Task 2

Objective: Implement a UART loopback mechanism where transmitted data is immediately received back, facilitating testing of UART functionality.

UART Serial Connection Circuit Diagram Components:

1. FPGA Mini Board (VSD-Squadron-FM)

2. USB-to-TTL Serial Converter (e.g., FTDI FT232RL)

Loopback Operation:

1. Transmit Data: Send data from the UART TX Pin

2. Receive Data: Receive data at the UART RX Pin (looped back from the TX Pin)

3. Compare Data: Verify that the transmitted data matches the received data

Connections:

1. UART TX Pin: Connect to UART RX Pin on the same FPGA (loopback connection)

2. UART RX Pin: Receives data transmitted by the UART TX Pin

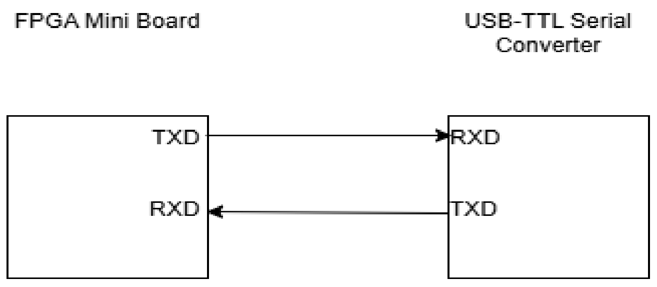
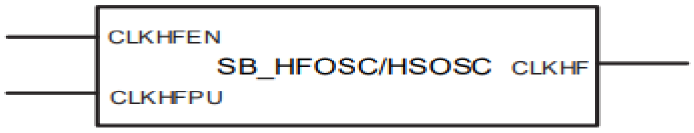
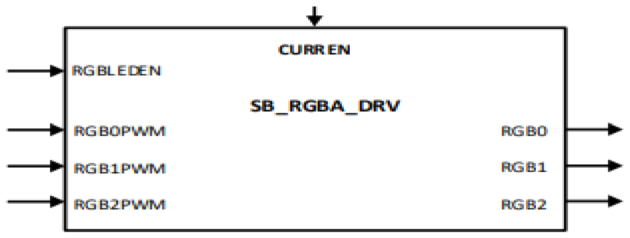


Figure1:Connection Diagram

Table 1 :Verilog Code1

Verilog Code	Explanations
include "uart_trx.v"	
module top (output wire led_red , output wire led_blue , output wire led_green , input wire hw_clk, output wire uarttx , input wire uartrx);	Module Declaration with output and input ports
wire int_osc ; reg [27:0] frequency_counter_i;	Declaration of intermediate signals
assign uarttx = uartrx;	UART RX pin is connected to UART TX pin.
SB_HFOSC #(	High Frequency Oscillator with output divider 
.CLKHF_DIV ("0b10"))	CLKHF_DIV =0b10 ;Divider setting for 12MHZ clock signal
u_SB_HFOSC (.CLKHFPU(1'b1),	CLKHFPU=1'b1; Power up signal Active HIGH
.CLKHFEN(1'b1),	CLKHFEN=1'b1; Clock Enable signal Active HIGH
.CLKHF(int_osc));	CLKHF=int_osc; Internal Oscillator Output is connected to the wire int_osc
always @(posedge int_osc) begin frequency_counter_i <= frequency_counter_i + 1'b1; end	Generation of 9600 Hz clock During positive edge of the clock signal int_osc, frequency counter value is incremented by one.
SB_RGBA_DRV RGB_DRIVER	RGB LED Driver primitive Instantiation  CURREN RGBLEDEN SB_RGBA_DRV RGB0PWM RGB1PWM RGB2PWM RGB0 RGB1 RGB2
(.RGBLEDEN(1'b1),	RGBLEDEN=1'b1 ;Enable control for RGB LED is Active HIGH
	RGB0PWM =1'b0 ;PWM signal for RGB_PAD0 is connected to

.RGB0PWM (uartrx), .RGB1PWM (uartrx), .RGB2PWM (uartrx), .CURREN (1'b1), .RGB0 (led_green), .RGB1 (led_blue), .RGB2 (led_red)); defparam RGB_DRIVER.RGB0_CURRENT = "0b000001"; defparam RGB_DRIVER.RGB1_CURRENT = "0b000001"; defparam RGB_DRIVER.RGB2_CURRENT = "0b000001";	UART RX. RGB1PWM =1'b0 ;PWM signal for RGB_PAD1 is is connected to UART RX. RGB2PWM =1'b1 ;PWM signal for RGB_PAD2 is is connected to UART RX. CURREN =1'b1 ; Power up signal Active HIGH RGB0=led_red ; RGB LED Driver output RGB0 is connected to red led. RGB1=led_green ; RGB LED Driver output RGB1 is connected to green led. RGB0=led_blue ; RGB LED Driver output RGB2 is connected to blue led. 0b000001 means 4mA current is supplied to red led. 0b000001 means 4mA current is supplied to green led. 0b000001 means 4mA current is supplied to blue led.
endmodule	Ending the Module.

