Vol.2011-SE-173 No.2
2011/7/21

時間ペトリネットでモデル化された非同期システムに対するモデル検査ツール UPPAAL の適用

三輪陽介[†1]　横川智教[†1]　宮崎仁[†2]
近藤真史[†1]　佐藤洋一郎[†1]

本論文では，非同期システムの一種である GALS(大域非同期局所同期) システムの特性を，モデル検査ツール UPPAAL を用いて検証する手法を提案する．ここでは，STPN(確率時間ペトリネット) でモデル化された GALS システムを対象として，与えられた STPN を UPPAAL の入力形式である XTA(拡張時間オートマトン) へと変換することで，UPPAAL による検証を行う．提案法では，STPN の各プレイスの振る舞いを XTA で表現し，同期チャネルを用いて XTA 間を同期することで STPN 全体の振る舞いを表現する．最後に，例題へと提案法を適用してその有効性を示すとともに，従来手法との比較も合わせて行う．

Verifying Asynchronous Systems Modeled by Timed Petri Nets Using UPPAAL

Yosuke MIWA,[†1] Tomoyuki YOKOGAWA,[†1]
Hisashi MIYAZAKI,[†2] Masafumi KONDO[†1]
and Yoichiro SATO[†1]

In this paper, we propose a method to verify a GALS system modeled by an STPN(Stochastic Timed Petri Net) using model checker UPPAAL. For this purpose, we propose the method to convert STPN to XTA(eXtended Timed Automata), which is an input of UPPAAL. In our method, the places in the STPN are represented by an XTA, and they are synchronized by using broadcast channels. We also develop a tool to convert a file describing the structure of a STPN to XTA as an input format of UPPAAL. Finally, we carried out a comparison experiment of the size of generated model and the cost of verification by UPPAAL.

1. まえがき

現在のディジタルシステムの多くは同期式回路として実現されるが，システムの大規模化に伴う配線遅延の増加により，同位相のクロックパルスの分配は困難となる．そのため，正常な動作を保証するためにはクロック周波数を低くせざるを得ず，同期式回路の動作速度には構造的な限界が生じつつある．この問題の解決を目的として，システムを完全非同期式回路として設計する方法が提案されてはいるが[1),2)]，非同期式回路の設計環境は十分に整っていないため，大規模システムを完全非同期式回路として設計することは現実的とはいえない．そこで，現在の大規模システムの多くは，システムを構成するコンポートネントを複数のサブシステムに分割した上で，サブシステムを同期式回路として構成し，サブシステムを 1 つのコンポーネントとみなしてシステムを非同期式回路として構成する大域非同期局所同期 (Globally Asynchronous Locally Synchronous, GALS) システムとして設計されている[3),4)]．GALS システムの性能はサブシステムの構成や非同期バスのアーキテクチャに大きく依存するため，その設計の最適化を目的として，GALS システムを確率時限ペトリネット (Stochastic Timed Petri Nets, STPN) によってモデル化し，シミュレーションにより性能評価を行う手法の開発が進められている[5)]．しかし，この手法はあくまで性能評価のためのものであり，GALS システムが与えられた仕様を満たすことを保証するためのものではない．

本論文では，実時間システムを対象としたモデル検査[6)] ツールの一種である UPPAAL[7),8)] を用いて，GALS システムの特性を検証する手法を提案する．ここでは，検証対象となる GALS システムが STPN によってモデル化されていることを仮定する．その上で，与えられた STPN を UPPAAL の入力形式である拡張時間オートマトン (eXtended Timed Automata, XTA) へと変換することにより，UPPAAL による検証を実現する．まず，UPPAAL では確率的振る舞いを検証の対象としないため，与えられた STPN から確率的要素を取り除き，時相ペトリネット (Timed Petri Net, TdPN) による表現を求める．そして，得られた TdPN の各プレイスの振る舞いをそれぞれ XTA によって表現し，その上でトランジションによる接続関係をもとに各 XTA の動作を同期することで，TdPN 全体の振る舞いを表現する．さ

らに，UPPAAL のクエリ言語によって，GALS システムが満たすべき特性を記述する．最後に，既存の TdPN から XTA への変換手法[9] との比較を行うことで，提案法の有効性を示している．

