TEAM- DIGICREW - EC01

Introduction:

Morse code is a character encoding scheme used in telecommunication that encodes text characters as standardized sequences of two different signal durations called dots and dashes.

Morse code representation:
DOT is represented with 1
DASH is represented with 111
Space between parts of same letter is represented with 0
The space between letters is represented with 000
The space between words is represented with 0000000

Reference:
https://upload.wikimedia.org/wikipedia/commons/b/b5/International_Morse_Code.svg

As a part of Hachthon under problem statement EC01 our Team - Digicrew has to implement Morse code to ASCII code.

Proposed Method:

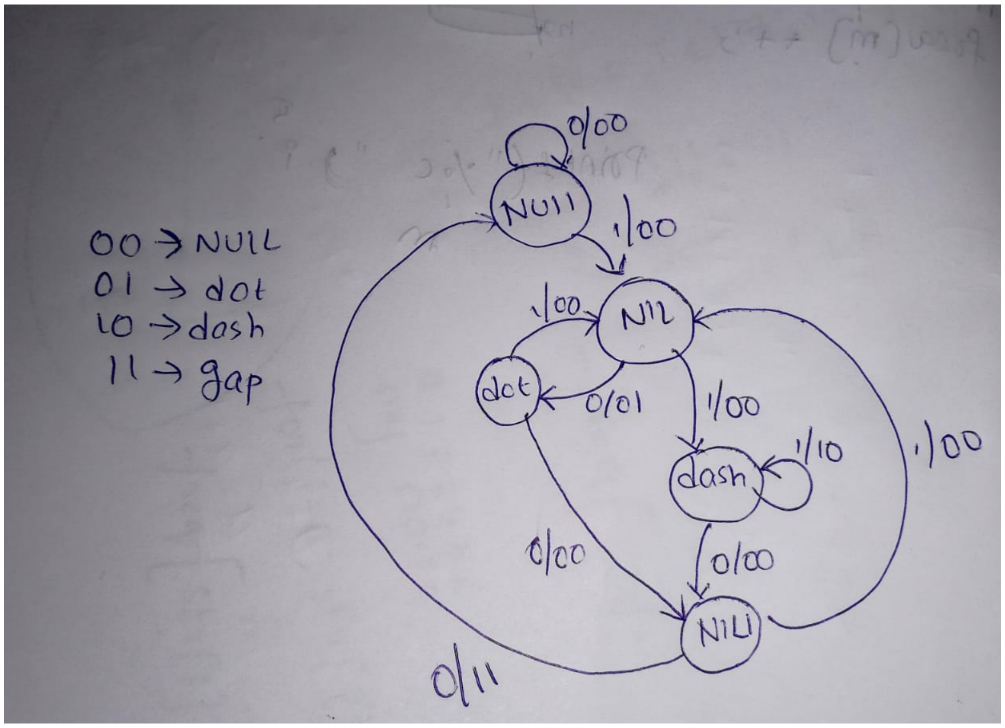
Our approach includes 2 steps. the 1st step is to detect the DOT,DASH,GAP. We used Mealy FSM for this purpose. Input to this fsm is from a single line data stream of morse code. The 2nd step is to decode the Morse code to the corresponding ASCII number. 1st step fsm outputs are inputs to 2nd step decoder. We used a 2hz clock for the entire circuit. Asynchronous Reset has been provided as input to above 2 steps circuits to ensure the circuit reaches known initial state when given signal ,when any erroneous output occurs.
Outputs of 1st step fsm is : DOT - 01
DASH - 10
GAP - 11
NULL - 00.

Output of 2nd step decoder is : 8bit ASCII number in HEXADECIMAL format. We used LOGISIM software to implement the circuit we designed.

