

**THE CONSTRUCTION OF A
FOUR-BIT PRIME NUMBER DETECTOR**

**UNIVERSITY OF THE PHILIPPINES VISAYAS
COLLEGE OF ARTS AND SCIENCES
DIVISION OF PHYSICAL SCIENCES AND MATHEMATICS**

**CMSC 130 - LOGIC DESIGN AND DIGITAL COMPUTER CIRCUITS
2ND SEMESTER AY 2021-2022**

ASSIGNMENT 1. INTRODUCTION TO VHDL

PREPARED BY:

DELA CRUZ	,	LLOYD WALLYS
LESCANO	,	RENMAR
ORAÑO	,	MYRTLLE GEM

MARCH 9, 2022

Abstract

In this laboratory exercise, a prime number detector program for 4-bit binary values was demonstrated. The detector was developed in VHDL. The team constructed a solution of the prime number detection architecture, which employs 4-bit vector inputs with each bit described by $sw(0)$, $sw(1)$, $sw(2)$, and $sw(3)$, as well as the system output named Prime. Using Boolean algebra, the solution $sw(3)'sw(2)'sw(1) + sw(2)'sw(1)sw(0) + sw(3)'sw(2)sw(0) + sw(2)sw(1)'sw(0)$ was formulated. The correctness of this solution was verified using truth tables. Ultimately, the expected outputs were generated proving its validity and applicability for this assignment. Accordingly, this solution was translated into logic, and finally, into codes. The program was built and compiled using GHDL, and the testbench simulation was visualized using GTKWave. The developed program processed 4-bit inputs and correctly generated high output signals for all values denoting prime numbers.

Table of Contents

Parts	Pages
Title Page	1
Abstract	2
List of Figures	4
Design	5
Problem Statement	5
Overview	5
Boolean Equation	6
Truth Tables	7
Logic Circuit	10
Testing and Debugging	11
Conclusion	13
References	14

List of Figures and Tables

Parts	Pages
Figure 1	5
Figure 2	6
Table 1	7
Table 2	7
Table 3	8
Table 4	8
Table 5	9
Table 6	9
Table 7	10
Figure 3	10
Figure 4	11
Figure 5	11
Figure 6	12
Figure 7	12
Figure 8	12
Figure 9	13

Design

Problem Statement:

In this assignment, the team is required to develop a VHDL program, *PrimeDetector*, that takes a 4-bit input representing the integers 0_{10} to 15_{10} . The resulting output *Prime* should return 1 (high output signal) if the given input is a prime number, and 0 (low output signal) if otherwise. By definition, a prime number is a number that is only divisible by itself and one. Thus, this excludes 0 and 1.

Overview:

For a systematic procedure, the following objectives were pursued:

1. develop a Boolean equation for Prime
2. prove its correctness using truth tables
3. simulate a comprehensive logic circuit
4. translate the solution in code using VHDL
5. visualize and verify the output for Prime

Boolean Equation:

To initialize the design process, the system must be mathematically formulated. Given the definition for prime numbers, all candidate inputs from 0_{10} to 15_{10} were isolated and represented through their bit values and position. Specifically, for bit N in position K , if its value is 1, it will be represented as NK . Meanwhile, if its value is 0, then it will be represented using NK' .

Integer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Prime	No	No	Yes	Yes	No	Yes	No	Yes	No	No	No	Yes	No	Yes	No	No

Integer	Bit Vector Representation				Output	Input Representation
	N3	N2	N1	N0		
0	0	0	0	0	0	
1	0	0	0	1	0	
2	0	0	1	0	1	$N3'N2'N1N0'$
3	0	0	1	1	1	$N3'N2'N1N0$
4	0	1	0	0	0	
5	0	1	0	1	1	$N3'N2N1'N0$
6	0	1	1	0	0	
7	0	1	1	1	1	$N3'N2N1N0$
8	1	0	0	0	0	
9	1	0	0	1	0	
10	1	0	1	0	0	
11	1	0	1	1	1	$N3N2'N1N0$
12	1	1	0	0	0	
13	1	1	0	1	1	$N3N2N1'N0$
14	1	1	1	0	0	
15	1	1	1	1	0	

Figure 1. The translation of decimal prime integers from 0 to 15 into binary logic.

