Assignment - 9 Report

Tanish Gupta , Divyanshu Agarwal

29th May 2022

1 Introduction

The objective of this assignment was to create a FIFO buffer by first instantiating a memory component. First we implemented memory using BRAM and then designed a FIFO with push button switches for WRITE and READ.

2 Strategy and Logic

We tried many different approaches to initialize a BRAM. The best one, that we finally used, was to make a custom design for BRAM. The input and output ports were kept exactly the same as a typical BRAM. This was then instantiated in top level design file for using further.
For the logic of empty and full, we maintained an integer called element_count, which would keep track of the number of elements in the queue. These empty and full signals were then displayed using the LEDs.

One difficulty we faced originally was for the debounced button, but we correctly handled it using FSMs.

3 Code and Working

The main directory contains the design.vhd file, which has the VHDL code logic for the FIFO buffer, the BRAM.vhd file for the BRAM RTL, the multiDisplay.vhd file and the singleDisplay.vhd, which are the same files as submitted in assignment 2 and 3 (The singleDisplay file contains the logic for single Display of LED, and the multiDisplay file constains the logic for multiple digits display). The directory also has the constraint file and the generated bitstream.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;



entity design is
    Port (
        clk: in std_logic;

        write: in std_logic;
        write_data: in std_logic_vector(15 downto 0);

        read: in std_logic;

        empty: out std_logic;
        full: out std_logic;
        LED: out std_logic_vector(6 downto 0);
        anode: out std_logic_vector(3 downto 0)
    );

end design;

architecture Behavioral of design is

component BRAM
    generic(
        ADDRESS_WIDTH: integer := 8;
        DATA_WIDTH: integer := 16
    );
    port(
        clk: in std_logic;
        w_enable: in std_logic;
        write_address: in std_logic_vector(ADDRESS_WIDTH-1 downto 0);
        read_address: in std_logic_vector(ADDRESS_WIDTH-1 downto 0);
        datain: in std_logic_vector(DATA_WIDTH-1 downto 0);
        dataout: out std_logic_vector(DATA_WIDTH-1 downto 0)
    );

end component BRAM;

type write_states is (idle, processing,completed);
type read_States is (idle, processing,completed);

signal head             : std_logic_vector(7 downto 0)  :=  x"00";
signal tail             : std_logic_vector(7 downto 0)  :=  x"00";
signal is_empty         : std_logic;
signal is_full          : std_logic;
signal element_count    : std_logic_vector(7 downto 0) := x"00";
signal data             : std_logic_vector(15 downto 0):= x"0000";
signal slow_clock       : std_logic := '0';
signal counter          : integer := 0;
signal write_en         : std_logic := '0';
signal read_en          : std_logic := '0';
signal done_read        : std_logic := '0';
signal done_write       : std_logic := '0';
signal write_state      : write_states := idle ;
signal read_state       : read_states :=idle ;
signal addr             : std_logic_vector(7 downto 0) := x"FF";
begin

    addr <= tail - 1;
    myBRAM: entity work.BRAM(Behavioral) generic map(
        ADDRESS_WIDTH => 8,
        DATA_LENGTH =>  16
    )
    port map(
        clock => clk,
        write_enable => write,
        write_address => head,
        read_address => addr,
        datain => write_data,
        dataout => data
    );
    empty <= is_empty;
    full <= is_full;

    process(clk, write_en, read_en)
        begin
            if rising_edge(clk) then

                if(counter = 2500000) then
                    counter <= 0;
                    slow_clock <= not(slow_clock);
                else
                    counter <= counter + 1;
                end if;

                if(head < tail) then
                    element_count <= head - tail + x"FF";
                else
                    element_count <= head - tail;
                end if;

                if(element_count = x"00") then
                    is_empty <= '1';
                else
                    is_empty <= '0';
                end if;

                if(element_count = x"FF" - 1) then
                    is_full <= '1';
                else
                    is_full <= '0';
                end if;

            end if;
    end process;

    process(slow_clock, write, read)
        begin
            if(rising_edge(slow_clock) and slow_clock'event) then
                if(write = '1') then
                         case (write_state) is
                        when idle =>
                            write_en <= '1';
                            write_state <= processing;

                        when processing =>
                            write_en <='0';
                            write_state<=completed;

                        when others =>
                            write_en <='0';
                            write_state<=completed;

                          end case;
                        else
                        write_en <= '0';
                        write_state <= idle ;
                        end if;

            if(read = '1') then
                case (read_state) is
                    when idle =>
                        read_en <= '1';
                        read_state<=processing;

                    when processing =>
                        read_en<='0' ;
                        read_state<=completed;

                    when others =>
                        read_en<='0' ;
                        read_state<=completed;

                end case;

            else
                read_en <= '0';
                read_state<=idle;

            end if;

                if(write_en = '1' and is_full = '0') then
                    head <= (head + 1);

                end if;

                if(read_en = '1' and is_empty = '0') then
                    tail <= (tail + 1);
                end if;
            end if;
    end process;

