**Name:** Dang Hung Thinh

**From:** HCMUTE

**Major:** Computer engineering

**--**

**Lab 05-GCD Calculator**

**The objectives of this project are as follows:**

* Become familiar with SystemC and develop a SystemC model at the behavioral level to calculate the Greatest Common Divisor (GCD) of two integers.
* Simulate and test the designed components within the SystemC environment.
* Then, implement the design in HDL (Verilog), simulate it, and observe input/output waveforms using Xilinx simulation tools.
* Next, use Yosys to map the Verilog netlist to the FreePDK45nm standard cell library.
* Finally, import the mapped Verilog netlist into Cadence Virtuoso to generate the schematic, functional, and symbol views, and analyze the waveform, delay, and power consumption of the circuit.

**Introduction**

The algorithm used to compute the GCD is as follows. Two numbers are compared ( x = y ?). If so the the GCD is found. If x > y, then x = x - y. The two numbers are then compared once again. If y > x, then y = y - x. The two numbers are then compared once again. Here is and example of our algorithim:  
  
x = 10  
y = 2  
  
Is x = y? No, x > y therefore x = x - y  
in our case, x = 10 - 2 = 8.  
  
Is x = y? No, x > y therefore x = x - y  
In our case, x = 8 - 2 = 6.  
  
Is x = y? No, x > y there fore x = x - y  
In our case, x = 6 - 2 = 4.  
  
Is x = y? No, x > y therefore x = x - y  
In our case, x = 4 - 2 = 2.  
  
Is x = y? Yes, therefore the GCD of 10 and 2 is 2.  
  
Note that 0 is not a valid input.

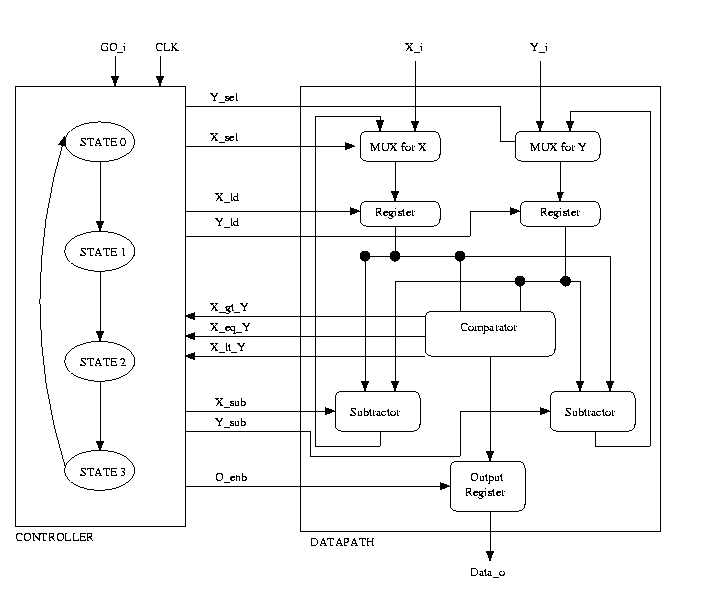
***Design Problems***

1. Implement a finite-state machine (FSM)  in SystemC to calculate the Greatest Common Divisor (GCD) of 2 numbers.

The design of the GCD calculator should be divided into 2 parts - a controller and a datapath. The controller is an FSM which issues commands to the datapath based on the current state and the external inputs. This can be a behavioral description. The datapath contains a netlist of functional units like multiplexors, registers, subtractors and a comparator, and hence this design is structural. The controller basically steps through the GCD algorithim shown above. If x = y, we have finished computing the GCD, and we go to the final state and assert the data output line. The Datapath does the actual GCD computation. It has the following components:

* **Mux:** takes 2 4-bit inputs and one select line. Based on the select line, it outputs either the 1st 4-bit number or the 2nd 4-bit number.
* **Register:** Takes a 4-bit input, a load signal, reset, and a clock signal. If the load signal is high and the clock is pulsed, it outputs the 4-bit number.
* **Comparator:** Takes 2 4-bit numbers, and assets one of 3 signals depending on whether the 1st number is less than, greater than or equal to the 2nd number.
* **Subtractor:** Takes 2 4-bit numbers, subtracts the smaller number from the larger.
* **Output Register:** Holds the GCD value. When x = y the GCD has been found and can be outputted. Because it is a register entity it should also take a clock and reset signal.

