Faculty of Engineering - Ain Shams University

Digital Design – NTI

Project Report

Universal Asynchronous Receiver-Transmitter (UART) with APB Wrapper

9 Sep 2025

Submitted to:

Eng. Mohamed Tareq

Submitted by:

Mai Fakhry Mohamed

Contents

1. Introduction

The Universal Asynchronous Receiver-Transmitter (UART) is a widely used hardware communication protocol enabling serial data exchange between digital systems. It transmits data one bit at a time, using configurable parameters such as baud rate, parity, stop bits, and data word length.

This project implements:

- **UART Transmitter**: Responsible for serializing and sending data frames.
- **UART Receiver**: Deserializes incoming serial data and validates it.
- **APB UART Wrapper**: Provides a standard AMBA APB interface to control the UART.

The design is verified using testbenches and simulated to validate correct operation.

2. Design Analysis

The UART follows asynchronous communication principles:

- **Transmitter**: Adds start bit, data bits and stop bits.
- **Receiver**: Detects start bit, samples incoming bits, checks stop conditions.
- **APB Wrapper**: Bridges APB bus transactions (write/read) to UART control and data registers.

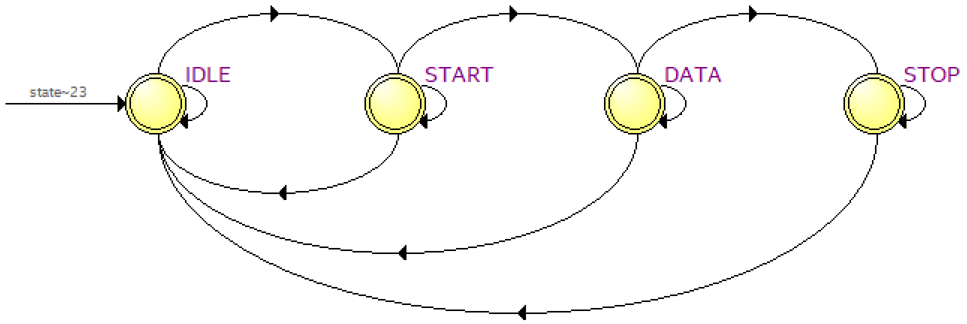
Key Design Parameters:

- Baud rate: Configurable via divider.
- Data width: 8 bits.
- Stop bits: 1.

3. State Diagrams

UART Transmitter FSM

- **IDLE → START → DATA → STOP → IDLE**



UART Receiver FSM

- **IDLE → START DETECT → DATA RECEIVE → STOP CHECK → IDLE**

4. Design Decisions

- Implemented FSM-based control for both TX and RX.
- Chose fixed 8N1 frame format for simplicity.
- APB wrapper provides register-based configuration and status reporting.
- Testbenches are used to automate verification.

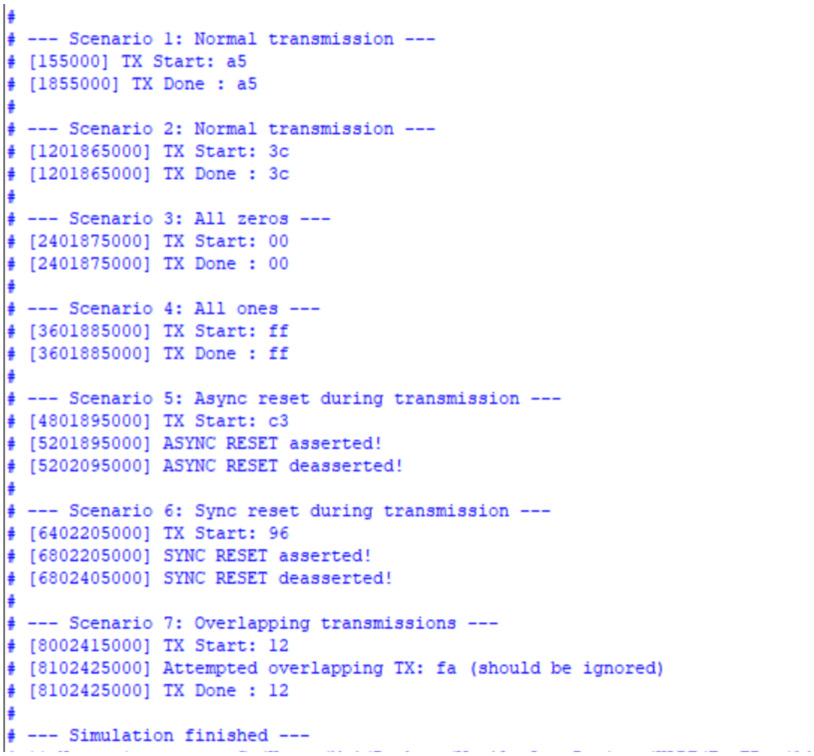
5. Verification Strategy

- **Unit-level verification**: Each module (TX, RX, Wrapper) tested independently.
- **Testbenches**: Compare expected vs. actual outputs automatically.
- **Integration testing**: Verified APB transactions trigger correct UART operations.
- **Corner cases**: Tested back-to-back transmission, framing errors, and buffer overflows.

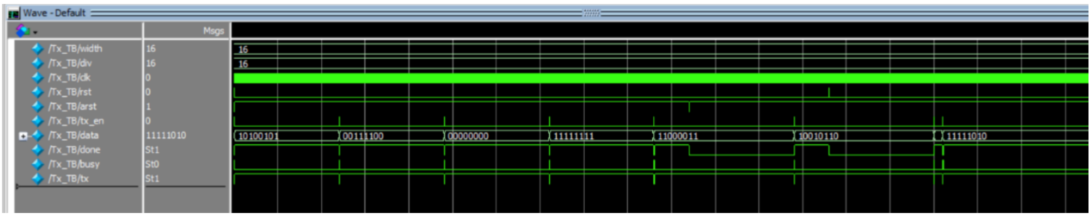
6. Simulation Results

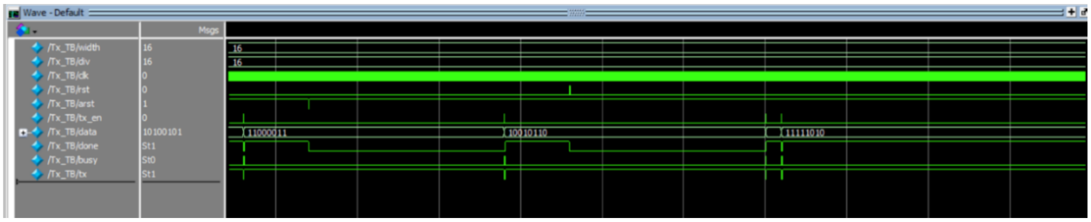
6.1 UART Transmitter (TX)

