Câu 3: Thiết kế mạch tính tổng của 2 số 32-bit sử dụng giải thuật CLA (Carry Lookahead Adder). Tách thành các bộ cộng 4 bit CLA.

Với các standard cell được sử dụng gồm: cổng not, các cổng logic 2, 3, 4 ngõ vào.

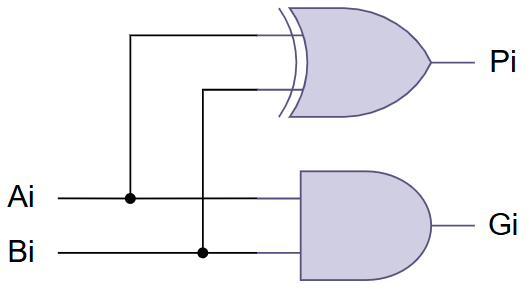
1. **Thiết kế mạch theo phương pháp đã cho và chỉ dùng các standard cell trên**

Theo lý thuyết, ta định nghĩa được 2 tín hiệu quan trọng là:

**+ Generate (Gᵢ)** – sinh carry ngay tại bit đó: .

**+ Propagate (Pᵢ)** – cho phép carry từ bit trước đi qua: .

Dựa vào đó, ta thiết kế PG Generator(Propagate–Generate Generator) như sau:

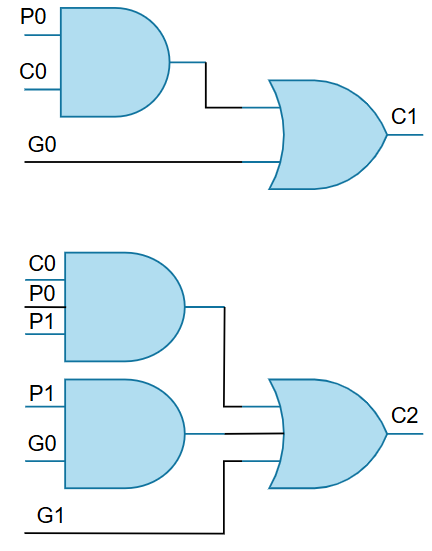


Hình 1. Sơ đồ logic của PG Generator.

Các tín hiệu và chính là đầu vào cho **carry generator** ở cấp kế tiếp. Biểu thức cho carry kế tiếp: . Nếu thay liên tục, được các biểu thức sau:

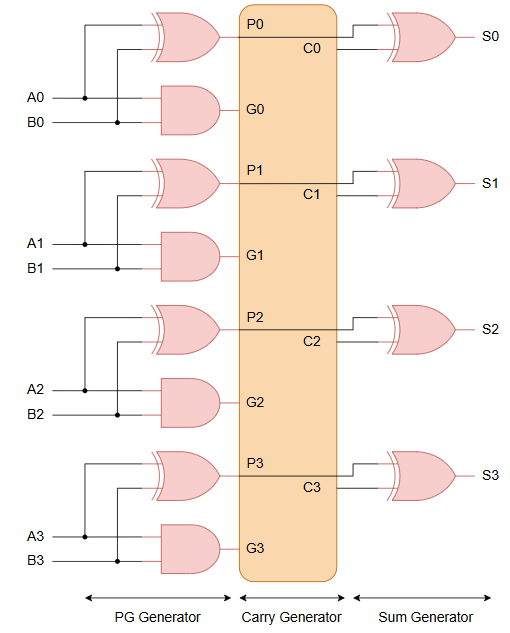
….

Tức là **tất cả các carry có thể tính song song** bằng các phép logic, không cần đợi ripple từng bit. Từ đó ta có thiết kế Carry Generator như sau:



