THE SERIAL CARAVAN

(Design and Verification of a UART Transmitter and Receiver System using Verilog HDL)

- **Category: Mid-Level Digital Design**
- **Team Name: FMS Fours**
- **Branch/year: Electronics and Communication Engineering (ECE)/2nd year**

Team Details:

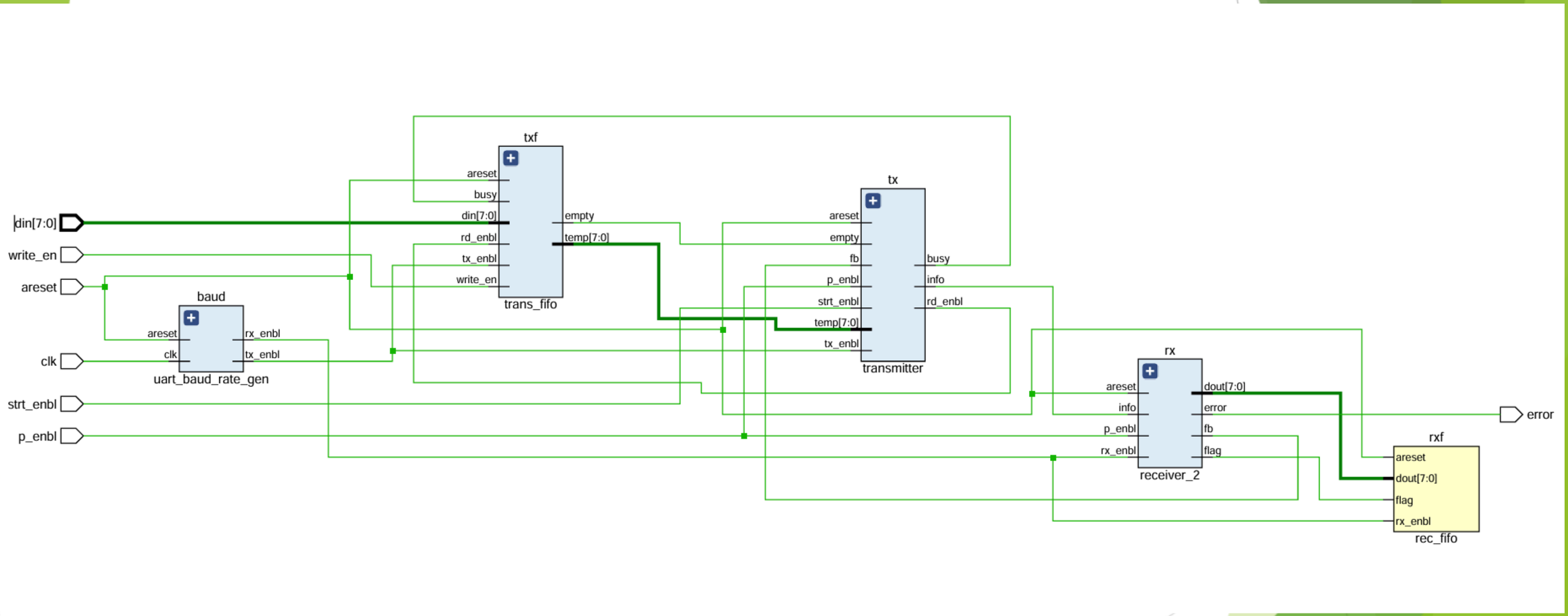
***Divyansh Awasthi(20244066),phone no: 62619 68850**
Email id: divyansh.awasthi616@gmail.com

*** Arun kumar(20244038),phone no.:9772335423**
Email id:Arun.20244038@mnnit.ac.in

***Anjanee Kumar(20244021), phone no.:7607 613 093**
Email id: anjaneekumar76076@gmail.com

***Krishna Mehrotra(20244096),phone no. :79058 44272**
Email id: Krishnamehrotra0406@gmail.com

BLOCK DIAGRAM



Baud Rate Generator

- **module uart_baud_rate_gen** //Defines the module name.
- **parameter integer CLK_FREQ = 48000000** // Sets the default system clock to 48 MHz. Parameters allow you to reuse this code for different boards by just changing this value.
- **parameter integer BAUD_RATE = 1000000:** Sets the target communication speed (1 Mbps).
- **input wire clk, areset:** The global system clock and an asynchronous reset (active high).
- **output reg tx_enbl, rx_enbl**: The generated clock signals. Note they are defined as reg because they are assigned inside an always block.
- **reg [$clog2(TX_DIV_HALF)-1:0] tx_cnt** //These are counters. The $clog2 function automatically calculates the minimum number of bits needed to store the division value
- **TX_DIV_HALF:** Calculates how many clock cycles make up half a baud period (48,000,000 / 1,000,000 = 48). Since the code toggles the output, the actual period will be 48 \times 2 = 96 cycles.

