# CS 303 CELL PHONE PROJECT

In this project, our duty was to design a sequential circuit for a simple one-sided telephone conversation and implement it using Verilog HDL. Then, we were supposed to simulate and show its functional correctness using Vivado 2018.2 tool. Basically, the caller would initiate a telephone conversation with the callee and the caller would send characters to the callee. Our circuit also calculates the cost of the call and sends the cost value and the characters (sent from the caller to callee) as outputs.

## *Inputs*

• *rst*: sets our circuitry to its initial state.

• *startCall* (1-bit): is used by the caller and represents that the caller pressing a button to start a call.

• *answerCall* (1-bit): is used by the callee and represents that the callee pressing a button to answer an incoming call.

• *endCall* (1-bit): is used by the callee and represents that the callee pressing a button to end a call.

• *charSent* (8-bit): is used to define 8-bit printable ASCII character to be sent from the caller to the callee according to printable ASCII table.

• *sendChar* (1-bit): is used by the caller and represents that the caller pressing a button to send an ASCII character (set by charSent input) to the callee.

## *Outputs*

• *statusMsg* (64-bit): is used to display the status of telephone using 8 printable ASCII character. For example, statusMsg output should be "IDLE " in its initial state.

• *sentMsg* (64-bit): is used to display the last 8 printable ASCII characters sent by caller.

## *Operation Steps*

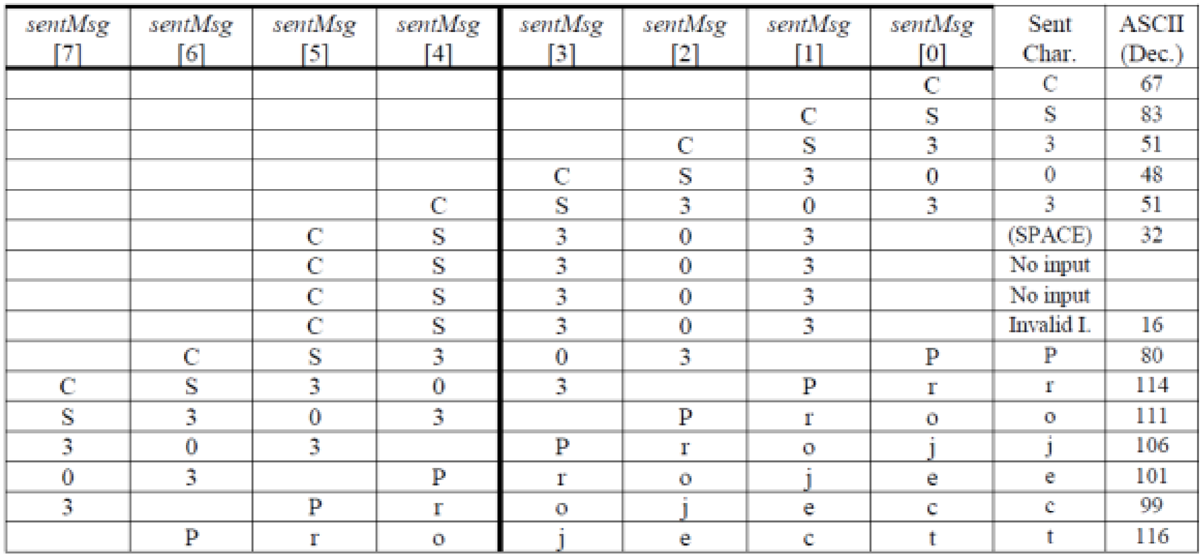
The circuitry starts in the IDLE state, in which it should output 'IDLE ' to statusMsg output (last 4 characters are space). We used rst input as an asynchronous reset input to put the sequential circuit into IDLE state. In IDLE state, the caller initiates a call by pressing startCall (by bringing startCall to 1 for one clock cycle) when the circuit is in IDLE state. Otherwise, the circuit stays in IDLE state. If the caller initiates a call by pressing startCall, the telephone starts ringing. When the telephone is ringing, the 8-bit ASCII output "RINGING " is at output statusMsg. When the telephone is ringing, there are three possibilities:

1. If the callee rejects the call by pressing endCall, our circuit outputs 'REJECTED' to statusMsg output for 10 clock cycles and then our circuit goes back to the IDLE state.

2. If the callee does not answer the call for 10 clock cycles, our circuit goes to the BUSY state. Our circuit outputs ‘BUSY’ to statusMsg output for 10 clock cycles. Then our circuit goes back to the IDLE state.

3. If the callee answers the call and the conversation starts, there are two possibilities during the conversations:

a. The caller sends character to the callee by setting charSent and pressing sendChar. If the caller does not press sendChar, telephone takes no input. During the caller sending character to the callee, statusMsg output is “CALLER “. Also, sentMsg output is the last 8 ASCII characters sent by the caller. For example, if caller sends characters “C”, “S”, “3”, “0”, “3”, then sentMsg should be “ CS303”. Another example is below:



| *sentMsg* [7] | *sentMsg* [6] | *sentMsg* [5] | *sentMsg* [4] | *sentMsg* [3] | *sentMsg* [2] | *sentMsg* [1] | *sentMsg* [0] | Sent Char. | ASCII (Dec.) |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | C | C | 67 |
| | | | | | | C | S | S | 83 |
| | | | | | C | S | 3 | 3 | 51 |
| | | | | C | S | 3 | 0 | 0 | 48 |
| | | | C | S | 3 | 0 | 3 | 3 | 51 |
| | | C | S | 3 | 0 | 3 | | (SPACE) | 32 |
| | | C | S | 3 | 0 | 3 | | No input | |
| | | C | S | 3 | 0 | 3 | | No input | |
| | | C | S | 3 | 0 | 3 | | Invalid I. | 16 |
| | C | S | 3 | 0 | 3 | | P | P | 80 |
| C | S | 3 | 0 | 3 | | P | r | r | 114 |
| S | 3 | 0 | 3 | | P | r | o | o | 111 |
| 3 | 0 | 3 | | P | r | o | j | j | 106 |
| 0 | 3 | | P | r | o | j | e | e | 101 |
| 3 | | P | r | o | j | e | c | c | 99 |
| | P | r | o | j | e | c | t | t | 116 |

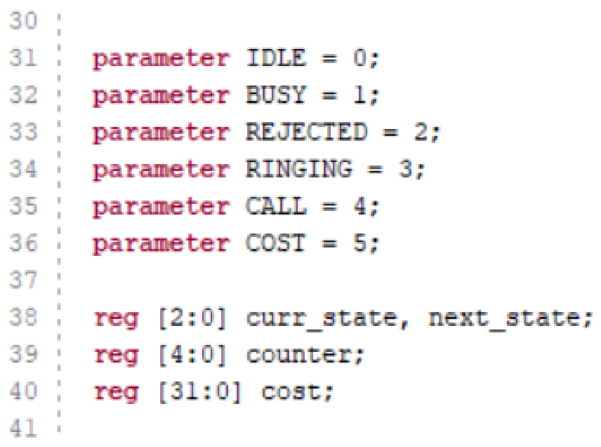
Only ASCII characters with decimal values between 32 and 126 are considered valid input characters. If the caller enters and sends an invalid character, our circuit should ignores that character (we did not include this input in cost calculation). If the caller sends the ASCI character with decimal value 127 (DEL) (we included this input in cost calculation), it will be caller’s last input character.

b. The callee ends the call at any time by pressing on endCall.

