Final Project

Morse Code Interpreter

Preston Call

12/17/2025

ECEN 340

Report

Description of project

We created a Morse Code Interpreter, with send and receive half-duplex communication to another FPGA board. As mentioned, this is designed for an FPGA. Originally it was just an interpreter that could interpret sequences of button presses as letters, without external communication (See Figure 8).

This project interprets button presses by comparing the edge of the button press to the edge of a clock. If the debounced button is pressed for less than 200 milliseconds (ms), or 0.2 seconds, it is interpreted as a dit. If the debounced button is pressed for longer than 200 ms, or 0.2 seconds, it is interpreted as a dash. Whether dit or dash, determines which state the machine will progress to next. If brought to a state other than the initial state, and no button is pressed for 500 ms, or half a second, then an inter-letter space is sent. This means that whichever letter is associated with the current state will be submitted to the output seven segment display. Then the machine returns to the initial state, waiting for the next button press. A switch may also be flipped so if no button is pressed for 1600 ms, or 1.6 seconds, then an inter-word space is submitted and the seven segment display output is shifted one character to add a space.

These characters and spaces are communicated to the connected FPGA along with its own display. If the FPGA detects that the signal comes from the external connection, and previously came from itself, then it will reset the display before displaying the next characters. There is also an LED indicator to light up, informing the user when the other board begins sending. This is to prevent confusion about who is sending signals and to prevent data clashes.

Information about operation is on the GitHub link included in the Code Section.

Peer review notes

We were reviewed twice through the process of creating this system. We originally were going to create a calculator with a keypad. First our project idea was reviewed by Korbin Boren and Noah Barnett. Something they liked about our idea was learning to use a keyboard using hardware design, and also displaying the characters on a display as they are pressed. They also thought that implementing a keypad would be difficult. They made a few suggestions, the first being to run our ideas by AI to see what could be done in a simpler, cleaner way and find ways to fix issues we had run into. Ultimately, they suggested that we switch our project to a half-duplex morse code interpreter to dive deeper into a previous lab project.

We did decide to switch projects; this switch allowed us to explore a deeper and greater complexity of a previous lab which proved enlightening. After working on this for a while, we were reviewed again, this time by Alex Turner. He liked our idea, especially that it would be able to communicate in both directions across two FPGA boards. He liked that we followed previous feedback to switch projects so that we could dig deeper into one project instead of starting over with a new idea. We discussed potential marketing, for which we could market this as a kids toy to talk to friends or siblings in another room, all while learning morse code.

Conclusion

This was a great lab to work on; we both learned a lot through research along with trial and error. We successfully created a half-duplex morse code interpreter with two FPGAs and three wires, for send, receive, and a common ground wire. In this implementation, we utilized several technologies we learned about in class. We implemented a Finite State Machine to manage each character that the morse interpreter must be able to output. We used the 7-segment display to display characters when there was an inter-letter space event, and to display a space via shift when there was an inter-word space event. We used on-board buttons and switches as input. We used the three wires for a serial interface between the two FPGAs. It was difficult to get the morse interpreter to function with incorrect syntax which Vivado couldn't catch, leading to significant portions of code being optimized away, despite fully functional simulations. Ultimately, it is complete and works fully.

Figures

Some additional figures/documents are included in the GitHub link below. We also recorded a video demonstration of the functional interpreter at this link: https://drive.google.com/file/d/1BfA0fsZIqq6prWtjpTnf8c3Ep23pCCnE/view?usp=drivesdk

