

Permutation Routing on Mesh model Parallel
Computers

Kaarthik Sivashanmugam
Dept. of Computer Science
University of Georgia

Topics to be covered:

1. Parallel Computers
2. Reconfigurable Interconnection networks
3. Mesh types
4. Permutation Routing
 - definition
 - algorithms in different models
5. Simulation of a PR algorithm using RMSIM
6. Hardware for Reconfigurable Switch
7. Summary of results
8. Future directions

Slides and Project proposal can be obtained from
<http://www.arches.uga.edu/~kaarthik>

Motivation:

- Need for parallelism in processing
- Upper bound for the computing power in a single processor is limited by speed of the processor
- Can be increased by integrating many processors
 - ~ Parallel computation
- Ways to interconnect?
- How to choose the type of interconnection?
- How to analyze parallel computers?
 - ~ Analyzing with algorithms
 - ~ Analyze connection complexity

About the paper:

- Studying the problem of permutation routing on different mesh models of parallel computation
- Permutation routing algorithms on mesh models with and without bus have been considered
- Simulating a simple algorithm in a simulator called RMSIM
- Hardware for reconfigurable switch

Introduction:

1. Reconfigurable networks

Reconfigurable Bus system.

- can be changed dynamically according to local or global information
- why?

2. Mesh models

Mesh with fixed row and column buses

Reconfigurable Mesh (Fig 1)

Mesh without bus (Fig 2)

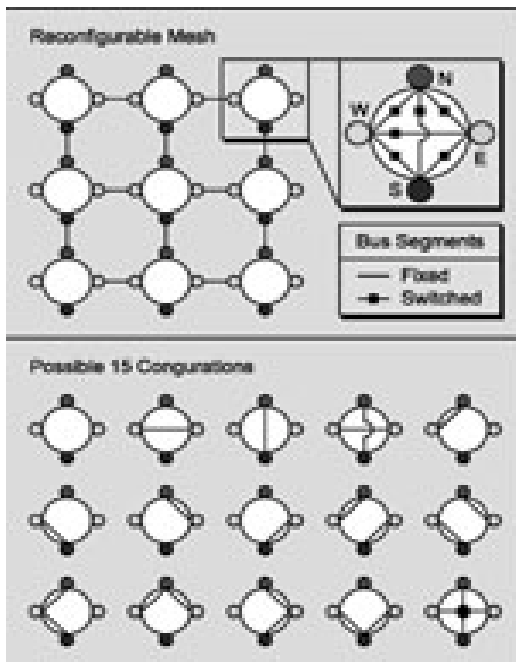


Figure 1

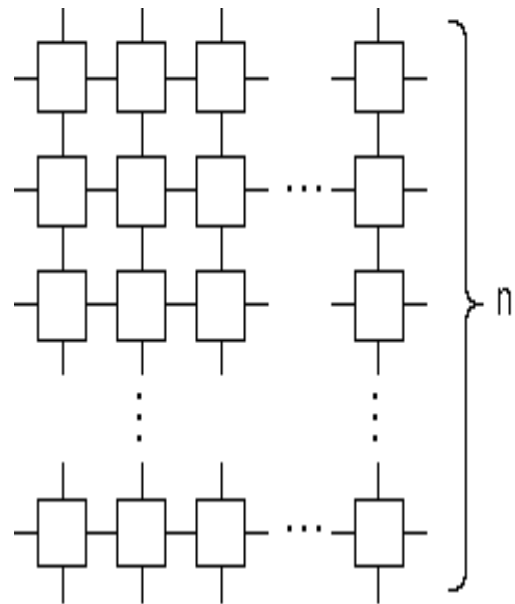


Figure 2

3. Packet Routing

- A single step of interprocessor communication
- Each node in the network has a packet of information to be sent to some other node.
- Circuit Switching Vs Packet Routing

4. Permutation Routing

- Special case of packet routing
- Each node is the origin of at most one packet and each node is the destination of no more than one packet
- Efficiency determined by number of communication steps(routing time), queue size

5. Significance of the problem

- a. Importance of interprocessor communication.
- b. To agree on the sub-problems each will work on.
- c. To see whether every one has finished its task.
- d. Speed of any parallel computer is primarily decided by the power of individual processing elements and efficiency of interprocessor communication.