Using the generated input representations for the identified prime numbers, the system architecture for PrimeDetector was then designed using Boolean algebra.

Solution. Let the pd be the system *PrimeDetector*.

	Steps	Laws
pd	$= N3'N2'N1N0' + N3'N2'N1N0 + N3'N2N1'N0$ $+ N3'N2N1N0 + N3N2'N1N0 + N3N2N1'N0$	
	$= (N3'N2'N1N0' + N3'N2'N1N0)$ $+ (N3'N2N1'N0 + N3'N2N1N0)$ $+ N3N2'N1N0 + N3N2N1'N0$	
pd	$= N3'N2'N1(N0' + N0) + N3'N2N0(N1 + N1')$ $+ N3N2'N1N0 + N3N2N1'N0$	OR Distributive
	$= N3'N2'N1 + N3'N2N0 + N3N2'N1N0$ $+ N3N2N1'N0$	Complement: $A + A' = 1$
	$= (N3'N2'N1 + N3N2'N1N0)$ $+ (N3'N2N0 + N3N2N1'N0)$	
pd	$= N2'N1(N3' + N3N0) + N2N0(N3' + N3N1')$	
	$= N2'N1(N3' + N0) + N2N0(N3' + N1')$	AND Distributive:
		$A + A'B$
		$= (A + A')(A + B)$
		$= 1(A + B) = A + B$
pd	$= N2'N1N3' + N2'N1N0 + N2N0N3' + N2N0N1'$	
pd	$= N3'N2'N1 + N2'N1N0 + N3'N2N0 + N2N1'N0$	

Figure 2. The Boolean equation for PrimeDetector system architecture.

Truth Tables:

The following truth tables (see Tables 2-6) were constructed to verify and prove the correctness of the Boolean equation in Figure 2. To facilitate a cohesive discussion, the notations in Table 1 will be used to represent the bits in the bit vector input *sw*.

Table 1. The tabular notations for input signals *sw*(3 down to 0) .

Input	Rep.
SW(3)	N3
SW(2)	N2
SW(1)	N1
SW(0)	N0

Moreover, for better comprehension, the truth tables were divided into 4 sections (see Tables 3-6) to illustrate the correctness of elements $\neg N3 \wedge \neg N2 \wedge N1$, $\neg N2 \wedge N1 \wedge N0$, $\neg N3 \wedge N2 \wedge N0$, and $N2 \wedge \neg N1 \wedge N0$ respectively. Finally, each resulting columns are combined and solved in Table 7.

Table 2. The truth table for all integers from 0 to 15.

Decimal Int	N3	N2	N1	N0
15	T	T	T	T
14	T	T	T	F
13	T	T	F	T
12	T	T	F	F
11	T	F	T	T
10	T	F	T	F
9	T	F	F	T
8	T	F	F	F
7	F	T	T	T
6	F	T	T	F
5	F	T	F	T
4	F	T	F	F
3	F	F	T	T
2	F	F	T	F
1	F	F	F	T
0	F	F	F	F

The table above shows the 16 possible combinations of the four bits (N3 down to N0) representing the integers 0_{10} until 15_{10} .

Table 3. The truth table for $\neg N3 \wedge \neg N2 \wedge N1$.

