طراحی سیستم های دیجیتال 1402-1403

تمرین 2

سیده شکیبا انارکی فیروز 99442047

(Q1

- Suggest a scenario where aliases could be beneficial in VHDL design.

In VHDL design, using aliases can simplify complex signal access within hierarchical structures. By creating aliases for frequently accessed signals, you can use shorter and more readable names in your code, improving code clarity and maintainability. This code can be a good example for it:

```
entity TopLevel is
    port (
        clk : in std_logic;
        reset : in std_logic;
        data_out : out std_logic
    );
end entity TopLevel;

architecture Behavioral of TopLevel is
    signal signalA : std_logic;
    signal signalB : std_logic;
    signal signalC : std_logic;
    -- Alias declaration to simplify signal access
    alias my_alias : std_logic is signalC;
begin
    process (clk, reset)
    begin
        if reset = '1' then
            data_out <= '0';
        elsif rising_edge(clk) then
            data_out <= my_alias;
        end if;
    end process;
end architecture Behavioral;
```

As we see in the code using the alias my_alias improves code readability and makes it easier to maintain, especially in designs with multiple nested components.

- When are the assignement operators <= , := and => typically utilized in VHDL code?

- Operator "<=" is used to assign a value to a SIGNAL within processes or concurrent signal assignments in architecture bodies. It reflects changes in the hardware model.
- Operator ":=" is used to assign a value to a VARIABLE, CONSTANT, or GENERIC. Used also for establishing initial values.
- Operator "=>" is used to assign values to individual vector elements or with OTHERS. Also used for associating values with labels in record declarations, port maps, and generic maps.

- What do language defined attributes do?

Attributes are a feature of VHDL that allow you to extract additional information about an object (such as a signal, variable or type) that may not be directly related to the value that the object carries. . Attributes also allow you to assign additional information (such as data related to synthesis) to objects in your design description. Some common attributes include 'event' for checking signal changes, 'active' for detecting actively driven signals, 'stable' for checking signal stability, 'transaction' for accessing transaction details, and attributes like 'left', 'right', 'high', and 'low' for array indexing.

- What is the difference between std_logic and std_ulogic types in VHDL?

They both can represent these values: 'U', 'X', '0', '1', 'Z', 'W', 'L' ,'H' and '-'.

- **std_logic** is a resolved type means it uses a resolution table to handle multiple drivers on a signal. The resolved function determines the output value based on the values of the driving signals. So the codes with this data type can be compiled successfully.
- **std_ulogic** is an unresolved type so it does not have a resolution function. It is primarily used for simulation purposes where unresolved values ('U', 'X') may occur due to uninitialized signals or unknowns. Conflicting drivers for this type generates error in VHDL and the simulation won't be compiled and synthesized.

- What is the usage of conv_std_logic_vector(p,b) and conv_signed(p.b)?

conv_std_logic_vector(p, b):

Converts a parameter p of type INTEGER, UN-SIGNED,SIGNED, or STD_LOGIC to a STD_LOGIC_VECTOR value with size b bits.