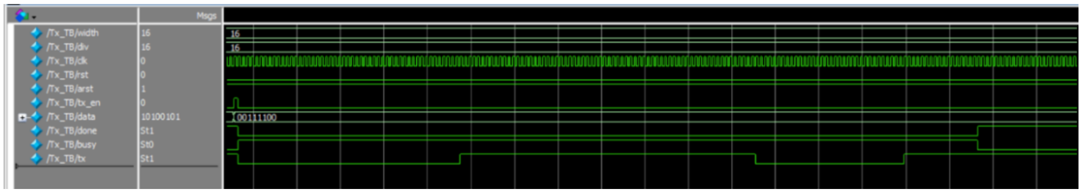
- Verified correct generation of start, data, and stop bits.
- Observed **serial waveform** on TX line in ModelSim



```
#
# --- Scenario 1: Normal transmission ---
# [155000] TX Start: a5
# [1855000] TX Done : a5
#
# --- Scenario 2: Normal transmission ---
# [1201865000] TX Start: 3c
# [1201865000] TX Done : 3c
#
# --- Scenario 3: All zeros ---
# [2401875000] TX Start: 00
# [2401875000] TX Done : 00
#
# --- Scenario 4: All ones ---
# [3601885000] TX Start: ff
# [3601885000] TX Done : ff
#
# --- Scenario 5: Async reset during transmission ---
# [4801895000] TX Start: c3
# [5201895000] ASYNC RESET asserted!
# [5202095000] ASYNC RESET deasserted!
#
# --- Scenario 6: Sync reset during transmission ---
# [6402205000] TX Start: 96
# [6802205000] SYNC RESET asserted!
# [6802405000] SYNC RESET deasserted!
#
# --- Scenario 7: Overlapping transmissions ---
# [8002415000] TX Start: 12
# [8102425000] Attempted overlapping TX: fa (should be ignored)
# [8102425000] TX Done : 12
#
# --- Simulation finished ---
```

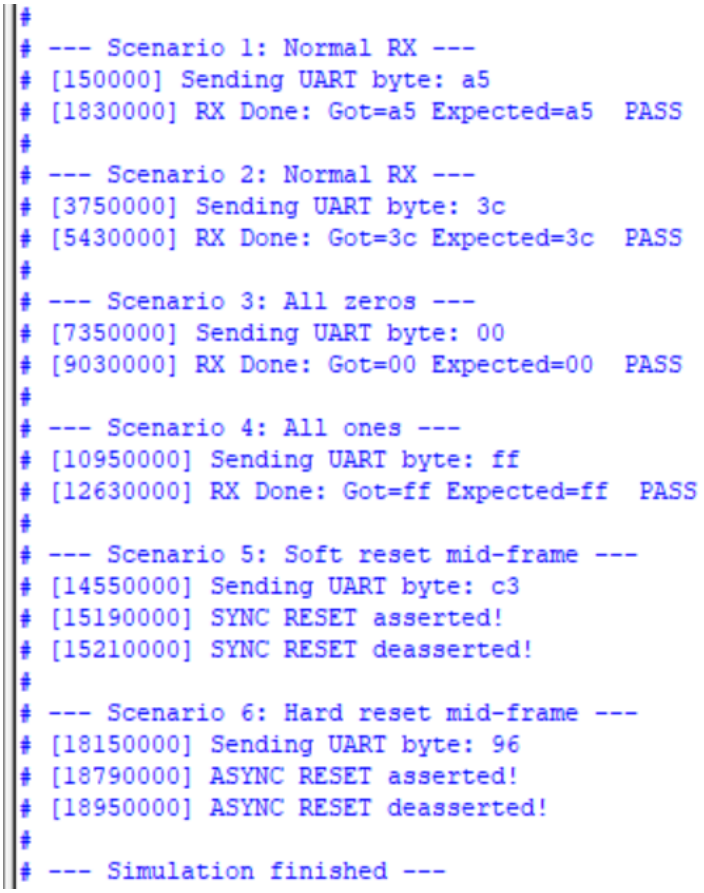




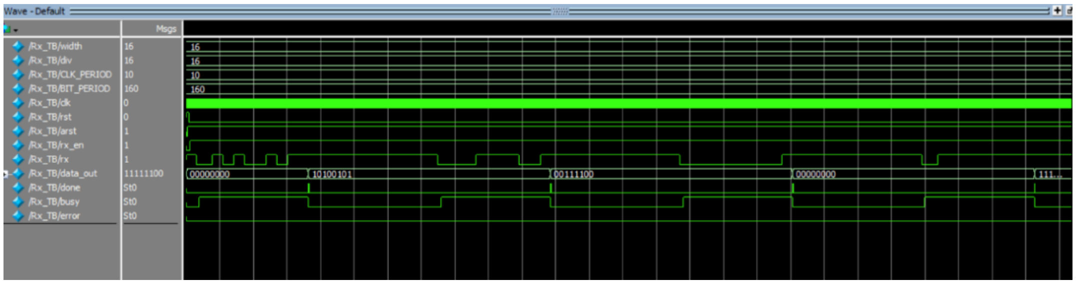


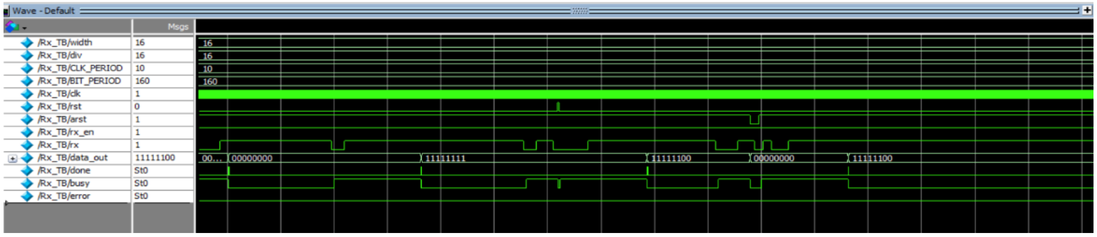
6.2 UART Receiver (RX)

- Correctly detected start bit, sampled 8 data bits, and validated stop bit.
- Compared received data against transmitted data