Work Done and Results :

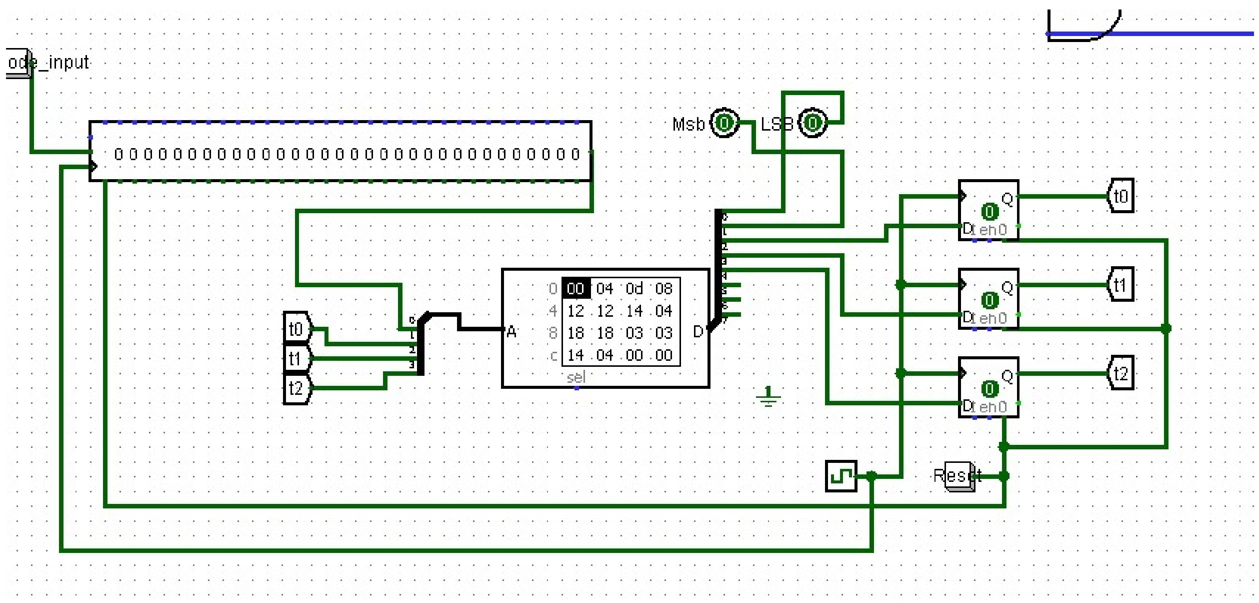
The 1st step FSM - DOT_DASH_GAP_DETECT :

Diagram :

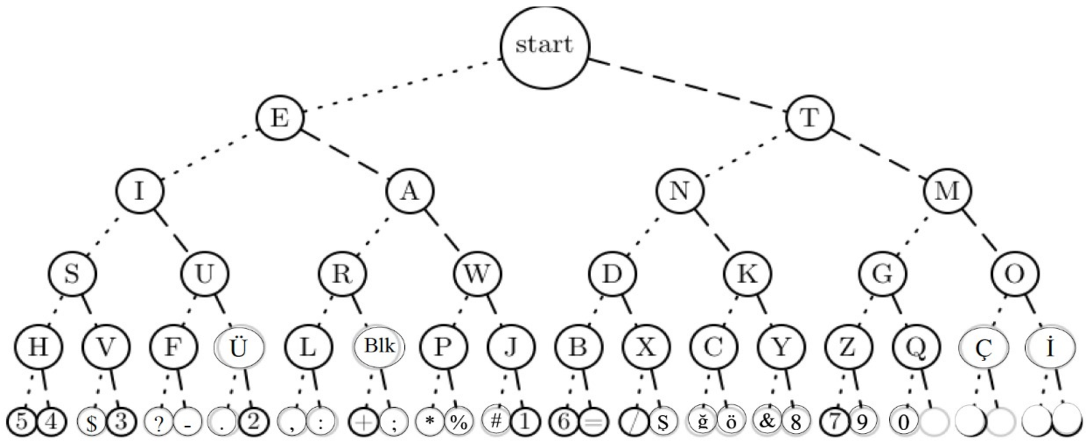


We implemented the above state diagram using ROM.
Corresponding present states and input is an input to ROM and respective 2bit outputs and states are output from ROM.

