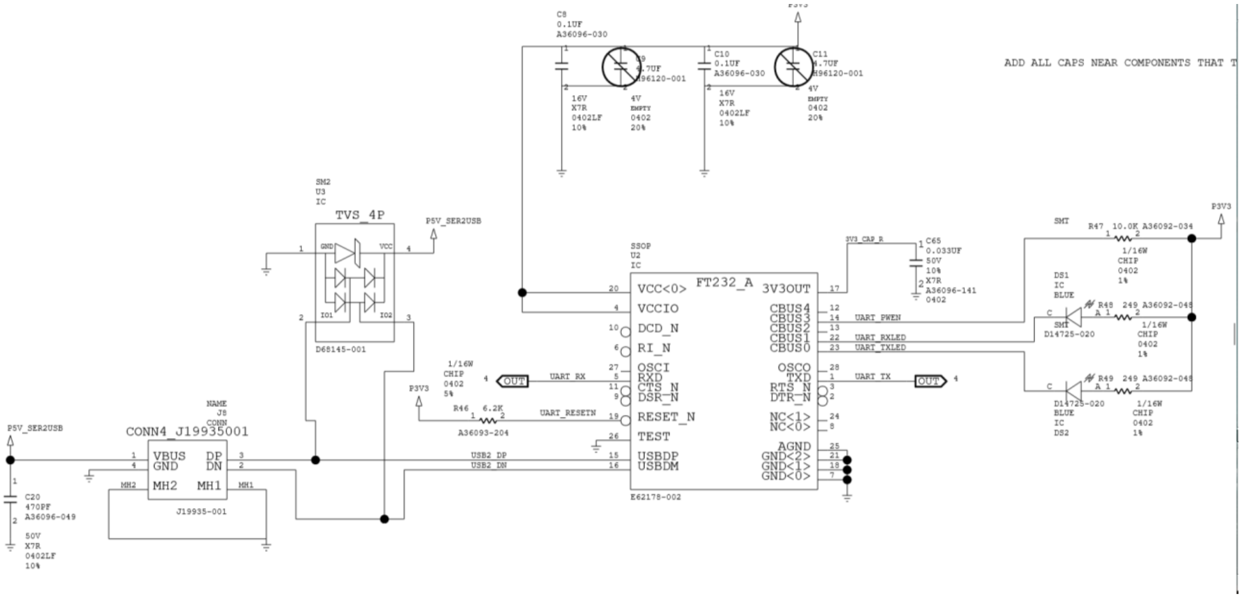
Abram Fouts

Homework 4

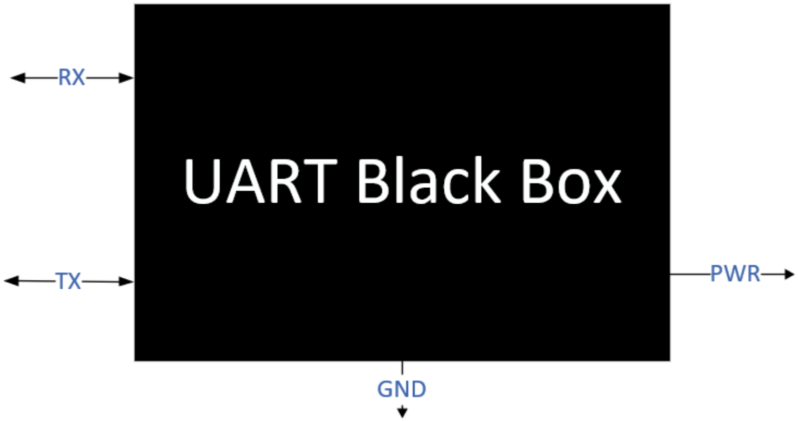
ECE 351

June 2, 2020

Block Diagram

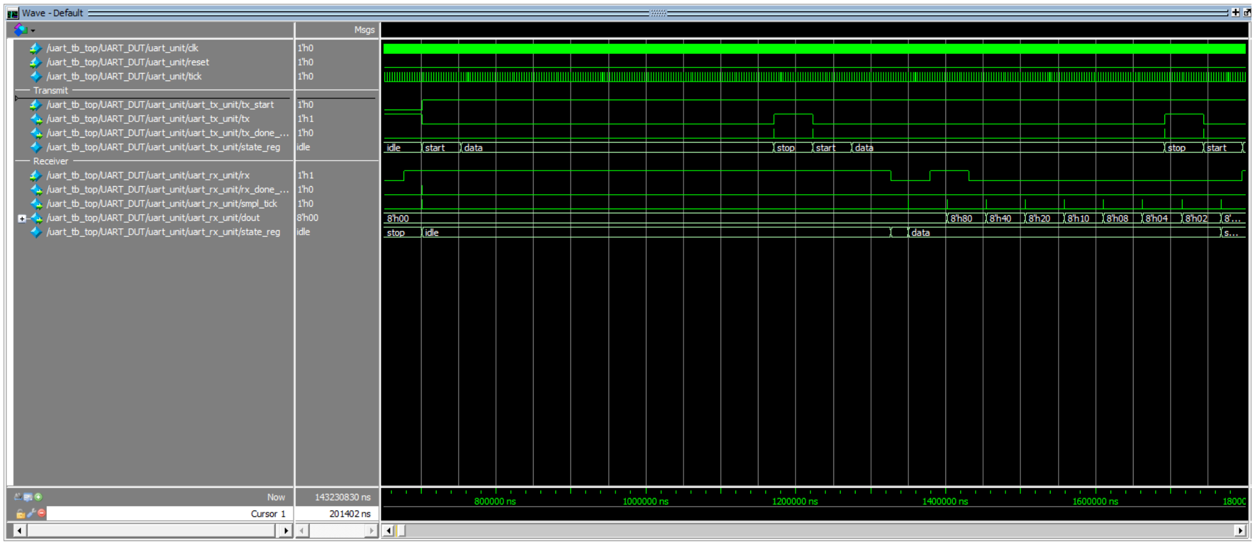


Here is a FT232 chip with a USB type B connector input on an FPGA development board I designed at intel. This will be sufficient to create a simple black box block diagram for UART. We have our transmit signal line that is bidirectional from source and destinations, along with a bidirectional