Tạp chí Tin học và Điều khiển học, T. 20, S. 1 (2004), 11–22

THUẬT GIẢI DI TRUYỀN SONG SONG CHO BÀI TOÁN THIẾT KẾ MẠCH TỔ HỢP TRÊN NHÓM MÁY TÍNH MẠNG

TRẦN VĂN LĂNG[1], NGUYỄN HOÀNG ANH[2]

[1]*Phân viện Công nghệ thông tin tại Tp. Hồ Chí Minh*
[2]*Trung tâm xử lý Golden Pacific*

Abstract. In this paper we present a discussion and application of Parallel Genetic Algorithms (PGA) on the supercluster (SC) running Parallel Virtual Machine (PVM) to solve the combinational logic circuit design problem. The results done on a 128- node SC proved that the solutions from PGA give a performance better than those of Genetic Algorithms (GA). We used the virtual topologies such as ring, torus, hypercube to create the migration operator for individuals. The system speedup in theory and practice was compared.

Tóm tắt. Bài báo trình bày một khảo sát và ứng dụng của thuật giải di truyền song song trên SC với môi trường PVM cho bài toán thiết kế mạch tổ hợp. Các kết quả được thực hiện trên một nhóm máy tính mạng (SC) sử dụng 128 nút (node) cho thấy rằng lời giải sinh ra từ thuật giải di truyền song song có chất lượng cao hơn từ thuật giải di truyền tuần tự. Các sơ đồ kết nối vành khung (ring), vòng lưới (torus) và siêu cầu phương (hypercube) được sử dụng để thực hiện phép toán di trú cho các cá thể. Độ tăng tốc (speedup) của hệ thống từ thực nghiệm và lý thuyết được so sánh.

1. GIỚI THIỆU

Thiết kế nói chung là một hoạt động thường gặp của con người trong các lĩnh vực đời sống, kỹ thuật và kiến trúc. Từ các vật liệu ban đầu, thiết kết cho phép con người tạo ra một sản phẩm theo một qui tắc, cách thức nào đó nhằm đạt được yêu cầu đặt ra. Do vậy, thiết kế là một quá trình luôn luôn gắn liền với đời sống con người và đó là một hoạt động mang tính sáng tạo. Xây dựng một công cụ tự động hóa việc thiết kế - thay thế con người trong công việc sáng tạo - có một ý nghĩa hết sức thiết thực và là một nhiệm vụ khó khăn. Bài toán thiết kế mạch tổ hợp cũng là một trong số đó, là bài toán cơ bản và thường gặp trong kỹ thuật số. Đã có nhiều phương pháp giải quyết bài toán này. Chẳng hạn, phương pháp đại số nhằm tối thiểu hóa hàm Bool, phương pháp bản đồ Karnaugh ([4]), phương pháp theo thủ tục Quine-Mc. Cluskey ([6, 8]). Các phương pháp này khá cồng kềnh, phức tạp, đòi hỏi nhiều kỹ năng của người thiết kế. Phương pháp thiết kế dùng thuật giải di truyền đã bước đầu đặt nền tảng cho việc thiết kế tự động. Tuy nhiên, các kết quả tính toán còn chậm ([1]). Cách tiếp cận của bài báo là song song hóa thuật giải di truyền, nhằm đề xuất phương pháp hiệu quả để giải bài toán thiết kế mạch tổ hợp, làm tiền đề cho việc thiết kế mạch.

1.1. Bài toán thiết kế mạch tổ hợp

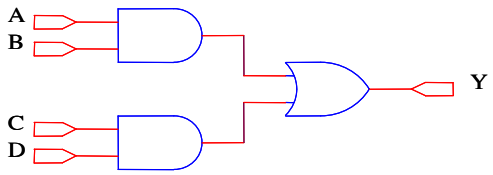
Thiết kế mạch tổ hợp là một hiện thực về mặt phần cứng của các hàm lôgic. Điều đó có nghĩa là thay các phép toán lôgic như AND, OR, XOR, NOT,... bằng các "vật liệu phần cứng" gọi là cổng AND, OR, XOR, NOT,... Không phải lúc nào cũng có sự tương ứng giữa hàm lôgic và mạch lôgic. Một hàm lôgic rất dễ biểu diễn bằng ký hiệu nhưng khi chuyển sang mạch lôgic thì sẽ gặp khó khăn rất lớn do phụ thuộc vào tập các cổng lôgic cho trước.

Các giá trị X_j ở đầu vào chỉ nhận các giá trị 0 và 1. Các giá trị Y_i ở đầu ra không phải bất kỳ mà bị ràng buộc vào các giá trị đầu vào X_j theo các quy tắc F_i cho trước. (Y_i cũng chỉ có giá trị là 0 và 1). Do đó trong việc thiết kế mạch tổ hợp, người thiết kế gặp phải khó khăn khi phải tính ra được hết các ràng buộc này. Số lượng đầu vào và số quy tắc ràng buộc càng nhiều thì khó khăn càng lớn.

Như vậy, bài toán thiết kế mạch tổ hợp được phát biểu như sau.

Cho trước các tín hiệu $X_1, X_2, ..., X_n$ ở đầu vào và tập hợp m hàm lôgic $Y_i = F_i(X_1, X_2, ..., X_n)$. Hãy chỉ ra một mạch tổ hợp dựa trên các cổng lôgic cơ sở AND, OR, NOT, ... cho trước.

Khi đó $X_1, X_2, ..., X_n$ được gọi là các đầu vào của mạch lôgic; $Y_1, Y_2, ..., Y_m$ được gọi là các đầu ra, còn mạch tổ hợp là một hiện thực của sơ đồ kết nối các cổng lôgic nhằm mô tả quan hệ giữa đầu vào và đầu ra. Hình 1 là một mạch tổ hợp đơn giải với 4 tín hiệu đầu vào là A, B, C, D; tín hiệu đầu ra là Y thỏa ràng buộc $Y = AB + CD$ (với phép toán nhân là AND, phép toán cộng là OR); mạch có 2 cổng AND và một cổng OR.



Hình 1. Mạch tổ hợp đơn giản

1.2. Tiếp cận theo hướng thuật giải di truyền

Từ một kiến trúc hai chiều theo kiểu mạch tổ hợp (hình 1 là một ví dụ), chuyển thành kiến trúc 1 chiều theo kiểu nhiễm sắc thể để sử dụng thuật giải di truyền. Độ thích nghi của một nhiễm sắc thể (một sơ đồ mạch tổ hợp) được xác định bằng cách đánh giá tất cả các cấu trúc kiểu hình hai chiều mà nó thỏa mãn các ràng buộc của mạch. Tổng số các cấu trúc kiểu hình hai chiều chính là độ thích nghi của mạch. Giả sử mạch có n đầu vào, khi đó 2^n tổ hợp các đầu vào sẽ được xem xét có thỏa mãn bài toán hay không. Số các trường hợp thỏa mãn chính là độ thích nghi của mạch đang xét. Như vậy, một mạch tổ hợp có n đầu vào và m đầu ra sẽ có độ thích nghi tối đa là $m \times 2^n$.

Hàm lượng giá được xây dựng như sau.

Giai đoạn đầu, chương trình thực hiện so sánh từng bit các ngõ ra của mạch được sinh bởi thuật giải di truyền với các ngõ ra từ bảng chân trị (sự thật). Nếu bảng chân trị của mạch cần thiết kế có n đầu vào và m đầu ra thì số trường hợp so sánh sẽ là $m \times 2^n$. Như vậy có thể có tối đa $m \times 2^n$ trường hợp mà cá thể (tức là sơ đồ mạch) chưa thỏa mãn bảng chân trị. Trong quá trình tìm kiếm, số trường hợp này sẽ giảm dần về 0.

Khi đã tìm được mạch thỏa mãn bảng chân trị, nghĩa là đã có một sơ đồ mạch thỏa mãn các chức năng yêu cầu thì số trường hợp nói trên bằng 0. Từ đây thuật giải di truyền lại khai thác không gian tìm kiếm bằng cách thay thế những cổng lôgic bằng các cổng WIRE (cổng có thể bỏ đi) mà vẫn bảo đảm chức năng lôgic của mạch. Khi đạt được một sự thay thế như vậy, fitness của mạch sẽ được cho thêm một đơn vị. Như vậy, fitness của một mạch theo tiếp cận GA sẽ là

$$\text{Fitness} = f_1 + f_2$$

trong đó, f_1 là số trường hợp thỏa mãn bản chân trị, f_2 là số điểm thưởng thu được khi GA tự tìm kiếm. Với cách biểu diễn mạch dạng kiểu hình hai chiều (là một ma trận kích thước J dòng và K cột) với số cổng WIRE là W, G là số cổng lôgic thực sự có trong mạch,

ta có

$$W + G = J \times K$$

Từ đây suy ra Fitness của một mạch sẽ là

$$\text{Fitness} = f_1 + f_2 = f_1 + W = f_1 + J \times K - G$$

Như vậy mạch có số cổng càng nhỏ thì giá trị Fitness của nó càng lớn.

2. THUẬT GIẢI DI TRUYỀN SONG SONG

2.1. Thuật giải di truyền song song

Có ba mô hình cơ bản khi song song thuật giải di truyền, đó là mô hình song song quần thể toàn cục, mô hình tế bào và mô hình đảo ([5]). Trên SC người ta thường sử dụng mô hình đảo, đây là mô hình mô phỏng theo sự tiến hóa của tự nhiên. Trong đó mỗi nút của SC thực thi giải thuật di truyền ([7]) một cách độc lập với kích thước quần thể con là $N/P, P$ là số bộ xử lý có trong SC, N là số kích thước toàn cục - kích thước của quần thể (tức là số cá thể có trong quần thể). Các quần thể con trao đổi các cá thể thích nghi nhất của mình với các quần thể con láng giềng sau mỗi thế hệ (hay một số thế hệ). Đây thường được gọi là quá trình di trú (migration). Thuật giải được mô tả một cách hình thức như sau với $P_s(t)$ là các quần thể con có kích thước N/P.

$t = 0$
initialize $P_s(t)$
evaluate $P_s(t)$
while (dieu kien chua thoa) **do**
  $t = t + 1$
  select $P_s(t)$ **from** $P_s(t-1)$
  crossover $(P_s(t))$
  mutation $(P_s(t))$
  evaluate $(P_s(t))$
  send (emigrant,...)
  receive (immigrant,...)
endwhile

trong đó, **send()** là phép toán gửi cá thể tốt nhất - gọi là cá thể di cư (emigrant) đến các quần thể láng giềng, **receive()** là phép toán nhận cá thể tốt nhất - gọi là cá thể nhập cư (immigrant) từ các quần thể láng giềng và thay thế các cá thể yếu nhất của quần thể hiện tại.

2.2. Thuật giải song song dùng PVM

Trên hệ thống PVM, tiến trình hay tác vụ chủ (master) khởi tạo tất cả các tiến trình con (slave) khác. Có tất cả P tiến trình con, mỗi tiến trình được ánh xạ vào một bộ xử lý của SC ([2]). Các tiến trình thực hiện thuật giải di truyền tuần tự một cách riêng biệt. Mỗi tiến trình sẽ gửi (nhận) các cá thể được lựa chọn (gọi là tác vụ emigrant) đến (từ) các tiến trình lân cận (gọi là tác vụ neighbours). Thuật giải ở trên có thể được hiện thực bằng PVM như sau:

if (my pid == master) **then**
  for $i = 1$ **to** P **do**
    pvm_spawn (slave S_i)

```
    endfor
endif
for all slave S_i(1 ≤ i ≤ P) do
    while (dieu kien chua thoa) do
        t = t + 1
        select P_s(t) from P_s(t − 1)
        crossover (P_s(t))
        mutation (P_s(t))
        evaluate (P_s(t))
        pvm_send (neighbours, emigrant,...)
        pvm_recv (neighbours, immigrant,...)
    endwhile
endforall
```

3. KHẢO SÁT ĐỊNH TÍNH CỦA THUẬT GIẢI

3.1. Thời gian thi hành của thuật giải tuần tự

Ký hiệu:

T_s là thời gian thực thi của chương trình tuần tự áp dụng thuật giải di truyền.

t_e là thời gian đánh giá một cá thể.

G là số thế hệ trong quá trình tiến hóa.

T_g là thời gian thực thi một hoạt động căn bản của thuật giải di truyền cho một cá thể. Thời gian này bao gồm thời gian tính toán độ thích nghi (đánh giá) của một cá thể, thời gian thực hiện phép lai, đột biến rồi sao chép cá thể đó vào quần thể.

Trên thực tế thì thời gian khởi tạo của một quần thể là rất nhỏ nên từ thuật giải trong hình 1 có thể suy ra

$$T_s = Nt_e + GNT_g = N(t_e + GT_g) \tag{1}$$

Vì $t_e < T_g$ nên $t_e << GT_g$, vậy có thể xem $t_e + GT_g \approx GT_g$, do đó (1) có thể viết lại

$$T_s = NGT_g \tag{2}$$

Công thức (2) cho thấy thời gian thực thi trong chương trình tuần tự có áp dụng thuật giải di truyền tỉ lệ với kích thước quần thể N và số thế hệ G. Nói cách khác, thời gian thực thi của thuật giải tuần tự có độ phức tạp là $O(GN)$. Hệ số T_g trong (2) phụ thuộc vào cấu trúc của mỗi loại máy tính và đặc trưng riêng của từng bài toán cụ thể. Hệ số này có thể xác định dựa vào thực nghiệm bằng cách chạy chương trình với thuật giải di truyền trên cùng một máy tính với các N, G khác nhau, đo kết quả thời gian T_s sau đó từ (2) suy ra giá trị T_g.

3.2. Thời gian thi hành song song

Ký hiệu:

T_p là thời gian thực thi áp dụng thuật giải di truyền song song như đã mô tả ở trên.

T_{comm} là thời gian truyền dữ liệu giữa các bộ xử lý.

T_{comp} là thời gian do việc tính toán.

s_m là kích thước di trú, tức là số cá thể trong quần thể con tham gia quá trình di trú.

Giả sử s_m là như nhau đối với mọi quần thể con.

Theo [3], thời gian thực thi chương trình song song bỏ qua thời gian chờ sẽ là

$$T_p = T_{comm} + T_{comp} \tag{3}$$

Thời gian T_{comm} bao gồm các thành phần:

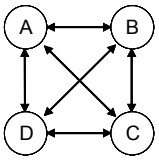
T_{bro} là thời gian tác vụ chủ truyền (broadcast) những tham số của bài toán cho các tác vụ con.

T_{mig} là thời gian thực hiện quá trình di trú giữa các tác vụ con.

T_{SM} là thời gian tác vụ con trả kết quả về tác vụ chủ.

Ta gọi $\theta(P)$ là hàm đặc trưng của một sơ đồ kết nối (topology) (số các kết nối của P tiến trình theo một mô hình mạng nào đó).

Thực chất $\theta(P)$ là số các phép toán gửi từ tiến trình này đến tiến trình kia. Nếu biểu diễn mỗi tiến trình như là một nút trong một đồ thị thì $\theta(P)$ là số cung vectơ của đồ thị đó.



Hình 2. Đồ thị đầy đủ có 4 nút

Chẳng hạn, với sơ đồ kết nối là một đồ thị đầy đủ có 4 nút như trên hình 2. Các kết nối là AB, BC, CD, DA, BA, CB, DC, DA, AC, CA, BD, DB. Nên $\theta(P) = \theta(4) = 12$.

Tổng quát, nếu sơ đồ kết nối là một đồ thị đầy đủ có P nút thì $\theta(P) = P(P-1)$.

Gọi T_f là thời gian lan truyền của một cá thể trên mạng. Để tăng tốc độ hội tụ, ở mỗi thế hệ ta phải cho thực hiện quá trình di trú. Số cá thể tham gia di trú ở mỗi thế hệ trên mạng là $\theta(P)s_m$. Do đó thời gian thực hiện di trú là

$$T_{mig} = G\theta(P)s_m T_f \tag{4}$$

Vì tác vụ con đầu tiên và tác vụ chủ cùng thuộc một bộ xử lý nên thời gian các tác vụ con phát cá thể về cho tác vụ chủ là

$$T_{SM} = (P-1)T_f \tag{5}$$

Gọi t_{MS} là thời gian tác vụ chủ gửi các thông số của bài toán về một tác vụ con. Theo [9] thời gian mà tác vụ chủ truyền các thông số này về P bộ xử lý có trong SC sẽ là

$$T_{bro} = (P-1)t_{MS} \tag{6}$$

Cộng (4), (5), (6) ta có được thời gian truyền dữ liệu giữa các bộ xử lý

$$T_{comm} = G\theta(P)s_m T_f + (P-1)T_f + (P-1)t_{MS} \tag{7}$$

Hay

$$T_{comm} = G\theta(P)s_m T_f + (P-1)(T_f + t_{MS}) \tag{8}$$

Thời gian t_{MS} có thể đo bằng thực nghiệm và phụ thuộc vào từng bài toán.

Đối với T_{comp}, vì mỗi quần thể con có kích thước N/P thực hiện giải thuật di truyền một cách độc lập nên thời gian tính toán theo (2) sẽ là

$$T_{comp} = \frac{NGT_g}{P} \tag{9}$$

Thay (8), (9) vào (3), ta có được biểu thức tổng quát tính thời gian chạy chương trình song song trên SC ứng với một sơ đồ kết nối nào đó được đặc trưng bởi hàm $\theta(P)$

$$T_p = \frac{NGT_g}{P} + G\theta(P)s_mT_f + (P-1)(T_f + t_{MS}) \tag{10}$$

Gọi α là tỉ số giữa thời gian thực hiện tính toán theo thuật giải di truyền cho một cá thể với thời gian lan truyền một cá thể qua mạng; $\alpha = T_g/T_f$.

Còn β là tỉ số giữa thời gian truyền các thông số của bài toán với thời gian lan truyền một cá thể qua mạng; $\beta = t_{MS}/T_f$.

3.3. Độ tăng tốc của hệ thống

Gọi S_p là độ tăng tốc của hệ thống gồm P bộ xử lý có hàm đặc trưng là $\theta(P)$. Theo định nghĩa là tỷ số giữa thời gian thực hiện thuật giải tuần tự và thuật giải song song ([3]), nên

$$S_P = \frac{NGT_g}{\frac{NGT_g}{P} + G\theta(P)s_mT_f + (P-1)(T_f + t_{MS})} \tag{11}$$

Hay

$$S_P = \frac{\alpha}{\frac{\alpha}{P} + \frac{\theta(P)s_m}{N} + \frac{(P-1)(1+\beta)}{NG}} \tag{12}$$

Trong (12), thường N và G rất lớn (hàng nghìn) do đó trong số hạng thứ ba của mẫu số rất nhỏ. Điều đó có nghĩa là chỉ có α ảnh hưởng trực tiếp đến độ tăng tốc, còn ảnh hưởng của β khá nhỏ. Như vậy độ tăng tốc phụ thuộc nhiều vào tỷ số thời gian tính toán theo giải thuật di truyền cho một cá thể với thời gian lan truyền của cá thể qua mạng.

Vấn đề là cần phải xác định số bộ xử lý cần thiết sao cho chi phí thời gian thực hiện thuật giải song song là cực tiểu, ta có

$$\frac{\partial T_p}{\partial P} = -\frac{NGT_g}{P^2} + Gs_mT_f\frac{\partial\theta(P)}{\partial P} + T_f + t_{MS} \tag{13}$$

Giá trị P làm cho T_p cực tiểu là nghiệm của phương trình

$$-\frac{NGT_g}{P^2} + Gs_mT_f\frac{\partial\theta(P)}{\partial P} + T_f + t_{MS} = 0$$

Hay

$$\frac{-\alpha NG}{P^2} + Gs_m\frac{\partial\theta(P)}{\partial P} + 1 + \beta = 0 \tag{14}$$

Gọi P_{opt} là nghiệm của phương trình (14), tùy theo sơ đồ kết nối của hệ thống, chúng ta sẽ có được số bộ xử lý tối ưu cần thiết.

3.4. Một số sơ đồ kết nối

a. Sơ đồ kết nối vòng (ring topology)

Hàm đặc trưng cho sơ đồ kết nối vành khung là $\theta(P) = P$, khi đó phương trình (14) được viết lại

$$-\frac{\alpha NG}{P^2} + Gs_m + 1 + \beta = 0 \tag{15}$$

Suy ra

$$P_{opt} = \sqrt{\frac{\alpha NG}{Gs_m + 1 + \beta}} \tag{16}$$

Nếu lấy $s_m = 1$, G rất lớn so với β thì $P_{opt} \approx \sqrt{\alpha N}$.

b. Sơ đồ kết nối vòng lưới (torus topology)

Hàm đặc trưng cho sơ đồ kết nối vòng lưới là $\theta(P) = 4P$, khi đó (14) được viết lại

$$-\frac{\alpha NG}{P} + 4Gs_m + 1 + \beta = 0 \tag{17}$$

Từ đây suy ra số bộ xử lý tối ưu là

$$P_{opt} = \sqrt{\frac{\alpha NG}{4Gs_m + 1 + \beta}} \tag{18}$$

Nếu chọn $s_m = 1$, G rất lớn so với β, từ (18) ta có thể suy ra $P_{opt} \approx \frac{\sqrt{\alpha N}}{2}$. So với sơ đồ kết nối vòng, số bộ xử lý đòi hỏi chỉ còn một nửa.

c. Sơ đồ kết nối siêu cầu phương (hypercube topology)

Hàm đặc trưng cho sơ đồ kết nối siêu cầu phương là $\theta(P) = P\log_2 P$, khi đó phương trình (14) được viết lại như sau

$$-\frac{\alpha NG}{P^2} + Gs_m\Big(\log_2 P + \frac{1}{\ln 2}\Big) + 1 + \beta = 0 \tag{19}$$

Giải phương trình trên bằng phương pháp xấp xỉ, hoặc các phần mềm tính toán khác có thể tìm được nghiệm P_{opt} tối ưu.

4. KẾT QUẢ THỰC NGHIỆM

Kết quả tính toán thực nghiệm được thực hiện trên hệ thống SC gồm 128 bộ xử lý loại Pentium 4, 1.7 GHz, bộ chuyển mạch Cisco 5500 tốc độ 100 Mbps. Hệ điều hành sử dụng là Linux phiên bản Red Hat 7.1, phần mềm quản trị và điều phối hệ thống song song là PVM 3.4.4.

Mỗi tác vụ của hệ thống PVM ánh xạ vào một bộ xử lý. Kích thước quần thể ban đầu là 10000. Số thế hệ cực đại là 3000. Kích thước di trú là 1. Các số liệu về hàm lượng giá và thời gian thực thi được lấy trung bình sau 5 lần chạy chương trình.

Bảng 1. Thời gian thực thi bài toán cộng

Số bộ xử lý		**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**
RING	**Thời gian**	1h30'8"	46'17''	23'37''	11'34"	5'50''	3'4''	1'35''	56''
	Giá trị Fitness	63.8	63.5	64.1	65.2	65.4	65.3	65.8	64.9
TORUS	**Thời gian**	1h30'8"	46'15''	24'34''	12'32''	6'21''	3'26''	2'17''	3'17''
	Giá trị Fitness	63.8	64.2	64.3	65.3	64.8	65.7	65.7	64.6
CUBE	**Thời gian**	1h30'8"	46'18''	24'	12'12''	6'45''	4'34''	3'57''	12'5''
	Giá trị Fitness	63.8	64.6	64.5	64.8	65.3	65.6	65.8	65.2

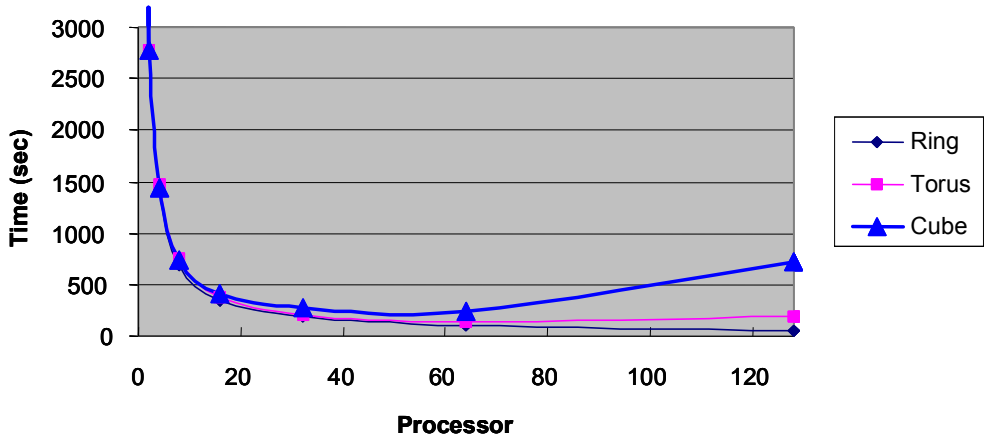
Tiếp theo, ta đi khảo sát 3 bài toán cơ bản trong thiết kế mạch tổ hợp: Bài toán cộng hai số nhị phân 2 bit (Add2), bài toán nhân hai số nhị phân 2 bit (Mult2) và bài toán kiểm tra chẵn - lẻ 6 bit (Par6) dùng hệ thống nêu trên với các bộ xử lý nối với nhau theo các sơ đồ kết nối vành khung, vòng lưới và siêu cầu phương.

4.1. Xác định các tham số T_g, T_f và t_{MS}

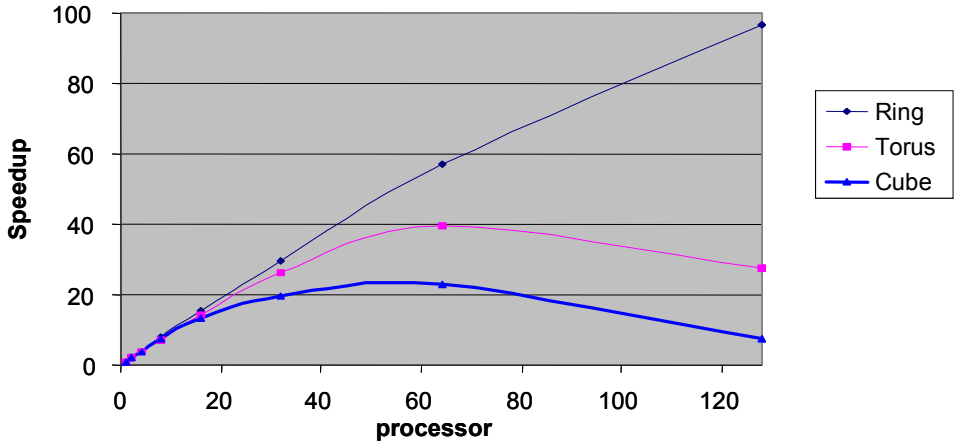
Để xác định giá trị T_g là thời gian tính toán độ thích nghi (đánh giá) của một cá thể, thời gian thực hiện phép lai, đột biến rồi sao chép cá thể đó vào quần thể. Chúng ta sử dụng thuật giải tuần tự để ước lượng với số cá thể N trong một quần thể và số thế hệ G trong quá trình tiến hóa, giá trị này được dùng để tính toán khi sử dụng thuật giải song song. Để có được các tham số T_f và t_{MS}, phương pháp ping-pong được sử dụng để đo thời gian ([9]).

4.2. Bài toán cộng

Bảng 1 cho biết thời gian thực thi chương trình đo được khi giải bài toán cộng với các sơ đồ kết nối khác nhau là vành khung, vòng lưới và siêu cầu phương. Đồ thị biểu diễn thời gian thực thi theo số bộ xử lý và độ tăng tốc cho ở hình 3, 4.



Hình 3. Đồ thị thời gian thực thi bài toán cộng



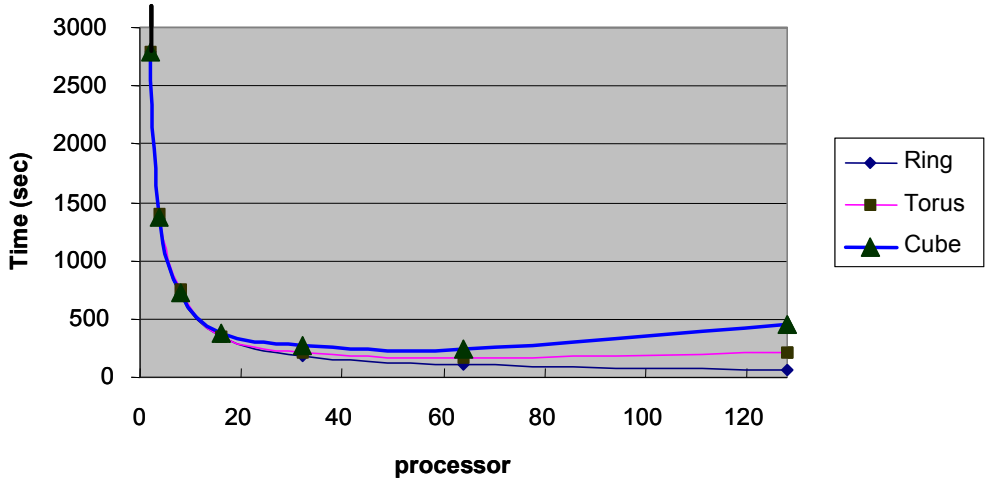
Hình 4. Độ tăng tốc bài toán cộng

4.3. Bài toán nhân

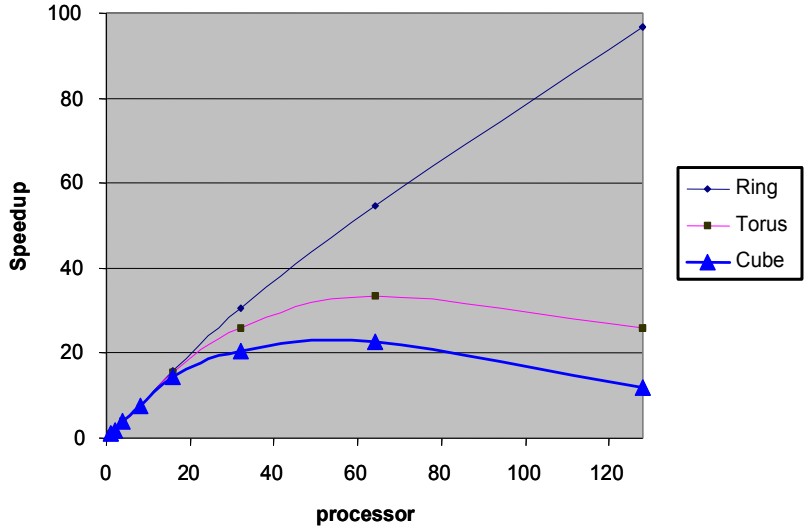
Bảng 2 cho biết thời gian thực thi chương trình đo được khi giải bài toán nhân với các sơ đồ kết nối khác nhau là vòng, vòng lưới và siêu cầu phương. Các hình 5, 6 là đồ thị biểu diễn thời gian thực thi, độ tăng tốc theo số bộ xử lý ứng với các loại sơ đồ kết nối này.

Bảng 2. Thời gian thực thi bài toán nhân

Số bộ xử lý		1	2	4	8	16	32	64	128
RING	Thời gian	1h30'9"	46'28"	22'40"	11'56"	5'40"	2'56"	1'39"	56"
	Giá trị Fitness	79.2	79.5	80.4	80.6	81.5	81.7	81.3	81.7
TORUS	Thời gian	1h30'9"	46'28"	23'15"	12'40"	5'53"	3'28"	2'41"	3'30"
	Giá trị Fitness	79.2	80.1	81.8	80.9	81.8	81.3	81.7	81.7
CUBE	Thời gian	1h30'9"	46'28"	22'52"	12'5"	6'15"	4'26"	3'59"	7'35"
	Giá trị Fitness	79.2	80.6	81.8	81.4	81.6	81.5	81.7	81.6



Hình 5. Đồ thị thời gian thực thi bài toán nhân



Hình 6. Độ tăng tốc bài toán nhân

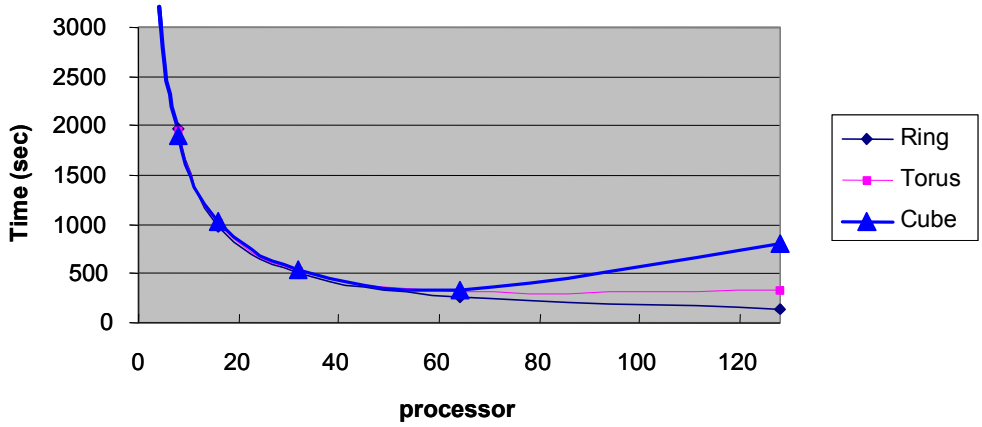
4.4. Bài toán kiểm tra chẵn - lẻ

Bảng 3 là kết quả đo được khi thực thi bài toán kiểm tra chẵn - lẻ 6 bit với các sơ đồ

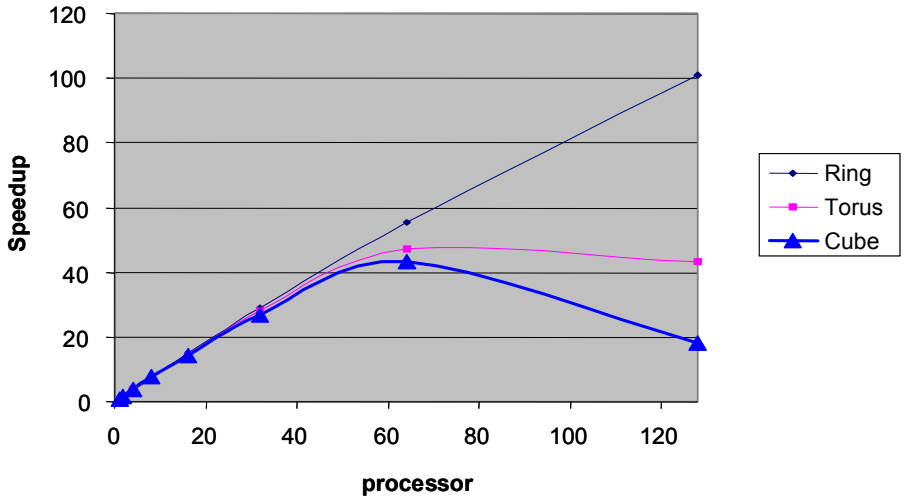
kết nối khác nhau là vòng, vòng lưới và siêu cầu phương. Hình 7, 8 là đồ thị biểu diễn thời gian thực thi và độ tăng tốc theo số bộ xử lý của thuật giải song song.

Bảng 3. Thời gian thực thi bài toán chẵn-lẻ

Số bộ xử lý		**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**
RING	Thời gian	4h4'20''	2h6'36''	1h3'58''	32'46''	16'25''	8'27''	4'25''	2'25''
	Giá trị Fitness	94.3	95	95	95	95	95	95	95
TORUS	Thời gian	4h4'20''	2h6'34''	1h4'23''	32'30''	16'57''	8'39''	5'10''	5'38''
	Giá trị Fitness	94.3	95	95	95	95	95	95	95
CUBE	Thời gian	4h4'20''	2h6'37''	1h4'5''	31'47''	17'8''	9'8''	5'39''	13'19''
	Giá trị Fitness	94.3	95	95	95	95	95	95	95