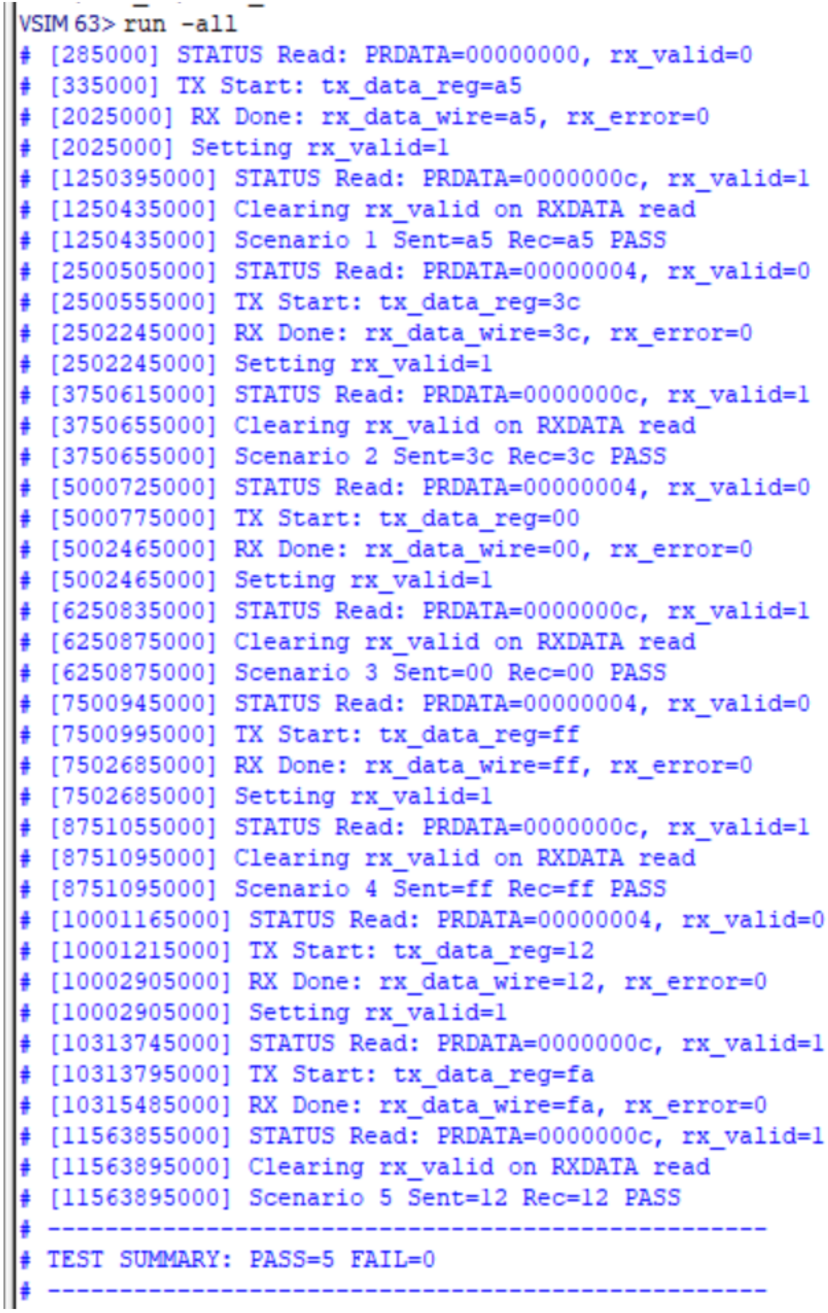
```
#
# --- Scenario 1: Normal RX ---
# [150000] Sending UART byte: a5
# [1830000] RX Done: Got=a5 Expected=a5  PASS
#
# --- Scenario 2: Normal RX ---
# [3750000] Sending UART byte: 3c
# [5430000] RX Done: Got=3c Expected=3c  PASS
#
# --- Scenario 3: All zeros ---
# [7350000] Sending UART byte: 00
# [9030000] RX Done: Got=00 Expected=00  PASS
#
# --- Scenario 4: All ones ---
# [10950000] Sending UART byte: ff
# [12630000] RX Done: Got=ff Expected=ff  PASS
#
# --- Scenario 5: Soft reset mid-frame ---
# [14550000] Sending UART byte: c3
# [15190000] SYNC RESET asserted!
# [15210000] SYNC RESET deasserted!
#
# --- Scenario 6: Hard reset mid-frame ---
# [18150000] Sending UART byte: 96
# [18790000] ASYNC RESET asserted!
# [18950000] ASYNC RESET deasserted!
#
# --- Simulation finished ---
```



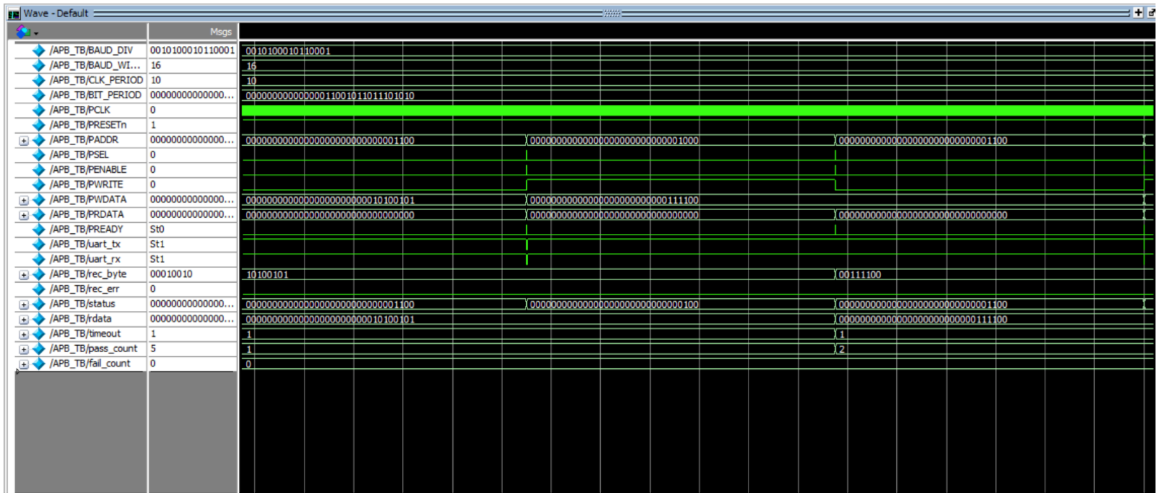


6.3 APB Wrapper

- Verified APB write/read transactions triggered UART operations.

