Chương I: Mở đầu quá trình thiết kế mạch vi điện tử

1.1 Các giai đoạn trong thiết kế các mạch tích hợp

Sự ra đời của các mạch vi điện tử đã làm cơ sở phát triển phần cứng và phần mềm của các hệ thống tính toán trong những thập kỷ gần đây. Việc tăng liên tục mức độ tích hợp của các mạch điện tử trên một nền đơn đã đưa tới việc chế tạo những hệ thống với độ phức tạp ngày càng tăng. Công nghệ chế tạo mạch tích hợp trên cơ sở các chất bán dẫn phát triển vũ bão. Tới giữa những năm 80 của thế kỷ 20 người ta đã có thể chế tạo được những mạch tích hợp chứa tới hàng triệu linh kiện điện tử trên một tinh thể chất bán dẫn. Những mạch được gọi là mạch tích hợp cao (VLSI) hoặc là mạch vi điện tử. Việc ra đời của những mạch vi điện tử đã làm nảy sinh sự cần thiết phải có một phương pháp luận và quy trình thiết kế, chế tạo thích hợp.

Trong công nghiệp, việc chế tạo các mạch tích hợp được thực hiện qua 4 giai đoạn:

- Giai đoạn thiết kế
- Giai đoạn chế tạo
- Giai đoạn kiểm tra
- Giai đoạn đóng gói

Giai đoạn thiết kế: từ các chức năng mà mạch sẽ thực hiện, chúng ta xây dựng mô hình của mạch trên nhiều mức độ chi tiết khác nhau. Các mức độ chi tiết có thể được chia thành mức kiến trúc, mức logic, mức vật lý. Kết quả cuả giai đoạn thiết kế là các mô hình của mạch đã được xác nhận không chứa lỗi trên phương diện thiết kế.

Giai đoạn chế tạo: mạch tích hợp sẽ được chế tạo theo các công nghệ cấy ghép các phần tử mạch lên các tinh thể chất bán dẫn bằng phương pháp mặt nạ che phủ và công nghệ xây dựng các mạch nhiều lớp. Kết quả của giai đoạn này là những vi mạch thực hiện những chức năng như trong thiết kế.

Giai đoạn kiểm tra: Những mạch đã chế tạo sẽ được kiểm nghiệm ngẫu nhiên để khẳng định rằng mạch không chứa lỗi về mặt chế tạo. Trong trường hợp có những lỗi gặp nhiều lần có thể rút ra kết luận lỗi đó có thể là lỗi trong quá rình chế tạo. Dựa vào việc kiểm tra quy trình công nghệ ta có thể rút ra kết luận về các khâu có thể sinh ra lỗi

Giai đoạn cuối cùng là giai đoạn đóng gói. Lúc đó các vi mạch sẽ được phân tách và được tạo vỏ bọc.

Ở đây ta sẽ đi sâu nghiên cứu giai đoạn đầu tiên là giai đoạn thiết kế. Quá trình thiết kế vi mạch điện tử trong công nghiệp được chia làm 3 phân đoạn:

- Mô hình hóa
- Tổng hợp và tối ưu hóa
- Kiểm nghiệm và phê chuẩn

Do đó ta tập trung vào bài toán mô hình hóa mạch và tổng hợp tối ưu hóa mạch.

a. Mô hình hóa:

Nhà thiết kế xây dựng các mô hình cấu trúc mạch và các chức năng mà mạch sẽ thực hiện. Các mô hình mạch là công cụ biểu diễn các ý tưởng thiết kế. Mô hình hóa đóng vai trò quan trọng trong thiết kế mạch vi điện tử bởi vì các mô hình là các phương tiện mang thông tin về các mạch sẽ được xây dựng một cách cô đọng và chính xác. Do đó mô hình cần phải chính xác, chặt chẽ cũng như có mức độ tổng quát, trong suốt và dễ hiểu đối với người thiết kế và máy. Với sự phát triển của các kỹ thuật mô phỏng, mô hình mạch có thể được xây dựng trên cơ sở các ngôn ngữ mô tả phần cứng HDL (hardware description languages). Trong nhiều trường hợp, các mô hình đồ họa như biểu đồ dòng thông tin, sơ đồ mạch và mô tả hình dạng hình học của các đối tượng cũng như cách sắp xếp chúng trên bản mạch đều có thể dùng để biểu diễn mạch. Đối với những mạch có độ tích hợp siêu lớn do độ phức tạp của mạch rất cao nên việc xây dựng mô hình mạch thường theo các mức độ chi tiết khác nhau. Điều đó cho phép người thiết kế tập trung vào từng phần của mô hình tại từng giai đoạn thiết kế.

b. Tổng hợp và tối ưu hóa

Tổng hợp là giai đoạn sáng tạo thứ hai của quá trình thiết kế. Giai đoạn đầu tuân theo các ý tưởng của nhà thiết kế hình thành dần các khái niệm về mạch và xây dựng những mô hình sơ bộ đầu tiên về mạch. Mục đích chính của giai đoạn này là xây dựng mô hình chi tiết của mạch như các chi tiết về dạng hình học phục vụ cho công đoạn lắp ráp và tạo vỏ bọc cho mạch. Điều này đạt được thông qua quá trình xây dựng và chính xác hóa thiết kế từng bước trong đó mô hình trừu tượng ban đầu được người thiết kế chi tiết hóa từng bước lặp đi lặp lại. Khi thực hiện quá trình tổng hợp mạch theo các bước cải tiến mô hình, người thiết kế cần nhiều thông tin liên quan tới các công nghệ chế tạo và các phong cách thiết kế mong muốn. Ta có thể thấy các chức năng của mạch có thể độc lập với các chi tiết thực hiện, trong khi đó các dạng

biểu diễn hình học của mạch hoàn toàn phụ thuộc vào các đặc tính của công nghệ như kích thước của các dây dẫn trong mạch phụ thuộc vào công nghệ chế tạo.

Bài toán tối ưu mạch luôn kết hợp chặt chẽ với bài toán tổng hợp mạch. Quá trình tối ưu đòi hỏi phải lựa chọn những chi tiết xác định của mạch với mục đích làm tăng khả năng của mạch về phương diện thiết kế tương ứng với những độ đo xác định. Vai trò của tối ưu là nâng cao chất lượng của mạch điện như tối ưu về chức năng, về diện tích, về tính dễ kiểm nghiệm và phát hiện lỗi. Chức năng liên quan đến thời gian để thực hiện một quá trình xử lý thông tin cũng như số lượng thông tin có thể được xử lý trong một đơn vị thời gian. Các tính năng của mạch là ảnh hưởng lớn tới khả năng cạnh tranh của mạch trên thị trường. Vấn đề chất lượng của mạch cũng liên quan tới kích thước cũng như diện tích của mạch. Diện tích cũng là đối tượng của tối ưu mạch. Kích thước nhỏ của mạch cho phép có thể phân bố nhiều mạch trên một lớp, điều đó làm giảm giá thành chế tạo và đóng gói. Trong công nghiệp chế tạo chúng ta mang muốn có những thiết kế cho phép phát hiện lỗi và xác định vị trí lỗi của mạch sau khi chế tạo. Khả năng này, trong nhiều trường hợp, ảnh hưởng lớn tới chất lượng của mạch. Một thông số quan trọng trong vấn đề phát hiện lỗi của mạch là phần trăm lỗi có thể được phát hiện đối với một bộ giá trị thử nghiệm. Nói chung người thiết kế mong muốn có những mạch dễ kiểm nghiệm, điều đó làm giảm giá thành chung của quá trình sản xuất.

c. Kiểm nghiệm và phê chuẩn

Quá trình phê chuẩn mạch là việc đạt được ở một mức độ chắc chắn hợp lý rằng mạch điện sẽ làm việc đúng với giả thiết không có lỗi chế tạo. Nhằm loại bỏ mọi lỗi thiết kế có thể có trước khi đưa vào sản xuất. Quá trình phê chuẩn mạch bao gồm việc xây dựng mô hình mô phỏng mạch dựa trên thiết kế và thực hiện kiểm tra. Mô phỏng mạch bao gồm phân tích các diễn biến hành vi của mạch điện theo thời gian đối với một hoặc nhiều bộ giá trị đầu vào. Quá trình mô phỏng có thể áp dụng trên nhiều mức thiết kế khác nhau tùy theo các mức trừu tượng của mô hình.

1.2 Mô hình hoá mạch điện

Mô hình mạch là biểu diễn trừu tượng trong đó chỉ ra những đặc tính thích hợp mà không có những chi tiết tương ứng. Quá trình tổng hợp mạch là quá trình tạo mô hình mạch bắt đầu từ những biểu diễn sơ lược nhất.

Các mô hình được phân loại theo các mức độ mô tả trừu tượng và các góc quan sát.

- Các mức độ mô tả trừu tượng được chia làm ba mức như sau:

- Mức kiến trúc

Mạch điện được thể hiện qua tập hợp các thao tác như các tính toán trên dữ liệu, các phép chuyển đổi và truyền thông tin. Ví dụ, trên mức kiến trúc, mạch có thể được biểu diễn qua những mô hình trên các ngôn ngữ mô tả phần cứng, những biểu đồ luồng thông tin.

- Mức logic

Mạch điện được thể hiện như tập hợp các chức năng logic và được chuyển thành các hàm logic. Ví dụ, trên mức logic mạch có thể được biểu diễn thông qua các biểu đồ chuyển trạng thái, các sơ đồ mạch lôgic

- Mức hình học

Mạch có thể biểu diễn như tập hợp các đối tượng hình học. Ví dụ đơn giản của biểu diễn hình học có thể là các lớp trong mạch nhiều lớp, dáng vẻ bề ngoài và phân bố của các phần tử cấu thành mạch.

- Các góc độ quan sát cũng được chia thành 3 góc độ:
  - Góc độ hành vi: mô tả các chức năng của mạch mà không quan tâm tới việc thực hiện các chức năng đó.
  - Góc độ cấu trúc: mô tả mô hình mạch bằng các thành phần cơ bản của mạch và các liên kết giữa các thành phần đó.
  - Góc độ vật lý: có liên quan tới các đối tượng vật lý xuất hiện trong thiết kế.

    Các mô hình có các mức độ mô tả trừu tượng khác nhau và có thể được quan sát theo những góc độ khác nhau.

    Ví dụ: Ở mức kiến trúc theo góc độ hành vi thì mạch điện là tập hợp các phép toán và sự liên quan giữa chúng với nhau, còn theo góc độ cấu trúc thì mạch là tập hợp các khối cơ sở và các liên kết ghép nối giữa các khối cơ sở đó.

    Nếu xét trường hợp thiết kế các mạch đồng bộ thì với các mô hình trên mức logic, góc độ hành vi có thể là các lưu đồ chuyển trạng thái, còn góc độ cấu trúc là các phần tử logic.

1.3 Tổng hợp và tối ưu hoá mạch dùng máy tính

Các công cụ trợ giúp thiết kế bằng máy tính cho phép nâng cao năng suất thiết kế. Các kỹ thuật thiết kế cho phép giảm thời gian nâng cao chu trình thiết kế và giảm công

sức con người. Các kỹ thuật tối ưu làm tăng chất lượng thiết kế. Do đó kỹ thuật tổng hợp và tối ưu hóa mạch với sự trợ giúp của máy tính được sử trong hầu hết các quá trình thiết kế mạch điện tử số.

- Tổng hợp mạch điện:

  Gồm các phân đoạn sau:

  Tổng hợp ở mức kiến trúc bao gồm việc tạo ra góc độ cấu trúc của mô hình ở mức kiến trúc, có nghĩa là xác định và phân các chức năng của mạch thành các phép toán. Các phép toán này được gọi là tài nguyên thiết kế. Phân đoạn này thường được gọi là tổng hợp ở mức cao hay tổng hợp cấu trúc vì ở đó người thiết kế phải xác định các cấu trúc vĩ mô (trên mức độ các sơ đồ khối) của mạch.

  Tổng hợp ở mức logic là phân đoạn tạo ra góc độ cấu trúc của mô hình ở mức logic, gồm các thao tác sử dụng kỹ thuật logic để tạo nên mô hình logic. Mô hình này bao gồm các phần tử logic cơ bản và kết nối giữa các phần tử đó. Như vậy bước tổng hợp logic là bước xác định cấu trúc vi mô (ở mức các phần tử logic cơ bản) của mạch.

  Tổng hợp ở mức hình học bao gồm việc tạo ra góc độ vật lý của mô hình ở mức hình học. Ở mức này mô hình được mô tả thông qua các đặc tính của tất cả các mẫu hình học tạo nên dạng của các mạch. Phân đoạn này thường được gọi là thiết kế vật lý.

- Tối ưu hóa mạch điện

  Bài toán tối ưu hóa luôn đi đôi với bài toán tổng hợp mạch. Tối ưu hóa không những để đạt được ở mức độ cao nhất về chất lượng mạch mà còn tạo ra những mạch có tính cạnh tranh cao.

  Xét hai độ đo chất lượng quan trọng: diện tích và hoạt động của mạch. Ngoài ra một độ đo chất lượng quan trọng nữa là khả năng dễ kiểm tra và phát hiện lỗi của mạch.

  Diện tích của mạch được xác định bằng tổng diện tích của các phần tử mạch. Do đó diện tích của mạch có thể được xác định thông qua góc độ cấu trúc của mạch nếu ta biết diện tích của từng thành phần mạch. Thông thường các phần tử cơ bản của mạch logic là các phầ tử logic, các thanh ghi, các phần tử này có diện tích biết trước tùy thuộc vào từng loại thiết kế. Diện tích các dây nối đóng vai trò quan trọng và không thể bỏ qua. Các

thành phần diện tích này có thể được xác định từ mô hình mạch trên góc độ vật lý hoặc ước lượng từ các mô hình theo góc độ cấu trúc theo các phương pháp thống kê.

Hiệu năng của mạch được tối ưu hóa dựa trên thời gian trễ, thời gian đồng bộ, cạnh tranh trên các phần tử,… Để tính toán độ đo hoạt động của mạch cần thiết phải phân tích cấu trúc và hành vi của mạch. Vấn đề này khác nhau đối với các loại mạch khác nhau.

Hiệu năng của mạch tổ hợp được xác định thông qua thời gian trễ truyền từ đầu vào đến đầu ra. Thông thường để giảm độ phức tạp của tính toán, ta luôn giả thiết rằng các giá trị đầu vào xuất hiện trong cùng một thời điểm và hiệu năng của mạch được tính qua thời gian trễ truyền theo đường dữ liệu dài nhất.

Đối với các mạch tuần tự đồng bộ, độ đo hiệu năng có thể được xác định thông qua thời gian quay vòng của mạch. Thời gian này là chu kỳ đồng bộ nhanh nhất có thể đặt vào mạch.

Các mạch đồng bộ có thể thực hiện dãy các phép toán theo chế độ dây chuyền, trong đó mạch sẽ thực hiện các phép toán song song trên những tập hợp dữ liệu khác nhau. Như vậy hiệu năng của mạch còn có thể được thực hiện qua khả năng xử lý dữ liệu, lượng dữ liệu mà mạch có thể xử lý. Độ đo đó gọi là thông lượng của mạch.

Với những độ đo nói trên, tối ưu hóa hiệu năng của mạch bao gồm việc giảm thiểu thời gian trễ truyền đối với mạch tổ hợp, thời gian quay vòng và thời gian thực hiện đối với mạch tuần tự đồng bộ, làm tăng tối đa thông lượng của mạch đối với những mạch thực hiện theo kỹ thuật dây chuyền.

Ngoài ra, hiệu năng của mạch còn liên quan tới khả năng phát hiện lỗi và định vị vị trí lỗi trong mạch.

Tóm lại bài toán tối ưu hóa thiết kế được đưa về kết hợp hai bài toán: giảm thiểu diện tích thực tế của mạch và tăng hiệu năng của mạch với khả năng cao nhất có thể có. Bài toán tối ưu hóa có thể phụ thuộc vào các ràng buộc như giới hạn trên về diện tích và giới hạn dưới về hiệu năng. Bài toán tối ưu hóa có thể được biểu diễn trong không gian vectơ như sau. Tập hợp các cấu trúc có thể có của mạch sẽ được thiết kế tạo thành một không gian.

Không gian này gọi là không gian thiết kế và chứa một số hữu hạn các điểm trong đó tương ứng với một thiết kế cụ thể. Mỗi điểm (tương ứng là thiết kế) sẽ có các giá trị diện tích và hiệu năng tương ứng. Ta sẽ lập hàm giá trị trên cơ sở các đối tượng như diện tích, thời gian trễ, thời gian thực hiện, thời gian quay vòng, thông lượng. Bài toán tối ưu hóa trở thành bài toán tìm kiếm điểm xác định trong không gian thiết kế sao cho các đối tượng đạt giá trị tối ưu.

Chương II: Cơ sở toán học của mạch

2.1 Đại số Boole và lý thuyết chuyển mạch

1. Đại số Bool và lý thuyết tập hợp

Lý thuyết chuyển mạch là cơ sở thiết kế các hệ thống số hiện đại. Lý thuyết này dựa trên logic ký tự do nhà toán học Bool sang tạo nên.

Định nghĩa: Đối với tập hợp B = {a, b,…} và hai toán tử '+' và '.', nếu bốn tiên đề sau thỏa mãn thì hệ thống đại số gọi là đại số Bool:

$1)\ \forall a, b \in B,\ a + b = b + a,\ a.b = b.a;$

$2)\ \forall a, b, c \in B,$

$a + (b.c) = (a + b).(a + c),\ a.(b + c) = (a.b) + (a.c)$

$3)\ \exists 1 \in B, \exists 0 \in B:$

$\forall a \in B,\ a + 0 = a,\ a.1 = a$

$4)\ \exists \overline{a} \in B : \forall a \in B,\ a + \overline{a} = 1,\ a.\overline{a} = 0$

Các định lý của đại số Bool:

$1.a + a = a$

$2.a.a = a$

$3.a + 1 = 1$

$4.a.0 = 0$

$5.\overline{\overline{a}} = a$

$6.(a.b).c = a.(b.c)$

$7.(a + b) + c = a + (b + c)$

$8.a + a.b = a$

$9.a.(a + b) = a$

$10.\overline{a + b} = \overline{a}.\overline{b}$

$11.\overline{a.b} = \overline{a} + \overline{b}$

2.2 Các hàm logic và dạng chuẩn tắc (đã được học, SV tự đọc tài liệu)

2.3 Tối thiểu hoá các biểu thức logic (đã được học, SV tự đọc tài liệu)

Chương III: Cơ sở của thiết kế Logic (3 tiết)

3.1 Đặc điểm của quá trình thiết kế mạch của máy tính

1. Đánh giá thời gian trễ trong các mạch lôgic

   Trong quá trình thiết kế các thiết bị tính toán, ngoài chức năng thực hiện các phép toán loogic của mạch nhà thiết kế cần phải tính đến cả thời gian trễ của tín hiệu khi đi qua các phần tử lôgic và các đoạn mạch. Thời gian trễ này ảnh hưởng lớn đến hoạt động của mạch trong thực tế.

   Thời gian trễ thuần túy t_d là thời gian truyền tín hiệu qua mạch. Trong trường hợp này, thời gian trễ của mạch gồm các phần tử chức năng mắc nối tiếp sẽ bằng tổng các thời gian trễ của các phần tử chức năng và thời gian trễ của các phần tử liên kết.

2. Các mạch tổ hợp và các mạch tuần tự

   Sự phân chia các mach số thành các mạch tổ hợp và các mạch tuần tự xuất phát từ các điểm khác biệt cơ bản giữa các đặc tính của chúng.
   Các biến đầu ra của mạch tổ hợp chỉ phụ thuộc vào các tác động vào mạch tại thời điểm hiện tại
   Các mạch tuần tự tính toán các giá trị ra dựa vào các giá trị đầu vào không chỉ tại thời điểm hiện tại mà còn phụ thuộc cả vào những trạng thái của mạch tính từ thời điểm đang xét trở về trước. Các trạng thái của mạch tuần tự được lưu trữ vào các phần tử nhớ trong thành phần của mạch.
   Các mạch tuần tự được cấu tạo bởi 2 phần: các bộ phận nhớ để lưu trữ các trạng thái của mạch; và mạch tổ hợp dùng để điều khiển các phần tử nhớ và hình thành các giá trị tín hiệu ra

   Trong kỹ thuật tính toán, các mạch tổ hợp là các mạch mã hóa, giải mã, bộ so sánh tín hiệu, bộ cộng,… Các mạch tuần tự là các trigơ , các mạch nhớ, thanh ghi, bộ đếm,… Các phương pháp phân tích và tổng hợp các mạch tổ hợp đơn giản hơn so với mạch tuần tự.

   Trong quá trình thiết kế, các mạch số thường được biểu diễn bằng nhiều phương pháp, ví dụ như các bảng, ma trận, đồ thị hoặc bằng các otomat.

3.2 Các phần tử logíc cơ bản

Trong quá trình thiết kế các mạch tích hợp có một số phần tử logic cơ bản được sử dụng phổ biến. Việc thực hiện các phần tử logic này phụ thuộc vào công nghệ sản xuất linh kiện điện tử như công nghệ transistor CMOS, công nghệ ttransistor trường, TTL,…

Các phần tử logic cơ bản gồm phần tử AND, OR, NOT, XOR, NOR, NAND, ngoài ra trong nhiều trường hợp phần tử đóng ngắt cũng được coi là phần tử cơ bản.

Trên quan điểm về khả năng xây dựng các hàm logic bất kỳ, một phần tử cơ bản hợp thành hệ đầy đủ. Điều đó có nghĩa là với các hàm cơ bản tham gia vào hệ đầy đủ, ta có thể xây dựng mọi hàm logic. Ta có hệ các phần tử AND, OR, NOT tạo thành một hệ đầy đủ vì ta có thể xây dựng mọi hàm logic theo các dạng chuẩn tắc tuyển hoặc hội với sự tham gia của các phần tử này.

Hệ các phần tử NOT, AND tạo thành một hệ đầy đủ vì phép toán OR có thể được biểu diễn qua NOT và AND.

$$z = x + y = \overline{\overline{x}.\overline{y}}$$

- Hệ các phần tử NOT, OR tạo thành một hệ đầy đủ
- Hệ phép toán chỉ có phần tử NAND hoặc NOR là một hệ đầy đủ
- Phần tử đóng ngắt và phần tử NOT tạo thành một hệ đầy đủ.

….

Ngoài các phần tử lôgic hai đầu vào nêu trên, trong công nghệ còn sử dụng những phần tử có nhiều hơn 2 đầu vào. Các phần tử có nhiều đầu vào có thể được biểu diễn như ghép nối nhiều lớp các phần tử lôgic có số lượng đầu vào ít hơn hoặc như một phần tử duy nhất.

Các phương pháp xây dựng mạch đó được lựa chọn dựa vào các tiêu chuẩn tối ưu về diện tích tinh thể bán dẫn của mạch, năng lượng mà mạch tiêu thụ và thời gian trễ truyền của tín hiệu khi đi qua mạch.

3.3 Thiết kế các mạch tổ hợp

1. Tổng hợp mạch theo biểu thức lôgic

Thông thường các hàm lôgic được biểu diễn bằng các biểu thức lôgic chứa các phép toán AND, OR, XOR, NOT. Những biểu thức đó có thể được thực hiện thành mạch thông qua những phần tử lôgic cơ sở.

Biểu thức có thể được phân tách dưới dạng các cây tính toán tương ứng với các dấu ngoặc và mức độ ưu tiên của các phép toán.

Ví dụ:

$$f = \overline{x + y.\overline{z}} \oplus w$$

Phương pháp xây dựng mạch trực tiếp từ các biểu thức lôgic như đã nêu trên là phương pháp đơn giản. Với phương pháp này ta có thể xây dựng mọi hàm lôgic với

những độ phức tạp khác nhau nhưng nó có một nhược điểm lớn là liên quan đến thời gian trễ của tín hiệu đi qua mạch và độ dài của tín hiệu

2. Thực hiện các mạch hai tầng

Trong các phương pháp thiết kế mạch hai tầng trên cơ sở các phần tử AND-OR và OR-AND gồm:

+ Xây dựng mạch trực tiếp từ các dạng chuẩn tắc, trên thực tế phương pháp này không hiệu quả. Khi xây dựng mạch các phần tử lôgic có n đầu vào cần nhiều diện tích trên tinh thể bán dẫn hơn khi xây dựng trên phần tử có hai đầu vào. Do đó giá thành mạch cũng phụ thuộc vào số lượng đầu vào của các phần tử lôgic. Đối với việc thiết các mạch tổ hợp sử dụng sơ đồ hai tầng thông qua các phần tử AND-OR. Bên cạnh việc giảm số lượng các phần tử cần giảm số lượng đầu vào của các phần tử đó.

Ví dụ:

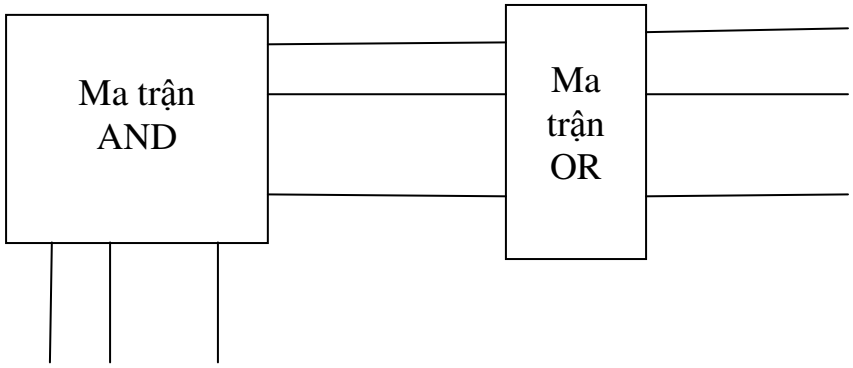
$$f = \sum(0,2,5,6,7,8,11,12,15)$$

Thiết kế theo dạng chuẩn tắc tuyển trực tiếp và thiết kế sau khi tối thiểu hóa

Xây dựng mạch trực tiếp từ các dạng chuẩn tắc, trên thực tế phương pháp này không hiệu quả. Khi xây dựng mạch các phần tử lôgic có n đầu vào cần nhiều diện tích trên tinh thể bán dẫn hơn khi xây dựng trên phần tử có hai đầu vào. Do đó giá thành mạch cũng phụ thuộc vào số lượng đầu vào của các phần tử lôgic. Đối với việc thiết các mạch tổ hợp sử dụng sơ đồ hai tầng thông qua các phần tử AND-OR. Bên cạnh việc giảm số lượng các phần tử cần giảm số lượng đầu vào của các phần tử đó.

3. Thực hiện mạch tổ hợp trên cơ sở các PLA/ROM

Ma trận logic lập trình là các khối phần tử vĩ mô được sử dụng để thiết kế những mạch LSI, VLSI theo cấu trúc các mạch hai tầng. Các PLA bao gồm các ma trận AND và ma trận OR được mắc nối tiếp. Ma trận thứ nhất là ma trận AND, ma trận thứ hai là ma trận OR. Như ta đã biết cấu trúc cấu trúc 2 tầng AND – OR tương đương với cấu trúc hai tầng của các phần tử NAND. Theo nguyên lý đối ngẫu, cấu trúc đó cũng tương đương với cấu trúc hai tầng NOR-NOR. Do đó trong kỹ thuật thiết kế cấu trúc hai tầng NAND-NAND và NOR-NOR được sử dụng rộng rãi.



Một phần tử hay được sử dụng của cấu trúc VLSI và cũng cho phép lập trình được như PLA là ROM. ROM khác PLA ở chỗ ROM là cấu trúc cho phép lập trình các giá trị 1 và 0 đối với các tích cực tiểu một cách tùy ý. Nhược điểm của chính của ROM so với PLA là ROM có hiệu suất sử dụng diện tích tinh thể thấp. So với PLA, ROM có độ mềm dẻo cao hơn trên quan điểm thay đổi các hàm logic trong hệ hàm, do đó ROM được sử dụng trong kỹ thuật tính toán rộng rãi hơn PLA. Đôi khi, để thay thế ROM, người ta có thể sử dụng các bộ nhớ truy cập ngẫu nhiên trong đó có ghi sẵn bảng chân lý.

3.4 Những vấn đề khi thiết kế mạch tổ hợp

1. Những giai đoạn thiết kế mạch tổ hợp:

Quá trình thiết kế mạch tổ hợp thường được thực hiện theo những bước sau:

- Khảo sát đặc điểm về chức năng của mạch tổ hợp, những liên kết của mạch với những mạch khác theo đầu vào/đầu ra, thiết lập các quan hệ tương ứng với các biến logic.

- Đánh giá kích thước của bài toán và giải quyết vấn đề phân chia mạch tổ hợp thành các phân hệ theo mức độ cần thiết.

- Xây dựng bảng thể hiện các chức năng của mạch tổ hợp.

- Tối thiểu hóa mạch.

- Lựa chọn các phần tử lôgic và biểu diễn hàm lôgic theo các hệ cơ sở đã lựa chọn.

2. Ảnh hưởng của thời gian trễ tới hoạt động của các mạch tổ hợp

- Làm thay đổi hoàn toàn chức năng hoạt động của mạch
Đối với các mạch tổ hợp, thời gian trễ không chỉ làm giảm tốc độ hoạt động của mạch mà còn có thể sinh ra các giá trị nhất thời bị sai ở đầu ra của mạch. Điều này có thể làm hoạt động của toàn hệ thống có thể bị thay đổi. Theo thời gian những giá trị

này sẽ biến mất và đầu ra của mạch sẽ nhận các giá trị được tính theo các hàm logic đã thiết kế. Nhưng các giá trị sai này rất nguy hiểm trong những trong những trường hợp khi mạch tổ hợp được nối với các mạch nhớ dung lưu trữ các trạng thái của hệ thống. Khi đó sẽ xuất hiện các trạng thái không dự đoán trước và hoạt động của toàn hệ thống có thể sai hoàn toàn. Những trường hợp này gọi là các ruit ro trong mạch.

Để loại trừ khả năng lỗi xuất hiện do rủi ro, trong các mạch tổ hợp người ta sử dụng đồng bộ quá trình nhận thông tin bằng các mạch nhớ nối với đầu ra của mạch tổ hợp. Thông tin được nhận vào mạch nhớ thông qua tín hiệu đồng bộ C. Tín hiệu này được tác động vào mạch nhớ sau khi các quá trình quá độ trong mạch tổ hợp kết thúc. Như vậy các tín hiệu sai sẽ không tác động đến phần tử nhớ và do đó không xuất trên đầu ra của mạch.

3.5 Thiết kế mạch dãy

Mạch tuần tự được thường được thiết kế theo cấu trúc bao gồm các mạch tổ hợp liên kết với các mạch nhớ. Các mach tổ hợp sẽ thực hiện các tính toán theo các hàm lô gic, còn các mạch nhớ dùng để lưu trữ các kết quả trung gian.

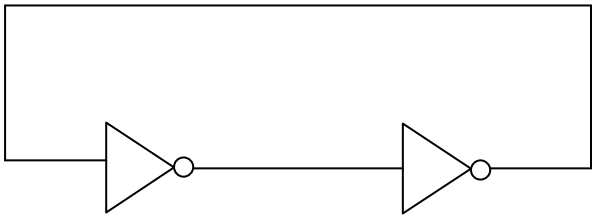
1. Nguyên lý của các mạch nhớ:

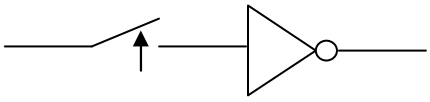
Các mạch tổ hợp cho phép thực hiện một số mạch phức tạp, như mạch nhân nhanh, nhưng đối với một số thao tác xử lý dữ liệu phức tạp hơn yêu cầu ghi nhớ các kết quả tính toán trung gian và thực hiện các thao tác lặp tương ứng với trình tự tính toán. Trong lĩnh vực xử lý số, các dữ liệu được biểu diễn dưới dạng nhị phân do đó cần thiết những mạch cho phép nhớ lại hai trạng thái ‘0’ và ‘1’.

Có hai loại sơ đồ nhớ kinh điển: mạch nhớ tĩnh và mạch nhớ động

Xét các dạng mạch nhớ được xây dựng trên những phần tử đơn giản nhất. Việc xây dựng các phần tử nhớ đều dựa trên các nguyên lý chung: hoặc phải xây dựng các vòng phản hồi tín hiệu trong mạch để duy trì giá trị ô nhớ hoặc là dùng thiết bị phụ trợ để duy trì giá trị ô nhớ.

Vòng phản hồi





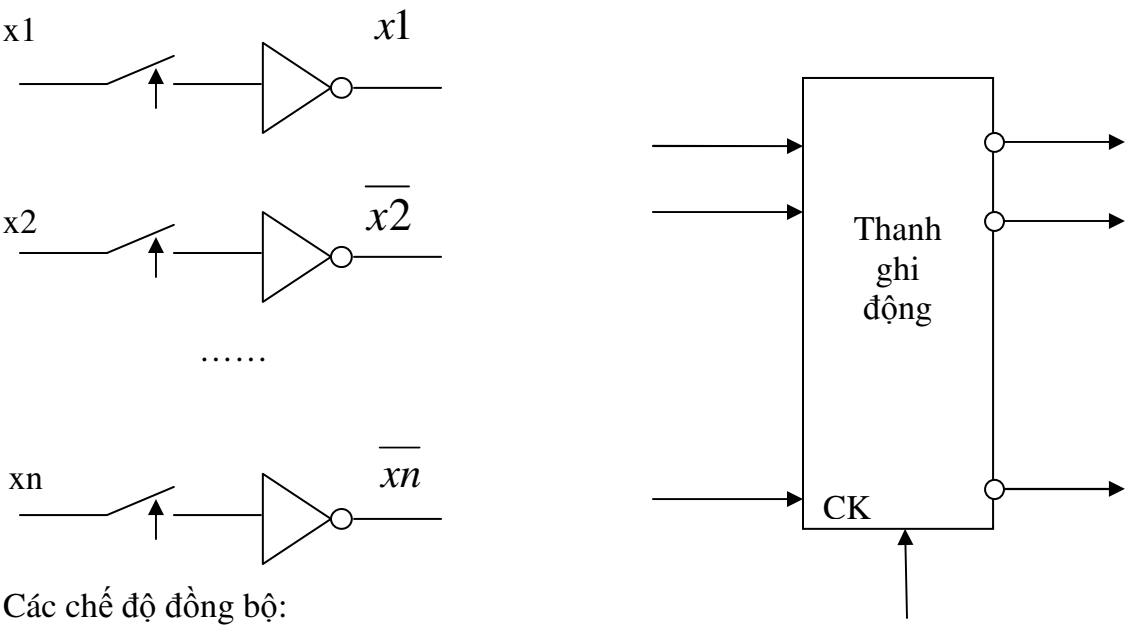
Mạch nhớ tĩnh

Mạch nhớ được xây dựng từ hai phần tử NOT mắc nối tiếp và một vòng phản hồi. Khi mạch ở trạng thái ổn định đầu ra của hai phần tử NOT lần lượt là ‘1’, ‘0’ hoặc ‘0’, ‘1’. Những giá trị đầu ra của các phần tử NOT cùng với vòng phản hồi có tác dụng duy trì trạng thái của phần tử nhớ . Như vậy mạch này có tác dụng lưu trữ các giá trị dữ liệu ‘1’ và ‘0’. Mạch nhớ này gọi là mạch nhớ tĩnh.

