On Multiplayer Techniques for Crowdsourcing Mapping onto Custom Domain-Specific Architectures

Gayatri Mehta*, Xiaozhong Luo, Anil Kumar Sistla, Andrew Marin, Mukund Malladi, Krunalkumar Patel and Brandon Rodgers
University of North Texas, Denton, TX, USA
Email: gayatri.mehta {at} unt.edu

Abstract— **One of the grand challenges in the design of portable/wearable devices is to achieve optimal efficiency and flexibility in a tiny low power package. Coarse-grained reconfigurable architectures (CGRAs) hold great promise for energy-efficient flexible designs for a domain of applications. CGRAs are very promising due to the ability to highly customize such architectures to an application domain. However, mapping applications onto such architectures is a perennially difficult problem - the greater the customization of architecture, the greater the difficulty of mapping applications onto that architecture efficiently. Our previous research has shown that crowdsourcing can be a highly effective means to solve small and moderate sized mapping problems. This paper presents a case study on multiplayer techniques with the goal of scaling to larger and more complex mapping problems in a graceful manner. To conduct this case study, we have designed a multiplayer version of our interactive mapping game, UNTANGLED. In this work, we used the benchmarks that are 2 to 5 times as large as those we have served to players to date. Our results show that multiplayer techniques are effective, and the most effective style of presentation depends on player experience, with more experienced players benefitting from a higher level of autonomy.**

Keywords-mapping, placement, coarse-grained reconfigurable architectures, custom domain-specific architectures, crowdsourcing

I. INTRODUCTION

Portable and wearable devices are ubiquitous, and the demand for these devices is increasing exponentially. These devices have a broad range of applications critical to health, safety and security, personal multimedia, and aerospace. Portable or wearable patient database displays can free doctors from being tied to cumbersome desktop computers and monitoring stations. Wearable sensors can monitor for dangerous toxins to protect firefighters and workers in other hazardous environments. Small, cheap, and low power computers and sensors can be deployed in planets and space systems to facilitate earth and planetary systems research.

Portable/wearable devices have many competing design goals - they must be light-weight, mobile, flexible, low-power, and high performance. The chip architecture that is selected is a primary determinant of how well a device meets these competing design goals. First of all, the architecture must be reconfigurable; it must be capable of running many different applications within a domain. Second, the architecture should be customized as much as possible so that it can run the intended applications with great efficiency and low power consumption. Field Programmable Gate Arrays, for example, while reconfigurable, are more general purpose and may be highly inefficient. Coarse-grained reconfigurable architectures (CGRAs) are an intriguing spot in the design space. They are reconfigurable and highly customizable [1], [2]. A typical use case for CGRAs today would be to run computational intensive kernels of a collection of related applications such as applications from the image-processing domain. However, they may be capable of doing much more.

CGRAs are composed of relatively high-level computational elements such as Arithmetic and Logic Units (ALUs), which are capable of doing operations such as addition, subtraction, and a simple interconnect such as communication between nearest neighbors. To make use of CGRAs, it is required to map a collection of applications in the form of data flow graphs (DFGs), onto a given architecture design. However, mapping dataflow graphs onto such architectures is especially challenging, precisely because of the customization that provides their advantage. The difficulty of mapping onto CGRAs may be one bottleneck to widespread adoption ([3], [4], [5], and [6]).

We focus on the mapping problem in this manuscript. Figure 1 shows an example of mapping a data flow graph of an application onto a stripe-based architecture. A DFG can be represented as a collection of nodes connected by edges. Edges are directional, representing flow of data from parent to child. A mapping of a DFG onto a reconfigurable architecture consists of an assignment of operators in the DFG to arithmetic and logic units (ALUs) in the reconfigurable architecture such that the logical structure of the application is preserved and the architectural constraints of the architecture are followed. In the stripe-based architecture shown in Figure 1, nodes are arranged in horizontal stripes, each of which is connected to all nodes in the stripe below them using a full crossbar interconnect.

Results of mapping the indicated DFG onto the stripe-based architecture are shown on the right side of the figure.

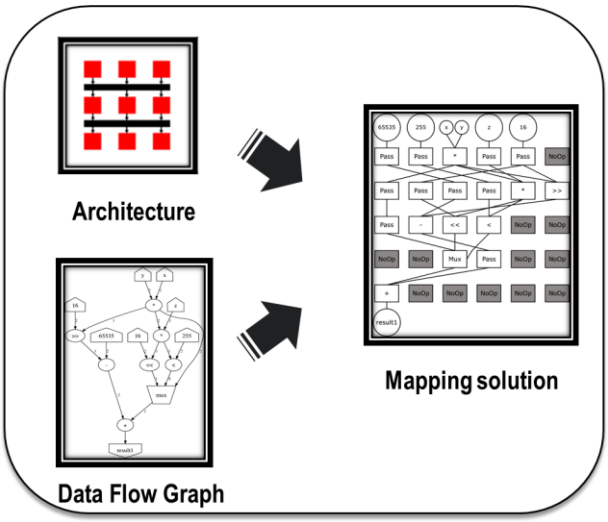


Figure 1. Mapping of a data flow graph onto a stripe-based architecture.

To begin to address the mapping problem bottleneck, we are pursuing a crowdsourcing approach, with the goals of both obtaining high quality mappings and uncovering human strategies. Specifically, we designed an interactive mapping game, UNTANGLED [7], which has been online and active for almost two years. UNTANGLED has received the People's Choice Award in the Games and Apps category of the 2012 International Science and Engineering Visualization Challenge conducted by the National Science Foundation and Science ([8], [9]). From our experimental studies conducted using single player version of UNTANGLED, it was clear that the mapping problem can be successfully crowdsourced [7]. After the success of the single player version of our game, one very important research challenge that we wanted to address and focus on was - how large and difficult mapping problems can be presented to the crowd when the problem size is too big and complexity is high for an individual player to handle.

In this paper, we describe a case study to explore an effective approach for presenting large and difficult mapping problems to the crowd using UNTANGLED. We explore how large and difficult mapping problems can be effectively crowdsourced. In order to perform the case study, we have developed a multiplayer version of our game that allows players to work in teams and communicate with each other to solve large and more difficult problems. This paper presents a case study to find out how multiplayer versions of UNTANGLED can be used to solve graphs that are up to 2 to 5 times as large as those we have served to players to date.

Our overall approach is to divide large graphs into clusters, so that individual players can focus on mapping smaller parts of the graph individually and then work together as a team to merge and adjust their solutions. We explored two options: (a) **Default clusters**, where the graph is sub-divided into clusters in-house and presented to the players, and (b) **Player clusters**, where players cooperate to divide the graph into clusters. We compared these two options from the multiplayer game to the single player option (**Single**).

We had two hypotheses entering this research work:

1) Working in teams will result in faster and better solutions than working alone. In other words, the two multiplayer options Default clusters and Player clusters should give better results than the Single option.

2) The multiplayer option Default clusters will result in faster and better solutions than the multiplayer option Player clusters. In other words, when players are given good initial clusters with which to begin, they will be able to launch right into solving the individual clusters, and the work of merging these clusters will be less, due to the care taken to provide players with clusters having minimal interconnections.

