*Tạp chí Tin học và Điều khiển học, T.25, S.4 (2009), 301–313*

# VỀ NGỮ NGHĨA ĐẠI SỐ CỦA CÁC CHƯƠNG TRÌNH TỔ HỢP

TRẦN VĂN DŨNG

*Khoa CNTT, Trường Đại học Giao thông Vận tải Hà Nội; zungtv@yahoo.com*

**Abstract.** In this paper, based on CSP (*Communicating sequential processes*) of C.A.R. Hoare, we formalise algebraic semantics of Combinational Programs in the forms of CSP formulas in order to simulate and reason about behaviours of sequential asynchronous circles. Based on the algebraic semantics and axioms of CSP, we give formal proofs of some properties of sequential asynchronous circles without dependent cycles within their components such as the stability, fairness, termination and uniqueness.

**Tóm tắt.** Trong báo cáo này chúng tôi sử dụng cách tiếp cận đại số CSP (*Communicating sequential processes*) của C.A.R. Hoare đưa ra định nghĩa ngữ nghĩa đại số của chương trình tổ hợp dùng để mô phỏng và suy luận về hoạt động của các mạch tuần tự không đồng bộ. Từ ngữ nghĩa đại số và bằng các luật CSP, bài báo đưa ra chứng minh hình thức cho một số tính chất của lớp mạch tuần tự không có chu trình phụ thuộc giữa các thành phần của nó, như: tính ổn định, tính công bằng, tính dừng và tất định.

## 1. MỞ ĐẦU

Ngôn ngữ mô tả phần cứng VERILOG [5] có ngữ nghĩa mô phỏng dựa trên sự kiện [1]. Ngữ nghĩa sự kiện này có thể mô tả hành vi không đồng bộ chi tiết của mạch tuần tự, nhưng nó không hỗ trợ kiểm chứng hình thức. Đã có một số tác giả tìm cách đưa ra ngữ nghĩa thao tác và ngữ nghĩa ký hiệu cho Verilog trong [9] và [10] để làm cơ sở hình thức cho việc lý giải và kiểm chứng các chương trình, nhưng khá phức tạp khi ứng dụng. Để sử dụng các kỹ thuật kiểm chứng phần mềm chuẩn, trong [7] chúng tôi đã đưa ra ngữ nghĩa quan hệ cho các mạch tuần tự không đồng bộ. Sau đó sử dụng đại số quan hệ trong [2, 3] ta có thể suy luận và chứng minh một số tính chất của chúng. Tuy nhiên nếu dùng các công cụ khá phổ biến như CSP để mô tả chương trình tổ hợp thì sẽ mở ra cơ hội lớn hơn nhờ việc sử dụng cơ sở và các công cụ toàn diện đã có của CSP. Chính vì vậy bài báo này tìm cách sử dụng mô hình đại số CSP của C.A.R. Hoare [4] để mô hình các mạch tuần tự không đồng bộ và chứng minh một số tính chất của chúng bằng các luật đại số.

Bài báo này được trình bày như sau: phần tiếp theo nêu định nghĩa và ngữ nghĩa quan hệ của chương trình tổ hợp [6, 7, 8] dùng đề mô tả các mạch tuần tự. Phần ba xây dựng mô hình đại số CSP của các chương trình tổ hợp. Hai phần cuối dùng các luật đại số chứng minh một số tính chất của chúng.

Trong các phần sau chúng ta sẽ sử dụng một số khái niệm và định nghĩa về đại số quá

trình CSP của C.A.R. Hoare trong [4] và lý thuyết thống nhất chương trình trong [3] như khái niệm quá trình và các phép toán trên quá trình: bảng chữ $\alpha$, phép chọn $|$, phép song song $\|$, phép chọn không tất định $\sqcap$, phép đan xen $|||$, phép chọn không tất định trong số các sự kiện có thể $\Box$, phép ghép $;$, phép che dấu $\backslash$, phép gán nhãn quá trình, sẽ được dùng làm mô hình để mô tả mạch tuần tự không đồng bộ trong các phần sau.

## 2. CHƯƠNG TRÌNH TỔ HỢP VÀ NGỮ NGHĨA QUAN HỆ

Sau đây ta nhắc lại các định nghĩa về chương trình tuần tự (xem [3]) được cấu tạo từ các lệnh gán bội và chương trình đơn vị *SKIP* bởi các phép ghép tuần tự, phép chọn điều kiện và không tất định. Các biến tổng thể của chương trình với các chỉ số sẽ được dùng để theo dõi quá trình thực hiện của nó ([7, 8]). Ta giả thiết mọi chương trình tuần tự xét đến đều kết thúc thành công.

**Định nghĩa 1.** (Chương trình tuần tự và chỉ số)

1. Chương trình tuần tự hữu hạn được sinh từ văn phạm sau:

   $$S ::= \Pi \mid \overline{v} := \overline{E} \mid S; S \mid \ S \lhd b \rhd S \mid S \sqcap S$$

   trong đó

   - $\Pi$ ký hiệu cho chương trình *SKIP*, tức là chương trình không làm gì cả, nhưng kết thúc thành công.
   - $\overline{v}$ là véctơ các biến Bool
   - $\overline{E}$ là véctơ các biểu thức Bool có cùng kích thước như $\overline{v}$ và
   - $\sqcap$ là phép toán chọn không tất định.

2. Mỗi chương trình thành phần $P$ sẽ gán cho một chỉ số $index(P)$ là một số tự nhiên. Phép gán chỉ số là một đơn ánh.

**Định nghĩa 2.** (Biến tín hiệu và kiểm soát sự kiện)

1. Ký hiệu $var(P)$ là tập các biến được gán trong chương trình $P$ và $acc(P)$ là tập các biến mà được truy cập đến trong các biểu thức của $P$. Ta có thể coi: $var(P) \subseteq acc(P)$. Chẳng hạn với $\overline{x} = (x_1, x_2, \ldots, x_n)$, thì $var(\overline{x} := \overline{E}) = \{x_1, x_2, \ldots, x_n\}$ và $acc(x := \overline{E})$ là tập các biến xuất hiện trong $\overline{E}$.