Mạch nhớ động: Phần tử khóa SW nối tiếp với phần tử NOT. Phần tử nhớ này lưu trữ giá trị dữ liệu của ô nhớ bằng phần tử điện dung ký sinh tại đầu vào của phần tử logic NOT. Khi tín hiệu điều khiển K=1, khóa SW đóng và điện dung ký sinh tại đầu vào phần tử NOT được tích điện. Khi K=0, khóa SW mở và điện tích ở đầu vào phần tử NOT bị cô lập với mạch ngoài. Do đó phần tử NOT cần phải có giá trị trở kháng đầu vào cao. Thời gian lưu trữ được xác định theo thời gian lưu giữ điện tích của phần tử điện dung đầu vào.

2. Các sơ đồ thanh ghi và trigơ

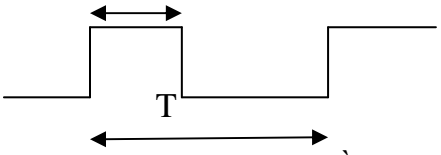
Sơ đồ nhớ đọng cho phép lưu trữ trực tiếp một bit thông tin. Để có thể ghi nhớ đồng thời được n bit thông tin người ta dung song song n phần tử nhớ động. Thiết bị đó được gọi là thanh ghi động n bit



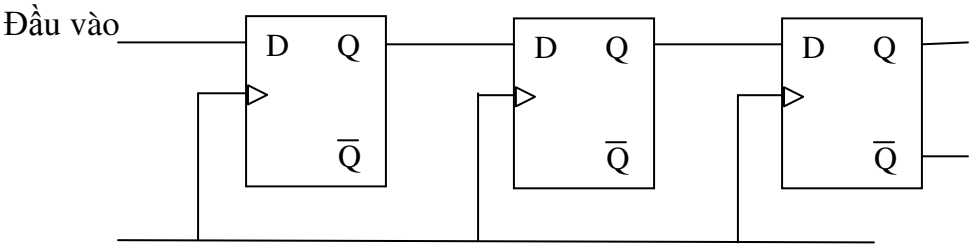
3. Các chế độ đồng bộ:

Tín hiệu điều khiển các đầu vào của mạch nhớ thường cung cấp bởi các mạch đồng bộ. Đầu vào tín hiệu đồng bộ tương ứng của mạch gọi là các đầu vào đồng bộ. Tín hiệu đồng bộ được xác định bởi độ dài khoảng thời gian t mà tín hiệu ở trạng thái ‘1’ và chu kỳ T.

t



Các mạch nhớ sử dụng tín hiệu đồng bộ thực hiện các thao tác đọc và ghi dữ liệu khi đầu vào đồng bộ nhận giá trị 1 hoặc 0. Các mạch loại này được gọi là các mạch làm việc trong chế độ đồng bộ theo mức. Khi thiết kế các mạch làm việc theo chế độ đồng bộ, độ dài của tín hiệu đồng bộ đóng một vai trò quan trọng trong hoạt động của mạch. Nếu các trigo D đồng bộ theo mức được ghép nối nối tiếp và nối chung các đầu tín hiệu đồng bộ ta sẽ có thanh ghi làm việc theo chế độ như sau:



Nếu tín hiệu đồng bộ có độ dài đủ lớn thì sau một khoảng thời gian tất cả các trigo sẽ ghi cùng một giá trị. Trong trường hợp ngược lại, khi độ dài t của tín hiệu đồng bộ thỏa mãn hệ thức:

$$\tau_{\max} < t < d_{\min}$$

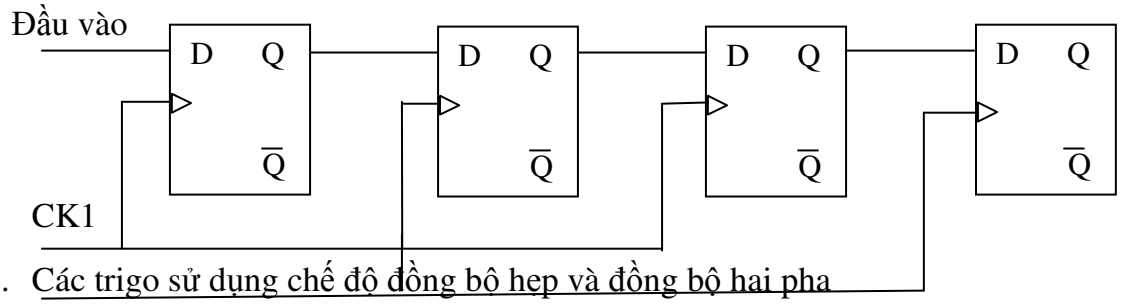
Trong đó $\tau_{\max}$ là giá trị cực đại trong các độ dài cực tiểu của các xung đảm bảo việc ghi dữ liệu; d_{min} là thời gian trễ cực tiểu khi tín hiệu lan truyền trong mạch nhớ từ đầu vào tới đầu ra. Khi đó thanh ghi thực hiện dịch dữ liệu sang phải một lớp sau mỗi lần nhận tín hiệu đồng bộ. Chế độ đồng bộ thỏa mãn bất đẳng thức trên gọi là chế độ đồng bộ hẹp.

Ưu điểm của chế độ đồng bộ hẹp là đơn giản, dễ thực hiện và để xây dựng mạch phát xung đồng bộ không cần nhiều thiết bị. Nhưng chế độ đồng bộ này ít được sử dụng trong các mạch VLSI vì trên tất cả các phần mạch ta khó có thể đạt được bất đẳng thức trên. Trong nhiều trường hợp để có thể kiểm soát được hành vi của các mạch người ta sử dụng một biến thể của chế độ đồng bộ hẹp, đó là chế độ làm việc theo sườn.

Đồng bộ theo sườn dốc là chế độ làm việc của các mạch với tín hiệu đồng bộ có độ rộng xung lớn và tác động thực hiện theo sườn lên hoặc sườn xuống của xung. Ở chế độ này các tác động của tín hiệu đồng bộ đối với mạch xảy ra cũng tương tự như trong chế độ đồng bộ hẹp.

Nhược điểm của phương pháp đồng theo sườn dốc và đồng bộ hẹp là miền dịch pha cho phép của xung đồng bộ. Độ dịch tương đối của các xung là hiện tượng xuất hiện độ dịch pha của các xung trên đầu vào đồng bộ của các phần tử nhớ trong mạch VLSI. Độ dịch pha của các xung đồng bộ xuất hiện chủ yếu do kết quả của sự phân bổ các thời gian trễ trong các mạch phân phối tín hiệu đồng bộ.

Đồng bộ hai pha:



4. Các trigo sử dụng chế độ đồng bộ hẹp và đồng bộ hai pha

   Trong kỹ thuật, nhiều mạch nhớ sử dụng các trigo hoạt động theo các chế độ đồng bộ theo sườn lên hoặc làm việc theo chế độ đồng bộ hai pha. Các chế độ đồng bộ này cho phép giải quyết vấn đề cạnh tranh giữa các phần tử logic trong hoạt động của các trigo. Điều này làm cho các mạch nhớ hoạt động tin cậy hơn và làm giảm tính bất định của mạch.

5. Thiết kế các mạch tuần tự bằng các otomat hữu hạn

   Trong các mạch tổ hợp, giá trị đầu ra được hoàn toàn xác định theo các tín hiệu đầu vào tại thời điểm hiện tại. Nhưng các mạch tuần tự có các giá trị đầu ra được xác định theo dãy các giá trị đầu vào tác động vào mạch tại những thời điểm trước thời điểm hiện tại.

3.6 Những vấn đề khi thiết kế mạch dãy

1. Hiện tượng cạnh tranh trong các mạch tuần tự

   Khác với các mạch tổ hợp, trạng thái của các mạch tuần tự được xác định không chỉ từ các tín hiệu vào mà còn phụ thuộc vào các trạng thái trước đó của mạch. Việc lưu trữ các trạng thái trước đó của mạch được thực hiện trên các phần tử nhớ như thanh ghi, trigo,… Như vậy mạch tuần tự có thể biểu diễn bằng các otomat có số trạng thái hữu hạn.

   Trong các otomat, do ảnh hưởng của thời gian trễ tín hiệu trên các phần tử mạch, người ta quan sát thấy những trạng thái trung gian chuyển tiếp giữa các trạng thái trong thiết kế. những trạng thái chuyển tiếp này có ảnh hưởng lớn

đến hoạt động của mạch. Sự cạnh tranh trong quá trình truyền tín hiệu giữa các phần tử mạch có thể dẫn đến việc thiết lập những trạng thái sai của các phần tử nhớ, do đó có thể dẫn đến hoạt động sai của mạch. Khi thiết kế các otomat, sự cạnh tranh trong các phần tử mạch phải được loại trừ.

Ví dụ:

Trong nhiều trường hợp cơ chế cạnh tranh trong các mạch tuần tự xảy ra như sau:

Các trạng thái của otomat, do các tín hiệu thiết lập thay đổi lại phụ thuộc vào trạng thái của các phần tử nhớ. Do hiện tượng trễ tín hiệu trong các phần tử logic, sự thiết lập các giá trị của ô nhớ không xảy ra đồng thời. Điều đó có thể ảnh hưởng tới tín hiệu thiết lập của các phần tử khác và do đó có thể dẫn đến những trạng thái không lường trước.

Phương pháp quan trọng để ngăn chặn hiện tượng cạnh tranh giữa các phần tử trong mạch là sử dụng các chế độ đồng bộ. Trong các chế độ này, việc nhận các giá trị vào các ô nhớ và đưa các giá trị đó vào tới đầu ra được thực hiện vào những thời điểm xác định sau khi các quá trình quá độ trong mạch kết thúc.

Chương IV: Những khái niệm chung về mô hình hoá phần cứng

4.1 Mô hình hoá phần cứng

Mô hình hóa phần cứng là những biểu diễn phần cứng trên các mức độ trừu tượng khác nhau. Mô hình cho ta thấy những phần tử liên quan mà không chỉ rõ những chi tiết của chúng.

Trong quá trình thiết kế, mô hình được sử dụng để đặc trưng cho mạch, thể hiện các tính chất của mạch và là phương tiện để trao đổi thông tin về thiết kế giữa những người thiết kế và máy tính.

Các đặc tả mạch là các mô hình mô tả chi tiết mạch sẽ được xây dựng. Các đặc tả này luôn đi đôi với các rang buộc về mặt thiết kế như hiệu năng, diện tích.

Mạch điện có thể được mô hình hóa theo những cách khác nhau tương ứng với các mức độ trừu tượng (mức kiến trúc, mức logic, mức hình học), theo các góc độ quan sát (góc độ hành vi, góc độ cấu trúc và góc độ vật lý) và tương ứng với các phương pháp mô hình hóa được sử dụng trong quá trình thiết kế (các ngôn ngữ mô tả thiết kế, các sơ đồ mạch hoặc các mô hình toán học).

Trong quá trình thiết kế mạch, hiện nay người ta thường sử dụng các ngôn ngữ mô hình hóa phần cứng (HDL), ngôn ngữ này cũng giống như các ngôn ngữ lập trình phần mềm. Tính súc tích dễ hiểu của các ngôn ngữ HDL giúp cho việc mô tả mạch bằng các ngôn ngữ đó được ưa chuộng hơn việc biểu diễn bằng các biểu đồ, sơ đồ trạng thái, sơ đồ logic, …

4.2 Các ngôn ngữ mô hình hoá phần cứng

Trong lĩnh vực thiết kế mạch, các ngôn ngữ mô hình hóa phần cứng (HDL) xuất hiện do nhu cầu phải có công cụ mô tả chính xác mạch về cấu trúc cũng như hành vi.

Sự khác biệt giữa các ngôn ngữ lập trình và ngôn ngữ mô hình hóa phần cứng:

- Các mạch phần cứng có thể thực hiện những pháp toán có mức độ song song (đồng thời) lớn còn các phần mềm thì ngược lại, trên những máy đơn xử lý chỉ thực hiện được những phép toán tuần tự. Về mặt này, các ngôn ngữ mô tả phần cứng sẽ gần giống với các ngôn ngữ lập trình cho các máy tính xử lý song song.

- Các mô hình phần cứng luôn phải chứa những thông tin về cấu trúc như sự tiếp xúc của mạch với những mạch khác làm nảy sinh yêu cầu phải mô tả các cổng vào ra của mạch và khuôn dạng dữ liệu được trao đổi qua những cổng đó. Do đó các ngôn ngữ mô

hình hóa phần cứng phải hỗ trợ việc mô tả thiết kế trên góc độ hành vi và cấu trúc nhằm biểu diễn các đặc trưng của mạch một cách có hiệu quả.

- Việc xác định thời gian và thời điểm thực hiện của các phép toán hết sức quan trọng trong phần cứng do các tương tác giữa các thành phần phần cứng với nhau. Trong khi đó vấn đề tương tác theo thời gian ít ảnh hưởng tới việc thực hiện các phép toán trong các chương trình phềm mềm, trừ một số trường hợp trong những ứng dụng thời gian thực.

Mạch điện có thể được mô tả dưới những góc độ quan sát khác nhau, do đó các ngôn ngữ HDL với những đặc trưng tương ứng cũng được phát triển.

1. Những đặc điểm khác biệt của các ngôn ngữ mô tả phần cứng.

   Các ngôn ngữ được đặc trưng nhờ các quy tắc cú pháp, ngữ nghĩa và thực tế sử dụng. Cú pháp liên quan đến các cấu trúc của ngôn ngữ và có thể được thực hiện qua các quy tắc ngữ pháp. Ngữ nghĩa chỉ ra ý nghĩa của các thành phần ngôn ngữ. Các quy tắc ngữ nghĩa tác động tương ứng với tới những thành phần ngôn ngữ thỏa mãn các quy tắc cú pháp. Thực tế sử dụng ngôn ngữ liên quan đến những khía cạnh khác của ngôn ngữ, bao gồm cả vấn đề sử dụng và thực hiện ngôn ngữ.

   Có thể chia ngôn ngữ thành 2 loại:

   + Ngôn ngữ thủ tục (procedural): các chương trình thể hiện các tác động mong muốn bằng cách mô tả dãy các bước cần thiết để thực hiện các tác động đó.

   + Ngôn ngữ khai báo (declarative): các mô hình thể hiện các vấn đề sẽ được giải quyết bằng tập hợp các đặc tả, khai báo mà không đưa ra chi tiết các phương pháp giải quyết. Do đó trình tự mô tả các khối cơ sở không quan trọng trong các ngôn ngữ khai báo.

Các ngôn ngữ mô hình hóa phần cứng được phân loại dựa trên cơ sở góc độ quan sát các đối tượng được mô tả. Ví dụ những ngôn ngữ mô tả thiết kế ở mức vật lý sẽ được hỗ trợ nhờ các đặc tả những đối tượng hình học nguyên thủy, các thao tác trên các đối tượng đó.

Các ngôn ngữ HDL thường được phát triển kèm theo các bộ mô phỏng. Tốc độ thực hiện là một trong những yêu cầu đối với bộ mô phỏng. Các thuật toán mô phỏng theo sự kiện được sử dụng rộng rãi bởi vì chúng cho phép bộ mô phỏng giảm thiểu số lượng các tính toán và do đó làm giảm thời gian thực hiện mô phỏng.

Quá trình mô phỏng bao gồm việc tính các giá trị cuiar tín hiệu trong một khoảng thời gian xác định. Khoảng thời gian được chia thành các khung thời gian. Trong mỗi khung thời gian chu trình mô phỏng gồm các bước sau:

- Tín hiệu được lan truyền trong mạch và được cập nhật

- Tất cả các quá trình nhạy cảm với các sự kiện được tính toán cho tới khi chúng được dừng lại.

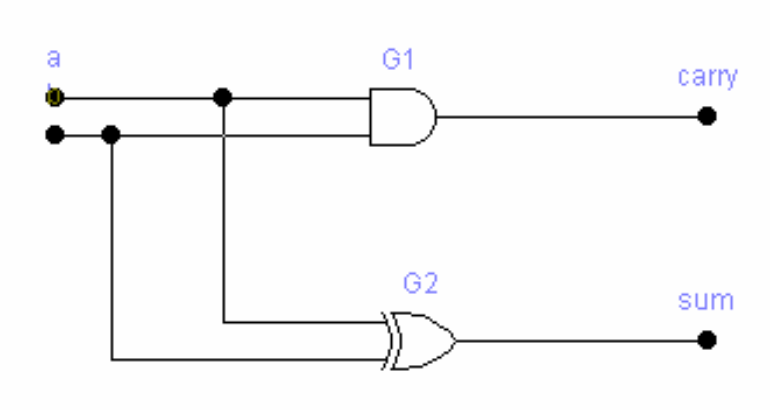
- Khi tất cả các quá trình tính toán đều dừng lại, thời gian trong bộ mô phỏng được chuyển sang khung tiếp theo và hình thành chu trình mô phỏng mới.

Phương pháp mô hình hóa mạch này đủ tổng quát để cho phép mô tả các mạch đồng bộ và không đồng bộ. Trong các mạch đồng bộ tất cả các quá trình có thể được kích hoạt tại mỗi chu kỳ đồng hồ. Nếu thời gian trễ của các thành phần mạch đã biết, chúng có thể được đặc tả như một thuộc tính của tín hiệu và bộ mô phỏng có thể mô tả chính xác hành vi theo thời gian của mạch. Trong trường hợp thời gian trễ là chưa biết, người thiết kế hoặc ohair tìm những thông tin về thời gian trễ từ các mô hình hành vi hoặc đưa ra các giả thiết và ràng buộc về thời gian trễ để thực hiện các phép toán trong khung thời gian xác định.

2. Các ngôn ngữ mô tả cấu trúc phần cứng

   Mô hình được mô tả bằng các ngôn ngữ cấu trúc thể hiện các kết nối giữa các phần tử. Do đó ngôn ngữ này có sức mạnh biểu cảm tương tự như các sơ đồ mạch mặc dù những đặc diểm của ngôn ngữ cho phép cung cấp những mô tả khái quát hơn. Các hệ thống thứ bậc trong ngôn ngữ cho phép tạo các mô hình có tính môđun hóa và ngắn gọn. Các thành phần cơ sở của ngôn ngữ cấu trúc cho phép xếp các ngôn ngữ này vào nhóm các ngôn ngữ khai báo (delarative), mặc dù một số ngôn ngữ mô tả cấu trúc có chứa những thành phần thủ tục. Các biến trong ngôn ngữ tương ứng với các cổng của các phần tử.

Ví dụ: Mô tả mạch nửa tổng bằng ngôn ngữ VHDL.



```
Architecture STRUCTURE of Half_Adder is
        Component AND2
            Port (x, y: in bit; 0: out);
        End Component
        Component EXOR2
            Port (x, y: in bit; 0: out);
        End Component
Begin
        G1: AND2
            Port map (a, b, carry);
        G2: EXOR2
            Port map (a, b, sum);
End STRUCTURE;
```

Trong mô hình này chứa hai khai báo của mô hình khác là AND2 và EXOR2 và hai khởi tạo mô hình là G1 và G2. Thông tin cụ thể về các mô hình AND2 và EXOR2 được khai báo ở một vị trí khác như trong thư viện chuẩn.

Dạng khác của biến là các siêu biến. Các biến này được dùng để làm mô hình mạch gọn hơn, các biến loại này có thể là các chỉ số của mảng. Các biến loại này không biểu diễn trực tiếp các thành phần của phần cứng và được loại trừ khỏi mô hình sau bước dịch đầu tiên..

Ví dụ: Xây dựng mô hình của mảng 32 bộ đảo tín hiệu nối giữa hai tuyến tín hiệu bằng ngôn ngữ VHDL. Từ khóa generate cho ta nhiều phiên bản của biến vòng lặp i:

```
Architecture STRUCTURE of BUS_INV is
Component INVERTER
        Port (i1: in bit; 01: out bit);
End component
Begin
  G: for i in 1 to 32 generate
```

INV: INVERTER **port map** (**inputs** (i), **output** (i));

End generate;

End STRUCTURE;

3. Ngôn ngữ mô tả chức năng phần cứng

Các mạch tổ hợp có thể được mô tả bằng tập hợp các phần tử logic và tập hợp các phương trình. Các cấu trúc này liên kết các biến thành các biểu thức logic. Phương thức khai báo ứng dụng tốt nhất cho trường hợp mô tả các mạch tổ hợp – những mạch được mô tả không cần bộ nhớ: Các mạch tổ hợp có thể coi là các ghép nối (về mặt cấu trúc) của các toán tử, trong đó mỗi toán tử xác định một hàm logic . Các mô hình này khác với mô hình cấu trúc ở chỗ không có tương quan "một - một" giữa các biểu thức và các cổng logic, vì đối với một số biểu thức sẽ không tồn tại phần tử logic thực hiện biểu thức đó.

Các ngôn ngữ thủ tục có thể sử dụng để mô tả các mạch tổ hợp. Phần lớn các ngôn ngữ cấu trúc cho phép thực hiện phép gán nhiều lần với một biến. Để tránh sự mập mờ về giá trị biến, trong các ngôn ngữ mô tả chức năng có các cơ chế giải quyết mập mờ như những phép toán sau sẽ xóa bỏ tác động của phép toán trước.

Ví dụ: Mô tả bộ nửa tổng trên VHDL dùng mô hình hành vi.

Architecture BEHAVIOR **of** HALF_ADDER **is**

Process

Begin

Carry <= (a **and** b);

Sum <= (a **xor** b);

End process

End BEHAVIOR;

Trong khối này ta thấy bên trong khối giới hạn bởi cặp từ khóa **Process** và **End** process các biểu thức được thực hiện một cách tuần tự. Hai phép gán tương ứng với hai cấu trúc trong mô hình bộ nửa tổng.

Mạch tuần tự cũng thường được mô hình hóa bằng các ngôn ngữ thủ tục. Các ôtômat có trạng thái hữu hạn biểu diễn hoạt động của mạch có thể được mô tả bằng các mô hình thủ tục trong đó các biến lưu giữ các thông tin trạng thái. Khi đó các thao tác của otomat hữu hạn được mô tả bằng những bước lặp đồng bộ

theo xung nhịp đồng hồ, với những phân nhánh chuyển trạng thái tương ứng với trạng thái hiện thời.

Ví dụ: Mô tả otomat hữu hạn thực hiện nhận biết các bit ‘1’ liên tiếp ở dòng dữ liệu vào.

4.3 Các mô hình trừu tượng

1. Các cấu trúc

Cấu trúc của mạch có thể mô hình hóa dựa vào các cấu trúc liên kết bao gồm tập hợp các môđun, tập hợp các mạng kết nối và quan hệ liên kết giữa các môđun và mạng kết nối. Mô hình cấu trúc có thể được biểu diễn bằng nhiều cách.

- Có thể biểu diễn mô hình cấu trúc một cách đơn giản bằng các siêu đồ thị, trong đó mỗi đỉnh của đồ thị tương ứng với các môđun và cung tương ứng với mạng liên kết. Quan hệ liên kết giữa các môđun và mạng được mô tả bằng các ma trận liên kết. Ta có siêu đồ thị tương đương với một đồ thị 2 phần có tập hợp các đỉnh chia làm hai phần, trong đó một phần tương đương với các môđun, phần còn lại tương đương với các mạng.
- Một cách biểu diễn khác của cấu trúc là biểu diễn mỗi môđun bằng các điểm đầu cuối hay là cổng và mô tả sự kết nối giữa các mạng với các cổng của môđun.

Thông thường các ma trận liên kết rất tản mạn, khi đó sử dụng danh sách mạng có hiệu quả hơn để mô tả cấu trúc. Trong danh sách mạng ta đánh số tất cả các mạng nối với từng môđun (danh sách loại này được gọi là danh sách mạng hướng môđun) hoặc đánh số tất cả các mô đun kết nối với một mạng (danh sách mạng hướng mạng).

Ví dụ:

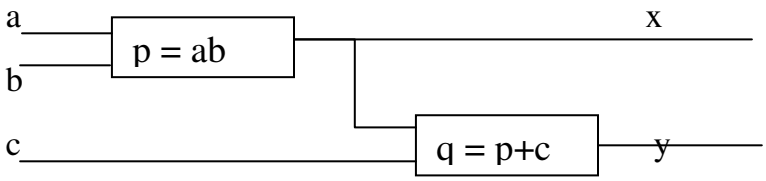
2. Mạng lôgic

Mạng logic tổng quát là một cấu trúc trong đó những mô đun tại lá tương ứng với các hàm logic tuần tự hoặc tổ hợp. Gồm mạng logic tổ hợp và mạng logic đồng bộ.

- Mạng logic tổ hợp hay còn gọi là mạng logic Bool là một cấu trúc phân cấp trong đó:
    - Mỗi mô đun tại lá tương ứng với một hàm logic có nhiều đầu vào và một đầu ra. Hàm này gọi là hàm cục bộ.

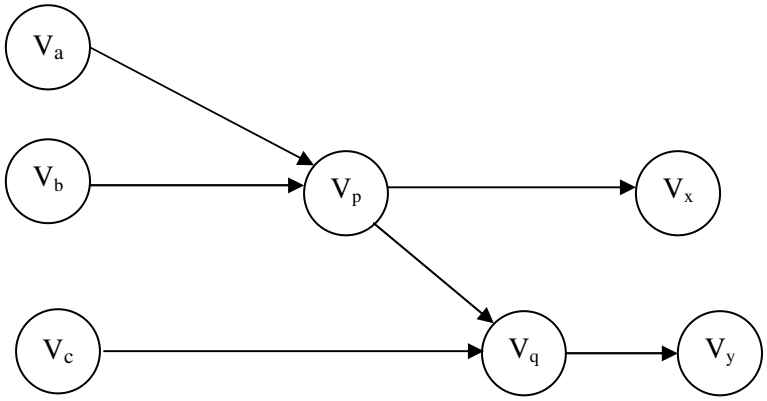
- Các cổng vào ra được chia làm hai nhóm: các cổng vào và các cổng ra. Các cổng không thuộc các mô đun con cũng được chia làm hai nhóm: các đầu vào và các đầu ra sơ cấp.
- Mỗi mạng có một cổng tách biệt gọi là cổng nguồn và có định hướng từ cổng nguồn tới các cổng khác. Các cổng nguồn của mạng có thể hoặc là đầu vào sơ cấp hoặc là đầu ra sơ cấp của mô đun thuộc mức thấp hơn.
- Quan hệ giữa các mạng trong mô đun là quan hệ được sắp một phần

Ví dụ:



Sơ đồ mạng logic (cấu trúc)

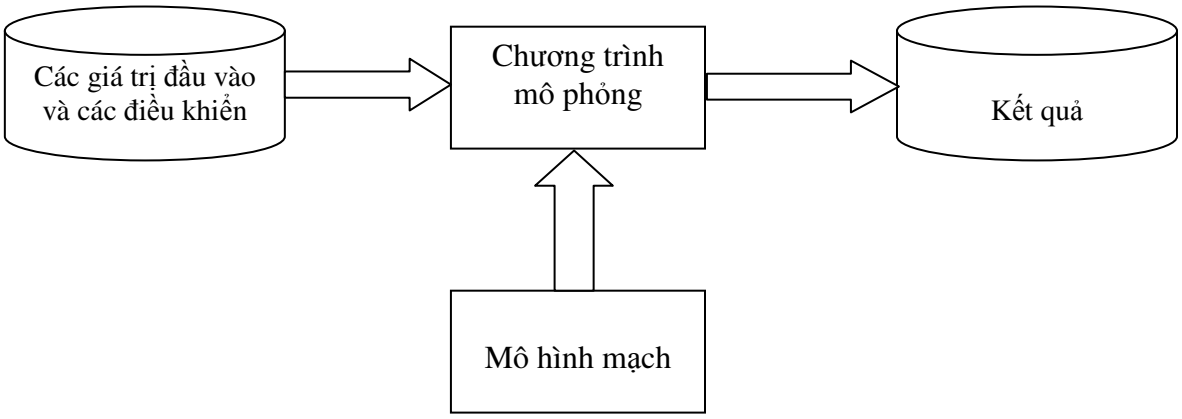
- V_a, V_b, V_c là 3 đỉnh vào
- V_x, V_y là 2 đỉnh ra
- $V_{p,}$ V_q là 2 đỉnh trung gian tương ứng với các hàm logic.



Đồ thị mạng logic của cấu trúc trên.

Chương V: Các phương pháp mô hình hoá logic

Mô hình hóa logic là hình thức kiểm tra thiết kế sử dụng các mô hình của mạch đã được thiết kế. Quá trình mô hình hóa logic và mô phỏng thiết kế có thể được biểu diễn theo sơ đồ sau:



Chương trình mô phỏng sẽ biểu diễn tín hiệu vào và tín hiệu điều khiển, phát triển quá trình tính toán trên các tín hiệu theo thời gian và hình thành các giá trị đầu ra dựa trên mô hình của mạch.

Việc kiểm chứng thiết kế logic là quá trình kiểm tra thiết kế mạch trên phương diện hoạt động về chức năng và theo thời gian. Quá trình này được thực hiện dựa trên việc so sánh các kết quả nhận được qua quá trình mô phỏng với những giá trị được tính toán từ trước dựa vào chức năng. Bên cạnh đó mô hình hóa logic còn có thể sử dụng để kiểm chứng các tính chất sau của hoạt động của mạch được thiết kế:

- Sự độc lập của các trạng thái ban đầu;
- Sự nhạy cảm của các biến (tín hiệu) vào tham số thời gian trễ của các phần tử.
- Trong hoạt động của mạch không tồn tại sự chạy đua giữa các phần tử, sự dao động, các điều kiện đầu vào không thích hợp hoặc các trạng thái treo.

Thông thường nhà thiết kế xây dựng những phiên bản mẫu của mạch theo thiết kế và kiểm tra hoạt động ccuar mẫu. Việc kiểm tra này sẽ cho phép tìm ra những lỗi tiềm ẩn trong thiết kế.

Ưu điểm của việc chọn mẫu là chúng cho phép nhà thiết kế kiểm nghiệm thiết kế theo tốc độ tính toán thực tế. Nhược điểm là giá thành xây dựng phiên bản mẫu thử nghiệm cao và tốn thời gian.

Mô hình hóa logic và mô phỏng thay thế việc xây dựng mẫu thử bằng các phần mềm. Điều này cho phép nhà thiết kế phân tích, kiểm nghiệm và hiệu chỉnh mô hình một cách dễ dàng. So với quá trình kiểm nghiệm trên mẫu, việc kiểm nghiệm thiết kế bằng mô hình có những ưu điểm sau:

- Cho phép kiểm tra các điều kiện sinh ra lỗi (như các mâu thuẫn trên đường tín hiệu)
- Cho phép thay đổi tham số thời gian trễ của các phần tử trong mô hình để kiểm tra những trường hợp xấu nhất về điều phối thời gian trong mạch.
- Kiểm tra những giá trị do nhà thiết kế xác định trong quá trình mô phỏng.
- Cho phép mạch được mô phỏng bắt đầu hoạt động tại bất kỳ một trạng thái nào.
- Cho phép kiểm soát một cách chính xác việc điều phối thời gian đối với những sự kiện không đồng bộ.
- Có khả năng tự động kiểm tra hoạt động của mạch được thiết kế trong môi trường liên kết với những mạch khác.

5.1 Cơ sở mô hình hoá logic

1. Các phương pháp mô hình hóa và các hệ mô phỏng

Trong kỹ thuật thiết kế mạch, mô hình hóa mạch gồm 2 phương pháp chính: Phương pháp mô hình hóa biên dịch và phương pháp mô hình hóa hướng sự kiện.

* Phương pháp mô hình hóa biên dịch:

Các hệ chương trình mô phỏng thực hiện các mô hình logic được dịch từ các ngôn ngữ mô hình hóa phần cứng được gọi là hệ mô phỏng bằng biên dịch. Các mã biên dịch được tạo ra từ những mô hình trên mức thanh ghi, từ các mô hình chức năng hoặc mô hình cấu trúc.

* Phương pháp mô hình hóa hướng sự kiện:

Giả sử ta khảo sát mạch điện khi mạch hoạt động và quan sát những tín hiệu thay đổi giá trị tại những thời điểm thời gian bất kỳ. Những tín hiệu này được gọi là những tín hiệu kích hoạt. Tỷ lệ giữa số lượng các tín hiệu kích hoạt và tổng số các tín hiệu trong mạch gọi là hoạt tính của mạch (khoảng 1% - 5%).

Trong mạch điện, sự thay đổi giá trị của tín hiệu trên một đường truyền tín hiệu được gọi là một sự kiện. Như vậy mỗi khi có sự kiện xuất hiện trên đường tín hiệu I, chúng ta nói rằng phần tử mạch nhận đường tín hiệu I làm đầu vào được kích hoạt. Quá trình tính

toán các giá trị đầu ra của phần tử được gọi là quá trình xác định giá trị tín hiệu. Phương pháp mô phỏng theo hoạt tính của mạch chỉ xác định giá trị tín hiệu đối với những phần tử được kích hoạt. Những phần tử được kích hoạt sẽ thay đổi các giá trị tín hiệu trên đầu ra của chúng và tạo ra các sự kiện mới. Như vậy, hoạt tính của mạch được xác định bởi các sự kiện trên các đường tín hiệu, do đó phương pháp mô phỏng theo hoạt tính còn được gọi là phương pháp mô phỏng hướng sự kiện. Để có thể truyền các sự kiện theo các đường liên kết trong mạch giữa các phần tử, hệ thống mô phỏng hướng sự kiện cần phải biết mô hình cấu trúc của mạch. Do đó mô hình hóa logic và mô phỏng hướng sự kiện thường dựa trên cách lập bảng.

Phương pháp mô hình hóa logic và mô phỏng bằng biên dịch phần lớn chỉ quan tâm tới việc kiểm chứng chức năng hoạt động của mạch mà không quan tâm tới việc điều khiển và điều phối các quá trình tính toán theo thời gian của mạch. Do đó phương pháp mô hình hóa logic và mô phỏng bằng biên dịch thích hợp với những mạch đồng bộ, trong đó, việc điều phối các tiến trình tính toán theo thời gian có thể được kiểm tra tách rời với việc kiểm tra chức năng của mạch.

Ngược lại phương pháp mô hình hóa lôgic và mô phỏng hướng sự kiện tập trung chủ yếu vào các mô hình điều khiển tiến trình tính toán theo thời gian và có thể làm việc với những mô hình thời gian chính xác. Như vậy, phương pháp mô hình hóa logic và mô phỏng hướng sự kiện có tính tổng quát cao hơn và có thể áp dụng cho cả những mạch không đồng bộ.

Phương pháp mô hình hóa logic và mô phỏng hướng sự kiện có thể thao tác với những đầu vào thời gian thực có nghĩa là: những đầu vào có số lần thay đổi trạng thái độc lập với hoạt tính của mạch được mô phỏng.