This paper describes the multiplayer version of our game UNTANGLED and the results obtained from our case study. We found out that working in teams typically produced better results than working alone. We also found to our surprise that players using the **Players cluster** option tended to produce better results, although it required significantly more time than the other options on average. Informally, player preferences were seen to evolve based on experience with the game. We present and discuss these results in the sections that follow.

II. Related Work

Our related research can be divided into three areas: coarse grained reconfigurable architectures (CGRAs), CGRA mapping algorithms, and games with a purpose. We discuss each of these areas below.

A. Coarse Grained Reconfigurable Architectures

A variety of coarse grained reconfigurable architectures have been proposed over the last 18 years, offering interesting alternatives and variations: PipeRench / Kilocore [10], RAPID [11], MATRIX [12], GARP [13], Morphosys [14], REMARC [15], KressArray [16], RAW [17], RSPA [2], ADRES [18], MORA [19], the CGRA architecture of Kim and his colleagues [20], and SmartCell [21]. However, research in CGRA architectural design on a broad scale continues. No single design has emerged as dominant, and there is a need for efficient design exploration techniques, along with high quality and flexible mapping algorithms to allow these techniques to achieve their full promise.

B. Mapping algorithms

The difficulty of the mapping problem has been well discussed in the literature, and most non-trivial formulations are NP-complete [22]. Most existing algorithms fall into one of several styles, including greedy algorithms, randomized

algorithms, clustering algorithms, Integer Linear Programming, and Analytical Placement. Comprehensive discussion and further references can be found in the following surveys: [23], [24], [25], [26], [27]. We note that other researchers have also suggested that the difficulty of mapping may be one bottleneck to widespread coarse grained reconfigurable architectures (CGRAs) adoption [4]. There is much room for improvement in any or all of these mapping algorithms, especially when we wish to explore novel highly customized architecture designs. In our previous research, we have shown that the mapping problem can be successfully crowdsourced [7] and in fact observations of human game play can lead to highly effective automatic mapping algorithms [28]. In the current work, we focus on large and complex mapping problems that are difficult for an individual to handle.

C. *Games with a Purpose*

The potential of human computation for Electronic Design Automation (EDA) has been nicely described in the research of DeOrio and Bertacco [29], [30], who mention, for example, the resemblance of the 1980's video game Pipe Dream (Pipe Mania [31]) to the problem of routing and the potential for casting placement as a packing puzzle in the manner of Tetris. DeOrio and Bertacco contribute a visual game environment for solving instances of the SAT problem [32], which has several interesting features for handling the problem of scalability, including presenting the problem to players at different scales and supporting multi-player versions of the game that involve collaboration and/or competition. In multiplayer FunSAT, one of the strategies players use is a collaborative strategy, where each player works a smaller chunk of the problem and players interact with each other to get to the final solution. Another strategy that players use is antagonistic strategy where more skilled players take control over other players' sub-problems. Our multiplayer game incorporates some features similar to FunSAT. In particular, we introduce a collaborative technique for solving the mapping problem. Throughout game play, players can communicate using a chat window.

To our knowledge, we are the only researchers to solve the mapping problem for custom reconfigurable architectures using a crowdsourcing approach. In our previous research, we have shown that the graphs containing approximately upto sixty nodes and 70 edges can be solved effectively with crowdsourcing [7]. The goal of the research described in this paper is to extend beyond that size in a graceful manner.

III. Problem Statement

Figure 2 shows one view of the design space exploration flow in which a designer may interact with various tools while exploring architectural designs for their applications. In this flow, the designer first configures a specific architecture, supplies a set of benchmarks and specifies objectives that should be optimized (e.g., a relative importance of power vs. performance vs. area). The benchmarks (presented in the form of Data Flow Graphs (DFGs) in our case) are mapped onto the proposed architecture, and the resulting mapped designs are simulated to obtain statistics relevant to the designer's objectives. These statistics are fed back to the designer, who then decides how to proceed to improve the design. The role of the designer could also be replaced by an automatic routine that iteratively attempts to optimize the architecture against a specific objective or cost function.

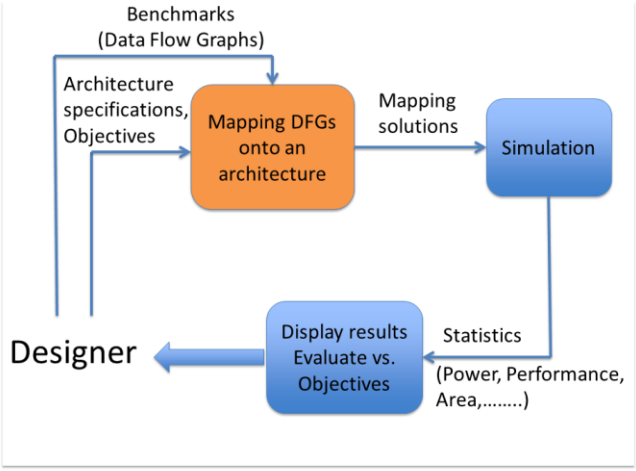


Figure 2. One view of the flow of information for a designer exploring various architectural design options.

In our research, we focus on the mapping problem shown in the orange block in Figure 2. As we have mentioned earlier that the mapping problem is very challenging for customized architectures. We have crowdsourced the mapping problem, relying on human flexibility, creativity, intuition, and persistence, rather than on a specific, potentially limited automatic mapping algorithm.

In our previous work, we have found out that the mapping problem can be successfully crowdsourced [7]. The graphs containing approximately upto sixty nodes and 70 edges can be solved effectively with crowdsourcing. Players outperformed Simulated Annealing by 2.5 standard deviations on average for a set of benchmarks and architectures [7]. In this research, we are exploring an effective approach for crowdsourcing large and difficult mapping problems. Our goals in this research are: (i) to find out if working in teams is a better approach for solving large and complex mapping problems than working alone; (ii) to find out if the problems should be divided into smaller chunks and then presented to the players or the raw graphs should be given to players and have them partition these graphs into sub-clusters and solve them.

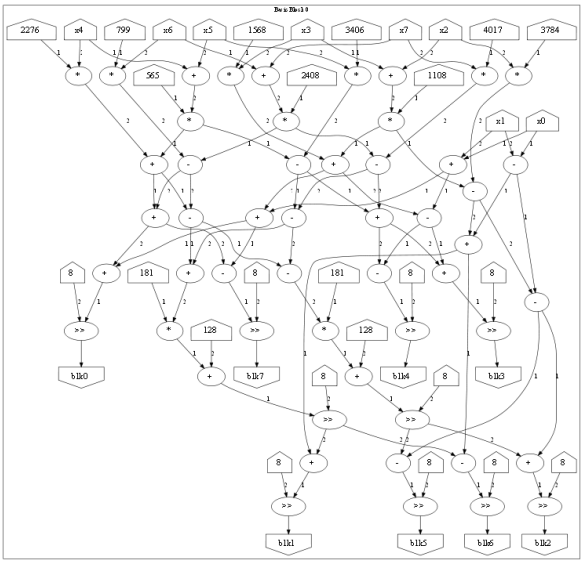


Figure 3 (a). A data flow graph (H1)

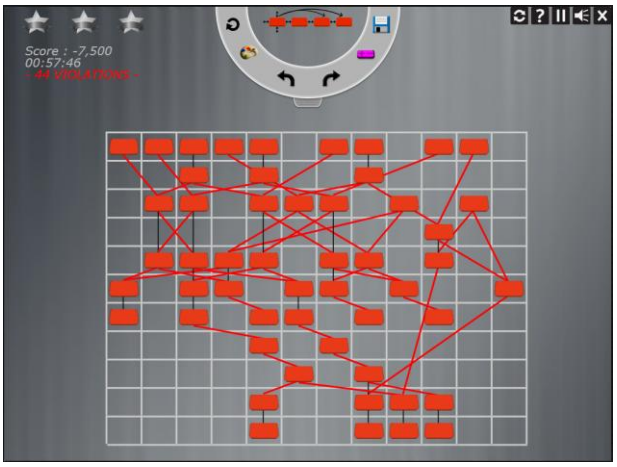


Figure 3 (b). The same graph shown on the game screen (H1).

