Lab 4: Code Conversion

ECE 211: Digital Circuits I — Spring 2025

Abstract. In the lab, you will design and build a code converter that takes the POSTNET 2-out-of-5 code as input and delivers the corresponding binary value as output. After implementing the POSTNET conversion module in SystemVerilog, you will instantiate multiple modules and connect them to the seven-segment display on your Nexys A7 board to show translations of POSTNET to decimal numbers.

POSTNET is a barcode system that was used by the United States Postal Service to automatically sort letters. To design your POSTNET to binary circuit, you will use new techniques including a five-variable K-map and having "don't care" values in your truth table. Before getting started on this lab, read the Background section to learn how POSTNET works!

Objectives.

- Use a ***five-variable Karnaugh map*** and ***don't cares ('X')*** to design a logic circuit that meets a given functional specification
- Implement a logic circuit using ***gate-level modeling*** in SystemVerilog
- Test the operation of a circuit using the input and output components provided by the Nexys A7

Materials and Equipment.

- Nexys A7-100T FPGA board with USB cable

Deliverables.

- In-person lab demonstration (per team)
- Lab report (per team)

Background. The task of efficiently sorting and delivering mail is momentous for the United States Postal Service (USPS). To address this task, the USPS developed ***POSTNET*** (Postal Numeric Encoding Technique), a representation of a ZIP Code that is printed directly on a piece of mail and can be read by automated sorting machines.

A POSTNET barcode consists of tall (full) bars and short (half) bars, where a tall bar represents "1," and a short bar represents "0." POSTNET is a 2-out-of-5 code, meaning that it is always composed of five digits, and two of those digits must be 1. This property allows for error detection since we will know the barcode has been smudged if more or less than two of the digits are 1. POSTNET is not a binary

number. Instead, it uses the positional weighting 7-4-2-1-0, as shown in the table below. All decimal numbers from 0 to 9 can be accommodated in POSTNET, except for 0. Instead, POSTNET uses the special value 11000 to represent zero.

POSTNET was replaced by the Intelligent Mail barcode in 2009, but it is still an important piece of mail history!

		7 4 2 1 0	
POSTNET Barcode	**V W X Y Z**	**Digit**	**A B C D**
	1 1 0 0 0	**0**	0 0 0 0
	0 0 0 1 1	**1**	0 0 0 1
	0 0 1 0 1	**2**	0 0 1 0
	0 0 1 1 0	**3**	0 0 1 1
	0 1 0 0 1	**4**	0 1 0 0
	0 1 0 1 0	**5**	0 1 0 1
	0 1 1 0 0	**6**	0 1 1 0
	1 0 0 0 1	**7**	0 1 1 1
	1 0 0 1 0	**8**	1 0 0 0
	1 0 1 0 0	**9**	1 0 0 1

Table 1. POSTNET Code Conversion (POSTNET to Binary)

Part 0: Set Up a Vivado Project

To begin the lab, we will set up a new Vivado project for Lab 4 and configure the Nexys A7 board. You can reference Lab 0 for a more detailed description of these steps.

1. Create a Xilinx Vivado project and import the provided constraints file (`main_constraints_NexysA7.xdc`) and SystemVerilog modules (`lab04_top.sv`, `disp_mux_seven_seg.sv`, `adv_seven_seg_n.sv`).

2. Connect your FPGA board, generate a bitstream, and program the FPGA.

Stop and Check: Toggle the switches switches (`SW[0]-SW[15]`) and check whether the corresponding LEDs (`LED[0]-LED[15]`) turn on and off.

Part 1: Design your POSTNET to Binary Converter

In this part, you will design Boolean equations to implement your POSTNET to binary converter. Create a truth table that uses the POSTNET code as inputs (variables V, W, X, Y, Z) and a binary number as output (variables A, B, C, D). Since there are five input variables, there will be 32 rows in your truth table! Not all of the rows are important to us. For example, the input combination 11111 is not a valid POSTNET barcode. We could choose to set the output ABCD=0000, since it is invalid. Alternatively, we can set the output as a "don't care" value, represented by a 'X' (*i.e.,* ABCD=XXXX). Don't cares are special values in a K-Map that we can treat as zeros *or* ones.! In other words, if we have an 'X' in our K-Map, we can choose to include it in a grouping (like a 1), but we are not required to group it. We can decide based on what will allow us to make the largest grouping.

Since there are five input variables, you can use a five-variable K-Map to form your equations, shown in Figure 1. A five-variable K-Map is three-dimensional, visualized in Figure 2. Groupings "between" the four-variable K-Maps will eliminate the variable V in the corresponding implicant.