- **RX_DIV_HALF**: Calculates the cycles for the 16x faster clock (48,000,000 / (16 * 1,000,000) = 3).
- **if (areset):** When the reset signal goes high, everything is cleared to zero immediately, regardless of the clock. This ensures the system starts in a known state.
- **rx_enbl** //It represents the clock signal used by the UART receiving logic to time the incoming serial data

- **if (tx_cnt == TX_DIV_HALF - 1):** The counter checks if it has reached the target number of cycles.
- **tx_cnt <= 0**: Once reached, it resets the counter to start over.

- **tx_enbl <= ~tx_enbl:** This is the "Toggle." If the signal was 0, it becomes 1, and vice versa. By toggling every "half" period, it creates a perfect 50% duty cycle square wave at the target Baud Rate.

UART Transmitter.

- **tx_enbl**: The baud rate clock pulses that drive the timing of each bit
- **strt_enbl:** A control pulse that triggers the start of a transmission.
- **p_enbl**: Parity Enable. If high, a parity bit will be calculated and sent.
- **fb**: Feedback signal from the receiver (used to trigger a re-transmission if an error occurred).
- **empty:** A signal from the FIFO; if high, there is no data to send.
- **temp**: The 8-bit parallel data currently waiting to be sent.
- **info**: The Serial Output Pin. This is where the physical transmission happens.
- **busy**: Tells the rest of the system the transmitter is currently occupied
- **rd_enbl:** The "Read Enable" sent to the FIFO to request the next byte
- **reg [1:0] ps, ns;**: Present State and Next State registers for the FSM (Finite State Machine).
- **reg [3:0] count**;: A counter used to track which bit (0 through 7, plus parity) is currently being shifted out.

Transmiter FIFO (First-In-First-Out) :-

- **input wire tx_enbl**: The clock source for the FIFO. It uses the Baud Rate clock to synchronize data movement with the transmitter.
- **input wire areset**: The asynchronous reset signal. When high, it clears the FIFO's pointers and status.
- **input wire write_en:** The "Push" signal. When this is high, the data on the din bus is saved into the memory.
- **input wire [7:0] din**: The 8-bit parallel data input coming into the FIFO.
- **input wire rd_enbl:** The "Pop" signal. This is controlled by the Transmitter. When the transmitter is ready for a new byte, it pulses this high to grab data from the FIFO
- **input wire busy:** A status signal from the transmitter indicating it is currently sending a byte.
- **output reg [7:0] temp**: The 8-bit output of the FIFO. This holds the byte that is currently being "read" and handed over to the transmitter.
- **output wire empty:** A status flag. It is high when the FIFO has no data left to send.

- **reg [7:0] mem [0:15]**;: This is the Memory Array. It consists of 16 "slots" (0 to 15), where each slot is 8 bits wide.
- **reg [3:0] rear_ptr;**: The Write Pointer (Tail). It points to the address where the next incoming byte from din will be stored.
- **reg [3:0] front_ptr;**: The Read Pointer (Head). It points to the address of the next byte that needs to be sent to the transmitter.
- **reg [4:0] count**;: This tracks the Fill Level (depth).

UART Receiver

- **input areset,**: The Asynchronous Reset. When this pin goes high, the receiver immediately clears all its internal memory and goes back to the idle state, regardless of what the clock is doing.
- **input p_enbl**,: Parity Enable. This is a control signal.
- **input rx_enbl**,: This is the Oversampling Clock (usually 16x the Baud Rate). This is the actual "heartbeat" used by the always block to time the sampling of the info line.
- **input info:**The serial input line (RX).
- **output dout**: The parallelized 8-bit data output.
- **output error**: High if parity is wrong or stop bit is missing.
- **output fb:** Feedback signal sent back to the transmitter to request re-send.
- **output flag**: Acts as a "Write Enable" for the FIFO memory.
- **reg [3:0] data_cnt:** Tracks which bit we are receiving (Bit 0 to 7, then Bit 8 for parity).
- **reg [3:0] s_cnt**: The oversampling counter. Since rx_enbl is 16x the baud rate, we count to 16 to move from one bit to the next.
- **reg [7:0] temp**: A shift register that holds bits as they arrive.
- **reg prev_p:** Stores the parity enable state to ensure logic remains consistent during a transaction

