Deniz Karakay – 2443307 08.12.2022

EE445 – Homework

1. Module Designs

a. Multiplexer

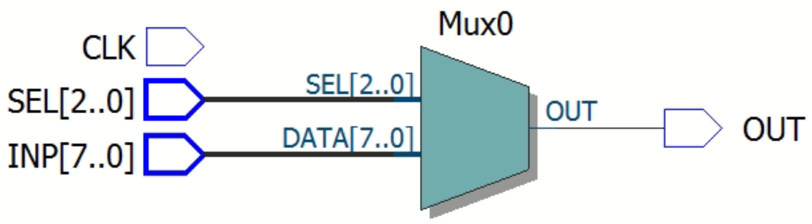


Figure 1. Multiplexer RTL from Quartus

b. Memory

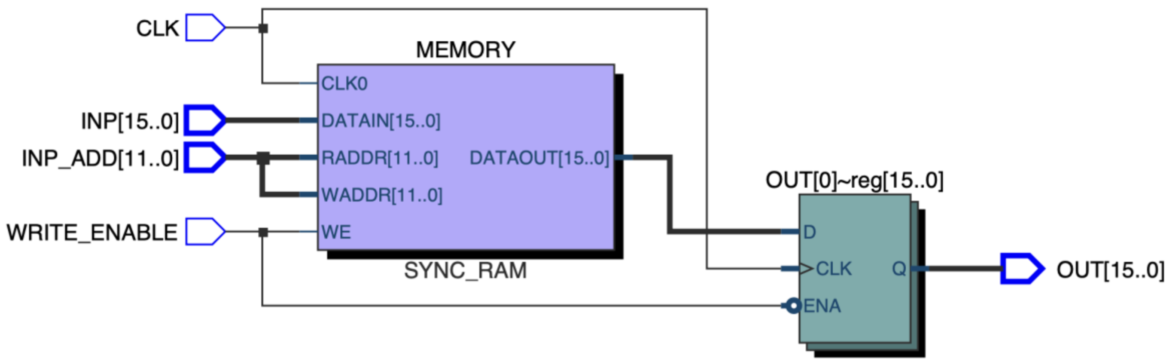


Figure 2. Memory RTL from Quartus

c. Register

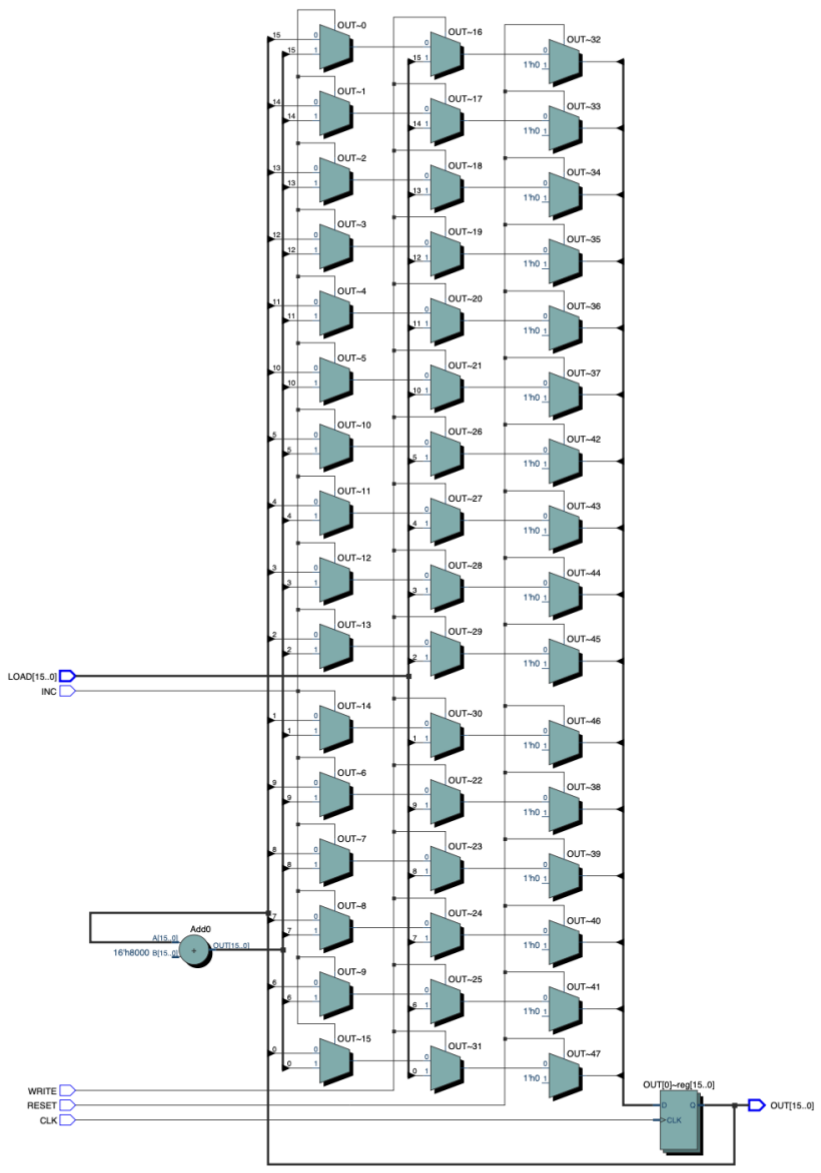


Figure 3. Register RTL from Quartus

d. ALU Unit

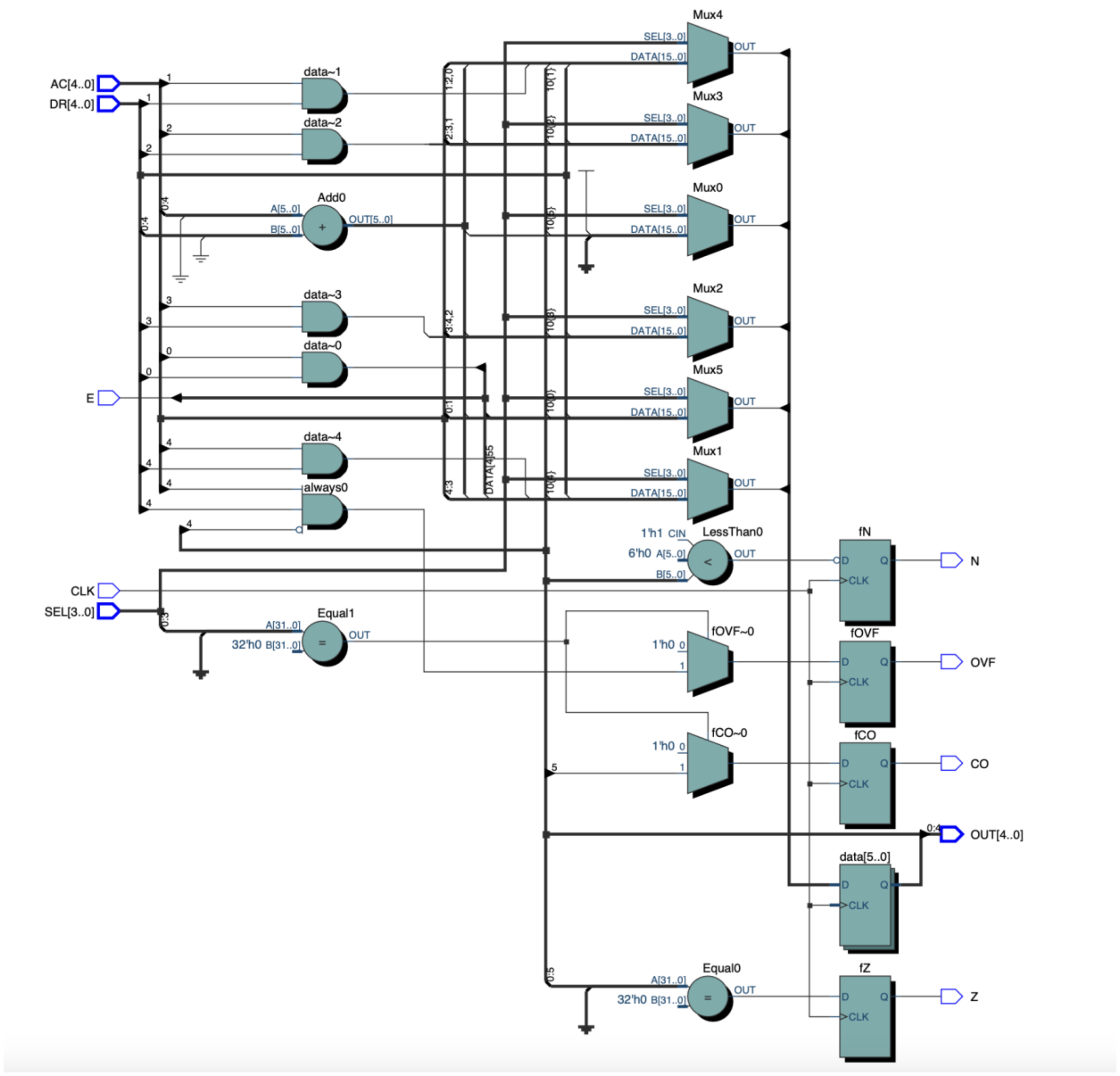


Figure 4. ALU RTL from Quartus

2. Datapath

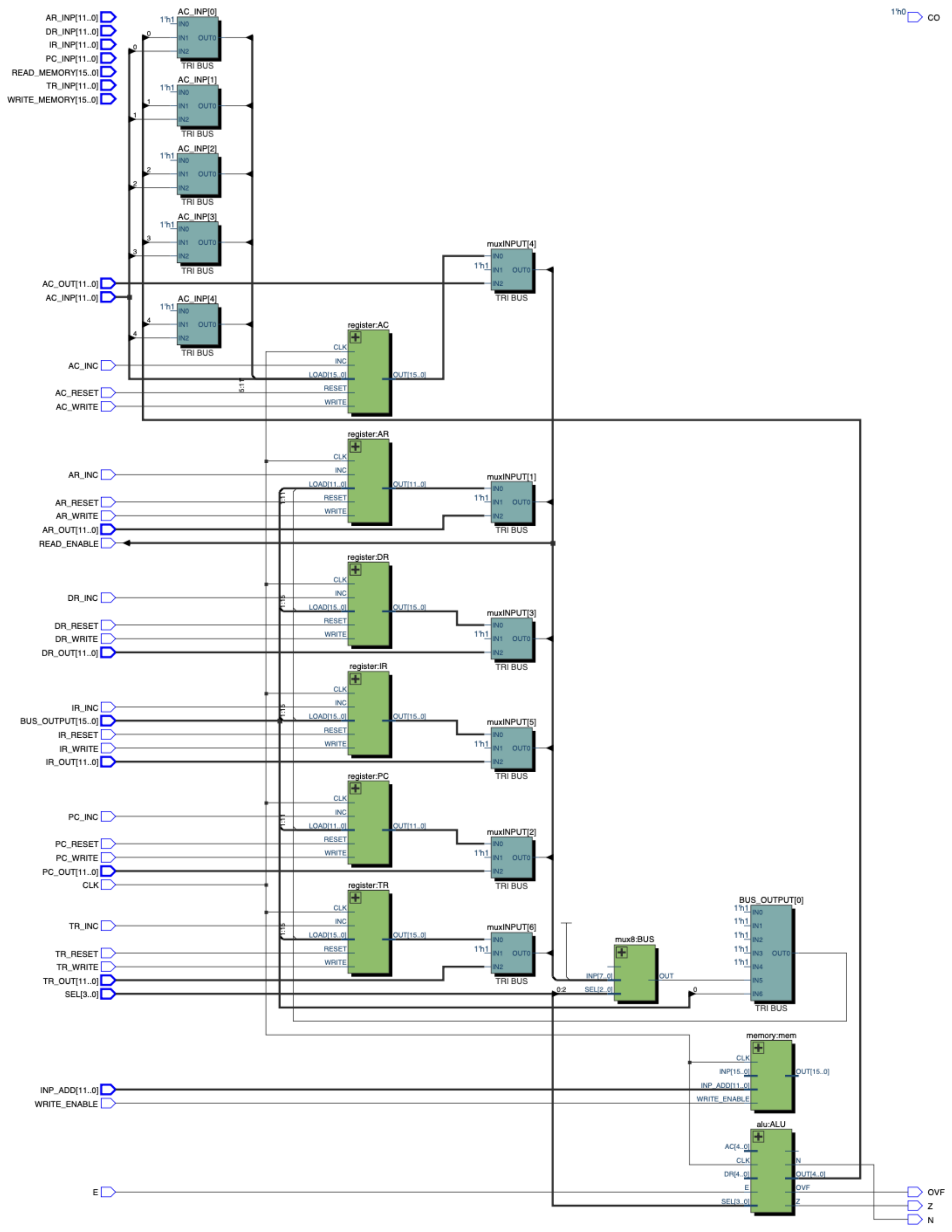


Figure 5. Datapath Unit RTL from Quartus

3. Control Unit

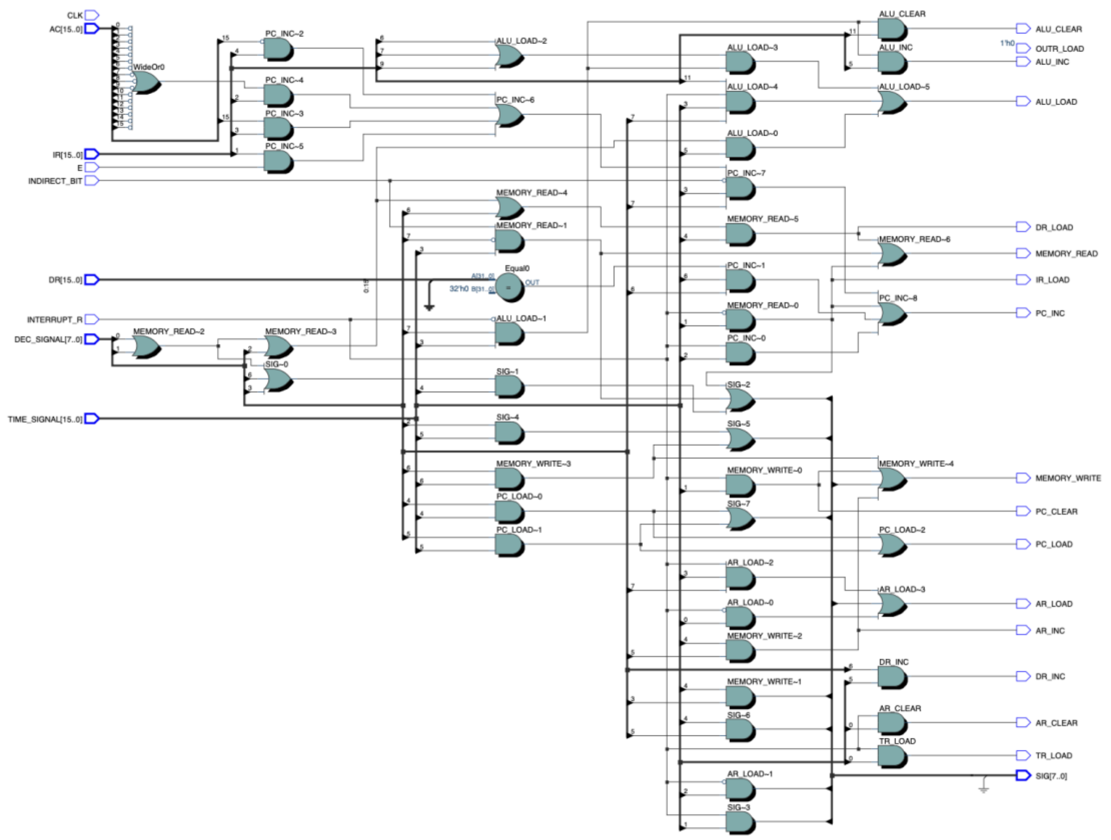


Figure 6. Control Unit RTL from Quartus

4. Cocotb Tests

a. MUX_cocotb_test.py

```
import random
import warnings

import cocotb
from cocotb.clock import Clock
from cocotb.triggers import FallingEdge



@cocotb.test()
async def mux_basic_test(dut):
