CS303 TERM PROJECT REPORT

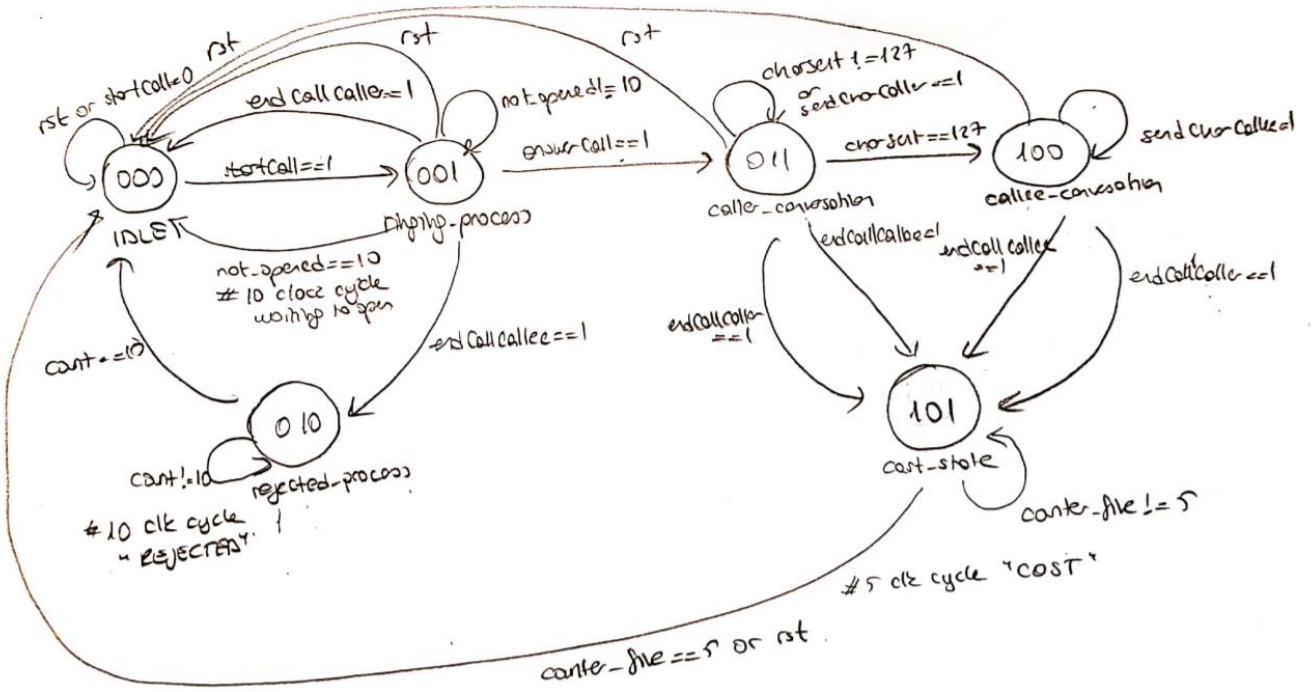
Name-Surname/SID: Zeynep Tandoğan/25200

Project Description

In this project, we are required to design a sequential circuit for a simple two-sided telephone conversation and implement it using Verilog HDL. For this purpose, we have 8 inputs that give details of the conversation and 2 outputs which show the last 8 char of the conversation and status of the conversation.

The details of states:

Before starting to write the Verilog code, I formed the state diagram of this problem in order to have a roadmap. I formed 6 states which are called as IDLE, ringing_process, rejected_process, caller_conversation, callee_conversation and cost_state.



We have asynchronous reset in this circuit. For this reason, every time the reset(rst) is 1, the state become IDLE. I used 6 states and I prefer to use binary encoding; for this reason, I have 3 bits for the states.

- IDLE: When rst is 1 or startCall is 0, it means that the next state will be IDLE, there won't be any change in the state. But if startCall is 1, it means that caller call the callee, and it moves to the ringing_process which shows the period of ringing the callee's phone.

