Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/1/29

DLHA によるCPUとDRPの協調動作の組込みシステムのシステム仕様記述

酒井辰典[†1]　柳瀬龍[†1]
酒井誠[†1]　山根智[†1]

CPU と DRP(Dynamically Reconfigurable Processor; 動的再構成可能プロセッサ) の協調動作の動的再構成可能組込みシステムを仕様記述するため，本研究では動的線形ハイブリッドオートマトン (以下 DLHA) を提案する．DRP の特徴としては，Co-タスクの生成・消滅，動作周波数の変化を動的に行うといった点が挙げられ，これらの特徴によりオートマトンの動的な生成・破棄，ハイブリッドオートマトンの性質を用いることで，従来手法よりロケーション数を削減した．

Specification of cooperated systems consisting of CPU and DRP by using DLHA

Tatsunori Sakai,[†1] Ryo Yanase,[†1] Makoto Sakai [†1]
and Satoshi Yamane[†1]

In this paper, we propose specification of cooperated systems consisting of CPU and DRP. DLHA(Dynamic Linear Hybrid Automaton) can describe the dynamic changes in configuration. By using DLHA, we have decreased the number of locations .

1. はじめに

近年，組込みシステムは多様な機能を持つようになり，その専用処理用プロセッサ数の増加が小型化・省電力化の障害となっている．そこで注目されているのが，動的再構成可能プロセッサ (Dynamically Reconfigurable Processor, DRP)[1)–3)] である．

DRP の仕様記述に関しては前例として南ら[7)] のものがあるが，検証時における状態爆発が問題であった．本論文では，仕様記述レベルでのロケーション数を削減することができる DLHA を提案する．ロケーション数を減らすことで，解析時における状態数を減らし状態爆発を抑制することが予想される．

尚，DRP の特徴及びモデル化については前例研究[7)] とほぼ同じであるため，紙面の都合上割愛する．

2. DLHA

ここでは，本稿で提案する仕様記述言語である DLHA とその動作について定義する．表記のルールとして，$=$ を「定義する ($\stackrel{\text{def}}{=}$)」，$==$ を「等しい」，$:=$ を「代入」の意味で用いる．

2.1 構　　文

関数，値の評価と制約条件について定義する．

定義 1 (サイクル関数とサイクル評価)
サイクル関数 $c \in C$ は，時間を定義域として，サイクル数を値域とする関数，$c : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ であり，ある時間 t の値 $c(t)$ をサイクル評価と呼び，μ と表記する．ある時間 t におけるサイクル評価値 $c(t)$ は，時間 0 から t までのサイクル数の積分値，すなわち経過サイクル数を表す． □

定義 2 (離散変数と離散値評価)
離散変数 $d \in D$ のある時間 t の値 $d(t)$ を離散値評価 ξ と呼ぶ． □

定義 3 (制約条件)
クロック関数とサイクル関数の制約条件を以下のように定義する．

$$\phi ::= x_1 \sim e \mid x_1 - x_2 \sim e \mid c \sim e \mid d \sim e \mid \phi_1 \wedge \phi_2 \mid true \mid asap$$

ただし，$x_1, x_2 \in X$, $c \in C$, $d \in D$, $e \in \mathbb{Z}$, $\sim \in \{<, \leq, >, \geq, ==\}$ である．なお，$asap$ がついた遷移関係は時間遷移より優先して行われる．全ての制約条件の集合を $\Phi(X \cup C)$ とする． □

†1 金沢大学
Kanazawa University

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

定義 4 (関数の傾き)
各ロケーションへ割り付ける関数の傾きを以下のように定義する．

$$f_c ::= \dot{c} = n$$

$n \in \mathbb{N}, \dot{c}$ は $c \in C$ の周波数 dc/dt である．この f_c の集合を $F(C)$ とする． □

DLHA は生成・消滅により動作中の構成変化を表すことができるオートマトンである．生成・消滅アクションの出力アクションに対する入力アクションはそれぞれ 1 つずつであり，1 対 1 対応で生成・消滅を行う．

定義 5 (DLHA の構文)
DLHA の構文 H は，$H = \langle L, l_0, X, C, D, I, flow, Act, T, T_{init}, T_{end} \rangle$ で表される．

- ロケーションの有限集合 L.
- 初期ロケーション $l_0 \in L$.
- サイクル関数の有限集合 C.
- 離散変数の有限集合 D.
- ロケーションに不変条件 ϕ を割り当てる関数 $I : L \to \Phi(C)$.
- 各ロケーションにサイクル関数の傾きを与える関数 $Flow$．ただし，$Flow : L \to F(C)$ である．
- アクションの有限集合 $Act = Act_{in} \cup Act_{out} \cup Act_{\tau}$．ここで，$Act_{in}$ は入力アクションの有限集合，Act_{out} は出力アクションの有限集合，Act_{τ} は内部アクションである．特に $Crt\_H_n! \in Act_{out}, Crt\_H_n? \in Act_{in}$ はオートマトン H_n を生成するアクション，$Dst\_H_n! \in Act_{out}, Dst\_H_n? \in Act_{in}$ はオートマトン H_n を消滅させるアクション，$\mathcal{Q}!H_n \in Act_{\tau}$ はオートマトン H_n の生成要求をキューに格納するアクション，$\mathcal{Q}?H_n \in Act_{\tau}$ はオートマトン H_n の生成要求をキューから取り出すアクションとする．その要素は $(l, \phi, a, UPD(X), l') \in T$ で表され，a は $a!, a?, a_{\tau}$ のいずれかである．
- エッジの有限集合 $T \subseteq L \times \Phi(C) \times Act \times 2^{UPD(V)} \times L$.
  その要素は $(l, \phi, a, UPD(X), l') \in T$ で表され，a は $a!, a?, a_{\tau}$ のいずれかである．
  - $l, l' \in L$ は遷移元及び遷移先ロケーション．
  - $\phi \in \Phi(C \cup D)$ はガード条件．
  - $a!, a?, a_{\tau} \in Act$ はアクションである．ただし，遷移とともにアクションを出力するアクションを $a!$，出力アクションを受け取り遷移するアクションを $a?$，内部アクションを a_{τ} と表す．
  - $UPD(V)$ は算術式 upd の集合である．upd は次のように定義される．