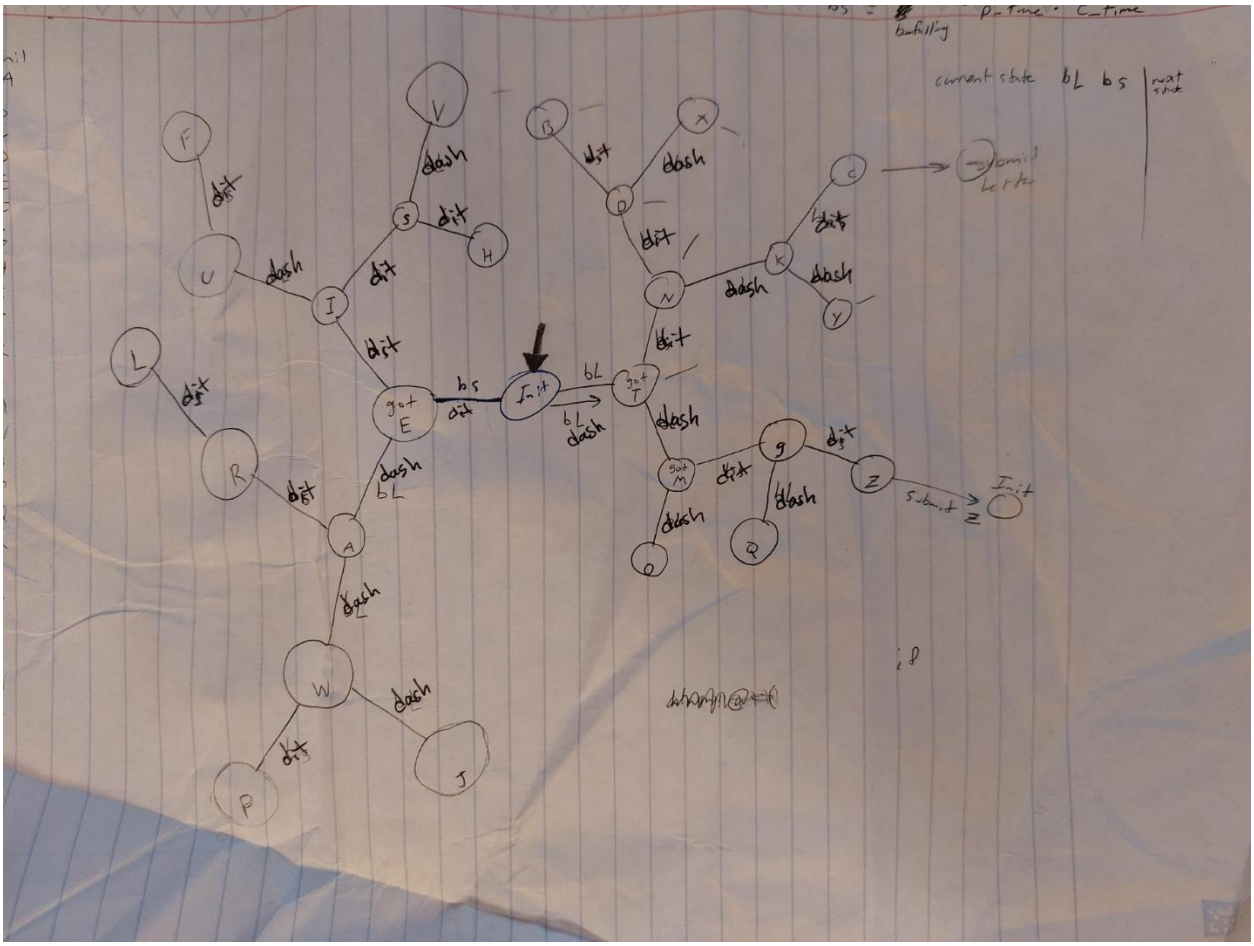

Figure 1) State diagram drawing of the morse code interpreter state machine.

ECEN – 340 FINAL LAB PEER REVIEW-1

Instructions

Each "team" meet with one other team and answer the following questions about the other team. Share one copy from your team's findings with the other team. One copy to Bro. Watson.

My Team Name (or Members) Korbin Boren, Noah Barnett

Other Team's Name (or Members) Preston call, Matthew spotten

1. Basic Description of the other Team's Project:
   A calculator with additional Keypad. Displays the answer with a Seven-Segment display

2. Areas of Learning from the other Team (One thing you like):
   how to use a keyboard and displaying characters

3. Potential Difficulties or Roadblocks you might see (if you were doing it):
   displaying characters and knowing how to use the keyboard

4. One thing the other team might consider to make it better/easier:
   Running things by AI to see what things could be fixed, simpler, and cleaner

Figure 2) Final Lab Peer Review – 1

ECEN – 340 FINAL LAB PEER REVIEW PART II.

Instructions

Each "team" meet with one other team and answer the following questions about the other team. Share one copy from your team's findings with the other team. One copy to Bro. Watson.

