# CASE STUDIES TO LEARN HUMAN MAPPING STRATEGIES IN A VARIETY OF COARSE-GRAINED RECONFIGURABLE ARCHITECTURES

Tika K. Malla

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

May 2017

APPROVED:

Gayatri Mehta, Major Professor
Kamesh Namuduri, Committee Member
Parthasarathy Guturu, Committee Member
Shengli Fu, Chair of the
Department of Electrical Engineering
Costas Tsatsoulis, Dean of the
College of Engineering
Victor Prybutok, Vice Provost of the
Toulouse Graduate School

Malla, Tika K. *Case Studies to Learn Human Mapping Strategies in a Variety of Coarse-Grained Reconfigurable Architectures*. Master of Science (Electrical Engineering), May 2017, 57 pp., 5 tables, 53 figures, 33 numbered references.

Computer hardware and algorithm design have seen significant progress over the years. It is also seen that there are several domains in which humans are more efficient than computers. For example in image recognition, image tagging, natural language understanding and processing, humans often find complicated algorithms quite easy to grasp. This thesis presents the different case studies to learn human mapping strategy to solve the mapping problem in the area of coarse-grained reconfigurable architectures (CGRAs). To achieve optimum level performance and consume less energy in CGRAs, place and route problem has always been a major concern. Making use of human characteristics can be helpful in problems as such, through pattern recognition and experience. Therefore to conduct the case studies a computer mapping game called UNTANGLED was analyzed as a medium to convey insights of human mapping strategies in a variety of architectures. The purpose of this research was to learn from humans so that we can come up with better algorithms to outperform the existing algorithms. We observed how human strategies vary as we present them with different architectures, different architectures with constraints, different visualization as well as how the quality of solution changes with experience. In this work all the case studies obtained from exploiting human strategies provide useful feedback that can improve upon existing algorithms. These insights can be adapted to find the best architectural solution for a particular domain and for future research directions for mapping onto mesh-and- stripe based CGRAs.

ACKNOWLEDGMENT

First and foremost, I would like to express my sincere appreciation to Dr. Gayatri Mehta for giving me the opportunity to do research and providing invaluable guidance throughout this research. It was a great privilege and honor to work and study under her guidance. Without her help and counsel, always generously and effortlessly given, the completion of this work would have been immeasurably more difficult.I would also like to express my sincere gratitude to Dr.Kamesh Namuduri and Dr.Parthasarthy Guturu for their time and consent to be on my thesis committee. During the course of this work, the constant association with the members of the Department of Electrical Engineering at University of North Texas has been most pleasurable. This thesis would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENT iii

LIST OF TABLES vi

CHAPTER 1 INTRODUCTION 1

1.1. Introduction 1

CHAPTER 2 BACKGROUND 3

2.1. UNTANGLED-The Mapping Game 3

2.2. Coarse-Grained Reconfigurable Architectures 3

2.3. Data Flow Graph Mapping 4

CHAPTER 3 RELATED WORK 5

CHAPTER 4 CASE STUDY-1: ANALYSIS OF STRATEGY IN MESH AND STRIPE ARCHITECTURE 8

4.1. Mesh Architecture 8

4.2. Stripe Architecture 18

CHAPTER 5 CASE STUDY 2- ANALYSIS OF STRATEGY IN RESOURCE CONSTRAINT ARCHITECTURE 22

5.1. Input Output (I/O) Constraint 22

5.2. Limited Connectivity (LC) Constraint 28

5.3. Dedicated Multiplier (DM) Constraint 33

CHAPTER 6 CASE STUDY 3- ANALYSIS OF STRATEGY IN VARIOUS REPRESENTATION OF THE SAME PROBLEM 36

CHAPTER 7 CASE STUDY 4- ANALYSIS OF STRATEGY FOR A GAME PLAYED

MULTIPLE TIMES. 50

CHAPTER 8 CASE STUDY 5- CORRELATION BETWEEN MOVE AND SCORE 52

8.1. Mesh architecture 52

8.2. I/O architecture 52

CHAPTER 9 CONCLUSION 53

BIBLIOGRAPHY 54

LIST OF TABLES


| | | Page |
|---|---|---|
| Table 6.1. | Top 10 Player's Feasible Solution | 38 |
| Table 6.2. | Top 10 Player's Average Grid Size | 39 |
| Table 6.3. | Graph Information (Easy Level) | 48 |
| Table 6.4. | Graph Information (Medium Level) | 48 |
| Table 6.5. | Graph Information (Hard Level) | 49 |

# CHAPTER 1

# INTRODUCTION

## 1.1. Introduction

Coarse-Grained Reconfigurable Architectures hold a promising future in hardware platform for high computation throughput, low cost, scalability, and efficient energy. One of the major challenges with CGRAs is placement and routing problem. To systematically tackle and solve the mapping problem, human computing abilities holds some unexplored potential. To drive more humans to engage in problem-solving exercise, games are considered to be an excellent platform in human computation. Over the years many games have been developed where computational tasks are solved by human power. Those games are developed in a way for humans to enjoy and at the same time perform computational tasks. To develop the dimensions in our work we analyzed a computer mapping game called UNTANGLED [2], [21] which utilizes human computation power where a player solves the important scientific problem of mapping or placement of logic blocks. The developers of the game believe that the next generation of mapping algorithm will demonstrate human characteristics consisting of pattern recognition, skill to learn from experience and recognition of creative solutions, thus suggesting directions for future mapping onto CGRAs. [22].

The game is used as a medium to provide all the required computational facilities to solve various problems in different application fields so that no deep knowledge of engineering is needed. Players participate in organizing most compact, visually appealing arrangements of circuit elements onto a grid. The circuit elements are represented as blocks in bold colors that are jumbled up. The player is required to untangle those blocks and arrange it in a compact manner while still following certain rules for the game. The game illustrates efficient crowd-sourcing of mapping problem where successful players have common strategies and can also improve upon existing algorithms [22].

Our case study revolves around how players strategies vary, as we give player puzzles in a variety of architecture styles in UNTANGLED. In the game, the styles are classified

into Mesh and Stripe architecture. We also analyzed players strategy when a constraint is added to the same Mesh and Stripe architecture. To further broaden our research for those architectures, the factors that were taken into consideration were visualization, average grid size, score, moves and effect of playing a puzzle in particular order. The remaining paper is organized as follows: Chapter-2 discusses game UNTANGLED, Coarse-Grained Reconfigurable Architectures and Data Flow Graph Mapping. Chapter 3 covers various related work. Chapter 4 presents Case Study -1 that reports the results and discusses player's strategies in Mesh and Stripe architecture. Chapter 5 discusses Case Study-2 that shows the analysis of strategy in resource constraint architecture. Chapter 6 covers Case Study-3 that reports how different visualization of the same problem can affect the quality of solution. In Chapter 7 we discuss Case Study-4 that analyzes a players strategy when a game is played multiple times. Correlation of moves and score for Mesh and Resource Constraint architecture is discussed in Chapter 8 followed by a conclusion.

# CHAPTER 2

# BACKGROUND

## 2.1. UNTANGLED-The Mapping Game

UNTANGLED is an interactive computer mapping game that has been developed to discover algorithms by making use of human intuition and their creativity to distinguish patterns and opportunities even in complex problems. The main purpose of UNTANGLED is to attract people from different domains and is developed focusing people who do not belong to any engineering domain or background. The problem is presented with different visualization in such a way that it is intuitive and challenging at the same time. The graphs consist of real algorithms and by arranging those blocks players are unknowingly mapping onto various chip architectures that can be fabricated in silicon. Solving those graphs, they contribute to the advancement of next-generation portable devices that are critical in various fields like health, aerospace, portable multimedia etc. An efficient solution to this problem is to make a grid more compact and consume less energy. UNTANGLED is easily accessible online and can be implied as a great platform to attract students for engineering education. With its interactive architecture players can use their intuition and creativity to discover strategies that yield more compact designs and are exposed to real-world chip design problems.

## 2.2. Coarse-Grained Reconfigurable Architectures

One of the fundamental challenges in modern microelectronics industry is the continuous demand for high-performance and high-power efficiency. With an increasing demand for low cost and low energy implementation, Coarse-Grained Reconfigurable Architectures are becoming an attractive platform with a promising future because they can be programmed to execute different instructions with very little power and increased hardware efficiency. CGRAs typically consist of an array of processing elements (PEs) that are interconnected by a 2-D grid. A PE consist of a few registers and an arithmetic logic unit and each PE can be programmed to perform different tasks.

2.3. Data Flow Graph Mapping

Data flow graphs are mostly used in hardware design to provide a good description. Dataflow graph consist of a directed graph where basic operations are represented by the nodes which needed to be mapped onto the PEs and the dependencies and data movement among the operations which are basically communication links of the interconnect are represented by the edges. Every node precisely corresponds to a hardware unit that is assigned on the chip therefore it is easy to translate those nodes into a hardware circuit.

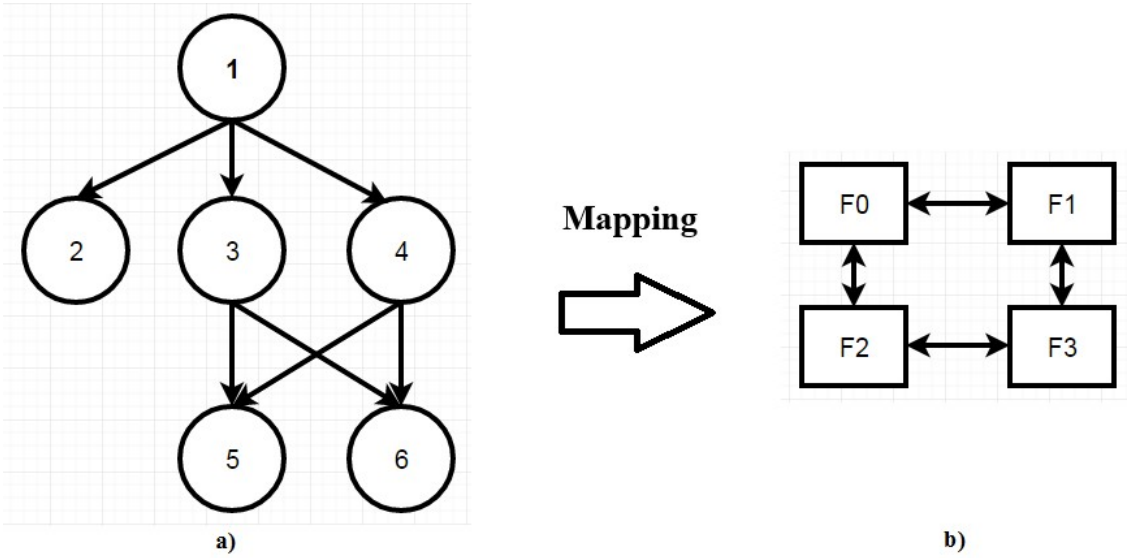



FIGURE 2.1. **a) Data Flow Graph; b) A 2x2 CGRA**

# CHAPTER 3

# RELATED WORK

Our research draws on previous research related to, Coarse-grained reconfigurable architectures, Data flow graph mapping, human computing, games with a purpose, crowd-sourcing, visualization and problem-solving. Due to large raw computation capabilities with low cost/energy implementation, coarse-grained reconfigurable architectures are becoming promising alternatives between ASICs and FPGAs.Multiple hardware implementations exist such as [3], [7], [24], [28]. CGRA architecture such as ADRES [23], MorphoSys, [19], and Silocon Hive [27] etc., have been proposed. Even though the possible performance and power efficiency of a CGRA is very high, one of the major challenges is mapping and routing in a compiler. To solve these hard problems, current CGRA compilers use search based heuristic to map applications to the CGRA, however the quality of mapping is not good [13]. A precise and general formulation of the application mapping problem on a CGRA is presented in [13]where a global heuristic called EPIMap is designed to transform the input specification to an Epimorphic equivalent graph that satisfies the necessary conditions for mapping on to a CGRA thus increasing the computation time. An architecture- adaptive CGRA mapping algorithm called SPR (Schedule, Place, Route) is proposed in [11]. Another algorithm called REGIMap is described in [14] which optimizes the mapping dependencies on the usage of registers. With CGRAs the challenge of mapping is critical, and that is where we believe humans are more efficient than computers. In several fields like image recognition, image tagging, etc., humans have excelled computers. Human-based computing was defined by Kosorukoff in [18] where human enhanced genetic algorithm is explained. Comparison of the emerging social-game-based human computation systems based on the game structure is presented in [8].

There are various games with a purpose that provide a great amount of meaningful information to their audiences with visualization tools. Game with a purpose emerged from the human computation technique that depends on the collaboration of game players to solve

problems. Several concepts of Game with a Purpose has been well discussed in [30], [33], [15] which explains about the games that are developed for amusement value but at the same time are designed in such a way that players contribute intellectually towards a problem. Human computing approach has been followed in a variety of applications. To harness human potential an online logic puzzle called FunSAT is developed for SAT solving. It uses visual pattern recognition skills, abstract perception and intuitive strategy skills of humans to solve complex SAT instances [8]. Advantages of human computing to index web pages in presented [32]. The popularity of computer games and how a game with a purpose approach holds some unexplored potential is explained in [4]. It presents a Game with a Purpose based system for verification of automatically generated mappings. CAPTCHA is a computer generated challenge response that differentiates humans from computers using a common sense problem [31]. Foldit is yet another online puzzle game to fold structure of selected proteins as well as use tools provided in the game [1]. Trouble hunters is another game where participants make up commissions of inquiry to analyze the failed cases of software development [12].An external automatic defibrillator game based learning was developed by providing scenarios of cardiac arrest where the user is required to apply a CPR and use an AED to save the patient [25]. jLegends another online game is developed to train programming skills. [29]. Since for a game which is played by all different kinds of people, it allows a crowd of people to collaborate on solving real-life problems and implement a variety of ideas. Several purposes can be solved by making use of innovative technology and gathering many knowledge sources with a different level of experience and expertise. Through Crowdsourcing anyone is allowed to solve a problem, thus both specialist, as well as the person with no background in that domain, can participate in solving the problem [17], [16]. Areas like creative design which does not require a specific answer to a problem, crowdsourcing has proven to be very successful [5]. However one of the main challenges in crowdsourcing is the need to design user interfaces that can appeal and retain the collaboration of numerous people [26], [10]. Thus to increase user participation game design techniques are considered to be an excellent platform, as games have successful strategies to grant enjoyable experience [9]. It is argued in [20] that if

games can engage players to solve fictitious problems of a virtual world, these same players could act together to solve real world problems. Gamification requirement and challenges to improve user experience in crowdsourcing systems is discussed in [6].Mapping problem in CGRAs can be solved systematically with a proper problem formulation. In summary the goal of this paper is to analyze human strategy in different architectures and observe how humans come up with better mapping architecture. To achieve that a computer mapping game called UNTANGLED was used as a medium.