Here is an example for its usage:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_arith.all;
...
SIGNAL a: IN UNSIGNED (7 DOWNTO 0);
SIGNAL b: IN UNSIGNED (7 DOWNTO 0);
SIGNAL y: OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
...
y <= CONV_STD_LOGIC_VECTOR ((a+b), 8);
```

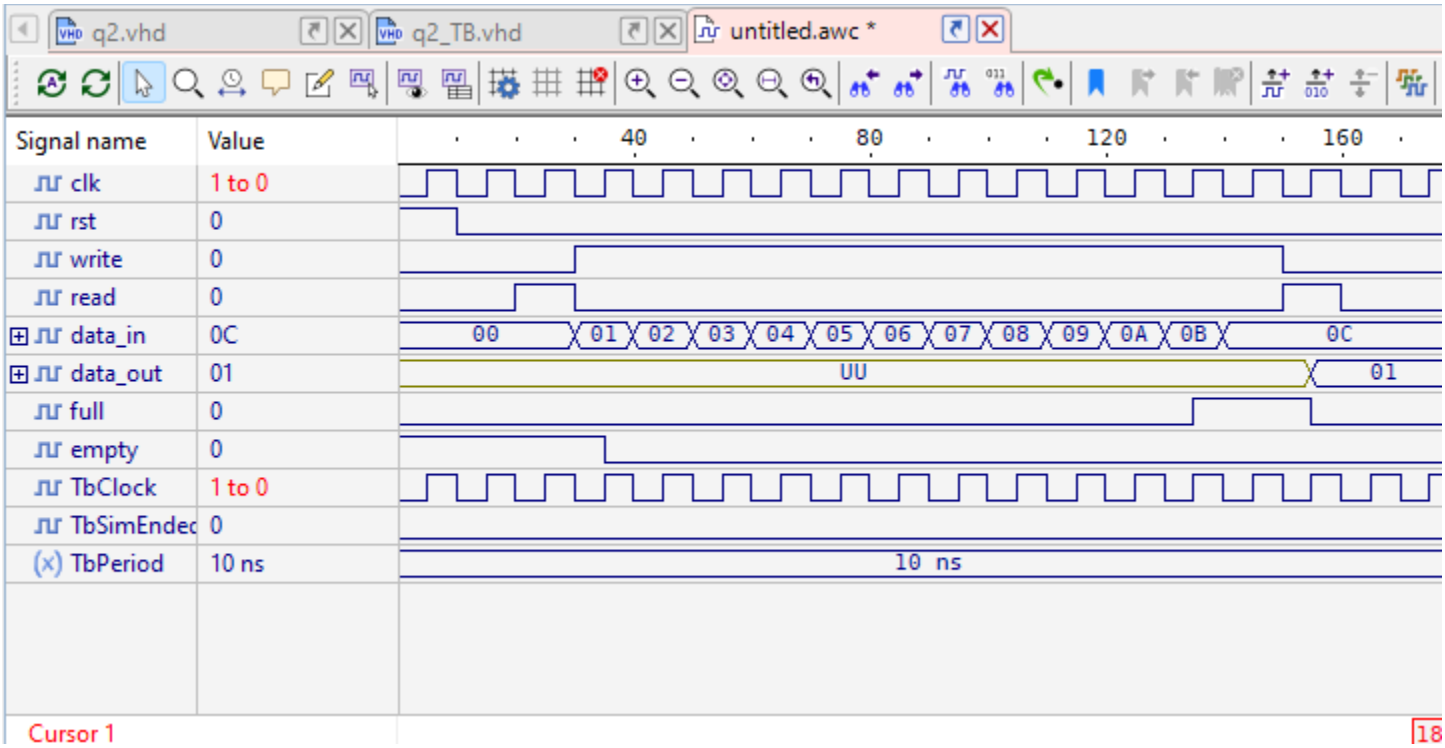
a+b is converted from UNSIGNED to an 8-bit STD_LOGIC_VECTOR value, then assigned to y.

conv_signed(p, b):

It is a data conversion function of std_logic_arith package of library IEEE. It converts a parameter p of type INTEGER, UNSIGNED, SIGNED,or STD_ULOGIC to a SIGNED value with size b bits.

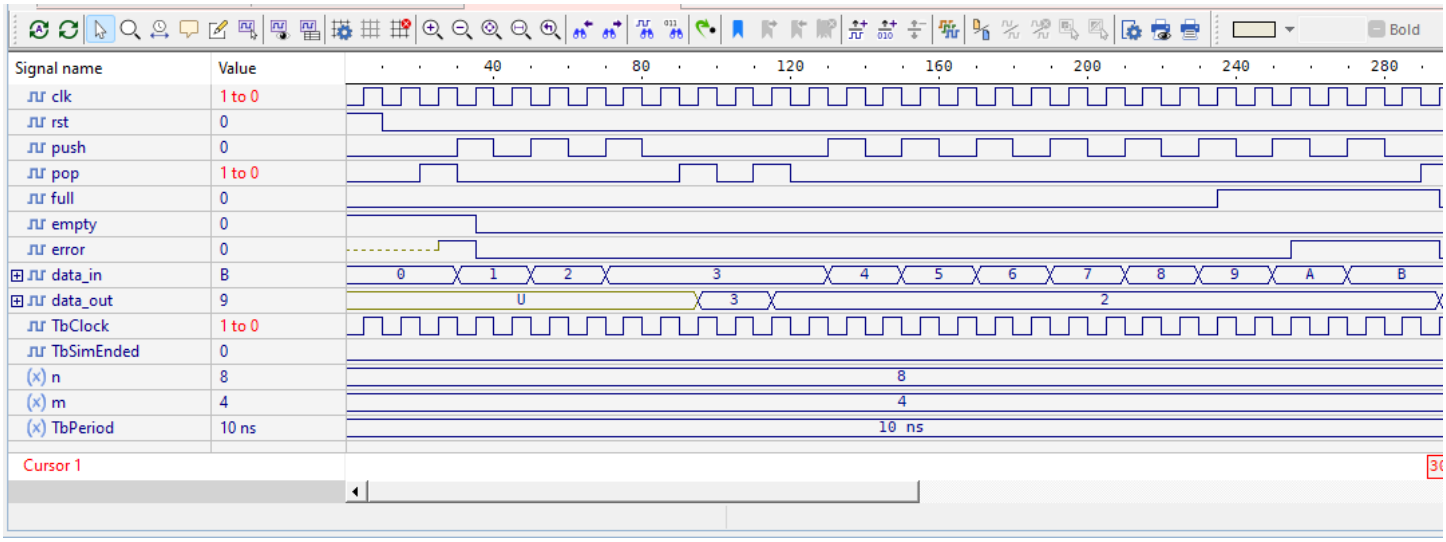
(Q2

Waveform:



(Q3

Wave form:



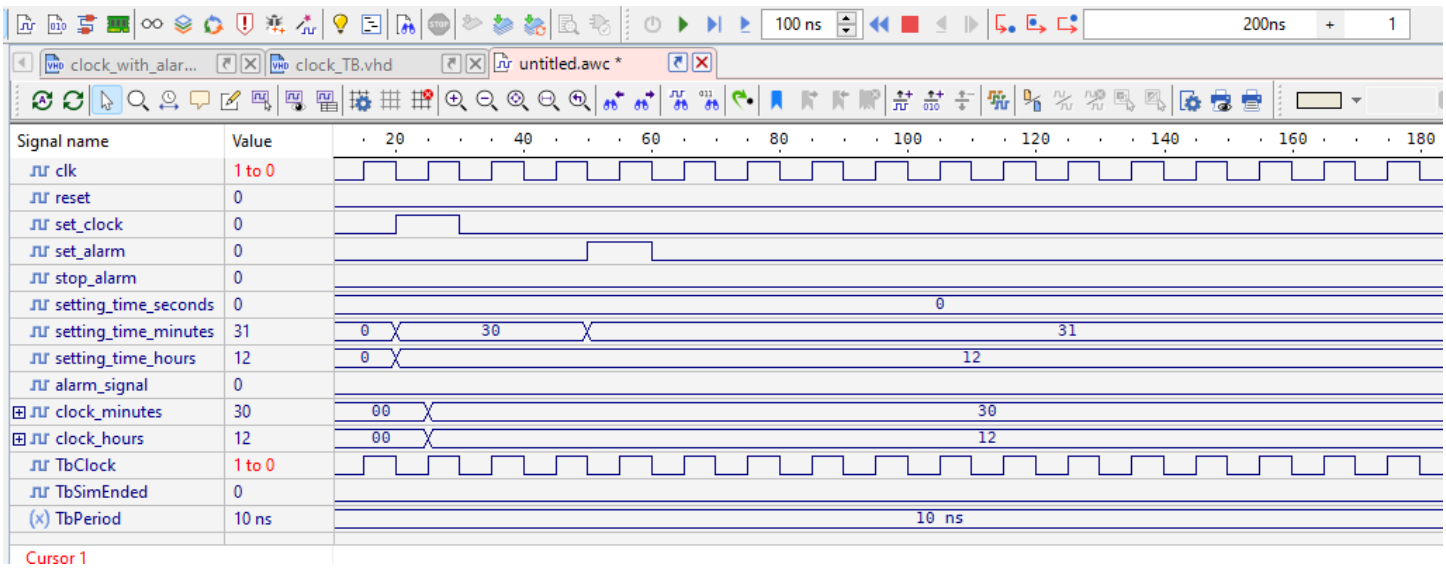
Challenge I faced with during coding:

I was reducing the top value before popping from the stack which was leading to popping from array with wrong index and the value was zero. Then I reduced the top value after popping and it was fixed (I guess the reason was assigning delay in signals)

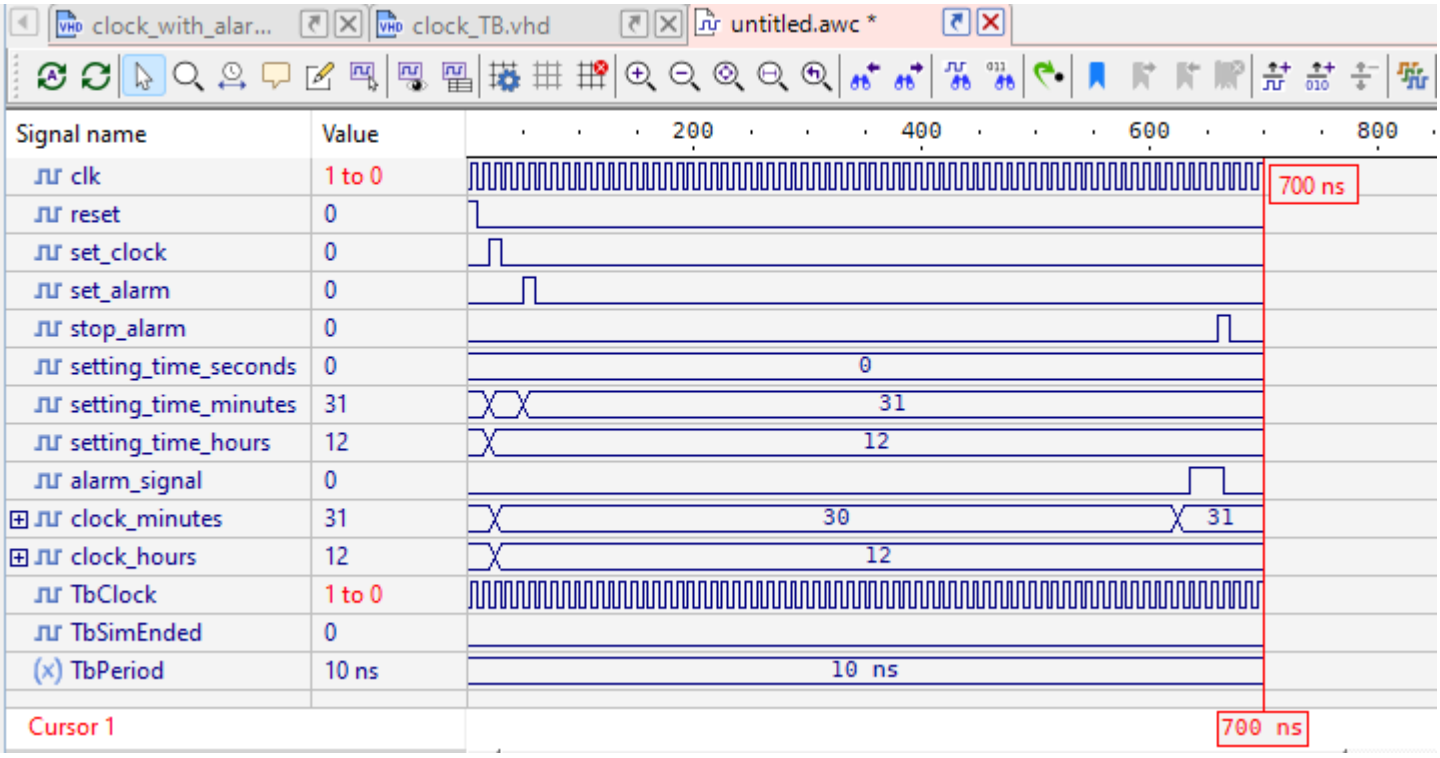
```
41              elsif pop = '1' then -- Pop operation
42                  if top > 0 then -- Stack is not empty
43                      data_out <= stack(top -1);
44                      top <= top - 1;
45                      error <= '0';
46                  else
47                      error <= '1'; -- Stack empty error