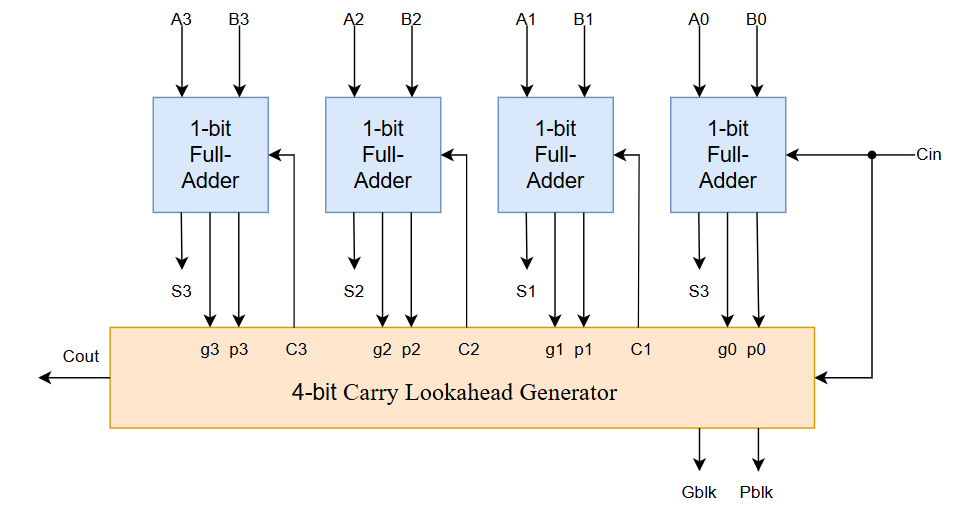
Hình 2. Sơ đồ logic của Carry Generator.

Sau khi biết các biết carry, ta tính tổng: . Từ đó, ta thiết kế bộ cộng CLA 4-bit như sau:

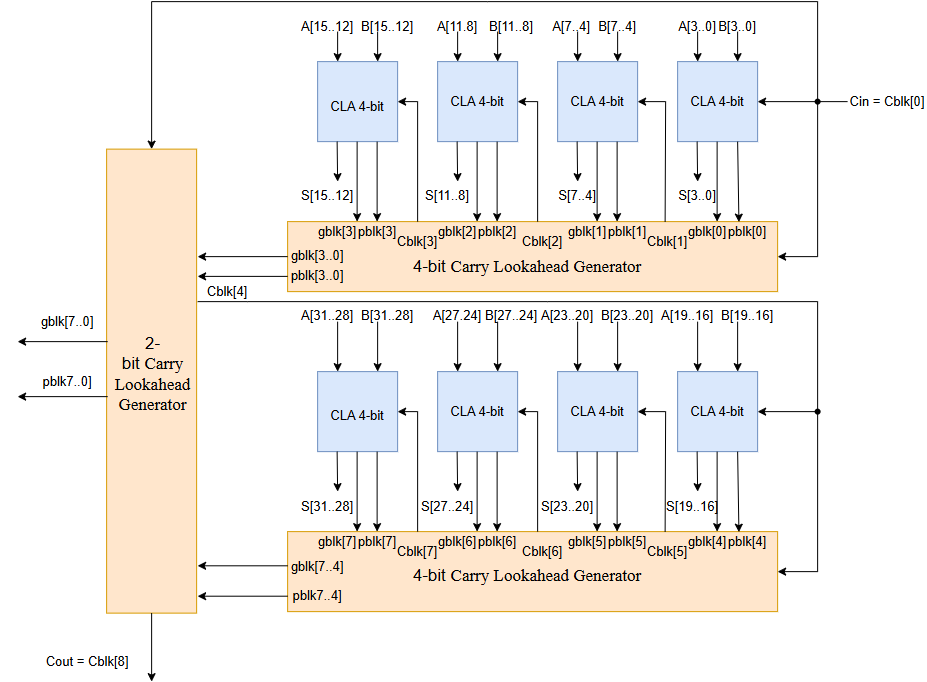


Hình 3. Sơ đồ logic của bộ cộng 4-bit CLA.

Ngoài ra, cũng là sơ đồ logic ấy, ta có thể biểu diễn thiết kế khác như hình bên dưới:



Hình 4. Block Diagram của bộ cộng 4 bit CLA.



Hình 5. Block Diagram của bộ cộng 32-bit CLA.