2. モ デ ル

2.1 STPN

GALS システムのモデル化においては，局所的な同期式動作と大域的な非同期動作を表現しなければならない．GALS システムでは，同一クロックで動作するサブシステム (クロックドメインという, Clock Domain, CD) 間の通信時に動作の同期化処理を行うが，その結果は確率的に決定される．さらに，メタステーブルの動作に起因して，同期化処理の所要時間は指数分布となる[10]．STPN は確率要素と時間要素を表現可能なモデルであり，GALS システムをはじめとして非同期システムをモデル化するために広く用いられている．

STPN は，プレイスの集合 P，トランジションの集合 T，アークの集合 $F \subseteq P \times T \cup T \times P$，遅延分布の集合 D，プレイスに対する遅延分布の割り当て $X : P \rightarrow D$，選択確率 $\mu : F \rightarrow (0,1)$，初期マーキング $M_0 \subset P$ による組 $(P, T, F, D, X, \mu, M_0)$ として定義される．

$p \in P$，$t \in T$ に対して，p から t(t から p) へのアークが存在するときかつそのときのみ，$(p,t) \in F((t,p) \in F)$ となる．アークは通常アーク F_{not_in} と抑止アーク F_{in} から構成される．$(F_{in} \cup F_{not\_in} = F, F_{not\_in} \cap F_{in} = \phi)$．$p \in P$ に対して，$(t,p) \in F_{not\_in}((p,t) \in F_{not\_in})$ を満たすトランジション $t \in T$ の集合をそれぞれ入力 (出力) トランジションと呼び，$\bullet p(p\bullet)$ と記す．また $t \in T$ に対して，$(p,t) \in F_{not\_in}((t,p) \in F_{not\_in})$ を満たす $p \in P$ の集合を入力 (出力) プレイスと呼び，$\bullet t(t\bullet)$ と記す．$p \in P$ に対して，$(p,t) \in F_{in}$ を満たす $t \in T$ の集合を被抑止トランジションと呼び，$p\circ$ と記す．$t \in T$ に対して，$(p,t) \in F_{in}$ を満たす $p \in P$ の集合を抑止プレイスと呼び，$\circ t$ と記す．

STPN で記述されたシステムの状態は，各プレイスに付与されたトークンの数により表現される．本論文で対象とする STPN では，プレイスがもつトークンの数を最大 1 個とし，トークンをもつプレイスの集合 $M \subset P$ でシステムの状態を表現する．M をマーキングといい，システムの初期状態におけるマーキングを初期マーキング M_0 という．

プレイス p が獲得したトークンが有効となるまでの時間は，遅延分布 $X(p) \in D$ に基づいて決定される．遅延分布としては，(1) 一様分布，(2) 正規分布，(3) 指数分布，(4) 固定値のいずれかを与える．

トランジション t は全ての $p \in \bullet t$ に対して，$p \in M$ かつ p のトークンが有効ならば発火可能であり，このとき $|p\bullet| = 1$ ならば確実に発火する．トランジション t が発火すると，全ての $p \in \bullet t$ のトークンが失われ，$p \in t\bullet$ の全てがトークンを獲得する．これによりマーキングが変化し，システムの状態遷移が行われる．

2 つ以上の出力トランジションをもつプレイス ($|p\bullet| > 1$) では，トランジションの競合処理が必要となる．このようなプレイスを *choice* プレイスといい，*choice* プレイスにおいては出力トランジション $t \in p\bullet$ のいずれが発火するかは選択確率 $\mu(p,t)$ に従って確率的に決定される．

抑止アークをもつプレイスに有効なトークンが存在する場合，被抑止トランジションは発火できない．抑止プレイスのトークンが消費された瞬間に，被抑止トランジションが発火可能であれば，直ちに発火処理が行われる．

2.2 XTA

UPPAAL は，XTA によってモデル化されたシステムが与えられた特性を満たすか否かをモデル検査の枠組みに従って自動検証する．XTA は，標準的な時間オートマトンに対して UPPAAL 独自の拡張を行ったものである．また，UPPAAL ではクエリ言語として，時相論理 TCTL(Timed Computational Tree Logic) のサブセットを用いて検証すべき特性を記述する．

