Coarse Grained Reconfigurable Architecture

Akeem Edwards

July 29 2012

Abstract:

This paper examines the challenges of mapping applications on to a Coarse-grained reconfigurable architecture (CGRA). Through crowd sourcing through a simple game and using the movement data from successful players the research project plans to produce better mappings, or improve existing algorithms through common patterns by the most successful players.

This paper also examines the case studies performed in this research project such as improving an existing algorithm used to map graphs on a CGRA, and to be compared to successful player graphs. This paper also examines a new game types that models a heterogeneous CGRA.

Introduction

Coarse-grained Reconfigurable Architectures (CGRA) have emerged as a promising reconfigurable platform, by providing operation-level programmability, wordlevel, datapaths, and area-efficient routing switches which is also very powerful[1], also promising for achieving energy-efficient flexible designs for an application domain. In order to use the CGRA's for practical applications there is a need for smart algorithms that are able to implement applications of interest onto these fabrics. This research is focused on discovering new mapping strategies for customized coarse-grained devices through crowd-sourcing. The main thrust of this research is to develop a science game to discover better mapping algorithms by making use of human intuition and ability to recognize patterns and opportunities even in complex problems. Players are presented with successively more difficult mapping problems in a game environment, and the vast dataset of players' moves is analyzed to recognize common patterns used by successful game players. The insights gained from strategic moves humans make while solving problems based on their visual intuition and experience can be used to discover new mapping approaches that are beyond what can be conceived with traditional algorithms. New energy-efficient architectures and mapping algorithms that are developed in this research will spur development of a broad range of portable/wearable computing applications critical to health, safety and security, personal multimedia, and aerospace.

2. CGRA Architecture.

A CGRA is basically an integrated circuit with an array of Processing Elements (PE) such as arithmetic logic units. The PEs are connected with each other by routing buses (figure 1), and thus a PE can use the results of their neighboring PEs. CGRAs can fully exploit the parallelism in an application, and therefore they are extremely well suited for the applications that require very high throughput [1].

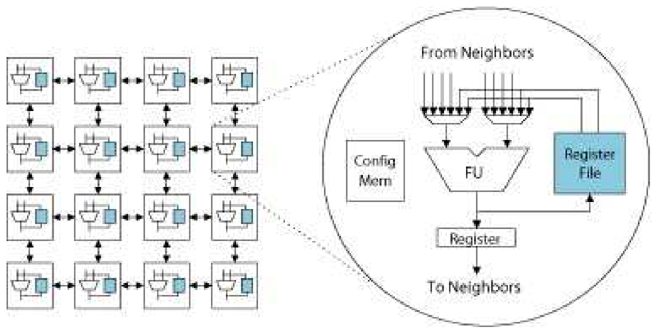

Figure 1 illustrating a Coarse grained reconfigurable architecture [2].

This Research examines two types of interconnections between PE; the first is called the Stripe architecture PE are connected to each other through a multiplexer, the second is called the Mesh architecture where the CGRA is interpreted as a 2D mesh structure, and as a result PE's are directly connected to their neighbors figure 2 shows the representation of each architecture in the game. Each architecture has quite a few of variations represented in the game that will be discussed later in this paper.

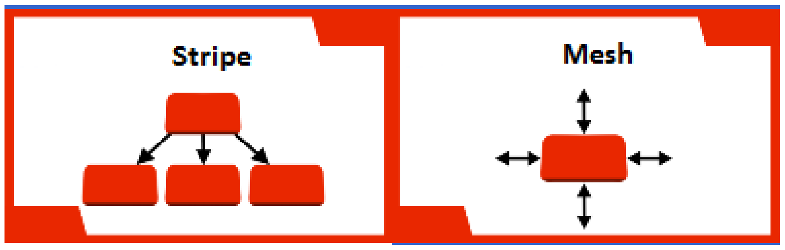

Figure 2 showing the different architectures represented in the game. The stripe architecture of the left and the mesh on the right

One essential component that is required to provide great performance in CGRA is the compiler. The compiler needs to be able to map applications onto a CGRA using the least amount of PE's as possible on to the given architecture.

