

*******DRAFT*******

**Testing MyHDL created `uart_tx.v` & `uart_rx.v`
for the CATBOARD**

based `UART_RX.v` & `UART_TX.v`

On a RPi2B

05/01/18

*******DRAFT*******

1.0)

`“cd uart_txrx/catboard”`

2.0)

`“./build-Prog-cat.sh 3”`

This invokes several scripts. The first creates the **`“catboard.blif”`**. The 2nd creates the **`“catboard.txt”`**. The 3rd creates the **`“catboard.bin”`**. If these files were created the last script programs the FPGA,

`“catboard.sh”`

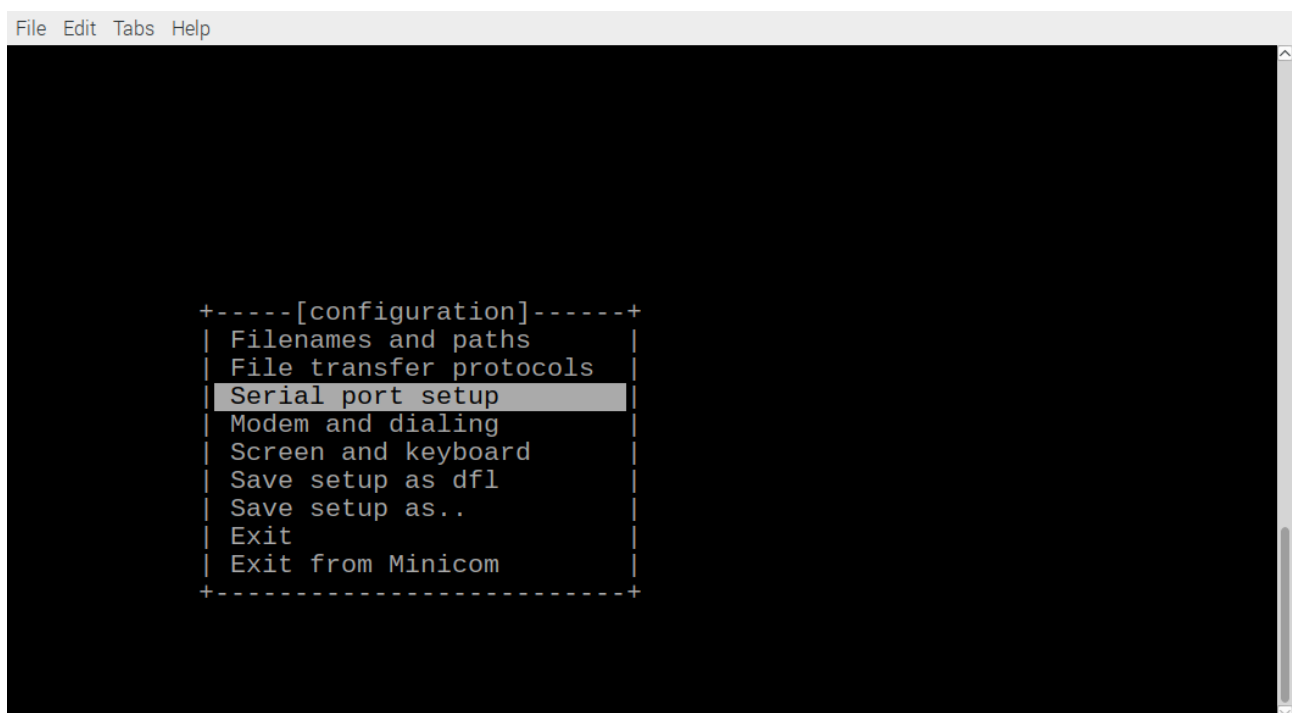
`“catboard_pnr.sh”`

`“catboard_bin.sh”`

`“sudo ~/catboard_yosys/config_cat catboard.bin”`

3.0)

`“sudo minicom -s”`

A screenshot of a terminal window with a menu displayed. The window has a title bar with 'File Edit Tabs Help'. The menu is titled '+-----[configuration]-----+' and lists several options: 'Filenames and paths', 'File transfer protocols', 'Serial port setup' (which is highlighted with a grey background), 'Modem and dialing', 'Screen and keyboard', 'Save setup as dfl', 'Save setup as..', 'Exit', and 'Exit from Minicom'. The menu is enclosed in a dashed border.