Trong kỹ thuật hai phương pháp mô phỏng và mô hình hóa này được sử dụng một cách kết hợp, trong đó những thủ tục hướng sự kiện sẽ truyền các sự kiện trên các đường tind hiệu qua các phần tử mạch còn những phần tử được kích hoạt sẽ thực hiện các thao tác lên tín hiệu bằng các mô hình xây dựng từ các mã biên dịch. Chúng ta có thể biểu diễn các mức mô phỏng sau:

- Mô hình hóa trên mức thanh ghi: hệ thống sẽ được mô tả hoàn toàn trên mức thanh ghi truyền đạt hoặc như liên kết giữa những thành phần của mô hình trên thanh ghi.

- Mô hình hóa trên mức chức năng: hệ thống được mô tả bằng các thành phần cơ bản và liên kết giữa các thành phần đó.

- Mô hình hóa trên mức các phần tử logic

- Mô hình hóa trên mức các transistor

- Mô hình hóa hỗn hợp

2. Các giá trị logic không xác định:

Trong quá trình mô hình hóa logic, để có thể mô tả chức năng và hoạt động của mạch theo thời gian, ta phải mô tả một cách chính xác các quá trình diễn ra trong mạch, có nghĩa là chúng ta phải mô tả được sự biến thiên giá trị tín hiệu trên các đường tín hiệu trong mạch. Do hoạt động của các mạch số dựa trên cơ sở của logic chuyển mạch, nhưng với 2 giá trị {0, 1} ta không thể mô tả được sự chuyển tiếp giá trị tín hiệu từ một mức sang một mức khác. Do đó phải mở rộng miền xác định của các phép toán logic truyền thống và mở rộng phép toán này trên miền xác định mới.

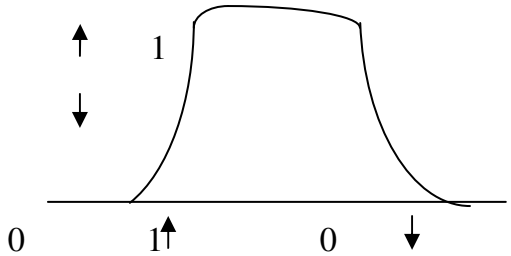
Trong quá trình mô hình hóa, để mô tả các giá trị tín hiệu trên các đường tín hiệu trong các quá trình tính toán, người ta thường sử dụng hai dạng giá trị tín hiệu sau:

- Các giá trị tín hiệu thực – các đối tượng được mô hình hóa trực tiếp và tương ứng với các giá trị tín hiệu trong sơ đồ thực.

- Các giá trị ảo – các giá trị chỉ tồn tại trong mô hình mạch khi thực hiện quá trình mô phỏng.

Trạng thái biến đổi giá trị từ ‘0’ -> ’1’ là

Trạng thái biến đổi giá trị từ ‘1’ -> ’0’ là



Các trạng thái này thể hiện các quá trình quá độ trong mạch trên các đường truyền tín hiệu. Ngoài ra đối với những phần tử có 3 trạng thái ta còn phải sử dụng them giá trị ‘Z’ để chỉ trạng thái cao. Như vậy miền xác định của các phép toán được mở rộng từ tập hợp {0, 1} sang tập {0, 1, ↑, ↓, Z}.

Các giá trị ảo được sử dụng trong quá trình mô hình hóa đối với một số trường hợp ta không thể thiết lập giá trị tín hiệu như là kết quả của các phép toán trên những giá trị thực hoặc khi ta phải mô hình hóa các phần tử của mạch trong điều kiện giá trị thời gian trễ của các phần tử không xác định.

Ví dụ: Khi mô phỏng hoạt động của phần tử NAND có 2 đầu vào bằng các giá trị tín hiệu thực {0, 1, ↑, ↓, Z}, chúng ta không thể thiết lập giá trị đầu ra bằng các giá trị tín

hiệu thực nêu trên khi giá trị đầu vào là '↑' và '↓'. Trong những trường hợp như vậy ta phải sử dụng them các giá trị tín hiệu không xác định ký hiệu là 'U'.

Đáp ứng của các mạch tuần tự đối với các tác động phụ thuộc vào các giá trị khởi tạo ban đầu, do đó ở giai đoạn đầu của quá trình mô phỏng, chúng ta cần xác định những giá trị tín hiệu tương ứng với trạng thái ban đầu của mạch. Khi mạch bắt đầu được cung cấp năng lượng, do giá trị thời gian trễ của các phần tử mạch là các đại lượng ngẫu nhiên nên các trạng thái của các phần tử trigo, thanh ghi, ô nhớ không xác định. Đó là nguyên nhân vì sao trước khi mạch bắt đầu thực hiện các chức năng tính toán thông thường, chúng ta thường đưa mạch về trạng thái ban đầu bằng một chuỗi các khởi tạo 'reset'.

Do đó, trong quá trình mô phỏng, khi cung cấp năng lượng cho mạch, ở thời điểm ban đầu tín hiệu thường được gán giá trị 'X'. Nếu giá trị của một tín hiệu là 'X' tại một điểm xác định có nghĩa là tín hiệu có thể nhận giá trị hoặc '0' hoặc '1' tại thời điểm đó.

Hai giá trị 'X' và 'U' có ý nghĩa khác nhau mặc dù chúng cùng là các giá trị không xác định. 'X' chỉ sự bất định về trạng thái của mạch tại thời điểm ban đầu. Do có sự chạy đua giữa các phần tử logic thành phần và sự biến thiên ngẫu nhiên của tham số thời gian trễ của chúng nên ở thời điểm ban đầu, các phần tử nhớ có thể nhận các giá trị '0' hoặc '1' một cách ngẫu nhiên. Khi đó ta biểu thị trạng thái của mạch là 'X'. Giá trị 'U' xuất hiện khi ta làm các phép toán trên các tín hiệu.

Ví dụ: tại thời điểm t, giá trị tín hiệu S_1 là dãy tín hiệu "↑1", giá trị tín hiệu S_2 là "↓0". Các dãy tín hiệu này biểu thị các quá trình quá độ xảy ra trên đường tín hiệu s_1 và s_2 tại thời điểm t khi giá trị tín hiệu chuyển từ '0' sang '1' và từ '1' về '0'. Nếu các đường tín hiệu s_1 và s_2 là các đầu vào của phần tử AND, khi đó phần tử AND sẽ thực hiện phép toán and('↑', '↓'). Kết quả của phép này là không xác định. Trên đầu ra của phần tử AND tín hiệu sẽ không nhận giá trị '0' cũng như '1'. Trong trường hợp này chúng ta biểu thị giá trị tín hiệu ký hiệu 'U'

Nếu số lượng các giá trị thực và ảo trong quá trình mô hình hóa logic và mô phỏng bằng n thì nhận được mô hình mô phỏng n – giá trị và hệ các phép toán logic tương ứng phải được mở rộng thành hệ logic n – giá trị.

Ví dụ:

- Nếu n = 3, tập hợp các giá trị mà tín hiệu có thể nhận được trong mô hình mô phỏng của mạch sẽ là {0, 1, X};

- Nếu n = 5, tập hợp các giá trị mà tín hiệu có thể nhận được trong mô hình mô phỏng của mạch sẽ là {0, 1, ↑, ↓, U};
- Nếu n = 7, tập hợp các giá trị mà tín hiệu có thể nhận được trong mô hình mô phỏng của mạch sẽ là {0, 1, ↑, ↓, U, X, Z};

5.2 Phương pháp mô hình hoá biên dịch

Phương pháp mô hình hóa logic và mô phỏng bằng biên dịch là phương pháp mô hình hóa trong đó các tín hiệu được xác định giá trị bằng cách tạo cho mỗi phần tử của mạch một mã lệnh tương ứng với các phép toán mà phần tử đó cần được thực hiện. Các mã lệnh nhận được sẽ được biểu diễn theo một trật tự tương ứng.

Trong phương pháp mô hình hóa và mô phỏng bằng biên dịch, mô hình bằng các mã lệnh là một thành phần của hệ thống mô phỏng. Trong những trường hợp đặc biệt, hệ mô phỏng chính là các mô hình trên các mã lệnh. Mô hình mã lệnh được kết nối với hệ mô phỏng , trong đó mỗi tiến trình bao gồm việc đọc các vecto đầu vào, thực hiện mô hình với từng vecto đầu vào và hiển thị kết quả.

Các bước mô hình hóa bằng phương pháp biên dịch:

- Phân hạng các phần tử của mạch cần mô hình hóa theo trật tự thực hiện các phép toán sao cho không có mâu thuẫn nảy sinh.

- Tạo các mã lệnh tương ứng với các phép toán do các phần tử thực hiện.

Để phân hạng các phần tử của mạch, trước tiên phải ngắt các vòng phản hồi nếu có. Các điểm ngắt được xác định tương ứng với các chức năng của mạch. Trong trường hợp mô hình hóa các mạch tuần tự đồng bộ việc ngắt mạch phản hồi thực hiện trên những phần tử trực tiếp nhận các tín hiệu đồng bộ.

Giả sử l(k) là hạng của phần tử k, khi đó quá trình được phân hạng được thực hiện như sau:

- Các đầu vào của mạch có hạng ‘0’

- Nếu $k_1, k_2, \ldots, k_p$ là các phần tử được nối với các đầu vào của phần tử k sẽ bằng:

$$l(k) = 1 + \max(l(k_1), \ldots, l(k_p)).$$

Nhận xét: phương pháp biên dịch không tính tới ảnh hưởng của thời gian trễ khi tín hiệu được truyền qua mạch do quá trình xây dựng mô hình mạch được thực hiện theo hạng.

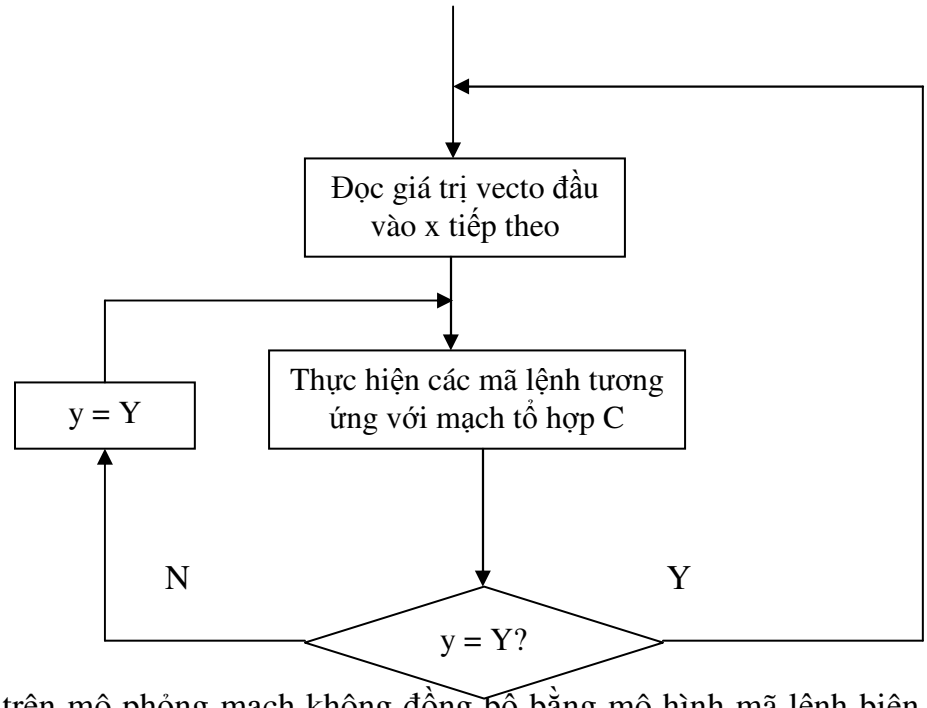
- Nếu ta chỉ giới hạn trường hợp thời gian trễ là các thời gian trễ lan truyền thì việc mô hình hóa hoạt động của có thể thục hiện được theo cách phân hạng. Các thời gian trễ lan truyền sẽ được tính đến một cách tường minh trong quá trình phân hạng và truyền tín hiệu qua từng lớp phân hạng.

- Đối với các dạng thời gian trễ khác như thời gian trễ ngẫu nhiên hoặc thời gian trễ quán tính thì việc mô hình hóa mạch theo phương pháp biên dịch không thể thực hiện được.

- Do ta phải ngắt các vòng phản hồi trong mạch, phương pháp biên dịch chỉ có thể sử dụng trong những trường hợp khi ý nghĩa của việc ngắt vòng phản hồi rõ rang

Ví dụ trong trường hợp các mạch đồng bộ, mô hình hóa mạch bằng phương pháp biên dịch có thể thực hiện được khá nhanh. Nhưng phương pháp này có nhược điểm quan trọng là không tính thời gian trễ trong mạch. Nếu ta coi thời gian trễ trên tất cả các phần tử mạch được coi là bằng nhau thì việc phân hạng sẽ thể hiện được sự trễ của tín hiệu khi đi qua mạch. Trong trường hợp thời gian trễ có những dạng phức tạp thì phương pháp biên dịch không thể thực hiện được chính xác. Khi mô hình hóa các mạch đồng bộ, việc ngắt các vòng phản hồi cũng làm cho ngữ nghĩa của mạch bị thay đổi. Trong trường hợp đối với những mạch không đồng bộ việc không tính đến thời gian trễ có thể dẫn tới những kết quả sai.

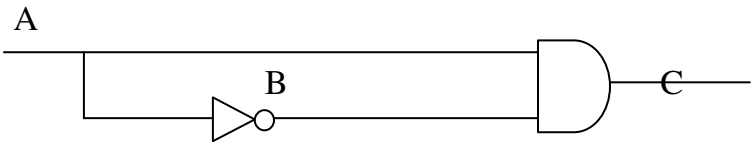
Để giải quyết một phần khó khăn nêu trên, đối với những mạch không đồng bộ, ta phải giả thiết giá trị thời gian trễ chỉ tập trung trên đường phản hồi. Để đáp ứng với vecto đầu vào x, mạch phải trải qua một chuỗi các lần chuyển trạng thái. Quá trình chuyển trạng thái này được biểu diễn bằng sự thay đổi giá trị của biến trạng thái y. Ta giả thiết rằng vector tín hiệu đầu vào chỉ tác động khi trạng thái của mạch ổn định y = Y. Đường tín hiệu phản hồi có hạng ‘0’ phải được xác định trước khi các mã lệnh tương ứng với mạch tổ hợp C được tạo ra.



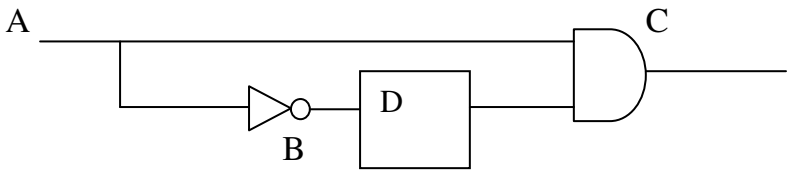
Hình trên mô phỏng mạch không đồng bộ bằng mô hình mã lệnh biên dịch. Khi thực hiện mô phỏng mô hình, quá trình mô hình hóa sẽ thực hiện tính giá trị tín hiệu z và Y dựa trên giá trị x và y.

Phương pháp này không chính xác khi thực hiện đối với các mạch không đồng bộ trong đó các thao tác tính toán đều dựa trên tham số thời gian trễ của các phần tử mạch.

Ví dụ: Ta xét mạch tạo xung sau:



Khi tín hiệu trên đường A có sự thay đổi giá trị từ ‘0’ sang ‘1’, nếu mạch đảo B không làm trễ tín hiệu thì tín hiệu trên C sẽ luôn nhận giá trị ‘0’ do tín hiệu trên hai đường A và B luôn ngược nhau. Nếu phần tử B làm trễ tín hiệu, trong khoảng thời gian có độ dài bằng giá trị thời gian trễ tín hiệu qua phần tử B, hai đầu vào của phần tử AND sẽ có cùng giá trị ‘1’ và trên đường tín hiệu C sẽ xuất hiện xung ‘0’ → ‘1’ → ‘0’ có độ rộng bằng giá trị tham số trễ của phần tử đảo B. Nếu không cẩn thận, xung ‘1’ này sẽ xuất hiện trên đường tín hiệu C bởi vì phương pháp biên dịch chỉ quan tâm tới hành vi tĩnh của mạch (nếu không quan tâm đến sự trễ tín hiệu, giá trị trên đường C luôn bằng ‘0’) Nếu ta them vào phần tử trễ D vào đường tín hiệu tại đầu ra của phần tử B, mạch sẽ được mô phỏng một cách đúng đắn.

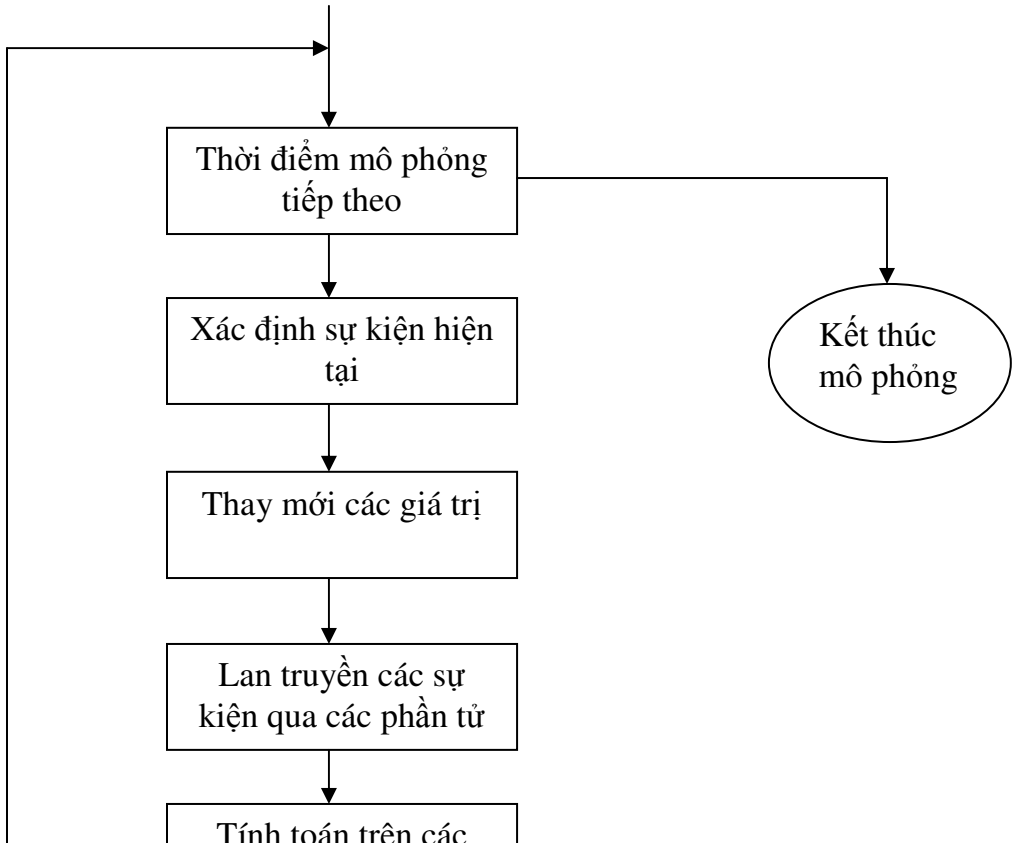


5.3 Phương pháp mô hình hoá hướng sự kiện

Phương pháp mô hình hóa hướng sự kiện cho phép xây dựng những mô hình mô phỏng trên máy tính với độ chính xác cao, trong đó ta có thể mô tả toàn bộ các liên kết trong mạch và dựa vào đó xây dựng các mô hình hoạt động của mạch theo tiến trình thời gian.

Hệ thống mô hình hóa logic và mô phỏng hướng sự kiện sử dụng mô hình cấu trúc của mạch để lan truyền các sự kiện. Sự thay đổi giá trị tại các đầu vào chính của mạch được xác định bằng các vecto tín hiệu kích hoạt. Mọi sự kiện trên các đường tín hiệu khác của mạch được tính toán theo các phần tử bị kích hoạt.

Các sự kiện xuất hiện tại những thời điểm thời gian mô phỏng xác định. Cơ chế điều phối thời gian cuả quá trình mô phỏng điều khiển sự xuất hiện của các sự kiện theo một trật tự xác định. Các tác động vào mạch sẽ được biểu diễn bằng dãy các sự kiện xuất hiện trên các đường tín hiệu tại các thời điểm thời gian xác định. Các sự kiện sẽ xuất hiện trong tương lai sẽ phải chờ và lưu trong danh sách các sự kiện. Những sự kiện trong danh sách sự kiện sẽ được điều phối và xử lý tại những thời điểm mô phỏng.



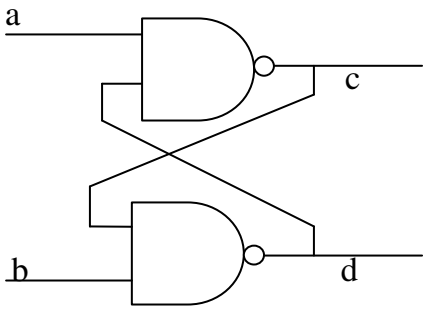
Để mô tả rõ quá trình mô hình hóa logic và mô phỏng hướng sự kiện, cụ thể hóa thuật toán mô phỏng hướng sự kiện như sau:

V_0: giá trị tín hiệu- đường truyền tín hiệu thay đổi ở thời điểm thời gian hiện tại.

V_1: giá trị tín hiệu – đường truyền tín hiệu thay đổi dưới tác dụng của V_0.

1. Thiết lập hai tập hợp V_a và V_b, gán a←0, b←1
2. Thiết lập giá trị trạng thái ban đầu của mạch
3. Đọc các giá trị đầu vào. Xét các đường tín hiệu mà trên đó xuất hiện các sự kiện (sự thay đổi giá trị tín hiệu). Các giá trị của sự kiện xảy ra này được ghi vao tập hợp V_a.
4. Kiểm tra tập hợp V_a, nếu V_a rỗng quay về bước 3; nếu V_a không rỗng, thực hiện bước 5.
5. Thực hiện mô hình hóa logic đối với các giá trị trong tập hợp V_a; các đường dữ liệu trên đó xuất hiện sự kiện và giá trị của chúng được ghi vào V_b.
6. Gán a←1, b←0, vai trò của V_a và V_b thay đổi. Kiểm tra sự xuất hiện dao động trong mạch. Nếu không có dao động quay về bước 4.

   Ví dụ: Mô hình hóa quá trình hoạt động của trigo RS



1. Thiết lập V_A và V_B (A←0, B←1);
2. Do trạng thái ban đầu của mạch không xác định, giá trị trên tất cả các đường tín hiệu được gán bằng ‘X’;

3. Đặt các giá trị đầu vào: a = 0, b = 1. Ta có V_A = {(a,0)(b,1)};
4. Ta thấy V_0<>rỗng, chuyển sang bước 5
5. Giá trị trên đường tín hiệu ra c thay đổi từ 'X' sang '1', tập hợp V_1 = {(c,1)};
6. Gán V_1 vào V_0. Tập hợp V_0 nhận giá trị V_0 = {(c,1)} và quay lại bước 4
7. Ta thấy V_0<> rỗng, chuyển tới bước 5
8. Giá trị tín hiệu trên đường d chuyển từ 'x' sang '0', V_1 = {(d,0)}
9. Gán V_1 vào V_0, V_0 = {(d,0)}, quay về bước 4
10. Ta thấy V_0<> rỗng, chuyển sang bước 5
11. Gán V_1 vào V_0, khi đó V_0 = rỗng, chuyển sang bước 4
12. Ta thấy V_0 = rỗng, quay về bước 3
13. Đọc các giá trị tiếp theo ở đầu vào

Như vậy với giá trị đầu vào a = '0', b = '1', ta nhận được các giá trị đầu ra c = '1', d = '0'.

5.4 Mô hình hoá quá trình trễ tín hiệu trong các phần tử mạch

Khi thực hiện mô hình hóa logic và mô phỏng hoạt động của mạch trên mức các phần tử logic, một nhân tố quyết định mức độ chính xác của mô hình so với mạch thực tế là tham số thời gian trễ của các phần tử logic. Nếu ta biểu diễn hoạt động của mạch không tương ứng với những tình huống thực tế sẽ diễn ra trong các phần tử logic, khi đó sẽ xuất hiện sự không chính xác trong các mô hình hoạt động của mạch chủ yếu theo những quan hệ về thời gian.

1. Mô hình hóa quá trình trễ tín hiệu qua các phần tử logic:

Mỗi phần tử logic đều tác động lên các tín hiệu vào và làm trễ các tín hiệu đó. Trong kỹ thuật thường sử dụng hai mô hình trễ qua các phần tử logic: mô hình trễ lan truyền và mô hình trễ quán tính

- Trễ lan truyền: là sự trễ tín hiệu phát sinh khi ta cho tín hiệu đi qua phần tử mạch.

- Trễ quán tính: là sự trễ tín hiệu gắn liền với năng lượng để kích hoạt phần tử mạch.

a. Trễ lan truyền:

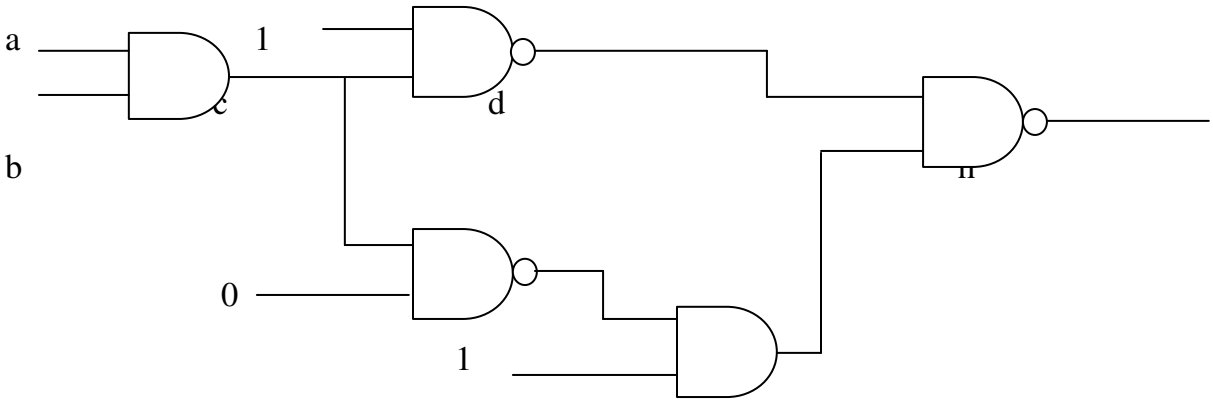
Thời gian trễ lan truyền là thời gian mà tín hiệu ra của mạch chậm pha so với tín hiệu vào mạch. Mô hình trễ lan truyền là mô hình cơ bản, trong đó xác định khoảng thời gian Δ_T

cách biệt giữa các sự kiện xuất hiện tại đầu ra với sự kiện xuất hiện tại đầu vào sinh ra chúng. Để đơn giản, các giá trị thời gian trễ sử dụng trong mô phỏng là các số nguyên.

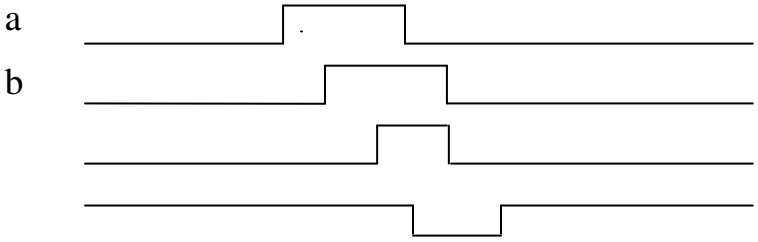
Trong một số trường hợp khi mô hình hóa mạch ta không tính đến thời gian trễ tín hiệu khi đi qua các phần tử. Lúc đó ta coi thời gian trễ của các phần tử bằng không, chúng ta sử dụng mô hình với thời gian trễ tín hiệu bằng không. Phương pháp mô hình hóa logic và mô phỏng bằng biên dịch sử dụng mô hình thuộc tính trễ này. Mô hình các phần tử với thời gian trễ bằng không chỉ được sử dụng để mô hình hóa các giá trị logic trong những mạch tổ hợp và mạch tuần tự đồng bộ.

Mô hình mạch trong đó thời gian trễ của tất cả các phần tử logic bằng nhau gọi là mô hình với thuộc tính trễ thuần nhất. Do các phần tử logic có thời gian trễ khác không nên chúng ta có khả năng mô hình hóa và xử lý các mạch không đồng bộ có phản hồi. Trong một số trường hợp ta có thể chọn giá trị thông số trễ làm đơn vị với tỷ lệ một, mô hình này được gọi là mô hình trễ đơn vị.

Trong trường hợp tổng quát, các phần tử của mạch có thể nhận những thời gian trễ khác nhau. Để thực hiện quá trình mô hình hóa hoạt động của mạch, ta sử dụng đơn vị thời gian để đo giá trị thông số thời gian trễ là ước số chung lớn nhất Δ_T của các thông số thời gian trễ trên các phần tử và phân phối các giá trị thông số thời gian trễ theo tỷ lệ tương ứng với Δ_T. Mô hình này được gọi là mô hình hoạt động với thông số thời gian trễ phân tán.



Sơ đồ mạch cho trường hợp trễ thuần nhất và trễ phân tán



c

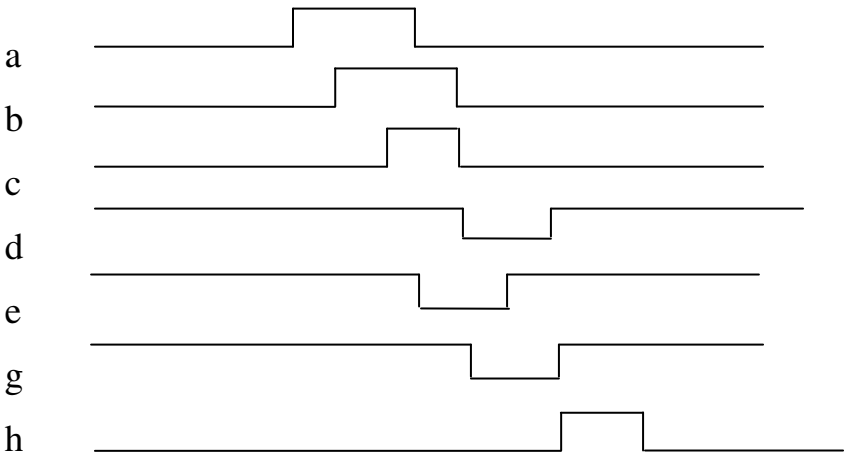
d

e

g

h

Hoạt động của mạch trên với mô hình trễ thuần nhất.

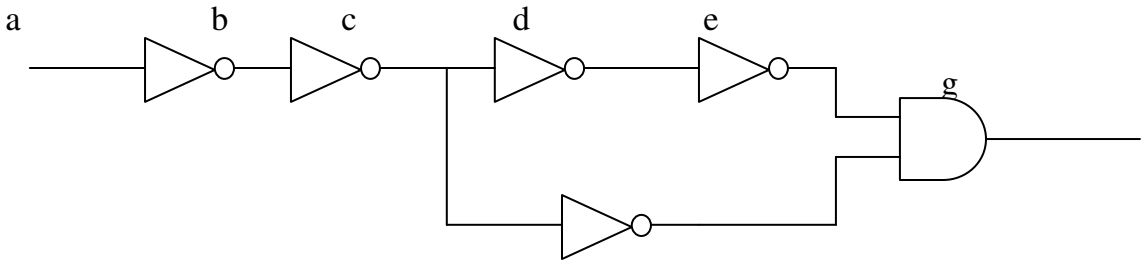


Hoạt động của mạch với mô hình trễ phân tán

Khi mô hình hóa hành vi của các phần tử logic, thường phân tách chức năng tính toán logic cơ chế điều phối thời gian của chúng. Như vậy một phần tử với thông số thời gian trễ là Δ_T sẽ tương ứng với phần tử logic có thông số thời gian trễ bằng 0 và phần tử trễ có thông số thời gian trễ là Δ_T. Trong quá trình mô hình hóa hành vi của phần tử logic, đầu tiên phần tử bị kích hoạt sẽ được tính toán theo chức năng logic, sau đó các thuộc tính trễ sẽ được tính toán và thể hiện qua quá trình điều phối thời gian.

Khi sử dụng mô hình trễ không xác định hoặc mô hình trễ cực đại – cực tiểu , trong hoạt động của mạch sẽ xuất hiện những hành vi không xác định và biểu thị qua những giá trị ‘U’.

Ví dụ:



Với mô hình trễ cực đại – cực tiểu ta có thể mô hình hóa phần lớn các trường hợp phức tạp xuất hiện trong hoạt động của mạch trên thực tế.

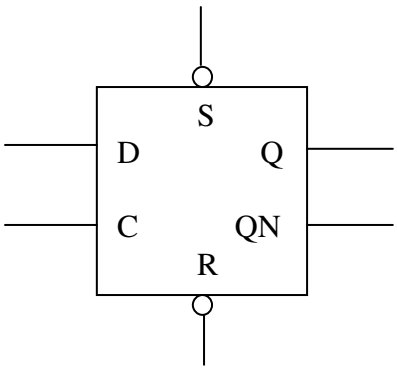
b. Trễ quán tính:

Mọi mạch điện đều cần năng lượng để chuyển trạng thái. Năng lượng của tín hiệu là hàm số của biên độ và độ dài tín hiệu. Nếu độ đài của tín hiệu quá ngắn, tín hiệu không thể kích hoạt để phần tử chuyển trạng thái. Độ dài tối thiểu của tín hiệu đầu vào đủ để kích hoạt mạch chuyển trạng thái được gọi là thời gian trễ quán tính đầu vào của phần tử, ký hiệu là Δ_I. Những tín hiệu có độ dài nhỏ hơn Δ_I sẽ được gọi là xung nhọn và sẽ không được phần tử cho đi qua. Nếu độ dài tín hiệu lớn hơn hoặc bằng Δ_I, tín hiệu sẽ được đi qua mạch với độ trễ bằng thời gian trễ lan truyền của phần tử. Việc đưa trễ quán tính đầu vào cho phép mô hình hóa những trường hợp đặc biệt trong các sơ đồ thực khi phần tử ngừng làm việc với những xung rất hẹp.

2. Mô hình hóa quá trình trễ tín hiệu qua các phần tử chức năng và thanh ghi.

Các chức năng logic và các đặc tính thời gian của các phần tử chức năng phức tạp hơn so với các phần tử logic cơ bản.

Ví dụ: xét hành vi hoạt động của phần tử Trigo D làm việc theo sườn lên với hai đường tín hiệu không đồng bộ thiết lập S (set) và khởi tạo R (Reset).



q	S	R	C	D	Q	QN	Tg Trễ
0	0	1	x	x	1	0	$\Delta_{S/Q} = 4$; $\Delta_{S/QN} = 3$
1	1	0	x	x	0	1	$\Delta_{R/Q} = 3$; $\Delta_{R/QN} = 4$
1	1	1	↑	0	0	1	$\Delta^f_{C/Q} = 8$; $\Delta^r_{C/QN} = 6$
0	1	1	↑	1	1	0	$\Delta^r_{C/Q} = 6$; $\Delta^f_{C/QN} = 8$

x	0	0	x	x	u	u	

$\Delta_{I/O}$: chỉ độ trễ của đáp ứng tại đầu ra ‘O’ đối với tác động tới đầu vào ‘I’

Δ^r, Δ^f: độ trễ sườn lên và sườn xuống của phần tử.