Each sent character costs 2 Krş except for integer digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) which cost 1 Krş. Our circuit calculates total cost of a call. When the call is ended, total cost of the conversation is sent as an output to sentMsg output in hexadecimal and statusMsg output is “COST ” for 5 clock cycles. For example, a call with a cost of 55 Krş is shown on sentMsg as “00000037” for 5 clock cycles. Then, our circuit goes to IDLE state.

## *Overview*

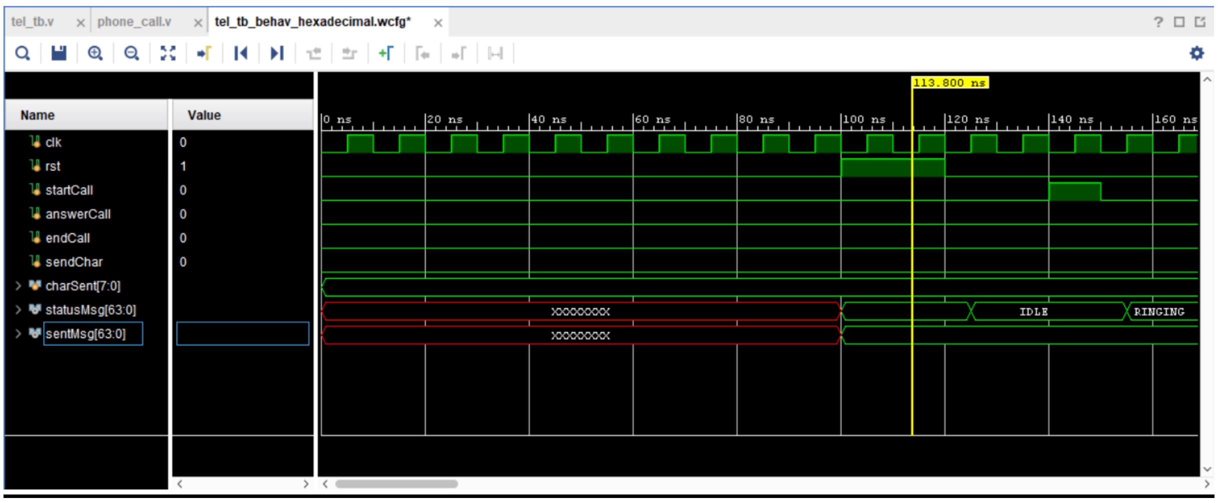
We have 6 states in total which are “IDLE”, “BUSY”, “REJECTED”, “RINGING”, “CALL”, and “COST”; but we also defined 2 state identifier ourselves for holding current and next state information: curr_state and next_state(using 3 bits since for representing 6 different states 3 bits are needed). Then, we created a counter which counts from 0 to 10 that we resetted whenever the program changes its state or rst input becomes 1. For cost calculation, we defined a parameter which is going to hold the cost as a number.



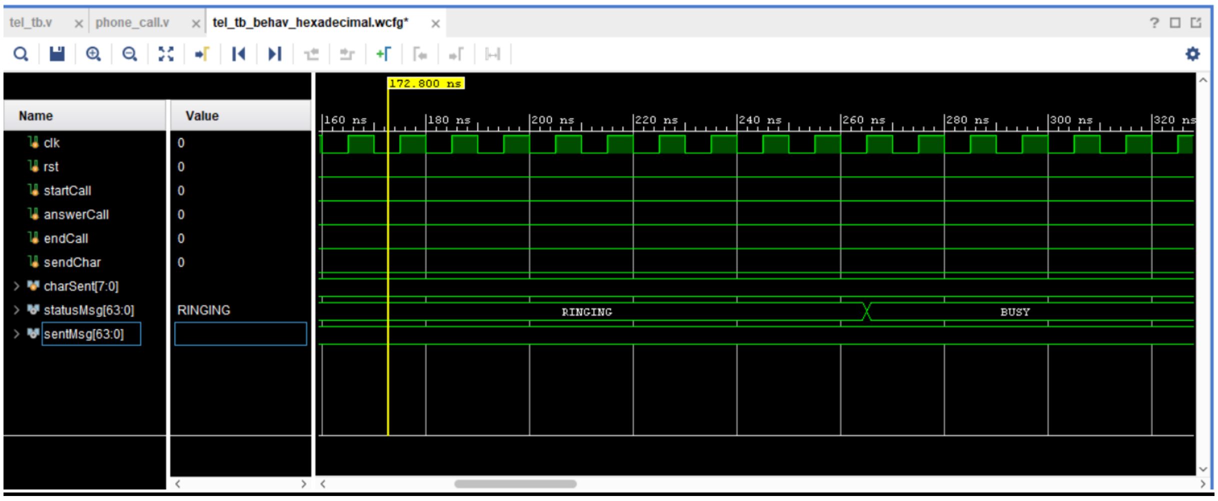
```

parameter IDLE = 0;
parameter BUSY = 1;
parameter REJECTED = 2;
parameter RINGING = 3;
parameter CALL = 4;
parameter COST = 5;

reg [2:0] curr_state, next_state;
reg [4:0] counter;
reg [31:0] cost;