Figure 3. The mapping problem consists of placing a DFG onto a given architecture. The figure on the top shows one of our example DFG's (H1), and the figure on the bottom shows the same graph shown on the game grid. This example maps the DFG onto 4Way2Hops mesh architecture.

IV. Presenting Mapping Problem As A Game

In this section, we describe how the mapping problem can be encapsulated in a game and presented to players. In our previous research, we have designed an interactive mapping game, UNTANGLED [7]. The game is available online at https://untangled.unt.edu. From the designer's perspective, UNTANGLED supports a variety of architectures and it allows customization of interconnect arrangements and placement of dedicated vertical routes and input/output (I/O) blocks.

From the player's perspective, UNTANGLED presents a succession of problems where a data flow graph (DFG) must be mapped onto a specific architecture. To construct winning mappings, players optimize their scores by moving nodes of the DFG within a game grid to create compact arrangements where parents are close to their children, where "close" is defined based on the interconnect structure of the architecture.

Figure 3(a) shows an example of a DFG used in the game. This is the H1 benchmark, which is the inverse discrete cosine transform of MPEG II video compression. Here, nodes are arithmetic operators, logical operators, multiplexers, or passthroughs. The DFG is presented to players in an abstract manner, with ALU's as red rectangles shown in Figure 3(b). This representation was adequate for our examples.

The game grid is an array of locations displayed to our players, within which nodes of a DFG can be placed. We can think of the game grid as an array of ALUs or other functional elements, which have not yet been assigned operations. Placing a node from the DFG in a grid location corresponds to assigning an operation to a functional element. The architecture's interconnect defines the paths along which data can flow efficiently within the game grid. In game play, players are presented with an architecture definition, which is abstracted as an icon showing legal connections between a node and its neighbors as shown in Figure 3(b).

The game interface allows players to drag and drop nodes within the game grid. Players can select and move clusters of nodes. The interface allows players to rotate and mirror their clusters. Players can add and remove passthroughs, which are often required to route data from producer nodes to consumer nodes within constraints of the architecture. Throughout the game play, players can see their score and the violations in their graph. They get incentives such as badges, medals during the game play and they can track their scores / rankings in comparison to other players around the world on the game leaderboard.

V. Untangled Multiplayer

In this section, we describe the multiplayer version of the UNTANGLED game, which has been designed to allow players to work both individually on their own clusters and also together as a team to solve a graph. The game is available online (https://untangled.unt.edu/multiplayer). We describe each of the three treatments, **Default clusters**, **Player clusters**, and **Single** below.

For the **Default clusters** version of the game, clusters were created in-house and presented to the players. To create the clusters, we used a combination of off the shelf clustering techniques and manual adjustment to obtain clusters that were nearly equal in size and had minimal connections between them.

Figure 4 shows some screenshots from the **Default clusters** version of our interface. Figure 4(a) shows the initial game

screen, where a player can start a new game, continue an existing game, or join an existing game from other teams who have made their games public. A player who has joined a game has the option to select one of the predefined clusters, as shown in Figure 4(b). Once a cluster has been selected, the players go to their own play screen, shown in Figure 4(c). In this screenshot, the player is playing the dark blue cluster and communicating with other players using a chat window. The orange, light-blue, and green nodes that we see in this dark blue colored cluster are called port nodes. This means that this cluster is connected to the orange, light-blue, and green clusters through these port nodes. In addition to manipulating their own cluster, players also have access to the complete initial graph at all times. Players can view the most recent clusters from other players, but cannot make modifications to clusters owned by other players. They can only manipulate their own clusters.

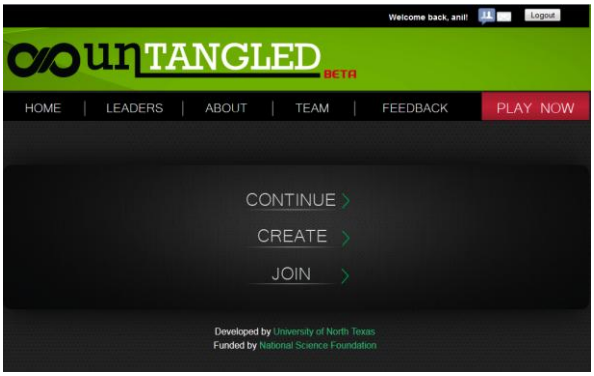


Figure 4 (a). Create a new game or Continue or Join an existing game.

Figure 4 (b). A screen with clusters, initial graph, and merged graph options.

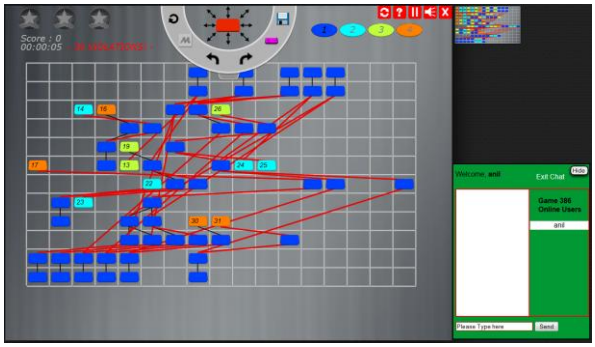


Figure 4 (c). A player playing a cluster and a chat window for talking to the other players.

Figure 4. Multiplayer game interface.

Once all the clusters have been individually played by the players of the team, the players have three options to choose from: (i) they can work together to merge the clusters and reach a final solution, or (ii) they can appoint a leader who is in charge of integrating all the clusters to get a final solution, or (iii) each player can merge the clusters and reach a final solution and the best final solution among their group will make to the leaderboard.

The **Player clusters** version of the game is very similar to **Default clusters**. However, it has one initial step. At the beginning of the game, the team works together to form the clusters from a raw, uncolored initial graph. Once clusters are selected, players individually sign up for clusters and play proceeds as with the **Default cluster** version. Once the team has formed clusters, the clusters cannot be changed.

The Single version of the game that was presented to users was identical to the original UNTANGLED game as described in [7]. No clusters are formed; the entire graph is played as a single cluster, with no merging required. Teams had the option to play the entire graph while sitting together, or play it individually. In some cases, all team members from a team or a few players from a team play individually a particular graph and they generate multiple final solutions. In those situations, we take the total time into account that went into getting all the legal final solutions from that team and the best solution can make it to the leaderboard.

