CSSE4010 – 2025 semester 2 - Lab 1

Introduction to Xilinx/AMD Vivado, VHDL and Nexys Board

Student name: Adnaan Buksh

Student number:47435568

Lab Session attended: Wednesday 12-2pm

Submission date: 11/08/2025, 16:00 (UTC+10)

(There is no page limit for reports, however you should not make it overly long)

Introduction

This lab introduced VHDL design and simulation. The task was to implement a 4 input prime number detector using dataflow modelling. Where the Output 'F' goes high when the binary input represents a prime number. The true table was developed followed by the k-map giving us the simplified logic expression that was implemented in VHDL and functionally verified through simulations.

Design Description

The system receives a 4-bit binary input **A,B,C,D,** where **A** is the most significant bit (MSB) and **D** is the least significant bit (LSB) as shown in figure 1. The binary input represents an unsigned integer from 0 to 15. The output **F** is asserted high (‘1’) only if the input corresponds to a prime number within this range.

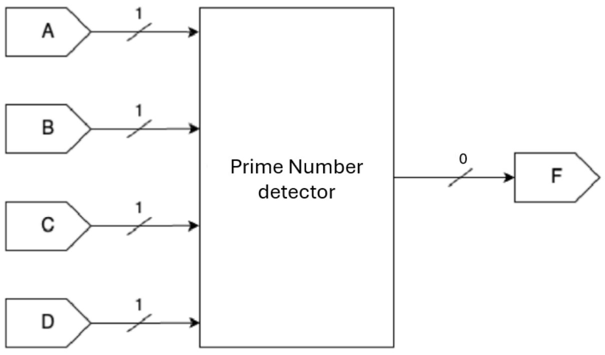


Figure 1 Block diagram for 4-input prime number detector

To design the system, the first step was to construct a truth table as shown in table 1, listing all possible 4-bit combinations and identifying which correspond to prime numbers. Within the 0–15 range, the prime numbers are: 2, 3, 5, 7, 11, and 13. All other values should produce an output F=0.

Table 2 True table for 4-input prime number detector

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Using this truth table, a **(K-map)** was constructed to simplify the logic.

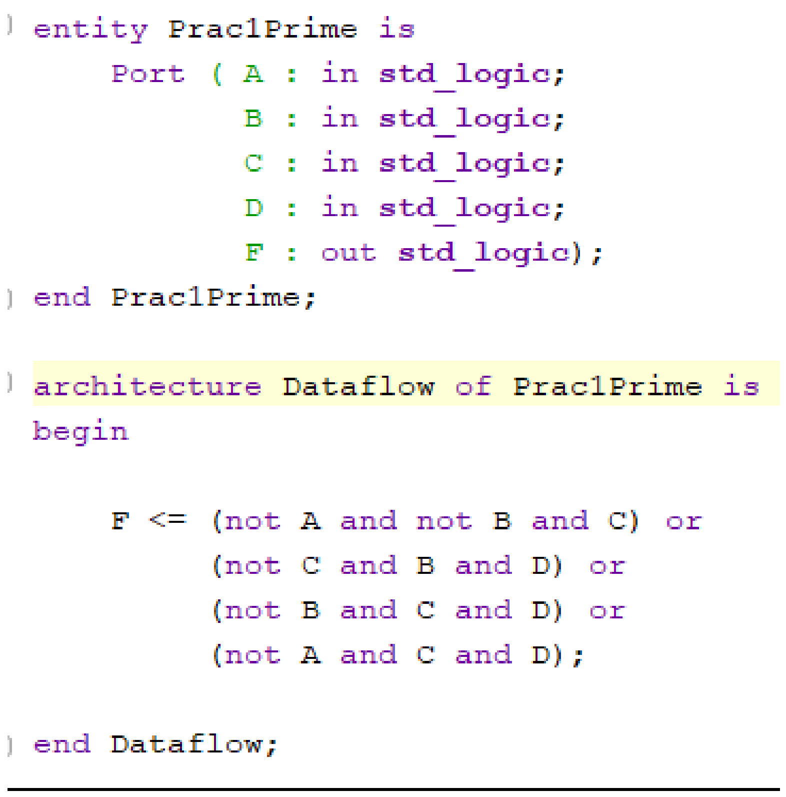
Table 2 K-map for 4-input prime number detector

