VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL AND ELECTRONICS ENGINEERING

—o0o—

BIG PROJECT 1 REPORT

Floating Point Adder and Subtractor

SUPERVISOR: Nguyễn Trung Hiếu

SUBJECT: Digital System Design and Verification

GROUP: 08

List of Members

STT	MSSV	Họ Và Tên	Lớp
1	2210780	Nguyễn Đại Đồng	L01
2	2213874	Nguyễn Thanh Tùng	L01
3	2213496	Nguyễn Quốc Tín	L01

Ho Chi Minh, ../../20..

Mục lục

Danh sách hình vẽ

Danh sách bảng

List of Listings

Chapter 1. Requirement

Thiết kế một bộ **Floating Point (FPU)** có thể cộng và trừ hai số **floating-point 32-bit**. Phương pháp thiết kế sử dụng thuần là bộ tổ hợp.

Sau khi thiết kế thành công, cần:

- Viết chương trình kiểm định thiết kế **FPU** trên.
- Thực hiện thiết kế trên KIT FPGA để kiểm tra chức năng của thiết kế.
- Tiến hành tổng hợp thiết kế sử dụng các standard cell trong thư viện có sẵn tại các trường hợp thực tế và loại bỏ các glitchs.

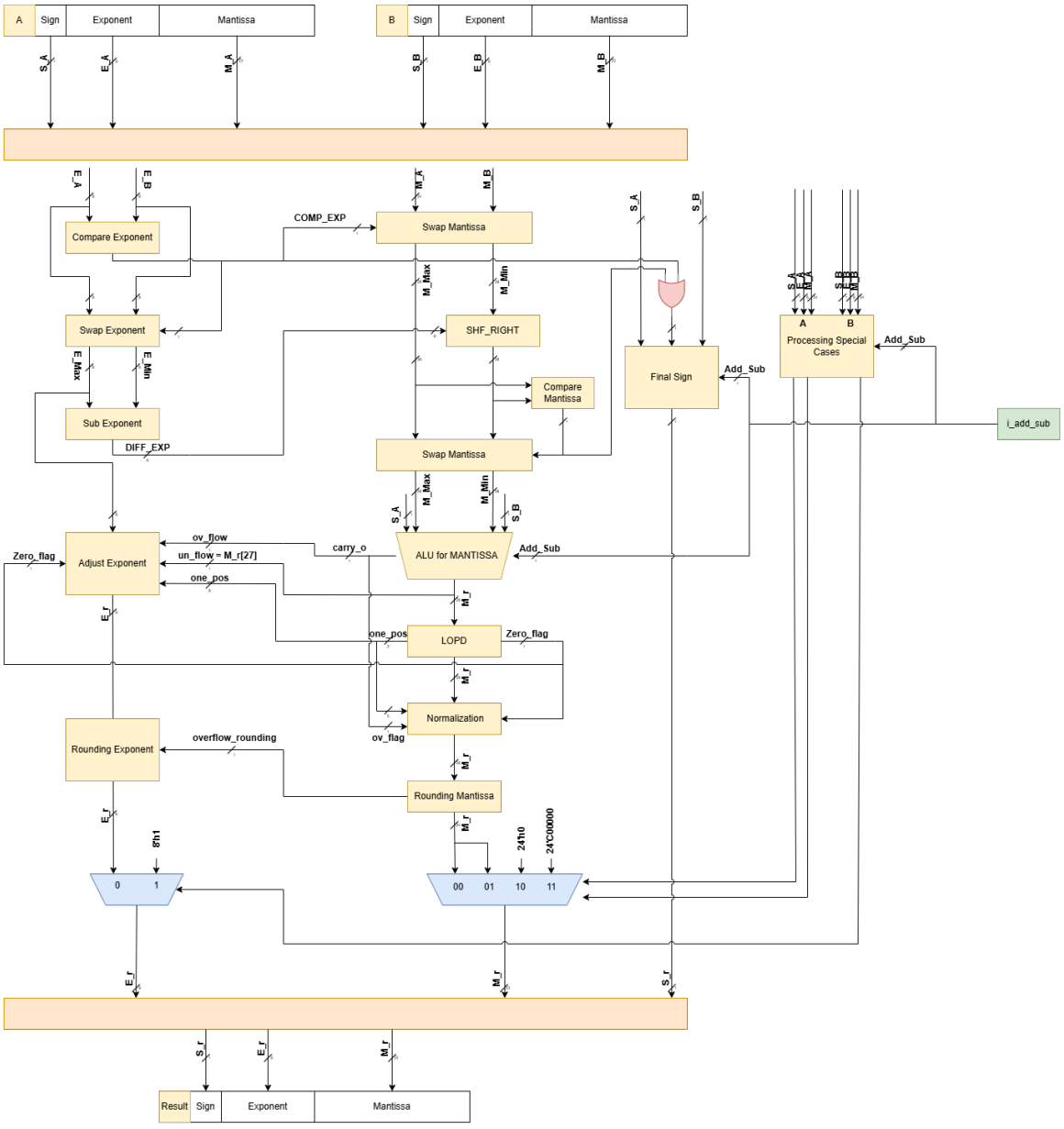
Ngõ vào và ngõ ra của thiết kế được mô tả như sau:

Name	Port Type	Width	Description
i_32_a	Input	32	32-bit floating point number A
i_32_b	Input	32	32-bit floating point number B
i_add_sub	Input	1	Operation: 0 = add, 1 = sub
o_32_s	Output	32	32-bit floating point output
o_ov_flag	Output	1	Overflow Flag: 1 = overflow, 0 = no overflow occurs
o_un_flaag	Output	1	Underflow Flag: 1 = underflow, 0 = no underflow occurs

Bảng 1.1: Input and Output Ports.

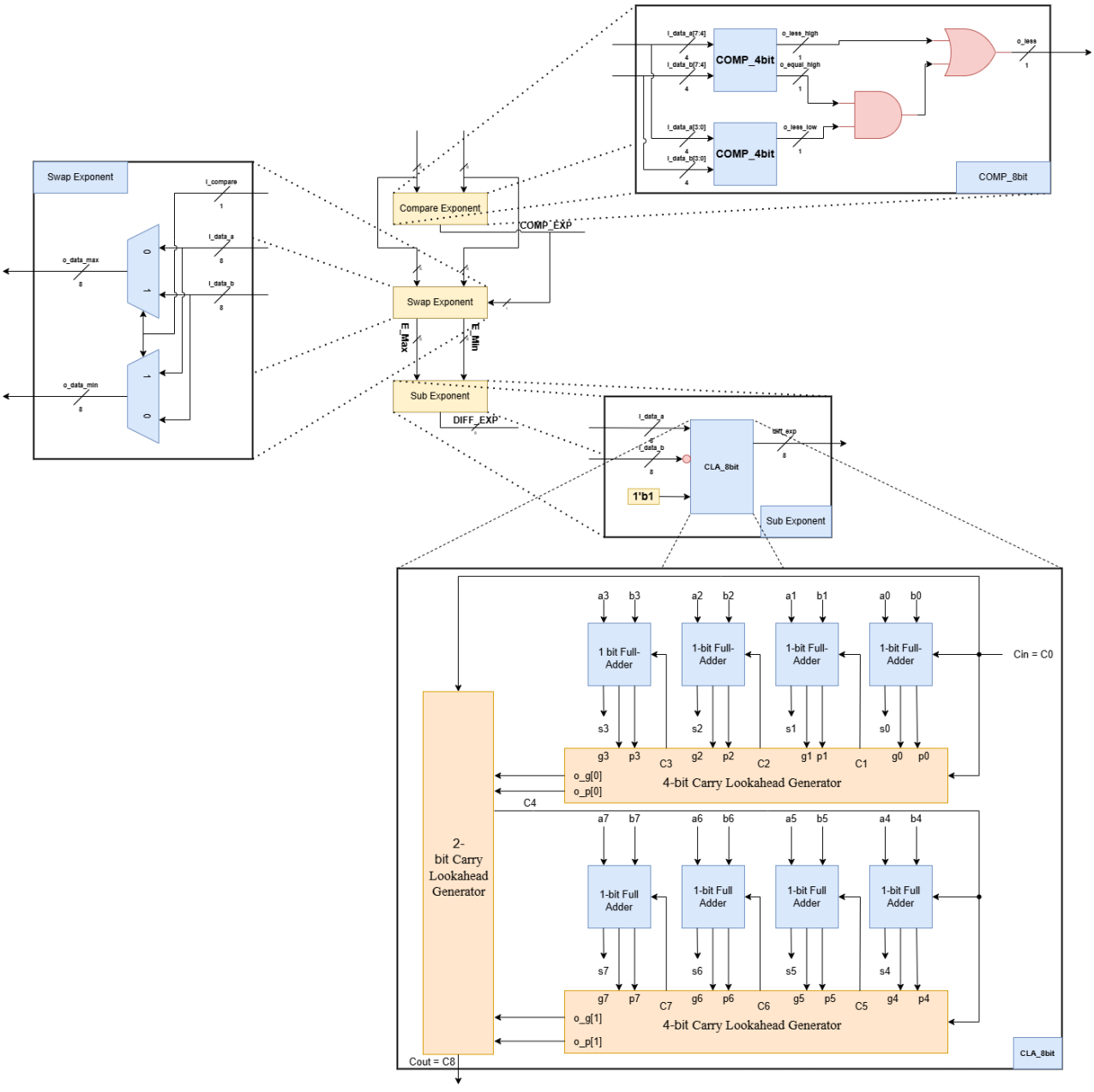
Chapter 2. Design FPU

Ta có thiết kế tổng quan của bộ tính toán FPU:



Hình 2.1: Sơ đồ khối tổng quan của bộ FPU.