Receiver FIFO (First-In-First-Out) buffer

- **input rx_enbl**: The oversampling clock (used here to synchronize internal logic).
- **input areset**: The asynchronous reset to clear the buffer.
- **input flag**: This is the write enable. It comes from the receiver; when the receiver finishes a byte, it pulses this high to tell the FIFO to "save" the data.
- **input [7:0] dout**: The 8-bit data byte arriving from the receiver.
- **reg [7:0] mem [0:15];**: This creates the actual memory array.
    - It is a 2D array: 16 rows deep, and each row is 8 bits (1 byte) wide.
    - This means this FIFO can hold up to 16 bytes of data at once
- **reg [3:0] rear_ptr**;: This is the Write Pointer
    - It is 4 bits wide (2^4 = 16), which allows it to point to any index in the mem from 0 to 15.
    - It tracks the next empty slot where the next incoming byte should be stored.
- **reg [4:0] count;**: This tracks the fill level of the FIFO.
    - It is 5 bits wide so it can represent values from 0 up to 16.
    - Why 5 bits? To distinguish between the FIFO being "Empty" (0) and "Full" (16). If it were only 4 bits, both 0 and 16 would look like 0000
    - **integer i**:for clearing the memory block

TOPMODULE

- **module topmodule(...)**: Defines the start of the module.
- **input clk**: The master system clock (usually 50MHz or 100MHz)
- **input areset**: Asynchronous reset signal to initialize all internal registers to zero.
- **input strt_enbl**: A control signal to trigger the transmitter to begin sending data.
- **input [7:0] din**: The 8-bit data input that you want to transmit.
- **input write_en:** A signal that tells the transmit FIFO to "capture" the data on the din bus.
- **input p_enbl:** Parity enable. If high, the system likely calculates a parity bit for error checking.
- **output error:** High if the receiver detects a mismatch in data or parity.
- **wire tx_enbl, rx_enbl**: Clock pulses from the baud generator to keep everything at the right speed.

- **wire info:** This is the most important wire. It is the single-bit serial line where data travels bit-by-bit from the transmitter to the receiver.
- **wire flag:** A signal from the receiver indicating a full byte has been successfully reconstructed.
- **wire [7:0] dout**: The 8-bit parallel data reconstructed by the receiver
- **wire rd_enbl:** Handshaking signal; the transmitter uses this to ask the FIFO for more data.
- **wire [7:0] temp:** Holds the byte currently being transferred from the FIFO to the transmitter.
- **wire empty**: Status signal from the FIFO; if high, the transmitter knows there is no data left to send.
- **wire busy:** High when the transmitter is currently shifting bits and cannot accept new data.

```
fifo content              463670000
addr[           0]: data=96
addr[           1]: data=66
addr[           2]: data=43
addr[           3]: data=53
addr[           4]: data=63
addr[           5]: data=01
addr[           6]: data=12
addr[           7]: data=24
addr[           8]: data=33
addr[           9]: data=46
addr[          10]: data=77
addr[          11]: data=81
addr[          12]: data=13
addr[          13]: data=15
addr[          14]: data=67
addr[          15]: data=69
fifo content              479030000
addr[           0]: data=69
addr[           1]: data=96
addr[           2]: data=66
addr[           3]: data=43
addr[           4]: data=53
addr[           5]: data=63
addr[           6]: data=01
addr[           7]: data=12
addr[           8]: data=24
addr[           9]: data=33
addr[          10]: data=46
addr[          11]: data=77
addr[          12]: data=81
addr[          13]: data=13
addr[          14]: data=15
addr[          15]: data=67
```