時間オートマトン TA は 6 項組 (L, l_0, C, A, E, I) である．ここで L は状態の集合であり，$l_0 \in L$ は初期状態，C はクロックの集合，A はアクションの集合，$E \subseteq L \times A \times B(C) \times L$ は状態間の遷移の集合，$I : L \rightarrow B(C)$ は各状態に対する不変条件の割当である．ここで $B(C)$ はクロックに対する条件式の論理積からなる集合である．アクションは他のプロセスとの同期と，クロックのリセット処理から構成される．遷移は状態とアクション，ガードから構成される．不変条件とは，状態が常に満たすべき条件である．アクションは他のプロセスとの同期とクロックのリセットから構成される．

時間オートマトンにおける状態遷移は，時間経過と遷移実行の 2 つがある．前者の場合は不変条件を違反しない範囲で全てのクロックが増加する．後者の場合は，遷移に従って，同期処理及びクロックのリセットが行われる．

UPPAAL における時間オートマトンに対する拡張は 7 種類あるが，ここでは本論文のモデルで用いる拡張についてのみ述べる．

Constant　定数を定義するための拡張．

Variable　整数型変数，二値変数および配列を定義するための拡張．ガード条件およびア

クションの記述に用いる．

Broadcast channel 複数のオートマトン間の遷移を同期するための拡張．同期は *broadcast channel* を用いた同期チャネルと呼ばれる信号の送受信によって行われる．

Committed location 特殊な状態として，時間経過がなく，その状態からの遷移が他の遷移よりも優先して実行される *committed location* を定義するための拡張．

2.3 クエリ言語

UPPAAL のクエリ言語は TCTL のサブセットであり，パス論理式と状態論理式から構成される．状態論理式は，モデル内の変数と状態に関する論理式として記述できる．また，デッドロック，すなわちその状態から実行可能な遷移が存在しない状態であることを，特殊な状態論理式 `deadlock` で表現することができる．

UPPAAL では以下の 5 種類の時相論理演算子が利用可能である．ここで，ϕ,ψ は状態論理式とする．

- `A[]` ϕ　全てのパスで ϕ は常に成立する．
- `A<>` ϕ　全てのパスで ϕ がいつか成立する．
- `E[]` ϕ　常に ϕ が成立するパスが存在する．
- `E<>` ϕ　いつか ϕ が成立するパスが存在する．
- ψ `~>` ϕ　ψ が成立するときは ϕ がいつか成立する．

3. STPN から XTA への変換

3.1 変換の枠組み

STPN は *choice* プレイスの出力トランジションの発火における選択確率およびプレイスに割り当てられた遅延分布として確率的動作を表現する．しかし，UPPAAL が採用しているモデル検査のアルゴリズムでは実行可能な遷移を網羅的に探索するため，選択確率で表現される遷移の重みは考慮する必要がない．同様に，遅延分布に関しても，とりうる遅延の範囲を最大値および最小値として考慮すればよい．従って，本論文では与えられた STPN$(P, T, F, D, X, \mu, M_0)$ に以下の変換を行うことで得られる時限ペトリネット (Timed Petri Nets，TPN)$(P, T, F, x_{\min}, x_{\max}, M_0)$ を対象として XTA への変換を行う．

遅延の最大値 $x_{\max} : P \to (0, \infty]$ および最小値 $x_{\min} : P \to [0, \infty)$ は STPN の遅延分布 D およびプレイスへの遅延分布の割り当て X に対して以下のように求める．

プレイス p_i に対して，

- $X(p_i)$ が固定値 d_f ならば，$x_{\min}(p_i) = x_{\max}(p_i) = d_f$.
- $X(p_i)$ が一様分布 $[u_{\min}, u_{\max}]$ ならば，$x_{\min}(p_i) = u_{\min}$，$x_{\max}(p_i) = u_{\max}$.
- $X(p_i)$ が正規分布もしくは指数分布ならば，$x_{\min}(p_i) = 0$，$x_{\max}(p_i) = \infty$.

