社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2005－SLDM－118（11）
2005／1／25

# FPGA上に実装されたPCMGTPを用いたSAT問題の解決

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

† 九州大学大学院システム情報科学府　〒816-8580 福岡県春日市春日公園 6-1
†† 九州大学大学院システム情報科学研究院　〒816-8580 福岡県春日市春日公園 6-1
E-mail: †{kino,matsuda,fujita,koshi,hasegawa}@ar.is.kyushu-u.ac.jp

**あらまし**　本論文では，FPGA 上に実装された定理証明器 PCMGTP（Propositional Constraint Model Generation Theorem Prover）の改良と評価について述べる．PCMGTP は一階述語論理の CMGTP を命題論理に限定したものであり，ハードウェア化に適している．我々が以前実装した PCMGTP を詳細に検討した結果，プログラム上のデータ表現や，推論エンジンの状態数等に冗長な部分を発見した．それらを改善し，新たに BCBE 回路の導入や，トーナメント回路の改良を行うことにより，SAT ソルバ全体の回路規模の大幅な削減や，実行時間の短縮を図ることができ，様々な充足可能性問題に対する実験で優れた実行結果を得ることができた．

**キーワード**　SAT ソルバ，FPGA，再構成可能論理デバイス，定理証明，モデル生成法

# Solving SAT problems by PCMGTP on FPGA

Shohei KINOSHITA†, Junichi MATSUDA†, Hiroshi FUJITA††, Miyuki KOSHIMURA††, and Ryuzo HASEGAWA††

† Department of Intelligent Systems, Graduate School of Information Science and Electrical Engineering, Kyushu University　Kasuga Kouen 6-1,Kasuga-shi,816-8580 Japan
†† Department of Intelligent Systems, Graduate School of Information Science and Electrical Engineering, Kyushu University　Kasuga Kouen 6-1,Kasuga-shi,816-8580 Japan
E-mail: †{kino,matsuda,fujita,koshi,hasegawa}@ar.is.kyushu-u.ac.jp

**Abstract**　In this paper, a new design of the SAT solver PCMGTP implemented on an FPGA chip is described. Although the previous implementation of PCMGTP achieved considerable speedup in solving SAT benchmark problems compared to the software counterpart of MGTP, it failed to compete with the state-of-the-art SAT solvers in speed and was unable to deal with large problems due to the limited capacity of FPGA. We tried to improve our design by removing some redundancy in the data representation and the redundant states in the deduction engine. Also, we developed a new circuit called BCBE for performing logical implication, and modified the tournament circuit so as to deal with very large candidates and to shorten the critical path. As the result, the new implementation achieved speedup by factor of five as well as an order of magnitude space reduction compared to the previous one.

**Key words**　SAT solvers, FPGA, Reconfigurable logic, Theorem proving, Model generation

## 1. はじめに

SAT ソルバは，命題論理式の充足性を判定する判定器である．SAT ソルバの分野においては，すでに数多くのソフトウェアによる優秀な SAT ソルバが開発されているが，それらの SAT ソルバを用いてもなお未解決の充足可能性問題（SAT Problem）は数多く存在している．このため，SAT Competition [1] 等においてより優秀な SAT ソルバの開発を目指し，世界各国で熱心な研究が行われている．

これらの SAT ソルバの多くは，完全性を保障するために Davis-Putnam 法（DP 法）[2] と呼ばれるアルゴリズムを基に作られている．他方，近年ハードウェアを用いることにより，ソフトウェア版 SAT ソルバにおける逐次実行の部分を，効率的に並列処理することで，より高速に解を求める SAT ソルバも開発され始めている [3]．ハードウェアを用いた SAT ソルバは，そのほとんどが，Verilog のようなハードウェア記述言語で記述され，FPGA 上に実装されている．

我々も 2003 年から SAT ソルバを FPGA 上に構築する試み

を始めており，一階述語論理の定理証明系 CMGTP [4] を命題論理に限定して，ハードウェア化に適したアルゴリズムである Propositional CMGTP (PCMGTP) [5] を開発し，それをFPGA 上に実装してきた．今回この PCMGTP について詳しく検討を行ったところ，Verilog-HDL プログラム上におけるデータ表現方法や，推論エンジンの状態数等に冗長な部分を発見した．データ表現においては，各変数の付値準位を bit 幅で表現していたが，それを整数表現に変更することによりレジスタの数を大幅に削減することができた．また，推論エンジンにおいては，以前の実装の 10 状態を 6 状態に削減することに成功した．さらに，BCBE (注1) 回路を新たに導入して効率的な implication の実装が可能となった他，未決定リテラルの少ない非ホーン節を決定するためのトーナメント回路の見直しを行い，多クロック化することによってクリティカルパスを短縮することができた．

これらの改善により，SAT ソルバ全体の回路規模の大幅な削減や，実行時間の短縮を図ることができた．

本稿では，新たに構築した PCMGTP の FPGA 上への実装とその評価について述べる．

## 2. PCMGTP

モデル生成法は，一階述語論理の定理証明法であり，これに基づいた定理証明器として，MGTP(*Model Generation Theorem Prover*) が開発され，それを命題論理に特化した定理証明器が PCMGTP である．そこで，初めにモデル生成法について説明し，その後 PCMGTP について述べる．

モデル生成法において，ある節集合 $\mathbf{C}$ が与えられたときその中の一本の節 $C$ は命題変数 $P_i$ を用いて，

$$P_1 \wedge \ldots \wedge P_n \rightarrow P_{n+1} \vee \ldots \vee P_m$$

のようにして含意式で表す．$\rightarrow$ の左辺を前件，右辺を後件と呼び，それぞれの節は後件の命題変数 $P_i$ の数によって，空の場合を負節，1 個の場合を確定節，それ以上の場合を非ホーン節と呼ばれる．

節集合 $\mathbf{C}$ 中に現れる命題変数のうち，真値が付値されたものの集合 $M$ を $\mathbf{C}$ のモデル候補という．ある節 $C$ がモデル候補 $M$ 中の命題変数の付値にしたがって真となるとき，$M$ が $C$ を充足するという．節集合 $\mathbf{C}$ 中のすべての節を充足するモデル候補 $M$ が存在すればそれは $\mathbf{C}$ のモデルとなる．ある節集合 $\mathbf{C}$ においてあるモデルが見つかった場合，$\mathbf{C}$ は充足可能 (SAT) となる．また逆にモデルが見つからなかった場合，$\mathbf{C}$ は充足不能 (UNSAT) となる．

モデル候補 $M$ の中で前件が充足（前件の命題変数がすべて真）され，後件が充足されない（後件の命題変数のいずれも真ではない）節を違反節と呼ぶ．

モデル生成法では，モデル候補集合 $\mathbf{M}$ を $\mathbf{M_0} = \{M_0 = \phi\}$ から始めて $\mathbf{M_0} \rightarrow \mathbf{M_1} \rightarrow \ldots$ のように拡張していく．モデル候補 $M_i^j \in \mathbf{M_i}$ を選んだとき，違反節を含まなければそれはモデルである（モデル確定）．$M_i^j$ が確定節を違反節として含む場合は，その後件を $P$ とすると $\mathbf{M_{i+1}}$ は $\mathbf{M_i}$ 中の $M_i^j$ を $M_{i+1}^j = M_i^j \cup \{P\}$ で置き換えたものとする（確定節拡張）．非ホーン節の場合は，その後件が $P_{n+1} \vee \ldots \vee P_m$ ならば

$$\mathbf{M_{i+1}} = \{M_i^j \cup \{P_{n+1}\}, \ldots, M_i^j \cup \{P_m\}\} \cup (\mathbf{M_i} \backslash \{M_i^j\})$$

とする（非ホーン拡張）．違反節が負節の場合は，$\mathbf{M_i}$ から $M_i^j$ を取り除く（モデル棄却）．この拡張の結果 $\mathbf{M_i}$ が空となったとき UNSAT，$\mathbf{M_i}$ 中の要素が全てモデルとなったとき SAT と判定する．

問題節集合 $\mathbf{C}$ の各節 $L_1 \vee L_2 \vee \ldots \vee L_n$ から，

$$\overline{L_1} \wedge \ldots \wedge \overline{L_{i-1}} \wedge \overline{L_{i+1}} \wedge \ldots \wedge \overline{L_n} \rightarrow L_i$$

のように n 本の確定節が導かれる．MGTP では $\rightarrow$ の両辺には正リテラルしか許されないが，CMGTP では負リテラルも認め，これらの確定節に対しても確定節拡張を適用する．このとき拡張されたモデル候補 $M$ において $\neg P \in M$ かつ $P \in M$ であるとき，単位反駁が成立するといい，モデル棄却の要件を満たす．そこで，モデル生成法における非ホーン拡張を以下のように変更する．非ホーン節の違反節を検出した場合，その後件が $P_{n+1} \vee \ldots \vee P_m$ ならば，$\mathbf{M_i}$ から $M_i^j$ を取り除き，

$$\left\{ \begin{array}{l} M_i^j \cup \{P_{n+1}, \neg P_{n+2}, \ldots, \neg P_m\}, \\ M_i^j \cup \{P_{n+2}, \neg P_{n+3}, \ldots, \neg P_m\}, \\ \ldots, \\ M_i^j \cup \{P_{m-1}, \neg P_m\}, \\ M_i^j \cup \{P_m\} \end{array} \right\}$$

を加えたものを $\mathbf{M_{i+1}}$ とする．この変更により単位反駁によるモデル棄却の機会を増大させ，探索空間を削減することが可能となる．このように負リテラルについての拡張を行なった CMGTP をさらに命題論理に限定したものが PCMGTP である．PCMGTP のアルゴリズムを以下に示す．

（1） implication による付値

implication : 値の付値されていないリテラル（未定リテラル）がただ一つ存在する節 (unit clause) が存在するとき，その節を真とするようにその未定リテラル（単位含意リテラル）の付値が一意に定まる．implication が可能な unit clause が複数存在するとき，そのすべてに関して同時に implication 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た (contradiction) 場合，そのモデル候補を棄却して (3) に飛ぶ．

unit propagation : implication の結果，新たな unit clause が生じるとき，引き続きその新たな unit clause に関して implication を行なう．

conflict : implication の衝突 (conflict) が生じない限り，implication と unit propagation を繰り返し適用する．implication 可能な unit clause が存在しなくなった場合は，違反節が存在するときは (2) に進み，そうでなければ問題は SAT であり，現モデル候補をモデル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．

(注1) : Burning Candle at Both Ends の略．

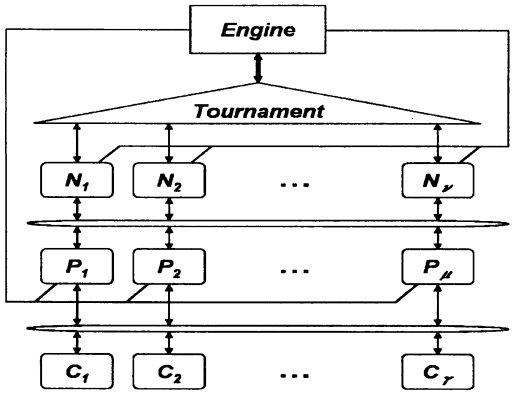


図 1　FPGA 上の PCMGTP のブロック図

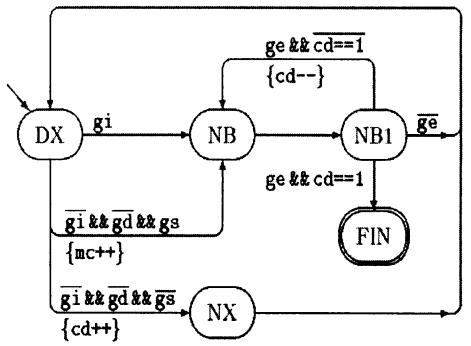


図 2　PCMGTP エンジンの状態遷移図

（2）　違反節の選択，投機的付値，モデル候補の拡張

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，モデル候補を拡張して (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，モデル候補を拡張して (1) に戻る．そうでなければ，その違反節に基づくモデル候補の拡張の場合分けは尽きているので，以前の違反節にバックトラックし，リテラルの再選択を試みる．バックトラック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

モデルが 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．

## 3. FPGA への実装

FPGA に実装する PCMGTP は図 1 に示すように，推論エンジンモジュール *Engine*，命題変数モジュール $P_i$ $(0 \leq i \leq \mu)$，節 (BCBE) モジュール $C_k$ $(0 \leq k \leq \gamma)$，非ホーン節モジュール ($N_j$ $(0 \leq j \leq \nu)$)，トーナメントモジュール *Tournament* によって構成される．$\mu$ は命題変数の総数，$\gamma$ は節の総本数，$\nu$ は非ホーン節の本数を表す．

以下，各モジュールの詳細について述べる．

### 3.1　推論エンジンモジュール

図 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．矢印は遷移を表し，矢印の周囲のラベル gi 等は，遷移を引き起こす条件であり，{ } 内のラベル {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．

gi は現モデル候補棄却，gs は充足，gd は “implication 演算の未了” をそれぞれ表す信号であり，cd は選択準位を表す整数で，cd 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 クロック内に現段階で存在する全ての implication が行われる．その結果，conflict が生じた (gi) ならば，NB に遷移．そうでなければ，
  - 新たな unit clause が生じている (gd) ならば，DX に留まる (unit propagation)．
  - さもなければ implication が完了しているので，充足している (gs) ならば，モデルカウンタを 1 増やして (mc++)，NB に遷移（単解でよい場合は FIN に遷移）．そうでなければ，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 1 個選んで付値を定め，DX に遷移．そうでなければ，
  - トップレベルにある (cd==1) ならば FIN に遷移．
  - そうで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)】: モデルが 0 個ならば UNSAT，そうでなければ SAT を報告．

なお，選択準位の上限は $\lambda = min(\mu, \nu)$ で与えられる．

### 3.2　命題変数モジュール

命題変数モジュールの Verilog コードの一部を図 3 に示す．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

```
assign ic = pr & nr;
assign df = ~(pa | na) & (pr | nr);
assign cancel = nb & (av>=cd);
always@(posedge ck or negedge rst)
  begin
    .....
    pa <= ~cancel & pr & ~na;
    na <= ~cancel & nr & ~pa;
    av <= ( cancel ? 0 : (df ? cd : av));
  end
assign gi = | icv;
assign gd = | dfv;
```

図 3　命題変数モジュールの Verilog コードの一部

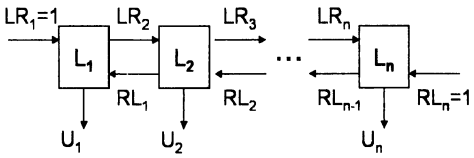


図 4　BCBE 回路

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 conflict が発生した，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．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．

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 implication の未完了を推論エンジンに伝えなければならない．

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．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．

$\mu$ 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 $\mu$ のベクタ icv, dfv を定義し，推論エンジンへの信号 gi, gd をそれぞれ icv，dcv のビット積で表す．

### 3.3　節モジュール（BCBE 回路）

節モジュールは implication のために特に考案された BCBE 方式に基づいて実装されている．図 4 に生成される回路を示す．

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．節を構成するリテラル $Li$ 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 $LRi$ と右端から左向きに流れる信号線 $RLi$ 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である．ある一つのリテラル $Li$ 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，すなわち

$$LRi \ \& \ RLi = 1$$

のとき，このリテラルが単位含意リテラル $Ui$ であることがわかる．

### 3.4　非ホーン節モジュール

図 5 に非ホーン節モジュールの Verilog コードの一部を示す．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 nlits のビット和により負リテラルのすべてが偽となること，csat は plits のビット積により正リテラルのいずれかが真となること，sat_nh は asat の否定と csat の積によりこの節が充足していること，

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

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

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

図 5　非ホーン節モジュールの Verilog コードの一部

および violated は asat と csat の否定を和により違反節であることを表す．

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．）disj_size で計算される未定正リテラルの個数の少ない非ホーン節が勝者となり，その非ホーン節に付値が行われる．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．

ある非ホーン節がトーナメントで優勝し，モデル候補拡張が行われる節となったとき，投機的付値の対象となる未定正リテラルを選択する．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．

$\nu$ 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 $\nu$ のベクタ stv, ebv を定義し，推論エンジンへの信号 gs を stv のビット積，ge を ebv のビット和として表す．

```
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
```

図 6　トーナメント回路の Verilog コード

### 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．図 6 にトーナメントモジュールの Velilog コード(注2) を示す．

ベクタ cnwrk のはじめの $\nu$ ビットには非ホーン節からの 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 $\nu$ 個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 $\nu$ 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決定する．

これらベクタで $\nu$ 以上の添字の信号線上には，そのときそのと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，$\nu$ 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，$\nu$ 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 $\nu$ が数 100 の場合は 2 ステージ分割が適切であった．

## 4.　比較・評価

上で示した PCMGTP を実際に FPGA 上に実装し，問題として DIMACS Challenge benchmark problems [6] の AIM ベンチマーク，N 王妃パズル，および準群問題を使用した．比較するために，ソフトウェア版の MGTP を用いて，同一問題の解を求めた．実験に使用した FPGA は ALTERA 社の EP20K1500EBC652-3 で論理セル数 51840 個であり，論理合成ツールは Quartus II Version4.0 を使用した．またソフトウェア版 MGTP の実行および，論理合成ツールの動作に用いたホストマシンは CPU が 2.8GHz，メインメモリが 2GB のものを使用した．その実行結果を表 1 に示す．なお，表 1 中の下段に示した ( ) 内の値は，今回の改良前の PCMGTP の結果を表し，{ } 内の値はその実行速度比を示している．

いずれの問題においても，正しいモデル数（M）が導き出された．回路規模（Cells）については，実装した中で最大規模の問題である準群問題 qg7-8（変数（Vars）512 個，節数（Cls）17430 本）においてチップ使用率が 91% であり，従来は，qg5-5 など 5 次の準群問題までしかチップにダウンロードできなかったことなどを考えると，今回の PCMGTP 設計がコンパクトになされたことが分かる．

実行時間は，ハードウェア版 PCMGTP（HW）については，問題を解くのに費やしたクロック数（Clks）を最大動作周波数（FMAX）で割ることによって求めた．クロック数は約 2 分の 1〜3 分の 1 に減少しており，最大動作周波数は約 1.5 倍〜2 倍上昇している．その結果，従来の PCMGTP と比較して 4.1〜5.5 倍の速度増加，また，ソフトウェア版 MGTP（SW）と比較しても最大で 6702 倍という桁違いの速度増加が見られた．aim-100-2_0-no-1 においては 86 倍という値に留まっているが，これは，両者の証明方式が異なり，ハードウェア版 PCMGTP の特徴である演算の並列化があまり行われなかった為と考えられる．問題によってこのようにハードウェアの長所が生かされにくいものもあるが，それでもなおハードウェア版 PCMGTP の方が高速に解を導くことができたという点では，ハードウェア版 PCMGTP の優位性がさらに示されたと考える．