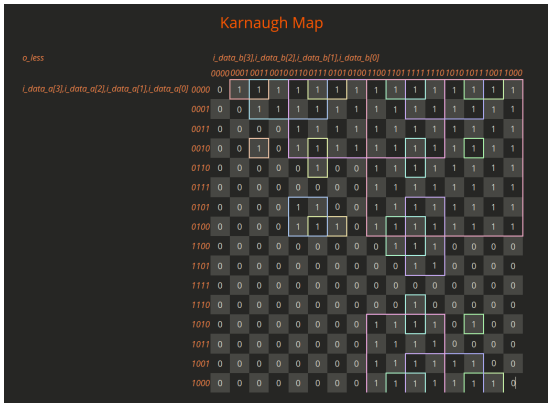
2.1 Bộ tiền xử lý liên quan đến Exponent



Hình 2.2: Bộ tính toán giá trị |EXP_A − EXP_B|.

Bộ có chức năng thực hiện tính giá trị trị khác nhau giữa hai Exponent của A và B. Trong đó,

- Bộ COMP_4bit: là bộ so sánh 4-bit hai số được triển khai dựa vào bảng sự thật và rút gọn bìa K như sau:

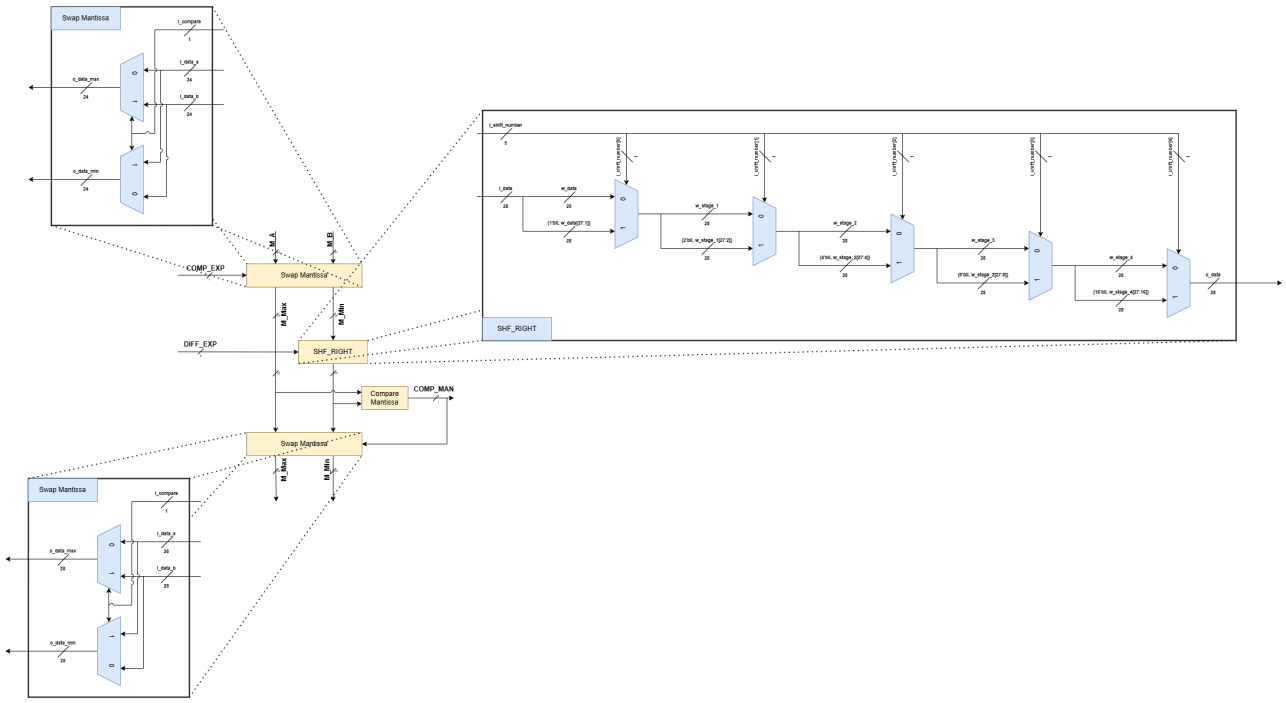


Hình 2.3: Bìa K của bộ so sánh 4-bit (COMP_4bit).

```
    assign o_less = (~i_data_a[3] & ~
i_data_a[2] & ~i_data_a[1] & ~i_data_a[0]
 & i_data_b[0]) | (~i_data_a[3] & ~
i_data_a[2] & ~i_data_a[1] & i_data_b[1])
 | (~i_data_a[3] & ~i_data_a[2] &
i_data_b[2]) | (~i_data_a[3] & i_data_b
[3]) | (~i_data_a[3] & ~i_data_a[2] & ~
i_data_a[0] & i_data_b[1] & i_data_b[0])
| (~i_data_a[3] & ~i_data_a[1] & ~
i_data_a[0] & i_data_b[2] & i_data_b[0])
| (~i_data_a[3] & ~i_data_a[1] & i_data_b
[2] & i_data_b[1]) | (~i_data_a[3] & ~
i_data_a[0] & i_data_b[2] & i_data_b[1] &
 i_data_b[0]) | (~i_data_a[2] & ~i_data_a
[1] & ~i_data_a[0] & i_data_b[3] &
i_data_b[0]) | (~i_data_a[2] & ~i_data_a
[1] & i_data_b[3] & i_data_b[1]) | (~
i_data_a[2] & i_data_b[3] & i_data_b[2])
| (~i_data_a[2] & ~i_data_a[0] & i_data_b
[3] & i_data_b[1] & i_data_b[0]) | (~
i_data_a[1] & ~i_data_a[0] & i_data_b[3]
& i_data_b[2] & i_data_b[0]) | (~i_data_a
[1] & i_data_b[3] & i_data_b[2] &
i_data_b[1]) | (~i_data_a[0] & i_data_b
[3] & i_data_b[2] & i_data_b[1] &
i_data_b[0]);

    assign o_equal = ~|(i_data_a ^
i_data_b);
```

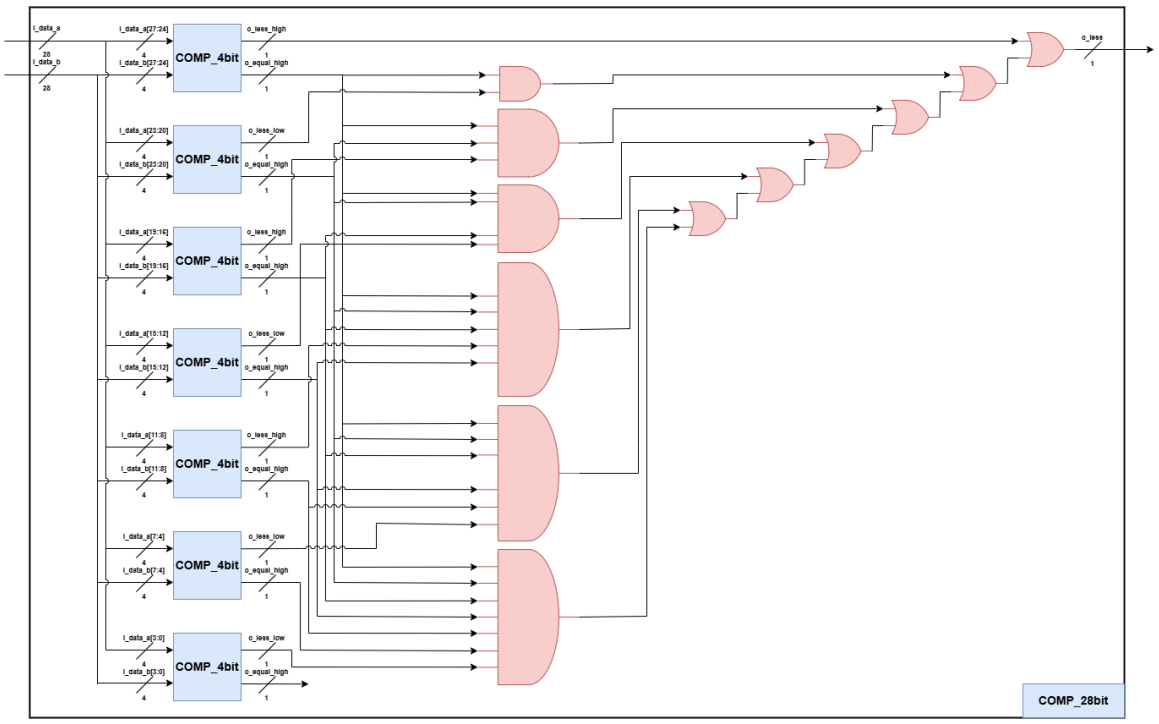
Listing 1: Rút gọn từ Bìa K của bộ so sánh 4-bit (COMP_4bit).

