Christopher Lall

CSc 342/343 – Lab 3: Adder

Professor Gertner

Due: March 6th, 2022

Table of Contents

[Objective: 3](#_Toc97458271)

[Screenshots: 3](#_Toc97458272)

[Half Adder: 3](#_Toc97458273)

[Full Adder: 5](#_Toc97458274)

[4-bit Adder: 7](#_Toc97458275)

[4-bit Adder/Subtractor: 8](#_Toc97458276)

[N-bit adder: 9](#_Toc97458277)

[N-bit adder as 4bit: 11](#_Toc97458278)

[N-bit adder as 32bit: 13](#_Toc97458279)

[LPM: 15](#_Toc97458280)

[Test Bench: 19](#_Toc97458281)

[Analysis & Explanations: 21](#_Toc97458282)

[4Bit Explanation for each operand case: 21](#_Toc97458283)

[32Bit Explanation for each operand case: 21](#_Toc97458284)

[Conclusion: 21](#_Toc97458285)

# Objective:

The goal of this lab is to design half adder, 1-bit full adder, 4-bit adder, 4-bit adder/subtractor, and n-bit adder/subtractor in VHDL. We are then going to verify our output from ModelSim with the respective truth tables for each adder.

# Screenshots:

Below will be the screenshots for all required tasks.

