DYRECT – A **DY**namic **REC**onfiguration **T**ool for Multicomputer Systems

Leslie D. Fife, Gopal Racherla and Steven E. Killian
School of Computer Science
University of Oklahoma
Norman, OK 73019

Abstract

In this paper we discuss the design and implementation of a tool to teach dynamic reconfiguration of multicomputer systems. DYRECT is a graphical tool used to visualize and analyze various topologies like stars, cubes, meshes, trees, and rings. Various existing re-configuration algorithms have been incorporated as a part of DYRECT's reconfiguration library. This tool allows the user to interactively design new reconfiguration algorithms. DYRECT has been implemented on PC-DOS machines running Microsoft Windows.

1. Introduction

Dynamic reconfiguration of multicomputer systems is an important topic in advanced computer architecture courses. It is anticipated that this topic would be taught in a computer architecture course at the senior or graduate level. However, even for advanced students dynamic reconfiguration can be difficult to visualize and understand. The complexities associated with reconfigurations suggest the use of a graphical tool to present, simulate, and experiment with different re configuration approaches. This tool would demonstrate dynamic re configuration in a clear and concise manner and could be used by the students outside of the classroom to supplement lectures and assignments. The lack of specific educational tools in this area has been the main motivation behind the development of DYRECT – DYnamic REConfiguration Tool for multicomputer systems.

DYRECT is an interactive tool that simulates dynamic reconfiguration networks and enhances the instruction of reconfiguration algorithms and strategies. DYRECT provides a user-friendly Graphical User Interface (GUI) to the dynamic reconfiguration simulator for interacting with the student. This allows the student to investigate the results of different reconfiguration strategies. The simulator has several common strategies available directly through a set of library files. Furthermore, new scenarios may be constructed and analyzed by the simulator. This ability to create and modify new and existing re configuration algorithms assists the student in understanding the effects of each a reconfiguration strategy. This will enhance the knowledge gained in the classroom, providing an additional dimension to the students' learning.

2. Related Work

Dynamic architectures were introduced by Kartashev in the early 1970's. A dynamic architecture can be viewed as a black box consisting of processor and memory units with differing computing structures such as multicomputers, arrays, and pipelines. Recon-figuration can be either architectural or fault-tolerant. DYRECT focuses on architectural reconfiguration of multicomputer systems which is an ensemble of stand- alone processors with distributed memory. DYRECT assumes the presence of a supervisory system monitor which decides when and how the reconfiguration occurs. A dynamic reconfiguration can assume a set of configurations during the program execution and improve the performance of the system. Rings, for ex- ample, are quite useful for pipelined computations and control algorithms while stars and trees are suited for divide-and-conquer algorithms like sorting. DYRECT uses the Shift Register with Variable Bias (SRVB) scheme proposed by Kartashev [1]. Each processor has a unique node number that is decided *a priori* by the system monitor. The monitor also broadcasts a bias number to all the nodes. The user can choose the resultant architecture and its parameters. Based on this information, each node performs a local computation to get the node numbers of its neighbors that it will connect to at the time of reconfiguration.

There has been some work in this area to date. Kartashev devised strategies for re configuring trees and rings [1] Biswas designed reconfigurable m-ary trees for any value of m [2]. Ruskey used transpositions to generate binary trees [3]. Racherla and Radhakrishnan developed parameterizable algorithms for rings and trees. These include reconfiguration algorithms for m-ary trees of variable height and a forest of trees with variable height and branching factor [4]. Lin and Wu worked on polymorphic and partitionable multicomputer systems creating meshes, trees, and rings [5]. Biswas and Srinivas designed a reconfiguration strategy for the binary tree configuration [6].

3. Description

DYRECT is composed of three primary subsystems. These are the Reconfiguration Libraries, Graphical User Interface, and the Reconfiguration Simulator. The modular design of each subsystem allows for rapid integration of reconfiguration algorithms and strategies, as well as improvements to either the GUI or reconfiguration simulator, without impacting on the other parts of the system.

DYRECT does not address the mapping of programs onto processors following reconfiguration. A module is being designed that intelligently performs this mapping. Using this module, a user can map his program, written in a higher level parallel language on to a multiprocessor system. Work is also in progress to add performance metrics to help the user in choosing an appropriate algorithm for reconfiguration depending on the application.

3.1 Reconfiguration Libraries

The first subsystem contains the reconfiguration libraries. Each reconfiguration strategy is included as an algorithm in a re configuration library file. The reconfiguration libraries are grouped by network topology. The topologies include Trees, Rings, Stars, Meshes, and Cubes.

