**********Draft***********
EchoTest
03/09/21
**********Draft***********

This simulation created using https://zipcpu.com/tutorial/lsn-09-serialrx.pdf
and the scala file "https://github.com/develone/SpinalTemplateSbt/blob/master/src/main/scala/mylib/UartCtrlGenerics.scala
privided by PasCal++@volatile-static on "https://gitter.im/SpinalHDL/SpinalHDL"
https://github.com/develone/SpinalTemplateSbt/blob/master/echotest/test_rx_ex-09/notes_testsimulator.txt & https://github.com/develone/SpinalTemplateSbt/blob/master/echotest/test_rx_ex-09/sim.png. Maybe this can help others. Thanks

and the scala file "https://github.com/develone/SpinalTemplateSbt/blob/master/src/main/scala/mylib/UartCtrlGenerics.scala
privided by PasCal++@volatile-static on "https://gitter.im/SpinalHDL/SpinalHDL"
Creating a simulation to test UartCtrlUsageExample.v using ZipCPU ex-09-uartrx.tgz

This simulation uses rxuart_tb.cpp, testb.h, uartsim.h, uartsim.cpp. The Verilog rxuart.v and f_txuart.v.
The file psalm.txt is sent to rxuart module during the simulation.
psalm.txt

```
==============================================================================
|                                                   |
| The LORD is my shepherd; I shall not want.                           |
| He maketh me to lie down in green pastures: he leadeth me beside the still |
|   waters.                                                     |
| He restoreth my soul: he leadeth me in the paths of righteousness for his  |
|   name's sake.
| Yea, though I walk through the valley of the shadow of death, I will fear  |
|   no evil: for thou art with me; thy rod and thy staff they comfort me.    |
| Thou preparest a table before me in the presence of mine enemies: thou     |
|   anointest my head with oil; my cup runneth over.                  |
| Surely goodness and mercy shall follow me all the days of my life: and I   |
|   will dwell in the house of the LORD for ever.                      |
|                                                   |
==============================================================================
```

step1
UartCtrlUsageExample.v was copied to rxuart.v. The module UartCtrlUsageExample replace module rxuart.
Made modoficatation rxuart.v, testb.h, rxuart_tb.cpp
make
./rxuart_tb psalm.txt

~~SUCCESS - all 0 characters matched~~
~~PASS~~

After adding o_wr to wrapper design rxuart.v for UartCtrlUsageExample.v.
Now the running of ./rxuart_tb psalm.txt provides the psalm.txt to std-out.
This was accomplished by adding the following Verilog code to rxuart.v

```
+  output              o_wr,
   input              clk
   //input              reset
 );
@@ -66,7 +67,14 @@ module rxuart (
   wire       [7:0]    write_m2sPipe_payload;
   reg                 write_m2sPipe_rValid;
   reg        [7:0]    write_m2sPipe_rData;
-
+
+  reg dum;
+
+  always @ (posedge clk) begin
+    dum <= uartCtrl_1_io_read_valid;
+  end
```

```
./rxuart_tb psalm.txt
===========================================================================
=========
|                                                                         |
| The LORD is my shepherd; I shall not want.                              |
| He maketh me to lie down in green pastures: he leadeth me beside the still |
|   waters.                                                               |
| He restoreth my soul: he leadeth me in the paths of righteousness for his |
|   name's sake.
| Yea, though I walk through the valley of the shadow of death, I will fear |
|   no evil: for thou art with me; thy rod and thy staff they comfort me.  |
| Thou preparest a table before me in the presence of mine enemies: thou  |
|   anointest my head with oil; my cup runneth over.                      |
| Surely goodness and mercy shall follow me all the days of my life: and I |
|   will dwell in the house of the LORD for ever.

SUCCESS - all 966 characters matched
PASS
```

A rxuat.vcd is created.

https://github.com/develone/SpinalTemplateSbt/blob/master/echotest/test_rx_ex-09/sim.png
Needed to decrease the simulator Baud rate to match the Verilog code.
Note : This is the change in uartsim.cpp
setup(868);   // Set us up for a baud rate of CLK/868
The original value was 25 instead of 868 which sets the Baudrate to 4MBaud.