48                  end if;
49              end if;
```

(Q4

Wave form first part : (setting the clock and reset)



Waveform second part: (alarm and stop alarm signals)



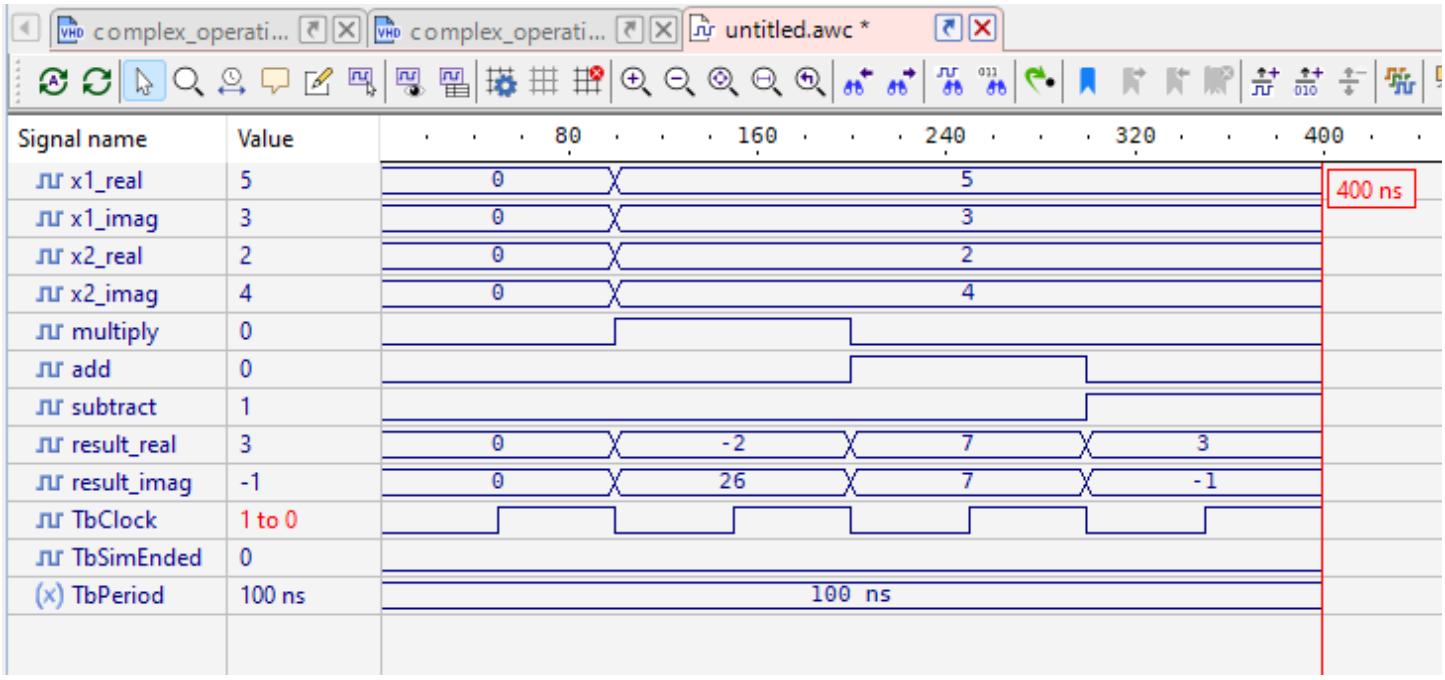
Challenge I faced: I used to set all of clock and alarm values to zero. But It leads alarm_signal to be one after resetting, so I changed the current time to 00:00:01 and alarm to 00:00:00 to prevent this phenomenon.

(Q5

Type definition part:

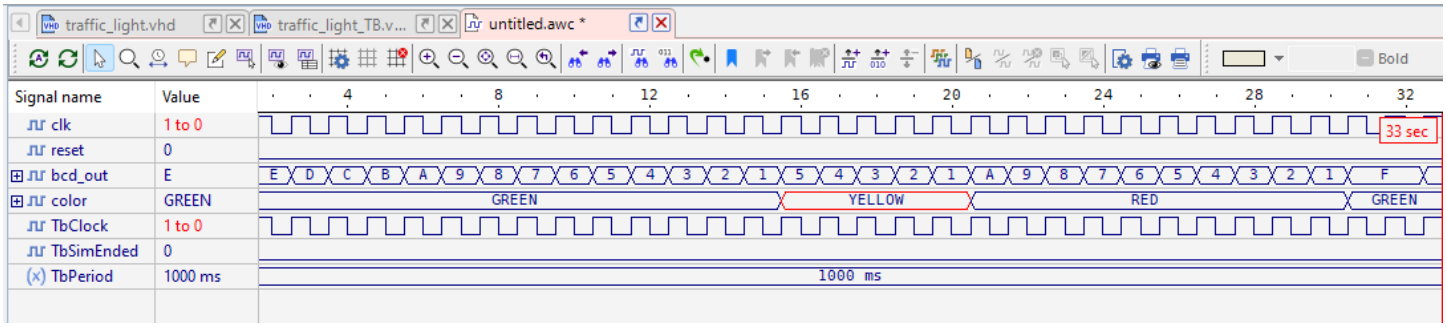
```
12
13  architecture behavioral of complex_operations is
14      -- Defining the type
15      type complex_number is record
16          real_part: integer;
17          imag_part: integer;
18      end record;
19
20      -- Signals to hold the input complex numbers
21      signal x1, x2: complex_number;
22
23  begin
```

Waveform picture is in the next page.



(Q6

Waveform:



Challenges: I had to initialize some additional signals to hold the counter as a countdown number for each light. Also synchronizing the clock with the counter was a bit tricky.

- Used links for Q1

http://pldworld.info/_hdl/2/_ref/acc-eda/language_overview/objects__data_types_and_operators/understanding_vhdl_attributes.htm#:~:text=Attributes%20are%20a%20feature%20of,value%20that%20the%20object%20carries.

https://gear.kku.ac.th/~watis/courses/198323/slide05.pdf

https://ece477.cankaya.edu.tr/uploads/files/LEC5%281%29.pdf