    """Setup testbench and run a test."""
    #Generate the clock
    await cocotb.start(Clock(dut.CLK, 10, 'us').start(start_high=False))

    #set clkedge as the falling edge for triggers
    clkedge = FallingEdge(dut.CLK)
    #wait until the falling edge
    await clkedge

    # Check MUX
    INP=8
    SEL=2
    dut.INP.value=INP
    dut.SEL.value=SEL
    await clkedge
    print('INP 1000 SEL 2')
    print(f'INP: {dut.INP.value}')
    print(f'SEL: {dut.SEL.value}')
    print(f'OUT: {dut.OUT.value}')
    assert dut.OUT.value == 0
    print('------')

    # Check MUX
    INP = 8
    SEL = 3
    dut.INP.value = INP
    dut.SEL.value = SEL
    await clkedge
    print('INP 1000 SEL 3')
    print(f'INP: {dut.INP.value}')
    print(f'SEL: {dut.SEL.value}')
    print(f'OUT: {dut.OUT.value}')
    assert dut.OUT.value == 1
    print('------')
```

Mux Results

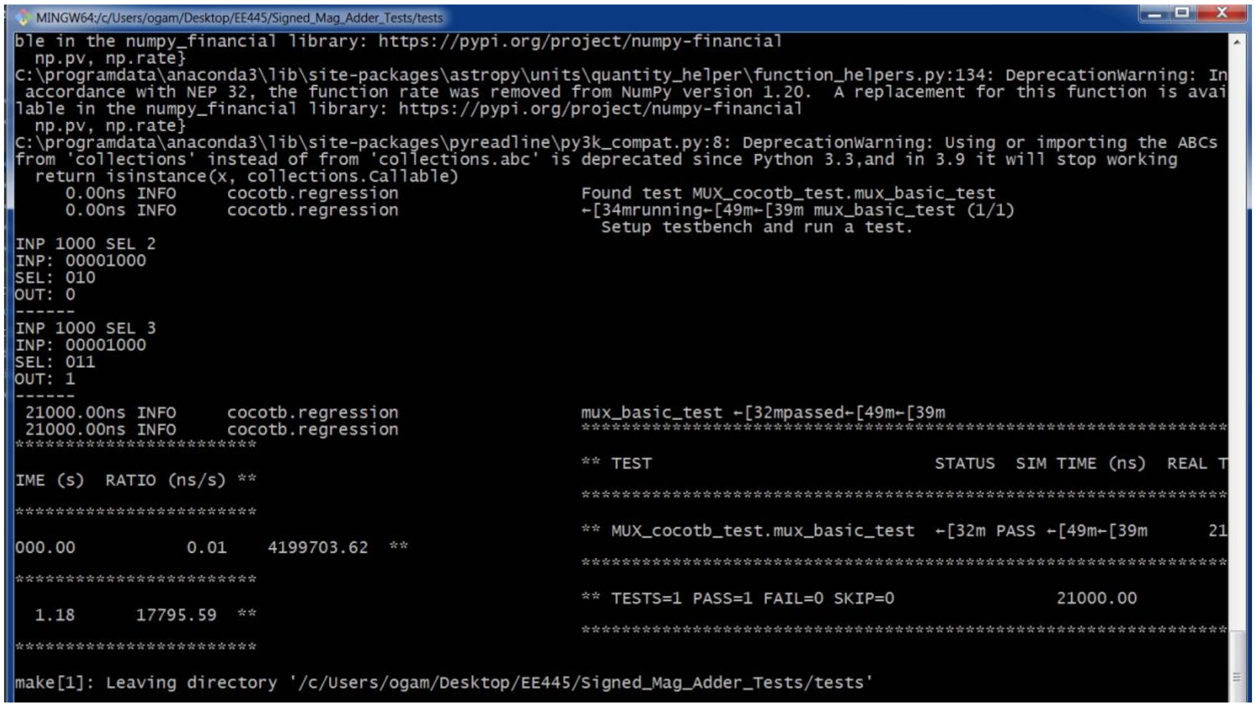
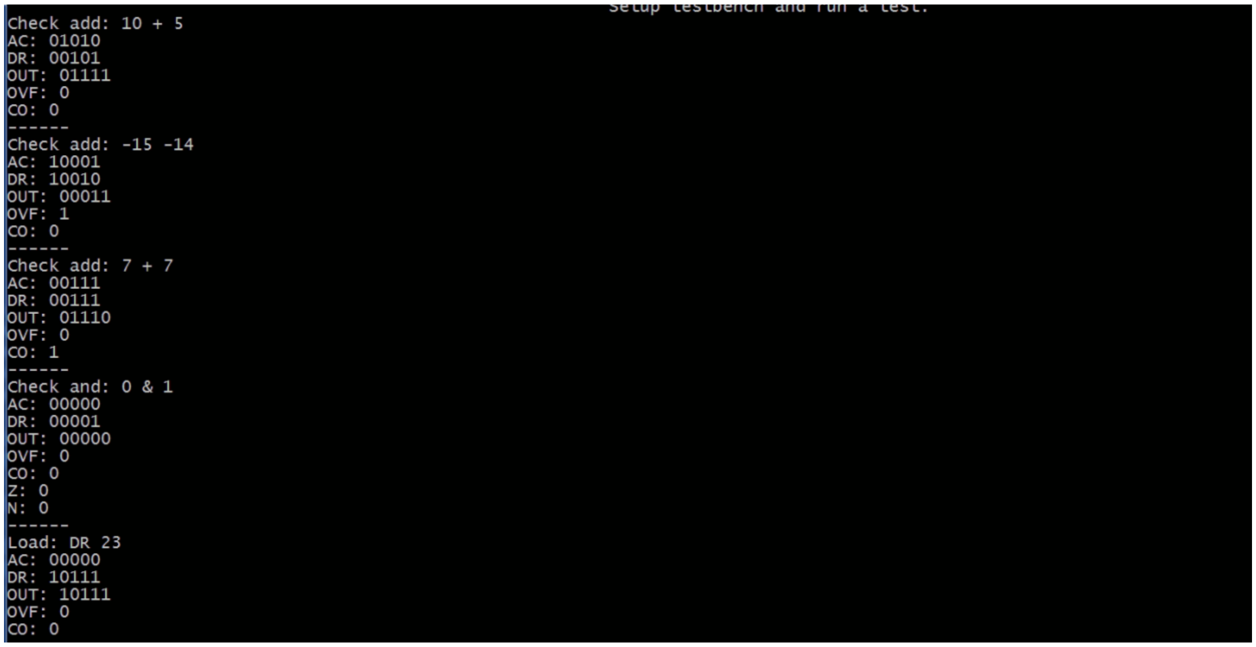

```
MINGW64:/c/Users/ogam/Desktop/EE445/Signed_Mag_Adder_Tests/tests
ble in the numpy_financial library: https://pypi.org/project/numpy-financial
  np.pv, np.rate}
C:\programdata\anaconda3\lib\site-packages\astropy\units\quantity_helper\function_helpers.py:134: DeprecationWarning: In
 accordance with NEP 32, the function rate was removed from NumPy version 1.20.  A replacement for this function is avai
lable in the numpy_financial library: https://pypi.org/project/numpy-financial
  np.pv, np.rate}
C:\programdata\anaconda3\lib\site-packages\pyreadline\py3k_compat.py:8: DeprecationWarning: Using or importing the ABCs
from 'collections' instead of from 'collections.abc' is deprecated since Python 3.3,and in 3.9 it will stop working