He restoreth my soul: he leadeth me in the paths of righteousness for his
https://github.com/develone/SpinalTemplateSbt/blob/master/echotest/test_rx_ex-09/4MBsim.pngPasCal++@volatile-static
PasCal++@volatile-static wrote @develone Could you complete the doc for them? As soon as I get it all working I will try and put together a document.

Will a pdf be the the desired format?
From "https://github.com/develone/SpinalTemplateSbt/blob/master/echotest/test_rx_ex-09/4MBsim.png". I see io_leds
matching what the simulator, is sending "He restoreth my soul: he lead". What I do not see "io_uart_txd" toggle like "io_uart_rxd" which is the output of the simulator.
When I compiled "https://github.com/develone/SpinalTemplateSbt/blob/master/src/main/scala/mylib/UartCtrlGenerics.scala"

```
  //Write the value of switch on the uart each 4000 cycles
  val write = Stream(Bits(8 bits))
  write.valid := CounterFreeRun(2000).willOverflow
  write.payload := io.switchs
  write >-> uartCtrl.io.write

  //Write the 0x55 and then the value of switch on the uart each 4000 cycles
  //val write = Stream(Fragment(Bits(8 bits)))
  //write.valid := CounterFreeRun(4000).willOverflow
  //write.fragment := io.switchs
  //write.last := True
```

The above is what I am testing the simulator and HW HX8K FPGA does not echo the charctertes back.

Am I wrong in thinking that the characters should be echo back?

This is the current testing that I am working on.
"https://gist.githubusercontent.com/develone/3462de27a0064bbf7bea847b9b1bc06b/raw/3e1a36bf4b8e30fd64c06e2b0f7d07b0fe79d7d2/rxuart.v.reset"
When I compiled "https://github.com/develone/SpinalTemplateSbt/blob/master/src/main/scala/mylib/UartCtrlGenerics.scala"

```
  //Write the 0x55 and then the value of switch on the uart each 4000 cycles
  val write = Stream(Fragment(Bits(8 bits)))
  write.valid := CounterFreeRun(4000).willOverflow
  write.fragment := io.switchs
  write.last := True
  write.m2sPipe().insertHeader(0x55).toStreamOfFragment >> uartCtrl.io.write
```

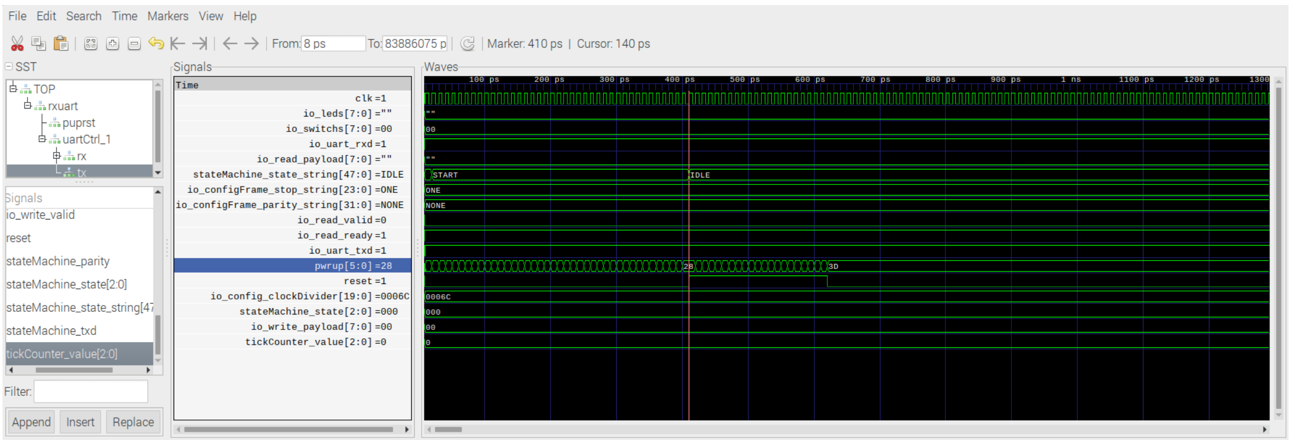
With the above lines uncommented I still needed add a power up module to to get the it to work in simulation and in HW
on my HX8K FPGA. "https://github.com/develone/SpinalTemplateSbt/blob/master/xmit55/XmitTestTop.v"

```
make
verilator -O3 -MMD --trace -Wall -cc rxuart.v
%Error-ASSIGNIN: rxuart.v:77:5: Assigning to input/const variable: 'reset'
   77 |     reset,
      |     ^~~~~
%Error: Exiting due to 1 error(s)
        ... See the manual and https://verilator.org for more assistance.
make: *** [Makefile:63: obj_dir/Vrxuart.cpp] Error 1
```

```
module rxuart (
  output              io_uart_txd,
  input               io_uart_rxd,
```