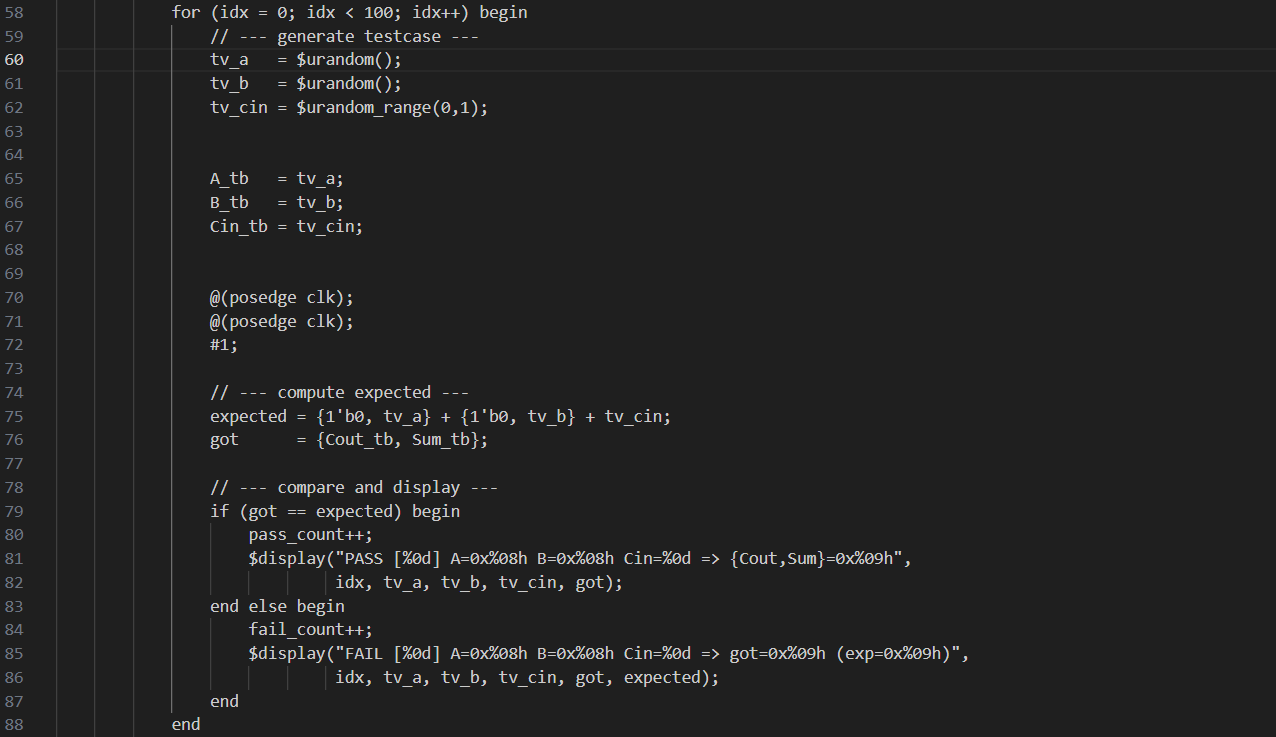
1. **Viết chương trình HDL mô tả mạch đã cho.**

Listing 1. Chương trình mô tả CLA 4-bit.

Listing 2. Chương trình mô tả CLA 32-bit.

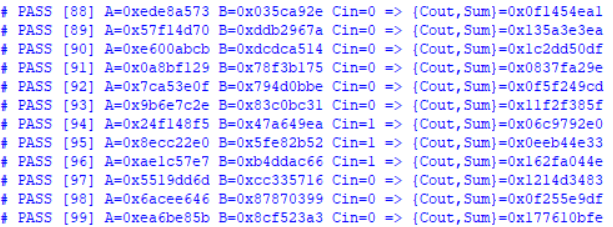
1. **Viết testbench cho mạch. Testbench thực hiện rải 100 mẫu và tính scoreboard của 100 mẫu đó.**

Thực hiện kiểm thử (self-checking) với 100 mẫu random để áp vào DUT. So sánh với expected để cập nhật scoreboard.



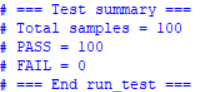
Listing 3. Sinh 100 mẫu random và thực hiện kiểm tra.

Kết quả:



Listing 4. Kết quả test từng mẫu.

Tổng kết sau khi chạy hết 100 mẫu:



Listing 5. Kết quả sau khi chạy hết 100 mẫu.

Câu 5: Thiết kế một ALU thực hiện tính toán theo bảng sau:

|  |  |  |
| --- | --- | --- |
| S1S0 | Cin = 0 | Cin = 1 |
| 00 | F = A + B (add) | F = A + B + 1 |
| 01 | F = A(transfer) | F = A + 1(increment) |
| 10 | F = (complement) | F = + 1 (negate) |
| 11 | F = A + | F = A + + 1(subtract) |

Bảng 1. Bảng tính toán để thiết kế ALU.

Với A và B là 2 số 8 bit. Cùng với các tín hiệu lựa chọn S1, S0 và Cin..

Yêu cầu: Chỉ sử dụng các cổng standard cell : cổng not, các cổng logic 2 ngõ vào, mux 2-1, mux 4-1.

