PGA上のSATソルバPCMGTPの改良について

藤田, 博
九州大学大学院システム情報科学研究院知能システム学部門

長谷川, 隆三
九州大学大学院システム情報科学研究院知能システム学部門

越村, 三幸
九州大学大学院システム情報科学研究院知能システム学部門

木之下, 昇平
九州大学大学院システム情報科学府知能システム学専攻 : 修士課程

他

https://doi.org/10.15017/1516056

出版情報：九州大学大学院システム情報科学紀要. 10 (1), pp.21-26, 2005-03-25. Faculty of Information Science and Electrical Engineering, Kyushu University
バージョン：
権利関係：

九州大学大学院
システム情報科学紀要
第10巻 第1号 平成17年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.10, No.1, March 2005

FPGA上のSATソルバPCMGTPの改良について

藤田 博* ・ 長谷川 隆三* ・ 越村 三幸* ・ 木之下 昇平** ・ 松田 純一**

On Improvements of a SAT-Solver PCMGTP on FPGA

Hiroshi FUJITA, Ryuzo HASEGAWA, Miyuki KOSHIMURA,
Shohei KINOSHITA and Jun'ichi MATSUDA

(Received December 24, 2004)

Abstract: In this paper, an improved design of a SAT-solver PCMGTP on FPGA is described. The previous implementation of PCMGTP achieved considerable speedup of SAT-solving compared to the software counterpart of MGTP. After intensive analyses and experiments, it turned out that the early design contains much redundancy and has room for improvement. Also, we developed a generic description style in Verilog using arrays and iterative constructs. Experimental results show that the new implementation outperforms the old one with regard to both execution time and circuit size.

Keywords: SAT solvers, FPGA, Verilog, Model generation, Theorem proving, Reconfigurable computing

1. まえがき

命題論理式の充足性判定器，いわゆるSATソルバの分野で，近年Zchaff[1]などのいくつかの優秀なソフトウェアがさらなる性能向上を目指し，国際競技の場[2]で上位入賞を果たすべく鎬を削っている．そのうち完全性を保障するSATソルバのほとんどは，基本的にDavis-Putnam (DP)アルゴリズム[3]に基いており，それぞれ独自のヒューリスティクスで差別化を図っている．

ハードウェア版SATソルバに関しては，ソフトウェアと対等に競争できるものは未だ現れていないが，我々が2003年に行なったFPGA上の実装経験から推量すれば，近い将来，優位に立てる可能性を有していると考えられる．我々のSATソルバは，一階述語論理の定理証明系MGTP[4]を命題論理に限定したPCMGTP[5]に基づくもので，問題ごとに回路を生成し，1チップのFPGA上にダウンロードして実行させる．

このPCMGTPの設計を詳細に検討したところ，データ表現や推論エンジンの状態数等に冗長性が含まれており，それらを除去／修正すれば回路規模，ならびに実行速度について改善が見込めることが明らかとなってきた．

今回実施した主な変更／改良は以下のとおりである．

- 付値準位を整数で表現
- 推論エンジンの状態数の削減
- BCBE回路による単位含意の実装
- 付値候補リテラル選択部の改良
- トーナメント回路の多クロック化

なおこの間，CADシステムの更新が行われ，回路合成時間は概ね半分ほどに短縮され，Verilogの配列やfor文による記述の回路合成可能性に関わる制限が緩和されるなど，開発環境についても大きな改善があった．

また，設計の機能検証のために既存の開発ツールに依らないシミュレータをJavaを用いて別途開発した．このシミュレータは，設計回路が実行に要する総クロック数について実機と厳密に一致する結果を与えることが最重要の要件である．そのため，当初はできる限りVerilog記述と1対1に対応がつくような記述を試みたが，ハードウェア並列性を活用する箇所については逐次化が避けられず，結果として実行時間が著しく増加してしまい実用に供さぬものとなったため，SATソルバの機能特性に依拠して適用可能ないくつかの最適化を行った．たとえば，PCMGTPの推論の遂行にしたがって値が変化するレジスタの個数は，各クロック周期内ではほぼ常に極めて少数に限られるので，値の変化しない大多数のレジスタに対する更新動作が省略できる．このシミュレータにより，回路設計の開発効率が格段に向上した．

