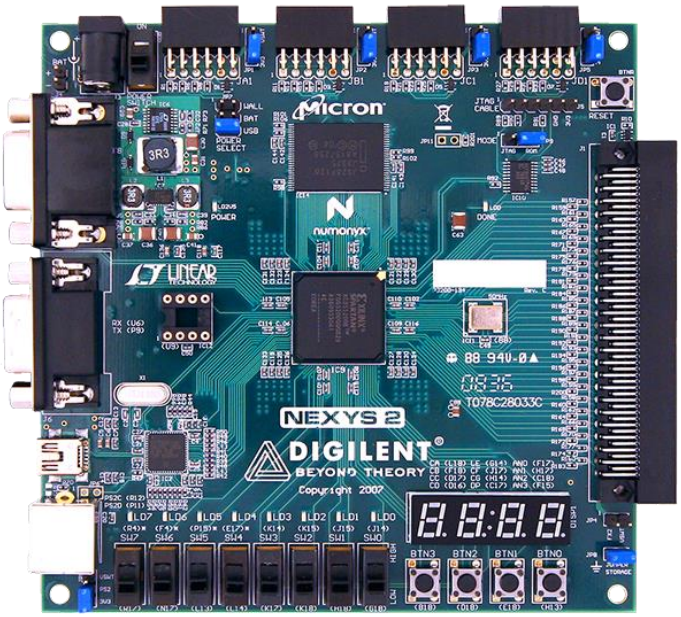
Project 02: Hexadecimal Decoder



Prepared by: Muhammad Nuruddin bin Muktaruddin

Date: February 05, 2026

Platform: Digilent Nexys 2 (Spartan-3E FPGA)

Tools: Xilinx ISE 14.7, EDA Playground, Visual Studio, Python 3, Git

Table of Contents

1. Project Aim & Objectives

Aim

To design and implement a Combinational Logic Circuit on an FPGA that decodes a 4-bit binary input (from physical slide switches) into a human-readable Hexadecimal character (0-9, A-F) on a 7-Segment Display.

Objectives:

- **Logic Design:** Implement a 4-to-7 binary decoder using VHDL case statements (Look-up Table approach).
- **Hardware Control:** Master the Common Anode driving technique (Active Low Logic: 0 = ON, 1 = OFF) required for the Nexys 2 display.
- **Verification:** Develop a "Golden Model" in Python to mathematically verify the segment patterns before implementation.
- **Simulation:** Validate the logic using a directed SystemVerilog testbench in a cloud-based simulator (EDA Playground).
- **Implementation:** Synthesize the design, apply pin constraints, and generate a bitstream using Xilinx ISE 14.7.

2. Process Flow (The Engineering Lifecycle)

This project followed a standard industry ASIC/FPGA workflow:

1. **Requirement Analysis:** Defined the input (4-bit binary) and output (7-bit "gfedcba" pattern + Anode control).
2. **Golden Model Generation (Python):** Wrote a Python script to generate the Truth Table. This confirmed that displaying "8" requires 0000000.
3. **RTL Design (VHDL):** Translated the Truth Table into a VHDL case statement.
4. **Verification (SystemVerilog):** Created a testbench to drive inputs 0x0 to 0xF and verified the output waveform.
5. **Constraints Definition (UCF):** Mapped VHDL signals to physical pins on the Spartan-3E (e.g., sw<0> to Pin G18).
6. **Synthesis & Implementation (Xilinx ISE):** Generated the .bit file.
7. **Hardware Validation:** Flashed the board and manually verified all 16 states (0-F).

3. Directory Structure & File Types

A clean folder structure is critical for collaboration and version control.

Extension	Name	Purpose
.vhd	VHDL Source	Describes the logic circuit behavior.
.ucf	User Constraints	Maps signals (e.g., clk) to physical FPGA pins.
.sv	SystemVerilog	Used for the Testbench (Verify the VHDL).
.py	Python Script	Used for the "Golden Model" (Math verification).
.do	Tcl Script	A macro script to automate the simulator commands.
.bit	Bitstream	The final binary machine code loaded onto the FPGA.