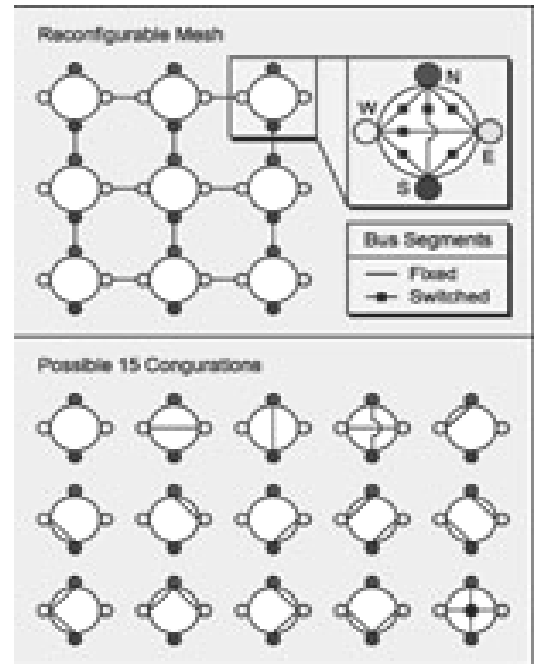
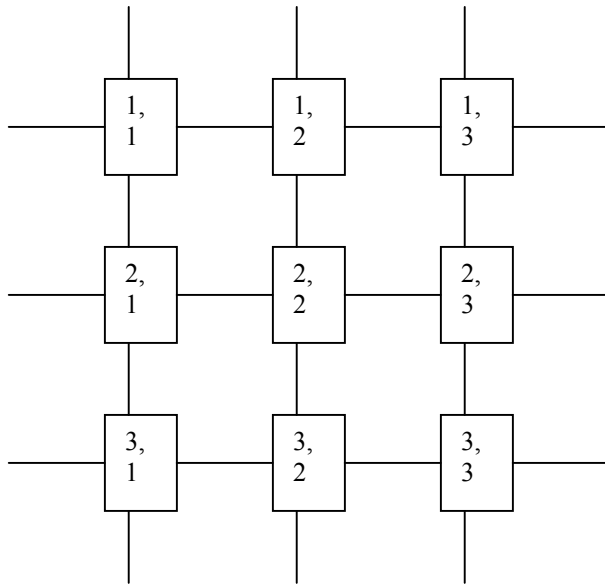
Goal

- to solve problems in a constant time interval
~ is it possible ?
- independent of problem size. ~ and this ?
- for a large number of problems.
~ AND what about this ?

Literature review/Works in this area/My work:

1. Most papers analyzed algorithms for Permutation routing on Meshes with and without buses. Refer [11], [13]
2. One paper discusses a C simulator for Reconfigurable Meshes. The name of the simulator is RMSIM. I will be using this in my analysis. Refer [12]
3. Some papers suggested different Reconfigurable models. Refer [1], [2], [7], [9]
4. My work considers a problem, “Permutation Routing”, analyzes the algorithms proposed by these papers, simulating an algorithm in the simulator and suggesting a switch for the reconfigurable meshes (for this problem?).
5. Most of the switches proposed so far for Meshes with reconfigurable buses are at “device level”. This paper suggests a switch at Digital components level (variant of a already proposed architecture?).

Problem Illustration:



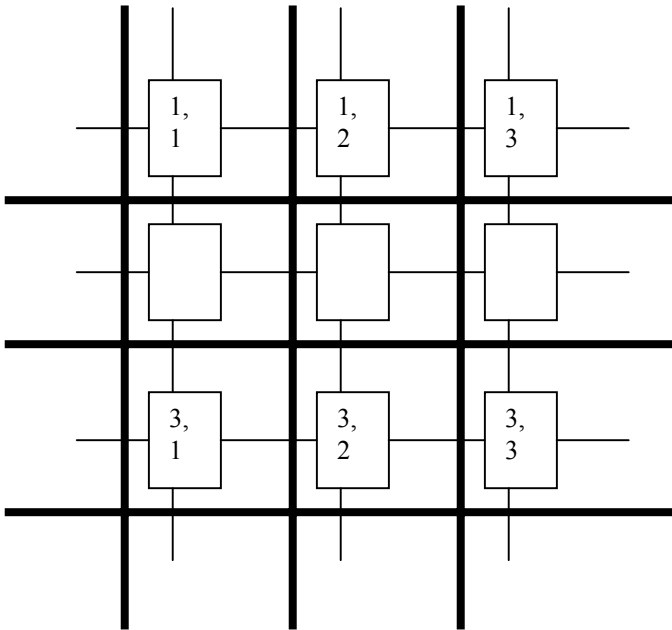
Simple Mesh

Reconfigurable Mesh

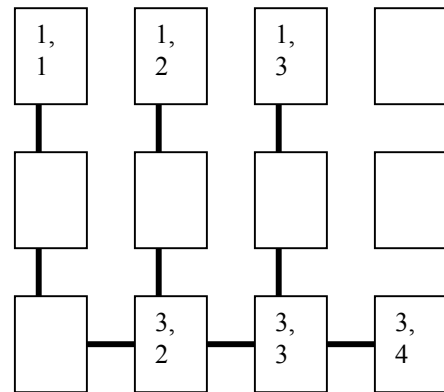
Routing (1,1) to (3,1) , (1,2) to (3,2) and (1,3) to (3,3)

1. Send (1,1) to (2,1), Send (1,2) to (2,2), Send (1,3) to (2,3)
 2. Send (2,1) to (3,1), Send (2,2) to (3,2), Send (2,3) to (3,3)
- 2 CLOCK CYCLES

Mesh with Bus



Mesh with Reconfigurable Bus



Is it really possible ?