- Ringing_process: In this state, we have to count the number of clock cycles that we wait for callee to pick up the phone. I count this time with "not_opened", if it is not 10 it means that we are in ringing_process but when it is equal to 10, we will go back to IDLE state. In addition to that, we have to know whether the phone is hung up whether by callee or caller. If it is ended by caller, we immediately go back to the IDLE; however, if it is ended by the callee, we have to go rejected_process, we have to do additional things to show it is rejected by the

callee. If answerCall is 1 it means that callee pick up the phone and the conversation will start, for this reason next state will be caller_conversation.

- Rejected_process: We have to show that the call is REJECTED by the callee in the statusMsg for 10 clock cycles. For this reason, I formed this state. I count the cycles with “count”, if it is equal to 10, we can go back to IDLE; otherwise, we would not change the state.

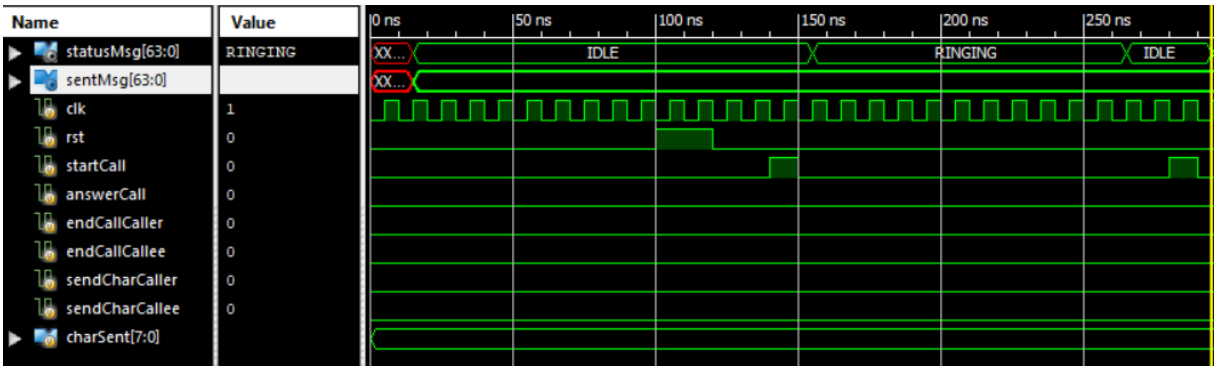
- Caller_conversation: In this state, while caller is sending char and the char is not 127, we will not change the state, caller will continue to talk. But it is known that any time caller or callee can end the conversation. If one of them end this call, the next state will be cost_state which will show the total cost of the conversation in statusMsg. If the char that is sent by callee is 127, it is the callee’s turn to talk; the next state will be callee_conversation.

- Callee_conversation: While the callee continues to talk, which means that sendCharCallee is 1, the state won’t be changed. However, if the call is ended by one of them, the next state will be cost_state.

- Cost_state: In this state, the calculated cost will be shown in the statusMsg in 5 clock cycle. For this, I used “counter_five”. Until it become 5, the state will remain the same, when it becomes 5, the state will be IDLE.

Simulation Results

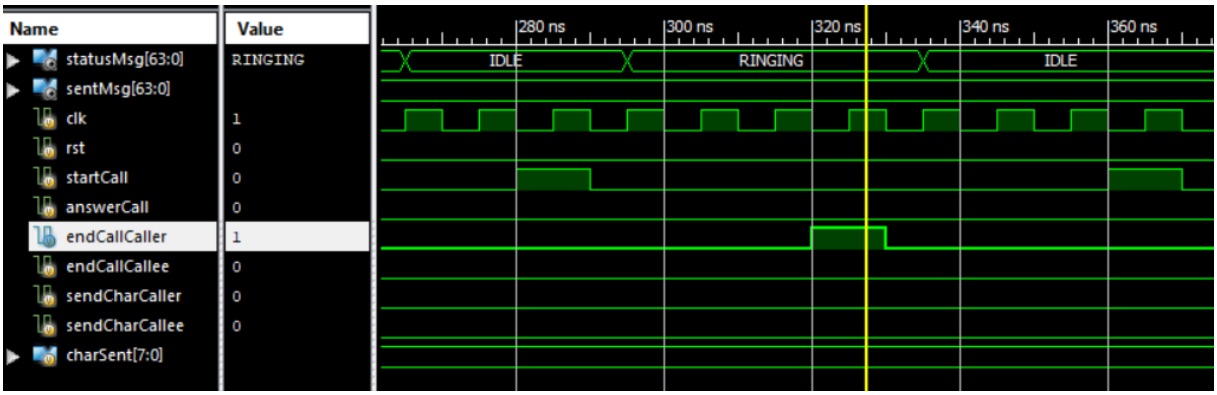
I will show the results with their corresponding codes in the testbench.




```
// reset your circuit
rst=1; #20; rst=0; #20; rst=0;              // reset

startCall=1; #10; startCall=0; #10;         // caller starts call
#100;                                       // statusMsg displaying "RINGING "
                                            // no answer for 10 clock cycles (and go back to IDLE)
#20;
```

