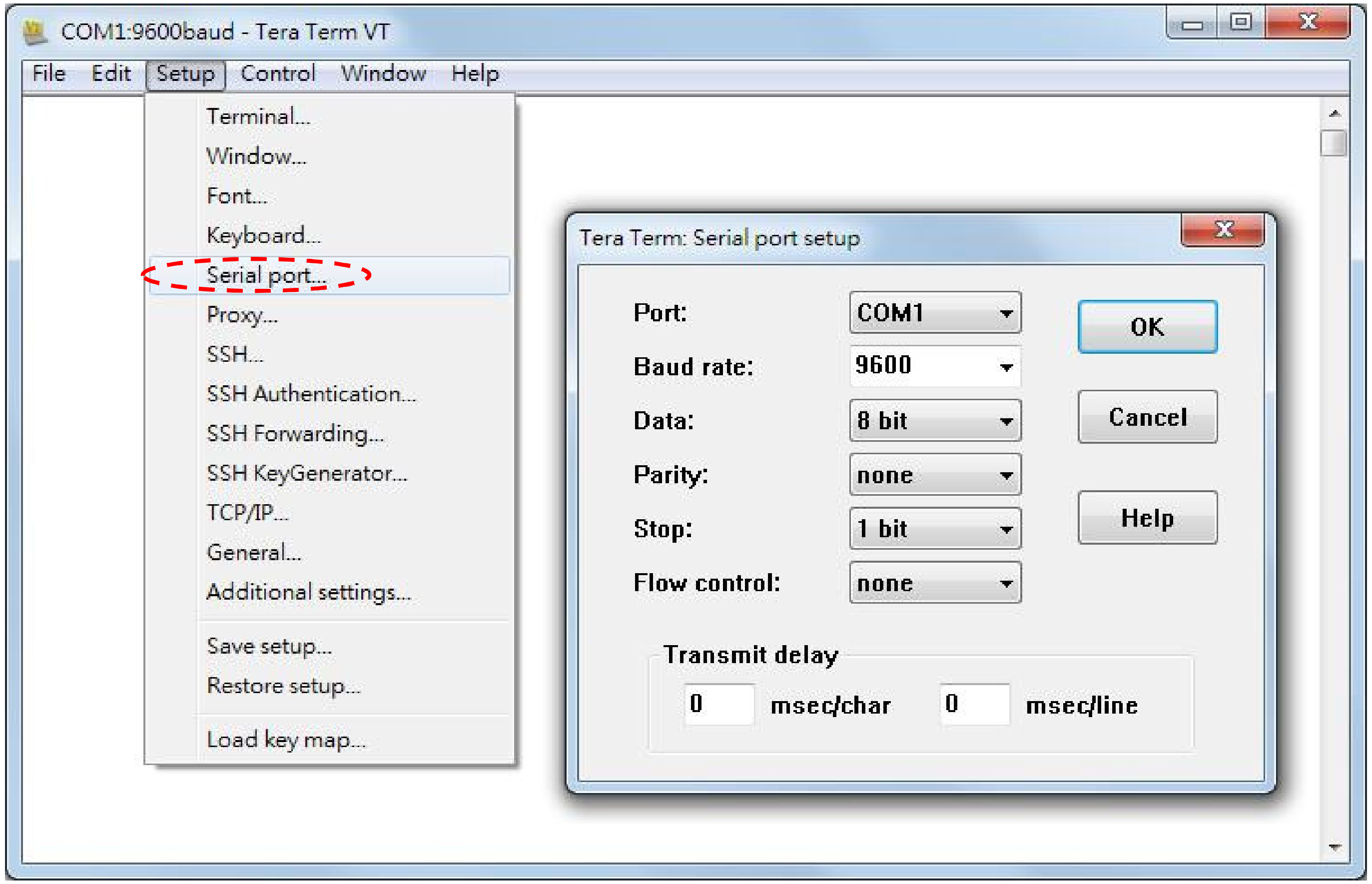
What is in the Lab5 Package

- The package contains a project that shows you how to do “Hello, World!” in hardware
- The source tree structure is as follows:


lab5.v
Top-level circuit, with a FSM for sending data thru UART
debounce.v
uart.v
An UART controller downloaded from www.opencores.org
lab5.ucf