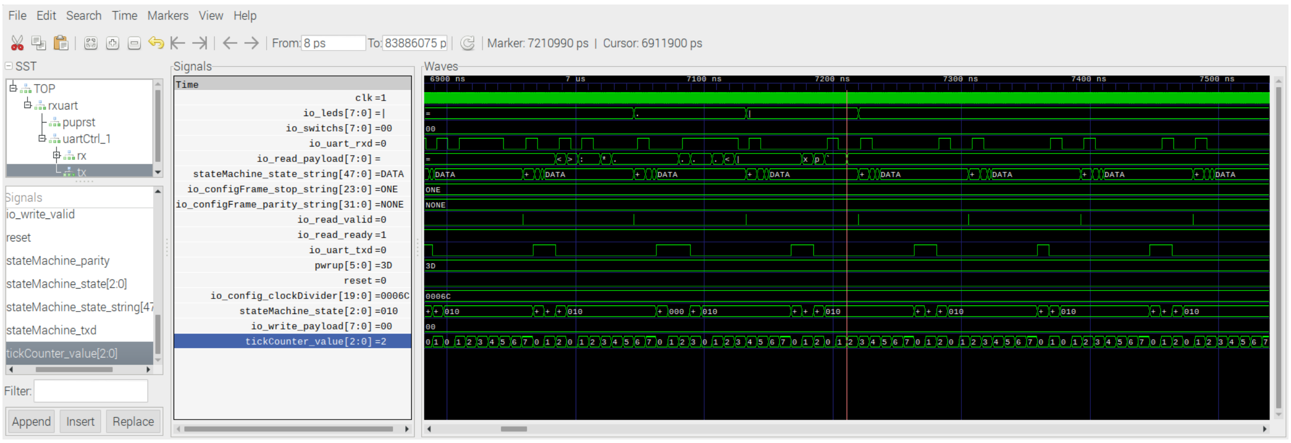
```
  input      [7:0]    io_switchs,
  output     [7:0]    io_leds,
  input               clk
  //input             reset
);

  /* verilator lint_off UNUSED */
  wire reset;
  /* verilator lint_off UNUSED */
```
Lines 78-22 of rxuart.v
```
pwruprst puprst (
    clk,
    reset,
    pwrup
);
```
Lines 782-814 of rxuart.v
```
module pwruprst (
    clk,
    reset,
    pwrup
);


input clk;
output reset;
reg reset;
output [5:0] pwrup;
reg [5:0] pwrup;




always @(posedge clk) begin: PWRUPRST_LOGIC2
    /* verilator lint_off SYNCASYNCNET */
    if (((reset == 0) && (pwrup == 40))) begin
        reset <= 1;
    end
    else begin
        if ((pwrup <= 60)) begin
            pwrup <= (pwrup + 1);
        end
    end
    if ((pwrup == 60)) begin
        reset <= 0;
    end
end

endmodule
```

Generating a reset on powerup. Signal reset is driven hi after 0x28 clocks and remains hi until the clock reaches 0x3C at which time is return lo.