Figure 5 shows initial and final graphs for the best solution in each of the three styles of game play. The benchmark is L1 (described in Section VII). The architecture is 4Way1Hop (also described in Section VII).

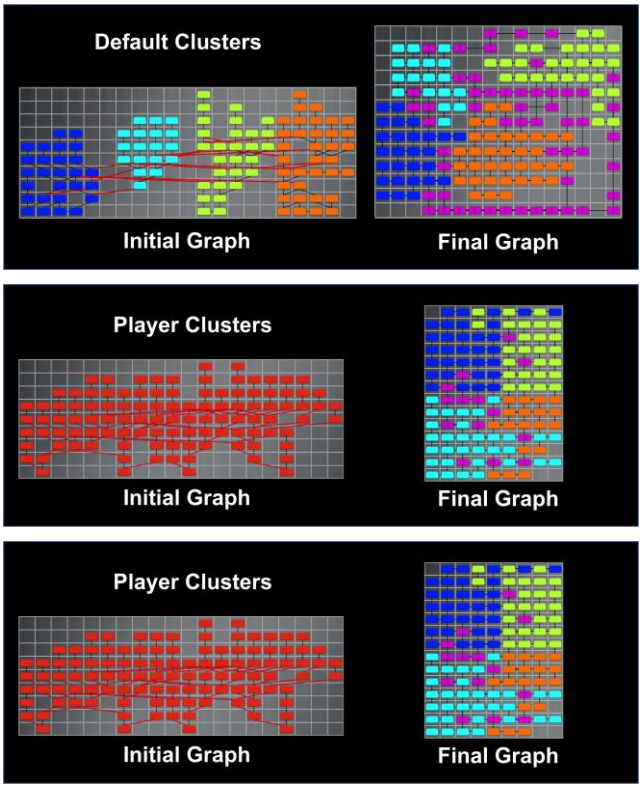


Figure 5. Benchmark L1 mapped onto the 4Way1Hop mesh architecture (Best solution using each technique).

VI. User Studies

The experimental protocol for our user studies was determined to qualify for an exemption from the Institutional Review Board of the University of North Texas. IRB protocols were followed in these studies. We conducted formal in-house studies in our lab where players were given an introduction to the game and invited to play, while talking out loud about what they were doing and their goals and expectations at each step. Observations were made by our players. We have around forty participants who have played the multiplayer version of the game so far.

VII. Experiments

For our case study, we used two architectures - 4Way1Hop and 8Way and two benchmarks in each architecture. 4Way1Hop (shown in Figure 6 Left) is a mesh architecture allowing connectivity to direct horizontal and vertical neighbors, as well as horizontal and vertical connections that skip one node. 8Way (shown in Figure 6 Right) is a mesh architecture where nodes can connect to any of their 8 neighbors. These are both common CGRA architectures with a good mix of power savings and performance but present quite difficult mapping problem to the players because of their difficult connectivity patterns. We assume that inputs can be loaded and outputs read from any ALU. The benchmark statistics are shown in Table I. The table shows the number of nodes, edges, highest degree, and number of nodes with degree 4 and higher in the benchmarks. We selected matrix multiplication, and matrix inversion benchmarks for our studies that appear in most of the signal processing applications. These benchmarks are up to 2 to 5 times as large as those we have served to players to date.

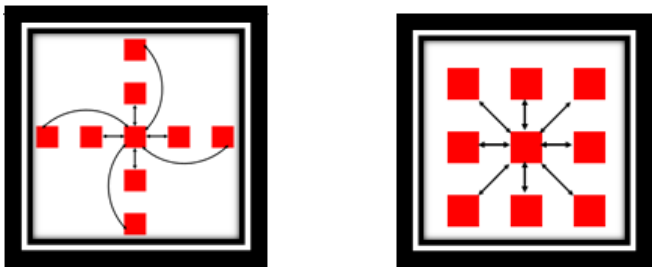


Figure 6. 4Way1Hop (Left) and 8Way (Right) mesh architectures.

We have studied three techniques to solve large and more difficult graphs. The first two techniques use the multiplayer version of the game where players solve the graphs in teams: (a) **Default clusters**, where the graph is sub-divided into clusters in-house and presented to the players, (b) **Player clusters**, where players cooperate to divide the graph into clusters. For the third option, we presented raw initial graphs to individuals in a single player game **Single**.

A total of 9 teams of 4 students each participated in our experiments. Students in our case study were recruited from the UNT EE student population and had some knowledge of engineering, although as we have shown in our previous research [7], this background is not required to perform well in the game.

Twelve total treatments were possible:

- Three variations of the game (Default clusters, Player clusters, or Single),
- Two architectures (4Way1Hop or 8Way), and
- Two benchmarks (L1 or L2)

Teams were presented with treatments in random order in order to minimize the influence of learning effects in our results. The number of treatments completed by each team ranged from 2 to 7. On average, teams completed 5 of the 12 treatments for a total of 43 completed solutions, representing 14362 total minutes of game time.

TABLE I. BASIC INFORMATION RELATED TO THE BENCHMARKS

	Nodes	Edges	Highest Degree	No. of Degree 4 Nodes	No. of Degree 5 Nodes	No. of Degree 6 Nodes	No. of Degree 7 Nodes
L1-Matrix Multiplication	108	116	5	0	4	0	0
L2-Matrix Inversion	336	354	7	4	0	4	1

A. Our Simulated Annealing Algorithm

Simulated Annealing (SA) is frequently used for placement because of its flexibility and good quality performance. We developed our own SA algorithm that closely follows that of Betz [33], with the exception that we do not use SA for placement only, but perform routing in the inner loop of the algorithm. Details of our custom SA can be found in [7]. The scoring function for the SA was designed in a way to encourage the algorithm towards good solutions. It takes into account the penalty for violations and area. We compare our players' results with SA.

VIII. RESULTS

In this section, we first present an overview of our results, and then answer the following questions - (i) Can players solve the mapping problem?, (ii) Is multiplayer better than single player?, and (iii) Which multiplayer technique is better?

A. Results Overview

We compare performance for the Default clusters, Player clusters, and Single variations for each benchmark and architecture separately. In particular, we ask the question for a given problem (e.g., mapping the L1 benchmark to the 4Way1Hop architecture), which of these three variations of the UNTANGLED game resulted in the best final mapping of that benchmark onto the architecture?

These best in class results are broken out in Table II, Table III, Table IV, and V. In particular, Table II gives the best mappings for benchmark L1 and the 4Way1Hop architecture, Table III gives the best mappings for benchmark L2 and the 4Way1Hop architecture, Table IV gives the best mappings for benchmark L1 and the 8Way architecture, and Table V gives the best mappings for benchmark L2 and the 4Way1Hop architecture. We consider these results one by one.

In these tables, we see the size of each final graph in terms of "Width X Height" as well as total number of "Functional Units." Functional units are counted as the number of units in the smallest axis aligned grid that surrounds the final graph. "Pass Gates" are the pink elements required to pass data between nodes that are too distant to be connected directly by the 4Way1Hop interconnect. "Scores" are given next. These are the scores seen by the teams and individual players. Players are evaluated and compete based on this score. Higher scores are better. Violations in the mapped solutions are also shown in the tables. These show the edges that cannot be routed under the given architectural constraints.