# CHAPTER 4

# CASE STUDY-1: ANALYSIS OF STRATEGY IN MESH AND STRIPE ARCHITECTURE

## 4.1. Mesh Architecture

CGRAs can be arranged into two categories, linear array/stripe-based and mesh based, depending on how the functional units are arranged. Linear array/stripe-based category comprises one or more linear arrays of elements that are connected with a full or partial crossbar interconnect between rows. Whereas in mesh category, the arrangement of functional units are done in a two-dimensional array that has horizontal and vertical interconnect which can support a nearest neighbor, a nearest neighbor with hops and hierarchical connections. The architectures in game UNTANGLED are designed in similar fashion. Each architecture has difficulty level ranging from easy to challenging. Levels (benchmark) consist of DFG obtained from real algorithms, that are to be placed onto a given architecture. Easy1, Easy2 and Easy3 benchmarks are obtained from Sobel Edge Detection, Laplace Edge Detection and GSM Channel Encoder respectively. Medium1 and Medium2 benchmarks are obtained from Adaptive differential pulse-code modulation (ADPCM) Decoder and Adaptive differential pulse-code modulation Encoder respectively. Furthermore Hard1 and Hard2 benchmarks are obtained from MPEG-II Decoder Inverse Discrete Cosine Transform technique (Row) and MPEG-II Decoder Inverse Discrete Cosine Transform technique (Column) respectively. This section highlights the collection of six various architectures within the Mesh and Stripe-based classes in UNTANGLED and different mapping strategies that were followed by players. Both classes also include architectures with resource constraints. The architectures are as follows.

(1)8Way. In this mesh architecture nodes can connect to any of their eight neighbors. (2) 4Way. This architecture is same as 8Way except the diagonal connections are not taken into consideration. (3) 4Way1Hop. This architecture is interpretation of 4Way, where it is allowed to skip one node both horizontally and vertically. (4) 4Way2Hop. In a similar

fashion as 4Way1Hop, here it is allowed to skip two nodes both horizontally and vertically.

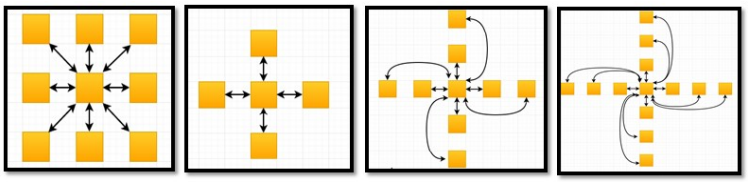


FIGURE 4.1. **Interconnect patterns. Left to right: 8Way, 4Way, 4Way1Hop, 4Way2Hop**

When selecting a scope for the case studies described in this work, our primary goal was to analyze the data to extract different patterns from each player to study their behavioral patterns and their approach towards the solution. One of the primary motivation was to identify common player strategies. The following figures shows interesting strategies followed by successful players to reach their optimum solution in **Mesh Architecture**.

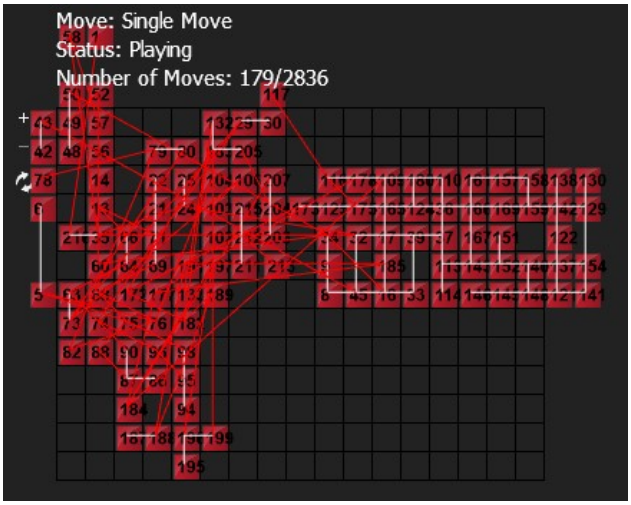



FIGURE 4.2. **Solving 4Way2Hop, Extreme1**

This players strategy was to separate the graph from right to left instead of top to bottom or bottom to top which serves as one of the most common strategies amongst most players. Player solved the right half of the graph first thus going towards the left from there, however, this approach seemed more rigorous as compared to the top to bottom or bottom to top approach as shown in Fig. 4.2

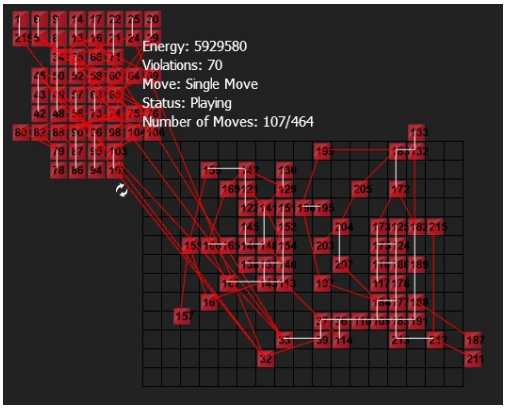



FIGURE 4.3. **Solving 4Way2Hop, Extreme1. Isolating the densest area to the outside of the graph.**

Experienced players would isolate the densest area or the nodes with higher number of connections to the outside of the graph then bring the solved sub-clusters into a compact arrangement. Fig.4.3

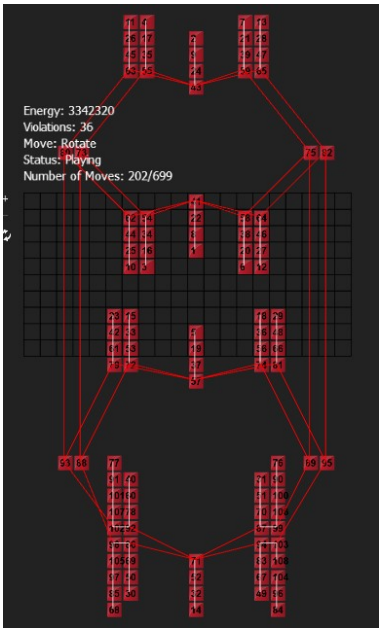
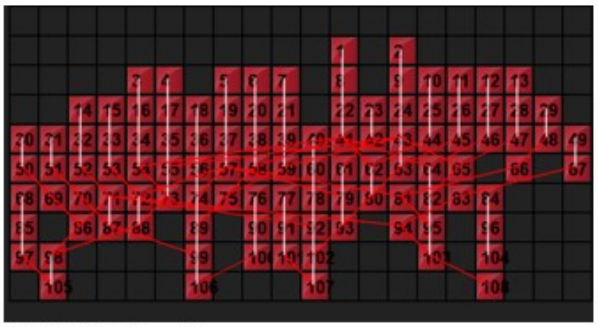
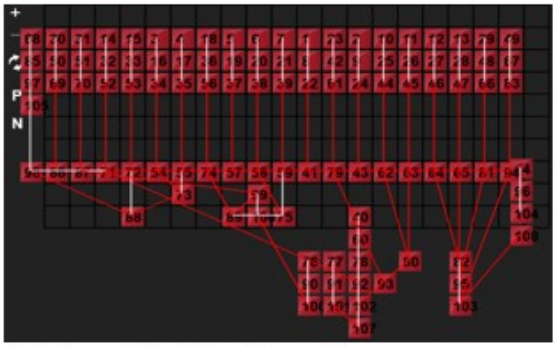



FIGURE 4.4. **Solving 4Way1Hop, Extreme3. Organizing the nodes in similar sub-clusters.**

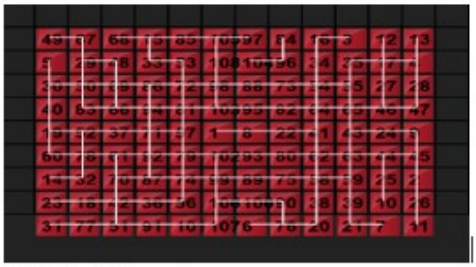
Initially, player started by spreading the blocks over the edges of the graph then organized them in similar clusters. Fig.4.4



(a) Initial Graph



(b) Player identified the pattern



(c) Final Graph

FIGURE 4.5. **Solving 4Way2Hop, Extreme3.**

While analyzing these games we observed that some of the experienced players would just look at the graph and recognize the pattern. Fig.6. Figuring out the pattern in the beginning of the game resulted in better graph. Several swapping of clusters and rotate move was observed all the along the game. Fig. 4.5

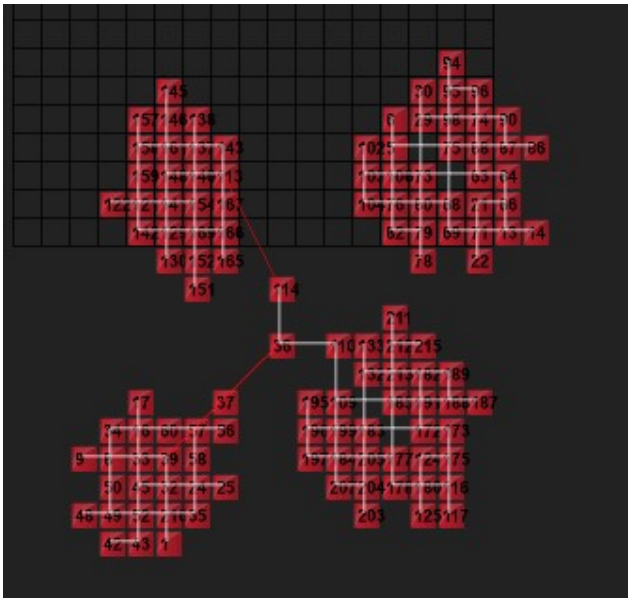


FIGURE 4.6. **Solving 4Way1Hop, Extreme1. Solving in small sub-clusters.**

Some experienced players would divide the whole graph into small sub-clusters and solve it, then make the connections as small as possible one at a time to make the arrangement more compact. Fig. 4.6

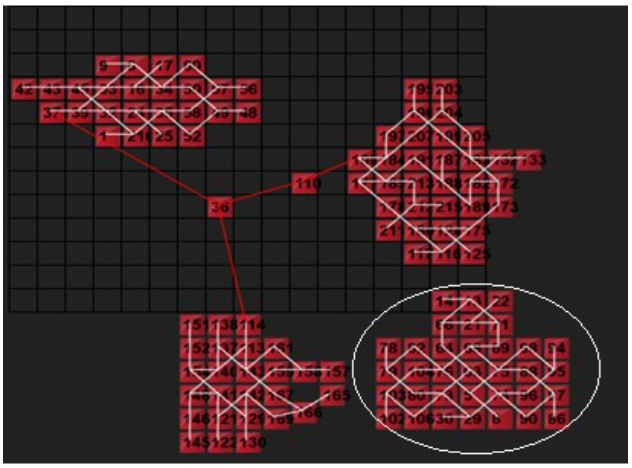


FIGURE 4.7. **Solving 8Way, Extreme1. Identifying hidden sub-cluster.**

Some players would also recognize that identifying the small sub-graph that are hidden in the big graph makes the graph less complicated and easier to solve. Fig. 4.7

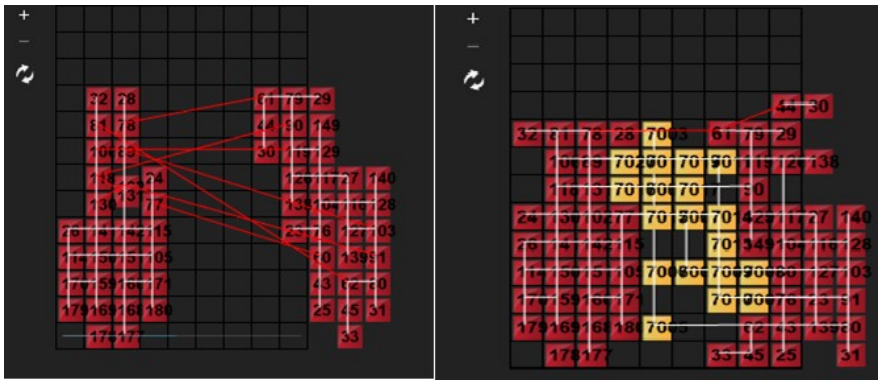
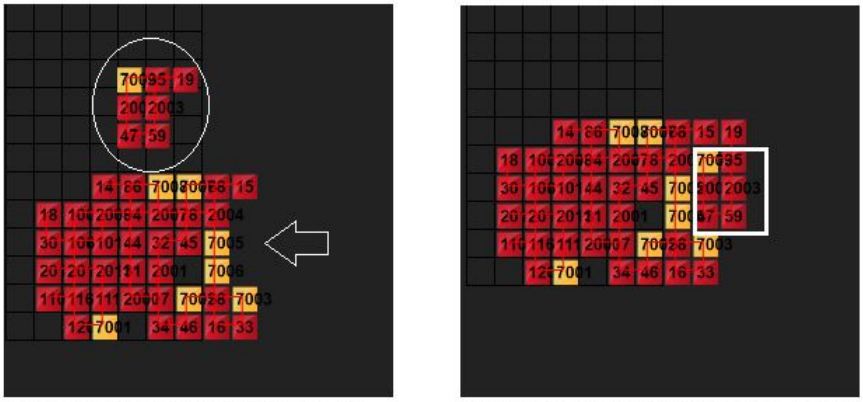


FIGURE 4.8. **Solving 4Way2Hop, H2.**

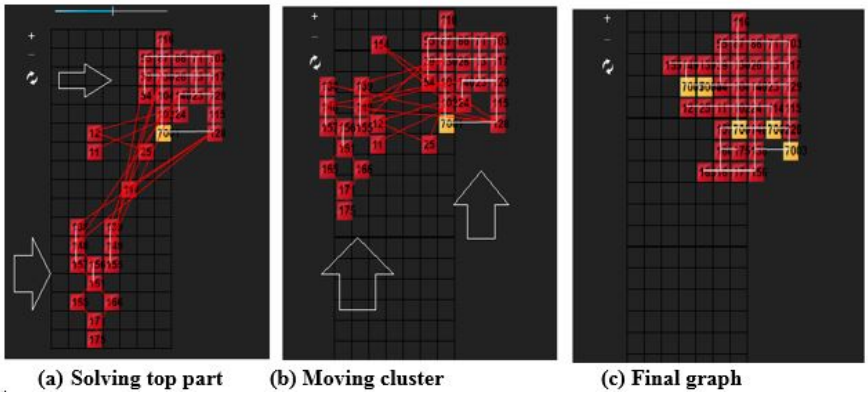
Fig.4.8 Shows how player solved the graph in two halves and once most of the violations were solved, to solve for the remaining violations player added pass through nodes (yellow). With this strategy player was able to solve for all the violations, however, the energy consumption was high.