  return isinstance(x, collections.Callable)
     0.00ns INFO     cocotb.regression                  Found test MUX_cocotb_test.mux_basic_test
     0.00ns INFO     cocotb.regression                  ←[34mrunning←[49m←[39m mux_basic_test (1/1)
                                                          Setup testbench and run a test.
INP 1000 SEL 2
INP: 00001000
SEL: 010
OUT: 0
------
INP 1000 SEL 3
INP: 00001000
SEL: 011
OUT: 1
------
 21000.00ns INFO     cocotb.regression                  mux_basic_test ←[32mpassed←[49m←[39m
 21000.00ns INFO     cocotb.regression                  ****************************************************************
************************
                                                        ** TEST                             STATUS  SIM TIME (ns)  REAL T
IME (s)  RATIO (ns/s) **
                                                        ****************************************************************
************************
                                                        ** MUX_cocotb_test.mux_basic_test  ←[32m PASS ←[49m←[39m      21
000.00           0.01    4199703.62  **
                                                        ****************************************************************
************************
                                                        ** TESTS=1 PASS=1 FAIL=0 SKIP=0                     21000.00
  1.18      17795.59  **
                                                        ****************************************************************
************************

make[1]: Leaving directory '/c/Users/ogam/Desktop/EE445/Signed_Mag_Adder_Tests/tests'
```