1. Trees:

- Kartashev and Kartashev [1]: This algorithm provides synthesis of both single and multiple tree structures. For single tree structures, it shows how to find the root and other nodes if the bias is given.
- Srinivas and Biswas [21[61: This algorithm produces m-ary tree structures. This approach is based on the extended multistage interconnection network which is a generalization of the shuffle exchange network. The main advantages of this algorithm include fast reconfiguration, simplified hardware in the nodes and the multistage interconnection network (Figure 1).
- Ruskey [3] : This algorithm produces trees by bit transpositions and inversion. It produces all binary trees with a fixed number of nodes using a gray code type generation.
- Racherla and Radhakrishnan [4] : This is an extension of the Srinivas-Biswas algorithm that allows parameterization of the height of the trees and their branching factor. It incorporates the advantages of all of the other algorithms described above.

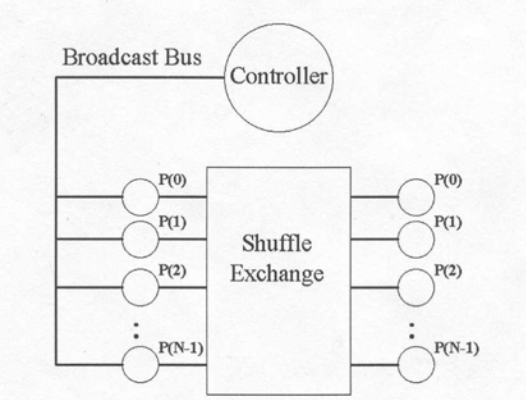

Figure 1: Shuffle Exchange Multistage Interconnection Network

2. Rings:

- Kartashev and Kartashev [1]: This algorithm produces single and multiple rings. It produces simple and compound rings depending on the input. Com- pound rings are rings that are inter-linked.
- Racherla and Radhakrishnan [4]: This algorithm parameterizes the ring size and number of rings to produce single rings.
- Lin and Wu [5]: This algorithm produces double and chordal rings and is parameterized on the ring size. It also produces single rings. This method uses a matching technique on the interconnection structure and the task graph.

3. Stars:

- Racherla and Radhakrishnan [4]: This algorithm produces stars based on the input parameters which are the size of the star and the number of stars.

4. Meshes:

- Lin and Wu [5]: This method produces a mesh given the dimensions. This work assumes a polymorphic and partitionable multiprocessor system. It relies on dynamic creation of subsystems.

5. Cubes:

- Racherla and Radhakrishnan [4]: This algorithm produces cubes of a given dimension. All the re configurations are done in parallel.

The user can use DYRECT to study, analyze, com- pare and contrast these predefined, as well as user specified reconfiguration algorithm thus gaining in- sight into the specifics of each algorithm. An important part of the DYRECT system is the ability to design and test new configuration algorithms. New methods are entered through the Equation editor (Figure 2), and can be saved for future use.

3.2 Graphical User Interface

DYRECT provides an easy to use GUI which accepts user input for all relevant user specified parameters for a given reconfiguration. As mentioned, DYRECT also allows the user to design and implement their own re configuration strategies, and have them simulated by the tool. These strategies can be saved in library files for later reuse, modification, and analysis.

The reconfiguration equation editor (Figure 2) is a tool for the user to build and test new reconfiguration equations for various topologies. This tool is invoked from the main menu under the Tools option. The tool provides basic primitives to build reconfiguration equations. These include binary and register operations. The binary operations consist of AND, OR, XOR, NOT, and Bit Flip. The register operations are shifting, partitioning, merging and building. Shifting can either be left or right as well as circular or non-circular. By using all the primitives the user can build various reconfiguration equations and test them. The modularity of the editor reflects the manner in which the entire system was constructed.

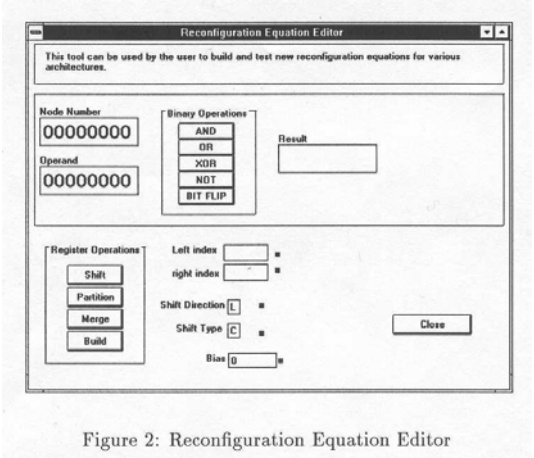

Figure 2: Reconfiguration Equation Editor

The GUI prototype was designed as a Windows application, running in a PC-DOS environment under Microsoft Windows. The GUI is designed as a separate subsystem. This allows rapid porting to other systems. For a new windowing environment, a new GUI would have to be provided. However, the libraries and simulator would only require recompilation on the new platform. As the crux of the work is done within the simulator, changing interfaces will have minimal impact on the tool as a whole.