(a) Identifying the hidden subcluster (b) Created the space for that subcluster

FIGURE 4.9. **Solving 4Way, M1.**

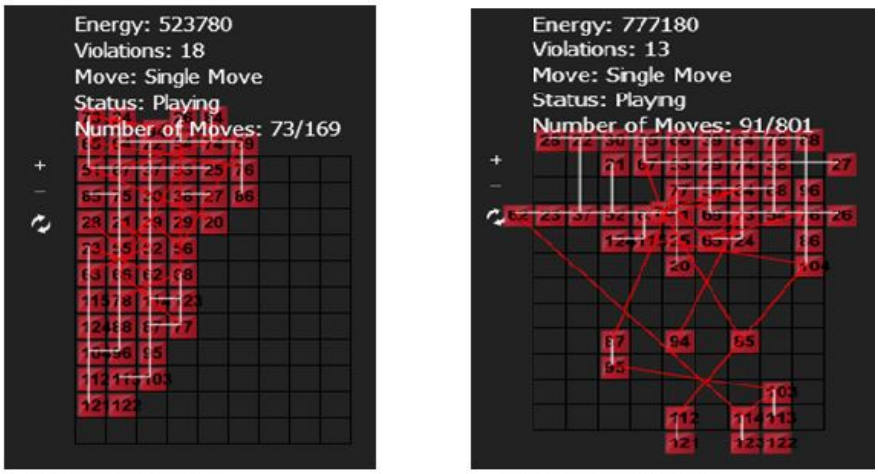
Some players were able to recognize and isolate the small graph and combine it with the big graph. Fig.4.9



(a) Solving top part (b) Moving cluster (c) Final graph

FIGURE 4.10. **Solving 4Way2Hop, M2.**

Fig.4.10 shows the solution process of the game where player solved for the top part first then decided to move the whole cluster from the bottom to the left of the graph then solve for the violations. With this strategy the player was able to solve for all the violations, however, in the end the player made use of several pass gates.




(a) First attempt (b) Second attempt

FIGURE 4.11. **Solving 4Way2Hop, H1.**

We also analyzed players multiple attempts for the same architecture to get more quantitative understanding of successful players. In the first attempt player started by working on the bottom blocks and solved them by organizing them in vertical manner. Player was not able to solve for all the violations and was left with few violations. Fig. 4.11 (a) For the same game, in the second attempt player scored highest when the graph was solved in a horizontal manner. With this strategy the player was able to solve up to 1 violation. Fig. 4.11 (b)

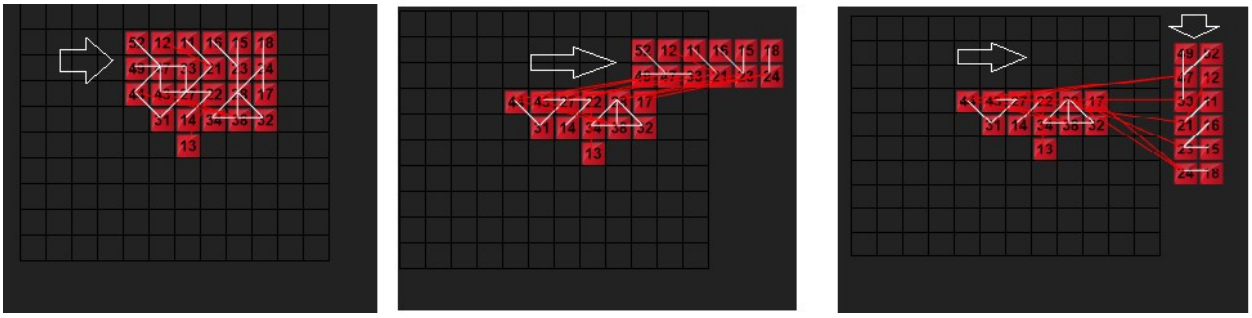


FIGURE 4.12. **Solving 8Way, E1 Rotating the whole cluster**

Another strategy for trying to organize the graph was to rotate the whole cluster then solve for the violations. Fig. 4.12

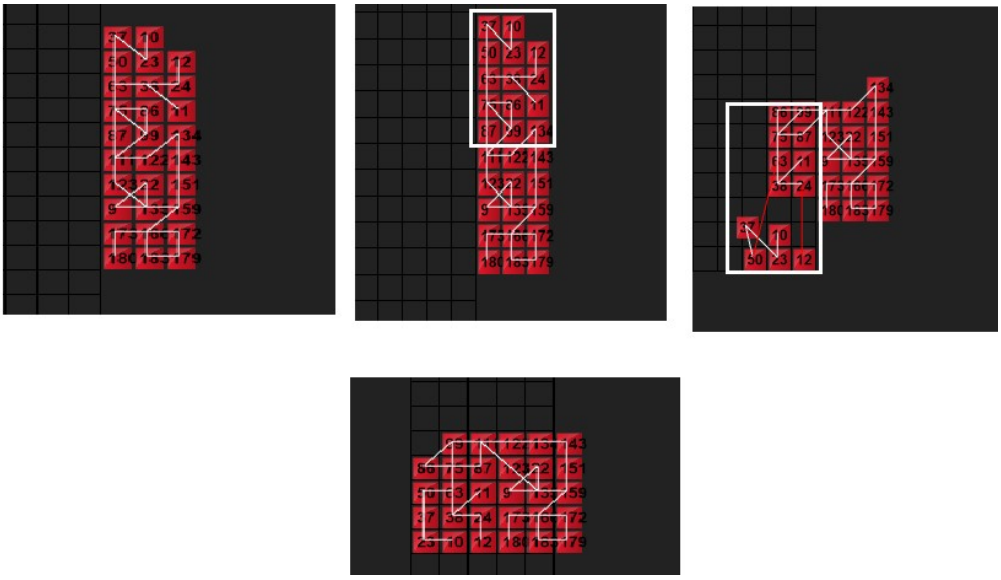


FIGURE 4.13. **Solving 8Way, E3 Optimizing the solution.**

Some experienced players would also focus on optimizing the graph. Fig. 4.13 To create compact graph player brought an entire cluster of nodes to the bottom of the graph. With this strategy, the player was able to make the graph more compact and with less energy.

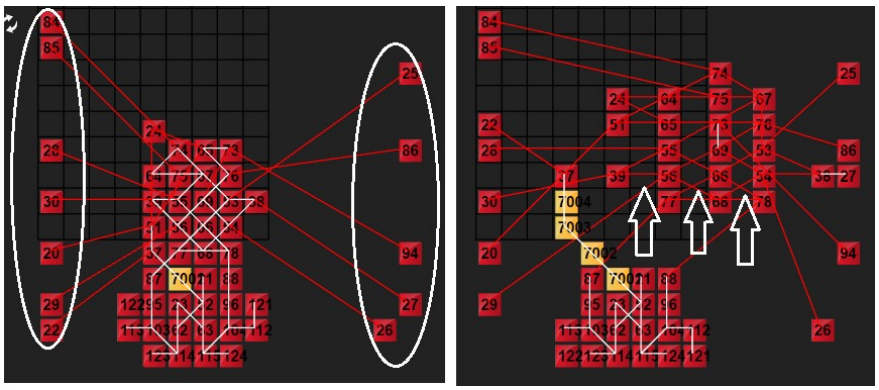


FIGURE 4.14. **Solving 8Way, H1.**

Experienced players would identify the nodes with less connectivity, then spread them on the edges of the graph followed by creating space for them. Fig.4.14

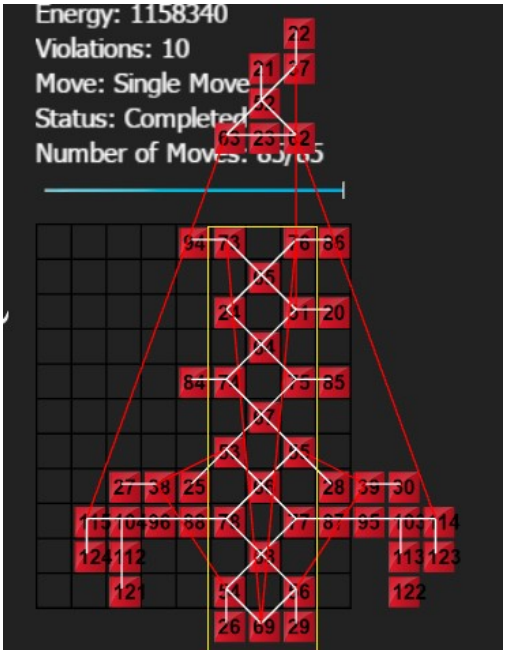



FIGURE 4.15. **Solving 8Way, H1. Diamond shape pattern.**

Some experienced players would try to organize the graph in a particular manner. An example of this is shown in Fig.4.15 Where the player tried to make the connections short by placing the nodes in diamond shape pattern. With this particular strategy the player was able to solve up to 10 violations, however, the energy level was too high.

4.2. Stripe Architecture

(1) Stripe. The stripe version of UNTANGLED simulates the data in one direction that is from top to bottom and nodes are arranged in a horizontal manner (stripes).That direction also represents how parent blocks should be oriented with respect to child blocks. Each of the stripe is connected to the bottom stripe using a full crossbar interconnect. It is assumed that inputs can be given to any ALU, but outputs must be appointed off chips, thus all nodes below output nodes are blocked from executing any computation. (2) Stripe dedicated routes (DR). This architecture follows the same convention as Stripe architecture, however, it also consists of dedicated routes together with stripe. Dedicated routes are green columns where only pass-through gates are allowed to be positioned.

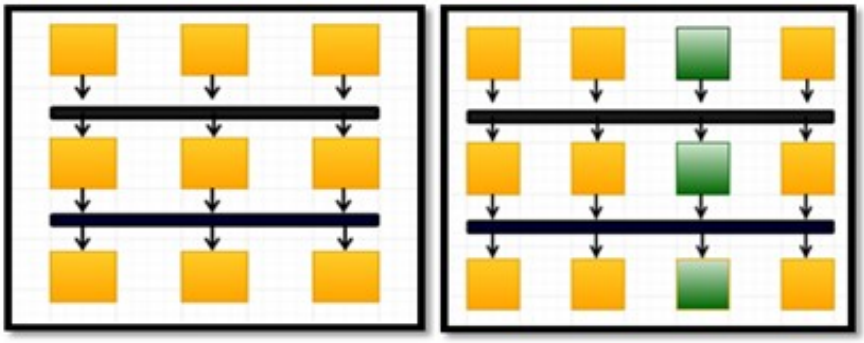
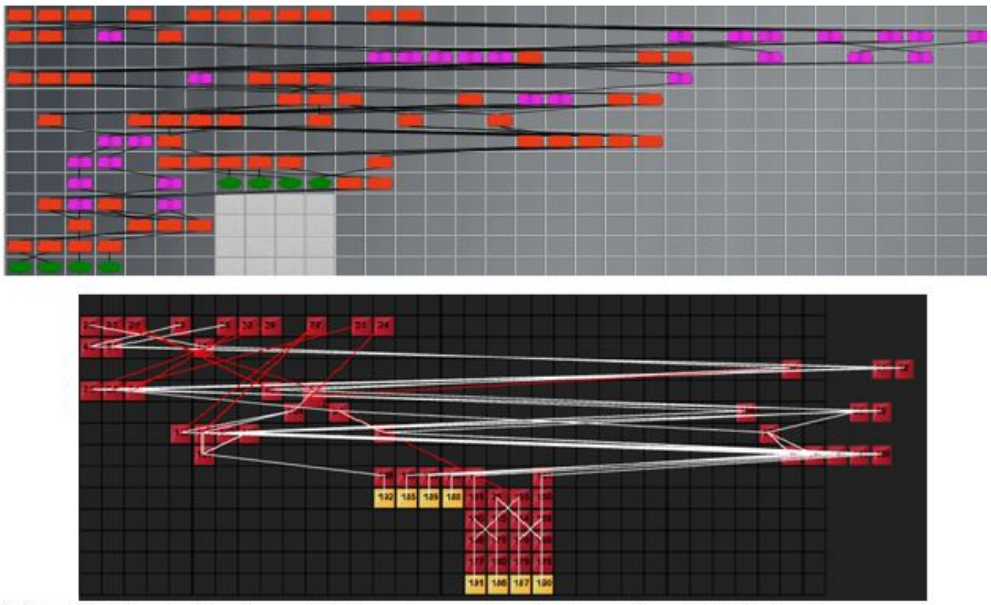