Depressing **`“Enter”`** brings up a new screen

```
File Edit Tabs Help

+-----+
| A -   Serial Device       : /dev/ttyAMA0 |
| B - Lockfile Location    : /var/lock     |
| C -   Callin Program      :              |
| D -   Callout Program     :              |
| E -   Bps/Par/Bits        : 115200 8N1   |
| F - Hardware Flow Control : No           |
| G - Software Flow Control : No           |
|                                     |
| Change which setting? █ |
+-----+
| Screen and keyboard      |
| Save setup as dfl        |
| Save setup as..          |
| Exit                     |
| Exit from Minicom        |
+-----+
```

*If the onboard Rpi is used /dev/ttyAMA0 is used as above.
If the pmodusbuart is connected to pm2-A /dev/ttyUSB0 is used as below.*

This requires that catboard.pcf be modified

<i>catboard.pcf.rpi</i>	<i>catboard.pcf.pm2-A</i>
<i>set_io i_UART_RX T15</i>	<i>set_io i_UART_RX B5</i>
<i>set_io o_UART_TX T14</i>	<i>et_io o_UART_TX B3</i>

```
File Edit Tabs Help

+-----+
| A -   Serial Device       : /dev/ttyUSB0 |
| B - Lockfile Location    : /var/lock     |
| C -   Callin Program      :              |
| D -   Callout Program     :              |
| E -   Bps/Par/Bits        : 115200 8N1   |
| F - Hardware Flow Control : No           |
| G - Software Flow Control : No           |
|                                         |
|   Change which setting?                |
+-----+
| Screen and keyboard |
| Save setup as dfl   |
| Save setup as..     |
| Exit                |
| Exit from Minicom   |
+-----+
```

Depressing ***“Enter”*** and scrolling down to ***“Exit”***.

```
File Edit Tabs Help

Welcome to minicom 2.7

OPTIONS: I18n
Compiled on Apr 22 2017, 09:14:19.
Port /dev/ttyAMA0, 15:05:52

Press CTRL-A Z for help on special keys

1234567890!@#$%^&*()-_+=, .<>/?abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
XYZ█
```

Typing characters get echoed. The programmed FPGA is transmitting the received characters correctly

Thu Apr 19 2018

Initial code was download from
<https://www.edaplayground.com/x/Pgf>

<http://www.nandland.com>

Command to create **testuart** from **UART_TB.v** **UART_RX.v** **UART_TB/v** includes **UART_TX.v**

iverilog -o testuart UART_TB.v UART_RX.v

vvp testuart

VCD info: dumpfile dump.vcd opened for output.