Ta xét dòng thứ 3: nếu trạng thái ban đầu của phần tử q = ‘1’ và tại các đường tín hiệu S, R không có tác động (S = ‘1’, R = ‘1’), sự chuyển trạng thái của xung đồng hồ C từ ‘0’ sang ‘1’ sẽ làm cho đầu ra Q nhận giá trị của đường D với thời gian trễ sườn xuống $\Delta^f_{C/Q}$ = 8; đầu ra QN chuyển từ ‘0’ sang ‘1’ với độ trễ $\Delta^r_{C/QN}$ = 6.

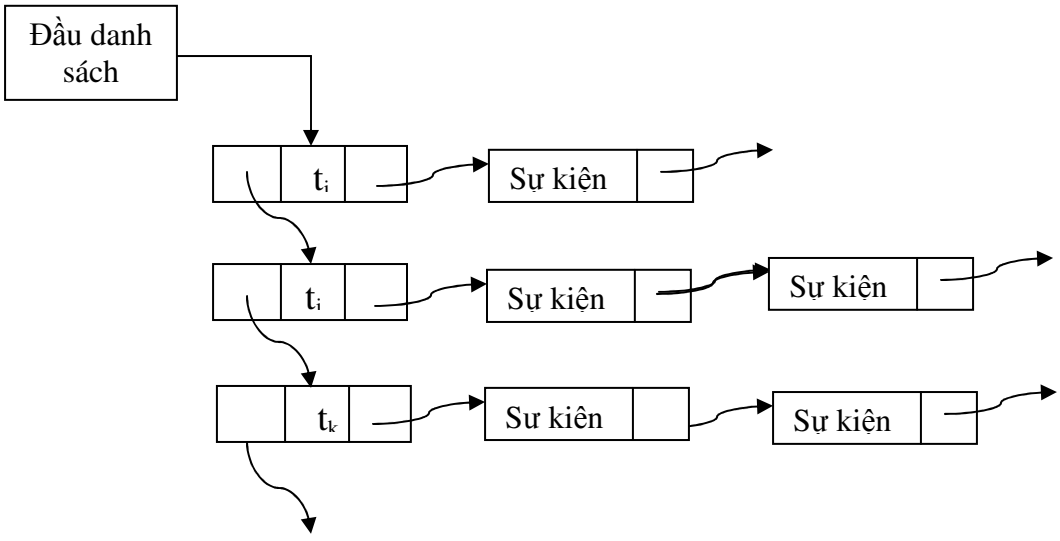
Dòng cuối cùng của bảng chỉ ra rằng nếu các đầu vào nhận giá trị cấm “00”, cả hai đầu ra sẽ nhận giá trị ‘U’.

5.5 Mô hình hoá trên mức các phần tử

Các phương pháp điều khiển quá trình thể hiện trình tự thực hiện mô hình hóa và những phương pháp xử lý gắn liền với quá trình mô phỏng mạch. Trong phương pháp biên dịch, trình tự thực hiện mô hình hóa được xác định bằng việc phân hạng các phần tử . Như vậy việc ngắt các vòng phản hồi trong các mạch tuần tự và phân hạng phần tử chính là phương pháp điều khiển quá trình mô hình hóa.

Khi sử dụng phương pháp mô hình hóa hướng sự kiện nếu các giá trị tín hiệu v’(j) tại thời điểm thời gian t_0 khác giá trị V(j) tại thời điểm trước, điều đó có nghĩa là xuất hiện sự kiện, ta sẽ coi rằng tín hiệu thay đổi giá trị vào thời điểm $t_0 + \Delta_T$ trong đó Δ_T là độ trễ lan truyền của phần tử đang xét.

Để mô tả sự xuất hiện sự kiện trên các đường tín hiệu trong mạch theo thời gian ta chứa các sự kiện vào danh sách tuyến tính có dạng biểu diễn như sau:

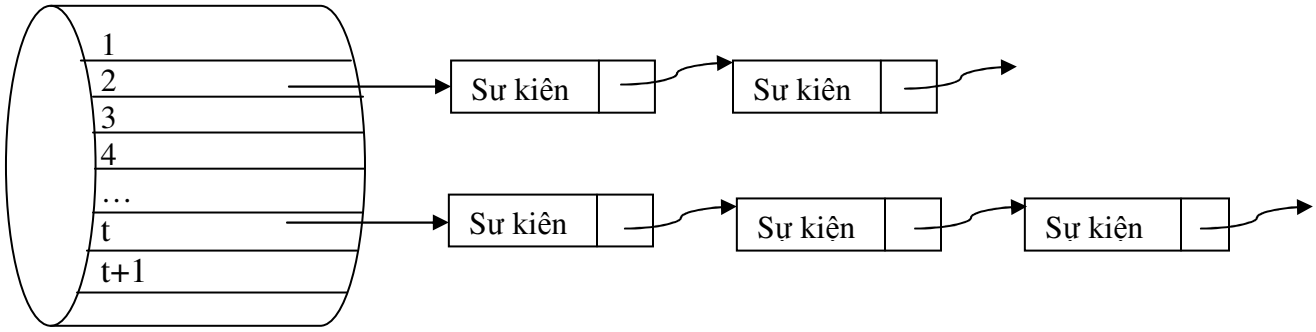


Danh sách các sự kiện

Tương ứng với mỗi một thời điểm thời gian sẽ có một danh sách các sự kiện xuất hiện vào thời điểm đó.

Khi có sự kiện mới xuất hiện, sự kiện này được đưa vào danh sách sự kiện gắn liền với một thời điểm thời gian xác định. Sự xuất hiện của một thời điểm thời gian được đánh dấu trong quá trình mô hình hóa không phụ thuộc vào chuỗi thời gian. Do đó để xác định vị trí của các thời điểm đã được tính toán trước trong danh sách sự kiện, chúng ta cần phải có chuỗi thời gian tương ứng với danh sách được xử lý. Điều này có thể được thực hiện dựa vào các ánh xạ thời gian.

Khi ta sử dụng phép ánh xạ thời gian, dãy các thời điểm thời gian được xác định với khoảng thời gian cách đều Δ, mọi sự kiện trong hệ thống sẽ được xác định dựa vào các thời điểm thời gian này. Khoảng thời gian Δ được xác định bằng ước số chung lớn nhất của thời gian truyền tín hiệu và độ trễ của các phần tử trong mạch. Do giới hạn vệ bộ nhớ dung để thực hiện quá trình mô hình hóa mạch, ta xác định giá trị giới hạn của thời điểm thời gian cực đại bằng M. Như vậy các thời điểm thời gian sau thời điểm cực đại sẽ quay lại bắt đầu từ thời điểm ban đầu và tạo thành vòng quay thời gian (bánh xe thời gian).

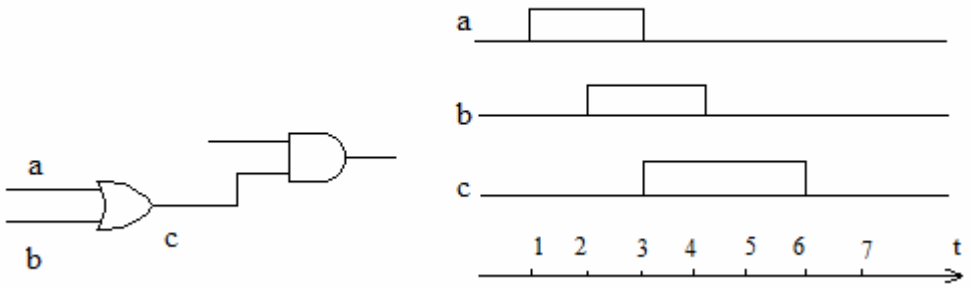


Thông thương khoảng thời gian Δ tương thích với độ trễ của các phần tử trong mạch. Trong những trường hợp khi trong mạch có những phần tử với độ trễ lớn, khoảng thời gian Δ có thể vượt quá thời gian giới hạn M, điều này dẫn tới việc thời điểm mô phỏng vượt ra ngoài vòng thời gian và gây nên hiện tượng tràn. Để giải quyết vấn đề này, ta xây dựng danh sách bổ trợ để lưu trữ các thời điểm tràn và gọi là danh sách tràn. Trong trường hợp sử dụng danh sách tràn, mỗi khi bánh xe thời gian thực hiện được một vòng quay, ta cần thực hiện thao tác trả các thời điểm trong danh sách tràn về chu trình xử lý theo vòng thời gian.

Khi thực hiện quá trình mô hình hóa theo vòng thời gian, các bước của quá trình mô phỏng có thể được biểu diễn dưới dạng sau:

- B1: Thiết lập các giá trị ban đầu của tín hiệu tại thời điểm t←t_0 và đặt các giá trị đó thành giá trị hiện thời.
- B2: Đọc các giá trị đầu vào. Nếu các giá trị này khác các giá trị hiện thời, ta đưa chúng vào tập hợp L_τ trong đó $\tau = t + \Delta_T$.
- B3: Nếu $L_\tau = Ø$, chuyển tới bước 5, ngược lại chuyển tới bước 4
- B4: Nếu giá trị trong L_τ trùng với giá trị hiện thời, ta giữ nguyên giá trị hiện thời; nếu các giá trị này khác nhau, ta làm giá trị cập nhật hiện thời và đưa vào L_τ, $\tau = t + \Delta_T$
- B5: Gán t←t=1 và quay về B2.

  Ví dụ: Minh họa thuật toán mô hình hóa hướng sự kiện đối với mạch sau:



Ví dụ minh họa thuật toán mô hình hóa hướng sự kiện đối với mạch trên. Các giá trị '1' và '0' trên một đường tín hiệu bất kỳ ; ví dụ trên đường a, sẽ được ký hiệu là (a;1) và (a;0). Giả thiết các tín hiệu đầu vào thay đổi theo giản đồ thời gian trên, phần tử OR có độ trễ lan truyền $\Delta_T = 2$. Trong trường hợp này quá trình mô hình hóa hướng sự kiện sẽ được thực hiện theo trình tự sau:

1. B1: Tại thời điểm ban đầu t=t_0 (t_0 = 1); Các giá trị ban đầu: (a; 0), (b; 0), (c; 0);
2. B2: Đọc giá trị đầu vào: (a; 1); $L_{\tau+\Delta T} = L_3 = \{(c; 1)\}$;
3. B3: Kiểm tra L_1, L_1=Ø, chuyển tới bước 5;
4. B5: t←t+1(t = 2); Quay lại B2;
5. B2: Đọc giá trị đầu vào (b; 1); $L_{\tau+\Delta T} = L_4 = \{(c; 1)\}$;
6. B3: Kiểm tra L_2, L_2 = Ø, chuyển tới B5;
7. B5: t←t+1(t = 3); Quay lại B2;
8. B2: Đọc giá trị đầu vào (a; 0);
9. B3: Kiểm tra L_3, L_3={(c; 1)}, $L_3 \neq Ø$; Thực hiện B4;

10. B4: Giá trị hiện thời được thiết lập bằng (c; 1);
11. B5: t←t+1(t = 3); Quay lại B2;
12. B2: Đọc giá trị đầu vào (b; 0); L_6 = {(c; 0)};
13. B3: Kiểm tra L_4, L_4 = {(c; 1)}, $L_4 \neq Ø$; Thực hiện B4;
14. B4: Do L_4 trùng với giá trị hiện thời (c; 1); thực hiện B5;
15. B5: t←t+1(t = 5); Quay lại B2;
16. B2: Các tín hiệu vào không thay đổi giá trị; Thực hiện B3;
17. B3: Kiểm tra L_5, $L_5 \neq Ø$; chuyển tới B5;
18. B5: t←t+1(t = 6);
19. B2: Các tín hiệu vào không thay đổi giá trị; Thực hiện B3;
20. B3: Kiểm tra L_6, $L_6 \neq Ø$. Thực hiện B4;
21. B4: Giá trị hiện thời được thiết lập bằng (c; 0);

Chương VI: Ngôn ngữ mô hình hoá VHDL

6.1 Giới thiệu về ngôn ngữ VHDL

1. Những khái niệm chung về ngôn ngữ VHDL

Các phương pháp thiết kế dựa trên cơ sở của các ngôn ngữ HDL ngày càng trở nên phổ biến. Các ngôn ngữ mô tả phần cứng HDL được các nhà thiết kế mạch sử dụng chủ yếu để mô tả cấu trúc hoặc hành vi của các hệ thống số cho quá trình mô phỏng hoặc thiết kế.

Phương pháp thiết kế trên cơ sở các ngôn ngữ HDL so với các phương pháp thiết kế truyền thống trên cơ sở của các cổng lôgic có các ưu điểm sau:

- Các phương pháp này cho phép tăng năng suất thiết kế, nó cho phép nhà thiết kế tốn ít thời gian hơn và cho phép những người không cần kiến thức sâu về phần cứng có thể thiết kế phần cứng.

-Phương pháp thiết ôkế trên các ngôn ngữ HDL khá cơ động với những công nghệ khác nhau. Các m tả trên các ngôn ngữ HDL cung cấp các tài liệu độc lập với phần cứng của mạch điện. Sử dụng các chương trình tiện ích hỗ trợ thiết kế ta có thể chuyển đổi các biểu diễn trên các ngôn ngữ HDL thành nhiều mức ứng dụng cho những công nghệ khác nhau.

Ngôn ngữ HDL (VHSIC Hardware Description Language) là ngôn ngữ được sử dụng phổ biến trong công nghệ chế tạo các mạch VLSI. VHDL được công nhận là ngôn ngữ tiêu chuẩn trong mô tả phần cứng của IEEE và của Bộ Quốc phòng Mỹ.

Về mặt cú pháp, ngôn ngữ VHDL là một ngôn ngữ được định kiểu chặt chẽ và có một tập hợp lớn các câu lệnh. Ngôn ngữ VHDL hỗ trợ các phương pháp mô tả nhiều lớp trong đó các thành phần cấu trúc hoặc mạng lưới các phần tử có thể đi đôi với các mô tả hành vi hoặc các thuật toán.

VHDL cung cấp khả năng mô tả của mạch số trên những mức độ trừu tượng khác nhau: mức thuật toán; mức các thạnh ghi, hàm truyền đạt; mức các cổng lôgic. Nhà thiết kế có thể sử dụng chiến lược thiết kế từ trên xuống, đầu tiên mô tả thiết kế trên mức kiến trúc, sau đó chi tiết hóa từng bước thiết kế. Ví dụ ta có thể mô tả mạch so sánh một bit bằng ngôn ngữ VHDL theo những mức độ chi tiết khác nhau. Mạch này bao gồm hai

đầu vào và một đầu ra với các tín hiệu tại các đầu này là các tín hiệu số. Như vậy mạch sẽ tương ứng với một thực thể có hai đầu vào và một đầu ra. Kiến trúc của mạch đặc tả quan hệ giữa đầu vào và đầu ra của mạch có thể mô tả theo hành vi, dòng truyền dữ liệu qua mạch, hoặc theo cấu trúc của mạch.

- Mô tả trên mức thực thể

Trên mức thực thể, ta mô tả về số lượng các cổng vào ra của mạch và các dạng tín hiệu tại các cổng đó. Trong ví dụ về mạch so sánh một bit, mạch được mô tả bằng một thực thể bao gồm hai cổng vào và một cổng ra. Các dữ liệu tại các cổng này là các bit.

```
entity COMPARE  is
        port ( A,B: in BIT;C:out BIT);
end COMPARE;
```

- Mô tả mạch băng hành vi

Để mô tả mạch bằng hành vi, trong ngôn ngữ VHDL người ta dùng cấu trúc process. Khi mô tả bằng hành vi, ta không cần thiết phải cung cấp chi tiết về việc thực hiện thiết kế. Trong ví dụ này ta thấy mạch so sánh được mô tả bằng một quá trình process. Quá trình này chịu tác dộng của hai tín hiệu là A và B.

```
ahitechture BEHAVIOR or COMARE is
begin
  process (A,B)
  begin
        if (A=B) then C<= '1';
        else C<='0';
        end iF
  end process;
end BEHAVIOR;
```

-Mô tả bằng dòng dữ liệu

Theo biểu diễn bằng dòng dữ liệu, hệ thống được biểu diến như các luồng chuyển động của các tín hiều điểu khiển và các dự liệu. Theo phương pháp này, hoạt động của mạch được biểu diến như các mạch lôgic tổ hợp, như mạch cộng, mạch so sánh, mạch giải mã. Đối với ví dụ về mạch so sánh, ta thấy tín hiệu C được gán giá trị của biểu thức lôgic của A và B sau một thời gian 10 ns sau khi có sự thay đổi giá trị tín hiệu trên các cổng A,B.

```
architechture DATAFLOW of COMPARE is
  begin
  C<= not (A or B) after 10 ns;
end DATAFLOW;
```

- Mô tả mạch bằng cấu trúc

Đối với cách mô tả mạch qua cấu trúc, ta phải mô tả các thành phần cấu trúc và mối liên kết giữa các thành phần đó. Trong ví dụ về mạch so sánh cấu trúc của mạch gồm một phần tử XOR có hai đầu vào I0,I1, kết nối với một phần tử NOT. Đầu ra của phần tử XOR nối với đầu vào của phần tử NOT. Cấu trúc kết nối trên được mô tả bằng đoạn chương trình VHDL.

```
Architecture STRUCTURE of COMPARE is
    component XOR_Gate
    port (I0,I1: in BIT; O: out BIT);
  end component;
component oNOT_Gate
    port (I0: in BIT; O: out BIT);
end component;
signal NET_I: BIT;
```

begin

U0: XOR_Gate **port map** (I0 => A,I1 =>B, O =>NET_I);

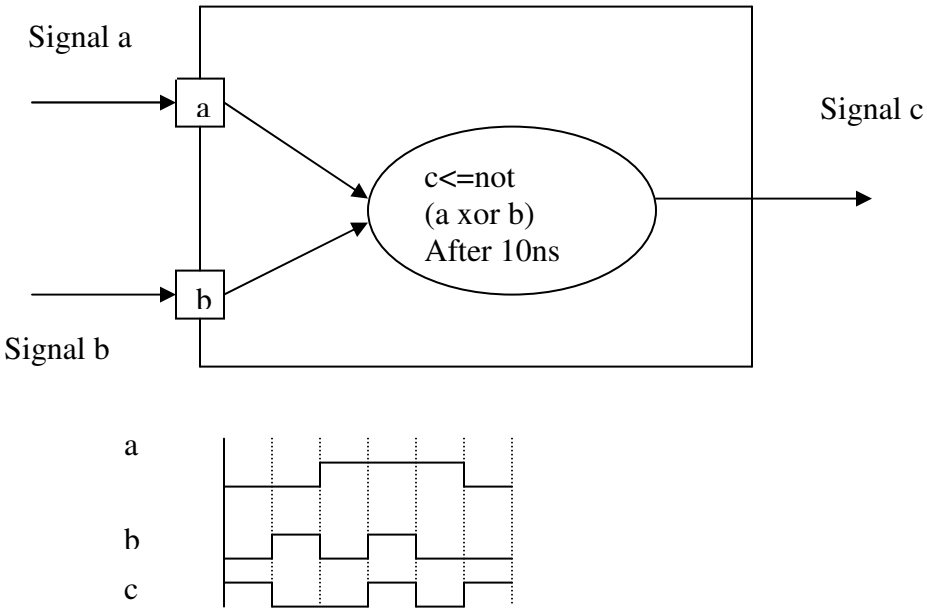
U1: NOT_Gate **port map** (I0 => NET_I, O => C);

end STRUCTURE;

2.Qúa trình mô phỏng

Như đã đề cập tới trong chương 4 ,các ngôn ngữ HDL luôn đi kèm với các bộ mô phỏng. Các thiết kế trên ngôn ngữ VHDL, sẽ thực hiện trên bộ mô phỏng VHDL để kiểm tra hành vi của hệ thống được mô hình hóa. Để tái tạo hoạt động của mô hình, nhà thiết kế cần cung cấp tập hợp các tác động vào mô hình. Chương trình mô phỏng sẽ gắn các tác động đó tới đầu vào của mô hình tại những thời gian xác định và dựa vào mô hình đưa ra các đáp ứng của mạch. Các kết quả đó được nhà thiết kế sử dụng để kiểm tra mức độ thích hợp của thiết kế.

Ta có thể thực hiện mô phỏng trên bất kỳ giai đoạn nào của quá trình thiết kế. Tại các mức bao quát nhất của thiết kế mô phỏng cung cấp cho ta thông tin về hoạt động của mạch đang thiết kế. Thông thường mô phỏng ở mức này thực hiện rất nhanh và không cung cấp cho ta những thông tin chi tiết về hoạt động của mạch và định chế độ định thời gian. Mô phỏng ở mức thấp chiếm nhiều thời gian hơn nhưng cung cấp cho ta nhiều thông tin chi tiết về hoạt động của mạch, chế độ thời gian đồng hồ. Ngôn ngữ VHDL cho phép sử dụng cơ chế mô phỏng hỗn hợp. Ưu điểm của phương pháp mô phỏng hỗn hợp là cho phép nhà thiết kế tập chung và xây dựng những phần mạch quan trọng. Để giảm giá thành thực hiện mô phỏng ở các mức thấp các bộ mô phỏng cần được dùng để phát hiện các lỗi càng sớm càng tốt. Trong quá trình mô phỏng chương trình VHDL, các nhà thiết kế cần cung cấp tập hợp các giá trị thử nghiệm tại những thời điểm mô phỏng xác định



Hình 6.1: Mô hình bộ so sánh 1 bít
và các tín hiêu qua mô phỏng

3. Tổng hợp mạch

Tổng hợp mạch là quá trình xây dựng các mô tả thiết kế từ một mức trừu tượng nào đó sang một mức trừu tượng thấp hơn. Quá trình này có thể là sự biến đổi từ hành vi này sang hành vi khác hoặc từ hành vi sang cấu trúc. Quá trình biến đổi này tương tự như quá trình biên dịch chương trình phần mềm viết trên các ngôn ngữ bậc cao sang các mã Assembly. Các đầu vào của các công cụ thiết kế thường là các mô tả trên các ngôn ngữ HDL, các cơ chế điều khiển thời gian các mục tiêu tối ưu, các thư viện kỹ thuật. Đầu ra của hệ thống hỗ trợ thiết kế là các danh sách các mạng lưới tối ưu, các hiệu năng của mạch, diện tích của thiết kế được xây dựng.

-Tổng hợp hành vi là quá trình biến đổi các mô ta bằng các ngôn ngữ thủ tục sang các mô tả ở mức thanh ghi truyền đạt. Thiết kế ở mức thanh ghi truyền đạt thường bao gồm các đường truyền dữ liệu, các mạch nhớ và các bộ điều khiển. Quá trình tổng hợp hành vi thường được gọi là quá trình tổng hợp ở mức cao hay còn gọi là tổng hợp kiến trúc.

-Tổng hợp ở mức thanh ghi truyền đạt là quá trình tạo ra cấu trúc mạng cho các mạch tuần tự từ tập hợp các hàm truyền đạt thanh ghi. Các trạng thái

tương ứng với chế độ đồng hồ cũng được xác định ở mức này. Các thao tác ở mức thanh ghi truyền đạt có thể mô tả bằng các ôtômat hữu hạn hoặc tập hợp các phương trình truyền đạt ở mức thanh ghi các thành phần của quá trình bao gồm : tối ưu hóa trạng thái, mã hóa trạng thái, tối ưu hóa lôgic , ánh xạ công nghệ.

-Tổng hợp lôgic là quá trình chuyển các biểu diễn mạch bằng các biểu diễn lôgic sang các mạch lôgic. Quá trình tối ưu hóa mạch lôgic thiết kế được chia làm 2 giai đoạn: các quá trình tối ưu mạch không phụ thuộc vào công nghệ; và các ánh xạ các sơ đồ và các phần tử vào các liên kết của các phần tử mô tả trong bộ thư viện.

6.2 Các cấu trúc cơ sở trong VHDL

Mỗi một hệ thống mạch số được thiết kế như một hệ phân cấp các môdun. Mỗi môdun tương ứng với một thực thể của thiết kế trên ngôn ngữ VHDL. Thực thể thiết kế thể hiện một đối tượng của thiết kế phần cứng. Đối tượng này có các đầu vào và đầu ra được xác định rõ ràng, đồng thời thực thể thiết kế cũng phải chỉ ra được chức năng của đối tượng thông qua phép toán được định nghĩa trước. Mỗi thực thể thiết kế gồm có 2 phần: phần khai báo thực thể và kiến trúc thực thể.

- Phần khai báo thực thể mô tả dạng bên ngoài của thực thể, các giao diện của thực thể với thực thể khác. Giao diện này được thể hiện qua các cổng vào và cổng ra của thực thể.

- Kiến trúc thực thể mô tả các thành phần bên trong của thực thể.

- Ngoài ra chúng còn có thể dùng các gói tiện ích để thiết kế.

- Các thực thể còn có cấu hình. Cấu hình là các dạng tương đương của thiết kế.

- Các thư viện là tập hợp các thực thể được mô tả sẵn. Các thực thể trong thư viện sẽ được sử dụng tùy theo mức quan trọng của thiết kế.

Các cấu trúc cơ sở của ngôn ngữ VHDL bao gồm:

- Các thực thể

- Các kiến trúc
- Các gói
- Các cấu hình
- Các thư viện

1. Mô tả các thực thể

Các khai báo thực thể cho ta cái nhìn đối với phần tử mạch cần được mô tả từ mặt bên ngoài. Bằng cách khai báo thực thể, phần tử mạch sẽ được mô tả bằng số lượng và chức năng của các cổng giao tiếp và các tính chất của dữ liệu tại các cổng này theo phương diện từ ngoài vào.

Ngôn ngữ VHDL được mô tả theo ngôn ngữ cú pháp. Đối với các mô tả thực thể cú pháp có dạng như sau:

entily ten_thực_thể **is**

[lệnh_khai_báo_**generic**]

[các_luật_tại_cổng]

{các_thành_phần_khai_báo_thực_thể}

[**begin**

Các_thành_phần_biểu_thức_thực_thể]

end [tên_thực_thể];

Lệnh khai báo generic dùng để khai báo các tham số được sử dụng để kiểm soat cấu trúc hoặc hành vi của thực thể. Các hằng số này được gọi là các tham số chung. Cấu trúc generic có cấu trúc như sau:

generic (

tên_hằng: kiểu_con [:=giá_tri_khởi_tạo]

{;tên_hằng: kiểu_con [:=giá_trị_khởi_tạo]}

);

tên_hằng: là tên của tham số chung

kiểu: kiểu dữ liệu của tham số.

giá_trị_khởi_tạo: giá trị khởi tạo của tham số.

Các luật cổng đặc tả cho các kênh giao tiếp của thực thể và có quy tắc cú pháp như sau.

port (tên_cổng: [mode] kiểu_con[:=giá_trị_khởi_tạo]

{; tên_cổng: [mode] kiểu_con[:=giá_trị_khởi_tạo]}

);

tên_cổng: tên của cổng được mô tả.

model: chỉ hướng của tín hiệu tại cổng.

kiều_con: kiểu dữ liệu tại cổng hoặc các tham số chung.

giá_trị_khởi_tạo: giá trị khởi tạo cho cổng.

Trong phần khai báo, thực thể và các cổng của thực thể luôn được đặt tên hoặc đánh định danh. Tên, định danh trong ngôn ngữ VHDL không phân biệt chữ hoa và thường. Một số định danh là các từ khóa của ngôn ngữ như: **entily**, **port**,**is**,**end**…Những từ này có ý nghĩa cố định và không thể thay đổi trong toàn bộ ngôn ngữ.

Các cổng là các tín hiệu kết nối thực thể với các thực thể khác. Những tín hiệu tại cổng được đặt tương ứng với các dạng in, out, buffer, inout và các kiểu dữ liệu. Ý nghĩa của các dạng cổng như sau:

- Cổng có dạng in là cổng chỉ dùng để đọc. Trong các mạch số, các cổng dạng in chỉ được sử dụng làm cổng tín hiệu vào.

- Cổng có dạng out là cổng chỉ dùng để gán giá trị. Trong thiết kế mạch, các cổng dạng out chỉ được sử dụng làm cổng tín hiệu ra.

- Các cổng dạng buffer là cổng cho phép cả 2 thao tác đọc và gán dữ liệu. Nhưng trong từng ngữ cảnh cổng chỉ có thể nhận 1 trong 2 chức năng đọc hoặc gán giá trị.

- Cổng inout là cổng có thể vừa đọc vừa gán giá trị. Các cổng này cho phép có nhiều điều khiển dữ liệu đồng thời trong mọi ngữ cảnh.

Trong phần *các_thành_phần_khai_báo_thực_thể* của thực thể chứa khai báo các hằng số, cách kiểu hoặc tín hiệu có thể sử dụng trong quá trình xây dựng thực thể. Phần

các_thành_phần_khai_báo_thực_thể chứa các biểu thức thực hiện đồng thời. Các biểu thức này được sử dụng để kiểm tra các điều kiện rằng buộc các phép toán trong hành vi của thực thể cần thiết kế.

2. Các kiến trúc

Trong ngôn ngữ VHDL các kiến trúc cung cấp cái nhìn bên trong của thực thể. Kiến trúc của thực thể xác định mối quan hệ giữa các đầu vào và đầu ra của thực thể và có thể biểu diễn theo hành vi, theo dòng vận chuyển dữ liệu hoặc theo cấu trúc.

Kiến trúc xác định chức năng của thực thể, kiến trúc chứa phần khai báo,trong đó bao gồm các khai báo của tín hiệu, khai báo kiểu, khai báo hằng, khai báo các thành phần và các chương trình con. Trong phần thân của kiến trúc chứa các câu kết cấu thực hiện đồng thời. Các kết cấu thực hiện đồng thời thể hiện tính chất thực hiện đồng thời của các thành phần phần cứng trong thiết kế khi có sự thay đổi trạng thái tín hiệu tác động vào mạch. Các kết cấu thực hiện đồng thời tương tác với nhau thông qua các tín hiệu. Mỗi kết cấu thực hiện đồng thời xác định một phần tử tính toán. Phần tử này đọc tín hiệu, thực hiện các phép toán trên các phép toán gán những giá trị tính được cho tín hiệu. Các kết cấu này biểu diễn khối các phần cứng và cách thức liên kết giữa chúng theo dạng cấu trúc hoặc dạng hành vi.

Kiến trúc được mô tả theo quy tắc sau:

architecture *ten_kiến_trúc* of *tên_thực_thể* **is**

{*phần_khai_báo_của_kiến_trúc*]

begin

{*các_kết_cấu_thực_hiện_đồng_thời*}

end [*tên_kiến_trúc*];

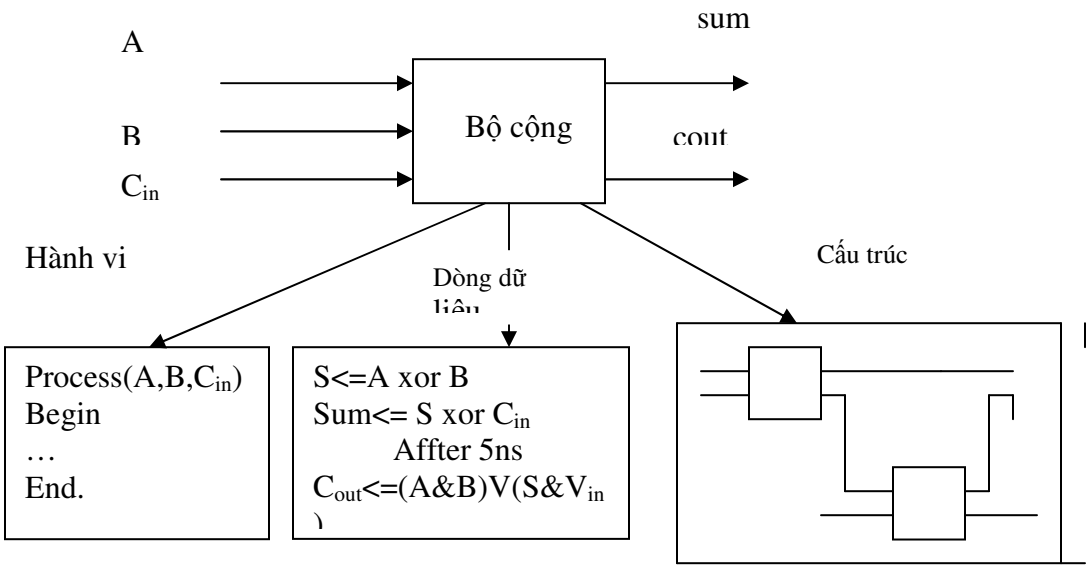
tên_thực_thể: là tên của đối tượng sẽ xây dựng.Tên này phải trùng với tên của thực thể tương ứng với kiến trúc đã khai báo trong phần khai báo thực thể.

phần_khai_báo_của_kiến_trúc chứa những khai báo trong mô tả kiến trúc. Hai dau {} có nghĩa là có thể không chứa một khai báo nào hoặc có chứa nhiều khai báo.

Phần nằm giữa 2 từ khóa begin và end,*các_kết_cấu_thực_hiện_đồng_thời* xác định các khối phần cứng theo dạng cấu trúc hoặc hành vi. Các thành phần signals dùng để kết nối các khối riêng biệt của kiến trúc. Mỗi tín hiệu tương ứng với một kiểu dữ liệu.

Một thực thể có thể có nhiều kiến trúc. Thông thường, ta có thể biểu diễn kiến trúc của một thực thể ở 3 phương diện: phương diện hành vi, phương diện dòng dữ liệu, phương diện cấu trúc. Một kiến trúc cũng có thể có hốn hợp cả 3 phương diện biểu diến thiết kế. Ta hãy xét ví dụ biểu diễn kiến trúc thiết kế của mạch cộng một bit. Theo hành vi, mạch cộng có thể được biểu diễn như một hàm lôgic tác động lên 3 biến độc lập là A, B, Cin để hình thành lên tín hiệu ra là Sum và Cout. Theo cách biểu diễn bằng dòng tín hiệu, mạch cộng thực hiện các tác động.

Ta xét ví dụ biểu diễn kiến trúc của mạch cộng một bít:



Hình 6.4:Mô tả thực thể mạch cộng và các kiến trúc

a. Biểu diến kiến trúc theo hành vi

Biểu diễn kiến trúc của thực thể theo hành vi là mô tả chức năng của hệ thống tương tự như các chương trình phần mềm bằng các quá trình tính toán. Trong biểu diễn này ta không cung cấp chi tiết việc thực hiện thiết kế. Trong ngôn ngữ VHDL để biệu diễn kiến trúc theo hành vi, cấu trúc chình của hành vi sẽ là quá trình, một quá trình

có thể coi như là chương trình và được xây dựng từ những kiến trúc thủ tục và có thể cho phép gọi các chương trình con giống như các ngôn ngữ lập trình truyền thống.

