RC-006

多重様相論理による遅延依存非同期回路の形式検証体系
Formal Verification Method of Delay Sensitive Asynchronous Circuit using Multimodal Logic

西村 俊二 † 久我 守弘 † 飯田 全広 †
Shunji Nishimura Morihiro Kuga Masahiro Iida
尼崎 太樹 † 末吉 敏則 †
Motoki Amagasaki Toshinori Sueyoshi

1. はじめに

デジタル回路は同期回路と非同期回路に分類できるが，相対的に設計・検証が容易であることから同期回路の方が広く普及している．一般に同期回路と比べて設計・検証が難しいとされる非同期回路であるが，回路自体の性能に注目すると，スピードや消費電力において非同期回路の方が有利であるといえる．これは，同期回路のスピードは組み合わせ回路部分のワースト遅延を基準としたものとなること，また，本質的に動作していない箇所についてもクロックだけは常に動いており，電力を消費し続けることからである．

非同期回路の利点を活かすため，欠点である設計・検証の難しさを解決する試みがこれまでになされてきた．遅延非依存 (Delay Insensitive, DI) 回路についてはフロー・テーブルを用いた設計 [1] が確立されており，ステートを中心概念とした設計・検証が可能である．しかしながら，厳密な DI 回路は非同期回路全体の中で極めて限定的なクラスである [2] ことが知られており，入力信号のタイミングについて基本モードと呼ばれる制約を課すなどの処置が必須である．

本稿では同期回路や DI 回路に限定しない，遅延依存の回路を含む広い範囲について適用できる検証手法を提案している．これにより遅延依存回路の検証コストを大幅に下げることができれば，遅延依存回路は同期回路や DI 回路に並ぶ選択肢と成り得ると考える．スピード・消費電力・面積において特に高い要求がある場合に遅延依存回路が最適解となる可能性は十分にあると思われる．

提案の検証手法は様相論理の考え方である可能世界と到達可能性に基づいており，回路中の各信号をそれぞれ独立した世界と見なすことにより遅延値が不明な段階での検証を可能としている．この検証手法を用いて遅延依存の FIFO の検証を行った結果を示す．論理合成を行う前の早い段階で，どのようなタイミング制約が必要となるかを確定させることができた．また，実用化に向けた試みとして定理証明言語 Agda 上の実装もあわせて示す．

†熊本大学大学院自然科学研究科

2. 多重様相論理による検証手法

本稿で提案する検証手法について述べる．まず回路モデルを定義し，その後タイミング制約の導入となる．

2.1. 様相論理による回路モデル

まずクリプキ意味論を提示した後に様相論理に基づいた回路モデルを定義する．クリプキ意味論は様相論理の意味論として用いられる [3].

定義 1 （クリプキ構造）
以下の三つの要素の組 (W, R, V) をクリプキ構造という．

- W は可能世界の集合である．
- R は到達可能性と呼ばれる世界の間の二項関係である．
- V は各世界における命題の解釈である．

$w_0, w_1 \in W$ で $w_0 R w_1$ のとき，本稿では“ w_0 から w_1 への経路がある”と表現したり，矢印を使って $w_0 \to w_1$ と記す場合がある．

ネットリスト形式で表現された回路 C に対して，到達可能性は C の各要素（ゲートなど）が接続されている信号について，入力信号から出力信号へ到達可能と定義する．次に C のいくつかの信号を選び，それらを互いに独立した可能世界の始点として扱う．可能世界は始点から到達可能あるいは逆方向（出力から入力へ）により到達可能な信号の全体として定義する．（以降，通常の意味での回路の信号と特に区別すべき場合には“信号世界”と表現する．）また，各信号世界における命題は“その信号値が何であるか”とする．具体例として図 1 の左に示す回路については，右に示す可能世界と到達可能性関係が得られる．ここでは，AND ゲートの c から d へのパスを ${}^*AND$，もう一方を ${}_*AND$ と表している．