```

***Simulation results***

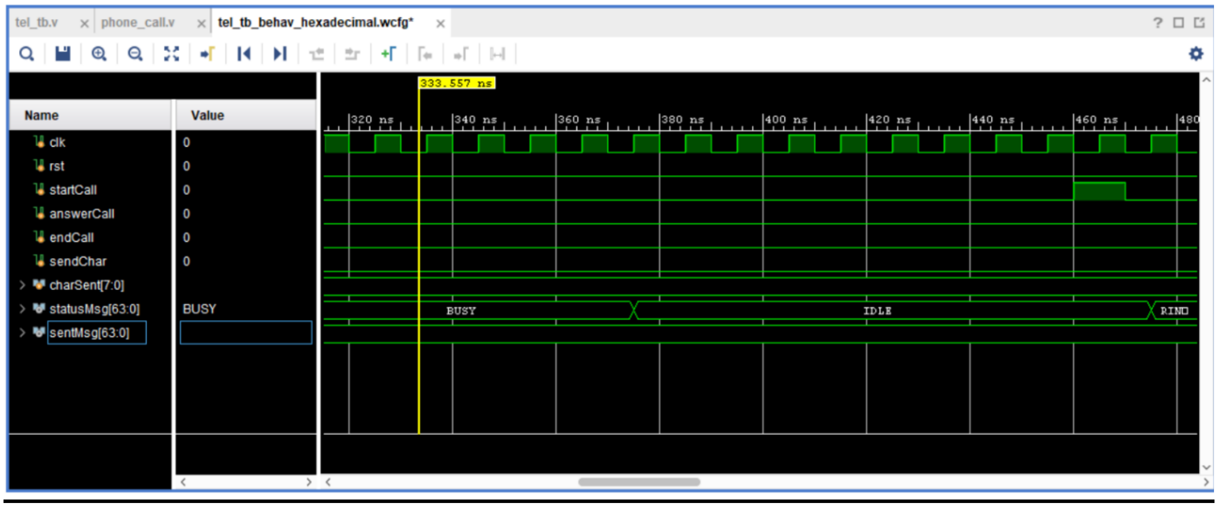
a) **<u>Simulation 1:</u>**



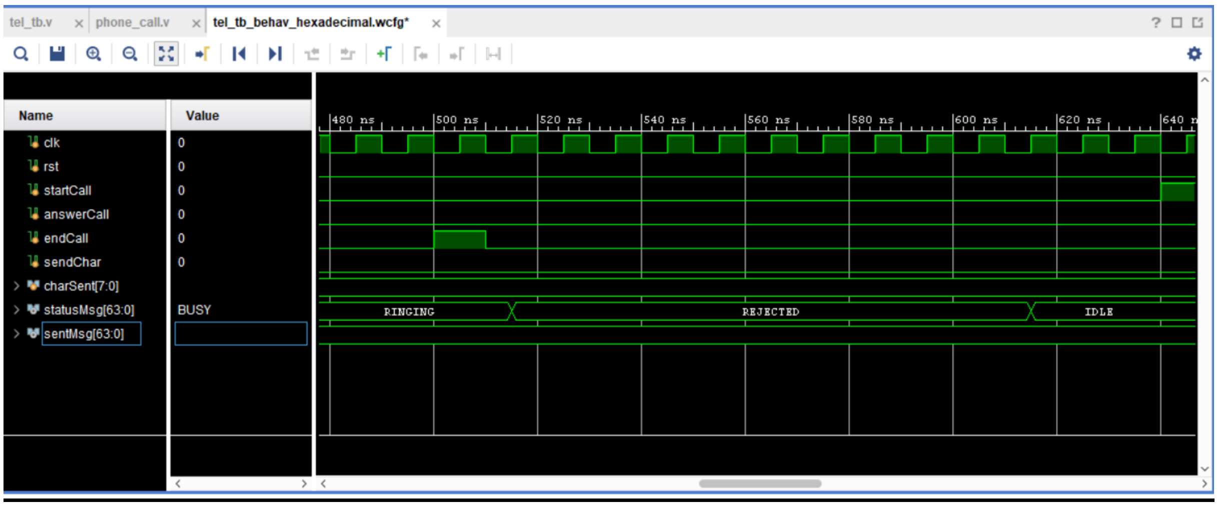
| Name | Value |
|---|---|
| clk | 0 |
| rst | 1 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | |
| sentMsg[63:0] | |



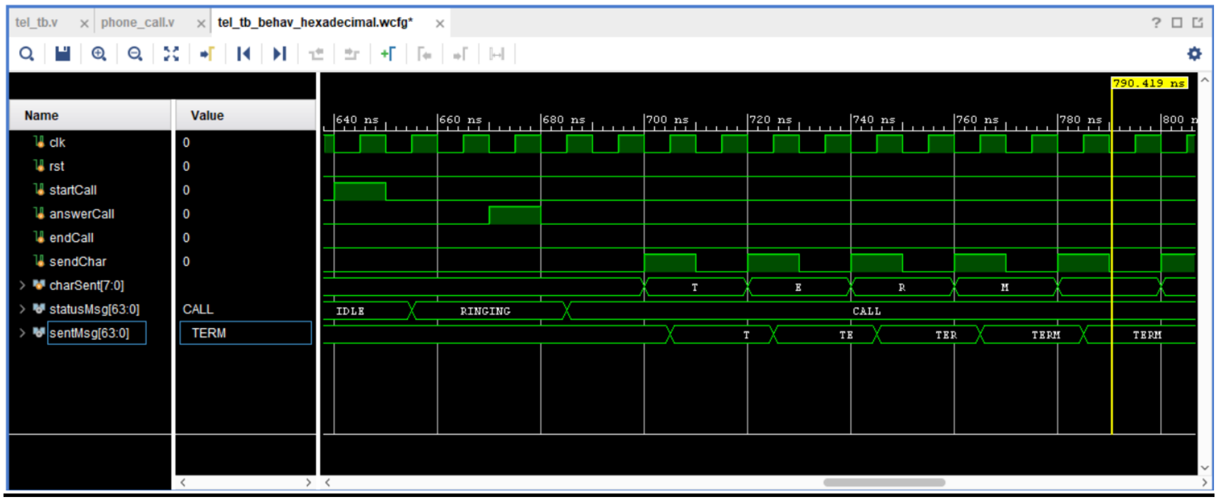
| Name | Value |
|---|---|
| clk | 0 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | RINGING |
| sentMsg[63:0] | |



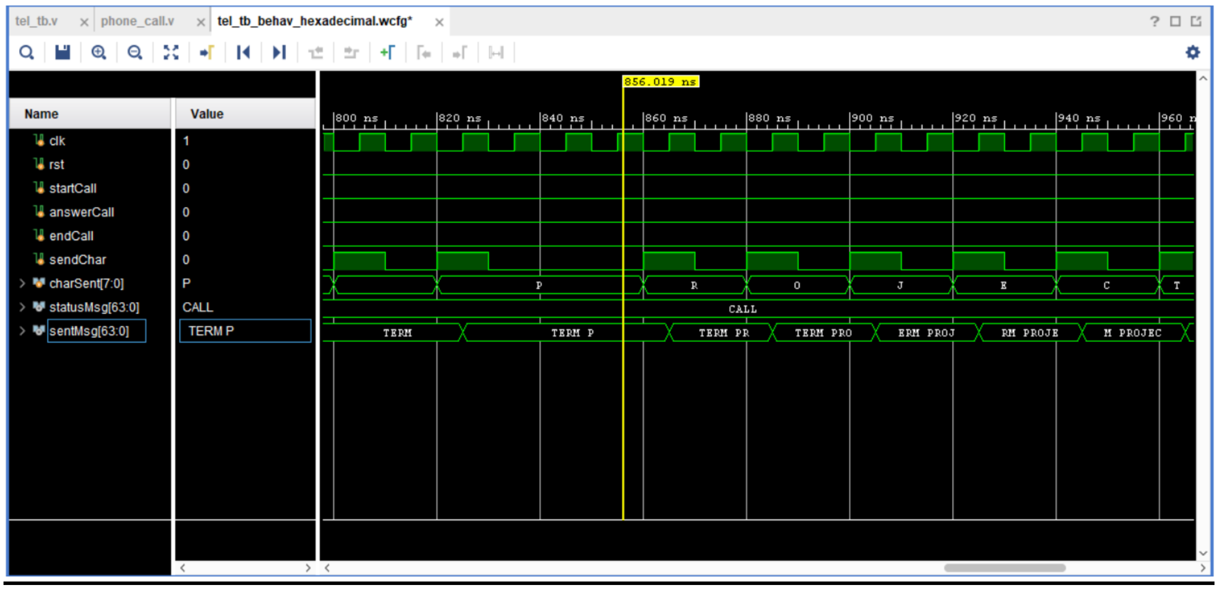
| Name | Value |
|---|---|
| clk | 0 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | BUSY |
| sentMsg[63:0] | |



