MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY OF UKRAINE

" IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE"

A. M. Sergiyenko

A. A. Molchanova

CAD of computer systems

Lab Work 2

Sequence Detector

Kulubecioglu Mehmet

Class Number 12

IM-14 FIOT

Kyiv

IHORY SIKORSKY KYIV POLYTECHNIC INSTITUTE

2024

1. sequence_detector.vhd (Main Design File)

This file contains the design for a sequence detector that recognizes a specific sequence (e.g., "1011"). It is written in VHDL and uses a Finite State Machine (FSM) to detect the sequence.

In the Entity section, input and output ports are defined. The inputs include the clock (clk), reset (reset), and the bit of the sequence being tested (input_bit). The output is a signal (detected) that indicates whether the sequence has been successfully detected.
In the Architecture section, the sequence detector is modeled as a finite state machine with five states: IDLE (idle state), S1, S2, S3, and S4. Each state corresponds to a particular step in the sequence detection process.
State transitions and detection conditions are defined in this section. For example, when the sequence of input_bit is correct, the detected output is set to high (1), indicating that the sequence has been detected successfully.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity sequence_detector is
    port (
        clk      : in std_logic;       -- Saat sinyali
        reset    : in std_logic;       -- Reset sinyali
        input_bit : in std_logic;      -- Giriş biti
        detected : out std_logic       -- Algılama çıktısı
    );
end sequence_detector;

architecture Behavioral of sequence_detector is
    type state_type is (IDLE, S1, S2, S3, S4); -- Durumlar
    signal current_state, next_state : state_type;
begin

    -- Durum geçiş işlemi
    process (clk, reset)
    begin
        if reset = '1' then
            current_state <= IDLE;
        elsif rising_edge(clk) then
            current_state <= next_state;
        end if;
    end process;

    -- Durum tablosu
    process (current_state, input_bit)
    begin
        -- Varsayılan değerler
        next_state <= IDLE;
        detected <= '0';

        case current_state is
            when IDLE =>
                if input_bit = '1' then
                    next_state <= S1;
                else
                    next_state <= IDLE;
                end if;
```

```
42                  end if;
43
44              when S1 =>
45                  if input_bit = '1' then
46                      next_state <= S2;
47                  else
48                      next_state <= IDLE;
49                  end if;
50
51              when S2 =>
52                  if input_bit = '1' then
53                      next_state <= S3;
54                  else
55                      next_state <= IDLE;
56                  end if;
57
58              when S3 =>
59                  if input_bit = '1' then
60                      next_state <= S4;
61                  else
62                      next_state <= IDLE;
63                  end if;
64
65              when S4 =>
66                  detected <= '1'; -- Dizi algılandı
67                  next_state <= IDLE;
68          end case;
69      end process;
70  end Behavioral;
71
```

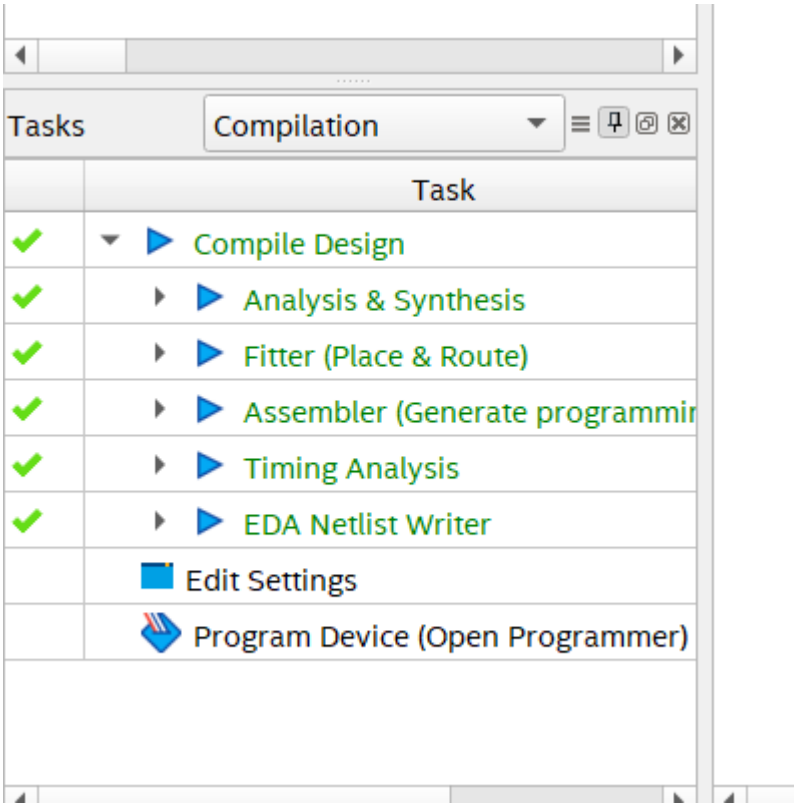
2. sequence_detector_tb.vhd (Testbench File)

This file is the testbench used to simulate the sequence_detector.vhd design. The testbench provides input signals and observes the outputs to verify if the design works correctly.

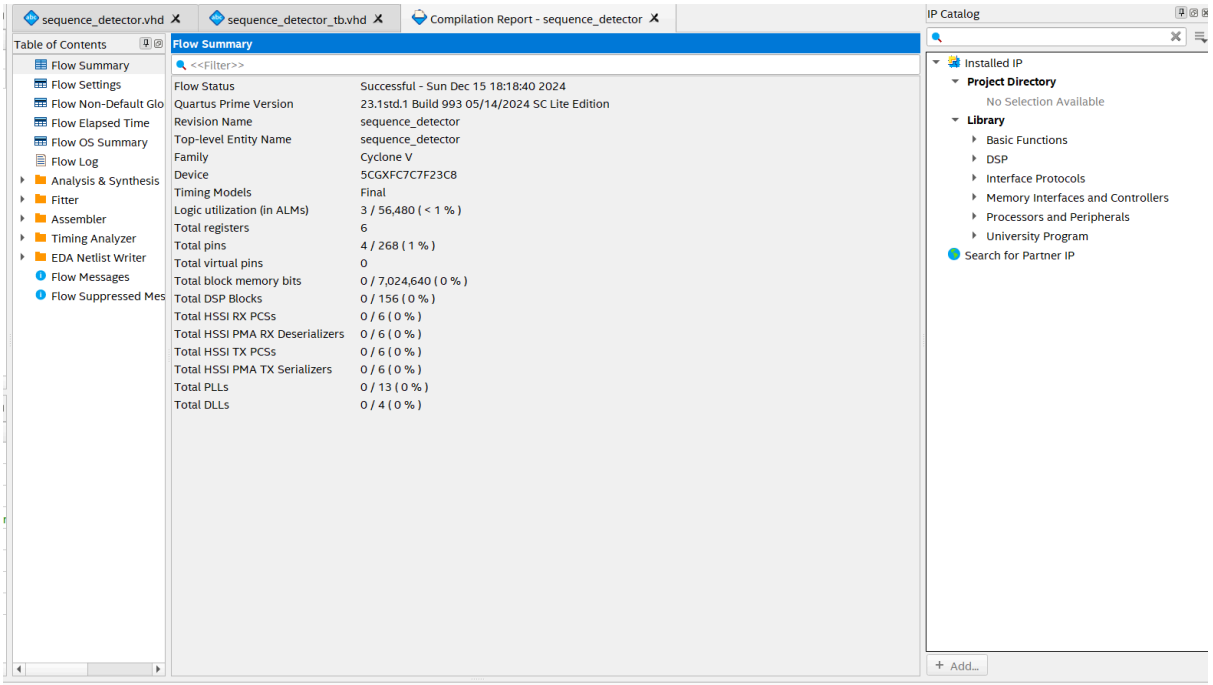
The purpose of the testbench file is to simulate and test the functionality of the design. The sequence_detector component is instantiated and input signals (clock, reset, and input bits) are applied during simulation.
The clock signal (clk) is generated at a specific frequency, typically 50 MHz, to provide the timing for the design.
During the stimulus process, the test input (input_bit) is set to the sequence "1011", and the design's response is monitored to check whether it detects the sequence correctly. The reset signal is asserted initially and then the input bits are applied sequentially.
The testbench initiates the simulation and observes the output signal, detected, to verify if the design behaves as expected.
This file provides the test environment for simulating the sequence_detector.vhd module by applying test input and timing, allowing the design's functionality to be verified.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity sequence_detector_tb is
end sequence_detector_tb;

architecture Behavioral of sequence_detector_tb is
    signal clk      : std_logic := '0';
    signal reset    : std_logic := '0';
    signal input_bit : std_logic := '0';
    signal detected : std_logic;

    component sequence_detector
        port (
            clk      : in std_logic;
            reset    : in std_logic;
            input_bit : in std_logic;
            detected : out std_logic
        );
    end component;

begin
    uut: sequence_detector
        port map (
            clk => clk,
            reset => reset,
            input_bit => input_bit,
            detected => detected
        );

    -- Saat sinyali üretimi
    clk_process : process
    begin
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
    end process;

    -- Test süreci
    stim_proc: process
    begin
        -- Reset
        reset <= '1';
        wait for 20 ns;
        reset <= '0';

        -- 0xcccc dizisini gönder
        input_bit <= '1';
        wait for 20 ns;
        input_bit <= '1';
        wait for 20 ns;
        input_bit <= '1';
        wait for 20 ns;
        input_bit <= '1';
        wait for 20 ns;

        wait;
    end process;
end Behavioral;

```

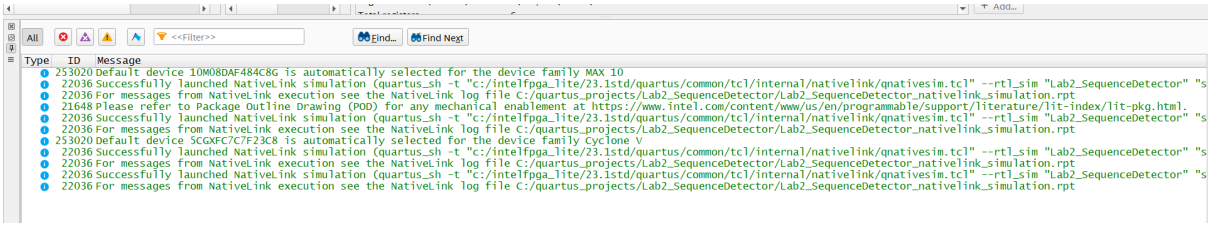
3- I compiled the codes and observed whether there was a line and noticed that there was no error.



4- Compilation report



4- start simulation

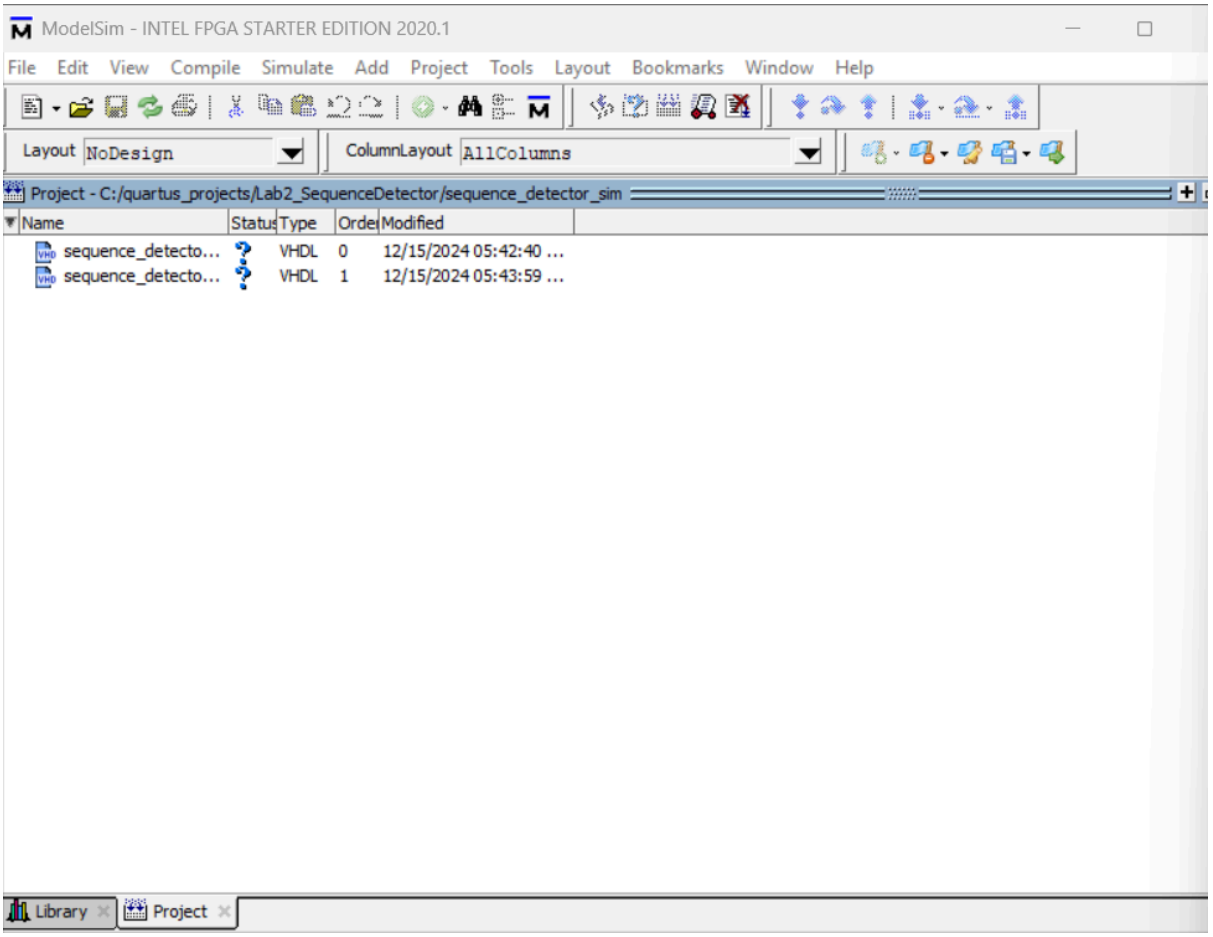


5- Modelsim

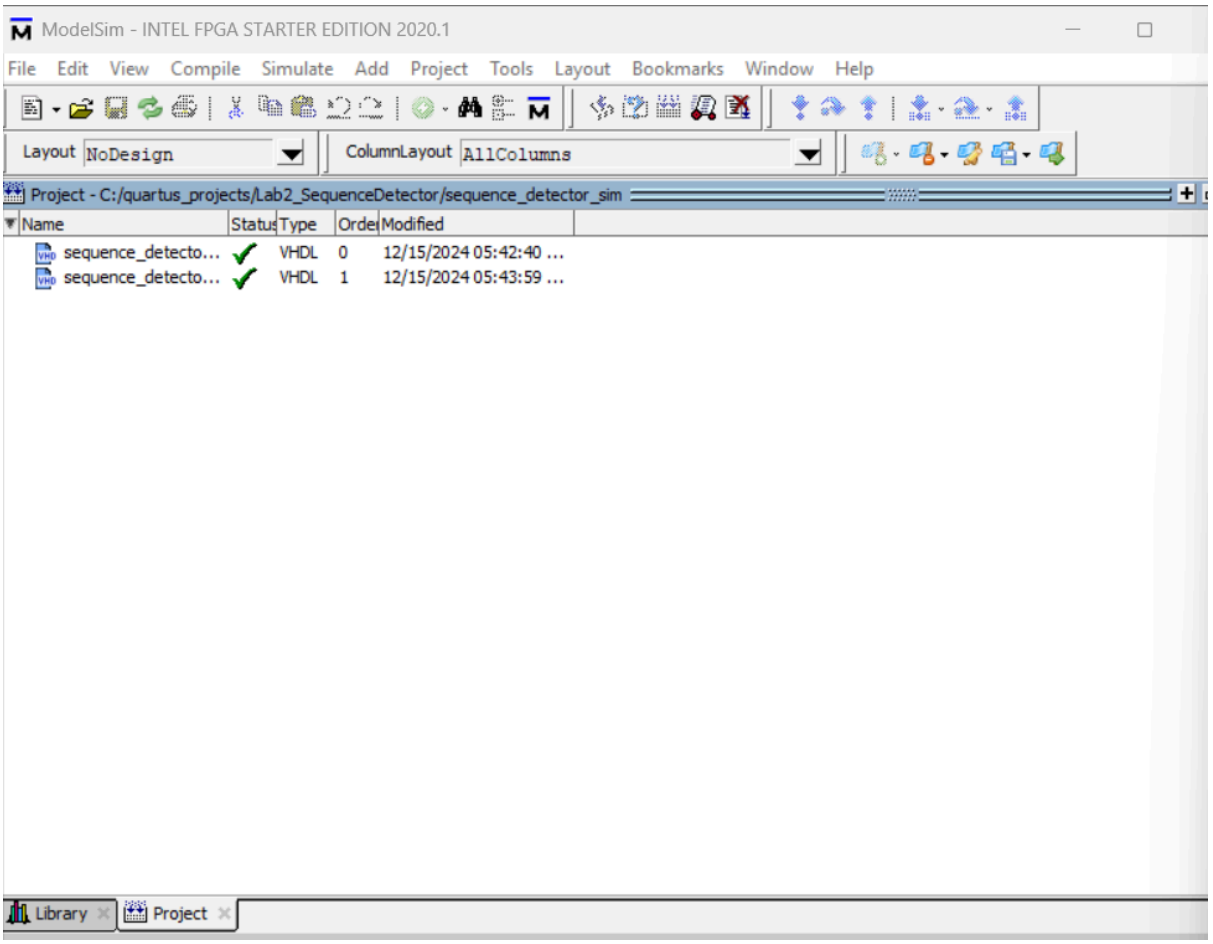
- I created a new project by selecting File > New > Project.

The project name was set to sequence_detector_sim, and the file path was selected.

- I added the design files (such as sequence_detector.vhd and sequence_detector_tb.vhd) to the project using Add > To Project > Existing File.

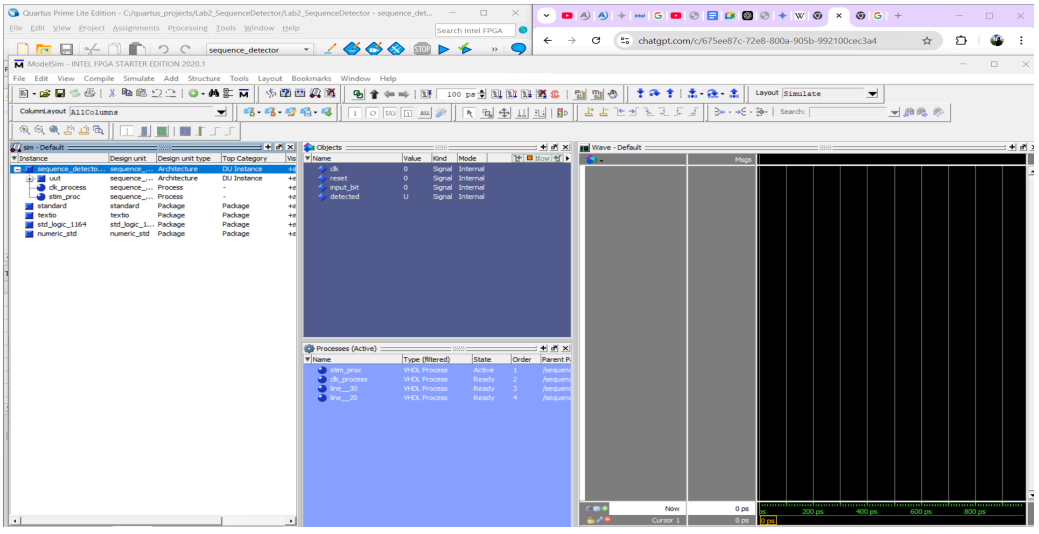


- I compiled all the files using the Compile > Compile All command.



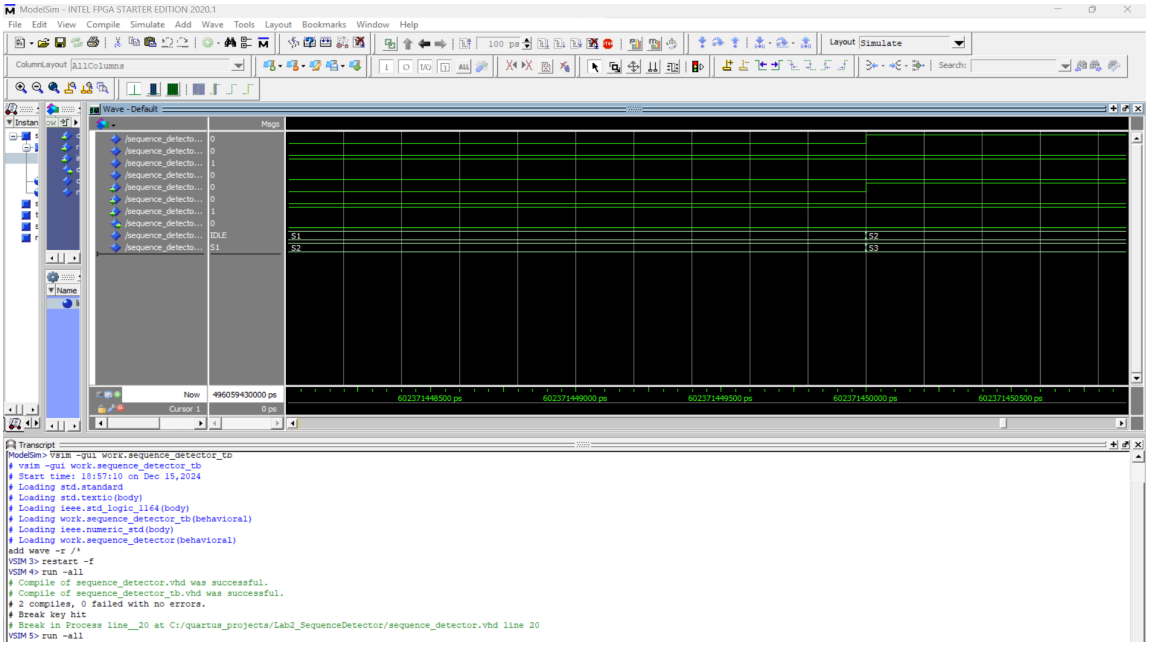
6 - Simulation With Modelsim

- I started the simulation using Simulate > Start Simulation and selected the testbench file.



7- To Wave

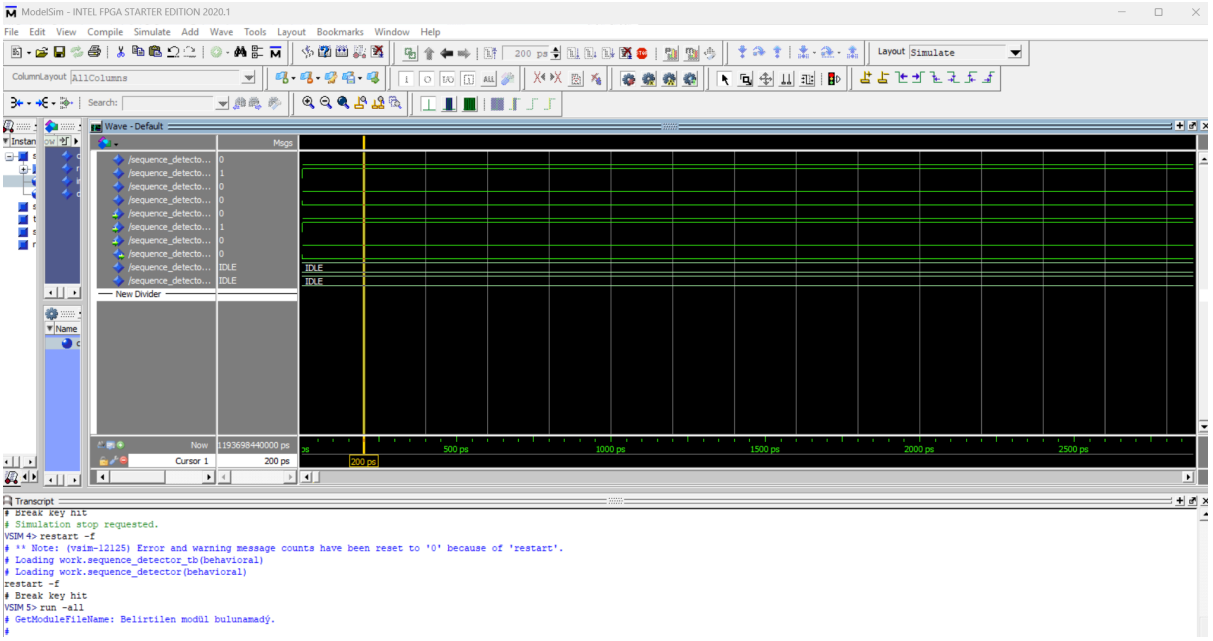
- I added all the signals in the project to the waveform by selecting Add > To Wave > All Items in Design.



8 Run for 200ns

- I ran the simulation using Run > Run All and let it run for 200ns.

At the end of the simulation, we observed and verified the output for correctness.



1. What is an automata language?

- An automata language refers to the set of strings that can be accepted or recognized by an automaton (a finite state machine, FSM). It consists of sequences of symbols from a given alphabet that the automaton can process and accept based on its state transitions.

2. What is a terminal symbol of a language?

- A terminal symbol (also called a token) is a symbol that appears in the strings of a language and cannot be further broken down or replaced by other symbols. In the context of regular expressions, terminal symbols represent the basic building blocks of a language (e.g., letters, digits).

3. What is a regular expression and what is it used for?

- A regular expression (regex) is a pattern used to match strings in a language. It is used for text searching, pattern matching, and validating input. It allows defining the structure of strings using symbols like *, +, and ?, among others.

4. What does the Kleene asterisk mean?

- The Kleene asterisk (*) is an operator in regular expressions that signifies zero or more occurrences of the preceding element. For example, a* matches "", "a", "aa", "aaa", and so on.

5. What is the union operation used in a regular expression?

- The union operation (denoted by | in regular expressions) specifies an alternation between two expressions. For example, a|b matches either "a" or "b".

6. The alphabet consists of the letters a, b, c. What does a regular expression that describes all words from these letters look like?

- A regular expression that describes all possible words formed from the alphabet {a, b, c} is:

[abc]*

This means any combination (including the empty string) of "a", "b", and "c".

7. Write a regular expression that recognizes the words aba, abababa.

- A regular expression that matches "aba" or "abababa" is:

(aba)+

This matches one or more repetitions of "aba", so it would match both "aba" and "abababa".

8. What does the nodal bar on the load of an edge of the FSM diagram mean?

- In an FSM diagram, the nodal bar on the load of an edge typically represents a latching or storage operation, where the state of a variable or signal is stored or updated when the FSM transitions.

9. How does a deterministic FSM differ from a nondeterministic one?

- A deterministic FSM (DFA) has exactly one state transition for each input symbol from any given state. In contrast, a nondeterministic FSM (NFA) can have multiple possible state transitions for a given input symbol, or even no transition at all.

10. Why should an FSM implemented in an FPGA be deterministic?

- An FSM implemented in an FPGA should be deterministic because FPGA implementations are based on logic circuits that must follow predictable paths. Non-deterministic behavior would require more complex logic to manage all possible transitions, which is inefficient and difficult to implement in hardware.

11. What are FIFO buffers used for?

- FIFO (First In, First Out) buffers are used to temporarily store data in a way that ensures the first data written to the buffer is the first data to be read. They are used in applications where data is transferred between processes or devices at different speeds or times, such as in communication protocols or data streaming.

12. How is a FIFO buffer implemented in an FPGA?

- A FIFO buffer in an FPGA is typically implemented using RAM (Random Access Memory), where data is written to one end and read from the other end, ensuring that data follows the FIFO order. Control logic manages the write and read pointers, as well as the empty and full flags.

13. What does the depth of a FIFO buffer depend on?

- The depth of a FIFO buffer depends on the required amount of data storage. It is determined by the number of elements that need to be temporarily stored in the buffer before being processed or transmitted.

14. Why is a FIFO buffer implemented in RAM, and not on a register chain?

- A FIFO buffer is implemented in RAM instead of a register chain because RAM allows more flexible and scalable storage for larger amounts of data, while a register chain would be inefficient for handling large data volumes or variable-length data.

15. What is the essence of a DFG and why is it used in device design?

- A DFG (Data Flow Graph) is a representation of the data dependencies between operations in a system. It is used in device design to optimize the flow of data and to visualize how computations and data transfer operations relate to each other, ensuring efficient resource utilization.

16. How is a FIFO buffer full signal generated in a detector circuit?

- A FIFO buffer full signal is generated when the write pointer reaches the last available location in the buffer. The control logic detects this condition and sets the full flag, preventing further data writes until space becomes available.

17. What is a sensitivity list?

- A sensitivity list in VHDL defines which signals a process is sensitive to. The process will be triggered (executed) whenever one of the signals in the list changes value.

18. How do you convert an octal character code to a binary number?

- To convert an octal number to a binary number, replace each octal digit with its 3-bit binary equivalent. For example, octal 7 is 111 in binary, octal 5 is 101, and so on.

19. How do sensitivity lists differ in process operators in the detector example and why?

- In the sequence detector example, the sensitivity list for the state transition process includes clk and reset to ensure the process is triggered on clock edges and when reset occurs. The sensitivity list for the state output process, on the other hand, may include current_state and input_bit, as the output depends on the current state and input bits.

20. How is an FSM programmed in VHDL?

- An FSM is programmed in VHDL by defining its states and transitions in a process block. State transitions are controlled by input signals and clock edges, and the next state is updated on each clock cycle.

21. How is a shift register programmed in VHDL?

- A shift register is programmed in VHDL by defining a register that shifts its stored value on each clock cycle. This is typically done using a signal or variable to hold the data and updating it with the previous value shifted by one position.

22. What does the label on a process operator mean?

- The label on a process operator in VHDL is an identifier for the process, used to distinguish between different processes within the same design. Labels are optional but help organize and document the code.

23. Why do literals usually have a type-defining prefix, rather than a suffix?

- Literals in VHDL often have a type-defining prefix (e.g., 4'b1101 for a 4-bit binary literal) because VHDL requires explicit type definitions to ensure the literal is treated in the correct context. Prefixes clarify the data type (binary, decimal, hexadecimal) and size, reducing ambiguity in the design.