    $$upd ::= v := const \mid v := v + const$$

    ただし v は変数の有限集合 $V = X \cup C \cup D$ の要素，$const$ は実数，整数と文字列を表す．この算術式 upd は変数のリセット，カウンタのカウントアップなどを表す．
- T_{init} は，以下のいずれかである．
  - アクションがある時，$T_{init} \subseteq Act_{in} \times 2^{UPD(V)} \times L$ その要素は $(a?, UPD(V), l) \in T_{init}$ で表される．
    - $a? \in Act_{in}$ は生成アクション．
    - $UPD(V) \in 2^{UPD(V)}$ は，算術式の集合．特定の値で変数を初期化する．
    - $l \in L$ は生成されたオートマトンの初期ロケーション．
  - アクションがない時，$T_{init} \subseteq 2^{UPD(V)} \times L$
    - $UPD(V) \in 2^{UPD(V)}$ は，算術式の集合．特定の値で変数を初期化する．
    - $l \in L$ は初期ロケーション．

  生成アクションがないオートマトンの場合は，そのオートマトンは既に生成され，動作しているオートマトンであり，生成アクションがあるオートマトンは，アクションにより生成され動作し始めるオートマトンである．また，$UPD(V)$ に初期化されなかった変数は暗黙に *0* で初期化されるものとする．
- $T_{end} \subseteq L \times \Phi(C) \times Act_{out}$ その要素は $(l, \phi, a!)$ で表される．
  - $l \in L$ は遷移元のロケーション．
  - $\phi \in \Phi(C \cup D)$ はガード条件．
  - $a! \in Act_{out}$ は消滅アクション．

  これにより，指定されたオートマトンが消滅する．

□

定義 6 (DLHA の並列合成)
2 つの *DLHA* H_1, H_2 の並列合成は，$H_1 || H_2 = \langle L_{H_1||H_2}, l_{0H_1||H_2}, C_{H_1||H_2}, D_{H_1||H_2}, I_{H_1||H_2}, Flow_{H_1||H_2}, Act_{H_1||H_2}, T_{H_1||H_2}, T_{initH_1||H_2}, T_{endH_1||H_2} \rangle$ によって定義される．ここでは，H_1 と新たに合成する *DLHA* を H_2 とする．

- $L_{H_1||H_2} = L^{\bar{*}}_{H_1} \cup (L^{*}_{H_1} \times L_{H_2})$．ただし，$L^{*}_{H_1}$ は以下に示すようなロケーション集合である．また，$L^{\bar{*}}_{H_1} = L_{H_1} \setminus L^{*}_{H_1}$ とする．
  - H_2 に $Crt\_H_2?$ アクションと $Dst\_H_2!$ アクションがある場合，$L^{*}_{H_1}$ は，L_{H_1} の中で $Crt\_H_2!$ アクションが付いている遷移の遷移先ロケーションから $Dst\_H_2?$ アク

