Prepared by Group 14

Morse Code Translator

Nandana Alwarappan, Lauren Monahan, Aidan Born, Sterling Wodzro

Goal/Motivation

Translation

- Morse code input sequence of dots and dashes translated to ASCII text in real time

Readability & Easy Use

- Utilizes VGA screen to display text
- Individual buttons for each input to avoid user error

Communication

- Used in emergencies when traditional communication systems fail (radio, internet-based messaging)

• • • — — — • • •

Functionality

User input:

Four button inputs: dot, dash, enter, reset

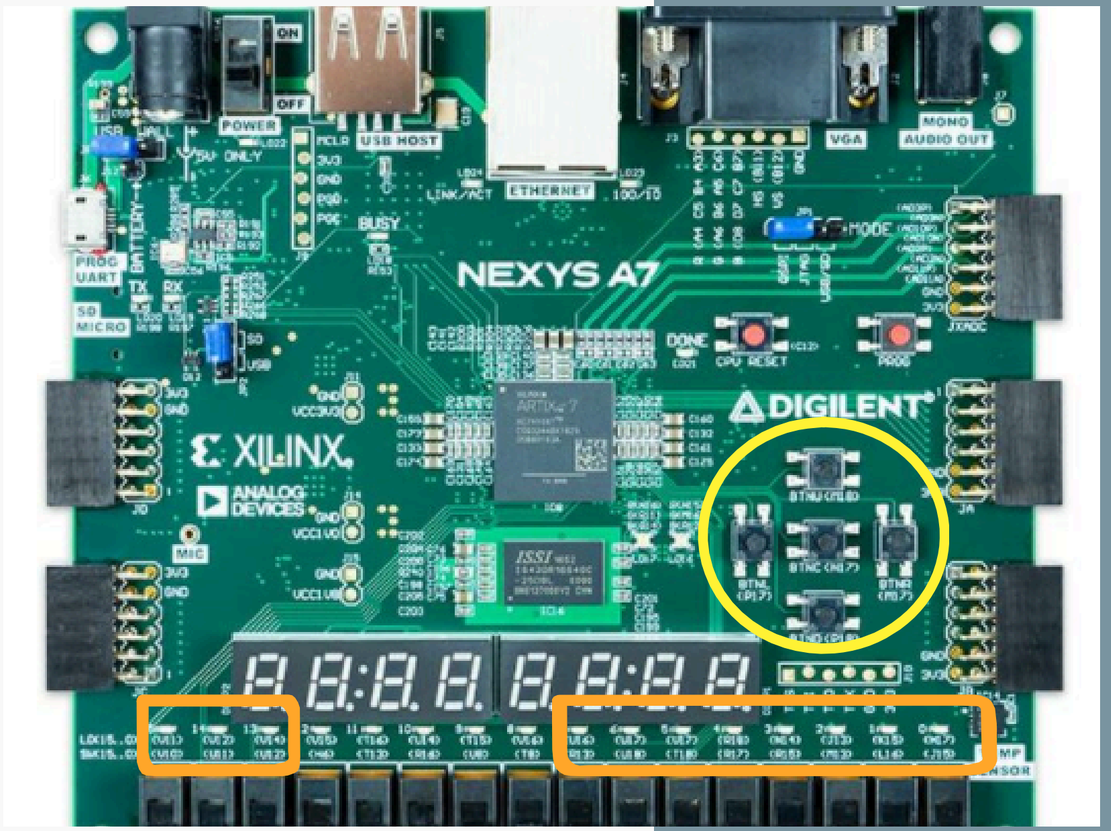
Decode:

Translate input sequence to ASCII text equivalent

Display:

1) VGA screen display: ASCII letter

2) LEDs on FPGA: 8-bit ASCII value,
   enter LED, dot LED & dash LED to indicate a press was registered