The circuit for the above description of step1 is :



The 2nd step FSM - MORSE_CODE_DECODER :



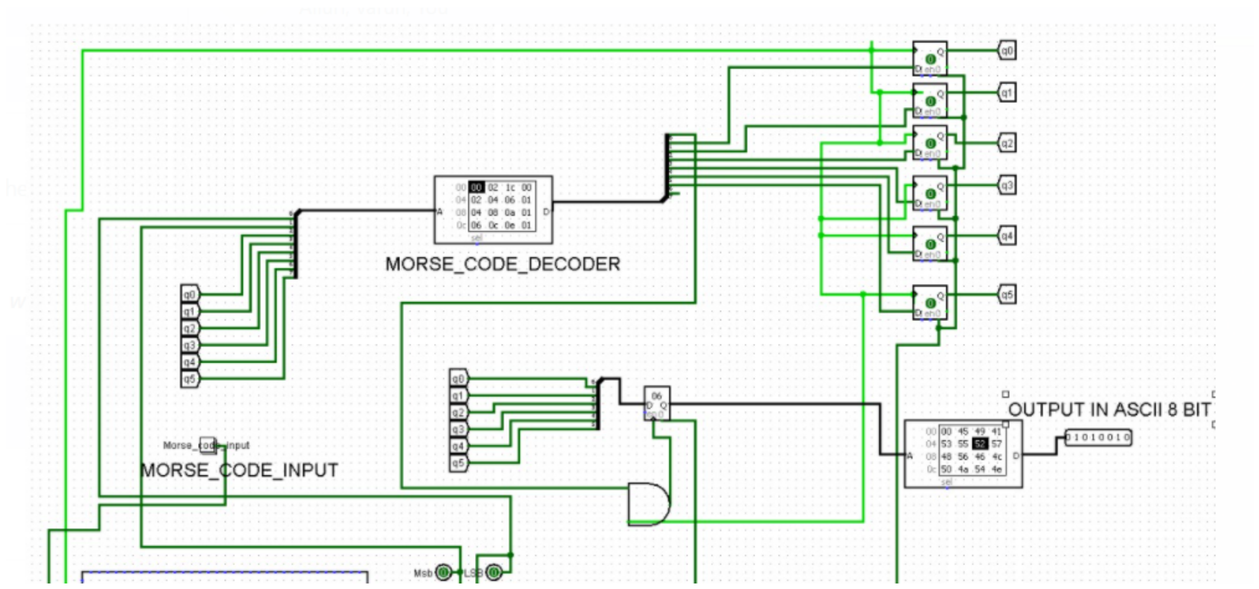
We have used the above Binary Tree as reference for our FSM for this step. We have designed fsm for detecting ALPHABETS. The initial state is idle

state. So the total number of states is 26+1. Number of inputs to this fsm is 2bits from the output of step1 fsm and 26 states (present states) . We used 6 Flip Flops for the implementation.
The Left link is taken as DOTS. That means if input is DOT(01) when state is in START/IDLE position the next state will be E, if the input is DASH(10) when state is in START/IDLE position the next state will be T. If the input is GAP(11) it goes to IDLE state leaving output_enable as 1. If the input is NULL(00) it stays in the same state. And the fsm design goes on until it reaches the last but one level in the above picture.