2.2. タイミング制約を考慮したプロパティ・チェック

様相論理による回路モデルの動作について図 1 の例を使って説明すると，インバータ及び AND ゲートの

Copyright © 2015 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

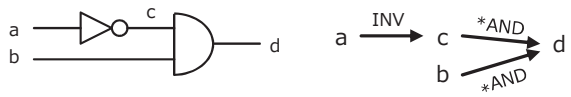


図 1: 簡単な回路のクリプキ構造

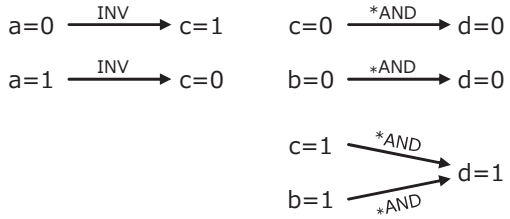


図 2: インバータ及び AND ゲートのプロパティ

プロパティを 2 の通りに定義するのが自然である．これを踏まえて，回路に入力 $a = 1$ が与えられたとすると，図 3 の推論により，$d = 0$ が導かれる．結論として

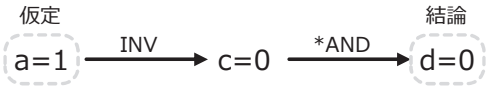


図 3: a=1 から導かれる命題

値が得られるだけでなく，図 3 の経路は $a = 1$ が与えられてから $d = 0$ となるまでの遅延を抽象的に表している．

様相論理による回路モデルで扱う抽象化された遅延は，入出力信号それぞれのパスを考慮に入れるため，古典的な遅延モデルであるワイヤーアンドゲート遅延モデル ([4] における Gate- and wire-state network) に相当すると言える．我々の回路モデルは時間の概念に従うことが求められるため，同じ信号についての信号世界の集合は全順序集合となっているものとする．

次に，2 つのパスの遅延に関して大小関係を規定する，すなわち相対的なタイミング制約を導入することができる．図 4 は簡単な遅延依存回路である．$a = 1$ に続いて $a = 0$ が入力された場合，$d = 0$ のままであるか短いパルスが出力されるかは，a から c を通って d に着くバスともう一方のパスの遅延の関係に依存する．

c を通るパスの方が遅いと仮定すると，図 5 上部のような Δ が存在する．この Δ を用いると図の下部のように，$d = 0$ のままであることが証明される．点線の矢印は“直前まで命題が成り立っていることの略記であり，厳密な定義は次の通りである．ある信号 S に関して $S = a \dashrightarrow S = b$ は，任意の $S = a \to S \to S = b$ となる間の S について $S = a$.

逆に，c を通るパスの方が早いと仮定すると，図 6 上部のような Δ が存在し，そのことから図中央の Δ' の存在も言える．Δ' を用いると図の下部のように，$d = 1$

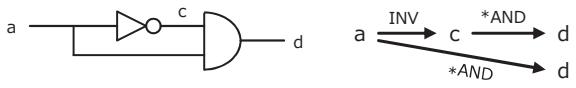


図 4: 簡単な遅延依存回路

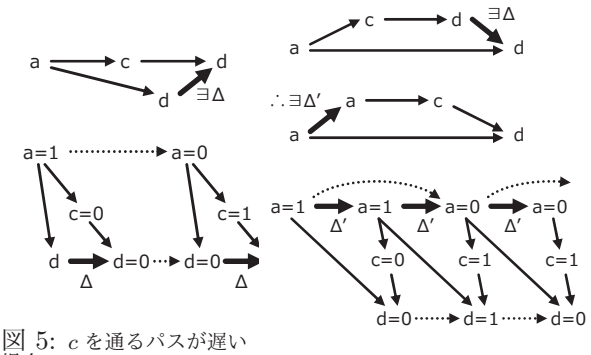


図 5: c を通るパスが遅い場合

図 6: c を通るパスが早い場合

の期間があることが証明される．

3. 非同期 FIFO の検証

Molnar らによる遅延依存の FIFO[5] を検証対象とする．この回路の性能は MOSIS による 0.6 ミクロン・プロセスで 1 秒間に 930 メガ個のスループットとされている．Molner らは検証手法にアナログ・シミュレータを用いており，網羅的な検証は行なわれていない．また，タイミング制約についてはシミュレータの結果を見ながらアドホックに構築する方法が採られた．これに対し，前節で提案した検証手法に基づけば，タイミング制約を含めた上で網羅的な検証が可能となる．