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

ションが付いている遷移の遷移元ロケーションの間で到達可能なロケーション集合.
  - H_2 に $Crt\_H_2?$ アクションがなく $Dst\_H_2!$ アクションがある場合，$L^*_{H_1}$ は，L_{H_1} の中で l_{0H_1} から $Dst\_H_2?$ アクションが付いている遷移の遷移元ロケーションの間で到達可能なロケーション集合.
  - H_2 に $Crt\_H_2?$ アクションがなく $Dst\_H_2!$ アクションがない場合，または，$Crt\_H_2?$ アクションがあり $Dst\_H_2!$ アクションがない場合，$L^*_{H_1} = L_{H_1}$.
- $l_{0H_1||H_2} = l^*_0$. ただし，l^*_0 は $l_{0H_1} \in L^*_{H_1}$ のとき $l^*_0 = \{l_{0H_1}, l_{0H_2}\}$，$l_{0H_1} \notin L^*_{H_1}$ のとき $l^*_0 = l_{0H_1}$ とする.
- $C_{H_1||H_2} = C_{H_1} \cup C_{H_2}$.
- $D_{H_1||H_2} = D_{H_1} \cup D_{H_2}$.
- $I_{H_1||H_2} = I^*$. ただし，I^* は，$L_{H_1||H_2}$ に対しては $I^* = I_{H_1} \cap I_{H_2}$，$L^{\bar{*}}_{H_1}$ に対しては $I^* = I_{H_1}$ とする.
- $Flow_{H_1||H_2} = Flow^*$. ただし，$Flow^*$ は，$L_{H_1||H_2}$ に対しては $Flow^* = \{flow_{H_1}\} \cup \{flow_{H_2}\}$，$L^{\bar{*}}_{H_1}$ に対しては $Flow^* = flow_{H_1}$ とする.
- $Act_{H_1||H_2} = (Act_{H_1} \setminus Act_{inoutH_1}) \cup (Act_{H_2} \setminus Act_{inoutH_2}) \cup Act_{\tau H_1||H_2}$. ただし，$Act_{inoutH_1}$ は H_1 で H_2 のオートマトンに入力アクション $a?$ 及び出力アクション $a!$ と同期するアクションが存在するアクションの有限集合．Act_{inoutH_2} は H_2 で H_1 のオートマトンに入力アクション $a?$ 及び出力アクション $a!$ と同期するアクションが存在するアクションの有限集合．それらは並列合成後，内部アクション a_τ の有限集合 $Act_{\tau H_1||H_2}$ となる.
- $T_{H_1||H_2} \subseteq L_{H_1||H_2} \times \Phi_{H_1||H_2} \times Act_{H_1||H_2} \times 2^{UPD(V_{H_1||H_2})} \times L_{H_1||H_2}$. ただし，$\Phi_{H_1||H_2} = \Phi_{H_1} \cup \Phi_{H_2}$．$2^{UPD(V_{H_1||H_2})} = 2^{UPD(V_{H_1}) \cup UPD(V_{H_2})}$ であり，その要素 t は2つの遷移 $(l_{H_1}, \phi_{H_1}, a_{H_1}, UPD(V_{H_1}), l'_{H_1})$ と $(l_{H_2}, \phi_{H_2}, a_{H_2}, UPD(V_{H_2}), l'_{H_2})$ に対して，以下の規則によって決まる.
  - $l_{H_1}, l'_{H_1} \in L^{\bar{*}}_{H_1}$ のとき，$t = (l_{H_1}, \phi_{H_1}, a_{H_1}, UPD(V_{H_1}), l'_{H_1})$.
  - $l_{H_1}, l'_{H_1} \in L^*_{H_1}$ のとき，
    - $a_{H_1} \in Act_{\tau H_1}$ または，H_2 に $a_{H_1} \in Act_{inH_1}$ もしくは $a_{H_1} \in Act_{outH_1}$ と同期するアクションがないとき，$t = ((l_{H_1}, l_{H_2}), \phi_{H_1}, a_{H_1}, UPD(V_{H_1}), (l'_{H_1}, l_{H_2}))$.
    - $a_{H_2} \in Act_{\tau H_2}$ または，H_1 に $a_{H_2} \in Act_{inH_2}$ もしくは $a_{H_2} \in Act_{outH_2}$ と同期するアクションがないとき，$t = ((l_{H_1}, l_{H_2}), \phi_{H_2}, a_{H_2}, UPD(V_{H_2}), (l_{H_1}, l'_{H_2}))$.
    - $a_{H_1} \in Act_{inH_1}$ かつそのアクションに同期する $a_{H_2} \in Act_{outH_2}$ または，$a_{H_1} \in Act_{outH_1}$ かつそのアクションに同期する $a_{H_2} \in Act_{inH_2}$ がある時，そのアクションは $a \in Act_{\tau H_1||H_2}$ となり，$t = ((l_{H_1}, l_{H_2}), \phi_{H_1} \wedge \phi_{H_2}, a, UPD(V_{H_1}) \cup UPD(V_{H_2}), (l'_{H_1}, l'_{H_2}))$
- T_{init} は以下のいずれかである.
  - $T_{init} \subseteq L_{H_1||H_2} \times Act_{\tau H_1||H_2} \times 2^{UPD(V)} \times L_{H_1||H_2}$. その要素 t は $t = (l_{H_1||H_2}, a_\tau, UPD(V), l'_{H_1||H_2})$ で表される.
    - $l_{H_1||H_2}, l'_{H_1||H_2} \in L_{H_1||H_2}$ は遷移先及び遷移元ロケーション.
    - $a_\tau \in Act_{\tau H_1||H_2}$ は生成アクション.
    - $UPD(V) \in 2^{UPD(V)}$ は算術式 upd の集合.
  - $T_{init} \subseteq 2^{UPD(V)} \times L_{H_1||H_2}$
    - $l_{H_1||H_2} \in T_{init}$ はロケーション.
    - $UPD(V) \in 2^{UPD(V)}$ は算術式 upd の集合.
- $T_{end} \subseteq L_{H_1||H_2} \times \Phi_{H_1||H_2} \times Act_{\tau H_1||H_2} \times L_{H_1||H_2}$. その要素 t は $t = (l_{H_1||H_2}, \phi_{H_1||H_2}, a_\tau, l'_{H_1||H_2})$ で表される.
  - $l_{H_1||H_2}, l'_{H_1||H_2} \in L_{H_1||H_2}$ は遷移元及び遷移先ロケーション.
  - $\phi_{H_1||H_2} \in \Phi_{H_1||H_2}$ はガード条件.
  - $a_\tau \in Act_{\tau H_1||H_2}$ は消滅アクション.

□

2.2 意　　味

次に，意味について定義する.

定義 7 (DLHA の状態)

$DLHA$ の状態は $q = (l, \mu, \xi)$ である.

