International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:2, 2013

Game-Tree Simplification by Pattern Matching and Its Acceleration Approach using an FPGA

Suguru Ochiai, Toru Yabuki, Yoshiki Yamaguchi, and Yuetsu Kodama

***Abstract*—**In this paper, we propose a Connect6 solver which adopts a hybrid approach based on a tree-search algorithm and image processing techniques. The solver must deal with the complicated computation and provide high performance in order to make real-time decisions. The proposed approach enables the solver to be implemented on a single Spartan-6 XC6SLX45 FPGA produced by XILINX without using any external devices. The compact implementation is achieved through image processing techniques to optimize a tree-search algorithm of the Connect6 game. The tree search is widely used in computer games and the optimal search brings the best move in every turn of a computer game. Thus, many tree-search algorithms such as Minimax algorithm and artificial intelligence approaches have been widely proposed in this field. However, there is one fundamental problem in this area; the computation time increases rapidly in response to the growth of the game tree. It means the larger the game tree is, the bigger the circuit size is because of their highly parallel computation characteristics. Here, this paper aims to reduce the size of a Connect6 game tree using image processing techniques and its position symmetric property. The proposed solver is composed of four computational modules: a two-dimensional checkmate strategy checker, a template matching module, a skilful-line predictor, and a next-move selector. These modules work well together in selecting next moves from some candidates and the total amount of their circuits is small. The details of the hardware design for an FPGA implementation are described and the performance of this design is also shown in this paper.

***Keywords*—**Connect6, pattern matching, game-tree reduction, hardware direct computation

I. Introduction

THE game of GO, Chess, Reversi and Connect6 are categorized as the kind of two-player, zero-sum, and logical-perfection-information games. The player who has a deeper insight wins the games and the fortuity has absolutely no influence on them. Thus, many earlier studies have proposed computational approaches to find the optimal strategy. Deep Blue [1] and Logistello [2] are representative examples in this field. Deep Blue was a Chess-playing computer developed by IBM, and it won against the Chess world champion in 1997. Logistello was a Reversi-playing computer developed by Michael Buro, and it also won against the Reversi world champion in 1997. In these studies, tree-search algorithms are used for finding the game strategy. Fig. 1 shows the game tree in the case of a 2-by-2 board game.

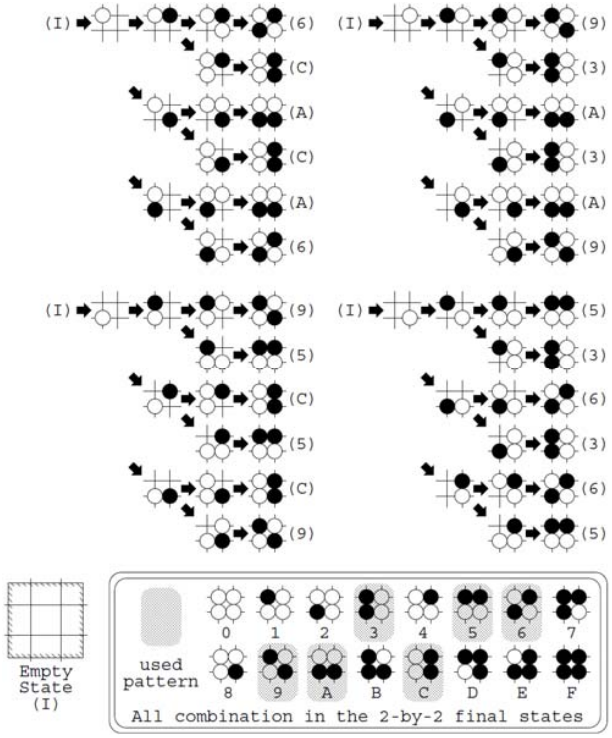

Fig. 1 This is an example of arrangements of GO stones on a 2-by-2-grid board. The initial state is empty and it is represented by a capital (I). In the game tree, the 24 leaf nodes are found. But, the number of arrangement is six and it can be found as used patterns in the double frame

As shown in Fig. 1, the computational complexity with full search is O(n!) based on the number of leaf nodes. The leaf-node check is the same as full-search, which is the best approach when the game tree is small. But the other complexity is $O(n^2)$ when the number of stone arrangements is chosen. This enables us to treat the large game trees; however, it is difficult to use simple game-tree searches in this case as evidenced in Fig. 1. For example, the game complexity of the GO becomes $O(10^{360})$ and it is impossible to compute even if we use the best supercomputer in the world.