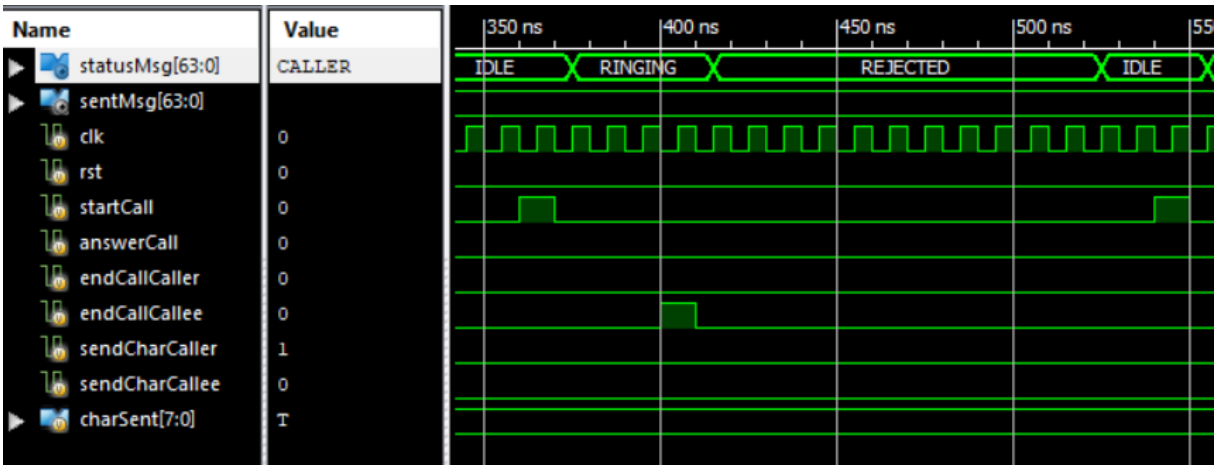
As it is wanted, after waiting 10 clock cycle in the ringing process. Since it is not opened, it goes back to the IDLE state.

```
startCall=1; #10; startCall=0; #10;          // caller starts call
#20;                                         // statusMsg displaying "RINGING "
endCallCaller=1; #10; endCallCaller=0; #10; // caller ends the call
#20;                                         // statusMsg displaying "RINGING "
```