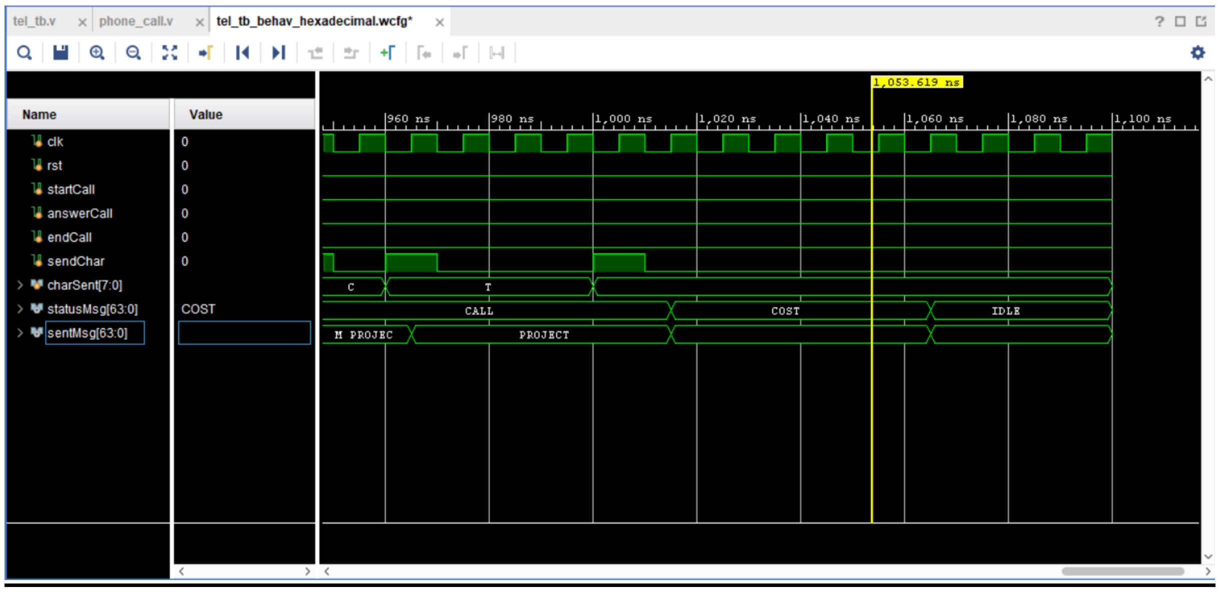
| Name | Value |
|---|---|
| clk | 0 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | BUSY |
| sentMsg[63:0] | |



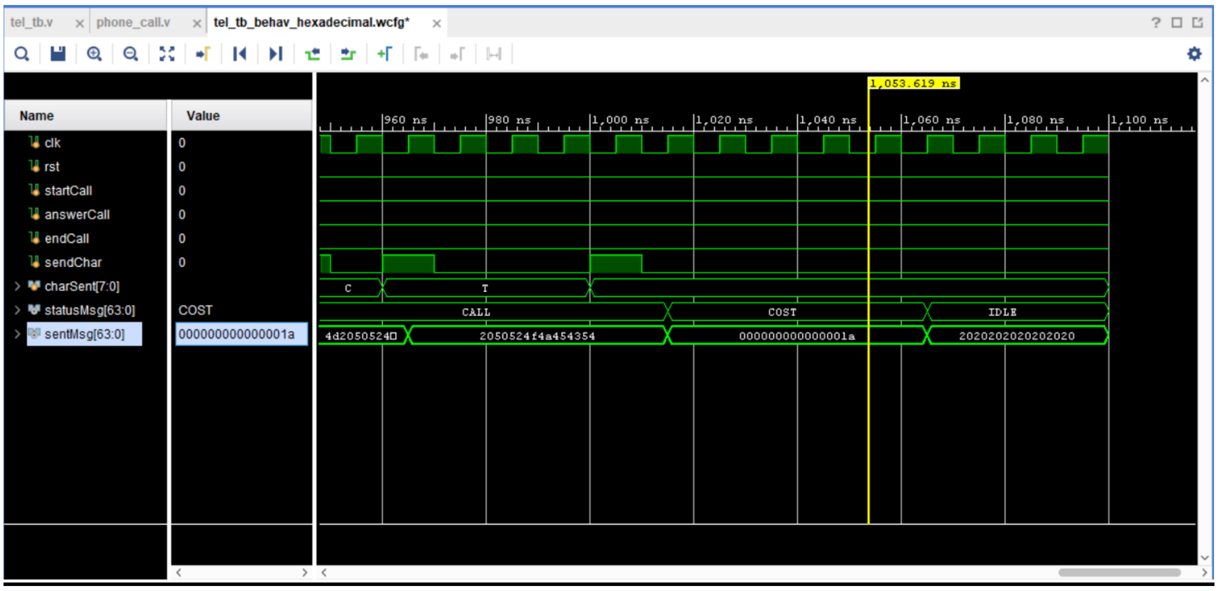
| Name | Value |
|---|---|
| clk | 0 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | TERM |



| Name | Value |
|---|---|
| clk | 1 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | P |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | TERM P |