2.2 Bộ tiền xử lý liên quan đến Mantissa



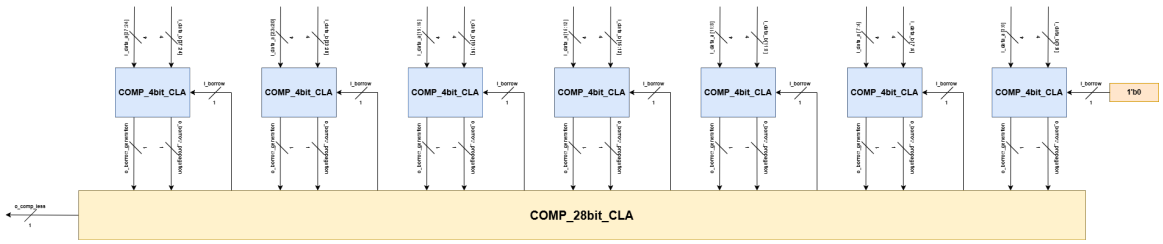
Bộ Compare Mantissa, nhóm em đề xuất hai phương pháp như sau:

Phương pháp 1. Sử dụng phương pháp xây dựng từ bộ COMP_4bit để tạo nên bộ COMP_28bit như sau:



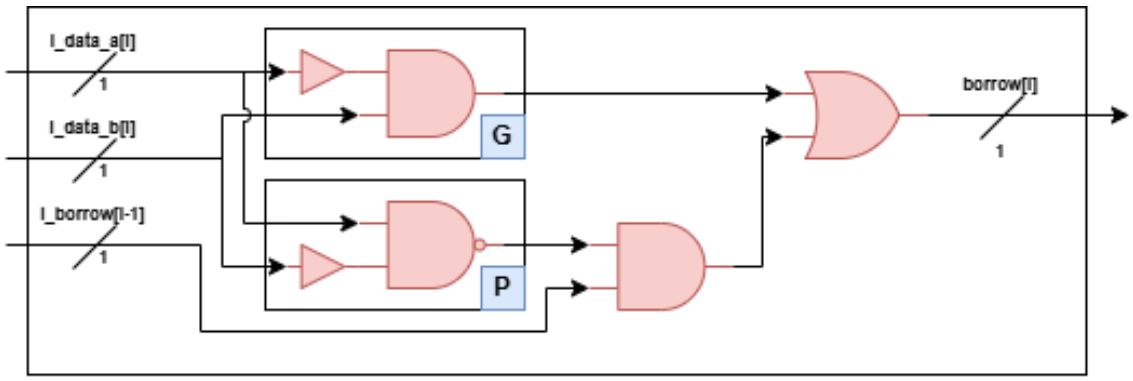
Hình 2.4: Bộ so sánh 28-bit sử dụng các so sánh lan truyền.

Phương pháp 2. Sử dụng giải thuật cộng của bộ CLA (Carry Lookahead Adder) như sau:



Hình 2.5: Bộ so sánh 28-bit sử dụng phép trừ sử dụng giải thuật của bộ CLA.

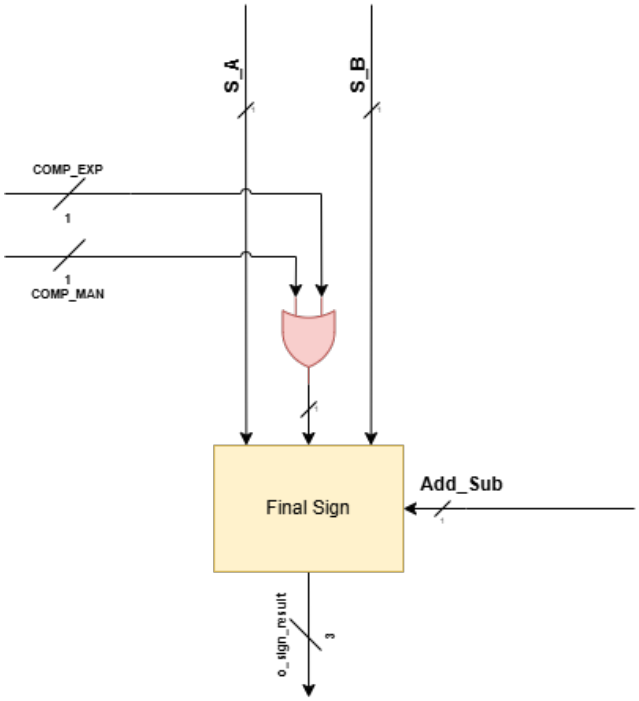
Tuy nhiên với bộ tạo ra tín hiệu `P` và `G` sẽ khác do đây là phép trừ hai bit được biểu dưới dạng như:



Và từ đó sẽ tiền hành như bộ CLA.

Từ hai phương pháp trên, nhóm em quyết định sử dụng bộ 2.4 để có thể dễ dàng triển khai bộ `COMP_28bit`.

2.3 Bộ xử lý dấu của phép tính

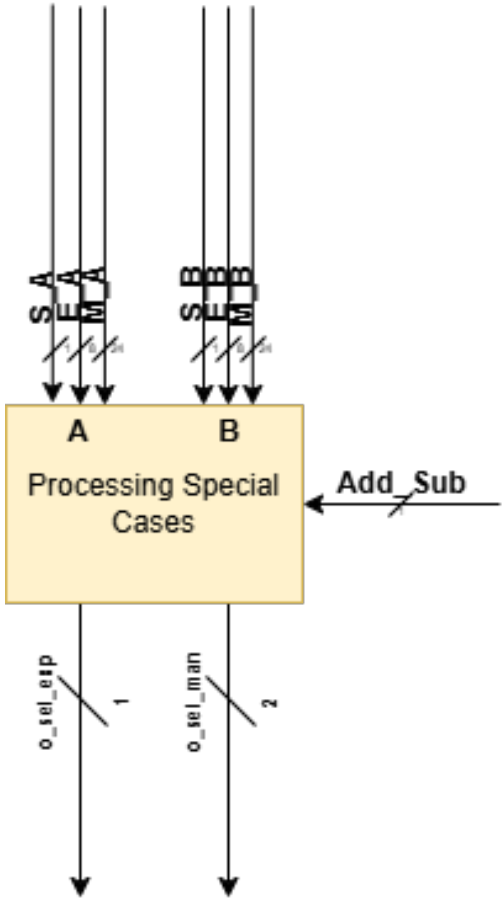


i_add_sub	i_sign_a	i_sign_b	i_comp	o_sign_result	
0	0	0	X	0	$(A+B)>0$
0	0	1	0	0	$(A \geq B) \rightarrow (A-B)>0$
0	0	1	1	1	$(A<B) \rightarrow (A-B)<0$
0	1	0	0	1	$(A \geq B) \rightarrow (-A+B)<0$
0	1	0	1	0	$(A<B) \rightarrow (-A+B)>0$
0	1	1	X	1	$(-A-B)<0$
1	0	0	0	0	$(A \geq B) \rightarrow (A-B)>0$
1	0	0	1	1	$(A<B) \rightarrow (A-B)<0$
1	0	1	X	0	$(A--B)>0$
1	1	0	X	1	$(-A-B)<0$
1	1	1	0	1	$(A \geq B) \rightarrow (-A--B)<0$
1	1	1	1	0	$(A<B) \rightarrow (-A--B)>0$

Bảng 2.1: Bảng sự thật của SIGN_RESULT.

Hình 2.6: Rút gọn bìa K theo bảng sự thật 2.1.

2.4 Bộ xử lý các trường hợp đặc biệt ở ngõ vào



Bảng 2.2: Các giá trị đặc biệt ở ngõ vào.

i_add_sub	i_data_a	i_data_b	o_data_fpu
0	$+\infty$	Number	$+\infty$
0	$+\infty$	$-\infty$	NaN
0	$+\infty$	$+\infty$	$+\infty$
0	$-\infty$	Number	$-\infty$
0	$-\infty$	$-\infty$	$-\infty$

Continued on next page ...

Bảng 2.2 – continued from previous page

i_add_sub	i_data_a	i_data_b	o_data_fpu
0	$-\infty$	$+\infty$	NaN
0	NaN	X	NaN
0	X	NaN	NaN
1	$+\infty$	Number	$+\infty$
1	Number	$+\infty$	$-\infty$
1	$+\infty$	$-\infty$	$+\infty$
1	$+\infty$	$+\infty$	NaN
1	$-\infty$	Number	$-\infty$
1	Number	$-\infty$	$+\infty$
1	$-\infty$	$-\infty$	NaN
1	$-\infty$	$+\infty$	$-\infty$
1	NaN	X	NaN
1	X	NaN	NaN

Với các giá trị đặc biệt ở ngõ vào có đặc tính đặc biệt như sau:

- Đối với các số liên quan đến $\pm\infty$ thì có
  - Exponent: $E = 8'hFF$
  - Mantissa: $M = 24'h00$
- Đối với các số liên quan đến NaN thì có
  - Exponent: $E = 8'hFF$
  - Mantissa: $M \neq 24'h00$

Từ đó ta tạo ra hai giá trị:

+ E = $\&i\_Exponent$

+ M = $\sim|(i\_Mantissa)$

Bảng 2.3: Bảng sự thật của bộ phát hiện các giá trị đặc biệt.