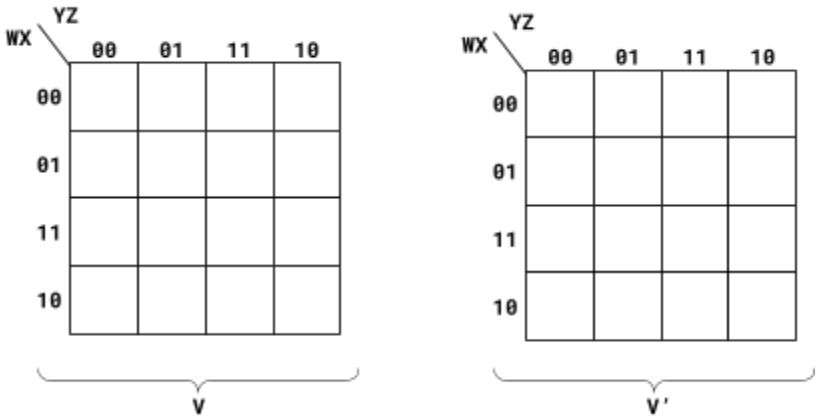

Figure 1. Five-Variable K-Map with inputs V, W, X, Y, Z

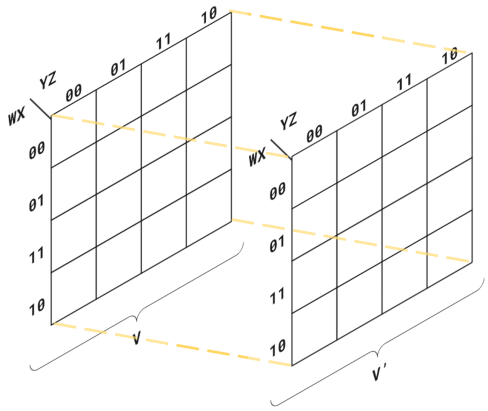

Figure 2. 3-D View of Five-Variable K-Map

Ultimately, you will need to create *four* five-variable K-Maps—one for each bit of binary output. Minimize each K-Map and document the resulting Boolean equations for A, B, C, and D.

3. Create a truth table that uses the POSTNET code as inputs (variables V, W, X, Y, Z) and a binary number as output (variables A, B, C, D). Your truth table should convert the POSTNET values shown in Table 1 to the corresponding binary values. You can use don't cares (X) for any invalid input combinations.

4. Using K-Maps and/or Boolean theorems, derive minimized equations for each of the outputs A, B, C, and D.

5. Sketch a schematic of your circuit. This schematic will come in handy when implementing the circuit in SystemVerilog.

Record: Record your truth tables, K-Maps, and Boolean equations to include in your lab report.

Part 2: Implement a Code Conversion Module in SystemVerilog

Now, implement your POSTNET to binary circuit as a SystemVerilog module. In the next part, we will instantiate this module multiple times to display multiple digits of POSTNET on the Nexys A7's seven-segment display.

6. Create a new source file for your `postnet_to_binary` module:
   a. In the Flow Navigator toolbar, select "Add Sources" listed under "Project Manager" to open the Add Sources wizard.
   b. Select "Add or create design sources."
   c. Select "Create File."
   d. Change the file type to SystemVerilog.
   e. Set the file name to `postnet_to_binary` (the file name and module name are the same). Then, click "OK" to create the new module.
   f. Click "Finish" to exit the Add Sources wizard.

7. Implement your `postnet_to_binary` module to perform the Boolean equations for A-D:
   a. Your module should take the POSTNET code (V, W, X, Y, and Z) as input
   b. Your module should output the binary value (A, B, C, and D)
   c. Implement the module body according to your equations for A, B, C, and D.

Part 3: Instantiate and Connect your POSTNET Module to the Seven-Segment Display

In the final part of the lab assignment, you will instantiate your POSTNET module to display three POSTNET values on the seven-segment display. Switches `SW[4]-SW[0]` will encode a 5-bit POSTNET input, `P0`, which will be displayed as the 4-bit binary number `B0` on the seven-segment display. Switches `SW[9]-SW[5]` will encode a POSTNET input, `P1`, which will be displayed as the 4-bit binary number `B1` on the seven-segment display. Switches `SW[14]-SW[10]` will encode a POSTNET input, `P2`, which will be displayed as the 4-bit binary number `B2` on the seven-segment display. This configuration is shown in Figure 3.

A module that configures the seven-segment display is provided as part of the lab starter code. You can look at these modules, but you do not need to understand them. To configure the seven-segment display, you will simply wire the input connections in the top-level module.

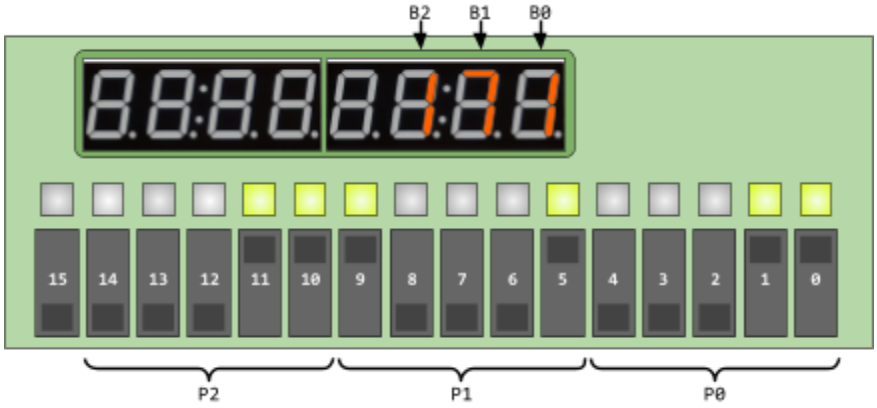

Figure 3. Input/Output Configuration for Part 3

8. Instantiate a POSTNET module for POSTNET digit `P0`. Connect the module's inputs to the switches shown in Figure 3 (`SW[4]-SW[0]`). Connect the module's outputs to a set of named wires (*e.g.,* `a0, b0, c0, d0`).