AB\CD	00	01	11	10
00	0	0	1	1
01	0	1	1	0
11	0	1	0	0
10	0	0	1	0

The final simplified Boolean expression for F was found to be:

$$F = \bar{A}\bar{B}C + B\bar{C}D + \bar{B}CD + \bar{A}CD$$

The design was implemented using **VHDL with dataflow abstraction**, translating the above Boolean equation directly into VHDL as shown in figure 2.



```
entity Prac1Prime is
    Port ( A : in std_logic;
           B : in std_logic;
           C : in std_logic;
           D : in std_logic;
           F : out std_logic);
end Prac1Prime;

architecture Dataflow of Prac1Prime is
begin

    F <= (not A and not B and C) or
         (not C and B and D) or
         (not B and C and D) or
         (not A and C and D);

end Dataflow;
```

Figure 2 Boolean equation in VHDL

Simulation Results

To verify the functionality of the prime number detector, behavioural simulations were performed in Vivado using a self-checking testbench. The testbench systematically applied all 16 possible 4-bit combinations (from 0000 to 1111), corresponding to decimal values 0 to 15, and compared the output against the expected results.

The waveform confirms correct output values for all test cases. Output F is asserted high ('1') only for binary inputs corresponding to prime numbers: 2, 3, 5, 7, 11, and 13. For all other input values, F remains low ('0'), which matches the expected behaviour as shown in figure 3.

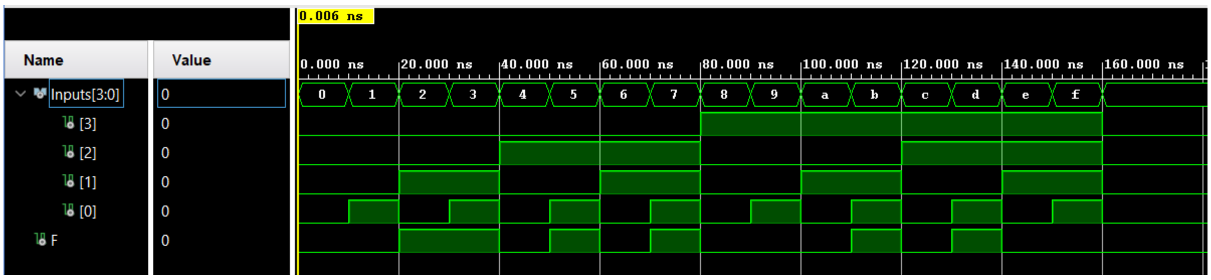


Figure 3 Correct functional simulation of design.

To simulate a propagation delay, a 10 ns delay was explicitly added to the dataflow architecture using the **after 10 ns** clause in the signal assignment. The waveform demonstrates that the output F changes 10 ns after any change in the input vector, emulating real hardware delay.