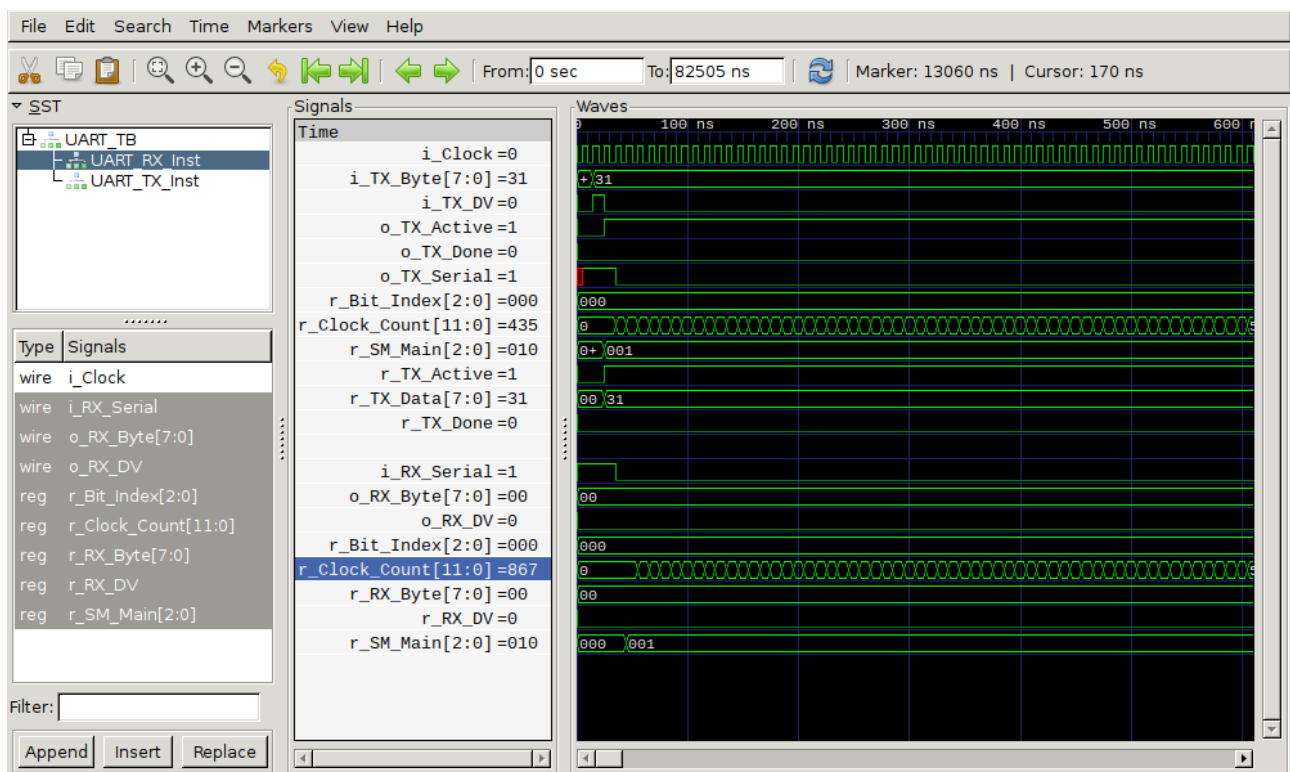
Test Passed - Correct Byte Received

This code was created

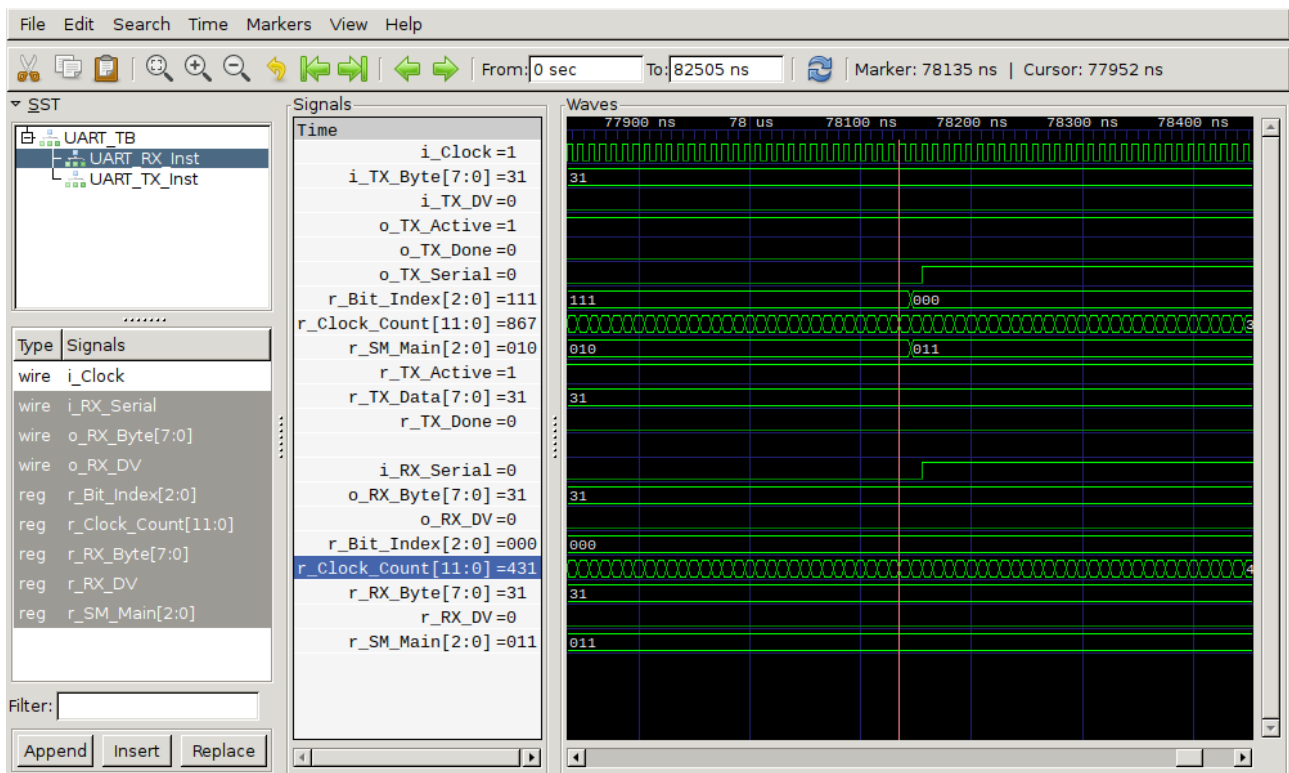
```
// Testbench uses a 25 MHz clock
// Want to interface to 115200 baud UART
// 25000000 / 115200 = 217 Clocks Per Bit.
parameter c_CLOCK_PERIOD_NS = 40;
At 100MHz the parameter c_CLKS_PER_BIT = 868; is required.
```

**Note: This does not fit in 8 bits which required a change in both UART_TX.v and UART_RX.v
reg [7:0] r_Clock_Count = 0; → reg [11:0] r_Clock_Count = 0;**