My Team Name (or Members) Alex Turner

Other Team's Name (or Members) Matthew & Preston

1. Basic Description of the other Team's Project:
   2way morse code interpreter.

2. Areas of Learning from the other Team (One thing you like):
   The fact that it is two way allowing others to communicate back and forth.

3. Ask the other team about one thing they corrected, due to feedback:
   Changeing their project as their morse code worked already so it was easier to change rather than start anew.

4. With your reviewer(s), think of at least one type of "product" your design could be used in
   a toy to teach kids morse code, communication in class or buisness

Figure 3) Final Lab Peer Review – 2

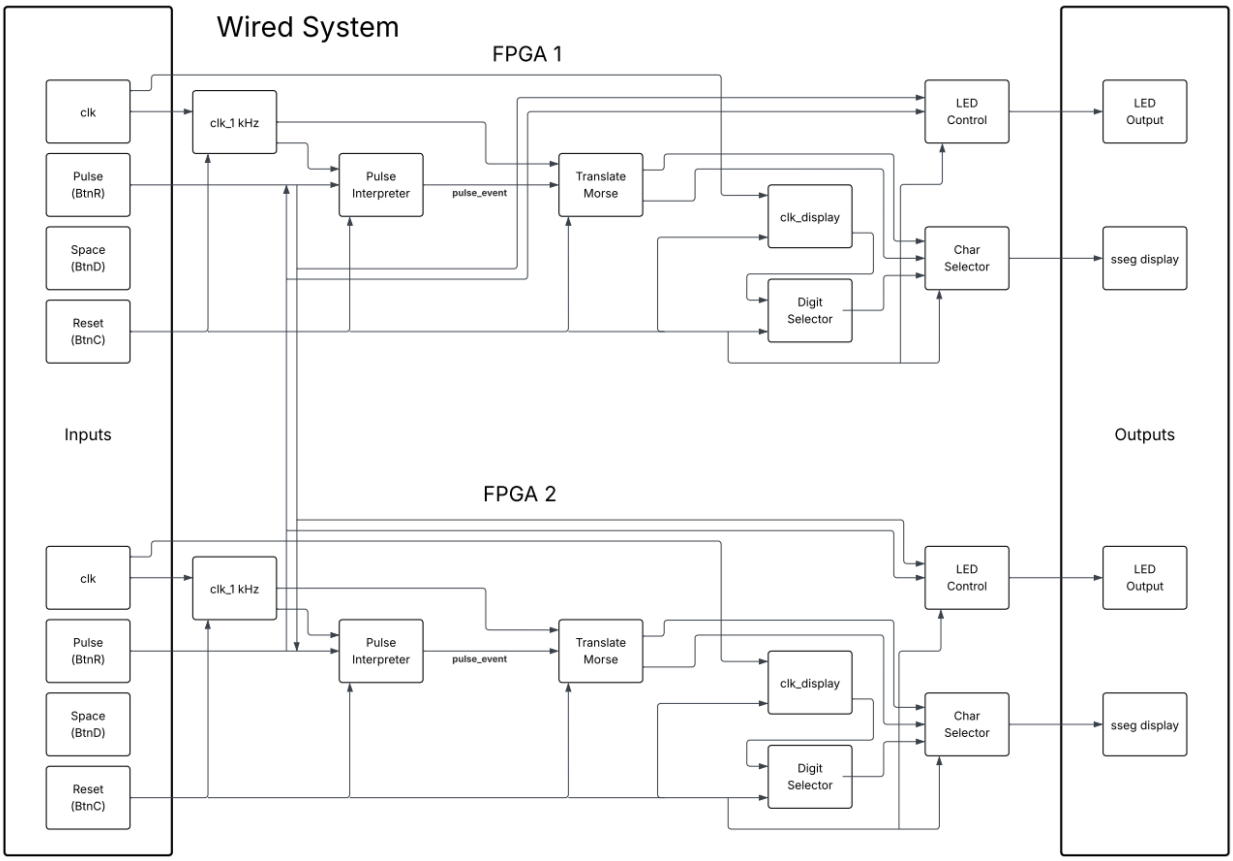

Figure 4) Block diagram of two FPGAs wired for serial half-duplex communication

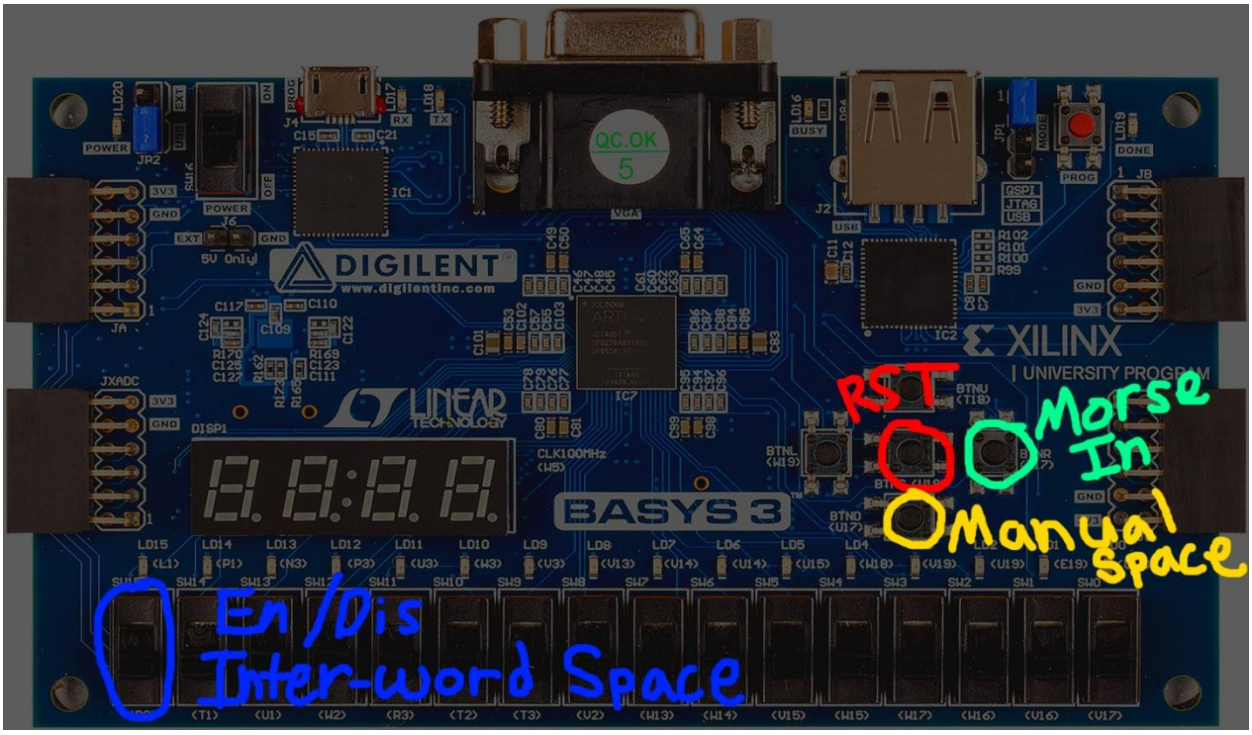

Figure 5) User guide to various inputs for morse code interpreter. Details to use below:

BUTTON PRESS TIMING:

Dit:	<	200ms	(dit if Morse In press time < 200ms)
Dash	>=	200ms	(dash if Morse In press time >= 200ms)
Inter-letter-space		500ms	(character is sent when nothing pressed for 500ms)
Inter-word-space		1600ms	(If Enabled: space is sent if nothing pressed for 1600ms)

