## Term Project

### A Simple Telephone Conversation

You will design a sequential circuit for a simple one-sided telephone conversation and implement it using Verilog HDL. Then, you will simulate and show its functional correctness using Vivado 2018.2 tool. Basically, the caller will initiate a telephone conversation with the callee and the caller will send characters to the callee. Your circuit will calculate the cost of the call and send the cost value and the characters (sent from the caller to callee) as outputs.

### Inputs

There will be 6 inputs in your circuitry:

- *rst* will set your circuitry to its initial state.
- *startCall* (1-bit) will be used by the caller and it will represent that the caller pressing a button to start a call.
- *answerCall* (1-bit) will be used by the callee and it will represent that the callee pressing a button to answer an incoming call.
- *endCall* (1-bit) will be used by the callee and it will represent that the callee pressing a button to end a call.
- *charSent* (8-bit) will be used to define 8-bit printable ASCII character to be sent from the caller to the callee according to printable ASCII table shown at Fig. 1.
- *sendChar* (1-bit) will be used by the caller and it will represent that the caller pressing a button to send an ASCII character (set by *charSent* input) to the callee.

### Outputs

There will be 2 outputs in your circuitry:

- *statusMsg* (64-bit) will be used to display the status of telephone using 8 printable ASCII character. For example, *statusMsg* output should be "`IDLE` " in its initial state. More details will be provided below.
- *sentMsg* (64-bit) will be used to display the last 8 printable ASCII characters sent by caller. More details will be provided below.

| Decimal | Hex | Char | Decimal | Hex | Char | Decimal | Hex | Char |
|---|---|---|---|---|---|---|---|---|
| 32 | 20 | [SPACE] | 64 | 40 | @ | 96 | 60 | ` |
| 33 | 21 | ! | 65 | 41 | A | 97 | 61 | a |
| 34 | 22 | " | 66 | 42 | B | 98 | 62 | b |
| 35 | 23 | # | 67 | 43 | C | 99 | 63 | c |
| 36 | 24 | $ | 68 | 44 | D | 100 | 64 | d |
| 37 | 25 | % | 69 | 45 | E | 101 | 65 | e |
| 38 | 26 | & | 70 | 46 | F | 102 | 66 | f |
| 39 | 27 | ' | 71 | 47 | G | 103 | 67 | g |
| 40 | 28 | ( | 72 | 48 | H | 104 | 68 | h |
| 41 | 29 | ) | 73 | 49 | I | 105 | 69 | i |
| 42 | 2A | * | 74 | 4A | J | 106 | 6A | j |
| 43 | 2B | + | 75 | 4B | K | 107 | 6B | k |
| 44 | 2C | , | 76 | 4C | L | 108 | 6C | l |
| 45 | 2D | - | 77 | 4D | M | 109 | 6D | m |
| 46 | 2E | . | 78 | 4E | N | 110 | 6E | n |
| 47 | 2F | / | 79 | 4F | O | 111 | 6F | o |
| 48 | 30 | 0 | 80 | 50 | P | 112 | 70 | p |
| 49 | 31 | 1 | 81 | 51 | Q | 113 | 71 | q |
| 50 | 32 | 2 | 82 | 52 | R | 114 | 72 | r |
| 51 | 33 | 3 | 83 | 53 | S | 115 | 73 | s |
| 52 | 34 | 4 | 84 | 54 | T | 116 | 74 | t |
| 53 | 35 | 5 | 85 | 55 | U | 117 | 75 | u |
| 54 | 36 | 6 | 86 | 56 | V | 118 | 76 | v |
| 55 | 37 | 7 | 87 | 57 | W | 119 | 77 | w |
| 56 | 38 | 8 | 88 | 58 | X | 120 | 78 | x |
| 57 | 39 | 9 | 89 | 59 | Y | 121 | 79 | y |
| 58 | 3A | : | 90 | 5A | Z | 122 | 7A | z |
| 59 | 3B | ; | 91 | 5B | [ | 123 | 7B | { |
| 60 | 3C | < | 92 | 5C | \ | 124 | 7C | \| |
| 61 | 3D | = | 93 | 5D | ] | 125 | 7D | } |
| 62 | 3E | > | 94 | 5E | ^ | 126 | 7E | ~ |
| 63 | 3F | ? | 95 | 5F | _ | 127 | 7F | [DEL] |

Fig.1. Printable ASCII Characters

## Operation Steps

The circuitry will start in the IDLE state, in which it should output '`IDLE `' to *statusMsg* output (last 4 characters are space). You should use *rst* input as an asynchronous reset input to put the sequential circuit into IDLE state. In IDLE state, the caller can initiate a call by pressing *startCall* when the circuit is in IDLE state. Otherwise, the circuit will stay in IDLE state. If the caller initiates a call by pressing *startCall*, the telephone will start ringing. When the telephone is ringing, the 8-bit ASCII output "`RINGING `" should be at output *statusMsg*. When the telephone is ringing, there are three possibilities:

1. If the callee rejects the call by pressing *endCall*, your circuit should output '`REJECTED`' to *statusMsg* output for 10 clock cycles and then your circuit should go back to the IDLE state.
2. If the callee does not answer the call for 10 clock cycles, your circuit should go to the BUSY state. Your circuit should output '`BUSY`' to *statusMsg* output for 10 clock cycles. Then your circuit should go 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 should be "`CALLER `". Also, *sentMsg* output should be 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 shown 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 |

Note that only ASCII characters with decimal values between 32 and 126 are considered valid input characters. If the caller enters and sends an invalid character, your circuit should ignore that character (do not include this input in cost calculation). If the caller sends the ASCI character with decimal value 127 (DEL) (include 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. Your circuit should calculate total cost of a call. When the call is ended, total cost of the conversation will be sent as an output to *sentMsg* output in hexadecimal and *statusMsg* output should be "`COST `" for 5 clock cycles. For example, a call with a cost of 55 Krş should be shown on *sentMsg* as "`00000037`" for 5 clock cycles. Then, your circuit should go to IDLE state.