## Half Adder:

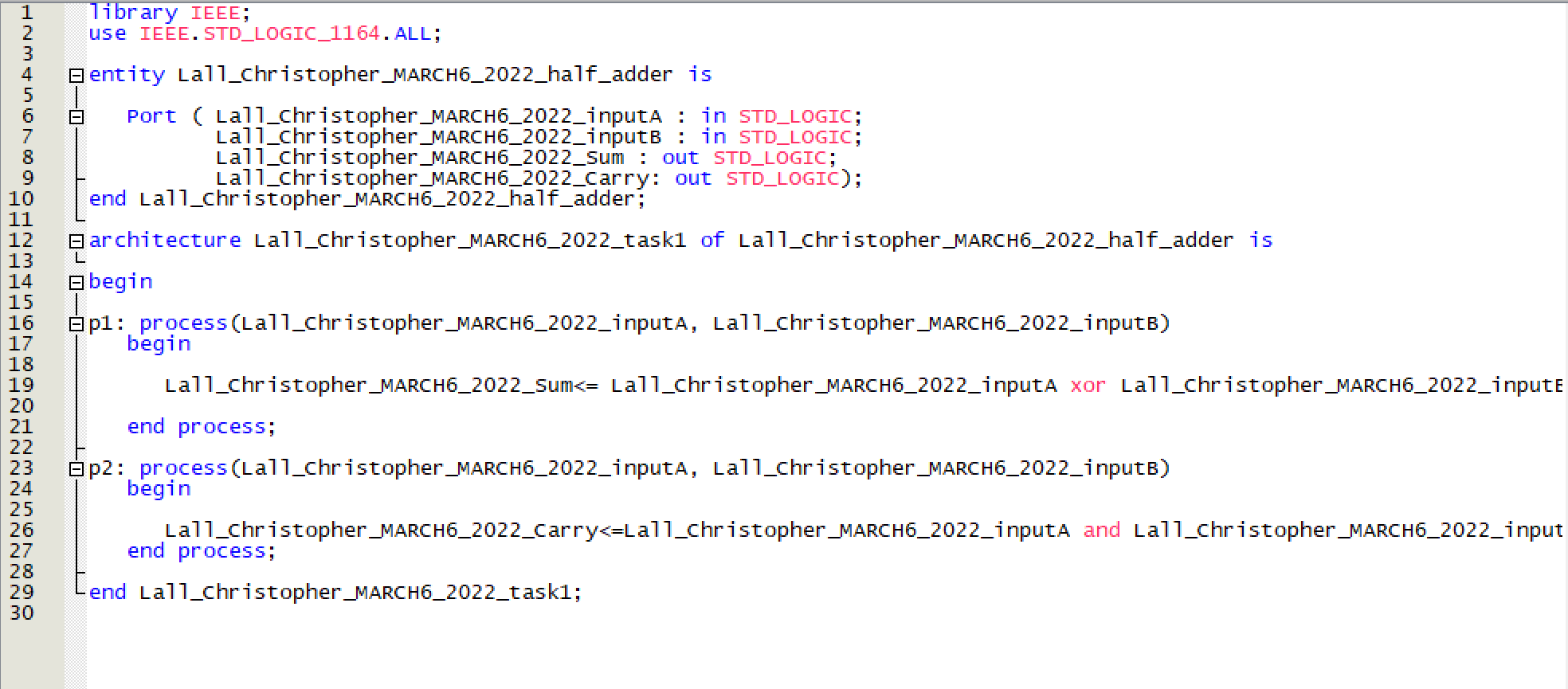


Figure 1. Half adder VHDL code

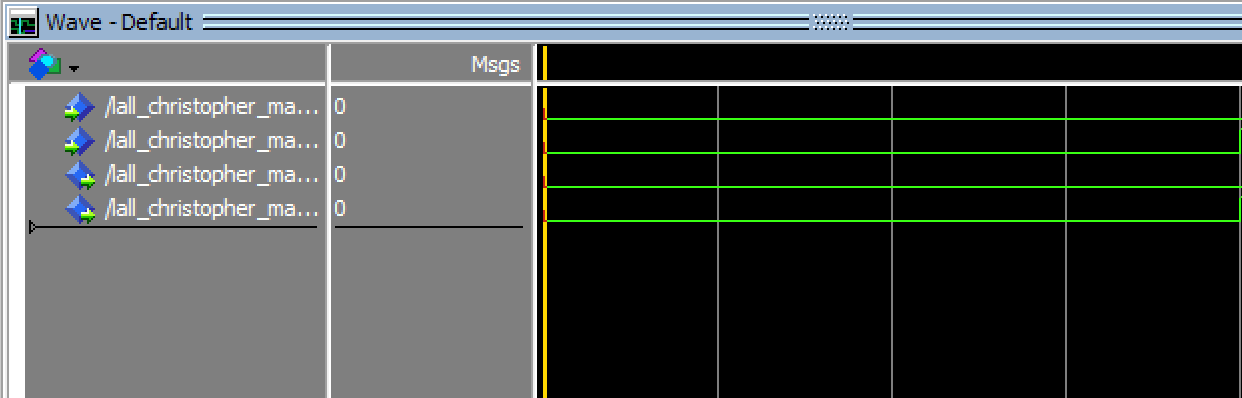


Figure 2. Half adder ouput from both inputs being 0

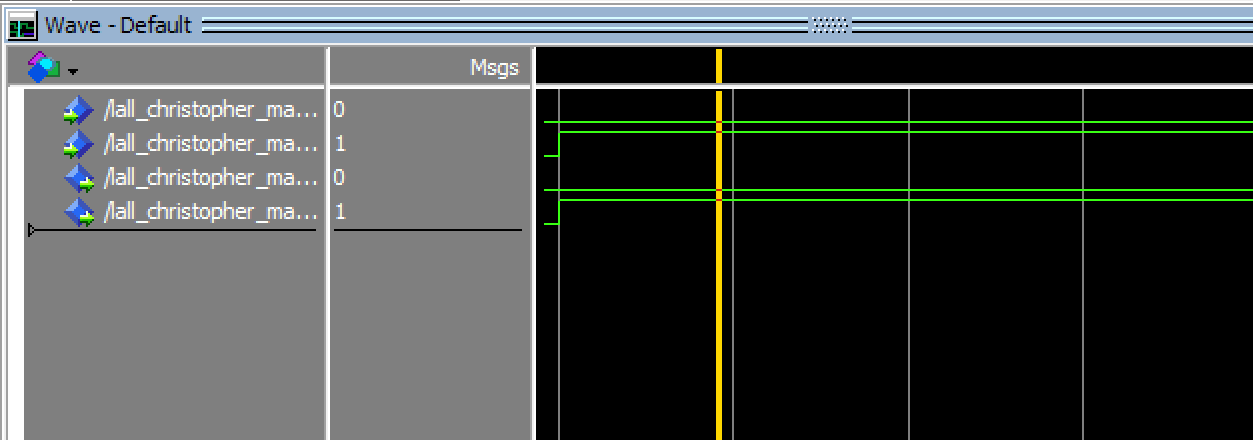


Figure 3. Half adder output where inputs are 0 and 1

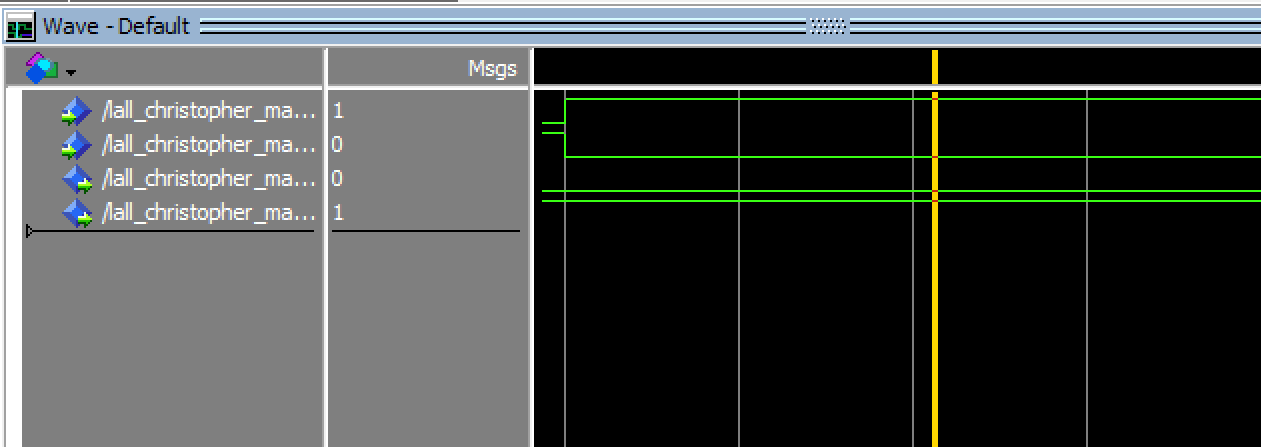


Figure 4. Half adder output where inputs are 1 and 0

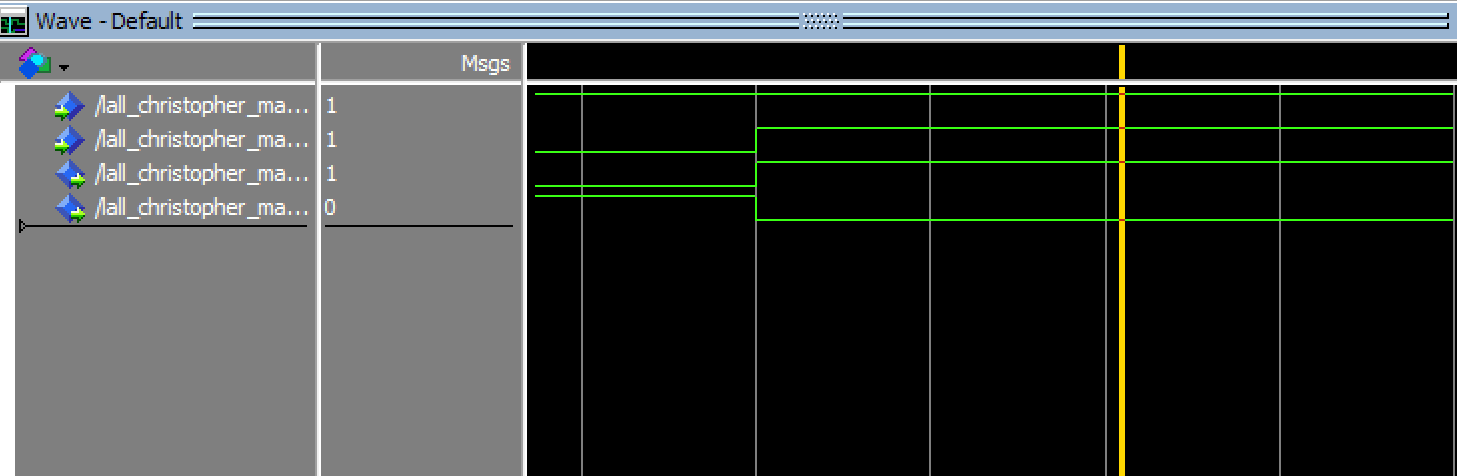


Figure 5. Half adder output where both inputs are 1

## Full Adder:

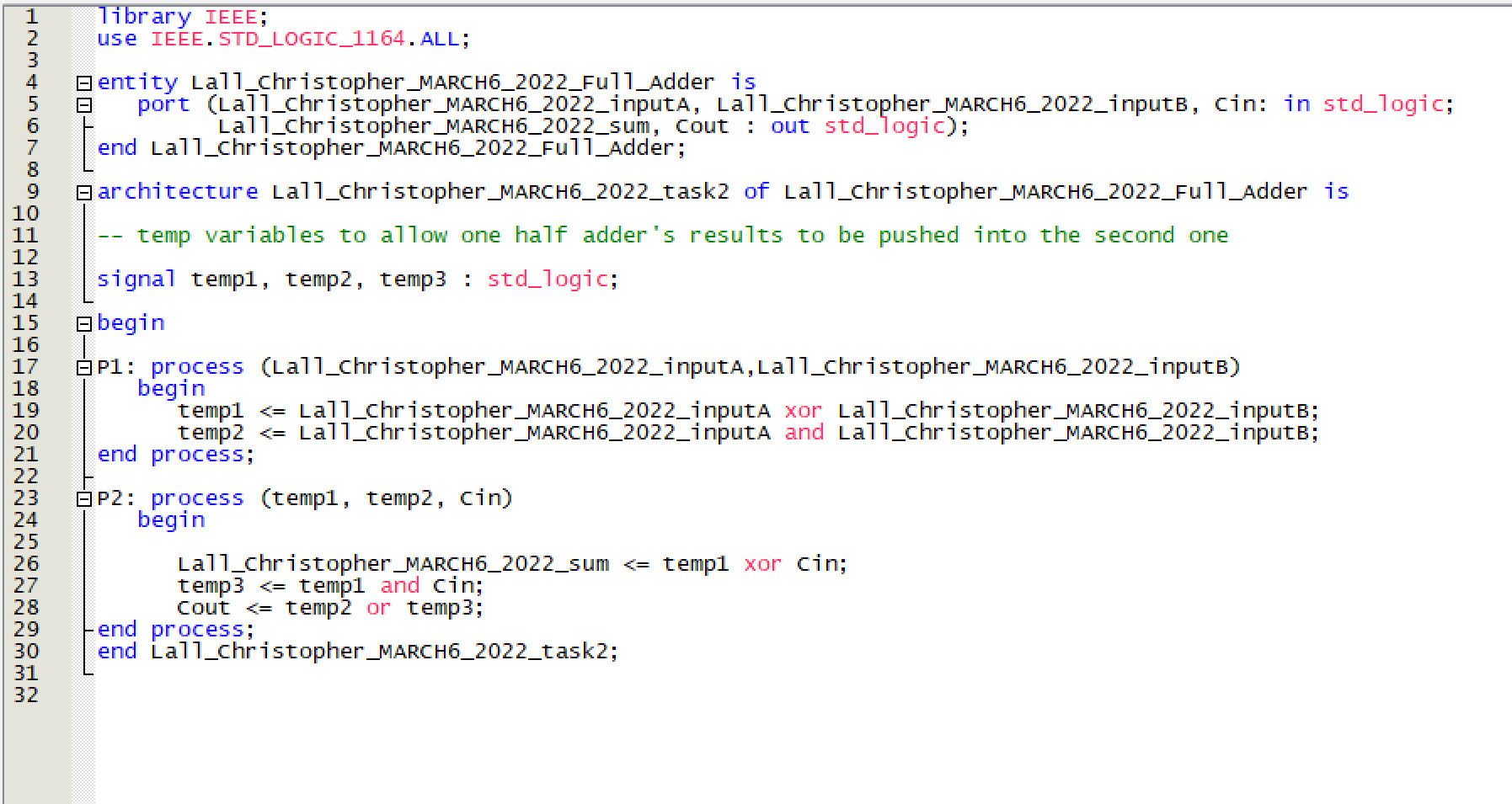