- $l \in L$ はロケーション.
- $\mu \in \mathbb{R}_{\geq 0}$ は時間 t におけるサイクル評価値.
- $\xi : D \to \mathbb{Z}$ は離散値評価.

□

定義 8 (DLHA の意味)

$DLHAH = \langle L, l_0, C, D, I, Flow, Act, T, T_{init}, T_{end} \rangle$ の意味は，$\mathcal{M} = \langle Q, \Rightarrow, q_0 \rangle$ として定義される.

- Q は状態 q の集合.
- $\Rightarrow$ は時間遷移 $\Rightarrow_\delta$ と離散遷移 $\Rightarrow_d$ の和集合.
  - 時間遷移

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

任意の状態 $(l,\mu,\xi) \in Q$ と時間経過 $t \in \mathbb{R}_{\geq 0}$ に対し，$l' = l$ かつ $\mu' = \mu + Flow(l) \cdot t \in I(l)$ の時かつその時に限り $(l,\mu,\xi) \Rightarrow_\delta (l,\mu',\xi)$ である．

– 離散遷移

ある遷移関係において条件が満たされている場合，ロケーションに留まる不変条件を満たしていても即座に離散遷移するような動作として，ガード条件 $\phi ::= asap$ が定義されている．これは時間遷移のない瞬時的な離散遷移の動作を行う場合に有用となる．

* アクション a が生成アクション及び消滅アクションでない時，任意の状態 $(l,\mu,\xi) \in Q$ に対し，ロケーション l からのエッジ $(l,\phi,a,UPD(V),l') \in T$ が存在し，μ と ξ が ϕ を満たし，かつ $UPD(V)$ で更新された μ' が $\mu' \in I(l')$ となるとき $(l,\mu,\xi) \Rightarrow_d (l',\mu',\xi')$ である．
* アクション a が生成アクションの時，任意の状態 $(l_1,\mu_1,\xi_1) \in Q$ に対し，ロケーション l_1 からのエッジ $(l_1,\phi,a,UPD(V),(l_1',l_2)) \in T$ が存在し，μ_1 が ϕ を満たし，かつ $UPD(V)$ で更新された μ_1' が $\mu_1' \in I(l_1')$ となるとき $(l_1,\mu_1,\xi_1) \Rightarrow_d ((l_1',l_2),(\mu_1',\mu_2)(\xi_1',\xi_2))$ である．
* アクション a が消滅アクションの時，任意の状態 $((l_1,l_2),(\mu_1,\nu_2)(\xi_1,\xi_2)) \in Q$ に対し，ロケーション (l_1,l_2) からのエッジ $((l_1,l_2),\phi,a,UPD(V),l_1') \in T$ が存在し，μ_1,μ_2,ξ_1,ξ_2 が ϕ を満たし，かつ $UPD(V)$ で更新された μ' が $\mu' \in I(l')$ となるとき $((l_1,l_2),(\mu_1,\mu_2)(\xi_1,\xi_2)) \Rightarrow_d (l_1',\mu_1',\xi_1')$ である．

- $q_0 \in Q$ は初期状態．

□

2.3 システム全体の構成

システムの構成図を図 1 に示す．

- *Ext* は外部を表したオートマトンである．外部は周期的に *CPU-Scheduler* にタスクの起動要求を出す．
- *Task* はタスクのオートマトンであり，外部から起動要求を受けると消滅状態から実行待ち状態に遷移し，*Task Dispatcher* にディスパッチ要求を出す．*Task Dispatcher* から実行タスクとして選択されると実行状態に遷移する．実行中のタスクは Co-タスクに処理を要求することがある．その際は，タスクを待ち状態に遷移させ，*Task Dispatcher* にディスパッチ要求を出し，*Wait Control* に Co-タスクの処理要求を出す．*Wait Control* より Co-タスクの処理終了応答が返されると実行待ち状態に遷移する．実行中のタスクが処理を終了すると *Task Dispatcher* にディスパッチ要求を出し，消滅状態に遷移する．
- *Task Dispatcher* はディスパッチャのオートマトンである．*Task* よりディスパッチ要求を受けると実行待ちもしくは実行状態のタスクの中で，最も優先度の高いタスクを実行状態に遷移させ，それ以外のタスクを実行待ち状態に遷移させる．ディスパッチ要求は *Wait Control* からもある．
- *Wait Control* はどのタスクがどの Co-タスクに処理を要求したかの管理を行うオートマトンである．*Task* より Co-タスクの処理要求を受けると，DRP のキューに Co-タスクの起動要求を出す．*Co-task* より Co-タスク処理終了応答を受けると，*Task* に Co-タスク処理終了応答を返し，*Task Dispatcher* にディスパッチ要求を出す．ここでは，Co-タスクは a と b からなるので，*Wait Control* a と *Wait Control* b からなる．
- *Queue* は CPU からの *Co-task*. 生成要求のバッファである．*Co-task* の生成順を記憶し出力するためキューとなる．
- *Tile-control* は DRP のタイルを管理するオートマトンである．キューの先頭の Co-タスクを実行する分のタイルの空きがあれば *Co-task* に生成要求を出す．また *Co-task* から消滅報告を受け取るとその Co-タスクの分だけタイルを開放する．
- *Frequency-manager* は DRP の周波数を管理するオートマトンである．動作周波数の遅い Co-タスクに合わせて，DRP 全体で実行中の Co-タスクの動作周波数を変更する．
- *Co-task* は Co-タスクのオートマトンで，*Tile-control* からの生成要求を受けて生成される．動作周波数は *Frequency-manager* によって管理される．処理が終了すると *Tile-control*，*Frequency-manager* に消滅報告をする．