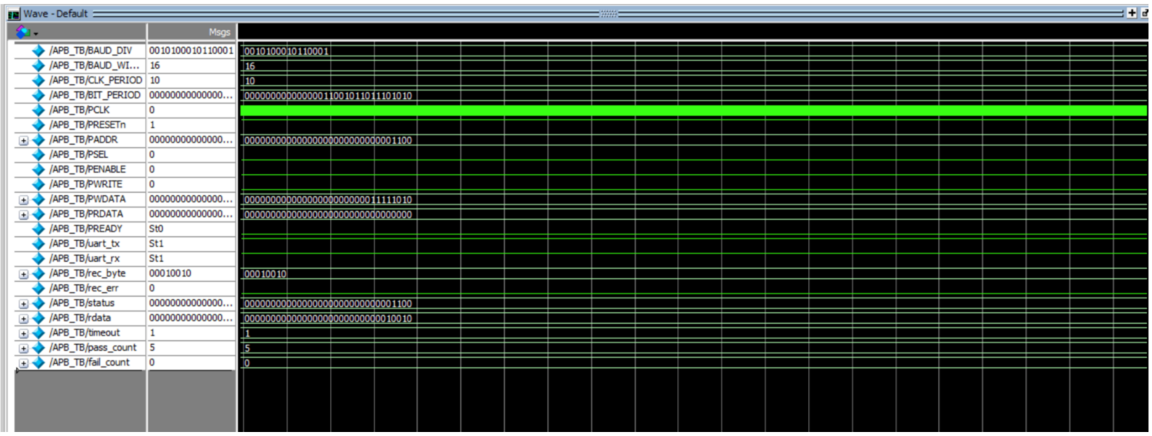


```
VSIM 63> run -all
# [285000] STATUS Read: PRDATA=00000000, rx_valid=0
# [335000] TX Start: tx_data_reg=a5
# [2025000] RX Done: rx_data_wire=a5, rx_error=0
# [2025000] Setting rx_valid=1
# [1250395000] STATUS Read: PRDATA=0000000c, rx_valid=1
# [1250435000] Clearing rx_valid on RXDATA read
# [1250435000] Scenario 1 Sent=a5 Rec=a5 PASS
# [2500505000] STATUS Read: PRDATA=00000004, rx_valid=0
# [2500555000] TX Start: tx_data_reg=3c
# [2502245000] RX Done: rx_data_wire=3c, rx_error=0
# [2502245000] Setting rx_valid=1
# [3750615000] STATUS Read: PRDATA=0000000c, rx_valid=1
# [3750655000] Clearing rx_valid on RXDATA read
# [3750655000] Scenario 2 Sent=3c Rec=3c PASS
# [5000725000] STATUS Read: PRDATA=00000004, rx_valid=0
# [5000775000] TX Start: tx_data_reg=00
# [5002465000] RX Done: rx_data_wire=00, rx_error=0
# [5002465000] Setting rx_valid=1
# [6250835000] STATUS Read: PRDATA=0000000c, rx_valid=1
# [6250875000] Clearing rx_valid on RXDATA read
# [6250875000] Scenario 3 Sent=00 Rec=00 PASS
# [7500945000] STATUS Read: PRDATA=00000004, rx_valid=0
# [7500995000] TX Start: tx_data_reg=ff
# [7502685000] RX Done: rx_data_wire=ff, rx_error=0
# [7502685000] Setting rx_valid=1
# [8751055000] STATUS Read: PRDATA=0000000c, rx_valid=1
# [8751095000] Clearing rx_valid on RXDATA read
# [8751095000] Scenario 4 Sent=ff Rec=ff PASS
# [10001165000] STATUS Read: PRDATA=00000004, rx_valid=0
# [10001215000] TX Start: tx_data_reg=12
# [10002905000] RX Done: rx_data_wire=12, rx_error=0
# [10002905000] Setting rx_valid=1
# [10313745000] STATUS Read: PRDATA=0000000c, rx_valid=1
# [10313795000] TX Start: tx_data_reg=fa
# [10315485000] RX Done: rx_data_wire=fa, rx_error=0
# [11563855000] STATUS Read: PRDATA=0000000c, rx_valid=1
# [11563895000] Clearing rx_valid on RXDATA read
# [11563895000] Scenario 5 Sent=12 Rec=12 PASS
# -------------------------------------------------
# TEST SUMMARY: PASS=5 FAIL=0
# -------------------------------------------------
```



Wave - Default

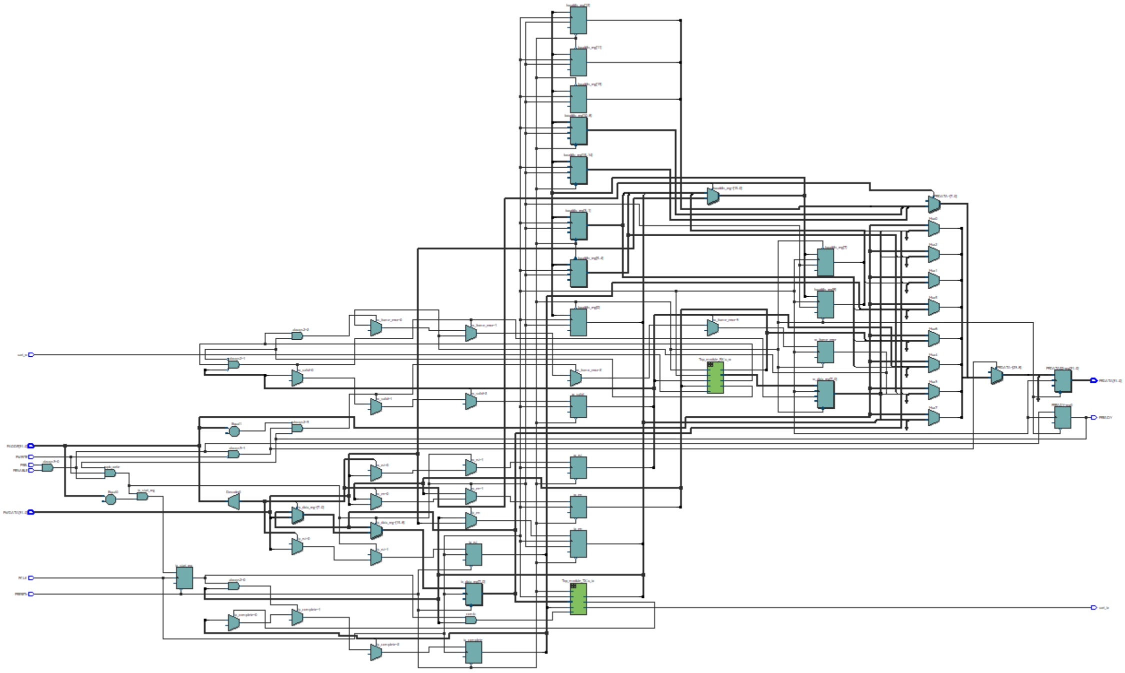
	Msgs
/APB_TB/BAUD_DIV	0010100010110001
/APB_TB/BAUD_WI...	16
/APB_TB/CLK_PERIOD	10
/APB_TB/BIT_PERIOD	00000000000000...
/APB_TB/PCLK	0
/APB_TB/PRESETn	1
/APB_TB/PADDR	00000000000000...
/APB_TB/PSEL	0
/APB_TB/PENABLE	0
/APB_TB/PWRITE	0
/APB_TB/PWDATA	00000000000000...
/APB_TB/PRDATA	00000000000000...
/APB_TB/PREADY	St0
/APB_TB/uart_tx	St1
/APB_TB/uart_rx	St1
/APB_TB/rec_byte	00010010
/APB_TB/rec_err	0
/APB_TB/status	00000000000000...
/APB_TB/rdata	00000000000000...
/APB_TB/timeout	1
/APB_TB/pass_count	5
/APB_TB/fail_count	0