For these reasons, tree-reduction algorithms have been discussed. Stochastic approaches such as Monte Carlo technique [3] and meta-heuristic approaches such as neural network approach [4] garner attention from the field of computational games. These approaches are very powerful and

S. Ochiai and T. Yabuki are with the Graduate School of Systems and Information Engineering at University of Tsukuba. 1-1-1 Ten-ou-dai Tsukuba Ibaraki, 3058573, Japan

Y. Yamaguchi and Y. Kodama are with the Faculty of Engineering, Information and Systems at University of Tsukuba, 1-1-1 Ten-ou-dai Tsukuba Ibaraki, 305-8573, Japan (corresponding author to provide e-mail: yoshiki@cs.tsukuba.ac.jp, and see http://www.cs.tsukuba.ac.jp/~yoshiki/).

can dramatically reduce the computational cost. But, there are many parameters in the evaluation function. It is sometimes difficult for us to optimize these parameters from our intuition though games are just games we play. Here, this paper does not treat these parameters directly but it indirectly optimizes them through the image processing. It can reduce the game complexity from $O(10^{360})$ to $O(10^{108})$ when the game of the GO is chosen.

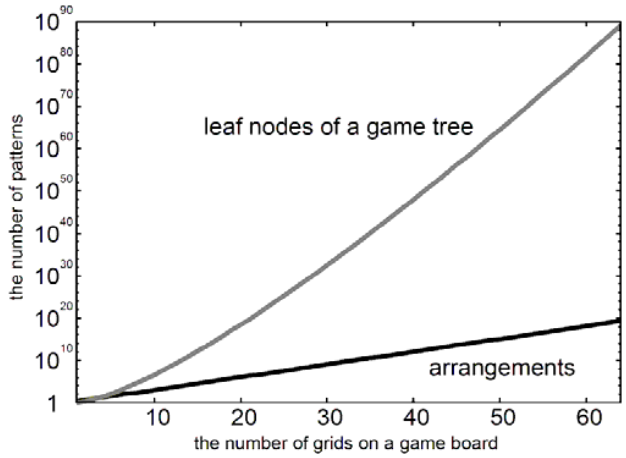

Fig. 2 This graph shows the comparison of game-tree search space. The number of grids means the size of a game board; the square root of them shows the length of the board. The gray-colored line represents the complexity of a simple game-tree search (the number of leaf nodes of a game tree) and the black-colored line is for that of image-processing approaches (the number of arrangements)

In this paper, to verify the proposed approach, the Connect6 is chosen for the target application. The Connect6 is a well-known *k*-in-a-row games and its impartiality is proven theoretically compared with other *k*-in-a-row games. The search domain of the Connect6 (10^{140}) is bigger than that of the Chess (10^{123}) and therefore hardware acceleration is required for constructing Connect6 solvers. This paper proposes an approach which differs from a common practice based on tree-search algorithm, and aims to achieve a highly efficient approach for hardware acceleration.

In related studies, we can find many efficient tree-search algorithms. They are a powerful method to treat this type of applications and some pruning algorithms may bring additional positive results [5]. However, from the viewpoint of hardware resources, it unfits for hardware acceleration. The amount of the circuits is too large to implement on medium-sized FPGAs [6]. The cooperative computing of hardware and software may be one of the solutions but it also requires large circuit utilizations [7]. Other papers discuss different implementations but it is based on a concept of a software implementation [8][9]. It is difficult to achieve a dramatic speedup unless we lay out the direction for functional change as the game progresses.

The basic concept of our proposed approach adopts image processing. It mimics a Chess player, which adopts different computation as a game phase changes. Thus, the proposed solver is composed of four modules: a two-dimensional checkmate strategy checker, a template matching module, a skilful-line predictor, and a next-move selector. Each module except for the next-move selector suggests a candidate for the next move, and the next-move selector chooses the optimal candidate of a stage at the moment. The advantage of this approach is easy to enhance module optimization because a single module does not need to handle the entire game stage. In addition, the approach goes well with hardware direct implementation. All modules can be implemented on a XILINX Spartan-6 XC6SLX45 FPGA which is middle-size FPGA at present. Any external devices are never used and the circuit utilization was about 6,866 LUTs (25.2%) and 16 BRAMs (4.6%).