2. Biến tín hiệu của biến Bool $x$ được ký hiêu là $\sim x$ và định nghĩa như sau: khi $x$ thay đổi giá trị của nó so với lần gán trước đó, ta có $\sim x = true$. Nếu $x$ là biến tổng thể, thì $\sim x$ là véctơ logic đa chiều, mỗi thành phần của nó $\sim x[i]$ được dùng để thông báo về sự thay đổi của $x$ cho chương trình con được đánh chỉ số $i$.

3. Kiểm soát sự kiện $g(S)$ của chương trình $S$ được định nghĩa như sau

- Giả sử
  - $S$ là phép gán $\overline{x} := \overline{E}$ có chỉ số $i$ và $acc(S) = \{x_1, \ldots, x_k, p_1, \ldots, p_l\}$, trong đó $p_1, \ldots, p_l$ là các đầu vào mà chỉ xuất hiện trong $\overline{E}$, và
  - $\uparrow p$ là biến Bool thể hiện sự tăng của giá trị của đầu vào $p$ và $\downarrow p$ là biến Bool thể hiện sự giảm của $p$.
- Định nghĩa $g(S) =_{df} \sim x_1[i] \vee \sim x_2[i] \vee \ldots \vee \sim x_k[i] \vee t(p_1)[i] \vee \ldots \vee t(p_l)[i]$, trong đó $t(z)$ là $\uparrow z$ hoặc $\downarrow z$ hoặc $\sim z$ hoặc *false* và $i$ là chỉ số của $S$. Thành phần thứ $i$ của biến tín hiệu được dùng để giữ thông tin về sự thay đổi của biến tương ứng trong chương trình $S$. Vì vậy mệnh đề kiểm soát sự kiện $g(S)$ của chương trình $S$ chỉ ra rằng có hay không sự thay đổi của biến nào trong $S$.
- Định nghĩa $g(P\, op\, Q) =_{df} g(P) \vee g(Q)$ đối với $op \in \{;, \lhd b \rhd, \sqcap\}$.

Chương trình tổ hợp dùng để mô phỏng các mạch tuần tự không đồng bộ, ở đây không có đồng hồ chung để đồng bộ các sự kiện. Để xét một số tính chất của mạch tuần tự, trong [7] đưa ra phép toán tương tác không đồng bộ. Rõ ràng, nó có tính chất giao hoán, kết hợp.

**Định nghĩa 3.** (Phép toán tương tác không đồng bộ) (xem [7], [8]).

1. *Phép toán tương tác không đồng bộ* của hai chương trình $P$ và $Q$ với hai chỉ số rời nhau được định nghĩa như sau

   $$P \sharp Q =_{df} (g(P) \to P) \sqcap (g(Q) \to Q)$$

2. Ta định nghĩa $g(g(P) \to P) =_{df} g(P)$ và $index(g(P) \to P) =_{df} index(P)$. Vậy $g(P \sharp Q) = g(P) \vee g(Q)$ và $index(P \sharp Q) = indexP \cup indexQ$.

**Định nghĩa 4.** (Chương trình tổ hợp)

1. Chương trình tổ hợp có dạng sau $Comb\ S$, trong đó $S = P_1 \sharp P_2 \sharp \ldots \sharp P_m$, và $P_1, P_2, \ldots, P_m$, được gọi là các luồng của $S$, bản thân chúng là các chương trình tổ hợp hay tuần tự với các tập chỉ số rời nhau.

2. Ta định nghĩa $g(Comb\ S) =_{df} g(S)$ và $index(Comb\ S) =_{df} index(S)$.

Nếu sự thay đổi của một biến $x$ nào đó làm $g(P_i)$ thay đổi từ *false* sang *true*, thì ta nói rằng biến $x$ kích hoạt kiểm soát sự kiện $g(P_i)$, khi đó luồng $P_i$ được kích hoạt và trở nên sẵn sàng hoạt động.

Trong [1] hành vi của chương trình tổ hợp được mô tả bởi tập các bước mô phỏng, mà mỗi bước bao gồm một dãy thao tác cơ sở như sau:

1. Khi bắt đầu một bước mô phỏng, giá trị của mỗi đầu vào được cung cấp. Giá trị mới của đầu vào có thể kích hoạt một số kiểm soát sự kiện. Các luồng tương ứng trở nên sẵn sàng thực hiện.

2. Môi trường lựa chọn tùy ý một trong các luồng sẵn sàng để thực hiện.

3. Việc thực hiện một luồng sẵn sàng đã chọn $P_i$, mà ta gọi là thao tác cơ sở bao gồm các bước nhỏ sau:

   (a) Thực hiện chương trình $P_i$.

   (b) Xoá kiểm soát sự kiện của $P$ để chỉ ra rằng nó đã được sử dụng để kích hoạt luồng đã chọn.

   (c) Thông báo sự thay đổi của các biến gán trong $var(P_i)$ do việc thực hiện chương trình $P_i$.

4. Nếu không còn luồng nào sẵn sàng nữa, thì bước mô phỏng này dừng và chờ đầu vào thực hiện bước mô phỏng tiếp theo. Nếu luôn có ít nhất một luồng sẵn sàng sau khi thực hiện mỗi thao tác cơ sở, thì bước mô phỏng này không kết thúc.

**Định nghĩa 5.** (Ngữ nghĩa sự kiện, xem [7, 8]). Giả sử $Comb\ S$ là chương trình tổ hợp với $m$ thành phần luồng khác nhau và $n$ biến tổng thể $var(S) = \{v_1, \ldots, v_n\}$. Giả sử một luồng $P$ nào đó có chỉ số là $i$. Khi đó ngữ nghĩa sự kiện của nó với chỉ số $i$ được định nghĩa và ký hiệu như sau:

$$(g(P) \to P_{(e,i)}) =_{df} (g(P))^{\top}; (P \| event(P,i))$$

trong đó

1. Giả thiết $b^{\top} =_{df} \Pi \lhd\ b\ \rhd \top$, trong đó $\top$ là chương trình không thể thực hiện được. Ở đây $g(P)$ là giả thiết để bắt đầu chương trình $P$.

