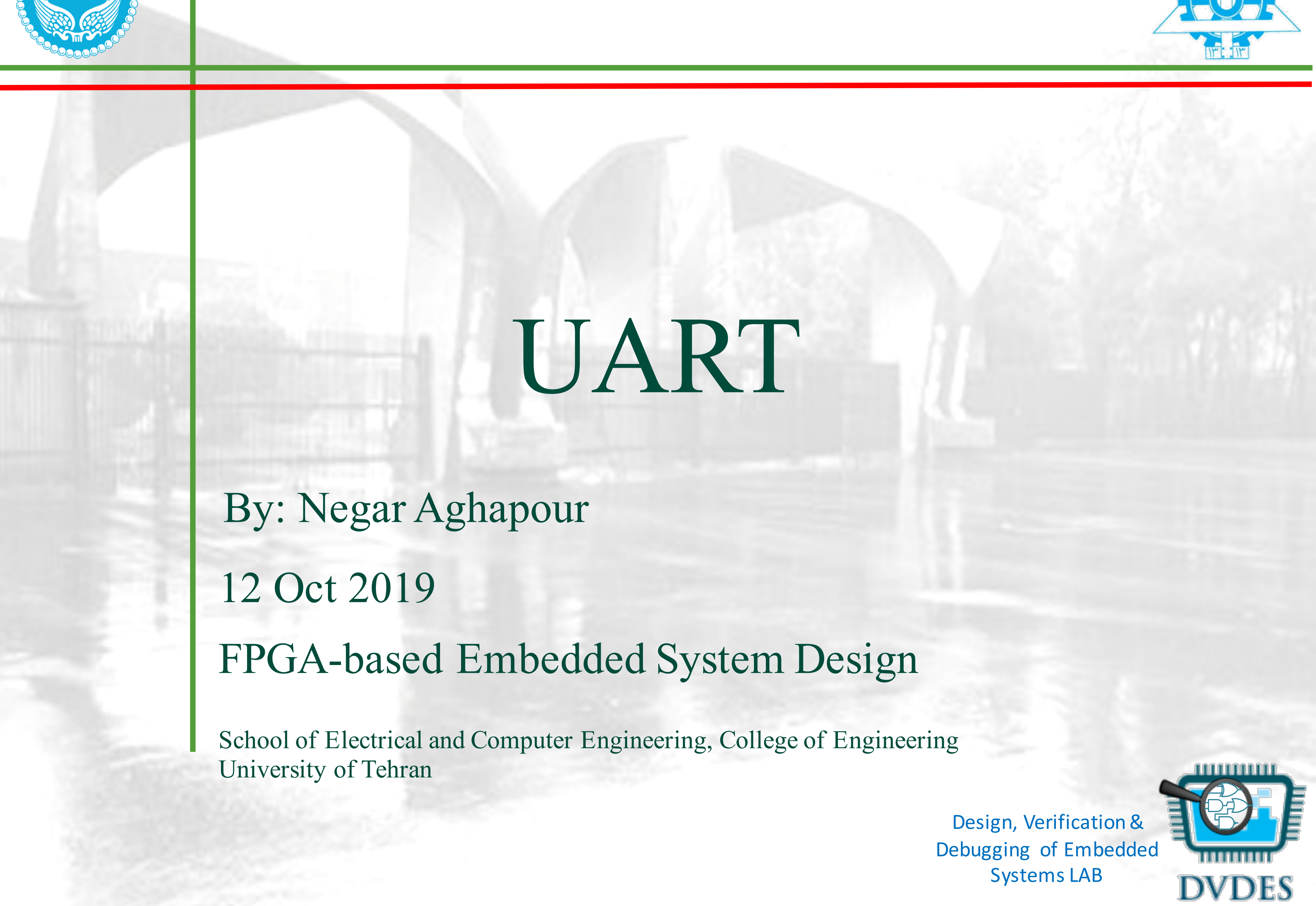


# UART

By: Negar Aghapour

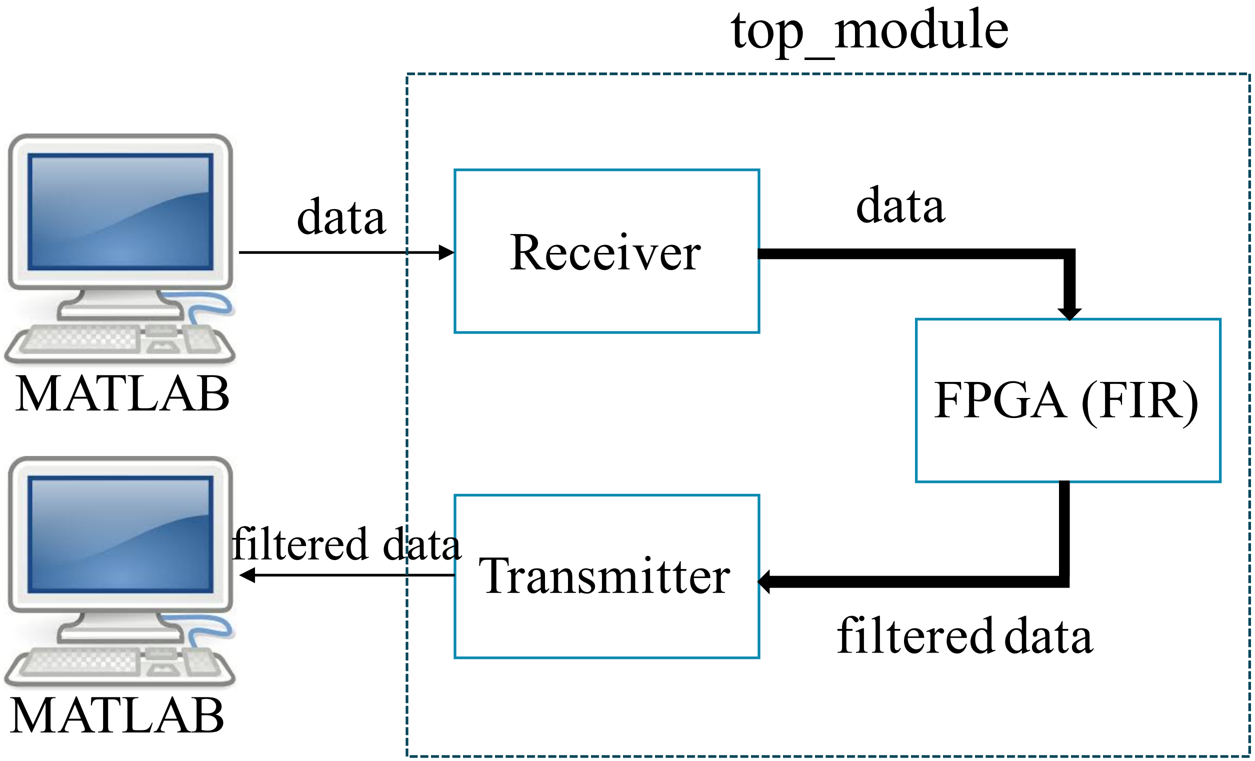
12 Oct 2019

FPGA-based Embedded System Design

School of Electrical and Computer Engineering, College of Engineering
University of Tehran

Design, Verification &
Debugging of Embedded
Systems LAB

# Lab1


top_module
data
Receiver
data
FPGA (FIR)
MATLAB
filtered data
Transmitter
filtered data
MATLAB

# Device Driver

- Computer program
- Operates or controls a particular type of device that is attached to a computer

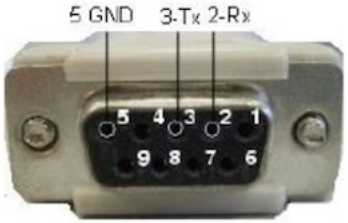
- Program device driver for serial connections based on RS232 protocol
- Using asynchronous UART communication

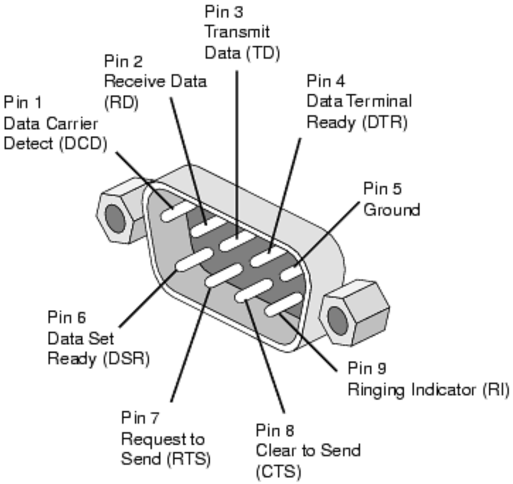
# UART

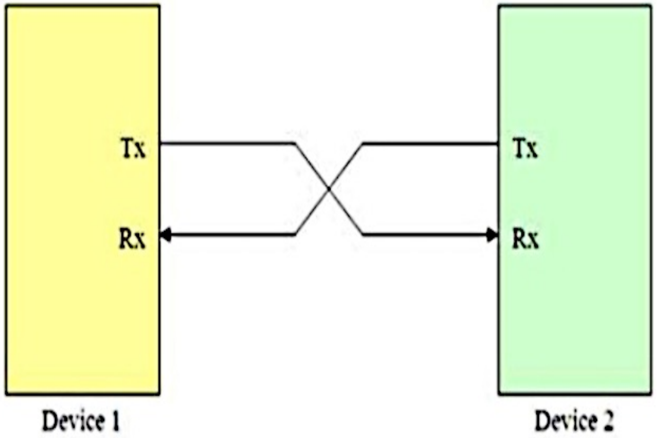
# UART

- Universal Asynchronous Receiver Transmitter
- The circuit sends parallel data through a serial line
- Serial communication without external clock signal
- In RS-232, serial port implemented with UART
- Very cheap communication
    - Needs one wire for serial communicate

# UART


5 GND
3-Tx
2-Rx
5
4
3
2
1
9
8
7
6


Pin 1
Data Carrier
Detect (DCD)
Pin 2
Receive Data
(RD)
Pin 3
Transmit
Data (TD)
Pin 4
Data Terminal
Ready (DTR)
Pin 5
Ground
Pin 6
Data Set
Ready (DSR)
Pin 7
Request to
Send (RTS)
Pin 8
Clear to Send
(CTS)
Pin 9
Ringing Indicator (RI)


Tx
Rx
Device 1
Tx
Rx
Device 2

# UART character transmission

Idle bus ⇢ Data frame ⇢ Idle bus

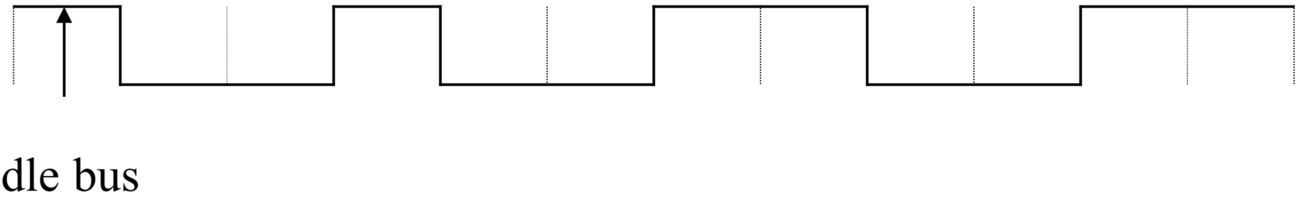
Data frame:
start bit ⇢ Data ⇢ parity (if needed) ⇢ stop bit

Idle state: The bus is in high voltage (*logic 1*)
Start bit: Put *logic 0* on bus
Data: Useful data
Parity: Is sent to check for transmission error
Stop bit: Put *logic 1* on bus