3.1. 対象回路: 非同期 FIFO

対象とする非同期 FIFO を図 7 に示す．FIFO の全

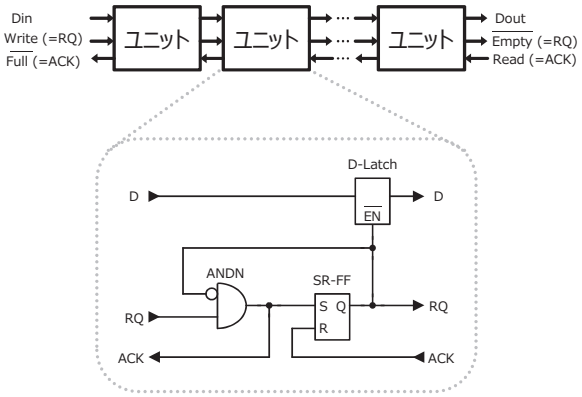


図 7: 非同期 FIFO

体は複数のユニットを直列に接続する構成となっており，名ユニットの内部は図に示される通り $D - Latch$

Copyright © 2015 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

（D ラッチ），$SR-FF$（SR-フリップフロップ）及び $ANDN$（片方がインバートされた AND ゲート）で構成される．Molnar らによる回路はすべてゲート又はトランジスタで構成してあるのに対し，本稿ではより粒度の粗いラッチや FF を用いており，これらを検証を行う上での最小単位としている．D はデータ信号，RQ はデータの取り込みを促すリクエスト信号，ACK はデータの取り込みを終えたことを示すアクノリッジ信号である．

この FIFO の動作について簡単に説明する．各 RQ，ACK 信号がすべて 0 で安定しているとする．このとき，$D-Latch$ は取込み状態となっている．この状態で左のユニットから $RQ=1$ が来た場合，$ANDN$ の出力として左のユニットへの $ACK=1$ が生成される．一方で $SR-FF$ の値がセットされるため右のユニットへの $RQ=1$ が生成され，また，$D-Latch$ は保持状態となる．すなわち，左から $RQ=1$ が来た時点での左からの D の値が保持される．その後右からの $ACK=1$ が来ると $D-Latch$ は再度取込み状態へ戻る．この様な一連の動作により左のユニットから来たデータが取り込まれ，次に右のユニットへと移されてゆく．

FIFO を様相論理モデルとして解釈した上で，各要素について仮定すべきプロパティを見てゆく．まず $ANDN$ については，左側の信号を ${}_L$，右側の信号を ${}_R$ で表すとプロパティが図 8 で規定される．（$ANDN$ のインバートされている方の入力から出力へのパスを ${}^*ANDN$，もう一方を ${}_*ANDN$ と表している．）$SR-FF$ については図 9 の通りである．ここで，一番

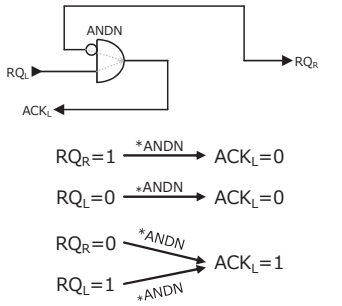


図 8: ANDN のプロパティ

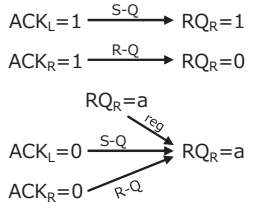
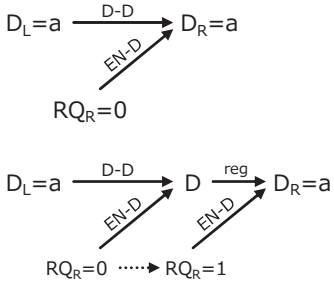


図 9: SR-FF のプロパティ

下は値を保持する場合を示しており，reg は $SR-FF$ 内に持つフィードバック・ループを表している．この reg の遅延が値の保持についての本質的な役割を果たしている．reg の遅延はその他の遅延と比べて極めて小さいと仮定する．

$D-Latch$ については若干複雑になり，図 10 のようになる．図の下のプロパティは値をラッチする瞬間を表している．

ここまで個別の要素について見てきたが，最後に FIFO の到達可能性関係の全体像を図 11 に表す．図は 3 つの連続したユニットを対象としており，L/R で表す代わりに点線でユニットの区切りを表している（以降の同様の図についてもこの方式に従う）．D 及び RQ