Ta hãy xét ví dụ mô tả kiến trúc của mạch cộng một bit bằng hành vi. Trong mô tả chứa một quá trình với 3 tham số là các tín hiệu A, B và Cin. Các tín hiệu này chứa trong danh sách các tín hiệu tác động vào quá trình. Việc thực hiện các quá trình sẽ dừng lại nếu không có các sự kiện xảy ra trên các đương tín hiệu xuất hiện trong danh sách hay nói một cách khác là các tín hiệu trong danh sách các tín hiệu tác động không thay đổi giá trị. Mỗi khi có một sự kiện xảy ra trên các đường tín hiệu, quá trình sẽ được kích hoạt và các câu lệnh bên trong cấu trúc sẽ được thực hiện tuần tự.

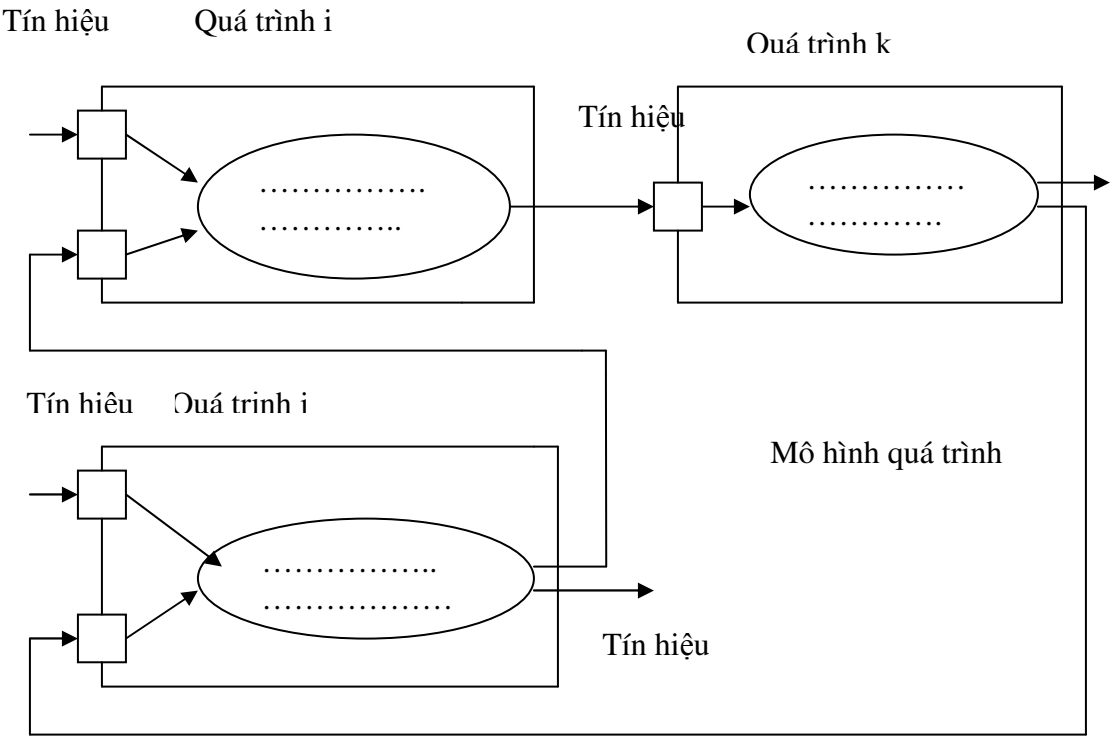
```
architecture BEHAVIOR of FULL_ADDER is
begin
    process (A,B,Cin)
    begin
        if (A = '0' and B = '0' and C = '0') then
            Sum <= '0';
            Cout <= '0';
        elsif (A= '0' and B = '0' and Cin= '1') or
            (A= '0' and B = '1' and Cin= '0') or
            (A= '1' and B = '0' and Cin= '0') then
            Sum <= '1';
            Cout <= '0';
        elsif (A= '1' and B = '1' and Cin= '1') then
            Sum <= '1';
            Cout <= '1';
        end if
    end process;
end BEHAVIOR;
```

Một quá trình mô tả hành vi của một phần hoặc toàn bộ thiết kế. Quá trình xác định các khối mã lệnh tuần tự độc lập. Các khối này có thể được kích hoạt ừng với các biến đổi trạng thái của tín hiệu. Khi có nhiều hơn một quá trình trong kiến trúc thì những quá trình đó được thực hiện đồng thời

i. Mô hình hành vi

Thiết kế số được mô hình hóa như một nhóm các phép toán tác động lên các giá trị dữ liệu truyền qua hệ thống. Trong mô hình hành vi của VHDL mỗi phép toán được gọi là một quá trình và các giá trị dữ liệu đi qua hệ thống được gọi là tín hiệu. Một hệ thống có thể coi là tập hợp các quá trình và các quá trình tương tác với nhau thông qua các tín hiệu. Tất cả các quá trình trong mô hình thực hiện đồng thời và các tìn hiệu được dùng để định vị các quá trình song song.

Chúng ta có thể coi việc thực hiện một quá trình như một vòng lặp vô hạn. Vòng lặp này bắt đầu từ việc thực hiện dòng lệnh đầu tiên, lần lượt đến dòng lệnh thứ hai, thứ ba… cho đến dòng lệnh cuối cùng và lại quay trở lại dòng lệnh đầu tiên. Việc thực hiện các câu lệnh trong quá trình được thực hiện cho đến khi gặp câu lệnh **wait.** Khi bị dừng lại, quá trình có thể được tiếp tục thực hiện trở lại. Điều kiện để quá trình thực hiện trở lại phụ thuộc vào thời gian ngừng cực đại trong câu lệnh **wait** trôi qua. Trong nhiều trường hợp quá trình có thể được kích hoạt trở lại tùy thuộc vào sự thay đổi trạng thái của các tín hiệu tác động hoặc khi các điều kiện đặt ra được thỏa mãn. Ngôn ngữ VHDL cung cấp khả năng mô tả những tín hiệu tác động vào các quá trình bằng danh sách các tín hiệu tác động. Khi giá trị của các tín hiệu trong danh sách tác động bị thay đổi, quá trình được kích hoạt và bắt đầu thực hiện.



Hình 6.6:Mô hình hoạt động của một quá trình process

Trên hình 6.6 ta có mô hình kiến trúc của hệ thống gồm ba quá trình i,j,k. Mỗi quá trình I và j có hai tín hiệu tác động còn quá trình k chỉ có một. Đầu ra của quá trình i nối với đầu vào của quá trình k và đầu ra của quá trình j nối với đầu vào của quá trình i. Cả ba quá trình cùng thực hiện đồng thời và các tín hiệu tác động dùng để kiểm soát tiến trình thực hiện các quá trình.

ii. Mô hình điều khiển thời gian

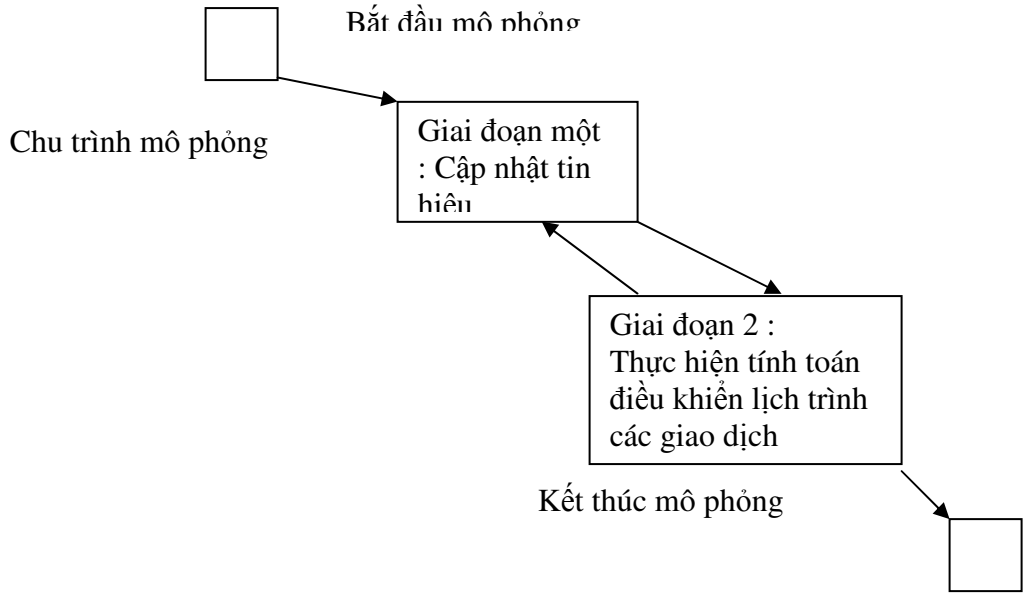
Sự thay đổi giá trị tín hiệu tại những thời điểm thời gian xác định được thể hiện qua "thời điểm mô phỏng " của hệ thống. Thời điểm mô phỏng trong ngôn ngữ VHDL là thời điểm tại đó có các sự kiện xuất hiện trên đường tín hiệu. Khái niệm thời điểm mô phỏng trên thực tế là khác với khái niệm thời gian đồng hồ nội tại. Các quá trình trong VHDL được kích hoạt lại một khi có sự thay đổi giá trị của tín hiệu trong danh sách tín hiệu tác động. Khi quá trình tạo nên giá trị cho tín hiệu đầu ra, hệ thống mô phỏng sẽ chỉ định lượng tử thời gian trước khi giá trị được gửi ra đầu ra. Lúc đó ta nói rằng hệ mô phỏng thực hiện việc định chương trình cho giao tác sau thời điểm mô phỏng xác định. Hệ thống mô phỏng của VHDL cũng cho phép định

chương trình cho một số bất kỳ các giao tác đối với đường tín hiệu ra. Tập hợp tất cả các giao tác đối với tín hiệu trong một quá trình gọi là VHDL cung cấp mô hình hai giai đoạn: chu trình mô phỏng.

- Trong giai đoạn 1: giá trị của các tín hiệu sẽ thực hiện giao dịch trong thời gian hiện tại được làm mới.

- Trong giai đoạn 2: những process nhận thông tin tại những tín hiệu tác động sẽ thực hiện tính toán cho tới khi bị treo. Kết thúc giai đoạn 2, thời điểm mô phỏng sẽ nhận giá trị mới và chu trình thực hiện lại.

Nhà thiết kế có thể chỉ định thời gian tính từ thời điểm hiện thời mà giá trị sẽ được gửi tới tín hiệu ra trong các câu lệnh gán tín hiệu. Nếu trong câu lệnh gán tín hiệu không chỉ rõ giá trị thời gian trễ hoặc giá trị này bằng '0' thì thời gian trêc mặc định delta của bộ mô phỏng sẽ được sử dụng để định lịch trình cho thao tác. Thời gian trễ này không thay đổi thời điểm xuất hiện tín hiệu đồng hồ mô phỏng nhưng được sử dụng để kết thúc một chu trình mô phỏng và bắt đầu một chu trình mới. Nếu giá trị mới được gán cho tín hiệu khác với giá trị cũ, trên đường tín hiệu xuất hiện sự kiện



Hình : chu trình mô phỏng của quá trình

.

b. Biểu diễn kiến trúc theo dòng dữ liệu

Phong cách mô tả kiến trúc theo dòng dữ liệu đặc tả hệ thống như các biểu diễn song song của các dòng dữ liệu và dòng các tín hiệu điều khiển. Theo phương pháp này chúng ta mô tả các dòng thông tin, hành vi luân chuyển các dòng dữ liệu theo thời gian của các hàm lôgic tổ hợp như mạch cộng, mạch so sánh, mạch giải mã và các phần tử lôgic cơ sở. Ví dụ.

```
architecture DATAFLOW of FULL_ADDER is
        signal S:BIT;
begin
        S<=A or B;
        Sum <= S xor Cin affer 10 ns;
        Cout <= ( A and B) or ( S and Cin) affer 5 ns;
end DATAFLOW;
```

Trong ví dụ này ta có ba biểu thức gán tín hiệu song song. Mỗi biểu thức gán này có thể hiểu như một quá trình với tín hiệu vế bên phải của phép gán là các tín hiệu nằm trong danh sách các tín hiệu tác động cuả quá trình. Ví dụ, trong phép gán thứ nhất, đường tín hiệu S sẽ nhận giá trị A xor B sau khi có sự kiện xuất hiện trên đường A, hoặc trên đường B hoặc trên cả hai đường A và B. Ở đây chúng ta không đặt giá trị tường minh của thời gian trễ. Do đó để mô phỏng tiến trình thực hiện quá trình theo thời gian, hệ thống sử dụng thời gian trễ mặc định là **delta.** Trong phép gán thứ hai, đường tín hiệu Sum nhận giá trị biểu thức S xor Cin sau 10ns so với thời điểm có các sự kiện trên các đầu vào S và Cin của phần tử xor. Cũng tương tự như vậy, đường tín hiện Cout sẽ nhận giá trị *(A and B) or (S and Cin)* sau 5ns so với thời điểm có sự kiện trên các đường tín hiệu A,B,S hoặc Cin.

Các hằng số cơ sở có thể được dùng như các tham số trễ. Trong ví dụ này, chúng ta thấy các khai báo của tham số trễ của phần khai báo thực thể và việc sử dụng chúng trong kiến trúc.

Ví dụ.

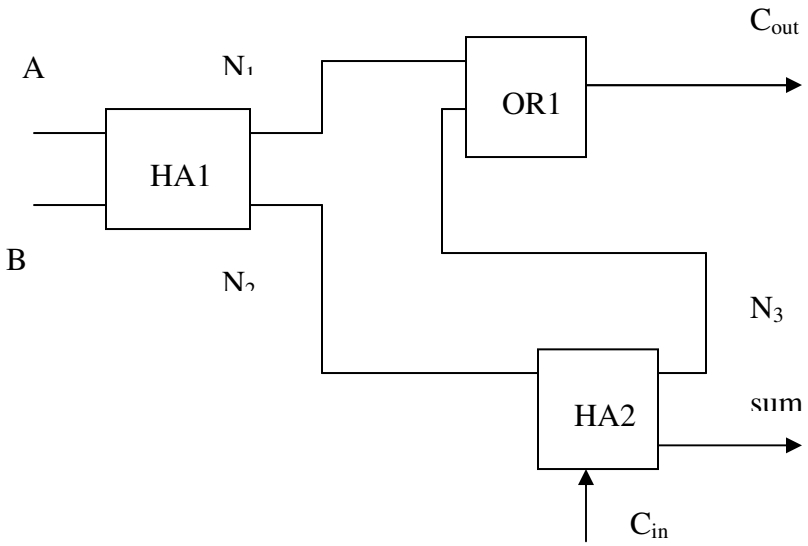
```
Entity FULL_ADDER is
        Generic ( N: TIME:= 5ns );
        Port ( A,B,Cin: in BIT; Sum, Cout: out BIT );
End FULL_ADDER;
Architecture  DATAFLOW of FULL_ADDER is
        Signal S: BIT;
        Begin
                S <= A xor B;
                Sum <= S xor Cin affer 2*N;
                Cout <= (A and B) or ( S and Cin) affer N;
End DATAFLOW;
```

c. Biểu diễn kiến trúc bằng cấu trúc

Phong cách mô tả kiến trúc thông qua cấu trúc xác định cấu trúc của thực thể sử dụng các khai báo của các phần tử thành phần và các phiên bản của các phần tử thành phần. Các mô tả cấu trúc chứa danh sách các phần tử hoạt động đồng thời và liên kết giữa các phần tử đó. Ví dụ, khi ta mô tả cấu trúc của mạch cộng một bit FULL_ADDER. Mạch FULL_ADDER được thiết kế trên cơ sở các thành phần là HALF_ADDER và OR_GATE. Cấu trúc được thiết kế của mạch cộng một bit chứa hai mạch nửa tổng và một phần tử lôgic cơ bản là OR_Gate. Các phần tử này liên kết với nhau bằng các tín hiệu.

```
architecture ATRUCTURE of FULL_ADDER is
        component HALF_ADDER
            port ( L1,L2: in BIT ;
```

```
        Carry,Sum ; out BIT );
    end component;
    component OR_GATE
        port ( L1,L2: in BIT;
            O: out BIT );
    end component;
    signal N1,N2,N3 : BIT;
begin
    HA1 : HALF_ADDER port map ( A,B,N1,N2 );
    HA2 : HALF_ADDER port map ( N2,Cin,N3,Sum );
    OR1 : OR_GATE port map (N1,N3,Cin );
end STRUCTURE;
```



Hình 6.8:Cấu trúc ở mức kiến trúc của mạch cộng 1 bít cuả hai tín hiệu

Khi độ phức tạp của thiết kế tăng lên, nhà thiết kế thường phân tích hệ thống thành những hệ thống con. Các hệ thống con này liên kết chặt chẽ theo chức năng

trong thành phần của hệ thống tổng thể. Mỗi hệ thống con lại có thể được phân tách thành những phân hệ ở mức thấp hơn nữa. Trong ngôn ngữ VHDL, ở mức cao nhất của thiết kế người ta sử dụng mô hình kiến trúc theo phong cách kiến trúc của thực thể.

Mỗi phiên bản của thành phần mạch được mô tả bằng các hộp đen trong biểu diễn cấu trúc với các liên kết đầu vào và liên kết đầu ra được mô tả rõ ràng. Các phiên bản thành phần phải tương thích với các thực thể. Các thực thể này sẽ mô tả các chức năng của thành phần mạch bằng các mô hình kiến trúc theo biểu diễn cấu trúc hoặc hành vi. Ví dụ, đối với mạch cộng một bit của hai tín hiệu, ta có mạch cộng sẽ được xây dựng từ hai mạch nửa tổng và một phần tử OR. Khi đó theo các mô tả kiến trúc của các thực thể bằng các mô tả cấu trúc, thực thể FULL_ADDER được tạo bởi hai thực thể HALF_ADDER và một thực thể OR_GATE. Trong đó thực thể HALF_ADDER có thể được xây dựng từ các phần tử XOR và AND.

Mô tả thực thể HALF_ADDER theo cấu trúc từ các phần tử AND và XOR.

```
entity HALF_ADDER is
port ( I0, I1: in BIT; S, C0: out BIT );
end HALF_ADDER;
architecture STRUCTURE of HALF_ADDER is
    component XOR_GATE
        port ( I0, I1: in BIT; S, C0: out BIT );
    end component;
    component AND2_GATE
        port ( I0, I1: in BIT; S, C0: out BIT );
    end component;
begin
    U1: XOR_GATE port map ( I0,I1,S);
    U2: AND2_GATE port map ( I0,I1,C0);
```

```
end STRUCTURE;
```

Một thành phần của thực thể nêu trên có thể xây dựng từ các thực thể khác mô tả các chức năng của chúng. Ví dụ phần tử XOR_GATE có thể được mô tả theo hành vi như sau.

```
Entity XOR_GATE;
    port ( I0, I1: in BIT; S, C0: out BIT );
end XOR_GATE;
architecture BEHAVIOR of XOR_GATE is
begin
    O <= I0 xor I1 affer 10 ns;
End BEHAVIOR;
```

Biểu diễn cấu trúc của các phân cấp thiết kế ành hưởng tới quá trình phân tách thiết kế. Điều này xuất phát từ các đặc điểm cảu hệ thống được thiết kế. Ở một mực phân cấp bất kỳ, hệ thống được cấu tạo bởi các liên kết cuả những thành phần ở mức đang xét. Biểu diễn cấu trúc của kiến trúc chứa danh sách các hộp đen. Ở mức thấp nhất của quá trình phân tách, ta phải mô tả hành vi của các phần tử nằm trong thiết kế ở mức này. Quá trình phân cấp có thể biểu diễn dưới dạng cây phân cấp. Tại mức thấp nhất của phân cấp, ta phải mô tả hành vi của các thực thể theo trình tự mà mô hình mạch sẽ được mô phỏng.

3. Các gói thiết kế

Mục đích chính của các gói là tập hợp các phần tử có thể dùng chung giữa hai hoặc nhiều đơn vị thiết kế. Một gói bao gồm hai thành phần: phần khai báo gói và phần thân gói.

- Phần khai báo gói chứa tất cả các khai báo của mọi tên. Những tên này sẽ được các đơn vị thiết kế dùng để khi sử dụng gói. Thông thường phần khai báo chứa một số kiểu dữ liệu chung, các hằng và mô tả của các chương trình con.

- Phần thân gói bao gồm các phần thân của các chương trình con mô tả trong phần khai báo gói. Phần thân này là ẩn đối với bên ngoài. Phần thân của gói không bắt buộc phải có nếu không có chương trình con được mô tả trong gói.

Ví dụ, ta có khai báo gói như sau. Gói này khai báo một số kiểu biến, hằng và chương trình con.

```
Package EX_PKG is
        Subtype INT8 is INTEGER range 0 to 255
        Constant ZERO : INT8 :=0;
        Constant MAX : INT8 := 100;
        Procedure Increcment ( variable count : inout INT8 N)
End EX_PKG;
```

Do trong khai báo có thủ tục Increcment nên ta cần phải có thân của gói tương ứng với khai báo gói trên.

```
Package body EX_PKG is
        Procedure Increcment ( variable Data : inout INT8 ) is
            Begin
                If ( Count >= MAX ) then
                        Count := ZERO;
                Else
                        Count := Count + 1;
            End if
        End Increcment;
End EX_PKG;
```

4. Các cấu hình

Một thực thể có thể có một vài kiến trúc. Trong quá trình thiết kế, ta có thể cần phải thử nghiệm một vài biến thể của thiết kế bằng cách sử dụng các kiến trúc khác

nhau. Cấu hình là thành phần cơ bản của đơn vị thiết kế. Cấu hình cho phép gắn các phiên bản của thực thể ào những kiến trúc khác nhau. Cấu hình cũng có thể được sử dụng để thay thế một cách nhanh chóng các phần tử của thực thể trong biểu diễn cấu trúc của thiết kế.

Cú pháp của mô tả cấu hình:

Configuration tên_cấu_hình of tên_thực_thể **is**

{ phần_khai_báo_của_cấu_hình }

For *đặc_tả_của_khối*

{ mệnh_đề_use }

{*các_phần_tử_của_cấu_hình* }

End for;

Vị từ *phần_khai_báo_của_cấu_hình* cho phép cấu hình sử dụng các phần tử trong các gói và các thư viện.

Vị từ *đặc_tả_của_khối* xác định cấu hình cho kiến trúc của thực thể

5. Các thư viện thiết kế

Phân tích VHDL là quá trình kiểm tra thiết kế VHDL cho đúng cú pháp và ngữ nghĩa. Sau khi phân tích VHDL, các đơn vị thiết kế sẽ được lưu dữ trong các thư viện để sử dụng sau này. Thư viện thiết kế có thể chứa các phần tử thư viện sau.

Gói: là những mô tả, khai báo được dùng chung

Thực thể: là những mô tả thiết kế được dùng chung.

Kiến trúc: những thiết kế chi tiết được dùng chung.

Cấu hình: là những phiên bản của thực thẻ được dùng chung.

Các đơn vị thư viện là cấu trúc VHDL có thể được phân tích riêng rẽ theo trình tự nhất định.

Trong ngôn ngữ VHDL có thư viện thiết kế đặc biệt có tên là “WORK” . Khi chúng ta biên dịch một chương trình viết trên ngôn ngữ VHDL nhưng không chỉ rõ thư viện đích, chương trình này sẽ được biên dịch và chứa vào thư viện “WORK”. Ví dụ lệnh

vc My-Design.vhd sẽ kiểm tra cú pháp chương trình nằm trong tệp "My-Design.vhd", dịch

Chương trình đó rồi chứa vào thư viện "WORK". Hình 6.10 chỉ ra các phương thức sử dụng các thư viện thiết kế trong VHDL.

6.3 Các kiểu dữ liệu

Ngôn ngữ VHDL có ba dạng đối tượng: biến, tín hiệu và hằng. Phần khai bào trong các cấu trúc ngôn ngữ sẽ liệt kê các đối tượng sẽ sử dụng, các kiểu của các đối tượng đó và giá trị ban đầu mà chúng sẽ nhận trong quá trình mô phỏng.

1. Các đối tượng dữ liệu

Trong ngôn ngữ VHDL người ta phân loại 3 loại đối tượng: biến, tín hiệu và hằng. Các đối tượng được đặc tả dựa vào các từ khóa. Những từ khóa này phải xuất hiện ở phần đầu của khai báo đối tượng.

a. Hằng

Hằng là đối tượng được khởi tạo bằng những giá trị nhất định khi được tạo nên trong quá trình thực hiện và sau đó giá trị của hằng không thay đổi. Hằng có thể được khai báo trong các gói, thực thể, kiến trúc, chương trình con, khối và quá trình. Cú pháp khai báo hằng

Constant tên_hằng { ,tên_hằng }: **kiểu** [:=giá_trị];

Ví dụ.

Constant CHAR7 : BIT_VECTOR (4downto 0) :="00111";

Constant MSB : INTEGER :=5;

b. Biến

Biến là đối tượng dữ liệu dùng để chữa những kết quả trung gian. Biến chỉ có thể được khai báo bên trong các quá trình hoặc chương trình con. Biến luôn đi đôi với kiểu, do đó biến phải được khai báo kiểu, xác định khoảng giới hạn hoặc giá trị khởi tạo ban đầu. Một cách mặc định, giá trị khởi tạo của biến là giá trị thấp nhất trong các giá trị thuộc miền xác định của kiểu. Biến có cú pháp khai báo như sau.

variable tên_biến { , tên_biến } : **kiểu** [: giá_trị_khởi_tạo];

c. Tín hiệu

Tín hiệu là đối tượng dữ liệu dùng để kết nối giữa các quá trình hoặc đồng bộ các quá trình. Khai báo tín hiệu sẽ tạo tín hiệu mới có các giá trị của kiểu xác định. Tín hiệu có thể được khai báo trong phần khai báo gói, khai báo thực thể, khai báo kiến trúc và trong khối. Các tín hiệu có cú pháp khai báo như sau.

signal tên_tín_hiệu {, tên_tín_hiệu } : kiểu [:= giá_trị_khởi_tạo];

2. Các kiểu dữ liệu

Trong đối tượng dữ liệu trong ngôn ngữ VHDL đều phải được định nghĩa bởi các kiểu dữ liệu. Ngôn ngữ VHDL cho phép sử dụng các kiểu cơ sở để tạo nên các đối tượng phức tạp hơn.

Kiểu phải được khai báo trước khi dử dụng. Khai báo kiểu xác định tên kiểu và miền xác định của kiểu.

Các kiểu dữ liệu chính trong ngôn ngữ VHDL:

- Kiểu liệt kê.
- Kiểu số nguyên.
- Kiểu được định nghĩa trước của VHDL.
- Kiểu mảng.
- Kiểu bản ghi.
- Kiểu STD_LOGIC.
- SIGNED và UNSIGNED.
- **Các kiểu con.**

a. Kiểu liệt kê

Kiểu liệt kê được định nghĩa bằng cách liệt kê tất cả các giá trị có thể có của kiểu. Các giá trị này do người sử dụng xác định và có thể là các tên hoặc những ký tự.

Cú pháp của kiểu liệt kê.

Type *tên_kiểu* **is** (*giá_trị_liệt_kê* {, *giá_trị_liệt_kê* });

Trong ngôn ngữ VHDL kiểu liệt kê có đặc điểm khác với kiểu liệt kê của các ngôn ngữ lập trình khác. Mỗi giá trị trong thành phần cuả biến có thể xuất hiện trong hai hoặc nhiều hơn kiểu liệt kê.

b. Kiểu số nguyên.

Kiểu nguyên là miền xác định của các số nguyên. Tất cả các phép toán toán học thông thường đều áp dụng được cho số nguyên. Các phép toán thực hiện trên kiểu nguyên là : +,- ,* ,/. Quy tắc cú pháp khai báo kiểu nguyên có dạng như sau.

type *tiên_kiểu* **is range** *miền_số_nguyên*;

miền_số_nguyên là miền con của tập hợp số nguyên.

c. Các kiểu được định nghĩa trước trong VHDL

Sau đây là một số kiểu chuẩn được mô tả trong gói standard.

- boolean: kiểu liệt kê có 2 giá trị false hoặc true với quan hê false < true. Các phép tác động lên đối tượng kiểu boolean là phép toán lôgic và quan hệ.

- Bit: kiểu liệt kê với 2 giá trị ‘0’ và ‘1’. Các phép toán lôgic có thể thực hiện trên các đối tượng kiểu Bit và trả lại giá trị kiểu Bit.

- Character: kiểu liệt kê với miền xác định là tập hợp các ký tự ASII. Các ký tự không hiện được biểu diễn bằng tên chứa 3 ký tự.

- Integer: kiểu số nguyên với những giá trị dương hoặc âm (từ -2,147,483,647 đến 2,147,483,647).

- Natural: là kiểu con của số nguyên và dùng để chỉ các số nguyên không âm – số tự nhiên.

- Positive: là kiểu con của kiểu số nguyên sử dụng để biểu diễn các số dương.

-Bit_vector: là kiểu biểu diễn mảng các Bit.

- String: kiểu dữ liệu bao gồm mảng các Charater

- Real: mô tả các số thực trong giới hạn từ -1.0E+38 đến 1.0E + 38

- Physical type Time: kiểu Time được sử dụng để biểu diễn các giá trị thời gian dùng trong quá trình mô phỏng.

d. Kiểu mảng

Tương tự như trong các ngôn ngữ lập trình truyền thống, trong ngôn ngữ VHDL, phần tử kiểu mảng là nhóm các phần tử cùng kiểu và được truy cập tới như một đối tượng. Phần tử kiểu mảng trong ngôn ngữ VHDL có những đặc điểm như sau.

- Các phần tử của mảng có thể là mọi kiểu trong ngôn ngữ VHDL.

- Số lượng các chỉ số của mảng có thể nhận mọi giá trị dương.

-Mảng chỉ có một và chỉ một chỉ số dùng để truy nhập tới phần tử.

Miền biến thiên của chỉ số xác định số phần tử của mảng và hướng sắp xếp chỉ số dương trongn mảng, từ chỉ số cao xuống các chỉ số thấp hoặc ngược lại.

- Kiểu của chỉ số có thể là kiểu nguyên hoặc kiểu liệt kê.

Khác với các ngôn ngữ lập trình truyền thống, trong ngôn ngữ VHDL mảng được chia làm hai loại: mảng có ràng buộc và mảng không ràng buộc.

-Mảng có ràng buộc là mảng trong đó kiểu của chỉ số có miền xác định được quy định một cách tường minh.

- Mảng không ràng buộc là kiểu mảng trong đó miền xác định của kiểu chỉ số và hướng sắp xếp của các chỉ số không xác định mặc dù số lượng chiều của mảng được chỉ rõ.

e. Kiểu bản ghi

Bản ghi là nhóm của một hoặc nhiều phần tử thuộc những kiểu khác nhau và được truy cập tới như một đối tượng. Bản ghi có những đặc điểm sau.

- Mỗi phần tử của bản ghi được truy nhập tới theo tên trường.

- Các phần tử của bản ghi có thể nhận mọi kiểu của ngôn ngữ VHDL kể cả mảng và bản ghi.

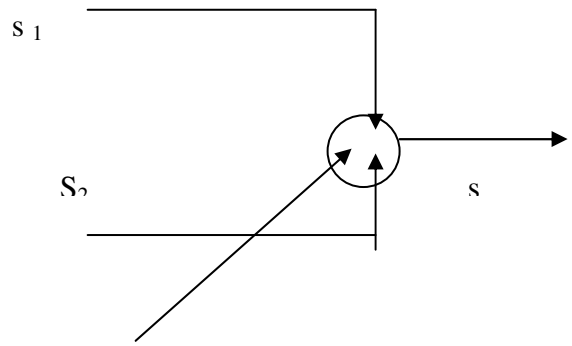
f. Các kiểu trong gói Standard Logic

Để mô hình hóa các tín hiệu có thể nhận nhiều hơn hai giá trị, trong ngôn ngữ VHDL xác định gói Standard Logic (tương ứng với chuẩn IEEE Std 1164 – 1993). Hai kiểu dữ liệu cơ sở trong gói này là STD_ULOGIC và STD_LOGIC.

STD_ULOGIC xác định kiểu dữ liệu gồm chín giá trị. Các giá trị tín hiệu kiểu STD_ULOGIC không thể tham gia vào các phép toán lôgic và số học cơ bản. Trong trường hợp này ta phải mở rộng các phép toán số học và lôgic cơ bản bằng cách cung cấp các hàm quyết định.

STD_LOGIC là kiểu được xác định. Các hàm quyết định của kiểu này được cung cấp bởi gói Standard Logic.

Tương tự như BIT và BIT_VECTOR, ngôn ngữ VHDL cung câp 2 kiểu STD_ULOGIC_VECTOR và STD_LOGIC_VECTOR.



g. Kiểu signed và unsigned

Kiểu có dấu signed và không dấu unsigned được sử dụng đối với những đối tượng sẽ được truy cập tới như các bit và như các số nguyên. Hai kiểu này được định nghĩa trong hai gói NUMERIC_BIT va NUMERIC_STD như sau.

type signed is array (natural range<>) **of** BIT/STD_LOGIC;

type unsigned is array (natural range<>) **of** BIT/STD_LOGIC;

h. Kiểu con

Các kiểu con là một tập hợp con của các kiểu đã được định nghĩa khác. Phép khai báo kiểu con có thể nằm ở mọi vị trí cho phép khai báo kiểu. Sự khác biệt giữa kiểu con và kiểu là ở chỗ kiểu con chỉ là tập con của một kiểu hoặc một kiểu con đã được định nghĩa trước.

Kiểu con thường dùng để giới hạn các dạng dữ liệu trong các phép gán và trên các đường dữ liệu.

6.4 Toán tử và biểu thức

Trong ngôn ngữ VHDL, các biểu thức là các công thức. Các công thức này xác định các tác động tính toán lên các đối tượng dữ liệu. Các biểu thức thực hiện các tính toán số học và lôgic sử dụng các toán tử với một số các toán hạng. Các toán tử đặc trưng cho phép toán sẽ được thực hiện còn các toán hạng là các nguông dữ liệu cho các phép toán.

1. Các toán tử

Các toán tử được phân chia theo các mức độ ưu tiên và trật tự tính toán. Trong bảng 6.1 đưa ra các nhóm phép toán với mức độ ưu tiên tăng dần. Các quy ước về trật tự thực hiện các phép toán trong biểu thức được thể hiện như sau:

- Trong biểu thức các phép toán có mức độ ưu tiên lớn hơn sẽ được thực hiện trước. Các dấu ngoặc đơn cũng giúp xác định đúng trật tự tính toán biểu thức.

Các phép toán trong nhóm với cùng một mức độ ưu tiên sẽ được thực hiện từ trái qua phải trong các biểu thức.

Bảng 6.1.Các toán tử và mức độ ưu tiên.