3. Untangled Game Project

There are many compilers that are able to map applications onto a CGRA, but many of the compilers are unable to map applications, even though a mapping exists, and are using too many processing elements (PEs) to map an application, mapping an application onto a CGRA to minimize the number of resources has been shown to be NP-complete [1]. The research project has two main parts, the first part is a web based game that allows players to solve graphs onto different architectures, each graph representing an application to be mapped onto a CGRA, and then analyzing that data. With these two objectives the major goal of this project will be to compare algorithms generated from players to traditional algorithms, and from these algorithms or common patterns generated from players try to improve traditional algorithms.

Players are presented with successively more difficult mapping problems in a game environment; with each graph players are to try to shrink the graphs and to reduce the length of connections between each node. Not only are players reducing the sizes of each node but they will also have to abide to the connectivity constraints (or rules) in each of the CGRA architectures also known as violations. While players are busy playing the game their moves will be analyzed, with successful players having their moves analyzed for common patterns.

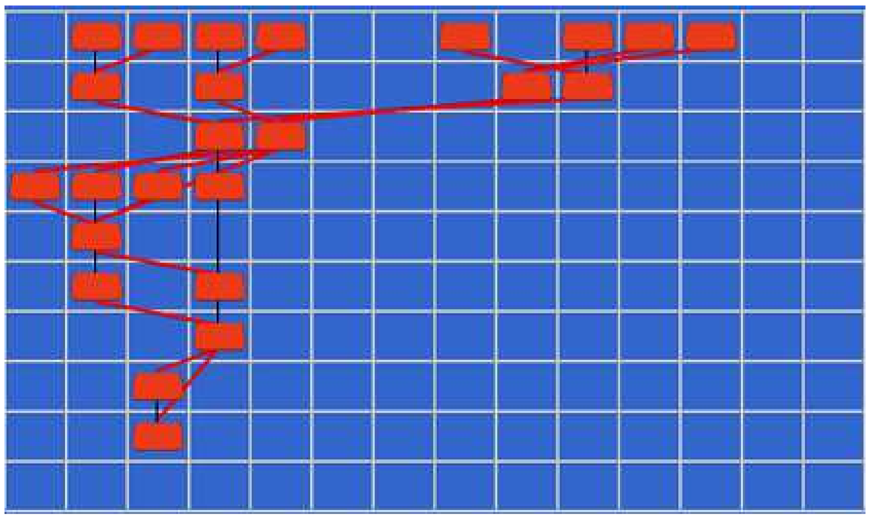

Figure 3 an unfinished graph filled with violations

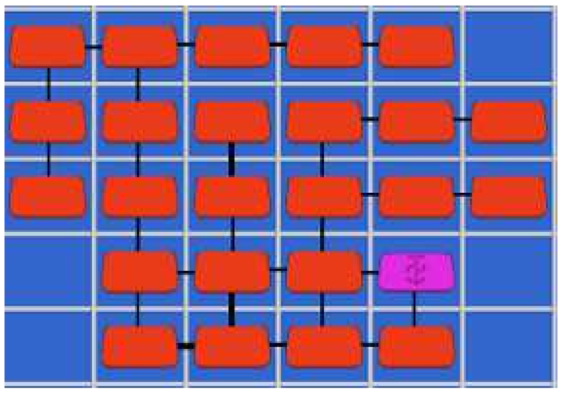

Figure 4 illustrating a finished graph with no violations.

The game provides players with two main architectures as discussed before the stripe based architecture where nodes are connected through multiplexors, and the mesh architecture where the graph is interpreted as a 2D mesh structure. Both stripe and mesh have many variations in the game. The stripe architecture has another variation called DR for dedicated-pass-gates (nodes used for transferring data). In the mesh architecture the player is presented with a few more variations such as the ability to for nodes to connect with other nodes directly above, below to the left and right of them. This is called four way, other variations come in the form of other nodes being able to hop or connect to nodes two steps away from them (4 way 2 hop) figure 5 shows all the variations of the mesh architecture represented in the game.