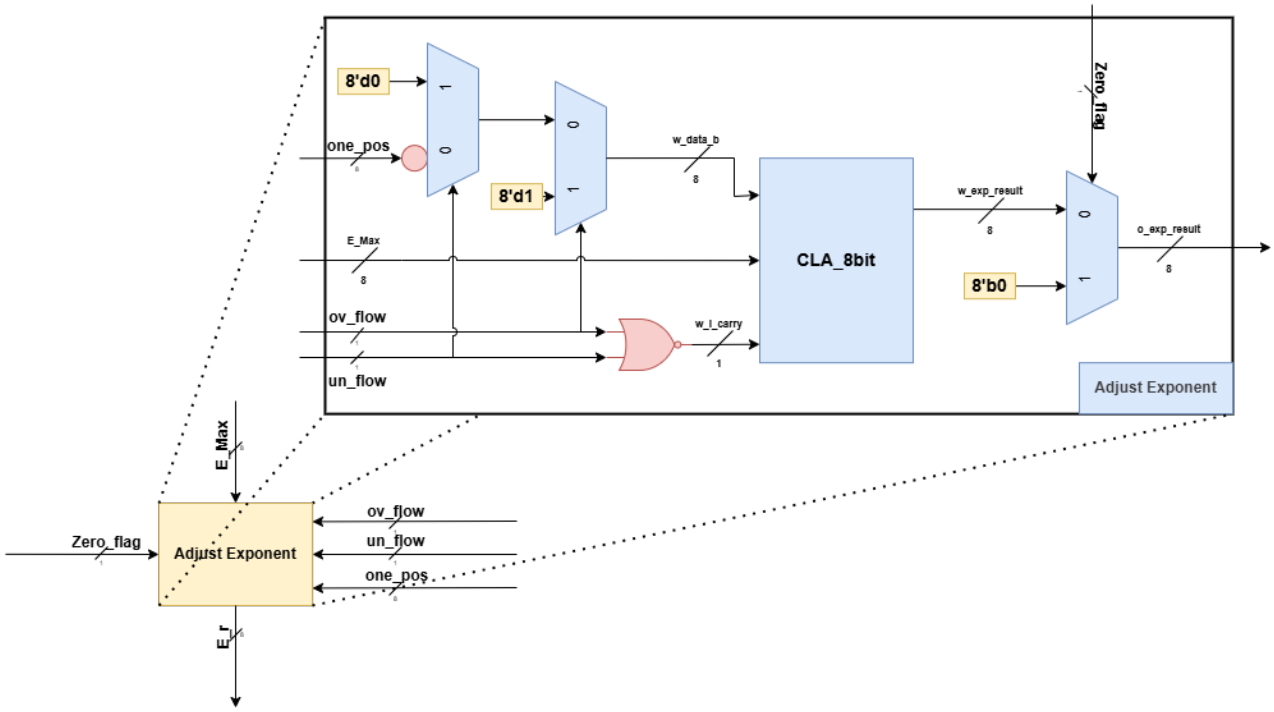
i_add_sub	A			B			S				
	S	E	M	S	E	M	S	E	M	S_M_1	S_M_0
0	0	1	1	X	0	X	0	1	1	1	0
0	0	1	1	1	1	1	0	1	0	1	1
0	0	1	1	0	1	1	0	1	1	1	0
0	1	1	1	X	0	X	1	1	1	1	0
0	1	1	1	1	1	1	1	1	1	1	0
0	1	1	1	0	1	1	1	1	0	1	1
0	X	1	0	X	X	X	X	1	0	1	1
0	X	X	X	X	1	0	X	1	0	1	1
1	0	1	1	X	0	X	0	1	1	1	0
1	X	0	X	0	1	1	1	1	1	1	0
1	0	1	1	1	1	1	0	1	1	1	0
1	0	1	1	0	1	1	X	1	0	1	1
1	1	1	1	X	0	X	1	1	1	1	0
1	X	0	X	1	1	1	0	1	1	1	0
1	1	1	1	1	1	1	X	1	0	1	1
1	1	1	1	0	1	1	1	1	1	1	0
1	X	1	0	X	X	X	X	1	0	1	1

Continued on next page ...

