ES 232 : Digital Systems and Microcontrollers project

Team Members: Mandlem Manikanta (15110070), Pansetty Karthik (15110082),

Veeramallu Giridhar Sai (15110144)

Project Title: Morse Code Detection (using FPGA)

Description of the Project:

Morse code comprises of a series of dots and dashes corresponding to a particular character of the alphanumeric characters (alphabets and digits). In this project, the Morse code is inputted using push buttons present on the FPGA. Generally in Morse code, a dot corresponds to one unit and a dash corresponds to three units, space between parts of the same letter is one unit, space between letters is three units and space between different words is seven units. The spaces correspond to low voltage value (or 0 in binary) and the dots and dashes correspond to high voltage value (or 1 in binary).

The basic idea is to set a particular range of time period to interpret the signal as a dot, a dash or different kinds of spaces. At the instant when we start to push the button (positive edge) or leave the button (negative edge), a new clock will start and the time period of each clock will determine the sequence of dots and dashes. Using an asynchronous FSM, we detect the input sequence and the output can be generated using the FPGA and the character(s) can be displayed on the LCD Screen/ 7-Segment display.

Distribution of Work:

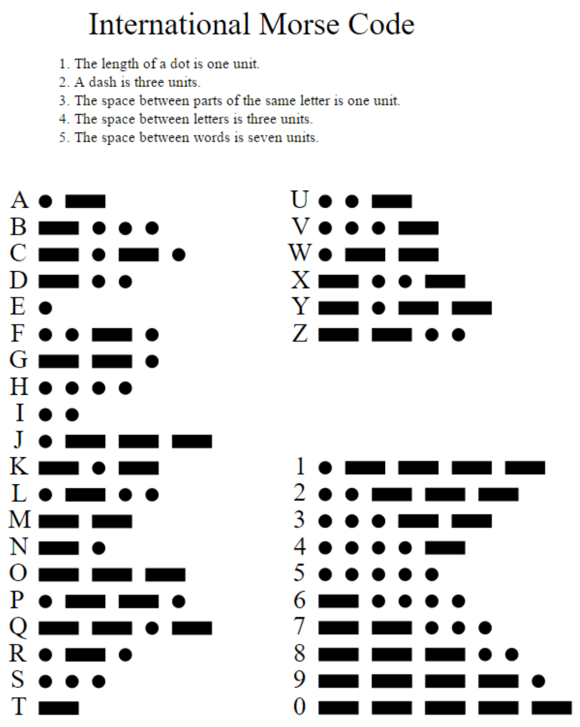
There are broadly two parts to this project: 1) Designing the FSM and 2) Coding the FSM using Verilog and implementing on the FPGA. Since there needs to be collective work done in both the parts, it cannot be distributed in a discrete way. So, all of us will be involved in all the aspects of the project.

Final Project Demonstration:

In the final working model of the Morse code detector, the user can give the input using the push buttons present on the FPGA. The output on the LCD Screen/ 7-Segment Display will show what is being inputted through the push buttons.

Introduction:

Morse code is one of the methods of transmitting information using dots and dashes. It is generally used in transmitting information through audio. Each character is associated with a series of dots and dashes. According to the International Standard Morse Code, the length of a dot is one unit and the length of a dash is three units. The space between parts of the same letter is one unit and the space between different letters is three units.



International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

```
—— ——— ·—· ··· ·      —·—· ——— —·· ·
M   O   R   S  E       C    O   D  E
```

This is one example of how the text “MORSE CODE” can be written in Morse code.

Source: Wikipedia

Motivation:

One of the places where Morse code was radio communication in the army in the olden days. This radio communication is done by sending short and long tone pulses. Radiotelegraphy using Morse code was very vital during the Second World War and it was very effective in communication. It was also one of the main modes of marine communication. It can also be used by people with disabilities to communicate. One important application now-a-days is to signal for help using SOS which is a distress signal. As, this has so much so much importance, we have chosen this field to explore and understand one of the oldest ways of communication which is of importance even today and to convert it into a digital form.

Aim:

To design and implement a Morse code detector using an FPGA and to display the transmitted characters on a 7-segment display. The inputs to the FPGA are taken through the push button. The FPGA is to be programmed by using Verilog.

Theory:

Our basic idea was to give the input of the Morse code through a single push button and detect it in the FPGA and give the corresponding output to a 7-segment display. In detail, we created two FSM's. One in the transmitter and another in the receiver. The FSM in transmitter was used to detect the dashes, dots and spaces given through the push button as inputs. The output of first FSM consisting of dashes, dots and spaces was fed into the second FSM in the receiver which maps the dots and dashes to the corresponding characters and those characters are displayed on the 7-segment display.