2.4 仕 様 記 述

本節では，前節で述べた各オートマトンの仕様記述を行う．なお，Task はタスク A とタスク B の 2 つのタスクからなる．以下では，図 2 の表記で仕様記述する．また今回は CPU 側は静的なモデルを使用するため，前例研究[7] で使用したモデルを使用する．従って本論文では割愛し，本節では DRP 側だけを説明する．

2.4.1 *Tile-control*

図 1 の *Tile-control* に相当する図はオートマトン *Tile-control* である (図 3)．ここで，離散変数 *tile* は，現在利用可能なタイル数を示し，定数 $tile.a_0$，$tile.b_0$ 等は *Co-task* が占有するタイル数を示す．その大まかな動作は以下のとおりである．

- *Tile-control* は，*tile* を設計上最大の値で初期化され動作を開始する．

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

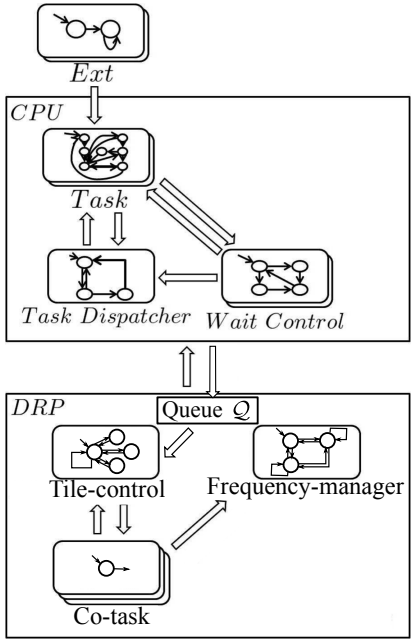

図 1 オートマトンの構成図

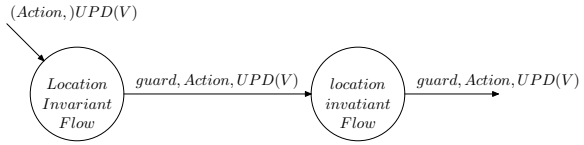

図 2 仕様記述の表記

- キューの先頭の *Co-task* が占有するタイル数より，利用可能タイル数 *tile* のほうが大きい場合，その *Co-task* の生成アクションを出力し利用可能タイル数 *tile* を *Co-task* が利用する分だけ減らす.
- *Co-task* の消滅アクションを受け取った場合，その *Co-task* が占有していたタイル数の分だけ利用可能タイル数 *tile* を増やす.

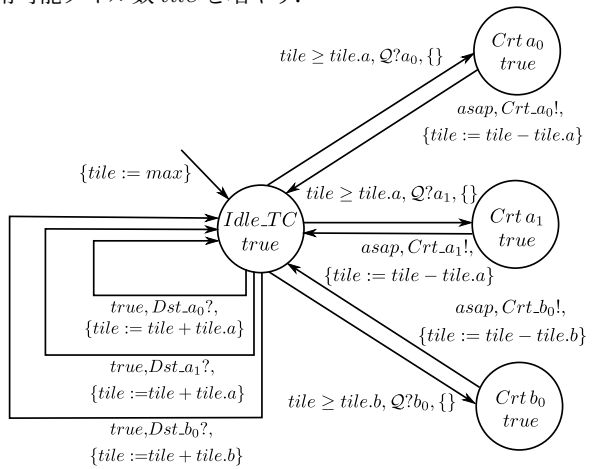

図 3 *Tile-control* のオートマトン

2.4.2 *Frequency-manager*

図 1 の *Frequency-manager* に相当する図はオートマトン *Frequency-manager* である (図 4)．ここで，離散変数 *count.a* は DRP 上で動作している *Co-task* a_0 と *Co-task* a_1 の数，*count.b* は DRP 上で動作している *Co-task* $b0$ の数である．*Co-task* a_0 と *Co-task* a_1 の動作周波数を f_a，*Co-task* b_0 の動作周波数を f_b とする．今回のモデルでは $f_a < f_b$ としている．大まかな動作は以下のとおりである．

- *count.a*，*count.b* を 0 で初期化し，初期ロケーション *Idle_FM* から動作を開始する．
- *Co-task* の生成または消滅アクションと同期して遷移し，遷移先は現在動作中の *Co-task* の中で最も動作周波数の低い *Co-task* にあわせるよう，動作中の各 *Co-task* の数によって決められる．生成アクションと同期して遷移する場合はその生成アクションにより生成される *Co-task* の動作中の数を 1 つ増やし，消滅アクションと同期して遷移する場合はその消滅アクションにより消滅される *Co-task* の数を 1 つ減らす.