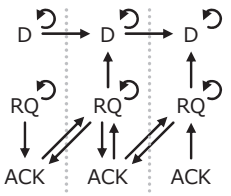


図 11: FIFO の到達可能性関係

図 10: D-Latch のプロパティ

が有している自己への矢印は reg である．これらの関係の中で前述のプロパティを使いながら推論を進めることが検証行為となる．

3.2. 必要となるタイミング制約

ここでは FIFO が正しく動作するために必要となるタイミング制約を示す．これらは我々が次節で示す検証を完成させる過程で経験的に確定させていったものである．すなわち，成立すべき命題が手持ちの仮定から証明できそうにない場合にタイミング制約の必要性を検討する，といった作業の繰り返しを経て定められた．本稿で提案する検証手法は合成前の上流工程で行うことを想定しているため，タイミング制約については具体的な遅延値は扱わず相対的な制約のみが対象である．

- タイミング制約 A

  図 12 において，RQ の分岐点から上がって最後に reg を通り $D-Latch$ の出力まで（点線）の遅延が，分岐点から右に伸びて同じく $D-Latch$ の出力まで（破線）の遅延よりも小さい．これは前の

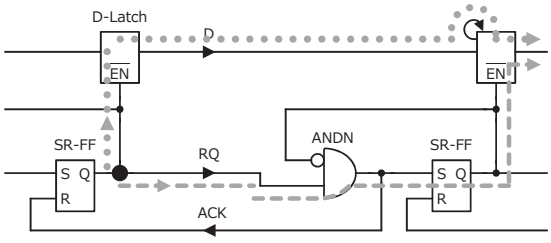


図 12: タイミング制約 A

（左の）ユニットが有効なデータを出し始めた後に $D-Latch$ がデータをラッチすべきという意図である．

- タイミング制約 B

  図 13 において，ACK の分岐点から右に行って $D-Latch$ の出力まで（点線）の遅延が，分岐点から下に行って最後に reg を通り同じく $D-Latch$ の出力まで（破線）の遅延よりも小さい．これは前の

Copyright © 2015 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

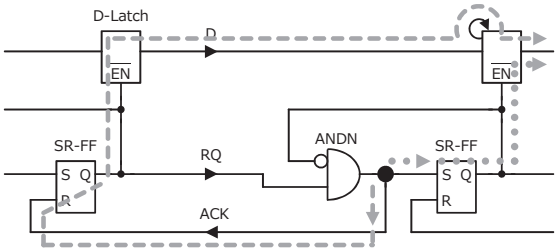
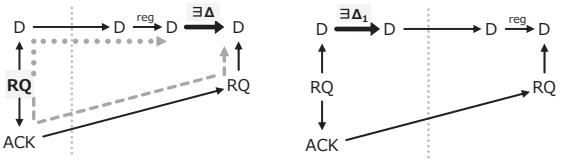


図 13: タイミング制約 B

（左の）ユニットがデータの保持を終えて $D-Latch$ を開放し始める前に $D-Latch$ がデータをラッチすべきという意図である．

タイミング制約 A を到達可能性関係上に示すと図 14(a) のようになる．RQ から始まった上側の経路は下

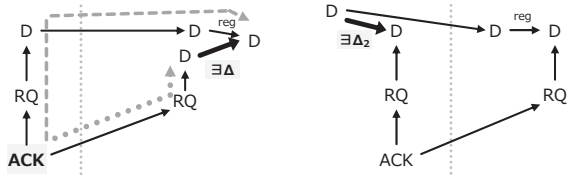


(a) (b)

図 14: 到達可能性関係上のタイミング制約 A

側の経路より短いため，ある Δ が存在して，それを加えると合流点が一致することになる．Δ の存在は合流点の直前でなくても言える．図 14(b) は Δ を手前側の D の後に求めた場合である．後のためにこれを Δ_1 とする．

同様にタイミング制約 B を到達可能性関係上に示すと図 15(a) のようになる．ACK から始まった下側の経



(a) (b)

図 15: 到達可能性関係上のタイミング制約 B

路は上側の経路より短いため，ある Δ が存在して，それを加えると合流点が一致することになる．タイミング制約 A の場合と同様に Δ の存在は合流点の直前でなくても言えるため，図 14(b) のような Δ_2 を取ることができる．次の項で行う検証ではタイミング制約の帰結である Δ_1，Δ_2 を使って FIFO の動作を証明する．