本稿では，SATソルバとして改善されたPCMGTPのFPGA実装とその評価について述べる．

2. PCMGTP

まず，MGTP手続き固有の用語／概念を導入しつつ標準的な命題論理の積和標準形に関する定義を与えた後，SATソルバとしてのPCMGTPの基本方式を示す．

命題変数pを正リテラル，命題変数の否定$\neg p$を負リテラルという．単にリテラルというとき，それは正リテラルか負リテラルのいずれかである．リテラルの集合を節といい，正リテラルを2個以上含む節を非ホーン節とい

平成16年12月24日受付

* 知能システム学部門
** 知能システム学専攻修士課程

う．所与の問題は節の集合として表現される．

命題変数やリテラル（単数／複数を混用，文脈で区別する）に対する真偽値の割当てを付値という．リテラルの付値が定まれば，対応する命題変数の付値は必然的に定まり，その逆も成立つ．付値されていないリテラル（命題変数）を未定リテラル（未定変数）という．

節中の少なくとも1個のリテラルが真値を付値されているとき，節の評価値が真であるといい，節中のすべてのリテラルが偽値を付値されているとき，節の評価値が偽であるという．（すなわち節はリテラルの論理和として解釈される．）評価値が真でも偽でもない節を未定節という．未定節は1個以上の未定リテラルを含む．未定リテラルが1個のみの未定節を単位未定節といい，その未定リテラルを単位含意リテラルという．2個以上の未定リテラルがすべて正リテラルの未定節を違反節[†1]という．

ある変数集合に関する付値のもとで単位未定節と違反節のいずれも存在しないとき，所与の問題は充足可能であり，そのときの付値を充足解[†2]という．（すなわち節集合は節の論理積として解釈される．）いかなる付値も充足解とならないとき，所与の問題は充足不可能である．一般に，充足可能性は未定変数を残しながら判定されることに注意．

未定変数が残っているために節集合の充足性が未だ判定できないような付値を解候補といい，これに未定変数の付値を追加することを解候補の拡張という．

以下にPCMGTPのアルゴリズムを示す．

(1) 含意に基づく付値

- 単位含意：単位未定節が存在するとき，その評価値を真とするような単位含意リテラル（したがって対応する変数）の付値が一意に定まる．
- 同時並行単位含意：単位含意が適用可能な単位未定節が複数存在するとき，そのすべてに関して同時並行的に単位含意を行なう．その結果，同一の命題変数に対して矛盾する付値が導かれる（単位含意の衝突）場合，現解候補を棄却して(3)に飛ぶ．
- 単位含意の連鎖：ある単位含意の結果，新たな単位未定節が生じるとき，引き続きその新たな単位未定節に関して単位含意を行なう．
- 単位含意の閉包：単位含意の衝突が生じない限り，同時並行単位含意の連鎖を繰り返し適用する．単位含意可能な単位未定節が存在しなくなったとき，違反節が存在するときは(2)に進む．さもなければ問題は充足可能で，現解候補を充足解としてカウントした後，棄却して(3)に（単解探索の場合は(4)に）飛ぶ．

(2) 違反節の選択，投機的付値，解候補の拡張

違反節を1本選び，その中の未定正リテラルを1個選んでそれに“投機的”に真値を付値し，解候補を拡張して(1)に戻る．

(3) リテラルの再選択

最近(2)で選択された違反節に未選択の未定正リテラルが存在するとき，それに真値を投機的に付値し，解候補を拡張して(1)に戻る．さもなければ，その違反節に基づく解候補の拡張の場合分けは尽きているので，以前の違反節に後戻りした上，リテラルの再選択を試みる．後戻りすべき違反節が存在しないとき，探索を終了し，(4)に進む．