Figure 7. Cocotb results for Mux

b. ALU_cocotb_test.py

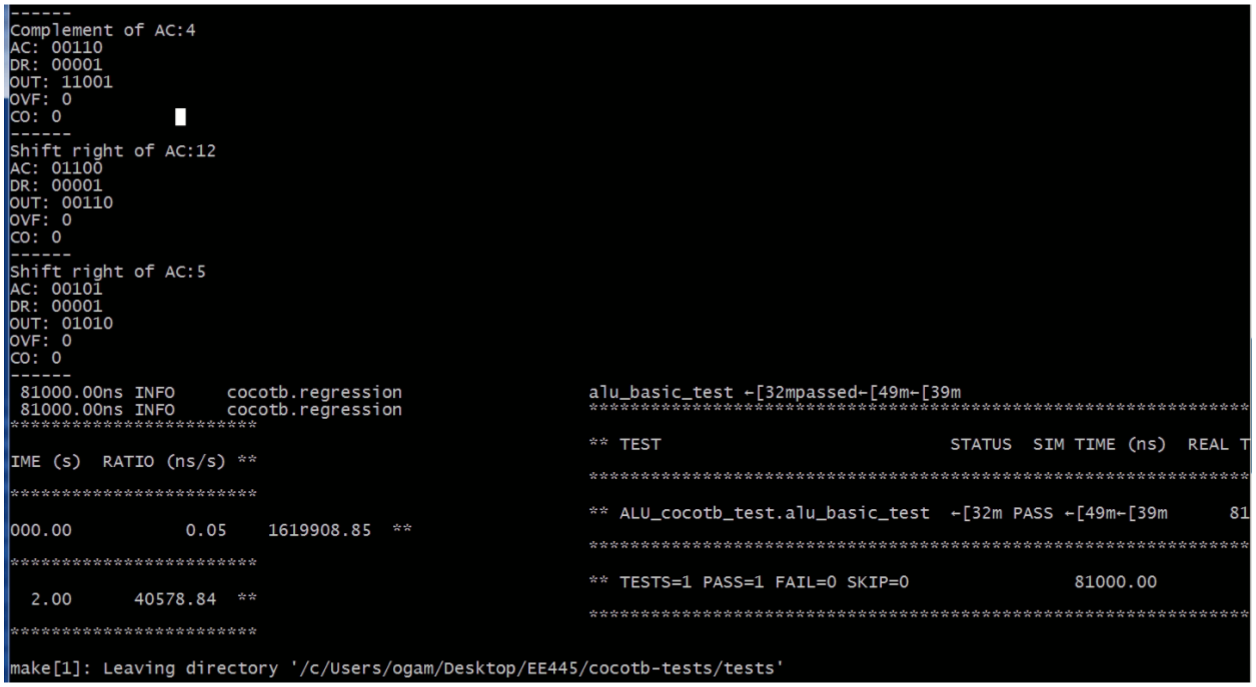
```
import random
import warnings

import cocotb
from cocotb.clock import Clock
from cocotb.triggers import FallingEdge



@cocotb.test()
async def alu_basic_test(dut):
    """Setup testbench and run a test."""
    #Generate the clock
    await cocotb.start(Clock(dut.CLK, 10, 'us').start(start_high=False))

    #set clkedge as the falling edge for triggers
    clkedge = FallingEdge(dut.CLK)
    #wait until the falling edge
    await clkedge

    # Check Add
    AC=10
    DR=5
    dut.AC.value=AC
    dut.DR.value=DR
    dut.SEL.value=0
    await clkedge
    print('Check add: 10 + 5 ')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    assert dut.OUT.value == AC + DR
    assert dut.OVF.value == 0
    assert dut.CO.value == 0
    print('------')

    # Check Add
    AC = -15
    DR = -14
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 0
    await clkedge
    print('Check add: -15 -14')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    assert dut.OUT.value != AC + DR
    assert dut.OVF.value == 1
    assert dut.CO.value == 0
    print('------')

    # Check Add
    AC = 7
    DR = 7
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 0
    await clkedge
    print('Check add: 7 + 7')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    assert dut.OUT.value == AC + DR
    assert dut.OVF.value == 0
    assert dut.CO.value == 1
    print('------')

    # Check Add
    AC = 0
    DR = 1
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 1
    await clkedge
    print('Check and: 0 & 1')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    print(f'Z: {dut.Z.value}')
    print(f'N: {dut.N.value}')
    assert dut.OUT.value == AC & DR
    assert dut.OVF.value == 0
    assert dut.CO.value == 0
    print('------')

    # Check Load
    AC = 0
    DR = 23
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 2
    await clkedge
    print('Load: DR 23')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    assert dut.OUT.value == DR
    assert dut.OVF.value == 0
    assert dut.CO.value == 0
    print('------')

    # Check Complement
    AC = 6
    DR = 1
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 3
    await clkedge
    print('Complement of AC:4')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    #assert dut.OUT.value == ~AC
    assert dut.OVF.value == 0
    assert dut.CO.value == 0
    print('------')


   # Check Shift right
    AC = 12
    DR = 1
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 4
    dut.E.value = 0
    await clkedge
    print('Shift right of AC:12')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    assert dut.OUT.value == AC/2
    assert dut.OVF.value == 0
    assert dut.CO.value == 0
    print('------')

    # Check Shift Left
    AC = 5
    DR = 1
    dut.AC.value = AC
    dut.DR.value = DR
    dut.SEL.value = 5
    dut.E.value = 0
    await clkedge
    print('Shift right of AC:5')
    print(f'AC: {dut.AC.value}')
    print(f'DR: {dut.DR.value}')
    print(f'OUT: {dut.OUT.value}')
    print(f'OVF: {dut.OVF.value}')
    print(f'CO: {dut.CO.value}')
    assert dut.OUT.value == AC * 2
    assert dut.OVF.value == 0
    assert dut.CO.value == 0
    print('------')
```

ALU Results



```
                                                            Setup testbench and run a test.
Check add: 10 + 5
AC: 01010
DR: 00101
OUT: 01111
OVF: 0
CO: 0
------
Check add: -15 -14
AC: 10001
DR: 10010
OUT: 00011
OVF: 1
CO: 0
------
Check add: 7 + 7
AC: 00111
DR: 00111
OUT: 01110
OVF: 0
CO: 1
------
Check and: 0 & 1
AC: 00000
DR: 00001
OUT: 00000
OVF: 0
CO: 0
Z: 0
N: 0
------
Load: DR 23
AC: 00000
DR: 10111
OUT: 10111
OVF: 0
CO: 0
```