Finally, the tables give time, in minutes, that the team spent solving the graphs, enumerated by time spent to make the clusters in the Player cluster version of the game, time spent solving clusters in both multiplayer versions of the game, time spent in merging the clusters for multiplayer games, and total time spent in any version of the game.

Table II shows the size of the grid, number of pass gates used, scores obtained, and time spent to achieve the highest scoring solutions for the 4Way1Hop architecture and L1 benchmark using all three techniques (**Default clusters**, **Player clusters**, and **Single**). Results for this case can be viewed visually in Figure 5. Players obtained the most compact mapping solution for the **Player clusters** option, with a 9x14 grid, or equivalently 126 functional units. SA created a compact mapping but its score is lower than all other options but **Default clusters**. In terms of time required to solve the graph, however, the Player clusters option took longer than the **Default clusters** option for players to reach their final mapping solution. In this case the **Single** variation resulted in an intermediate score of the three options, and required the greatest amount of total time.

Table III shows the size of the grid, number of pass gates used, scores obtained, and time spent to achieve the best solution for the 4Way1Hop architecture and L2 benchmark using all three techniques (Default clusters, Player clusters, and Single). In this case, the Player clusters option again gave the highest score, followed by Single and then Default clusters. Solving the Player clusters option required an extraordinary amount of time in this case, even compared to the Single player variation. In this case, some of the teams tried several different strategies to solve each cluster and then tried several ways to integrate these clusters together to get the final solution. We took into account all the time players of the team (who provided the best solution) put in trying out different options to get to the final solution. Note that although the SA solution is more compact and has a higher score, it is

not a valid mapping, having 6 violations, edges that cannot be routed using this architecture. In this case, SA failed to find any valid solution.

TABLE II. GRID SIZE, PASS GATES, SCORES, TIME SPENT FOR THE BEST SOLUTION IN EACH CATEGORY FOR 4WAY1HOP ARCHITECTURE AND L1 BENCHMARK. BEST VALUES ARE SHOWN IN BOLD

	Default clusters	**Player clusters**	**Single**	**SA**
Width X Height	16x14	**9x14**	11x14	14x14
Functional Units	192	**126**	154	134
Pass Gates	54	**14**	24	26
Scores	413930	**462880**	447660	429960
Violations	**0**	**0**	**0**	**0**
Time Spent in making clusters (mins)	-	56	-	-
Time Spent in solving clusters (mins)	57	**40**	-	-
Time Spent in merging clusters (mins)	**37**	122	-	-
Total Time Spent (mins)	**94**	218	232	1320

TABLE III. GRID SIZE, PASS GATES, SCORES, TIME SPENT FOR THE BEST SOLUTION IN EACH CATEGORY FOR 4WAY1HOP L2. BEST VALUES ARE SHOWN IN BOLD

	Default clusters	**Player clusters**	**Single**	**SA**
Width X Height	27x26	24x18	18x24	**20x20**
Functional Units	675	408	432	**362**
Pass Gates	138	61	69	**26**
Scores	2543470	**2682300**	2679000	**2685190**
Violations	**0**	**0**	**0**	6
Time Spent in making clusters (mins)	-	12	-	-
Time Spent in solving clusters (mins)	**239**	344	-	-
Time Spent in merging clusters (mins)	**70**	579	-	-
Total Time Spent (mins)	**309**	935	499	2747 (approx. 2 days)

Table IV shows the size of the grid, number of pass gates used, scores obtained, and time spent to achieve the best solution for the 8Way architecture and L1 benchmark using all three techniques (Default clusters, Player clusters, and Single). The results show that SA produced the most compact mapping and highest score, followed closely by Default clusters. For this example, the Default clusters result was superior in terms of score and far superior in terms of time required to solve the graph.

Table V shows the size of the grid, number of pass gates used, scores obtained, and time spent to achieve the best solution for the 8Way architecture and L2 benchmark using all three techniques (Default clusters, Player clusters, and Single). The results show that Player clusters option provides us the most compact mapping solution, followed by Default clusters and then Single. For this example, you may note that the Default clusters result required an inordinate amount of time, most of it spent merging the clusters. In this case, the winning team elected to have all four team members separately attempt the merge and kept the highest scoring result. Time required for the three failed attempts is also included in the final timing result. In this case, SA again failed to find a valid mapping.

TABLE IV. GRID SIZE, PASS GATES, SCORES, TIME SPENT FOR THE BEST SOLUTION IN EACH CATEGORY FOR 8WAY L1. BEST VALUES ARE SHOWN IN BOLD

	Default clusters	**Player clusters**	**Single**	**SA**
Width X Height	12x14	15x15	14x13	**13x13**
Functional Units	156	210	182	**134**
Pass Gates	38	96	53	**26**
Scores	**364380**	318020	352550	**368820**
Violations	**0**	**0**	**0**	**0**
Time Spent in making clusters (mins)	-	14	-	-
Time Spent in solving clusters (mins)	**29**	345	-	-
Time Spent in merging clusters (mins)	**129**	228	-	-
Total Time Spent (mins)	**158**	588	602	1358

TABLE V. GRID SIZE, PASS GATES, SCORES, TIME SPENT FOR THE BEST SOLUTION IN EACH CATEGORY FOR 8WAY L2. BEST VALUES ARE SHOWN IN BOLD

	Default clusters	**Player clusters**	**Single**	**SA**
Width X Height	30x22	27x22	31x27	**20x20**
Functional Units	630	567	837	**372**
Pass Gates	201	145	183	**36**
Scores	2352120	**2400800**	2288000	**2493230**
Violations	**0**	**0**	**0**	4
Time Spent in making clusters (mins)	-	1	-	-
Time Spent in solving clusters (mins)	232	**74**	-	-
Time Spent in merging clusters (mins)	927	**888**	-	-
Total Time Spent (mins)	1159	**963**	398	2851 (approx. 2 days)

B. *Can players solve the mapping problem?*

In our previous research, we have shown that the graphs containing approximately up to 60 nodes and 70 edges can be solved effectively using crowdsourcing [7]. Players outperformed Simulated Annealing by 2.5 standard deviations on average for a set of benchmarks and architectures [7]. Players generated high quality, reliable mappings, and outperformed our custom SA algorithm in 37 out of 42 trials [34]. In this research, we answer the following question - For larger and difficult mapping problems, how did our players do? Players did better than SA 3 out of 4 times. They solved the large and complex graphs and generated feasible solutions in all the test cases. SA did not find a valid solution 2 out of 4 times in this case study.

C. *Is Multiplayer Better than Single Player?*

Our first hypothesis was that **working in teams would result in faster and better solutions than working alone**. In view of that hypothesis, we first present overall results comparing the three techniques in terms of score and time. Table VI shows the overall comparison of the three techniques studied here. Broadly, we see that players who solved the same graph with more than one variation of the game performed 1.6% better than their own average score when they played the **Player clusters** variation, 0.9% better than their own average when they played the **Default clusters** variation, and 2.4% worse, when they played the Single variation, suggesting a trend favoring **Player clusters** followed by **Default clusters** followed by **Single**.