FIGURE 4.16. **Interconnect patterns. Left to Right: Stripe, Stripe DR.**

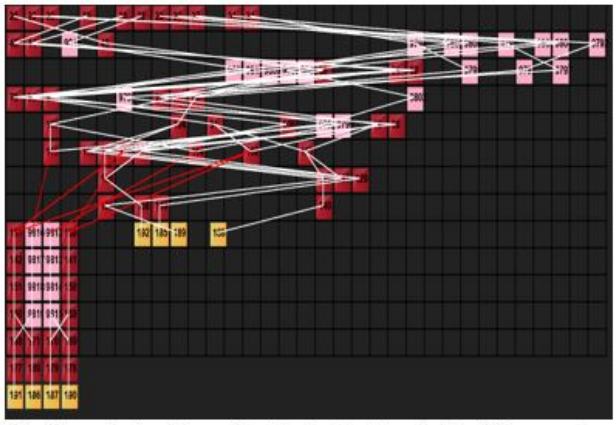
The following figures shows interesting strategies followed by players to reach their optimum solution:



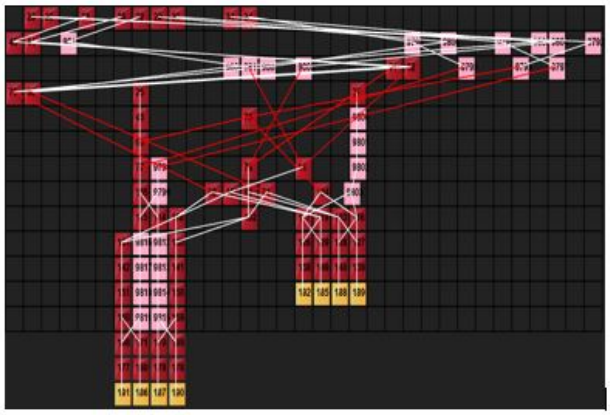
(a) Initial Graph (Top) (b) Moved an entire cluster to the right (bottom)

FIGURE 4.17. **Solving Stripe, H2.**

Experienced players would initiate by removing all the pass gates first. Once all the pass gates are removed, the player would move an entire cluster of nodes to either the right side of the graph or the left side of the graph then followed by solving towards the bottom or top of the graph. Furthermore, bringing the parent nodes from left and right side of the graph towards the center is also done. Fig.4.17



(a) **Organizing the nodes to the bottom left of the graph**



**(b) Solving in subclusters**

FIGURE 4.18. **Solving Stripe, H2**

Players would incorporate the same strategies as performed in mesh architecture for example here player started at the bottom of the graph and started shifting the nodes within the row. Then as the game progressed player started making the connections short by placing the nodes in two sub-clusters. Fig.4.18 In Stripe architecture it was observed that experienced players would initiate by making all the connections short by performing frequent swaps and shifting of nodes within a row. Once most of the connections are short, towards the end of the game the focus was on removing the pass gates. Some players would also initiate by making the connections short, and while making the connections short the players also remove the pass-gates in the process. Solving for the graph in sub-clusters seemed to be the players key strategy. Once most of the violations were solved in sub-clusters, the player would move the sub-cluster to the top or bottom of the graph, placing

the child nodes as close as possible to parents. In Stripe-DR similar strategies were observed as mentioned above, moreover experienced players would also initiate by placing all the pass through nodes on the dedicated routes first, then solving the rest of the graph.

# CHAPTER 5

# CASE STUDY 2- ANALYSIS OF STRATEGY IN RESOURCE CONSTRAINT ARCHITECTURE

Both Mesh architecture and Stripe architecture also include architectures with resource constraints where a player has few restrictions that must be taken into account in order to achieve the optimum solution.

## 5.1. Input Output (I/O) Constraint

The I/O resource constraints are as follows. (1) 4Way1Hop I/O. This architecture is the generalized form of 4Way1Hop where a player is restricted to position the input-output nodes (blue nodes) on the peripheral of the graph thus all nodes outside of input-output nodes are blocked from executing any computation. (2) 4Way2Hop I/O. This architecture also imposes similar restrictions as 4Way1Hop. (3) 8Way I/O. Same restrictions are applied in this architecture as mentioned above.

The following figures show interesting strategies followed by players to reach their optimum solution when the game is presented with constraints.

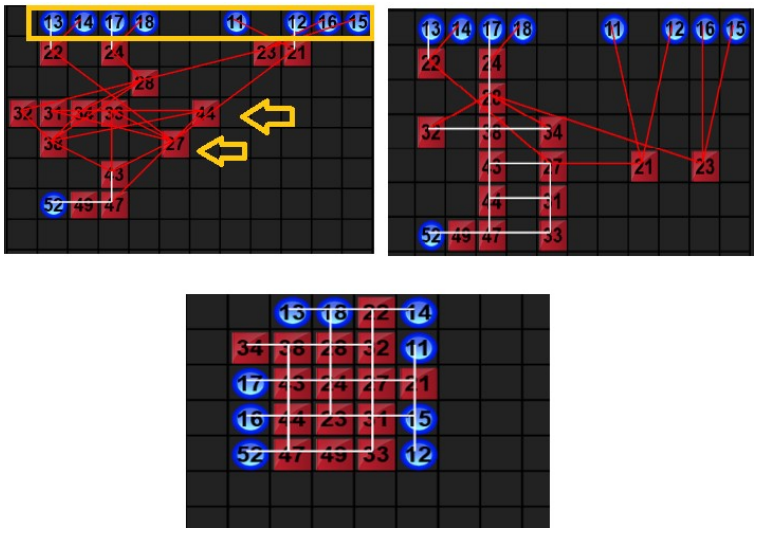
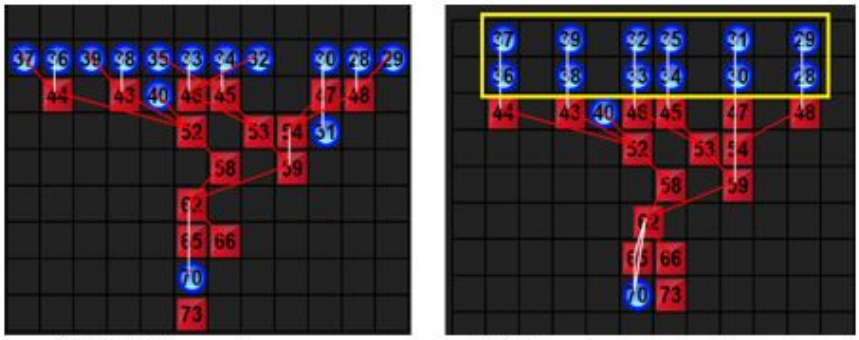


FIGURE 5.1. **Solving 4Way1Hop I/O, E1. Focusing on internal violations**

Most players would initially focus only on the red nodes and leave all the I/O nodes

along the perimeter of the graph. The player would sideline blue nodes till most of the internal violations were solved while constantly trying to make space for those nodes. Some experienced players would all along create an interior blank space for the interior nodes and pivot the nodes to fill those spaces. Fig.5.1 shows the solution process.



(a) Initial graph (b) Player's approach

FIGURE 5.2. **Solving 4Way1Hop I/O, E2.**

In I/O architecture, it was also observed that typically a player always initiates by engaging on the red nodes. However, some player would place all the input-output nodes to the top of the graph, then work on the violations, Fig. 5.2 (Normally, all the players placed those blue nodes at the bottom)

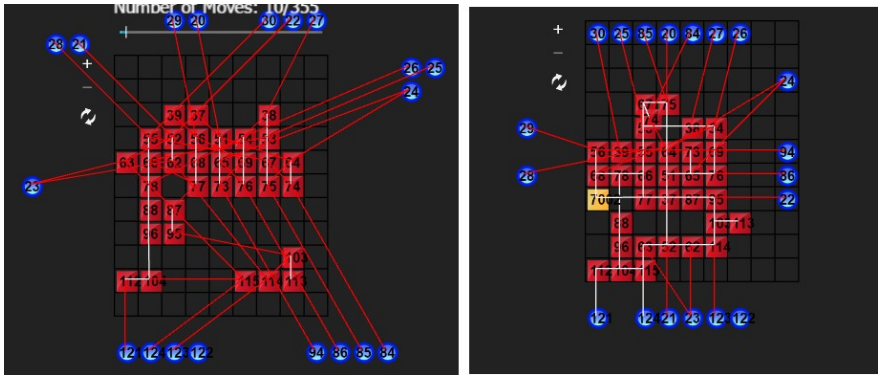


FIGURE 5.3. **Solving 4Way1Hop I/O, H1. Spreading the input-output nodes to the outside of graph**

Some players would also initiate by widening the input, output nodes on the peripheral of the graph, then placing the red nodes to the bottom of the graph. Fig. 5.3 shows the approach.

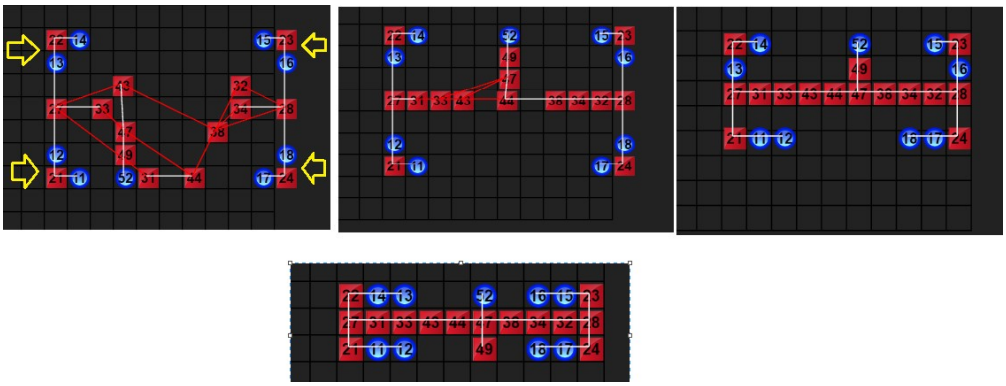


FIGURE 5.4. **Solving 4Way1Hop I/O, H1.**

Another strategy that was observed was positioning the nodes in a similar manner on both sides of the graph, i.e. the left and right in the similar manner, then solving for the nodes in the center of the graph as shown in Fig. 5.4

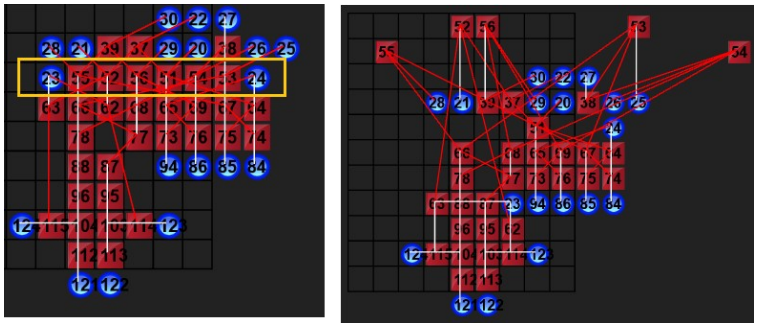


FIGURE 5.5. **Solving 4Way2Hop I/O, H1. Isolating high-degree nodes.**

Players would also isolate the high-degree nodes (second row) to the outside perimeter and make the connections short towards those nodes. Fig. 5.5

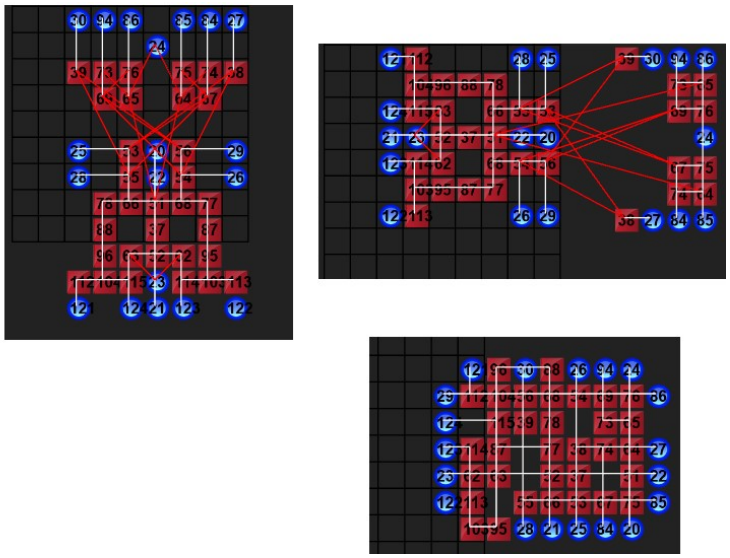


FIGURE 5.6. **Solving 4Way2Hop I/O, H1. Rotating the entire graph.**

Another strategy that served effective for some players was rotating an entire graph as shown in Fig. 5.6 . Player initiated with the top to bottom to strategy and later rotated the entire graph. Rotating the entire graph turned out to be very productive as it made the graph look less complex and making the connections short was much easier than before as a player simply made the connections short by taking the nodes from left to right.

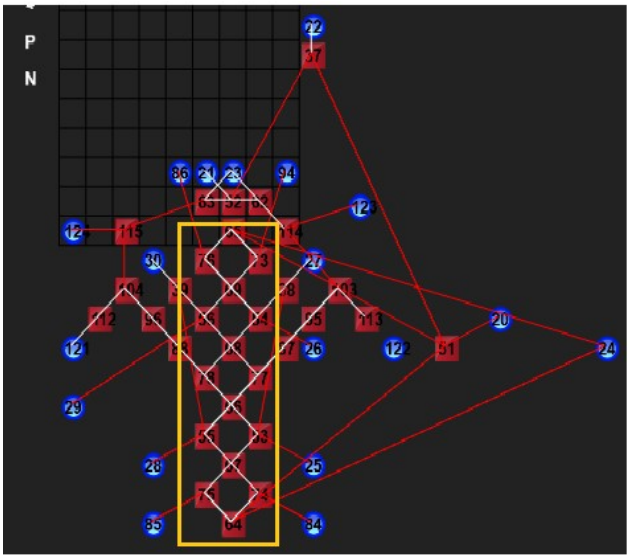



FIGURE 5.7. **Solving 4Way2Hop I/O, H1**

This strategy was observed in a mesh architecture also where the player initiates by placing the nodes from the top to the bottom of the graph and the focus would be on untangling the densest part of the graph. The player was seen to be untangling the graph and placing the nodes in a diamond shape pattern while still making sure that I/O nodes are on the outside perimeter. Fig. 5.7 .