Working:

Clock:

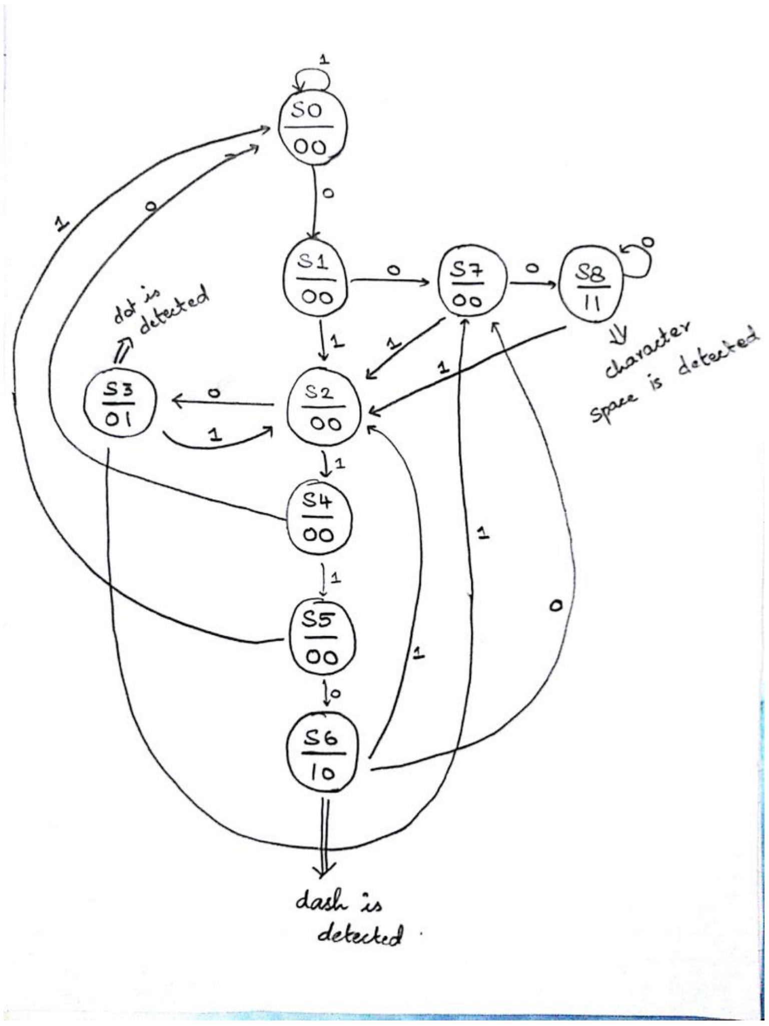
Using the 50MHz clock present in the FPGA, we used a clock divider and created a 1Hz clock. We used a counter and a comparator. The counter counts up to 25,000,000 and resets to zero. This count is compared with 24,999,999 using a comparator which gives output '1' when both the numbers are equal. The output from the comparator is fed to a flip flop which toggles its output for every half second. This will create a 1Hz clock.

FSM in the transmitter:

In this FSM, the inputs are given as 0 or 1. This is a Moore FSM. The input is taken into the FPGA at every positive edge triggered clock pulse. Depending on the timing of the button being pressed or not pressed, the input is taken as a dot, dash, or a character space. If a single '1' preceded and succeeded by a zero (010) are detected, then a dot is recorded. If three '1's preceded and succeeded by a zero (01110) are detected, then a dash is recorded. The output of the FSM is of two bit and are as given below:

00:	space in between dashes and dots in a single letter (when input is 0)
01:	dot (when input is 010)
10:	dash (when input is 01110)
11:	character space (when input is 000)

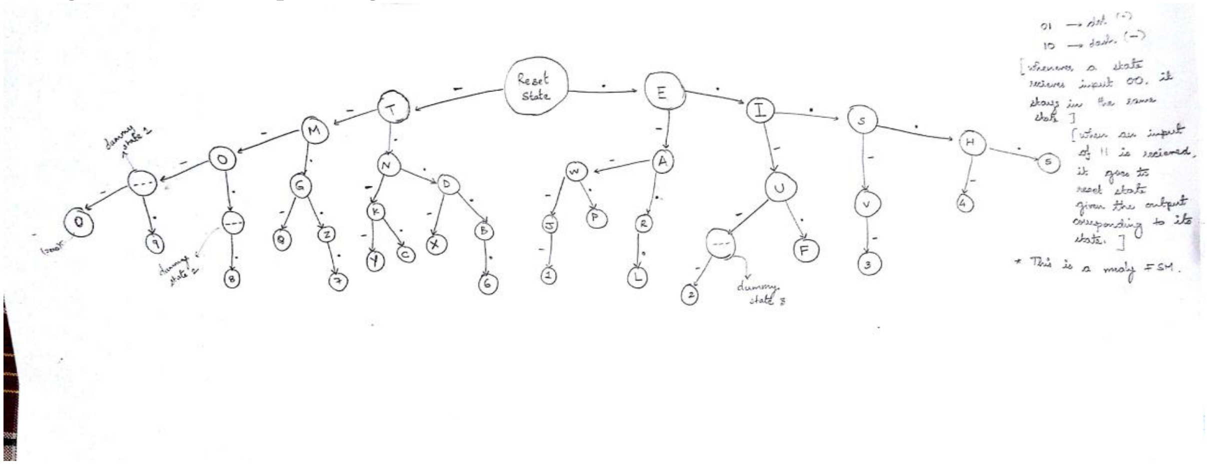
This 2-bit output is transmitted and then is received by the receiver.



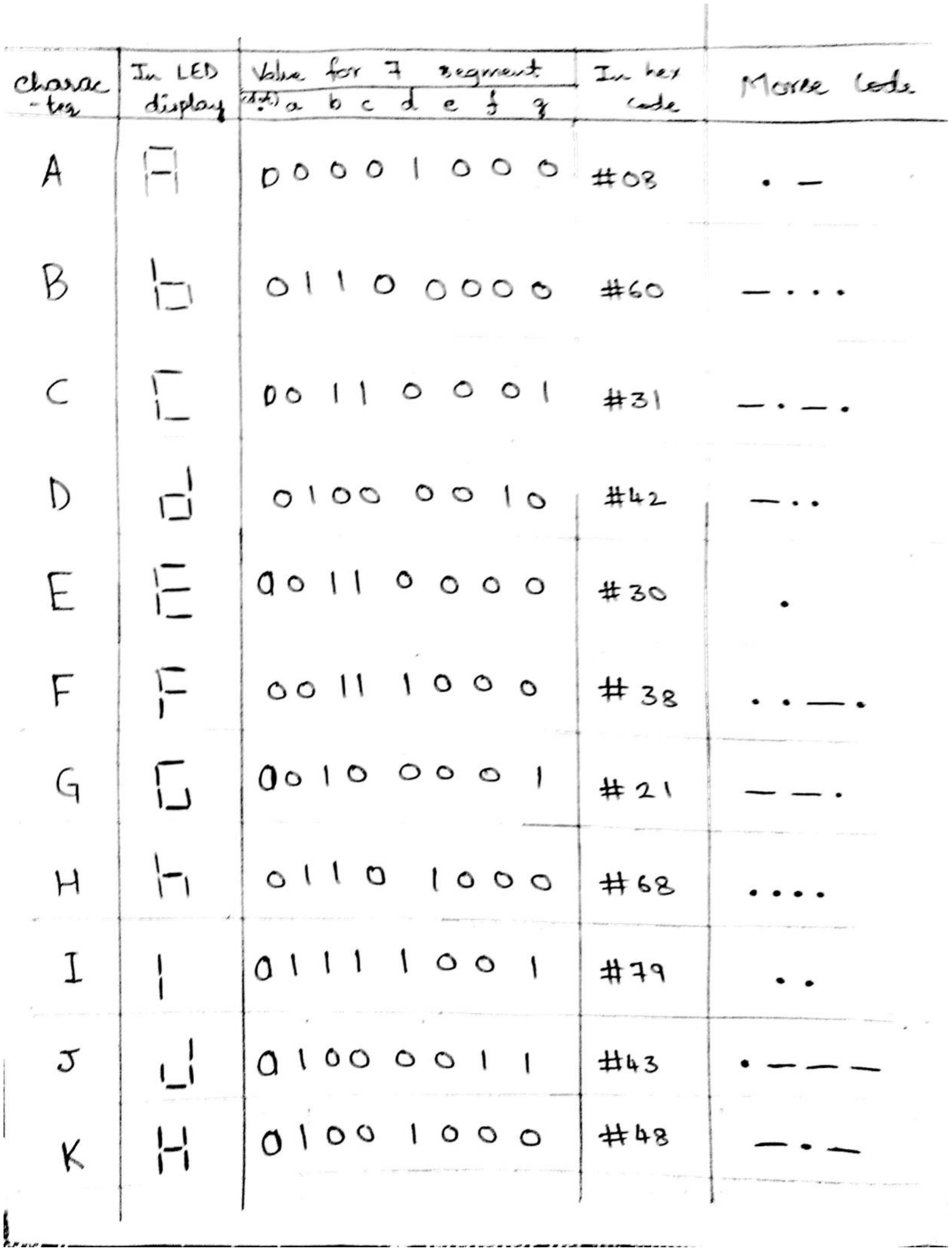
State diagram of the FSM present in the transmitter.