TABLE VI. OVERALL COMPARISON OF THREE TECHNIQUES

	Default clusters	**Player clusters**	**Single**
Difference from average	0.9%	1.6%	-2.4%
Count of "wins"	4 wins	6 wins	3 wins
Count of "losses"	4 losses	3 losses	6 losses

We can view these results differently in terms of "wins" and "losses," for each game variation, here expressed as the number of times a treatment resulted in a score that was above a team's own average for a given benchmark and architecture (a "win") or below it (a "loss"). In Table VI, we can see that for **Player clusters**, teams beat their own average six of ten times. For **Default clusters** teams showed as many "wins" as "losses." For **Single**, teams beat their own average only four of ten times.

For more specific detail, to obtain the results in Table VI, we considered results for each team, architecture, and benchmark individually. We computed the average of the scores obtained by that team using **Default clusters**, **Player clusters**, and **Single** variations, including only whichever variations that team actually played. We then calculated the percentage difference from average for all variations for this team, architecture, and benchmark. These results were averaged over all the teams, benchmarks, and architectures to obtain the compiled results shown in Table VI.

D. *Which Multiplayer Technique is Better?*

Our second hypothesis was that **the multiplayer option Default clusters will result in faster and better solutions than the multiplayer option Player clusters**.

Table VI has already given some indication to the contrary. In terms of score, on average, teams performed better with the **Player clusters** option than for **Default clusters**.

Table II, Table III, Table IV, and V also present information to the contrary, in that the **Player cluster** variation produced the best result for three of the four benchmark / architecture combinations.

However, we can note from Table II, Table III, Table IV, and V that **Player clusters** may be more time consuming to play. In this section, we dig more deeply into the timing results.

Table VII shows the breakdown of the average time in minutes spent by our players in building clusters, solving clusters, and merging clusters to reach a final solution for benchmark L1 and the 4Way1Hop architecture. This table shows that the least time was required for the **Default clusters** variation, followed by **Single**, followed by **Player clusters**.

TABLE VII. BREAKDOWN OF TIME SPENT IN MAKING CLUSTERS, SOLVING CLUSTERS, AND MERGING CLUSTERS FOR THE FINAL SOLUTION - 4WAY1HOP L1 AND AVERAGED OVER ALL THE TEAMS

	Default clusters	**Player clusters**	**Single**
Time Spent in making clusters	-	28	-
Time Spent in solving clusters	68	50	-
Time Spent in merging clusters	55	181	-
Total Time Spent	123	259	188

Table VIII shows the timing breakdown for benchmark L2 and the 4Way1Hop architecture. Here, the trend is similar, although the larger graph required more time to solve. In this case, there is virtually no difference from the time required for the **Default clusters** vs. the **Single** variations.

TABLE VIII. BREAKDOWN OF TIME SPENT IN MAKING CLUSTERS, SOLVING CLUSTERS, AND MERGING CLUSTERS FOR THE FINAL SOLUTION - 4WAY1HOP L2 AND ALL THE TEAMS

	Default clusters	**Player clusters**	**Single**
Time Spent in making clusters	-	19	-
Time Spent in solving clusters	284	198	-
Time Spent in merging clusters	124	333	-
Total Time Spent	408	549	407

TABLE X. BREAKDOWN OF TIME SPENT IN MAKING CLUSTERS, SOLVING CLUSTERS, AND MERGING CLUSTERS FOR THE FINAL SOLUTION - 8WAY L2 AND ALL THE TEAMS

	Default clusters	**Player clusters**	**Single**
Time Spent in making clusters	-	25	-
Time Spent in solving clusters	250	116	-
Time Spent in merging clusters	458	249	-
Total Time Spent	709	390	170

Table IX shows the timing breakdown for benchmark L1and the 8Way architecture. Again, the trend shows that the least time was required for the **Default clusters** variation, followed by **Single**, followed by **Player clusters**.

Table X shows the timing breakdown for benchmark L2 and the 8Way architecture. Here, timing information does not follow the typical trend, showing relatively little time on average applied to the Single variation, followed by Player clusters then with the greatest time spent on Default clusters. In this case, we have noticed that a couple of teams tried several different strategies to solve their clusters and several different ways to merge them together to get to the final solutions. This is the most difficult of the four benchmark / architecture combinations.

IX. DISCUSSION AND CONCLUSION

In this paper, we have described a multiplayer version of the game UNTANGLED, with the goal of crowdsourcing the problem of mapping large algorithms to custom domain specific architectures. Our objective was to make it possible to crowd source the mapping of larger and more complex graphs than have been seen to date, while keeping the process manageable and enjoyable for the players.

TABLE IX. BREAKDOWN OF TIME SPENT IN MAKING CLUSTERS, SOLVING CLUSTERS, AND MERGING CLUSTERS FOR THE FINAL SOLUTION - 8WAY L1 AND ALL THE TEAMS

	Default clusters	**Player clusters**	**Single**
Time Spent in making clusters	-	15	-
Time Spent in solving clusters	62	345	-
Time Spent in merging clusters	173	203	-
Total Time Spent	235	563	257

Results of our case study show that the multiplayer game produces better results than the previous, single player version. Two sources of analysis support this result. First, teams playing more than one variation obtain better scores on average with either multiplayer variation than with the single player variation. This result can be seen in Table VI, both as a higher percentage difference from average score for the multiplayer variations, as well as a greater number of "wins" vs. "losses" for multiplayer.

Second, the overall best scores obtained for each benchmark / architecture combination were obtained using a multiplayer variation rather than a single player variation, as shown in Table II, Table III, Table IV, and V. Specifically, the **Player clusters** variation performed best in three out of four cases, and the Default cluster variation performed best in one out of four cases. The Single variation sometimes took second place, but it was never the best of the three solutions.

Our second hypothesis was that the Default clusters variation would produce better results than the Player clusters variation. However, in terms of scores, the opposite hypothesis was better supported, both in the Table VI and cumulatively Table II, Table III, Table IV, and V. Results of our case study show that the Player clusters variation typically produces better scores (i.e., better final graphs) than the Default clusters variation.

However, we observed that playing the Player clusters variation required considerably more time for teams than playing the Default clusters variation. In fact, the extra time taken cannot be explained by the extra time required to make the clusters in the first place, as shown in Tables VII through X. Time required to form the clusters in the first place is typically less than 10% of total time, although time differences between the two variations are much greater.

We believe that teams spent more time on the Player clusters variation because they tended to create more interconnected clusters, which required more time both to solve and to merge. Table XI shows the interconnections among clusters for both architectures and graphs for Default clusters and Player clusters options. For example, for 4Way1Hop L1, player clusters have interconnections with other clusters ranging from 12 to 20. Nevertheless, once teams did complete the merge, after investing the extra time, they were often able

to come up with top quality solutions showing that they were sufficiently invested in the process to push it forward to a good solution. Some of the players spent good amount of time to fix the raw graph before they even partition it. That also helps them in getting good quality final solutions. Perhaps defining their own clusters created more investment in producing high quality results, but this is a topic for further research.