TdPN を XTA で表現するために，まずプレイス p_i に対して，p_i が有効なトークンをもつときかつそのときのみ `p_i` $= 1$ となる二値変数 `p_i` と，トークン獲得からの経過時間をカウントするクロック変数 `x_i` を定義する．そして，トランジション t_j に対して t_j の発火を表す同期チャネル `t_j` を定義する．その上で，定義した変数および同期チャネルを用いて各プレイスの振る舞いを XTA で表現する．以下，与えられた TdPN の各プレイスの XTA 表現法を，通常のプレイスの場合と抑止プレイスの場合に分けてそれぞれ述べる．

3.2 プレイスの XTA 表現

プレイス p_i は，入力トランジション $t_k \in {}^\bullet p_i$ の発火によりトークンを獲得する．そして $x_{\min}(p_i)$ から $x_{\max}(p_i)$ が経過するまでにトークンは有効となる．トークンが有効となった瞬間に出力トランジション $t_j \in p_i{}^\bullet$ のいずれかが発火可能であれば，トランジション t_j はただちに発火し，p_i のトークンは消費される．発火可能なトランジションが存在しなければ，トークンをもったまま出力トランジションが発火可能となるまで待機する．

この動作は図 1 に示す構造をもつ XTA で表現できる．以下，この XTA がもつ状態および遷移について，対応するプレイスの振る舞いとともに述べる．

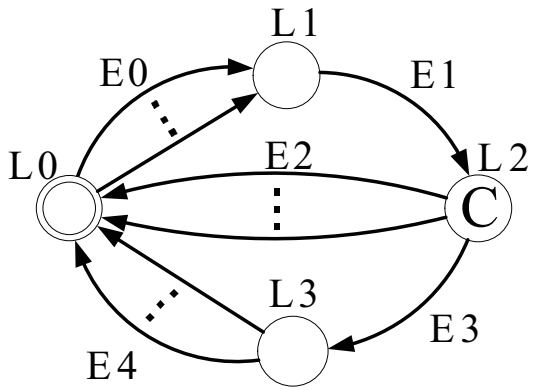

図 1 プレイスを表現する XTA の構造

L0： プレイス p_i がトークンをもたない状態を表す．

E0： **L0** から **L1** への遷移であり，入力トランジション $t_k \in \bullet p_i$ の発火によるトークンの獲得を表す．全ての $t_k \in \bullet p_i$ に対して，t_k の発火を表す同期チャネル受信 `t_k?`をもち，経過時間の初期化のためのアクション `x_i`:=0 をもつ．同期チャネルの受信は 1 つの遷移に 1 つしか記述できないため，この遷移は入力トランジションと同数だけ設ける必要がある．

L1： プレイス p_i がもつトークンがまだ有効でなく，定められた遅延時間が経過するのを待っている状態を表す．p_i が獲得したトークンは，最大遅延値 $x_{\max}(p_i)$ が経過するまでには必ず有効となる．有効となったトークンは必ず発火に使用されるため，$x_{\max}(p_i)$ が経過した後にこの状態に留まることは許されない．そのため，不変条件として経過時間が最大遅延値を越えないことを表す論理式

$$\texttt{x\_i} \leq x_{\max}(p_i)$$

を与える．ただし，$x_{\max}(p_i) = \infty$ ならば，経過時間が最大遅延値を越えることはないため，この不変条件は不要である．

E1： **L1** から **L2** への遷移であり，プレイス p_i のトークンが有効となることを表す．従って，プレイス p_i のトークンを有効にするアクション `p_i:=1` をもつ．ここで，p_i が獲得したトークンは，最小遅延値 $x_{\min}(p_i)$ を経過するまで有効とはならない．そのため，ガード条件として経過時間が最小遅延値以上であることを表す論理式

$$\texttt{x\_i} \geq x_{\min}(p_i)$$

を与える．

L2： プレイス p_i のトークンが有効になった瞬間の状態を表す．この状態は，トークンが有効となることにより出力トランジションが発火できるか否かを判定し，処理を分岐するために設けられる．もし出力トランジションが発火可能であれば，ただちに発火が行われるため，状態 **L2** は *committed location* として定義する．

E2： **L2** から **L0** への遷移であり，出力トランジション $t_j \in p_i\bullet$ のいずれかが発火可能であったとき，t_j の発火によるトークンの消費を表す．t_j は入力プレイス $\bullet t_j$ の全てが有効なトークンをもち，かつ抑止プレイス $\circ t_j$ の全てが有効なトークンをもたないとき，発火可能となる．従って，ガード条件として論理式

$$\bigwedge_{p_l \in \bullet t_j \setminus p_i} (\texttt{p\_l} == 1) \wedge \bigwedge_{p_m \in \circ t_j} (\texttt{p\_m} == 0)$$