3.3. 検証

本稿で検証する FIFO のプロパティは次である．“回路がステーブルな状態からデータ及び $RQ=1$ が与えられると，然るべき後にそのデータが次のユニットの $D-Latch$ に保持される．”ステーブルな状態をより具体的に表すと全ての $RQ=0$，$ACK=0$ となる．このとき，$D-Latch$ は透過状態である．図 16 は証明の概要を表している．まず“データ及び $RQ=1$ が与えられ

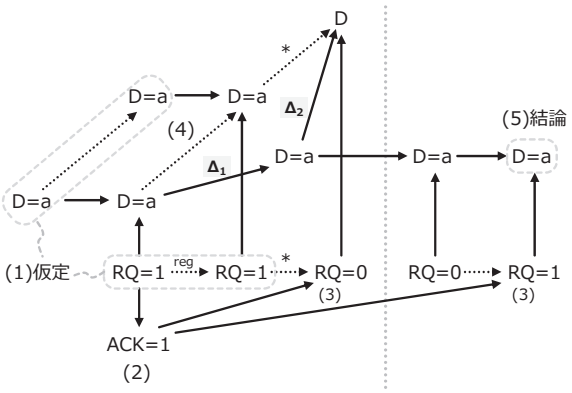


図 16: 動作プロパティの証明

ると”の仮定は (1) に示される $D=a$ 及び $RQ=1$ で表わされている．$RQ=1$ を受けて (2)$ACK=1$ となり，その後 (3)$RQ=0$ に戻り次のユニットの $RQ=1$ となる．一方，仮定の RQ の上の経路では (4)$D=a$ がしばらくの間保たれることが分かる．アスタリスク (*) で示した箇所は前のユニットが持つ条件が必要となるため省略しているが，成立する．前の項で示した Δ_1，Δ_2 をここに適用することによりラッチされるデータは与えられたもの，すなわち (5)$D=a$ が結論として成り立つ．対象の FIFO がタイミング制約 A,B を満たしていれば求めるプロパティが成立することが示された．

4. 定理証明言語 Agda 上への実装

前節では非同期 FIFO の検証結果を示したが，その証明は日本語で示したのみであった．人手による証明ではミスの可能性を否めないため，提案の検証手法について計算機への実装の実現性を確認すべく，定理証明言語 Agda[6] への実装を試みた．Agda は定理証明言語であり，かつ，依存型をもつ関数型プログラミング言語でもある．

4.1. 定義の実装

回路記述と到達可能性関係については [7] を参考に Arrow の概念をベースとして実装した．Arrow は関数型プログラミング言語 Haskell のクラスであり，抽象的な計算を表現する枠組である．図 17 に示されるような演算子を用い，小さな単位の計算からより大きな単位の計算を構築する．Verilog 等多くの HDL が“信号

Copyright © 2015 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

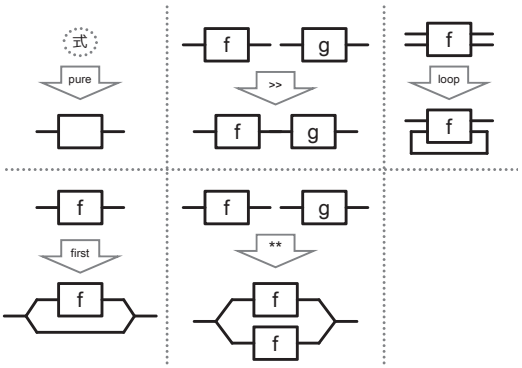


図 17: Arrow の代表的な演算

をどう処理するか”を記述するのに対して，Arrow を用いた回路記述では“要素をどう構成するか”を記述することとなる．

Arrow を用いた回路記述の定義を示す．

```
data Arrow where
  pure : (A → B) → Arrow A B
  _>>_ : Arrow A B → Arrow B C
    → Arrow A C
  _**_ : Arrow A0 B0 → Arrow A1 B1
    → Arrow (A0 × A1) (B0 × B1)
  first : Arrow A0 B0
    → Arrow (A0 × A1) (B0 × B1)
  loop : Arrow (A × C) (B × C)
    → Arrow A B
  fork : Arrow A (A × A)
```

fork 以外は一般の Arrow に従う．fork は二股に分かれる配線を表しており，すなわち論理的には pure(λ a → (a, a)) に等しいが，タイミング制約を定めるために必要となる．