Tera Term

- On the PC connected to the Spartan-3E board, we use a TeraTerm to send/receive data through RS-232:

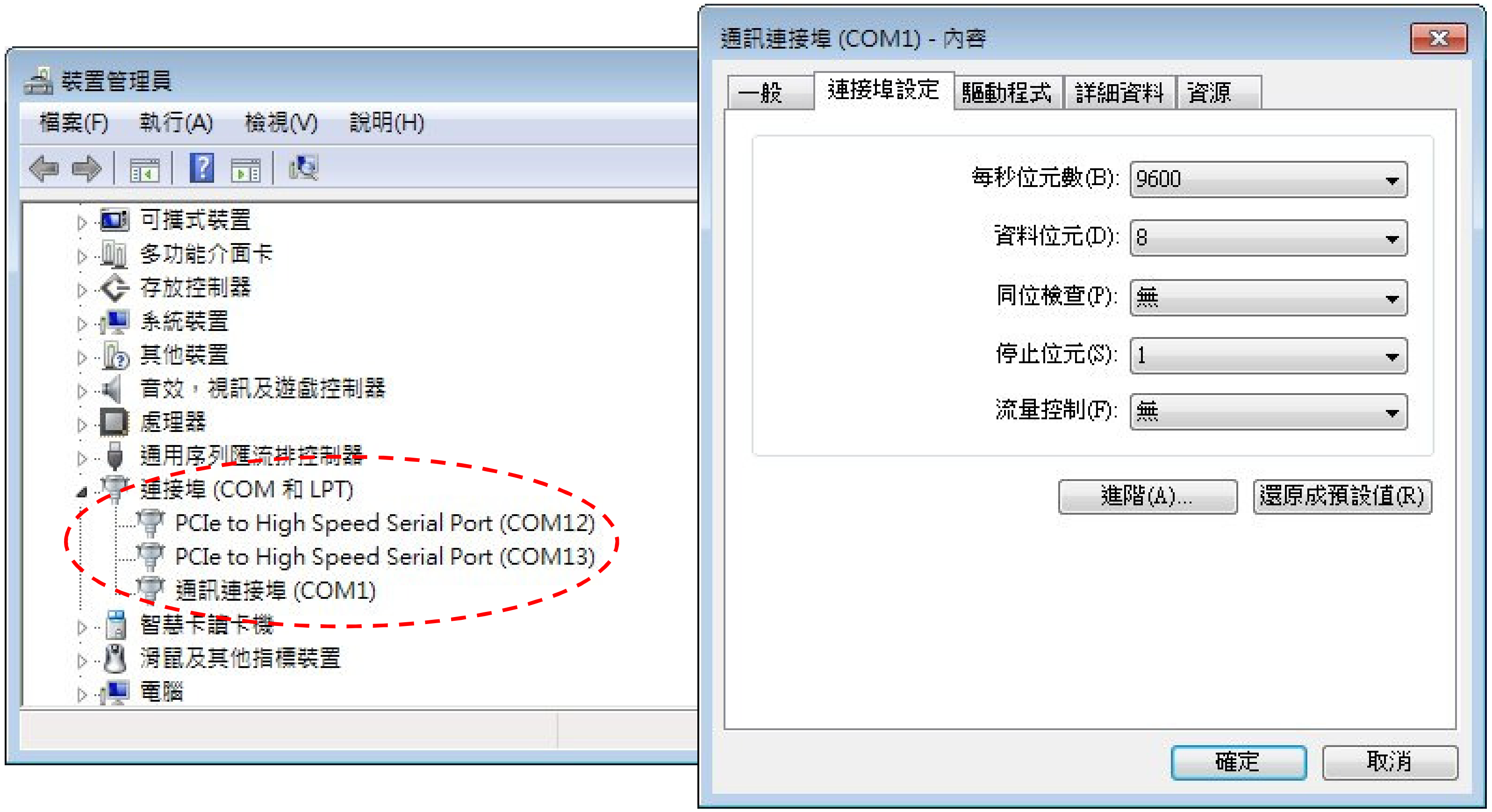


Port: COM1
Baud rate: 9600
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none