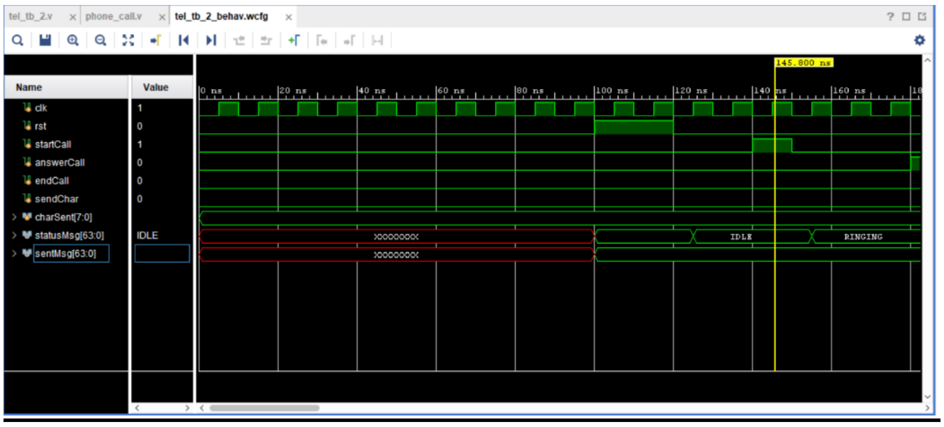


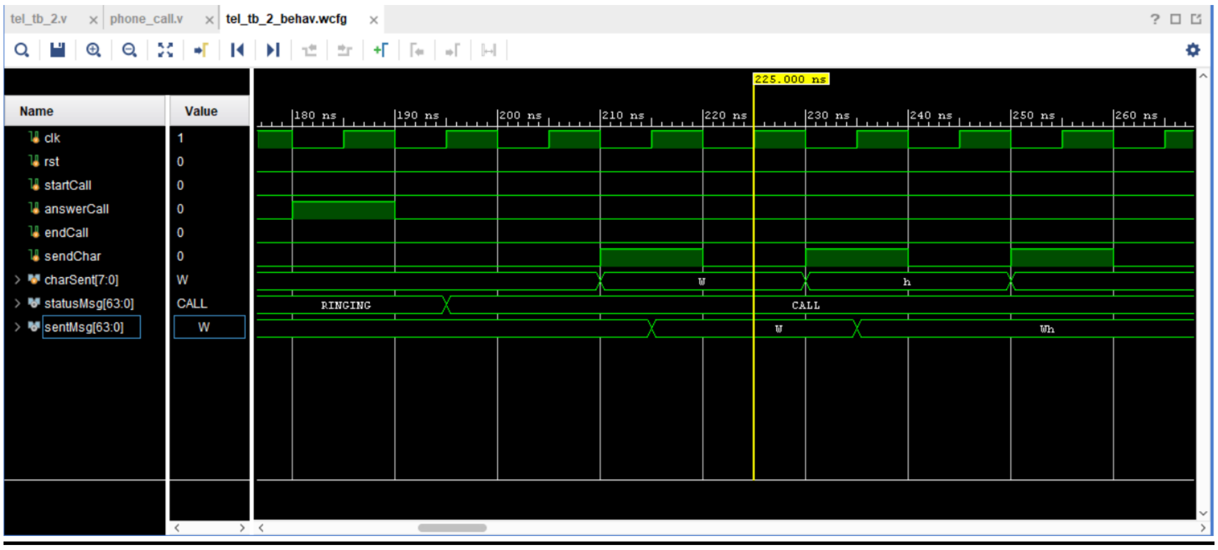
| Name | Value |
|---|---|
| clk | 0 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | COST |
| sentMsg[63:0] | |

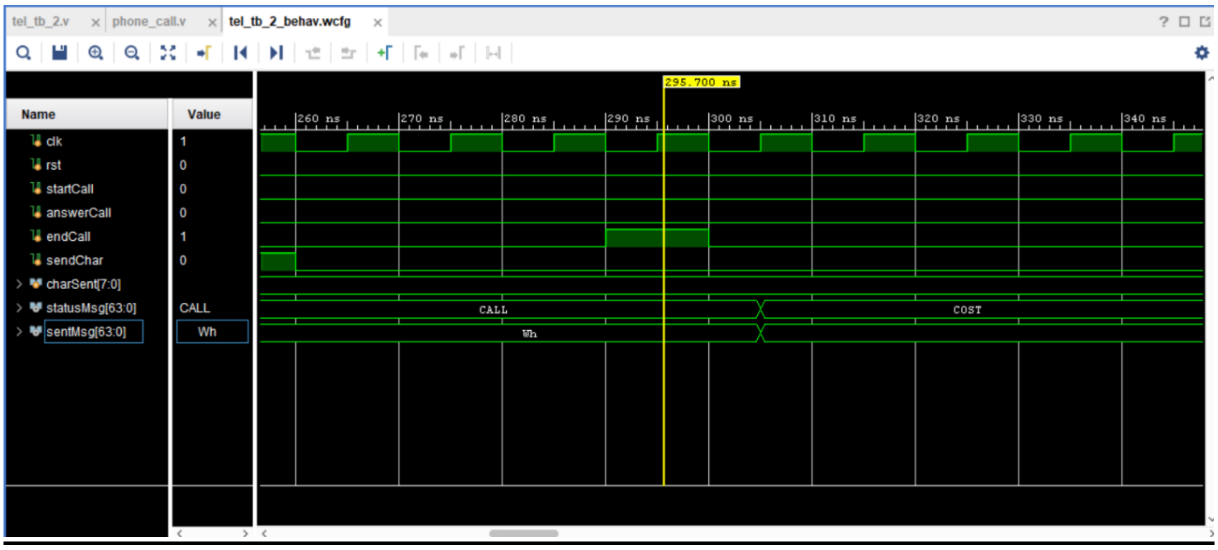


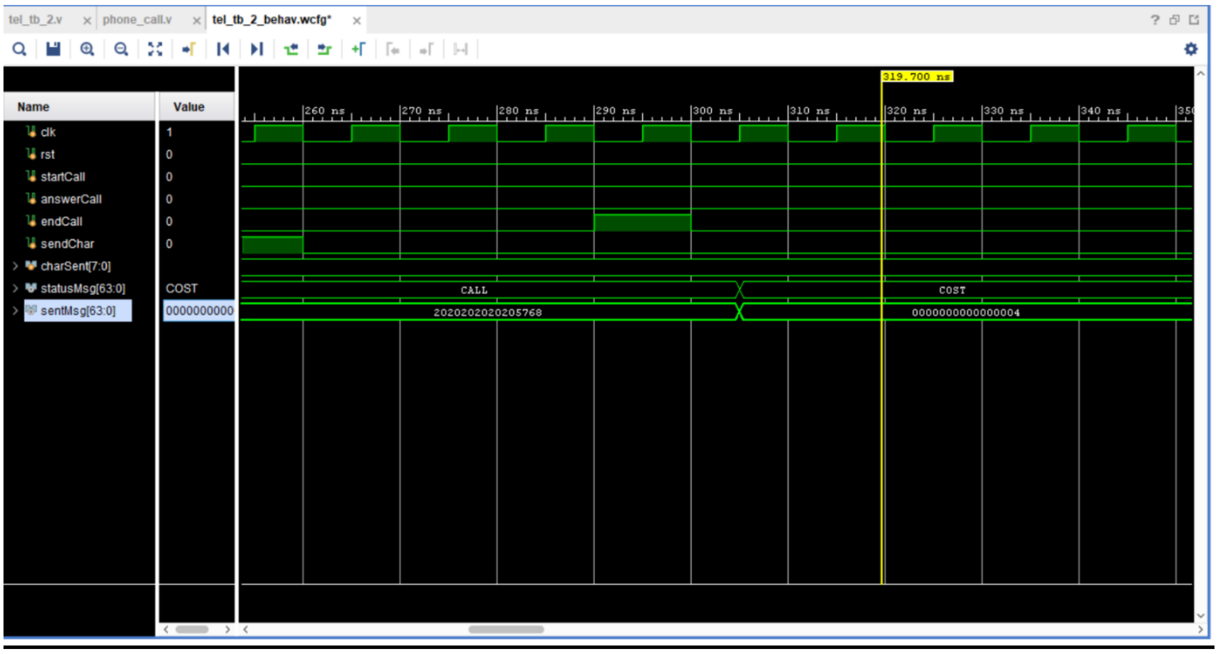
At the end, we convert *sentMsg* to hexadecimal format in order to see the cost.