# UART character transmission

Idle bus ⇢ Data frame ⇢ Idle bus

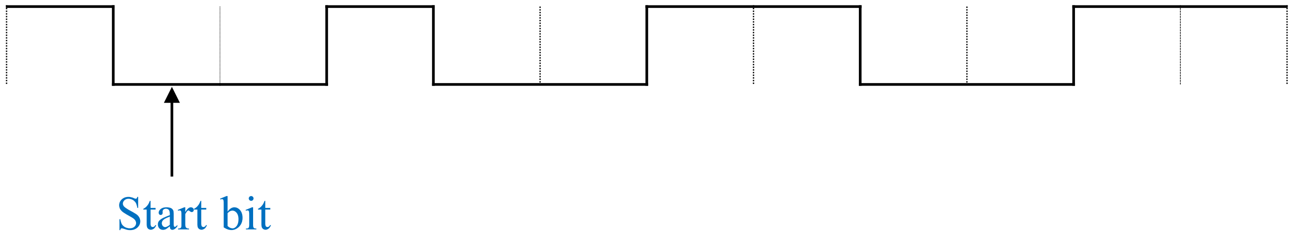
Start bit: 1 bit
Data: 5, 6, 7 or 8 bit
Parity: 0 or 1 bit
Stop bit: 1, 1.5 or 2 bit



# UART character transmission

Idle bus ⇢ Data frame ⇢ Idle bus

Start bit: 1 bit
Data: 5, 6, 7 or 8 bit
Parity: 0 or 1 bit
Stop bit: 1, 1.5 or 2 bit



# UART character transmission

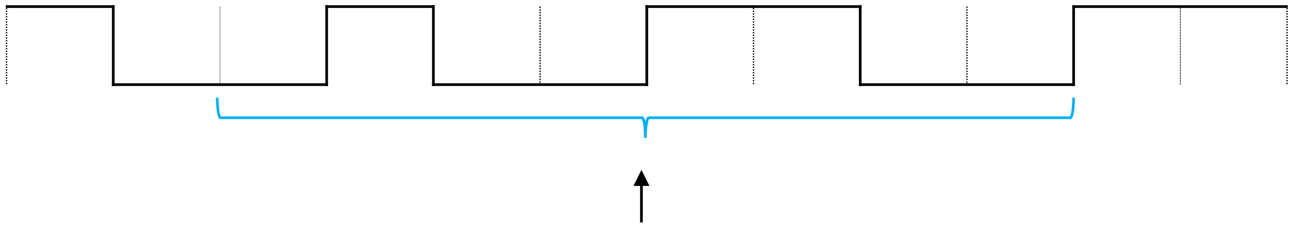
Idle bus ⇢ Data frame ⇢ Idle bus

Start bit: 1 bit

Data: 5, 6, 7 or 8 bit (from LSB to MSB)

Parity: 0 or 1 bit

Stop bit: 1, 1.5 or 2 bit

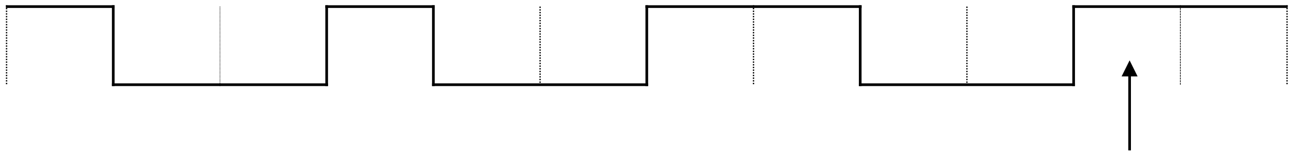


8 bit data $(00110010)_2$

# UART character transmission

Idle bus ⇢ Data frame ⇢ Idle bus

Start bit: 1 bit
Data: 5, 6, 7 or 8 bit
Parity: 0 or 1 bit
Stop bit: 1, 1.5 or 2 bit

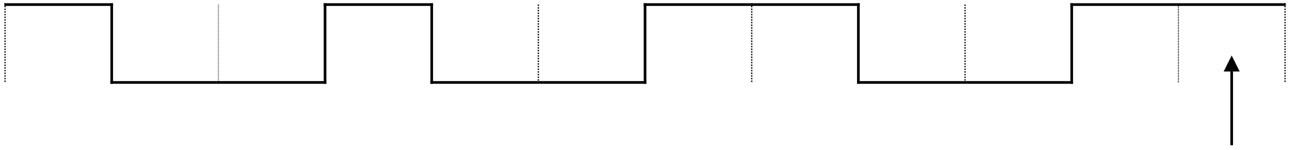


Even parity

# UART character transmission

Idle bus ⇢ Data frame ⇢ Idle bus

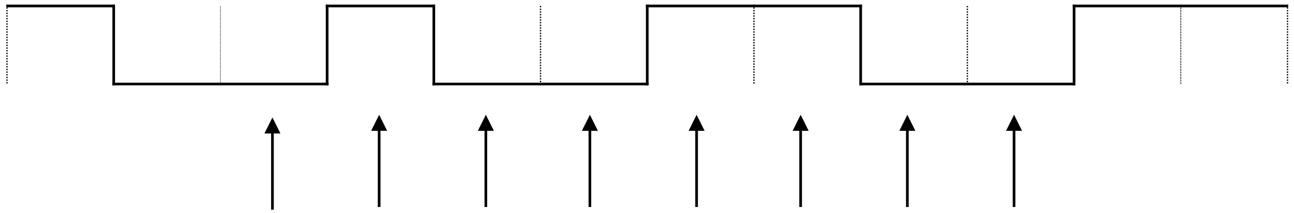
Start bit: 1 bit
Data: 5, 6, 7 or 8 bit
Parity: 0 or 1 bit
Stop bit: 1, 1.5 or 2 bit



Stop bit

# UART character transmission

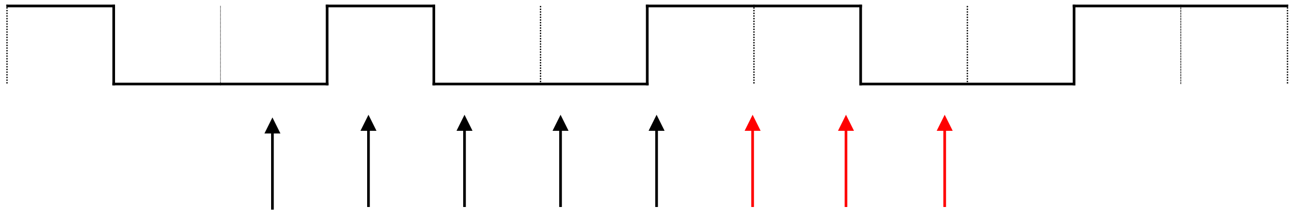
- ➢ Receiver should sample in middle of bits



- ✓ Transmitter and receiver agree on the same baud rate

# UART character transmission

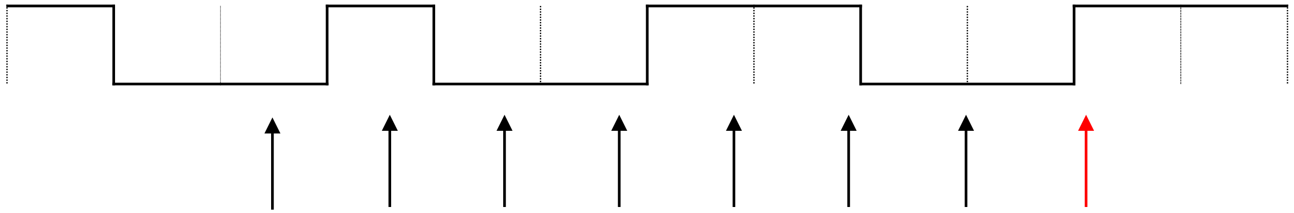
- ➢ If receiver samples quickly:



✗ Receiver receives wrong data

# UART character transmission

- ➢ If receiver samples slowly:



✗ Receiver receives wrong data

## Basics of serial communication

- **Bit rate:**
  Number of bits sent every second (BPS)
- **Baud rate:**
  Number of symbols sent every second

Standard bit rates:
100, 200, 300, 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 bps.

# Basics of serial communication

- Example:
- 9600 baud rate
- 10MHz clock frequency

In one seconds clock signal has $10^7$ cycles
We have 2400 symbol every second
Baud tick should clock every $10^7/9600=1041.66$ cycle

Clock divided by 1041

# Basics of serial communication

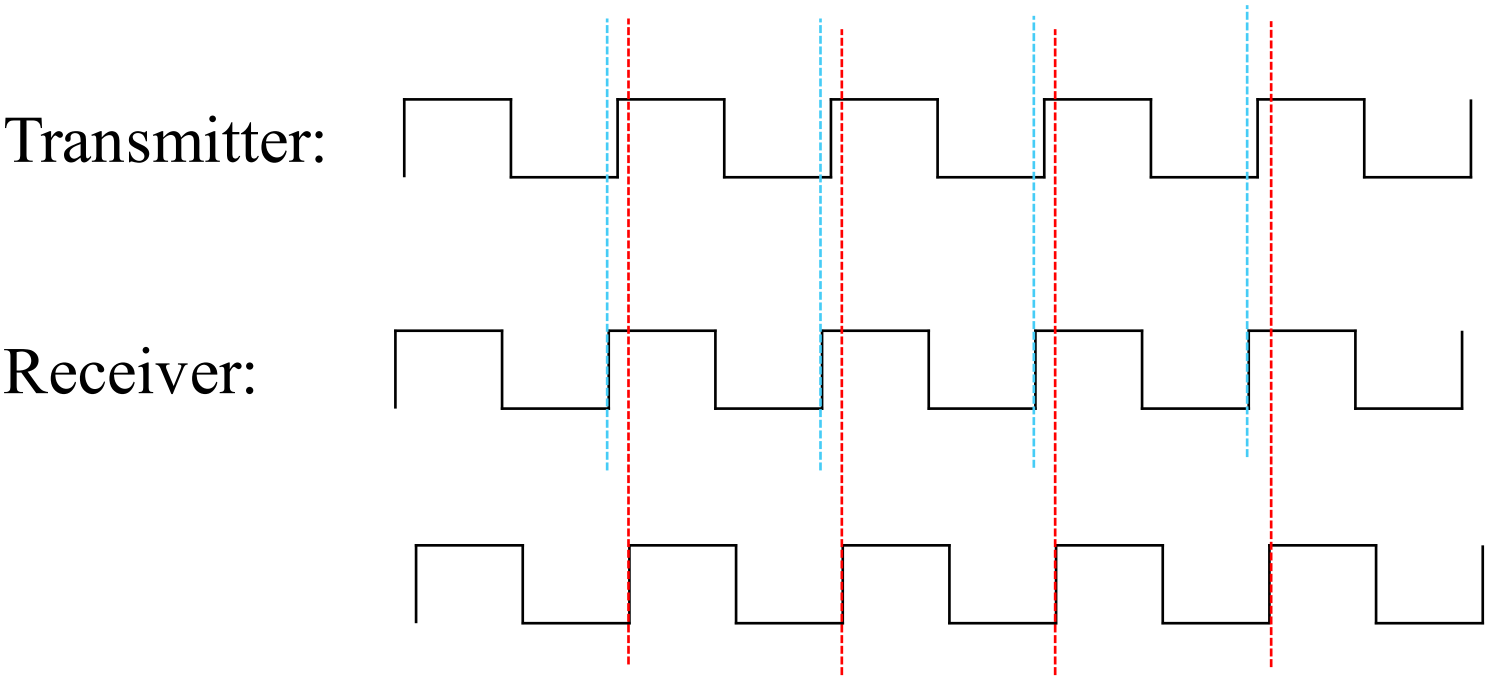
➢ Example:

- 9600 baud rate
- 10MHz clock frequency

New baud rate = 10MHz/1041 = 9606.147

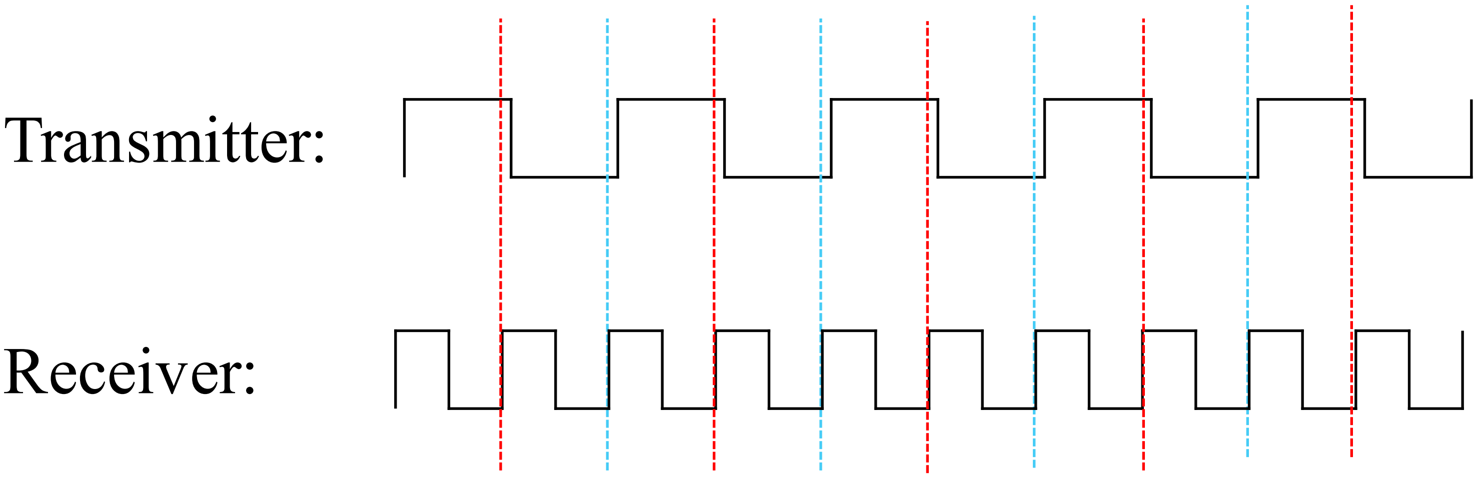
Error = (9606-9600)/9600 * 100 = 0.06% ≤ 0.3% ✓

# UART character transmission



Oversampling helps receiver get correct data

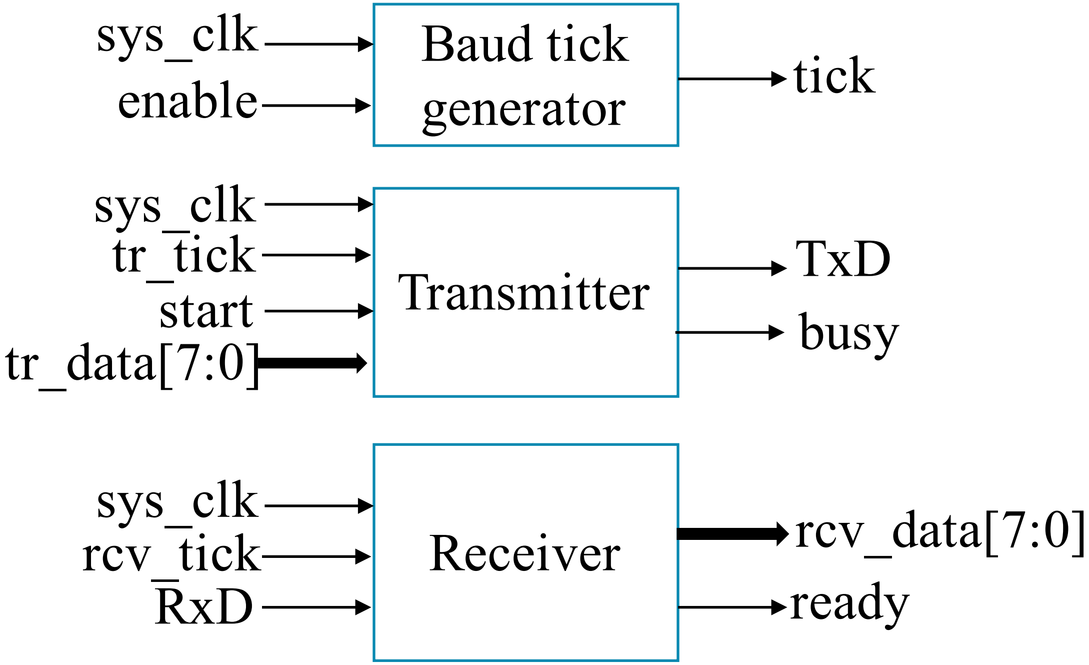
# UART character transmission



# LAB1

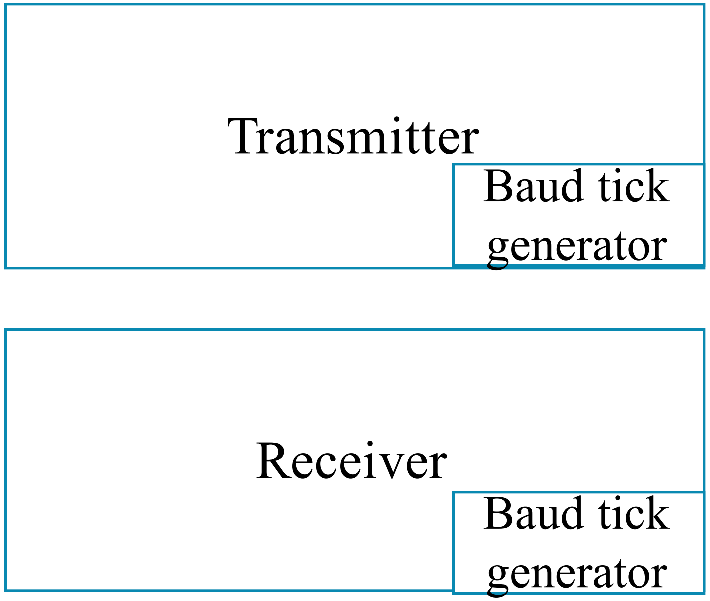
# Required modules

1. Baud tick generator
2. Transmitter
3. Receiver



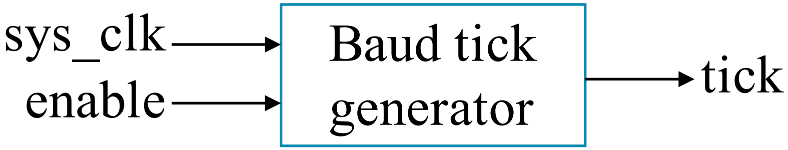
# Required modules

1. Baud tick generator
2. Transmitter
3. Receiver



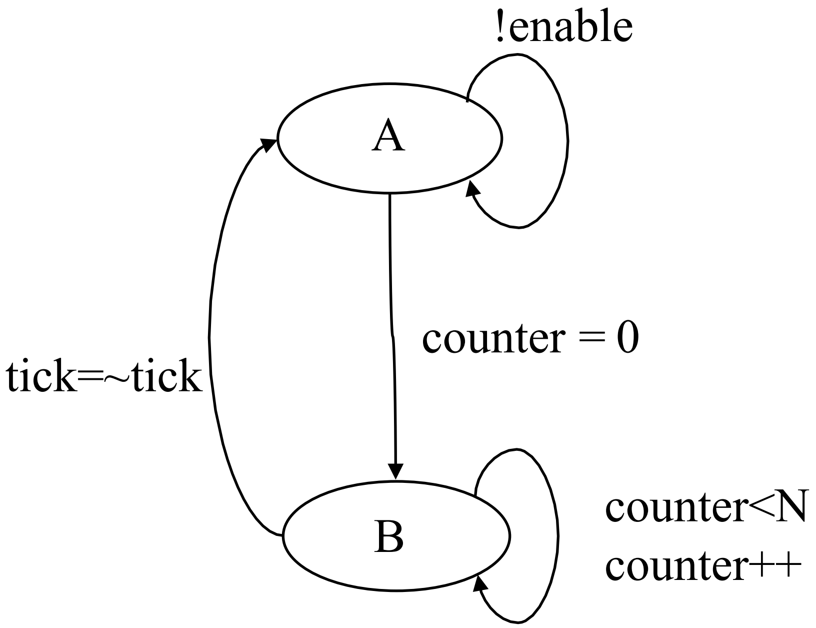
# Baud tick generator

# Baud tick generator



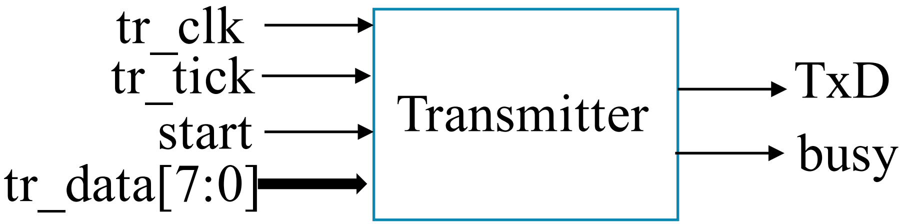
1. Calculate division factor (N)
   - clock frequency
   - baud-rate
   - oversampling
2. Use up-counter counting to N
3. Tick every N cycle