Decimal Int	N3	N2	N1	$\neg N3$	$\neg N2$	$\neg N3 \wedge \neg N2$	$\neg N3 \wedge \neg N2 \wedge N1$
15	T	T	T	F	F	F	F
14	T	T	T	F	F	F	F
13	T	T	F	F	F	F	F
12	T	T	F	F	F	F	F
11	T	F	T	F	T	F	F
10	T	F	T	F	T	F	F
9	T	F	F	F	T	F	F
8	T	F	F	F	T	F	F
7	F	T	T	T	F	F	F
6	F	T	T	T	F	F	F
5	F	T	F	T	F	F	F
4	F	T	F	T	F	F	F
3	F	F	T	T	T	T	T
2	F	F	T	T	T	T	T
1	F	F	F	T	T	T	F
0	F	F	F	T	T	T	F

Table 3 illustrates the result for the first equation element, $N3'N2'N1$. The final column indicates that only the inputs for decimal integers 2 and 3 should result to a high output signal.

Table 4 . The truth table for $\neg N2 \wedge N1 \wedge N0$.

Decimal Int	N2	N1	N0	$\neg N2$	$N1 \wedge N0$	$\neg N2 \wedge N1 \wedge N0$
15	T	T	T	F	T	F
14	T	T	F	F	F	F
13	T	F	T	F	F	F
12	T	F	F	F	F	F
11	F	T	T	T	T	T
10	F	T	F	T	F	F
9	F	F	T	T	F	F
8	F	F	F	T	F	F
7	T	T	T	F	T	F
6	T	T	F	F	F	F
5	T	F	T	F	F	F
4	T	F	F	F	F	F
3	F	T	T	T	T	T
2	F	T	F	T	F	F
1	F	F	T	T	F	F
0	F	F	F	T	F	F

Table 4 illustrates the result for the second equation element, $N2'N1N0$. The final column indicates that only the inputs for decimal integers 3 and 11 should result to a high output signal.

Table 5. The truth table for $\neg N3 \wedge N2 \wedge N0$.

Decimal Int	N3	N2	N0	$\neg N3$	$\neg N3 \wedge N2$	$\neg N3 \wedge N2 \wedge N0$
15	T	T	T	F	F	F
14	T	T	F	F	F	F
13	T	T	T	F	F	F
12	T	T	F	F	F	F
11	T	F	T	F	F	F
10	T	F	F	F	F	F
9	T	F	T	F	F	F
8	T	F	F	F	F	F
7	F	T	T	T	T	T
6	F	T	F	T	T	F
5	F	T	T	T	T	T
4	F	T	F	T	T	F
3	F	F	T	T	F	F
2	F	F	F	T	F	F
1	F	F	T	T	F	F
0	F	F	F	T	F	F

Table 5 illustrates the result for the third equation element, $N3'N2N0$. The final column indicates that only the inputs for decimal integers 5 and 7 should result to a high output signal.

Table 6 . The truth table for $N2 \wedge \neg N1 \wedge N0$.

Decimal Int	N2	N1	N0	$\neg N1$	$\neg N1 \wedge N0$	$N2 \wedge \neg N1 \wedge N0$
15	T	T	T	F	F	F
14	T	T	F	F	F	F
13	T	F	T	T	T	T
12	T	F	F	T	F	F
11	F	T	T	F	F	F
10	F	T	F	F	F	F
9	F	F	T	T	T	F
8	F	F	F	T	F	F
7	T	T	T	F	F	F
6	T	T	F	F	F	F
5	T	F	T	T	T	T
4	T	F	F	T	F	F
3	F	T	T	F	F	F
2	F	T	F	F	F	F
1	F	F	T	T	T	F
0	F	F	F	T	F	F

Table 6 illustrates the result for the fourth equation element, $N2N1'N0$. The final column indicates that only the inputs for decimal integers 5 and 13 should result to a high output signal.