The rest of this paper is organized as follows: Section II explains the Connect6 and what is the difficulty point for the implementation on an FPGA. Section III shows the implementation design and our proposed architecture. It includes three main circuits and one decision making function. Complex arrangement can be judged by this combination. Section IV and Section V show the performance of propose implementation and our conclusions, respectively. The future work is also included in the Section V.

II. Rule of the Connect6

Connect6 is a two-player game classified into a member of *k*-in-a-row games. A summary of Connect6 is the following.

- Connect6 is played on the GO board whose size is from 19-by-19 grids to 59-by-59 grids.
- Each stone can be placed on an unoccupied grid on the board.
- The player ***B*** can get the first move and put one black stone.
- The player ***W*** can put two white stones after ***B***.
- ***B*** and ***W*** alternate to place two black stones and two white stones.
- If a player places six or more consecutive stones in the row, the player is a winner.

Connect6 ensures fairness between ***B*** and ***W*** though the game is very simple like the above [10][11]. On the other hand, the space complexity is at least 10^{171} even if the 19-by-19-grid board is used. It is impossible to store all situations in the memory. A game tree structure may reduce the size because the average length of game records is 30 but the complexity is still large, approximately 10^{140}. Thus, various acceleration approaches have been discussed for Connect6.

III. Design Principle

A. Overview of the Proposed Approach

Fig. 3 shows a resign diagram of Connect6 and its record. In the Connect6, six-consecutive stones decide a winner and it is expected from a game record. For example, two four-consecutive stones decide victory and defeat in Fig. 3.

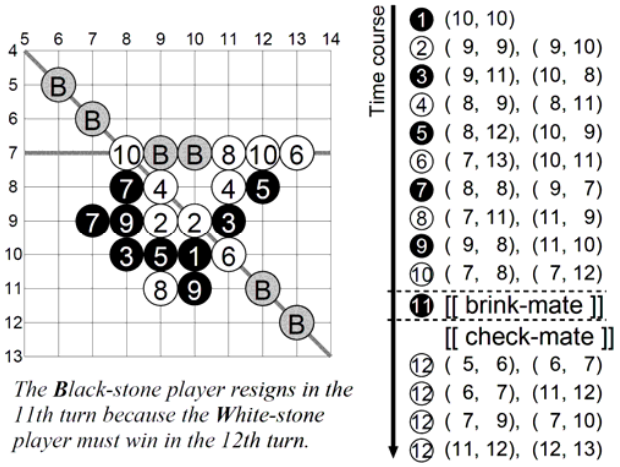

Fig. 3 A Connect6 resign diagram and its game record

A Connect6 solver detects the best move and places a stone on the board if it is able to expect the entire game records. However, it is too hard to detect the best move since all of them cannot be simply implemented on a solver as explained in Section II. Tree-search algorithm and pruning algorithm are the powerful approach used for this type of two-player games in general. If a Connect6 solver alters its computation in response to the game progress, it has the possibility to achieve highly efficient acceleration. Thus, the concept of our system is inspired by chess solvers, whose computation is changed to fit the game stages: opening, middle-game, and end-game stage. It is composed of three main computational modules: a checkmate strategy checker, a template matching module, and a skilful-line predictor. Fig. 4 shows the top design of our proposed Connect6 solver.

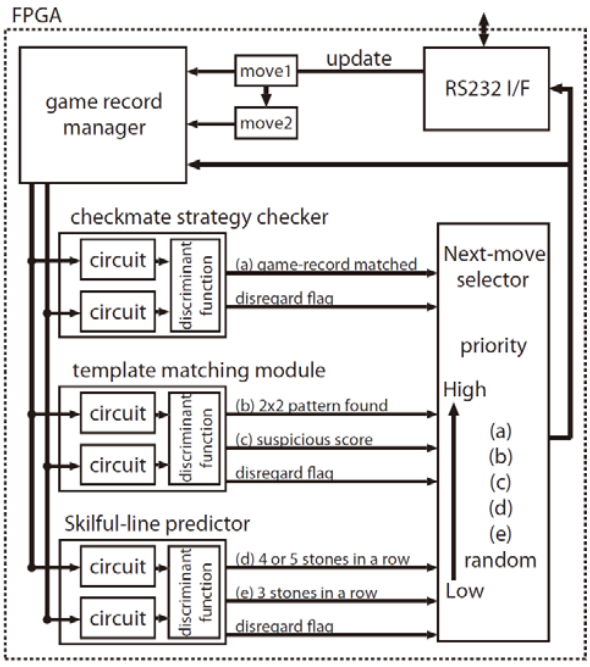

Fig. 4 Top-level Design for Connect6 computation

The following section describes the details of these modules.