3.3 Reconfiguration Simulator

The third subsystem is the reconfiguration simulator. The reconfiguration simulator performs the primary work, and is the central subsystem of DYRECT . The simulator performs the processing required for each dynamic reconfiguration. The simulator works as a standalone program. The simulator is started by the GUI when the user chooses to perform a reconfiguration. The GUI provides any user inputs required. The simulator uses these inputs and one of the reconfiguration library files to perform the necessary processing. The simulator is designed as a C program.

The simulator is designed so that the reconfiguration equation that is applied to each node is identical, with the exception of each node's unique node number . Because of this, the application of a re configuration equation on each processor node can be parallelized. The determination of each node's neighbors is a local operation and is independent of the determination of neighbors by the other processor nodes.

4. Example

To illustrate how DYRECT works a sample session has been captured and is presented here. Given below is an example reconfiguration session.

Upon starting DYRECT, the primary window appears as shown in Figure 3. This window initially displays a default configuration. The default values may be changed using the Options menu choice. In this ex- ample, a configuration of 16 nodes has been used as an illustration. Further, the nodes are not interconnected and are considered as standalone processors.

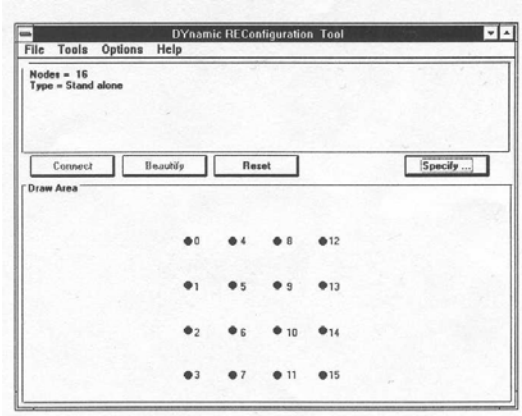

Figure 3: Initial Configuration

From the main DYRECT window the configuration specifications are made for the node reconfiguration. This is exhibited in Figure 4. For this example, a tree configuration has been selected. From this point further specifications are made based on the tree con- figuration. These include the number of nodes for the tree, the bias, tree height, branching factor, and the reconfiguration algorithm to be used. Once this information is specified the user is returned to the main DYRECT window (Figure 3).

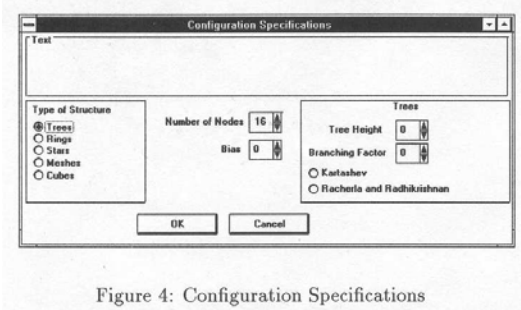

Figure 4: Configuration Specifications

On returning to the main window, the configuration information is updated and displayed in the text box at the top of the window. Figure 5 shows the information selected earlier in the text box. At this point the nodes are ready for reconfiguration, which may be achieved with the connect button. Figure 6 shows the state of the nodes after connection. The node connection is shown based on the reconfiguration information specified, albeit disorderly and confusing to visualize at this point.

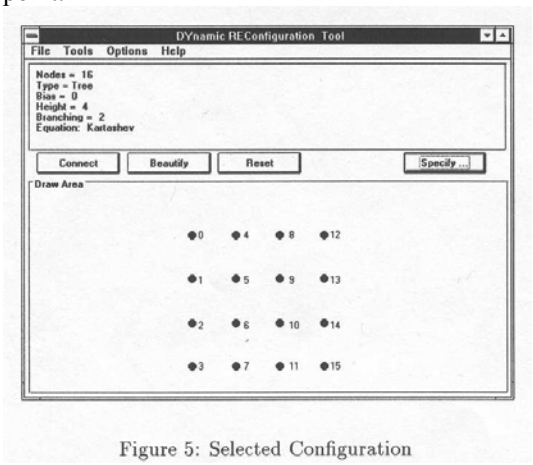

Figure 5: Selected Configuration

To clarify the connection reconfiguration exhibited in Figure 6, the beautify function is used. The performance of this function is dependent on the reconfiguration specifications. The rules used to rearrange the nodes are predetermined by the reconfiguration type and once the beautify function is invoked, the nodes are ordered based on the topology specific rules. The effects of the beautify function are shown in Figure 7.