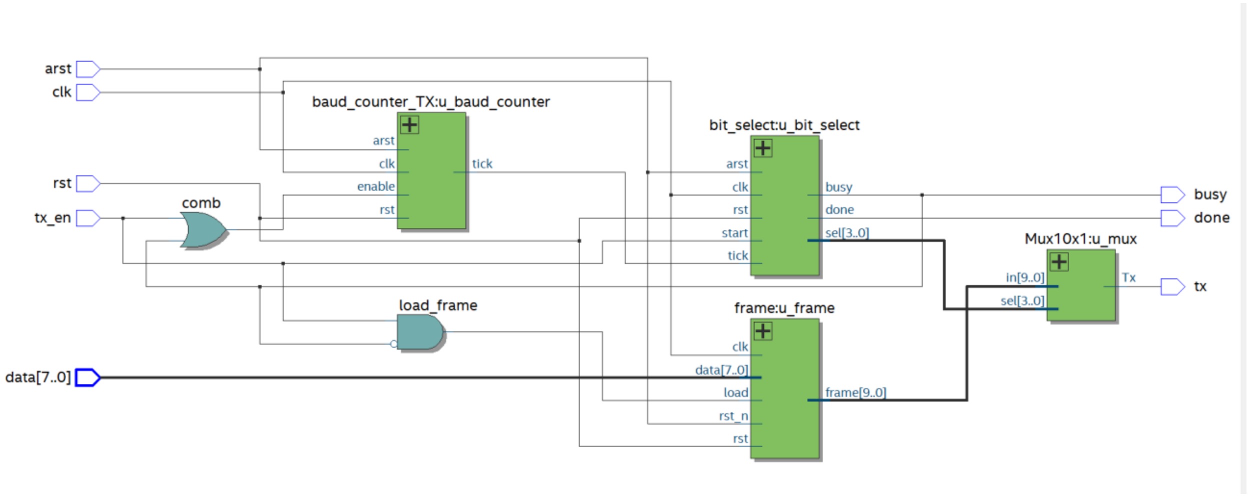


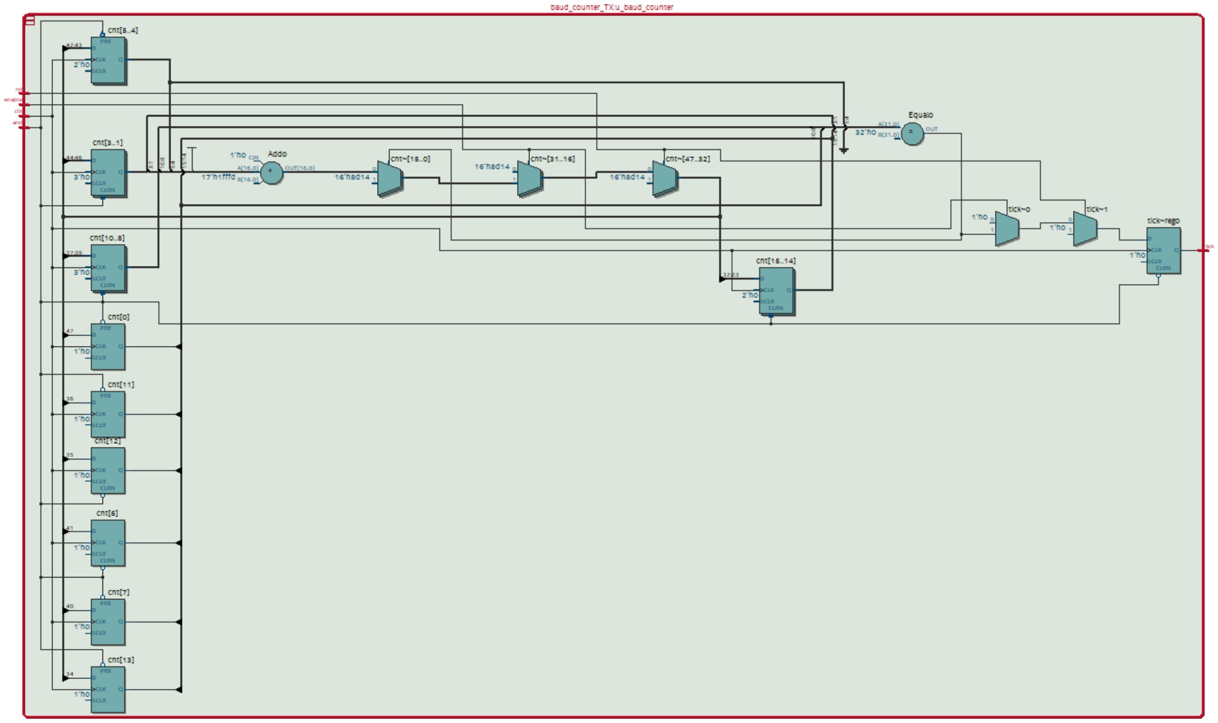
Wave - Default

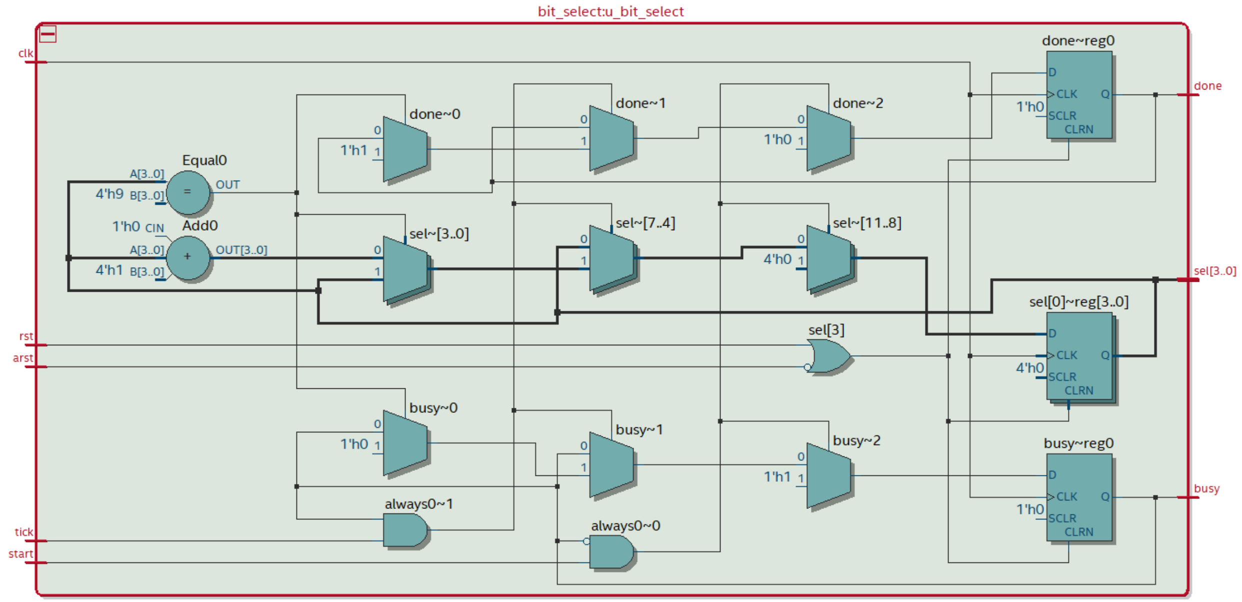
	Msgs
/APB_TB/BAUD_DIV	0010100010110001
/APB_TB/BAUD_WI...	16
/APB_TB/CLK_PERIOD	10
/APB_TB/BIT_PERIOD	00000000000000...
/APB_TB/PCLK	0
/APB_TB/PRESETn	1
/APB_TB/PADDR	00000000000000...
/APB_TB/PSEL	0
/APB_TB/PENABLE	0
/APB_TB/PWRITE	0
/APB_TB/PWDATA	00000000000000...
/APB_TB/PRDATA	00000000000000...
/APB_TB/PREADY	St0
/APB_TB/uart_tx	St1
/APB_TB/uart_rx	St1
/APB_TB/rec_byte	00010010
/APB_TB/rec_err	0
/APB_TB/status	00000000000000...
/APB_TB/rdata	00000000000000...
/APB_TB/timeout	1
/APB_TB/pass_count	5
/APB_TB/fail_count	0