B. Checkmate Strategy Checker

In Connect6, the degree of interdependence among stones falls in inverse proportion to the distance of them. For example, the spatial implication of a stone is limited and the maximum number is five. If an 11-by-11-grid Connect6 space is computed in parallel, the solver fulfils desirable features. This paper breaks the space down into a 5-by-5-grid space as shown in Fig. 5 since the trade-off among computational efficiency and circuit resource is considered.

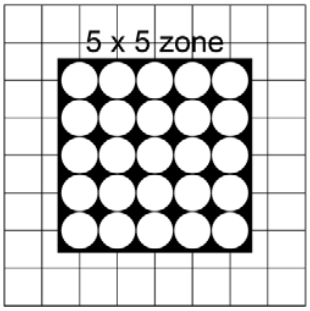

Fig. 5 Region-dividing technique for template search

Here, the entire board was also divided into nine areas based on a 5-by-5-grid space as shown in the top of Fig. 6. The new strategy can be considered on the basis of these subspaces. For example, in the opening stage, the relationship of a long distance between two stones does not have to be considered because the number of stones is few. The difficulty of a tactics increases when the game shifts to the middle stage. However, the computational space can be limited aside from a couple of exceptions because the average length of Connect6 game records implies a sufficient size. The exceptions will be computed by the skilful-line predictor. The spatial influence decreases drastically in the end-game stage since the most of stones interrupts each other's candidates.

To achieve the further reduction of circuit resources, a triangular search region is considered in a 5-by-5-grid subspace as shown in the bottom of Fig. 6. If the four same-color stones are placed in a single region, victory and defeat are decided at that time with high probability. At least two stones should be put on the subspace for the viewpoint of the defense player when the offence player places some stones in the triangular region. By limiting the search space into the triangular region, the number of how to put the other stones can be reduced up to 105 ($=_{15}C_2$). The small number enables a solver to store the tactics in on-chip memories of an FPGA. The triangular region should be rotated as shown in the bottom of Fig. 6 because of further optimization. In this paper, the next-move candidate of offensive and defensive tactics is generated by the same circuits and the same procedure.

C. Template Matching Module

The number of cases where only the same stones exist in a subspace decreases as the hand advances. As the game advances, it is necessary to judge which shape of ***W*** or ***B*** is

superior in the subspace and to generate the next two moves. It means, it is necessary to think about the case where the checkmate strategy checker cannot be applied. Here, the next moves are also decided by the pattern matching technique using the template of 3-by-2 grids or less at this time. The template used is shown in Fig. 7.