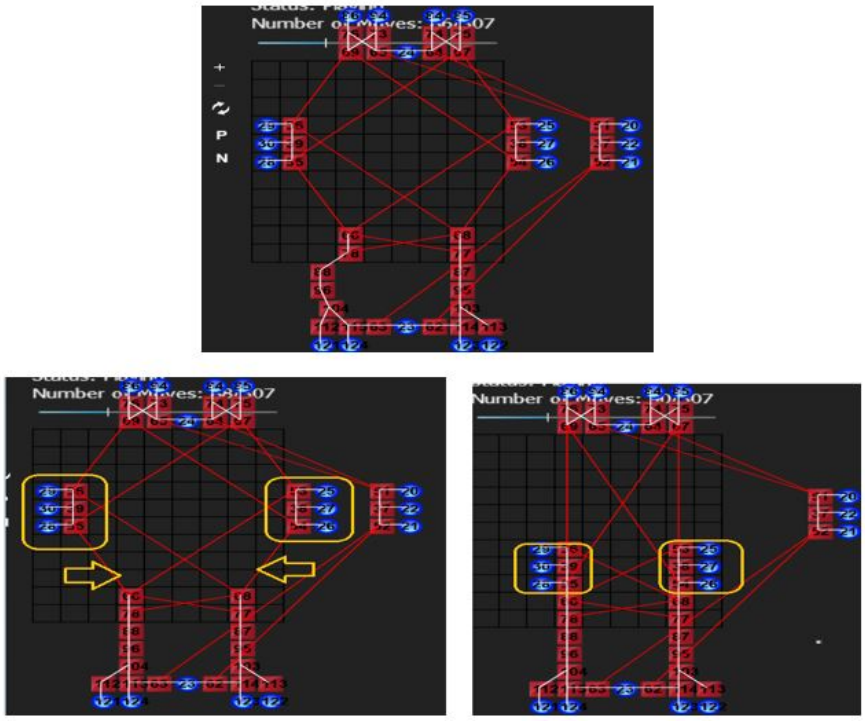


FIGURE 5.8. **Solving 8Way I/O, H1. Separating the graph into familiar sub-clusters.**

Fig. 5.8 .shows how player focused on solving for the graph in sub-clusters then combined them together at the bottom of the graph. As the game progressed, we observed that player separates the graph into familiar sub-clusters and positions them in a similar manner while combining the sub-clusters. Every move that was made on the left side of the grid, similar move was made on the right side of the grid. With this strategy player was able to solve for most of the violations.

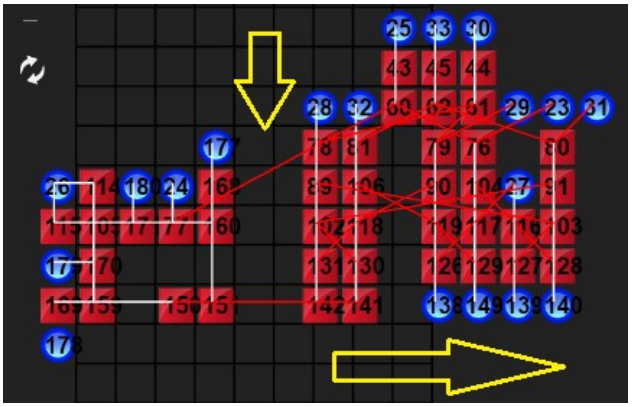


FIGURE 5.9. **Solving 4Way1Hop I/O, H2. Creating space by moving the graph to the right.**

As the game progress, some experienced players would also plant the nodes to the left and create space all along for the nodes. Fig. 5.9 shows how the player is creating space. That exact same strategy was followed all along, to solve for most of the violations. The player would identify the opportunity to fit the cluster from the right in that blank space. In the end the player used several pass gates to solve for all the violations.

In general, in an I/O architecture player would initially focus only on the red nodes and leave all the I/O nodes along the perimeter of the graph. Few experienced players would sideline blue nodes till most of the internal violations are solved while constantly trying to make space for those nodes. Players would also initially focus on placing the highly connected nodes to the left or right side of the graph, then making the connections short by bringing the child nodes closer to them. For some players rotating the entire graph was very effective as it makes the graph looks less complex thus making it easier to approach.

## 5.2. Limited Connectivity (LC) Constraint

(1) Stripe LC. This architecture is Stripe with limited connectivity. The data simulates only in top to bottom direction, that direction also represents how parent blocks should be oriented with respect to child blocks, thus applying LC constraint. Therefore, it follows the same convention as Stripe, however, is also restricted with a limited connectivity as it is a violation if a child node is more than two columns away from the parent node. (2) Stripe

DR-LC. This version of UNTANGLED is generalized form of Stripe LC with DR where only pass-through gates are allowed to be placed.

The following figures show interesting strategies followed by players to reach their optimum solution when the game is presented with constraints.

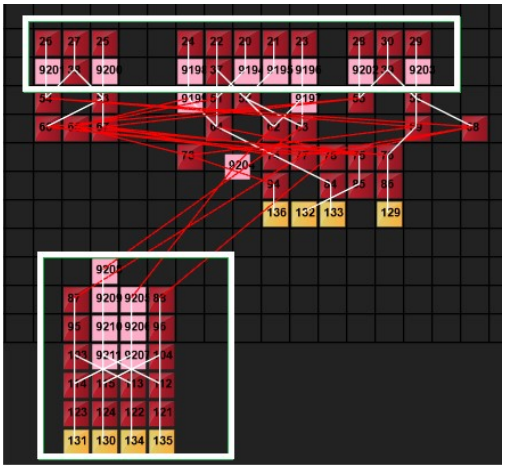



FIGURE 5.10. **Solving Stripe-LC, H1.**

One of the strategies was to initiate from the top of the graph. Once most of the parent children nodes were as close as possible on top, the player did the same at the bottom of the graph. Thereafter the player attacked the dense part of the graph. Several swapping of clusters were done to make the right connections. Fig. 5.10

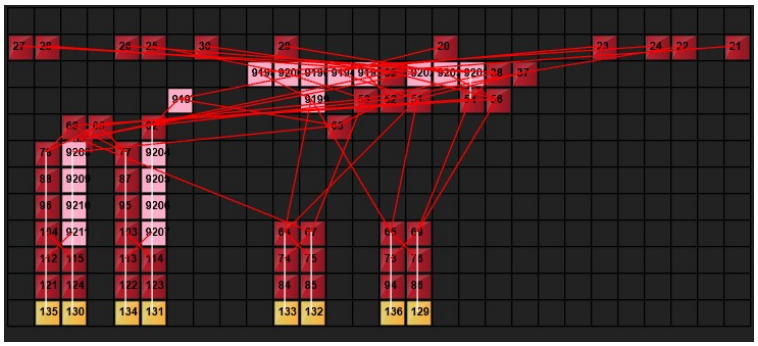


FIGURE 5.11. **Solving Stripe-LC, H1.**

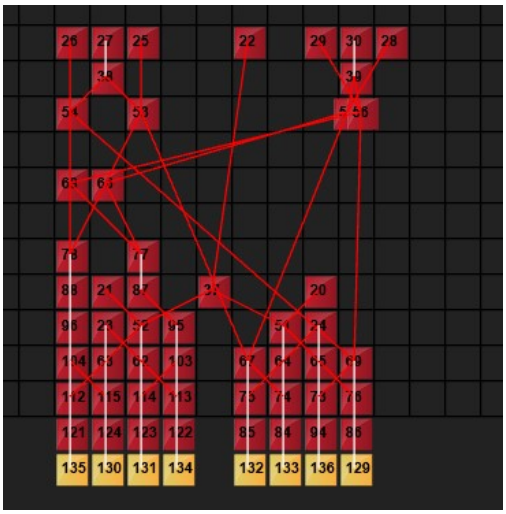


FIGURE 5.12. **Solving Stripe-LC, H1.**

Another interesting strategy that was observed was once most of the parent and child nodes were as close as possible Fig. 5.11, player would remove all the pass gates. Fig. 5.12.

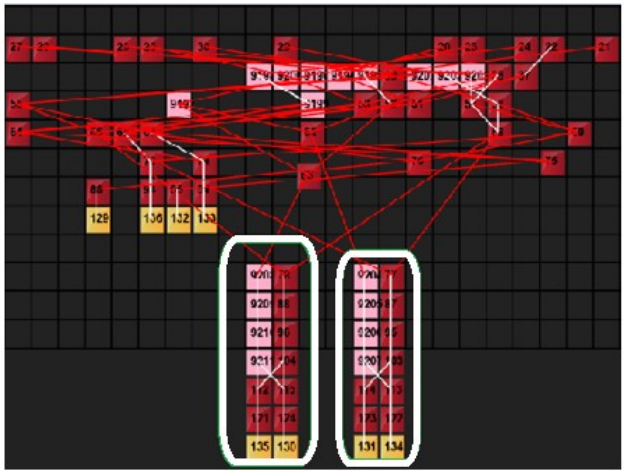


FIGURE 5.13. **Solving Stripe-LC, H1.**

This strategy has been successful in other architectures also, where the player would organize the nodes to the bottom of the graph in an identical manner. Fig.5.13.

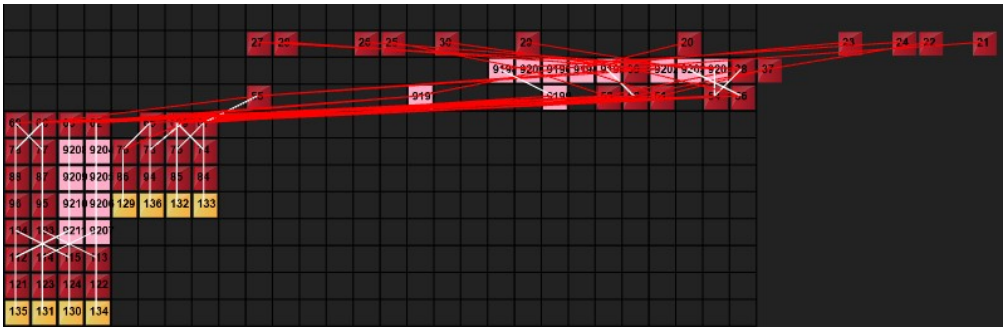


FIGURE 5.14. **Solving Stripe-LC, H1.**

Another strategy that was successful was shifting of nodes to one side of the graph to shorten edge lengths. Fig.5.14.

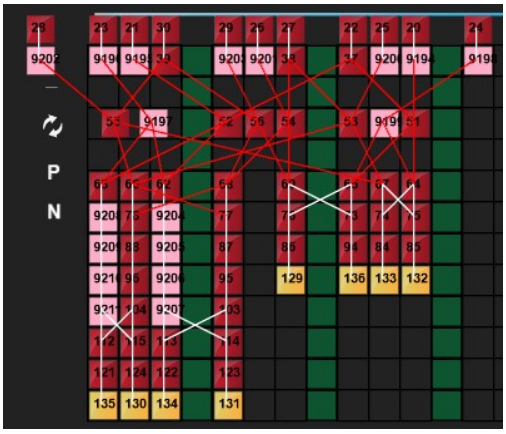



FIGURE 5.15. **Solving Stripe DR-LC, H1.**

Moving the nodes within the same row one at a time followed by swapping of clusters vertically served as a good strategy amongst few experienced players in this architecture. Fig.5.15.

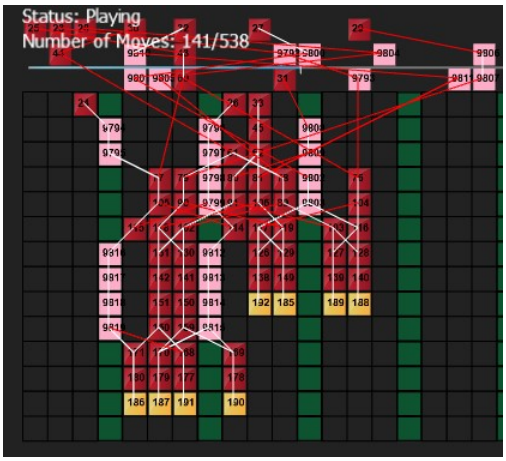



FIGURE 5.16. **Solving Stripe DR-LC, H2.**

Initiating by placing most of the pass gates in DR columns together with bringing the parent nodes closer to children nodes was yet another successful strategy in this architecture. Fig.5.16.

5.3. Dedicated Multiplier (DM) Constraint

(1) 4Way1Hop DM. This architecture is 4Way1Hop with Dedicated Multiplier which can only be placed in Dedicated Routes (DR). DR are arranged in vertical columns in alternate fashion within which only DM can be placed, unlike Stripe-DR which allows only pass through gates to be placed. (2) 4Way2Hop DM. 4Way2Hop with Dedicated Multiplier follows the same rule as mentioned above.

The following figures show interesting strategies followed by players to reach their optimum solution.