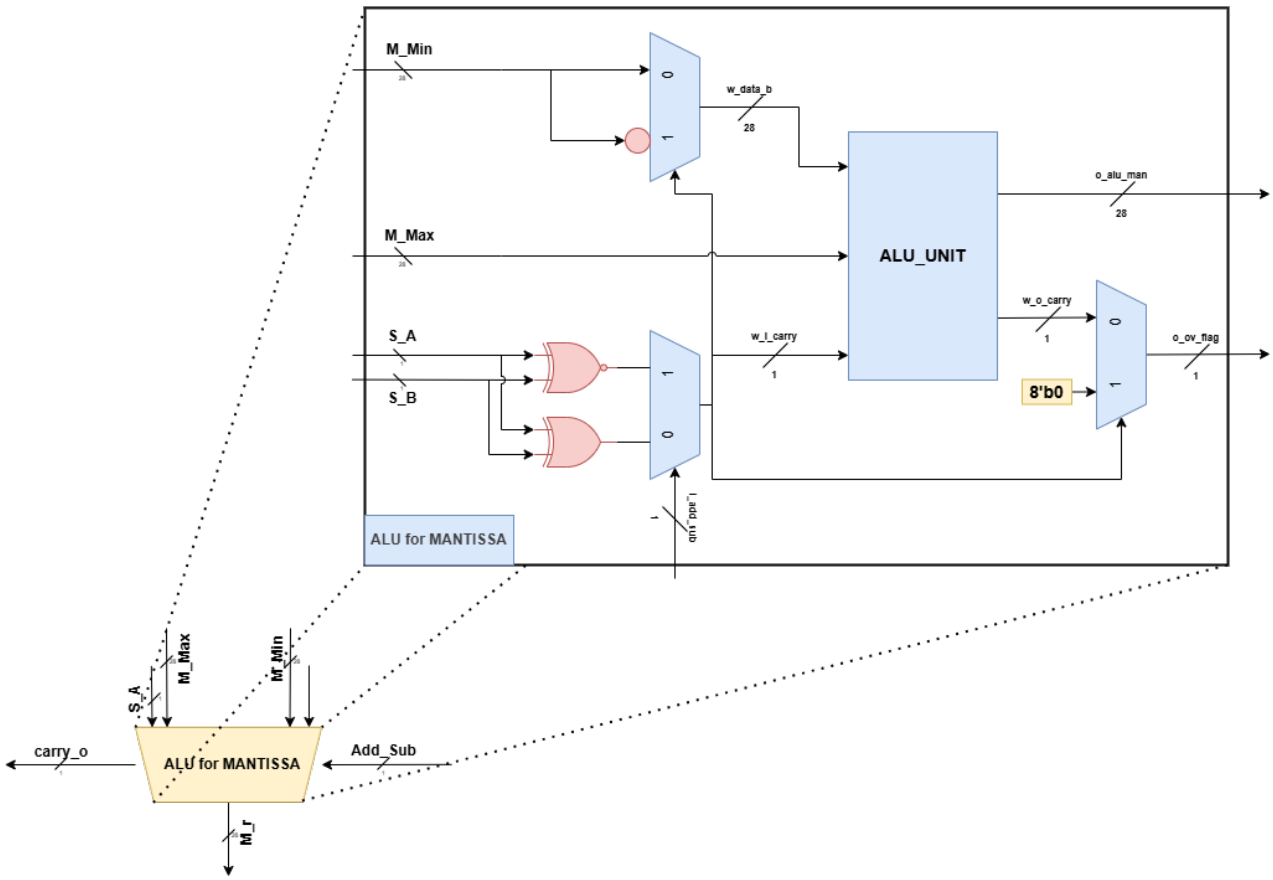
Bảng 2.3 – continued from previous page

1	X	X	X	X	1	0	X	1	0	1	1

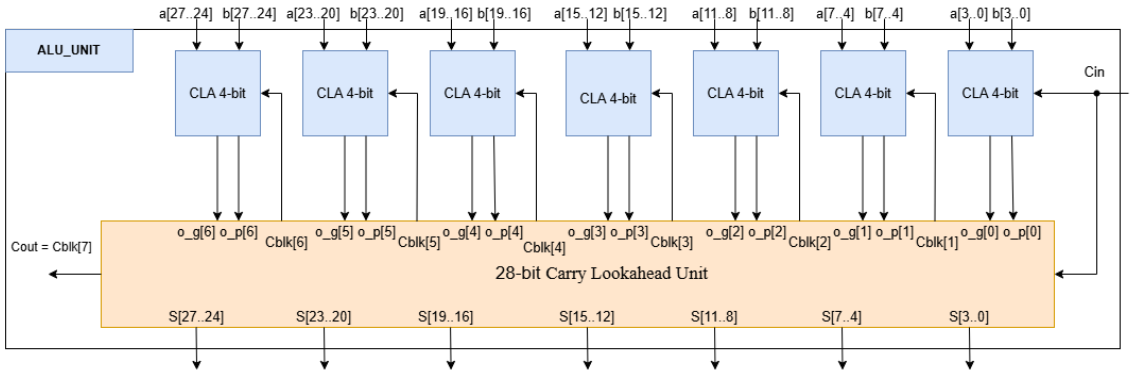
2.5 Bộ căn chỉnh Exponent



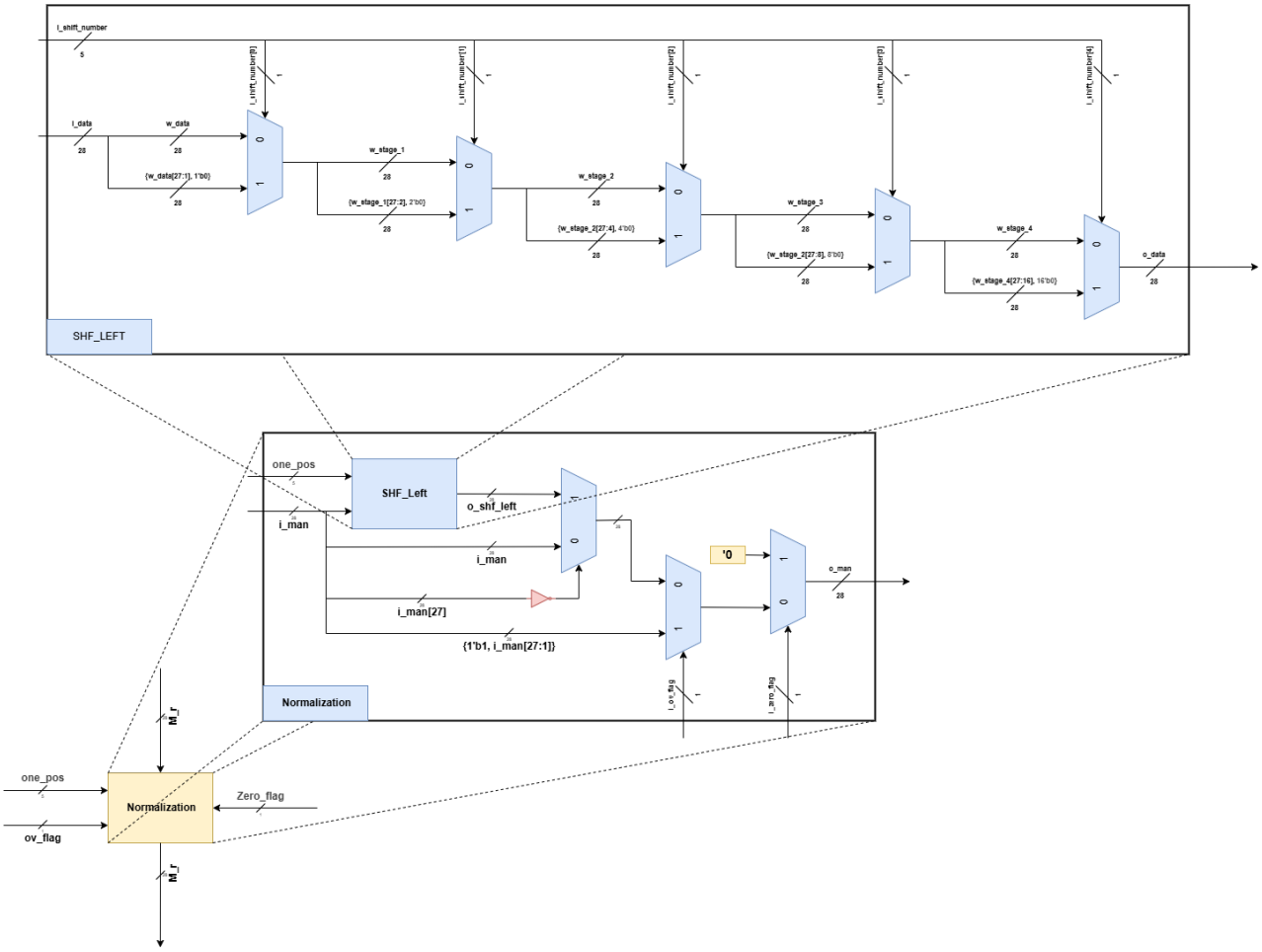
2.6 Bộ cộng trừ Mantissa



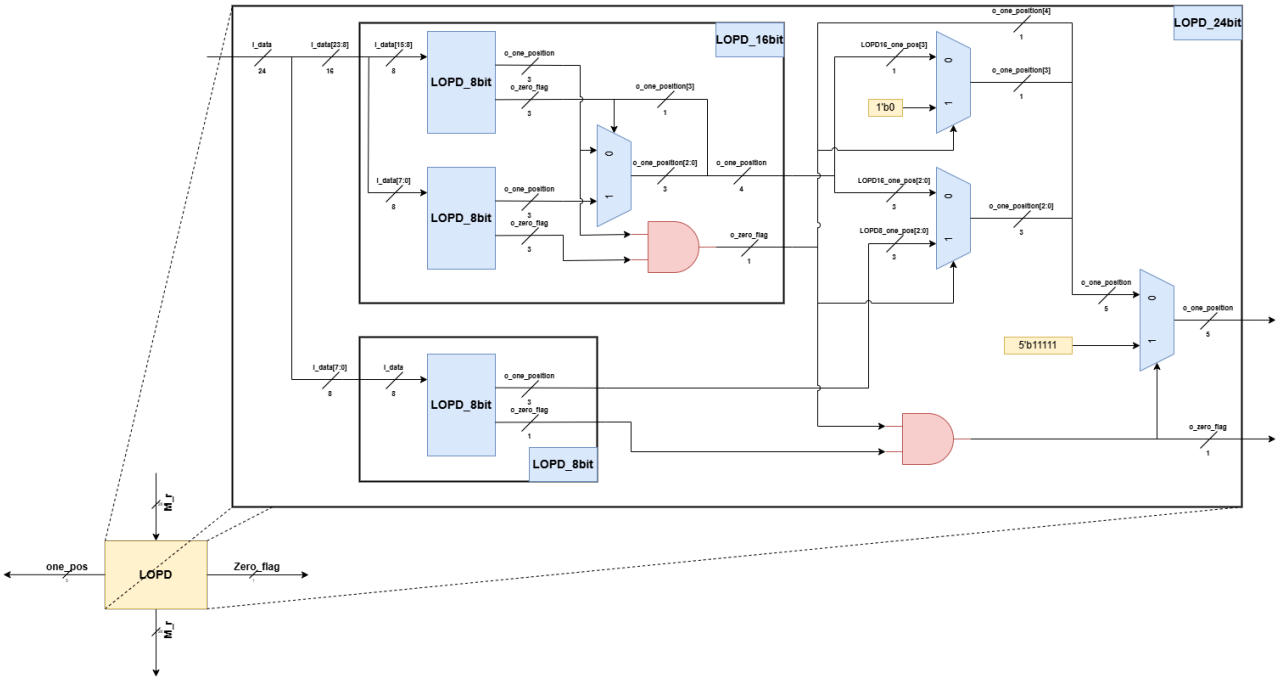
Tại bộ ALU_UNIT, nhóm em sử dụng một bộ cộng với 28-bit



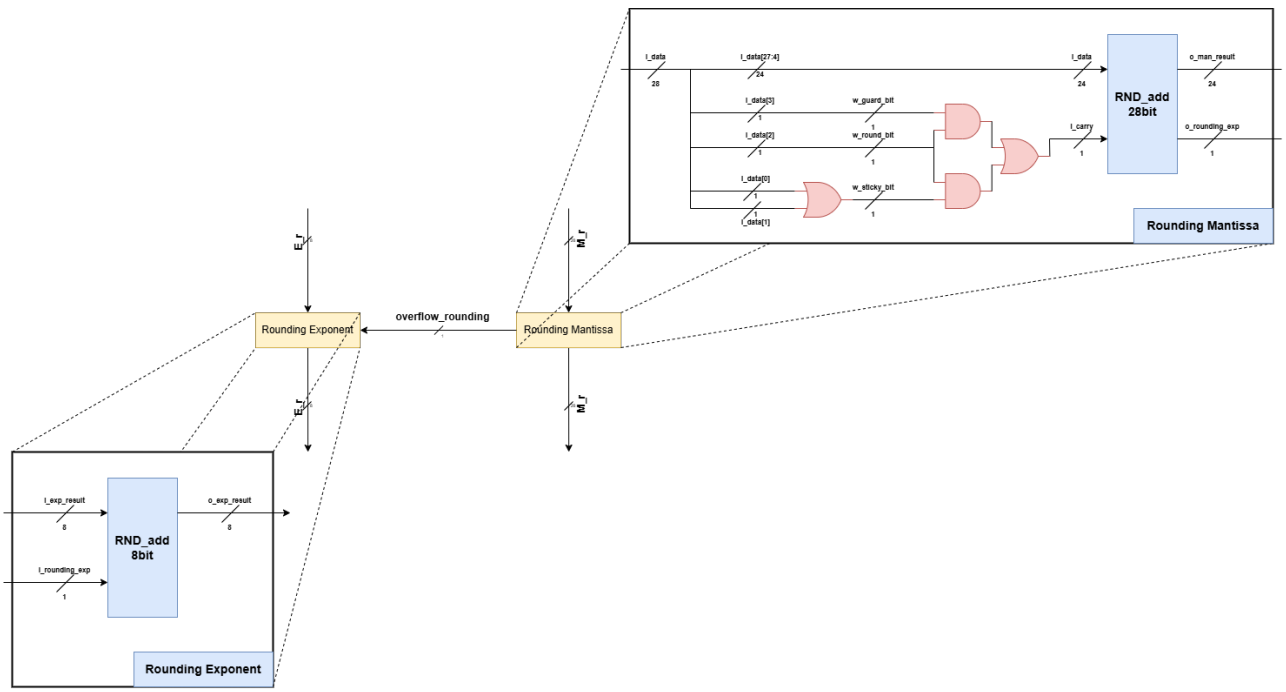
2.7 Bộ chuẩn hóa Mantissa



2.8 Bộ tìm vị trí bit 1 của MANTISSA sau khi chuẩn hóa



2.9 Bộ làm tròn cho giá trị Mantissa và Exponent



Ở đây bộ `RND_unit` là bộ cộng giá trị làm tròn cho dữ liệu đầu vào, với kiểu làm tròn là làm tròn gần nhất. Vì vậy, ở đây ta thấy ở đây chỉ lấy i_data + i_carry bởi vì chỉ là cộng thêm 1 hoặc không.

```
1       p = a ^ b;
2       g = a & b;
3       c[i] = g[i] | (p[i] & c[i-1])
4
```

⇒

```
1       p = a;
2       c[i] = g[i] | (p[i] & c[i-1])
3
4       s  = p ^ c
5       c_o   = &a & c[0]
6
```

Chapter 3. Simulation and Verification

Để thực hiện kiểm định thiết kế, nhóm em sử dụng kiểu dữ liệu `shortreal` của SystemVerilog để thực hiện làm một golden module như sau:

```
function automatic shortreal Cal_FPU_expected(
   input logic         f_i_add_sub ,
   input shortreal     f_i_32_a    ,
   input shortreal     f_i_32_b
);
   begin
      return (f_i_add_sub) ? f_i_32_a - f_i_32_b : f_i_32_a + f_i_32_b;
   end
endfunction
```

Listing 2: Hàm golden model để thực hiện test.