(4) 充足性判定

充足解が1個以上求まっている場合はSAT，さもなければUNSATと判定する．

上の(2),(3)において，違反節に基づく未定変数の選択の方式を別の基準に適宜置き換えることにより，本アルゴリズムを標準的なDP法と同等にすることができる．

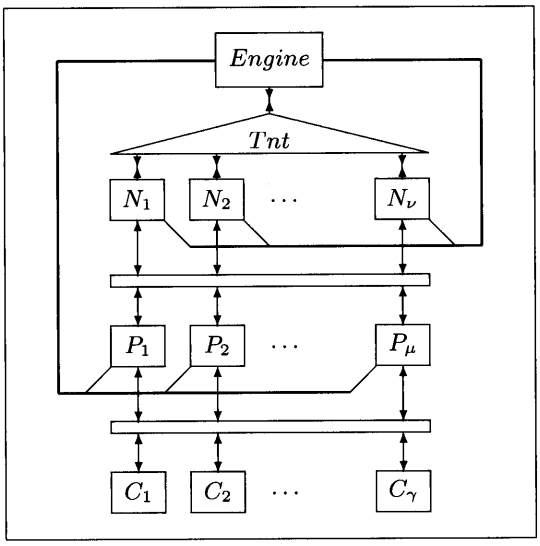

Fig. 1　Block diagram of the improved PCMGTP.

3. FPGA上のPCMGTP

Fig. 1に改良版PCMGTPのFPGA上のモジュール構成を示す．*Engine*，非ホーン節 N_j $(0 \le j \le \nu)$，および命題変数 P_i $(0 \le i \le \mu)$ の各モジュールは順序回路である．トーナメント Tnt と節 C_k $(0 \le k \le \gamma)$ の両モジュールは組合せ回路である．ただし，命題変数の総数をμ，節の総数をγ，非ホーン節の総数をν とする．

以下，各モジュールの詳細について述べるが，説明に対応するVerilogコード断片を本文中に枠で囲って示す．assign文により回路の任意の点に適宜信号線名を付ける

†1　モデル生成法の用語．含意式で前件が充足，後件が非充足のものをいう．

†2　MGTPでは充足性（SATかUNSATか）だけでなく，充足解の内容と個数に興味がある．

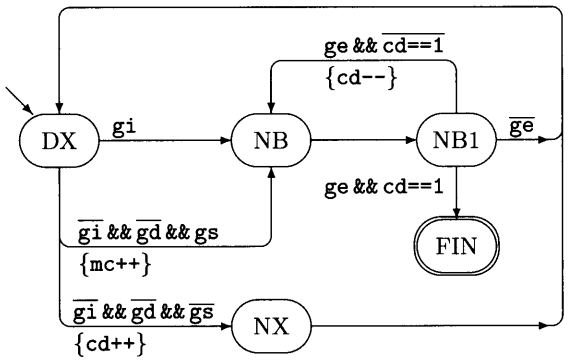

Fig. 2 State transition diagram for new PCMGTP.

が，その右辺は一般に定数，信号線名，レジスタ（の出力）名，を基本項とする論理式，算術式，あるいはfunction呼出しであり，組合せ回路の合成対象となる．一方，順序回路の動作はシステムクロックを遷移契機とするalways構文内に，レジスタに対するノンブロッキング代入(<=)を用いて記述される．

3.1 推論エンジンモジュール

Fig. 2 に推論エンジンの状態遷移図を示す．長円は状態を表し，DXが始状態，FINが終状態である．矢は遷移を表し，矢の上か左のラベル（gi等）は，遷移を引き起こす条件である．矢の下のラベル（{cd++} 等）は，状態遷移と同時に行われるレジスタの更新を示す．複数クロック間留まる可能性のある状態はDXとFINのみであり，他の状態は必ず1クロック後に別の状態へ遷移する．

giは現解候補棄却，gsは充足，gdは“単位含意の閉包の未了”をそれぞれ表す信号であり，cdは選択準位[†3]を表す整数で，cd==1は“トップレベルにあること” を表す．

