CS254 Project Presentation
UART Design

Team Number 6 : Digital Logic Design Lab
KATURI SAI KIRAN
130050051,130050051@iitb.ac.in
LINGALA RAHUL
130050056,130050056@iitb.ac.in
ADDANKI RAVICHANDRA
130050061,130050061@iitb.ac.in
VEGULLA KRANTHI
130050062,130050062@iitb.ac.in
THATIPAMULA AKHIL
130050068,130050068@iitb.ac.in

May 1, 2015
1. What is UART

2. Modules inside UART design

3. Algorithmic State Machines (ASM)
   - ASM specifying receiver design
   - ASM specifying transmitter design

4. State Diagrams
   - State Diagram for receiver
   - State Diagram for transmitter
UART Definition

- Universal Asynchronous receiver/transmitter.
- **Universal** : Any device can run uart protocol.
- **Asynchronous** : The sequential components on the receiver and sender can be asynchronous.
- **transmitter/receiver** : used for communication purposes.
**Terminology used**

**Baud Rate**

A unit for symbol rate or modulation rate in *symbols* per second.
Terminology used

Baud Rate
A unit for symbol rate or modulation rate in symbols per second

Bit Rate
It is the number of bits that are conveyed or processed per unit of time.
### Terminology used

**Baud Rate**

A unit for symbol rate or modulation rate in *symbols* per second.

**Bit Rate**

It is the number of *bits* that are conveyed or processed per unit of time.

**Sampling Rate**

The number of times we sample during the transmission of a particular bit sent by the transmitter.
Terminology used

**Baud Rate**
A unit for symbol rate or modulation rate in *symbols* per second

**Bit Rate**
It is the number of *bits* that are conveyed or processed per unit of time.

**Sampling Rate**
The number of times we sample during the transmission of a particular bit sent by the transmitter.

**Example**
In our project the baud rate is 19200 Hz while the sampling rate is 16*19200 Hz
Overview of modules

![Diagram of modules]

Team Number 6: Digital Logic Design Lab

Members:
- KATURI SAI KIRAN 130050051, 130050051@iitb.ac.in
- LINGALA RAHUL 130050056, 130050056@iitb.ac.in
- ADDANKI RAVICHANDRA 130050061, 130050061@iitb.ac.in
- VEGULLA KRANTHI 130050062, 130050062@iitb.ac.in
- THATIPAMULA AKHIL 130050068, 130050068@iitb.ac.in
Overview of modules

Our design contains 4 modules in total. They are

- CLOCK_GEN
- RX_UART
- TX_UART
- UART_TOP
RTL schematic showing different modules

RTL schematic of our UART design
CLOCK_GEN

- Generates a virtual clock with frequency 16*19200 Hz.
CLOCK_GEN

- Generates a virtual clock with frequency $16 \times 19200$ Hz.
- This clock now behaves as a clock to RX_UART and TX_UART modules.
Generates a virtual clock with frequency 16*19200 Hz.

This clock now behaves as a clock to RX_UART and TX_UART modules.

This clock is 16 times faster than the baudrate since its frequency is 16 times baudrate.
Generates a virtual clock with frequency 16*19200 Hz.
This clock now behaves as a clock to RX_UART and TX_UART modules.
This clock is 16 times faster than the baudrate since its frequency is 16 times baudrate.
This module helps for the purpose of sampling the teraterm inputs.
Generates a virtual clock with frequency $16 \times 19200$ Hz.

This clock now behaves as a clock to RX_UART and TX_UART modules.

This clock is 16 times faster than the baudrate since its frequency is 16 times baudrate.

This module helps for the purpose of sampling the teraterm inputs.

Uses a counter in order to make a clock of lower frequency than that of 100 MHz.
Receive module that receives the serial input from tera term and converts it into a parallel output.
Receive module that receives the serial input from tera term and converts it into a parallel output.

This module uses two counters one to count for the purpose of sampling and other to count the index in the vector.
Receive module that receives the serial input from tera term and converts it into a parallel output.