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

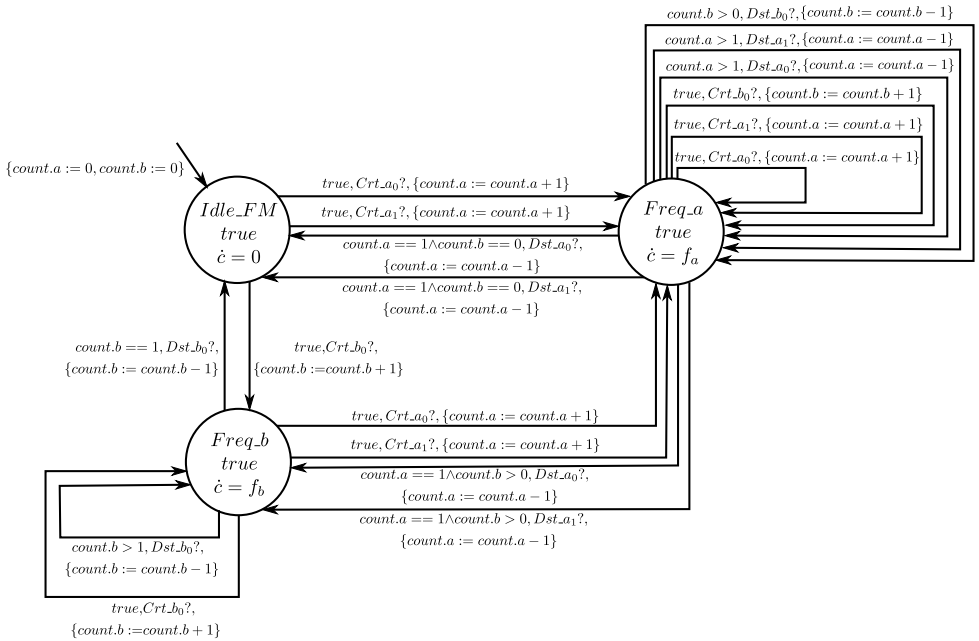

図 4 *Frequency-manager* のオートマトン

2.4.3 *Co*-タスク

図 1 の *Co-task* に相当する図は *Co-task* の内のひとつであるオートマトン *Co-task* a_0 である (図 5)．ここで，関数 x_{a_0} は起動からの経過時間を表すクロック関数，c_{a_0} は実行時間を表すサイクル関数，定数 E_a は *Co-Task* a の処理に必要な時間を示す．c_{a_0} の傾きは，*Frequency-manager* の $\dot{c}$ より取得する．

(1) 生成アクション Crt_a_0 を *Tile-control* から受け取って生成される．

(2) x_{a_0} の傾きは常に 1，c_{a_0} の傾きは *Frequency-manager* の $\dot{c}$ に制御され，タスクの生成・消滅による動作周波数の変化に合わせてこの値も変化する．

(3) c_{a_0} が E_a に達したとき，消滅アクション Crt_a_0 を *Tile-control*，*Frequency-manager* に出力し破棄される．

Co-task a_1，*Co-task* b_0 も同様の動作をする．

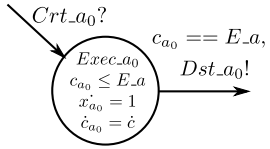

図 5 *Co-task* a_0 のオートマトン

3. 解析手法

本節では，到達可能な状態集合を求めるための解析手法を述べる．まず，キューの内容を含めたシステム全体の状態について形式的に定義し，次に解析のための探索手法について述べる．

3.1 キューと大域状態

本節では，キューの構文とシステム全体の状態 (大域状態) について形式的に定義する．

3.1.1 キュー

ここでは，キューとキューに対する操作について定義する．キューには，CPU 側のタスクからキューへの追加アクションによって，Co-タスクの生成要求が FIFO(First In First Out) 方式で追加される．

定義 9 (キュー)
FIFO キューを $\mathcal{Q}$，キューに格納されうる生成要求の集合を M とする．ここで，生成要求は *Co*-タスク名としており，*Co*-タスク a_0 の生成要求は a_0 と表される．このとき，キューの内容，すなわちキュー内の生成要求列は

$$w_{\mathcal{Q}} \in M^*$$

で表される． □

定義 10 (キューの操作)
キューの操作に関するアクションは，キューを $\mathcal{Q}$ として $\mathcal{Q}!w$ もしくは $\mathcal{Q}?w$ の形で表される．ここで，$w \in M^*$ である．

システム内で動作中のあるハイブリッドオートマトンを $H = \langle L, l_0, X, C, D, I, Flow, Act, T, T_{init}, T_{end} \rangle$，キューを $\mathcal{Q}$ としたとき，$t_1 = (l_1, \phi_1, \mathcal{Q}!w, UPD(V_1), l_1') \in T$ であれば，遷移 t_1 を行う際に生成要求列 w をキューの最後尾に追加し，$t_2 = (l_2, \phi_2, \mathcal{Q}?w, UPD(V_2), l_2') \in T$ であれば，遷移 t_2 を行う際に生成要求列 w をキューの先頭から削除する． □

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

本研究においては，生成要求の集合は $M = \{a_0, a_1, b_0\}$ であり，キュー操作に関するアクションは，$\mathcal{Q}!a_0$, $\mathcal{Q}!a_1$, $\mathcal{Q}!b_0$, $\mathcal{Q}?a_0$, $\mathcal{Q}?a_1$, $\mathcal{Q}?b_0$ となる．