各状態での動作を以下に述べる．

- DX【アルゴリズム(1)】: 1クロック内に同時並行単位含意が行われる．その結果，単位含意の衝突が生じた(gi)ならば，NBに遷移．さもなければ，
  - 新たな未定含意節が生じている(gd)ならば，DXに留まる（単位含意の連鎖）．
  - さもなければ単位含意の閉包計算が完了しているので，充足している(gs)ならば，充足解個数を1増やして(mc++)，NBに遷移（単解でよい場合はFINに遷移）．さもなければ，選択準位を1増やして(cd++)，NXに遷移．
- NX【アルゴリズム(2)】: 選ばれた違反節中の未定正リテラルを1個選び，付値を定めてDXに遷移．
- NB: NB1に遷移．
- NB1【アルゴリズム(3)】: 現選択準位で選ばれた違反節中に未選択の未定正リテラルがあれば($\overline{\text{ge}}$)，1個選んで付値を定め，DXに遷移．さもなければ，
  - トップレベルにある(cd==1)ならばFINに遷移．
  - さもなければ選択準位を1減じて(cd--)，NBに遷移．
- FIN【アルゴリズム(4)】: 充足解個数が0ならばUNSAT，さもなければSATを報告．

なお，選択準位の上限はμ, ν の小さい方$\lambda = min(\mu, \nu)$で抑えられる．

3.2 命題変数モジュール

命題変数1個につき，本モジュールのインスタンスが1個生成される．

命題変数pについて，正リテラルpと負リテラル$\neg p$に対する付値の要請が独立に外部で定められ，それぞれpr, nrとして本モジュールに入力される．pr(nr) が1のとき正（負）リテラルに対して真値が付値されるべきこと，0のとき付値が未定であるべきことを意味する．

pr, nrがともに1のときは付値の衝突，すなわち矛盾を意味するので，ただちにicを1として推論エンジンにこれを伝えなければならない．

```
assign ic = pr & nr;
```

pr, nrの値は本モジュール内のレジスタpa, naにそれぞれ記憶される．

```
pa <= ~cancel & pr & ~na;
na <= ~cancel & nr & ~pa;
```

本命題変数が未定変数であるときに付値の要請がある場合には，dfを1として単位含意の末了を推論エンジンに伝えなければならない．

```
assign df = ~(pa | na) & (pr | nr);
```

付値が行われるときの選択準位cdがレジスタavに記憶される．cancel は推論エンジンが状態NBにあり，かつ本命題変数に対する付値がキャンセル時の選択準位より深い準位で行われている場合に真となる．

```
assign cancel = nb & (av>=cd);
```

レジスタavは，付値のキャンセル時にリセットされ，付値が行われた時点でcdを記憶し，それ以降は値を保持する．

```
av <= ( cancel ? 0 : (df ? cd : av));
```

μ個の命題変数インスタンスからの出力線ic, dfを束ねて幅がμのベクタicv, dfvを定義し，推論エンジンへの信号gi, gdを次のように結線する．

```
assign gi = | icv;
assign gd = | dfv;
```

3.3 節モジュール（BCBE回路）

入力節1本につき，本モジュールのインスタンスが1個生成される．

このモジュールは単位含意のために特に考案された

†3 前論文では分岐深度，構築途中のMGTP証明木の高さ．

```
module bcbe(lit, unit);
   parameter N = 3;
   input [N-1:0] lit;
   output [N-1:0] unit;
   reg [N-1:0]    out;
   always @(lit)
     begin: block
        integer i;
        reg [N-1:0] l2r,r2l;
        l2r[0] = 1; r2l[N-1] = 1;
        for (i=0; i<N-1; i=i+1) begin
           l2r[i+1] = l2r[i] & lit[i];
           r2l[N-2-i] = r2l[N-1-i]
                        & lit[N-1-i];
           out[i] = l2r[i] & r2l[i];
        end
        out[N-1] = l2r[N-1] & r2l[N-1];
     end
   assign unit = out;
endmodule
```

Fig. 3　Verilog code for a BCBE circuit.

BCBE†4方式に基づき，**Fig. 3**に示すVerilog記述で与えられる．

節を構成するリテラル`lit`を横1列に並べ，リテラル間を左端から右向きに流れる信号線`l2r`と右端から左向きに流れる信号線`r2l`で結線する．

左端からの信号は偽のリテラルを通過して右に進み，真あるいは未定のリテラルのところで止まる．右端からの信号も同様．あるリテラルに左端からの信号と右端からの信号の両方が到達したとき，このリテラルが単位含意リテラルであることがわかる．こうして，出力ベクタ`unit`の単位含意リテラルに対応したビット位置に1をたてる．