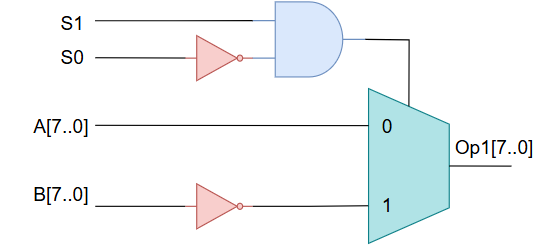
1. **Thiết kế mạch chỉ sử dụng các standard cell trên và chỉ được thiết kế 1 bộ cộng**

Trước tiên, dựa vào Bảng 1, ta có thể rút ra được công thức tổng quát như sau:

F = A + B + Cin

Mà Cin sẽ được điều khiển trực tiếp để đưa vào bộ cộng nên không cần dùng mux. Tóm lại ta chỉ cần quan tâm đến 2 toán hạng còn lại gọi là Op1 và Op2.

Ngoài ra, quan sát thấy tại S1S0 = 10 chỉ có 1 toán hạng là Dùng mux 2-1 để chọn ra trường hợp đặc biệt này, dưới đây là hình ảnh thiết kế.



Hình 6. Mux 2-1 cho toán hạng thứ nhất Op1.

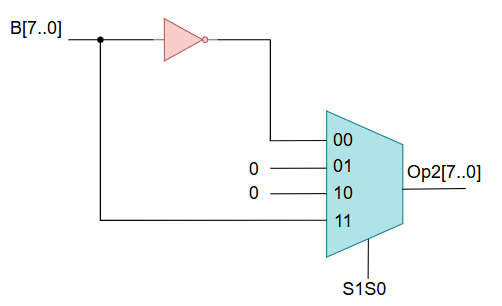
Hình 6 là thiết kế được tóm gọn lại. Thực chất, vì là A và B là số 8-bit nên sẽ là 8 bộ mux 2-1 cho mỗi một bit.

Sau khi có toán hạng thứ nhất, ta tiến đến toán hạng thứ hai dựa vào toán hạng thứ nhất như sau:

|  |  |  |
| --- | --- | --- |
| S1S0 | Toán hạng 1(Op1) | Toán hạng 2(Op2) |
| 00 | A | B |
| 01 | A | 0 |
| 10 |  | 0 |
| 11 | A |  |

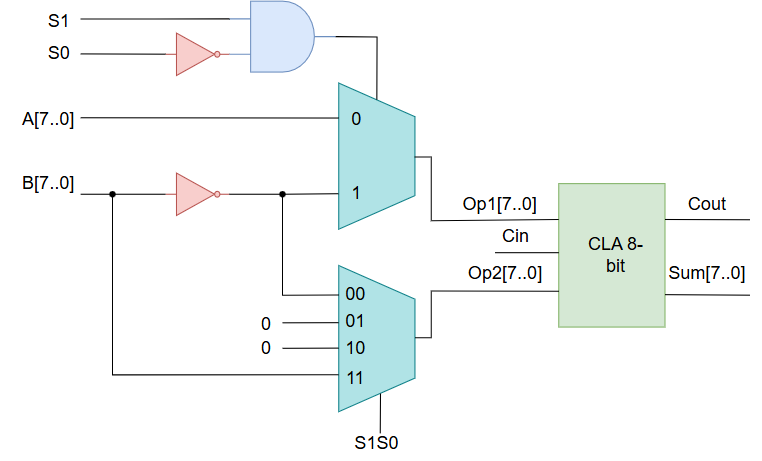
Bảng 2. Bảng lựa chọn toán hạng thứ hai.

Dựa vào bảng 2, nhóm em chọn mux 4-1 để thiết kế cho phù hợp với 4 trường hợp trên. Tương tự, vì là thiết kế cho mỗi bit nên thực chất sẽ có đến 8 bộ mux 4-1.