3.1.2 大域状態と大域遷移

ここでは，システム全体の状態として大域状態を定義し，システムの意味について述べる．

定義 11 (大域状態)
大域状態 γ は

$$\gamma = \langle q, w_{\mathcal{Q}} \rangle$$

で表される．

- $q \in Q$ は既に生成されている *DLHA* の状態．
- $w_{\mathcal{Q}} \in M^*$ はキュー $\mathcal{Q}$ に格納されている生成要求列． □

定義 12 (システムの意味)
システムの意味 $\mathcal{M}_S$ は，大域状態間の遷移関係 *(*以下，大域遷移*)*$\rightarrow$ を用いて以下のように定義する．$\mathcal{M}_S = \langle \Gamma, \rightarrow, \gamma_0 \rangle$

- $\Gamma \subseteq Q \times M^*$ は大域状態 γ の集合．
- $\rightarrow \subseteq \Gamma \times \{\tau\} \times \Gamma$ は大域遷移．
  - τ は内部アクション．大域状態においては全てのアクションが内部アクションとなる．
  - 任意の状態 q について，$q \Rightarrow_\delta q'$ となる時間遷移 $\Rightarrow_\delta$ が存在するとき，$\langle q, w_{\mathcal{Q}} \rangle \xrightarrow{\tau} \langle q', w_{\mathcal{Q}} \rangle$ である．
  - 任意の状態 q について，$q \Rightarrow_d q'$ となる，キューに対する追加アクション $\mathcal{Q}!w$ による離散遷移 $\Rightarrow_d$ が存在するとき，$\langle q, w_{\mathcal{Q}} \rangle \xrightarrow{\tau} \langle q', w'_{\mathcal{Q}} \rangle$ である．ただし，$w'_{\mathcal{Q}} = w_{\mathcal{Q}} \cdot w$ である．
  - 任意の状態 q について，$q \Rightarrow_d q'$ となる，キューに対する削除アクション $\mathcal{Q}?w$ による離散遷移 $\Rightarrow_d$ が存在するとき，$\langle q, w_{\mathcal{Q}} \rangle \xrightarrow{\tau} \langle q', w'_{\mathcal{Q}} \rangle$ である．ただし，$w_{\mathcal{Q}} = w \cdot w'_{\mathcal{Q}}$ である．
  - 任意の状態 q について，$q \Rightarrow_d q'$ となる，キューに対する追加，削除アクション以外のアクションによる離散遷移離散遷移 $\Rightarrow_d$ が存在するとき，$\langle q, w_{\mathcal{Q}} \rangle \xrightarrow{\tau} \langle q, w_{\mathcal{Q}} \rangle$ である．
- $\gamma_0 = \langle q_0, \varepsilon \rangle$ は初期大域状態．初期大域状態とは，初期状態 q_0 においてキューの内容が空文字 ε の状態である．

3.2 大域状態空間の探索

初期大域状態から到達可能な全ての大域状態の集合を大域状態空間という．大域状態空間の探索を行う際，キューに格納されている生成要求列を単に保持すると状態爆発が起こる可能性が極めて高くなる．これを抑えるために，本研究では QDD(Queue-content Decision Diagram) を用いてキューの内容を保持し，ループ優先探索による状態空間の探索を行うものとする[8)]．

3.2.1 キューの内容の表現

キューの内容を表現するための QDD と，QDD に対する操作について定義する．

定義 13 (QDD)
QDD は以下のような有限語 $w_{\mathcal{Q}}$ 上の決定性有限オートマトン $\mathcal{A} = (M, S, \Delta, s_0, F)$ で定義される．

$$\forall w_{\mathcal{Q}} \in L(\mathcal{A})$$

ここで，$L(\mathcal{A})$ は $\mathcal{A}$ が受理する言語．

- M はキューに格納されうる生成要求の集合．
- S は状態の集合．
- $\Delta \subseteq S \times \Sigma \times S$ は遷移の集合．
- $s_0 \in S$ は初期状態．
- $F \subseteq S$ は受理状態の集合．

□

定義 14 (キューへの操作)
キュー $\mathcal{Q}$ の最後尾に生成要求列 w を追加したのち *QDD* $\mathcal{A}$ が受理する言語を $L_{\mathcal{Q}!w}(\mathcal{A})$，キュー $\mathcal{Q}$ の先頭から生成要求列 w を削除したのち *QDD* $\mathcal{A}$ が受理する言語を $L_{\mathcal{Q}?w}(\mathcal{A})$ とすると，

$$L_{\mathcal{Q}!w}(\mathcal{A}) = \left\{ w'_{\mathcal{Q}} \mid \exists w_{\mathcal{Q}} \in L(\mathcal{A}) \colon w'_{\mathcal{Q}} = w_{\mathcal{Q}} \cdot w \right\}$$
$$L_{\mathcal{Q}?w}(\mathcal{A}) = \left\{ w'_{\mathcal{Q}} \mid \exists w_{\mathcal{Q}} \in L(\mathcal{A}) \colon w_{\mathcal{Q}} = w \cdot w'_{\mathcal{Q}} \right\}$$

と表される． □

3.2.2 ループ優先探索

大域状態空間の探索に用いる，メタ遷移とループ優先探索について定義する．

定義 15 (メタ遷移)
メタ遷移とは，システムにおける *DLHA* の記述にループが存在するとき，ループを繰り返した後，到達可能な全ての大域状態を生成する操作であり，以下の *2* つの操作がある．

ⓒ 2011 Information Processing Society of Japan

Vol.2011-OS-119 No.10
Vol.2011-EMB-23 No.10
2011/11/29

- $(\mathcal{Q}!w)^*$：キュー $\mathcal{Q}$ の最後尾に生成要求列 $w \in M^*$ を $k(\geq 0)$ 個追加したのち取りうるキューの内容の和集合を求める操作.
- $(\mathcal{Q}?w)^*$：キュー $\mathcal{Q}$ の先頭から生成要求列 $w \in M^*$ を $k(\geq 0)$ 個削除したのち取りうるキューの内容の和集合を求める操作.