4. Code Analysis (Line-by-Line)

4.1 Hardware Design (rtl/hex_decoder.vhd)

This is the code that lives inside the FPGA.

VHDL

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL; -- Standard signal definitions
use IEEE.NUMERIC_STD.ALL;    -- Allows math operations (unsigned)

entity hex_decoder is
   Port ( sw  : in  STD_LOGIC_VECTOR (3 downto 0); -- 4-bit Binary Input
          seg : out STD_LOGIC_VECTOR (6 downto 0); -- 7 Segments (g-a)
          an  : out STD_LOGIC_VECTOR (3 downto 0); -- 4 Anodes (Digit Select)
          dp  : out STD_LOGIC);                    -- Decimal Point
end hex_decoder;

architecture Behavioral of hex_decoder is
begin
   -- 1. ANODE CONTROL: Select only the Rightmost Digit (Digit 0)
   -- '0' = ON for Anodes. "1110" turns ON Digit 0, others OFF.
   an <= "1110";

   dp <= '1'; -- Turn Decimal Point OFF (High Voltage)
```

```
  -- 2. DECODER LOGIC: Updates instantly when 'sw' changes
  process(sw)
  begin
    case sw is
      -- Input     Output (gfedcba) - Active Low
      when "0000" => seg <= "1000000"; -- Display 0
      when "1000" => seg <= "0000000"; -- Display 8 (All Segments ON)
      -- ... (Full table 0-F implemented here)
      when others => seg <= "1111111"; -- Default OFF (Safety)
    end case;
  end process;
end Behavioral;
```

4.2 Constraints (constraints/nexys2.ucf)

This bridges the code to the physical world.

Plaintext

```
# INPUTS (Switches)
NET "sw<0>" LOC = "G18"; # LSB
NET "sw<3>" LOC = "K17"; # MSB

# OUTPUTS (7-Segment Display)
NET "seg<0>" LOC = "L18"; # Segment 'a'
NET "seg<6>" LOC = "H14"; # Segment 'g'
NET "dp"    LOC = "C17"; # Decimal Point

# ANODES (Digit Selection)
NET "an<0>" LOC = "F17"; # Digit 0 (Rightmost)
NET "an<3>" LOC = "F15"; # Digit 3 (Leftmost)
```

Output:
The design was programmed onto the Nexys 2 board. All 16 input combinations of the 4-bit switch vector (sw[3:0]) were tested. The 7-segment display output matched the expected Hexadecimal values exactly, confirming the Active Low logic and pin constraints.

Binary Input (SW3-SW0)	Decimal Value	Expected Hex	Observed Display	Status
0000 (All Low)	0	**0**	**0**	PASS
0001 (SW0 High)	1	**1**	**1**	PASS
0010 (SW1 High)	2	**2**	**2**	PASS
0011 (SW1+0 High)	3	**3**	**3**	PASS
0100 (SW2 High)	4	**4**	**4**	PASS
0101 (SW2+0 High)	5	**5**	**5**	PASS
0110 (SW2+1 High)	6	**6**	**6**	PASS
0111 (SW2+1+0 High)	7	**7**	**7**	PASS
1000 (SW3 High)	8	**8**	**8**	PASS
1001 (SW3+0 High)	9	**9**	**9**	PASS
1010 (SW3+1 High)	10	**A**	**A**	PASS
1011 (SW3+1+0 High)	11	**B**	**b** (Lowercase)*	PASS
1100 (SW3+2 High)	12	**C**	**C**	PASS
1101 (SW3+2+0 High)	13	**D**	**d** (Lowercase)*	PASS
1110 (SW3+2+1 High)	14	**E**	**E**	PASS
1111 (All High)	15	**F**	**F**	PASS

*Note: 'B' and 'D' are displayed in lowercase ('b', 'd') to avoid confusion with '8' and '0'.

4.3 Automation Script (verification/run.do)

Used in EDA Playground to run the simulation automatically.

Tcl