3.4 非ホーン節モジュール

非ホーン節1本につき，本モジュールのインスタンスが1個生成される．

`nlits` (`plits`) をこの非ホーン節中の負（正）リテラルに対する付値のベクタとすると，`asat`は負リテラルのすべてが偽なること，`csat`は正リテラルのいずれかが真なること，`sat_nh`はこの節が充足していること，および`violated`は違反節であることを表し，次のように与えられる．

```
assign asat = & nlits;
assign csat = | plits;
assign sat_nh = ~asat | csat
assign violated = asat & ~csat;
```

違反節の状態となった非ホーン節は，トーナメント回路においてほかの違反節との対戦資格を有する．(違反節でないときは不戦敗となる．) 勝つのは**Fig. 4**に示す関数`disj_size`で計算される未定正リテラルの個数の少ない方である．

```
function [CBITS-1:0] disj_size;
  input [CWIDTH-1:0] nv;
  integer i,j;
  begin
    j=0;
    for (i=0; i<CWIDTH; i=i+1) begin
      if (nv[i]==0) begin j = j+1; end
    end
    disj_size = j;
  end
endfunction
```

Fig. 4　Verilog code for a "disj-size" circuit.

```
reg [CWIDTH-1:0] ncv, dor;
reg [CWIDTH:0] dol;
assign eb = dor[CWIDTH-1];
integer i;
always @*
  begin
    dol[0] = ena1;
    for (i=0; i<CWIDTH; i=i+1) begin
      ncv[i] = ~sel & cv[i] |
               dol[i] & ~nav[i];
      dor[i] = sel & cv[i] |
               dol[i] & nav[i];
      dol[i+1] = dor[i];
    end
  end
```

Fig. 5　Verilog code for a "next literal" circuit.

`CWIDTH`はこの非ホーン節中の正リテラルL_iの個数，`nv`は対応する負リテラル$\neg L_i$に対する付値であり，これが0であればこの命題変数L_iは偽値を付値されていない．違反節のときは真値に付値されてもいない．よって未定リテラルといえる．

Fig. 5は，この非ホーン節がトーナメントで優勝し，解候補拡張の主体となったとき，投機的付値の対象となる未定正リテラルを選択する回路の記述である．`eb`は，リテラルの再選択の末，未定正リテラルが尽きたときに1となる．

ν個の非ホーン節インスタンスからの出力線`sat_nh`, `eb`を束ねて幅がνのベクタ`stv`, `ebv`を定義し，推論エンジンへの信号`gs`, `ge`を次のように結線する．

```
assign gs = & stv;
assign ge = | ebv;
```

3.5 トーナメントモジュール

トーナメントモジュールは，違反節で未定正リテラルの個数が最小のものを決定する．**Fig. 6**にVerilogコードを示す．ベクタ`cnwrk`のはじめのνビットには非ホーン節

†4 Burning Candle at Both Ends の略．

からのviolated信号が結線される．この値が1のとき違反節であることを意味している．配列winのはじめのν語には非ホーン節の未定正リテラル個数の値が代入される．ベクタenwrkのはじめのνビットには非ホーン節に対戦結果を伝える信号が結線される．この値が1のときその非ホーン節の優勝が判る．

これらベクタでν以上の添字の信号線上には，そのときどきの対戦の勝者に関する情報が流れる．cnwrkとwinはセレクタを通して勝ち上がり方向へ併合収束し，enwrkは分配器を通して勝ち上がりの逆方向へ分岐拡散するような信号経路をそれぞれ形成する．こうして，セレクタで形成された2分木と分配器で形成された2分木を根の箇所で直列に繋いだような蝶ネクタイ形の回路が合成される．こうして，ν本の非ホーン節のエントリから優勝者決定までの時間はおよそ$log_2\nu$に比例する．

ところで，実際にはトーナメント優勝者は状態NXに進む直前の状態DXの段階で決定しており，状態NXにおいては，優勝した違反節が最初の未定正リテラルを速やかに選択決定することができる．