Nhóm em sử dụng các test theo sai số, với sai số nhóm em chọn là:

$$\text{Error\_standard} = \frac{2.0^{-23}}{1.0} \times 100 \approx 11.9209\,\mu\%$$

Có nghĩa là sai ở bit LSB của Mantissa.

Tiếp theo, ta thực hiện xây dựng hàm tính sai số giữa kết quả của golden model và kết quả từ bộ DUT,

```
function automatic shortreal Error_actual(
   input shortreal         f_i_32_s    ,
   input shortreal         f_i_32_e
);
   logic [31:0] f_t_32_s, f_t_32_e;
   logic is_E_one_S, is_E_one_E;
   logic is_M_zero_S, is_M_zero_E;
   logic is_INF_S, is_INF_E;
   logic is_NAN_S, is_NAN_E;

   begin
      f_t_32_s    = REAL_TO_HEX(f_i_32_s);
      f_t_32_e    = REAL_TO_HEX(f_i_32_e);
      is_E_one_S  = &f_t_32_s[30:23];
      is_M_zero_S = ~|f_t_32_s[22:0];
      is_INF_S    = is_E_one_S & is_M_zero_S;
      is_NAN_S    = is_E_one_S & ~is_M_zero_S;
      is_E_one_E  = &f_t_32_e[30:23];
      is_M_zero_E = ~|f_t_32_e[22:0];
      is_INF_E    = is_E_one_E & is_M_zero_E;
      is_NAN_E    = is_E_one_E & ~is_M_zero_E;

      if(is_NAN_S) begin
         Error_actual = (is_NAN_E) ? 0.0 : 100.0;
      end else if(is_NAN_E) begin
         Error_actual = (is_NAN_S) ? 0.0 : 100.0;
      end else if(is_INF_S || is_INF_E) begin
         Error_actual = (f_t_32_s == f_t_32_e) ? 0.0 : 100;
      end else if((f_i_32_e == 0.0) || (f_i_32_e == -0.0)) begin
```

```
            Error_actual = ((f_i_32_s == 0.0) || (f_i_32_s == -0.0)) ? 0.0 : 100.0;
        end else begin
            Error_actual = ((ABS_value(f_i_32_e - f_i_32_s)) / ABS_value(f_i_32_e)) * 100.0;
        end
    end
endfunction
```

Listing 3: Hàm tính sai số giữa kết quả của DUT và kết quả của golden model để thực hiện test.

Với xử lý các trường hợp như sau:

- Nếu kết quả của DUT là một NaN-(Not A Number), hoặc kết quả của bộ golden model là NaN, và khi kiểm tra trường hợp có xuất hiện NaN thì không quan tâm đến dấu của phép tính, vậy ta có kết quả sẽ kiểm tra như sau:
  - Nếu cả hai đều là NaN thì Error_actual = 0.0%
  - Còn nếu một kết quả là NaN còn kết quả còn lại không phải là NaN thì Error_actual = 100.0%
- Nếu kết quả của DUT là một $\pm\infty$, hoặc kết quả của bộ golden model là $\pm\infty$, thì kiểm tra kết quả của bộ DUT và golden model có giống nhau không:
  - Nếu cả hai giống nhau thì Error_actual = 0.0%
  - Còn nếu hai kết qua không giống nhau thì Error_actual = 100.0%
- Còn các trường hợp còn lại thì thực hiện việc tính sai số như sau:

$$\text{Error\_actual} = \frac{|f\_i\_32\_e - f\_i\_32\_s|}{|f\_i\_32\_e|} \times 100.0$$

Từ đó ta có hàm test chính như sau:

```
initial begin
i_rst_n = 0;
i_add_sub       = 1'b0;
i_32_a          = 32'h0;
i_32_b          = 32'h0;
w_i_addr        = '0;
#100;
i_rst_n = 1;
#100;
TestCase_Display_result("ZERO", "(0.0 & 0.0)", 32'h00000000, 32'h00000000);
TestCase_Display_result("ZERO", "(0.0 & -0.0)", 32'h00000000, 32'h80000000);
TestCase_Display_result("ZERO", "(0.0 & -0.0)", 32'h4016A197, 32'h4016A197);
TestCase_Display_result("ZERO", "(0.0 & -0.0)", 32'h40AED834, 32'h40AED834);
TestCase_Display_result("INF", "(inf & inf)", 32'h7f800000, 32'h7f800000);
TestCase_Display_result("INF", "(-inf & -inf)", 32'hff800000, 32'hff800000);
```

```
    TestCase_Display_result("INF", "(inf & -inf)", 32'hff800000, 32'h7f800000);
    TestCase_Display_result("INF", "(inf & 0)", 32'h7f800000, 32'h00000000);
    TestCase_Display_result("INF", "(-inf & 0)", 32'hff800000, 32'h00000000);
    TestCase_Display_result("INF", "(inf & Number)", 32'h7f800000, 32'h40533333);
    TestCase_Display_result("INF", "(-inf & Number)", 32'hff800000, 32'h40533333);
    TestCase_Display_result("INF", "(inf & -Number)", 32'h7f800000, 32'hc00ccccd);
    TestCase_Display_result("INF", "(-inf & -Number)", 32'hff800000, 32'hc00ccccd);
    TestCase_Display_result("NaN", "(NaN & -Number)", 32'h7f800001, 32'hc00ccccd);
    TestCase_Display_result("NaN", "(-NaN & -Number)", 32'hff800001, 32'hc00ccccd);
    TestCase_Display_result("NaN", "(NaN &  Number)", 32'hff800001, 32'h40533333);
    TestCase_Display_result("NaN", "(-NaN &  Number)", 32'h7f800001, 32'h40533333);
    TestCase_Display_result("APPRO", "APPR INF", 32'h7f21616f, 32'h007fffff);
    TestCase_Display_result("APPRO", "APPR INF", 32'h7f7fffff, 32'h00ffffff);
    TestCase_Display_result("APPRO", "APPR INF", 32'h7f7fffff, 32'h007fffff);
    TestCase_Display_result("APPRO", "APPR ZERO", 32'h00ffffff, 32'h007fffff);
    TestCase_Display_result("APPRO", "APPR ZERO", 32'h00ffffff, 32'h00ffffff);
    TestCase_Display_result("SIGN", "(-A + B)", 32'hc00ccccd, 32'h40533333);
    TestCase_Display_result("SIGN", "TEST SIGN", 32'hc00ccccd, 32'hc0533333);
    TestCase_Display_result("SIGN", "TEST SIGN", 32'hc00ccccd, 32'hc1b1999a);
    TestCase_Display_result("PRE_NOR_EXP", "Overflow rouding", 32'h0cffffff, 32'h00f80000);
    repeat(2**SIZE_ADDR) begin
        TestCase_Display_result("Random", "Read data from ROM", w_o_data_rom_a, w_o_data_rom_b);
        @(posedge i_clk);
        #1;
        w_i_addr = w_i_addr + 1;
    end

    Display_SummaryResult(test_count, test_pass);
    #100;
    $finish;
    end
```

Listing 4: Hàm Testbench chính.

Kết quả của Testbench (Compile.log)