Figure 6. Full adder VHDL code

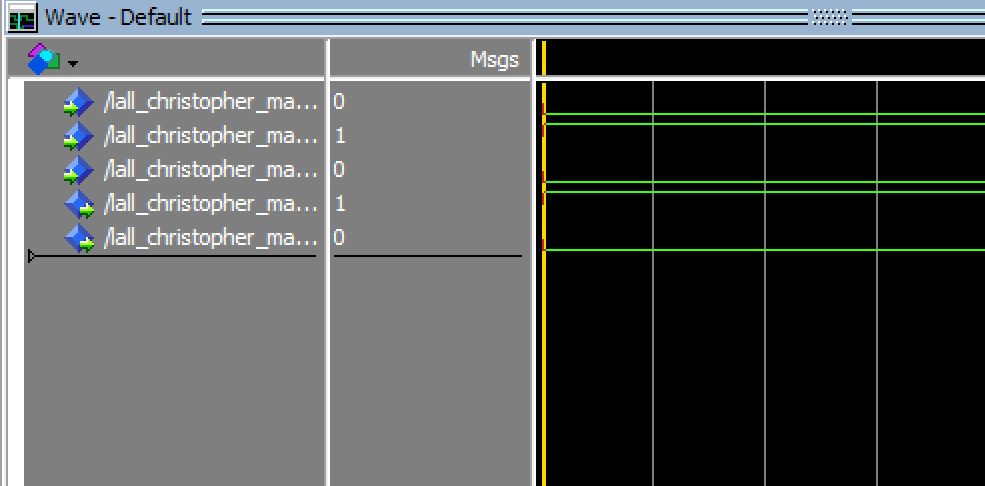


Figure 7. Full adder output where inputs are 0 1 0

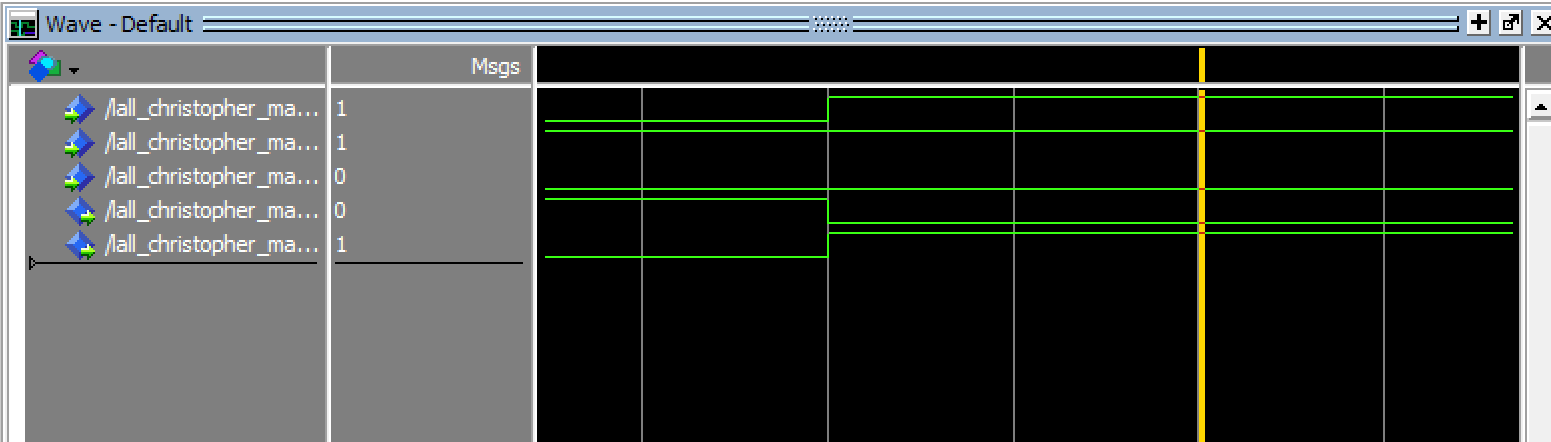


Figure 8. Full adder output where inputs are 1 1 0

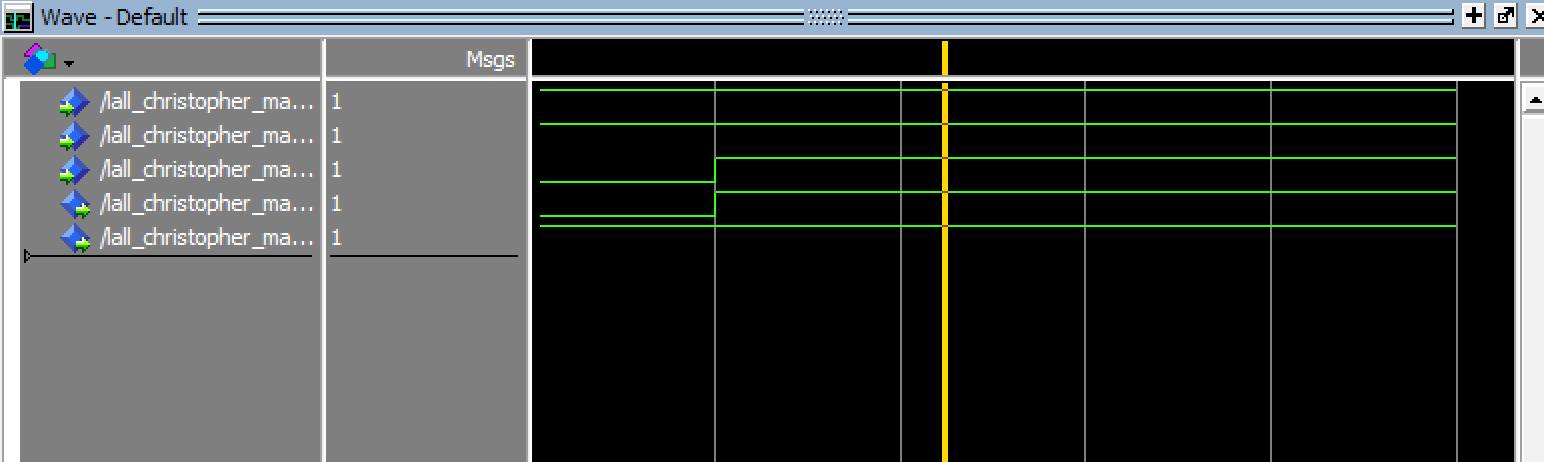


Figure 9. Full adder output where input is 1 1 1

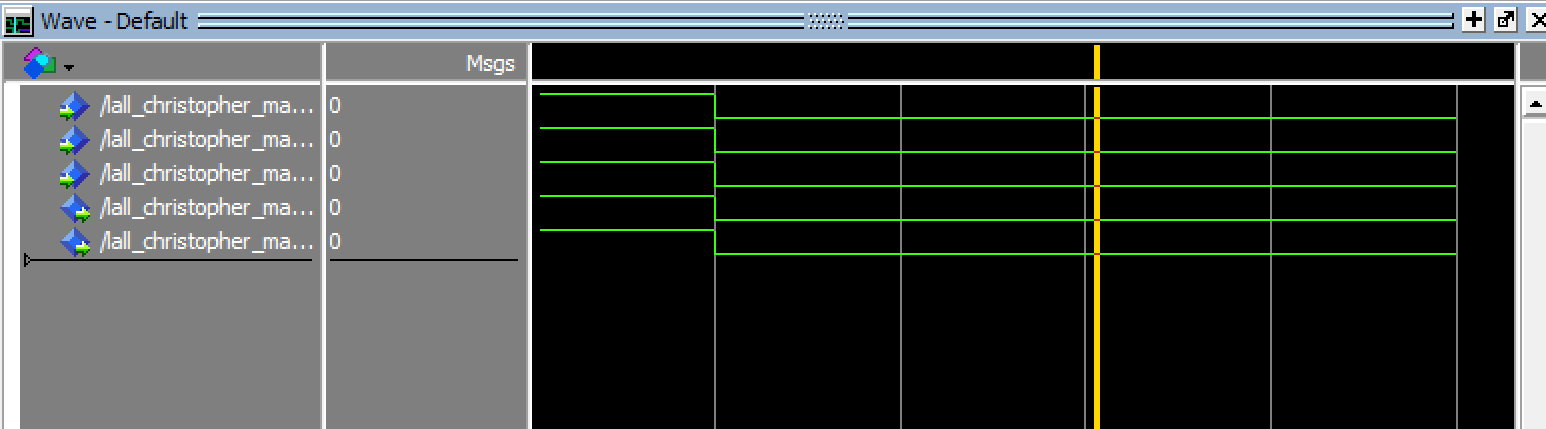


Figure 10. Full adder output where input is 0 0 0

## 4-bit Adder:

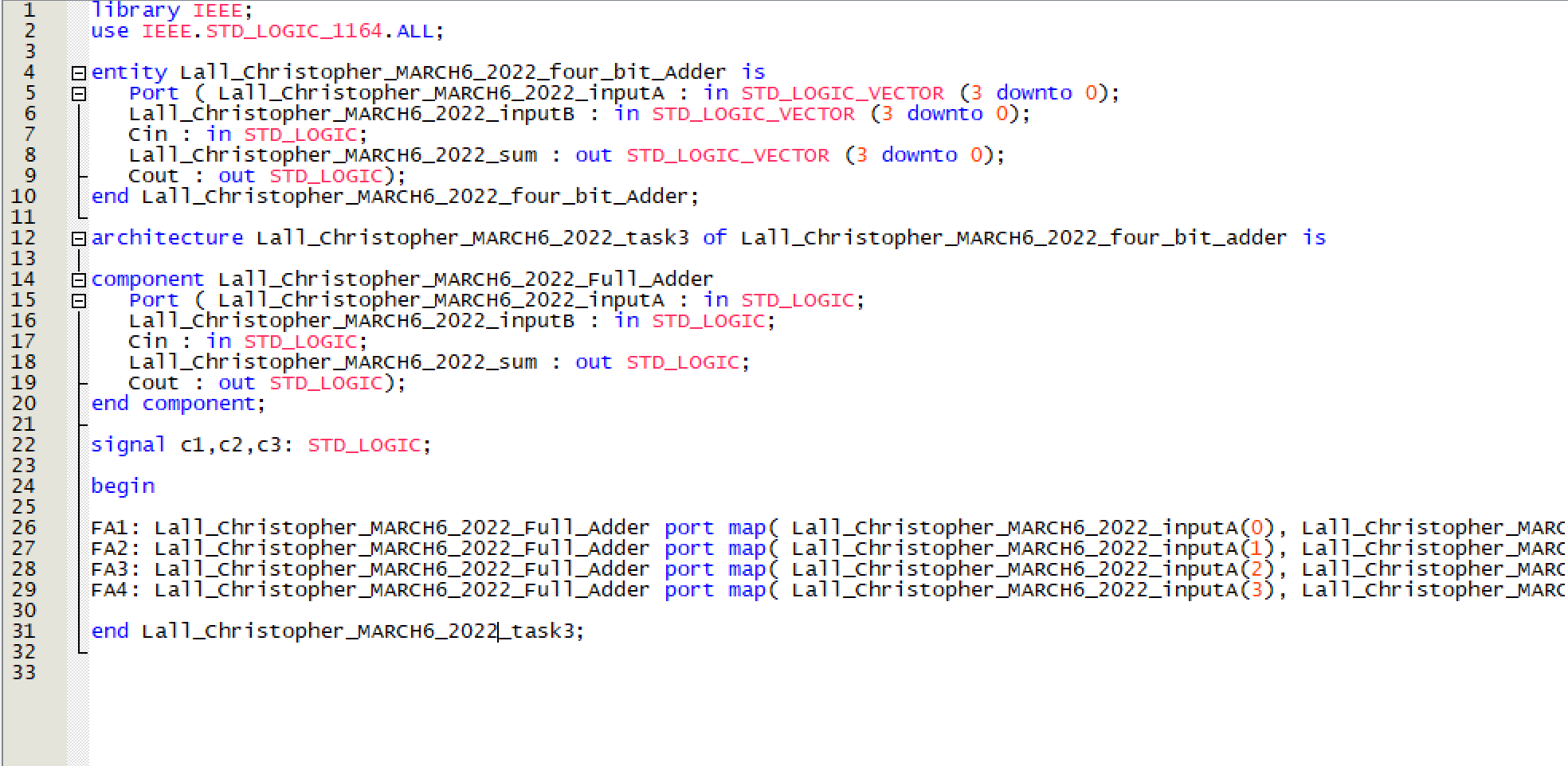


Figure 11. 4-bit Adder VHDL code

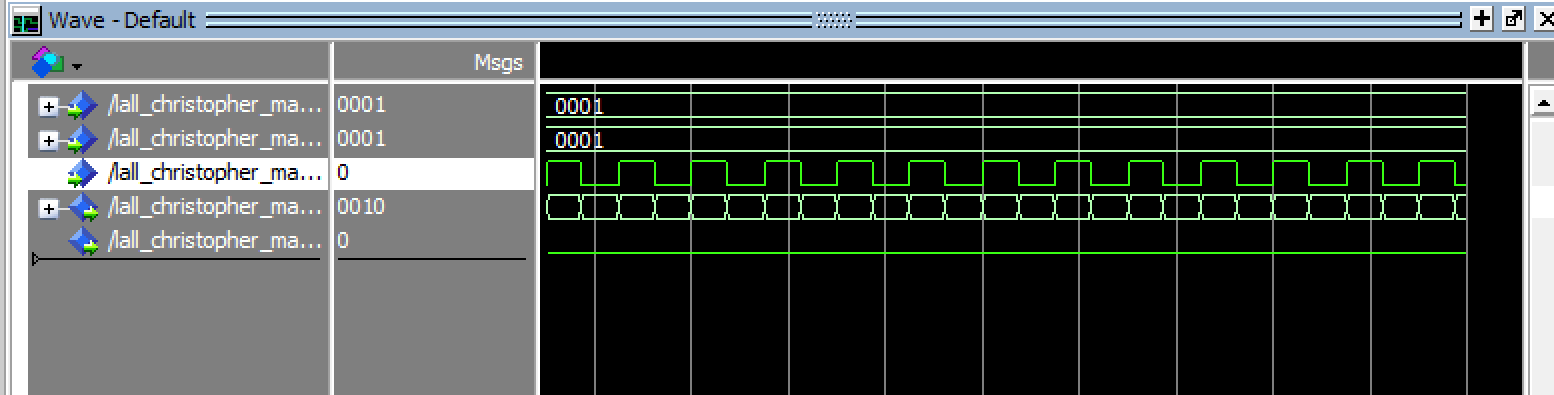