receive line both named TX and RX respectively. There is a mutual ground between the two. There needs to be proper power being supplied to the 232 chips (UART chip). In this case, there is a 5V USB in and a 3.3V from an LDO VR.

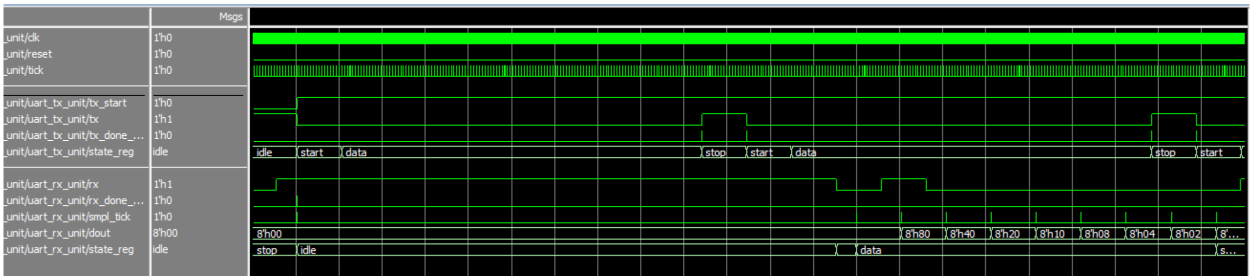


This is the black box diagram. It is very simple because UART is for serialized data meaning data only travels on one line. In UART's case it travels on two lines one for receiving and one for sending.