# Baud tick generator

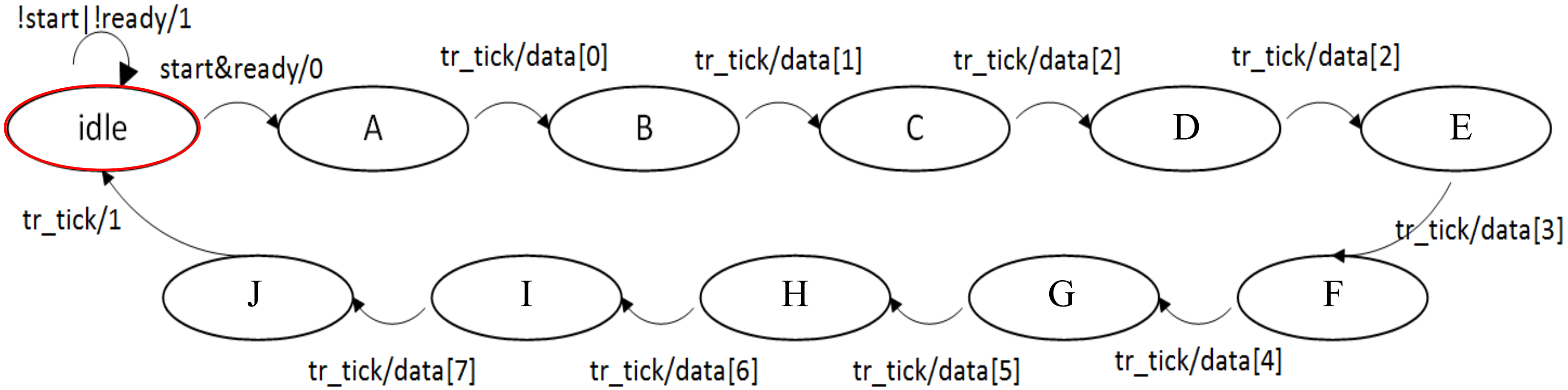


# Transmitter

# Transmitter

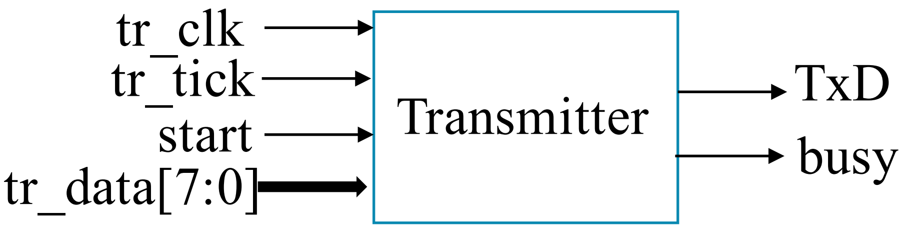


8n1 = No parity, 1 stop bit

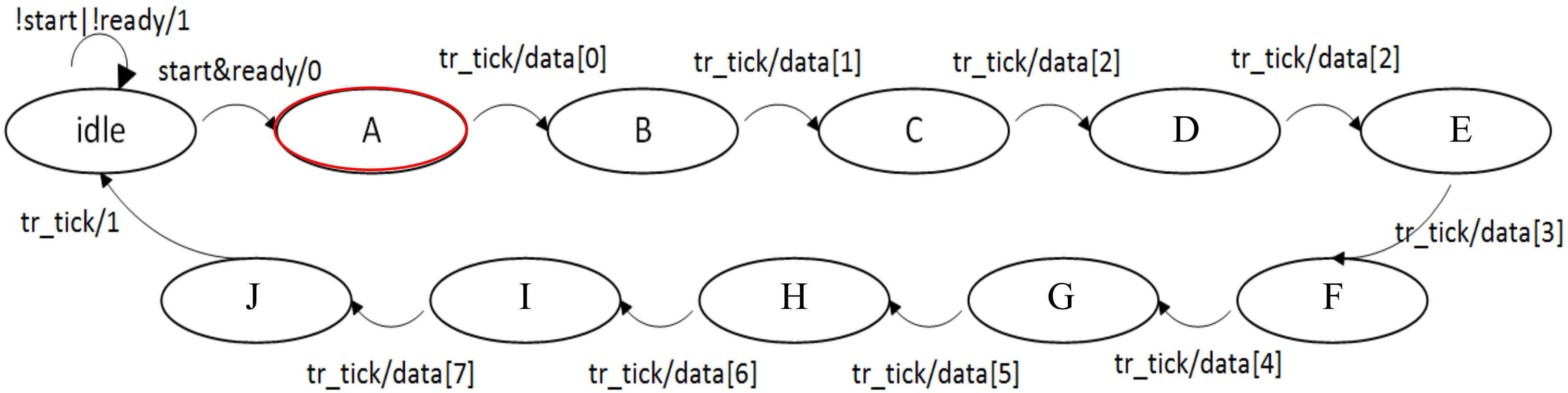


Send: idle bit = 1

# Transmitter

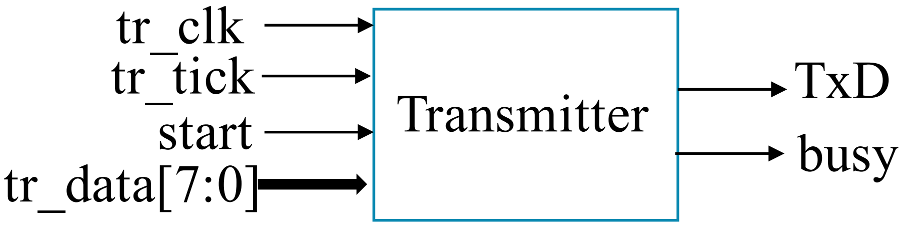


8n1 = No parity, 1 stop bit

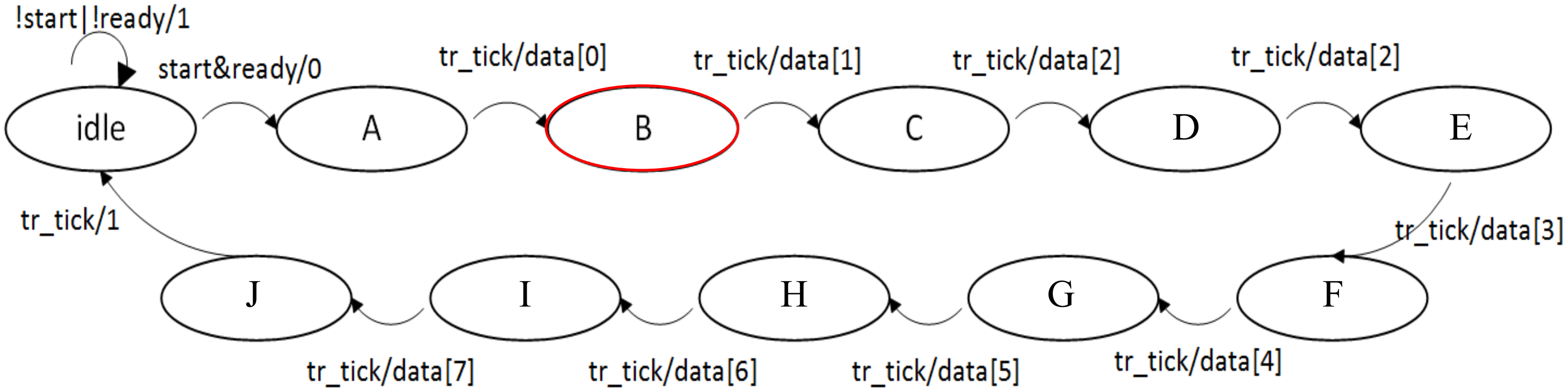


Send: start bit = 0

# Transmitter

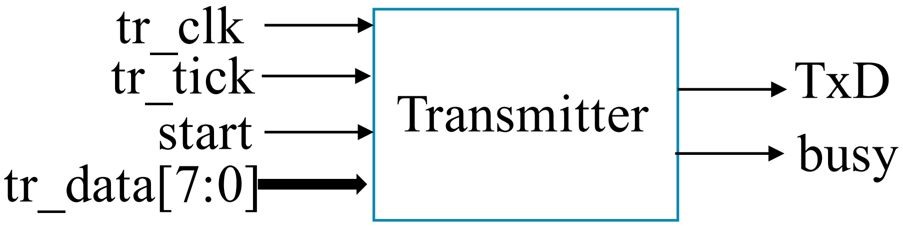


8n1 = No parity, 1 stop bit

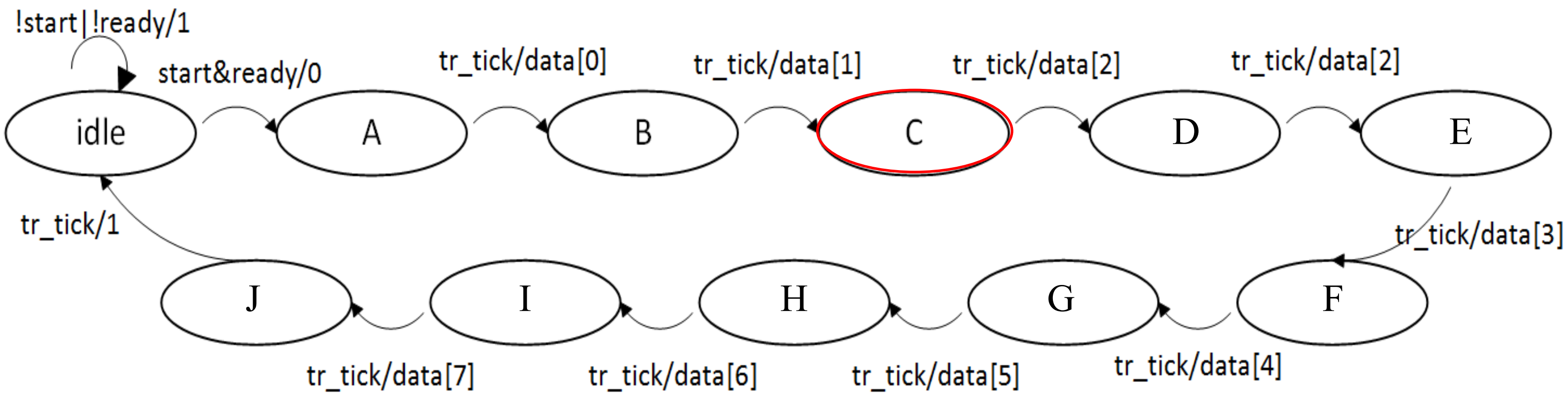


Send: data[0]

# Transmitter

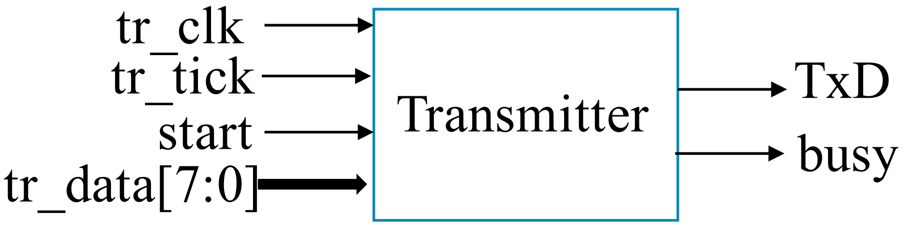


8n1 = No parity, 1 stop bit

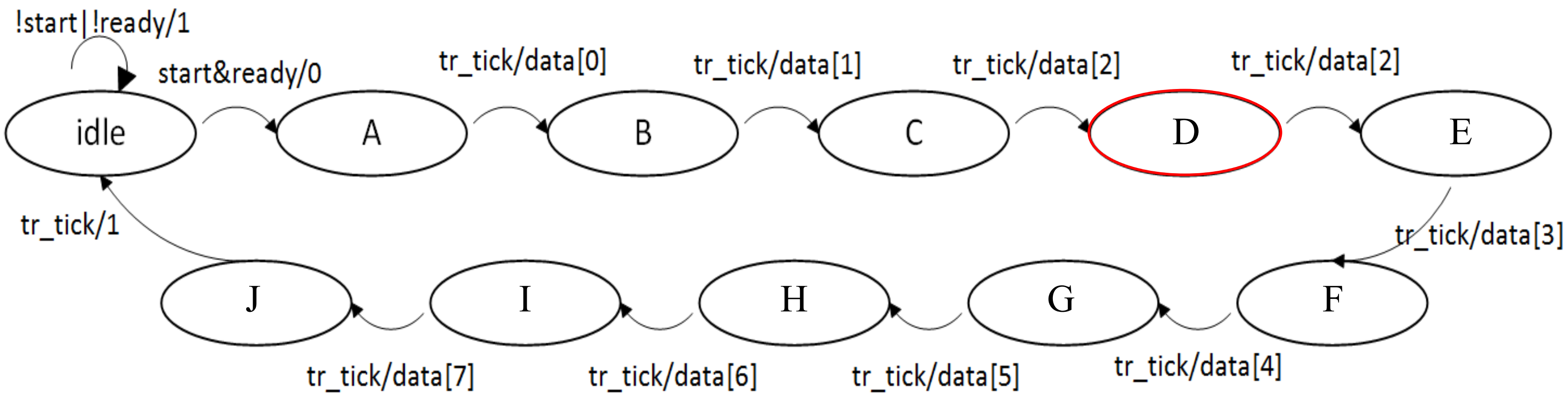


Send: data[1]

# Transmitter

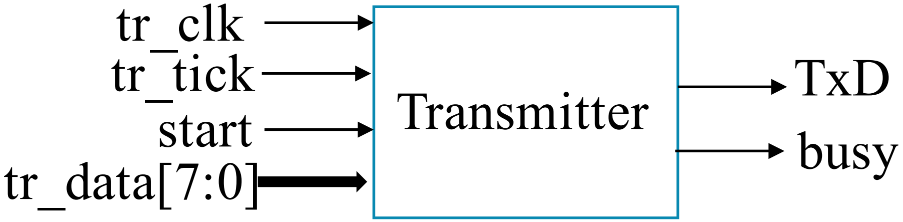


8n1 = No parity, 1 stop bit

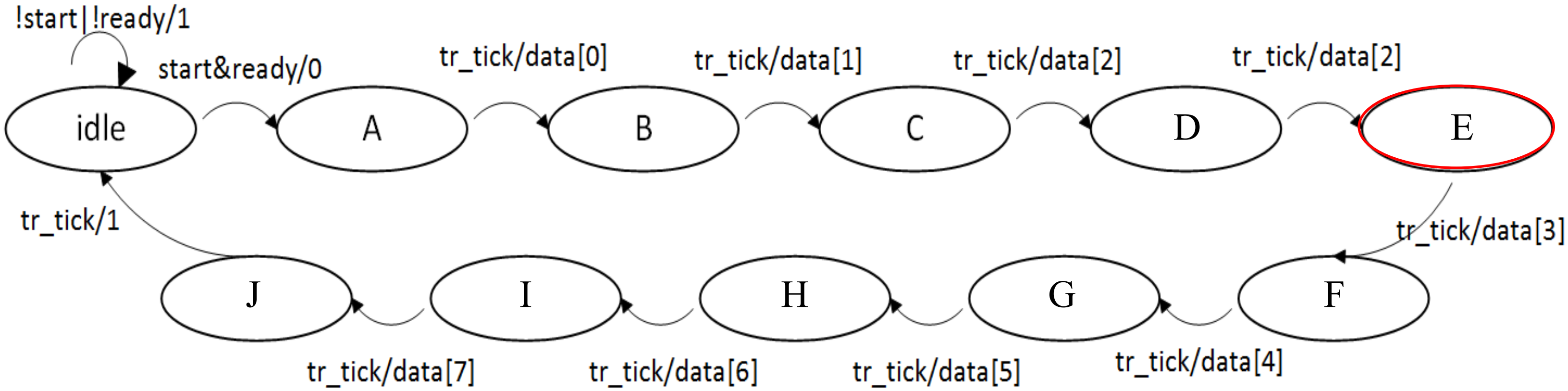


Send: data[2]

