**Lab Report**

**Name: Dang Hung Thinh**

**ID: 22119137**

**Class: Monday morning class (1-5)**

*Note: every individual must submit a unique lab report form.*

1. **Summarize (in your own words) the subject of this lab:**

* This lab focuses on learning how to model and simulate simple combinational logic circuits using *SystemC* at the *behavioral level*. Specifically, we are asked to design a *4-to-1 multiplexer* and *a 2-to-4 decoder*, including creating *testbenches* and *simulation files* to verify their correct functionality.

1. **Describe the new concepts covered in this lab:**

* Introduction to *SystemC behavioral modeling*
* Designing *combinational logic* (MUX, Decoder) in *SystemC*
* Writing and using *testbenches* to simulate and verify digital designs
* Understanding the basic flow of SystemC simulation (e.g., SC\_MODULE, SC\_METHOD, sensitivity list)

1. **Describe how this lab built upon previous ones:**

* The previous lab introduced us to the *SystemC development environment* and basic syntax. This lab builds upon that foundation by applying SystemC to design *functional digital components* and validating them through *simulation and testbenches,* which is a key step in digital system design.

1. **Describe the most difficult part of this lab for you:**

* The most difficult part was writing the *testbench* to thoroughly verify the functionality of the multiplexer and decoder. It required careful planning to cover all input combinations and to ensure that timing and signal sensitivity were handled correctly in SystemC.

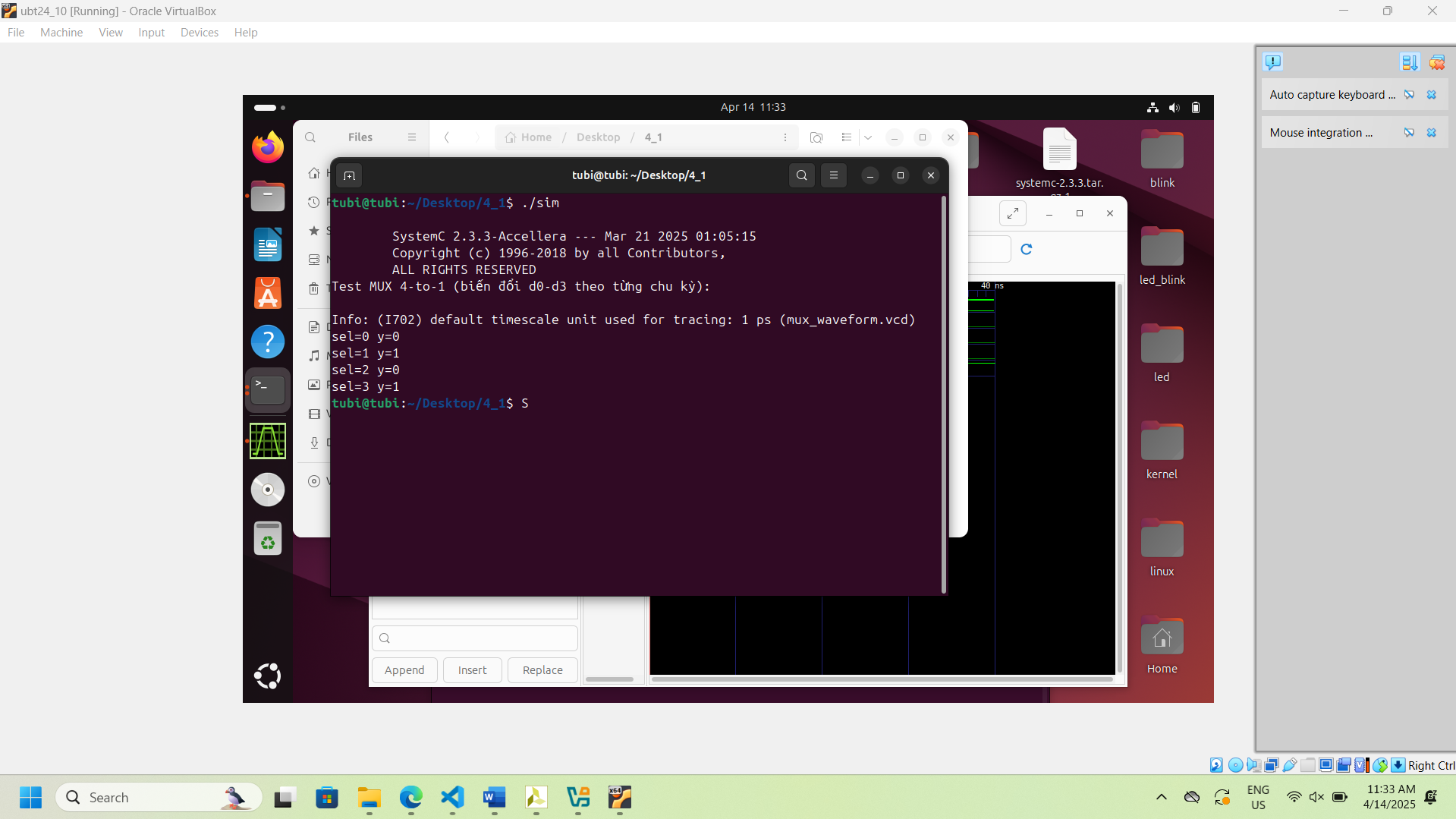
1. **Describe problems you faced and how you solved them:**