次に，クリプキ構造の経路を定める．Arrow に以下を追加したものを ExArrow とする．

```
  id : ExArrow A A
  L| : ExArrow (A0 × A1) B → ExArrow A0 B
  R| : ExArrow (A0 × A1) B → ExArrow A0 B
  |L : ExArrow A (B0 × B1) → ExArrow A B0
  |R : ExArrow A (B0 × B1) → ExArrow A B0
  -  : ExArrow A B → ExArrow B A
  LR : ExArrow A B
  RL : ExArrow A B
```

id は自身への経路である．f : Arrow (A0 × A1) B としたとき，L|f は f を入力側の（A0 側）の引数のみに制限した経路であり，A0 から B への経路である．-（マイナス）は入力側から出力側への経路である．LR は A × B における A から B への経路として使われる．

加えて，ExArrow についての等式を導入する．Agda のコードでは“{}”内に表わされるのは暗黙の引数である．

```
data _≡_ : {A B : Set} →
 ExArrow A B → ExArrow A B → Set1 where
  refl : {A B : Set}{f : ExArrow A B} →
     f ≡ f
  symm : {A B : Set}{f g : ExArrow A B} →
    f ≡ g → g ≡ f
  trans : {A B : Set}{f g h : ExArrow A B} →
    f ≡ g → g ≡ h → f ≡ h
  assoc : {A B C D : Set}{f : ExArrow A B}
    {g : ExArrow B C}{h : ExArrow C D} →
     (f >> g) >> h ≡ f >> (g >> h)
   :
```

等式≡を，二項関係の反射律（refl）・対称律（symm）・推移律（trans）を満たすものとして定義している．assoc は>>が結合律を満たすことを表している．その他，ExArrow の演算に関する等式も必要となる．

```
   :
  +- : {A B : Set}{f : ExArrow A B} →
     f >> - f ≡ Id
  -+ : {A B : Set}{f : ExArrow A B} →
     - f >> f ≡ Id
  -#- : {A B : Set}{f : ExArrow A B} →
    - (- f) ≡ f
   :
```

などである．

あるクリプキ経路 r : ExArrow が検証対象の回路中の経路かどうかを判定できなければならない．そのために，包含関係 SubArrow を次のように定義する．

```
data SubArrow : {S' B' : Set}{A  B  : Set} →
 ExArrow S' B' → ExArrow A  B  → Set1 where
  refl : {A B : Set}{f : ExArrow A B} →
     SubArrow f f
   :
  trans :
   {A B C D E F : Set} →
   {f : ExArrow A B} →
   {g : ExArrow C D} →
   {h : ExArrow E F} →
    SubArrow f g →
    SubArrow g h →
     SubArrow f h
   :
  L|# :
   {A0 A1 B : Set} →
   {f  : ExArrow (A0 ×  A1) B} →
     SubArrow (L| f) f
   :
```

refl は，等しいものは包含関係にあると定義している．trans により包含関係は推移的であるとし，最後に，L|f は f に包含されるとしている．続いて，正の経路の定義を行う．

```
data Positive : {A B : Set} →
    ExArrow A B → Set1 where
   :
  fork△|R : {A : Set} → (f : ExArrow A A) →
     Positive (fork |R >> f >> - (fork |L))
  fork△|L : {A : Set} → (f : ExArrow A A) →
     Positive (fork |L >> f >> - (fork |R))
  fork<△  : {A : Set} → (f : ExArrow A A) →
     Positive (fork< |R >> f >> - (fork< |L))
   :
```

fork については fork|R から及び fork|L からの経路のどちらもが正であるのに対し，fork<については fork<|R からの経路のみを正とする．

最後に，命題を表す PROP を定義する．

```
\beginCode \begin{verbatim}
data PROP : {A B : Set} →
```


Copyright © 2015 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.

```
Time → ExArrow A B → B → Set1 where
  :
 pure : {t : Time}{A B C : Set} →
  {f : ExArrow A B}{g : B → C}{b : B} →
   PROP t f b → PROP t (f >> pure g) (g b)
  :
 LR : {t : Time}{A B0 B1 : Set} →
  {f  : ExArrow A (B0 × B1)} →
  {f' : ExArrow A B0} →
  {b0 : B0}{b1 : B1} →
   SubArrow f' f →
   PROP t f' b0 → PROP t (f' >> LR) b1 →
    PROP t f (b0 , b1)
  :
```