2. $P \| Q$ thể hiện phép toán song song rời nhau của $P$ và $Q$ (xem [3], [4]).

3. Chương trình $event(P,i)$ được sử dụng để xoá kiểm soát sự kiện một khi luồng $P$ được thực hiện và thông báo về sự thay đổi của các biến được tạo ra bởi việc thực hiện $P$:

$$event(P,i) =_{df} \exists v'_1, \ldots, v'_n \bullet result(P)\ \wedge \forall k : 1 \leq k \leq n \bullet$$
$$\sim v'_k := (\sim v_k \wedge clear(i)) \vee (bool(v_k \neq v'_k))^m$$

   trong đó

   - Dấu tròn đậm chỉ ra rằng biến ràng buộc thỏa mãn khẳng định sau;
   - $result(P)$ là mệnh đề mô tả chương trình $P$, mà liên kết bộ giá trị của các biến $v_1, \ldots, v_n$ trước khi thực hiện với bộ giá trị mới của chúng $v'_1, \ldots, v'_n$ sau khi thực hiện chương trình;
   - $m$ là số các chỉ số khác nhau, tức là số luồng khác nhau;
   - Mệnh đề $bool(b)$ có giá trị *true*, nếu $b$ là đúng, ngược lại nó nhận giá trị sai *false*;
   - Véctơ logic $m$ chiều $(bool(b))^m$ với mọi thành phần có giá trị $bool(b)$

- $clear(i)$ là véc tơ logic $m$ chiều sao cho $clear(i)[i] = false$ , ngược lại $clear(i)[j] = true$, nếu $j \neq i$.

Chương trình $event(P, i)$ so sánh giá trị cũ và mới của mỗi biến trong $P$; nếu mọi giá trị này như nhau thì nó sẽ đặt mọi thành phần của vectơ biến tín hiệu với chỉ số $i$ là *false*, tức là, xoá chúng; nếu các giá trị này khác nhau thì nó đặt tất cả $m$ thành phần của vec tơ biến tín hiệu là *true*, tức là, thông báo về thay đổi của biến.

4. Ta sử dụng ký hiệu $P_{(e,i)} =_{df} (P \| event(P, i))$

**Định nghĩa 6.** (Bước mô phỏng) Giả sử $Comb\ S$ là chương trình tổ hợp, trong đó $S = P_1 \sharp \ldots \sharp P_n$ với các tập chỉ số rời nhau tương ứng là $1, 2, \ldots, n$. Ta định nghĩa bước mô phỏng là vòng lặp các sự kiện của các luồng sẵn sàng thực hiện:

$Comb\ S =_{df} g(S) * S_e$, trong đó

$g(S) = g(P_1) \vee \ldots \vee g(P_n)$ và $S_e = g(P_1) \to P_{1(e,1)} \sqcap \ldots \sqcap g(P_n) \to P_{n(e,n)}$.

## 3. CHƯƠNG TRÌNH TỔ HỢP VÀ NGỮ NGHĨA ĐẠI SỐ

Bây giờ chúng ta dựa trên mô hình CSP của C.A.R. Hoare để xây dựng ngữ nghĩa đại số của chương trình tổ hợp đã nêu trong phần trước và hơn nữa chúng ta muốn mô tả không chỉ trong một chu kỳ mà cả một quá trình hoạt động liên tục trao đổi tương tác với môi trường.

Giả sử $Comb\ P$ là chương trình tổ hợp được xét trong một chu kỳ như ở phần trước, đơn giản có một luồng thành phần $P$ với chỉ số là $i$ và chương trình thao tác $P_{e,i}$ được ký hiệu là $pr$. Ta sẽ mở rộng số luồng thành phần thông qua phép toán tương tác không đồng bộ.

Giả sử $P$ có tập các biến gán, tập các biến truy cập, tập các đầu vào tương tác với nó và bảng chữ được sử dụng trong $P$ là

$$var,\ acc,\ inp,\ \alpha \quad \text{với} \quad var \subseteq acc \subseteq \alpha \quad \text{và} \quad inp \subseteq acc$$

Ta ký hiệu $\#var$ là số biến trong $var$. Để đơn giản chúng ta quan tâm đến các biến tín hiệu của các biến là điểm nhấn đặc biệt của chương trình tổ hợp, còn bản thân các biến được dùng để chia sẻ nói chung được mô tả bởi kỹ thuật chung đã biết trong CSP.

Chúng ta xây dựng chương trình tổ hợp $Circ(P)$ tương tác với môi trường để nhập giá trị đầu vào ở đầu mỗi chu kỳ mà trong đó $Comb\ P$ là một bước mô phỏng trong một chu kỳ như đã xét trong phần trước:

$$Circ(P) = Comb\ P \to Circ(P)$$

Bây giờ ta xây dựng từng bước các thành phần của $Circ(P)$ hay viết ngắn gọn là $Circ$:

1. Chương trình tổ hợp tương tác $Circ$ là sự kết hợp song song giữa hai quá trình thao tác $Exec$ và điều khiển kích hoạt $Cont$ và hai quá trình đó phối hợp lựa chọn các thành phần sẵn sàng để thực hiện

$$Circ = Cont \| Exec$$

2. Quá trình thao tác $Exec$ mô phỏng việc nhập dữ liệu và thực hiện mạch tuần tự cho đến khi dừng. Dùng quá trình $Atom$ để mô phỏng các thao tác cơ sở của $pr$ và quá trình $Input$ thực hiện việc nhập đầu vào.