をもつ．また，t_j の発火を表す同期チャネル送信 `t_j!`と，p_i のトークンを消費するアクション `p_i`:=0 をもつ．$t_j \in p_i\bullet$ の全てに対して，t_j の発火はそれぞれ個別に起こりうるため，この遷移は出力トランジションごとに設ける必要がある．

E3： **L2** から **L3** への遷移であり，出力トランジション $t_j \in p_i\bullet$ のいずれも発火できなかったときの分岐を表す．従って，ガード条件として出力トランジション $t_j \in p_i\bullet$ の全てが発火可能でないことを表す論理式

$$\bigwedge_{t_j \in p_i\bullet} \left\{ \neg \Big(\bigwedge_{p_l \in \bullet t_j \setminus p_i} (\texttt{p\_l} == 1) \wedge \bigwedge_{p_m \in \circ t_j} (\texttt{p\_m} == 0) \Big) \right\}$$

をもつ．

L3： プレイス p_i が有効なトークンをもつが，出力トランジションがいずれも発火不可能である状態を表す．

E4： **L3** から **L0** への遷移であり，他のプレイスのトークンが有効になることによる p_i の出力トランジション $t_j \in p_i\bullet$ の発火および p_i のトークンの消費を表す．t_j の発火を表す同期チャネル受信 `t_j?`と，p_i のトークン消費のためのアクション `p_i`:=0 をもつ．**E2** と同様の理由により，この遷移は出力トランジションごとに設ける必要がある．

初期マーキング M_0 に含まれるプレイス p_i に対しては，p_i は有効なトークンをもつため，`p_i` の初期値は 1 となり，かつ **L2** が初期ロケーションとなる．一方，p_i が M_0 に含まれなければ，p_i はトークンをもたないため，`p_i` の初期値は 0 となり，かつ **L0** が初期ロケーションとなる．

3.3 抑止プレイスの XTA 表現

プレイス p_i が抑止プレイスであるとき，p_i のトークンが有効ならば，被抑止トランジション $t_h \in p_i\circ$ は発火できない．抑止プレイス p_i のトークンが消費された瞬間に被抑止トランジション $t_h \in p_i\circ$ のいずれかが発火可能であれば，トランジション t_h はただちに発火し，p_i のトークンは消費される．

この動作は図 2 に示す構造をもつ XTA で表現できる．以下，図 1 の XTA に対して追加もしくは変更された状態および遷移について，対応する抑止プレイスの振る舞いとともに述べる．

L4: p_i のトークンが消費された瞬間の状態を表す．この状態は，トークンが消費されることにより，被抑止トランジションが発火できるか否かを判定し，処理を分岐するために設けられる．もし，被抑止トランジションが発火可能であれば，ただちに発火が行われるため，状態 **L4** は *committed location* として定義する．また，発火可能となった被抑止トランジションは全て発火するため，全ての被抑止トランジションが発火可能でな

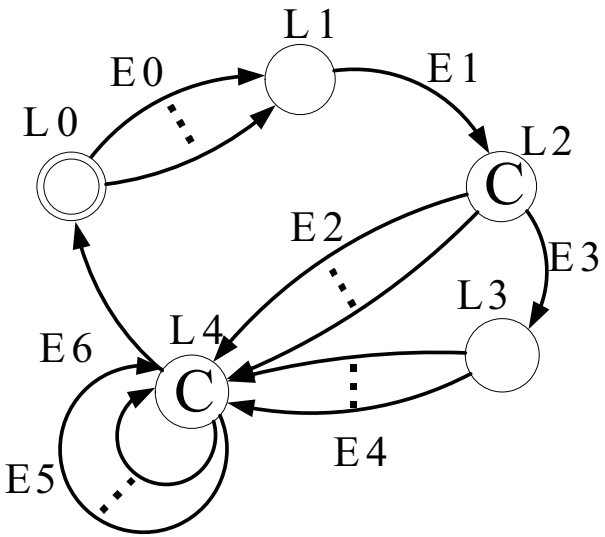

図 2 抑止プレイスを表現する XTA の構造

くなるまで，発火可能か否かの判定は繰り返し行われる．

E5： **L4** 自身への自己遷移であり，p_i のトークンの消費による被抑止トランジション $t_h \in p_i\circ$ の発火を表す．従って，ガード条件として被抑止トランジション $t_h \in p_i\circ$ が発火可能であることを表す論理式

$$\bigwedge_{p_l \in \bullet t_h} (\mathtt{p\_l} == 1) \wedge \bigwedge_{p_m \in \circ t_h \setminus p_i} (\mathtt{p\_m} == 0)$$