Figure 12. 4-bit Adder output where inputs are 0001 and 0001 with cin as 0

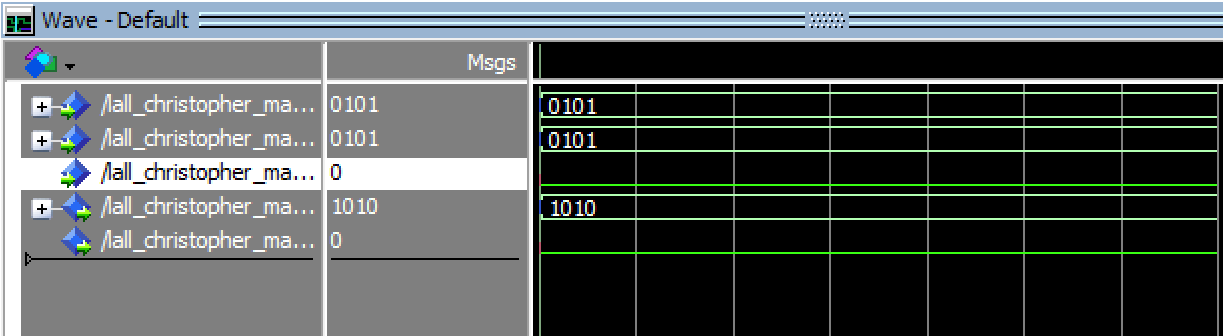


Figure 13. 4-bit Adder output where inputs are 0101 and 0101 with cin as 0

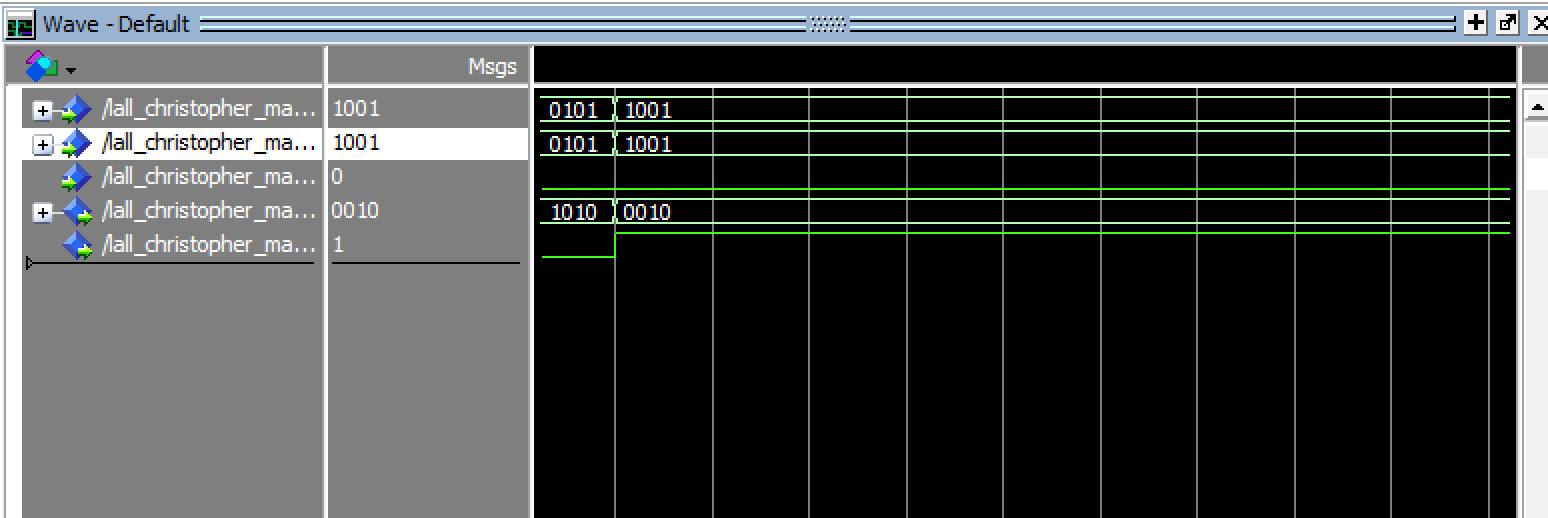


Figure 14. 4-bit Adder output where input is 1001 and 1001 with cin 0

## 4-bit Adder/Subtractor:

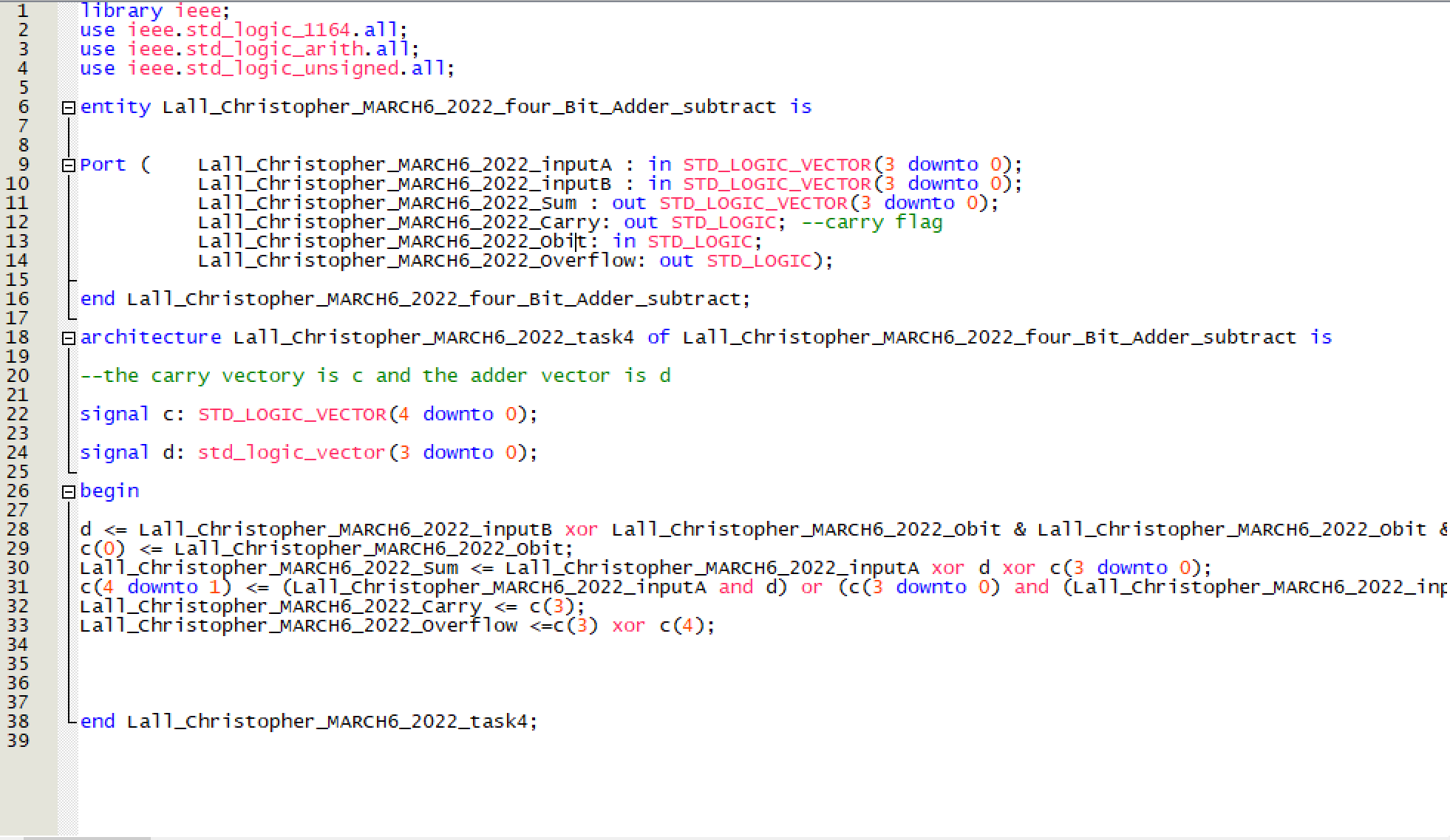


Figure 15. 4-bit Adder/Subtractor VHDL code

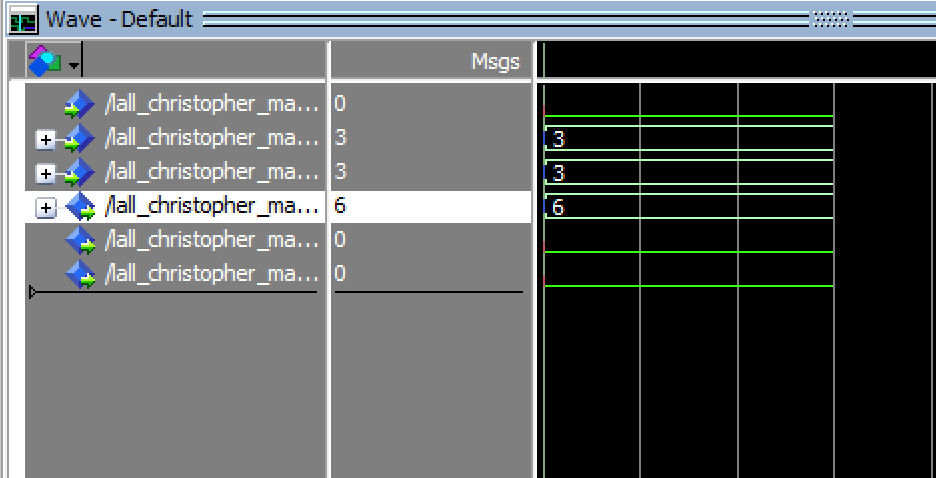


Figure 16. 4-bit Adder/Subtractor adder output where input is 0011(3) and 0011(3)