    Multi_Display : entity work.lightDisplay(structure) port map (
        data, clk, LED, anode
        );

end Behavioral;
```

4 Simulation and Synthesis

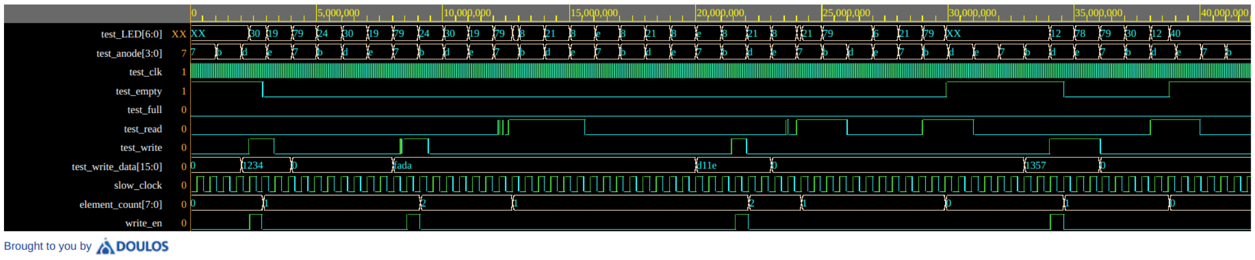

Figure 1: Generated EP Wave - 1

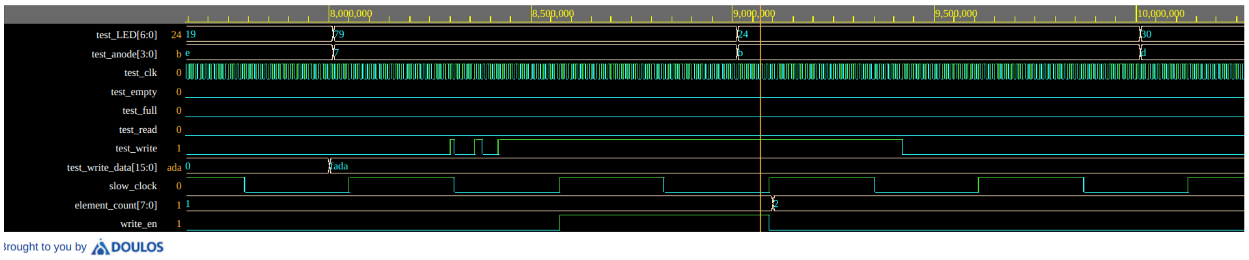

Figure 2: Generated EP Wave - 2

These EPWaves were carefully analysed to check if all outputs were correct!
The Vivado synthesis report is as follows:

```
Copyright 1986-2019 Xilinx, Inc. All Rights Reserved.
--------------------------------------------------------------------------------
| Tool Version : Vivado v.2019.1 (lin64) Build 2552052 Fri May 24 14:47:09 MDT 2019
| Date         : Sat May 28 20:01:17 2022
| Host         : divyanshu-HP-ENVY-x360-Convertible-13-bd0xxx
                 running 64-bit Ubuntu 20.04.2 LTS
| Command      : report_utilization -file design_utilization_synth.rpt
                 -pb design_utilization_synth.pb
| Design       : \design
| Device       : 7a35tcpg236-2
| Design State : Synthesized
--------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
```

```
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------

+----------------------------+------+-------+-----------+-------+
|          Site Type         | Used | Fixed | Available | Util% |
+----------------------------+------+-------+-----------+-------+
| Slice LUTs*                |   99 |     0 |     20800 |  0.48 |
|   LUT as Logic             |   99 |     0 |     20800 |  0.48 |
|   LUT as Memory            |    0 |     0 |      9600 |  0.00 |
| Slice Registers            |  103 |     0 |     41600 |  0.25 |
|   Register as Flip Flop    |  103 |     0 |     41600 |  0.25 |
|   Register as Latch        |    0 |     0 |     41600 |  0.00 |
| F7 Muxes                   |    0 |     0 |     16300 |  0.00 |
| F8 Muxes                   |    0 |     0 |      8150 |  0.00 |
+----------------------------+------+-------+-----------+-------+

1.1 Summary of Registers by Type
--------------------------------

+-------+--------------+-------------+--------------+
| Total | Clock Enable | Synchronous | Asynchronous |
+-------+--------------+-------------+--------------+
| 0     |            _ |           - |            - |
| 0     |            _ |           - |          Set |
| 0     |            _ |           - |        Reset |
| 0     |            _ |         Set |            - |
| 0     |            _ |       Reset |            - |
| 0     |          Yes |           - |            - |
| 0     |          Yes |           - |          Set |
| 0     |          Yes |           - |        Reset |
| 0     |          Yes |         Set |            - |
| 103   |          Yes |       Reset |            - |
+-------+--------------+-------------+--------------+


