Lab4 - Report

Name: Dang Hung Thinh ID: 22119137

Class: Monday morning class (1-5)

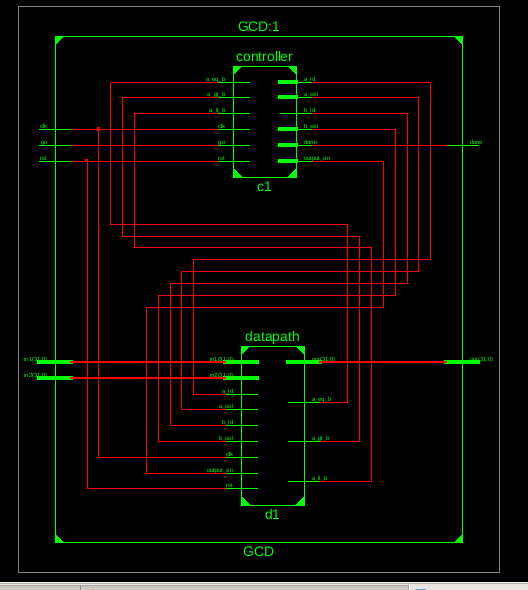
*Note: every individual must submit a unique lab report form.*

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

In this lab, the primary task was to design a circuit that computes the Greatest Common Divisor (GCD) of two numbers. This involved implementing a complete hardware system using SystemC.

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

The key new concept introduced in this lab is the FSMD model—**Finite State Machine with Datapath**. This combines control logic (FSM) and datapath elements into a structured design. Additionally, previously developed modules (like comparators, registers, muxes, etc.) were instantiated and connected using internal signals to form a cohesive system.



*2. Diogram of FSMD+Datapath*

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

This lab builds on earlier exercises by applying concepts of *SC\_METHOD* and *SC\_CTHREAD*, distinguishing when to use combinational versus sequential logic. It emphasizes integrating multiple modules into a complete system, requiring deeper understanding of signal communication and synchronization.

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

The most difficult aspect of this lab was **debugging**—specifically monitoring and verifying the behavior of internal signals during simulation, which is crucial for validating both control and datapath logic.

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

Several challenges arose, particularly around:

- Deciding when to use combinational vs sequential constructs.

- Integrating modules cleanly into the top-level GCD module.

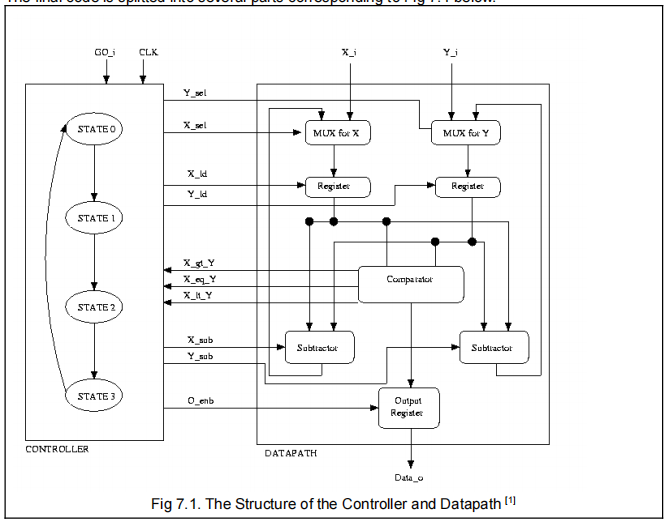
- Monitoring internal signals for debugging.

To address these, an alternative version of the top module was created where internal signals were also made external (as outputs) for easier observation during simulation. Additionally, a partial **V-model** verification approach was used, where each unit module was tested independently to catch and isolate errors early.

1. **Do you verify that the code included with this report is your's original work (yes/no)?**

Yes, I verify that all the code included with this report is my original work.