いずれにしても，νが大きくなるにつれ，トーナメント回路がクリティカルパスとなってしまう可能性が最も高い．その場合，トーナメントを勝ち上がり回数に関してできるだけ均等な数ステージに分割し，そのステージ数分のクロック数をかけて優勝者を決めるようにする．ただし，その場合，各ステージごとの勝者の記憶が必要になるので回路規模は増大する．実験によれば，非ホーン節数νが数100の場合は2ステージ分割が適切であった．

```
parameter TN = 2*NU+1;
reg [CBITS-1:0] win[TN:0];
reg [NN:0] cnwrk,enwrk;
integer i,j,k,mpp,mp,m;
always @*
  begin
    mpp = 0; m = NU; k = m-mpp;
    while (k>1) begin
      mp = m;
      for (i=0; i<k-1; i=i+2) begin
        j = i+mpp;
        if (cnwrk[j+1]==0 || cnwrk[j]==1
            && (win[j]<=win[j+1])) begin
          cnwrk[m] = cnwrk[j];
          win[m] = win[j];
          enwrk[j] = enwrk[m];
          enwrk[j+1] = 0;
        end else begin
          cnwrk[m] = cnwrk[j+1];
          win[m] = win[j+1];
          enwrk[j] = 0;
          enwrk[j+1] = enwrk[m];
        end
        m = m+1;
      end
      j = i+mpp;
      if (k%2==1) begin
        cnwrk[m] = cnwrk[j];
        win[m] = win[j];
        enwrk[j] = enwrk[m];
        m = m+1;
      end
      k = m-mp; mpp = mp;
    end
  enwrk[m-1] = 1;
end
```

Fig. 6　Verilog code for a tournament circuit.

4. 性 能 評 価

使用したFPGAはALTERA社のEP20K1500EBC652-3で論理セル数51840個，論理合成ツールはQuartus II Version4.0，ホストマシンはCPUが2.8GHz，メインメモリメモリが2GBである．問題はDIMACS Challenge benchmark problems[6] のAIMベンチマーク，N王妃パズル，および準群問題を用いた．比較用にソフトウェアMGTPを上記マシンで走行させ，同じ問題を解かせた．実験の結果を **Table 1** に示す．すべて正しい充足性判定結果（充足解個数M）が得られている．表中，( )内は旧実装による値，{ }内は新実装の旧実装に対する実行速度比である．

回路規模(Cells)については，命題変数の数(Vars)が50，節の数(Cls)が80〜200のaim系統の問題において前実装と比べて約5分の1に縮約できている．命題変数個数$\mu = 512$，節数$\gamma = 17430$で本リスト中最大規模のqg7-8では，新実装のチップ使用率が91%近くに達している．

証明実行時間について，HWではクロック数(Clocks)を回路ごとの許容最大周波数(FMAX)の値で除して見積った．新実装は旧実装に比べ，5倍ほど高速化されていることがわかる．旧実装の許容回路規模が小さいため，50変数200節を超える問題での比較はできていない．

ソフトウェアとの比較に関して，aim-100-2_0-no-1を除きHW, SWともに1秒に満たない易しすぎる問題であって，回路合成時間を考慮すればHWのメリットは全くないが，証明実行時間の高速化の目安として，概ね数100〜数1000倍というオーダーでHWがSWより高速に解く能力を有するということが言えよう．aim-100-2_0-no-1が86倍という値に留まっているが，証明のサイズが変わって見かけの効率が低下する場合もあるということである．

aim系統の問題(aim-$\star$)は乱数を用いて人工的に作られた問題で，冗長な節を多く含む傾向があり，補題生成機能を有するSATソルバがその機能を極めて有効に発揮して効率よく解くことが知られている．一方，N王妃パズルや準群問題(qg$\star$)は冗長な節が少なく，却って補題の効果を期待できないことが知られている．また，SATソルバの競技会では工業分野における実践的問題の部門があり，問題の特性についてaim系統とqg系統を両極端とするとき，概ね中間に位置するものと考えられている．ただし，

Table 1 Numerical results for some benchmark problems.