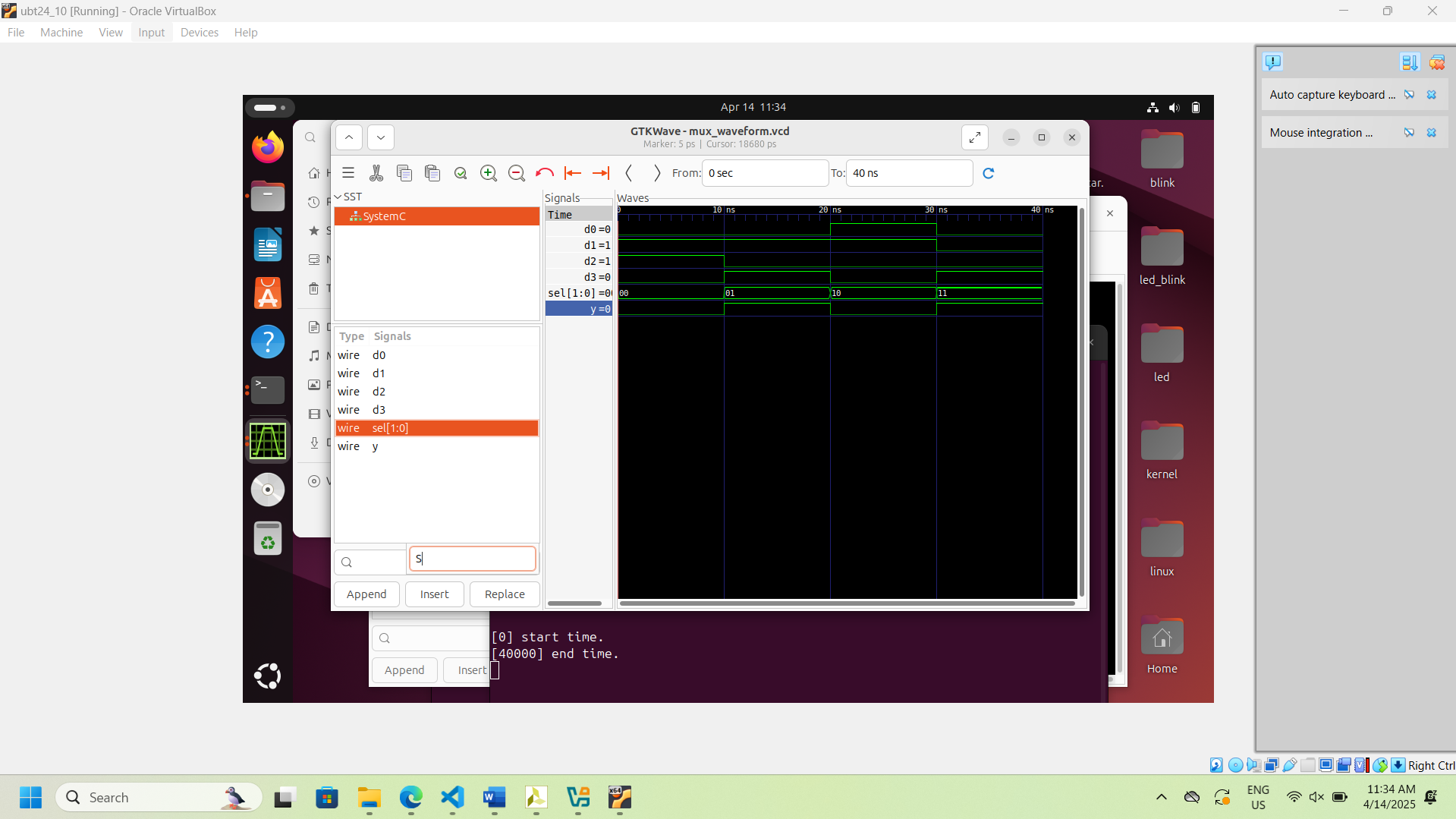
* One problem was that the *multiplexer output didn't change* when expected. After debugging, I realized I forgot to register the process method with the correct *sensitivity list* using sensitive << sel << d0 << d1 << d2 << d3;. After fixing this, the output updated correctly. Also, getting used to SC\_MODULE structure took some practice.

