5P-2

Prolog を用いた ハードウェア同期部検証の効率化

西山　智、　藤田　昌宏、　田中　英彦、　元岡　達
（東京大学　工学部）

1.　はじめに

我々は、時相論理（Temporal Logic）を用いてハードウェア同期部の仕様を記述し、処理系にProlog を使用してゲート回路やＤＤＬによる設計の検証を行なうことについて研究してきた。［１、２］仕様は命題論理の範囲で記述し、設計に対し必要な全ての場合を調べることによって検証している。本稿では、次に示すような検証手法の効率化について報告する。

① 検証に必要な部分の回路のみ検証プログラムが処理するように回路を絞り込む（回路の絞り込み）

② 検証プログラムを工夫し、一度調べた状態を記憶しておき、以降同じ状態がでてきても処理しないようにする（状態の記憶）

③ 外部回路に対する仕様を利用し、検証対象回路の各端子の値の変化を押さえる（外部条件の利用）

以下、これらの手法について、ゲート回路を例にとって説明していく。なお使用するProlog は、Ｃ－Prolog ［３］である。

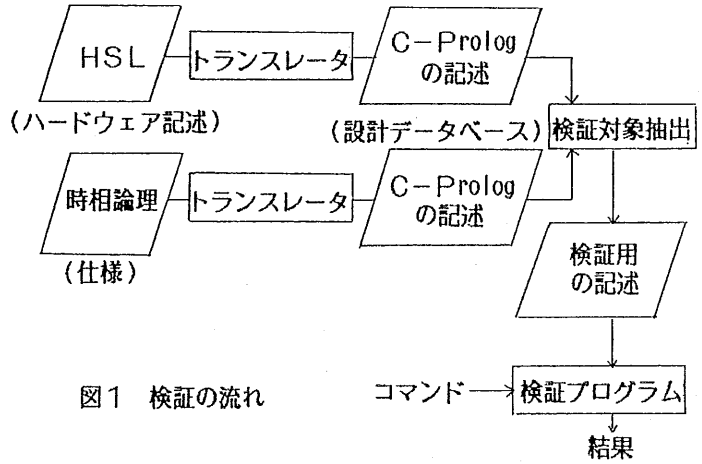

図１　検証の流れ

2.　同期部の検証

ゲート回路に対する検証は図１に示す流れに従って行なわれる。ゲート回路の記述としてＨＳＬ［４］を用い、Ｃで記述されたトランスレータによってＣ－Prolog の記述（設計データベース）に変換される。例として文献［１］のReceiver （図２）をＨＳＬで記述し、Ｃ－Prolog の記述に変換したものを図３に示す。この段階では、ＨＳＬの階層的記述や配列の記述がそのまま残っている。

次に時相論理で与えられる仕様を元に、設計データベースから検証に必要な部分のみを抽出し、回路の絞り込みを行ない、検証プログラムの要求する形に変換する。

検証手法の詳細については、文献［１、２］を参照されたい。

3.　回路の絞り込み

回路全体の記述から、仕様を元に検証に真に必要な部分のみを絞り込むことにより、検証の効率化が図れる。

回路の絞り込みアルゴリズムは基本的に次のようになる。

```
program 回路絞り込み
  begin
    仕様から検証に必要なすべての外部出力端子を抽出する；
    全ての外部出力端子について、
    procedure “端子絞り込み”を実行する
  end .

procedure 端子絞り込み（端子）
  begin
    端子を出力から入力へ逆方向にネットを溯る；
    if（外部入力端子にぬける）  then  return
    else if （一度通ったゲートにあたる）  then  return
    else begin  {始めてのゲートにあたったので}
              そのゲートを記憶する；
              そのゲートの全ての入力端子について、
              このprocedure をrecurcive に呼ぶ；
         end
  end ；
```

実際のプログラムでは、階層的記述や配列の記述に対応するため、もう少し複雑になっている。

例えば、図２のReceiver について、仕様：

□（Call →▽Hear ）［１］　・・・①

を検証するために、出力端子Hear を元に回路を絞り込むと図２の点線内の部分が抽出される。

4. 状態の記憶

検証は背理法で行うため、時相論理で与えられる仕様の否定を各時刻毎に展開して状態遷移図を作り、検証対象回路がこれを満さないことを調べることにより行う。この時、同じ状態が現れることがあるため、同じ処理を何度も繰り返すことが多い。従って、一度計算した状態を全てAssert して記憶しておき、以前に現れていない状態ができた場合のみ処理することにより効率化が図れる。文献［５］に示すようにこれはある程度大きな回路に対しては非常に有効である。

5. 外部条件の利用

検証対象回路はそれ自身のみで動作することはなく、外部に何らかの回路がある場合が多い。この時、回路自身では解らないが、外部回路との接続によって入力の条件が定められる場合がある。これらを時相論理によって記述し、これを元に回路の各端子の値の変化を押えることにより、検証の効率化を図る。