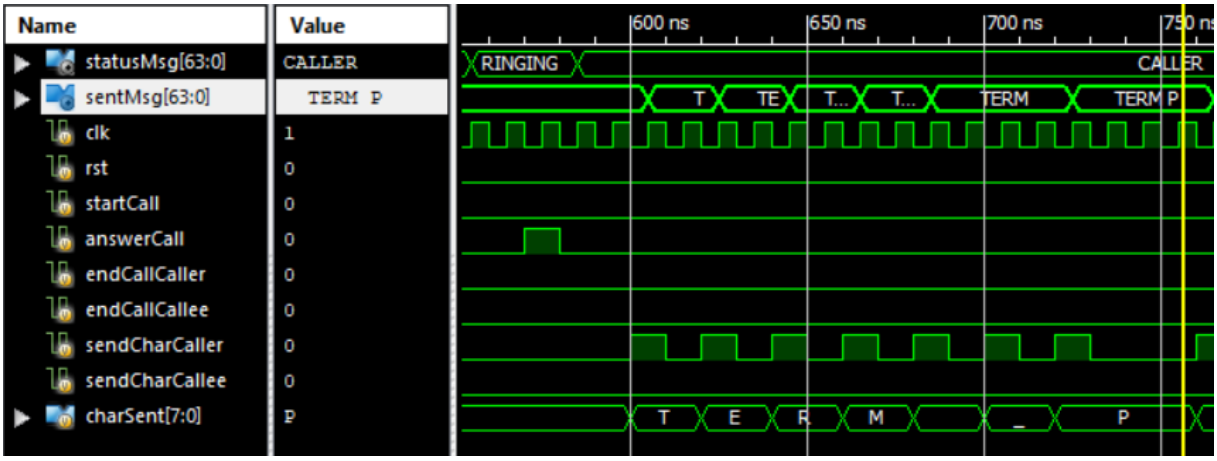
After it is ended by the caller, it goes back to the IDLE state.

```
startCall=1; #10; startCall=0; #10;          // caller starts call
#20;                                         // statusMsg displaying "RINGING "
endCallCallee=1; #10; endCallCallee=0; #10; // callee rejects the call
#100;                                        // statusMsg displaying 'REJECTED' for 10 clock cycles
#20;
```



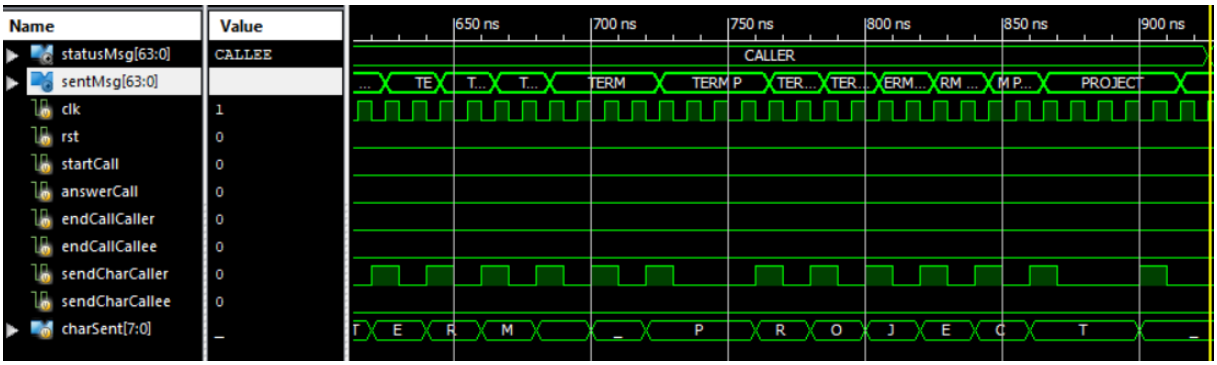
StatusMsg becomes REJECTED for 10 clock cycle since the callee ended the call.

```
startCall=1; #10; startCall=0;              // caller starts call
#20;                                        // statusMsg displaying "RINGING "
answerCall=1; #10; answerCall=0;            // callee answer call
#20;                                       // statusMsg displaying "CALLER  "

sendCharCaller=1; charSent="T"; #10; sendCharCaller=0; #10; // caller sends "T", sentMsg displaying "      T", cost is 2

sendCharCaller=1; charSent="E"; #10; sendCharCaller=0; #10; // caller sends "E", sentMsg displaying "     TE", cost is 4
sendCharCaller=1; charSent="R"; #10; sendCharCaller=0; #10; // caller sends "R", sentMsg displaying "    TER", cost is 6
sendCharCaller=1; charSent="M"; #10; sendCharCaller=0; #10; // caller sends "M", sentMsg displaying "   TERM", cost is 8
sendCharCaller=1; charSent=" "; #10; sendCharCaller=0; #10; // caller sends " ", sentMsg displaying "  TERM ", cost is 10
sendCharCaller=1; charSent= 12; #10; sendCharCaller=0; #10; // caller sends (invalid char), no change on sentMsg and cost
sendCharCaller=1; charSent="P"; #10; sendCharCaller=0; #10; // caller sends "P", sentMsg displaying "  TERM P", cost is 12
```