## b) **Simulation 2:**

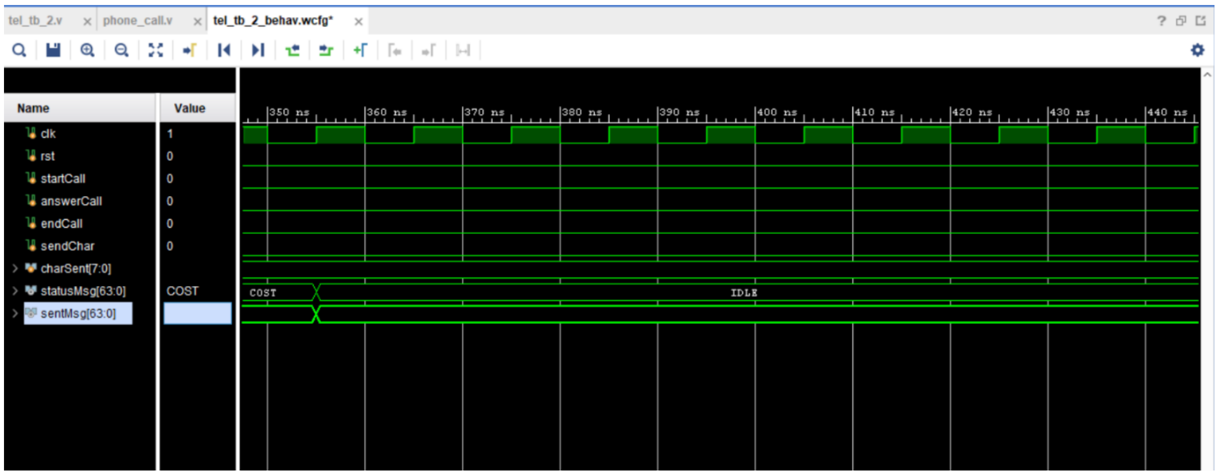


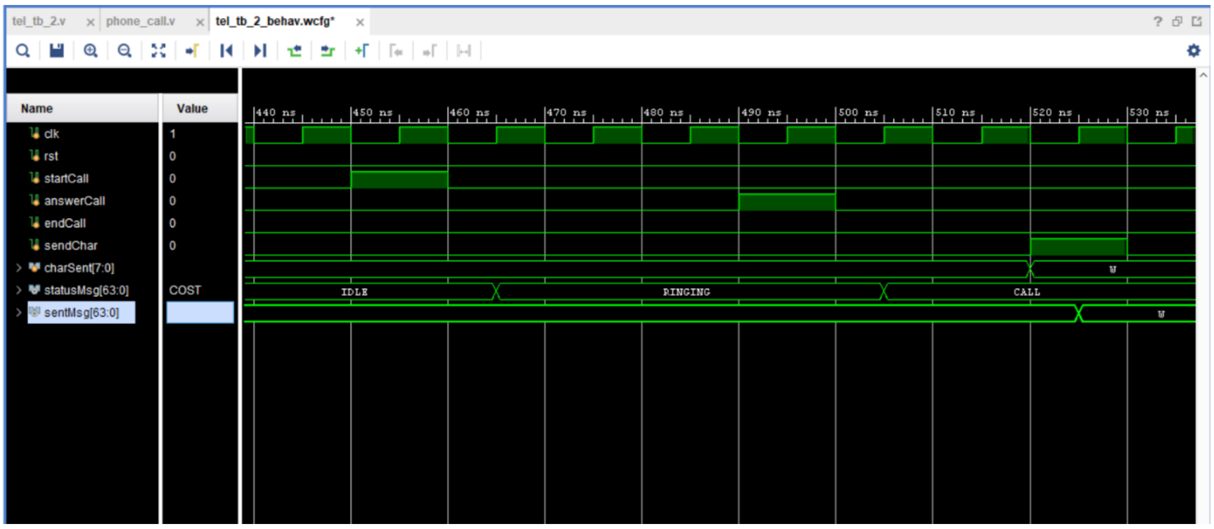




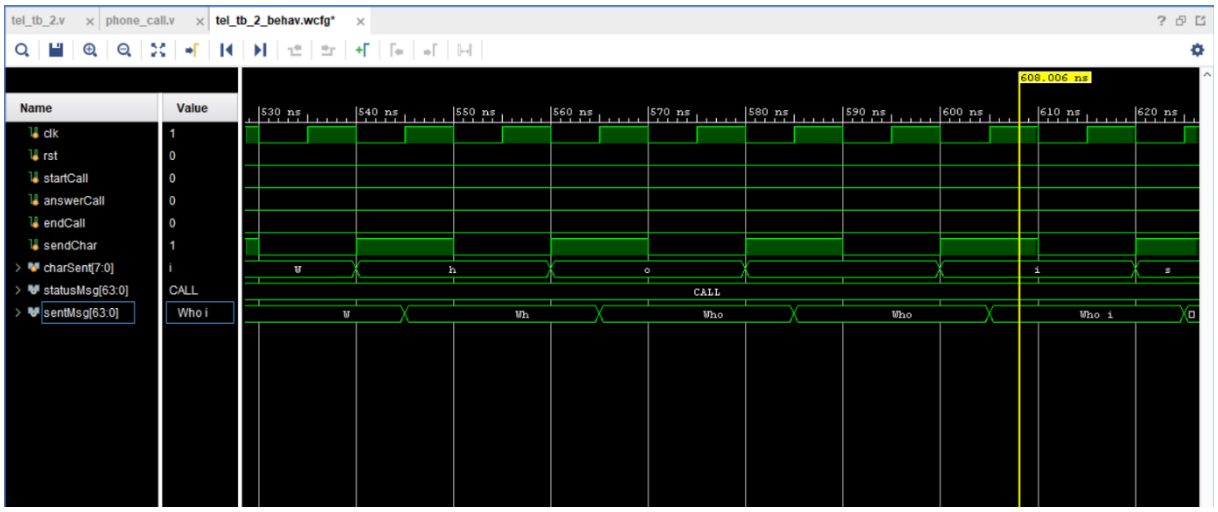


Here, we do hexadecimal conversion in order to see the cost.