6. 検証例

ここでは、Receiver の回路に対して仕様①の検証を検証例として示す。演算部のデータ幅に対して検証時間は表１のようになった。（数字はＶＡＸ１１／７３０　Ｃ－Ｐrolog でのＣＰＵ時間（sec ）である。）　この例では、５の効率化は行っていない。

7. おわりに

文献［５］でも示してあるように、ここで述べた３つの効率化の手法によりかなり大きな回路でも検証時間を短く押えることができる。今後は検証システムとしてまとめあげると共に検証の効率化についてさらに検討していく予定である。

8. 参考文献

［１］第26情報処理全国大会、３Ｋ－１、３Ｋ－２
［２］第27情報処理全国大会、３Ｌ－１１
［３］Ｃ－Prolog User's Manual Version1.2a
［４］階層仕様記述言語ＨＳＬ　Manual （ＮＴＴ）
［５］通信学会電子計算機研究会資料ＥＣ 83-29

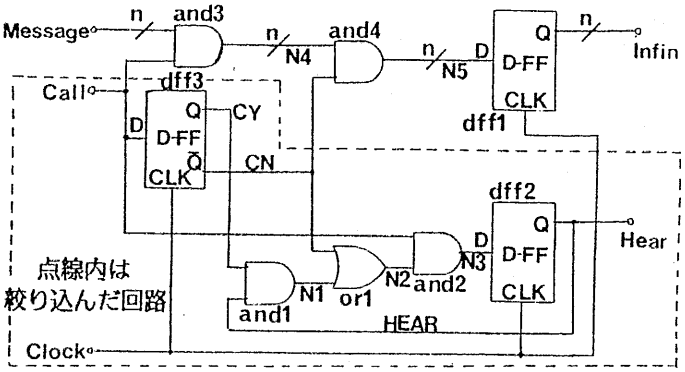

図２　ハンドシェイク・モジュール　Receiver［１］

	状態の記憶あり	状態の記憶なし
絞り込んだ回路	0．93	0．87
演算部データ幅1bit	4．28	9．88
〃　2bit	28．25	204．80
〃　3bit	253．30	5645．80

表１　□（Call →▽Hear ）の検証に要するCPU時間
（単位：秒、VAX11／730上のC－Prolog による）

```
ident('HANDSHAKE').
version('1.0').
date('83/12/11').
project('VERIFIER').
extern_name('RECEIVER',['MESSAGE',[0,7]]).
extern_name('RECEIVER',['CALL']).
extern_name('RECEIVER',['CLOCK']).
extern_name('RECEIVER',['HEAR']).
extern_name('RECEIVER',['INFIN',[0,7]]).
gate_interface('RECEIVER',[['MESSAGE',[0,7]],['CALL'],['CLOCK']],[['HEAR'],['INFIN',[0,7]]]).
type_def('RECEIVER',[   [['AND1'],'AND2'],
                        [['AND2'],'AND2'],
                        [['AND3',[0,7]],'AND2'],
                        [['AND4',[0,7]],'AND2'],
                        [['OR1'],'OR2'],
                        [['DFF1',[0,7]],'DEPE'],
                        [['DFF2'],'DEPE'],
                        [['DFF3'],'DEPE']]).
all_net_def('RECEIVER',[['MESSAGE',[0,7]],['MESSAGE',[0,7]],[[['AND3',[0,7]],['1']]]]).
all_net_def('RECEIVER',[['CALL'],['CALL'],[[['AND3',[0,7]],['2']],[['AND2'],['1']],[['DFF3'],['D']]]]).
all_net_def('RECEIVER',[['CLOCK'],['CLOCK'],[[['DFF1',[0,7]],['CLK']],[['DFF2'],['CLK']],[['DFF3'],['CLK']]]]).
all_net_def('RECEIVER',[['HEAR'],[['DFF2'],['Q']],[['HEAR'],[['AND1'],['2']]]]).
all_net_def('RECEIVER',[['INFIN',[0,7]],[['DFF1',[0,7]],['Q']],[['INFIN',[0,7]]]]).
all_net_def('RECEIVER',[['CY'],[['DFF3'],['Q']],[[['AND1'],['1']]]]).
all_net_def('RECEIVER',[['CN'],[['DFF3'],['QB']],[[['OR1'],['1']],[['AND4',[0,7]],['2']]]]).
all_net_def('RECEIVER',[['N1'],[['AND1'],['3']],[[['OR1'],['2']]]]).
all_net_def('RECEIVER',[['N2'],[['OR1'],['3']],[[['AND2'],['2']]]]).
all_net_def('RECEIVER',[['N3'],[['AND2'],['3']],[[['DFF2'],['D']]]]).
all_net_def('RECEIVER',[['N4',[0,7]],[['AND3',[0,7]],['3']],[[['AND4',[0,7]],['1']]]]).
all_net_def('RECEIVER',[['N5',[0,7]],[['AND4',[0,7]],['3']],[[['DFF1',[0,7]],['D']]]]).
```

図３　C－Prolog の記述例（Receiver ）