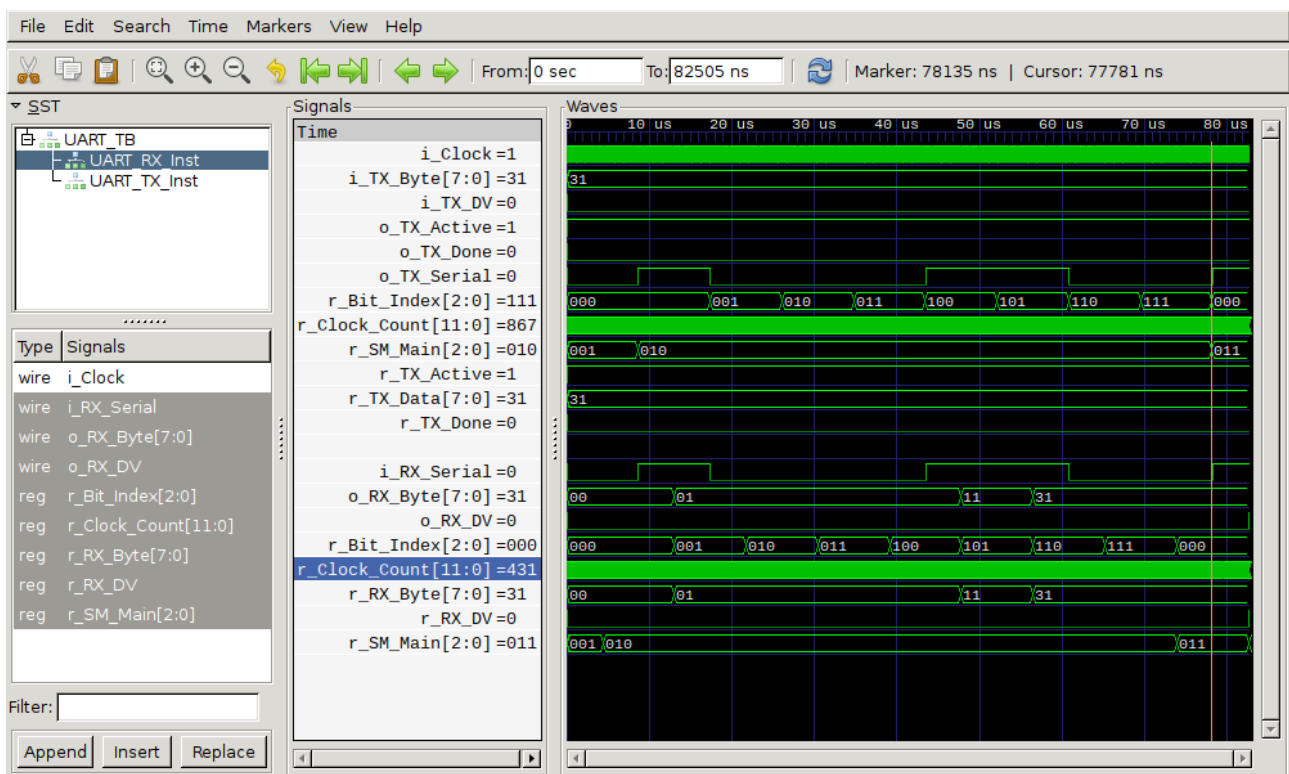
In UART_TB.v parameter c_CLKS_PER_BIT = 217 → parameter c_CLKS_PER_BIT = 868;



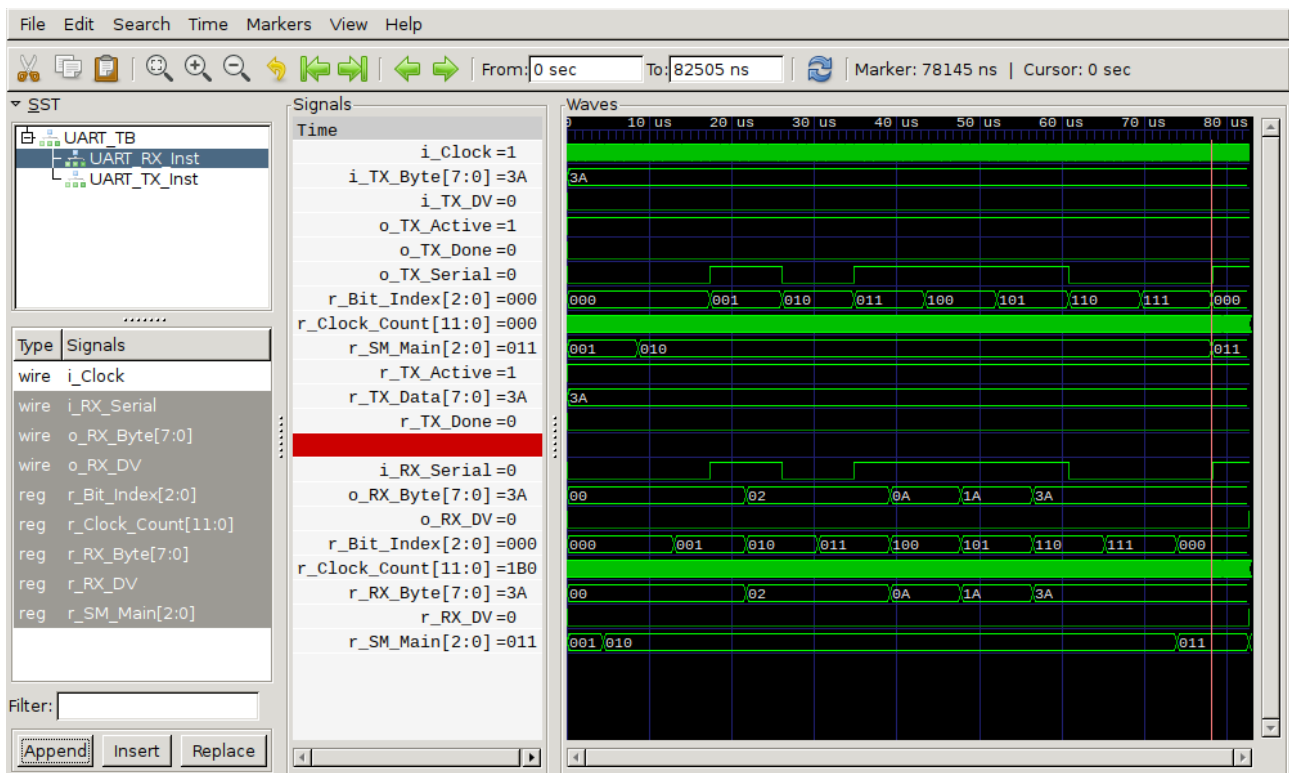
Checking that **r_Clock_Count** does not overflow with 868.