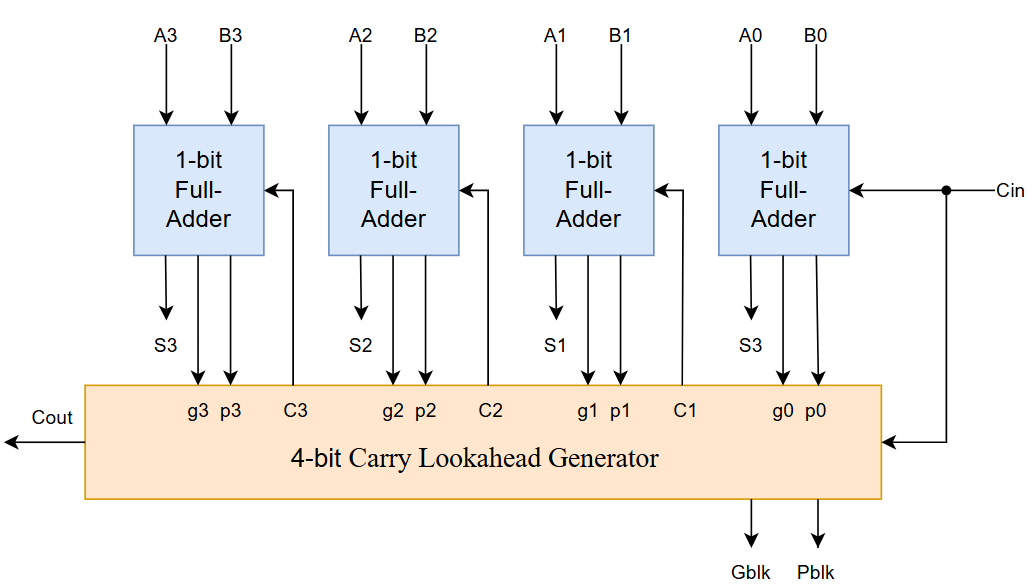


Hình 7. Mux 4-1 cho toán hạng thứ hai (Op2).

Sau khi đã có đầy đủ hai toán hạng, việc còn lại là thiết kế bộ cộng với đầu vào là toán hạng thứ nhất và thứ hai đã chọn cùng với Cin. Tổng quát, bộ ALU sẽ có thiết kế như sau:

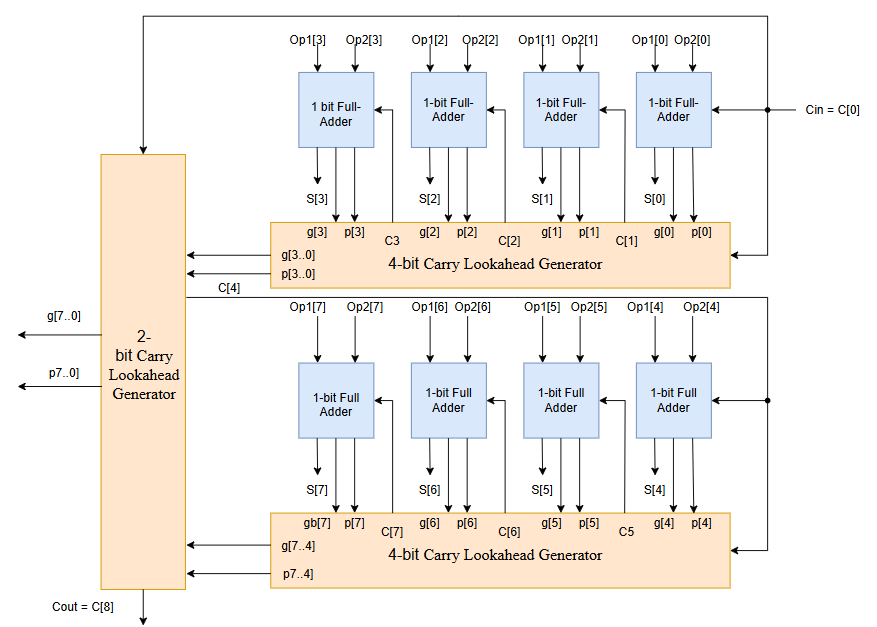
Hình 8. Sơ đồ logic của bộ ALU.

Đối với bộ cộng, nhóm em sử dụng bộ cộng CLA 8-bit. Trước hết, cần phải xây dựng một khối CLA 4-bit như sau.



Hình 9. 4-bit Carry Lookahead Adder.

Dựa vào thiết kế này, ta nối 2 khối CLA 4-bit thành bộ cộng CLA 8-bit, cụ thể như hình dưới:



Hình 10. 8-bit Carry Lookahead Adder.

1. **Viết chương trình HDL mô tả mạch đã cho và viết testbench cho mạch**

Listing 6. Chương trình mô tả CLA 4-bit.

Listing 7. Chương trình mô tả CLA 8-bit.

Listing 8. Chương trình mô tả bộ ALU 8-bit.

**Về testbench:**

Mục tiêu: Xác minh chức năng ALU 8-bit theo bảng S1, S0, Cin; đảm bảo F và Cout của DUT khớp với mô hình tham chiếu trên tập mẫu Directed + Random (100 mẫu tổng).

