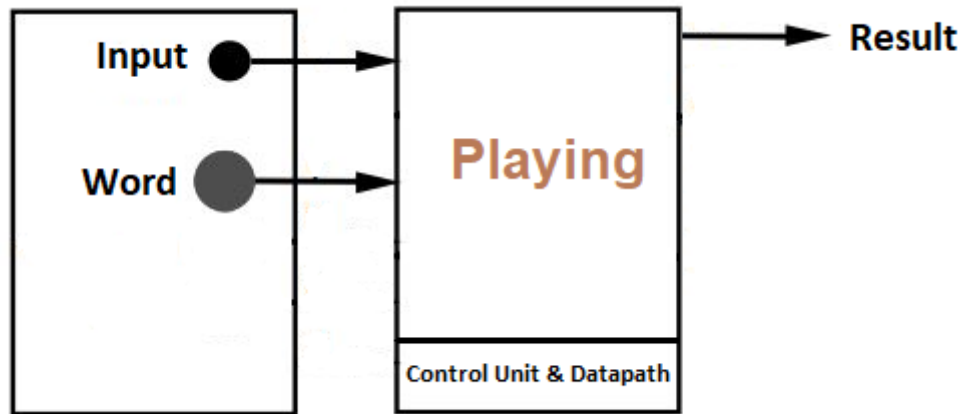


**GTU Department of Computer
Engineering CSE 232 - Spring 2020
Final Project Report**

**Akif KARTAL
171044098**

Problem Definition

The problem is to implement a modified hangman game machine with Logisim. A basic diagram to show the problem is;



Inputs: Start (Button), Random Word(1-bit number), Word(64-bit input), Letter(Buttons).

Outputs: Number of characters in the word (on 7-segment display), Hanging man(LED matrix), Found letters in the word (using a TTY).

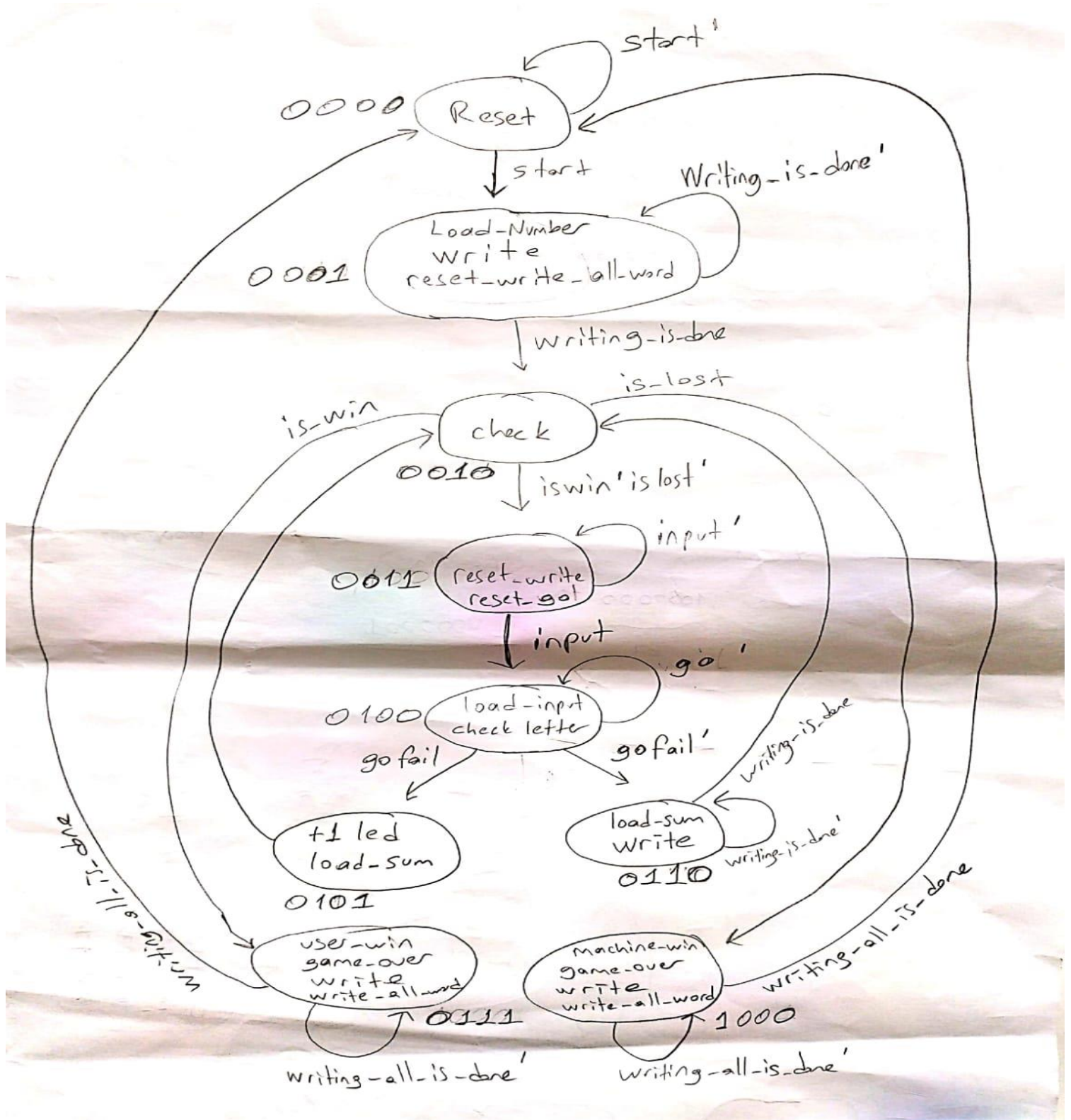
C language version of problem;