QDD を $\mathcal{A}$ として，メタ遷移 $(\mathcal{Q}!w)^*$ を行ったのち QDD $\mathcal{A}$ が受理する言語を $L_{(\mathcal{Q}!w)^*}(\mathcal{A})$，メタ遷移 $(\mathcal{Q}?w)^*$ を行ったのち QDD $\mathcal{A}$ が受理する言語を $L_{(\mathcal{Q}?w)^*}(\mathcal{A})$ とすると，

$$L_{(\mathcal{Q}!w)^*}(\mathcal{A}) = \{w'_{\mathcal{Q}} \mid \exists w_{\mathcal{Q}} \in L(\mathcal{A}), k \geq 0 \colon w'_{\mathcal{Q}} = w_{\mathcal{Q}} \cdot w^k\}$$
$$L_{(\mathcal{Q}?w)^*}(\mathcal{A}) = \{w'_{\mathcal{Q}} \mid \exists w_{\mathcal{Q}} \in L(\mathcal{A}), k \geq 0 \colon w_{\mathcal{Q}} = w^k \cdot w'_{\mathcal{Q}}\}$$

と表される． □

定義 16 (ループ優先探索)

ループ優先探索は，以下の手順で実行される．

(1) 初期大域状態を到達可能状態の集合に加え，探索を開始する．

(2) 各状態において，メタ遷移が実行可能であれば優先的にメタ遷移を実行する．

(3) 求められた状態が到達可能状態の集合に含まれていなければ，到達可能状態の集合にこれを加え，ステップ *(2)* に戻る．含まれていた場合，そのパスについての探索を終了する．

(4) 全てのパスについて探索が終了したところで大域状態空間の探索を終了する．

4. まとめと今後の課題

本論文で我々は CPU と DRP とが協調動作するような組み込みシステムの仕様記述するため，DLHA を提案した．提案法では動的なオートマトンの生成・破棄を記述できるため，静的なものと比べロケーション数を削減することが可能となった．前例研究[7] の静的モデルを扱った場合と比べて，本稿で提案した DLHA を用いた場合では，*Co-task* の生成待ちの待機ロケーション等を削除できたため，*Co-task* のロケーション数を 3 から 1 に抑えることができた．さらに不要なオートマトンは消滅させることができるので解析中は状態数は減ると思われる．

また今後の課題として，提案法を用いた検証法，及び検証器の開発が挙げられる．静的モデルを扱った前例研究[7] では検証において状態爆発が発生していたが，ロケーション数を削減することができた提案法を用いれば検証時の状態空間の削減をすることができ，状態爆発が回避できると考えられる．

参 考 文 献

1) 天野英晴, 安達義則, 堤聡, 石川健一郎: 動的リコンフィギャブルプロセッサにおける可変クロック機構の導入. *IEICE technical report*, Vol.104, pp.13-16, 2005.
2) V. Tuan, Y. Hasegawa, N. Katsura and H. Amano. Performance Evaluation of Hardware Multi-process Execution on the Dynamically Reconfigurable Processor, *IEICE technical report*, Vol.106, pp.25-30, 2006.
3) 本村真人, 藤井太郎, 古田浩一朗, 安生健一朗, 矢部義一, 戸川勝巳, 山田順也, 伊澤義貴, 佐々木僚子. 新世代マイクロプロセッサアーキテクチャ（後編）:3. 実例 4. 動的再構成プロセッサ (DRP). 情報処理学会誌, Vol.46, pp.1259-1265, 2005.
4) R. Alur, C. Courcoubetis, T. A. Henzinger and P. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. *LNCS*, Vol.736, pp.209-229, 1992.
5) V. M. Tuan and H. Amano. A Mapping Method for Multi-Process Execution on Dynamically Reconfigurable Processors. *IEICE Transactions on Information and Systems*, Vol.11, pp.2312-2322, 2008.
6) 長谷川陽平, 天野英晴, 阿部昌平, 黒瀧俊輔, トウアン ヴ マン. 動的リコンフィギャラブルプロセッサにおける時分割多重実行の性能と消費電力の解析. *IEICE technical report*, Vol.105, No.287, pp.31-36, 2005.
7) 南翔太，瀧内新悟，瀬古口智，中居佑輝，山根智．動的再構成可能プロセッサのモデル化，仕様記述とモデル検査．コンピュータソフトウェア, Vol.28, No.1, pp.190-216, 2011.
8) B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. *Formal Methods in System Design*, Vol.14, pp.237-255, 1999.
9) P. C. Attie and N. A. Lynch. Dynamic Input/Output Automata: A Formal Model for Dynamic Systems. *LNCS*, Vol.2154, pp.137-151, 2001.

ⓒ 2011 Information Processing Society of Japan