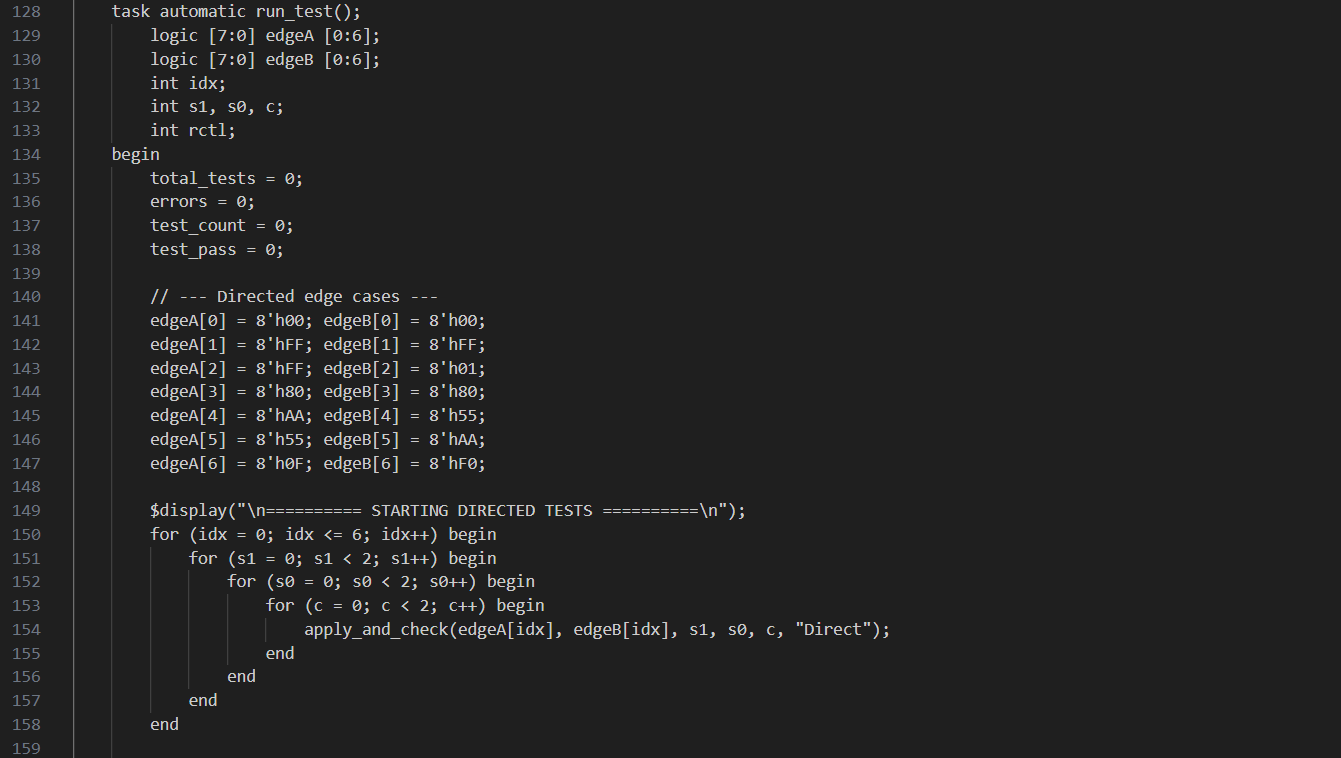
Ở đây, phương pháp nhóm em áp dụng là Functional Verification - Self-checking testbench. Cụ thể hơn, đây là Direct + Random Testing kết hợp Reference Model Comparison.

Nhóm sử dụng Reference Model để tính toán giá trị expected . Sau đó, thực hiện task để so sánh với giá trị expected và thực hiện cập nhật PASS/FAIL. Nhóm thực hiện test 100 trường hợp với những trường hợp và phương pháp test khác nhau.