1. **Submit your source code, testbench, and simulation output.**



|  |
| --- |
| *7.2 Mux block (4 bit)*  - **LIBRARY MUX(INPUT 4BIT):**  *#ifndef MUX\_H*  *#define MUX\_H*  *#include <systemc.h>*  *SC\_MODULE(Mux) {*  *sc\_in<sc\_uint<4>> in0, in1;*  *sc\_in<bool> sel;*  *sc\_out<sc\_uint<4>> out;*  *void do\_mux() {*  *if (sel.read() == 0)*  *out.write(in0.read());*  *else*  *out.write(in1.read());*  *}*  *SC\_CTOR(Mux) {*  *SC\_METHOD(do\_mux);*  *sensitive << in0 << in1 << sel;*  *}*  *};*  *#endif* |
| *7.3 Register 4 bit*   * **LIBRARY REGISTER 4BIT:**   *#ifndef REGISTER\_H*  *#define REGISTER\_H*  *#include <systemc.h>*  *SC\_MODULE(Register) {*  *sc\_in<sc\_uint<4>> d;*  *sc\_in<bool> load, reset, clk;*  *sc\_out<sc\_uint<4>> q;*  *sc\_uint<4> temp;*  *void do\_register() {*  *if (reset.read()) {*  *temp = 0;*  *} else if (load.read() && clk.posedge()) {*  *temp = d.read();*  *}*  *q.write(temp);*  *}*  *SC\_CTOR(Register) {*  *SC\_METHOD(do\_register);*  *sensitive << clk.pos();*  *}*  *};*  *#endif* |
| *7.3. C omparator 4 bit*  **- LIBRARY COMPARATOR 4 BIT:**  *#ifndef COMPARATOR\_H*  *#define COMPARATOR\_H*  *#include <systemc.h>*  *SC\_MODULE(Comparator) {*  *sc\_in<sc\_uint<4>> a, b;*  *sc\_out<bool> a\_gt\_b, a\_lt\_b, a\_eq\_b;*  *void do\_compare() {*  *if (a.read() > b.read()) {*  *a\_gt\_b.write(true);*  *a\_lt\_b.write(false);*  *a\_eq\_b.write(false);*  *} else if (a.read() < b.read()) {*  *a\_gt\_b.write(false);*  *a\_lt\_b.write(true);*  *a\_eq\_b.write(false);*  *} else {*  *a\_gt\_b.write(false);*  *a\_lt\_b.write(false);*  *a\_eq\_b.write(true);*  *}*  *}*  *SC\_CTOR(Comparator) {*  *SC\_METHOD(do\_compare);*  *sensitive << a << b;*  *}*  *};*  *#endif* |
| *7.4. Subtractor 4 bit*  **- LIBRARY COMPARATOR 4 BIT:**  *#ifndef SUBTRACTOR\_H*  *#define SUBTRACTOR\_H*  *#include <systemc.h>*  *SC\_MODULE(Subtractor) {*  *sc\_in<sc\_uint<4>> a, b;*  *sc\_out<sc\_uint<4>> out;*  *void do\_subtract() {*  *if (a.read() > b.read())*  *out.write(a.read() - b.read());*  *else*  *out.write(b.read() - a.read());*  *}*  *SC\_CTOR(Subtractor) {*  *SC\_METHOD(do\_subtract);*  *sensitive << a << b;*  *}*  *};*  *#endif* |
| *7.4. Register output 4 bit*  **- LIBRARY REGISTER OUTPUT 4 BIT:** *// OutputRegister.h*  *#include <systemc.h>*  *SC\_MODULE(OutputRegister) {*  *sc\_in<bool> clk, reset, enable;*  *sc\_in<sc\_uint<4>> d\_in;*  *sc\_out<sc\_uint<4>> d\_out;*  *sc\_uint<4> reg;*  *void do\_output() {*  *if (reset.read())*  *reg = 0;*  *else if (clk.posedge()) {*  *if (enable.read())*  *reg = d\_in.read();*  *}*  *d\_out.write(reg);*  *}*  *SC\_CTOR(OutputRegister) {*  *SC\_METHOD(do\_output);*  *sensitive << clk.pos();*  *}*  *};* |
| *7.5. Diogram FSM of controller*  *#ifndef CONTROLLER\_H*  *#define CONTROLLER\_H*  *#include <systemc.h>*  *SC\_MODULE(Controller) {*  *sc\_in<bool> clk, reset;*  *sc\_in<bool> a\_gt\_b, a\_lt\_b, a\_eq\_b;*  *sc\_out<bool> sel\_x, sel\_y;*  *sc\_out<bool> load\_x, load\_y, load\_gcd;*  *sc\_out<sc\_uint<2>> state;*  *enum States {IDLE = 0, CALCULATE = 1, DONE = 2};*  *sc\_signal<States> current\_state, next\_state;*  *void state\_transition() {*  *if (reset.read())*  *current\_state = IDLE;*  *else if (clk.posedge())*  *current\_state = next\_state;*  *}*  *void next\_state\_logic() {*  *switch (current\_state.read()) {*  *case