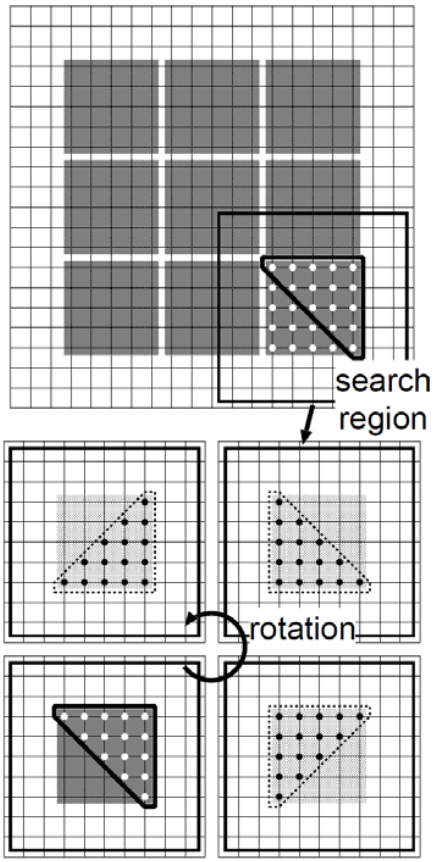

Fig. 4 Nine 5-by-5-grid subspaces and its triangle folding search

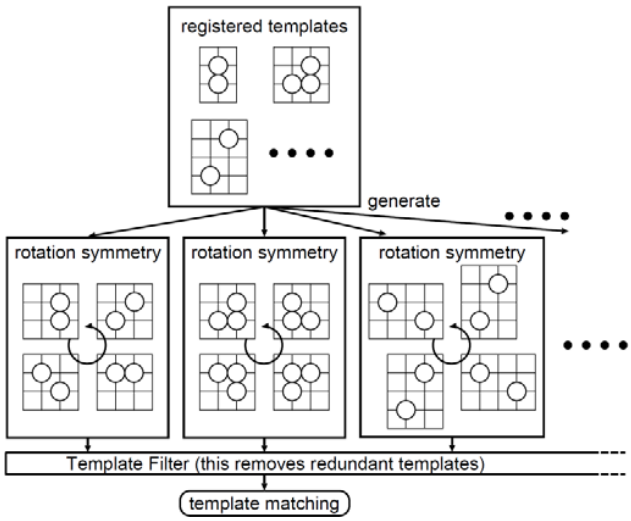

Fig. 7 Template generation for pattern matching

In this template matching, each grid has a priority point which accumulates a severity index inherent to a template. These points indicate next-move candidates and therefore they are used in the next-move selector. Then, a scan window is introduced to the process. The scan window where the opponent's stone is placed at centre is a 5-by-5-grid space, and it is clipped from the color 15-by-15-grid region shown in the top of Fig. 6. The 5-by-5-grid window is shifted when it protrudes from the end of the 15-by-15-grid region, and the distance at that time is minimized. The template matching is also done by rotating the template by 45 degrees, and therefore it corresponds to the point symmetry. This paper selected 12 templates from the limitation of the circuit size though it was preferable to use more templates.

D.Skilful-Line Predictor

The Connect6 game which is composed of ***B***, ***W***, and the unoccupied state requires the triplet search tree for the decision of the strategy. Although the average length of the game record is empirically-deduced to 30, the complexity is still too large (approximately 10^{140}). To achieve the further reduction, an image processing technique of 11-by-11 pixels was thought. There was impossibility in the use of the loss-less compression technique and the processing because of the limitation of the circuit resource. Thus, the search for two dimensions uses the template matching in Section III-C. One-dimensional search is used for a long-distant search as shown in Fig. 8.

In this computation, the distance was extended from 11 to 13 as a result by considering the tradeoffs between distance-extension and circuit difficulty. The reason is that there are some difficult-checkmating situations; an example is shown in the bottom of Fig. 8. It is difficult for the simple circuits to predict a defeat and/or to find out the optimal place to avoid being a loser. In this circuit, the scanning of eight directions can be completed by repeating the rotation of 45 degrees three times.

E. Next-Move Selector and I/O Interface

First of all, the coordinates of two moves of the opponent are received via the RS232C interface, and it stores in the game-record manager. Secondly, each move is transformed to an address of memories, and the surrounding game record is sent to each module as shown in Fig. 4. All the circuits are made double because the number of moves is always two in these modules.

In each discriminate function in three computational modules, relative coordinates are converted into the absolute coordinate, and

- when the both is independent, it keeps the processing.
- when the both is dependent, both next moves are adjusted based on the absolute coordinate.

The output of each circuit has priority. The next moves are decided based on the priority in the next-move selector. Here, the "random" move in Fig. 4 means processing where the stone is put at random when the next moves cannot decide in the next-move selector. In "random" mode, the two stones will be placed in a 5-by-5-grid region, as shown in the top of Fig. 6, in

which it becomes empty most. It is important that the circuits where the next move is generated are efficiently designed compact.