Figure 8. Cocotb results for ALU Part 1



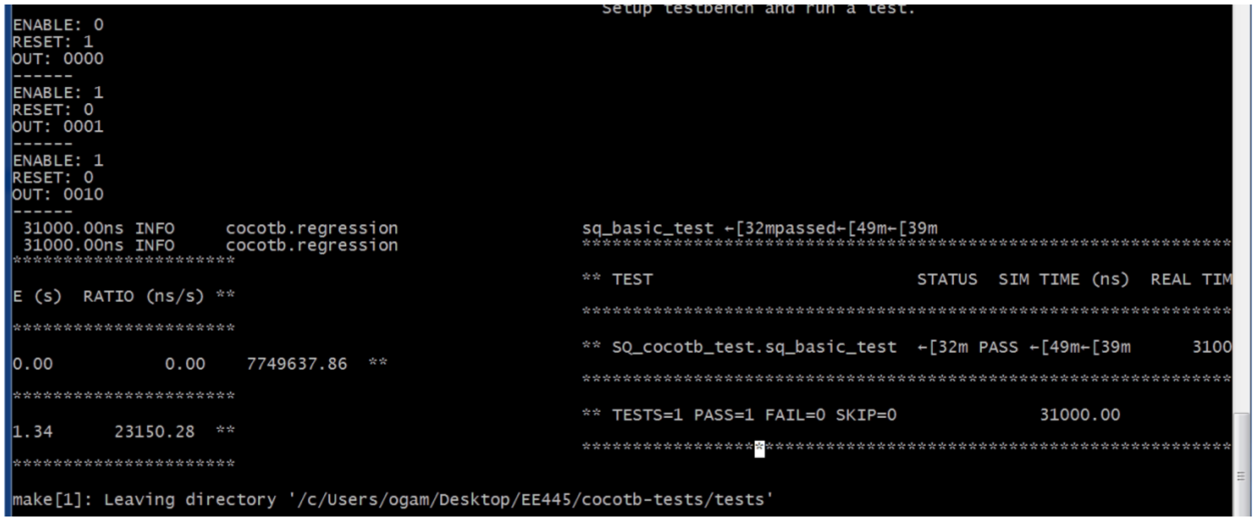
```
------
Complement of AC:4
AC: 00110
DR: 00001
OUT: 11001
OVF: 0
CO: 0
------
Shift right of AC:12
AC: 01100
DR: 00001
OUT: 00110
OVF: 0
CO: 0
------
Shift right of AC:5
AC: 00101
DR: 00001
OUT: 01010
OVF: 0
CO: 0
------
 81000.00ns INFO     cocotb.regression                    alu_basic_test ←[32mpassed←[49m←[39m
 81000.00ns INFO     cocotb.regression                    ****************************************************************
************************
                                                          ** TEST                              STATUS  SIM TIME (ns)  REAL T
IME (s)  RATIO (ns/s) **
                                                          ****************************************************************
************************
                                                          ** ALU_cocotb_test.alu_basic_test  ←[32m PASS ←[49m←[39m      81
000.00           0.05    1619908.85  **
                                                          ****************************************************************
************************
                                                          ** TESTS=1 PASS=1 FAIL=0 SKIP=0                  81000.00
  2.00      40578.84  **
                                                          ****************************************************************
************************

make[1]: Leaving directory '/c/Users/ogam/Desktop/EE445/cocotb-tests/tests'
```

Figure 9. Cocotb results for ALU Part 2

c. SQ _cocotb_test.py

```
import random
import warnings

import cocotb
from cocotb.clock import Clock
from cocotb.triggers import FallingEdge



@cocotb.test()
async def sq_basic_test(dut):
    """Setup testbench and run a test."""
    #Generate the clock
    await cocotb.start(Clock(dut.CLK, 10, 'us').start(start_high=False))

    #set clkedge as the falling edge for triggers
    clkedge = FallingEdge(dut.CLK)
    #wait until the falling edge
    await clkedge

    # Check MUX
    INC=0
    dut.INC.value=INC
    dut.RESET.value=1
    await clkedge
    print(f'ENABLE: {dut.INC.value}')
    print(f'RESET: {dut.RESET.value}')
    print(f'OUT: {dut.count.value}')
    assert dut.count.value == 0
    print('------')


  # Check MUX
    INC=1
    dut.INC.value=INC
    dut.RESET.value=0
    await clkedge
    print(f'ENABLE: {dut.INC.value}')
    print(f'RESET: {dut.RESET.value}')
    print(f'OUT: {dut.count.value}')
    assert dut.count.value == 1
    print('------')
    out = dut.count.value

    # Check MUX
    INC = 1
    dut.INC.value = INC
    dut.RESET.value = 0
    await clkedge
    print(f'ENABLE: {dut.INC.value}')
    print(f'RESET: {dut.RESET.value}')
    print(f'OUT: {dut.count.value}')
    assert dut.count.value == out+1
    print('------')
```

Sequence Counter Results



```
                                                          Setup testbench and run a test.
ENABLE: 0
RESET: 1
OUT: 0000
------
ENABLE: 1
RESET: 0
OUT: 0001
------
ENABLE: 1
RESET: 0
OUT: 0010
------
 31000.00ns INFO     cocotb.regression                    sq_basic_test ←[32mpassed←[49m←[39m
 31000.00ns INFO     cocotb.regression                    ****************************************************************
**********************
                                                          ** TEST                             STATUS  SIM TIME (ns)  REAL TIM
E (s)  RATIO (ns/s) **
                                                          ****************************************************************
**********************
                                                          ** SQ_cocotb_test.sq_basic_test  ←[32m PASS ←[49m←[39m      3100
0.00          0.00    7749637.86  **
                                                          ****************************************************************
**********************
                                                          ** TESTS=1 PASS=1 FAIL=0 SKIP=0              31000.00
1.34      23150.28  **
                                                          ****************************************************************
**********************
make[1]: Leaving directory '/c/Users/ogam/Desktop/EE445/cocotb-tests/tests'
```