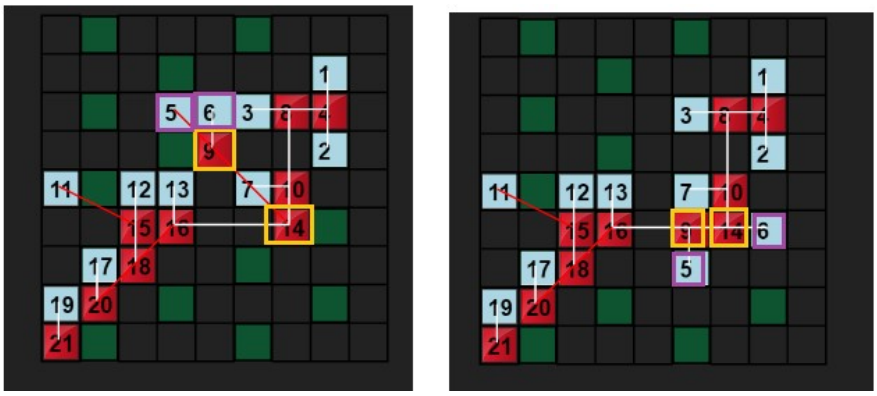



FIGURE 5.17. **Solving 4way2hops DM Pattern-1 E1.**

Some players would place the red nodes on the graph such that there are dedicated routes for the connected DM nodes to fit in. For example, observe nodes 5, 6, 9 and 14 which are connected to each other in Fig.5.17 (Left). So as a player made the connections short, player positioned the red nodes such that DM nodes are also satisfied on DR.

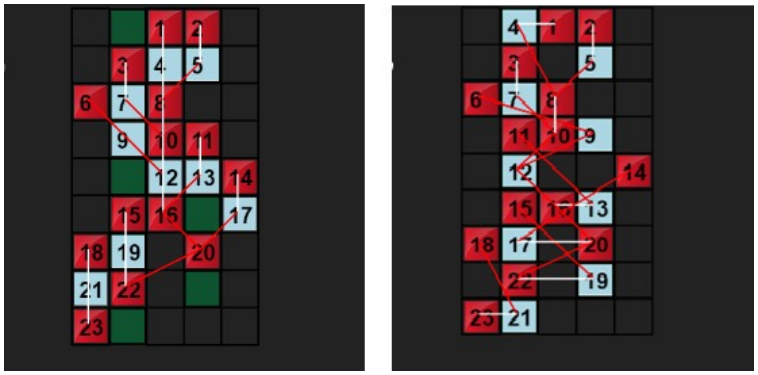


FIGURE 5.18. **Solving 4way2hops DM Pattern-1 E2.**

We also observe that some players initial strategy would be to place all the DM nodes on the dedicated routes, thereafter worked on satisfying the violations of red nodes. Fig.5.15. The same strategy was also seen for solving big graphs. Fig.5.18.

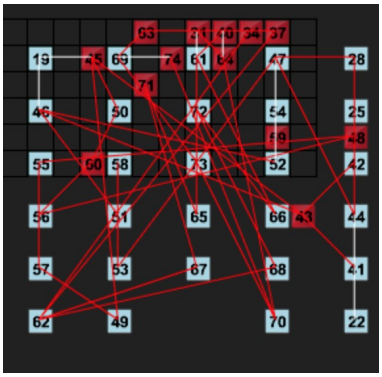


FIGURE 5.19. **Solving 4way1hops DM Pattern-2 M2.**

Few experienced players would initiate by positioning all the DM nodes in dedicated routes, then worked on positioning the red nodes. As the game progressed several swap moves were made and player worked on the violations in sub-clusters. Fig.5.19.

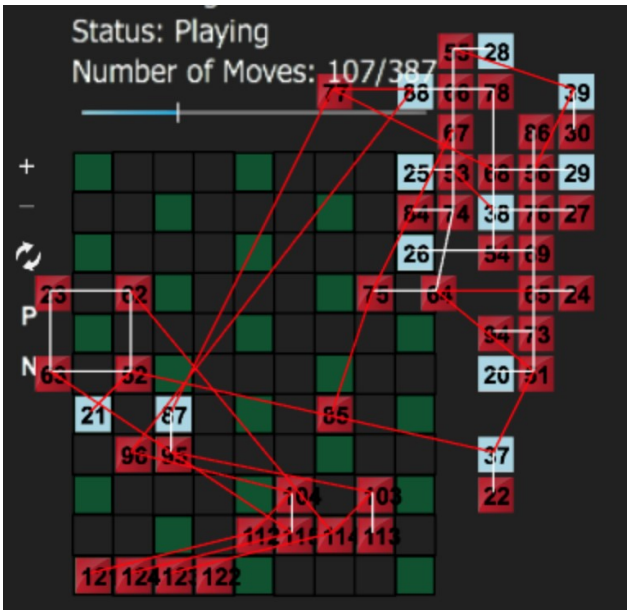


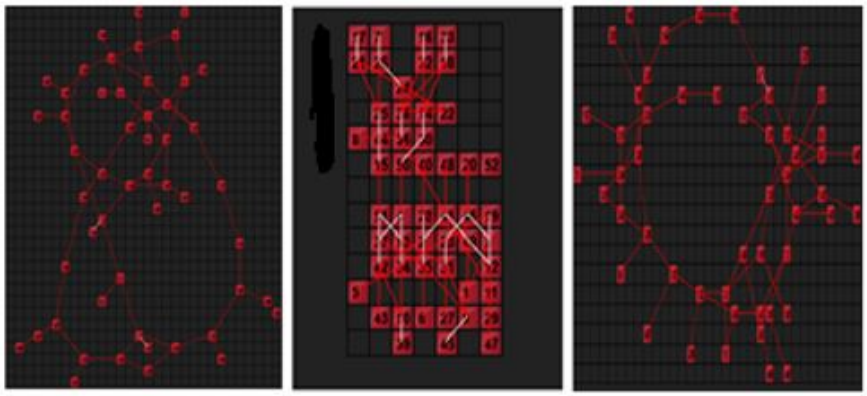

FIGURE 5.20. **Solving 4way1hops DM Pattern-1 H1**

Players would also initiate by placing the nodes from the densest part of the graph to the right side of the graph then making the connections short. Fig.5.20.

# CHAPTER 6

## CASE STUDY 3- ANALYSIS OF STRATEGY IN VARIOUS REPRESENTATION OF THE SAME PROBLEM

In this section we studied the effect of visualization by taking into consideration various aspects of the game. We observed top 10 players and analyzed how different visualization of the same graph affects the quality of the solution. For that purpose, we particularly discussed Mesh architecture: 8Way, 4Way1Hop and 4Way2Hop as its the same graph with different visualization. To perform our case studies, the factors that were taken into consideration were Average grid size, Score, Strategy and Moves.The score was taken as the first parameter to calculate the quality of the solution in each visualization. It was observed that amongst E1, M1 and H1 representation, E1 visualization succeeded in scoring highest for the first level in all three architectures. Similarly, H2 representation scored the highest in the second level, H3 representation scored highest in third level and H4 representation scored the highest in fourth level in all three architectures.



(a) 8WayE1 (b) 8Way M1 (c) 8Way H1

FIGURE 6.1. **Different Visualization of the same architecture**

Which brought us to the conclusion that players get better scores when the graph is more spread out. For example, Fig.6.1 shows graphs of E1 M1 and H1, it can be observed that E1 graph is more scattered and less dense than M1 and H1 thus players scored highest in E1. Even though H1 is also spread out the area towards the right has densely connected set of nodes. Thus making E1 visualization as best scorer in first level. For all the other levels the conclusions were made in similar manner.

TABLE 6.1. Top 10 Player's Feasible Solution

| 8Way | 4Way1hop | 4Way2hop |
|---|---|---|
| E1 12 | E1 14 | E1 26 |
| M1 12 | M1 13 | M1 24 |
| H1 9 | H1 18 | H1 23 |
| E2 9 | E2 17 | E2 26 |
| M2 9 | M2 10 | M2 22 |
| H2 12 | H2 11 | H2 27 |
| E3 16 | E3 21 | E3 28 |
| M3 10 | M3 13 | M3 20 |
| H3 16 | H3 18 | H3 28 |
| E4 7 | E4 8 | E4 18 |
| M4 4 | M4 7 | M4 12 |
| H4 4 | H4 9 | H4 20 |

**Feasible Solution**- Furthermore, we also looked at the number of feasible solutions in all three architectures, in all but three, Easy and Hard graph gave more number of feasible solutions than Medium graph this again bringing us to the conclusion that if a graph is more spread out, it makes the graph more approachable and easier to solve than the graph that is more compact. Table I 6.1

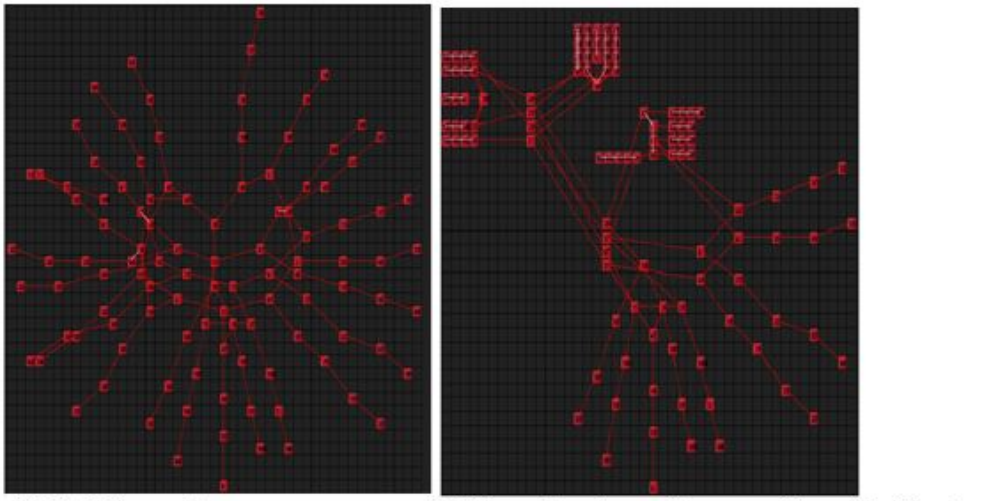
TABLE 6.2. Top 10 Player's Average Grid Size

| 8Way | 4Way1hop | 4Way2hop |
|---|---|---|
| E1 111.1 | E1 88.8 | E1 61 |
| M1 121.9 | M1 97.3 | M1 70 |
| H1 116.7 | H1 78.9 | H1 72.5 |
| E2 139 | E2 90.5 | E2 75.1 |
| M2 122.8 | M2 113 | M2 86.6 |
| H2 131.8 | H2 120 | H2 77.7 |
| E3 205.4 | E3 209.1 | E3 168.2 |
| M3 242.7 | M3 218.4 | M3 183 |
| H3 254.2 | H3 217 | H3 170.11 |
| E4 250 | E4 221.4 | E4 137.8 |
| M4 232.1 | M4 217.2 | M4 154.8 |
| H4 211.3 | M4 182.3 | H4 127.2 |

**-Area.-** Table II 6.2 compares the average grid size of top 10 players in 8Way, 4way1hop and 4Way2hop. For each architecture, we compared the average grid size. For example, compare E1 of all three architectures. In all cases but one, players were able to create most compact graphs in 4Way2hop architectures concluding again, given a substantially larger grid within which to work, players perform well.

**-Player Strategies**. In this section, results from player moves and mappings in 8Way, 4Way1hop and 4Way2hop are reported.

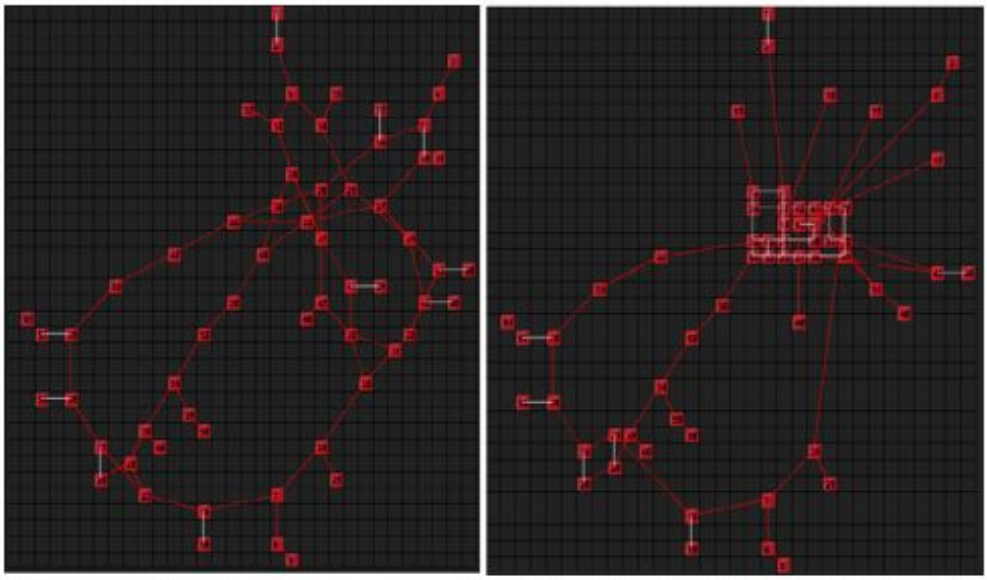
**-Moving nodes from center to outside**. Player would move the center nodes on the peripherals of the graph. This particular strategy was not opted by too many players, however the ones who followed it were successful in scoring highest in that level.Fig.6.2



(a) Initial graph (b) Dragging the nodes away from the Center

FIGURE 6.2. **8WayH4**

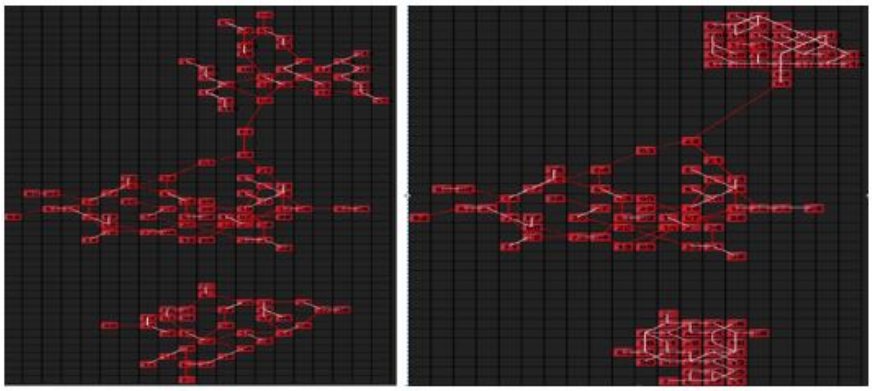
**-Moving nodes from outside to center**.: This is the opposite of the strategy that we mentioned above. Player would make the connections short by placing the nodes towards the center from the outside of the graph. Fig.6.3



(a) Initial graph. (b) Placing the nodes in the Center.