This module uses two counters one to count for the purpose of sampling and other to count the index in the vector.

It uses a finite state machine with 4 states which will be explained later.
Receive module that receives the serial input from tera term and converts it into a parallel output.

This module uses two counters one to count for the purpose of sampling and other to count the index in the vector.

It uses a finite state machine with 4 states which will be explained later.

This makes sampling by running the clock 16 times faster than tera term’s clock with the help of CLOCK_GEN module.
Transmitter module that takes the parallel input and outputs it in series.
- Transmitter module that takes the parallel input and outputs it in series.
- This module uses two counters one to count for 16 clock cycles and other to count the index in the vector.
Transmitter module that takes the parallel input and outputs it in series.

This module uses two counters one to count for 16 clock cycles and other to count the index in the vector.

It uses a finite state machine with 4 states which will be explained later.
Transmitter module that takes the parallel input and outputs it in series.

This module uses two counters one to count for 16 clock cycles and other to count the index in the vector.

It uses a finite state machine with 4 states which will be explained later.

It sends each bit 16 times since its clock is 16 times faster than the tera term’s clock. It uses the CLOCK_GEN module as its clock.
This is the top module that connects the two receiver and transmitter modules.
- This is the top module that connects the two receiver and transmitter modules.
- It loop backs the two receiver and transmitter modules so that we can test it on tera term.
This is the top module that connects the two receiver and transmitter modules.

It loop backs the two receiver and transmitter modules so that we can test it on tera term.

It just takes rx as a serial input and gives tx as another serial output.
Outline

1. What is UART

2. Modules inside UART design

3. Algorithmic State Machines (ASM)
   - ASM specifying receiver design
   - ASM specifying transmitter design

4. State Diagrams
   - State Diagram for receiver
   - State Diagram for transmitter
ASM for receiver

RX_DATA_VALID=1

count=1111

inc_count

RX_DATA(index)=rx

inc_index

RX_DATA_VALID=0

idle

rx

rst

clr_count

clr_index

start

rst

inc_count

count=1110

clr_count

Receiving

rst

count=1000

inc_count

index(3)=1

count=1111

Stop
Block diagram for RX_UART

Control unit

rst
rx
check_count
check_index
inc_count
inc_index
clr_count
clr_index
set_RX_DATA_VALID
set_RX_DATA

Data path component

Rx

Clk

RX_DATA RX_DATA_VALID
Outline

1. What is UART
2. Modules inside UART design
3. Algorithmic State Machines (ASM)
   - ASM specifying receiver design
   - ASM specifying transmitter design
4. State Diagrams
   - State Diagram for receiver
   - State Diagram for transmitter
ASM for transmitter

- Idle: TX = 1
- Start: rst
- Send: inc_count
- TX = temp(index)
- inc_index
- count = 1111
- inc_count
- TX = 0
- clr_count
- clr_index
- temp = TX_DATA
- count = 1111
- rst
- count = 1111
- stop
- index(3) = 0
- rst
- inc_count
- inc_index
Block diagram for TX_UART

Control unit

Clk

rst
TX_DATA_VALID
check_count
check_index
inc_count
inc_index
clr_count
clr_index
set_TX
set_temp

Data path component

TX_DATA
TX_DATA_VALID

Block diagram for TX_UART
Outline

1. What is UART

2. Modules inside UART design

3. Algorithmic State Machines (ASM)
   - ASM specifying receiver design
   - ASM specifying transmitter design

4. State Diagrams
   - State Diagram for receiver
   - State Diagram for transmitter
State Diagram for receiver

- **IDLE**
  - RX_DATA_VALID = 0
  - RST = 1

- **START**
  - RX = 0
  - 15 clock cycles

- **RECEIVING**
  - RX_DATA_VALID = 1
  - RST = 1
  - At 16th clock cycle
  - Receive 1 byte of data for 16*8 clock cycles

- **STOP**
  - RX = 1 for 16 clock cycles
  - At the end of sending 1 byte of data
State Diagram for receiver

- Consists of 4 states in total: IDLE, START, RECEIVING and STOP.
State Diagram for receiver