Problem	Vars	Cls	M	Cells	Comp. (sec)	FMAX (MHz)	Clocks	HW (μsec)	SW (msec)	Ratio
aim-50-1_6-yes1-1	50	80	1	2371 (9542)	275 (839)	33.03 (19.6)	294 (854)	8.9 (43.6)	15	1685 {4.9}
aim-50-2_0-yes1-1	50	100	1	2818 (14303)	255 (1474)	33.52 (17.89)	461 (1354)	13.8 (75.7)	15	1087 {5.5}
aim-50-3_4-yes1-1	50	170	1	3735 (22281)	344 (3565)	32.22 (15.5)	1579 (3153)	49.0 (203)	62	1265 {4.1}
aim-100-2_0-no-1	100	200	0	6052	613	28.89	7.3×10^7	2.4×10^6	2.1×10^5	86
8queen	64	6708	92	1999 (6708)	242 (557)	28.79 (13.72)	4138 (11258)	143.7 (820.6)	31	216 {5.7}
qg5-8	512	17430	1	34507	46732	13.16	175	13.3	63	1737
qg6-8	512	17374	2	32788	19268	14.67	138	9.4	63	6702
qg7-8	512	17430	0	34884	47017	13.75	1041	75.7	31	410

変数の個数と節数が数10万～数100万という極めて大規模なものになっている．

我々のFPGA版SATソルバは，

- 単位含意等でハードウェア並列性を引き出す
- 現状では補題生成機能を有しない
- 命題変数の個数が数100，節数が数万程度まで

という特徴ならびに制限をもつということが言える．したがって，qg系統のような本質的に困難だが適正サイズの問題に適用するのが最適と考えられる．

実際，以前の実装を用いた実験の一つとして，qg問題の原型ともいえるラテン方陣の作成問題について全解探索を試みたところ，ソフトウェアでは1日以上費やした計算をFPGAのSATソルバが回路合成時間を含めて3時間ほどで解いたという例がある．このようにハードウェアのメリットが充分に活かせる規模の問題が存在することは事実である．

5. むすび

本稿は，FPGA上のSATソルバPCMGTPの改良について述べた．主としてデータ表現の変更により，素子利用効率が10倍近く向上し，変数100個，節200本規模のAIMベンチマーク問題が1個のFPGA上に実装できるようになった．実行速度についても，クリティカルパス短縮化によりクロック周波数を約3倍上げることができたほか，推論エンジンにおける冗長な状態が削減でき，約5倍高速化された．

開発ツールの更新により，Verilogにおけるパラメタ化の機能やfor文と配列の活用によって，同型の回路インスタンスの生成と結線が従来よりも自由自在になり，BCBE回路，リテラル選択回路，トーナメント回路などのスケーラブルな記述が可能となった．実際に合成された回路についても，RTL表現レベルで概ね意図したとおりの最適なものであることが確認できた．

PCMGTPの推論機能の拡張に関しては，folding-up技法に基づく補題の自動生成・利用機構について設計を行い，シミュレーションにより有効性を確認している．ただし，回路規模の著しい増大を伴うことが予想されるため，実機への実装にあたっては何らかの制限を設ける修正が必要である．

従来は1チップ上の実装に限定していたが，今後は多チップ構成や外部メモリを利用したスケーラブルな回路構成，あるいはソフトウェアとの分担協調動作を基本とする設計について本格的な検討が必要な段階にあるものと考えている．

謝　辞

本研究の一部は，日本学術振興会科学研究費補助金・(C)(2)(課題番号：14580380)，萌芽研究(課題番号：14658094) の補助を受けた．

参考文献

1) L. Zhang, C. F. Madigan, M. W. Moskewicz and S. Malik, "Efficient conflict driven learning in a Boolean satisfiability solver," *International Conference on Computer-Aided Design,* 2001, pp.279–285.
2) "SAT Competitions," `http://satlive.org/SATCompetition/index.jsp`
3) M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of ACM,* Vol.7(3), 1960, pp.394–397.
4) 長谷川 隆三, 藤田 博, 越村 三幸, "分岐補題の抽出による極小モデル生成の効率化," **人工知能学会論文誌,** 第16巻, 第2号, 2001, pp.234–245.
5) 藤田 博, 河野　真史, 長谷川 隆三, "定理証明PCMGTPのFPGA上の実装," **九州大学大学院システム情報科学紀要,** 第9巻, 第1号, 2004, pp.13–18.
6) `ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/`.