FIGURE 6.3. **4Way1HopE2**

**-Initiating with less dense cluster. Successful players would begin by attempting less dense cluster, thereupon moving to the dense one.** Fig.6.4



(a) Initial graph. (b) Solving less dense cluster.

FIGURE 6.4. **8WayE3**

**-Dividing the graph.** Some players recognized that in a more compact level like medium, it was helpful to divide the graph to better. This move was particularly helpful in seeing the connections better. Player would then make the connections short by placing the nodes in the center from either sides.Fig.6.5

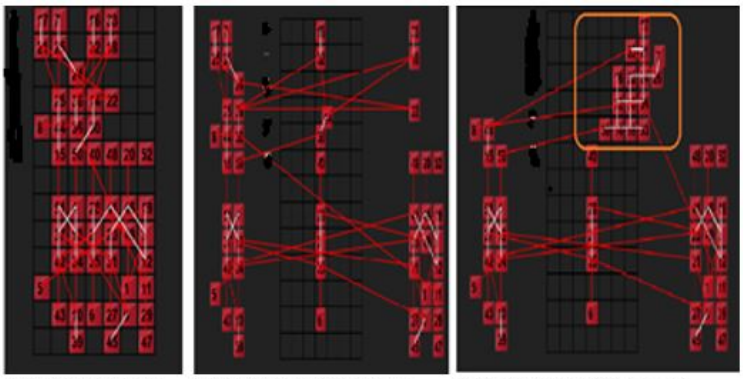



FIGURE 6.5. **8WayM1.**

**-Identifying the pattern.** Another strategy that was observed was placing the nodes in a similar pattern. Fig.6.6. shows how a player initially placed the nodes that were on the edges in vertical manner then arranged them in a common form. This strategy was effective in lowering most of the violations.

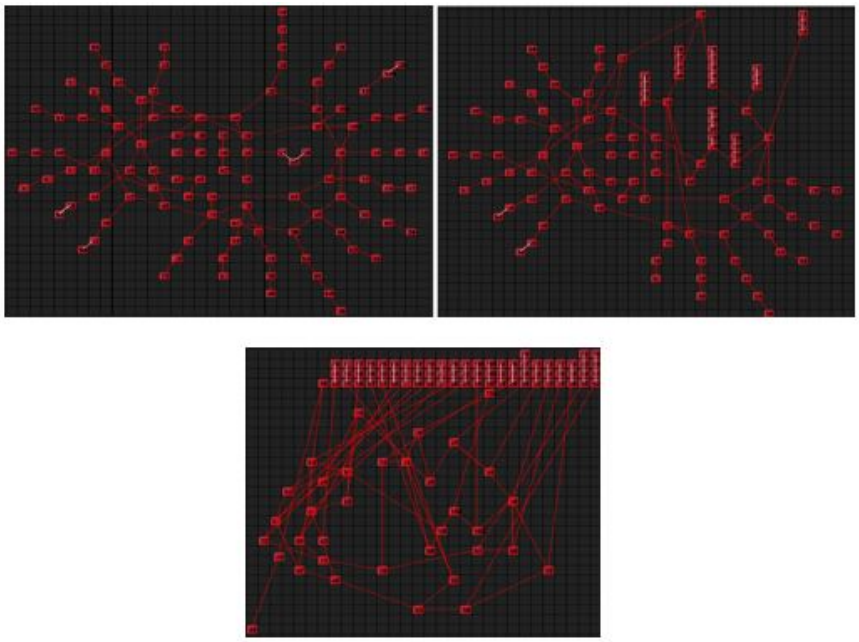



FIGURE 6.6. **8WayE4**

Generally **Easy levels** were mostly dominated by bottom to top strategy in all three architectures. Fig.6.7.

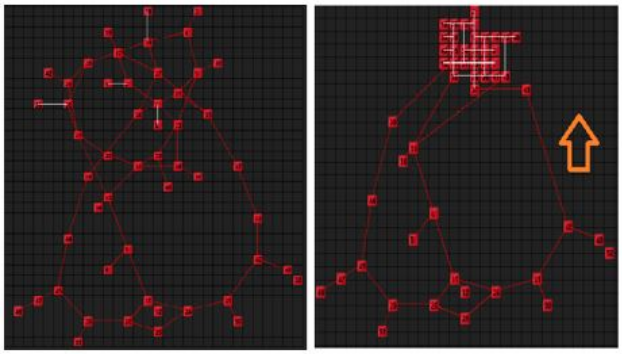



FIGURE 6.7. **4way2Hop E1**

In **Medium levels** successful players would initiate by untangling the dense part of the graph and solve it in sub-clusters. Fig.6.8.

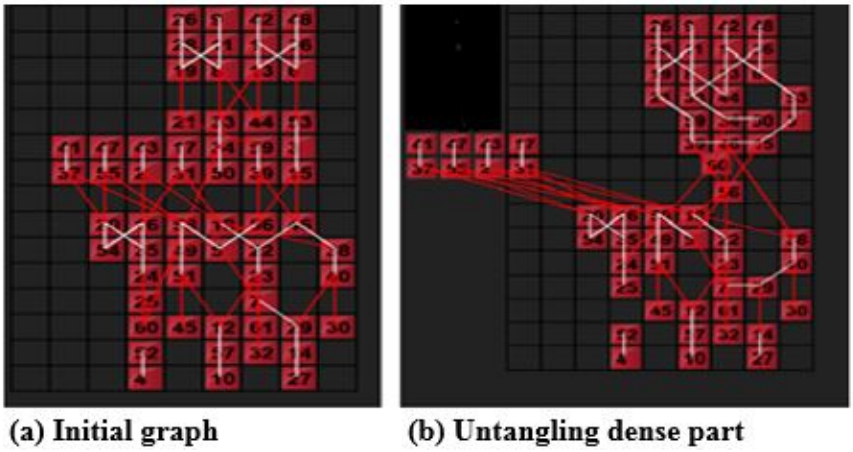
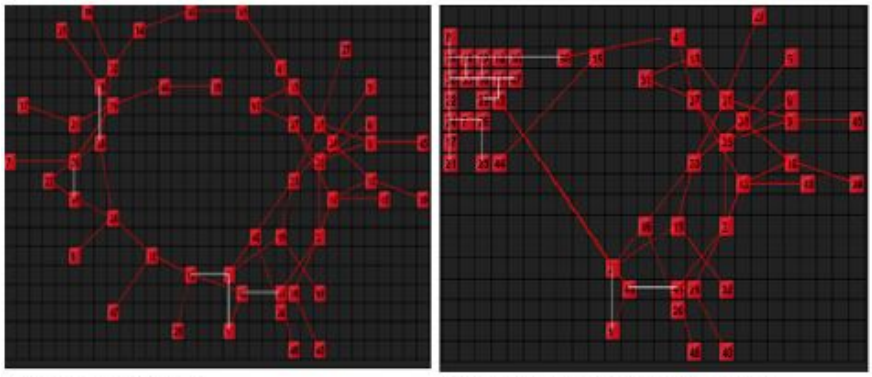


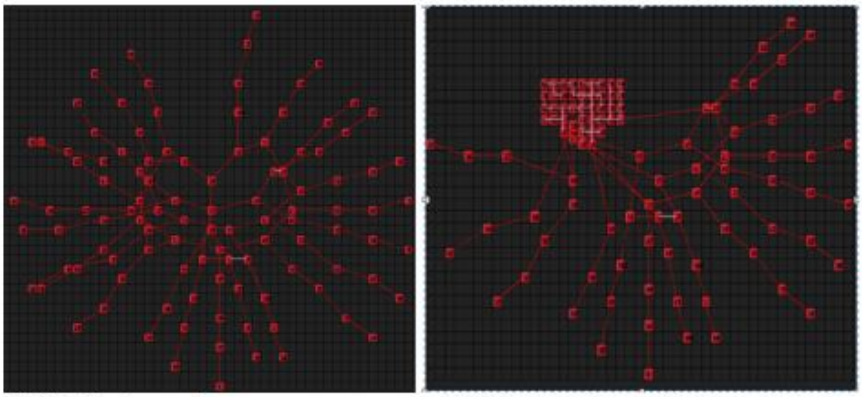

FIGURE 6.8. **8Way M2**

**Hard levels** were very spread out on the peripherals of the graph. Two successful common strategies were observed that were opted by many players in this level. First was to solve less dense area of the graph Fig.6.9. Second strategy was to solve the graph by making the connections short towards the center of the graph. Fig.6.10.



(a) Initial graph. (b) Solving less dense part.

FIGURE 6.9. **4Way2Hop H1**



(a) Initial graph. (b) Placing the nodes towards the center from outside.

FIGURE 6.10. **4Way1Hop H4**

**-Moves**. All three architectures consist of same kind of graph for all levels, only differing in the way the connections can be made to their neighbors. For example, its the same graph for E-1 in 8way, 4way1hop and 4way2hops, same for other levels. We analyzed the moves made by the top 10 players of each architecture to gain more quantitative understanding of successful player actions. We observed how player position their nodes when they initiate with the problem. The way the nodes were placed on the graph while solving for the violations were different in each architecture. In 8Way experienced players would begin by arranging the nodes in diagonal fashion Fig.6.11. Thereafter work on the connected violations.

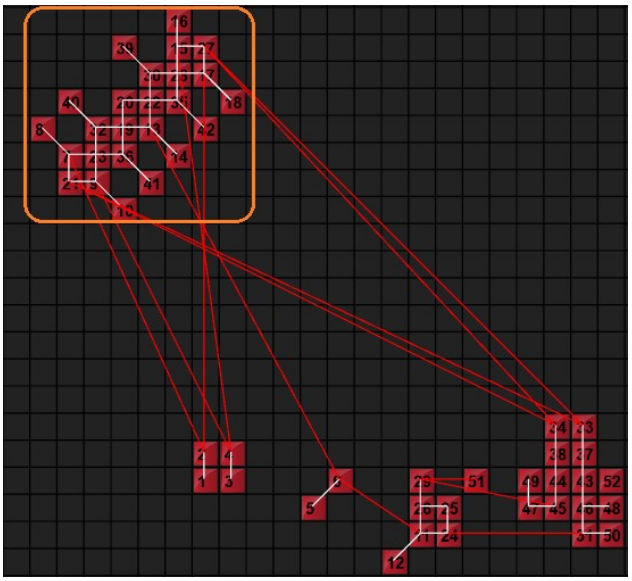


FIGURE 6.11. **Solving 8Way E1.**

Whereas in 4Way1hop player would position the nodes with 1hop then fill that empty space in the middle by placing its direct connections. Fig.6.12.Once most of the violations were solved in that fashion, player would focus on making the graph compact. Similar arrangement was followed by successful players in 4Way2hop. Nodes were positioned in both 1hop and 2hop manner followed by making its direct connections short and solving for violations.

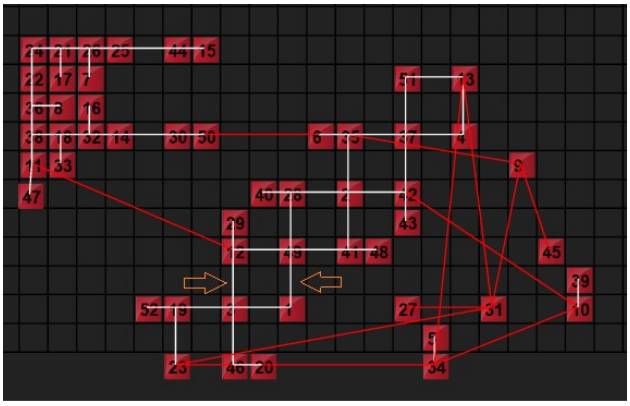


FIGURE 6.12. **Solving 4way1hop H1.**

TABLE 6.3. Graph Information (Easy Level)

| Player 1 | 8Way E1 | 4Way1hop E1 | 4Way2hop E1 |
|---|---|---|---|
| Height | 13 | 10 | 10 |
| Width | 10 | 10 | 6 |
| Nodes | 59 | 68 | 52 |
| ALU | 52 | 52 | 52 |
| Pass-Gate | 7 | 16 | 0 |
| ALU$_N$*oop* | 21 | 22 | 14 |
| Area | 80 | 90 | 66 |

TABLE 6.4. Graph Information (Medium Level)

| Player 2 | 8Way M1 | 4Way1hop M1 | 4Way2hop M1 |
|---|---|---|---|
| Height | 13 | 10 | 10 |
| Width | 7 | 8 | 7 |
| Nodes | 76 | 60 | 60 |
| ALU | 52 | 52 | 52 |
| Pass-Gate | 24 | 8 | 8 |
| ALU$_N$*oop* | 15 | 20 | 10 |
| Area | 91 | 80 | 70 |

We also analyzed the top 10 players who played all three architectures: 8Way, 4Way1Hop and 4Way2Hop and observed their quality of the solution. For example, Player 1 who played E1 in 8Way, 4Way1Hop and 4Way2Hop. Table III 6.3 , Table IV 6.4 and Table V 6.5 shows few examples of players who played all three architectures.

TABLE 6.5. Graph Information (Hard Level)

| Player 3 | 8Way H1 | 4Way1hop H1 | 4Way2hop H1 |
|---|---|---|---|
| Height | 10 | 8 | 10 |
| Width | 10 | 10 | 9 |
| Nodes | 75 | 60 | 52 |
| ALU | 52 | 52 | 52 |
| Pass-Gate | 23 | 8 | 0 |
| $ALU_N oop$ | 25 | 20 | 38 |
| Area | 100 | 80 | 90 |

It was found that on an average 4Way2Hop gives more compact solution followed by 4Way1Hop and 8Way. It was also observed that towards the final stages of the game 8Way used more number of pass gates followed by 4Way1hop and 4Way2hop. Furthermore, overall 4way2hops has high number of feasible solutions followed by 4way1hop and 8way.