- Check hangman.c file to see the C code version of the problem.

Solution Step by Step

1) Decide states and draw the state diagram for your FSM controller.

State Diagram with values;



Encoding → Reset=0000, Initialize=0001, Check=0010, Get Input=0011, Determine Fail or not = 0100, Increment Led number= 0101, Write found letter= 0110, User win = 0111, Machine win= 1000

2) Draw datapath.

Modules in Datapath

- **Select word**
 - This module determines where will the word load? In other words from outside or ROM. If random input is 1 word will load from ROM, otherwise it will load from outside.
- **Check word**
 - This module determines which letter are okay for 64-bit 8 letter word. If letter ascii value is between 97 and 122 then this letter is good.
- **Get Choose**
 - This module gets an input letter from user.
- **is_in**
 - This module checks given input is in the word or not .
- **Hanging man**
 - This module draw the man on LED matrix according to faults that user made.
- **is_win**
 - This module determine the user is found all letters in the word.
- **Write word**
 - This module writes the word tty according to situation of the game by using line feed(a) end of text(3) and *(2a)
- **Datapath**
 - This module uses all other modules to merge all modules.

➤ Please check modules in the 171044098.circ file.

3) Draw truth table.

Encode states

Encoding → Reset=0000, Initialize=0001, Check=0010, Get Input=0011, Determine Fail or not = 0100, Increment Led number= 0101, Write found letter= 0110, User win = 0111, Machine win= 1000

Truth Table

inputs													outputs											
s3	s2	s1	s0	start	Is win	Is lost	fail	go	input	writing is_ done	Writing all_is done		load_ number	write	Write reset	load input	increment led	load sum	User win	Machine win	Game over	n3	n2	n1
0	0	0	0	0	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	x	x	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	x	x	x	x	x	x	0	x	0	1	1	0	0	0	0	0	0	0	0	0	1
0	0	0	1	x	x	x	x	x	x	1	x	0	1	1	0	0	0	0	0	0	0	0	1	0
0	0	1	0	x	1	0	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	1	1	1
0	0	1	0	x	0	1	x	x	x	x	x	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	x	0	0	x	x	x	x	x	0	0	0	0	0	0	0	0	0	0	0	1	1
0	0	1	1	x	x	x	x	x	0	x	x	0	0	1	0	0	0	0	0	0	0	0	1	1
0	0	1	1	x	x	x	x	x	1	x	x	0	0	1	0	0	0	0	0	0	0	1	0	0
0	1	0	0	x	x	x	x	0	x	x	x	0	0	0	1	0	0	0	0	0	0	1	0	0
0	1	0	0	x	x	x	0	1	x	x	x	0	0	0	1	0	0	0	0	0	0	1	1	0
0	1	0	0	x	x	x	1	1	x	x	x	0	0	0	1	0	0	0	0	0	0	1	0	1
0	1	0	1	x	x	x	x	x	x	x	x	0	0	0	0	1	1	0	0	0	0	0	1	0
0	1	1	0	x	x	x	x	x	x	1	x	0	0	1	0	0	1	0	0	0	0	0	1	0
0	1	1	0	x	x	x	x	x	x	0	x	0	0	1	0	0	1	0	0	0	0	1	1	0
0	1	1	1	x	x	x	x	x	x	x	1	0	0	1	0	0	0	1	0	1	0	0	0	0
0	1	1	1	x	x	x	x	x	x	x	0	0	1	0	0	0	0	1	0	1	0	1	1	1
1	0	0	0	x	x	x	x	x	x	x	1	0	1	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	x	x	x	x	x	x	x	0	0	1	0	0	0	0	0	1	1	1	0	0	0