Specifications

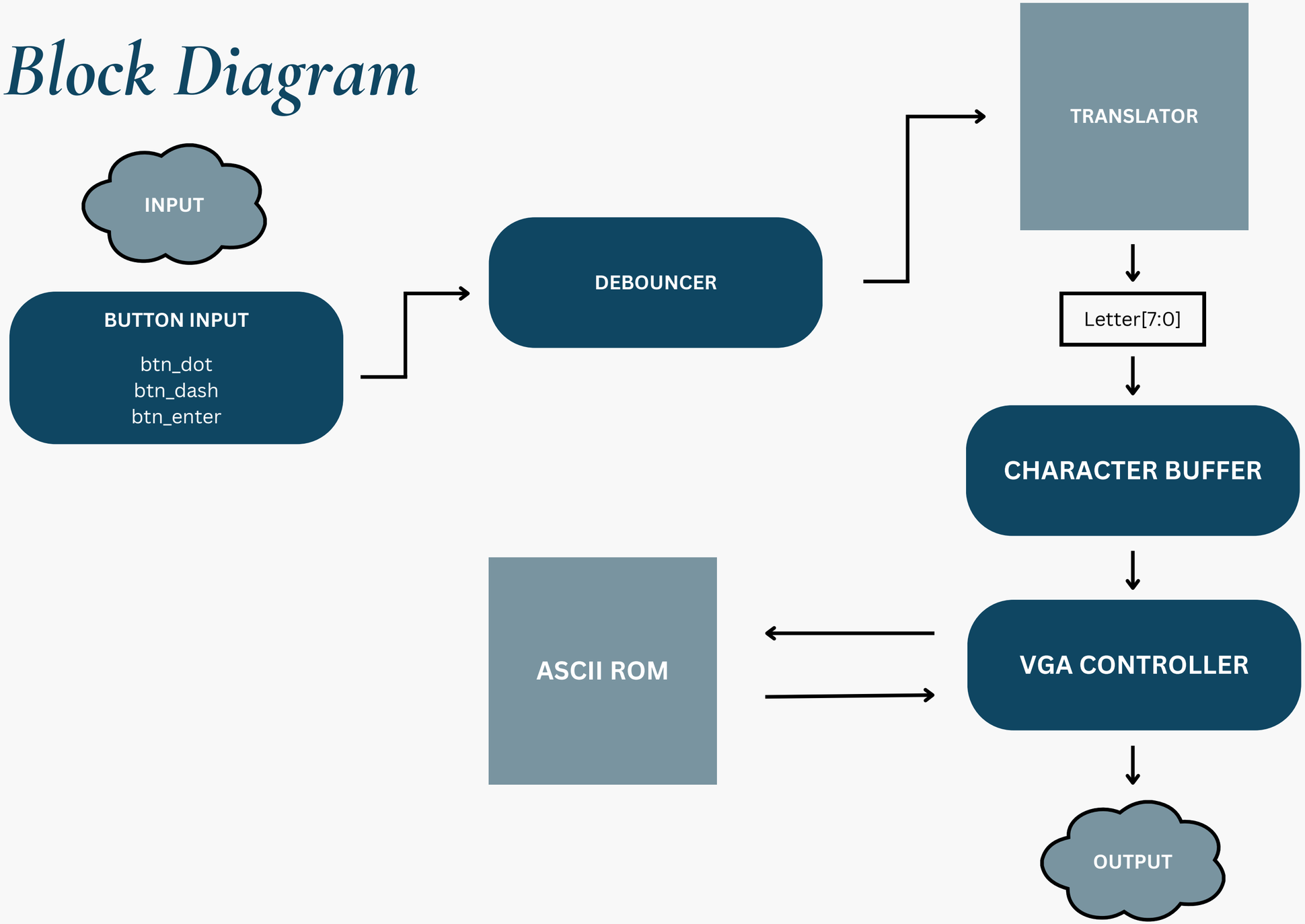
Requirements:

- International Morse Code Alphabet
- Button Inputs, VGA output
- 6-bit binary input translated to 8-bit ASCII value -> matched to bit pattern to display letter

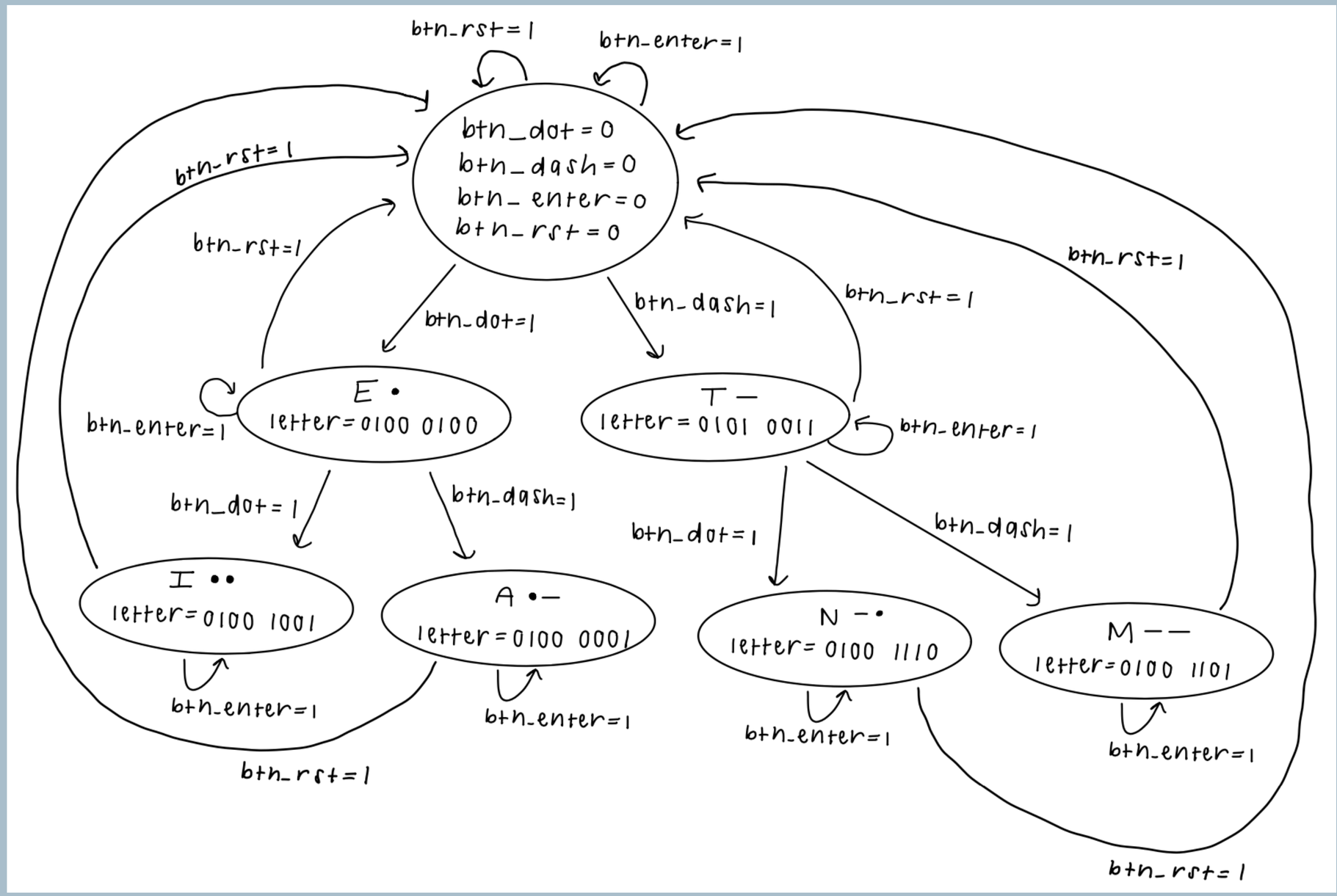
Constraints:

- VGA screen spacing
  - text output 640x480 resolution
- Real-time Translation

Block Diagram



Simplified FSM



Code Snippet

Morse code translation Logic

```
always@(negedge clk)
begin
```

```
if(btn_dot_posedge)begin
    morse_code = (morse_code << 1);
    morse_code[0] = 0;
    code_length = code_length + 1;
    check_dot=1;
    ready <= 0;
end
else if (btn_dash_posedge) begin
    morse_code = (morse_code << 1);
    morse_code[0] = 1;
    code_length= code_length + 1;
    check_dash =1;
    ready <= 0;

end
```

```
else if (btn_enter_posedge)begin
    if (morse_code == 6'b000001 && code_length == 4'b0010) begin
        letter = "A";
        end
    else if (morse_code == 6'b001000 && code_length == 4'b0100) begin
        letter = "B";
        end
    else if (morse_code == 6'b001010 && code_length == 4'b0100) begin
        letter = "C";
        end
    else if (morse_code == 6'b000100 && code_length == 4'b0011) begin
        letter = "D";
        end
```

Code Snippet

VGA Logic:

```
// Adder ROM logic for each character
always @(*) begin
    adder_rom = 11'h000; // Default value
    for (i = 0; i < MAX_LETTERS; i = i + 1) begin
        if (letter_enables[i]) begin
            case (memory[i])
                8'b00000000: adder_rom = 11'h2e0;
                8'b01000001: adder_rom = 11'h410;
                8'b01000010: adder_rom = 11'h420;
                8'b01000011: adder_rom = 11'h430;
                8'b01000100: adder_rom = 11'h440;
                8'b01000101: adder_rom = 11'h450;
                8'b01000110: adder_rom = 11'h460;
```

```
always @(posedge clk or posedge rst) begin
    if (rst) begin
        letter_count <= 0;
        current_x_pos <= 200;
        for (i = 0; i < MAX_LETTERS; i = i + 1) begin
            memory[i] <= 8'h00;
            letter_positions[i] <= 0;
        end
    end else if (btn_enter_rising_edge && (letter_count < MAX_LETTERS)) begin
        memory[letter_count] <= letter;
        letter_positions[letter_count] <= current_x_pos;
        letter_count <= letter_count + 1;
        current_x_pos <= current_x_pos + CHAR_WIDTH;
    end
end
```

Successes

VGA Display

- Multiple letter inputs can display on the screen at once

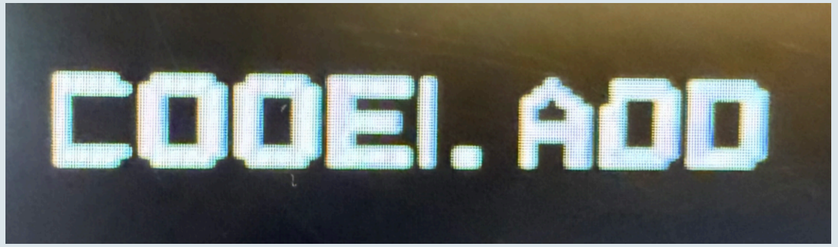
Morse Code Translation

- Correctly processes button inputs as morse code sequence
- Matches morse code sequence to its corresponding ASCII value/bit pattern
- Displays correct letter

Failures

Visuals

- User has to reset when number of letter inputs = the max memory array size (50)
  - Won’t move to next line
- Line of white space after each letter on screen, at times distorts the letter displayed to its right
  - Example: Confusion between O and D when side by side



Character Limitations

- Did not include numbers/symbols
- Period represented by “- - - - ”

Thank you!

Questions?