IDLE:*  *next\_state = CALCULATE;*  *break;*  *case CALCULATE:*  *if (a\_eq\_b.read())*  *next\_state = DONE;*  *else*  *next\_state = CALCULATE;*  *break;*  *case DONE:*  *next\_state = IDLE;*  *break;*  *}*  *state.write(current\_state.read());*  *}*  *void output\_logic() {*  *switch (current\_state.read()) {*  *case IDLE:*  *sel\_x.write(0);*  *sel\_y.write(0);*  *load\_x.write(1);*  *load\_y.write(1);*  *load\_gcd.write(0);*  *break;*  *case CALCULATE:*  *if (a\_gt\_b.read()) {*  *sel\_x.write(1);*  *sel\_y.write(0);*  *load\_x.write(1);*  *load\_y.write(0);*  *load\_gcd.write(0);*  *} else if (a\_lt\_b.read()) {*  *sel\_x.write(0);*  *sel\_y.write(1);*  *load\_x.write(0);*  *load\_y.write(1);*  *load\_gcd.write(0);*  *} else {*  *sel\_x.write(0);*  *sel\_y.write(0);*  *load\_x.write(0);*  *load\_y.write(0);*  *load\_gcd.write(1);*  *}*  *break;*  *case DONE:*  *sel\_x.write(0);*  *sel\_y.write(0);*  *load\_x.write(0);*  *load\_y.write(0);*  *load\_gcd.write(0);*  *break;*  *}*  *}*  *SC\_CTOR(Controller) {*  *SC\_METHOD(state\_transition);*  *sensitive << clk.pos();*  *SC\_METHOD(next\_state\_logic);*  *sensitive << current\_state << a\_gt\_b << a\_lt\_b << a\_eq\_b;*  *SC\_METHOD(output\_logic);*  *sensitive << current\_state << a\_gt\_b << a\_lt\_b << a\_eq\_b;*  *}*  *};*  *#endif* |
| **-LIBRARY DATAPATH:**  *#ifndef DATAPATH\_H*  *#define DATAPATH\_H*  *#include <systemc.h>*  *#include "Mux.h"*  *#include "Register.h"*  *#include "Comparator.h"*  *#include "Subtractor.h"*  *SC\_MODULE(Datapath) {*  *sc\_in<bool> clk, reset;*  *sc\_in<bool> sel\_x, sel\_y;*  *sc\_in<bool> load\_x, load\_y, load\_gcd;*  *sc\_in<sc\_uint<4>> x\_in, y\_in;*  *sc\_out<bool> a\_gt\_b, a\_lt\_b, a\_eq\_b;*  *sc\_out<sc\_uint<4>> gcd\_out;*  *sc\_signal<sc\_uint<4>> x, y, mux\_x\_out, mux\_y\_out, subtract\_out;*  *Mux \*mux\_x, \*mux\_y;*  *Register \*reg\_x, \*reg\_y, \*reg\_gcd;*  *Comparator \*comp;*  *Subtractor \*sub;*  *SC\_CTOR(Datapath) {*  *mux\_x = new Mux("mux\_x");*  *mux\_x->in0(x\_in);*  *mux\_x->in1(subtract\_out);*  *mux\_x->sel(sel\_x);*  *mux\_x->out(mux\_x\_out);*  *mux\_y = new Mux("mux\_y");*  *mux\_y->in0(y\_in);*  *mux\_y->in1(subtract\_out);*  *mux\_y->sel(sel\_y);*  *mux\_y->out(mux\_y\_out);*  *reg\_x = new Register("reg\_x");*  *reg\_x->d(mux\_x\_out);*  *reg\_x->load(load\_x);*  *reg\_x->reset(reset);*  *reg\_x->clk(clk);*  *reg\_x->q(x);*  *reg\_y = new Register("reg\_y");*  *reg\_y->d(mux\_y\_out);*  *reg\_y->load(load\_y);*  *reg\_y->reset(reset);*  *reg\_y->clk(clk);*  *reg\_y->q(y);*  *sub = new Subtractor("sub");*  *sub->a(x);*  *sub->b(y);*  *sub->out(subtract\_out);*  *comp = new Comparator("comp");*  *comp->a(x);*  *comp->b(y);*  *comp->a\_gt\_b(a\_gt\_b);*  *comp->a\_lt\_b(a\_lt\_b);*  *comp->a\_eq\_b(a\_eq\_b);*  *reg\_gcd = new Register("reg\_gcd");*  *reg\_gcd->d(x);*  *reg\_gcd->load(load\_gcd);*  *reg\_gcd->reset(reset);*  *reg\_gcd->clk(clk);*  *reg\_gcd->q(gcd\_out);*  *}*  *};*  *#endif* |