**Sample Structure of the Controller and Datapath**



--

**1. Using SystemC to implement the algorithm described above**

|  |
| --- |
| *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* |

**--**

**2. Using Verilog to describe the hardware and simulate it using Xilinx tools**

2.2. Tree files

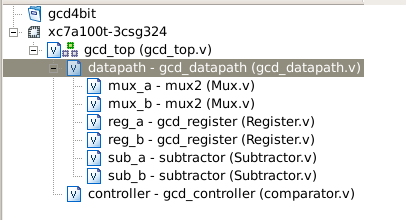


Figure 2.1. Tree files in Xilinx

2.3. Source

|  |
| --- |
| - Subtractor module:  *module subtractor (*  *input [7:0] a,*  *input [7:0] b,*  *output [7:0] result*  *);*  *assign result = a - b;*  *endmodule* |
| * Register (save input a,b):   *module gcd\_register (*  *input clk,*  *input reset,*  *input load,*  *input [7:0] d,*  *output reg [7:0] q*  *);*  *always @(posedge clk or posedge reset) begin*  *if (reset)*  *q <= 0;*  *else if (load)*  *q <= d;*  *end*  *endmodule* |
| * Mux module:   module mux2 (  input [7:0] in0,  input [7:0] in1,  input sel,  output [7:0] out  );  assign out = sel ? in1 : in0;  endmodule |
| * Datapath Module:   *smodule gcd\_datapath (*  *input clk,*  *input reset,*  *input load\_a, load\_b,*  *input sel\_a, sel\_b,*  *input [7:0] data\_in1, data\_in2,*  *output [7:0] a\_out, b\_out,*  *output eq, gt*  *);*  *wire [7:0] a\_mux\_out, b\_mux\_out;*  *wire [7:0] a\_reg\_out, b\_reg\_out;*  *wire [7:0] a\_sub, b\_sub;*  *// MUX chọn giữa đầu vào ban đầu và dữ liệu sau khi trừ*  *mux2 mux\_a (.in0(data\_in1), .in1(a\_sub), .sel(sel\_a), .out(a\_mux\_out));*  *mux2 mux\_b (.in0(data\_in2), .in1(b\_sub), .sel(sel\_b), .out(b\_mux\_out));*  *// Đăng ký giá trị a, b*  *gcd\_register reg\_a (.clk(clk), .reset(reset), .load(load\_a), .d(a\_mux\_out), .q(a\_reg\_out));*  *gcd\_register reg\_b (.clk(clk), .reset(reset), .load(load\_b), .d(b\_mux\_out), .q(b\_reg\_out));*  *// Bộ trừ*  *subtractor sub\_a (.a(a\_reg\_out), .b(b\_reg\_out), .result(a\_sub));*  *subtractor sub\_b (.a(b\_reg\_out), .b(a\_reg\_out), .result(b\_sub));*  *// Comparator*  *assign eq = (a\_reg\_out == b\_reg\_out);*  *assign gt = (a\_reg\_out > b\_reg\_out);*  *assign a\_out = a\_reg\_out;*  *assign b\_out = b\_reg\_out;*  *endmodule* |
| * Module Controller (FSM):   *module gcd\_controller (*  *input clk,*  *input reset,*  *input start,*  *input eq, gt,*  *output reg load\_a, load\_b,*  *output reg sel\_a, sel\_b,*  *output reg done*  *);*  *// Các trạng thái*  *parameter IDLE = 2'b00,*  *LOAD = 2'b01,*  *CALC = 2'b10,*  *DONE = 2'b11;*  *reg [1:0] state, next;*  *// Bộ logic trạng thái kế tiếp + tín hiệu điều khiển*  *always @(\*) begin*  *// Mặc định*  *load\_a = 0;*  *load\_b = 0;*  *sel\_a = 0;*  *sel\_b = 0;*  *done = 0;*  *case (state)*  *IDLE: begin*  *if (start)*  *next = LOAD;*  *else*  *next = IDLE;*  *end*  *LOAD: begin*  *load\_a = 1;*  *load\_b = 1;*  *next = CALC;*  *end*  *CALC: begin*  *if (eq) begin*  *next = DONE;*  *end else if (gt) begin*  *sel\_a = 1;*  *load\_a = 1;*  *next = CALC;*  *end else begin*  *sel\_b = 1;*  *load\_b = 1;*  *next = CALC;*  *end*  *end*  *DONE: begin*  *done = 1;*  *next = IDLE;*  *end*  *default: next = IDLE;*  *endcase*  *end*  *// Cập nhật trạng thái*  *always @(posedge clk or posedge reset) begin*  *if (reset)*  *state <= IDLE;*  *else*  *state <= next;*  *end*  *endmodule* |
| * Schematic symbol: |
|  |
| * GCD(28,16)=4 |
| * GCD(0,4) =4 |
| TCL, rtl yosys ubuntu:   * Schematic in cadence: |
| * GCD(28,16) * Input: * Output: |
| * GCD(56,8) * Input: * Output: |