UART Echo System with FIFO Buffers

Technical Specification and Implementation Guide

VHDL Development Project

July 31, 2025

Contents

1 Executive Summary

The UART Echo System is a complete FPGA-based communication interface designed for the Basys3 development board. The system implements a full-duplex UART communication protocol with integrated FIFO buffering, seven-segment display visualization, and comprehensive status monitoring. Operating at 115200 baud rate with 8N1 configuration, the system provides reliable serial communication with real-time character echo and display capabilities.

2 System Overview

2.1 Key Features

- Full-duplex UART communication at 115200 baud (8N1)
- Integrated RX/TX FIFO buffers (32 words each)
- Real-time character display on seven-segment display
- 16 status LEDs for system monitoring
- ASCII character support with echo functionality
- Overflow protection and error handling
- Configurable FIFO depths and timing parameters

2.2 Target Platform

- **Primary:** Digilent Basys3 FPGA Development Board
- **FPGA:** Xilinx Artix-7 XC7A35T-1CPG236C
- **Tools:** Xilinx Vivado 2025.1 or later
- **Clock:** 100MHz system clock

3 System Architecture

3.1 Block Diagram

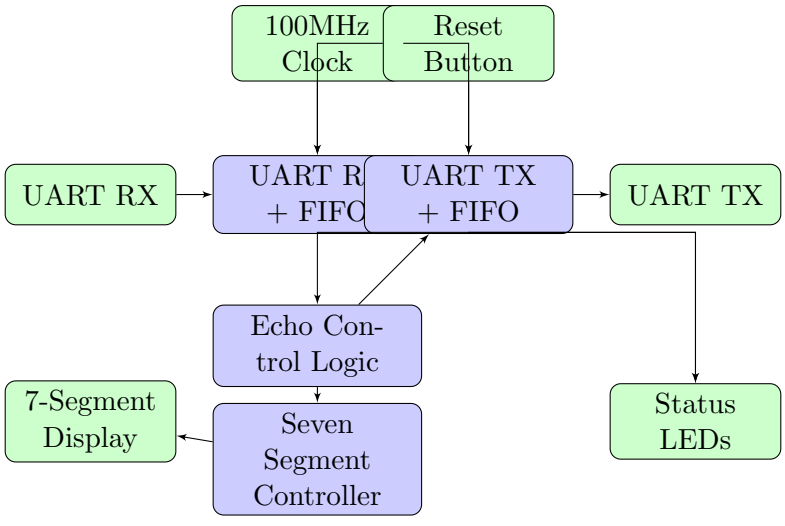

Figure 1: UART Echo System Block Diagram

3.2 Module Hierarchy

Module	File	Description
top	top.vhd	Top-level integration module
uart_rx	uart_rx.vhd	UART receiver with FIFO
uart_tx	uart_tx.vhd	UART transmitter with FIFO
fifo	fifo.vhd	Generic FIFO buffer
seven_segment_controller	seven_segment_controller.vhd	Display controller

Table 1: Module Hierarchy

4 Detailed Module Specifications

4.1 Top Module (top.vhd)

4.1.1 Entity Declaration

```
entity top is
    port (
        clk         : in  std_logic;  -- 100MHz system clock
        btnC        : in  std_logic;  -- Center button for reset
        RsRx        : in  std_logic;  -- UART receive
        RsTx        : out std_logic;  -- UART transmit
        seg         : out std_logic_vector(6 downto 0);
        an          : out std_logic_vector(3 downto 0);
        led         : out std_logic_vector(15 downto 0)
    );
end top;
```

Listing 1: Top Module Entity

4.1.2 Key Parameters

- **CLKS_PER_BIT:** 868 (100MHz / 115200 baud)
- **RX_FIFO_DEPTH:** 32 words
- **TX_FIFO_DEPTH:** 32 words
- **DATA_WIDTH:** 8 bits

4.2 UART Receiver (uart_rx.vhd)

4.2.1 State Machine

The UART RX implements a 5-state finite state machine:

1. **s_Idle:** Wait for start bit detection
2. **s_RX_Start_Bit:** Validate start bit at middle sampling
3. **s_RX_Data_Bits:** Receive 8 data bits (LSB first)
4. **s_RX_Stop_Bit:** Validate stop bit
5. **s_Cleanup:** Complete reception cycle

4.2.2 Timing Specifications

Parameter	Value
Baud Rate	115200 bps
Bit Period	8.68 s
Sampling Point	Middle of bit period
Clock Cycles per Bit	868
Start Bit Detection	Falling edge

Table 2: UART RX Timing Parameters

4.3 UART Transmitter (uart_tx.vhd)

4.3.1 State Machine

The UART TX implements a 5-state finite state machine:

1. **IDLE:** Wait for data from FIFO
2. **TX_START_BIT:** Transmit start bit (0)
3. **TX_DATA_BITS:** Transmit 8 data bits (LSB first)
4. **TX_STOP_BIT:** Transmit stop bit (1)
5. **CLEANUP:** Complete transmission cycle