   - Các sự kiện của $Input$ và $Atom$ được thực hiện đan xen nhau

     $$Exec = Input|||Atom$$

   - $Exec$ cũng như các quá trình thành phần có cùng bảng chữ gồm mọi biến truy cập, biến tín hiệu và bản thân ký hiệu chương trình $pr$:

     $$\alpha(Exec) = \alpha(Input) = \alpha(Atom) = acc(pr) \ \cup \sim acc(pr) \ \cup \ \{pr\}$$

3. Quá trình $Cont$ điều khiển việc kích hoạt luồng thành phần thông qua việc nhận các biến tín hiệu được gửi bởi $Exec$. Vậy $\alpha(Cont) = acc(pr) \cup \sim acc(pr)$. Kênh gửi thông tin từ $Exec$ đến $Cont$ được gọi là $upd$, khi đó ta sử dụng ký hiệu $upd : Cont$.

   Bây giờ chúng ta đưa ra định nghĩa của các quá trình.

4. Sau khi nhập giá trị đầu vào từ môi trường, $Input$ gửi thông báo giá trị biến tín hiệu đầu vào cho các quá trình để điều khiển kích hoạt:

   $$Input \ = \ (upd.right! \sim inp) \rightarrow Input$$

5. Khi được lựa chọn ở một bước cơ sở nào đó, mỗi quá trình thành phần sẵn sàng sẽ xoá mọi biến kích hoạt nó và thực hiện thân chương trình tuần tự của mình, rồi gửi thông báo giá trị của các biến tín hiệu cho mọi chương trình liên quan khác. Ta định nghĩa

   $$\begin{aligned} Atom \ = \ & aquire \rightarrow pr \rightarrow (\text{if change then upd.right!} \sim var \\ & else \ \Pi) \rightarrow release \rightarrow Atom \end{aligned}$$

   Sự kiện $pr$ ký hiệu việc thực hiện bản thân chương trình trong luồng. Biến $change$ cho biết có ít nhất một biến gán nào đó thay đổi giá trị hay không

   $$change = bool((\sim var)^{\#var} \neq (false)^{\#var})$$

   Như vậy sau khi thực hiện $pr$ kèm theo việc so sánh các giá trị mới với giá trị cũ của các biến, nếu có ít nhất một biến nào đó thay đổi giá trị, thì $change = true$ và sẽ thực hiện việc thông báo sự thay đổi đến các luồng thành phần có sử dụng các biến đó.

6. Gọi trạng thái của chương trình là bộ giá trị các biến của nó, tức là ánh xạ từ tập $var$ vào tập các giá trị $val$ của các biến. Khi đó tập các trạng thái $state$ được xác định như sau $state$: $var \rightarrow val$. Trạng thái của một chương trình tổ hợp được gọi là ổn định nếu mọi biến tín hiệu của nó đều nhận giá trị sai: $false$, được đặc trưng bởi biến Bool sau:

$$stable = bool((\sim var)^{\#var} = (false)^{\#var})$$

Điều kiện ban đầu để chương trình tổ hợp tương tác với môi trường bên ngoài là nó đang ở trạng thái ổn định và được đặc trưng bởi phương trình:

$$Input = stable^{\top}; Input$$

7. Quá trình điều khiển kích hoạt cho phép các thành phần tiếp nhận các biến tín hiệu kể cả đầu vào để kích hoạt làm cho chúng sẵn sàng thực hiện

$$Cont \quad = \quad ((\text{upd.left?} \sim inp \to \Pi)|||(\text{upd.left?} \sim acc \to \Pi)) \to Trig$$

8. Quá trình thành phần đã sẵn sàng có thể tiếp tục tiếp nhận thêm các biến tín hiệu hoặc đồng bộ thực hiện cùng chương trình thành phần *Atom* có cùng chỉ số

$$\begin{array}{rcl} Trig & = & \mu X \bullet (\text{upd.left?} \sim acc \to X) \to Trig \mid \\ & & acquire \to \ \text{if change then release} \to Trig \\ & & \qquad\qquad \text{else release} \to Cont \end{array}$$

tức là mọi sự thay đổi mà nó tạo ra có thể kích hoạt chính nó.

9. Giả sử chương trình tổ hợp $P$ có chỉ số $i$, để phân biệt với các luồng khác ta thêm chỉ số $i$ cho các đặc trưng của nó: chương trình luồng thành phần là $pr.i$ và có tập các biến gán, tập các biến truy cập, tập các đầu vào và bảng chữ tương ứng là: $var[i]$, $acc[i]$, $inp[i]$, $\alpha[i]$. Bây giờ chúng ta sử dụng phép gán nhãn quá trình (xem [4]) cho các quá trình thành phần của $Circ(P)$ như sau $i : Trig; i : (upd : Cont); i : Input, i : Atom; i : Exec$ để chỉ rõ các quá trình tương ứng của $P$ có chỉ số $i$. Và với mỗi sự kiện *action* trong các quá trình trên được gán thêm chỉ số có dạng $action.i$. Ta có thể định nghĩa

$$Circ(P) = i : (upd : Cont)\|(i : Input|||i : Atom).$$

**Định nghĩa 7.** Giả sử cho hai chương trình tổ hợp $Circ(P)$ và $Circ(Q)$ với hai chỉ số rời nhau là $i$ và $j$. Khi đó ta định nghĩa

$$Circ(P)\sharp Circ(Q) = \{i, j\} : (upd : Cont)\|(\{i, j\} : Input)|||\{i, j\} : Atom) \setminus \{\alpha(conn)\},$$

trong đó *conn* là kênh trao đổi giữa hai chương trình. Rõ ràng phép toán $\sharp$ có tính chất giao hoán, kết hợp.

**Ví dụ**. Xét mạch tuần tự có hai cổng NOT-OR là $p$ và $q$ được kết nối như sau: đầu ra của cổng $p$ cùng đầu vào $r$ là hai đầu vào của $q$ và đầu ra của cổng $q$ quay lui cùng với đầu vào $s$ là hai đầu vào của $p$. Các biến đều là biến Bool và có thể được đặc trưng bởi hai phép gán

$$p := \neg s \wedge \neg q, \tag{1}$$

$$q := \neg r \wedge \neg p. \tag{2}$$

Ta mô tả từng mạch thành phần, rồi kết hợp chúng với nhau qua phép toán $\sharp$.