Các phép toán lôgic	And, or, nand, nor, xor
Các phép toán quan hệ	=, /=, <, >, <=, >=
Các phép toán cộng	+, -, &
Toán tử dấu	+,-

Các phép toán nhân	*,/,mod, rem
Các phép toán khác	**, abs, not

a. Các phép toán lôgic

Trong ngôn ngữ VHDL, các phép toán lôgic gồm có and, or, nand, nor, xor và not. Các phép toán này có tác động lên các dữ liệu kiểu BIT, Boolean và mảng một chiều các BIT. Đối với các phép toán hai ngôi and, or, nand, nor, xor, các toán hạng phải cùng kiểu. Trong trường hợp các toán hạng là các mảng một chiều các BIT thì các mảng phải có cùng độ dài. Các phép toán nhị phân xác định các hàm trên các bit của những phần tử mảng có cùng chỉ số, kế quả sẽ là một mảng có cùng độ dài và giới hạn các chỉ số. Phép toán một ngôi **not** xác định phép đảo bit đối với toán hạng của nó. Trong trường hợp toán hạng là một mảng thì phép toán **not** tác động lên tất cả các phần tử của mảng.

b. Các phép toán quan hệ

Các phép toán quan hệ gồm các phép toán so sánh " =", " /=", " <", " <=", " >", " >=". Các phép toán so sánh là các phép toán hai ngôi và các toán hạng phải có cùng kiểu. Kết quả so sánh nhận kiểu Boolean. Kết quả trả về giá trị **true** nếu quan hệ được nghiệp đúng và nhận giá trị **false** trong trường hợp ngược lại.

Các quan hệ " =" và " /=" xác định với mọi kiểu dữ liệu. Còn các quan hệ còn lại " <", " <=", " >", " >=" chỉ xác định với các kiểu liệt kê, số nguyên, mảng một chiều các liệt kê hoặc số nguyên.

c. Các phép toán cộng

Các phép toán cộng bao gồm " +", " –", " &". Các phép toán " +", " –" thực hiện trên các đối tượng kiểu **integer.** Phép toán nối " &" áp dụng với các đối tượng là mảng thanh ghi. Phép toán này xây dựng mảng mới bằng cách ghép nối hai mảng nằm trong toán hạng với nhau. Mỗi toán hạng của phép toán " &" có thể là một mảng hoặc phần tử của mảng. Các phép toán này cũng thực hiện với các toán hạng có kiểu **signed** và **unsigned**.

d. Các phép toán định dấu

Các phép toán một ngôi " +", " –", "abs" thực hiện với các toán hạng dạng số và trả lại giá trị cùng kiểu.

e. Các phép toán nhân

Các phép toan nhân " **", "*" , "/" , **mod, rem** thực hiện trên các kiểu **integer**. Phép toán "**" thể hiện nâng lên lũy thừa;"*" phép nhân; "/" phép chia; **mod** lấy môdun; **rem** lấy phần dư.

2. Các toán hạng

Trong một biểu thức, các toán tử dùng các toán hạng để tính các giá trị của chúng. Thông thường trong ngôn ngữ VHDL có nhiều dạng toán hạng. Các toán hạng cũng có thể là chính các biểu thức.

Các dạng toán hạng trong ngôn ngữ VHDL bao gồm:

- Các hằng, ký hiệu như 'x', "1001", 345;
- Các tên, định danh;
- Các chỉ số;
- Các tên ngắn và biệt danh;
- Các tên thuộc tính;
- Các biểu thức định kiểu;
- Các phép gọi hàm;
- Các biểu thức chuyển đổi kiểu.

a. Các hằng và giá trị ký hiệu

Các hằng ký và giá trị ký hiệu hoặc là giá trị số, giá trị ký tự, các giá trị liêtk kê, các giá trị xâu.

- Các giá trị số là các hằng giá trị nguyên. Các giá trị số được biểu diễn tùy thuộc theo hệ cơ số mà giá trị đó biểu diễn.
    - Trong hệ cơ số mười các giá trị số được viết như bình thường.

- Trong hệ cơ số khác từ 2 → 16, ta viết dưới dạng

cơ_số # giá_trị_hằng

- Các hằng ký tự là các ký tự riêng lẻ được viết trong hai ngoặc đơn.

- Các hằng liệt kê là các hằng được xác định trong định nghĩa kiểu liệt kê. Trong ngôn ngữ VHDL, các giá trị liệt kê của các kiểu liệt kê khác nhau có thể trùng nhau.

- Các giá trị xâu ký tự là biểu diễn của mảng ký tự một chiều các ký tự. Có hai dạng hằng xâu: các xâu ký tự và xâu các bit.

- Xâu ký tự là chuỗi các ký tự nằm giữa hai dấu móc kép.
- Xâu các bít được biểu diễn tương tự như xâu ký tự, nhưng phân biệt các dạng xâu nhị phân, octal và hexa bằng các ký tự chỉ hệ cơ số.

b. Các tên và định danh

Các định danh đôi khi còn gọi một cách đơn giản là tên. Định danh là các tên đối với hằng, biến, tín hiệu, thực thể, của các cổng, chương trình con và của các khai báo tham số. Các từ khóa trong ngôn ngữ VHDL cũng là định danh. Các tên phải bắt đầu từ chữ cái và chỉ chứa chữ cái, chữ số và dấu gạch nối '_'. Dấu gạch nối không thể là ký tự sau cùng trong một định danh. Trong ngôn ngữ VHDL các tên và định danh không phân biệt theo chữ hoa và chữ thường.

c. Tên được chỉ số hóa

Những tên này xác định một phần tử của đối tượng mảng. Cú pháp mô tả của tên chỉ số như sau:

tên_mảng (*biểu_thức*)

trong đó *biểu_thức* phải trả lại giá trị là chỉ số của phần tử mảng trong miền xác định của chỉ số.

d. Tên gán và biệt danh

Tên ngắn xác định chuỗi các phần tử của đối tượng mảng. Hướng của các chỉ số phần tử mảng là **to** hoặc **downto.** Tuy nhiên chiều chỉ số của tên ngắn phải tương thích với

chiều chỉ số của kiểu mảng tương ứng. Tên ngắn có thể sử dụng cùng với các biệt danh (alias).

e. Thuộc tính

Thuộc tính là dữ liệu gắn liền với đối tượng trong ngôn ngữ VHDL. Thuộc tính trong ngôn ngữ VHDL của các biến hoặc tín hiệu tương ứng với những giá trị cụ thể và được xác định theo quy tắc cú pháp sau:

Tiền_tố_thuộc_tính

Các thuộc tính được định nghĩa trước trong ngôn ngữ VHDL là **left, right, low, high, range, reverse-range,length.**

- Các thuộc tính **left** hoặc **right** trả lại chỉ số của phần tử bên trái nhất hoặc bên phải nhất của kiểu dữ liệu.

- Các thuộc tính **high, low** trả lại chỉ số của phần tử cao nhất hoặc thấp nhất của kiểu dữ liệu.

-Các thuộc tính **range** và **reverse_range** xác định khoảng cách của chỉ số .

- Thuộc tính **length** đưa ra số lượng các phần tử của một BIT_VECTOR.

-Các thuộc tính **even** và **stable** chỉ có đối với các tín hiệu. Các thuộc tính này chỉ rằng trên đường tín hiệu đang xét có xuất hiện sự kiện hay giá trị trên đường tín hiệu ổn định tại thời điểm hiện tại.

f. Các nhóm

Các nhóm kết hợp một hoặc nhiều giá trị vào một giá trị kết hợp của kiểu mảng hoặc kiểu bản ghi. Nhóm được dùng để gán giá trị cho đối tượng kiểu mảng hoặc bản ghi khởi tạo trong các biểu thức gán. Việc đánh địa chỉ số các phần tử được đặc tả theo tên hoặc theo vị trí .

-Đặc tả theo tên: Sự tương ừng của các phần tử của nhóm và phần tử của đối tượng được gán được chỉ rõ theo tên của từng phần tử và gián trị của chúng.

- Đặc tả theo vị trí: Mỗi phần tử nhận giá trị của chúng trong biểu thức theo trật tự .

Khi gán giá trị cho nhóm, ta có thể không cần thiết đặt giá trị cho tất cả các phần tử của nhóm .

g. Các biểu thức định kiểu

Các biểu thức định kiểu là những biểu thức hoặc nhóm dùng để xác định rõ những tình trạng mập mờ. Ví dụ như trong trường hợp hai kiểu liệt kê có những giá trị liệt kê giống nhau. Cú pháp của biểu thức định kiểu như sau:

Tên_kiểu(biểu_thức)

Biểu _thức phải có kiểu trùng với tên_kiểu.

h. Chuyển đổi kiểu

Phép chuyển đổi kiểu cung cấp phương tiện biến đổi giá trị của những đối tượng thuộc những kiểu có quan hệ chặt chẽ với nhau. Ví dụ như kiểu **real** và kiểu **integer**. Cú pháp của phép biến đổi kiểu như sau:

Tên_kiểu(biểu_thức)

6.5 Các cấu trúc tuần tự

Trong ngôn ngữ VHDL, kiến trúc xác định chức năng của thực thể. Trong kiến trúc chứa phần khai báo của các kiểu, các tín hiệu, các hằng, các thành phần và các chương trình con. Theo sau phần khai báo là các cấu trúc thực hiện đồng thời. Các cấu trúc thực hiện đồng thời có thể là các biểu thức gán tín hiệu song song, các khối và các lệnh khởi tạo phiên bản của thành phần. Các lệnh thực hiện đồng thời được kết nối với nhau bằng những tín hiệu. Mỗi khối lệnh thực hiện đồng thời trong một kiến trúc xác định một đơn vị tính toán bao gồm các thao tác tính toán như: Đọc tín hiệu, thực hiện các tính toán trên các tín hiệu đó và gán các giá trị tính được cho các tín hiệu ra.

Trong ngôn ngữ VHDL, một kiến trúc thực hiện đồng thời là một quá trình (process). Qúa trình là một cấu trúc quan trọng được sử dụng để mô tả hành vi hoạt động của mạch. Trong một kiến trúc tất cả các quá trình sẽ được thực hiện đồng thời khi mô phỏng.

Một quá trình được xây dựng từ những cấu trúc tuần tự - hay còn gọi là các lệnh tuần tự. Trong thời gian mô phỏng, các lệnh tuần tự trong một quá trình sẽ được thực hiện lần lượt trong một chu trình vô hạn bắt đầu từ lệnh thứ nhất đến lệnh thứ n và sau đó việc thực hiện quá trình lại quay trở lại lệnh đầu. Việc thực hiện một quá trình trong quá trình mô phỏng trên ngôn ngữ VHDL bị dừng lại khi gặp lệnh **wait** và được kích hoạt lại khi có sự

thay đổi trạng thái của ít nhất một trong các tín hiệu nằm trong danh sách các tín hiệu tác động.

Các lệnh tuần tự trong ngôn ngữ VHDL gồm có:

- Câu lệnh gán cho biến.

- Câu lệnh gán cho tín hiệu.

- Câu lệnh **if;**

- Câu lệnh **case;**

- Câu lệnh rỗng **null;**

- Các lệnh vòng lặp;

Phép gán biến

Trong ngôn ngữ VHDL, phép gán biến có tác dụng tương tự phép gán ở ngôn ngữ lập trình truyên thống. Phép gán biến thiết lập giá trị mới cho biến. Cú pháp của phép gán biến như sau:

Biến:=biểu_thức;

Vế trái của biểu thức phải là biến được khai báo từ trước. Vế phải của phép gán là biểu thức. Để phép gán có thể thực hiện được biểu thức ở vế trái và biến ở vế phải của phép gán phải cùng kiểu.

Khi một biến được gán giá trị, phép gán được thực hiện với thời gian mô phỏng bằng không. Điều đó có nghĩa là sự thay đổi giá trị của biến được xảy ra tức thời ngay tại thời điểm mô phỏng hiện tại.

Các biến chỉ được khai báo trong các quá trình hoặc chương trình con và được sủ dụng để lưu trữ các kết quả trung gian. Một biến được khai báo bên trong môth quá trình hoặc chương trình con sẽ tồn tại cục bộ trong quá trình hoặc chương trình con đó và không thể được truy cập tới từ cấu trúc song song khác.

2. Phép gán tín hiệu

Trong ngôn ngữ VHDL, tín hiệu là một dạng đối tượng đặc biệt. Phép gán tín hiệu dùng để thay đổi giá trị của tín hiệu. Các tín hiệu luôn được biểu diễn kết hợp với biểu diễn thời gian. Phép gán tín hiệu thay đổi giá trị của tín hiệu tương ứng theo thời gian và phụ thuộc vào các mô hình quá trình trễ trong các phần tử mạch. Khi tín hiệu được gán giá trị, giá trị mới đó không được gắn với tín hiệu một cách tức thời mà phải sau một thời gian được định lịch trình trước trong thời điểm mô phỏng tiếp theo tương ứng với mô hình trễ. Phép gán tín hiệu có cú pháp như sau:

tín_hiệu_đích <= [**transport**] *biểu_thức* [**after** *thời_gian*]

Trong một quá trình, việc gán giá trị của biểu thức cho tín hiệu sẽ được làm trễ khi chu trình mô phỏng đang thực hiện và được kiểm soát bởi toán tử **wait**.

Khi tín hiệu được gán giá trị trong quá trình, phép gán sẽ xác định một bộ điều khiển tín hiệu. Trong một quá trình, một tín hiệu chỉ có thể có một điều khiển, điều đó có nghĩa là bên trong một quá trình tín hiệu chỉ có thể xuất phát từ một nguồn. Nếu tín hiệu được gán giá trị trong nhiều quá trình khác nhau, chúng ta nói rằng tín hiệu có nhiều điều khiển.

Như đã đề cập trong chương ba, trong thiết kế mạch, chúng ta phân biệt hai dạng thời gian trễ khi ta điều khiển các thao tác trên tín hiệu theo thời gian: thời gian trễ quán tính và thời gian trễ lan truyền.

- Thời gian trễ quán tính được thể hiện mặc định trong ngôn ngữ VHDL. Giá trị thời gian trễ quán tính là độ dài giới hạn cần thiết của tín hiệu tác động để thiết kế có thể phản ứng với sự xuất hiện tín hiệu đầu vào. Nếu thời gian tồn tại của tín hiệu đầu vào không vượt quá giá trị thời gian trễ quán tính thì mạch sẽ không phản ứng với sự thay đổi của tín hiệu.

- Thời gian trễ lan truyền là thời gian trễ xuất hiện khji tín hiệu đi qua mạch.

Từ khóa **transport** được dùng trong trường hợp thời gian trễ trong phép gán là thời gian trễ lan truyền. Nếu không sử dụng từ khóa **transport** trong phép gán tín hiệu, thời gian trễ sẽ được coi là thời gian trễ quán tính một cách mặc định.

3. Câu lệnh if

Câu lệnh **if** tạo nên phân nhánh trong khi thực hiện chương trình. Tùy theo kết quả của biểu thức điều kiện mà có thể hoặc một số lệnh hoặc không có lệnh nào sẽ được thực hiện. Câu lệnh **if** có cấu trúc như sau:

if < *điều_kiện*> **then**

{ < *câu_lệnh_tuần_tự*>}

{**elsif <***điều_kiện*> **then** }

{ < *câu_lệnh_tuần_tự*>}

[**else** { < *câu_lệnh_tuần_tự*>}]

end if

Trong mỗi nhánh của toán tử **if** có thể chứa một hoặc nhiều câu lệnh tuần tự. Đầu tiên biểu thức điều kiện được tính toán, nếu kết quả cho giá trị **true**, các câu lệnh tuần tự nằm sau từ khóa **then** sẽ được thực hiện tức thời. Trong trường hợp ngược lại, biểu thức điều kiện sau từ khóa **elsif** cho giá trị **true**, các câu lệnh sau từ khóa **then** tiếp theo sẽ được thực hiện…

4. Câu lệnh case

Câu lệnh **case** được sử dụng trong trường hợp một biểu thức để kiểm soát nhiều rẽ nhánh trong chương trình VHDL. Các lệnh tương ứng với một trong các lựa chọn sẽ được thực hiện nếu biểu thức kiểm soát có giá trị bằng giá trị tương ứng của lựa chọn. Câu lệnh **case** có cú pháp như sau:

case <*biểu_thức*> **is**

when <*lựa_chọn*> =>

{ < *câu_lệnh_tuần_tự*>}

{ **when** <*lựa_chọn*> =>

{ < *câu_lệnh_tuần_tự*>}}

end case

5. Câu lệnh rỗng null

Câu lệnh rỗng có cú pháp như sau:

Null;

Trong ngôn ngữ VHDL, khi chương trình mô phỏng gặp phải câu lệnh null, nó sẽ bỏ qua lệnh này và thực hiện câu lệnh tiếp theo. Thông thường câu lệnh null dùng để chỉ trường hợp không thực hiện lệnh một cách tường minh khi có các điều kiện trả lại giá trị true. Do đó câu lệnh null thường được dùng trong các câu lệnh case đối với các lựa chọn không cần thao tác.

6. Các lệnh vòng lặp

Lệnh lặp loop chứa thân vòng lặp gồm dãy các câu lệnh sẽ được thực hiện không hoặc nhiều lần. Câu lệnh loop có cú pháp như sau:

[<nhãn>:][<sơ_đồ_lặp>] **loop**

{ < lệnh_tuần_tự>}

{**next** [<nhãn>][**when**<điều_kiện>];}

{**exit** [<nhãn>][**when**<điều_kiện>];}

end loop

Với những vòng lặp không chứa <sơ_đồ_lặp>, các lệnh trong dãy lệnh tuần tự sẽ được thực hiện cho tới khi được ngắt bởi câu lệnh exit. Trong ngôn ngữ VHDL, câu lệnh next cũng có thể được dùng để thay đổi trình tự thực hiện thân của vòng lặp.

Vòng lặp chứa <sơ_đồ_lặp> dạng for là một dạng khác của vòng lặp. Vòng lặp for là câu lệnh tuần tự nằm trong quá trình process, và cho phép thân của vòng lặp thực hiện theo số lượng xác định các lần lặp.

Sơ đồ lặp while là sơ đồ lặp trong đó quá trình lặp được thực hiện nếu biểu thức điều kiện lặp nhận giá trị true. Vòng lặp dừng khi giá trị của biểu thức điều kiện trở thành false hoặc quá trình thực hiện thân vòng lặp gặp lệnh exit. Cũng tương tự như vòng lặp for, câu lệnh next cũng được dùng để thay đổi trật tự lặp.

Đối với các vòng lặp trong các nhánh chứa phép gán tín hiệu phải có ít nhất một câu lệnh wait. Nếu không thỏa mãn điều kiện này, quá trình mô phỏng có thể khác đi. Chúng ta hãy xét ví dụ đoạn chương trình sau.

```
Signal S: integer range 0 to 10;
Process
Variable I: integer range 0 to 10;
Begin
    Wait until ( CLK'event and CLK= '0');
    I := 0;
    While (I < 10) loop
        S <=I;
        I := I+1;
    End loop;
End process;
```

7. Câu lệnh next

Câu lệnh next chỉ dùng trong các vòng lặp. Lệnh này có tác dụng loại bỏ việc thực hiện các câu lệnh nằm giữa câu lệnh next và cuối vòng lặp khi điều kiện trong câu lệnh được nghiệm đúng. Lệnh next có cấu trúc cú pháp như sau:

Next [< nhãn_vòng_lặp >][**when** <điều_kiện>]

Trong trường hợp có các vòng lặp lồng nhau thì việc thực hiện lệnh next sẽ được xác định một cách tường minh băng < nhãn_vòng_lặp >. Nếu không có nhán vòng lặp trong câu lệnh, lệnh next có tác dụng lên vòng lặp trong cùng chứa lệnh next.

8. Câu lệnh exit

Câu lệnh exit có thể được dùng bên trong các vòng lặp. Câu lệnh này có tác dụng bỏ qua các lệnh còn lại của vòng lặp và thực hiện ngay lệnh tiếp sau của vòng lặp vừa kết thúc. Lệnh exit có cấu trúc cú pháp như sau:

Exit [< nhãn_vòng_lặp >][**when** <điều_kiện>]

9. Câu lệnh wait

Lệnh wait điều khiển bộ mô phỏng ngừng việc thực hiện các quá trình hoặc các chương trình con cho tới khi điều kiện bên trong câu lệnh được nghiệm đúng.

Ta có thể nói rằng điều kiện trong câu lệnh wait chỉ có thể được nghiệm đúng khi xuất hiện các sự kiện trên đường tín hiệu. Như vậy, các đối tượng dữ liệu tham gia trong điều kiện phải là các tín hiệu. Các điều kiện để tiếp tục của quá trình bị dừng có thể được biểu thị dưới 3 dạng sau đây trong ngôn ngữ VHDL:

Wait

[**on** <tên_tín_hiệu >{, <tên_tín_hiệu>}]

[**until** <biểu_thưc_lôgic>]

[**for** <biểu_thức_thời_gian>];

- Câu lệnh wait on: chỉ cho chúng ta danh sách các đường tín hiệu và bộ mô phỏng sẽ chờ sự kiện (sự thay đổi trạng thái các tín hiệu).

- Câu lệnh wait until: sẽ dừng việc thực hiện quá trình cho tới khi biểu thức nhận giá trị true. Câu lệnh wait loại này sẽ tạo ra một danh sách ngầm định các tín hiệu tác động trong biểu thức lôgic. Mỗi khi có bất ký một sự kiện xuất hiện trên đường tín hiệu trong danh sách này, biểu thức lôgic sẽ được tính. Trong trường hợp lệnh wait until không chứa biểu thức lôgic, chúng ta hiểu rằng câu lệnh sẽ là wait until true.

-Câu lệnh wait for: sẽ dừng việc mô phỏng quá trình một thời gian bằng giá trị thời gian được chỉ định bên trong điều kiện. Sau khoảng thời gian được chỉ định, bộ mô phỏng thực hiện lệnh tiếp theo sau lệnh wait. Nếu biểu thức thời gian không có, chũng ta hiểu câu lệnh wait có ý nghĩa như sau:

Wait for time ‘hight’

Điều này có nghĩa là chúng ta không chỉ định tường minh về thời gian chờ.

Trong mô phỏng mạch, lệnh wait có thể được dùng để thiết lập đồng hồ cho chế độ đồng bộ. Trong các mô hình thiết kế VHDL, lệnh wait ngừng quá trình thực hiện cho tới khi xuất hiện sườn dương hoặc sườn âm của tín hiệu.

10. Phép gọi chương trình con và lệnh return

Trong ngôn ngữ VHDL có 2 dạng chương trình con:

- Thủ tục: procedure có thể trả lại nhiều giá trị;

- Hàm: function chỉ trả lại 1 giá trị và có thể tham gia vào các biểu thức.

Câu lệnh return dùng để kết thúc hoạt động của một chương trình con và chỉ được sử dụng trong hàm hoặc thủ tục. Đối với hàm, sự có mặt của câu lệnh return là bắt buộc còn trong thủ tục thì không bắt buộc. Lệnh return có cú pháp như sau:

Return <biểu_thức>

6.6 Các cấu trúc song song

Trong ngôn ngữ VHDL, một cấu trúc có thể chứa một hoặc nhiều cấu trúc song song. Mỗi cấu trúc song song xác định một đơn vị tính toán bao gồm các thao tác đọc tín hiệu, thực hiện các tính toán trên giá trị tín hiệu và gán các tín hiệu cho các tín hiệu ra. Cấu trúc song song xác định các thành phần và các quá trình liên kết những thành phần đó bằng những cấu trúc và hành vi của các thực thể. Các cấu trúc song song sẽ được thực hiện đồng thời trong quá trình mô phỏng không phụ thuộc vào trật tự xuất hiện của chũng trong kiến trúc.

Trong ngôn ngữ VHDL có các cấu trúc song song sau:

- Cấu trúc process;

- Các phép gán tín hiện song song;

- Phép gán tín hiệu có điều kiện;

- Phép gán tín hiệu có lựa chọn;

- Khối;

-Phép gọi chương trình con song song;

1. Quá trình process

Quá trình tính toán process được tạo thành từ một tập hợp các câu lệnh tuần tự. Tất cả các quá trình process trong một thiết kế được thực hiện một cách song song. Tuy vậy, tại một thời điểm xác định chỉ có một câu lệnh tuần tự được thực hiện trong mỗi quá trình process. Một quá trình process liên kết với phần còn lại của thiết kế thông qua các thao tác đọc các giá trị từ các tín hiệu đầu vào, các cổng được khai báo ngoài quá trình

hoặcghi giá trị vào các tín hiệu, cổng đó. Một quá trình tính toán process được mô tả theo quy tắc cú pháp sau:

[<nhãn>:] **process**[(< danh_sách_các_tín_hiệu_tác_động>)]

{<phần_khai_báo>}

Begin

{<lệnh_tuần_tự>}

End process [<nhãn>];

- Các khai báo biến, khai báo hằng, khai báo kiểu, kiểu con;

- Thân chương trình con, khai báo các biệt danh, luật use.

Nếu quá trình chứa <danh_sách_các_tín_hiệu_tác_động> thì lúc đó quá trình này sẽ tương tự như quá trình không chứa danh sách tín hiệu tác động nhưng lại chứa lệnh wait ở vị trí câu lệnh cuối cùng trong quá trình:

Wait on <danh_sách_các_tín_hiệu_tác_động>;

Việc thực hiện một quá trình process bao gồm việc thực hiện lặp lại các cấu trúc tuần tự chứa bên trong thân chương trình. Sau khi câu lệnh cuối cùng được thực hiện, việc mô phỏng quá trình sẽ được bắt đầu lại từ câu lệnh tuần tự đầu tiên của quá trình. Điều này làm cho việc mô phỏng hoạt động của quá trình như một vòng lặp vô hạn bao gồm tất cả các câu lệnh tuần tự bên trong quá trình. Việc thực hiện mô phỏng quá trình process có thể bị dừng lại bằng câu lệnh wait và có thể bị kích hoạt lại khi xuất hiện sự kiện trên các đường tín hiệu trong danh sách tín hiệu tác động.

2. Các phép gán tín hiệu song song

Một dạng khác của phép gán tín hiệu trong ngôn ngữ VHDL là phép gán tín hiệu song song. Phép gán này được sử dụng bên ngoài các **process** và bên trong các kiến trúc. Dạng đơn giản nhất của phép gán tín hiệu song song co cấu trúc cú pháp như sau:

< tín_hiệu_đích><=<biểu_thức>[**after** <biểu_thức_thời_gian>];

Trong đó, <tín_hiệu_đich> là tín hiệu nhận giá trị của <biểu_thức>. Cũng giống như trường hợp phép gán tín hiệu tuần tự, luật **after** sẽ được bộ tổng hợp mạch bỏ qua.

Phép gán tín hiệu song song tương đương với một quá trình **process** chứa phép gán tín hiệu.

3. Phép gán tín hiệu có điều kiện

Phép gán tín hiệu có điều kiện là câu lệnh song song thực hiện phép gán giá trị của các biểu thức cho một tín hiệu đích tùy theo các điều kiện đặt ra. Trong các biểu thức của lệnh gán, ngoại trừ biểu thức cuối cùng, những biểu thức khác được đi kèm với điều kiện gán. Khi một điều kiện nào đó nhận giá trị băng **true,** biểu thức tương ứng với điều kiện sẽ được gán cho tín hiệu đích. Tại một thời điểm thời gian, chỉ có một biểu thức được sử dụng cho phép gán. Phép gán tín hiệu có điều kiện được mô tả theo quy tắc cú pháp sau:

<tín_hiệu_đích><={ <biểu_thức>[**after** <biểu_thức_thời_gian>]

When<điều_kiện> else}

<biểu_thức>[**after** <biểu_thức_thời_gian>];

4. Phép gán tín hiệu theo lựa chọn

Phép gán tín hiệu theo lựa chọn thực hiện phép gán cho một tín hiệu đích với biểu thức **with**. Giá trị của biểu thức lựa chọn nằm sau từ khóa **with** được sử dụng giống như lệnh **case**. Phép gán được thực hiện khi có xuất hiện sự kiện làm thay đổi giá trị của biểu thức lựa chọn. Cú pháp của câu lệnh được biểu diễn như sau:

With <biểu_thức_lựa_chọn> select

<tín_hiệu_đích><={<biểu_thức>[**after** <biểu_thức_thời_gian>]

When <tín_hiệu_lựa_chọn>,}

<biểu_thức>[**after** <biểu_thức_thời_gian>]

When <giá_trị_lựa_chọn>;

5. Khối

Khối bao gồm một tập hợp các câu lệnh song song. Một kiến trúc có thể được phân tách thành một số các cấu trúc lôgic. Mỗi khối biểu diễn một thành phần của mô hình và thường được sử dụng để tổ chức một tập hợp các cấu trúc song song phân cấp. Khối được biểu diễn theo quy tắc cú pháp sau:

< nhãn >: **block**

{< phần_khai_báo>}

Begin

{<câu_lệnh_song_song>}

End block [<nhãn>];

< phần_khai_báo> xác định các đối tượng tồn tại cục bộ trong khối và có thể có các dạng sau:

- Khai báo hằng, kiểu, kiểu con và tín hiệu;
- Thân chương trình con;
- Khai báo các biệt danh;
- Khai báo các thành phần;
-Luật use;

Trình tự của mỗi <câu_lệnh_song_song> trong khối không quan trọng bởi vì các câu lệnh luôn được kích hoạt. Các <câu_lệnh_song_song> luôn truyền thông tin thông qua các tín hiệu. Các đối tượng được khai báo trong một khối block thì xác định trong toàn bộ khối, bao gồm cả các khối con. Nếu trong một khối con có khai báo một đối tượng trùng tên với một đối tượng ở khối bao nó, khi đó khai báo của đối tượng ở khối con sẽ che lấp đối tượng ở khối bên ngoài.

6. Gọi chương trình con song song

Phép gọi chương trình con song song tương đương với các quá trình process bao gồm các phép gọi chương trình con tuần tự tương ứng. Mỗi phép gọi chương trình con song song tương đương với một quá trình process không chứa dãy danh sách các tín hiệu tác động, phần khai báo rỗng và phần thân chứa một phép gọi chương trình con, tiếp theo là một câu lệnh **wait.**

6.7 Các chương trình con và các gói chương trình

1. Các chương trình con

Trong ngôn ngữ VHDL có hai dạng chương trình con là procedure và function. Các chương trình con có thể được sử dụng tại mọi vị trí trong mô tả VHDL. Các gói chương trình xác định tên của mọi đối tượng có thể được chia sẻ giữa các thực thể.

- Các thủ tục procedure sẽ được gọ đến như một câu lệnh và có thể trả lại nhiều giá trị. Một thủ tục procedure được phép thay đổi giá trị các đối tượng tương ứng với các tham số hình thức của procedure. Như vậy, các tham số của một thủ tục procedure có thể có các dạng in, out và inout.

- Các hàm function sẽ được sử dụng như một biểu thức và chỉ được phép trả lại duy nhất một giá trị. Hàm function thường được sử dụng để thực hiện tính toán trên các giá trị của tham số và không có mục đích thay đổi giá trị của các đối tượng được gắn kết với tham số. Do đó các tham số của hàm phải có dạng in và thuộc nhóm các tín hiệu signal hoặc hằng constant. Đối với hàm function, chúng ta phải mô tả kiểu của giá trị trả lại.

Một mô tả chương trình con được phân chia thành hai phần: phần khai báo chương trình con và thân của chương trình con. Phần khai báo đưa ra các thông tin về giao diện của chương trình con và phần thân chương trình con mô tả các chức năng của chương trình con.

Phần khai báo chương trình con được mô tả theo quy tắc cú pháp sau:

< khai_báo_chương_trình_con>::=

Procedure <ten_thu_tuc><danh_sach_tham_so>

Funtion < tên_hàm> <danh_sach_tham_số>

Return<kiểu_giá_trị_lại>;

< danh_sach_tham_so> ::= [class]ư<danh_sach_tên>

[mode] <kiểu>[:= <biểu_thức>];

Trong đó:

Nhóm class: constant, variable, signal;

Dạng mode: in, out , inout.

Nếu không được chỉ rõ, tham số sẽ được coi rằng có mode là in một cách mặc định. Cũng tương tự như vậy, tham số có dạng in sẽ có class là hằng constant một cách mặc định và tương ứng của dạng out và inout là biến variable.

Phần thân của chương trình con được mô tả theo quy tắc cú pháp sau:

< khai_báo_chương_trình_con> **is**

{ < phần_khai_báo_của_chương_trình_con>}

Begin

{< lệnh_tuần_tự >}

End [<tên_chương_trình_con >];

Khi gọi chương trình con, các đối tượng thực tế tương ứng với tham số hình thức lớp variable phải là các biến; tương ứng với lớp constant phải là hằng số hoặc biểu thức và tương ứng với lớp signal phải là tín hiệu. Các hằng số và biến được truyền theo giá trị, còn tín hiệu được truyền theo địa chỉ. Do đó đối với lớp đối tượng tín hiệu thì mọi tác động lên tham số truyền vào thân chương trình con cũng chính là tác động trực tiếp lên tín hiệu được truyền vào.

2. Các hàm quyết định

Chúng ta biết rằng, mỗi tín hiệu đều xuất phát từ một nguôn. Nói cách khác là mỗi tín hiệu có một điều khiển. Trong nhiều trường hợp, ví dụ như khi các đường tín hiệu bị chậm lại, tín hiệu đi ra khỏi nút chập sẽ được tổng hợp từ các tín hiệu đi vào nút theo một luật xác định. Trong trường hợp này, chúng ta nói rằng tín hiệu đi ra khỏi nút chập có nhiều điều khiển. Ngôn ngữ VHDL cho phép chúng ta có thể xác định các tín hiệu xuất phát từ nhiều nguồn nếu sử dụng những hàm quyết định. Những hàm quyết định này dùng để xác định giá trị của đường tín hiệu từ những giá trị nhận được từ nhiều nguồn điều khiển.

Đối với quá trình mô phỏng, các hàm quyết định có thể là mọi hàm bao gồm cả chương trình viết trên ngôn ngữ VHDL. Trong những mạch thực, kết quả của các hàm quyết định là liên kết các tín hiệu và cùng cho qua một phần tử lôgic đặc biệt có một đầu ra. Trong kỹ thuật, chỉ có một số giới hạn. Trong các phần cứng chỉ có một vài kiểu thực

hiện các hàm quyết định như liên kết dạng Or, liên kết dạng And hoặc liên kết dạng ba trạng thái

Cac tín hiệu được các hàm quyết định tạo ra nếu khai báo của tín hiệu chứa cả các hàm quyết định hoặc khai báo kiểu con của tín hiệu chứa hàm quyết định. Ví dụ:

Signal NODE: WIRE_AND BIT;

Subtype RESOLVED_STD is WIRE-OR STD_ULOGIC;

Trong ví dụ này, khai bao thứ nhất cho chúng ta thấy tín hiệu NODE là tín hiệu được xác định với hàm quyết đinh là WIRE_AND. Mỗi khi xuất hiện sự kiện trên đường tín hiệu NODE, hàm WIRE_AND được gọi và trả lại giá trị kiểu BIT – là giá trị của tín hiệu NODE. Khai báo thứ hai xác định kiểu con được quyết định RESOLVED_STD. Các tín hiệu được khai báo thuộc kiểu RESOLVED_STD là những tín hiệu được quyết định. Dưới đây chúng ta thấy cách để xác định và sử dụng các tín hiệu được quyết định:

- Xác định kiểu tín hiệu (nếu cần thiết);

- Xác định hàm quyết định. Hàm này sẽ nhận các tín hiệu vào và trả lại tín hiệu thuộc kiểu đã được xác đinh.