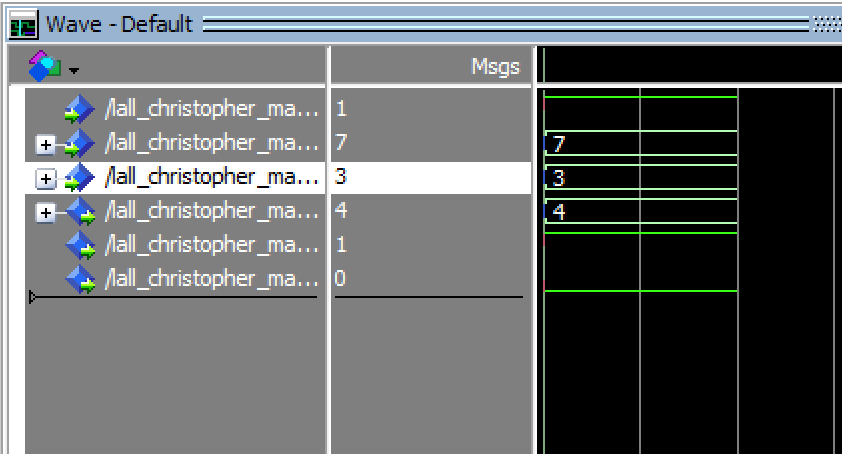


Figure 17. 4-bit Adder/Subtractor subtraction output where input is 0111 (7) and 0011(3)

## N-bit adder:

Text

Description automatically generated with medium confidence

Figure 18. N-bit adder VHDL code

Text

Description automatically generated

Figure 19. n-bit adder with flags VHDL code

Text

Description automatically generated

Figure 20. Waveform requirements

### N-bit adder as 4bit:

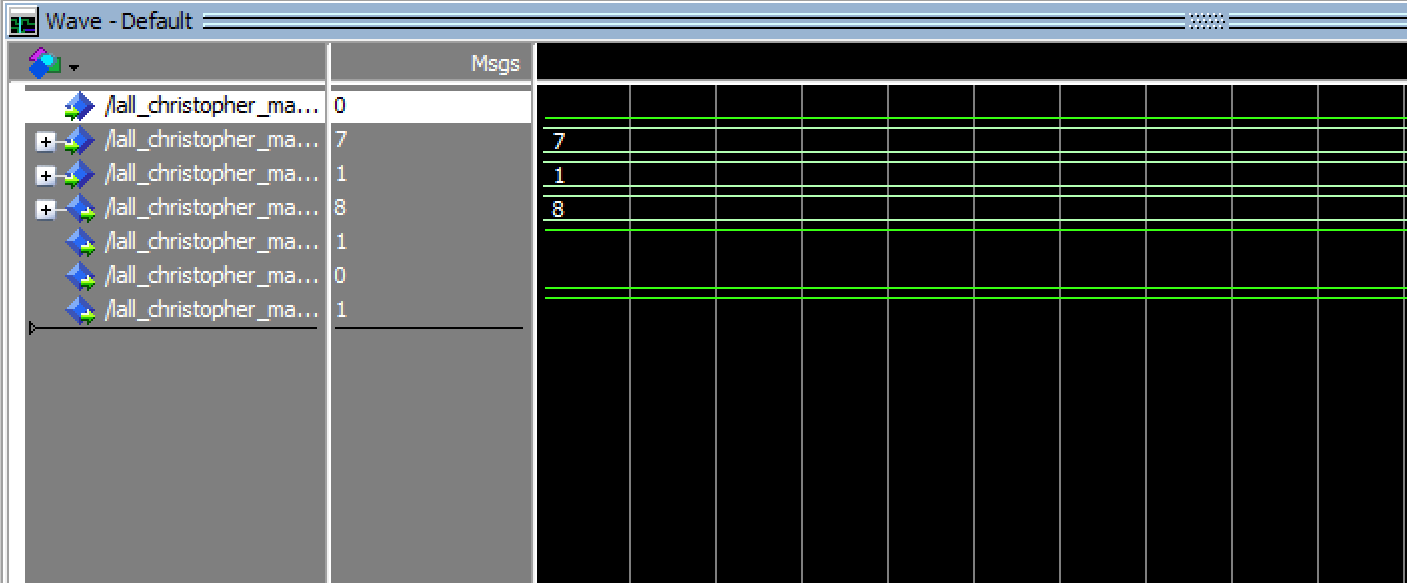


Figure 21. N-bit adder/sub as 4bit. Case 3a waveform

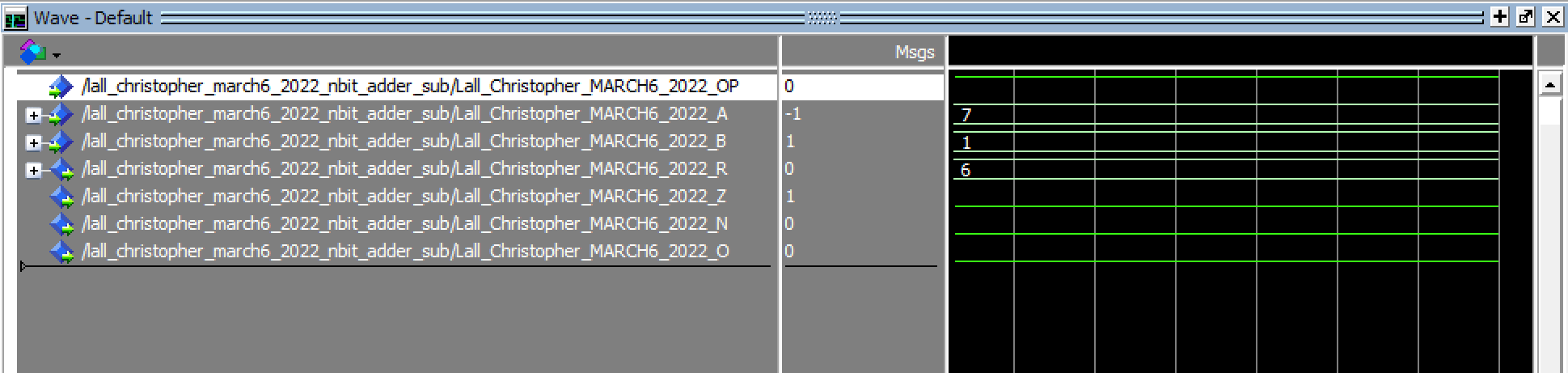


Figure 22. N-bit adder as 4bit. Case 3b waveform

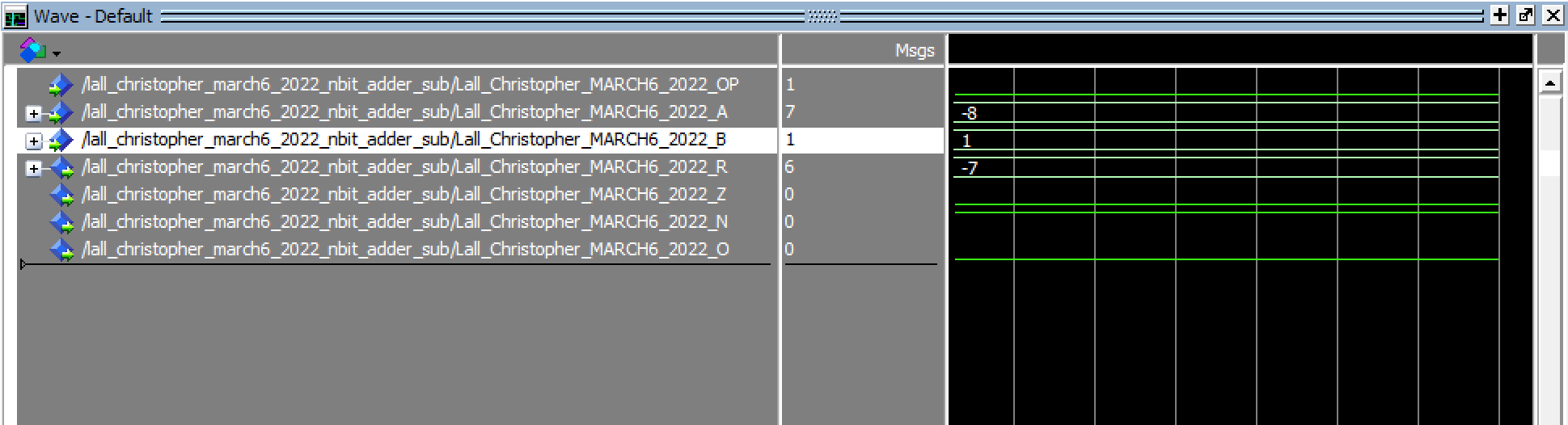


Figure 23. N-bit adder as 4bit. Case 3c waveform

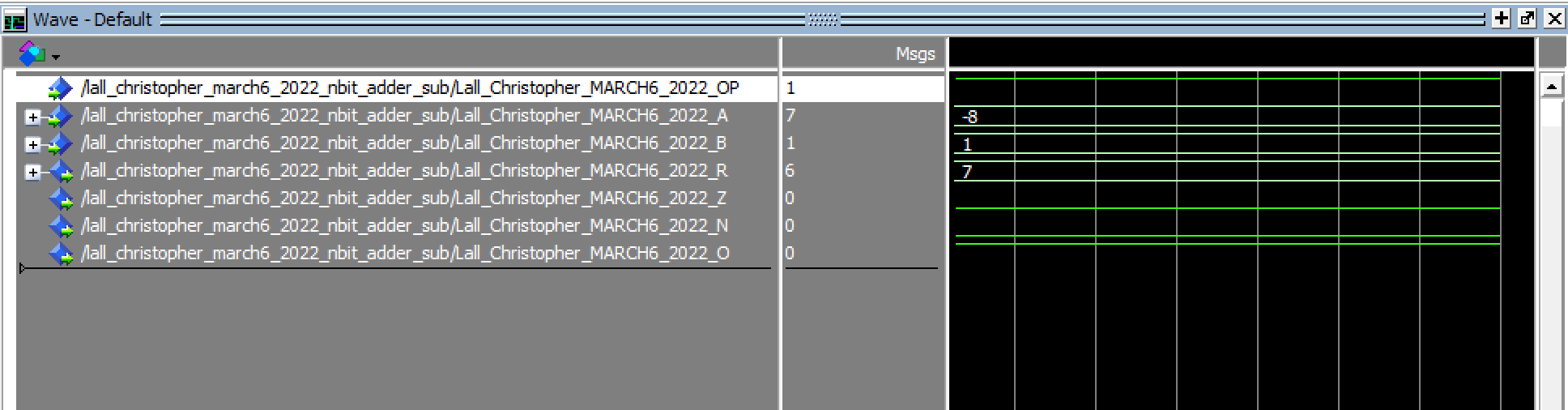


Figure 24. N-bit adder as 4bit. Case 3d waveform

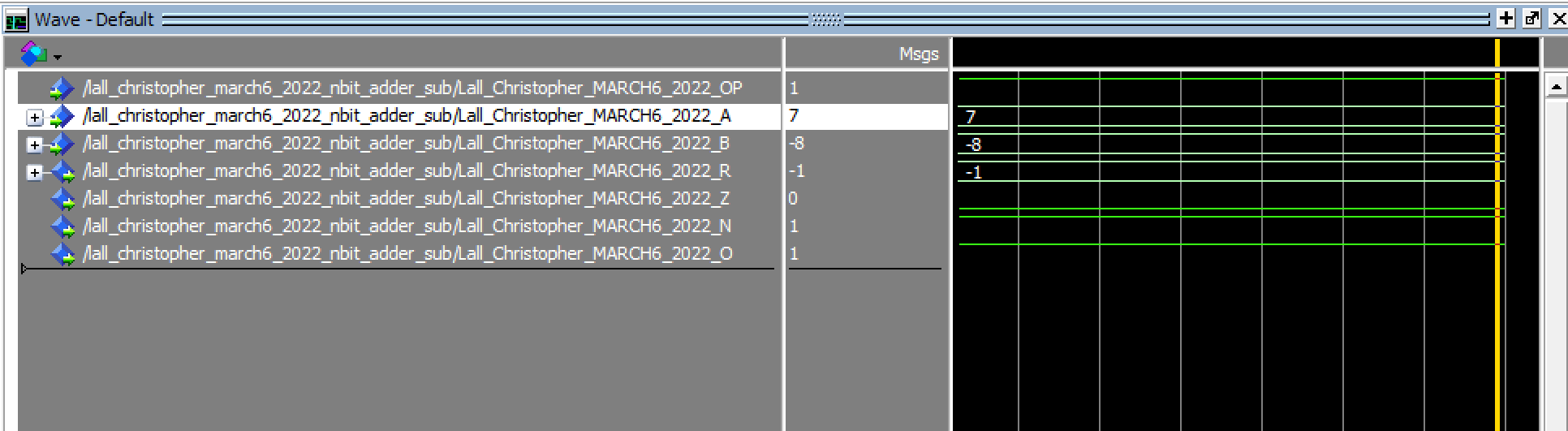


Figure 25. N-bit adder as 4bit. Case 3e waveform

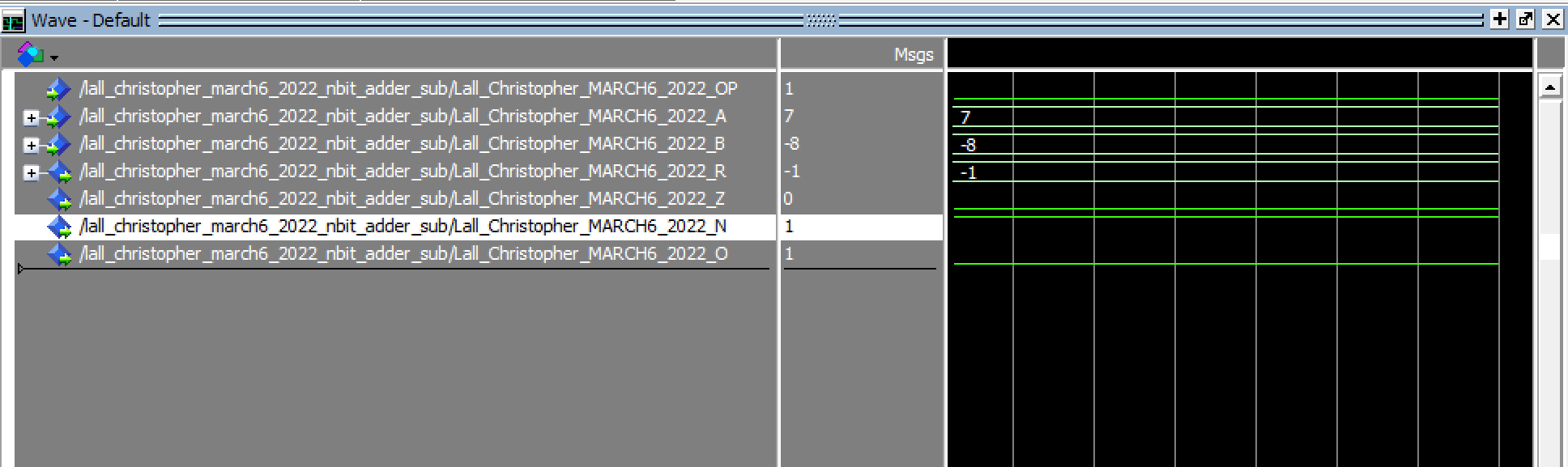


Figure 26. N-bit adder as 4bit. Case 3f waveform

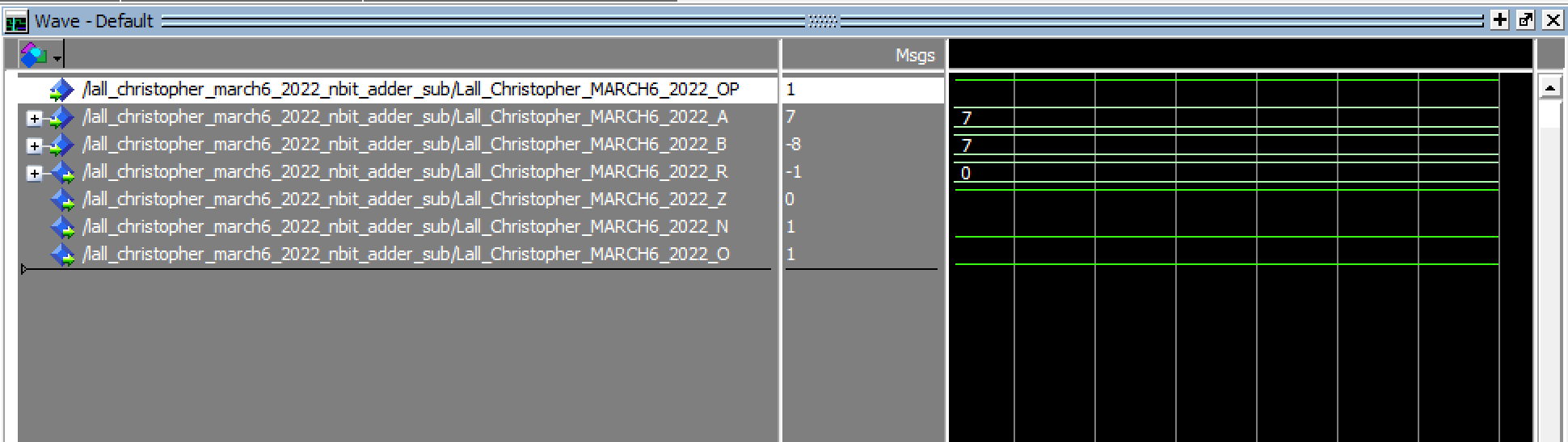


Figure 27. N-bit adder as 4bit. Case 3g waveform

### N-bit adder as 32bit:

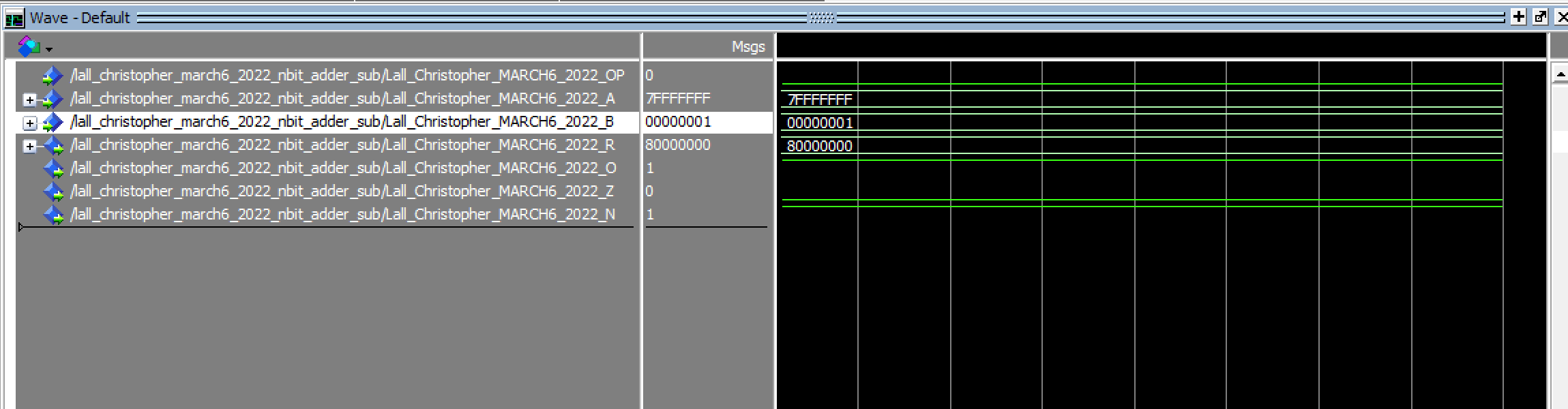


Figure 28. N-bit adder as 32bit. Case 3a waveform

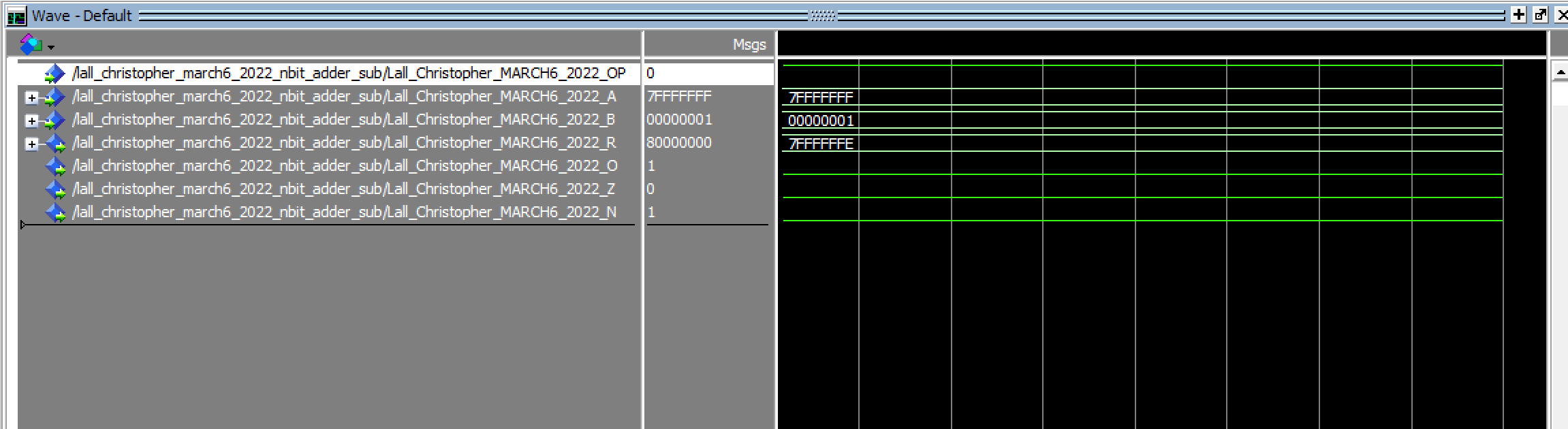


Figure 29. N-bit adder as 32bit. Case 3b waveform

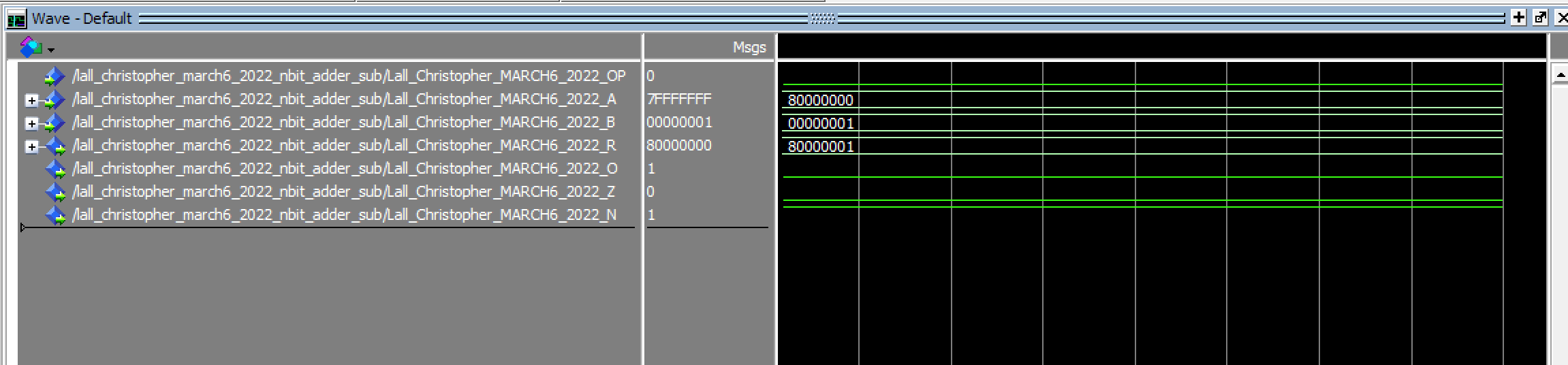


Figure 30. N-bit adder as 32bit. Case 3c waveform

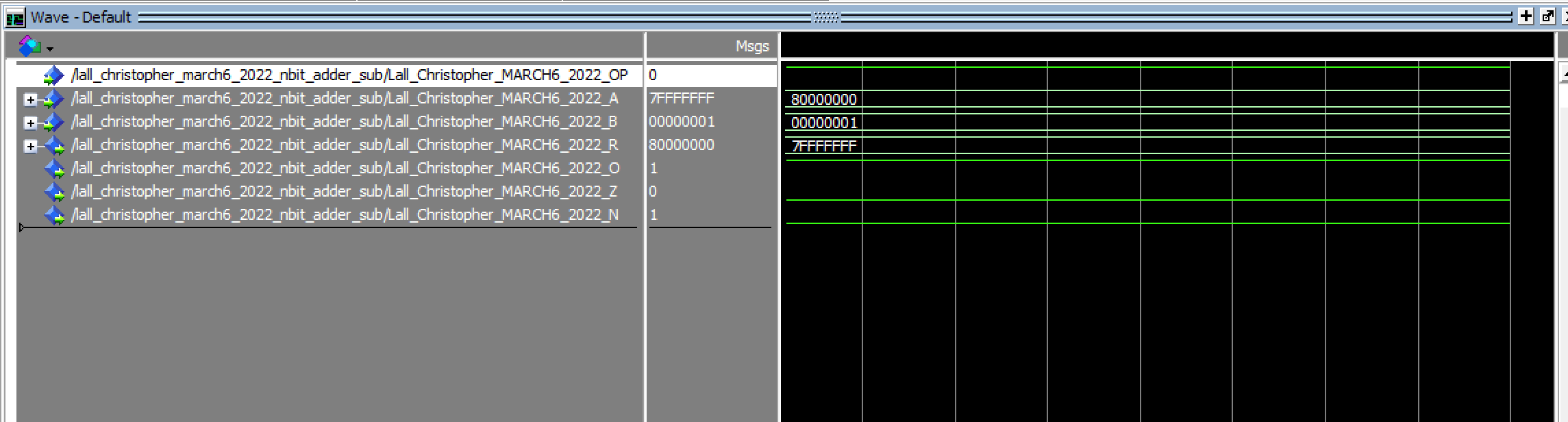


Figure 31. N-bit adder as 32bit. Case 3d waveform

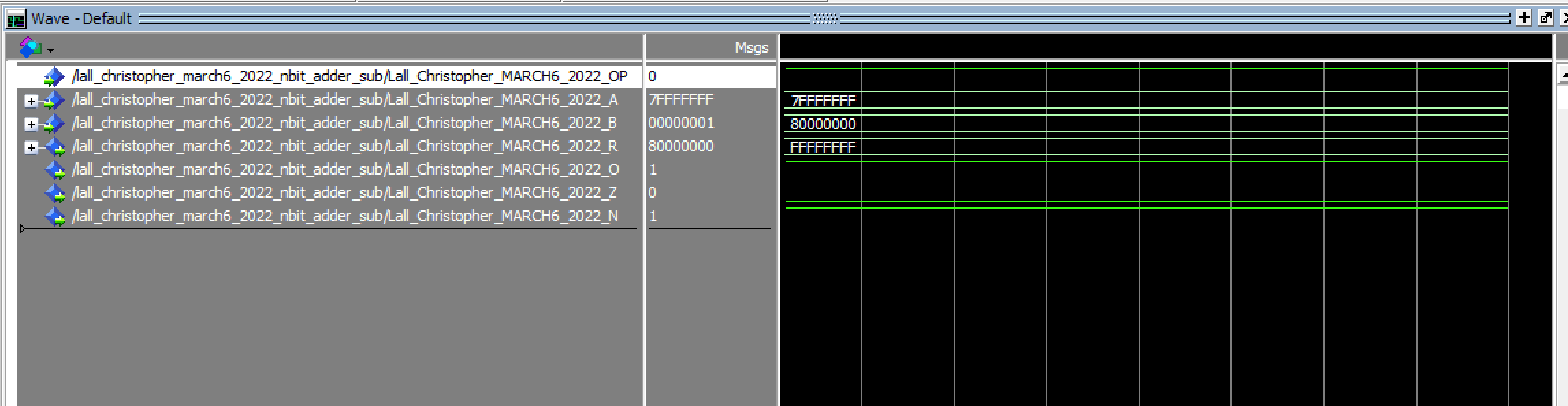


Figure 32. N-bit adder as 32bit. Case 3e waveform

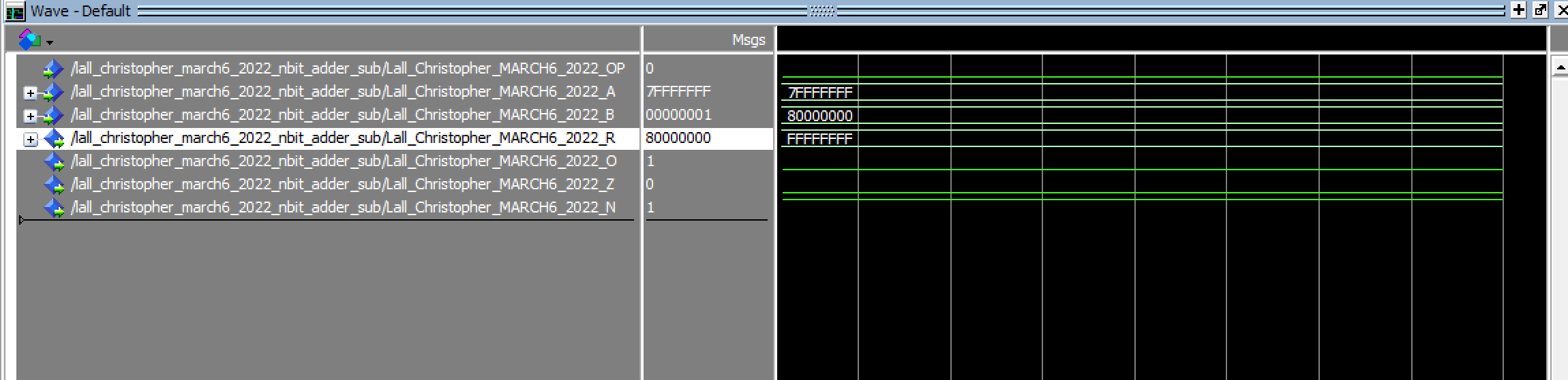


Figure 33. N-bit adder as 32bit. Case 3f waveform