Table 2 :Verilog Code2

Verilog Code	Explanations
include "uart_trx.v"	
module uart_tx_8n1 (clk, txbyte, senddata, txdone, tx);	Module Declaration with output and input ports input clock outgoing byte trigger tx outgoing byte sent tx wire

input clk;input[7:0] txbyte; input senddata;	Declaration of input signals
output txdone; output tx;	Declaration of output signals.

parameter STATE_IDLE=8'd0;
 	
parameter STATE_STARTTX=8'd1;
 	
parameter STATE_TXING=8'd2;
 	
parameter STATE_TXDONE=8'd3
	Parameters

reg[7:0] state=8'b0;
 	
reg[7:0] buf_tx=8'b0;
 	
reg[7:0] bits_sent=8'b0;
 	
reg txbit=1'b1;
 	
reg txdone=1'b0;
	State variables

assign tx=txbit	

always @ (posedge clk) begin
 	
if (senddata == 1 && state == STATE_IDLE) begin
 	
state <= STATE_STARTTX;
 	
buf_tx <= txbyte;
 	
txdone <= 1'b0;
 	
end
 	
else if (state == STATE_IDLE) begin
 	
txbit <= 1'b1;
 	
txdone <= 1'b0;
 	
end
 	
if (state == STATE_STARTTX) begin
 	
txbit <= 1'b0;
 	
state <= STATE_TXING;
 	
end
 	
if (state == STATE_TXING && bits_sent < 8'd8) begin
	start sending?
	
	idle at high
	
	send start bit (low)

txbit <= buf_tx[0];
buf_tx <= buf_tx>>1;
bits_sent = bits_sent + 1;
end
else if (state == STATE_TXING) begin
txbit <= 1'b1;
bits_sent <= 8'b0;
state <= STATE_TXDONE;
end
if (state == STATE_TXDONE) begin
txdone <= 1'b1;
state <= STATE_IDLE;
end
end
	clock data out
	
	send stop bit (high)
	
	tx done

endmodule	Ending the Module.

Procedure:

1.Connect the board to the computer using USB-C and ensuring FTDI connection.

Ensure the connection by typing ***lsusb_***command on terminal window.

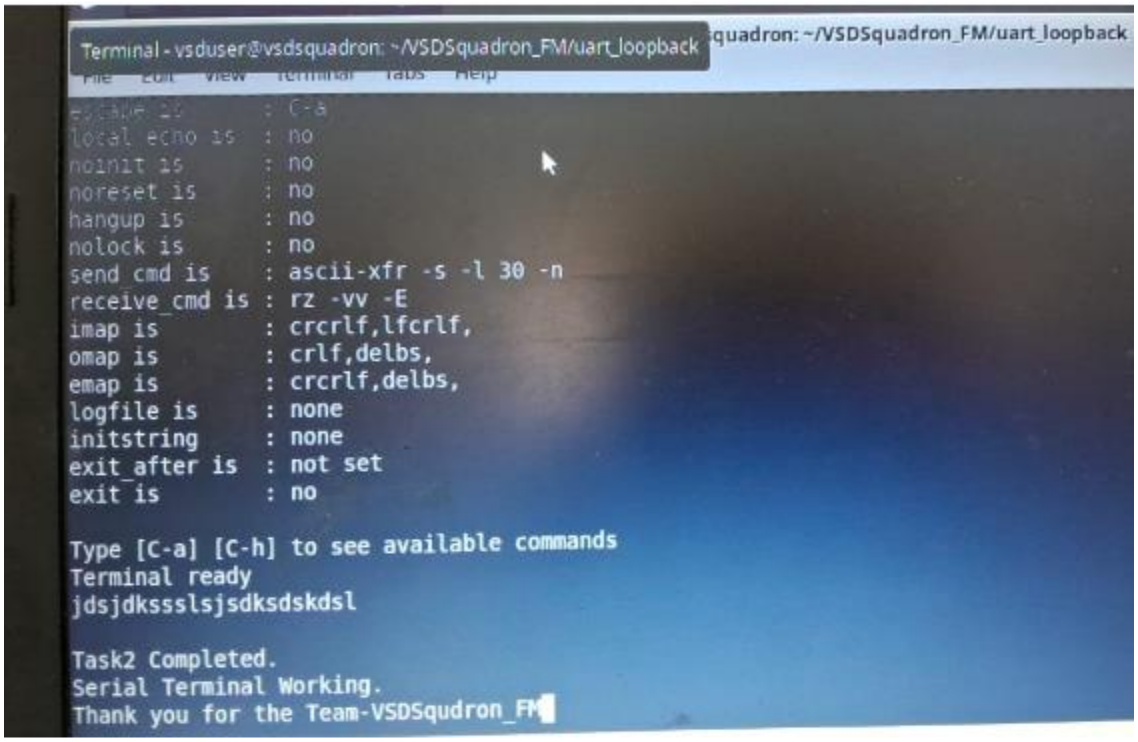
2.Commands for building and flashing the Verilog code:

- Run ***'make clean'*** to clear any previous builds
- Run ***'make build'*** to compile the design
- Run ***'sudo make flash'*** to program the FPGA board

3. Install picocom in the Linux terminal by running command ***sudo apt install picocom***

4. Run ***make terminal*** to to run the picocom

Testing Results:



Terminal - vsduser@vsdsquadron: ~/VSDSquadron_FM/uart_loopback

```
escape is        : C-a
local echo is    : no
noinit is        : no
noreset is       : no
hangup is        : no
nolock is        : no
send_cmd is      : ascii-xfr -s -l 30 -n
receive_cmd is   : rz -vv -E
imap is          : crcrlf,lfcrlf,
omap is          : crlf,delbs,
emap is          : crcrlf,delbs,
logfile is       : none
initstring       : none
exit_after is    : not set
exit is          : no

Type [C-a] [C-h] to see available commands
Terminal ready
jdsjdksslsjsdksdskdsl

Task2 Completed.
Serial Terminal Working.
Thank you for the Team-VSDSqudron_FM
```