- Khai báo kiểu con của kiểu tín hiệu kèm với hàm quyết định;

- Khai báo và sử dụng các tín hiệu được quyết định

3. Các gói chương trình

Các kiểu dữ liệu, hằng số và chương trình con có thể được khai báo bên trong các khai báo thực thể hoặc bên trong thân của các kiến trúc. Các khai báo này là cục bộ trong những kiến trúc tương ứng và thực thể khác không thể truy cập tới những đối tượng đó. Tuy nhiên trong nhiều trường hợp những kiến trúc khác nhau cũng có thể phải chia sẻ những đối tượng chung nào đó. Các gói chương trình trong ngôn ngữ VHDL cho phép chúng ta thực hiện các khai báo chung cho nhiều thực thể khác nhau. Một gói chương trình của ngôn ngữ VHDL được biểu diến thành hai phần: phần khai báo gói và thân của gói.

Phần khai báo gói mô tả các giao diện của gói và có cấu trúc cú pháp như sau:

Package < tên_của_gói **>is**

{< các_khai_báo_thuộc_gói>}

End [< tên_gói >];

Các khai báo thuộc gói có thể là các:

- Khai báo kiểu, kiểu con, tín hiệu, hằng, biệt danh;

- Khai báo thành phần, chương trình con;

- Luật use;

- Và có thể chứa cả các package khác.

Các khai báo tín hiệu trong gói tạo nên một số vấn đề trong quá trình tổng hợp mạch bởi vì một tín hiệu không thể phân phối giữa hai thực thể. Cách giải quyết thông dụng vấn đề này là tín hiệu sẽ được khai báo như một tín hiệu cục bộ. Nói cách khác, nếu hai thực thể sử dụng cùng một tín hiệu trong gói, mỗi thực thể sẽ nhận được một phiên bản sao chép của tín hiệu này.

Thân của một gói xác định các hành vi của gói. Thân của một gói luôn phải cùng tên với khai báo gói. Phần thân gói được bắt đầu bởi từ khóa package body. Các thông tin trong thân của gói không thể nhìn thấy được từ các thiết kế hoặc thực thể sử dụng gói đó. Như vậy từ các thiết kế và các thực thể chúng ta chỉ có thể truy cập tới các đối tượng trong gói thông qua các giao diện được xác định trong phần khai báo gói mà không thể can thiệp trực tiếp vào bên trong gói. Nói một cách khác gói là một hộp đen với các giao diện được đưa ra phần khai báo. Thân của gói được mô tả theo quy tắc cú pháp như sau:

Package body <ten_của_goi > **is**

{ < các_khai_báo_trong_than_gói>

End [< tên_của_gói>];

Trong phần khai báo trong thân gói có thể chứa các:

- Khai báo kiểu, kiểu con, hằng;

- Thân chương trình;

- Luật use.

Các phần tử được mô tả trong phần khai báo gói không thể nhìn thấy được một cách tự động từ các thư viện khác. Luật **use** đứng trước một đơn vị chương trình sẽ làm cho tất cả các phần tử trong phần khai báo gói có thể được truy cập từ đơn vị chương trình đó.

Hội đồng chuẩn hóa IEEE xác định hai thư viện cho ngôn ngữ VHDL là STD và IEEE. Mối thư viện chứa một số gói như:

- STANDARD và TEXTIO đối với thư viện STD. Gói STANDARD xác định các kiểu dữ liệu quan trọng như integer, boolean, Bit…

- STD_LOGIC_1164, NUMERIC_BIT, NUMERIC_STD đối với thư viện IEEE. Các gói này chứa các kiểu và những hàm quan trọng cho quá trình tổng hợp và mô phỏng bằng ngôn ngữ VHDL.

Trong chương này chúng ta đã khảo sát những đặc điểm cơ bản của ngôn ngữ VHDL – một ngôn ngữ mô tả phần cứng điển hình. Ngôn ngữ này được dùng để mô hình hóa mạch trong công nghệ thiết kế, chế tạo mạch với độ tích hợp cao và siêu cao. Những thành phần chính của ngôn ngữ như các lệnh tuần tự, các lệnh song song, các chương trình con, các phương pháp mô tả thực thể và phương pháp luận biểu diễn mạch theo cấu trúc, hành vi hoặc theo dòng dữ liệu đã được đề cập tới. Trong chương tiếp theo chúng ta sử dụng ngôn ngữ VHDL để xây dựng các mô hình những mạch lôgic cơ bản.

Chương VII: Mô hình hoá mạch bằng ngôn ngữ VHDL

7.1 Mô hình hoá trên mức cấu trúc

Các mạch số thường được biểu diễn theo một cấu trúc phân cấp các thành phần. Mỗi thành phần có tập hợp các cổng để liên kết với các thành phần khác. Một thiết kế bao gồm một tập các phiên bản của mạch được xét trên chi tiết mức độ khác nhau. Một phiên bản mạch trên một mức chi tiết cụ thể của thiết kế được tạo thành từ các mô tả những thành phần trên mức chi tiết đó và những phần tử này được liên kết thông qua các tín hiệu tại các cổng của chúng xét trên mức chi tiết này. Khi mô tả mạch bằng ngôn ngữ VHDL, việc phân tích thiết kế thông qua các khai báo thành phần mạch và các biểu thức mô tả của thành phần.

Khi mô tả hành vi của mạch, thành phần cơ sở của hành vi là các câu lệnh mô tả quá trình (**process**), còn khi mô tả cấu trúc, đơn vị cơ sở sẽ là các các câu lệnh mô tả phiên của thành phần. Những câu lệnh mô tả quá trình và các câu lệnh mô tả phiên bản thành phần cần được gói vào trong thân của mô tả kiến trúc. Các chức năng mô tả sẽ có chức năng phân cấp các đơn vị thiết kế. Một trong những đặc tính quan trọng của VHDL là ngôn ngữ này cho phép mô hình hóa thiết kế trên những mức độ trừu tượng và các chi tiết khác nhau. Nói một cách khác các biểu diễn kiến trúc có thể bao gồm các câu lệnh mô tả quá trình, các câu lệnh mô tả thành phần.

1. Khai báo thành phần

Một kiến trúc có thể sử dụng những thực thể đã được mô tả một cách độc lập. Những thực thể này chứa trong thư viện thiết kế và có thể được sử dụng trong các khai báo thành phần và các câu lệnh mô tả các phiên bản của thành phần. Trong phần mô tả của thiết kế, mỗi câu lệnh khai báo thành phần tương ứng với một thực thể. Để có thể mô phỏng hoặc tổng hợp một thiết kế nào đó, thực thể và các mô tả kiến trúc của tất cả các thành phần mạch phải được biên dịch trước trong thư viện thiết kế.

Trong quá trình mô phỏng hoặc tổng hợp thiết kế, thực thể và mô tả kiến trúc đối với mọi thành phần cần phải được biên dịch trong thư viện thiết kế.

Cách mô tả cấu trúc thành phần cũng tương tự như cách thức mô tả thực thể. Trước hết để mô tả cấu trúc của thành phần, chúng ta phải xác định rõ các giao diện của thành phần, các giao diện này chính là các đường tín hiệu vào và ra khỏi thành phần. Trước khi được sủ dụng các thành phần phải được khai báo một cách tường minh theo quy tắc cú pháp sau:

Component<tên_thành _phần>

[**port**(<khai_báo_cục_bộ>)]

End Component;

2. Mô tả các phiên bản của thành phần

Các thành phần mô tả trong kiến trúc có thể được khỏi tạo bàng các câu lệnh tạo phiên bản thành phần. Tại các vị trí được tạo ra, phiên bản của thành phần được biểu diễn thông qua các thuộc tính bên ngoài như tên, kiểu hoặc hướng tín hiệu tại các cổng vào/ra, nhưng các tín hiệu bên trong thành phần không được biểu diễn tường minh, các câu lệnh khởi tạo sinh ra các phiên bản của thành phần và kết nối chúng để tạo lên một danh sách liên kết của thiết kế.

Câu lệnh tạo phiên bản thành phần có cấu trúc cú pháp như sau:

<nhãn_khởi_tạo>:<tên_thành_phần>

Portmap([<tên_cổng_cục_bộ>=>]<biểu_thức>

{[<tên_cổng_cục_bộ>=>]<biểu_thức>});

cấu truc **port map** ánh xạ các cổng của phần tử vào các tín hiệu. Ánh xạ này có thể hiểu như việc kết nối cổng tương ứng của phần tử vào đường tín hiệu. Các cổng được mô tả tromg khai báo thành phần là những cổng cục bộ. Trong khi đó, các cổng của các phiên bản thành phần được gọi là công thức. Trong các câu lệnh tạo phiên bản thành phần, cấu trúc **port map** đặt tương ứng mỗi cổng thực của phiên bản với một cổng cục bộ của thành phần. Mỗi cổng thực phải là đối tượng dạng *tín hiệu*. Trong ngôn ngữ VHDL đưa ra hai phương pháp ánh xạ các cổng cục bộ vào các cổng thực: ánh xạ theo vị trí và ánh xạ theo tên.

- Khi sử dụng ánh xạ theo vị trí, chúng ta đưa ra danh sách các tín hiệu tuân theo đúng trật tự mà cổng được khai bao.

-Đối với trường hợp ánh xạ theo tên, chúng ta sử dụng cấu trúc ánh xạ tường minh đặt tương ứng mỗi cổng với các tín hiệu thực.

3. Cấu trúc Generate

Câu lệnh khởi tạo Generate là câu lệnh song song được định nghĩa bên trong các kiến trúc và được dùng để mô tả các phiên bản thành phần. Câu lệnh **Generate** có cấu trúc như sau:

<nhán_khởi_tạo> : <sơ_đồ_khởi_tạo**> generate**

{<các_câu_lệnh_song_song>}

End generate [<nhãn_khởi_tạo >];

Trong ngôn ngữ VHDL, có hai loại sơ đồ khởi tạo: sơ đồ khởi tạo **for** và sơ đồ khởi tạo **if.**

- Sơ đồ **for** dùng để mô tả các cấu trúc có tính quy luật. Sơ đồ này đưa ra các tham số khởi tạo và các bước lặp.

- Thông thường trong nhiều thiết kế sự xuất hiện và liên kết giữa các phần tử không tuân theo quy luật lặp. Đối với những trường hợp này, ngôn ngữ VHDL cung cấp khả năng mô tả mạch sử dụng sơ đồ **if.** Khác với câu lệnh **if** tuần tự, câu lệnh **if** khởi tạo không chứa các nhánh **else** và **elsif.**

4. Các cấu hình

Các khai báo thành phần và các phiên bản của thành phần chỉ chứa mô tả bên ngoài của một phần tử mạch trên một mức chi tiết nhất định. Các thực thể này không thể sử dụng được trong quá trình mô phỏng bởi vì chúng không chứa những thông tin mô tả về chức năng của phần tử. Trong ngôn ngữ VHDL, cơ chế liên kết này được thể hiện trong các *cấu hình* của phần tử.

Sau khi tất cả các thực thể và kiến trúc của từng phần tử được biên dịch và chứa trong các thư viện, chúng ta có thể tiến hành dịch mô hình của bộ cộng đầy đủ, chương trình dịch sẽ đối sánh từng định dạng của các thành phần trong mô tả thiết kế với các thực thể trong thư viện đang sử dụng. Nếu ứng với mỗi thành phần trong thiết kế có thể tìm thấy thực thể mô phỏng cho mô hình thiết kế của bộ cồng FullAdder.

Trong trường hợp một thực thể có nhiều kiến trúc thì việc chọn mặc định kiến trúc được dịch sau cùng có thể không lựa chọn được kiến trúc cần thiết, do đó cần phải biểu diễn cấu hình một cách tường minh.

5. Biểu diễn cấu hình

Một thực thể có thể có nhiều kiến trúc. Mỗi kiến trúc có thể là mô hình thuật toán, kiến trúc khác có thể là mức thanh ghi truyền đạt hoặc có thể là mô hình cấu trúc. Trên khía cạnh thiết kế, chúng ta cần lựa chọn các kiến trúc của phần tử một cách thích hợp và phải ghi rõ các khai báo thực thể và các kiến trúc sẽ được lựa chọn. Như vậy, một cấu hình sẽ cung cấp cho chúng ta cách lựa chọn những khai báo thực thể và kiến trúc đối với những phiên bản của thành phần được mô tả trong thân kiến trúc của thiết kế.

Cấu hình của một đặc tả thành phần được biểu diễn theo quy tắc cú pháp sau:

For<danh_sách_phiên_bản>:<tên_thành_phần>**use**<đặc_tả_gắn_kết>

7.2 Mô hình hoá trên mức thanh ghi truyền đạt

Một thiết kế trên mức thanh ghi truyền đạt bao gồm một tập hợp các thanh ghi lien kết với các mạch tổ hợp. Trong mục này, chúng tôi sẽ trình bày mối quan hệ giữa các cấu trúc trên mức thanh ghi trong ngôn ngữ VHDL và mạch lôgic sẽ được tổng hợp.

1.Mô hình hóa tổ hợp

Một quá trình không chứa câu lệnh **if** với tín hiệu điều khiển theo sườn lên hoặc sườn xuống hoặc không chứa câu lệnh **wait** với các sự kiện của tín hiệu gọi là quá trình tổ hợp. Một quá trình tổ hợp sẽ được tổng hợp bằng các mạch tổ hợp.

Để mô tả mạch lôgic tổ hợp, các biến và tín hiệu trong một quá trình **process** không được nhận giá trị gán khởi tạo trước bởi vì mạch tổ hợp không chứa các phần tử nhớ. Khi trong mô hình mạch có các biến hoặc tín hiệu được khởi tạo giá trị trước, điều này sẽ tương đương với việc trong mạch phải có những phần tử lưu trữ các giá trị khởi tạo. Như vậy khi mô hình hóa cấu trúc, chương trình mô phỏng sẽ sinh ra các phần tử nhớ để lưu trữ các giá trị khởi tạo. Mạch trở thành mạch có nhớ. Thêm vào đó, trong các mô hình mạch tổ hợp, các tín hiệu và biến cần phải được gán giá trị trước khi được sử dụng.

Mọi câu lệnh tuần tự trừ các lệnh **wait, loop** và **if** với những tín hiệu điều khiển theo sườn đều có thể dùng để mô tả các mạch lôgic tổ hợp. Các phép toán số học như +, - , *,.. ; các phép toán quan hệ và các phép toán lôgic đều có thể được sử dụng trong biểu thức.

Công cụ tổng hợp có thể thực hiện chia sẻ tài nguyên nếu có các thao tác loại trừ trong biểu diễn thiết kế.

2. Các mạch trigo điều khiển theo mức (mạch lật)

Các mạch lật là các trigo làm việc với chế độ đồng bộ theo mức. Các trigo làm việc theo sườn và trigo làm việc theo mức thường được sử dụng trong các phần tử nhớ một bit.

Nói chung, hành ci của các mạch trigo làm việc theo mức sẽ được xây dựng trên những câu lệnh điều kiện **if** không đầy đủ. Có thể hiểu câu lệnh điều kiện **if** không đầy đủ là câu lệnh **if** chỉ có một nhánh **then** và không chứa nhánh **else.** Như vậy, mọi tín hiệu hoặc biến không được điều khiển bởi tất cả các khả năng có thể có của điều kiện đều được mô phỏng thành những phần tử trigo làm việc theo mức.

Để tránh sự xuất hiện của các phần tử trigo làm việc theo mức không mong muốn, chúng ta phải gán tín hiệu với tất cả các khả năng có thể có của điều kiện trong các câu lệnh rẽ nhánh.

Chúng ta có thể mô tả những phần tử trigo làm việc theo mức với hai đầu tín hiệu thiết lập giá trị ‘0’ hoặc ‘1’ không đồng bộ. Đoạn chương trình dưới đây sẽ biểu diễn phần tử trigo làm việc theo mức có giá trị sẽ thiết lập về ‘0’ khi tín hiệu

đầu vào không đồng bộ RST nhận giá trị băng ‘ 1’. Như vậy trong ví dụ này, tín hiệu RST là tín hiệu kích hoạt với mức giá trị tín hiệu cao.

Ví dụ, nếu RST = 1 => thiết lập trạng thái của trigo về ‘0’.

```
Signal S,RST, Din, Dout : BIT;
Process (S,RST, Din )
Begin
    If (RST =’1’) then
        Dout <= ‘0’;
    Else (S =’1’ ) then
        Dout <= Din;
    End iF;
End process;
```

Nếu chũng ta muốn đổi tín hiệu RST thành tín hiệu kích hoạt với mức giá trị thấp, điều kiện trong câu lệnh **if** sẽ chuyển từ (RST = ‘1’) thành (RST =’ 0’).

3. Xây dựng mạch đồng hồ hai pha

Các mạch đồng hồ hai pha có thể được mô tả bằng cách sử dụng các trigo làm việc theo mức. Mạch đồng hồ hai pha được biểu diễn bằng hai quá trính. Trong đó, một quá trình mô tả mạch tổ hợp và mạch lật tầng một, quá trình mô tả mạch tổ hợp và mạch tật tầng thứ hai.

Ví dụ, thiết kế của mạch đồng hồ hai pha sẽ được mô tử bằng đoạn chương trình trên ngôn ngữ VHDL như sau:

```
Entity TwoPhase is
    Port( A,B: in BIT ;Ph1, Ph2: in BIT; Z: buffer BIT );
End TwoPhase;
Architecture Implement oF TwoPhase is
    Signal D:BIT;
Begin
```

```
Process (A,Z,Phi_1)
Begin
    If (Phi_1 = '1') then
        D <= A or Z;
    End iF;
End process;
Process( B,D, Phi_2)
Begin
    If (Phi_2 ='1') then
        Z <= B and (not D);
    End iF
End process
End Implement;
```

4. Các mạch trigo làm việc theo sườn (flip –flop)

Các quá trình chứa các tín hiệu **if** hoặc **wait** điều khiển theo sườn lên (hoặc sườn xuống) là các quá trình được định giờ. Các mạch trigo điều khiển theo sườn lên hoặc sương xuống của tín hiệu sẽ được tạo ra từ mô tả trên ngôn ngữ VHDL nếu phép gán tín hiệu (hoặc phép gán biến) được thực hiện theo sườn lên hoặc sườn xuống của các tín hiệu điều khiển.

Thuộc tính **event** của tín hiệu được dùng để biểu diễn sự biến thiên của tín hiệu. Khi chúng ta cần xác định sự biến thiên của tín hiệu, thuộc tính **event** sẽ cho ra giá trị lôgic tùy theo trên đường tín hiệu có xuất hiện sự kiện hay không. Thuộc tính **stable** cũng cho giá trị lôgic và có s nghĩa ngược lại với thuộc tính **event.**

Trong các chương trình trên ngôn ngữ VHDL, các biến cũng có thể siinh ra các trigo điều khiển theo sườn tín hiệu. Chúng ta biết rằng, khi một biến được khai báo trong một quá trình **process,** giá trị của biến sẽ không rời khỏi quá trình (có nghĩa là giá trị cảu biến được khai báo bên trong một quá trình sẽ không được sử

dụng ở bên ngoài quá trình). Như vậy, thời điểm mà một biến sẽ sinh ra một trigo điều khiển theo sườn trong quá trình mô phỏng là thời điểm khi biến được sử dụng trước lúc được gán giá trị bên trong quá trình VHDL sẽ tính hai phần tử làm việc theo sườn khi được mô phỏng.

Ví dụ: đoạn chương trình tạo ra hai phần tử trigo điều khiển theo sườn trong quá trình mô phỏng.

```
Signal CLK, Din, Dout: BIT;
Process (CLK)
Variable TMP: BIT;
begin
    if (CLK'event and CLK ='1' ) then
        DOut <= TMP;
        TMP := Din;
    End if;
End process;
```

5. Thiết lập và xóa giá trị đồng bộ và không đồng bộ trong các mạch trigo điều khiển theo sườn tín hiệu (flip – flop)

Giá trị của các phần tử flip – flop có thể được lập hoặc xóa một cách đồng bộ dựa vào sự xuất hiện tín hiệu trên các đầu vào tương ứng của phần tử. Phép gán giá trị chỉ được thực hiện khi xuất sườn của tín hiệu đồng bộ. Còn trong các thời điểm không có tín hiệu đồng bộ, sự thay đổi giá trị tín hiệu trên các đầu thiết lập hoặc xóa không ảnh hưởng tới trạng thái của phần tử nhớ.

Tất cả các phép gán bên trong câu lệnh **if** phản ứng với sườn (lên hoặc xuống) của tín hiệu sẽ có trác dụng thiết lập trạng thái của phần tử nhớ trong toàn bộ thời gian hình thành sườn tín hiệu đồng hồ. Phần tử flip – flop với các đầu thiết lập hoặc xóa trạng thái có thể được mô hình hóa bằng cách xác định thời điểm đầu vào đồng bộ sẽ được thiết lập theo sự xuất hiện sườn tín hiệu đồng hồ. Đoạn chương trình dưới đây sẽ tương đương với mạch flip – flop có cấu trúc được mô tả hình 7.10.

Ví dụ:

Signal CLK, Din, Dout, SRDT: BIT;

Process (CLK);

If (CLK'event **and** CLK ='1') **the**n

If (SRST = '1') **then**

Dout ← '0';

Else

Dout ← Din;

End if

End if;

End process;

Trong nhiều trường hợp, việc thiết lập hoặc xóa giá trị của phần tử nhớ không phụ thuộc vào thời gian đồng hồ.

6. Sơ đồ chung mô tả mạch có nhớ

Nếu xét trên quan điểm cấu trúc và chức năng, mạch có nhớ luôn được chia thành hai phần: các thành phần tổ hợp và các mạch nhớ đồng bộ. Tương ứng với cách phân chia chức năng và cấu trúc như vậy, mô hình của mạch có nhuos bao gồm hai phần: một phần mô tả các cấu trúc và chức năng tổ hợp; một phần mô tả các cấu trúc và chức năng mạch nhớ đồng bộ.

- Trong phần mô tả các thành phần tổ hợp, chúng ta biểu diễn những thành phần mạch có hành vi phụ thuộc vào sự thay đổi giá trị tín hiệu nằm trong danh sách tín hiệu tác động. Tất cả các tín hiệu được tham chiếu tới trong phần mô tả các thành phần tổ hợp phải được liệt kê trong danh sách các tín hiệu tác động.

-Trong phần mô tả các mạch nhớ đồng bộ, chúng ta biểu diễn các thành phần mạch có hành vi phụ thuộc vào sự xuất hiện các sườn của tín hiệu đồng bộ. Các thao tác đối với tín hiệu trong phần này chứa các phép kiểm tra sự kiện trên đường tín hiệu và kiểm tra sườn tín hiệu .

Ví dụ, đoạn chương trình sau mô tả mạch tạo xung có sơ đồ thiết kế ở mức lôgic như trên hình 7.12.

```
Entity Pulser is
    Port ( CLK, PB: in BIT; Pulse: out BIT);
End Pulser;
Architechture BHV of Pulser is
    Signal Q1, Q2: BIT;
Begin
    Process( CLK, Q1, Q2 )
    Begin
        If ( CLK'event and CLK ='1' ) then
            Q1 <= PB;
            Q2 <= Q1;
        End if;
        Pulse <= ( not Q1 ) nor Q2;
    End process;
End BHV;
```

Còn phần chức năng nhớ đồng bộ theo sườn lên được biểu diễn bằng câu lệnh if kiểm tra sự kiện:

```
If ( CLK'event and CLK = '1') then
    Q1 <= PB;
    Q2 <= Q1;
End if;
```

7. Các thanh ghi

Các dạng thanh ghi khác nhau cũng thường được sử dụng trong các mạch có nhớ. Hình vẽ biểu diễn sơ đồ của thanh ghi bốn bit làm việc trong chế độ đồng bộ

bằng tín hiệu đồng hồ CLK. Thanh ghi được thiết lập giá trị ban đầu bằng "1111" khi xuất hiện tín hiệu thiết lập không đồng bộ ASYNC. Khi xuất hiện sườn lên của tín hiệu đồng bộ CLK, các bit đầu vào của thanh ghi được truyền tới đầu ra đồng thời. Với thanh ghi như thế, chúng ta có đoạn chương trình trên ngôn ngữ VHDL tương ứng mô tả hành vi của mạch.

Ví dụ:

```
Signal CLK, ASYNC : BIT;
Signal Din, Dout: BIT_VECTOR (3 downto 0);
Process (CLK, ASYNC )
Begin
    If (ASYNC ='1') then
        Dout  <= "1111";
    Elsif (CLK'event and CLK ='1' ) then
        Dout <= Din;
    End if;
End process;
```

7.3 Mô hình hoá các automat hữu hạn

Một thiết kế mạch số có thể được chia làm hai thành phần: bộ xử lý dữ liệu và bộ điều khiển. Mối quan hệ giữa bộ điều khiển và bộ xử lý dữ liệu trong mạch được biểu diễn trên hình 7.14.

Bộ xử lý dữ liệu thực hiện các thao tác đối với dữ liệu chứa trong các phần tử nhớ theo các lệnh do bộ điều khiển đưa ra. Bộ điều khiển đưa ra các lệnh điều khiển thích hợp cho bộ xử lý dữ liệu tại mỗi thời điểm thời gian. Với những lệnh điều khiển từ bộ điều khiển, bộ xử lỹ dữ liệu có thể thực hiện chức năng và thao tác thích hợp để xác định được các tín hiệu đầu ra cần thiết. Bộ điều khiển nhận những thông tin phản hồi từ bộ xử lý dữ liệu – các thông tin trạng thái xử lý dữ liệu. Các thông tin phản hồi này được sử dụng làm các biến quyết định để xác định dãy chuyển trạng thái của mạch.

Bộ điều khiển là những mạch tuần tự có các trạng thái được dùng để xác định các lệnh điều khiển cho hệ thống. Tại một trạng thái hiện thời, dựa vào những thông tin trạng thái và thông tin đầu vào (thông tin đầu vào do bộ xử lý dữ liệu cung cấp hoặc là các tín hiệu thiết lập từ bên ngoài) Bộ điều khiển chuyển sang trạng thái mới và khởi tạo các lệnh điều khiển mới và tạo ra các giá trị ra tương ứng. Bộ điều khiển thường được xây dựng từ các mạch tuần tự - các thanh ghi trạng thái và các mạch tổ hợp. Các thanh ghi trạng thái lưu giữ các trạng thái hiện thời, còn mạch tổ hợp tạo ra các mạch điều khiển và trạng thái mới dựa trên trạng thái hiện thời và các tìn hiệu đầu vào. Các mạch tuần tự có một số hữu hạn các trạng thái gọi là otomat hữu hạn.

Máy otomat hữu hạn là một bộ sáu $< X,Y,S,s_0, \delta, \lambda >$, trong đó :

1 X – Tập các tín hiệu vào của otomat:

$X=\{x_1(t),\ldots, x_n(t)\}$

2 Y- Tập các tín hiệu ra của otomat:

$Y=\{y_1(t),\ldots, y_n(t)\}$

3 S-Tập các trạng thái của otomat:

$S=\{s_1(t),\ldots, s_n(t)\}$

4 s_0 – Trạng thái ban đầu của otomat:

$s_0(t) \in S$

5 Hàm δ (s,x) – hàm chuyển trạng thái của otomat

6 Hàm λ (s,x) – hàm đầu ra của otomat

Tương ứng với các phương pháp tính toán, hàm chuyển trạng thái và hàm ra, chúng ta có các loại otomat khác nhau. Hai loại otomat thông dụng là otomat Moore và otomat Mealy. Chúng ta sử dụng ngôn ngữ VHDL để mô tả 2 mô hình otomat này.

1. Mô hình hóa Otomat Moore

Otomat Moore là một otomat hữu hạn có hàm chuyển trạng thái và hàm ra có dạng như sau:

$$s(t) = \delta(s(t-1),x(t))$$

$$y(t) = \lambda(s(t)) \quad t= 1,2,\ldots$$

Trong otomat moore, tín hiệu đầu ra ở thời điểm hiện thời chỉ phụ thuộc vào trạng thái hiện thời của otomat; còn trạng thái ở thời điểm hiện thời sẽ được tính thông qua tín hiệu đầu vào tại thời điểm hiện thời và trạng thái trước đó của otomat

Theo sơ đồ khối trên hình 7.15, otomat moore có thể biểu diễn bao gồm một mạch tổ hợp để xác định trạng thái mới của otomat thông qua tín hiệu vào và trạng thái trước đó; một hệ mạch nhớ để lưu giữ trạng thái. Đối với otomat moore, tín hiệu đầu ra chỉ phụ thuộc vào trạng thái hiện thời, do đó, chúng ta cần một mạch tổ hợp nữa để xác định tín hiệu ra. Mạch tổ hợp xác định tín hiệu ra sẽ được nối trực tiếp với hệ mạch nhớ trạng thái bởi vì tín hiệu ra chỉ phụ thuộc vào trạng thái của otomat ở thời điểm hiện thời. Các phần tử nhớ của otomat moore thường được điều khiển bằng tín hiệu đồng hồ, do đó các tín hiệu ra cũng được đồng bộ hóa theo đồng hồ. Như vậy otomat moore là một otomat với đầu ra đồng bộ. Vì lẽ đó chúng ta không cần quan tâm tới những khó khăn có thể xuất hiện do các quá trình quá độ hoặc cạnh tranh giữa các phần tử trong mạch của otomat gây ra. Các tín hiệu ra được tính qua mạch tổ hợp nhờ có tín hiệu từ các hệ nhớ của trạng thái.

Chúng ta có thể biểu diễn otomat moore trên ngôn ngữ VHDL tương tự như khi biểu diễn các mạch có nhớ trên mức thanh ghi. Quá trình biểu diễn otomat hữu hạn sẽ được chia thành 2 phân hệ: phân hệ tổ hợp và phân hệ mạch tuần tự. Tín hiệu hóa trạng thái không đồng bộ khởi tạo giá trị cho các thanh ghi và đưa otomat về trạng thái ban đầu. Cùng với sự xuất hiện sườn tín hiệu đồng bộ CLK, giá trị của trạng thái mới được gán cho trạng thái hiện thời. Đối với otomat moore, tín hiệu đầu ra chỉ phụ thuộc vào trạng thái hiện thời nên mạch tổ hợp xác định đầu ra không kết nối với các tín hiệu đầu vào.

2. Mô hình hóa otomat Mealy

Otomat mealy có các hàm chuyển trạng thái và hàm ra được biểu diễn theo hệ thức sau:

$$S(t) = \delta(s(t-1),x(t))$$

$$y(t) = \lambda(s(t)) \quad t= 1,2,\ldots$$

Đối với otomat mealy, các tín hiệu đầu ra phụ thuộc vào trạng thái và tín hiệu đầu vào ở thời điểm hiện thời. Để ngăn chặn sự thay đổi giá trị đầu ra trong thời gian hiện tại của xung đồng bộ, chúng ta phải đồng bộ hóa hoạt động của otomat mealy không đồng bộ. Để đạt được điều này, tín hiệu đi vào hệ mạch nhớ trạng thái phải được đồng bộ bằng xung đồng hồ và khi đó tín hiệu đầu ra phải được xác định chỉ trong thời gian là sườn của xung đồng hồ.

Otomat mealy cũng được mô tả trong ngôn ngữ VHDL bằng 2 phân hệ: phân hệ mạch tổ hợp xác định hàm ra với hàm chuyển trạng thái và phân hệ mạch nhớ đồng bộ.

Otomat mealy có thể được xây dựng theo 2 dạng: dạng mạch không đồng bộ đầu ra và dạng mạch đồng bộ đầu ra

Xét ví dụ xây dựng mạch điều khiển thực hiện cộng hai số dấu phẩy tính. Otomat mealy tương ứng có giản đồ chuyển trạng thái. Otomat mealy nhận được các thông số sau:

Các tín hiệu vào là: $X= \{x_1,x_2,x_3\}$;

Các tín hiệu ra là: $Y= \{y_1,y_2,y_3,y_4\}$;

Các trạng thái là: $S= \{S_0,S_1,S_2,S_3\}$;

Như vậy chúng ta thấy rằng, với cùng một thuật toán thiết kế theo mô hình mealy sẽ tiết kiệm trạng thái hơ so với thiết kế theo mô hình moore.Đoạn chương trình dưới đây biểu diễn otomat mealy theo thiết kế không đồng bộ của thuật toán cộng hai số có dấu phẩy tĩnh nêu ở mục trước:

Entity AsyncMealy **is**

Port (CLK,RST: in BIT;

```
        X: in BIT- VECTOR ( 3 downto 1);
        Y: out BIT – VECTOR ( 4 downto 1);
End AsyncMealy;
Architecture Implement of AsyncMealy is
  Begin
        Process (CLK,RST,X)
              Type StateType is (S0,S1,S2,S3);
              Variable State, NextState: StateType;
        Begin
              If(RST ='1') then
                    State  = S0;
              Elsif (CLK'event and CLK ='1') then
                    State = NextState;
              End if;
              Case State is
                    When S0 =>
                          If (X(1) ='1') then
                                Y(1) = '1';
                                NextState := S1;
                          Elsif (X(2) = '1')then
                                Y(3) = '1';
                                NextState :=S2;
                          Else  Y(2) = '1';
                                NextState := S2;
                          End if;
                    When S1 =>
```

if (X(2) = ‘1’) **then**

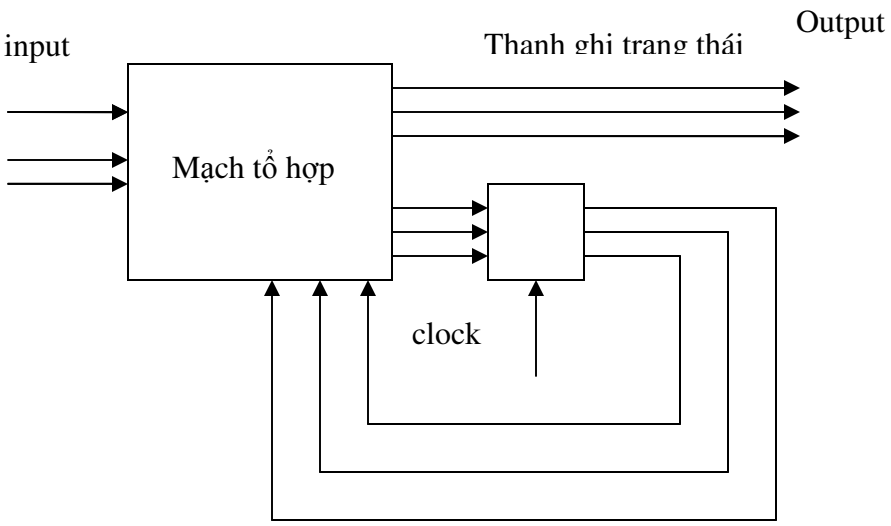
Y(3) =’1’;

NextState :=S2;

Else Y(2) = ‘1’;

NextState :=S2;

End if;



Hình 7.19:Sơ đồ nguyên lý của ôtômat mealy
Hoạt động theo cơ chế không đồng bộ

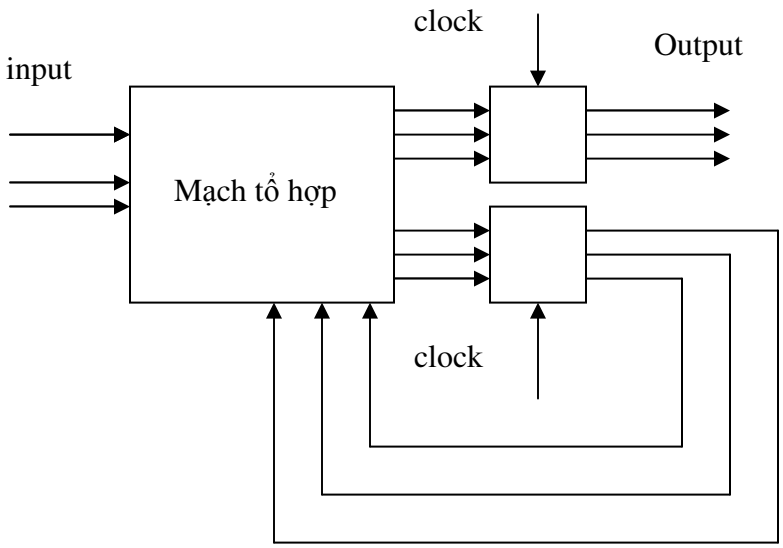
Ta thấy rằng trong otomat mealy không đồng bộ sự xuất hiện ra không đồng bộ với sự xuất hiện của tín hiệu đầu vào, điều này có thể dẫn tới những kết quả thực hiện phép tính sai. Để khắc phục nhược điểm này của oto mat mealy không đồng bộ, chúng ta sẽ thêm vào trong mạch một hệ nhớ đồng bộ. Hệ nhớ này nằm giữa mạch tổ hợp tính toán hàm ra với đầu ra của otomat. Thông thường các phần tử của mạch

nhớ đồng bộ này là các phần tử flip-flop. Các phần tử flip-flop này được đồng bộ theo 2 sườn giống như thanh ghi trạng thái. Khi xuất hiện xung động hồ, hệ mạch nhớ đầu ra nhận giá trị ra do mạch tổ hợp xác định hàm ra tính toán được. Trong toàn bộ khoảng thời gian sau khi xung đồng hồ thiết lập sườn, hệ mạch nhớ không thay đổi trạng thái. Đối với VD trên thiết kế với mạch đồng bộ đầu ra sẽ được biểu diễn bằng ngôn ngữ VHDL như sau đây:

```
Entity AsyncMealy is
Port ( CLK,RST: in BIT;
       X: in BIT- VECTOR ( 3 downto 1);
       Y: out BIT – VECTOR ( 4 downto 1);
End AsyncMealy;
Architecture Implement of AsyncMealy is
  Begin
       Process (CLK,RST,X)
            Type StateType is (S0,S1,S2,S3,S);
            Variable State: StateType;
       Begin
            If(RST ='1') then
                 State  = S0;
            Elsif (CLK'event and CLK ='1') then
                 State = NextState;
            End if;
            Case State is
```

```
When S0 =>
        If (X(1) ='1') then
                Y(1) = '1';
                State := S1;
        Elsif (X(2) = '1'               )then
                Y(3) = '1';
               State :=S2;
        Else  Y(2) = '1';
                State := S2;
        End if;

…..
```



Hình :Sơ đồ nguyên lý của ôtômat mealy
Hoạt động theo chế độ đồng bộ đầu ra

Kết luận: Như vậy, trong chương này chúng ta đã đưa ra những phương pháp mô hình hóa cấu trúc và chức năng của những mạch logic cơ bản- các mạch tổ hợp và các mạch tuần tự sử dụng những cấu trúc cơ bản trong ngôn ngữ VHDL.