4) Derive Boolean expressions from the truth table.

load - number

$S_3'S_2'S_1'S_0$

Increment led

$S_3'S_2'S_1'S_0$

load sum

$S_3'S_2'S_1'S_0 + S_3'S_2'S_1'S_0$

User-win

$S_3'S_2'S_1'S_0$

machine win

$S_3'S_2'S_1'S_0$

Game over

$S_3'S_2'S_1'S_0 + S_3'S_2'S_1'S_0$

$S_3'S_2$	00	01	11	10
$S_1'S_0$	00			1
01	1			
11		1		
10		1		

Write

$S_3'S_2'S_1 + S_3'S_2'S_1'S_0 + S_3'S_2'S_1'S_0$

Write-read

$S_3'S_2'S_1'S_0$

load-input

$S_3'S_2'S_1'S_0$

Other outputs:

```
n3 = (! s3 && ! s2 && s1 && ! s0 && ! is_win && ! is_lost) || (s3 && !s2 && !s1 && !s0 && !
writing_all_is_done)
```

```
n2 = (! s3 && s2 && s1 && s0 && ! writing_all_is_done) || (! s3 && s2 && ! s1 && ! s0) || (! s3 &&
s2 && ! s0 && ! writing_is_done) || (! s3 && ! s2 && s1 && s0 && input) || (! s3 && ! s2 && s1 &&
! s0 && is_win && ! is_lost)
```

```
n1 = (! s3 && s2 && s1 && ! s0) || (! s3 && s2 && ! s1 && s0) || (! s3 && s2 && s0 && !
writing_all_is_done) || (! s3 && s2 && ! s0 && ! fail && go) || (! s3 && ! s2 && s1 && s0 && !
input) || (! s3 && s1 && ! s0 && ! is_lost) || (! s3 && ! s1 && s0 && writing_is_done)
```

```
n0 = (! s3 && s2 && s1 && s0 && ! writing_all_is_done) || (! s3 && s2 && ! s1 && ! s0 && go && fail) || (!
s3 && ! s2 && s1 && s0 && ! input) || (! s3 && ! s2 && s1 && ! s0 && ! lost) || (! s3 && ! s2 && ! s1 && s0
&& ! writing_is_done) || (! s3 && ! s2 && ! s1 && ! s0 && start)
```

5) Draw the circuit on Logisim.

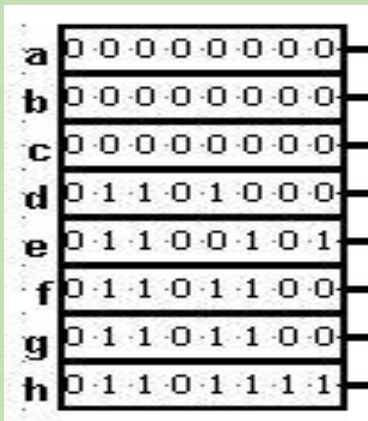
➤ Check **main** circuit in **171044098.circ** file to see circuit.

6) Simulate and see whether it works.

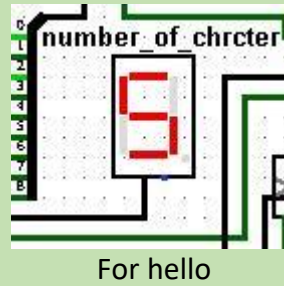
❖ This machine can perform the all the minimum requirements and all **bonus parts**.