# CHAPTER 7

## CASE STUDY 4- ANALYSIS OF STRATEGY FOR A GAME PLAYED MULTIPLE TIMES.

In most cases player scored better in later attempts. The score would mostly decrease in the initial phase of the game which was mostly dominated by single moves. In some cases, the player would try to perform different moves in the later attempts. For example, if a player added few pass gates in the first half of the game in the first attempt, then in the second attempt, the player would do the opposite and would add the pass gates in the second half of the game. Or a player would perform more multimove in the first attempt, however, in the second attempt player would perform more swap moves. Similarly with the strategy, experienced players would take the opposite direction in the second attempt. To further gain more quantitative understanding of the successful players strategies, we also analyzed how playing a puzzle in particular order can affect the quality of the solution. On an average Player performs well in all architectures if EMH path is followed. However, it was also observed that in some cases, player performs better if H level is attempted first as shown in Fig.7.1, Fig.7.2.

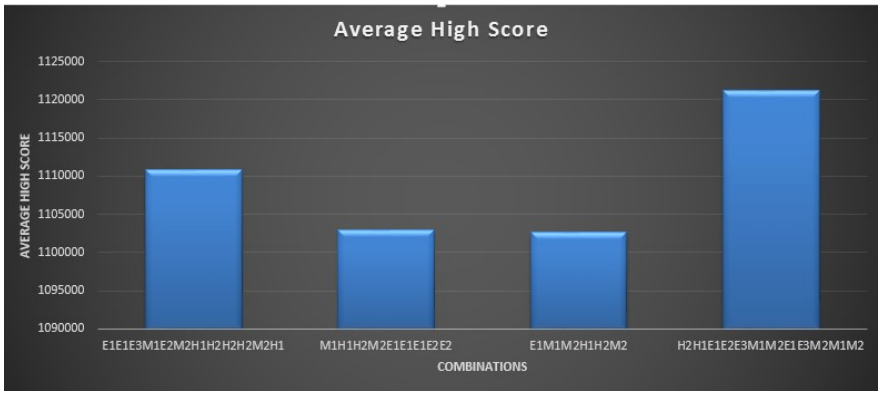



FIGURE 7.1. **4Way Architecture**

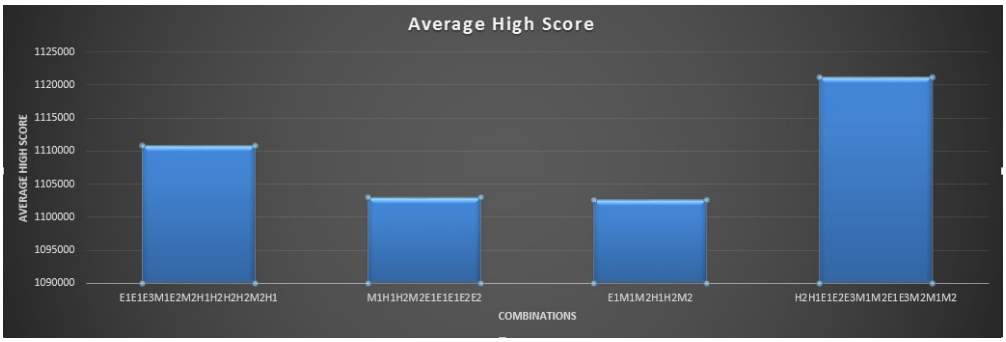



FIGURE 7.2. **Stripe Architecture**

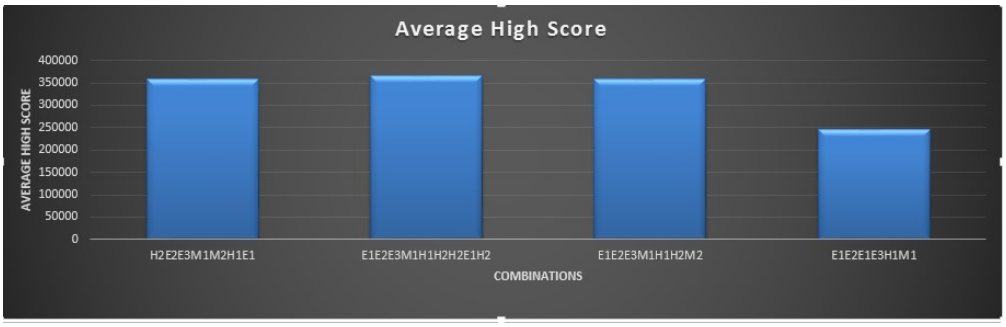



FIGURE 7.3. **4Way1Hop Architecture**

In Fig.7.3. we observe how the performances for attempting H level first are almost same as EMH combinations. Significant improvement in performance was observed, when a player attempts H level and goes on to play EM levels, then returns again to attempt the H level. This trend was observed in all architectures.

# CHAPTER 8

# CASE STUDY 5- CORRELATION BETWEEN MOVE AND SCORE

We also observed the correlation between moves and scores for top 10 players in regular Mesh architecture and Mesh architecture with constraint to further understand players strategy.

## 8.1. Mesh architecture

In most of the cases score decreased in the initial phase of the game which was mostly dominated by single moves. A drastic increase or decrease in score was mostly observed in the first half of the game. In the second half of the game, the change in the score was not very significant. Most of the time a player would come back from a break, an increase in the score was observed after that break. In some cases, multi-move decreased the score, meaning player might be solving the graph in cluster, solves for that cluster, which then later resulted in a drastic rise in the score. Adding pass gates in the initial phase of the game mostly reduced the score. Adding pass gates in the end, solved for most of the violations, however it reduced the score in some cases. We also observed the move type and which move type was performed more often and it was found as following: Single Moves > Multi-Moves> Swap Moves> Addition of Pass Gates >Removal of Pass Gates > Rotate. (In some Hard Levels, addition of pass gates was performed more frequently than swap moves).

## 8.2. I/O architecture

In most of the cases score decreased in the initial phase of the game which is mostly dominated by single moves. A drastic increase or decrease in score was mostly observed in the first half of the game. In the second half of the game, the change in the score was not very significant. Observing the move type and which move type was performed more often, the trend was found as following: Single moves > Multi-Moves > Swap Moves > Addition of Pass Gates >Removal of Pass Gates> Rotate.

# CHAPTER 9

# CONCLUSION

Coarse-Grained Reconfigurable Architectures hold a promising future in hardware platform for high performance, improved flexibility, low cost and power efficiency for various application domains. To really exploit the potential of Coarse-Grained Reconfigurable Architecture, efficient mapping problem in CGRAs is incredibly critical. In this paper, we conducted various case studies with the goal to learn human strategy to solve mapping problem for Coarse-Grained Reconfigurable architecture. To perform our experiments we did data analysis of players of a computer mapping game called UNTANGLED to identify opportunities for improved automatic mapping algorithms. All the case studies using UNTANGLED provides useful feedback from the game players that would be difficult to obtain from an automatic mapper. These insights can be useful in finding the best architectural solution for a particular domain. The different case studies show a qualitative evaluation of difficulty of mapping onto Mesh architecture, Stripe architecture and the architecture with constraints, thus also showing that crowd-sourcing the game with a purpose has the potential to have an influence to various other exciting new mapping algorithms.

# BIBLIOGRAPHY


[1] *Foldit*.

[2] *Untangled*.

[3] Mythri Alle, Keshavan Varadarajan, Alexander Fell, Ramesh Reddy C., Nimmy Joseph, Saptarsi Das, Prasenjit Biswas, Jugantor Chetia, Adarsh Rao, S. K. Nandy, and Ranjani Narayan, *Redefine: Runtime reconfigurable polymorphic asic*, ACM Trans. Embed. Comput. Syst. 9 (2009), no. 2, 11:1–11:48.

[4] T. Boiski, *Game with a purpose for mappings verification*, 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), Sept 2016, pp. 405–409.

[5] Daren C. Brabham, *Crowdsourcing as a model for problem solving*, Convergence 14 (2008), no. 1, 75–90.

[6] J. Brito, V. Vieira, and A. Duran, *Towards a framework for gamification design on crowdsourcing systems: The g.a.m.e. approach*, 2015 12th International Conference on Information Technology - New Generations, April 2015, pp. 445–450.

[7] Anupam Chattopadhyay, *Ingredients of adaptability: A survey of reconfigurable processors*, VLSI Des. 2013 (2013), 10:10–10:10.

[8] Andrew DeOrio and Valeria Bertacco, *Human computing for eda*, Proceedings of the 46th Annual Design Automation Conference (New York, NY, USA), DAC '09, ACM, 2009, pp. 621–622.

[9] Sebastian Deterding, *Gamification: Designing for motivation*, interactions 19 (2012), no. 4, 14–17.

[10] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy, *Crowdsourcing systems on the world-wide web*, Commun. ACM 54 (2011), no. 4, 86–96.

[11] Stephen Friedman, Allan Carroll, Brian Van Essen, Benjamin Ylvisaker, Carl Ebeling, and Scott Hauck, *Spr: An architecture-adaptive cgra mapping tool*, Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (New York, NY, USA), FPGA '09, ACM, 2009, pp. 191–200.

[12] M. C. Gmez-lvarez, R. Snchez-Dams, and A. Barn-Salazar, *Trouble hunters: A game for introductory subjects to computer engineering*, 2016 XLII Latin American Computing Conference (CLEI), Oct 2016, pp. 1–8.

[13] M. Hamzeh, A. Shrivastava, and S. Vrudhula, *Epimap: Using epimorphism to map applications on cgras*, DAC Design Automation Conference 2012, June 2012, pp. 1280–1287.

[14] Mahdi Hamzeh, Aviral Shrivastava, and Sarma Vrudhula, *Regimap: Register-aware application mapping on coarse-grained reconfigurable architectures (cgras)*, Proceedings of the 50th Annual Design Automation Conference (New York, NY, USA), DAC '13, ACM, 2013, pp. 18:1–18:10.

[15] Chien-Ju Ho, Tao-Hsuan Chang, Jong-Chuan Lee, Jane Yung-jen Hsu, and Kuan-Ta Chen, *Kisskissban: A competitive human computation game for image annotation*, Proceedings of the ACM SIGKDD Workshop on Human Computation (New York, NY, USA), HCOMP '09, ACM, 2009, pp. 11–14.

[16] Jeff Howe, *Crowdsourcing: Why the power of the crowd is driving the future of business*, 1 ed., Crown Publishing Group, New York, NY, USA, 2008.

[17] 27. Howe J, *The rise of crowdsourcing, wired*.

[18] A. Kosorukoff, *Human based genetic algorithm*, 2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236), vol. 5, 2001, pp. 3464–3469 vol.5.

[19] Guangming Lu, Hartej Singh, Ming-hau Lee, Nader Bagherzadeh, Fadi Kurdahi, and Eliseu M. C. Filho, *The morphosys parallel reconfigurable system*, pp. 727–734, Springer Berlin Heidelberg, Berlin, Heidelberg, 1999.

[20] J McGonigal, *Reality is broken: Why games make us better and how they can change the world 2011, penguin group, the.*

[21] G. Mehta, X. Luo, N. Parde, K. Patel, B. Rodgers, and A. K. Sistla, *Untangled - an interactive mapping game for engineering education*, 2013 IEEE International Conference on Microelectronic Systems Education (MSE), June 2013, pp. 40–43.

[22] Gayatri Mehta, Carson Crawford, Xiaozhong Luo, Natalie Parde, Krunalkumar Patel, Brandon Rodgers, Anil Kumar Sistla, Anil Yadav, and Marc Reisner, *Untangled: A game environment for discovery of creative mapping strategies*, ACM Trans. Reconfigurable Technol. Syst. 6 (2013), no. 3, 13:1–13:26.

[23] Bingfeng Mei, F. J. Veredas, and B. Masschelein, *Mapping an h.264/avc decoder onto the adres reconfigurable architecture*, International Conference on Field Programmable Logic and Applications, 2005., Aug 2005, pp. 622–625.

[24] Bingfeng Mei, Serge Vernalde, Diederik Verkest, Hugo De Man, and Rudy Lauwereins, *Adres: An architecture with tightly coupled vliw processor and coarse-grained reconfigurable matrix*, pp. 61–70, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[25] M. A. V. Orjuela, A. Uribe-Quevedo, N. Jaimes, and B. Perez-Gutierrez, *External automatic defibrillator game-based learning app*, 2015 IEEE Games Entertainment Media Conference (GEM), Oct 2015, pp. 1–4.

[26] Y. Pan and E. Blevis, *A survey of crowdsourcing as a means of collaboration and the implications of crowdsourcing for interaction design*, 2011 International Conference on Collaboration Technologies and Systems (CTS), May 2011, pp. 397–403.

[27] Marc Quax, Jos Huisken, and Jef van Meerbergen, *A scalable implementation of a reconfigurable wcdma rake receiver*, Proceedings of the Conference on Design, Automation and Test in Europe - Volume 3 (Washington, DC, USA), DATE '04, IEEE Computer Society, 2004, pp. 30230–.

[28] Z. E. Rkossy, T. Naphade, and A. Chattopadhyay, *Design and analysis of layered coarse-grained reconfigurable architecture*, 2012 International Conference on Reconfigurable Computing and FPGAs, Dec 2012, pp. 1–6.

[29] K. Tsalikidis and G. Pavlidis, *jlegends: Online game to train programming skills*, 2016 7th International Conference on Information, Intelligence, Systems Applications (IISA), July 2016, pp. 1–6.

[30] L. von Ahn, *Games with a purpose*, Computer 39 (2006), no. 6, 92–94.

[31] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford, *Captcha: Using*

*hard ai problems for security*, pp. 294–311, Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[32] Luis von Ahn and Laura Dabbish, *Labeling images with a computer game*, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (New York, NY, USA), CHI '04, ACM, 2004, pp. 319–326.

[33] M. C. Yuen, L. J. Chen, and I. King, *A survey of human computation systems*, 2009 International Conference on Computational Science and Engineering, vol. 4, Aug 2009, pp. 723–728.