- Consists of 4 states in total: IDLE, START, RECEIVING and STOP.
- When it is receiving idle bit from tera term it stays in idle state.
- Consists of 4 states in total: IDLE, START, RECEIVING and STOP.
- When it is receiving idle bit from tera term it stays in idle state.
- When ever it receives start bit it comes to START state and listens it for 16 cycles.
State Diagram for receiver

- Consists of 4 states in total: IDLE, START, RECEIVING and STOP.
- When it is receiving idle bit from tera term it stays in idle state.
- When ever it receives start bit it comes to START state and listens it for 16 cycles.
- Then it comes to RECEIVING state to receive data bits by sampling.
State Diagram for receiver

- Consists of 4 states in total: IDLE, START, RECEIVING and STOP.
- When it is receiving idle bit from tera term it stays in idle state.
- When ever it receives start bit it comes to START state and listens it for 16 cycles.
- Then it comes to RECEIVING state to receive data bits by sampling.
- After receiving all data bits it goes to STOP state to receive the stop bit and after it goes back to idle state.
Outline

1. What is UART

2. Modules inside UART design

3. Algorithmic State Machines (ASM)
   - ASM specifying receiver design
   - ASM specifying transmitter design

4. State Diagrams
   - State Diagram for receiver
   - State Diagram for transmitter
State Diagram for transmitter

- **Idle**: TX=1
  - RST=1
  - TX_DATA_VALID=1

- **Start**: TX=0
  - At 16th clock cycle

- **Send**: RX=1 for 16 clock cycles
  - Receive 1 byte data for 16+8 clock cycles

- **Stop**: TX=1
  - At the end of sending 1 byte of data
  - RX=1 for 16 clock cycles

- **RST=1**

15 clock cycles
State Diagram for transmitter

- Consists of 4 states in total: IDLE, START, SEND and STOP.
State Diagram for transmitter

- Consists of 4 states in total: IDLE, START, SEND and STOP.
- When it is not receiving any valid parallel vector it stays in idle state and sends TX as idle bit.
State Diagram for transmitter

- Consists of 4 states in total: IDLE, START, SEND and STOP.
- When it is not receiving any valid parallel vector it stays in idle state and sends TX as idle bit.
- When it receives a parallel vector it goes to START state to start transmitting and transmits start bit in this state.
State Diagram for transmitter

- Consists of 4 states in total: IDLE, START, SEND and STOP.
- When it is not receiving any valid parallel vector it stays in idle state and sends TX as idle bit.
- When it receives a parallel vector it goes to START state to start transmitting and transmits start bit in this state.
- After this it goes to SEND state to send the data bits.
State Diagram for transmitter

- Consists of 4 states in total: IDLE, START, SEND and STOP.
- When it is not receiving any valid parallel vector it stays in idle state and sends TX as idle bit.
- When it receives a parallel vector it goes to START state to start transmitting and transmits start bit in this state.
- After this it goes to SEND state to send the data bits.
- Then it goes to STOP state to send the stop bit and after this it goes to its idle state.
Conclusions

- Sampling is an effective way to overcome asynchrony.
- For a successful transmission of data, receiver frequency > transmitter frequency.
- FSMs and Counters which are themselves a simple device can be used to build many complex outcomes like UART which handles many cases like sampling etc...
Wave form simulation for the top module
The code is a result of the contribution of all the team members.
The project report and presentation are made by Katuri Sai Kiran(130050051) and Ravichandra(130050061)
The state transition diagrams are made by Rahul Lingala(130050056)
The Block diagrams are due to Vegulla Kranthi(130050062)
The ASM’s are made by Akhil thatipamula(130050068)
References

(1) http://en.wikipedia.org/wiki/8250_UART
(2) http://en.wikipedia.org/wiki/Universal_asynchronous_receiver/transmitter
(3) http://en.wikipedia.org/wiki/16550_UART
(4) http://en.wikipedia.org/wiki/Tera_Term
(5) http://download.cnet.com/Tera-Term/3000-20432_4-75766675.html