FSM in the receiver:

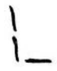
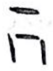
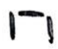
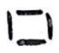
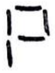
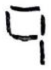
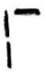
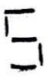
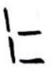
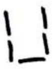
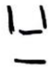
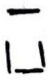
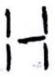
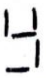
In this FSM the inputs are 2-bit inputs which are the outputs of the previous FSM. This is Mealy FSM. In this FSM also, input is taken into the FPGA at every positive edge triggered clock pulse. The main purpose of this FSM is to detect the series of dashes and dots and generate the corresponding output. And when a character space is received, the output is changed to the corresponding of the state and the state is set to reset.


Reset State
dummy state 1
dummy state 2
dummy state 3
01 → dot (·)
10 → dash (–)
[whenever a state recieves input 00, it stays in the same state]
[when an input of 11 is recieved, it goes to reset state given the output corresponding to its state.]
* This is a mealy FSM.

State diagram of the FSM present in the transmitter.



Charac-ter	In LED display	Value for 7 segment (d.p.)	a	b	c	d	e	f	g	In hex code	Morse code
A	A	0	0	0	0	1	0	0	0	#08	· —
B	b	0	1	1	0	0	0	0	0	#60	— · · ·
C	C	0	0	1	1	0	0	0	1	#31	— · — ·
D	d	0	1	0	0	0	0	1	0	#42	— · ·
E	E	0	0	1	1	0	0	0	0	#30	·
F	F	0	0	1	1	1	0	0	0	# 38	· · — ·
G	G	0	0	1	0	0	0	0	1	# 21	— — ·
H	h	0	1	1	0	1	0	0	0	#68	· · · ·
I	I	0	1	1	1	1	0	0	1	#79	· ·
J	J	0	1	0	0	0	0	1	1	#43	· — — —
K	H	0	1	0	0	1	0	0	0	#48	— · —

character	In LED Display	Value for 7 segment								In hex code	Morse code
		(dot) •	a	b	c	d	e	f	g		
L		0	1	1	1	0	0	0	1	#71	•—••
M		0	0	1	0	1	0	1	0	#2A	——
N		0	1	1	0	1	0	1	0	#6A	—•
O		0	1	1	0	0	0	1	0	#62	———
P		0	0	0	1	1	0	0	0	#18	•——•
Q		0	0	0	0	1	1	0	0	#0C	——•—
R		0	0	1	1	1	0	0	1	#39	•—•
S		0	0	1	0	0	1	1	0	#24	•••
T		0	1	1	1	0	0	0	0	#70	—
U		0	1	0	0	0	0	0	1	#41	••—
V		0	1	0	1	0	1	0	0	#54	•••—
W		0	0	1	0	0	0	1	1	#23	•——
X		0	1	0	0	1	0	0	0	#48	—••—
Y		0	1	0	0	0	1	0	0	#44	—•——

character	In LED Display	Value for 7 segments								In hex code	Morse code
		(dot) .	a	b	c	d	e	f	g		
Z	2	0	0	0	1	0	0	1	0	#12	— — · ·
0	0	0	0	0	0	0	0	0	1	#01	— — — — —
1	1	0	1	0	0	1	1	1	1	#4F	· — — — —
2	2	0	0	0	1	0	0	1	0	#12	· · — — —
3	3	0	0	0	0	0	1	1	0	# 06	· · · — —
4	4	0	1	0	0	1	1	0	0	#4C	· · · · —
5	5	0	0	1	0	0	1	0	0	# 24	· · · · ·
6	6	0	0	1	0	0	0	0	0	# 20	— · · · ·
7	7	0	0	0	0	1	1	1	1	# 0F	— — · · ·
8	8	0	0	0	0	0	0	0	0	# 00	— — — · ·
9	9	0	0	0	0	0	1	0	0	#04	— — — — ·