1. Luồng $P$ có 2 đầu vào là $s$ và $q$ và một biến trạng thái là $p$, có chỉ số là 1.

$$\begin{aligned}
(1 : Input) &= (upd.1.right! \sim s[1] \to \Pi \| \\
&\quad upd.1.right! \sim q[1] \to \Pi)) \to (1 : Input) \\
(1 : Atom) &= \mu X \bullet (aquire.1 \to (p := \neg s \wedge \neg q) \to (\text{if } bool(p \neq p') \ then \\
&\quad upd.1.right! \sim p[2] \ else \ \Pi) \to release.1 \to X) \\
1 : (upd : Cont) &= (\text{upd.1.left?} \sim s[1] \to \Pi) \| (\ \text{upd.1.left?} \sim q[1] \to \Pi) \\
&\quad \to (1 : Trig) \\
(1 : Trig) = \mu X &\bullet ((\text{upd.1.left?} \sim s[1] \mid \text{upd.1.left?} \sim q[1]) \to X) \\
&\quad \mid (acquire.1 \to release.1) \to 1 : (upd : Cont)
\end{aligned}$$

$$Circ(P) = 1 : (upd : Cont) \| (1 : Input| \ as \ Atom)$$

2. Luồng $Q$ có 2 đầu vào là $r$ và $p$ và một biến trạng thái là $q$, có chỉ số là 2.

$$\begin{aligned}
(2 : Input) &= (upd.2.right! \sim r[2] \to \Pi \| \\
&\quad upd.2.right! \sim q[2] \to \Pi)) \to (2 : Input) \\
(2 : Atom) &= \mu X \bullet (aquire.2 \to (q := \neg r \wedge \neg p) \to (\text{if } bool(q \neq q') \ then \\
&\quad upd.2.right! \sim q[1] \ else \ \Pi) \to release.2 \to X) \\
2 : (upd : Cont) &= (\text{upd.2.left?} \sim r[2] \to \Pi) \| \| (\text{upd.2.left?} \sim p[2] \to \Pi) \\
&\quad \to (2 : Trig) \\
(2 : Trig) = \mu X &\bullet ((\text{upd.2.left?} \sim r[2] \mid \text{upd.2.left?} \sim p[2]) \to X) \\
&\quad \mid (acquire.2 \to release.2) \to 2 : (upd : Cont)
\end{aligned}$$

$$Circ(Q) = 2 : (upd : Cont) \| (2 : Input| \ \| 2 : Atom)$$

3. Ta mô tả hoạt động của mạch tổ hợp trên bằng quá trình tương tác không đồng bộ giữa $P$ và $Q$ che dấu các biến phụ và trung gian:

$$(Circ(P) \sharp Circ(Q)) = \{1, 2\} : (upd : Cont) \| (((\{1, 2\} : Input) \| \| \{1, 2\} : Atom) \setminus \mathcal{I}$$

trong đó

$$\mathcal{I} = \{upd.2.left? \sim p[2], upd.2.right! \sim p[2], upd.1.left? \sim q[1], upd.1.right! \sim q[1]\}.$$

## 4. MỘT SỐ TÍNH CHẤT CỦA CHƯƠNG TRÌNH TỔ HỢP

Giả sử $Circ(S) = Circ(P_1) \sharp Circ(P_2) \sharp \ldots \sharp Circ(P_n)$ là chương trình tổ hợp với $n$ quá trình thành phần $\{P_i\}$ có chỉ số tương ứng là $i$.

Ta sử dụng một số ký hiệu sau:

1. $Comb\ S(\bar{x}^0, \bar{p}^0|\bar{p})$ là chương trình thực hiện trong một chu kỳ của $Circ(S)$ đang ở trạng thái ổn định ban đầu $\bar{x}^0$ với đầu vào cũ $\bar{p}$ và đầu vào mới $\bar{p}^0$ được nhập vào kích hoạt chương trình. Đôi khi ta dùng trạng thái $(\bar{x}^0, \bar{p}^0)$, nếu không muốn nói đến giá trị đầu vào cũ. Ta ký hiệu như sau

   $$Circ(S)(\bar{x}^0, \bar{p}^0|\bar{p}) = Comb\ S(\bar{x}^0, \bar{p}^0|\bar{p}) \to Circ(S).$$

2. Cho $D \subseteq N = \{1, 2, \ldots, n\}$ là một tập con các chỉ số luồng trong $n$ luồng thành phần, khi đó ký hiệu

   $$N : (upd : Cont(D)) = (D : Trig)\|((N \setminus D) : (upd : Cont)).$$

   Ban đầu trước khi nhập đầu vào mọi thành phần đều chưa sẵn sàng, khi đó $D = \emptyset$ và chương trình tổ hợp cũng sẽ dừng khi $D = \emptyset$.

3. Ký hiệu $D_0 = D(\bar{p}^0|\bar{p})$ là tập các thành phần $i : Cont$ được kích hoạt trở nên sẵn sàng do thay đổi đầu vào từ $\bar{p}$ thành $\bar{p}^0$.

4. Gọi tập các hành động bên trong cần che dấu là

   $$\mathcal{C} = \{acquire, release, upd, \sim var, \sim inp, change\}.$$

Sau khi thay đổi đầu vào, nó tác động kích hoạt mọi thành phần bị ảnh hưởng trực tiếp.

**Bổ đề 1.**

$$Comb\ S(\bar{x}^o, \bar{p}^0|\bar{p}) \setminus \mathcal{C} = (N : (upd : Cont(D_0))\|N : Atom)(\bar{x}^o, \bar{p}^o),$$

*trong đó* $D_0 = D(\bar{p}^0|\bar{p})$.