1. **Do you verify that the code included with this report is your's original work (yes/no)?**
2. **Submit your source code, testbench, and simulation output.**
   1. **4-to-1 multiplexer** 
      1. **On Ubuntu**

|  |
| --- |
| * **SOURCE CODE:**   #include <systemc.h>  **SC\_MODULE(Mux4to1) {**  **sc\_in<sc\_uint<2>> sel;**  **sc\_in<bool> d0, d1, d2, d3;**  **sc\_out<bool> y;**  **void do\_mux() {**  **switch (sel.read()) {**  **case 0: y.write(d0.read()); break;**  **case 1: y.write(d1.read()); break;**  **case 2: y.write(d2.read()); break;**  **case 3: y.write(d3.read()); break;**  **}**  **}**  **SC\_CTOR(Mux4to1) {**  **SC\_METHOD(do\_mux);**  **sensitive << sel << d0 << d1 << d2 << d3;**  **}**  **};** |
| * **TEST BENCH:**   **#include <systemc.h>**  **#include "mux4to1.cpp"**  **int sc\_main(int argc, char\* argv[]) {**  **sc\_signal<sc\_uint<2>> sel;**  **sc\_signal<bool> d0, d1, d2, d3;**  **sc\_signal<bool> y;**  **Mux4to1 mux("MUX");**  **mux.sel(sel);**  **mux.d0(d0);**  **mux.d1(d1);**  **mux.d2(d2);**  **mux.d3(d3);**  **mux.y(y);**  **// Tạo waveform**  **sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("mux\_waveform");**  **sc\_trace(wf, sel, "sel");**  **sc\_trace(wf, d0, "d0");**  **sc\_trace(wf, d1, "d1");**  **sc\_trace(wf, d2, "d2");**  **sc\_trace(wf, d3, "d3");**  **sc\_trace(wf, y, "y");**  **cout << "Test MUX 4-to-1 (biến đổi d0-d3 theo từng chu kỳ):\n";**  **// Chu kỳ 1: sel = 0, d0 = 1**  **sel = 0;**  **d0 = 0; d1 = 1; d2 = 1; d3 = 0;**  **sc\_start(10, SC\_NS);**  **cout << "sel=" << sel.read() << " y=" << y.read() << endl;**  **// Chu kỳ 2: sel = 1, d1 = 1**  **sel = 1;**  **d0 = 0; d1 = 1; d2 = 0; d3 = 1;**  **sc\_start(10, SC\_NS);**  **cout << "sel=" << sel.read() << " y=" << y.read() << endl;**  **// Chu kỳ 3: sel = 2, d2 = 1**  **sel = 2;**  **d0 = 1; d1 = 1; d2 = 1; d3 = 0;**  **sc\_start(10, SC\_NS);**  **cout << "sel=" << sel.read() << " y=" << y.read() << endl;**  **// Chu kỳ 4: sel = 3, d3 = 1**  **sel = 3;**  **d0 = 0; d1 = 0; d2 = 0; d3 = 1;**  **sc\_start(10, SC\_NS);**  **cout << "sel=" << sel.read() << " y=" << y.read() << endl;**  **sc\_close\_vcd\_trace\_file(wf);**  **return 0;**  **}** |