Thực hiện Directed Test kiểm tra trường hợp đầu vào đặc biệt mà logic dễ sai như sau:

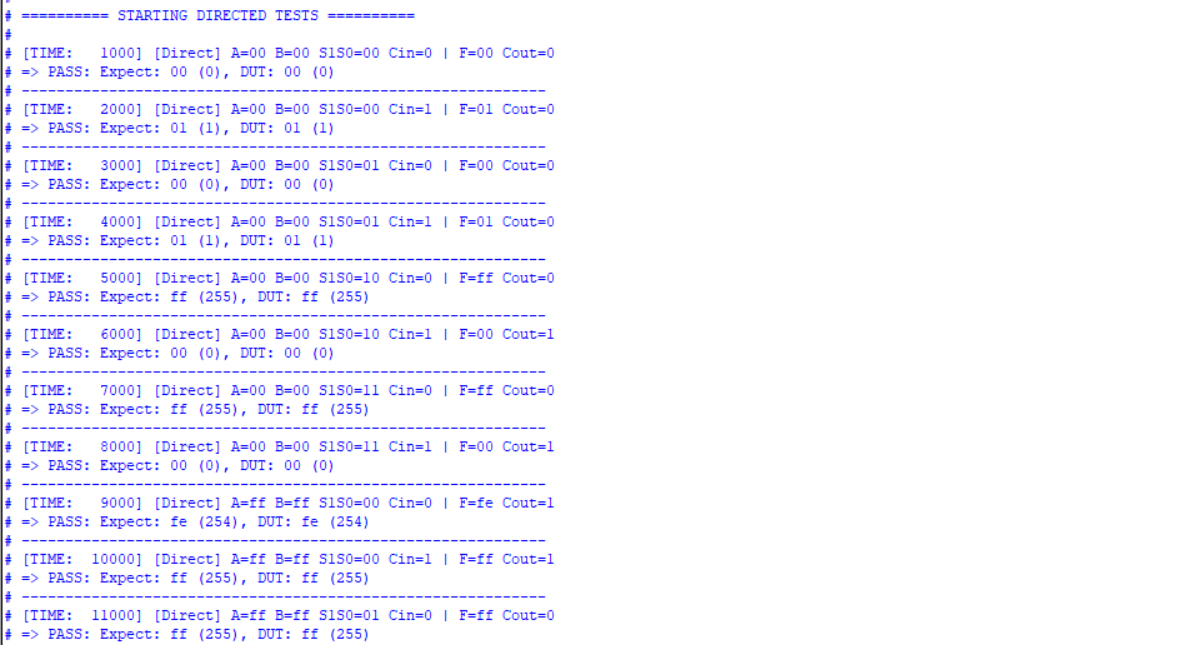
|  |  |  |
| --- | --- | --- |
| Trường hợp | Giá trị test A và B | Tác dụng test |
| 1 | A = 8'h00, B = 8'h00 | Kiểm tra xem ALU có giữ đúng tính cộng cơ bản (0 + 0 = 0) không; đồng thời xác nhận không có bit rác hoặc lỗi sign extension. |
| 2 | A = 8'hFF, B = 8'hFF | Giúp kiểm tra tràn (overflow / carry), vì khi A = B = FFh, phép cộng cho kết quả 0xFE + carry = 1. |
| 3 | A = 8'hFF, B = 8'h01 | Kiểm tra xem khi cộng giá trị lớn nhất với 1 thì Cout có được sinh ra đúng không. Cũng xác nhận mạch carry propagate hoạt động đúng trên bit thấp. |
| 4 | A = 8'h80, B = 8'h80 | Dùng để xem ALU có bị lỗi khi xử lý bit 7 (MSB) hay không, vì 0x80 là 1000\_0000. Nếu là signed operation, đây là vùng âm; còn unsigned thì kiểm overflow. |
| 5 | A = 8'hAA, B = 8'h55 | Kiểm tra carry propagate xen kẽ. Các bit xen kẽ 1–0 giúp xem carry có truyền đúng qua bit liền kề không. Vì nếu mạch cộng sai (ví dụ nối nhầm bit hoặc XOR lỗi), kết quả sẽ lệch rõ. |
| 6 | A = 8'h55, B = 8'hAA | Kiểm tra tính đối xứng của logic cộng (A+B phải giống B+A). |
| 7 | A = 8'h0F, B = 8'hF0 | Test trường hợp chỉ nửa thấp của A có bit 1, và nửa cao của B có bit 1. Kiểm tra carry propagate giữa nibble thấp–cao và tính cộng bit chéo |

Bảng 3. Trường hợp đặc biệt được kiểm tra bằng Directed Test.



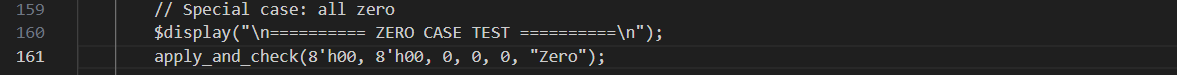
Listing 9. Thực hiện Directed Test.