# Transmitter

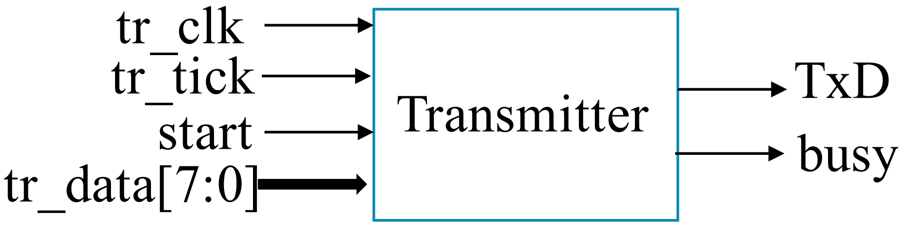


8n1 = No parity, 1 stop bit

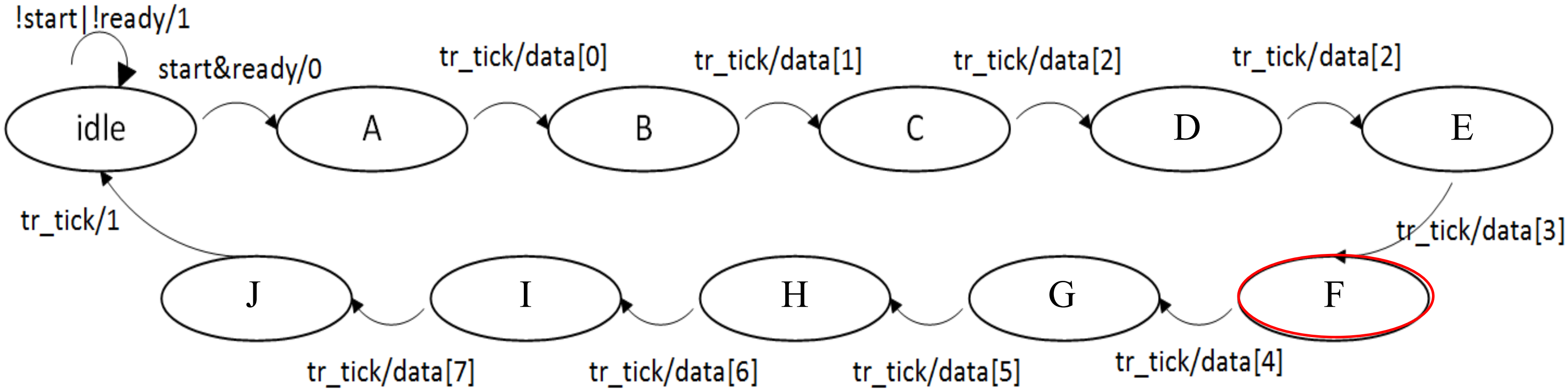


Send: data[3]

# Transmitter

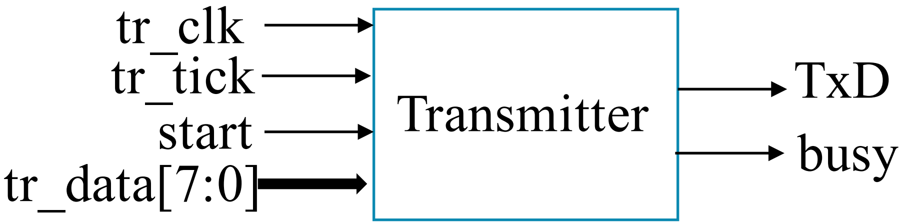


8n1 = No parity, 1 stop bit

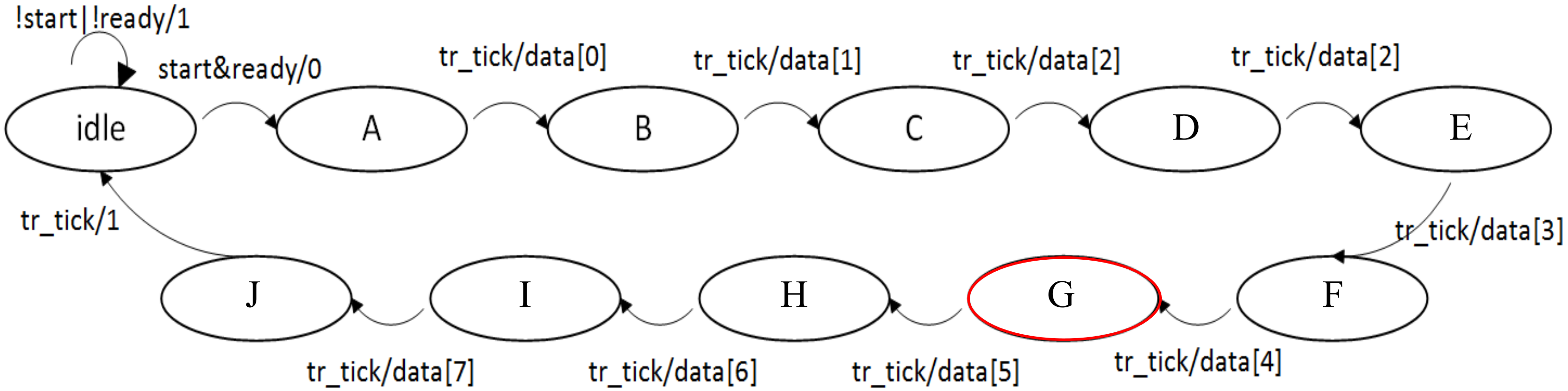


Send: data[4]

# Transmitter



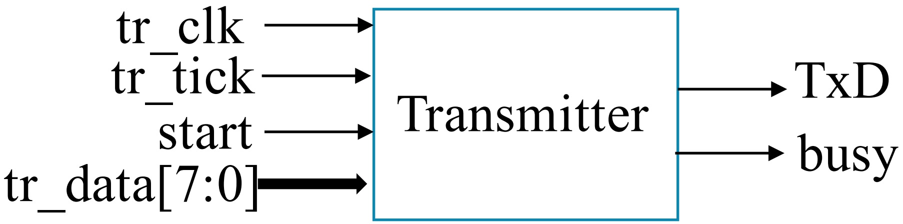
8n1 = No parity, 1 stop bit



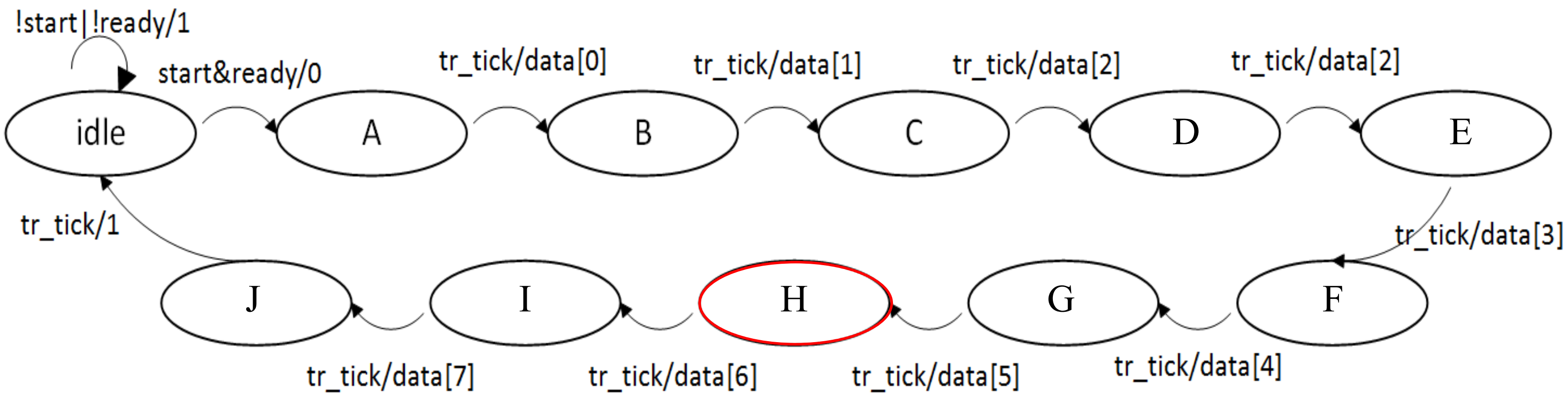
Send: data[5]

DVDES

# Transmitter

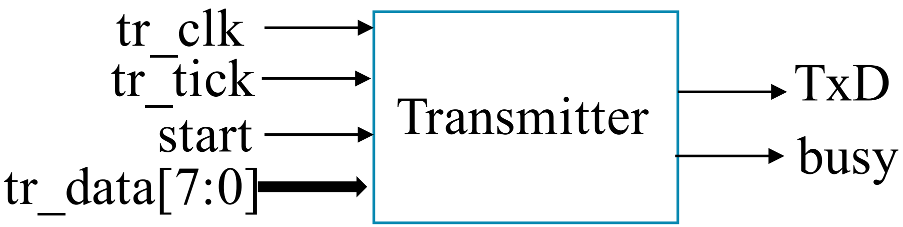


8n1 = No parity, 1 stop bit



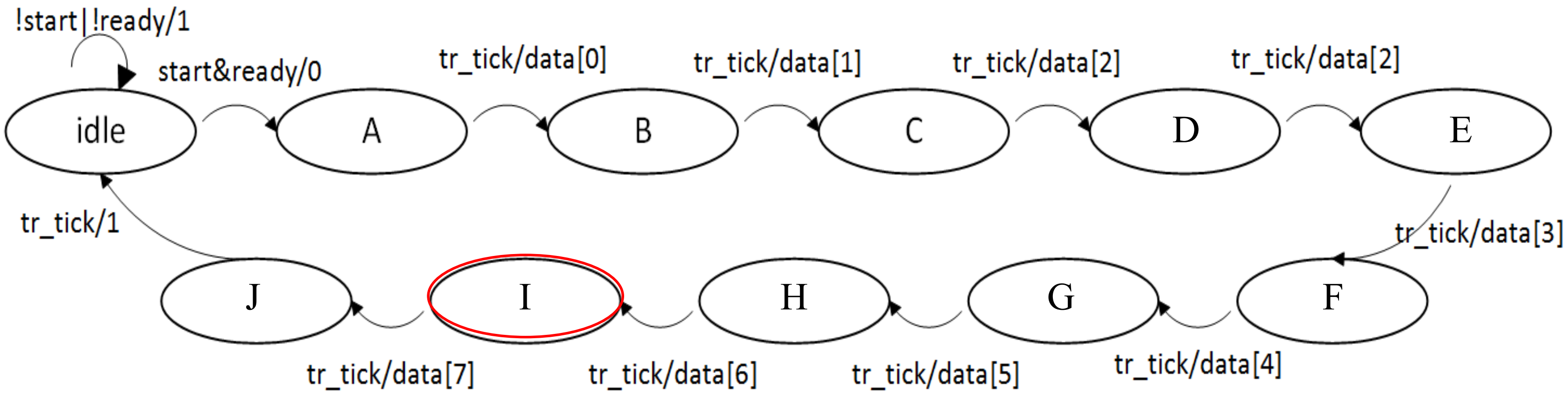
Send: data[6]

# Transmitter



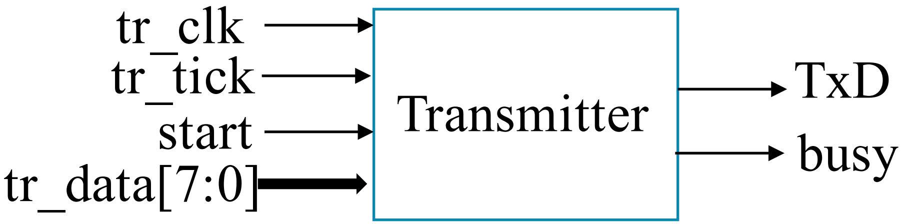
tr_clk
tr_tick
start
tr_data[7:0]
Transmitter
TxD
busy