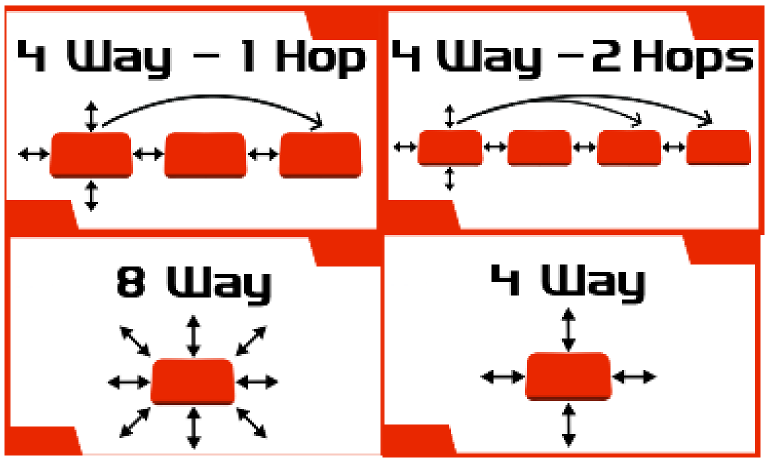

Figure 5 illustrating the variations of the mesh architecture in the game

4 Case Studies

The rest of the paper will discuss the case studies performed during research. Since this research is being worked on by a big team, only the case studies I worked on will be covered. Some of the major case studies performed included bettering algorithms used for mapping graphs that will be compared to successful player's graphs. The other major case study involved prototyping for a new game type that models a heterogeneous CGRA, a new game type to study players moves to be analyzed.

4.1 Simulated annealing placement

Simulated annealing is perhaps the main algorithm compared to successful player's graphs. In this project we use simulated annealing as a good approximation of graphs that use minimal resources on a CGRA. The next step is to take these graphs and look for any improvements that can be made in each graph. After improvements have been made, the cost function for the algorithm was fine tuned to reflect each of the improvements noted which will provide a next iteration which better paces graphs onto the grid or CGRA. The Simulated annealing algorithm will not only have to place graphs as efficiently as possible but will have to abide to the connectivity constrains or violations as well. Ideally the algorithm will take a graph as input and given specific constrains output a valid graph that also uses little resources as possible.

The first iteration of the simulated annealing algorithm was performed on the stripe architecture levels in the game and stored into a database. Then the finished graphs of were loaded up and simple improvements were to be made on each of the graphs which were little adjustments that could reduce the amount of resources needed on a CGRA. Although the algorithm was able to perform some valid mappings for a few graphs, there were some final graphs that were produced with violations, and the algorithm also didn’t seem to produce graphs with any width considerations or constraints. Figure 6 shows an example of the final graph as an output