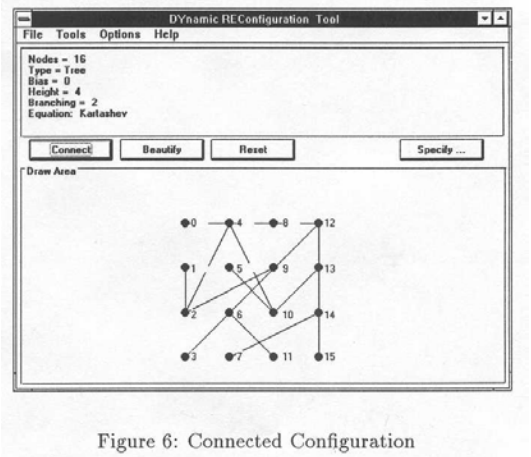

Figure 6: Connected Configuration

The example shown here was produced in Visual Basic version 3.0 for Windows, running under PC- DOS. The window designs were taken from versions initially developed for X Windows. The code for the interface is also written in Visual Basic.

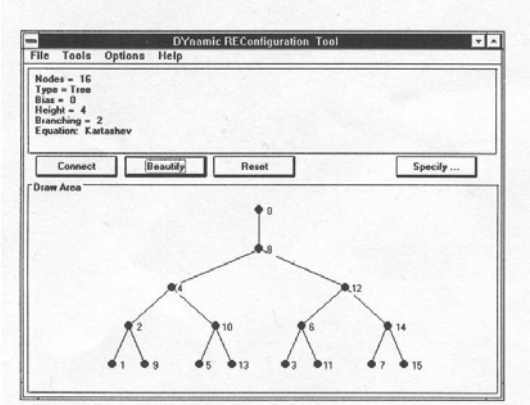

Figure 7: Beautified Output

5. Conclusions

The study of dynamically reconfigurable networks is an important topic in advanced computer architecture courses. The complexity of dynamic reconfiguration makes this a difficult topic for many students to visualize and understand. The use of a graphical tool can assist in the educational process and supplement the content of lectures, readings, and course assignments.

The DYRECT system was developed modularly as a collection of three primary subsystems. These subsystems are the re configuration libraries, graphical user interface, and the reconfiguration simulator. These systems can be modified and updated individually. The simulator was initially developed for the Unix environment. In an effort to make DYRECT available to a larger audience, it was ported to the PC-DOS environment.

DYRECT provides an important illustrative ex- tension to the standard lecture format of university courses in computer architecture. By providing an interactive tool, the learning process is enhanced, allowing the student to experiment and implement various approaches to reconfiguration. After each reconfiguration, the results can be analyzed and compared thus augmenting the student's knowledge of the benefits and weaknesses of the various reconfiguration approaches. This increases the ability of the student to understand and integrate the theory presented in class on dynamic reconfiguration of multicomputer systems with its application.

Acknowledgements

The authors would like to thank the Graduate College and the School of Computer Science at the University of Oklahoma for research funding. Special thanks are due to Prof. Rex Page for his encouragement and support. Thanks are also due to Prof. S. Lakshmivarahan and Dr. Sridhar Radhakrishnan for their insightful comments on our work.

References

[1] S. P. Kartashev, S. I. Kartashev, "Analysis and Synthesis of Dynamic Multicomputer Networks that Reconfigure into Rings, Trees, and Stars" *IEEE Transactions on Computers,* Vol. C-36, No. 7, July 1987.

[2] S. Srinivas, N. N. Biswas, "Design and Analysis of a Generalized Architecture for Reconfigurable m-ary Tree Structures" *IEEE Transactions on Computers,* Vol. 41, No.11, November 1992.

[3] F. Ruskey, et al, "Generating Binary Trees by Transpositions", *Journal of Algorithms,* Vol. 11, 1990.

[4] Gopal Racherla, R. Sridhar, "Parameterized Re-configuration of a General Architecture into Tree Structures and Binary Hypercube", *manuscript.*

[5] Woei Lin, Chuan-Lin Wu, "Reconfiguration Procedures for a Polymorphic and Partitionable Multiprocessor" *IEEE Transactions on Computers,* Vol. 35, No.10, November 1986.

[6] N. Biswas and S.Srinivas, "A Reconfiguration Tree Architecture with Multistage Interconnection Network" *IEEE Transactions on Computers,* Vol. 39, No.12, November 1990.