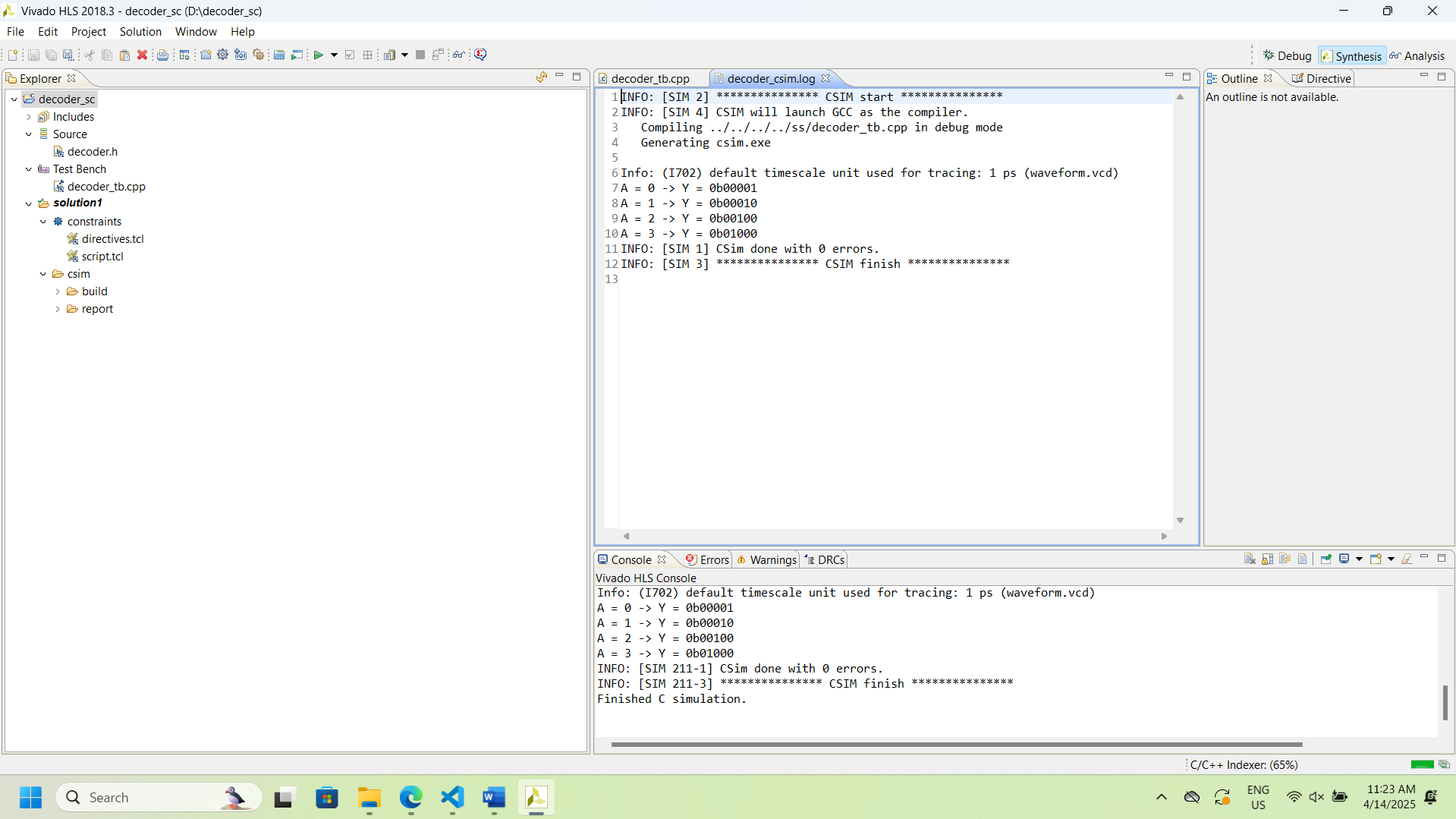
* **SIMULATION OUTPUT:**

****

****

* + 1. **On HLS vivado**

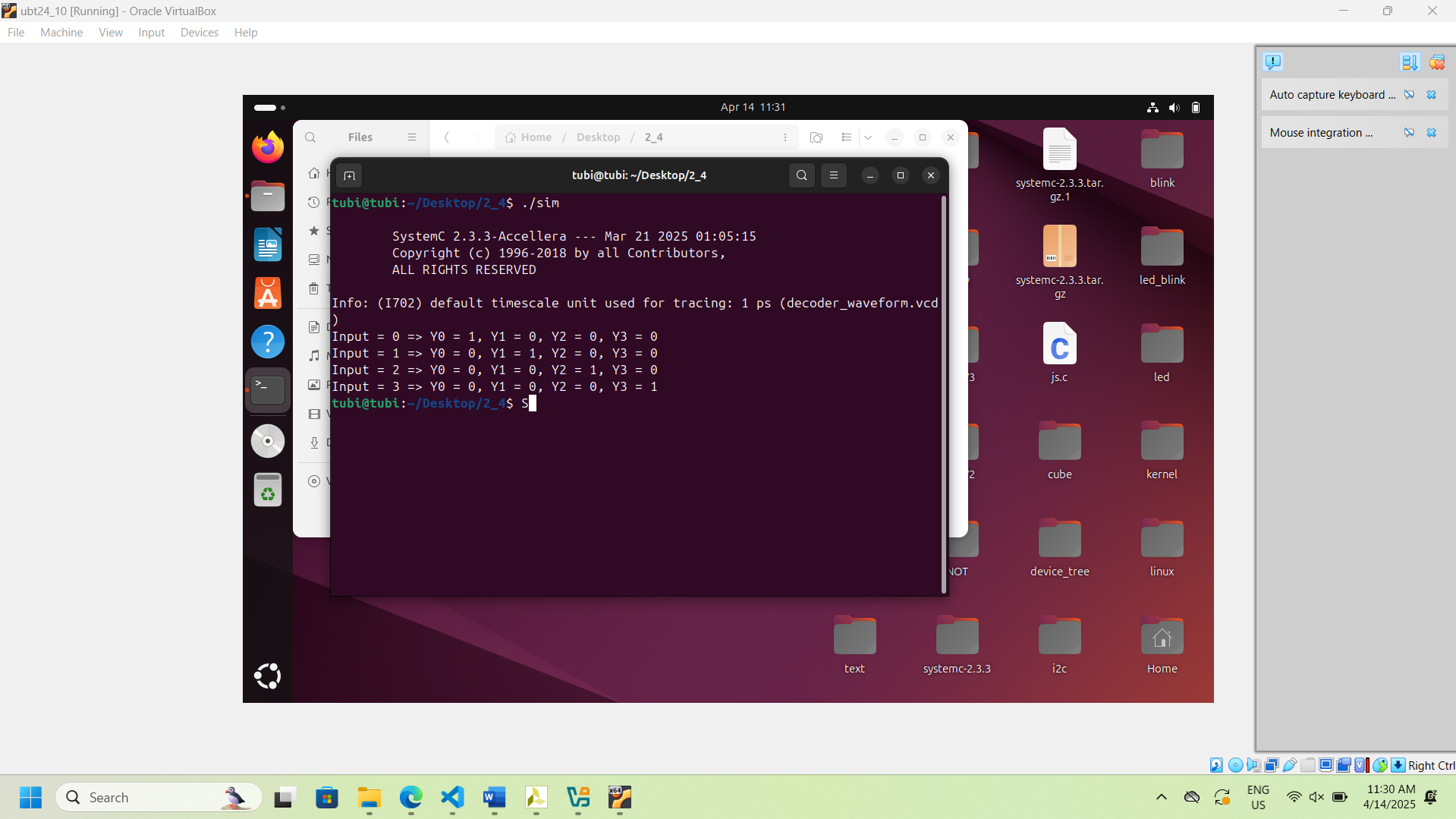
|  |
| --- |
| * **Source code:**   **#include** "mux\_func.h"  **void** **mux4to1**(  ap\_uint<2> sel,  ap\_uint<1> d0,  ap\_uint<1> d1,  ap\_uint<1> d2,  ap\_uint<1> d3,  ap\_uint<1> &y  ) {  **switch** (sel) {  **case** 0: y = d0; **break**;  **case** 1: y = d1; **break**;  **case** 2: y = d2; **break**;  **case** 3: y = d3; **break**;  **default**: y = 0; **break**;  }  } |
| * **Library:**   **#ifndef** MUX\_FUNC\_H  **#define** MUX\_FUNC\_H  **#include** <ap\_int.h>  // Hàm top-level cho Vivado HLS  **void** **mux4to1**(  ap\_uint<2> sel,  ap\_uint<1> d0,  ap\_uint<1> d1,  ap\_uint<1> d2,  ap\_uint<1> d3,  ap\_uint<1> &y  );  **#endif** |
| * **Test bench:**   **#include** <iostream>  **#include** "mux\_func.h"  **int** **main**() {  ap\_uint<2> sel;  ap\_uint<1> d0 = 1, d1 = 0, d2 = 1, d3 = 0;  ap\_uint<1> y;  std::cout << "=== MUX 4:1 TEST ===" << std::**endl**;  **for** (**int** i = 0; i < 4; ++i) {  sel = i;  mux4to1(sel, d0, d1, d2, d3, y);  std::cout << "sel = " << sel << " -> y = " << y << std::**endl**;  }  **return** 0;  } |