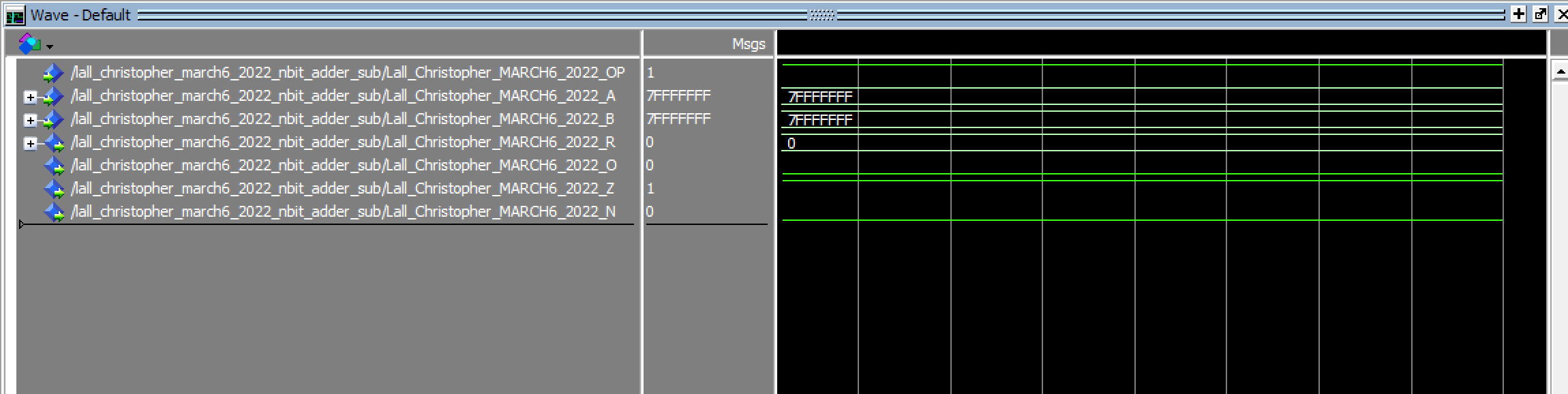


Figure 34. N-bit adder as 32bit. Case 3g waveform

## LPM:

Text

Description automatically generated

Text

Description automatically generated

Figure . VHDL Code lpm

Graphical user interface

Description automatically generated

Figure . LPM case 3a

Graphical user interface

Description automatically generated

Figure . LPM case 3b

Graphical user interface

Description automatically generated

Figure . LPM case 3c

Graphical user interface, diagram

Description automatically generated

Figure . LPM case 3d

Graphical user interface

Description automatically generated

Figure . LPM case 3e

Graphical user interface

Description automatically generated with medium confidence

Figure . LPM case 3f

Graphical user interface

Description automatically generated with medium confidence

Figure . LPM case 3g

## Test Bench:

A picture containing text

Description automatically generatedText, application

Description automatically generated

Figure 43. VHDL code for test bench

Text

Description automatically generated

Figure 44. VHDL code for add\_sub

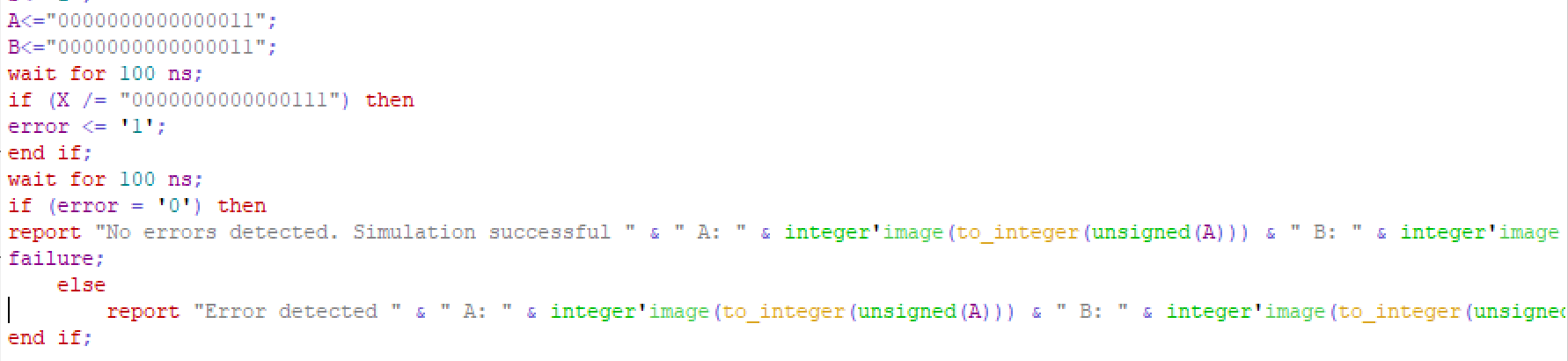


Figure 45. Error code lines in code

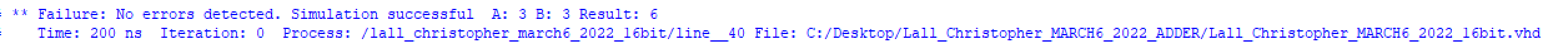


Figure 46. test bench success

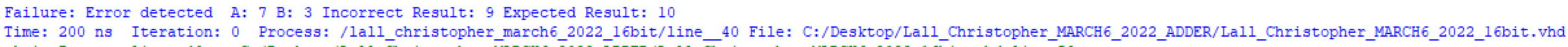


Figure 47. test bench fail case

# Analysis & Explanations:

Each of the VHDL codes allowed us to run a simulation in ModelSim to provide the operand cases. For some files such as the 4bit adder, we are required to add a subtractor which allow us to use operands in ModelSim. With