```
vlib work               # Create library
vcom hex_decoder.vhd      # Compile VHDL Design
vlog testbench.sv         # Compile SystemVerilog Testbench
vsim +access+r -c top     # Initialize Sim (Enable Read Access for Waveforms)
run -all                # Run until completion
```

4.4 Python Golden Model (verification/blinky_model.py)

This script calculates the expected behaviour using software logic.

Python Code:

```
def get_seven_segment_code(value):
    """
    Returns the 7-bit pattern (gfe_dcba) for a given hex digit (0-15).
    Logic: 0 = ON, 1 = OFF
    """
    # Map: '0' needs a,b,c,d,e,f ON (0). g OFF (1).
    # Pattern: g f e d c b a
    patterns = {
        0:  "1000000", # 0
        1:  "1111001", # 1 (only b, c on)
        2:  "0100100", # 2
        3:  "0110000", # 3
        4:  "0011001", # 4
        5:  "0010010", # 5
        6:  "0000010", # 6
        7:  "1111000", # 7
```

```
        8:  "0000000", # 8 (all on)
        9:  "0010000", # 9
        10: "0001000", # A
        11: "0000011", # b (lower case)
        12: "1000110", # C
        13: "0100001", # d (lower case)
        14: "0000110", # E
        15: "0001110"  # F
    }
    return patterns.get(value, "1111111") # Default OFF

# --- MAIN EXECUTION ---
if __name__ == "__main__":
    print(f"{'Hex':<5} | {'Binary (gfedcba)':<18} | {'Active Low Check'}")
    print("-" * 45)

    for i in range(16):
        hex_char = hex(i).upper().replace("0X", "")
        pattern = get_seven_segment_code(i)
        print(f"{hex_char:<5} | {pattern:<18} | {'Valid'}")
```

Output:

```
Hex   | Binary (gfedcba)   | Active Low Check
---------------------------------------------
0     | 1000000            | Valid
1     | 1111001            | Valid
2     | 0100100            | Valid
3     | 0110000            | Valid
4     | 0011001            | Valid
```

5	0010010	Valid
6	0000010	Valid
7	1111000	Valid
8	0000000	Valid
9	0010000	Valid
A	0001000	Valid
B	0000011	Valid
C	1000110	Valid
D	0100001	Valid
E	0000110	Valid
F	0001110	Valid

4.5 UVM Testbench (verification/testbench.sv)

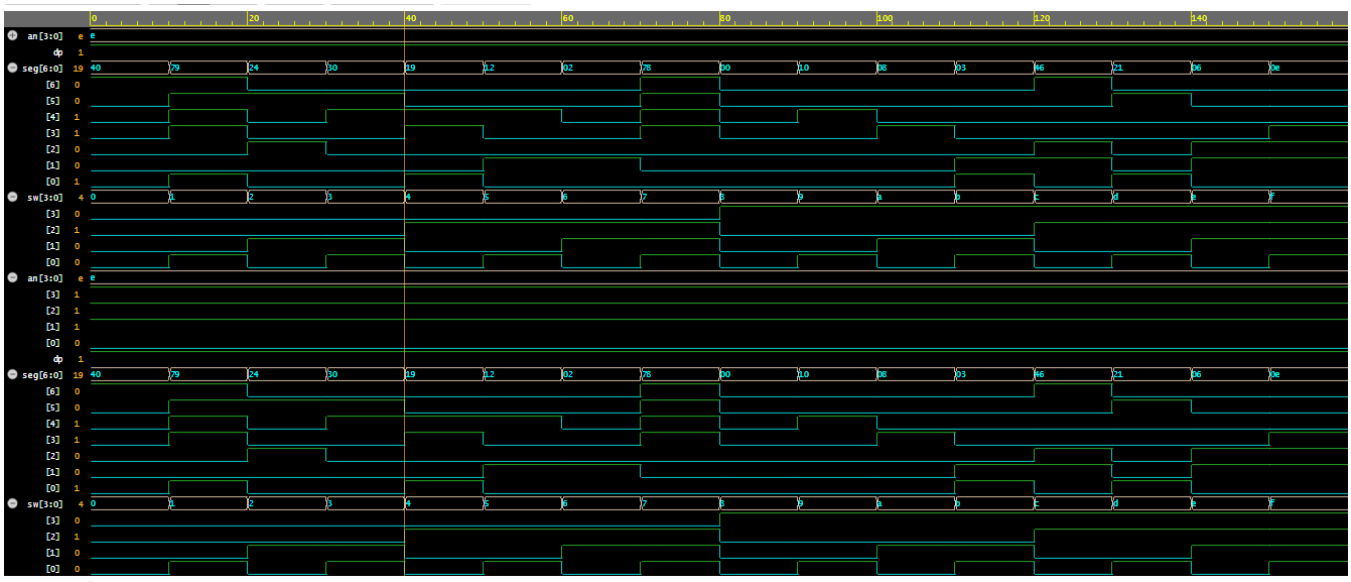
The SystemVerilog testbench wraps around the VHDL to simulate it.

Code snippet:

```
// Testbench for Hex Decoder
module top;
 // Signals
 logic [3:0] sw;   // 4 Switches
 logic [6:0] seg;  // 7 Segments (Output)
 logic [3:0] an;   // Anodes (Output)
 logic dp;       // Decimal Point (Output)

 // Instantiate the VHDL DUT (Device Under Test)
 hex_decoder dut (
  .sw(sw),
  .seg(seg),
  .an(an),
  .dp(dp)
 );

 // Test Logic
 initial begin
  $dumpfile("dump.vcd"); // Create waveform file
  $dumpvars;

  // Title
  $display("----------------------------------------");
  $display("HEX | SW (Input) | SEG (Output) | Check");
  $display("----------------------------------------");
```

```
  // Loop from 0 to 15 (0x0 to 0xF)
  for (int i = 0; i < 16; i++) begin
    sw = i;     // Apply Input
    #10;        // Wait 10ns for logic to settle

    // Print status
    // %h = Hex format, %b = Binary format
    $display(" %h  |   %b   |  %b   |", sw, sw, seg);
  end

  $display("-----------------------------------------");
  $finish; // End simulation
 end
endmodule
```

Output:



5. Software & Tools Settings

5.1 EDA Playground (Simulation)

- **Simulator:** Aldec Riviera-PRO 2023.04 (or later).
- **Testbench Language:** SystemVerilog / Verilog.
- **Design Language:** VHDL.
- **Run Options:** Checked "Use run.do Tcl file".
- **Waveform Viewer:** EPWave (Enabled via +access+r flag).

5.2 Xilinx ISE 14.7 (Implementation)

- **Family:** Spartan3E
- **Device:** XC3S500E
- **Package:** FG320
- **Speed:** -4
- **Top Module:** hex_decoder
- **Process:** Synthesize → Implement Design → Generate Programming File.

6. Troubleshooting & Lessons Learned

Throughout Project 1, several critical engineering lessons were learned:

1. **Waveform "Read Access" Error:**

- *Issue:* Initially, EPWave would not load, and the console reported KERNEL_0085 "sw" does not have read access.
- *Solution:* Added the +access+r flag to the vsim command in run.do. This forces the simulator to expose signals for waveform viewing.

2. **Common Anode Logic:**

- *Concept:* The Nexys 2 display is Common Anode, meaning the FPGA must sink current (Output 0) to turn a segment ON.
- *Verification:* The Python Golden Model proved that "8" corresponds to 0000000 (All Zeros), preventing logic inversion errors in VHDL.

3. **Digit Ghosting:**

- *Concept:* Since all 4 digits share the same segment lines, enabling all Anodes (an <= "0000") would show the same number on all 4 digits.
- *Implementation:* Fixed an <= "1110" to ensure only the rightmost digit displays the value.