PROP は時刻，クリプキ経路，信号値を引数に取り集合を返す．t : Time，r : ExArrow A B，v : B に対し，PROP t r v が要素を持つことは，時刻 t において経路 r を通った後の点について信号値 v が成り立っていることを表す．pure は時刻 t において経路 f を通った後の点について b であれば，さらに pure g を通った後は g b となることを表している．LR はその引数に従い，f' と f' >> LR により B0 × B1 の値を構成可能であることが分かる．

4.2.INVAND の検証

図 4 に挙げた回路を INVAND 回路と呼び，c を通るパスの方が遅いと仮定すると $d = 0$ のままであることを証明する．

INVAND 回路に入力波形を与えることは以下で表現される．

```
record Hypo : Set1 where
 field
  thH : {f : ExArrow Sig Sig} →
    Positive f → PROP t (- f) H
  th↓ : PROP t Id ↓
  thL : {f : ExArrow Sig Sig} →
    Positive f → PROP t f L
```

thH は時刻 t 以前が H レベルであることを示している．これを仮定することにより，以下の二つの結論が得られた．（Agda 処理系の言葉を使うと証明項が書け，型チェックを通った．）

```
Tht : PROP t INVAND L
```

すなわち，時刻 t において経路 INVAND を通った後の点は L である．また，

```
Th-Δ : PROP t (- Path Δ >> INVAND) L
```

すなわち，時刻 t において経路- Path Δ >> INVAND を通った後の点は L である．（Path Δ は図 4 の説明における Δ に相当する経路である．）このように，提案した検証方法を定理証明言語に実装することができた．

4.3. 実装に関する課題

タイミング制約の記述に fork<を用いたが，この方法は比較的単純なパスを持つ場合にしか適用できないと思われる．すなわち，fork<で分岐した先にまた別の分岐がある場合などに対応できていない．検証例に挙げた非同期 FIFO も対応できないケースとなっており，実用に向けてはクリアすべき課題である．また，Agda による実装については，信号世界において到達し得る点について全て述べられていること（関数の全域性）の判定が実現できておらず，可能かどうかも含めて課題である．

5. まとめと今後の課題

本稿では，遅延依存回路の形式検証手法及びそれを用いた非同期 FIFO の検証を示した．これにより，遅延依存回路について，そのタイミング制約を含めて網羅的な検証が可能となった．

今後の課題としては，まず提案した検証手法の妥当性について評価を行う必要がある．誤った結果が導かれる可能性（健全性），あるいは条件の下で導かれるべき結果を必ず得ることができるのか（完全性）について，理論面と実際の評価の双方から確認すべきである．また，相対的なタイミング制約の記述方法については回路が単純な場合についてしか実現できておらず，今後の課題となっている．

参考文献

[1] C. J. Myers, *Asynchronous circuit design.* John Wiley & Sons, 2004.

[2] J. A. Brzozowski and J. Ebergen, “On the delay-sensitivity of gate networks,” *Computers, IEEE Transactions on*, vol. 41, no. 11, pp. 1349–1360, Nov 1992.

[3] 萩谷昌己 and 西崎真也, **論理と計算のしくみ**. 東京: 岩波書店, 2007, ch. 3.

[4] J. A. Brzozowski and C.-J. H. Seger, *Asynchronous circuits.* Springer, 1995.

[5] C. Molnar, I. Jones, W. Coates, and J. Lexau, “A fifo ring performance experiment,” in *Advanced Research in Asynchronous Circuits and Systems, 1997. Proceedings., Third International Symposium on*, Apr 1997, pp. 279–289.

[6] U. Norell, “Dependently typed programming in agda,” in *Advanced Functional Programming.* Springer, 2009, pp. 230–266.

[7] 西村俊二, 尼崎太樹, and 末吉敏則, “タイミング制約を含んだ回路記述方式とその意味論,” in **電子情報通信学会技術研究報告** *VLD2014-82*, vol. 114, no. 328. 電子情報通信学会 VLD 研究会, Nov 2014, pp. 81–86.

Copyright © 2015 by Information Processing Society of Japan and
The Institute of Electronics, Information and Communication Engineers
All rights reserved.