Waveform



```
Wave - Default
                                                              Msgs
/uart_tb_top/UART_DUT/uart_unit/clk                            1'h0
/uart_tb_top/UART_DUT/uart_unit/reset                          1'h0
/uart_tb_top/UART_DUT/uart_unit/tick                           1'h0
Transmit
/uart_tb_top/UART_DUT/uart_unit/uart_tx_unit/tx_start          1'h0
/uart_tb_top/UART_DUT/uart_unit/uart_tx_unit/tx                1'h1
/uart_tb_top/UART_DUT/uart_unit/uart_tx_unit/tx_done_...       1'h0
/uart_tb_top/UART_DUT/uart_unit/uart_tx_unit/state_reg         idle
Receiver
/uart_tb_top/UART_DUT/uart_unit/uart_rx_unit/rx                1'h1
/uart_tb_top/UART_DUT/uart_unit/uart_rx_unit/rx_done_...       1'h0
/uart_tb_top/UART_DUT/uart_unit/uart_rx_unit/smpl_tick         1'h0
/uart_tb_top/UART_DUT/uart_unit/uart_rx_unit/dout              8'h00
/uart_tb_top/UART_DUT/uart_unit/uart_rx_unit/state_reg         idle

Now        143230830 ns
Cursor 1   201402 ns
```



```
                               Msgs
_unit/clk                      1'h0
_unit/reset                    1'h0
_unit/tick                     1'h0
_unit/uart_tx_unit/tx_start    1'h0
_unit/uart_tx_unit/tx          1'h1
_unit/uart_tx_unit/tx_done_... 1'h0
_unit/uart_tx_unit/state_reg   idle
_unit/uart_rx_unit/rx          1'h1
_unit/uart_rx_unit/rx_done_... 1'h0
_unit/uart_rx_unit/smpl_tick   1'h0
_unit/uart_rx_unit/dout        8'h00
_unit/uart_rx_unit/state_reg   idle
```

Here are two of the same waveforms one being closer than the other, and as it may be hard to see, there are receives and transmits happening. I think this is what is being required of us.

Transcript

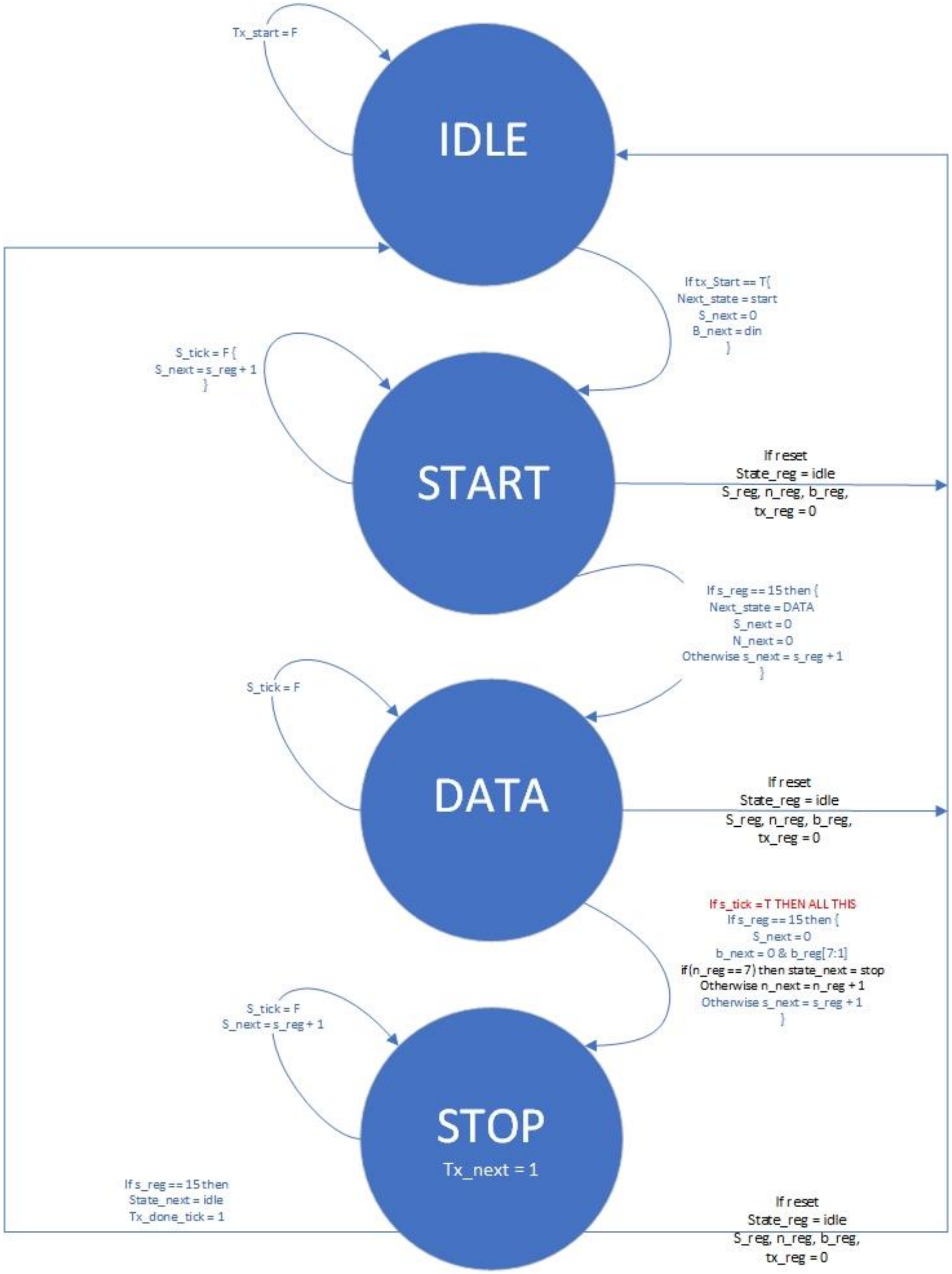
```
# Compile of uart_tx.sv failed with 4 errors.

# Compile of uart_tx.sv was successful.

restart -f

# Loading work.uart_tx

# ** Warning: (vsim-3017) C:/Users/Bram/OneDrive/Documents/School/Spring2020/ECE351/HW4/uart.sv(49): [TFMPC] - Too few port connections. Expected 7, found 6.

#    Time: 0 ns  Iteration: 0  Instance: /uart_tb_top/UART_DUT/uart_unit/uart_rx_unit File: C:/Users/Bram/OneDrive/Documents/School/Spring2020/ECE351/HW4/uart_rx.sv

# ** Warning: (vsim-3722) C:/Users/Bram/OneDrive/Documents/School/Spring2020/ECE351/HW4/uart.sv(49): [TFMPC] - Missing connection for port 'smpl_tick'.

run -all

# TX RX Loopback 128 tests PASSED!!!

# Time = 143230830:          END OF UART TEST!!!

# ** Note: $stop    : C:/Users/Bram/OneDrive/Documents/School/Spring2020/ECE351/HW4/uart_tb_top.sv(79)

#    Time: 143230830 ns  Iteration: 1  Instance: /uart_tb_top

# Break in Task run_test at C:/Users/Bram/OneDrive/Documents/School/Spring2020/ECE351/HW4/uart_tb_top.sv line 79
```