8n1 = No parity, 1 stop bit

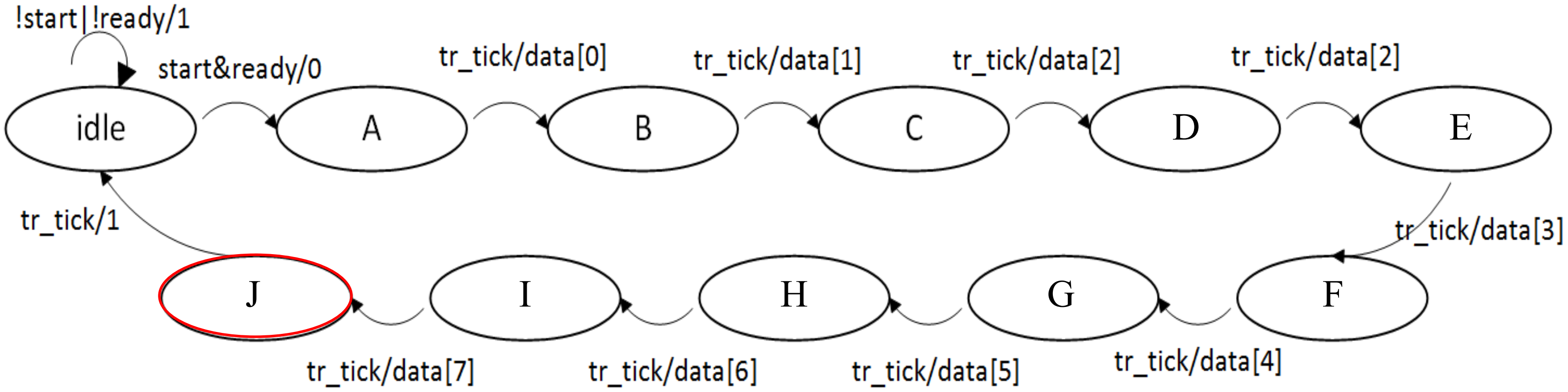


Send: data[7]

# Transmitter



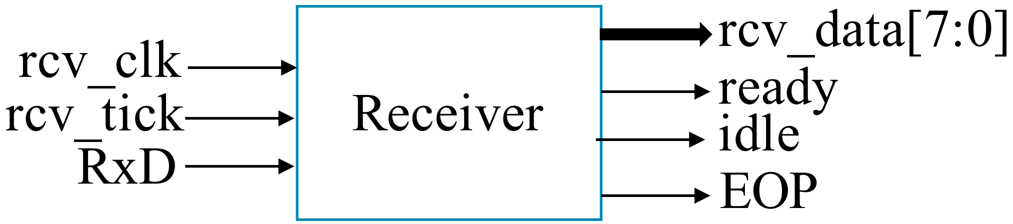
8n1 = No parity, 1 stop bit



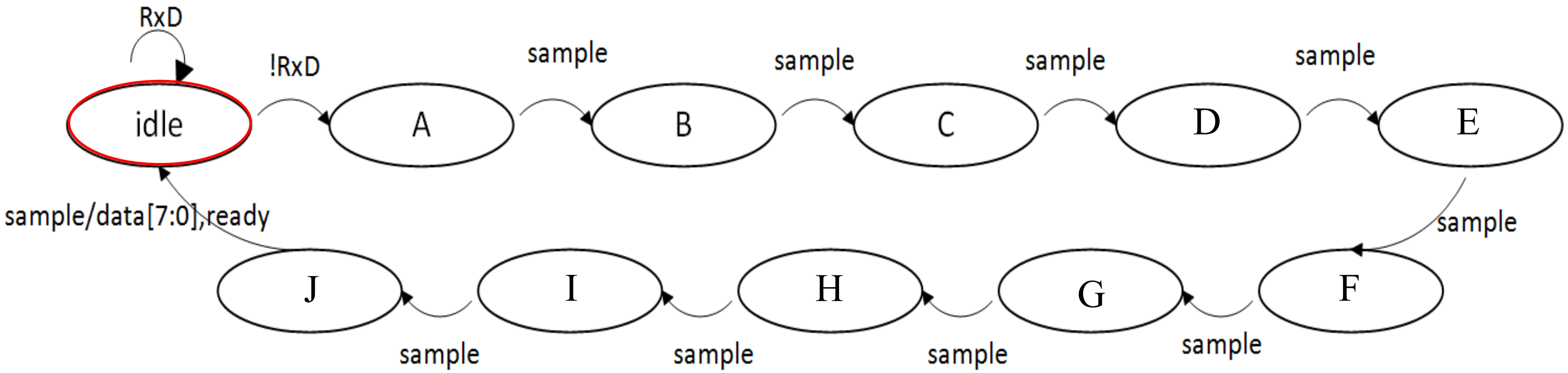
Send: stop bit = 1

# Receiver

# Receiver

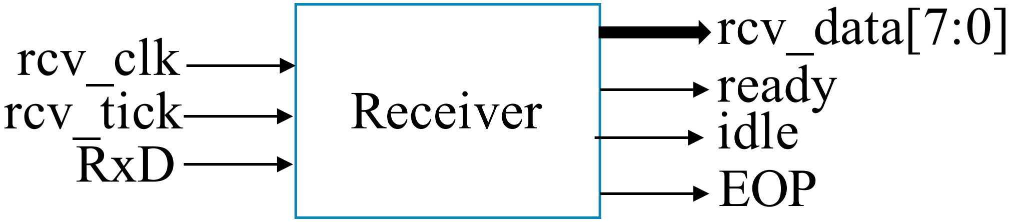


## 8n1 = No parity, 1 stop bit

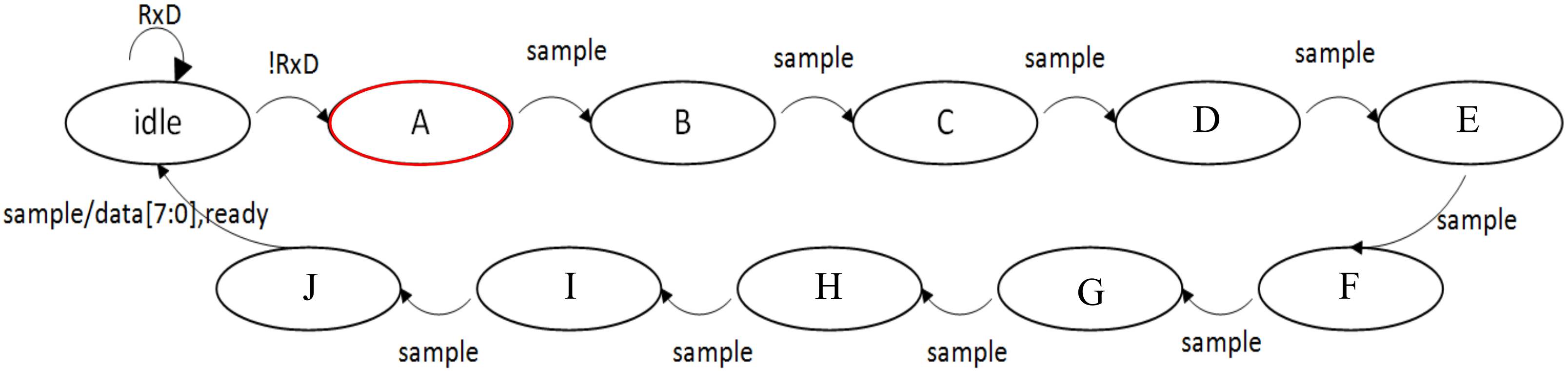


## Receive: idle bit = 1

# Receiver

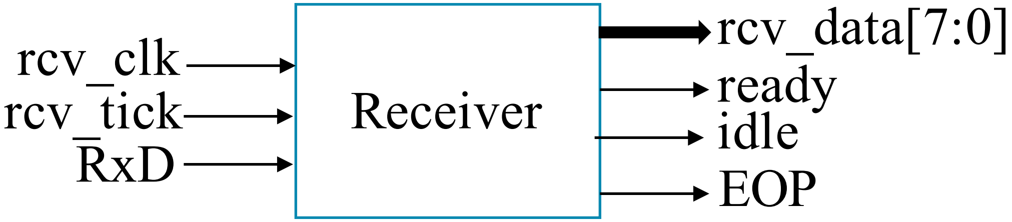


8n1 = No parity, 1 stop bit

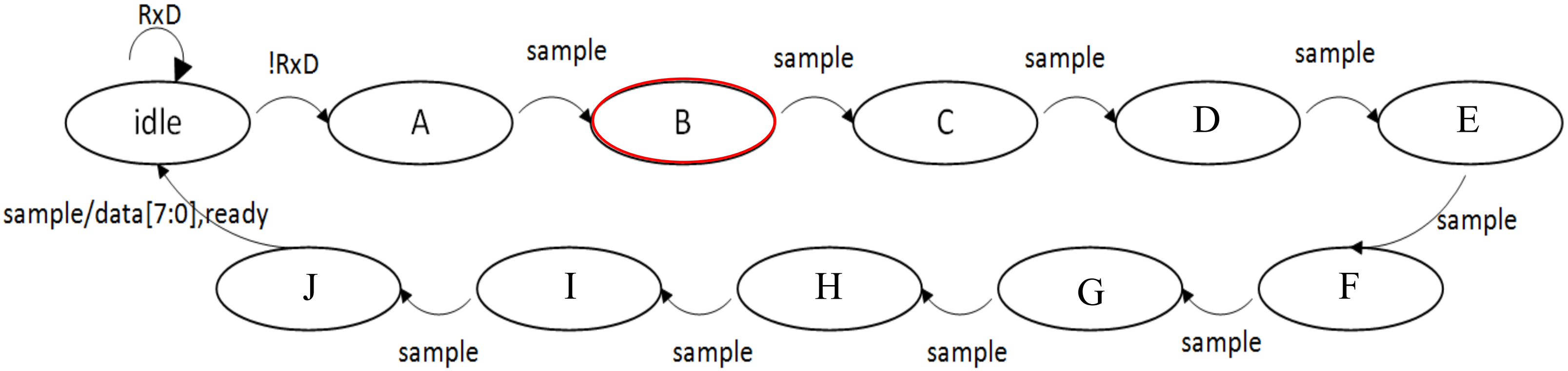


Receive: start bit = 0
Wait for sample signal

# Receiver

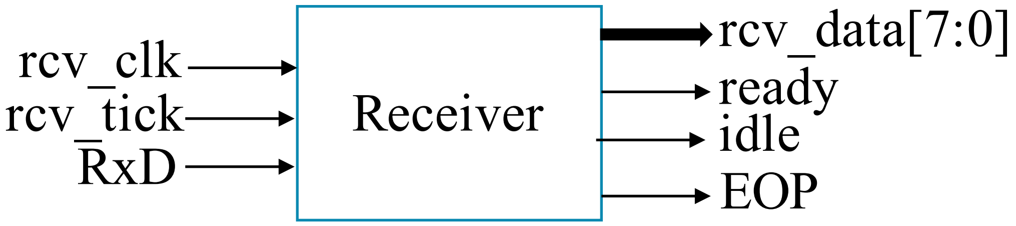


8n1 = No parity, 1 stop bit



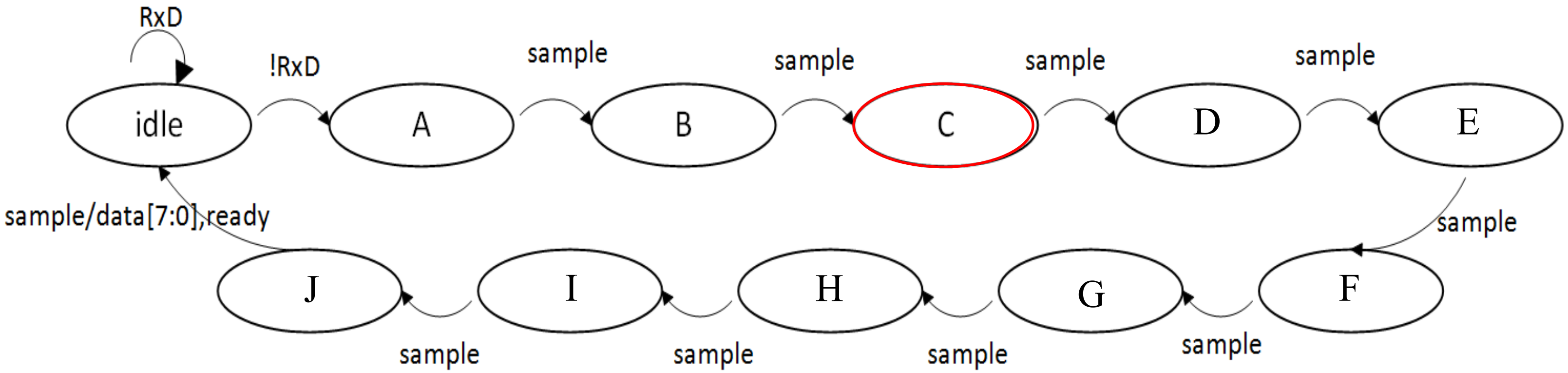
Receive : data[0]