TABLE XI. INTERCONNECTIONS AMONG CLUSTERS FOR BOTH ARCHITECTURES AND GRAPHS FOR DEFAULT CLUSTERS AND PLAYER CLUSTERS

	Default clusters	**Player clusters**
4Way1Hop L1	12	12, 13, 14, 18, 19, 20
4Way1Hop L2	15	13, 15, 18
8Way L1	12	13
8Way L2	15	14, 15, 19, 22

Because the **Players clusters** treatment was generally the most successful, we examine players' strategies for this treatment. Figure 7 shows an example where players were given an initial raw graph, how players take a sequence of steps to divide the graph into clusters, solve those clusters individually and put them back together to solve the whole graph. We have noticed that players give high priority to nodes with higher degrees. They first separate out the nodes with more connections, try to align along either a horizontal or vertical line, then move the lower degree nodes behind that line and keep all the long distance violations in the middle as shown in Figure 7(b). They follow this approach until they get distinctive clusters and then color these clusters using different colors. Players solve these clusters individually and then put the solved clusters together during the merge phase. They often use rotate/flip tool to rotate or flip a group of nodes to fix their alignments to get the final solution. They use pass gates to remove the violations and rearrange the blocks to get the compact arrangement.

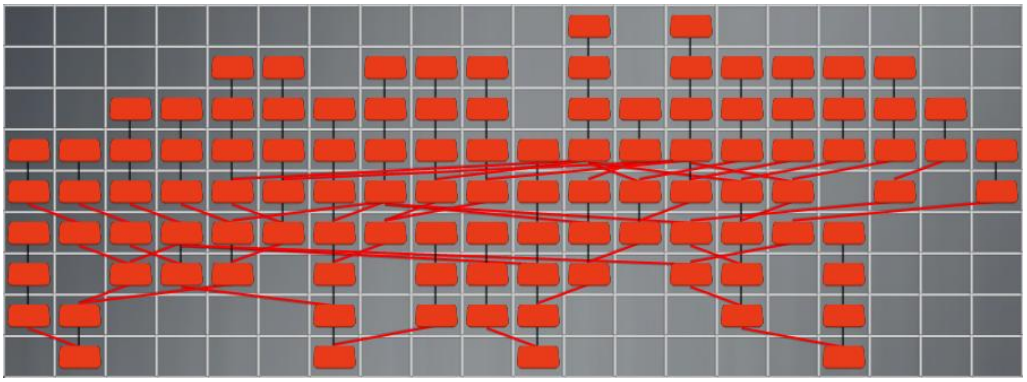
Players' results and process illustrate difficulty with routing between partitions that is evident in automated algorithms as well. Perhaps if partitioning strategies are to dominate, interconnect should be designed non-homogeneously with the difficulty of routing between partitions in mind. We have also observed a variety of interesting strategies while following players as they solved their own clusters, including rotation, pivoting, and sifting nodes or sub graphs up / down or across. Exploiting how such strategies may be exploited in automated algorithms is also a topic of future research. We also plan to extend our experimental studies by adding more benchmarks and architectures to our test suite.

Informal discussion with teams also provides some additional insight. We found that players encountering the game for the first time were happier with the **Default clusters** variation, because they felt that they began from a more manageable starting point. However, more experienced players wanted more control over the game. As they developed their own strategies, they became more aware of the properties that were needed to separate the graph into good vs. poor clusters. Eventually, as players became even more experienced, they often preferred to take over and solve the entire graph as a whole (i.e., play using the **Single** variation). However, even these more experienced players noted that when the problem size becomes large, they prefer to play as a team to get to the final solution in a reasonable amount of time. These observations and our results indicate that a training process for new players may be especially beneficial.

ACKNOWLEDGMENT

This work was supported by the National Science Foundation, under grants CCF-1117800 and CCF-1218656. We would like to thank Emily Lofaro for designing graphics for the game, and all the players for their participation and feedback.

Figure 7 (a). An initial graph L1



.

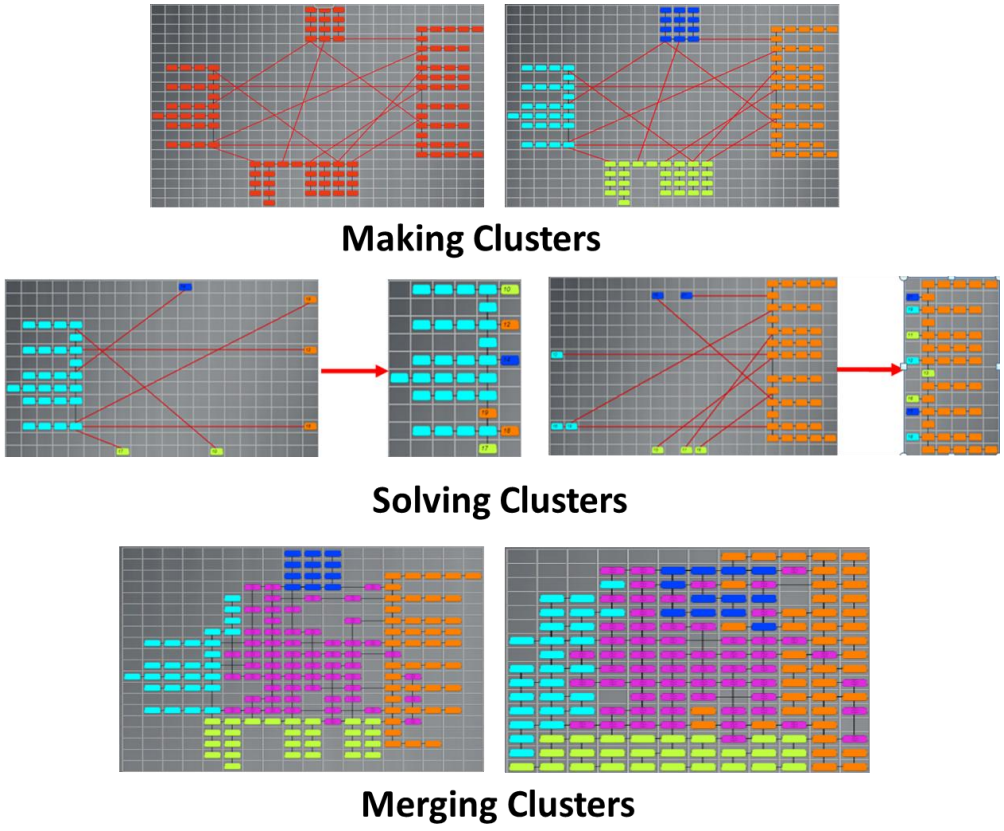


Figure 7 (b). A team is making clusters, solving them, and merging them to get the final solution.

Figure 7. A team is given an initial graph, players divide the graph into clusters, solve clusters, and merging clusters to get the final mapping solution. This example is for 4Way1Hop architecture and L1 benchmark.

REFERENCES

[1] J. Cong, "Era of customization and specialization," Keynote Speech, IEEE International Conference on. Application-specific Systems, Architectures and Processors (ASAP), 2011.

[2] Y. Kim, M. Kiemb, C. Park, J. Jung, and K. Choi, "Resource sharing and pipelining in coarse-grained reconfigurable architecture for domainspecific optimization," in Proceedings of the conference on Design, Automation and Test in Europe - Volume 1, ser. DATE '05, 2005, pp.12–17.