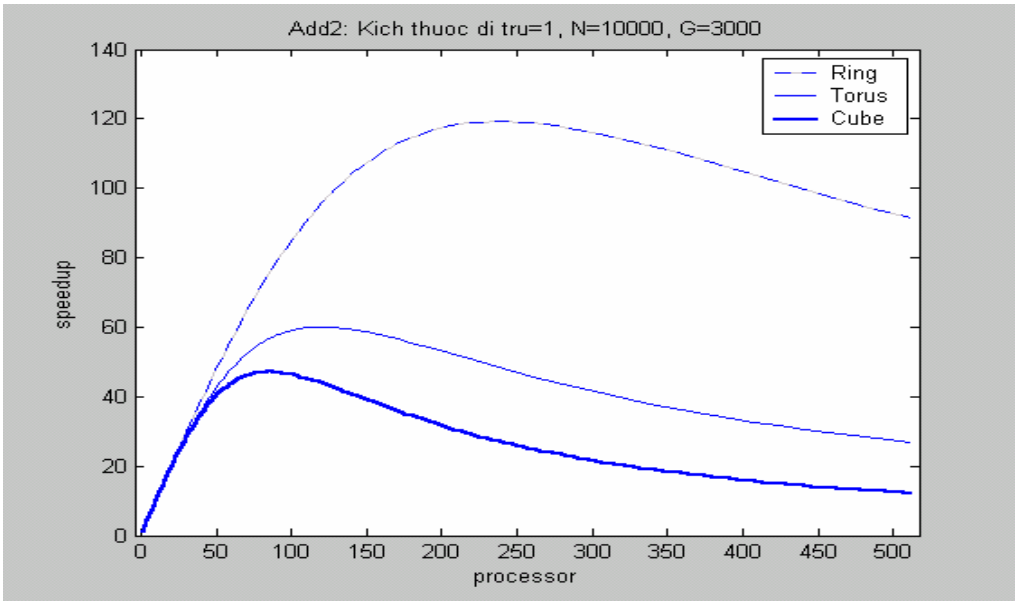


Hình 7. Đồ thị thời gian thực thi bài toán chẵn - lẻ



Hình 8. Độ tăng tốc bài toán chẵn - lẽ

5. KẾT LUẬN



Hình 9. Đồ thị độ tăng tốc lý thuyết

Bảng 4. Hiệu suất (Efficiency) bài toán cộng

Số bộ xử lý	2	4	8	16	32	64	128
RING	0.975	0.955	0.973	0.965	0.906	0.889	0.754
TURUS	0.975	0.917	0.898	0.886	0.82	0.616	0.214
CUBE	0.975	0.940	0.923	0.834	0.616	0.343	0.058

Bảng 5. Hiệu suất bài toán nhân

Số bộ xử lý	2	4	8	16	32	64	128
RING	0.97	0.995	0.945	0.994	0.960	0.852	0.754
TURUS	0.97	0.970	0.890	0.957	0.812	0.525	0.201
CUBE	0.97	0.985	0.932	0.901	0.635	0.353	0.092

Bảng 6. Hiệu suất bài toán chẵn - lẽ

Số bộ xử lý	2	4	8	16	32	64	128
RING	0.965	0.955	0.932	0.935	0.903	0.864	0.789
TURUS	0.965	0.947	0.940	0.901	0.882	0.738	0.338
CUBE	0.965	0.955	0.961	0.891	0.835	0.675	0.143

Hình 9 so sánh kết quả thực nghiệm với kết quả lý thuyết trên cơ sở biểu thức (12) biểu diễn giá trị độ tăng tốc của thuật giải song song. Qua hình vẽ, độ tăng tốc trong

trường hợp vành khung khá khớp với đường cong thực nghiệm trong đoạn [1, 128]. Còn với vòng lưới và siêu cầu phương, kết quả thực nghiệm thấp hơn so với lý thuyết và các điểm cực đại đạt được cũng sớm hơn. Bên cạnh đó, chúng ta có thể tính toán để tìm ra hiệu suất của thuật giải song song của ba bài toán với ba sơ đồ kết nối khác nhau (bảng 4, 5, 6) để từ đó chọn lựa sơ đồ kết nối phù hợp.

TÀI LIỆU THAM KHẢO

[1] C. Carlos, A. D. Christiansen, and A. A. Hernandez, Use of Evolutionary Techniques to Automate the Design of Combinational Circuits, *International Journal of Smart Engineering System Design* **2** (4) (2000) 299 – 314.

[2] A. Geist, A. Beguelin, et al., *PVM: Parallel Virtual Machine - A User' Guide and Tutorial for Networked Parallel Computing,* MIT Press, Cambridge, Massachusetts, 1994.

[3] Ian Foster, *Designing and Building Parallel Programs,* Addision - Wesley, 1995.

[4] M. Karnaugh, A Map Method for Synthesis of Combinational Logic Circuits, Transactions of the AIEE, *Communication and Electronics* **72** (1) (1953) 593 – 599.

[5] G. Lin, Xin Yao, et al., Parallel Genetic Algorithm on PVM, http://citeseer.nj.nec.com/33753.html.

[6] E. J. McCluskey, Minimization of Boolean Functions, *Bell System Technical Journal* **35** (5) (1956) 1417 – 1444.

[7] Z. Michalewics, *Genetic Algorithms + Data Structure = Evolution Programs,* Springer, 1996.

[8] W. V. Quine, A way to Simply Truth Functions, American, *Mathematic Monthly* **62** (9) (1955) 627–631.

[9] B. Wilkinson, A. Michael, *Parallel Programming,* Prentice Hall, 1999.

Nhận bài ngày 02 - 7 -2003

Nhận lại sau sửa ngày 12 - 8 - 2003