| **- TOP MODULE:**  *// Top.h*  *#ifndef TOP\_H*  *#define TOP\_H*  *#include <systemc.h>*  *SC\_MODULE(Top) {*  *sc\_in<bool> clk, reset, go;*  *sc\_in<sc\_uint<4>> x\_in, y\_in;*  *sc\_out<sc\_uint<4>> gcd\_out;*  *sc\_out<sc\_uint<2>> state\_out;*  *sc\_signal<sc\_uint<4>> x, y;*  *enum State { IDLE, WORK, DONE };*  *sc\_signal<State> state;*  *void fsm() {*  *if (reset.read() == 1) {*  *state.write(IDLE);*  *gcd\_out.write(0);*  *} else {*  *switch(state.read()) {*  *case IDLE:*  *if (go.read() == 1) {*  *x.write(x\_in.read());*  *y.write(y\_in.read());*  *// --- Fix ở đây: check nếu x hoặc y = 0 ---*  *if (x\_in.read() == 0) {*  *gcd\_out.write(y\_in.read());*  *state.write(DONE);*  *} else if (y\_in.read() == 0) {*  *gcd\_out.write(x\_in.read());*  *state.write(DONE);*  *} else {*  *state.write(WORK);*  *}*  *}*  *break;*  *case WORK:*  *if (x.read() > y.read()) {*  *x.write(x.read() - y.read());*  *} else if (y.read() > x.read()) {*  *y.write(y.read() - x.read());*  *} else {  // x == y*  *gcd\_out.write(x.read());*  *state.write(DONE);*  *}*  *break;*  *case DONE:*  *// Stay here until reset*  *break;*  *}*  *}*  *state\_out.write(state.read());*  *}*  *SC\_CTOR(Top) {*  *SC\_METHOD(fsm);*  *sensitive << clk.pos();*  *}*  *};*  *#endif // TOP\_H* |
| **- TEST BENCH:** *#include <systemc.h>*  *#include "Top.h" // Đảm bảo bạn có Top.h và Top.cpp đúng*  *int sc\_main(int argc, char\* argv[]) {*  *sc\_signal<bool> clk, reset, go;*  *sc\_signal<sc\_uint<4>> x\_in, y\_in;*  *sc\_signal<sc\_uint<4>> gcd\_out;*  *sc\_signal<sc\_uint<2>> state\_out;*  *Top top("top");*  *// Kết nối*  *top.clk(clk);*  *top.reset(reset);*  *top.go(go);*  *top.x\_in(x\_in);*  *top.y\_in(y\_in);*  *top.gcd\_out(gcd\_out);*  *top.state\_out(state\_out);*  *// Tạo file VCD trace*  *sc\_trace\_file \*wf = sc\_create\_vcd\_trace\_file("gcd\_waveform");*  *sc\_trace(wf, clk, "clk");*  *sc\_trace(wf, reset, "reset");*  *sc\_trace(wf, go, "go");*  *sc\_trace(wf, x\_in, "x\_in");*  *sc\_trace(wf, y\_in, "y\_in");*  *sc\_trace(wf, gcd\_out, "gcd\_out");*  *sc\_trace(wf, state\_out, "state\_out");*  *// Danh sách các cặp test (x,y)*  *int test\_vectors[][2] = {*  *{7, 14},*  *{12, 8},*  *{9, 6},*  *{15, 5},*  *{0, 5},*  *{7, 5}*  *};*  *int num\_tests = sizeof(test\_vectors) / sizeof(test\_vectors[0]);*  *// Bắt đầu test từng cặp*  *for (int t = 0; t < num\_tests; ++t) {*  *cout << "\n========== Test case " << t+1*  *<< ": x = " << test\_vectors[t][0]*  *<< ", y = " << test\_vectors[t][1]*  *<< " ==========\n";*  *// Reset hệ thống*  *reset.write(1);*  *clk.write(0); sc\_start(5, SC\_NS);*  *clk.write(1); sc\_start(5, SC\_NS);*  *reset.write(0);*  *clk.write(0); sc\_start(5, SC\_NS);*  *clk.write(1); sc\_start(5, SC\_NS);*  *// Set input*  *x\_in.write(test\_vectors[t][0]);*  *y\_in.write(test\_vectors[t][1]);*  *// Gửi tín hiệu go = 1 trong 1 chu kỳ clock*  *go.write(1);*  *clk.write(0); sc\_start(5, SC\_NS);*  *clk.write(1); sc\_start(5, SC\_NS);*  *go.write(0);*  *// Chạy clock liên tục đến khi state == DONE*  *int cycle = 0;*  *while (state\_out.read() != 3) { // 3 == DONE*  *clk.write(0);*  *sc\_start(5, SC\_NS);*  *clk.write(1);*  *sc\_start(5, SC\_NS);*  *cout << "Time: " << sc\_time\_stamp()*  *<< " | State: " << state\_out.read()*  *<< " | GCD: " << gcd\_out.read()*  *<< endl;*  *cycle++;*  *if (cycle > 50) { // tránh bị kẹt vô hạn*  *cout << "Error: Timeout!\n";*  *break;*  *}*  *}*  *// In kết quả cuối cùng*  *cout << "\*\* Result: GCD(" << test\_vectors[t][0]*  *<< ", " << test\_vectors[t][1] << ") = "*  *<< gcd\_out.read() << "\n";*  *}*  *sc\_close\_vcd\_trace\_file(wf);*  *return 0;*  *}*  **- OUTPUT:**  *Screenshot 2025-04-27 130411* |