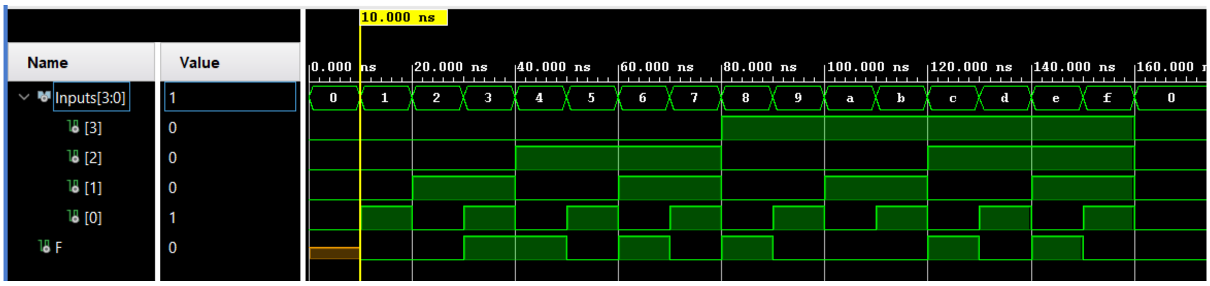
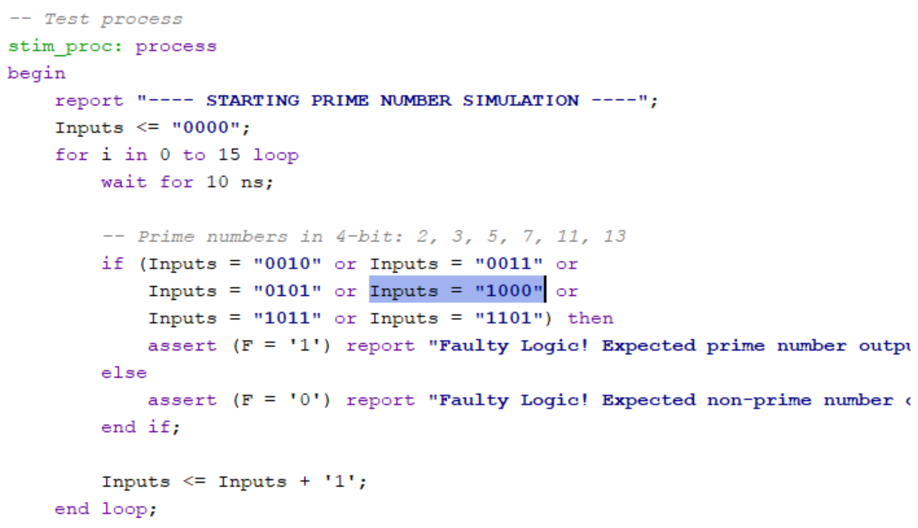


Figure 4 10ns output delay

To demonstrate the testbench's error-detection capability, an intentional bug was introduced in the VHDL logic by replacing a valid prime (e.g., 7 = 0111) with a non-prime (e.g., 8 = 1000) as shown in figure 5. This mismatch triggered an assertion failure in the simulation, as shown in figure 6. The waveform also confirms the incorrect logic, helping verify that the testbench is working correctly.

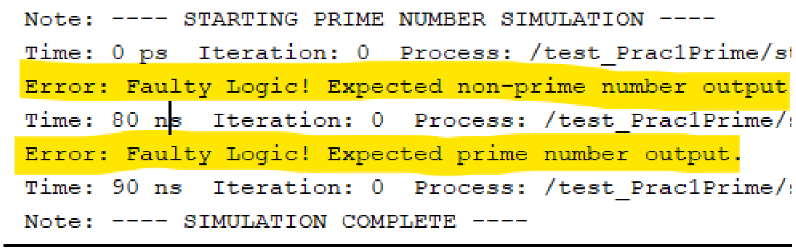


```
-- Test process
stim_proc: process
begin
    report "---- STARTING PRIME NUMBER SIMULATION ----";
    Inputs <= "0000";
    for i in 0 to 15 loop
        wait for 10 ns;

        -- Prime numbers in 4-bit: 2, 3, 5, 7, 11, 13
        if (Inputs = "0010" or Inputs = "0011" or
            Inputs = "0101" or Inputs = "1000" or
            Inputs = "1011" or Inputs = "1101") then
            assert (F = '1') report "Faulty Logic! Expected prime number outpu
        else
            assert (F = '0') report "Faulty Logic! Expected non-prime number
        end if;

        Inputs <= Inputs + '1';
    end loop;
```

Figure 5 Error in code test bench



```
Note: ---- STARTING PRIME NUMBER SIMULATION ----
Time: 0 ps  Iteration: 0  Process: /test_Prac1Prime/s
Error: Faulty Logic! Expected non-prime number output
Time: 80 ns  Iteration: 0  Process: /test_Prac1Prime/
Error: Faulty Logic! Expected prime number output.
Time: 90 ns  Iteration: 0  Process: /test_Prac1Prime/
Note: ---- SIMULATION COMPLETE ----
```

Figure 6 Error message in Tcl Console

Synthesis/Implementation Results

The following figures and tables provide an overview of the hardware synthesis process and resource utilisation.

Figure 7 shows the register-transfer level representation of the design, generated directly from the VHDL code. It includes the input ports A, B, C, D, internal logic blocks, and the output port F.