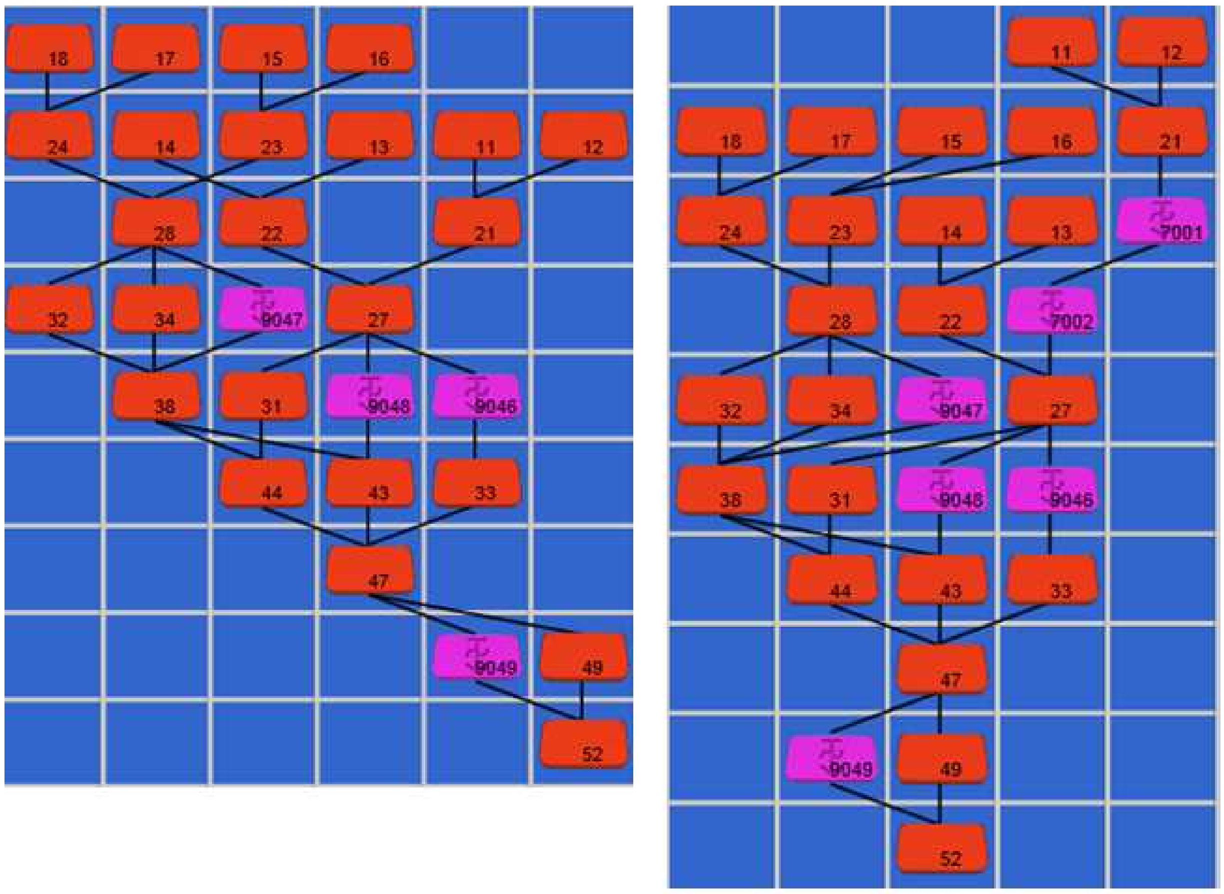

Figure 6 output of simulated annealing graph results on the left along with improvements made on the right.

To measure the performance of this algorithm each cell in the grid will be interpreted as a processing element. In this case reach the final mapped graph will be using a 6 x 9 area a total of 54 PE's, compared to the improved version (figure 7) which is using an array of 5 x 10 or a total of 50 PE's. Table 1 shows a comparison between the area of simulated annealing algorithm graph and the improved version graphs for all the levels in the game. In some levels such as the previous graph examined the improvements were minimal while there are others that had huge improvements that were done to them, and finally there were only two where there wasn't any improvement available.

Level	Amount of nodes	Simulated annealing results	Improved results
Encoder	36	144	112
Decoder	29	98	65
Gsm	28	90	90
idct col	61	144	144
idct row	52	120	110
Laplace	29	96	72
Sobel	24	54	50

Table 1 showing a comparison between the simulated annealing algorithm graphs and the improved versions.

The next iteration of the simulated annealing algorithm was deployed onto a variation of the mesh architecture 4 way 1 hop. In this setup each node has access to the neighboring nodes from above, below, left and right of it, and is able to access nodes one node a away from the same direction as well. For this experiment we first ran the new and improved cost function several times on each level, with the results we recorded the best, worst and the average cases. For each case recorded each graph was examined for small improvements that could be made to reduce the amount of resources needed. With the new cost function there were much less recorded violations that the previous iteration and the reduction of resources need in the graph was also reduced as well. At the algorithms best there were no improvements that could be performed to reduce the area of the graphs on the grid for any level in the game. On average and worst cases there were only two levels that allowed for improvement.

4.2 Heterogeneous prototype

The next major case study performed in the research was prototyping for a new game type that models a different version of a CGRA. As explained before a CGRA is made up of an array of PEs, one assumption that is given to players is that every processing element will be able to provide any operation given to them. This type of CGRA is considered a Homogenous Architecture every Processing element is identical to the other, the next major step in this project has been to model a Heterogeneous CGRA where different PE's provide different operations.