9. Repeat Step 8 to instantiate POSTNET modules for the digits `P1` and `P2`. Connect these modules to the corresponding switches.

10. Connect the output of each POSTNET module to the seven-segment display by completing the following code found in `lab4_top.sv`:

```
// Display binary output on seven-segment display
disp_mux_seven_seg DISP(.clk(clk), .reset(1'b0),
    .in0(/*Connect B0 vector here*/),
    .in1(/*Connect B1 vector here*/),
    .in2(/*Connect B2 vector here*/),
```

```
    .in3(4'b0000), .seg_dis(4'b1000), .seg_n(SEG), .an(AN));
```

The seven-segment display takes in the three binary numbers on ports `in0`, `in1`, and `in2` Each of these inputs is a 4-bit vector type in SystemVerilog. To connect the scalar outputs of your first POSTNET module to the seven-segment port `in0`, you will need to use the concatenation operator. The ***concatenation operator*** `{ , }` combines (concatenates) two or more scalar values into a vector. The following code snippet shows an example of combining four scalar values into a 4-bit vector called `B0`.

```
// Example of using the concatenation operator {} to combine
// multiple scalar values into a 4-bit vector
logic [3:0] B0;
assign B0 = {a0, b0, c0, d0};

// You can also do this all on the same line!
logic [3:0] B0 = {a0, b0, c0, d0};
```

After using the concatenation operator, you can connect the signal `B0` directly to the pin `in0` on the seven-segment module. Repeat this step for the remaining outputs `B1` and `B2`.

11. Verify that the circuit schematic matches your proposed design. You can view the circuit schematic under “RTL Analysis/Open Elaborated Design/Schematic” in the Flow Navigator.

12. Generate a bitstream, program the FPGA, and test that your implementation functions correctly.

Stop and Check: Toggle the switches for inputs `P0`, `P1`, and `P2` to ensure that the implementation is working correctly.

Record: Screenshot the circuit schematic (produced by Vivado) for this step. Record your test cases to include in the Results section of your lab report.

Part 4: Wrap-Up

Before leaving the lab, make sure to complete the following steps!

13. ***Demonstrate the operation of your circuit to the instructor.*** A portion of your lab grade will come from the lab report in addition to this demonstration.

14. Turn the Nexys board OFF and disconnect it from the computer. Make sure to save your Vivado project and sign out of your machine!

Lab Report

Lab Report. In collaboration with your lab partner, submit one lab report on Moodle that describes the lab activity and your team's circuit design. Your written report is worth 60% of your grade for each lab assignment. Your lab report should include the following sections:

- **Title Page:** Your lab report should begin with a title block that includes the course number, the title of the lab assignment, the names of each team member, and the date of the lab assignment. The title page should also include a "Statement of Collaboration" (detailed below) and an estimate of the total amount of time spent on the lab assignment. The inclusion of these components are graded, but the content of each statement is not used in determining your grade.

- **Statement of Collaboration:** Describe the contributions of each team member to the lab assignment, including writing of the lab report. What is specified in this section will not affect your grade, however it is expected that you periodically rotate roles within your lab group if you do divide work, including writing the report.

- **Introduction:** Include a brief paragraph summarizing the objectives of the lab and what your team achieved. This section should be written in your own words; it is not acceptable to copy text for this from the lab handout.

- **Design:** Include a concise yet descriptive explanation of the circuit design, following all steps in the process—from the problem specification and initial truth table, to equations and circuit schematics. Tables and equations should be typeset and clearly labeled. Technical information should be accompanied by some text narrating each step or aspect of the design.

- **Testing & Results:** Describe how you tested your circuit, including rationale for why you chose specific test cases if the circuit is not tested exhaustively. If you tested your circuit at multiple points (*i.e.,* using a testbench and then on the Nexys board), this section should include all test cases at each point. Test cases can be included in the form of tables or simulation waveforms. Results should be neatly formatted and labeled, indicating what step is being tested. This section should also include any recorded measurements, tests, diagrams, or images specified by the lab document.

- **Discussion/Conclusion:** Your discussion/conclusion section should comment on any observations about the technique you employed to get your results. Briefly describe any difficulties that you encountered and how you resolved them. Then, summarize what you concluded from your work. (It may also be helpful to think negatively, *i.e.* what your results did not show. Remember that a negative result is often as valuable as a positive result!)

  A number of technical questions may be posed in the assignment sheet. If questions are given, include your answers to them in this section.

Your lab report should be typed and submitted as a .pdf document. Only one member per team needs to submit the lab report.