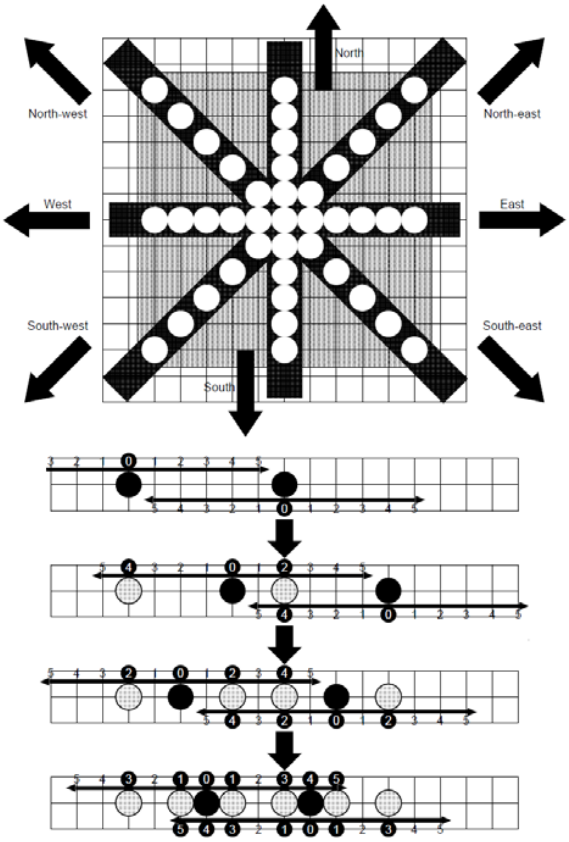

Fig. 8 The eight directions and an difficult checkmate problem

IV. Performance Evaluation

A. Hardware Resources

All circuits in Fig. 4 are implemented on one single XILINX Spartan-6 XC6SLX45 FPGA[1] and any external devices such as DDR-SDRAM modules are never used. The FPGA is embedded on a Atlys FPGA board produced by Memec Design Inc [12]. Fig. 9 shows the overview of this FPGA board.

Table I shows the resource usage. All modules can be implemented on one single small FPGA.

TABLE I
Circuit Usage of the Proposed Connect6 Solver

	Used	Usage Ratio
Logic Element	6,866	25.2%(=6,966/27,288)
Flip Flop	2,877	5.3%(=2,877/54,576)
Block RAM	16	4.6%(=16/348)

[1] The same circuit can be implemented on one 10-years-old FPGA, XILINX Virtex-II XC2V1000 device. The compact design enables it to be implemented on most kinds of FPGAs.

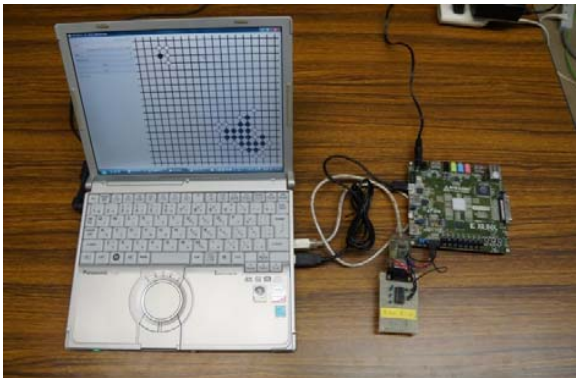

Fig. 9 Atlys™ board (Spartan-6 XC6SLX45) connected to a laptop PC

B. Strategic effectiveness

The strategic effectiveness should be statistically valid to proof the proposed solver. It was evaluated using a software program provided by the FPGA design contest committee in conjunction with the International Conference of Field-Programmable Technology 2011 (ICFPT'11)[2] [13]. In this trial experiment, 100 matches of the Connect6 game are tested. The result is shown in Figs. 10 and 11.

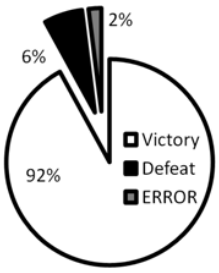

Fig. 10 Winning Percentage of Our Trial Experiment

The results show that our proposed solver has 91 victories and 8 defeats. The solver achieves a tolerable strategy since the winning percentage is 92.0%.

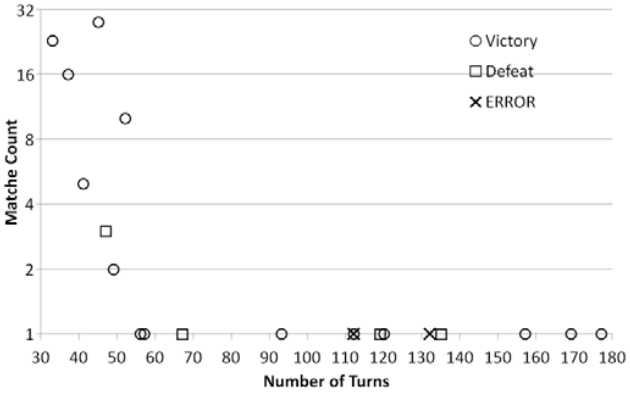

Fig. 11 Detail analysis of our trial experiment

[2] The FPGA Design Competition whose theme was the Connect6 was held also in ICFPT12 [14].