2. Memory
---------

+-------------------+------+-------+-----------+-------+
|     Site Type     | Used | Fixed | Available | Util% |
+-------------------+------+-------+-----------+-------+
| Block RAM Tile    |  0.5 |     0 |        50 |  1.00 |
|   RAMB36/FIFO*    |    0 |     0 |        50 |  0.00 |
|   RAMB18          |    1 |     0 |       100 |  1.00 |
|     RAMB18E1 only |    1 |       |           |       |
+-------------------+------+-------+-----------+-------+
* Note: Each Block RAM Tile only has one FIFO logic available and
        therefore can accommodate only one FIFO36E1 or one FIFO18E1.
        However, if a FIFO18E1 occupies a Block RAM Tile,
        that tile can still accommodate a RAMB18E1


3. DSP
------

+-----------+------+-------+-----------+-------+
| Site Type | Used | Fixed | Available | Util% |
+-----------+------+-------+-----------+-------+
| DSPs      |    0 |     0 |        90 |  0.00 |
+-----------+------+-------+-----------+-------+


4. IO and GT Specific
---------------------

+-----------------------------+------+-------+-----------+-------+
|          Site Type          | Used | Fixed | Available | Util% |
+-----------------------------+------+-------+-----------+-------+
| Bonded IOB                  |   32 |     0 |       106 | 30.19 |
| Bonded IPADs                |    0 |     0 |        10 |  0.00 |
| Bonded OPADs                |    0 |     0 |         4 |  0.00 |
| PHY_CONTROL                 |    0 |     0 |         5 |  0.00 |
| PHASER_REF                  |    0 |     0 |         5 |  0.00 |
| OUT_FIFO                    |    0 |     0 |        20 |  0.00 |
| IN_FIFO                     |    0 |     0 |        20 |  0.00 |
| IDELAYCTRL                  |    0 |     0 |         5 |  0.00 |
| IBUFDS                      |    0 |     0 |       104 |  0.00 |
| GTPE2_CHANNEL               |    0 |     0 |         2 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY   |    0 |     0 |        20 |  0.00 |
| PHASER_IN/PHASER_IN_PHY     |    0 |     0 |        20 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY |    0 |     0 |       250 |  0.00 |
| IBUFDS_GTE2                 |    0 |     0 |         2 |  0.00 |
| ILOGIC                      |    0 |     0 |       106 |  0.00 |
| OLOGIC                      |    0 |     0 |       106 |  0.00 |
+-----------------------------+------+-------+-----------+-------+


5. Clocking
-----------

+------------+------+-------+-----------+-------+
|  Site Type | Used | Fixed | Available | Util% |
+------------+------+-------+-----------+-------+
| BUFGCTRL   |    1 |     0 |        32 |  3.13 |
| BUFIO      |    0 |     0 |        20 |  0.00 |
| MMCME2_ADV |    0 |     0 |         5 |  0.00 |
| PLLE2_ADV  |    0 |     0 |         5 |  0.00 |
| BUFMRCE    |    0 |     0 |        10 |  0.00 |
| BUFHCE     |    0 |     0 |        72 |  0.00 |
| BUFR       |    0 |     0 |        20 |  0.00 |
+------------+------+-------+-----------+-------+
6. Specific Feature
-------------------

+-------------+------+-------+-----------+-------+
|  Site Type  | Used | Fixed | Available | Util% |
+-------------+------+-------+-----------+-------+
| BSCANE2     |    0 |     0 |         4 |  0.00 |
| CAPTUREE2   |    0 |     0 |         1 |  0.00 |
| DNA_PORT    |    0 |     0 |         1 |  0.00 |
| EFUSE_USR   |    0 |     0 |         1 |  0.00 |
| FRAME_ECCE2 |    0 |     0 |         1 |  0.00 |
| ICAPE2      |    0 |     0 |         2 |  0.00 |
| PCIE_2_1    |    0 |     0 |         1 |  0.00 |
| STARTUPE2   |    0 |     0 |         1 |  0.00 |
| XADC        |    0 |     0 |         1 |  0.00 |
+-------------+------+-------+-----------+-------+


7. Primitives
-------------

+----------+------+---------------------+
| Ref Name | Used | Functional Category |
+----------+------+---------------------+
| FDRE     |  103 |        Flop & Latch |
| LUT2     |   54 |                 LUT |
| CARRY4   |   29 |          CarryLogic |
| LUT4     |   27 |                 LUT |
| LUT1     |   26 |                 LUT |
| IBUF     |   19 |                  IO |
| OBUF     |   13 |                  IO |
| LUT6     |   12 |                 LUT |
| LUT5     |    8 |                 LUT |
| LUT3     |    7 |                 LUT |
| RAMB18E1 |    1 |        Block Memory |
| BUFG     |    1 |               Clock |
+----------+------+---------------------+


8. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


9. Instantiated Netlists
------------------------

+----------+------+
| Ref Name | Used |
+----------+------+
```