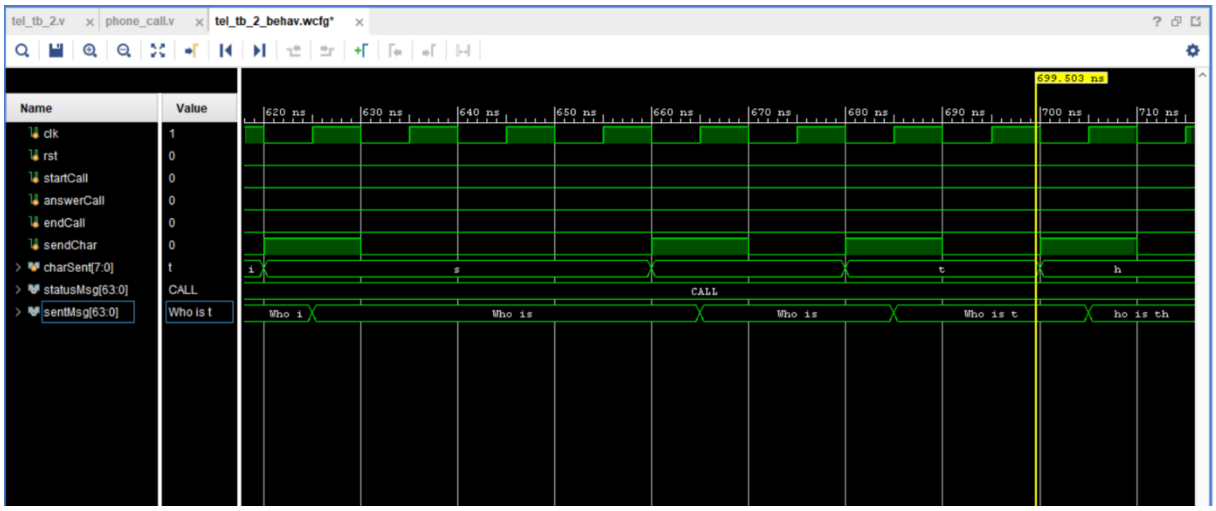




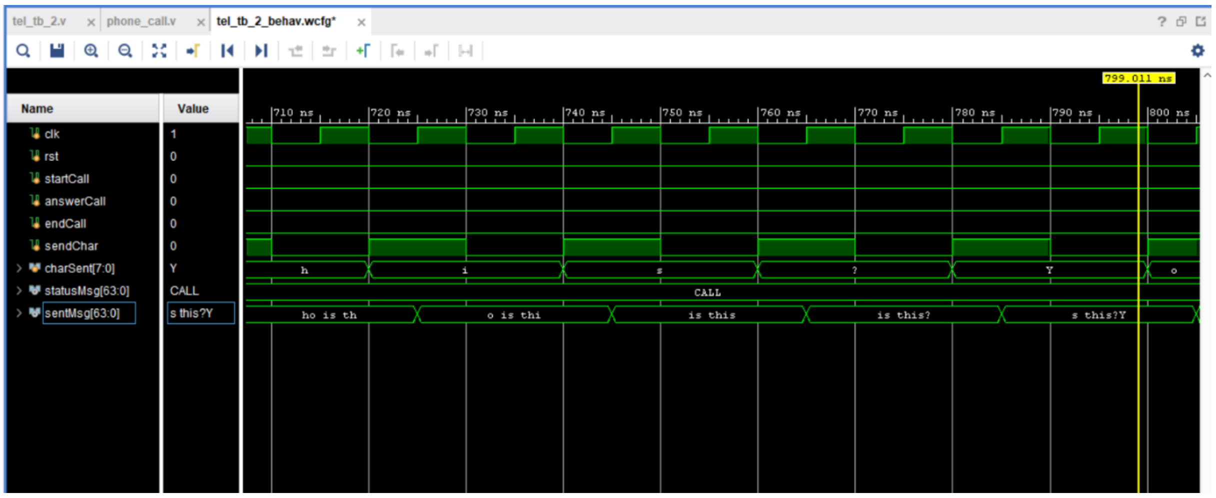
| Name | Value |
|---|---|
| clk | 1 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | |
| statusMsg[63:0] | COST |
| sentMsg[63:0] | |



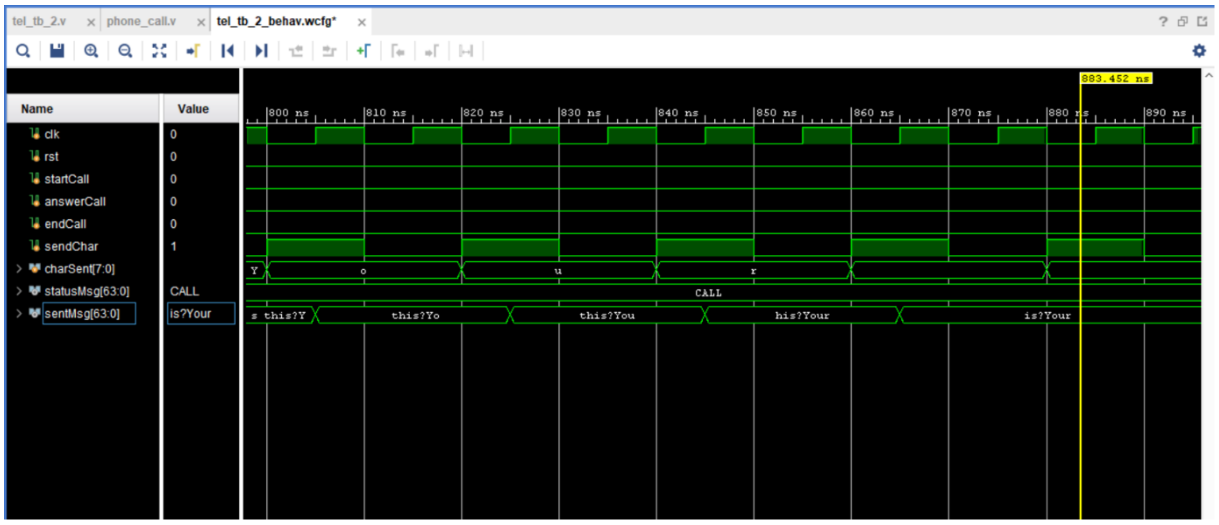
| Name | Value |
|---|---|
| clk | 1 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 1 |
| charSent[7:0] | i |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | Who i |



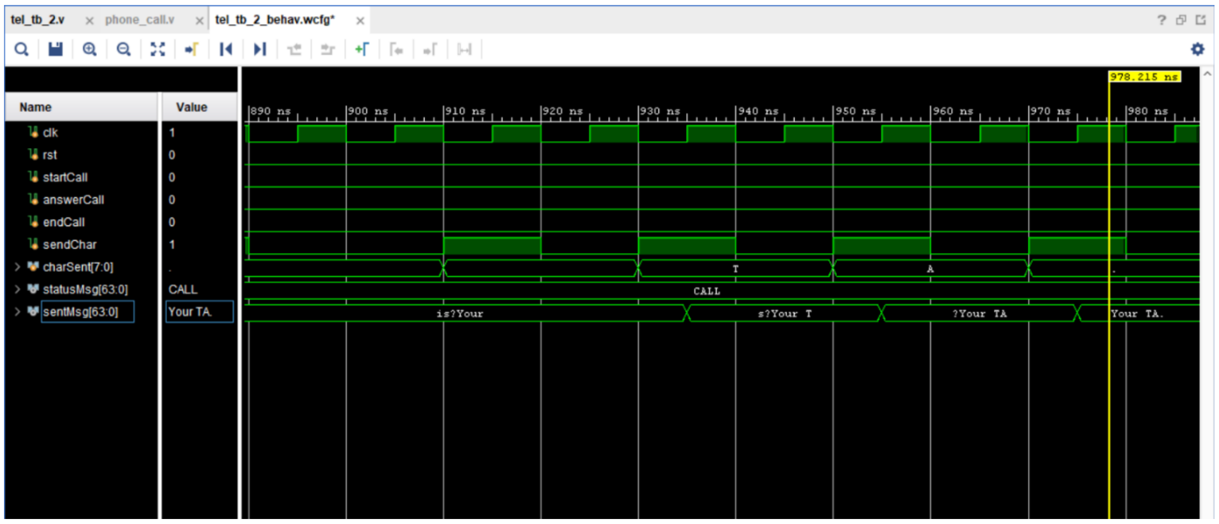
| Name | Value |
|---|---|
| clk | 1 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | t |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | Who is t |