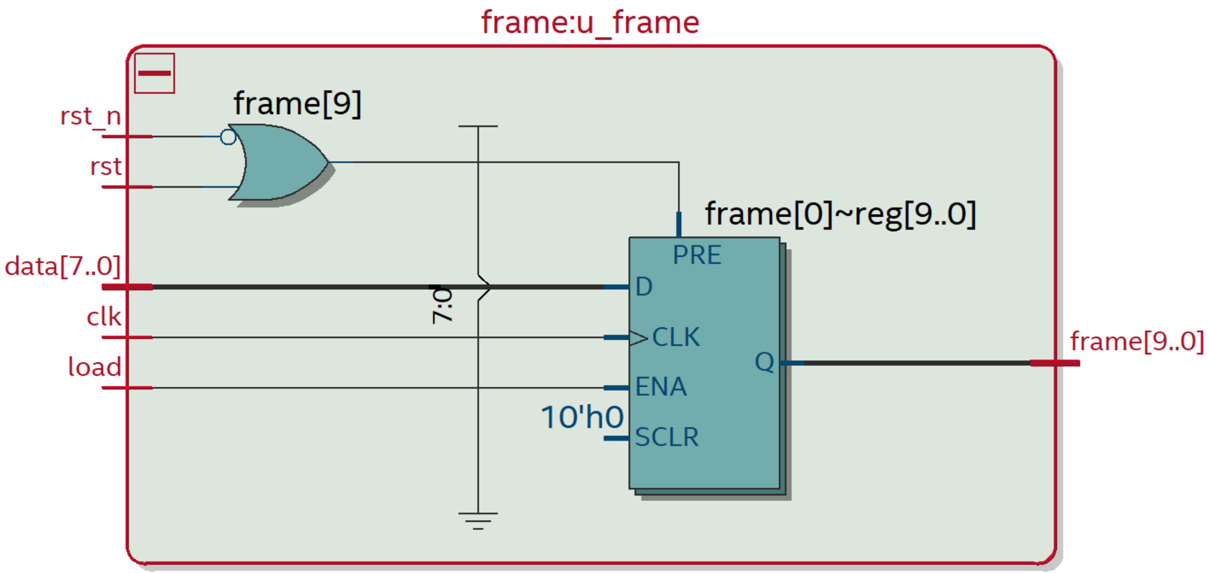
6.4 Quartus Prime Results

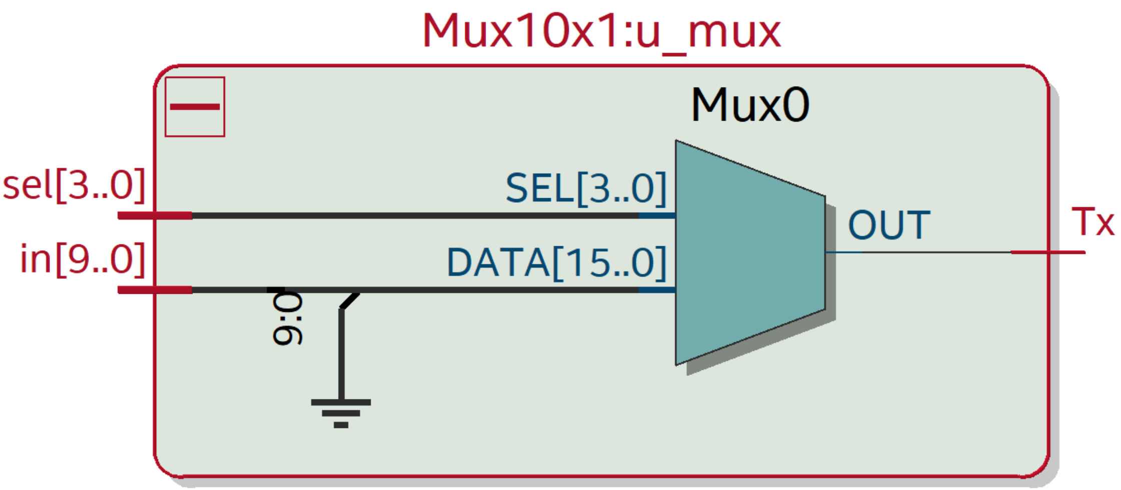


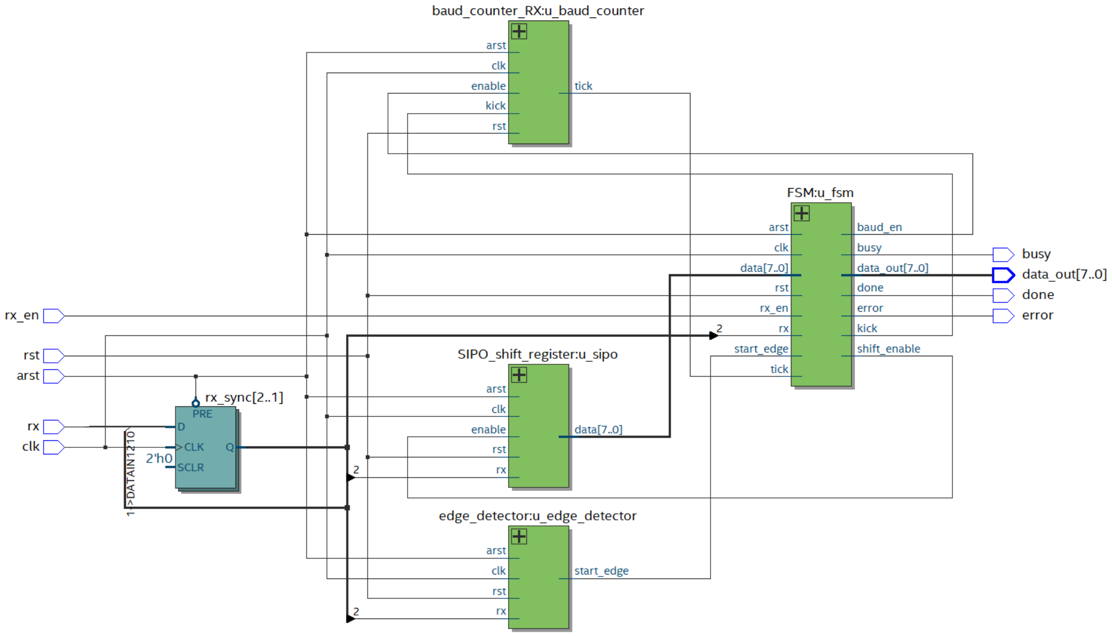

arst
clk
rst
tx_en
data[7..0]
comb
load_frame
baud_counter_TX:u_baud_counter
arst
clk
enable
rst
tick
bit_select:u_bit_select
arst
clk
rst
start
tick
busy
done
sel[3..0]
frame:u_frame
clk
data[7..0]
load
rst_n
rst
frame[9..0]
Mux10x1:u_mux
in[9..0]
sel[3..0]
Tx
busy
done
tx