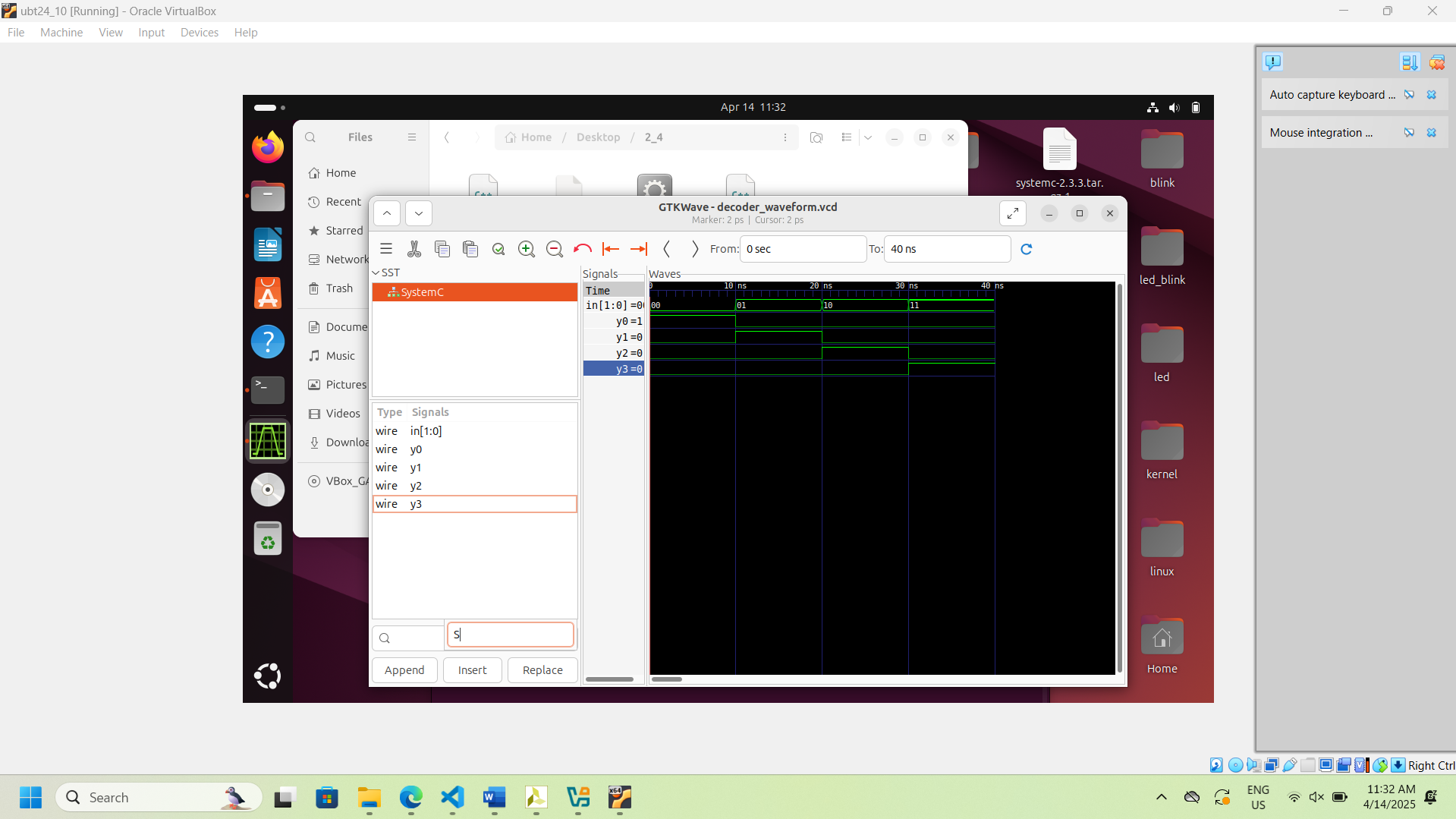


* 1. **2-to-4 decoder** 
     1. **On Ubuntu**

|  |
| --- |
| * **SOURCE CODE:**   **#include <systemc.h>**  **#include "decoder2to4.cpp"**  **int sc\_main(int argc, char\* argv[]) {**  **sc\_signal<sc\_uint<2>> in;**  **sc\_signal<bool> y0, y1, y2, y3;**  **Decoder2to4 decoder("DECODER");**  **decoder.in(in);**  **decoder.y0(y0);**  **decoder.y1(y1);**  **decoder.y2(y2);**  **decoder.y3(y3);**  **for (int i = 0; i < 4; i++) {**  **in = i;**  **sc\_start(1, SC\_NS);**  **cout << "Input = " << in.read() << " => "**  **<< "Y0 = " << y0.read() << ", "**  **<< "Y1 = " << y1.read() << ", "**  **<< "Y2 = " << y2.read() << ", "**  **<< "Y3 = " << y3.read() << endl;**  **}**  **return 0;**  **}** |
| * **TEST BENCH:**   **#include <systemc.h>**  **#include "decoder2to4.cpp"**  **int sc\_main(int argc, char\* argv[]) {**  **sc\_signal<sc\_uint<2>> in;**  **sc\_signal<bool> y0, y1, y2, y3;**  **Decoder2to4 decoder("DECODER");**  **decoder.in(in);**  **decoder.y0(y0);**  **decoder.y1(y1);**  **decoder.y2(y2);**  **decoder.y3(y3);**  **// Tạo file waveform**  **sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("decoder\_waveform");**  **sc\_trace(wf, in, "in");**  **sc\_trace(wf, y0, "y0");**  **sc\_trace(wf, y1, "y1");**  **sc\_trace(wf, y2, "y2");**  **sc\_trace(wf, y3, "y3");**  **// Test lần lượt các giá trị đầu vào**  **for (int i = 0; i < 4; i++) {**  **in = i;**  **sc\_start(10, SC\_NS);  // Đợi 10ns cho mỗi giá trị**  **cout << "Input = " << in.read() << " => "**  **<< "Y0 = " << y0.read() << ", "**  **<< "Y1 = " << y1.read() << ", "**  **<< "Y2 = " << y2.read() << ", "**  **<< "Y3 = " << y3.read() << endl;**  **}**  **// Đóng waveform**  **sc\_close\_vcd\_trace\_file(wf);**  **return 0;**  **}** |

* **SIMULATION OUTPUT:**





* + 1. **On HLS vivado**

D:.