These are the results when the testbench is ran. It successfully runs! Woohoo!

Challenges & Discussion

A large challenge that I faced was learning about the loopback section and oversampling. When I was an intern at Intel, I was tasked with designing a FPGA development board to assist in future boards using serialized data. We name it the DIB for Debug Interface Board, and one of its components was UART. I found online IP through Intel's resources that created UART. I was able instantiate it in Verilog, and I wrote a program to take keyboard input and display it on the screen through UART. This made this much easier, but I never had to deal with the pulling of the lines in UART (only had to write my own I2C program for the board). So, ultimately it was fairly straight forward what was needed to be done, but the learning curve of understanding the given source code, and the slight tweaks in this code from previous code I have worked with added to the difficulty.

Transmit Transition



The parts that I had to write were data and stop. For data we wanted to wait until the oversampling of 16 was done (15) then mask the upper 7 bits of the byte being sent and send the LSB. Then we want to do these 7 more times for sending all the data and transition into stop. Inside stop we are polling for oversampling, then once that's done, we change the state to go back to idle and assert the tx done tick.

Conclusion

In the end the homework to my knowledge was done successfully. It was more challenging than all the others, and I think there are ways to ease into this one more than the way we did it. As I previously talked about, I used someone else's UART programs and added my own code to take input from the user and display it on an COM port on the screen. I think this would be a cool assignment to see in the future of this class. Maybe as a build up to this homework. It will allow more people to become adjusted to UART prior to writing UART code. It could act as the instantiation and testbenches homework assignments because that's mostly what it was when I did. Then again, I think you would have to have physical hardware to do this. The program I used was Tera Term for the COM ports.