をもつ．また，t_h の発火を表す同期チャネル送信 `t_h!`をもつ．$t_h \in p_i\circ$ にの全てに対して，t_h の発火はそれぞれ個別に起こりうるため，この遷移は被抑止トランジションごとに設ける必要がある．

E6 **L4** から **L0** への遷移であり，発火可能な被抑止トランジションが存在せず，トークンが無い状態へと戻ることを表す．従って，ガード条件として発火可能な被抑止トランジションが存在しないことを表す論理式

$$\bigwedge_{t_h \in p_i\circ} \Big\{ \neg \Big(\bigwedge_{p_l \in \bullet t_h} (\mathtt{p\_l} = 1) \wedge \bigwedge_{p_m \in \circ t_h \setminus p_i} (\mathtt{p\_m} = 0) \Big) \Big\}$$

をもつ．

初期状態については，プレイスの XTA 表現と同様に決定される．

4. 比 較 実 験

本節では，これまでに開発した変換手法[9] との比較を通して，提案した変換法についての評価を行う．

4.1 実 験 環 境

本実験では，OS として Fedora 12 がインストールされており，Intel(R) Core(TM)2 Duo 3.16GHz の CPU および 3.7GB のメモリをもつ計算機上で検証を行った．また，UPPAAL のバージョンは 4.0.10 であり，検証オプションの設定はそれぞれ以下の通りである．

探索順序	:	幅優先
状態空間の削減	:	保守的
状態空間表現	:	DBM
外挿法	:	自動

検証対象とする GALS システムの構成は，3 コンポーネント ×1 クロックドメインのもの (モデル 1) と，2 コンポーネント ×2 クロックドメインのもの (モデル 2) の 2 種類とする．各モデルのサイズについては表 1 に示す．

表 1 検証に用いるモデルの構成およびサイズ

	GALS 構成	プレイス数	トランジション数
モデル 1	$3C \times 1CD$	55	63
モデル 2	$2C \times 2CD$	140	159

4.2 検証する特性

本実験では，以下の 2 つの特性を対象として検証を行った．

- あるコンポーネントがデータ送信を要求すると将来必ずデータ転送が完了する（可達性）．
- 常にデッドロックが発生しない（安全性）．

それぞれの特性は UPPAAL のクエリ言語を用いて以下のように表現できる．

可達性 本手法が対象とする STPN モデルは，対応する GALS システムの各コンポーネントに対して，データ送信の要求およびデータ送信の完了を表すプレイスをそれぞれも

つ．そのプレイスを p_{req} および p_{end} とおくと，上記の可達性は次のように記述できる．

```
p_req==1 ~> p_end==1
```

安全性 UPPAAL ではデッドロックであることを特殊な状態論理式 deadlock で表現することができるため，これを用いて安全性は次のように記述できる．

```
A[] not deadlock
```

4.3 比較結果

モデル 1 およびモデル 2 に対して，従来法および提案法を用いて XTA への変換を行い，前述の 2 つの特性に関して UPPAAL を用いて検証を行った．生成されたモデルサイズおよび UPPAAL による検証時間を比較結果を表 2 および表 3 に示す．

表 2 モデルサイズの比較

		従来法	提案法	削減率
モデル 1	XTA 数	118	55	53.4 %
	状態数	333	226	32.1 %
	遷移数	571	582	-1.9 %
モデル 2	XTA 数	229	140	38.9 %
	状態数	826	612	25.9 %
	遷移数	1830	1340	26.8 %

表 3 検証時間の比較 (単位：秒)

	検証特性	従来法	提案法	検証結果
モデル 1	可達性	0.5	0.5	False
	安全性	1074	1938	True
モデル 2	可達性	6.47	3.25	False
	安全性	N/A	N/A	N/A