Kết quả:



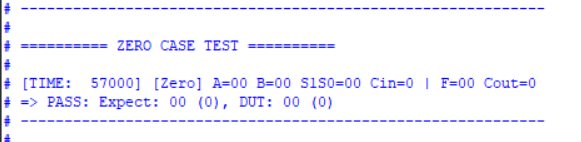
Listing 10. Kết quả mô phỏng Directed Test.

Tiếp theo, là test trường hợp toàn zero. Để xác nhận là mô phỏng ổn, đảm bảo mạch không tạo nhiễu hay glitch.



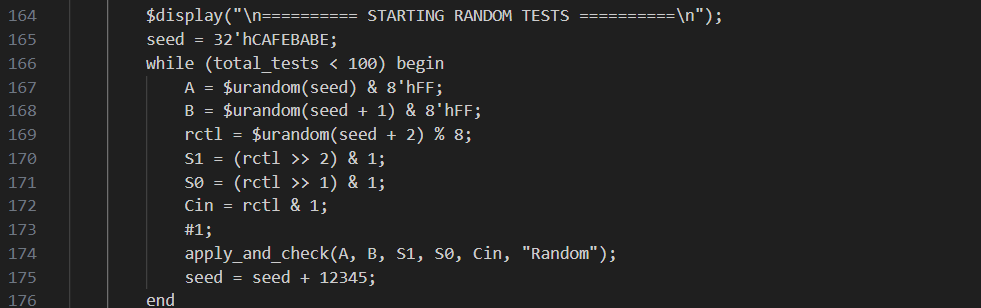
Listing 11. Thực hiện kiểm tra zero case.

Kết quả:



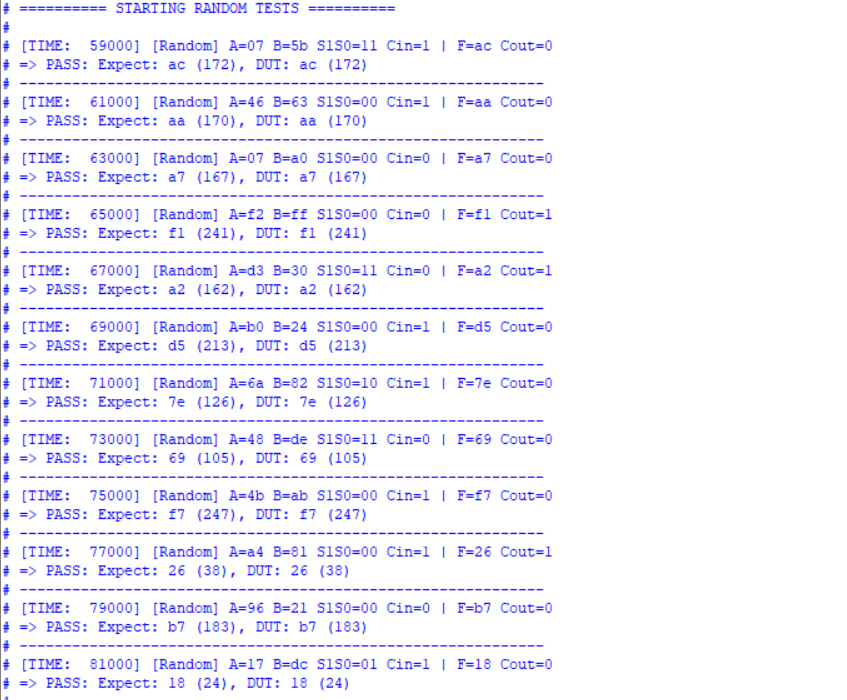
Listing 12. Kết quả mô phỏng zero case.

Cuối cùng là Random Test sau khi chạy Directed test và zero case trước đó: Sinh ngẫu nhiên giá trị A, B, S1, S0, Cin bằng hàm $urandom(seed). Dừng khi đạt 100 mẫu test tổng cộng.



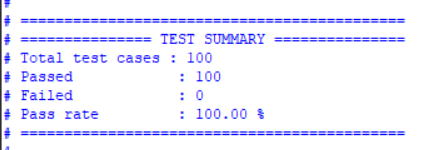
Listing 13. Thực hiện Random Test.

Kết quả:



Listing 14. Kết quả mô phỏng Random Test.

Kết quả tổng kết:



Listing 15. Kết quả sau khi chạy hết 100 mẫu.