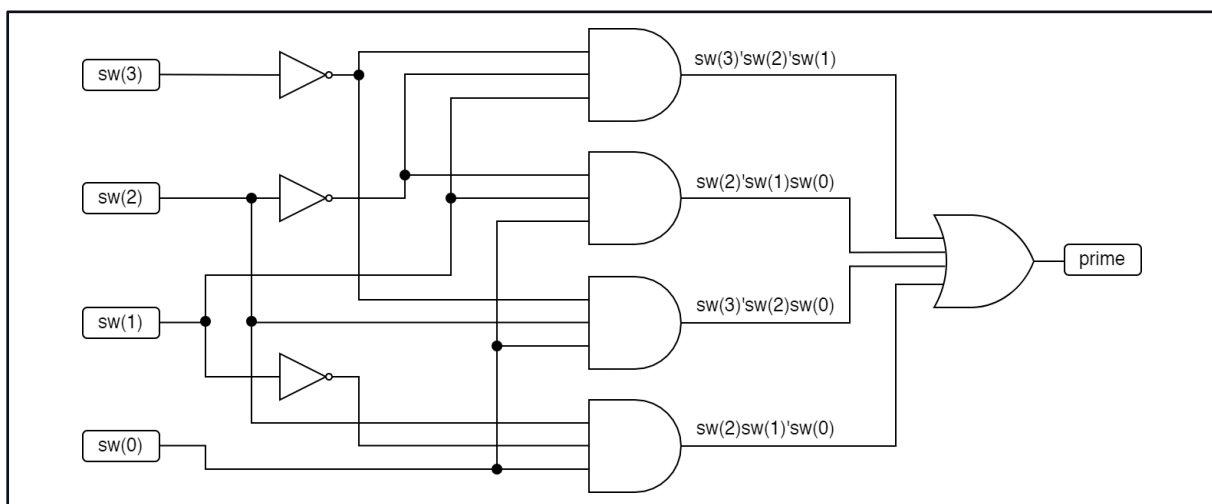
Table 7. The summarized truth table for the PrimeDetector system architecture.

Decimal Int	$\neg N3 \wedge N2 \wedge N0$	$\neg N3 \wedge \neg N2 \wedge N1$	$\neg N2 \wedge N1 \wedge N0$	$N2 \wedge \neg N1 \wedge N0$	$(\neg N3 \wedge N2 \wedge N0)$ $\vee (\neg N3 \wedge \neg N2 \wedge N1)$ $\vee (\neg N2 \wedge N1 \wedge N0)$ $\vee (N2 \wedge \neg N1 \wedge N0)$
15	F	F	F	F	F
14	F	F	F	F	F
13	F	F	F	T	T
12	F	F	F	F	F
11	F	F	T	F	T
10	F	F	F	F	F
9	F	F	F	F	F
8	F	F	F	F	F
7	T	F	F	F	T
6	F	F	F	F	F
5	T	F	F	T	T
4	F	F	F	F	F
3	F	T	T	F	T
2	F	T	F	F	T
1	F	F	F	F	F
0	F	F	F	F	F

Table 7 combines the respective results of all the elements from the equation in Figure 2. The final column indicates that the inputs for decimal integers 2, 3, 5, 7, 11, and 13, should yield a high output signal. Since this coincides with the specifications for the assignment, the Boolean equation is hereby proven to be correct.

