**Tổng hợp các phép toán trong Quartus**

Tổng hợp các phép toán trong Quartus có nghĩa là chuyển đổi một mô tả cấp cao của một mạch (được viết bằng một ngôn ngữ như Verilog) thành một netlist cấp thấp của các cổng logic và các thành phần phần cứng khác có thể được triển khai trên một FPGA cụ thể.

* **Các phép toán được Quartus hỗ trợ:** Quartus, thông qua công cụ tổng hợp của nó, hỗ trợ hầu hết các phép toán số học, logic và quan hệ tiêu chuẩn có sẵn trong Verilog. Bao gồm:
  + **Số học:** +, -, \*, /, % (chia lấy dư)
  + **Logic:** &&, ||, !
  + **Bitwise:** &, |, ^, ~, ~^ (hoặc ^~)
  + **Quan hệ:** ==, !=, <, >, <=, >=
  + **Dịch chuyển:** <<, >>
  + **Nối và lặp lại:** { }
* **So sánh Kích thước (LE):** Kích thước của một mạch đã được tổng hợp thường được đo bằng **Logic Elements (LEs)**, là các khối xây dựng cơ bản của một FPGA của Intel. Số lượng LEs mà một mạch chiếm phụ thuộc vào độ phức tạp của phép toán. Các phép toán đơn giản như cộng hoặc AND bitwise sử dụng ít LEs hơn so với các phép toán phức tạp như nhân, thường yêu cầu các khối phần cứng chuyên dụng (khối DSP) hoặc được triển khai bằng cách sử dụng số lượng LEs lớn hơn.

**So sánh các bộ cộng (CRA so với CLA)**

Phần này của câu hỏi tập trung vào việc so sánh các kiến trúc bộ cộng khác nhau, cụ thể là **Bộ cộng truyền carry (CRA)** và **Bộ cộng lookahead carry (CLA)**, khi được triển khai theo cấu trúc.

**Lý thuyết**

* **Bộ cộng truyền carry (CRA):** Đây là loại bộ cộng song song đơn giản nhất. Nó bao gồm một loạt các bộ cộng đầy đủ (full adder) được kết nối thành một chuỗi. Carry-out của một bộ cộng đầy đủ trở thành carry-in của bộ cộng tiếp theo.
  + **Nhược điểm:** Đường truyền quan trọng được xác định bởi "sự truyền" của tín hiệu carry từ bit ít quan trọng nhất (LSB) đến bit quan trọng nhất (MSB). Điều này làm cho nó chậm đối với các bit-width lớn, vì độ trễ tăng tuyến tính với số bit.
* **Bộ cộng lookahead carry (CLA):** Kiến trúc bộ cộng này tăng tốc quá trình cộng bằng cách tính toán các tín hiệu carry song song. Nó sử dụng các tín hiệu "propagate" và "generate" để xác định carry-out của mỗi giai đoạn mà không cần chờ carry truyền qua tất cả các giai đoạn trước đó.
  + **Ưu điểm:** Nhanh hơn nhiều so với CRA, đặc biệt đối với các bit-width lớn hơn. Độ trễ tăng theo logarit với số bit.
  + **Nhược điểm:** Phức tạp hơn và yêu cầu nhiều phần cứng hơn (nhiều LEs hơn) so với CRA để triển khai logic lookahead.

**Mã Verilog**

Dưới đây là một ví dụ về cách bạn có thể mô hình hóa các bộ cộng này theo cấu trúc trong Verilog.

1. Verilog cho một Bộ cộng đầy đủ (FA):

Trước tiên, bạn cần một module full adder, vì nó là khối xây dựng cho cả hai loại.

Verilog

module full\_adder(

input a,

input b,

input cin,

output sum,

output cout

);

assign sum = a ^ b ^ cin;

assign cout = (a & b) | (cin & (a ^ b));

endmodule

2. Verilog cho Bộ cộng truyền carry 4-bit (CRA):

Mã này tạo ra bốn bộ cộng đầy đủ và kết nối chúng trong một chuỗi truyền carry.

Verilog

module cra\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

wire c1, c2, c3;

full\_adder fa0 (.a(a[0]), .b(b[0]), .cin(cin), .sum(sum[0]), .cout(c1));

full\_adder fa1 (.a(a[1]), .b(b[1]), .cin(c1), .sum(sum[1]), .cout(c2));

full\_adder fa2 (.a(a[2]), .b(b[2]), .cin(c2), .sum(sum[2]), .cout(c3));

full\_adder fa3 (.a(a[3]), .b(b[3]), .cin(c3), .sum(sum[3]), .cout(cout));

endmodule

3. Verilog cho Bộ cộng lo5okahead carry 4-bit (CLA):

Mã này cho thấy một mô hình cấu trúc cho một CLA. Nó phức tạp hơn, liên quan đến logic để tạo ra và truyền carry.

Verilog

module cla\_4bit(

input [3:0] a,

input [3:0] b,

input cin,

output [3:0] sum,

output cout

);

// Tín hiệu Propagate và Generate

wire [3:0] p, g;

// Tính toán P và G cho mỗi bit

assign p = a ^ b;

assign g = a & b;

// Tính toán các tín hiệu carry

wire c1, c2, c3;

assign c1 = g[0] | (p[0] & cin);

assign c2 = g[1] | (p[1] & g[0]) | (p[1] & p[0] & cin);

assign c3 = g[2] | (p[2] & g[1]) | (p[2] & p[1] & g[0]) | (p[2] & p[1] & p[0] & cin);

// Tính toán các bit tổng song song

assign sum[0] = p[0] ^ cin;

assign sum[1] = p[1] ^ c1;

assign sum[2] = p[2] ^ c2;

assign sum[3] = p[3] ^ c3;

// Carry-out cuối cùng

assign cout = g[3] | (p[3] & c3);

endmodule

**So sánh CRA và CLA**

* **Kích thước (Size/LEs):** CLA nhìn chung **lớn hơn** CRA vì nó yêu cầu logic tổ hợp phức tạp hơn để triển khai các phương trình lookahead carry. Logic bổ sung này tiêu tốn nhiều Logic Elements (LEs) hơn trên FPGA.
* **Độ trễ (Delay):** CLA **nhanh hơn** đáng kể so với CRA. Độ trễ của CRA tỷ lệ thuận với số bit, O(n), vì tín hiệu carry phải truyền qua từng giai đoạn tuần tự. Độ trễ của CLA là logarit, O(logn), vì các carry được tính toán song song.

Tóm lại, **CRA** đơn giản và có kích thước nhỏ nhưng chậm, trong khi **CLA** nhanh hơn nhưng yêu cầu nhiều tài nguyên phần cứng hơn. Việc lựa chọn giữa chúng phụ thuộc vào yêu cầu hiệu suất và hạn chế tài nguyên của thiết kế.