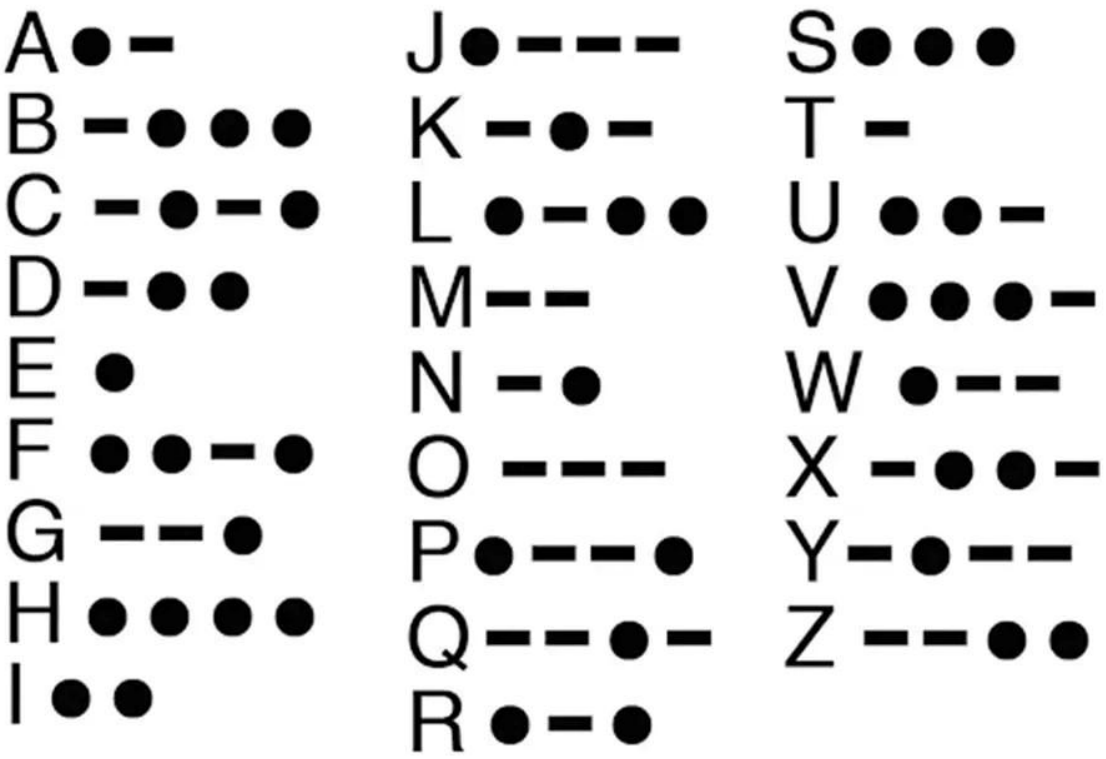

Figure 6) Morse encoder table.

Figure 7) Morse character decoder table, from 7-seg display to alphabet.

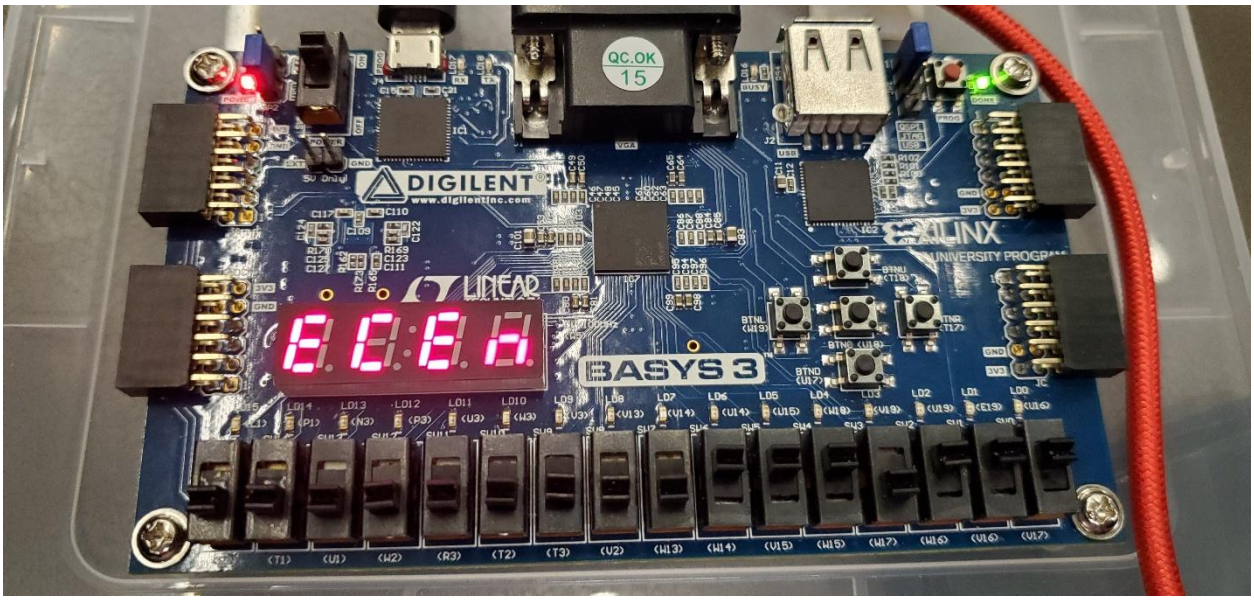

Figure 8) Successfully interpreting timed button presses into characters to display and communicate.