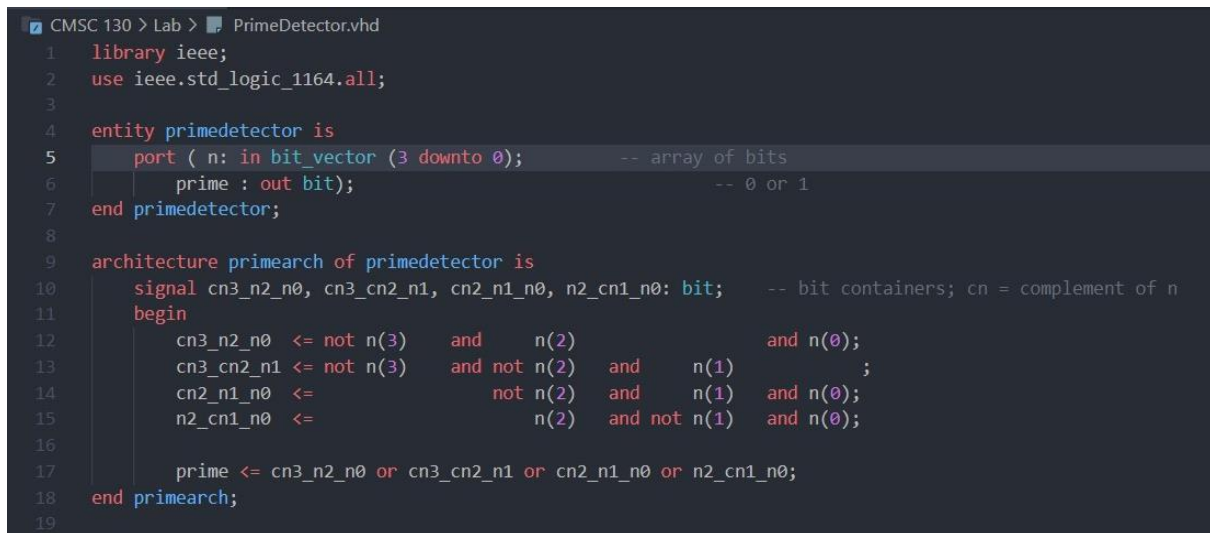
Logic Circuit:

In addition to the truth tables, external virtual simulations were performed using the corresponding logic gates to further verify the solution. The figure below (*see Figure 3*), constructed using *Diagram.io*, summarizes the design and connections for the logic circuit.

**Figure 3.** The logic circuit design for the solution.

Testing and Debugging

Finally, using an Integrated Development Environment (IDE), the solution was translated in to VHDL programming language (see *Figure 4*). Given that beforehand, a Boolean Algebra solution was developed as the basis for the program, no significant semantic issues or bugs have been raised. However, a runtime error was observed. Specifically, there was a conflict with the names of the ports between the program and test bench (see *Figure 5*). Evidently, it was quickly addressed by simply checking the specifications of PrimeDetector_TB and modifying the port names accordingly (see *Figure 6-7*).

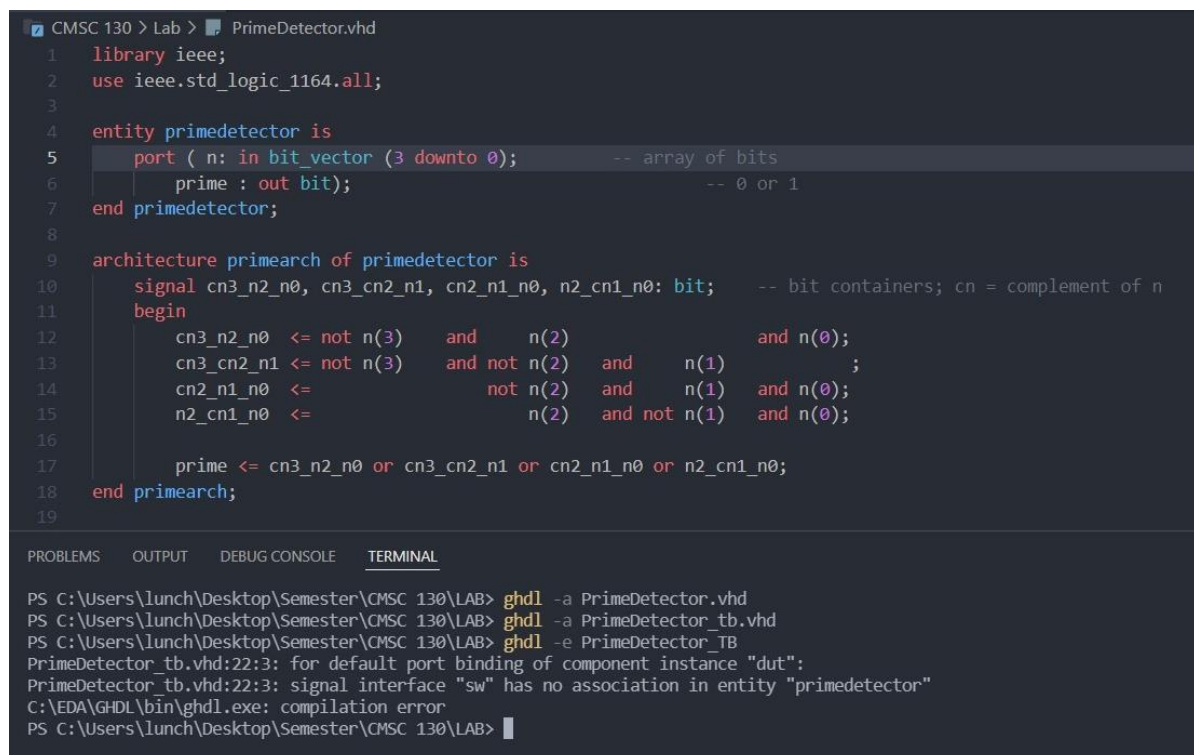


```

CMSC 130 > Lab > PrimeDetector.vhd
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity primedetector is
5      port ( n: in bit_vector (3 downto 0);      -- array of bits
6            prime : out bit);                    -- 0 or 1
7  end primedetector;
8
9  architecture primearch of primedetector is
10     signal cn3_n2_n0, cn3_cn2_n1, cn2_n1_n0, n2_cn1_n0: bit;    -- bit containers; cn = complement of n
11     begin
12         cn3_n2_n0 <= not n(3) and n(2) and n(0);
13         cn3_cn2_n1 <= not n(3) and not n(2) and n(1);
14         cn2_n1_n0 <= not n(2) and n(1) and n(0);
15         n2_cn1_n0 <= n(2) and not n(1) and n(0);
16
17         prime <= cn3_n2_n0 or cn3_cn2_n1 or cn2_n1_n0 or n2_cn1_n0;
18     end primearch;
19

