ES 4 Lab 7: Using memory (ROM)

Katherine Deane
November 4th/11th 2024
6-8pm Tuesday Section, TAs: Matt, Quan, Griffin

The objective of this lab is to combine a counter and ROM to create a blinking animation on the 7-segment display; ROM stores a sequence of frames for the animation, and the counter provides the address for the ROM. There is no input, and the output is the 7-segment display.

Design – Truth Table

0 indicates when the segment is on, and 1 indicates when the segment is off. This should create a snake-like animation.

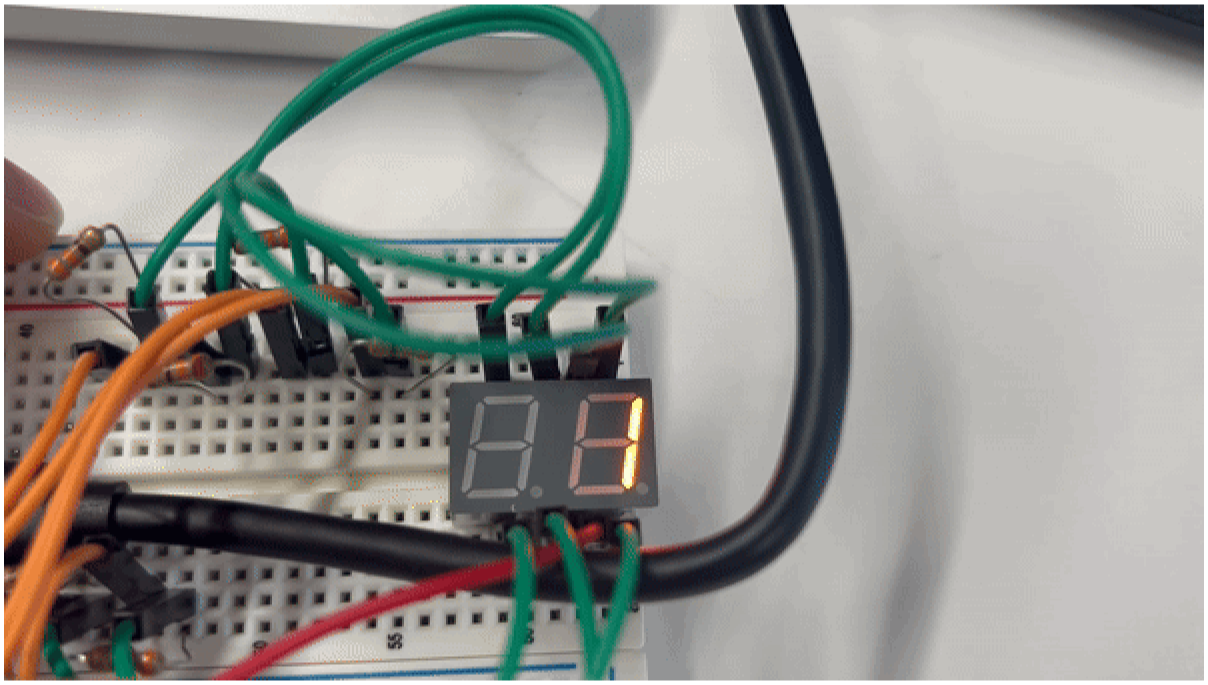
frame	**A**	**B**	**C**	**D**	**E**	**F**	**G**
000	1	0	0	1	1	1	1
001	0	0	1	1	1	1	1
010	0	1	1	1	1	0	1
011	1	1	1	1	1	0	0
100	1	1	0	1	1	1	0
101	1	1	0	0	1	1	1
110	1	1	1	0	0	1	1
111	1	1	1	1	0	0	1
others	0	0	0	0	0	0	0

Completed Circuit:

The circuit uses exactly the same wiring from the previous lab, except there is no need for the DIP switch inputs, so we only need the orange and green wires connecting to the 7-segment. Additionally, the tens digit and decimal point are grounded, while 3.3V power is connected to the ones digit.