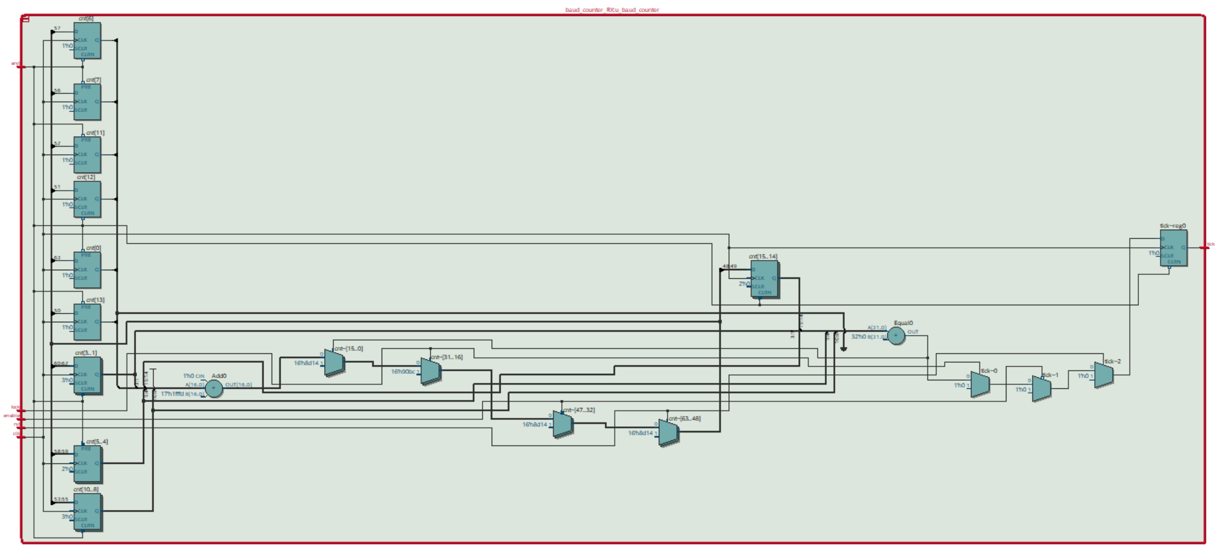

baud_counter_TX:u_baud_counter
cnt[9..4]
cnt[3..1]
cnt[10..8]
cnt[0]
cnt[11]
cnt[12]
cnt[6]
cnt[7]
cnt[13]
Add0
17'h1ffff
1'h0
cnt~[15..0]
cnt~[31..16]
cnt~[47..32]
16'h8d14
Equal0
32'h0
cnt[15..14]
tick~0
tick~1
tick~reg0

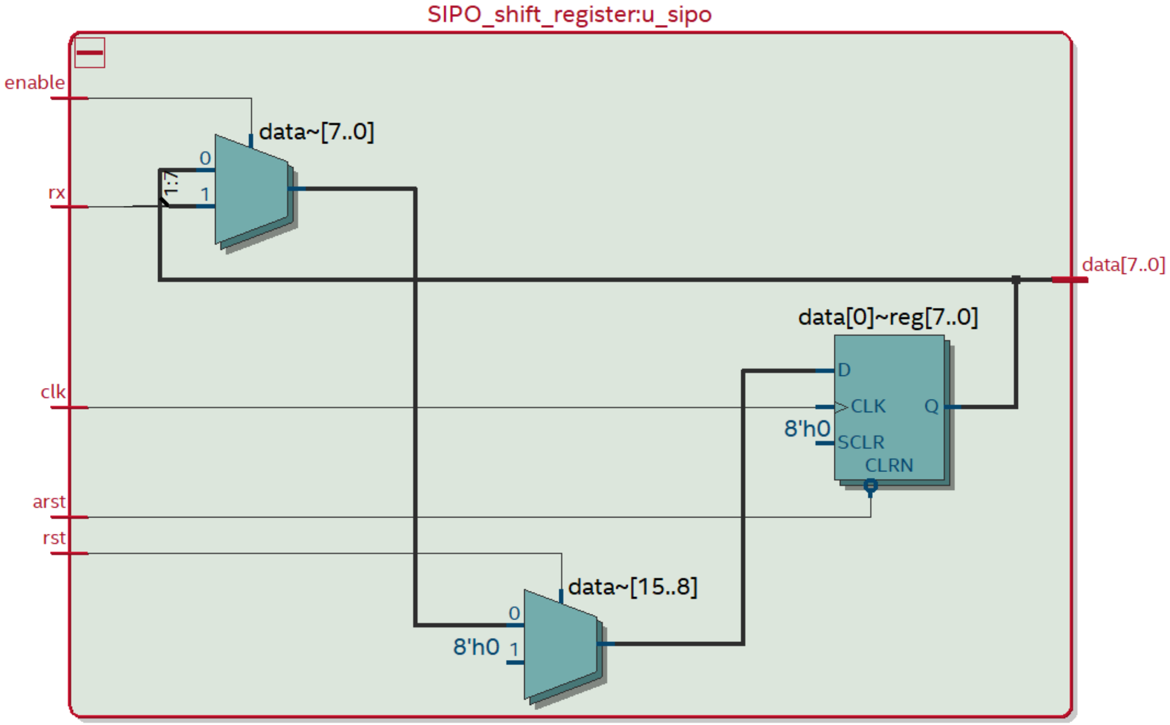

bit_select:u_bit_select
clk
rst
arst
tick
start
Equal0
A[3..0]
4'h9
B[3..0]
OUT
1'h0
CIN
Add0
A[3..0]
4'h1
B[3..0]
OUT[3..0]
done~0
1'h1
done~1
done~2
1'h0
done~reg0
D
CLK
Q
1'h0
SCLR
CLRN
done
sel~[3..0]
sel~[7..4]
sel~[11..8]
4'h0
sel[3]
sel[0]~reg[3..0]
D
CLK
Q
4'h0
SCLR
CLRN
sel[3..0]
busy~0
1'h0
busy~1
busy~2
1'h1
busy~reg0
D
CLK
Q
1'h0
SCLR
CLRN
busy
always0~1
always0~0

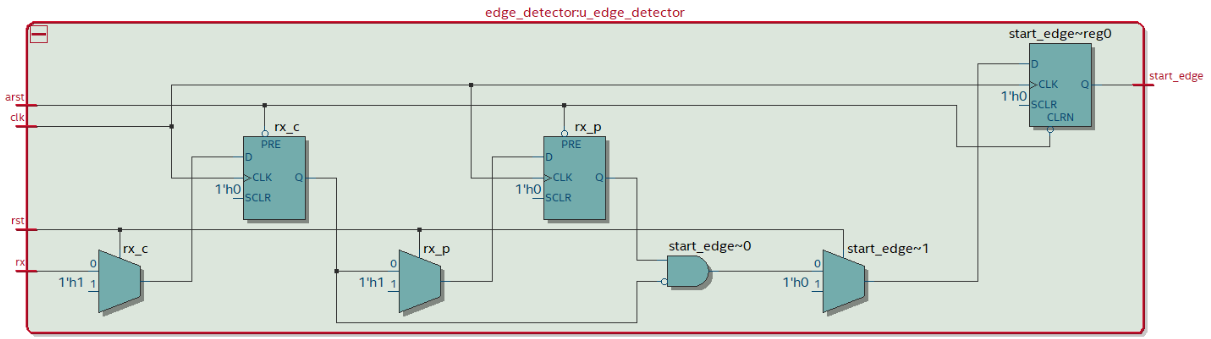

frame:u_frame
rst_n
rst
data[7..0]
clk
load
frame[9]
7:0
frame[0]~reg[9..0]
PRE
D
CLK
ENA
SCLR
Q
10'h0
frame[9..0]