```

Figure 4. The initial implementation of the solution into VHDL.



```

CMSC 130 > Lab > PrimeDetector.vhd
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity primedetector is
5      port ( n: in bit_vector (3 downto 0);      -- array of bits
6            prime : out bit);                    -- 0 or 1
7  end primedetector;
8
9  architecture primearch of primedetector is
10     signal cn3_n2_n0, cn3_cn2_n1, cn2_n1_n0, n2_cn1_n0: bit;    -- bit containers; cn = complement of n
11     begin
12         cn3_n2_n0 <= not n(3) and n(2) and n(0);
13         cn3_cn2_n1 <= not n(3) and not n(2) and n(1);
14         cn2_n1_n0 <= not n(2) and n(1) and n(0);
15         n2_cn1_n0 <= n(2) and not n(1) and n(0);
16
17         prime <= cn3_n2_n0 or cn3_cn2_n1 or cn2_n1_n0 or n2_cn1_n0;
18     end primearch;
19

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL**

```

PS C:\Users\lunch\Desktop\Semester\CMSC 130\LAB> ghdl -a PrimeDetector.vhd
PS C:\Users\lunch\Desktop\Semester\CMSC 130\LAB> ghdl -a PrimeDetector_tb.vhd
PS C:\Users\lunch\Desktop\Semester\CMSC 130\LAB> ghdl -e PrimeDetector_TB
PrimeDetector_tb.vhd:22:3: for default port binding of component instance "dut":
PrimeDetector_tb.vhd:22:3: signal interface "sw" has no association in entity "primedetector"
C:\EDA\GHDL\bin\ghdl.exe: compilation error
PS C:\Users\lunch\Desktop\Semester\CMSC 130\LAB>

```

Figure 5. The runtime error that was raised pertaining to the conflict in port names.