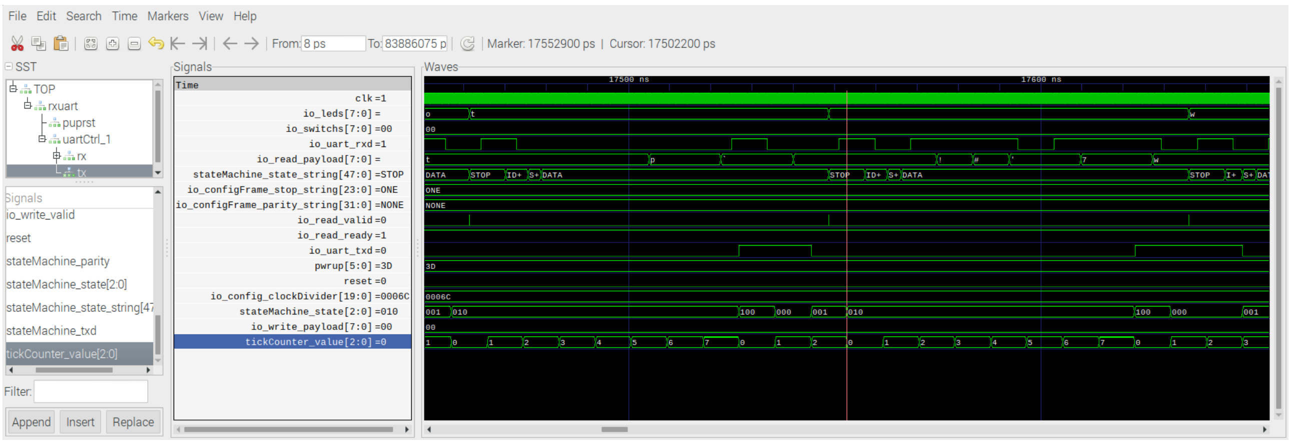
```
From: 8 ps   To: 83886075 p   Marker: 410 ps | Cursor: 140 ps

Time
                              clk =1
                     io_leds[7:0] =""
                  io_switchs[7:0] =00
                      io_uart_rxd =1
              io_read_payload[7:0] =""
  stateMachine_state_string[47:0] =IDLE
 io_configFrame_stop_string[23:0] =ONE
io_configFrame_parity_string[31:0] =NONE
                    io_read_valid =0
                    io_read_ready =1
                      io_uart_txd =1
                       pwrup[5:0] =28
                            reset =1
     io_config_clockDivider[19:0] =0006C
          stateMachine_state[2:0] =000
             io_write_payload[7:0] =00
           tickCounter_value[2:0] =0
```

Drilling down why the io_uart_txd only goes low at start bit and no other toggles during the tickCounter_values 0 thru 7.



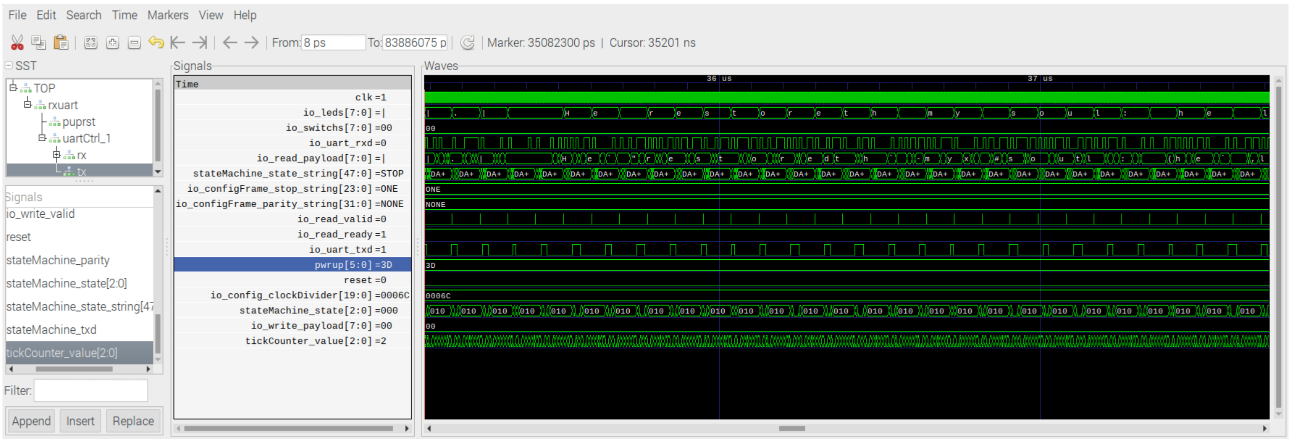
```
From: 8 ps   To: 83886075 p   Marker: 7210990 ps | Cursor: 6911900 ps

Time
                              clk =1
                     io_leds[7:0] =|
                  io_switchs[7:0] =00
                      io_uart_rxd =0
              io_read_payload[7:0] =
  stateMachine_state_string[47:0] =DATA
 io_configFrame_stop_string[23:0] =ONE
io_configFrame_parity_string[31:0] =NONE
                    io_read_valid =0
                    io_read_ready =1
                      io_uart_txd =0
                       pwrup[5:0] =3D
                            reset =0
     io_config_clockDivider[19:0] =0006C
          stateMachine_state[2:0] =010
             io_write_payload[7:0] =00
           tickCounter_value[2:0] =2
```