Transmit delay
0 msec/char 0 msec/line

COM PORT Number

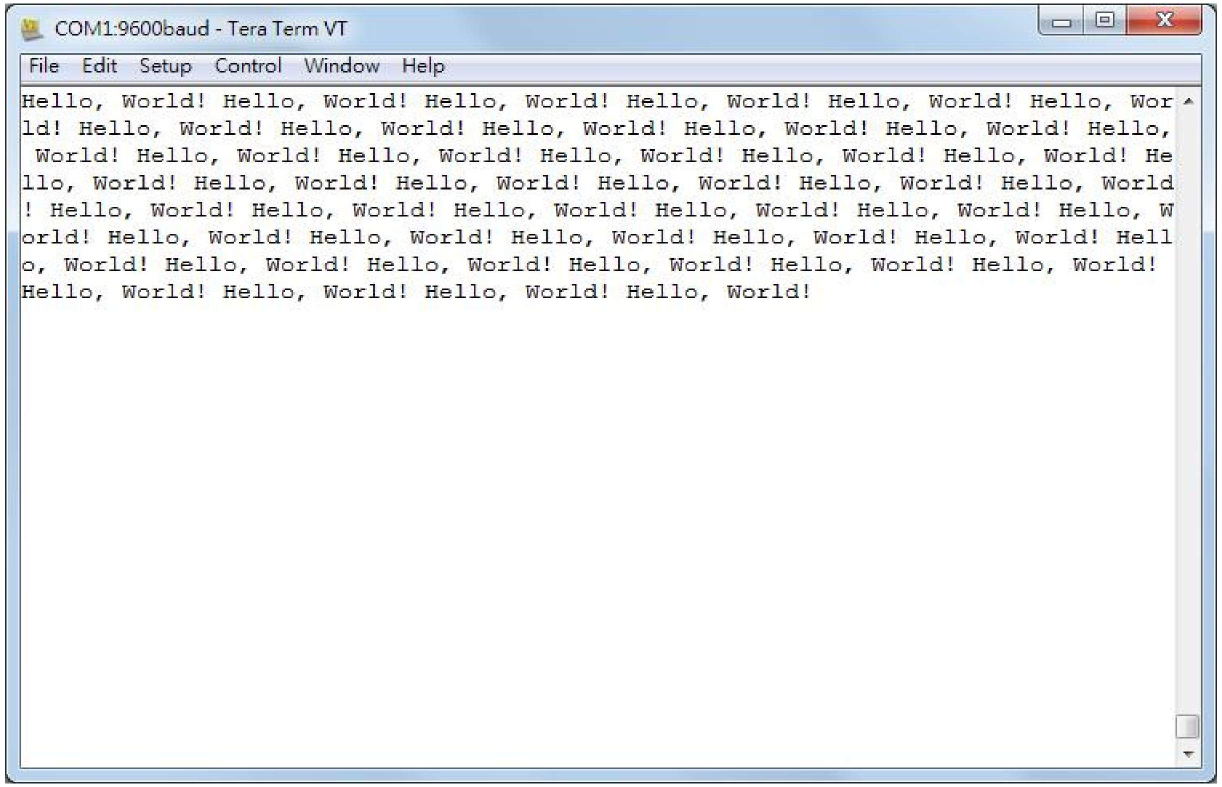
- The COM port number of your computer can be obtained from the device manager as follows:



每秒位元數(B): 9600
資料位元(D): 8
同位檢查(P): 無
停止位元(S): 1
流量控制(F): 無

Screen Shot

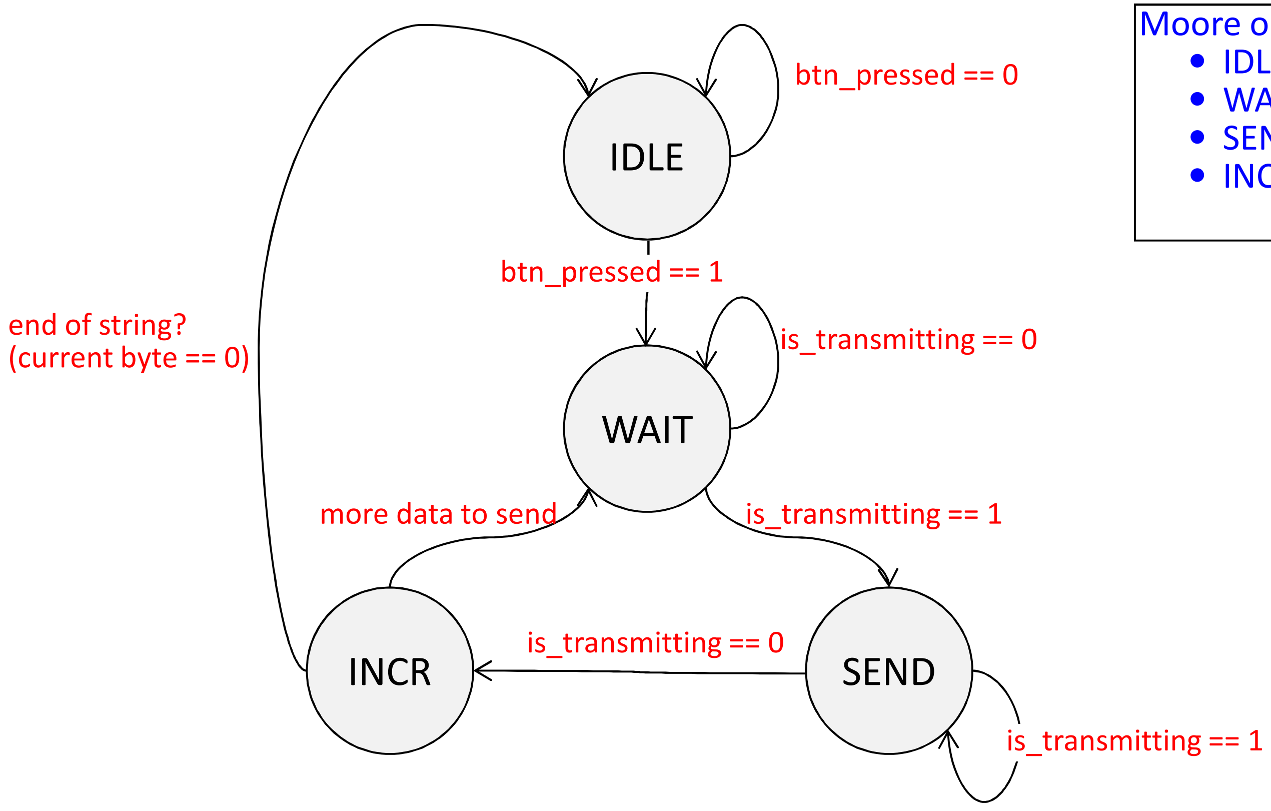
- The TeraTerm window shows the following output when the west button on the Sparan-3E board is pressed:


COM1:9600baud - Tera Term VT
File Edit Setup Control Window Help

```
Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, Wor
ld! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello,
 World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! He
llo, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World
! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, W
orld! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World! Hell
o, World! Hello, World! Hello, World! Hello, World! Hello, World! Hello, World!
Hello, World! Hello, World! Hello, World! Hello, World!
```

FSM of the Sample Code

- We can send data to the UART controller when it's free:

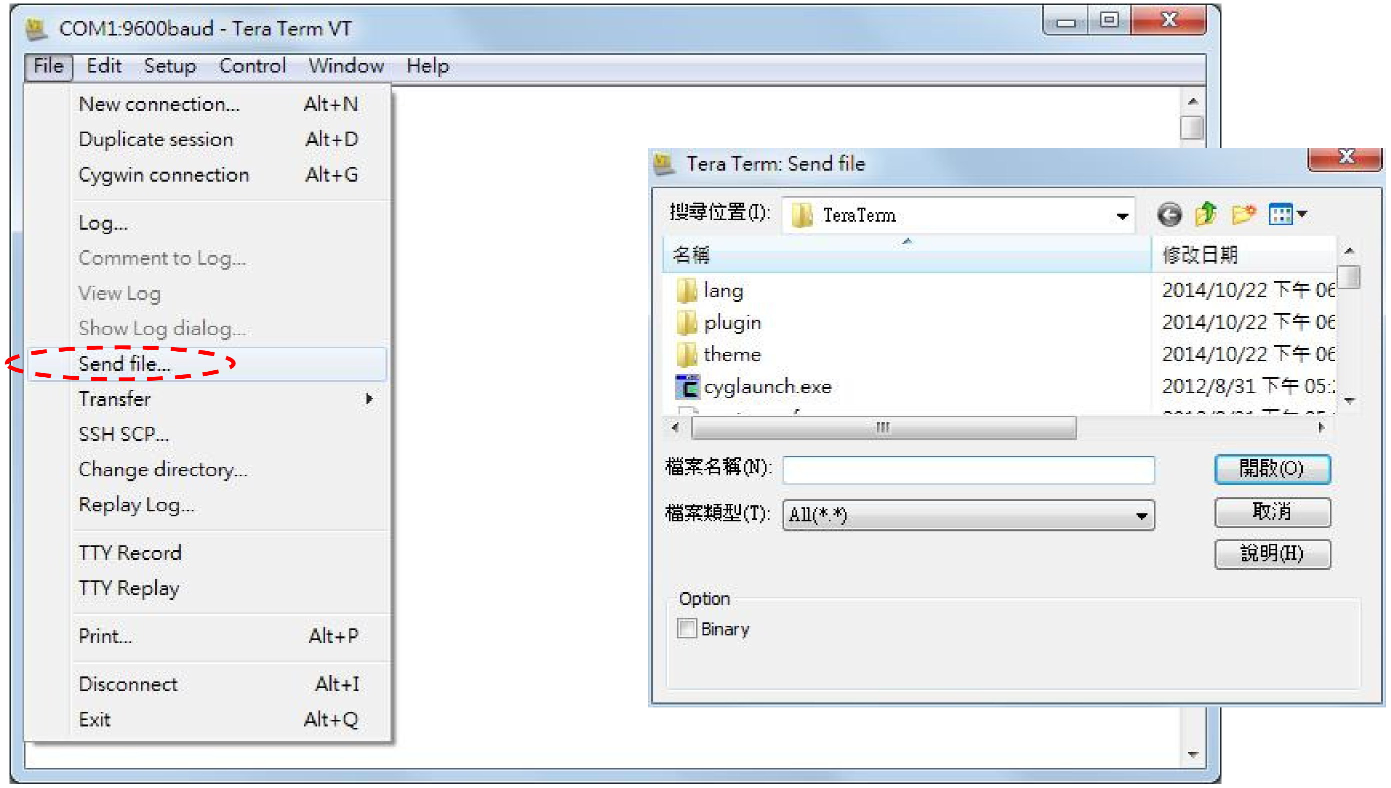


Moore output actions:

- IDLE : transmit $\leftarrow$ 0;
- WAIT: transmit $\leftarrow$ 1;
- SEND: transmit $\leftarrow$ 0;
- INCR : transmit $\leftarrow$ 0, send_counter++;