# Receiver



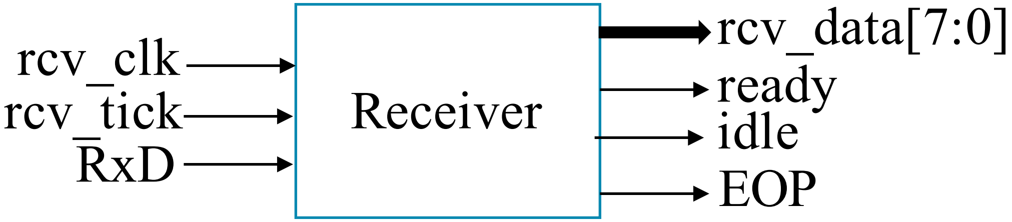
rcv_clk
rcv_tick
RxD
Receiver
rcv_data[7:0]
ready
idle
EOP

8n1 = No parity, 1 stop bit

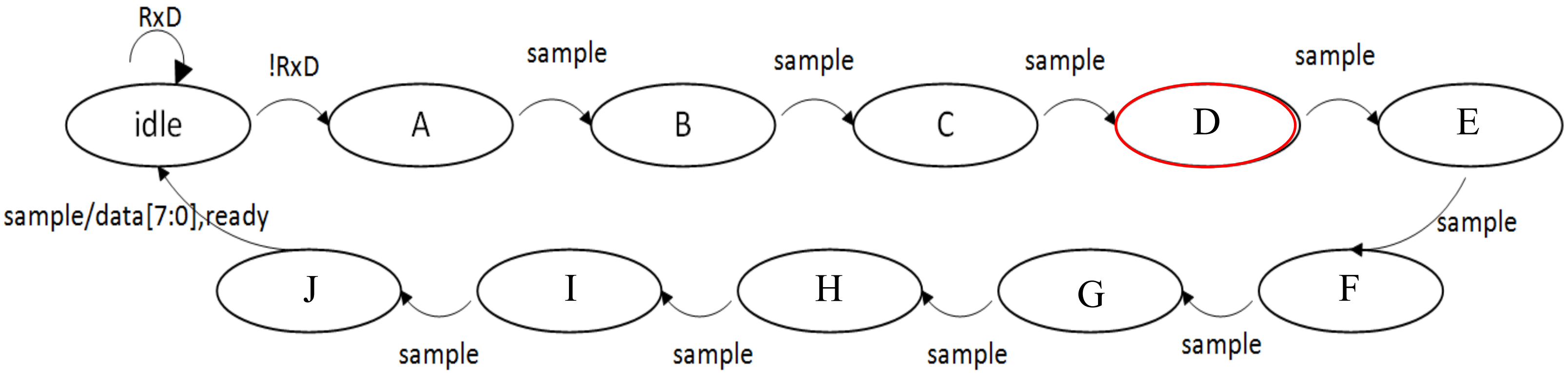


Receive : data[1]

# Receiver

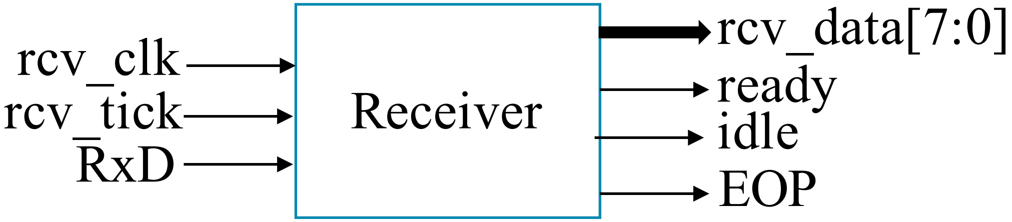


8n1 = No parity, 1 stop bit

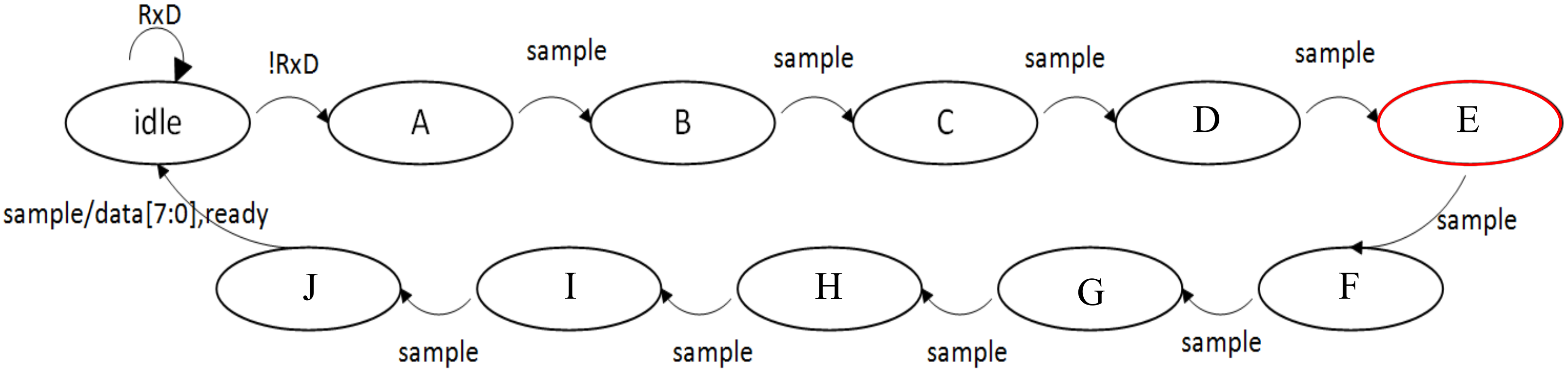


Receive : data[2]

# Receiver

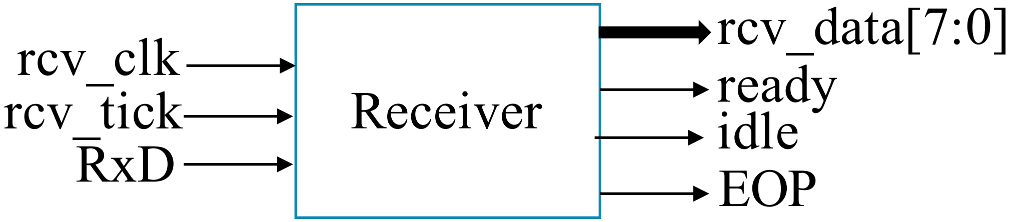


8n1 = No parity, 1 stop bit

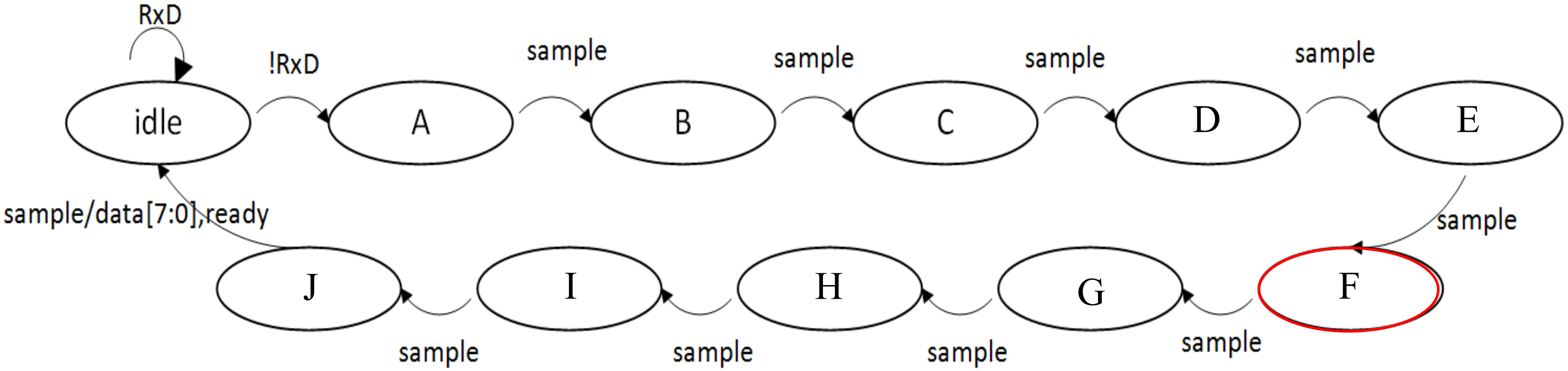


Receive : data[3]

# Receiver

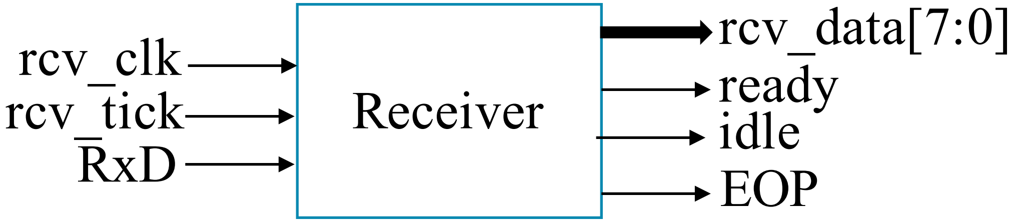


8n1 = No parity, 1 stop bit

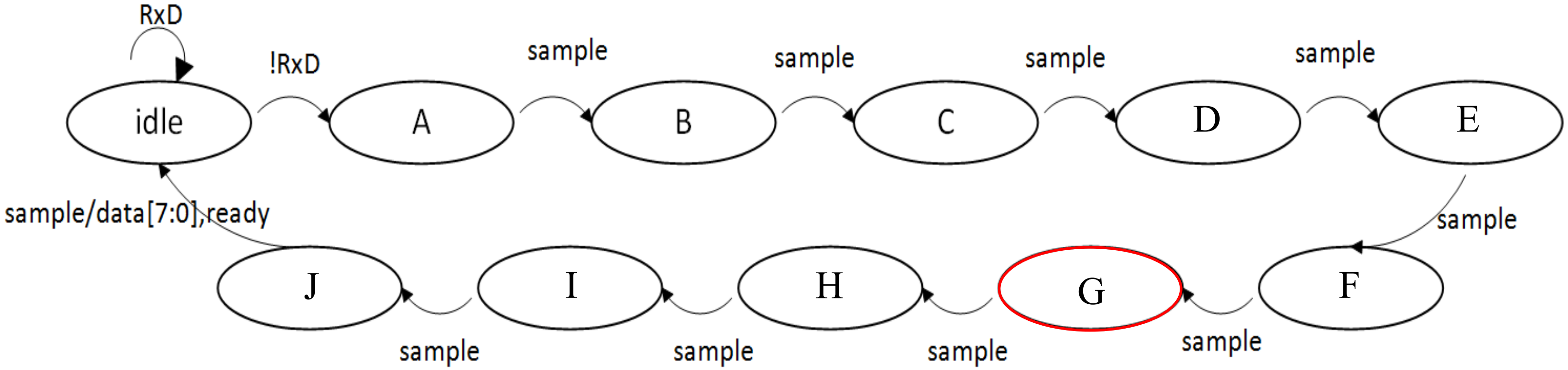


Receive : data[4]

