ECE2330 – Digital Logic Design

Opcode Decoder

Learning Objective

After this activity, you will be able to

- Design an opcode decoder using analytical digital design techniques, and
- Verify that the implementation you specify satisfies the requirements

Problem Statement

For this learning activity, you are **required** to design a digital decoding circuit to decode the 3-bit opcode for the instruction format shown in Figure 1. The 8 instructions are listed in Table 1.

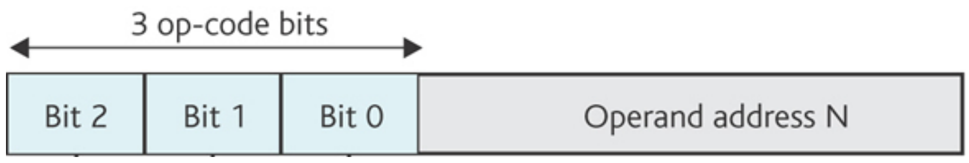


Figure 1: Simple CPU Instruction Format

Table 1: Simple CPU Instructions

Opcode	Mnemonic	Operation
000	Load N	Load data register, D0, with the contents at memory address = N
001	Store N	Store contents of data register, D0, at memory address = N
010	Add N	The contents at memory address = N are added to the contents of data register, D0, and then stored in data register, D0
011	Sub N	The contents at memory address = N are subtracted from the contents of data register, D0, and then stored in data register, D0
100	Inc N	The contents at memory address = N are incremented by 1
101	Dec N	The contents at memory address = N are decremented by 1
110	Bra N	The Program Counter (PC) is loaded with the memory address = N
111	Beq N	The Program Counter (PC) is loaded with the memory address = N if the last arithmetic operation produced a result of zero (indicated by the zero bit of the ALU, Z = 1).

In order to verify your design, you will create a testbench similar to the one shown in Figure 2, which includes a **Test** component with an exhaustive set of input test vectors, shown in Figure 4, along with the results.

The **Execute** input is a signal generated by the Fetch-Execute Finite State Machine (FSM), which equals 0 when an instruction is being fetched and 1 when an instruction is being executed. Thus, the output that indicates which instruction to execute should equal 1 if and only if **Execute** = 1. Otherwise, if **Execute** = 0, they should all equal 0.

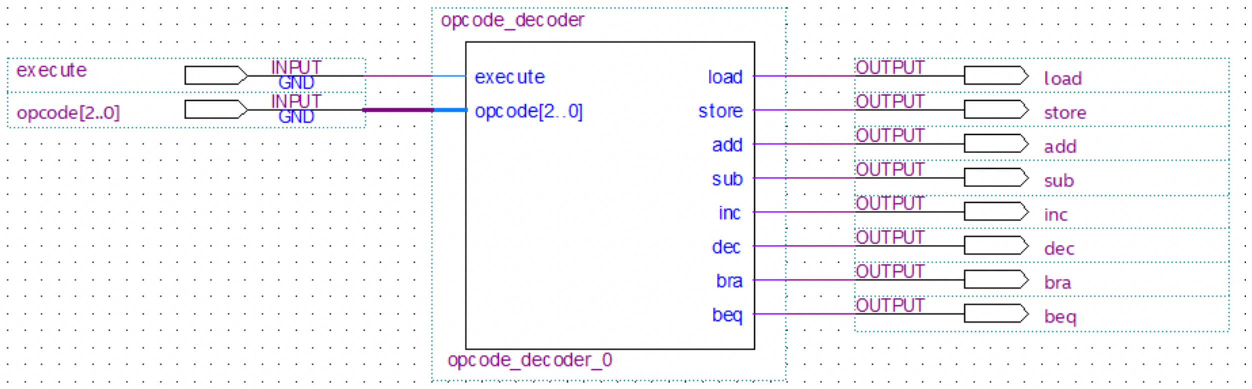
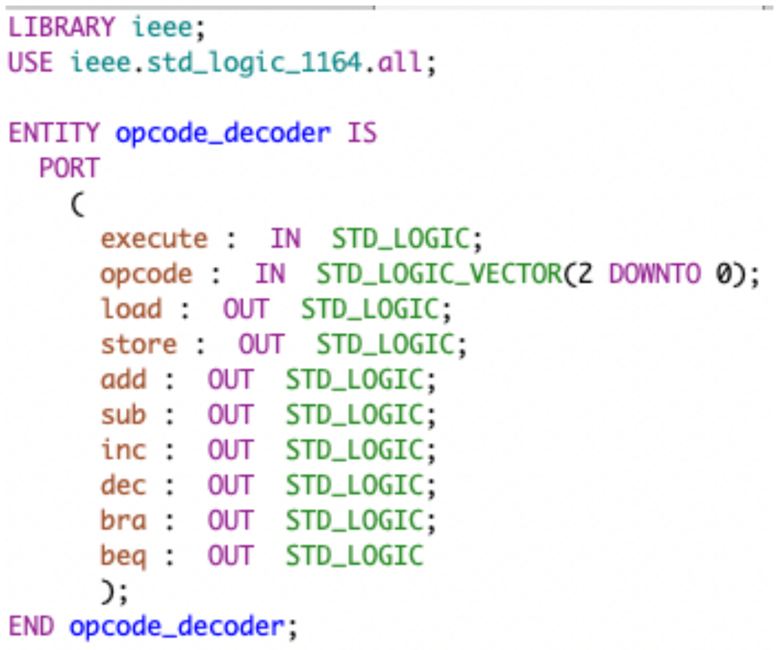


Figure 2: Opcode Decoder Testbench



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY opcode_decoder IS
  PORT
    (
      execute :  IN  STD_LOGIC;
      opcode :  IN  STD_LOGIC_VECTOR(2 DOWNTO 0);
      load :  OUT  STD_LOGIC;
      store :  OUT  STD_LOGIC;
      add :  OUT  STD_LOGIC;
      sub :  OUT  STD_LOGIC;
      inc :  OUT  STD_LOGIC;
      dec :  OUT  STD_LOGIC;
      bra :  OUT  STD_LOGIC;
      beq :  OUT  STD_LOGIC
    );
END opcode_decoder;
```

Figure 3: VHDL Entity declaration for opcode decoder

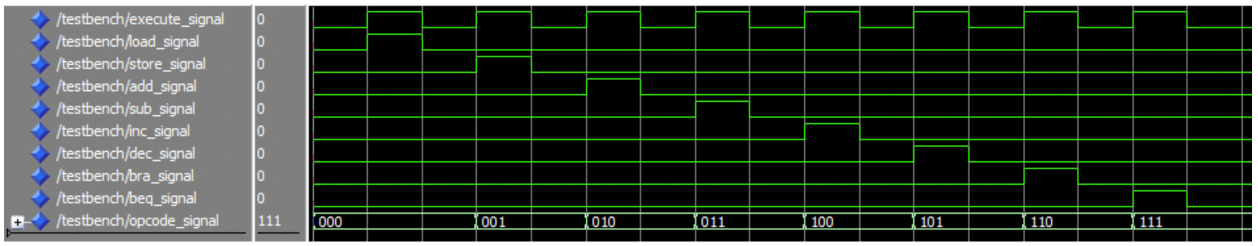


Figure 4: Test Vectors and Results

What To Submit

You will submit the VHDL file you create as part of this assignment, along with simulation results for verification (as a PDF file, annotated appropriately where necessary).

Grading Rubric

This assignment is worth a total of 10 points:

- VHDL file of opcode decoder design
- VHDL file of testbench
- Numerical Verification