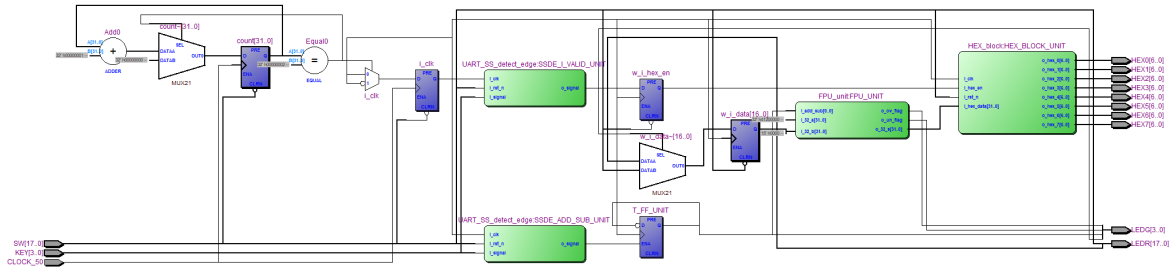
```
==========[ (0.0 & 0.0) ]==========
[ZERO][ADD]i_32_a=00000000 (0.00000000000000000000000) + i_32_b=00000000 (0.00000000000000000000000)  | o_32_s=00000000
    (0.00000000000000000000000)  | o_ov_flow=1, o_un_flow=1
=> PASS: expect=0.00000000000000000000000 (00000000), dut=0.00000000000000000000000 (00000000), rounding_error=0.00000000 % (exp_error = 0.00001192
    %)
[ZERO][ADD]i_32_a=00000000 (0.00000000000000000000000) + i_32_b=00000000 (0.00000000000000000000000)  | o_32_s=00000000
    (0.00000000000000000000000)  | o_ov_flow=1, o_un_flow=1
=> PASS: expect=0.00000000000000000000000 (00000000), dut=0.00000000000000000000000 (00000000), rounding_error=0.00000000 % (exp_error = 0.00001192
    %)
[ZERO][SUB]i_32_a=00000000 (0.00000000000000000000000) - i_32_b=00000000 (0.00000000000000000000000)  | o_32_s=00000000
    (0.00000000000000000000000)  | o_ov_flow=0, o_un_flow=1
=> PASS: expect=0.00000000000000000000000 (00000000), dut=0.00000000000000000000000 (00000000), rounding_error=0.00000000 % (exp_error = 0.00001192
    %)
[ZERO][SUB]i_32_a=00000000 (0.00000000000000000000000) - i_32_b=00000000 (0.00000000000000000000000)  | o_32_s=00000000
    (0.00000000000000000000000)  | o_ov_flow=0, o_un_flow=1
=> PASS: expect=0.00000000000000000000000 (00000000), dut=0.00000000000000000000000 (00000000), rounding_error=0.00000000 % (exp_error = 0.00001192
    %)
...
==========[ (inf & inf) ]==========
[INF][ADD]i_32_a=7f800000 (inf) + i_32_b=7f800000 (inf)  | o_32_s=7f800000 (inf)  | o_ov_flow=1, o_un_flow=1
=> PASS: expect=inf (7f800000), dut=inf (7f800000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[INF][ADD]i_32_a=7f800000 (inf) + i_32_b=7f800000 (inf)  | o_32_s=7f800000 (inf)  | o_ov_flow=1, o_un_flow=1
=> PASS: expect=inf (7f800000), dut=inf (7f800000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[INF][SUB]i_32_a=7f800000 (inf) - i_32_b=7f800000 (inf)  | o_32_s=7fc00000 (nan)  | o_ov_flow=0, o_un_flow=1
=> PASS: expect=-nan (ffc00000), dut=nan (7fc00000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[INF][SUB]i_32_a=7f800000 (inf) - i_32_b=7f800000 (inf)  | o_32_s=7fc00000 (nan)  | o_ov_flow=0, o_un_flow=1
=> PASS: expect=-nan (ffc00000), dut=nan (7fc00000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
...
==========[ (-NaN &  Number) ]==========
[NaN][ADD]i_32_a=7f800001 (nan) + i_32_b=40533333 (3.2999999523162841796875000)  | o_32_s=7fc00000 (nan)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=nan (7fc00001), dut=nan (7fc00000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[NaN][ADD]i_32_a=40533333 (3.2999999523162841796875000) + i_32_b=7f800001 (nan)  | o_32_s=7fc00000 (nan)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=nan (7fc00001), dut=nan (7fc00000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[NaN][SUB]i_32_a=7f800001 (nan) - i_32_b=40533333 (3.2999999523162841796875000)  | o_32_s=7fc00000 (nan)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=nan (7fc00001), dut=nan (7fc00000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[NaN][SUB]i_32_a=40533333 (3.2999999523162841796875000) - i_32_b=7f800001 (nan)  | o_32_s=ffc00000 (-nan)  | o_ov_flow=0, o_un_flow=0
```

```
=> PASS: expect=nan (7fc00001), dut=-nan (ffc00000), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
...
==========[ APPR INF ]==========
[APPRO][ADD]i_32_a=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000) + i_32_b=007fffff (0.000000000000000000000000)   |
    o_32_s=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=21451161146411533915289745299396257382 4.000000000000000000000000 (7f21616f), dut
    =21451161146411533915289745299396257382 4.000000000000000000000000 (7f21616f), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[APPRO][ADD]i_32_a=007fffff (0.000000000000000000000000) + i_32_b=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000)   |
    o_32_s=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=21451161146411533915289745299396257382 4.000000000000000000000000 (7f21616f), dut
    =21451161146411533915289745299396257382 4.000000000000000000000000 (7f21616f), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[APPRO][SUB]i_32_a=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000) - i_32_b=007fffff (0.000000000000000000000000)   |
    o_32_s=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=21451161146411533915289745299396257382 4.000000000000000000000000 (7f21616f), dut
    =21451161146411533915289745299396257382 4.000000000000000000000000 (7f21616f), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
[APPRO][SUB]i_32_a=007fffff (0.000000000000000000000000) - i_32_b=7f21616f (21451161146411533915289745299396257382 4.000000000000000000000000)   |
    o_32_s=ff21616f (-21451161146411533915289745299396257382 4.000000000000000000000000)   | o_ov_flow=0, o_un_flow=0
=> PASS: expect=-21451161146411533915289745299396257382 4.000000000000000000000000 (ff21616f), dut
    =-21451161146411533915289745299396257382 4.000000000000000000000000 (ff21616f), rounding_error=0.00000000 % (exp_error = 0.00001192 %)
==========[ Overflow rouding ]==========
[PRE_NOR_EXP][ADD]i_32_a=0cffffff (0.000000000000000000000000) + i_32_b=00f80000 (0.000000000000000000000000)   | o_32_s=0d000000
    (0.000000000000000000000000)  | o_ov_flow=1, o_un_flow=0
=> PASS: expect=0.000000000000000000000000 (0d000000), dut=0.000000000000000000000000 (0d000000), rounding_error=0.00000000 % (exp_error = 0.00001192
     %)
[PRE_NOR_EXP][ADD]i_32_a=00f80000 (0.000000000000000000000000) + i_32_b=0cffffff (0.000000000000000000000000)   | o_32_s=0d000000
    (0.000000000000000000000000)  | o_ov_flow=1, o_un_flow=0
=> PASS: expect=0.000000000000000000000000 (0d000000), dut=0.000000000000000000000000 (0d000000), rounding_error=0.00000000 % (exp_error = 0.00001192
     %)
[PRE_NOR_EXP][SUB]i_32_a=0cffffff (0.000000000000000000000000) - i_32_b=00f80000 (0.000000000000000000000000)   | o_32_s=0cfffffe
    (0.000000000000000000000000)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=0.000000000000000000000000 (0cfffffe), dut=0.000000000000000000000000 (0cfffffe), rounding_error=0.00000000 % (exp_error = 0.00001192
     %)
[PRE_NOR_EXP][SUB]i_32_a=00f80000 (0.000000000000000000000000) - i_32_b=0cffffff (0.000000000000000000000000)   | o_32_s=8cfffffe
    (-0.000000000000000000000000)   | o_ov_flow=0, o_un_flow=0
=> PASS: expect=-0.000000000000000000000000 (8cfffffe), dut=-0.000000000000000000000000 (8cfffffe), rounding_error=0.00000000 % (exp_error =
    0.00001192 %)
...
==========[ Read data from ROM ]==========
[Random][ADD]i_32_a=d09f76d2 (-21402914816.000000000000000000000000) + i_32_b=501fb38a (10717374464.000000000000000000000000)   | o_32_s=d01f3a1a
    (-10685540352.000000000000000000000000)   | o_ov_flow=0, o_un_flow=1
=> PASS: expect=-10685540352.000000000000000000000000 (d01f3a1a), dut=-10685540352.000000000000000000000000 (d01f3a1a), rounding_error=0.00000000 % (
    exp_error = 0.00001192 %)
[Random][ADD]i_32_a=501fb38a (10717374464.000000000000000000000000) + i_32_b=d09f76d2 (-21402914816.000000000000000000000000)   | o_32_s=d01f3a1a
    (-10685540352.000000000000000000000000)   | o_ov_flow=0, o_un_flow=1
=> PASS: expect=-10685540352.000000000000000000000000 (d01f3a1a), dut=-10685540352.000000000000000000000000 (d01f3a1a), rounding_error=0.00000000 % (
    exp_error = 0.00001192 %)
[Random][SUB]i_32_a=d09f76d2 (-21402914816.000000000000000000000000) - i_32_b=501fb38a (10717374464.000000000000000000000000)   | o_32_s=d0ef5097
    (-32120289280.000000000000000000000000)   | o_ov_flow=0, o_un_flow=0
=> PASS: expect=-32120289280.000000000000000000000000 (d0ef5097), dut=-32120289280.000000000000000000000000 (d0ef5097), rounding_error=0.00000000 % (
    exp_error = 0.00001192 %)
[Random][SUB]i_32_a=501fb38a (10717374464.000000000000000000000000) - i_32_b=d09f76d2 (-21402914816.000000000000000000000000)   | o_32_s=50ef5097
    (32120289280.000000000000000000000000)  | o_ov_flow=0, o_un_flow=0
=> PASS: expect=32120289280.000000000000000000000000 (50ef5097), dut=32120289280.000000000000000000000000 (50ef5097), rounding_error=0.00000000 % (
    exp_error = 0.00001192 %)

==================================
========== TEST SUMMARY ==========
Total test cases:   8296
Passed          :   8296
Failed          :      0
Pass rate       : 100.00%
==================================

Simulation complete via $finish(1) at time 207371 NS + 0
../Topmodule/tb_FPU_unit.sv:234     $finish;
xcelium> exit
TOOL: xrun(64)
```

Listing 5: Kết quả của Testbench.