*Chứng minh.* Thực vậy,

$$\begin{aligned}
& Circ(S)(\bar{x}^0, \bar{p}^0|\bar{p}) \\
= \ & (N : (upd : Cont)\|(N : Input(\bar{p}^0|\bar{p})\|\|N : Atom(\bar{x}^0, \bar{p}^0))) \\
= \ & ((\|_{i\in N}\text{upd.i.left?} \sim inp \to \Pi)\|\|_{i\in N}\ (\text{upd.i.left?} \sim acc \to \Pi) \to Trig)\| \\
& ((\|_{i\in N}(upd.i.right! \sim inp) \to Input)\|\|(N : Atom))(\bar{x}^0, \bar{p}^0) \\
= \ & ((D_0 : Trig)\|(N \setminus D_0) : (upd : Cont))\|((N : Input)(p^0)\|\|(N : Atom)(x^0, \bar{p}^0) \\
= \ & (N : (upd : Cont(D_0))\|N : Atom)(\bar{x}^o, \bar{p}^o) \to Circ(S)
\end{aligned}$$

Suy ra điều cần phải chứng minh. ■

Sau khi lựa chọn một quá trình đã sẵn sàng $i : Trig$, chúng ta cần đồng bộ hai sự kiện $acquire.i \to \ldots \to release.i$ trong $i : Atom$ và trong $i : Trig$. Và mọi sự kiện của thao tác cơ sở đều nằm trong hai sự kiện đồng bộ đó, do đóđảm bảo rằng thao tác cơ sở hiện tại sẽ được hoàn thành trước khi vào thao tác cơ sở tiếp theo.

**Bổ đề 2.** (Bổ đề triển khai)

$$\begin{aligned}
& (N : (upd : Cont(D_0))\|N : Atom)(\bar{x}^o, \bar{p}^0) \setminus \mathcal{C} \\
& = \Box_{i\in D_0} pr.i \to (N : (upd : Cont(D_{i1}))\|N : Atom)(\bar{x}^{i1}, \bar{p}^0),
\end{aligned}$$

*trong đó $\bar{x}^{i1}$ là trạng thái đã được cập nhật các biến gán trong $pr.i$ và $D_{i1}$ là tập các thành phần đang sẵn sàng sau khi thực hiện bước cơ sở đầu tiên là $pr.i$.*

*Chứng minh.* Thực vậy,

$$\begin{aligned}
& (N : (upd : Cont(D_0))\|N : Atom)(\bar{x}^o, \bar{p}^0) \setminus \mathcal{C} \\
= \; & ((D_0 : Trig\|N \setminus D_0 : (upd : Cont))\|N : Atom)(\bar{x}^o, \bar{p}^0) \setminus \mathcal{C} \\
= \; & (D_0 : (acquire \to release) \to \\
& (\text{if change } then\ Trig\ else\ Cont)\ \|N \setminus D_0 : (upd : Cont))\| \\
& (N : aquire \to pr \to (\text{if change } then\ upd.right! \sim var \\
& else\ \Pi) \to release) \to Atom \setminus \mathcal{C} \\
= \; & \Box_{i \in D_0} acquire.i \to pr.i \to (\text{if change.i } then\ upd.i.right! \sim var \\
& else\ \Pi) \to release.i \to (N : (upd : Cont(D_{i1}))\|N : Atom)(\bar{x}^{i1}, \bar{p}^0) \setminus \mathcal{C} \\
= \; & \Box_{i \in D_0} pr.i \to (N : (upd : Cont(D_{i1}))\|N : Atom)(\bar{x}^{i1}, \bar{p}^0)
\end{aligned}$$

Bổ đề triển khai cho thấy, khi chương trình dừng không có luồng nào được kích hoạt, mọi biến không có thay đổi, do đó chương trình đạt trạng thái ổn định. ■

**Định lý 1.** *Trong một chu kỳ nếu chương trình tổ hợp dừng, thì khi đó nó đạt trạng thái ổn định:*

$$stable^{\top}; (Comb\ S \to Circ(S)) = (stable^{\top}; Comb\ S;\ stable_{\perp}) \to Circ(S),$$

*ở đây sử dụng ký hiệu $b_{\perp} =_{df} \Pi \lhd\ b\ \rhd \perp$, trong đó $\perp = RUN$ là chương trình thực hiện bất cứ điều gì. Một chương trình sau khi kết thúc có tính chất $b$ nào đó, thường được biểu diễn qua đơn vị phải $b_{\perp}$ của nó.*

*Chứng minh.* Theo bổ đề triển khai, chương trình tổ hợp kết thúc chu kỳ khi không có thành phần nào kích hoạt và khi đó không có sự thay đổi của mọi biến, tức là đã đạt trạng thái ổn định:

$$\begin{aligned}
& stable^{\top}; (Comb\ S(\bar{x}^o, \bar{p}^0|\bar{p}) \to Circ(S)) \setminus \mathcal{C} \\
= \; & stable^{\top}; (N : (upd : Cont(D_0))\|N : Atom)(\bar{x}^o, \bar{p}^0) \setminus \mathcal{C} \\
= \; & stable^{\top}; (\Box_{i_1 \in D_0} pr.i_1 \to (N : (upd : Cont(D_{i1}))\|N : Atom)(\bar{x}^{i1}, \bar{p}^0)) \setminus \mathcal{C} \\
= \; & stable^{\top}; (\Box_{i_1 \in D_0} pr.i_1 \to \Box_{i_2 \in D_{i1}} pr.i_2 \\
& \to (N : (upd : Cont(D_{i2}))\|N : Atom)(\bar{x}^{i2}, \bar{p}^0)) \setminus \mathcal{C} \\
= \; & stable^{\top}; (\Box_{i_1 \in D_0} pr.i_1 \to \Box_{i_2 \in D_{i1}} pr.i_2 \to \ldots \to \Box_{i_k \in D_{ik-1}} pr.i_k \\
& \to (N : (upd : Cont(\emptyset)); stable_{\perp}\|N : Atom)(\bar{x}^{ik}, \bar{p}^0)) \setminus \mathcal{C} \\
= \; & stable^{\top}; (\Box_{i_1 \in D_0} pr.i_1 \to \Box_{i_2 \in D_{i1}} pr.i_2 \to \ldots \to \Box_{i_k \in D_{ik-1}} pr.i_k)(\bar{x}^{ik}, \bar{p}^0); \\
& stable_{\perp} \to (N : (upd : Cont(\emptyset))\|N : Atom) \setminus \mathcal{C},
\end{aligned}$$

trong đó $D_0$ là tập các luồng được kích hoạt khi nhập đầu vào, $pr_{i_h}$ là thao tác cơ sở được lựa chọn thực hiện trong $D_{ih-1}$ và $\bar{x}^{ih}$ là trạng thái đạt được sau đó. Khi đạt đến trạng thái ổn định tiếp theo thì chương trình tổ hợp kết thúc một chu kỳ hoạt động và chờ môi trường thay đổi đầu vào. ■