Some Information;

- ✓ To execute the system get clock tick frequency to be 64 hz.
- ✓ To use random word from ROM make random word input to be 1, otherwise enter your word.
- ✓ To start game click start game button after game is start you can not start new game until this game ends after game is over you can start new game with start button. (6th rule from project pdf file) .
- Check following test results.
 - Minimum Requirements

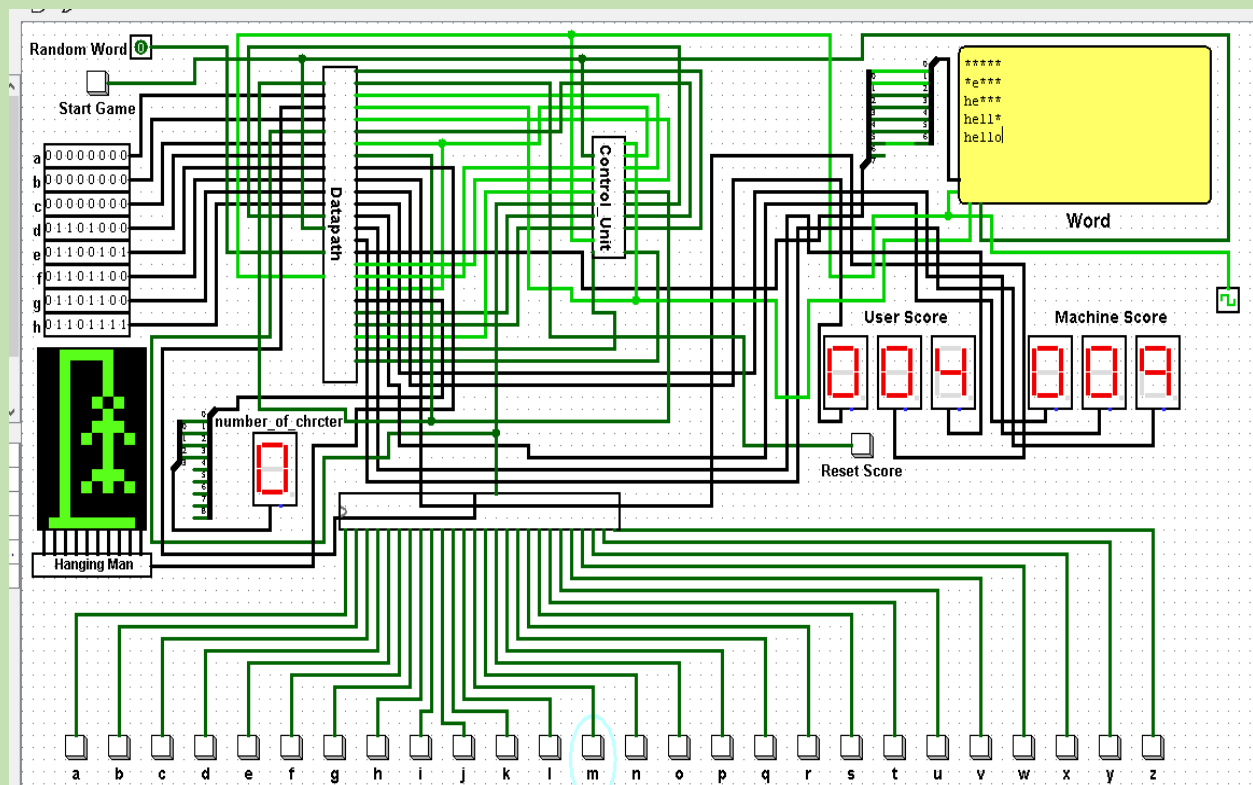
Description	Test Result
<p>1. The word is taken as 64-bit input.</p> <p>2. Each 8 bit refers to the ascii code of a character. Therefore, the word is 8 characters at most.</p>	 <p>hello word</p>

3. The machine will show the number of characters in the word on 7-segment display.

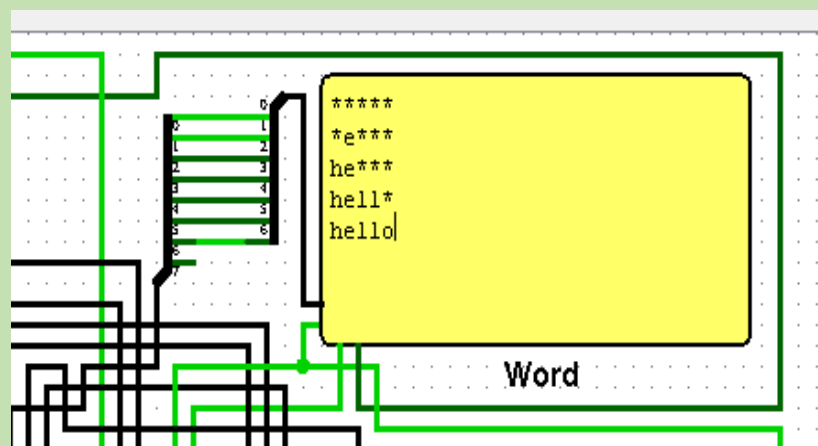


For hello

4. The user selects a character by pushing a button, which means that for each character there is a different button input for the machine. If the selected character is absent in the word a LED is turned ON. If ten LEDs are turned ON the user loses, otherwise if the user can find all letters in the word, he wins.

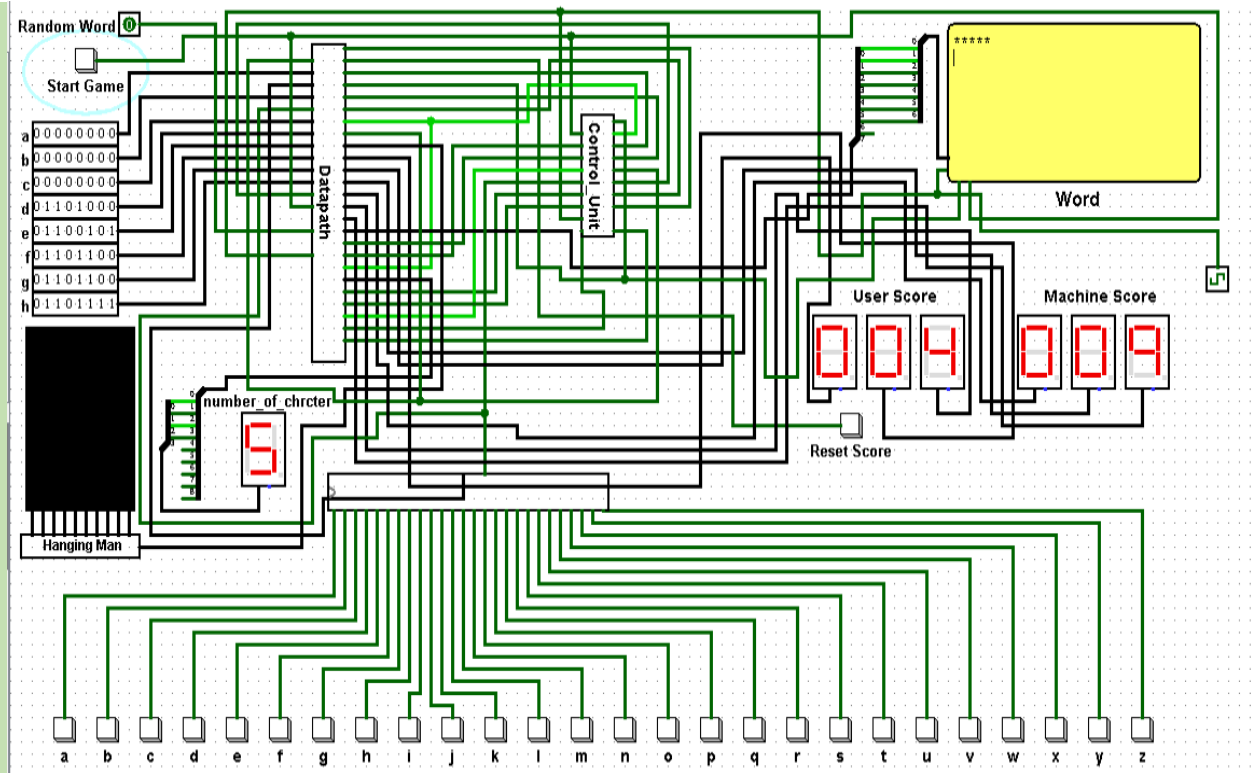


5. The machine will show the found letters in the word using a TTY display in Logisim. It also shows the whole word at the end even if the user loses.



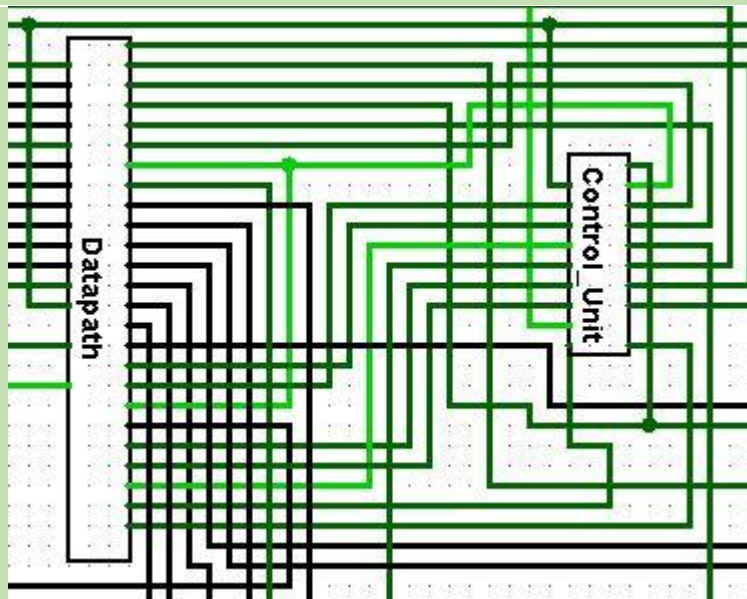
User is lost

6. When the game ends the user can restart a new game with another button.



Start button clicked again after game is end.

7. If you build your machine without FSM and sequential design and instead you perform only combinational design.



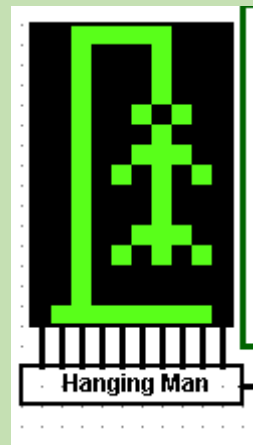
(Control Unit(FSM) and Datapath together not just datapath)