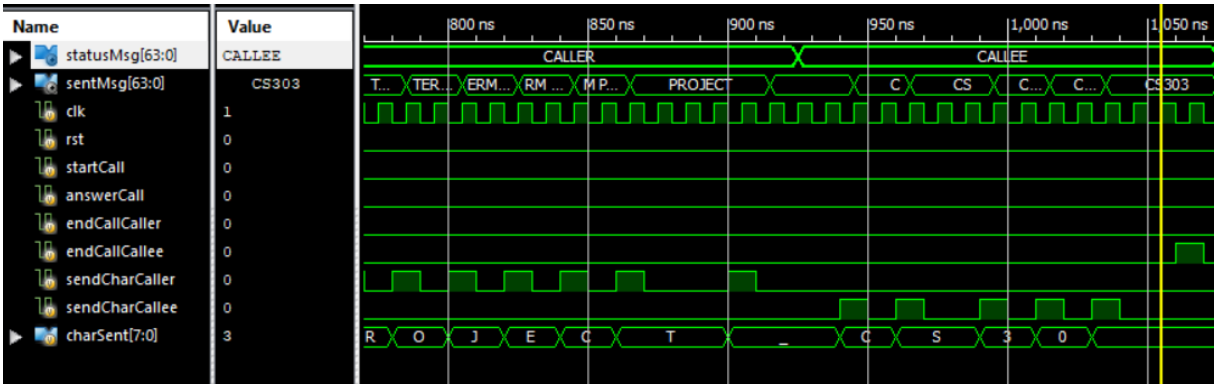
As it is seen above, the characters are seen in the sentMsg in a correct manner. The caller still continues to talk it doesn't enter 127(del).

```
#10;
#10;
sendCharCaller=1; charSent="R"; #10; sendCharCaller=0; #10; // caller sends "R", sentMsg displaying " TERM PR", cost is 14
sendCharCaller=1; charSent="O"; #10; sendCharCaller=0; #10; // caller sends "O", sentMsg displaying "TERM PRO", cost is 16
sendCharCaller=1; charSent="J"; #10; sendCharCaller=0; #10; // caller sends "J", sentMsg displaying "ERM PROJ", cost is 18
sendCharCaller=1; charSent="E"; #10; sendCharCaller=0; #10; // caller sends "E", sentMsg displaying "RM PROJE", cost is 20
sendCharCaller=1; charSent="C"; #10; sendCharCaller=0; #10; // caller sends "C", sentMsg displaying "M PROJEC", cost is 22
sendCharCaller=1; charSent="T"; #10; sendCharCaller=0; #10; // caller sends "T", sentMsg displaying " PROJECT", cost is 24
#10;
#10;
sendCharCaller=1; charSent=127; #10; sendCharCaller=0; #10; // caller sends DEL to change turn, cost is 26
```