Routing (1,1) to (3,1) , (1,2) to (3,2) and (1,3) to (3,3)

1. Send (1,1) to (3,1), Send (1,2) to (3,2), Send (1,3) to (3,3)
- 1 Clock cycle and very expensive

Routing (1,1) to (3,4) , (1,2) to (3,2) and (1,3) to (3,3)

Mesh with bus does not achieve routing in one clock cycle. But a Reconfigurable mesh achieves this.

Project details:

Assumptions

- dealing with atomic packet (store and forward model)
- these packets can be transmitted along the communication channel in one unit time.
- the packet not only contains message but also source and destination names.
- Ignoring Queue details, problems like Livelock, deadlock and starvation

PR on Meshes without buses:

SIMD:

$4N-4$ steps is the lower bound for almost any problem. This matches with the Simple Greedy algorithm for Permutation Routing.

Algorithm:

- All packets are sent to their destination columns along their rows
- Send all packets to destination through columns
- $O(n)$
- What about multiple destination in a column?

MIMD:

$2N-2$ is the lower bound diameter. Randomised Algorithms have been proposed to achieve this bound.

PR on Meshes with Row and Column buses:

~ CREW model

Off-line PR: Permutation is known in advance and hence optimal routing schedule can be precomputed.

Every Permutation can be routed in $n+1$ steps after scheduling has been computed.

Calculating the schedule:

1. Columns are processes $P_1, P_2, P_3 \dots$
2. Rows are Resources $R_1, R_2, R_3 \dots$
3. Before a packet is transmitted along row bus it is to be routed to each row using column bus. Hence each column bus to access row bus according to the destination of packets.

Eg. If k packets in column j need to reach row i then P_i needs to access R_j k times. These k steps can be scheduled in any order.

Finding minimum time schedule is Open Shop Scheduling problem.

Let D_{ij} be the demand by Process P_i for Resource R_j .

$$\sum_i D_{ij} = n$$

$$\sum_j D_{ij} = n$$

Minimum matching is computed from the maximum matching in the bipartite graph $G = (U, V, E)$

U - set of processes

V - set of resources

$E = (P_i, R_j)$ for $D_{ij} > 0$

The algorithm computes maximum matching M of G and schedules each process with its matched resource from D_{min} ($D_{min} = \text{minimal value in set } D_{ij} \text{ for } M$)

Remove D_{min} from D_{ij} and then recompute matching for new set.

This is repeated till D_{ij} becomes zero.

This schedule can have length of at most n

At most n matching has to be computed.

Maximum matching computation – $(n^{5/2})$

So the entire schedule can be computed in $n^{7/2}$ steps

$n^{7/2}$ is not a good figure to be used. So an algorithm can be designed in such a way that this value is reduced.

Ideally $n \rightarrow n^{2/7}$ will achieve best results.

An algorithm has been proposed that partition mesh into blocks of size $n^\alpha \times n^\alpha$ and the total number of blocks will be $n^{2-2\alpha}$ where α is some constant. The same algorithm is implemented. And here the packet need not reach the actual destination but should reach the block that contains the destination.

Here α decides the number of steps required for routing.

Reconfigurable Buses:

- ~ EREW model
- ~ all PEs connected to a single bus which can be segmented / disconnected by switches and made to act like several small buses
- ~ trivial bound is n

Interesting feature:

- On a reconfigurable $n \times n$ mesh n numbers can be sorted in constant time.
- $O(1)$, constant value hidden could be large, which depends on the Sorting algorithms we choose