- **Bonus Parts**

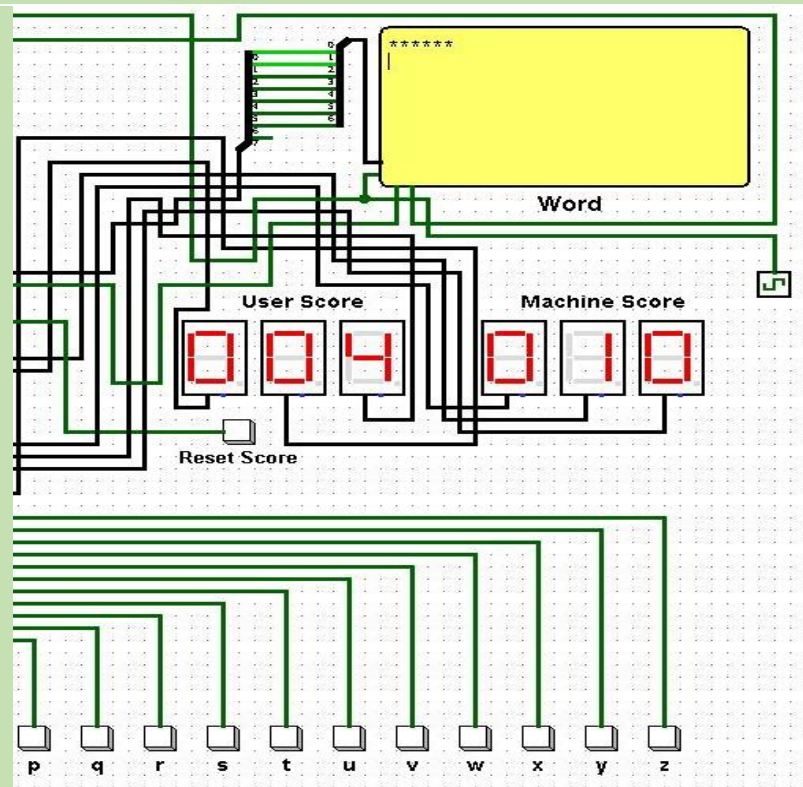
Description

Test Result

1. You can show a hanging man instead of turning ON leds by using LED matrix in Logisim.



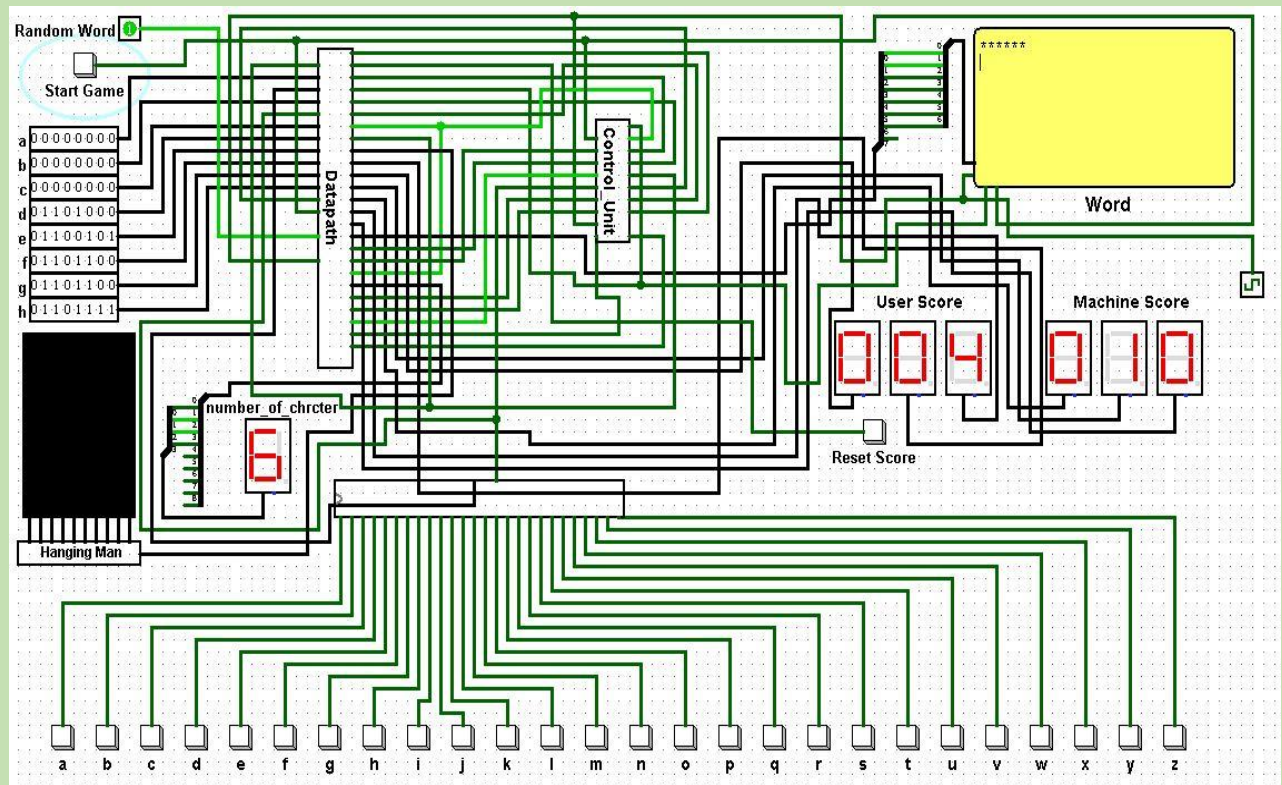
2. You can remember the number of times the user won and show that on scoreboard using 7-segment displays for the scoreboard. Show the score as user vs machine.



3. You can implement a word memory using 128 64-bit registers (two 128x32-bit ROM in Logisim) each

holding a different word and select one of them randomly so that you do not need to take a word as input. Your circuit should also be able to take word from outside

Random input is 1



Random word Module in datapath