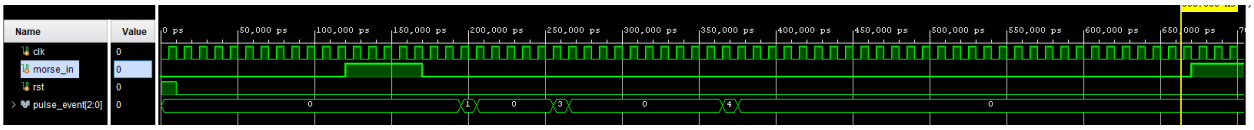

Figure 9) Test bench simulation of pulse_interpreter module. This module turns a button press into a dit or dash.

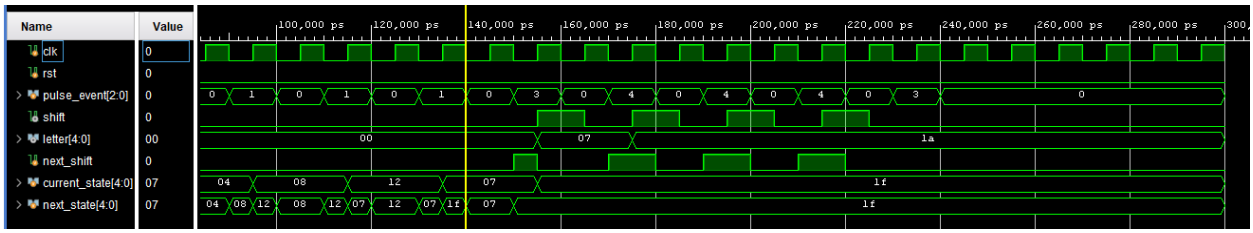

Figure 10) Simulation of translate_morse module. This shows four dits which output ‘H’

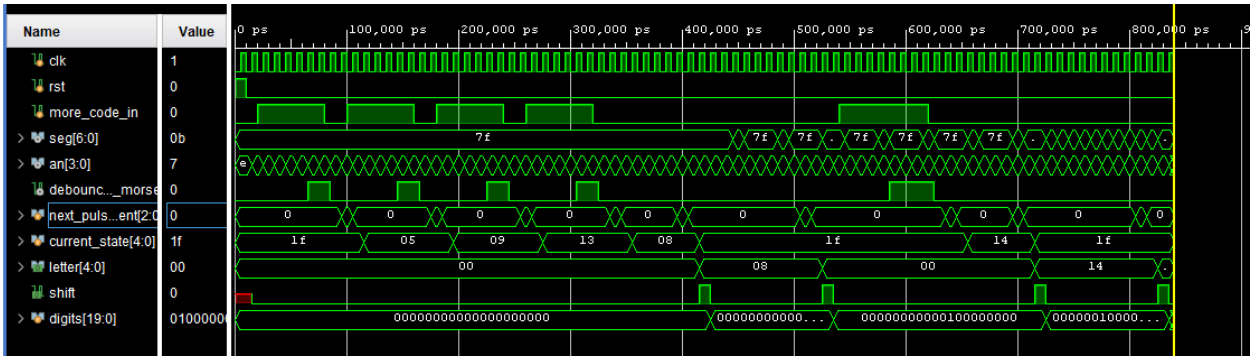

Figure 11) Simulation of the morse_code_interpreter_top. This example shows the display of another ‘H’, then shifts the output once, displays a ‘T’, and shifts the output again. “h_t_”

Code

Our system included a very large code base that can be viewed here on GitHub.

https://github.com/PCall1/ECEN340_Morse_Code_Interpreter/tree/main