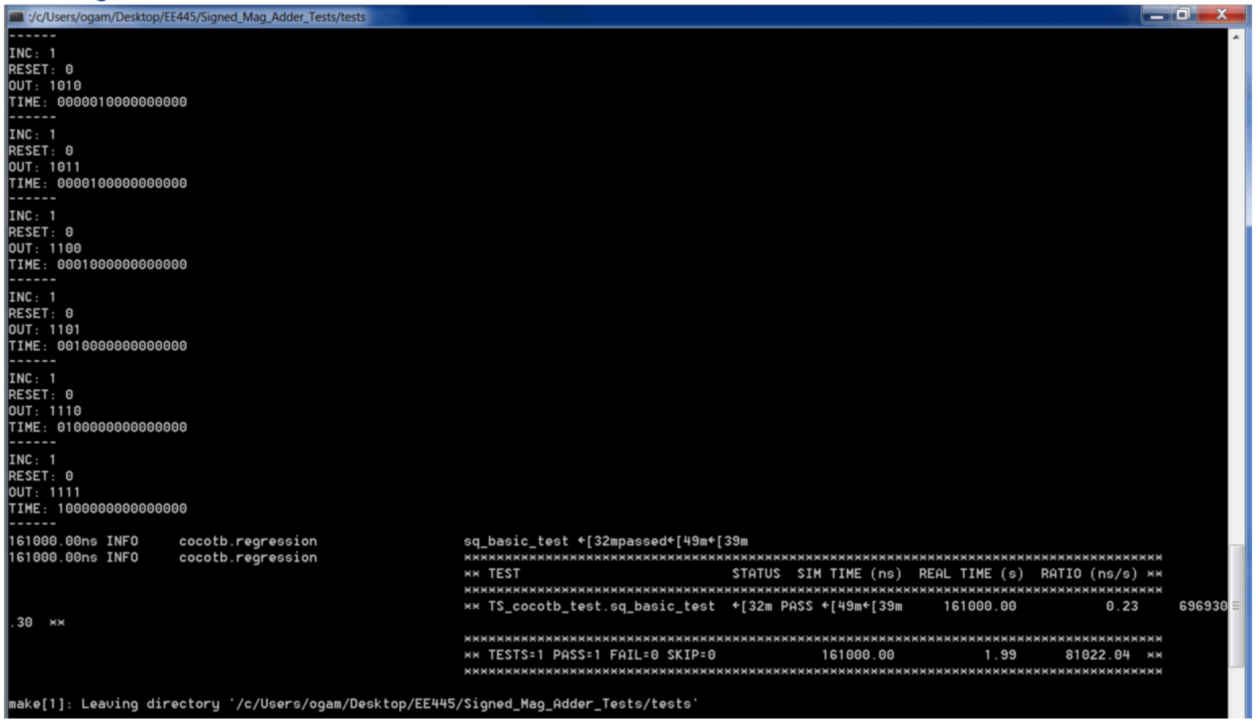
Figure 10. Cocotb results for Sequence Counter

d. TS_cocotb_test.py

```
import random
import warnings

import cocotb
```

```
from cocotb.clock import Clock
from cocotb.triggers import FallingEdge



@cocotb.test()
async def sq_basic_test(dut):
    """Setup testbench and run a test."""
    #Generate the clock
    await cocotb.start(Clock(dut.CLK, 10, 'us').start(start_high=False))

    #set clkedge as the falling edge for triggers
    clkedge = FallingEdge(dut.CLK)
    #wait until the falling edge
    await clkedge

    # Check MUX
    INC=0
    dut.INC.value=INC
    dut.RESET.value=1
    await clkedge
    print(f'INC: {dut.INC.value}')
    print(f'RESET: {dut.RESET.value}')
    print(f'OUT: {dut.OUT_SQ.value}')
    print(f'TIME: {dut.TIME_SIGNAL.value}')
    assert dut.OUT_SQ.value == 0
    print('------')

    # Check MUX
    INC = 1
    dut.INC.value = INC
    dut.RESET.value = 0
    await clkedge
    print(f'INC: {dut.INC.value}')
    print(f'RESET: {dut.RESET.value}')
    print(f'OUT: {dut.OUT_SQ.value}')
    print(f'TIME: {dut.TIME_SIGNAL.value}')
    assert dut.OUT_SQ.value == 1
    print('------')

    out = 2
    for i in range(14):
        INC = 1
        dut.INC.value = INC
        dut.RESET.value = 0
        await clkedge
        print(f'INC: {dut.INC.value}')
        print(f'RESET: {dut.RESET.value}')
        print(f'OUT: {dut.OUT_SQ.value}')
        print(f'TIME: {dut.TIME_SIGNAL.value}')
        assert dut.OUT_SQ.value == out
        print('------')
        out+=1
```

Time Signals Result



```
:/c/Users/ogam/Desktop/EE445/Signed_Mag_Adder_Tests/tests
------
INC: 1
RESET: 0
OUT: 1010
TIME: 0000010000000000
------
INC: 1
RESET: 0
OUT: 1011
TIME: 0000100000000000
------
INC: 1
RESET: 0
OUT: 1100
TIME: 0001000000000000
------
INC: 1
RESET: 0
OUT: 1101
TIME: 0010000000000000
------
INC: 1
RESET: 0
OUT: 1110
TIME: 0100000000000000
------
INC: 1
RESET: 0
OUT: 1111
TIME: 1000000000000000
------
161000.00ns INFO     cocotb.regression                  sq_basic_test ←[32mpassed←[49m←[39m
161000.00ns INFO     cocotb.regression                  ****************************************************************************************
                                                        ** TEST                             STATUS  SIM TIME (ns)  REAL TIME (s)  RATIO (ns/s) **
                                                        ****************************************************************************************
                                                        ** TS_cocotb_test.sq_basic_test  ←[32m PASS ←[49m←[39m     161000.00           0.23     696930
.30  **
                                                        ****************************************************************************************
                                                        ** TESTS=1 PASS=1 FAIL=0 SKIP=0             161000.00           1.99      81022.04  **
                                                        ****************************************************************************************

make[1]: Leaving directory '/c/Users/ogam/Desktop/EE445/Signed_Mag_Adder_Tests/tests'
```

Figure 11. Cocotb results for Time Signals