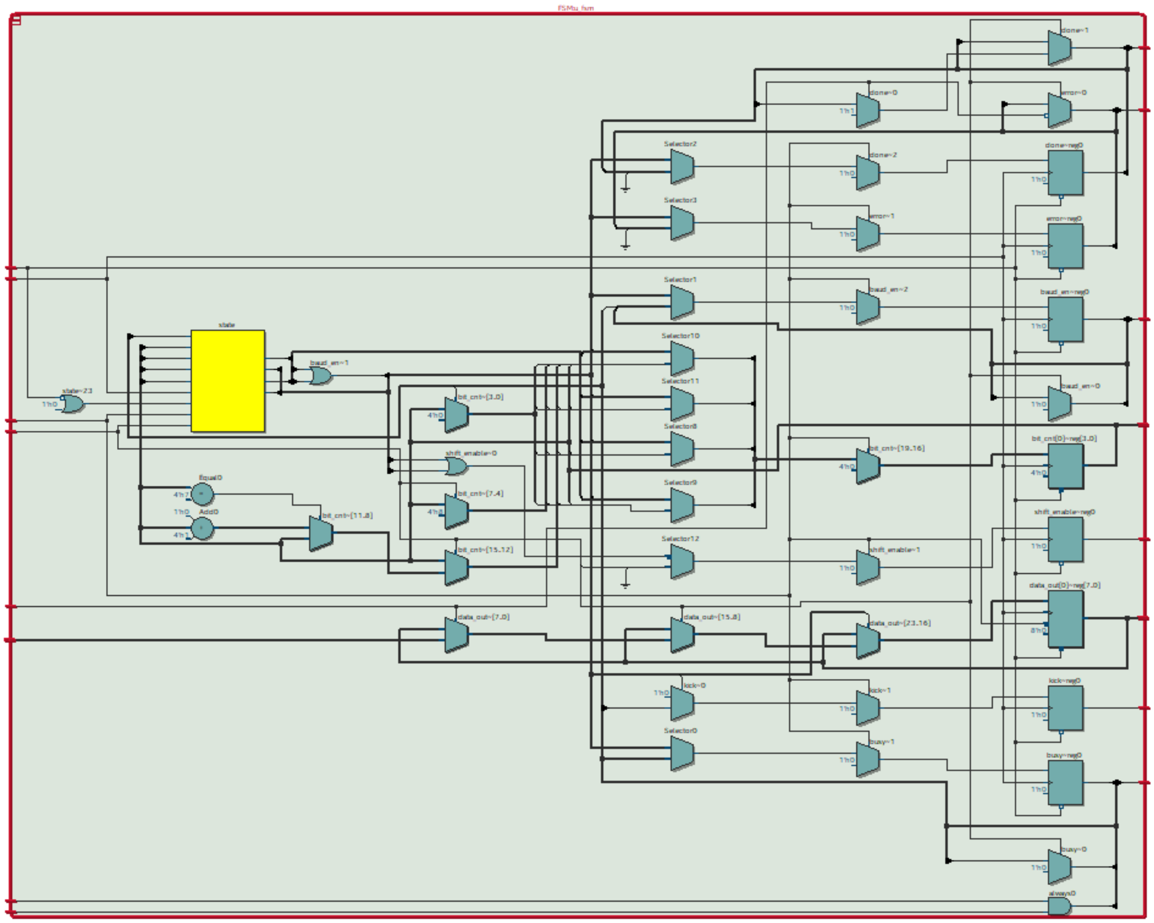

Mux10x1:u_mux
Mux0
sel[3..0]
SEL[3..0]
in[9..0]
DATA[15..0]
9:0
OUT
Tx


baud_counter_RX:u_baud_counter
arst
clk
enable
kick
rst
tick
FSM:u_fsm
arst
clk
data[7..0]
rst
rx_en
rx
start_edge
tick
baud_en
busy
data_out[7..0]
done
error
kick
shift_enable
busy
data_out[7..0]
done
error
rx_en
rst
arst
rx
clk
rx_sync[2..1]
PRE
D
CLK
Q
SCLR
2'h0
1'>DATAIN1210
2
SIPO_shift_register:u_sipo
arst
clk
enable
rst
rx
data[7..0]
edge_detector:u_edge_detector
arst
clk
rst
rx
start_edge


baud_counter_RX:u_baud_counter
cnt[6]
cnt[7]
cnt[11]
cnt[12]
cnt[0]
cnt[13]
cnt[3..1]
Add0
cnt~[15..0]
cnt~[31..16]
cnt~[47..32]
cnt~[63..48]
cnt[5..4]
cnt[10..8]
cnt[15..14]
Equal0
tick~0
tick~1
tick~2
tick~reg0


SIPO_shift_register:u_sipo
enable
rx
clk
arst
rst
data~[7..0]
0
1
1:7
data[7..0]
data[0]~reg[7..0]
D
CLK
Q
8'h0
SCLR
CLRN
data~[15..8]
0
8'h0
1


edge_detector:u_edge_detector
start_edge~reg0
D
CLK
Q
SCLR
CLRN
1'h0
start_edge
arst
clk
rst
rx
rx_c
PRE
D
CLK
Q
SCLR
1'h0
rx_p
PRE
D
CLK
Q
SCLR
1'h0
rx_c
0
1'h1 1
rx_p
0
1'h1 1
start_edge~0
start_edge~1
0
1'h0 1


FSM:u_fsm
state
baud_en~1
state~23
1'h0
Equal0
Add0
4'h7
1'h0
4'h1
bit_cnt~[3..0]
shift_enable~0
bit_cnt~[7..4]
bit_cnt~[11..8]
bit_cnt~[15..12]
data_out~[7..0]
Selector2
Selector3
Selector1
Selector10
Selector11
Selector8
Selector9
Selector12
data_out~[15..8]
kick~0
Selector0
done~1
done~0
error~0
done~2
done~reg0
error~1
error~reg0
baud_en~2
baud_en~reg0
baud_en~0
bit_cnt~[19..16]
bit_cnt[0]~reg[3..0]
shift_enable~reg0
shift_enable~1
data_out[0]~reg[7..0]
data_out~[23..16]
kick~reg0
kick~1
busy~1
busy~reg0
busy~0
always0

7. Conclusion

The ModelSim simulation results confirm that:

- **UART TX** outputs correct bitstream.
- **UART RX** successfully reconstructs transmitted data.
- **APB wrapper** correctly manages register access.

Quartus Prime synthesis further confirms the design’s **hardware feasibility**, showing reasonable resource usage and timing closure.