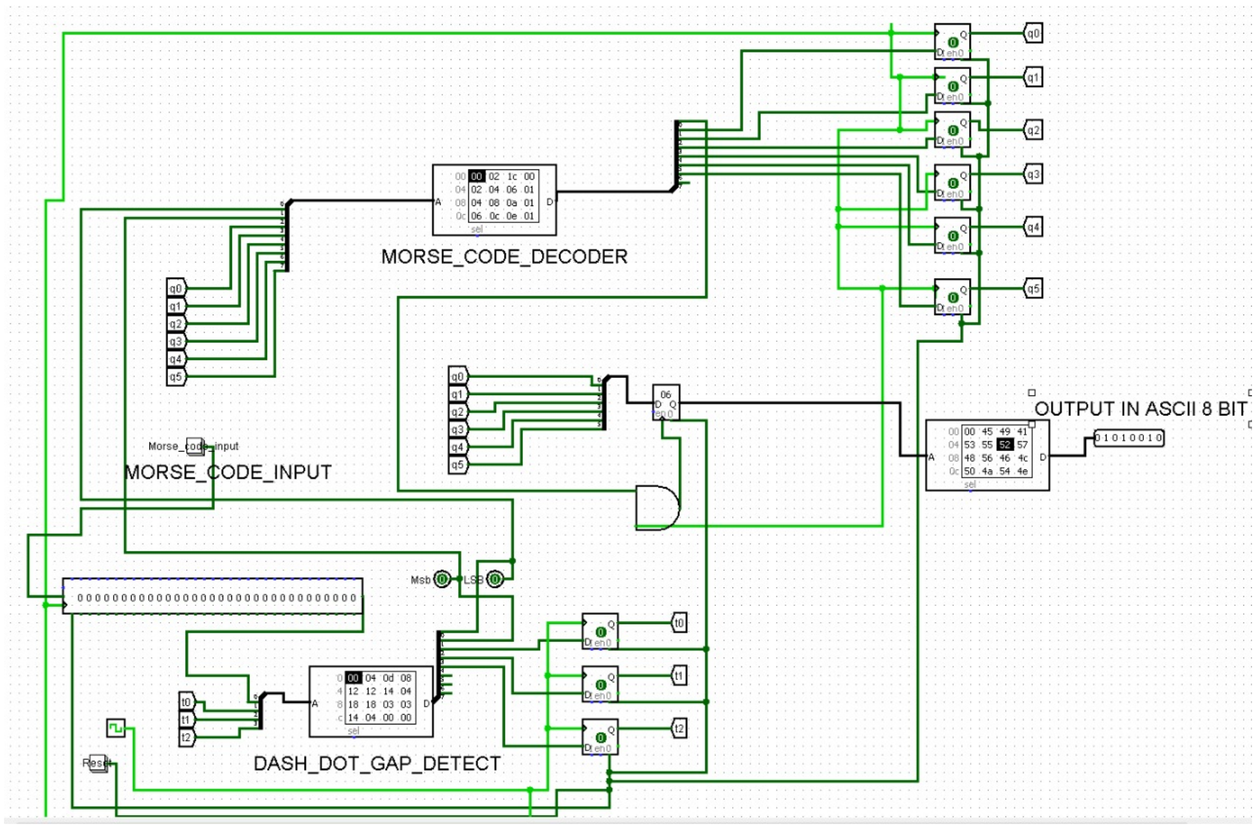
As soon as output_enable is 1 the corresponding state is stored in registers and is given as input to another ROM which give ASCII number of respective state ALPHABET.

The complete FSM table for above picture can be seen from below link in "copy of sheet2":
morse
We used ROM to implement the above fsm.

The circuit of above step2 description:

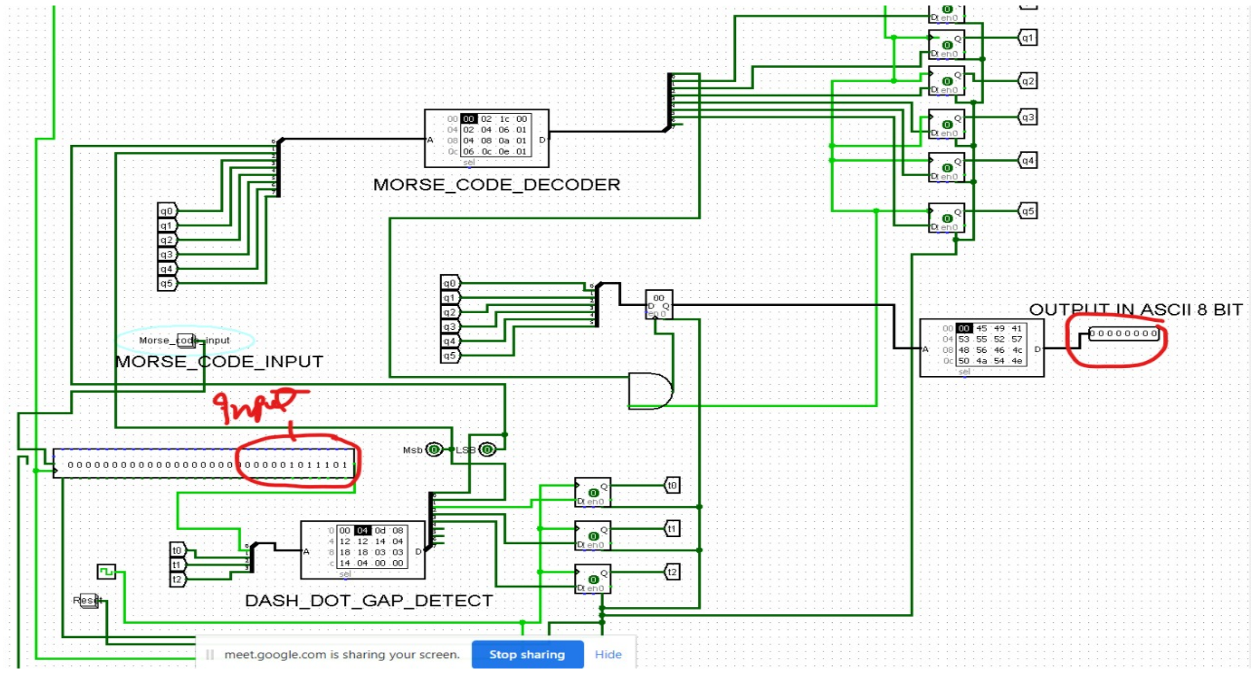


Total circuit combining 2 steps is :

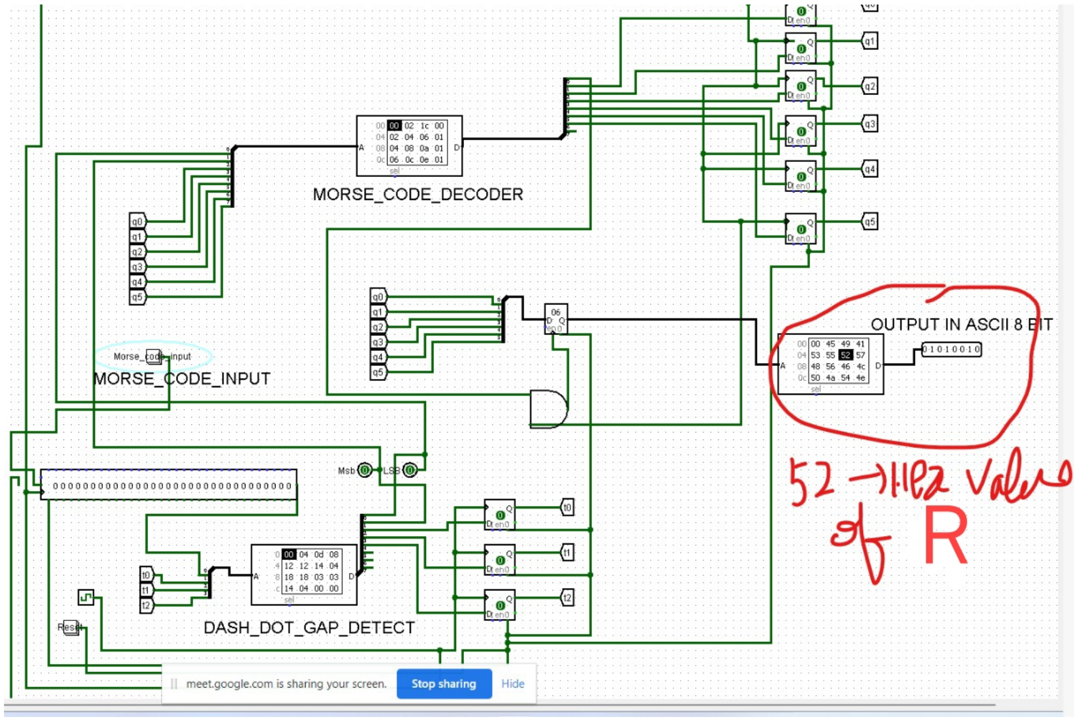


<u>Results:</u>

Input provided is 1011101 which means R.



Output is :



Conclusion:

Morse code to ASCII number circuit is successfully designed and implemented using LOGISIM software. We have tested our circuit 1st by giving letters morse code and after getting successful results we went further by giving morse code of word ELECTRONICS and we successfully got corresponding ASCII numbers in display.