these operands, we can change the type of math we want to be done. In these cases, except for the lpm file, 0 is addition and 1 is subtraction, whereas in the lpm file, it is the opposite. We also have two input waves which allow us to set each operand case and an output which we set to print in hexadecimal.

## 4Bit Explanation for each operand case:

3a) Most positive 4-bit integer is 0111 which is 7, 7+1 is 8

3b) Most positive 4-bit integer is 0111 which is 7, 7-1 is 6

3c) Most negative 4-bit integer is 1000 which is 8, 8+1 is 9

3d) Most negative 4-bit integer is 1000 which is 8, 8-1 is 7

3e) Most positive 4-bit int – most negative 4-bit int is -1

3f) Most positive 4-bit int + most negative 4-bit int is -1

3g) Most positive 4-bit int - most positive 4-bit int is 0

## 32Bit Explanation for each operand case:

3a) Most positive 32-bit integer is 7FFFFFFF + 00000001 = 80000000

3b) Most positive 32-bit integer is 7FFFFFFF – 00000001 = 7FFFFFFE

3c) Most negative 32-bit integer is 8000000 + 00000001 = 80000001

3d) Most negative 32-bit integer is 8000000 – 00000001 = 80000001

3e) Most positive 32-bit integer 7FFFFFFF - most negative 32-bit integer 80000000 = FFFFFFFF

3f) Most positive 32-bit integer 7FFFFFFF + most negative 32-bit integer 8000000 = FFFFFFFF

3g) Most positive 32-bit integer 7FFFFFFF - most positive 32-bit integer 7FFFFFFF = 0

# Conclusion:

In conclusion, I was able to create each required tasks VHDL file. I then imported the files to ModelSim in which I got my wave files. I was able to successfully do each task.