Simulation of 00110001 being transmitted and recived.



Simulation of 00110101 being transmitted and recived.



Both the UART_TX and UART_RX are a case statement
 IDLE, TX_START_BIT, TX_DATA_BITS, TX_STOP_BIT, and C:EANUP.
 IDLE, RX_START_BIT, RX_DATA_BITS, RX_STOP_BIT, and C:EANUP.
 In MyHDL the case statement was written for uart_tx.

if(r_SM_Main==IDLE):

•
•

elif (r_SM_Main==TX_START_BIT):

•

```

        .
    elif (r_SM_Main==TX_DATA_BITS):
        .
        .
    elif (r_SM_Main==TX_STOP_BIT):
        .
        .
    else:
        .
        .
and the uart_rx was similar
if(r_SM_Main==IDLE):
    .
    .
    elif (r_SM_Main==RX_START_BIT):
        .
        .
    elif (r_SM_Main==RX_DATA_BITS):
        .
        .
    elif (r_SM_Main==RX_STOP_BIT):
        .
        .
    else:
        .
        .

```

In the simulation the r_TX_BYTE is set to byte to be transmitted and a 1 clock wide r_TX_DV.

```

r_TX_DV  <= 1'b1;
r_TX_Byte <= 8'h3A;
@(posedge r_Clock);
r_TX_DV <= 1'b0;

```

Testing at 100 MHz appears okay
Using MyHDL to create *uart_rx.v* with *uart_rx.py*
& *uart_tx.v* with *uart_tx.py*

iverilog -o testuart tb_dut_uart_txx.v uart_rx.v

vvp testuart
VCD info: dumpfile dump.vcd opened for output.
Test Passed - Correct Byte Received

Several simulations were tested 0x3F, 0x32, 0x38, 0x3A, 0x41.
3F transmitted & received

00111111
32 transmitted & received
00110010

38 transmitted & received

00111000
3A transmitted & received

00111010
41 transmitted & received

01000001

uart_txx.sh catboard.sh

uart_txx_pnr.sh catboard_pnr.sh
uart_loopback.pcf catboard.pcf


```
uart_txx_bin.sh      catboard_bin.sh
```

```
/home/pi/uart_rxtx/MyHDL  
sudo ~/catboard_yosys/config_cat uart_loopback.bin
```

```
/home/pi/uart_rxtx/catboard  
sudo ~/catboard_yosys/config_cat catboard.bin
```

```
.
```