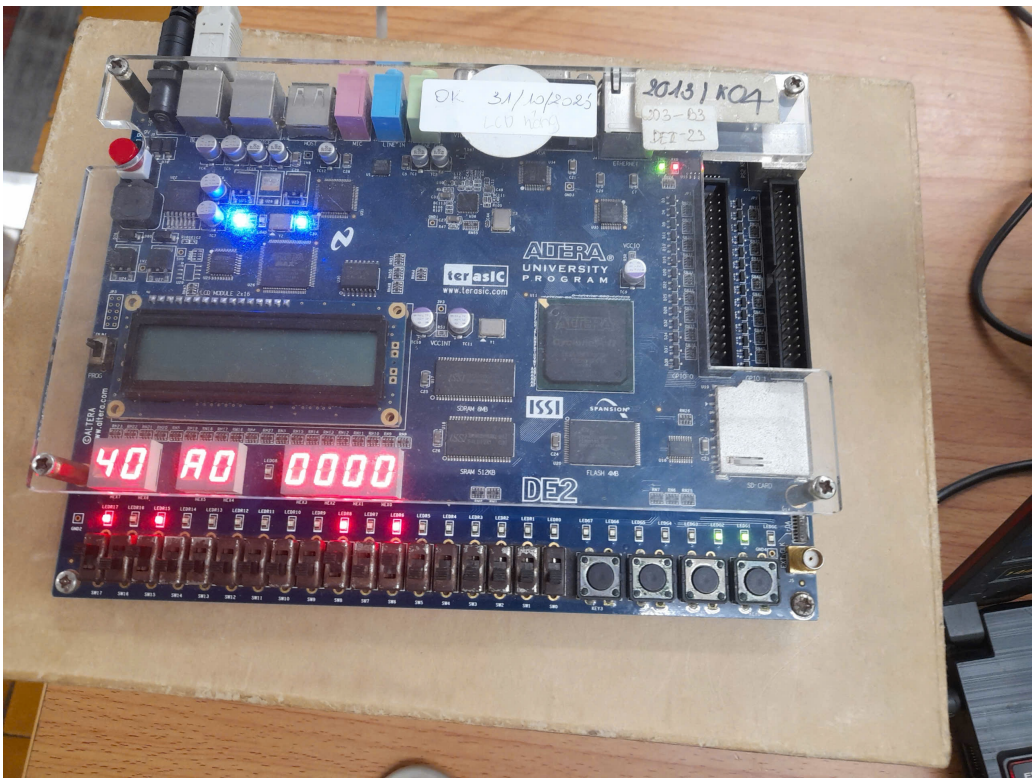
Chapter 4. Implement on FPGA

Sử dụng DE2 để thực hiện kiểm chứng bộ FPU, tổng quan thiết kế của bộ khi thực hiện như sau:



Hình 4.1: RTL Viewer của bộ FPU.

Với việc cố định giá trị `i_32_a` = `32'h41200000` (tương ứng với 10.0). Kết quả sau khi đổ kit như sau:



Hình 4.2: Kết quả khi thực hiện trên kit DE2.

Chapter 5. Synthesis my design

Tiến hành sử dụng file .tcl sau để thực hiện trên Genus:

```
1
2   ################################################
3   #
4   # EE3213_04 (cre: Khai Pham)
5   # Digital system design and Verification
6   #
7   ################################################
8   set Top_module  FPU_unit
9   set LIB_DIR     "./../PDK45nm/gpdk045_lib"
10  set LIB_NAME    "fast_vdd1v0_basicCells_hvt"
11  set LIB_FILE    "${LIB_DIR}/${LIB_NAME}.lib"
12  set SDC_FILE    "./constraint.sdc"
13  set SRC_FILE    "./../../02_rtl/"
14  if {![file exists ${LIB_FILE}]} {
15     error "ERROR: Library file not found: ${LIB_FILE}"
16  }
17  put "TOP_MODULE = $Top_module"
18  put "LIB_DIR = $LIB_DIR"
19  put "LIB_NAME = $LIB_NAME"
20  put "LIB_FILE = $LIB_FILE"
21  put "SDC_FILE = $SDC_FILE"
22  put "SRD_FILE = $SRC_FILE"
23
24  read_libs ${LIB_FILE}
25  read_hdl -sv [glob ${SRC_FILE}*.sv]
26  elaborate
27  set_top_module ${Top_module}
28
29  set elab_v "./${LIB_NAME}/${Top_module}_elab.v"
30  write_hdl > ${elab_v}
31  check_design -all
32
33  read_sdc ${SDC_FILE}
34
35  set_db syn_generic_effort medium
36  set_db syn_map_effort medium
37  set_db syn_opt_effort medium
38
39  syn_generic
40  set generic_v "./${LIB_NAME}/${Top_module}_generic.v"
41  write_hdl > ${generic_v}
42
43  syn_map
44  set techmap_v "./${LIB_NAME}/${Top_module}_tech_map.v"
45  write_hdl > $techmap_v
46
47  remove_assigns_without_optimization -verbose
48  syn_opt
49
50  report timing -lint
51  set netlist_v "./${LIB_NAME}/${Top_module}_netlist.v"
52  write_hdl > ${netlist_v}
53
54  set sdf_out "./${LIB_NAME}/${Top_module}.sdf"
55  write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge -setuphold split  > $sdf_out
56
57  report_timing -max_paths 1 > ./${LIB_NAME}/report/${Top_module}_timing.rpt
58  report hierarchy > ./${LIB_NAME}/report/${Top_module}_hier.rpt
```

```
report gates > ./${LIB_NAME}/report/${Top_module}_gates.rpt
report datapath > ./${LIB_NAME}/report/${Top_module}_datapath.rpt
report_qor > ./${LIB_NAME}/report/${Top_module}_qor.rpt
report_area > ./${LIB_NAME}/report/${Top_module}_area.rpt
report_power > ./${LIB_NAME}/report/${Top_module}_power.rpt

puts "Done. Using library: $LIB_FILE"
puts "Elaborated    -> $elab_v"
puts "Generic       -> $generic_v"
puts "Tech-mapped   -> $techmap_v"
puts "Netlist       -> $netlist_v"
puts "SDF           -> $sdf_out"
puts "Reports       -> ./${LIB_NAME}/report/"

################################################
# Open genus GUI
gui_show
```

Listing 6: genus_synthesis.tcl

Với file **constraint.sdc** như sau:

```
set_max_delay ?.0 -from [all_inputs] -to [all_outputs]
set_input_delay 0.1 -clock [get_clocks *] [all_inputs]
set_output_delay 0.1 -clock [get_clocks *] [all_outputs]
set timing_report_unconstrained true
```

Listing 7: File constraint.sdc

1. Kết quả của **timing.rpt** như sau:

   - **slow_vdd1v2_basicCells_lvt.lib**:

```
Path 1: MET (1638 ps) Path Delay Check
     Startpoint: (R) i_32_b[24]
       Endpoint: (F) o_32_s[30]

                    Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900
                       |
   Required Time:=    7900
       Data Path:-    6262
           Slack:=    1638

Exceptions/Constraints:
  max_delay             7900            constraint.sdc_line_9
```

Hình 5.1

   - **slow_vdd1v2_basicCells_hvt.lib**:

```
Path 1: MET (0 ps) Path Delay Check
     Startpoint: (R) i_32_b[23]
       Endpoint: (F) o_32_s[30]

                   Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900

   Required Time:=    7900
       Data Path:-    7900
           Slack:=       0

Exceptions/Constraints:
  max_delay             7900           constraint.sdc_line_9
```

Hình 5.2

- **slow_vdd1v0_basicCells_lvt.lib**:

```
Path 1: MET (24 ps) Path Delay Check
     Startpoint: (R) i_32_b[23]
       Endpoint: (F) o_32_s[30]

                   Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900

   Required Time:=    7900
       Data Path:-    7876
           Slack:=      24

Exceptions/Constraints:
  max_delay             7900           constraint.sdc_line_9
```

Hình 5.3

- **slow_vdd1v0_basicCells_hvt.lib**:

```
Path 1: MET (0 ps) Path Delay Check
     Startpoint: (F) i_32_b[24]
       Endpoint: (R) o_32_s[9]

                   Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900

    Required Time:=    7900
        Data Path:-    7900
            Slack:=       0

Exceptions/Constraints:
  max_delay             7900            constraint.sdc_line_9
```

Hình 5.4

- **fast_vdd1v2_basicCells_lvt.lib**:

```
Path 1: MET (5390 ps) Path Delay Check
     Startpoint: (R) i_32_b[24]
       Endpoint: (R) o_32_s[30]

                   Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900

    Required Time:=    7900
        Data Path:-    2510
            Slack:=    5390

Exceptions/Constraints:
  max_delay             7900            constraint.sdc_line_9
```

Hình 5.5

- **fast_vdd1v2_basicCells_hvt.lib**:

```
Path 1: MET (4172 ps) Path Delay Check
     Startpoint: (F) i_32_b[28]
       Endpoint: (F) o_32_s[30]

                   Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900

    Required Time:=    7900
        Data Path:-    3728
            Slack:=    4172

Exceptions/Constraints:
  max_delay             7900            constraint.sdc_line_9
```

Hình 5.6

- **fast_vdd1v0_basicCells_lvt.lib**:

```
Path 1: MET (4709 ps) Path Delay Check
     Startpoint: (R) i_32_b[23]
       Endpoint: (F) o_32_s[30]

                   Capture    Launch
      Path Delay:+    7900         -
      Drv Adjust:+       0         0
         Arrival:=    7900

    Required Time:=    7900
        Data Path:-    3191
            Slack:=    4709

Exceptions/Constraints:
  max_delay             7900            constraint.sdc_line_9
```

Hình 5.7

- **fast_vdd1v0_basicCells_hvt.lib**:

```
Path 1: MET (3063 ps) Path Delay Check
     Startpoint: (R) i_32_b[23]
       Endpoint: (F) o_32_s[30]

                    Capture    Launch
      Path Delay:+     7900         -
      Drv Adjust:+        0         0
         Arrival:=     7900

    Required Time:=     7900
        Data Path:-     4837
            Slack:=     3063

Exceptions/Constraints:
  max_delay             7900           constraint.sdc_line_9
```

Hình 5.8

2. Kết quả của Synthesis: