FPGAを用いた詰将棋専用ハードウェアプラットフォームの開発

堀 洋平　斎藤 尚徳　丸山 勉

hori@darwin.esys.tsukuba.ac.jp

筑波大学

概要

人工知能の分野では，木探索による問題解決の手法を探るためにチェスや将棋等のゲームが例題として取り上げられ，盛んに研究されている．チェスにおいては，専用ハードウェア DEEP BLUE が人間の世界チャンピオンに勝利し成功を収めたが，将棋プログラムは現段階では人間のプロ棋士の実力に及ばない．将棋プログラムの棋力の向上のためには，専用ハードウェアの開発が必要不可欠である．本研究では，専用ハードウェア開発の第一段階として，Field-Programmable Gate Array (FPGA) を用いて詰将棋の専用ハードウェアの開発を行った．本論文において，ハードウェアのアーキテクチャと性能について報告する．

An FPGA-Based Hardware Platform for Tsume-Shogi

Yohei Hori, Hisanori Saito and Tsutomu Maruyama

University of Tsukuba

Abstract

Games like chess and shogi are good applications to study methods of problem solving with tree search in Artificial Intelligence. In chess, DEEP BLUE defeated the then-reigning human world champion, while shogi programs are not strong enough to play against human professional players. To improve play strength of shogi programs, developing dedicated hardware is an essential approach. As a first step to shogi-playing hardware, we developed a hardware Tsume-shogi solver with a Field-Programmable Gate Array (FPGA). In this paper we report the architecture and the performance of the hardware.

1　はじめに

ゲームプログラミングは，人工知能の最も初期の段階から研究課題として取り上げられてきた．近年，チェスの専用マシンである Deep Blue が人間の世界チャンピオンである Garry Kasparov に勝利し一定の成功を収めたことにより，チェスに代わる研究対象として将棋プログラムが注目されている．将棋では持駒の再利用が可能であるため 1 局面における合法手の総数が多く，将棋はコンピュータにとってチェスよりも複雑なゲームであると認識されている [8, 13].

将棋はチェスと同じくタクティカルなゲームであり，最善手の決定のためにはある程度深い探索が要

求される．ゆえに将棋においても全幅探索は非現実的であり，ヒューリスティックな前向き枝刈りが行われる．しかし，チェスと同程度まで枝刈りを行った場合，合法手に対して刈り取られる枝の割合がチェスより大きくなり最善手を逃す可能性が高くなると考えられる．精度の高い (最善手を残す確率の高い) 枝刈りを実現するためには，正確な局面評価関数と深い探索の実現が必要であるが，将棋のルールの複雑さと合法手数の多さを考えると，チェスに比べてより多くの計算量が要求される．将棋プログラムの棋力の向上のためには，新しい評価関数や木探索法の開発等のソフトウェア面からのアプローチと，より高速な演算を実現するためのハードウェア面からのアプローチの両方が必要である．

近年，ソフトウェアの開発競争は研究者やメーカーあるいは個人によって盛んに進められており，その中で様々なアルゴリズムが提案されて大きな成果を上げている [9, 10, 11, 12]．しかし，専用ハードウェアに関する研究は，将棋においてはほとんど行われていない．これは，ハードウェア開発における典型的な 2 つの問題 — 時間的・金銭的コストとアーキテクチャの非柔軟性の問題 — が原因であると考えられる．

筆者等は過去の研究において，将棋専用ハードウェアにおけるこれらの問題を解決する手段として，Field-Programmable Gate Array (FPGA) を使用することを提案した [3, 4]．FPGA は，回路構成をユーザが自由に変更することができる LSI である．FPGA の回路構成は，ハードウェア記述言語 (Hardware Description Language: HDL) を使用してテキスト形式のプログラムで記述される．プログラムは計算機上でコンパイルされ，回路構成情報へと変換される．回路の構成および変更は，この回路構成情報をチップにダウンロードすることで直ちに実現される．FPGA を用いた場合，回路の設計から実装までをすべてユーザの環境で実現できるため，開発にかかる時間とコストを大幅に削減することが可能である．

従来の筆者等の研究では，詰将棋における王手と防手 (後手の応手) の生成モジュールを FPGA に実装し，ソフトウェアより高速な指手の生成が実現されていることを示した．今回の研究では，指手生成回路に加えて木探索回路を FPGA に実装し，実際に詰将棋の問題を解いてハードウェアの性能を評価した．実装された木探索アルゴリズムは，証明数を閾値とする多重反復深化を行う PN* [14] である．本論文において，詰将棋ハードウェアのアーキテクチャと性能について説明する．

2 チェスと将棋のハードウェアの比較

チェスの専用ハードウェアの研究は，1970 年代から始まった．最も初期の頃に作られたハードウェアの一つである BELLE は，その後の研究開発に大きな影響を与え，チェス・マシンのアーキテクチャの基礎を確立した．BELLE のアーキテクチャは，DEEP THOUGHT [7], DEEP BLUE [6] にも採用されている．本章では，BELLE の基本的なアーキテクチャについて説明し，将棋ハードウェアへの応用について考察する．

2.1 BELLE のアーキテクチャ

BELLE はベル研究所の Condon と Thompson によって開発され，1977 年に専用ハードウェアとして初めてコンピュータチェスの競技会に参加した [2]．このときのハードウェアは 325 個のチップから構成されていたが，その後も改良が重ねられ，1980 年につくられた BELLE は 1700 個のチップを搭載し 1 秒間に 16 万局面の探索が可能であった．

BELLE は図 1 のように 8x8 個のロジック・ブロック (chess square) から構成され，各 chess square がチェス盤のマスのそれぞれに対応している．Chess square には transmitter (XMIT) と receiver (RECV) が 1 つずつ実装されており，これら 2 つが隣接する square と信号の受送信を行うことによって指手の生成を行う．図 1 中の XMIT の矢印は，各方向へ出力される駒の効きを表しており，RECV の矢印は各方向から到達する駒の効きを表している．L 字型の矢印は，Knight の効きを表している．

BELLE は，捕獲される駒を見つける “find-victim (FV)” と捕獲する駒を特定する “find-aggressor (FA)” という 2 つのサイクルによって指手の生成

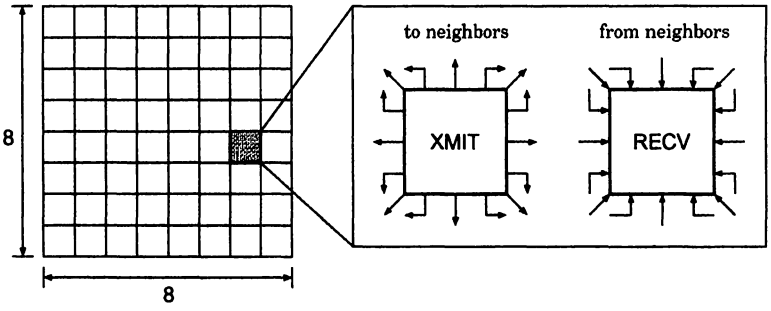

図 1: BELLE の chess square array

を行う．FV では，XMIT が square を占有する駒に応じて，効きのある方向へ信号を送信する．この信号を受けた square 上に相手の駒があった場合，それは捕獲される駒 (victim) の候補となる．これらの候補のうち，最も価値の高い駒を Most Valuable Victim (MVV) という．FA では，MVV のある square から全方向に信号を送信する．この信号を受け取った square 上に味方の駒がある場合，それは victim を捕獲することのできる駒 (aggressor) の候補となる．これらの候補のうち，最も価値の低い駒を Least Valuable Aggressor (LVA) という．このように，MVV/LVA に基づいて最も評価値の高いものから順に指手が生成される．

2.2 将棋との比較

将棋においても，チェスと同じように 9x9 のロジック・ブロックを使う方法が可能であると考えられる．しかし，我々の研究では小規模で低価格なハードウェアの作成を目標としているため，この方法は様々な点で問題がある．

将棋の駒は 8 種類であり，このうち 6 種類が成ることができる．成駒のうち 4 種類は「金」と全く同じ動きをするが，持駒の使用が可能であるためこれらを単純に「金」として扱うことができない．よって実質上 14 種類の駒すべてを区別する必要がある．敵味方を合わせると，将棋の 1 マスには最大で 12 方向から駒が移動してくる．将棋において BELLE と同様の構造をとる場合，各種類の駒の入力が各方向からあるため，1 つのロジック・ブロックはチェスと比べて非常に大きく複雑になると考えられる．また，持駒による指手の生成や，複雑なルール下での成り生成も，ロジックを複雑にする大きな要因である．そのためブロック間の配線量が増加し，ハードウェアの性能が低下することが懸念される．また，将棋盤は 9x9 とチェス盤に比べて大きいので，より多くのハードウェア・リソースが必要となる．

これらの理由から，筆者等は 9x9 のロジック・ブロックを用いる方法ではなく，独自のアーキテクチャによって詰将棋回路を実装した．そのアルゴリズムとアーキテクチャについて，第 3 章以降で説明する．

3 詰将棋のハードウェア・アルゴリズム

3.1 王手の生成

ソフトウェアにおいては，後手玉の位置から上下左右，斜め，桂馬方向に向かって盤面をスキャンし，先手の駒が発見された場合に王手を生成するという方法が一般的であろう (盤面のスキャンとは，対象となるマスのデータをメモリから次々と読み出すことである)．この方法をハードウェアで実現することは可能である．しかし，後手玉の位置は一定ではないので，読み出すメモリのアドレスや読み出しの順番が毎回変わるためハードウェアが複雑になる．また，各方向へのスキャンを順番に行っていては高い性能を得ることができない．

我々のハードウェアでは，王手を生成する前に，まず盤上の先手の駒の効きをすべて調べる．このた

めに，横1段の9マスのデータを同時に読み出しながら，1段目から9段目および9段目から1段目に向かって盤面のスキャンを行う．この2方向からのスキャンは並列に実行される．また，各段のデータはパイプライン回路によって処理されるため，盤面のデータの読み出しは9サイクル連続で行われる．この方法では，王手の生成にまったく関係のない駒の効きも生成されるが，ハードウェアにおいては，求める駒の効きの数によって所要クロック数が変わることはない．第2.2節で述べた様に，本研究では回路が複雑化するのを防ぐため9x9のロジック・ブロックを用いる方法は採用していないが，上で述べたような9マスのデータの並列処理，9段のデータのパイプライン処理によって高速な演算が実現されている．

駒の効きとは駒の可動範囲のことであるから，駒の効きはすべての可能な指手を表している．しかし，詰将棋においては先手の指手は王手でなければならないため，すべての指手の中から王手となるものだけを抜き出す必要がある．我々のハードウェアでは，王手のみを選択するために王手マスクを使用する．王手マスクは局面が更新されるたびに新たに作成する必要があり，計算量の観点からは冗長的であるが，マスクの作成は駒の効きの計算と並列に実行されるので，所要クロック数は増加しない．このように，局面に依存しない手法を用いることにより，ハードウェアの制御が単純化され，より高速な動作周波数の実現が可能となる．

駒の効きとマスクが得られた後，これらのデータを用いることで王手のみが生成される．本研究では先手の指手を「直接王手」「間接王手」「持駒王手」に分類し，これらの王手を別々の回路で生成する並列処理を行っている．

生成された王手はマルチプレクサを経由して木探索回路へと送られ，スタックに保存される．このスタックをどのような順番で読み書きするかによって，様々な木探索の実現が可能となる．木探索回路によって選択された王手によって局面が更新されると，手番は後手側に移り，続いて防手の生成が行われる．

ハードウェアにおいて王手を生成する手順は，以下の様にまとめられる．

1. 駒の効きの計算，マスクの生成
2. 直接王手，持駒王手，開き王手の生成
3. マルチプレクサを経由した王手のスタックへの書き込み
4. 木探索回路による指手の決定，局面の更新

3.2 防手の生成

防手の場合も，王手生成の場合と同様な盤面のスキャンによって駒の効きを求め，その後各カテゴリの防手を並列に生成する．防手は「玉移動防手」「捕獲防手」「合駒防手」に分類されており，これらが並列に生成される．ハードウェアにおいて防手を生成する手順は基本的に王手の生成と同様であり，以下のようにまとめられる．

1. 後手の直接効きデータの生成
2. 玉移動防手，捕獲防手，合駒防手の生成
3. マルチプレクサを経由した防手のスタックへの書き込み
4. 木探索回路による指手の決定，局面の更新

3.3 木探索アルゴリズム

本研究ではPN*アルゴリズムによって木探索を行う．ハードウェアでは，木探索は複雑なステートマシンによって制御されている．2に，木探索回路におけるステートマシンの状態遷移図を示す．

ステートマシンの各状態では，以下に述べる処理によって木探索を制御している．ステートマシンの状態遷移は複雑であるため，ここでは簡単な説明に留める．

初期化 (INI) レジスタ，メモリの初期化を行う．

局面更新 (UDP) メモリより局面データを読み出し，選択された指手によって局面を更新する．また，更新された局面データよりハッシュコードを生成する．

王手生成 (BMG) 指手生成回路に王手の生成命令を出す．

防手生成 (WMG) 指手生成回路に防手の生成命令

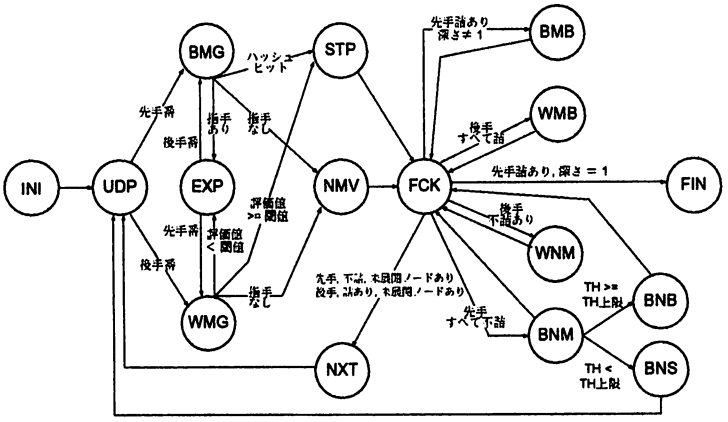

図 2: 木探索回路の状態遷移図

を出す．生成された指手によって局面を更新し，評価値の和を計算する．

展開 (EXP) 現在局面を子節点に設定し，ゲーム木を 1 段進む．

生成中止 (STP) ハードウェアでは，ハッシュ表の参照は指手の生成と並列に実行されている．ハッシュ表には，その節点の証明数と局面データが登録されている．先手局面では，この値が多重反復深化における閾値を超えていた場合に指手の生成を中止する．後手局面では，登録されている兄弟節点の証明数の和が閾値を超えていた場合に指手の生成を中止する．

指手なし (NMV) 指手生成回路において指手が生成されなかった場合，先手局面ならば詰フラグを "01" (不詰) とし，ハッシュ表に証明数 ∞ を登録する．後手局面ならば詰フラグを "10" (詰) とし，証明数 0 を登録する．

詰フラグチェック (FCK) 先手局面の場合は，子節点に詰が 1 つでもあればその局面は詰であるとし，詰フラグを "10" とする．子節点に詰がない場合で，未展開の節点がある場合は，詰フラグを "00" (未展開) とする．すべての子節点が不詰であれば，詰フラグを "01" とする．後手局面の場合は，子節点に 1 つでもあればその局面は不詰であるとし，詰フラグを "01" とする．不詰がない場合で，未展開の節点がある場合は詰フラグを "00" とする．すべての子節点が詰である場合は，詰フラグを "10" とする．

節点移動 (NXT) 詰フラグが "00" であった場合に，現在局面を未展開の兄弟節点に移動する．

先手詰 (BMB) 詰フラグが "10" であった場合に，指手データをスタックから読み出してに詰フラグデータを記入し，再びスタックに戻す．また，ハッシュ表に証明数 0 を登録する．

後手詰 (WMB) BMB と同様．

先手不詰 (BNM) 詰フラグが "01" であった場合に，現在の閾値とその反復における閾値の上限を比較する．

先手不詰戻る (BMB) 現在の閾値がその反復における閾値の上限以上であった場合，探索木を 1 段戻る．

先手不詰不戻 (BNS) 現在の閾値がその反復における閾値の上限未満であった場合，現在の閾値に 1 を加え，兄弟節点の詰フラグをすべて "00" に設定し直して再び探索を行う．

終了 (FIN) 終了フラグを立てる．

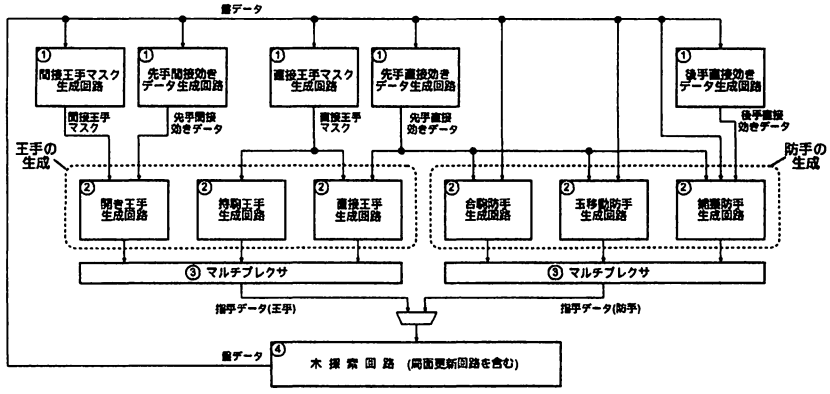

図 3: 詰将棋回路のブロック図

4 詰将棋回路

4.1 ハードウェアのブロック図

詰将棋回路のブロック図を図 3 に示す．図 3 中のモジュールにつけられている番号はハードウェア処理のステージを表しており，第 3.1 節および第 3.2 節で述べた手順中の番号に対応している．各ステージのモジュールはすべて並列に動作する．さらに，各モジュールはその内部において並列・パイプライン処理が行われており，高速な指手の生成が実現されている．各モジュールのアーキテクチャについては，文献 [5] を参照されたい．

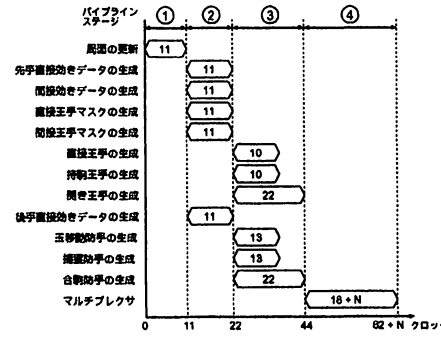

図 4: 詰将棋回路のタイミングチャート

4.2 性能

本研究では，Alpha Data 社製の FPGA ボード ADM-XRC-2 に詰将棋回路を実装し，性能の評価を行った．ADM-XRC-2 は，Xilinx 社 Virtex-II シリーズの FPGA である XC2V6000 と，8 バンク 40 MB (6 バンク × 4 MB，2 バンク × 8 MB) の SSRAM モジュールを搭載している [1]．XC2V6000 は，論理回路を実現する Slice 33,792 個，メモリを実現する 18 Kb の Block RAM 144 個から構成されている [15]．また，ハードウェアの設計には，Xilinx 社製の EDA ツールである Foundation 4.1 を使用した．

実装された詰将棋回路のハードウェアリソースの使用率，および最大動作周波数を表 1 に示す．また図 4 に，ハードウェアにおける指手生成のタイミングチャートを示す．各行程中の数値は演算のステップ数を表し，マルチプレクサの所要ステップ中の N は 1 局面に対する指手の数を表す．マルチプレクサは，各指手生成回路の出力より王手・防手のみを選択しメモリへ送るモジュールであり，この処理は逐次的にしか実行することができないため，所要ステップ数は生成される指手の数に依存する．

本研究では，内藤による詰将棋選集の 100 題の中から双玉問題を除く 98 題をテスト問題として使用し，ハードウェアおよびソフトウェアによるゲーム木のノードの展開能力を比較した．表 2 に，各手数の問題において展開されたノード数の平均と，ソフトウェア (SW) およびハードウェア (HW) による実

表 1: 詰将棋回路のリソース使用率と動作周波数

モジュール	Slice 使用数	(使用率)	RAM 使用数	(使用率)	動作周波数 [MHz]
王手生成回路	13,460/33,792	(39%)	66/144	(45%)	53.348
防手生成回路	8,007/33,792	(23%)	25/144	(17%)	46.553
木探索回路	1,544/33,792	(4.6%)	43/144	(29%)	69.633
回路全体	23,011/33,792	(68%)	134/144	(93%)	46.553

表 2: 詰将棋問題において展開されたノード数

手数	問題数	平均展開ノード数	SW 平均 [msec]	HW 平均 [msec]	性能比
3 手詰	2	1940	55	5.3	10.3
5 手詰	9	5494	137	20.5	6.67
7 手詰	17	3282	80	10.2	7.82
9 手詰	23	12452	270	44.8	6.02
11 手詰	23	19901	379	68.1	5.57
13 手詰	17	28389	546	97.9	5.58
15 手詰	6	60889	1012	181	5.60
17 手詰	1	37801	590	114	5.20
全問題	98	19874	330	59.3	5.51

行時間の平均を示す．ハードウェアの実行時間は，所要クロック数と動作周波数から計算されたものである．所要クロック数は，EDA ツールに付属のシミュレーションツールを用いて測定した．ソフトウェアは性能比較のために独自に作成したものであり，実行環境は，CPU が Pentium4-2.53 GHz，OS は Windows2000 上の Unix エミュレータ Cygwin である．ソフトウェアにもハードウェアと同様の木探索アルゴリズムと局面表が実装されているが，跳駒の位置を覚えて指手の高速化を図るなど，できる限りのチューンナップがされている．ただし，生成される指手の順序およびノードの展開順序はハードウェアとまったく同じくなるように設計されており，正確なノードの展開能力を測定することができる．現段階では，ハッシュ表の大きさの制約や効率化アルゴリズムの未実装により長手詰の問題が解けていないが，ここでは同じアルゴリズムによるノードの展開能力に注目する．

図 2 が示すように，本研究で作成した詰将棋回路はソフトウェアに対して 5 倍以上の性能を得ることができた．またハードウェアのノード展開能力は，1 秒間あたり約 34 万局面であった．

4.3 考察

図 4 より，局面の更新が開始されてから全王手または全防手が生成されるまでのステップ数は $62+N$ であるから，指手の生成に要する時間は

$$(62+N)\times\frac{1}{46.553} \qquad (1)$$

となる．脊尾らによって求められた詰将棋のゲーム木の分岐数の平均は 5.23 であり [14]，上式 (1) において $N=5$ を代入すると，指手 5 個の生成にかかる演算時間は 1.44 μsec となり高速な指手の生成が実現されていることがわかる．

実装した詰将棋ハードウェアでは，並列度の高い演算によって指手の生成が行われており，全指手の生成に要する時間は生成する指手の数にはあまり依存しない．たとえば生成する指手数 N が 10 である場合，ソフトウェアによる逐次処理での所要時間は $N=5$ である場合の約 2 倍と予想されるが，ハードウェアによる並列処理では，式 (1) よりわずか 1.07 倍増加するのみである．ゆえに，1 局面あたりの指手数が多い複雑な問題ほど，ハードウェアのソフトウェアに対する性能は向上する．詰将棋は芸術作品としての側面があるため，先手に与えられる持駒が必要最低限であり，さらに詰局面において持駒が残ってはならないという制約がある．実際の将

棋の終盤では，詰ませるためには必要のない持駒が存在するなど，1局面あたりに可能な指手数は詰将棋よりも多くなると考えられる．ゆえにハードウェアは，実戦局面においてさらに高い性能を示すと考えられる．

5 おわりに

本研究では，将棋専用ハードウェアの開発の第一段階として，FPGAを用いた詰将棋専用ハードウェアの作成を行った．王手生成回路，防手生成回路，木探索回路をFPGAに実装し，実際に詰将棋の問題を解いてソフトウェアと処理時間を比較したところ，5倍以上の性能を得ることができた．ハードウェア処理は，複雑な詰将棋の問題や候補手の極めて多い本将棋において高い性能を示すことが期待される．今後，未実装のアルゴリズムのハードウェア化を進めるとともに，詰将棋回路を利用して本将棋ハードウェアの開発を行う．

参考文献

[1] Alpha Data Parallel Systems: *ADM-XRC-II User Manual, ver.1.5*, Edinburgh, UK (2002).

[2] Condon, J. and Thompson, K.: BELLE chess hardware, *Advances in Computer Chess 3* (Clarke, M.(ed.)), Pargamon Press, Oxford, pp. 45–54 (1982).

[3] Hori, Y., Seki, M., Grimbergen, R., Maruyama, T. and Hoshino, T.: A Shogi Processor with a Field Programmable Gate Array, *Computers and Games*, pp. 297–314 (2000).

[4] Hori, Y., Sonoyama, M. and Maruyama, T.: An FPGA-Based Processor for Shogi Mating Problems, *IEEE International Conference on Field-Programmable Technology*, pp. 117–124 (2002).

[5] 堀洋平, Grimbergen, R., 丸山勉: Field-Programmable Gate Arrayによる将棋専用プロセッサの開発, アマ4段を超える — コンピュータ将棋の進歩4 — (松原仁 (編)), 共立出版, pp. 41–67 (2003).

[6] Hsu, F.-h.: IBM's Deep Blue Chess Grandmaster Chips, *IEEE Micro*, Vol. 19, No. 2, pp. 70–81 (1999).

[7] Hsu, F.-h., Anantharman, T., Campbell, M. and Nowatzyk: Deep Thought, *Computers, Chess, and Cognition* (Marsland, T. and Schaeffer, J.(eds.)), Springer, Berlin, pp. 55–78 (1990).

[8] Matsubara, H. and Grimbergen, R.: Differences between Shogi and western Chess from a computational point of view, *Board Game in Academia* (1997).

[9] 松原仁 (編): コンピュータ将棋の進歩, 共立出版 (1996).

[10] 松原仁 (編): コンピュータ将棋の進歩2, 共立出版 (1998).

[11] 松原仁 (編): コンピュータ将棋の進歩3, 共立出版 (2000).

[12] 松原仁 (編): アマ4段を超える —コンピュータ将棋の進歩4—, 共立出版 (2003).

[13] 松原仁, 半田剣一: ゲームとしての将棋のいくつかの性質について, 情報処理学会人工知能研究会資料 96-3, pp. 21–30 (1994).

[14] Seo, M., Iida, H. and Uiterwijk, J. W. H. M.: The PN*-search algorithm: Application to tsume-shogi, *Artificial Intelligence*, Vol. 129, pp. 253–277 (2001).

[15] Xilinx Inc.: *Virtex-II 1.5V Field Programmable Gate Arrays v1.7*, San Jose, CA (2001).