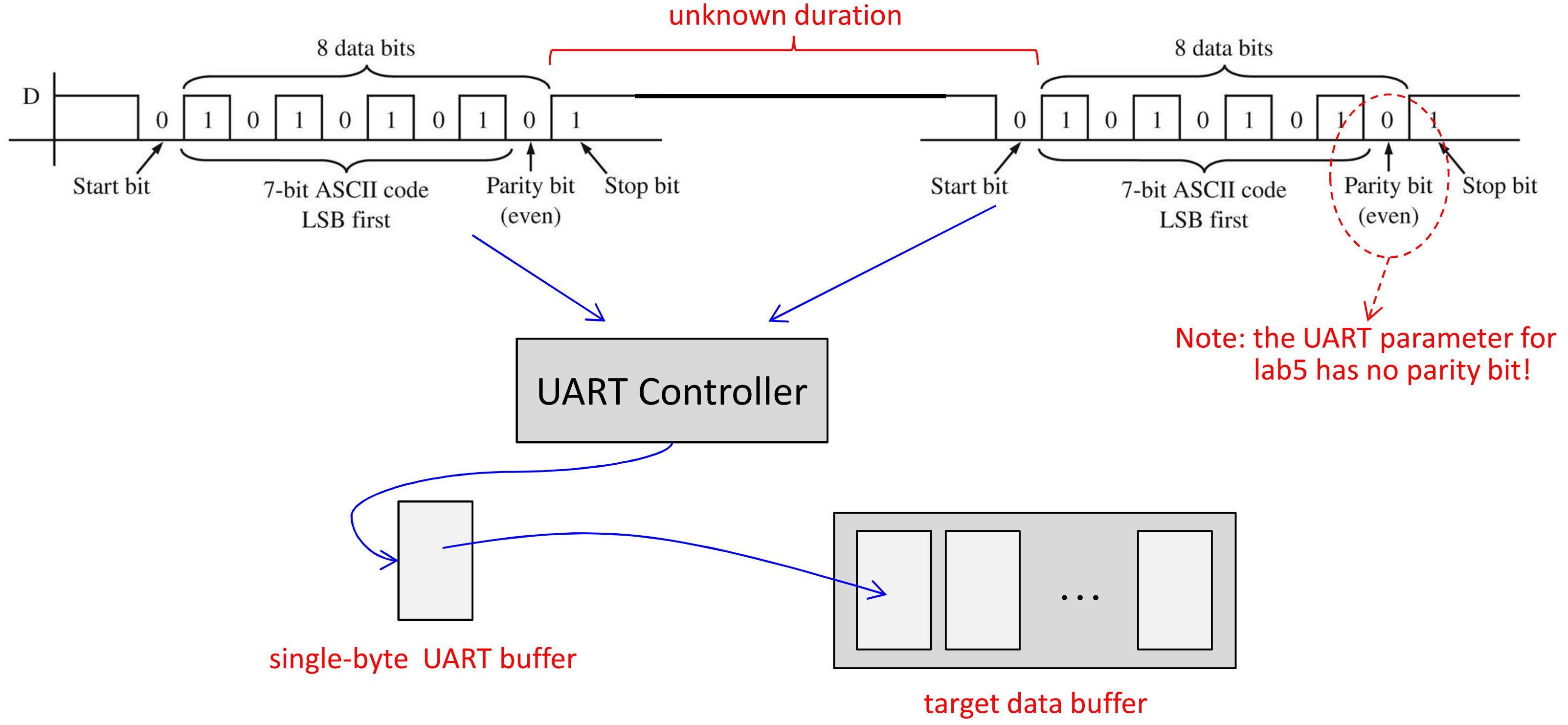
Your Main Task in Lab5

- ❑ Add Verilog code to store a text file sent from TeraTerm in a register buffer and convert it to upper-case text



Comments on the Receiver

- FPGA runs at 50MHz, but data arrives at 9600 bps
  $\rightarrow$ do not "duplicate" or "overwrite" the input data!
  - The "`received`" notification from the UART controller only lasts for one system clock cycle (20 nano seconds)



Reference

- Chapter 8 of P. Chu's *FPGA Prototyping by Verilog Examples*, Wiley, 2008.