```

CMSC 130 > Lab > PrimeDetector_TB.vhd
1  entity PrimeDetector_TB is
2  end entity;
3
4  architecture PrimeDetector_TB_arch of PrimeDetector_TB is
5
6  -- Constant Declaration
7  constant t_wait : time := 10 ns;
8
9  -- Component Declaration
10 component PrimeDetector
11 port ( SW : in bit_vector (3 downto 0);
12       Prime : out bit);
13 end component;
14
15 -- Signal Declaration
16 signal SW_TB : bit_vector (3 downto 0); -- these signals are used to connect to the device under test (DUT)
17 signal Prime_TB : bit;

```

Figure 6. The source of conflict as traced in PrimeDetector_TB.

```

CMSC 130 > Lab > PrimeDetector.vhd
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity primedetector is
5  port ( sw: in bit_vector (3 downto 0); -- array of bits
6        prime : out bit); -- 0 or 1
7  end primedetector;
8
9  architecture primearch of primedetector is
10 signal cn3_n2_n0, cn3_cn2_n1, cn2_n1_n0, n2_cn1_n0: bit; -- bit containers; cn = complement of n
11 begin
12 cn3_n2_n0 <= not sw(3) and sw(2) and sw(0);
13 cn3_cn2_n1 <= not sw(3) and not sw(2) and sw(1) and sw(0);
14 cn2_n1_n0 <= not sw(2) and sw(1) and sw(0);
15 n2_cn1_n0 <= sw(2) and not sw(1) and sw(0);
16
17 prime <= cn3_n2_n0 or cn3_cn2_n1 or cn2_n1_n0 or n2_cn1_n0;
18 end primearch;
19

```

Figure 7. Renaming of the input ports to fix the runtime error.

Afterwards, the program was built and compiled successfully using GHDL (see Figure 8). Lastly, the signals were simulated and visualized through GTKWave as illustrated in Figure 9. The high output signal from Prime is interpreted as 1 or *true* while the low output signal represents 0 or *false*. The simulation time was set to 160 nanoseconds for one full cycle. To note, the default viewing format of GTKWave is in hexadecimal which was modified to decimal to mirror the expected results for this laboratory activity.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/powershell

PS C:\Users\lunch\Desktop\Semester\CMSC 130> cd lab
PS C:\Users\lunch\Desktop\Semester\CMSC 130\lab> ghdl -a PrimeDetector.vhd
PS C:\Users\lunch\Desktop\Semester\CMSC 130\lab> ghdl -a PrimeDetector_TB.vhd
PS C:\Users\lunch\Desktop\Semester\CMSC 130\lab> ghdl -e PrimeDetector_TB
PS C:\Users\lunch\Desktop\Semester\CMSC 130\lab> ghdl -r PrimeDetector_TB --vcd=PrimeDetector.vcd --stop-time=160ns
.\primedetector_TB.exe:info: simulation stopped by --stop-time @160ns
PS C:\Users\lunch\Desktop\Semester\CMSC 130\lab> gtkwave PrimeDetector.vcd

GTKWave Analyzer v3.3.100 (w)1999-2019 BSI

[0] start time.
[160000000] end time.

```

Figure 8. The GHDL and GTKWave commands to build and run PrimeDetector.