The gif works on my google doc but not when I save it to a PDF, so here is a link to the video. Hope it works!

animation.MOV



Debugging Log

- Since I utilized most of my circuit from the previous lab—I only removed the DIP switch inputs, grounded the tens digit and decimal point, and connected the ones digit to 3.3V of power—I didn't have any issues to debug in regards to my hardware.
- I had an issue with my VHDL code, though: it synthesized with no problems, but the display was showing all 7 segments constantly lit. I enlisted Quan's help.
    - In my rom.vhd file, the segments should've only all been lit if the address was invalid (`when others=> data <= "0000000"`). My port mapping was completely fine, and my counter.vhd and rom.vhd files should not have been causing the issue, so I changed the `when others=> data <= "0000000"` statement in my `rom.vhd` file to be `when others=> data <= "0000001"`, which should have made the middle bar of the 7-segment turn off if the issue was that data was getting the 'others' value. We flashed the FPGA again, and same issue.
    - We combed through the code for 30 minutes and tried accessing data in different ways (for example, port mapping
        - `count(25 downto 23) => address`
        - `addr => std_logic_vector(address)`
    - instead of
        - `count => counttoaddr`
        - `addr => std_logic_vector(counttoaddr(14 downto 12))`

    ◦ even though they should have the same effect. This went on until Quan tried using higher bits of the counter to give us the address, and finally that worked. The display appeared to be completely lit the whole time because it was flashing so fast we couldn't see any segments turning off at all.
- Then I had a small issue with my animation because I forgot that F comes before G, but after that it worked.
- I was checked off by Quan.

Reflection:

1) The most valuable thing I learned was how to use submodules and synchronize clock signals to write VHDL code that infers a ROM block. To be honest this lab wasn't that crazy, so the "hardest" part was still just getting more comfortable with turning a circuit diagram into VHDL. Knowing how to connect multiple components using submodules and a top module and control the transfer of changing data with a synchronized clock is clearly important because it enabled me to basically make a ROM block in VHDL, which is useful for infinitely many applications, the least of which is making cool light animations.
2) I'm still working on getting VHDL syntax down. I feel like I've gotten a little bit better since Griffin told me that wires connecting submodules in the top module should be signals, and the rest (like port mapping) is getting a little bit more familiar too. To learn these skills more I'm going to practice on VHDLweb and just do the final project because at this point it seems like it's sink or swim.
3) It took me about 2.5 hours to complete the lab. It took me about two hours to do the actual lab and about a half hour to write up this report. The biggest time-consuming issue with the circuit was the fact that I was taking digits of my counter that were too low.

VHDL Code:

top.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity top is
  port(
        display : out std_logic_vector(6 downto 0)
  );
end top;

architecture synth of top is
signal clk : std_logic;
```

```
component HSOSC is
    generic (
        CLKHF_DIV : String := "0b00"); -- Divide 48MHz clock by 2^N (0-3)
    port(
        CLKHFPU : in std_logic := 'X'; -- Set to 1 to power up
        CLKHFEN : in std_logic := 'X'; -- Set to 1 to enable output
        CLKHF : out std_logic := 'X'); -- Clock output
end component;

component counter is
      port(
                  clk : in std_logic;
                  count : out unsigned(25 downto 0) := (others => '0')
            );
end component;

component rom is
      port (
                  clk : in std_logic;
                  addr : in std_logic_vector (2 downto 0); --make 2 downto 0
later on
                  data : out std_logic_vector (6 downto 0)
            );
end component;

signal address : unsigned(2 downto 0);

begin
    osc : HSOSC generic map ( CLKHF_DIV => "0b00")
            port map (CLKHFPU => '1',
            CLKHFEN => '1',
            CLKHF => clk); -- clk is run by CLKHF now

      counter1: counter port map (clk => clk, count(25 downto 23) => address);
      rom1: rom port map (clk => clk, addr => std_logic_vector(address), data
            => display);
end;
```

---

counter.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity counter is
      port(
                clk : in std_logic;
                count : out unsigned(25 downto 0) := (others => '0')
          );
end counter;

architecture synth of counter is
begin
process (clk) begin
    if rising_edge(clk) then
        count <= count + '1';
    end if;
end process;

end;
```

rom.vhd

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;


entity rom is
     port (
                clk : in std_logic;
                addr : in std_logic_vector (2 downto 0);
                data : out std_logic_vector (6 downto 0)
          );
end rom;

architecture synth of rom is
begin

process(clk) is
begin
     if rising_edge(clk) then
          case addr is
                when "000" => data <= 7b"1001111";
                when "001" => data <= 7b"0011111";
                when "010" => data <= 7b"0111101";
                when "011" => data <= 7b"1111100";
                when "100" => data <= 7b"1101110";
                when "101" => data <= 7b"1100111";
                when "110" => data <= 7b"1110011";
                when "111" => data <= 7b"1111001";
                when others => data <= 7b"0000000";
           end case;
      end if;
end process;

end;
```