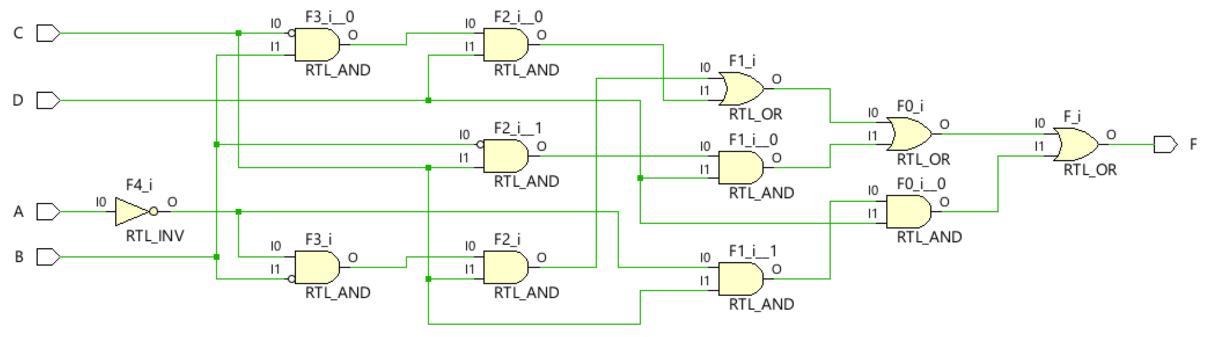


Figure 7 RTL schematic

Figure 8 is the synthesis schematic is a lower-level representation of the logic after optimisation by the synthesis tool. It shows how logic gates and components are interconnected to implement the prime number detection function.

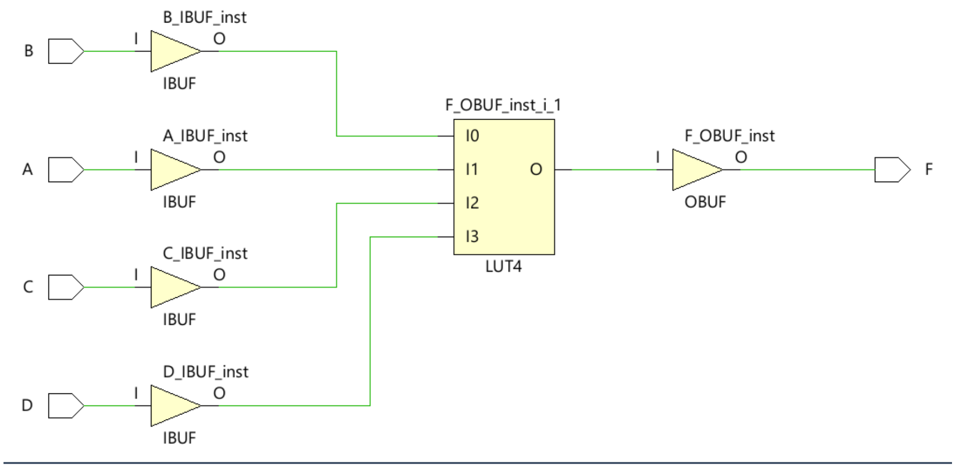
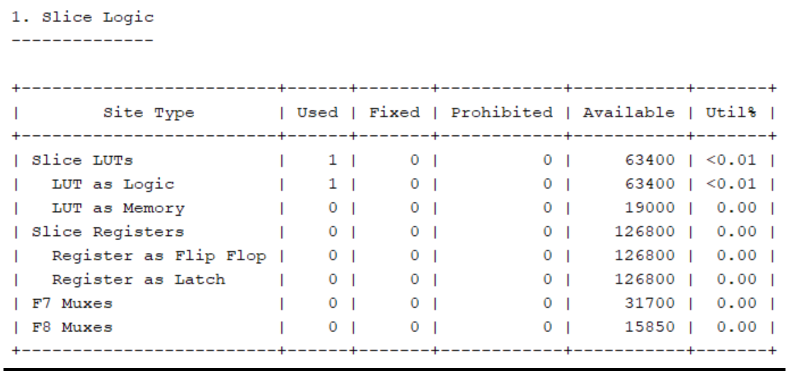


Figure 8 Synthesis Schematic

As expected, the logic usage is extremely low, with only one LUT used for the entire logic function and five IOBs allocated for the four inputs and one output, as shown in figure 9.