This prototype uses the same graphs as the homogenous game type as well as the same operations. The CGRA in this prototype distinguishes between two types of operations multipliers and non-multipliers (Multiplication is an expensive operation and is separated from the rest). In this experiment each level is given a fixed amount of multipliers on each row. Each node is drawn with a symbol of its operation, and different colors depending on the type of operation required. For the prototype nodes requiring multiplication are drawn in green, while the rest of the nodes are drawn orange (later in the prototype design red). The Grid also has also been changed to reflect this new architecture as well, with columns that will be able to provide multiplication in green and the rest in red as well. Figure 7 shows the new game type designed in the prototype. The architecture in the prototypes is based off of the variation of the mesh architecture 4 way 2 hop. Players are only able to place nodes in the correct cells of the grid (for example green nodes can only be placed in green columns).

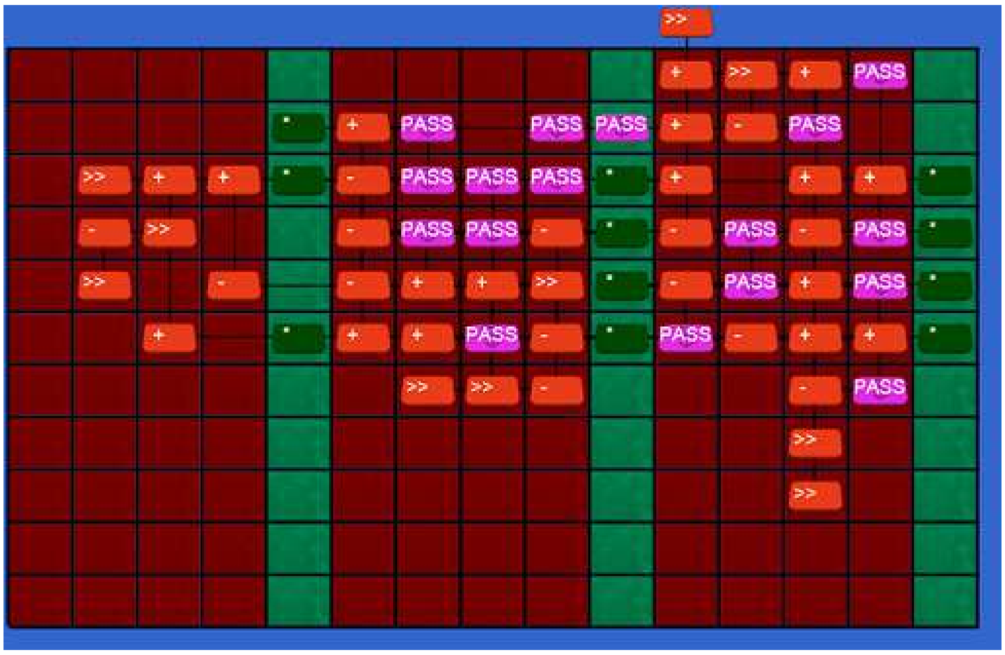

Figure 7 illustrating the prototype for the heterogeneous architecture a multiplier row is given at every 5 interval.

With this new game type there is also a greater difficulty curve associated with this new game type. Part of creating this new prototype was to measure the difficulty curve associated with the increase in difficulty level, by experimenting with the amount of intervals a multiplier column will appear. The difficulty level seems to be unbearable with more multiplier columns provided. This is a result of less space being provided to non-multiplier nodes which stretches the connections between nodes.

Conclusion

A lot of lessons have been learned while working on this project, and although there are not any results ready for the final project, there have been many milestones that have been made through the work of major case studies performed a great framework has been laid out for the final project to achieve the final goal of the project.

References

[1] W. Yoon, A Shrivastava, S Park, M Ahn, and Y Paek. A Graph Based Spatial Mapping Algorithm for Coarse-Grained Reconfigurable Architectures. IEEE Transactions on Very Large Integration (VLSI) Systems. 1565 – 1578 November 2009.

[2]Coarse-Grained Reconfigurable Architecture. Compilers Creating Custom Processors(CCCP) http://cccp.eecs.umich.edu/research/cgra.php