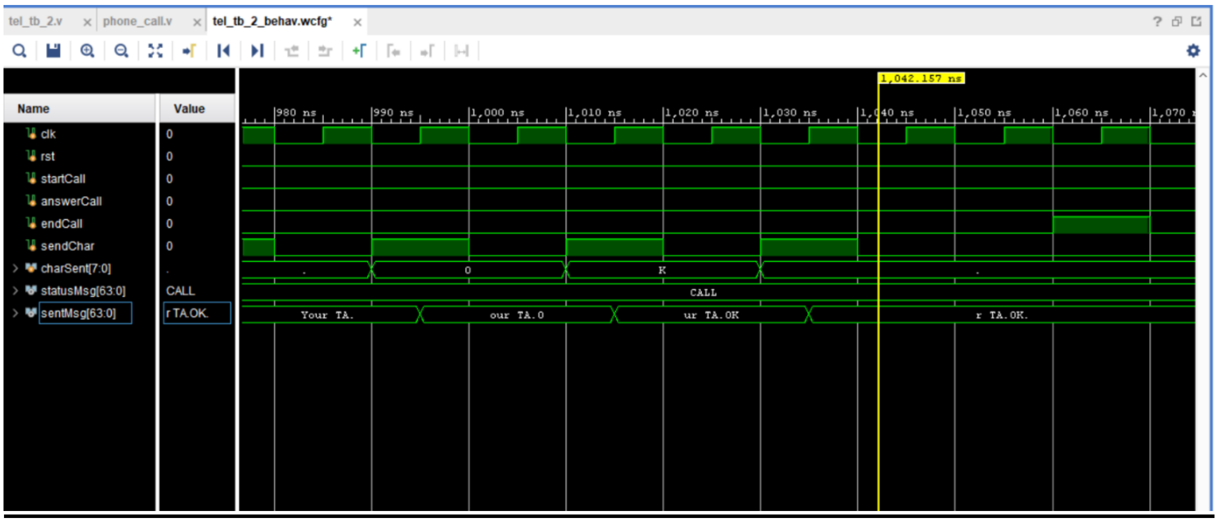
| Name | Value |
|---|---|
| clk | 1 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 0 |
| charSent[7:0] | Y |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | s this?Y |

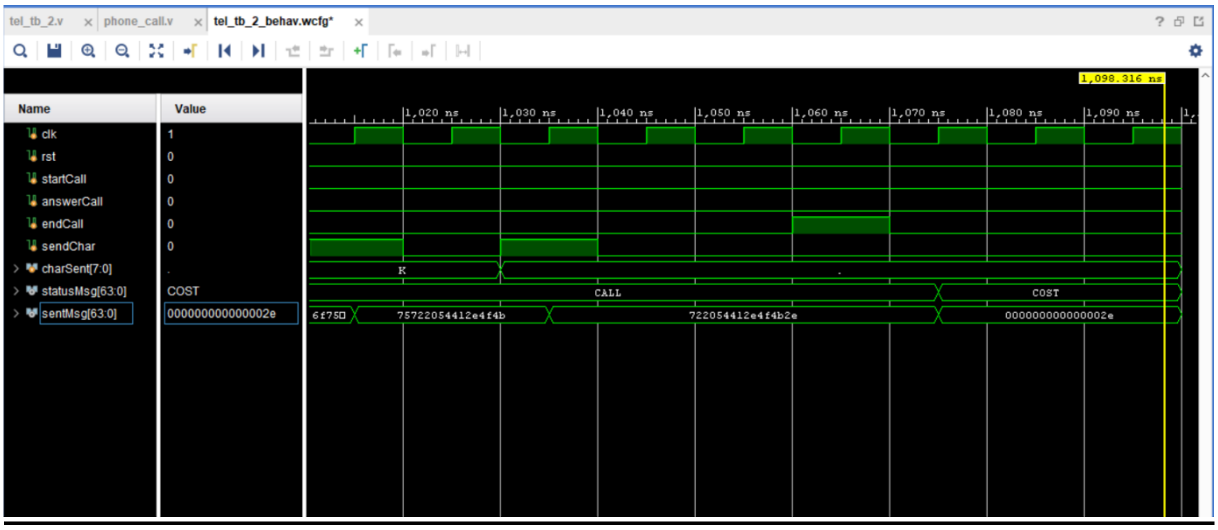


| Name | Value |
|---|---|
| clk | 0 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 1 |
| charSent[7:0] | |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | is?Your |

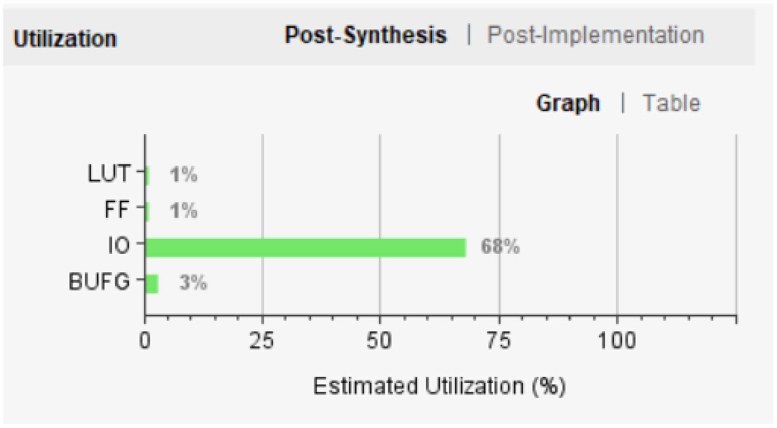


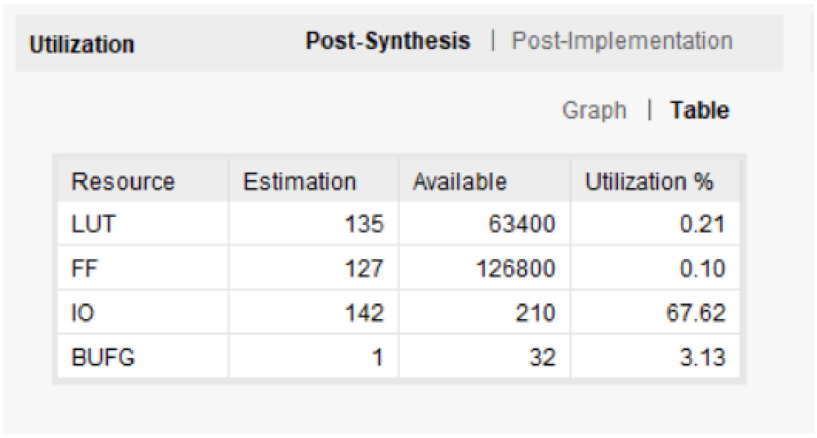
| Name | Value |
|---|---|
| clk | 1 |
| rst | 0 |
| startCall | 0 |
| answerCall | 0 |
| endCall | 0 |
| sendChar | 1 |
| charSent[7:0] | . |
| statusMsg[63:0] | CALL |
| sentMsg[63:0] | Your TA. |





## *Synthesis Results*





Utilization **Post-Synthesis** | Post-Implementation

Graph | **Table**

| Resource | Estimation | Available | Utilization % |
|---|---|---|---|
| LUT | 135 | 63400 | 0.21 |
| FF | 127 | 126800 | 0.10 |
| IO | 142 | 210 | 67.62 |
| BUFG | 1 | 32 | 3.13 |

***Group Members:***

| | |
|---|---|
| Uğur Kağan Çakır | 27058 |
| Selin Tokman | 28495 |