4.4 FIFO Buffer (fifo.vhd)

4.4.1 Architecture

- **Memory Type:** Dual-port RAM using FPGA block RAM
- **Addressing:** Circular buffer with read/write pointers
- **Depth:** Configurable (default: 32 words)
- **Width:** Configurable (default: 8 bits)
- **Status Flags:** Full, Empty, Count

4.4.2 Control Logic

```
-- Control signals combination
signal control_sig : std_logic_vector(1 downto 0);
control_sig <= wr_valid & rd_valid;

case control_sig is
    when "10" => fifo_count <= fifo_count + 1;  -- Write only
    when "01" => fifo_count <= fifo_count - 1;  -- Read only
    when "11" => fifo_count <= fifo_count;      -- Simultaneous
    when others => fifo_count <= fifo_count;    -- No operation
end case;
```

Listing 2: FIFO Control Logic

5 Seven-Segment Display Controller

5.1 Character Mapping

The seven-segment controller supports full ASCII character set with the following mappings:

Range	Characters	Display	Pattern
0x30-0x39	0-9	Digits	Standard 7-segment
0x41-0x5A	A-Z	Letters	Custom patterns
0x61-0x7A	a-z	Letters	Custom patterns
0x20	Space	Blank	All segments off
Others	Special	Dash	Middle segment only

Table 3: Character Mapping for Seven-Segment Display

5.2 Multiplexing

- **Refresh Rate:** 380Hz (100MHz / 2^18)**Digit Update:**$Every 2.6ms$

- **Display Buffer:** 4 characters (32 bits)
- **Shift Direction:** Left shift for new characters

6 Status LED Assignments

LED	Function	Description
LED[0]	RX Data Available	RX FIFO not empty
LED[1]	TX Data Queued	TX FIFO not empty
LED[2]	RX FIFO Full	RX buffer full
LED[3]	TX FIFO Full	TX buffer full
LED[4]	TX Active	Currently transmitting
LED[5]	RX Error	Overflow error
LED[6]	Character Received	New character indicator
LED[7]	Heartbeat	System alive indicator
LED[12:8]	RX Count	RX FIFO word count
LED[15:13]	TX Count	TX FIFO word count (3 MSBs)

Table 4: Status LED Functions

7 Implementation Details

7.1 Clock Domain Management

- **System Clock:** 100MHz for all digital logic
- **UART Timing:** Derived from system clock using counters
- **Reset:** Asynchronous assert, synchronous deassert
- **Clock Enable:** Used for UART bit timing

7.2 Reset Strategy

```
reset_sync_process : process(clk)
begin
    if rising_edge(clk) then
        reset_sync <= reset_sync(1 downto 0) & (not btnC);
    end if;
end process;

reset_n <= reset_sync(2);
```

Listing 3: Reset Synchronizer

8 Timing Analysis

8.1 Critical Paths

1. FIFO read/write operations
2. UART state machine transitions
3. Seven-segment display multiplexing
4. Status LED updates

8.2 Timing Constraints

```
## Clock constraint
create_clock -add -name sys_clk_pin -period 10.00 [get_ports clk]

## UART timing constraints
set_max_delay -from [get_ports RsRx] -to [get_clocks sys_clk_pin] 30.0
set_max_delay -from [get_clocks sys_clk_pin] -to [get_ports RsTx] 30.0

## Seven segment timing
set_max_delay -from [get_clocks sys_clk_pin] -to [get_ports {seg[*]}]
   20.0
```

Listing 4: Key Timing Constraints (from master.xdc)

9 Resource Utilization

9.1 Estimated Resource Usage

Resource	Usage	Percentage (XC7A35T)
LUTs	450-600	2.7-3.6%
Flip-Flops	300-400	0.9-1.2%
Block RAM	1-2	2.5-5.0%
DSP Slices	0	0%
IO Pins	27	13.5%

Table 5: Resource Utilization Estimates

10 Testing and Verification

10.1 Testbench Strategy

- Unit tests for each module
- Integration testing at top level
- UART protocol compliance verification
- FIFO boundary condition testing
- Display controller validation

10.2 Hardware Testing

1. Connect Basys3 to PC via USB-UART
2. Open terminal emulator (115200, 8N1)
3. Type characters and verify echo
4. Monitor seven-segment display updates
5. Observe status LED behavior
6. Test overflow conditions

11 Synthesis and Implementation

11.1 Synthesis Settings

- **Strategy:** Vivado Synthesis Defaults
- **Optimization:** Balanced
- **Resource Sharing:** Enabled
- **FSM Encoding:** Auto

11.2 Implementation Flow

1. Synthesis
2. Optimization
3. Placement
4. Routing
5. Timing analysis
6. Bitstream generation

12 Conclusion

The UART Echo System demonstrates a complete FPGA-based communication interface with robust buffering and comprehensive monitoring capabilities. The modular design allows for easy customization and extension, making it suitable for educational purposes and as a foundation for more complex communication systems.

12.1 Future Enhancements

- Variable baud rate support
- Hardware flow control (RTS/CTS)
- Larger character buffer for seven-segment display
- Protocol analyzer functionality
- Multi-channel UART support