[3] M. Hamzeh, A. Shrivastava, and S. Vrudhula, "Epimap: Using epimorphism to map applications on cgras," in Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE, 2012, pp. 1280–1287.

[4] J. Cong, "Era of customization and implications to eda," Keynote Speech, Synopsys University Reception, 48th Annual Design Automation Conference (DAC), 2011.

[5] H. Park, K. Fan, S. A. Mahlke, T. Oh, H. Kim, and H.-S. Kim, "Edgecentric modulo scheduling for coarse-grained reconfigurable architectures," in Proceedings of the 17th international conference on Parallel architectures and compilation techniques, ser. PACT '08, 2008, pp. 166–176.

[6] J. W. Yoon, A. Shrivastava, S. Park, M. Ahn, and R. Jeyapaul, "SPKM : A novel graph drawing based algorithm for application mapping onto coarse-grained reconfigurable architectures," 2008 Asia and South Pacific Design Automation Conference, pp. 776–782, Jan. 2008. [Online].Available:http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4484056

[7] G. Mehta, C. Crawford, X. Luo, N. Parde, K. Patel, B. Rodgers, A. K. Sistla, A. Yadav, and M. Reisner, "Untangled: A game environment for discovery of creative mapping strategies," ACM Trans. Reconfigurable Technol. Syst., vol. 6, no. 3, pp. 13:1–13:26, Oct. 2013. [Online].Available: http://doi.acm.org/10.1145/2517325

[8] "Winners of the 2012 international science & engineering visualization challenge," http://www.nsf.gov/news/special reports/scivis/winners 2012.jsp , National Science Foundation, 2013.

[9] "2012 international science & engineering visualization challenge," http://www.sciencemag.org/site/special/vis2012/ , Science, 2013.

[10] S. C. Goldstein, H. Schmit, M. Budiu, S. Cadambi, M. Moe, and R. Taylor, "Piperench: A reconfigurable architecture and compiler," in IEEE Computer, vol. 33, no. 4, April 2000.

[11] C. Ebeling, D. Cronquist, and P. Franklin, "RaPiD – Reconfigurable Pipelined Datapath," Field-Programmable Logic Smart Applications, New Paradigms and Compilers, pp. 126–135, 1996. [Online]. Available: http://www.springerlink.com/index/f45122367533h852.pdf

[12] E. Mirsky and A. Dehon, "Matrix: A reconfigurable computing architecture with configurable instruction distribution and deployable resources," in Proc. of IEEE Symposium on FPGAs for Custom Computing Machines (FCCM), 1996.

[13] J. R. Hauser and J. Wawrzynek, "Garp: A mips processor with a reconfigurable coprocessor," in IEEE Symposium on FPGAs for Custom Computing Machines, K. L. Pocek and J. Arnold, Eds. Los Alamitos, CA: IEEE Computer Society Press, 1997, pp. 12–21.

[14] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications," IEEE Transactions on Computers, vol. 49, no. 5, pp. 465–481, 2000.

[15] T. Miyamori and K. Olukotun, "Remarc: Reconfigurable multimedia array coprocessor," in IEICE Transactions on Information and Systems E82-D, 1998, pp. 389–397.

[16] R. Hartenstein, M. Herz, T. Hoffmann, and U. Nageldinger, "KressArray Xplorer: a new CAD environment to optimize reconfigurable datapath array architectures," in DAC, vol. 2. Ieee, 2000, pp. 12 163–168. [Online].Available:http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=835089

[17] M. Taylor, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, and J. Kim, "Evaluation of the raw microprocessor: an exposed-wire-delay architecture for ilp and streams," in Computer Architecture, 2004. Proceedings. 31st Annual International Symposium on, june 2004, pp. 2 – 13.

[18] F. Bouwens, M. Berekovic, A. Kanstein, and G. Gaydadjiev, Eds., Architecture exploration of the ADRES coarse-grained reconfigurable array, ser. Reconfigurable Computing: Architectures, Tools and Applications. Springer, 2007.

[19] M. Lanuzza, S. Perri, P. Corsonello, and M. Margala, "A new reconfigurable coarse-grain architecture for multimedia applications," in Adaptive Hardware and Systems, 2007. AHS 2007. Second NASA/ESA Conference on, aug. 2007, pp. 119 –126.

[20] Y. Kim and R. N. Mahapatra, "A new array fabric for coarse-grained reconfigurable architecture," in DSD, 2008, pp. 584–591.

[21] C. Liang and X.-M. Huang, "Mapping parallel fft algorithm onto smartcell coarse-grained reconfigurable architecture," in ASAP, 2009, pp. 231–234.

[22] M. Garey and D. Johnson, "Crossing number is NP-complete," SIAM Journal on Algebraic and Discrete Methods, vol. 4, no. 3, p. 312, 1983. [Online].Available:http://link.aip.org/link/SJMAEL/v4/i3/p312/s1n&Agg=doihttp://link.aip.org/link/?SJMAEL/4/312/1

[23] V. Tehre and R. Kshirsagar, "Survey on coarse grained reconfigurable architectures," International Journal of Computer Applications, vol. 48, no. 16, June 2012.

[24] K. Choi, "Coarse-grained reconfigurable array: Architecture and application mapping," IPSJ Transactions on System LSI Design Methodology, vol. 4, pp. 31–46, 2011.

[25] G. Theodoridis, D. Soudris, and S. Vassiliadis, "A survey of coarse-grain reconfigurable architectures and cad tools," in Fine- and Coarse-Grain Reconfigurable Computing, S. Vassiliadis and D. Soudris, Eds. Springer Netherlands, 2008, pp. 89–149.

[26] R. Hartenstein, "A decade of reconfigurable computing: a visionary retrospective," Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001, pp. 642–649, 2001. [Online]. Available:http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=915091

[27] ——, "Coarse grain reconfigurable architecture (embedded tutorial)," in Proceedings of the 2001 Asia and South Pacific Design Automation Conference, ser. ASP-DAC '01. New York, NY, USA: ACM, 2001, pp. 564–570.

[28] G. Mehta, K. Patel, N. Parde, and N. Pollard, "Data-driven mapping using local patterns," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 32, no. 11, pp. 1668–1681, 2013.

[29] A. DeOrio and V. Bertacco, "Human computing for eda," in Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE. IEEE, 2009, pp. 621–622.

[30] V. Bertacco, "Humans for eda and eda for humans," in Proceedings of the 49th Annual Design Automation Conference. ACM, 2012, pp. 729–733.

[31] The Assembly Line, "Pipe mania," 1989.

[32] J. Krzemien, A. DeOrio, and V. Bertacco, "Funsat - multi-player," http://funsat.eecs.umich.edu/, 2011.

[33] V. Betz, "VPR: A new packing, placement and routing tool for FPGA research," Field-Programmable Logic and Applications, pp. 1–10, 1997. [Online].Available:http://www.springerlink.com/index/1673hwlmm7720606.pdf

[34] A. Sistla, N. Parde, K. Patel, and G. Mehta, "Cross-architectural study of custom reconfigurable devices using crowdsourcing," in Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2013 IEEE 27th International, 2013, pp. 222–230.