## 5. CHƯƠNG TRÌNH CÚ PHÁP KHÔNG CHU TRÌNH

Sau đây chúng ta xét lớp các chương trình tổ hợp mà không có chu trình phụ thuộc giữa

các luồng theo nghĩa đầu ra của luồng trước tham gia đầu vào của luồng sau. Ta chứng minh lại một số tính chất theo ngữ nghĩa quan hệ nêu trong [8] bằng việc áp dụng ngữ nghĩa đại số và các luật của CSP: trong mỗi chu kỳ chúng có các tính chất tính công bằng (fairness), dừng và tất định.

**Định nghĩa 8.** Giả sử $Circ(S) = Circ(P_1)\sharp Circ(P_2)\sharp \ldots \sharp Circ(P_n)$ là chương trình tổ hợp với $n$ quá trình thành phần là $\{P_i\}$ có chỉ số tương ứng là $i$. Khi đó

- Nếu $var_i \cap acc_j \neq \emptyset$, thì ta nói chương trình $j$ phụ thuộc cú pháp vào chương trình $i$ và ký hiệu $P_i \prec P_j$.
- Chương trình tổ hợp được gọi là cú pháp không chu trình, nếu $\prec$ là quan hệ thứ tự bộ phận trên tập các thành phần của nó và mọi chương trình thành phần đều lũy đẳng, tức là $pr.i^2 = pr.i$, với $i = 1, 2, \ldots, n$.
- Một chương trình tổ hợp gọi là có tính công bằng nếu mọi thành phần sẵn sàng sẽ được chọn để thực hiện ở một bước nào đó.

Việc thực hiện chương trình tổ hợp cú pháp không chu trình trong mỗi thao tác cơ sở bảo toàn tính ổn định của các biến mà không phụ thuộc vào đầu ra của thao tác đó và làm cho các biến gán mới ổn định.

**Bổ đề 3.** *Giả sử Circ S là chương trình thành phần cú pháp không có chu trình, tức là* $P_i \prec P_j \Rightarrow i < j$ *và* $pr.j^2 = pr.j$, $i, j = 1, 2, \ldots, n$. *Khi đó*

$$(\wedge_{i<j} \neg change.i)^{\top};\ pr.j = (\wedge_{i<j} \neg change.i)^{\top};\ pr.j;\ (\wedge_{i\leq j} \neg change.i)_{\perp}.$$

*Chứng minh.* Rõ ràng từ các tính chất của chương trình tổ hợp thành phần cú pháp. Ta có $j > i \Rightarrow var_j \cap acc_i = \emptyset$. Do đó các biến có mặt trong các biểu thức vế phải của các chương trình $P_i$, với $i < j$, sẽ không chịu tác động của chương trình $pr.j$, nên

$$(\wedge_{i<j} \neg change.i)^{\top}; pr.j = (\wedge_{i<j} \neg change.i)^{\top}; pr.j; (\wedge_{i<j} \neg change.i)_{\perp}.$$

Kết hợp với tính chất lũy đẳng các thành phần: $pr.j^2 = pr.j \Rightarrow pr.j = pr.j; (\neg change.j)_{\perp}$ suy ra điều phải chứng minh.

Nếu sau một vòng lặp có ít nhất một biến trở nên ổn định, thì mọi biến sẽ ổn định sau một số vòng lặp nhất định. Nếu mọi luồng sẵn sàng cần phải được thực hiện ở bước nào đó, khi đó một biến bất kỳ nhất định sẽ trở nên ổn định trong quá trình thực hiện chương trình tổ hợp cú pháp. Điều đó suy ra tính dừng của nó. ■

**Định lý 2.** *Trong một chu kỳ hoạt động mọi chương trình tổ hợp cú pháp không có chu trình có tính công bằng và sẽ dừng.*

*Chứng minh.* Nếu chỉ có một luồng thành phần, thì theo định nghĩa của chương trình cú pháp, thành phần đó lũy đẳng nên sẽ dừng sau một bước thực hiện. Bây giờ giả sử $n$ là số

thành phần của chương trình. Giả thiết qui nạp rằng chúng ta đã chứng minh định lý cho chương trình với số thành phần ít hơn.

Nếu thành phần thứ nhất không được kích hoạt khi nhập đầu vào, thì nó sẽ không sẵn sàng trong suốt quá trình thực hiện các thao tác cơ sở đó của chu kỳ đó, khi đó theo giả thiết qui nạp chương trình chỉ có nhiều nhất $n-1$ thành phần sẵn sàng, do đó có tính công bằng và sẽ dừng.

Bây giờ ta giả thiết thành phần thứ nhất sẵn sàng sau khi nhập đầu vào:

$$
\begin{aligned}
& stable^{\top}; Comb\ S(\bar{x}^o, \bar{p}^0|\bar{p}) \to Circ(S) \setminus \mathcal{C} && \{\text{Theo Bổ đề 1 }\}\\
= \;& stable^{\top}; (N : (upd : Cont(D_0)) \| N : Atom)(\bar{x}^o, \bar{p}^0) \setminus \mathcal{C} && \{\text{Theo Bổ đề 2 }\}\\
= \;& stable^{\top}; (pr.1 \to (N : (upd : Cont(D_{11})) \| N : Atom)) \setminus \mathcal{C}\\
& \quad \{1 \in D_0, \text{ từ Bổ đề } 3 : 1 \notin D_{11}, \text{ nên theo giả thiết qui nạp sẽ dừng}\}\\
& \Box_{i_1 \in D_0} pr.i_1 \to (N : (upd : Cont(D_{i1})) \| N : Atom)) \setminus \mathcal{C}\\
& \qquad \{\text{Với } i_1 \neq 1 \wedge 1 \in D_{i1}, \text{theo giả thiết qui nạp có tính fairness}\}\\
= \;& stable^{\top}; (pr.1 \to \ldots \to pr.j_k \to (N : (upd : Cont(\emptyset)) \| N : Atom)\\
& \Box_{i_1 \in D_{01}} pr.i_1 \to \ldots \to pr.1 \to (N : (upd : Cont(D_{1k})) \| N : Atom)) \setminus \mathcal{C}\\
& \qquad \{\text{Từ Bổ đề } 3 : 1 \notin D_{1k}, \text{ Theo giả thiết qui nạp sẽ dừng}\}\\
= \;& stable^{\top}; (pr.1 \to \ldots \to pr.j_k \to (N : (upd : Cont(\emptyset)) \| N : Atom)\\
& \Box_{i_1 \in D_{01}} pr.i_1 \to \ldots \to pr.1 \to \ldots \to pr.i_t); stable_{\perp}\\
& \to (N : (upd : Cont(\emptyset)) \| N : Atom) \setminus \mathcal{C}. && \{\text{Theo Định lý 1 }\}
\end{aligned}
$$

Trong một chu kỳ hoạt động chương trình cú pháp không chu trình sẽ tương đương với việc thực hiện tuần tự các chương trình thành phần theo đúng trình tự phụ thuộc của chúng, tức là chúng có tính chất tất định và có thể đưa về dạng chuẩn từ các chương trình thành phần. ■

**Định lý 3.** *Cho chương trình cú pháp không chu trình Circ S có dãy chương trình thành phần $pr.1, pr.2, \ldots, pr.n$ thỏa mãn tính chất $P_i \prec P_j \Rightarrow i \leq j$. Khi đó trong một chu kỳ hoạt động*

$$stable^{\top}; Comb\ S \setminus \mathcal{C} = stable^{\top}; pr.1; pr.2; \ldots; pr.n; stable_{\perp}.$$

*Chứng minh.* Thực vậy

$$
\begin{aligned}
& stable^{\top}; Comb\ S \to Circ(S) \setminus \mathcal{C} && \{\text{Theo Định lý 1 và 2 }\}\\
= \;& stable^{\top}; Comb\ S; stable_{\perp} \to Circ(S) \setminus \mathcal{C} && \{(a \wedge b)_{\perp} = a_{\perp}; b_{\perp}\}\\
= \;& stable^{\top}; Comb\ S; (\neg change.1); stable_{\perp} \to Circ(S) \setminus \mathcal{C}\\
= \;& stable^{\top}; Comb\ S; (\neg change.1); pr.1; stable_{\perp} \to Circ(S) \setminus \mathcal{C}\\
= \;& stable^{\top}; Comb\ S; pr.1; (\neg change.2); stable_{\perp} \to Circ(S) \setminus \mathcal{C}\\
& && \{\text{Theo Định lý 2 }\}\\
= \;& stable^{\top}; Comb\ S; pr.1; (\neg change.2); pr.2; stable_{\perp} \to Circ(S) \setminus \mathcal{C}\\
= \;& stable^{\top}; Comb\ S; pr.1; pr.2; \ldots; pr.n; stable_{\perp} \to Circ(S) \setminus \mathcal{C}\\
& \{(x := e; z := g; x := f) = (z := g; x := f), x \text{ không có trong } g\}\\
= \;& stable^{\top}; pr.1; pr.2; \ldots; pr.n; stable_{\perp} \to Circ(S) \setminus \mathcal{C}
\end{aligned}
$$

■

## 6. KẾT LUẬN

Bài báo đã đề xuất việc sử dụng đại số quá trình CSP của C.A.R. Hoare để đưa ra ngữ nghĩa đại số của các chương trình tổ hợp mô tả các mạch tuần tự không đồng bộ. Như vậy, ta có thể dùng các luật và các bộ công cụ hỗ trợ của CSP để suy luận về các tính chất của các mạch tuần tự. Cụ thể bài báo đã đưa ra chứng minh tính công bằng, tính dừng và tất định của lớp các chương trình cú pháp không chu trình. Chúng tôi sẽ đề xuất cách biểu diễn các chương trình tổ hợp một cách có cấu trúc rõ ràng hơn và đưa ra một số luật suy luận về các tính chất của chúng trong những nghiên cứu tiếp theo.

## TÀI LIỆU THAM KHẢO

[1] M. Gordon, "Event and Cycle Semantics of Hardware Description Languages", University of Cambridge Computer Laboratory, (1998).

[2] C.A.R Hoare, *et. al.*, Laws of programming, *Comm. of the ACM* **30** (8) (1987) 672–686.

[3] C.A.R. Hoare and He Jifeng, *Unifying Theory of Programming*, Prentice - Hall International, 1998.

[4] C.A.R. Hoare *et al*, *Communicating Sequential Processes,* Prentice Hall International, 1985-2004.

[5] Open VERILOG International. VERILOG Hardware Description Language Reference Manual, Version 1.0.

[6] T.V. Dung and He JiFeng. "A theory of combinational programs", UNU/IIST Report No 162.

[7] T.V. Dung and He JiFeng. A theory of combinational programs, *Proceedings, APSEC 2001*, Macao, China (325–328).

[8] T.V. Dung, On the stability semantics of combinational programs, *Proceedings, ICTAC 2005, LNCS 3722*, Hanoi, Vietnam (180–194).

[9] Z. Huibiao, J. Bowen and He JiFeng, Deriving operational semantics from denotational semantics for verilog, *Proceedings, APSEC 2001*, Macao, China (177–184).

[10] J. Dimitrov, Operational semantics for verilog, *Proceedings, APSEC 2001*, Macao, China (161–168).

*Nhận bài ngày 6 - 7 - 2009*
*Nhận lại sau sửa ngày 17 - 1 -2010*