# Receiver

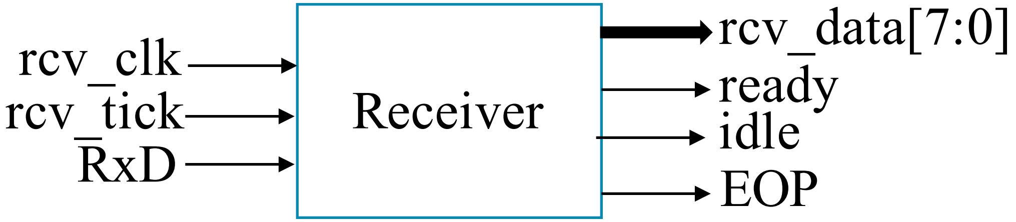


8n1 = No parity, 1 stop bit

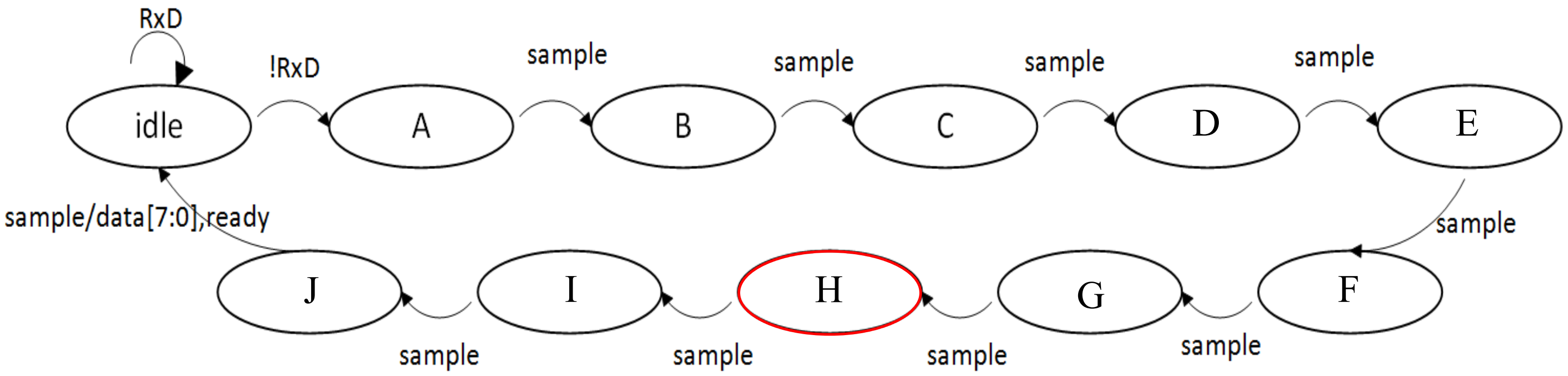


Receive : data[5]

# Receiver

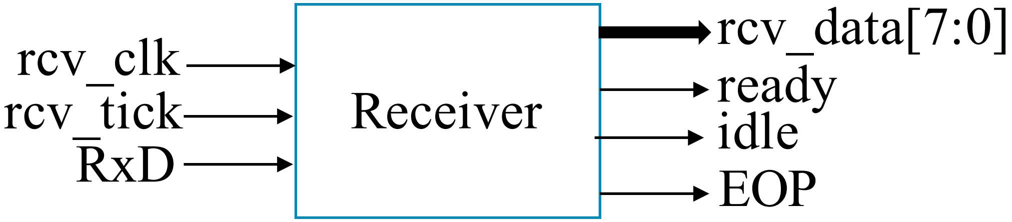


## 8n1 = No parity, 1 stop bit

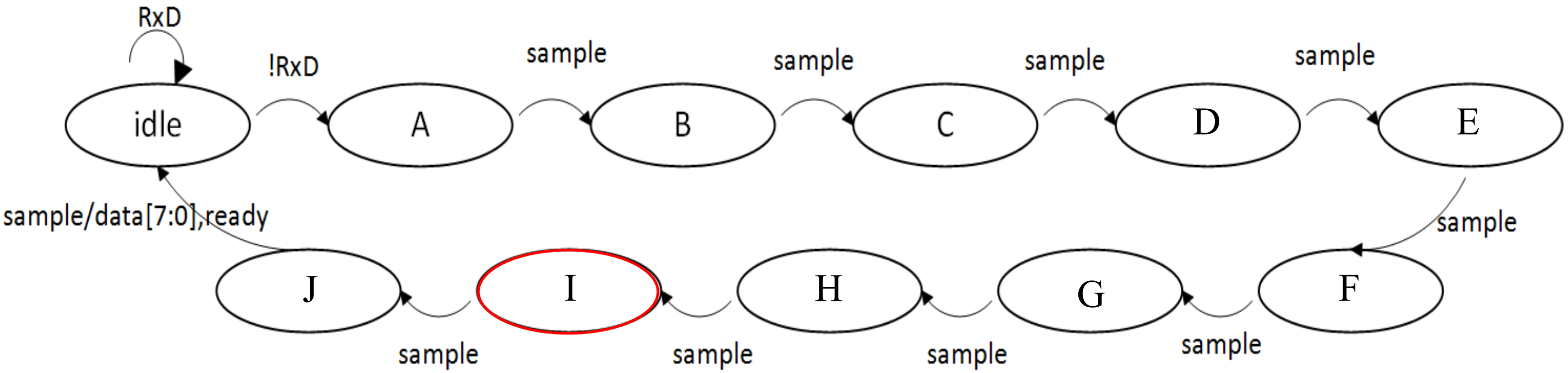


## Receive : data[6]

# Receiver

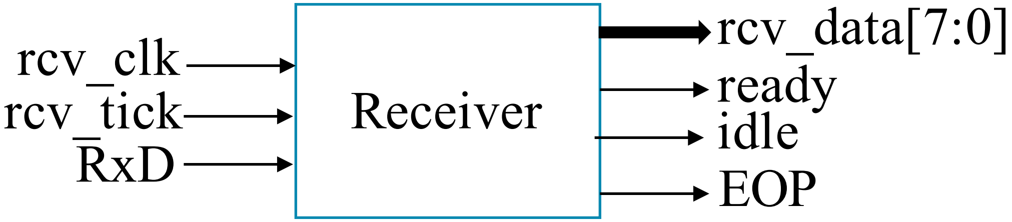


8n1 = No parity, 1 stop bit

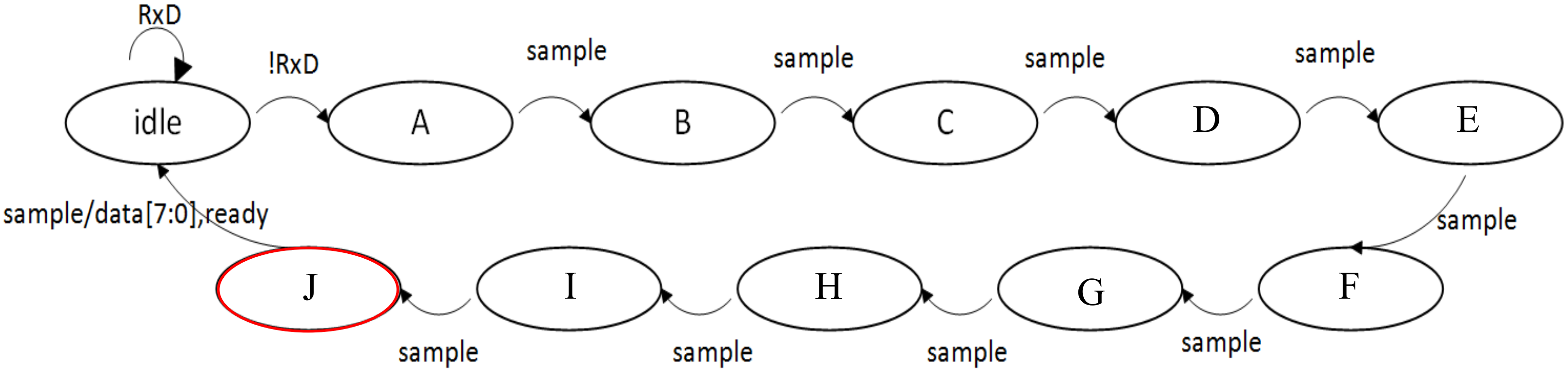


Receive : data[7]

# Receiver



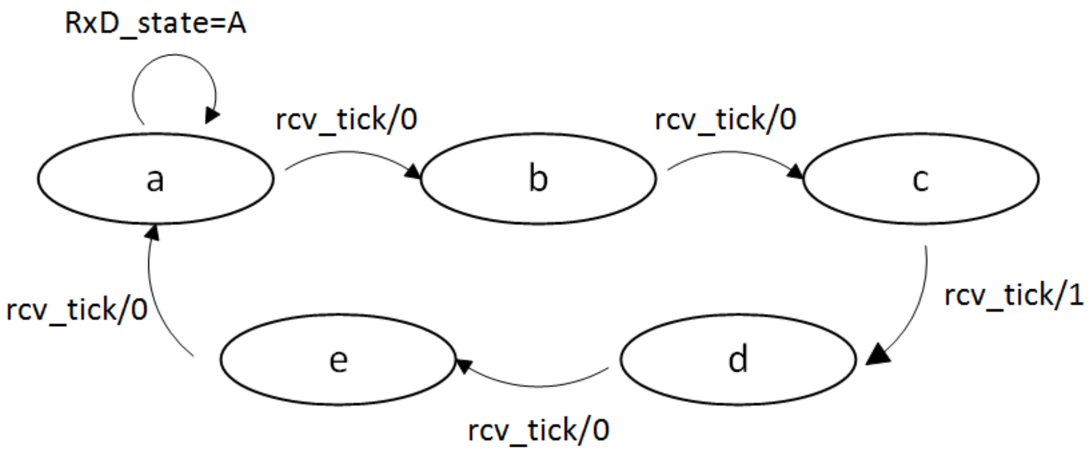
8n1 = No parity, 1 stop bit



Receive : stop bit = 1

# Receiver

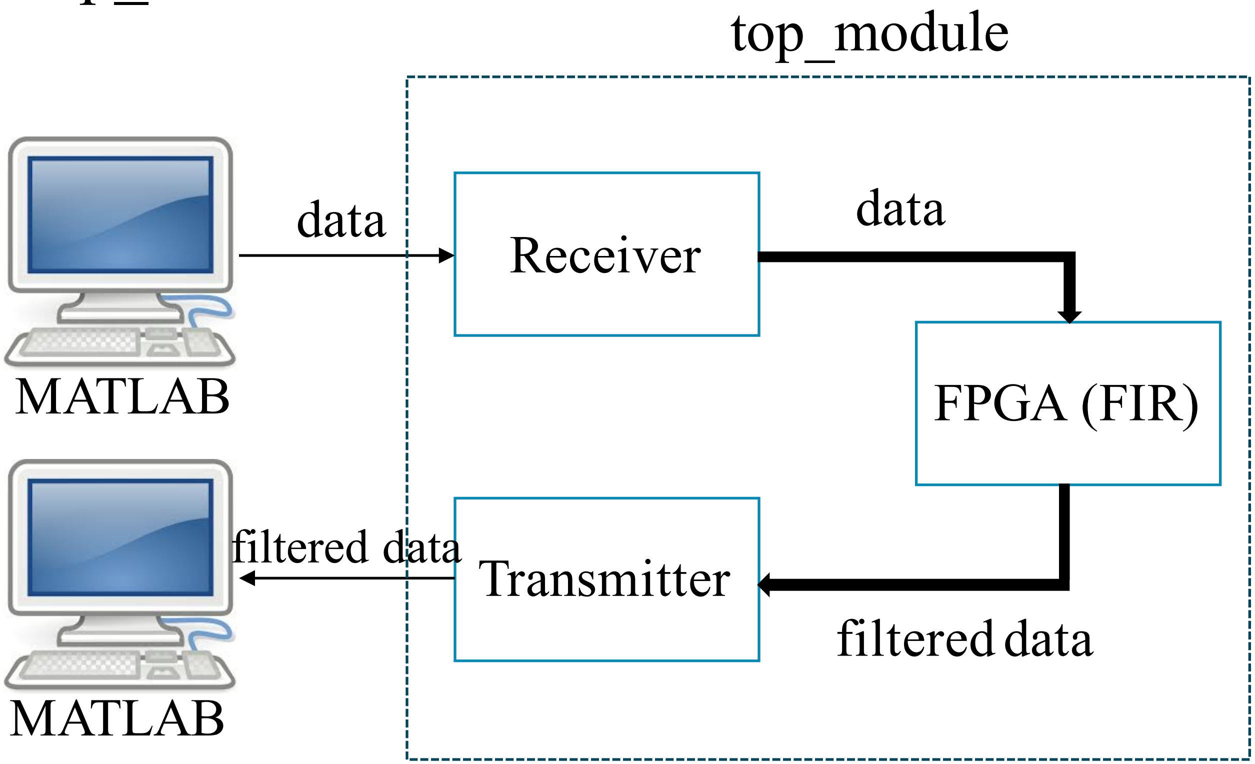
sample signal state machine



DVDES

# Lab1

Write top_module code



# Thanks for your attention