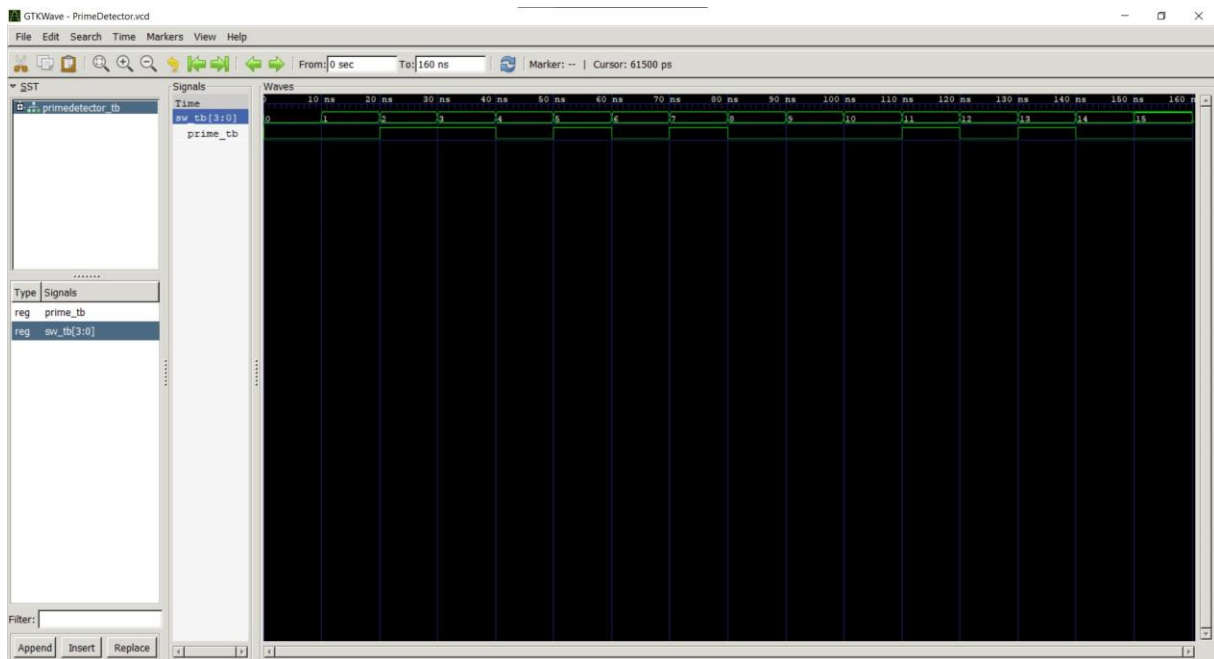


Figure 9. The wave output of PrimeDetector in GTKWave.

As shown in the figure, decimal inputs 2, 3, 5, 7, 11, and 13 resulted to high output signals indicating that they are detected as prime numbers. This corresponds accurately with the mathematical solution (see Figure 2) and the assignment specifications.

Conclusion

The purpose of this laboratory exercise was to design a prime number detector program which takes in 4-bit inputs that represent the integers between 0_{10} and 15_{10} and determine the prime numbers within that range of values. Utilizing the appropriate tools or procedures, the team was able to successfully design, develop, prove, and simulate all the specifications. Precisely, using the formulated Boolean equation, the program, *PrimeDetector*, correctly recognized and processed each bit vector input through the logic tests to generate a corresponding output signal. While a minor runtime error was initially encountered, it was immediately resolved by inspecting the test bench and making the necessary adjustments in the program code. Finally, as visualized in GTKWave, the program has effectively detected all the prime numbers between 0_{10} and 15_{10} therefore confirming the functionality and correctness of the solution. While there are no apparent technical deficiencies, it is recommended for future experiments to regularly crosscheck the testing environment to avoid unforeseen errors.

References

- ASPENCORE, "Laws of boolean algebra and Boolean Algebra Rules," *Basic Electronics Tutorials*, 01-May-2021. [Online]. Available: https://www.electronicstutorials.ws/boolean/bool_6.html. [Accessed: 06-Mar-2022].
- J. F. Wakerly, "Chapter 4 Combinational Logic Design Principles," in *Digital Design Principles & Practices*, 3rd ed., Upper Saddle River, New Jersey: Prentice Hall International, Inc., 2001, pp. 282–292.
- "Operators," *VHDL Reference Guide - Operators*, 06-Oct-2009. [Online]. Available: <https://www.ics.uci.edu/~jmoorkan/vhdlref/operator.html>. [Accessed: 06-Mar-2022].
- P. Mehta, "VHDL Syntax Reference," *VHDL syntax reference*, 2003. [Online]. Available: http://webdocs.cs.ualberta.ca/~amaral/courses/329/labs/VHDL_Reference.html#library. [Accessed: 06-Mar-2022].