検証結果に関しては，表 3 に示すように 2 つの特性のいずれに対しても同じ結果が得られた．ここで，可達性に対してはモデルに誤りがあるという結果が得られた．これは各プレイスに対する遅延値の割り当てに原因があり，意図した通りに遷移が実行されなかったためだった．また，安全性に関しては，モデル 1 についてはデッドロックがないことが示された．しかしながら，モデル 2 については，いずれの XTA に対しても状態爆発によるメモリ不足のため検証が終了しなかった．

従来の変換法と比較すると，モデルサイズについては表 2 に示すように，モデル 1 の遷移数を除く全てに対してモデルサイズの大きな削減に成功している．モデル 1 の遷移数についてもほぼ同等のモデルサイズであり，より規模の大きなモデル 2 については全体的に削減傾向にあることから，提案法はモデルサイズの削減に有効に働いていると考えられる．

検証時間については，モデル 1 の可達性はほぼ同等の時間で検証が完了したが，安全性については，従来法に基づく検証の方が提案法に比べてより短い時間で終了した．これは，提案法では XTA 数を削減するため一つ一つの XTA の処理はより複雑となっているが，システムの規模が小さいため，処理の複雑化によるコスト増加が XTA 数の減少による効果を上回っているためだと考えられる．よりシステムの規模の大きなモデル 2 に対しては，安全性については前述の通りどちらも終了しなかったが，可達性では提案法で生成した XTA による検証の方がより短い時間で終了している．

以上の結果から，提案法によるモデルサイズ削減は検証コストに影響を与えていると考えられるが，大規模なシステムを対象として検証を行うためにはまだ不十分である．従って，大規模なモデルに対しても検証可能となるよう，提案法をもとにさらに遷移数を削減するよう改良を加える必要がある．

5. あとがき

本論文では，GALS システムをモデル化する STPN を XTA に変換し，UPPAAL を用いて検証する手法を提案した．本手法では，STPN をプレイス単位で XTA 表現することにより，従来の変換法に比べてモデルサイズの大幅な削減を可能としている．比較実験として，2 つの GALS システムの STPN モデルを提案法と従来法を用いて XTA へと変換し，モデルサイズおよび検証コストの比較を行った．今後の課題として，検証コスト削減への効果をより大きくするためのモデルサイズ縮小手法の検討が必要である．

Vol.2011-SE-173 No.2
2011/7/21

参 考 文 献

1) Sutherland, I. and Fairbanks, S.: GasP: a minimal FIFO control, *Seventh International Symposium on Asynchronous Circuits and Systems (ASYNC 2001)*, pp. 46–53 (2001).
2) Ebergen, J.: Squaring the FIFO in GasP, *Seventh International Symposium on Asynchronous Circuits and Systems (ASYNC 2001)*, pp.194–205 (2001).
3) Stein, M.: Crossing the abyss: asynchronous signals in a synchronous world, *EDN Magazine*, pp.59–69 (2003).
4) 今井　雅，Ozcan, M.，南谷　崇：SPI モデルに基づく局所同期式 VLSI 設計方式，情報処理学会論文誌，Vol.44, No.5, pp.1232–1243 (2003).
5) 田代和幸，横川智教，茅野　功，佐藤洋一郎，早瀬道芳：Globally Asynchronous Locally Synchronous システムの性能評価に関する一検討，電子情報通信学会技術研究報告. VLD，VLSI 設計技術，Vol.106, No.549(20070302), 社団法人電子情報通信学会，pp.57–62 (2007).
6) Clarke, E., Grumberg, O. and Peled, D.: *Model Checking*, MIT Press (1999).
7) Behrmann, G., David, A. and Larsen, K.G.: A Tutorial on Uppaal, *SFM* (Bernardo, M. and Corradini, F., eds.), Lecture Notes in Computer Science, Vol.3185, Springer, pp.200–236 (2004).
8) *UPPAAL*, http://www.uppaal.com (2004).
9) 桐田和明，横川智教，宮崎　仁，佐藤洋一郎，早瀬道芳：UPPAAL を用いた GALS システムの形式的検証手法の改良，平成 22 年度電気・情報関連学会中国支部連合大会 (2010).
10) Chaney, T.J. and Molnar, C.E.: Anomalous behavior of synchronizer and arbiter circuits, *IEEE transactions on Computers*, Vol.C-22, No.4, pp.421–422 (1973).