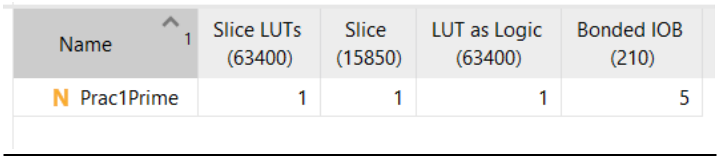


```
1. Slice Logic
--------------

+-------------------------+------+-------+------------+-----------+-------+
|        Site Type        | Used | Fixed | Prohibited | Available | Util% |
+-------------------------+------+-------+------------+-----------+-------+
| Slice LUTs              |    1 |     0 |          0 |     63400 | <0.01 |
|   LUT as Logic          |    1 |     0 |          0 |     63400 | <0.01 |
|   LUT as Memory         |    0 |     0 |          0 |     19000 |  0.00 |
| Slice Registers         |    0 |     0 |          0 |    126800 |  0.00 |
|   Register as Flip Flop |    0 |     0 |          0 |    126800 |  0.00 |
|   Register as Latch     |    0 |     0 |          0 |    126800 |  0.00 |
| F7 Muxes                |    0 |     0 |          0 |     31700 |  0.00 |
| F8 Muxes                |    0 |     0 |          0 |     15850 |  0.00 |
+-------------------------+------+-------+------------+-----------+-------+
```

Figure 9 Resource Summary Overview Table

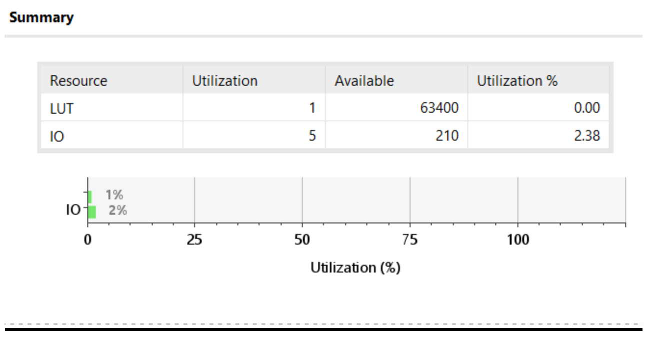
Figure 10 shows that only 1 look up table is being used and no flip-flop, latches or muxes.



Name 1	Slice LUTs (63400)	Slice (15850)	LUT as Logic (63400)	Bonded IOB (210)
N Prac1Prime	1	1	1	5

Figure 10 Detailed Slice Logic Table

Figure 11, utilisation summary chart shows clear breakdown of the percentages, LUT using close to 0% percent and 2.38% of I/O being used.


Summary

Resource	Utilization	Available	Utilization %
LUT	1	63400	0.00
IO	5	210	2.38

Figure 11 Utilisation Summary Chart

Discussion and Conclusion

The 4-bit prime number detector was successfully implemented and functionally verified using VHDL dataflow modelling. The design was tested with all 16 possible input combinations from 0000 (0) to 1111 (15). The output F correctly asserted high ('1') only for the prime numbers 2, 3, 5, 7, 11, and 13, and remained low for all other values.

The behavioural simulation confirmed correct functionality, with waveforms showing the expected output for each test case. The addition of a 10 ns output delay successfully modelled propagation delay in hardware, and the delay effect was clearly visible in the waveform results. The self-checking testbench proved effective, as intentionally introducing a logic error caused assertion failures that were captured in both the waveform and the Tcl console output.

Synthesis and implementation results demonstrated that the design is highly efficient, using only 1 LUT and 5 I/O blocks. No registers, latches, or multiplexers were required, which is consistent with the purely combinational nature of the design.

Potential Improvements:

- The design could be expanded to handle larger input sizes by increasing the number of bits and updating the detection logic or using a lookup table approach.
- A sequential version could be implemented to dynamically check for primality rather than hardcoding known prime values, which would improve scalability.

In conclusion, the project met all functional and implementation requirements. The design was simple, resource-efficient, and robustly tested, making it an effective introduction to VHDL design, simulation, and synthesis using Vivado.

References

List any references used, including gen AI tools.