- Reading contents of transmitter fifo memory block

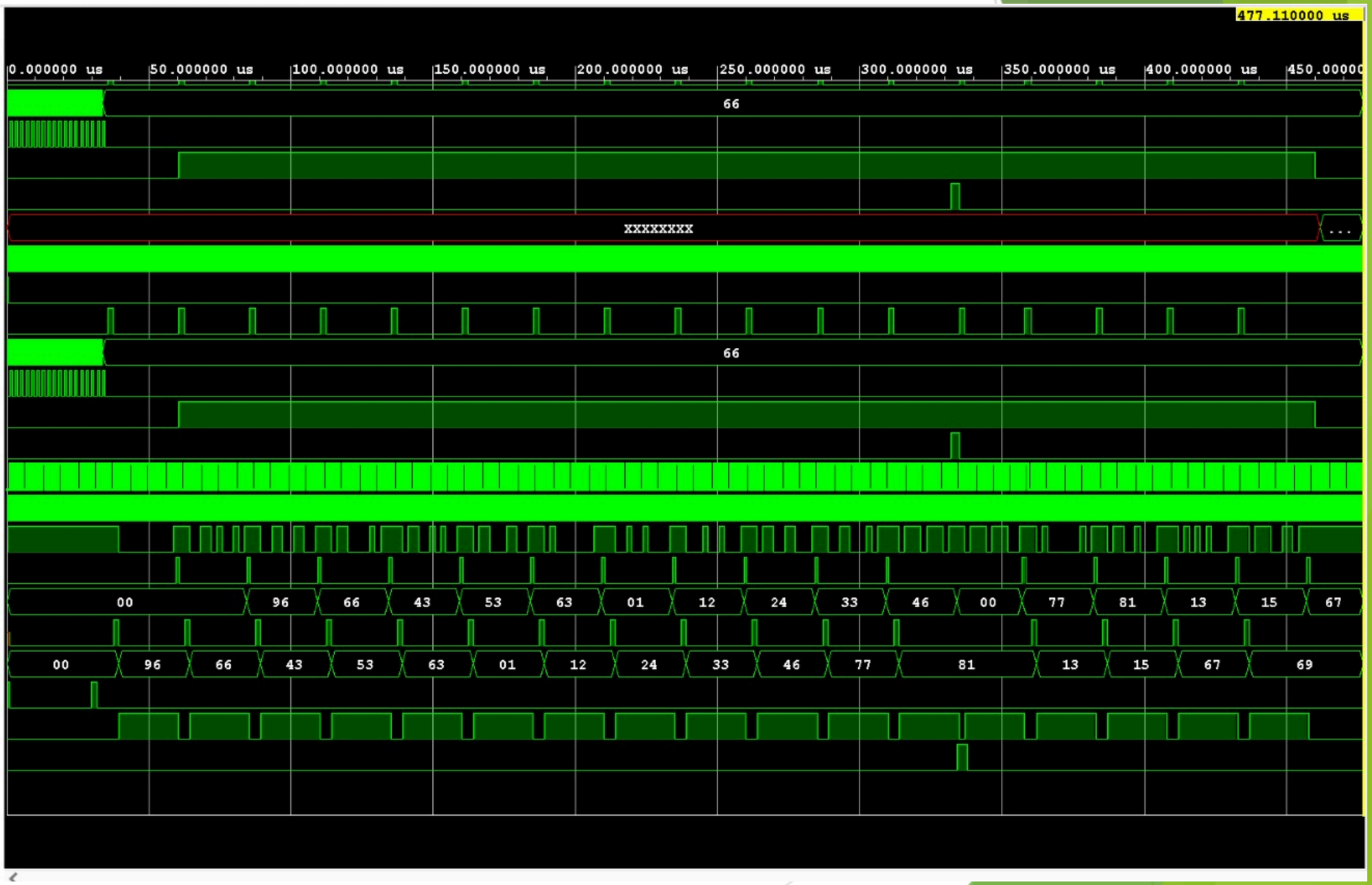
- reading contents of receiver fifo memory block

```
fifo content              461750000
addr[           0]: data=96
addr[           1]: data=66
addr[           2]: data=43
addr[           3]: data=53
addr[           4]: data=63
addr[           5]: data=01
addr[           6]: data=12
addr[           7]: data=24
addr[           8]: data=33
addr[           9]: data=46
addr[          10]: data=77
addr[          11]: data=81
addr[          12]: data=13
addr[          13]: data=15
addr[          14]: data=67
addr[          15]: data=69
fifo content              477110000
addr[           0]: data=00
addr[           1]: data=96
addr[           2]: data=66
addr[           3]: data=43
addr[           4]: data=53
addr[           5]: data=63
addr[           6]: data=01
addr[           7]: data=12
addr[           8]: data=24
addr[           9]: data=33
addr[          10]: data=46
addr[          11]: data=77
addr[          12]: data=81
addr[          13]: data=13
addr[          14]: data=15
addr[          15]: data=67
$finish called at time : 477110 ns
```

- with feedback high at 77 ---> error was made high

Your paragraph text

OUTPUT


477.110000 us
0.000000 us
50.000000 us
100.000000 us
150.000000 us
200.000000 us
250.000000 us
300.000000 us
350.000000 us
400.000000 us
450.00000
66
xxxxxxxx
...
66
00 96 66 43 53 63 01 12 24 33 46 00 77 81 13 15 67
00 96 66 43 53 63 01 12 24 33 46 77 81 13 15 67 69