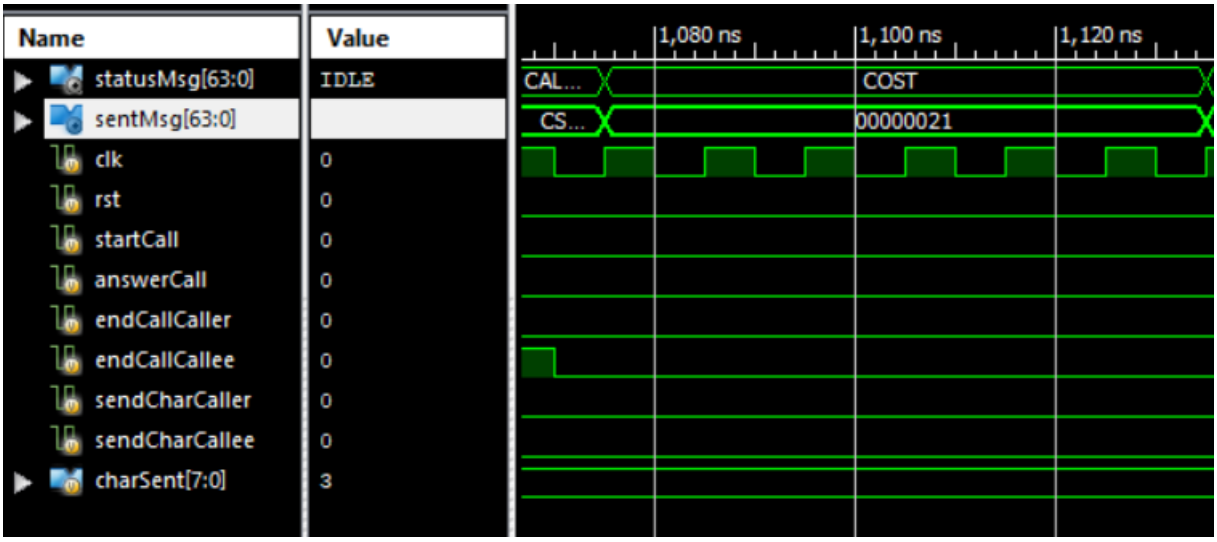
Since del came from the caller, callee will start to talk.

```
#10;                                                    // statusMsg displaying "CALLEE  "
#10;

sendCharCallee=1; charSent="C"; #10; sendCharCallee=0; #10; // callee sends "C", sentMsg displaying "       C", cost is 28
sendCharCallee=1; charSent="S"; #10; sendCharCallee=0; #10; // callee sends "S", sentMsg displaying "      CS", cost is 30

#10;
sendCharCallee=1; charSent="3"; #10; sendCharCallee=0; #10; // callee sends "3", sentMsg displaying "     CS3", cost is 31
sendCharCallee=1; charSent="0"; #10; sendCharCallee=0; #10; // callee sends "0", sentMsg displaying "    CS30", cost is 32

sendCharCallee=1; charSent="3"; #10; sendCharCallee=0; #10; // callee sends "3", sentMsg displaying "   CS303", cost is 33
#10;
```



sentMsg is cleared when the turn is the callee. If it was not the case it would be like ROJECT C.

```
endCallCallee=1; #10; endCallCallee=0; // callee ends the call

#50;                                   // statusMsg displaying "COST    ", sentMsg displaying "00000021"
```



The total cost of the conversation is seen in the statusMsg, in the form of hexadecimal. After it is shown in 5 clock cycle. The state come back to the IDLE state.

Synthesis Results

```
=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : tel_project.ngr
Top Level Output File Name         : tel_project
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 144

Cell Usage :
# BELS                             : 392
#      GND                         : 1
#      LUT1                        : 30
#      LUT2                        : 22
#      LUT2_D                      : 1
#      LUT2_L                      : 8
#      LUT3                        : 61
#      LUT3_L                      : 40
#      LUT4                        : 134
#      LUT4_D                      : 4
#      LUT4_L                      : 23
#      MUXCY                       : 31
#      MUXF5                       : 5
#      XORCY                       : 32
# FlipFlops/Latches                : 142
#      FDC                         : 75
#      FDCE                        : 43
#      FDP                         : 18
#      LD                          : 6
# Clock Buffers                    : 1
#      BUFGP                       : 1
# IO Buffers                       : 143
#      IBUF                        : 15
#      OBUF                        : 128
=========================================================================

Device utilization summary:
---------------------------

Selected Device : 3s100etq144-4

 Number of Slices:                      172  out of    960    17%
 Number of Slice Flip Flops:            142  out of   1920     7%
 Number of 4 input LUTs:                323  out of   1920    16%
 Number of IOs:                         144
 Number of bonded IOBs:                 144  out of    108   133% (*)
 Number of GCLKs:                         1  out of     24     4%
```

There are 323 four input LUTs in this design. The area usage is observed by looking at the number of LUTs.

```
Timing Summary:
---------------
Speed Grade: -4

   Minimum period: 6.565ns (Maximum Frequency: 152.323MHz)
   Minimum input arrival time before clock: 10.213ns
   Maximum output required time after clock: 4.496ns
   Maximum combinational path delay: No path found

Timing Detail:
--------------
All values displayed in nanoseconds (ns)
```

Although maximum combinational path delay is not found, the other values are calculated in the timing summary.