Chương VIII: Các phương pháp kiểm tra lỗi mạch logic

8.1 Các mô hình lỗi logic

Bài toán phát hiện lỗi trong các mạch lôgic là bài toán xác định sơ đồ lôgic được thiết kế thực hiện được các chức năng đã đề ra. Để giải quyết bải toán này chúng ta cần phải xây dựng phương pháp phát hiện lỗi và cuối cùng là xây dựng mô hình lỗi. Ta đã thấy rằng theo quan điểm phân loại mạch dựa vào hoạt động của chúng, các mạch lôgic được chia thành các mạch tổ hợp và các mạch tuần tự.

Như đã thấy trong các mục trước, trên quan điểm về hoạt động, các mạch tổ hợp được thể hiện ở các trạng thái của các đầu ra ở mọi thời điểm thời gian được xác định hoàn toàn bằng các trạng thái của đầu vào tại cùng một thời điểm.

Nếu xét trên quan điểm cấu trúc, các mạch tổ hợp hoàn toàn không chứa vòng tín hiệu phản hồi. Nếu xét trên khía cạnh phát hiện lỗi, các mạch tổ hợp là những đối tượng nghiên cứu khá đơn giản.

Nếu xét trên quan điểm hành vi, trong hoạt động của các mạch tuần tự xuất hiện các trạng thái bên trong; còn xét trên quan điểm cấu trúc các mạch này chứa các vòng phản hồi. Điều đó làm cho việc phát hiện lỗi trong các mạch tuần tự là vô cùng phức tạp.

Vi mạch cần kiểm tra là mạch lôgic nên ta giả thiết rằng trong khi mạch có lỗi, mạch vẫn thực hiện các chức năng như một mạch lôgic. Các lỗi thỏa mãn điều kiện này đề gọi là các lỗi lôgic.

Các lối lôgic biểu hiện ảnh hưởng của các lỗi vật lý lên hành vi của các hệ thỗng được mô hình hóa. Vì trong quá trình mô hình hóa, các phần tử mạch chúng ta tách biệt các chức năng lôgic và hành vi thời gian; nên ta phân chia thành các nhóm lỗi sau:

- Nhóm các **lỗi ảnh hưởng đến các chức năng** lôgic của phần tử;
- Nhóm các **lỗi ảnh hưởng đến độ trễ** của tín hiệu đi qua phần tử;

Khi ta mô tả các lỗi vật lý như lỗi lôgic ta được các lợi điểm như sau:

- Thứ nhất là, bài toán lôgic lỗi trở thành bài toán lôgic hơn là bài toán vật lý.

- Thứ hai là, một số lỗi lôgic trở lên không phụ thuộc vào công nghệ theo nghĩa: cùng một mô hình lỗi có thể sử dụng trong nhiều công nghệ khác nhau.
- Thứ ba là, các bộ giá trị thử nghiệm để phát hiện lỗi lôgic có thể được sử dụng đỗi với các lỗi vật lý có hành vi trong mạch chưa hoàn toàn được hiểu rõ hoặc quá phức tạp để có thể phân tích.

Một mô hình lỗi có thể là mô hình ẩn hoặc mô hình tường minh.

- **Mô hình lỗi tường minh** xác định một không gian lỗi, trong đó từng lối được xác định độc lập.
- **Mô hình lỗi ẩn** xây dựng không gian lỗi bằng cách xác định có lựa chọn các lỗi tùy theo mức độ quan tâm, chủ yếu thông qua các tính chất của chúng.

Nếu cho trước lỗi lôgic và mô hình mạch, về nguyên lý chúng ta sẽ xác định được chức năng lôgic của mạch đối với sự tồn tại của lỗi này trong mạch. Tùy theo mô hình mạch các lỗi lôgic có thể được chia thành các loại sau:

- Các lỗi được xác định gắn liền với mô hình cấu trúc được gọi là các lỗi cấu trúc. Ảnh hưởng của các lỗi cấu trúc là làm thay đôi sự liên kết giữa các thành phần mạch.
- Các lỗi được xác định gắn liền với mô hình chức năng của mạch được gọi là các lỗi chức năng.

Mặc dù các lỗi ngẫu nhiên hoặc lỗi đột biến có mặt thường xuyên trong mạch, việc mô hình hóa những lỗi đó yêu cầu các dữ liệu thống kê về sự xuất hiện sắc xuất của chúng. Thông thường chúng ta không có đầy đủ những thông tin về mặt thống kê, do đó đối với những đột biết hoặc xuất hiện không thường xuyên, tốt nhất là sử dụng các phương pháp kiểm nghiệm trực tuyến. Trong giáo trình này chúng ta nghiên cứu đến **các lỗi thường trực**.

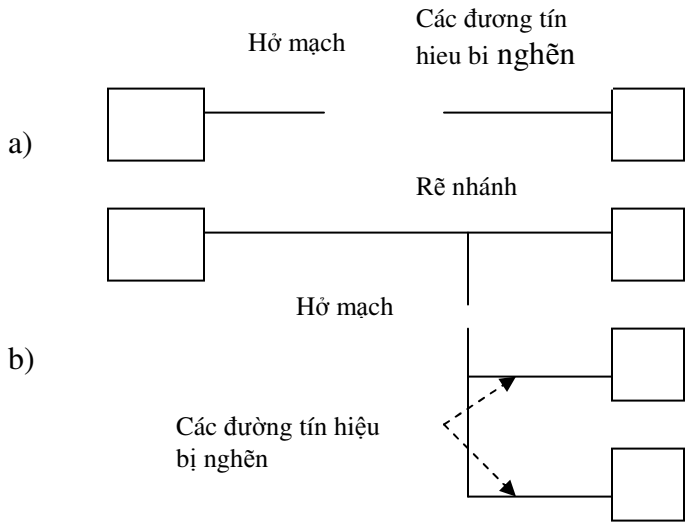
Việc làm đơn giản hóa bài toán phát hiện lỗi bằng **giả thiết về lỗi đơn lẻ** được biện minh bởi *chiến lược kiểm tra thương xuyên*. Chiến lực kiểm tra thường xuyên có thể được phát biểu như sau: Chúng ta cần kiểm tra hệ thống một cách

thường xuyên sao cho sác xuất xuất hiện nhiều hơn một lỗi giữa hai lần kiểm tra liên tiếp là đủ nhỏ. Những trường hợp xuất hiện nhiều hơn một lỗi có thể là:

- Các lỗi vật lý có thể xuất hiện trong mạch giữa hai lần kiểm tra là trong những lỗi vật lý đó, một số lỗi có thể tương ứng với nhiều lỗi lôgic. Điều này có khả năng xảy ra lớn đối với những mạch có độ tích hợp cao trong đó nhiều lỗi vật lý có thể ảnh hưởng tới bề mặt tinh thể trên đó có một số các thành phần mạch.
- Trong những mạch mới được sản xuất, thì trong những lần thử nghiệm đầu những lỗi kép có thể xuất hiện.
- Trong trường hợp những phép thử không phát hiện được những lỗi đơn lẻ ở bất kỳ lúc nào, mạch có thể chứa những lỗi chưa thể phát hiện. Những lỗi ẩn này, khi xuất hiện các lỗi đơn lẻ thứ hai giữa hai lần kiểm tra, sẽ tạo ra nhiều lỗi kép trong mạch.

Các lỗi đặc trưng do các đường kết nối tạo lên thường là: ngắn mạch và hở mạch.

- **Các lối ngắn mạch** là những lỗi xuất hiện các đương truyền không được phép liên kết bị chập.
- **Các lỗi hở mạch** là kết quả của sự đứt các kết nối. Trong nhiều công nghệ, sự hở mạch trên những đường tín hiệu một chiều với một nhánh phân kỳ sẽ làm cho đường tín hiệu vào đó trở thành bị ngắt và nhận một giá trị lôgic cố định.



Hình :Lỗi hở mạch
a)loic hở mạch đơn b)lỗi hở mạch kép

8.2 Bài toán phát hiện lỗi

1. Phát hiện lỗi trong các mạch tổ hợp.

Cho Z(x) là hàm logic của mạch tổ hợp N với x là vector giá trị đầu vào bất kỳ và Z(x) biểu diễn ánh xạ thực hiện bởi mach N. Với một vec tơ giá trị đầu vào cụ thể t = (x_1, x_2, …,x_n) ta sẽ có Z(t) là đáp ứng của mạch N đối với vecto t. Đối với mạch có nhiều đầu ra Z(t) cũng là vecto

Nếu trong mạch xuất hiện lỗi f, mạch N sẽ chuyển thành mạch N_f. ta giả thiết mạch N_f cũng là mạch tổ hợp với hàm chức năng $Z_f(x)$. Mạch được kiểm nghiệm bằng cách đặt dãy T của các vecto giá trị thử nghiệm t_1, t_2, …, t_m lên các đầu vào của mạch và so sánh những giá trị thu được trên đầu ra theo lý thuyết của mạch N tương ứng với những vecto đầu vào đó.

Ta có định nghĩa sau:

Vecto giá trị kiểm nghiệm t được gọi là phát hiện lỗi f nếu: $Z_f(t)<>Z(t)$. Khi áp dụng định nghĩa này cần chú ý:

- Các vecto giá trị kiểm nghiệm trong dãy T có thể được sử dụng không phụ thuộc vào trình tự áp dụng, đối với mạch tổ hợp N, dãy T được gọi là tập hợp các vecto giá trị kiểm nghiệm .

- Định nghĩa này không áp dụng được nếu mạch chứa lỗi N_f trở thành mạch tuần tự.

- Trong định nghĩa này chúng ta giả thiết rằng việc kiểm tra lỗi bằng cách đặt các giá trị thử nghiệm và thu nhận các kết quả thông qua các chân của phần tử và so sánh hoàn toàn của các kết quả nhận được.

2.Phát hiện lỗi trong mạch tuần tự

Kiểm tra mạch tuần tự phức tạp hơn so với mạch tổ hợp. Để có thể phát hiện được lỗi, chúng ta cần phải sử dụng chuỗi các vecto giá trị kiểm nghiệm và đáp ứng của mạch tuần tự sẽ là hàm đối với các trạng thái ban đầu của mạch. Có thể minh họa như sau:

Giả thiết T là chuỗi các vecto thử nghiệm và R(q, T) là đáp ứng của mạch tuần tự N đối với chuỗi T bắt đầu từ trạng thái q. Giả thiết rằng khi trong mạch N xuất hiện lỗi f, mạch N trở thành mạch N_f và $R_f(q_f, T)$ là đáp ứng của mạch chứa lỗi N_f đối với chuỗi T bắt đầu từ trạng thái q_f.

8.3 Các phương pháp thuật toán tổng hợp các giá trị thử nghiệm

1. Phương pháp dựa trên sự kích hoạt đường truyền

Các phương pháp tạo dãy giá trị thử nghiệm được chia làm hai nhóm:

- Các phương pháp tìm các vecto giá trị đầu vào có thể phát hiện một lỗi cho trước.

- Các phương pháp tìm các lỗi có thể được phát hiện sử dụng một vecto giá trị kiểm nghiệm cho trước. Các phương pháp thuộc nhóm này được gọi là các phương pháp mô hình hóa lỗi.

Phương pháp mô hình hóa lỗi không thể gọi là phương pháp thuật toán vì nếu quá trình mô hình hóa không được thực hiện với mọi khả năng có thể có của các vecto giá trị đầu vào, không thể đảm bảo được việc tìm ra các vecto giá trị kiểm nghiệm cho tất cả các lỗi có thể được phát hiện.

Các phương pháp thuật toán tạo các vecto giá trị thử nghiệm để phát hiện lỗi được sử dụng hiện nay đều dựa trên khái niệm kích hoạt đường truyền.

Kích hoạt đường truyền là phương pháp xác định các vecto giá trị thử nghiệm dựa trên việc tìm đường mà theo đó sự khác biệt giữa các giá trị tín hiệu khi mạch chứa lỗi và khi mạch không chứa lỗi được truyền ra bên ngoài. Nối cách khác là chúng phải xác định được các đường dẫn mẫn cảm với lỗi.

8.4 Phương pháp mô hình hoá lỗi

1. Bài toán mô hình hóa lỗi:

Việc tạo các vecto thử nghiệm để tìm lỗi bằng các phương pháp thuật toán cho phép ta thực hiện đối với mọi lỗi có thể phát hiện được. Như vậy phương pháp thuật toán là phwong pháp tạo vecto giá trị thử nghiệm tìm lỗi đầy đủ. Khi độ phức tạp của mạch tăng, thời gian để tạo bộ giá trị phát hiện lỗi sẽ tăng nhanh. Theo lý thuyết thời gian tạo bộ giá trị thử nghiệm sẽ tăng theo luật hàm mũ đối với số lượng phần tử logic tham gia vào mạch. Đối với những mạch trong thực tế, thời gian tạo bộ giá trị thử nghiệm sẽ tỷ lệ bậc hai – bậc ba so với sự tăng lên của số lượng các phần tử logic cơ bản trong mạch.

Phương pháp mô hình hóa lỗi là như sau: ta thực hiện mô hình hóa logic mạch chứa lỗi tương ứng với một vecto giá trị đầu vào nào đó. Nếu nhận được các giá trị đầu ra của mạch trong trường hợp xuất hiện lỗi và trong trường hợp không bị lỗi khác nhau, như vậy vecto giá trị đầu vào được coi là vecto giá trị thử nghiệm phát hiện lỗi đang xét. Phương pháp xác định các lỗi là phương pháp xác định các lỗi có thể được phát hiện sử dụng một vecto giá trị đầu vào cho trước.

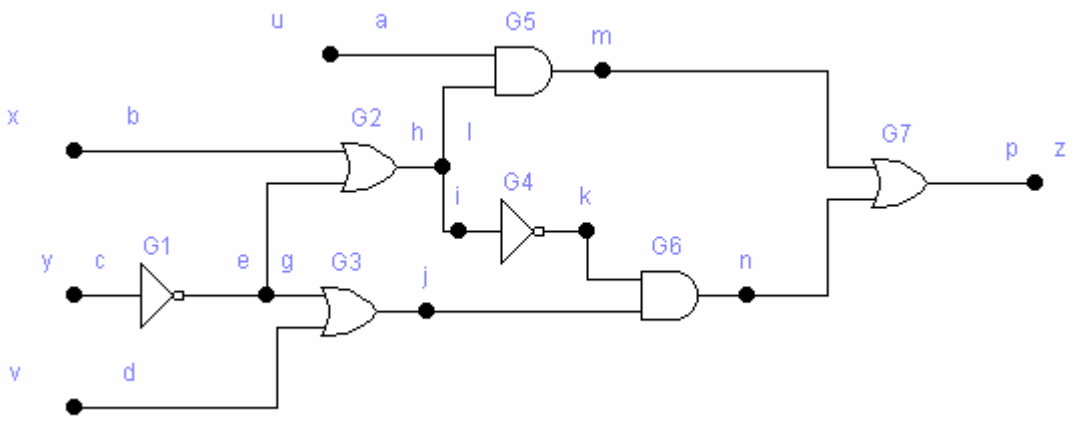
Phương pháp mô hình hóa lỗi về cơ bản dựa trên quá trình quá trình mô hình hóa logic mạch trong hai trường hợp: chứa lỗi và không chứa lỗi.

Có 3 phương pháp mô hình hóa lỗi: Phương pháp mô hình hóa lỗi song song; phương pháp mô hình hóa lỗi suy diễn; phương pháp mô hình hóa lỗi cạnh tranh.

1. Phương pháp mô hình hóa lỗi song song:

Khi thực hiện mô hình hóa lỗi, giá trị trên các đường tín hiệu được mô tả bằng một từ máy. Do đó một từ máy có chứa n hang chữ số nhị phân, thì chúng ta có thể thực hiện được quá trình mô hình hóa đối với n vecto giá trị đầu vào một cách song song.

Ví dụ:



Với một từ máy gồm 16 bit, ta có thực hiện quá trình mô hình hóa song song đối với tất cả các vecto giá trị đầu vào. Trong trường hợp này, các lỗi có thể được biểu diễn bằng cách thiết lập cố định các giá trị ‘0’ , ‘1’ trên các đường tín hiệu bị lỗi, không phụ thuộc vào giá trị đầu vào. Đối với mạch trên, chúng ta thực hiện quá trình mô hình hóa lỗi với các vecto giá trị đầu vào được biểu diễn trên các từ máy 8-bit; các giá trị trên các đường tín hiệu và trên đầu ra được biểu diễn như sau:

a:01010101 e:11110000 11110000 l:11110011 11111111

b:00110011 f:11110000 11110000 m:01010001 01010101

c:00001111 g:11110000 11110000 n:00001100 00000000

d:00000001 h:11110011 11111111 p:01011101 01010101

I:11110011 11111111 không lỗi bị lỗi

J:00001110 00001110

K:00001100 00000000

Khi mô hình hóa theo phương pháp trên, chúng ta thấy các giá trị đầu vào tham gia vào quá trình mô hình hóa một cách đồng thời trên tất cả các bit của từ máy do đó phương pháp này gọi là mô hình hóa lỗi với đầu vào song song.

Với mạch trên, thực hiện quá trình mô hình hóa một cách đồng thời với các lỗi:

g/0, g/1, h/0, j/0, j/1, k/0, k/1, m/1;

tương ứng với bộ giá trị đầu vào: (a, b, c, d) = (1, 0, 1, 0).

a:11111111 e:00000000 i:01000000 m:01000001

b:00000000 f:00000000 j:11101111 n:10101011

c:11111111 g:01000000 k:10111011 p:11101011

d:00000000 h:01000000 l:01000000

Mô hình hóa lỗi song song

Ta thấy bộ giá trị đầu vào như nhau (cùng băng “1010”) đối với tất cả các lỗi nêu trên. Các giá trị được sắp xếp trong từ máy tương ứng với các lỗi theo trình tự nêu trên. Sau khi thực hiện quá trình mô hình hóa logic với tất cả các lỗi đã cho, chúng ta nhận được từ máy biểu diễn các giá trị đầu ra p = “1101011”. Trong trường hợp nếu xuất hiện các lỗi hằng số j/0 hoặc k/0 giá trị tín hiệu ở đầu ra p khác với giá trị đầu ra của mạch trong trường hợp không có lỗi (trong trường hợp không có lỗi, đầu ra p nhận giá

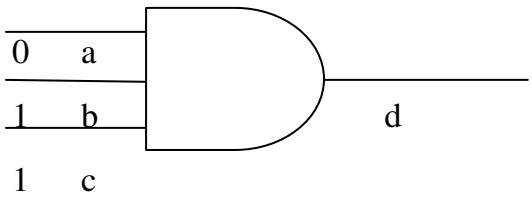
trị bằng ‘1’ tương ứng với vecto giá trị đầu vào “1010”, còn trong trường hợp xuất hiện lỗi j/0 hoặc k/0, đầu ra p nhận giá trị bằng ‘0’). Từ đó suy ra vecto giá trị đầu vào (1, 0, 1, 0) là vecto giá trị kiểm nghiệm để phát hiện các lỗi hằng số j/0, k/0.

2. Mô hình hóa lỗi suy diễn:

Theo phương pháp mô hình hóa lỗi song song, ta đưa ra vecto các giá trị đầu vào, các lỗi và sau đó kiểm tra khả năng phát hiện các lỗi đưa ra bằng các vecto giá trị đầu vào đang xét. Trong trường hợp thực hiện kiểm tra thành công, vecto giá trị đầu vào đang xét được coi là vecto giá trị kiểm tra phát hiện lỗi đã cho; nếu thực hiện mô hình hóa lỗi không thành công, chúng ta phải thực hiện quá trình mô hình hoá lỗi cho bộ giá trị đầu vào và lỗi khác. Như vậy, trong phương pháp mô hình hóa lỗi này, mối quan hệ logic giữa các lỗi và các vecto giá trị đầu vào không được thiết lập và do đó hiệu quả của quá trình tạo các bộ giá trị kiểm nghiệm phát hiện lỗi không cao.

Phương pháp mô hình hóa lỗi suy diễn là phương pháp sử dụng các phép toán suy diễn tập hợp để thực hiện quá trình mô hình hóa.

Ví dụ: Xét phần tử AND có 3 lối vào.



Nếu các đầu vào nhận giá trị a = ‘0’, b = ‘1’, c = ‘1’, đầu ra sẽ nhận giá trị d = ‘0’. Do đó các lỗi hằng số do vecto giá trị đầu vào (a, b, c) = (0, 1, 1) phát hiện được sẽ là: a/1, d/1 . Nếu các đầu vào a = b = c = ‘1’ ta có d = ‘1’. Suy ra tập hợp các lỗi hằng số có thể phát hiện được bằng vecto giá trị đầu vào (a, b, c) = (1, 1, 1) sẽ là $\{a_0, b_0, c_0, d_0\}$.

Chúng ta có thể nhận được tập hợp này theo những luật sau:

Luật suy diễn C: các phép tập hợp đối với phần tử A và các giá trị đầu vào của phần tử đó:

- Đường tín hiệu vào a của phần tử là đường vào từ bên ngoài. Nếu giá trị tín hiệu này bằng ‘1’, ta có $L_a = \{a_0\}$; nếu giá trị đường tín hiệu bằng ‘0’ thì $L_a = \{a_1\}$;

- Hàm ra của phần tử A được biểu diễn dưới dạng tổng hoặc tích. Nếu giá trị đúng của đường tín hiệu a bằng ‘0’, thì ký hiệu a trong biểu thức sẽ thay bằng L_a , ký hiệu $\bar{a}$ sẽ

được thay bằng $\overline{L_a}$. Nếu giá trị đúng của đường tín hiệu a bằng '1', thì ký hiệu a trong biểu thức sẽ thay bằng $\overline{L_a}$, ký hiệu $\bar{a}$ sẽ được thay bằng $\overline{L_a}$.

- Tích logic và tổng logic trong biểu thức được thay bằng các phép toán tập hợp giao và hợp. Nếu giá trị đúng của đầu ra bằng '0' thì biểu thức này được thay bằng L, nếu là '1'- $\overline{L}$

- Trong trường hợp giá trị đúng của đường tín hiệu ra f bằng '0' thì ta hợp tập hợp $\{f_1\}$ vào L; nếu giá trị đầu ra bằng '1', ta sẽ hợp tập hợp $\{f_0\}$ vào L. Kết quả là nhận được tập hợp L_f.

4. Mô hình hóa lỗi cạnh tranh:

Trong phương pháp mô hình hóa suy diễn đã trình bày ở trên, danh sách các lỗi có thể phát hiện bằng một vecto giá trị đầu vào cho trước được tìm thấy bằng cách lan truyền tuần tự lỗi theo các mức phân hạng.

8.5 Một số phương pháp làm đơn giản hoá quá trình kiểm tra phát hiện lỗi

Với sự phát triển của công nghệ VLSI, kích thước và độ phức tạp của các mạch số tăng nhanh. Khi số lượng các phần tử mạch tăng, thời gian cần thiết để tạo các bộ giá trị thử nghiệm phát hiện lỗi tăng. Đối với các mạch phức tạp như các bộ vi xử lý, bài toán kiểm tra lỗi trở nên hết sức phức tạp. Để giải quyết vấn đề này, chúng ta không thể chỉ giới hạn ở việc tìm ra những phương pháp kiểm tra có hiệu quả mà còn phải tìm cách thiết kế những mạch cho phép làm dơn giản hóa bài toán phát hiện lỗi ngay từ đầu.

Để làm đơn giản hóa quá trình kiểm tra và phát hiện lỗi, chúng ta cần phải giải quyết 2 bài toán:

- Bài toán thứ nhất là giảm thời gian tạo các vecto giá trị thử nghiệm. Bài toán này được giải quyết bằng cách xây dựng những thuật toán tạo vecto giá trị thử nghiệm hiệu quả hơn.
- Bài toán thứ 2 là giảm số lượng các phép kiểm tra và nhằm mục đích là làm giảm thời gian kiểm tra mạch.

Bên cạnh đó, khi thiết kế những mạch dễ kiểm tra theo các mục đích nêu trên lại nảy sinh ra các vấn đề mới:

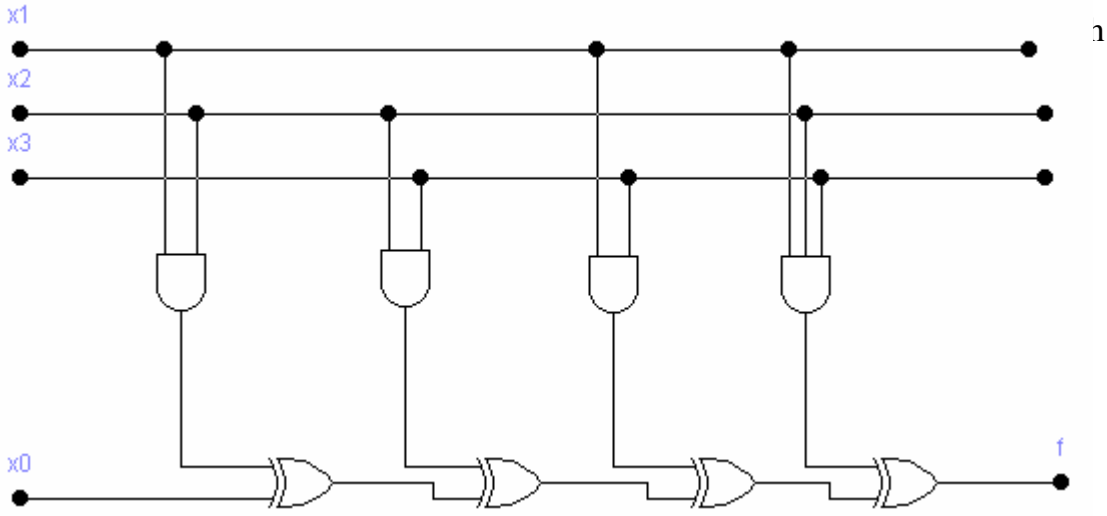
- Tăng số lượng thiết bị để làm đơn giản hóa quá trình kiểm tra mạch.
- Giảm tốc độ làm việc do hệ quả của việc làm đơn giản hóa quá trình kiểm tra.

Như vậy thiết kế mạch dễ kiểm tra có thể coi là thiết kế mạch logic với khả năng them một số lượng tối thiểu thiết bị bổ sung vào mạch mà không làm thay đổi chức năng và các tham số của mạch ban đầu kết hợp với khả năng thực hiện kiểm tra mạch bằng một số lượng nhỏ các vecto giá trị kiểm nghiệm nhận được từ những phương pháp đơn giản.

1. Các giới hạn về cấu trúc của sơ đồ:

   Xét phương pháp thiết kế những mạch dễ kiểm tra bằng việc dựa trên xây dựng những mạch logic có một dạng nhất định. Những mạch đó thường có cấu trúc dạng ma trận hai chiều. Vì cấu trúc của mạch có tính quy luật cao cho nên chúng ta có thể tổng hợp các vecto giá trị kiểm nghiệm cho những mạch loại này một cách tương đối dễ dàng.

   Mọi hàm logic có thể biểu diễn dưới dạng tổ hợp các phép toán XOR của các tích



   Các vecto giá trị thử nghiệm cho những mạch loại này có cấu trúc khá đơn giản. Cấu trúc này bao gồm giá trị ’0’ đối với một biến và giá trị ‘1’ đối với các biến còn lại.

   Nếu số lượng các tích của n biến logic bằng p, chúng ta có thể thấy rằng số lượng các vecto giá trị thử nghiệm sẽ bằng (2n+4) và số lượng cực đại các tầng mạch sẽ bằng p. Nếu so sánh với các phương pháp xây dựng các vecto giá trị thử nghiệm thông thường , số lượng các véc tơ giá trị thử nghiệm không lớn.

2. Biến thể của sơ đồ mạch và các phần tử them mới:

   Để đơn giản hóa bài toán phát hiện lỗi trong các mạch logic, ta cần tăng tính dễ quan sát và khả năng kiểm soát mạch. Do đó ta đưa vào trong mạch những phần

tử bổ sung và thực hiện việc biến đổi sơ đồ của mạch sao cho mạch không thay đổi về chức năng. Bên canh đó, ta cần phải xác định được mục đích chính khi nghiên cứu xây dựng mạch để kiểm tra.

Nếu xuất phát điểm là số lượng cực tiểu các bộ giá trị thử nghiệm, như vậy chúng ta có thể biến đổi mạch điện sao cho có thể phát hiện được lỗi chỉ cần ít nhất là ba bộ giá trị thử nghiệm. Số lượng này được đưa ra dựa vào nhận xét như sau: để có thể phát hiện được lỗi của các phần tử AND và OR có hai đầu vào, ta cần ít nhất 3 bộ giá trị kiểm nghiệm. Vì vậy có thể nói 3 bộ giá trị thử nghiệm là cận dưới theo lý thuyết.

Để phát hiện lỗi cho mạch AND có 2 đầu vào, chúng ta cần phải kiểm tra phần tử với 3 bộ giá trị đầu vào: (0, 1), (1, 0), (1, 1). Do đó phần tử AND có thể được kiểm tra bởi bất kỳ 2 trong số các dãy giá trị đầu vào sau:

$$S_A = \{011, 101, 110\}$$

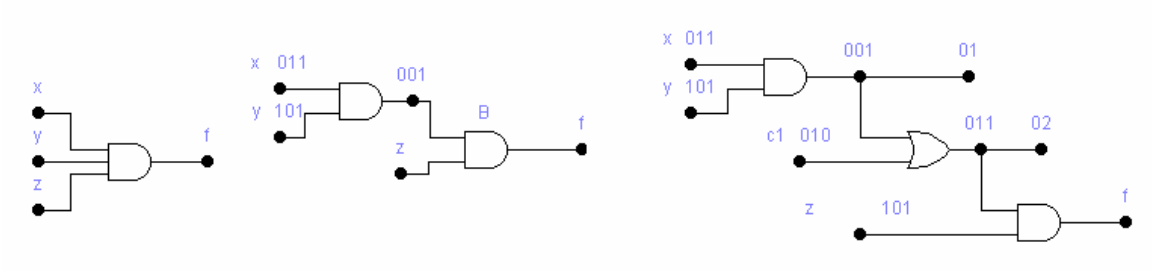
Đối với phần tử OR, việc kiểm tra có thể được đảm bảo bởi bất kỳ 2 trong số các dãy giá trị đầu vào sau:

$$S_0 = \{001, 010, 100\}$$

Để có thể phát hiện được lỗi dùng các dãy giá trị đầu vào nêu trên thì mạch cần phải được biến đổi như sau:

1. Mạch phải được chuyển đổi thành mạch chỉ chứa các phần tử AND, OR, NOT.
2. Phần tử AND (hoặc OR) có số lượng đầu vào là 3 hoặc hơn sẽ được triển khai thành tổ hợp liêm kết tuần tự của các phần tử AND (hoặc OR) có hai đầu vào.
3. Bắt đầu từ các đầu vào, chúng ta chọn các bộ giá trị thử nghiệm từ trong các chuỗi S_A và S_0 tương ứng với dạng của phần tử. Truyền các giá trị này tới đầu ra của mạch.
4. Nếu trong quá trình truyền giá trị, không nhận các dãy tín hiệu từ S_A hoặc S_0, chúng ta cần phải thêm vào đường tín hiệu phần tử AND hoặc OR có 2 đầu vào
5. Lặp lại bước 3 và bước 4 cho tới khi trên tất cả các phần tử trong mạch thiết lập được các giá trị thử nghiệm.

   Ví dụ:



Tài liệu học tập, tham khảo:

[1] Nguyễn Linh Giang, Thiết kế mạch bằng máy tính, nhà xuất bản KHKT

[2] Tống Văn On, Thiết kế mạch số với VHDL, nhà xuất bản KHKT

[3] Phạm Việt Bình, Thiết kế mạch số với VHDL, nhà xuất bản KHKT

[4] VHDL programing

[5] Thomas J.Wilderotter : A designer's guide to VHDL synthesis

[6] Zainalabedin Navabi : VHDL Analysis and Modeling of Digital Systems

[7] http://www.vhdl-online.de