Drilling down why the io_uart_txd only goes low for the



```
From: 8 ps   To: 83886075 p   Marker: 17552900 ps | Cursor: 17502200 ps

Time
                              clk =1
                     io_leds[7:0] =
                  io_switchs[7:0] =00
                      io_uart_rxd =1
              io_read_payload[7:0] =
  stateMachine_state_string[47:0] =STOP
 io_configFrame_stop_string[23:0] =ONE
io_configFrame_parity_string[31:0] =NONE
                    io_read_valid =0
                    io_read_ready =1
                      io_uart_txd =0
                       pwrup[5:0] =3D
                            reset =0
     io_config_clockDivider[19:0] =0006C
          stateMachine_state[2:0] =010
             io_write_payload[7:0] =00
           tickCounter_value[2:0] =0
```

The signal io_uart_rxd toggles correctly since the signal io_leds displays “ He restoreth my soul: he”. This matches with the what the uartsim is sending.



```
Signals
io_write_valid
reset
stateMachine_parity
stateMachine_state[2:0]
stateMachine_state_string[47
stateMachine_txd
tickCounter_value[2:0]
```

```
Time
clk =1
io_leds[7:0] =|
io_switchs[7:0] =00
io_uart_rxd =0
io_read_payload[7:0] =|
stateMachine_state_string[47:0] =STOP
io_configFrame_stop_string[23:0] =ONE
io_configFrame_parity_string[31:0] =NONE
io_read_valid =0
io_read_ready =1
io_uart_txd =1
pwrup[5:0] =3D
reset =0
io_config_clockDivider[19:0] =0006C
stateMachine_state[2:0] =000
io_write_payload[7:0] =00
tickCounter_value[2:0] =2
```

The Verilator simulator provides to std-out the txt of psalm.txt file.

```
-                       //uart->setup(baudclocks);
+                       baudclocks = 868;
+                       uart->setup(baudclocks);
            .
            .
            .
-                                   //if (tb->m_core->o_wr) {
-                                               //num_received++;
-                                               //putchar(tb->m_core->o_data);
-                                   //}
+                                   if (tb->m_core->o_wr) {
+                                               num_received++;
+                                               putchar(tb->m_core->io_leds);
+                                   }
```

Scala code

```
  //Write the value of switch on the uart each 4000 cycles
  val write = Stream(Bits(8 bits))
  write.valid := CounterFreeRun(2000).willOverflow
  write.payload := io.switchs
  write >-> uartCtrl.io.write
```

```
  assign _zz_2 = 1'b1;
  assign write_valid = _zz_6;
  assign write_payload = io_switchs;
  assign write_ready = ((1'b1 && (! write_m2sPipe_valid)) || write_m2sPipe_ready);
  assign write_m2sPipe_valid = write_m2sPipe_rValid;
  assign write_m2sPipe_payload = write_m2sPipe_rData;
  assign write_m2sPipe_ready = uartCtrl_1_io_write_ready;
  always @ (posedge clk) begin
    if(uartCtrl_1_io_read_valid)begin
      _zz_1 <= uartCtrl_1_io_read_payload;
    end
    if(write_ready)begin
      write_m2sPipe_rData <= write_payload;
  end
end
```