(注2)：ただし，1 ステージ版．

表 1　ベンチマーク問題に対する実験結果

| Problem | Vars | Cls | M | Cells | Comp. (sec) | FMAX (MHz) | Clks | HW ($\mu$sec) | 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 | $7.3\times10^7$ | $2.4\times10^6$ | $2.1\times10^5$ | 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 |

しかしハードウェアの論理合成時間（Comp.）については，以前と比較すると，冗長な部分の改善や論理合成ツールの性能向上により，その時間は短縮されたが，未だ相当な時間を費やしていることに変わりはない．これは深刻な問題であるが，未解決 SAT 問題など更に難易度の高い問題の解決が最終目標であり，その場合，実行時間が数日～数ヶ月かかることも珍しくない．ハードウェア版 PCMGTP の演算処理能力がソフトウェアの数百～数千倍であることを考えると，難易度の高い問題に関しては，論理合成時間を考慮に入れても，十分にハードウェア版 PCMGTP のメリットを生かすことができると考えている．

## 5. むすび

本稿は，一階述語論理の定理証明系 MGTP を命題論理に特化した SAT ソルバ PCMGTP の FPGA 上での実装の改良と，比較・評価について述べた．

今回作成した PCMGTP は，主にデータ表現方法の変更により我々が以前作成した PCMGTP に比べ，SAT ソルバ全体の回路規模を約 80%縮小し，FPGA の論理素子使用効率を格段に向上させた．これにより，命題変数 512 個，節数約 17000 本である 8 次の準群問題を 1 つの FPGA チップに実装することが可能となった．また，実行速度についても，トーナメント回路の多クロック化によるクリティカルパスの短縮により，最大動作周波数が約 2 倍になったこと，および，推論エンジンモジュールにおいて，冗長な状態数を削減することができ，状態遷移回数が減少したことによって約 5 倍の速度で実行することが可能となった．さらに，ソフトウェア版 SAT ソルバとの比較においても，その優位性を確認することができた．

証明方式の課題としてソフトウェア版 SAT ソルバに標準的に備わっている補題の自動生成・利用，知的バックトラック等の採用が挙げられる．現在までに folding-up 機能に基づく補題の自動生成・利用についてシミュレーションを行い，その有効性は示されている．しかし，folding-up 機能の実装では回路規模の著しい増加が予想されるため，導入にあたっては何らかの制限を設ける工夫が必要となる．

今回，FPGA 論理合成ツールの発達により，コンパイル時間を多少削減することができたが，その時間はやはり証明実行時間に比べると，かなりの時間を費やしている．ソフトウェアの SAT ソルバに対して更に優位に立つには，論理合成ツールの更なる発展が望まれる．

また，従来は 1 チップの FPGA に回路を実装してきたため，問題の規模に限界が生じていたが，今後は，FPGA 間の相互データ通信を可能とした，多チップ構成の FPGA を用いたより大規模な問題への取り組みや，Flash link 等の外部メモリを使用した FPGA を用いたよりスケーラブルな回路の構築，PCI バスを用いて PC とリンクさせることによって，ソフトウェアとのハイブリットな SAT ソルバの設計等を検討している．

## 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．

## 文　献

[1] "SAT Competitions," http://satlive.org/SATCompetition/index.jsp

[2] M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of ACM*, Vol.7(3), 1960, pp.394–397.

[3] 須山 敬之, 横尾 真, 澤田 宏, 名古屋 彰, "再構成可能なハードウェアを用いた充足可能性問題の解法," **電気情報通信学会論文誌**, Vol.J84-D1, No.4, 2001, pp.410–420.

[4] 白井 康之, 長谷川 隆三, "モデル生成型定理証明システムによる制約充足問題の解決とその並列化," **電子情報通信学会論文誌**, Vol.J80-D-II, No.1, 1997, pp.224–236.

[5] 河野　真史, 藤田 博, 長谷川 隆三, "モデル生成型定理証明器の FPGA 上の実装," **情報処理学会 研究報告**, 2004-SLDM-113, pp.119–124.

[6] ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/