In Fig. 11, we can find defeats after 45 moves. It comes from our over-reduction of the computational efforts. As shown in Fig. 4, our solver omits the edge region of the board because of the reduction of computation. At the early stage, it is apparent that both players do not put their stones on the edge region. But, when the stone is filled on the board, this influence could not be disregarded. A few errors were seen in matches and they also cause in an edge region.

Well, we gave priority to the miniaturization of the circuit and it was achieved. It is possible to correspond by adding as an exceptional computation because the circuit resources remain on the FPGA. The exceptional circuit will be realized by 2,000 LUTs or less; the total amount of the circuit use will be less than 10,000 LUTs derived from Table I. This achievement is significant experiment to find the tradeoff of its hardware usage and high performance. The idea will be effective in a similar tree search algorithm.

V. Conclusion

The three combination functions work well and seem to be efficient methods for generating Connect6 moves. However, there is room for improvement about the enumerated point.

- Template matching: we could not implement larger number of templates. Especially they only take care of the opponent's move. It is important to put both moves in the templates.
- Skilful-line predictor (1-D pattern-guessing tree): we are satisfied with this approach as one method. On the other hand, we would like to reuse the template matching circuit, and implement a bit complicated functions.
- Move generation from the viewpoint of offensive side: we use nine subspaces for generating the next moves, but that is still random moves. The more efficient moves should be generated by using the result of the template matching.

We would like to keep examining the improvement of this implementation though the circuit is limited.

VI. Acknowledgment

This work was partially supported by the Program for Enhancing Systematic Education in Graduate Schools, “Program for Development of ICT Solution Architects”, from the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Japan.

References

[1] Murray Campbell, A. Joseph Hoane Jr., and Feng hsiung Hsu. Deep Blue. Artificial Intelligence, 134:57–83, 2002.
[2] Michael Buro. Entertainment Computing - Technology and Applications, volume 112 of IFIP Advances in Information and Communication Technology, chapter The Evolution Of Strong Othello Programs, pages 81–88. Springer, 2002.
[3] Remi Coulom, Computing Elo Ratings of Move Patterns in the Game of GO, in Proceedings of the Computer Games Workshop, pp1-11, 2007.
[4] Helmut A. Mayer and Peter Maier. Coevolution of neural go players in a cultural environment. In The 2005 IEEE Congress on Evolutionary Computation, volume 2, pages 1017–1024, 2005.
[5] I-Chen Wu and Ping-Hung Lin. Relevance-Zone-Oriented Proof Search for Connect6. IEEE Transactions on Computational Intelligence and AI in Games, 2(3):191–207, 2010.
[6] Takahiro Watanabe, Retsu Moriwaki, Yuichiro Yamaji, Yuki Kamikubo, Yuki Torigai, Yuki Nihira, Takashi Yoza, Yumiko Ueno, Yuji Aoyama, and Minoru Watanabe. An FPGA Connect6 Solver with a Two-Stage Pipelined Evaluation. In Proceedings of the International Conference on Field-Programmable Technology, pages 1-4, 2011.
[7] Kentaro Sano. SW and HW Co-design of Connect6 Accelerator with Scalable Streaming Cores. In Proceedings of the International Conference on Field-Programmable Technology, pages 1-4, 2011.
[8] Kizheppatt Vipin and Suhaib A. Fahmy. A Threat-based Connect6 Implementation on FPGA. In Proceedings of the International Conference on Field-Programmable Technology, pages 1-4, 2011.
[9] Tobias Ziermann, Bernhard Schmidt, Moritz Muhlenthaler, Daniel Ziener, Josef Angermeier, and Jurgen Teich. An FPGA Implementation of a Threat-based Strategy for Connect6. In Proceedings of the International Conference on Field-Programmable Technology, pages 1–4, 2011.
[10] I-Chen Wu and Dei-Yen Huang. A New Family of *k*-in-a-row Games. In Proceedings of the International Conference on Advances in Computer Games, pages 180–194, 2006.
[11] I-Chen Wu, Dei-Yen Huang, and Hsiu-Chen Chang. Connect6. ICGA Journal (SCI), 28(4):234–241, 2005.
[12] Atlys™ Board Reference Manual, Dec 2012. Rev. C.
[13] The 2011 International Conference on Field-Programmable Technology, 2011. http://www.cse.iitd.ac.in/ icfpt11/.
[14] The 2012 International Conference on Field-Programmable Technology, 2012. http://icfpt2012.blogspot.jp/.