| .cproject

| .project

| .vivado\_hls\_log\_all.xml

| vivado\_hls.app

|

+---.apc

| | autopilot.apfmapping

| |

| +---.src

| \---.tb

+---.settings

| decoder\_sc.Debug.launch

| decoder\_sc.Release.launch

| language.settings.xml

|

\---solution1

| directives.tcl

| script.tcl

| solution1.aps

| solution1.directive

| solution1.log

|

+---.autopilot

| | .autopilot\_exit

| |

| \---db

| .message\_csim.xml

| .message\_syn.xml

| autopilot.flow.log

| dsp\_style

|

+---.tcls

\---csim

| .lst\_opt.tcl

|

+---build

| | csim.exe

| | csim.mk

| | Makefile.rules

| | run\_sim.tcl

| | sim.bat

| | waveform.vcd

| |

| \---obj

| .dir

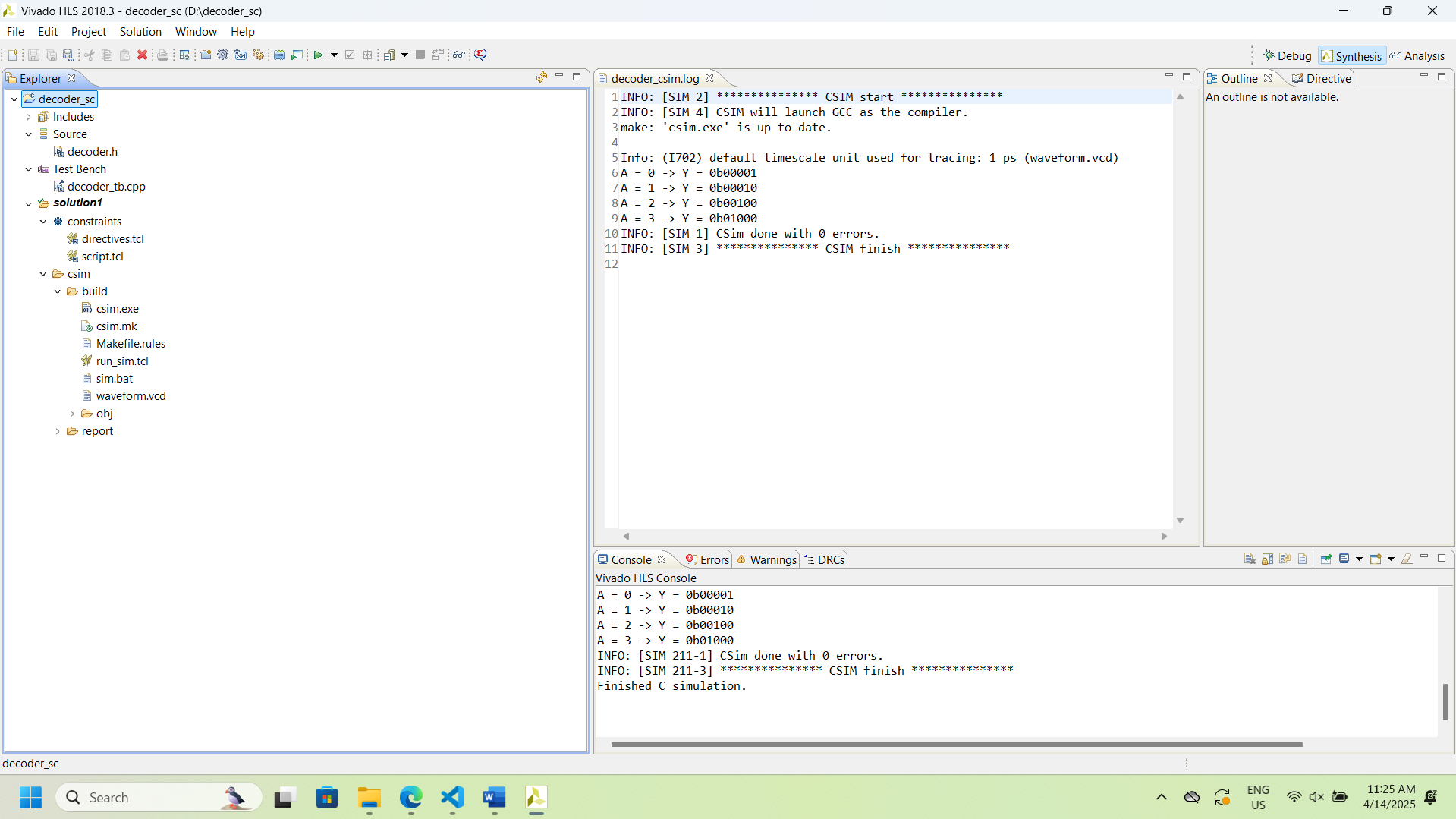
| decoder\_tb.d

| decoder\_tb.o

|

\---report

|  |
| --- |
| * **Library & source code:**   **#ifndef** DECODER\_H  **#define** DECODER\_H  **#include** <systemc.h>  SC\_MODULE(decoder) {  sc\_in<sc\_uint<2>> A;  sc\_out<sc\_uint<4>> Y;  **void** **decode\_process**() {  **switch** (A.read()) {  **case** 0: Y.write(0b0001); **break**;  **case** 1: Y.write(0b0010); **break**;  **case** 2: Y.write(0b0100); **break**;  **case** 3: Y.write(0b1000); **break**;  **default**: Y.write(0); **break**;  }  }  SC\_CTOR(**decoder**) {  SC\_METHOD(decode\_process);  sensitive << A;  }  };  **#endif** // DECODER\_H |
| * **Test bench:**   **#include** <iostream>  **#include** "systemc.h"  **#include** "decoder.h"  **int** **sc\_main**(**int** argc, **char**\* argv[]) {  sc\_signal<sc\_uint<2>> A\_sig;  sc\_signal<sc\_uint<4>> Y\_sig;  decoder uut("decoder");  uut.A(A\_sig);  uut.Y(Y\_sig);  // Ghi VCD  sc\_trace\_file \*wf = **sc\_create\_vcd\_trace\_file**("waveform");  sc\_trace(wf, A\_sig, "A");  sc\_trace(wf, Y\_sig, "Y");  **for** (**int** i = 0; i < 4; ++i) {  A\_sig.write(i);  sc\_start(1, *SC\_NS*);  std::cout << "A = " << A\_sig.read() << " -> Y = " << Y\_sig.read().**to\_string**(*SC\_BIN*) << std::**endl**;  }  **sc\_close\_vcd\_trace\_file**(wf);  **return** 0;  } |

****