RMSIM: Serial simulator for Reconfigurable Meshes

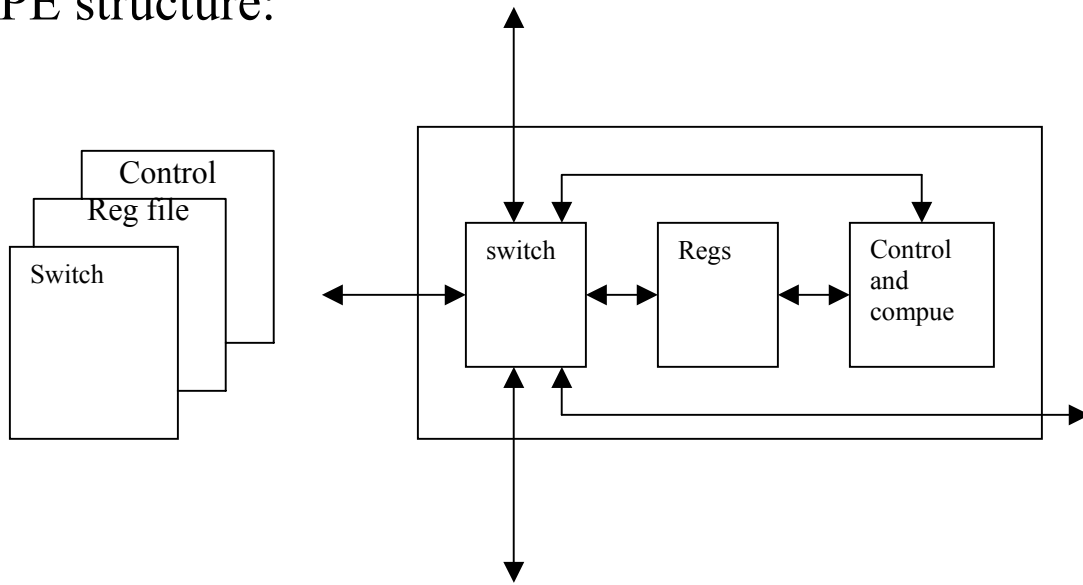
- written in C
- to study algorithms in 3-D reconfigurable mesh
- any axis can be selected for analysis
- has snapshotter to generate LATEX pictures of the planar segment of the simulated mesh in any step of program execution

Features:

- this operates in SIMD mode
- along with switches each PE has computing unit with fixed number of registers which can be specified during run time
- Single time step of an RM has following sub-steps
 - ~ Bus step: every PE switches internal connectors between IO ports
 - ~ Write step: along each bus one or more PE (called speakers) can transmit messages. If collision messages are discarded.
 - ~ Read step: some or all Pes connected to a bus read the message transmitted by a single speaker
 - ~ Compute step: a constant time local computation is done in each PE

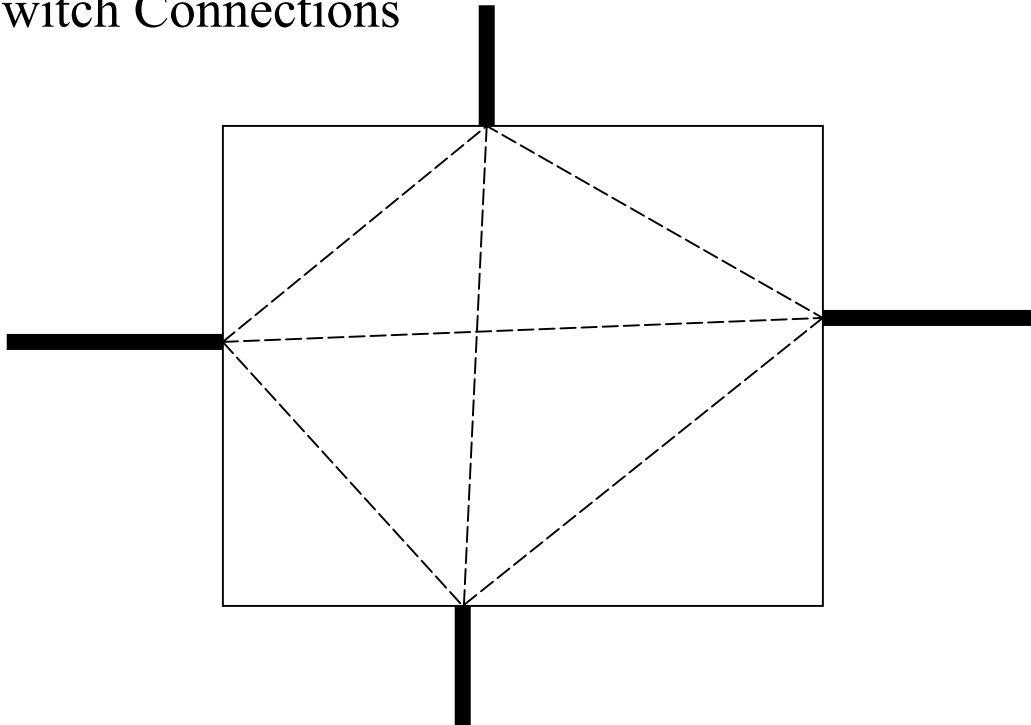
Switch hardware for Reconfigurable Meshes:

PE structure:



Control unit in PE?

Switch Connections



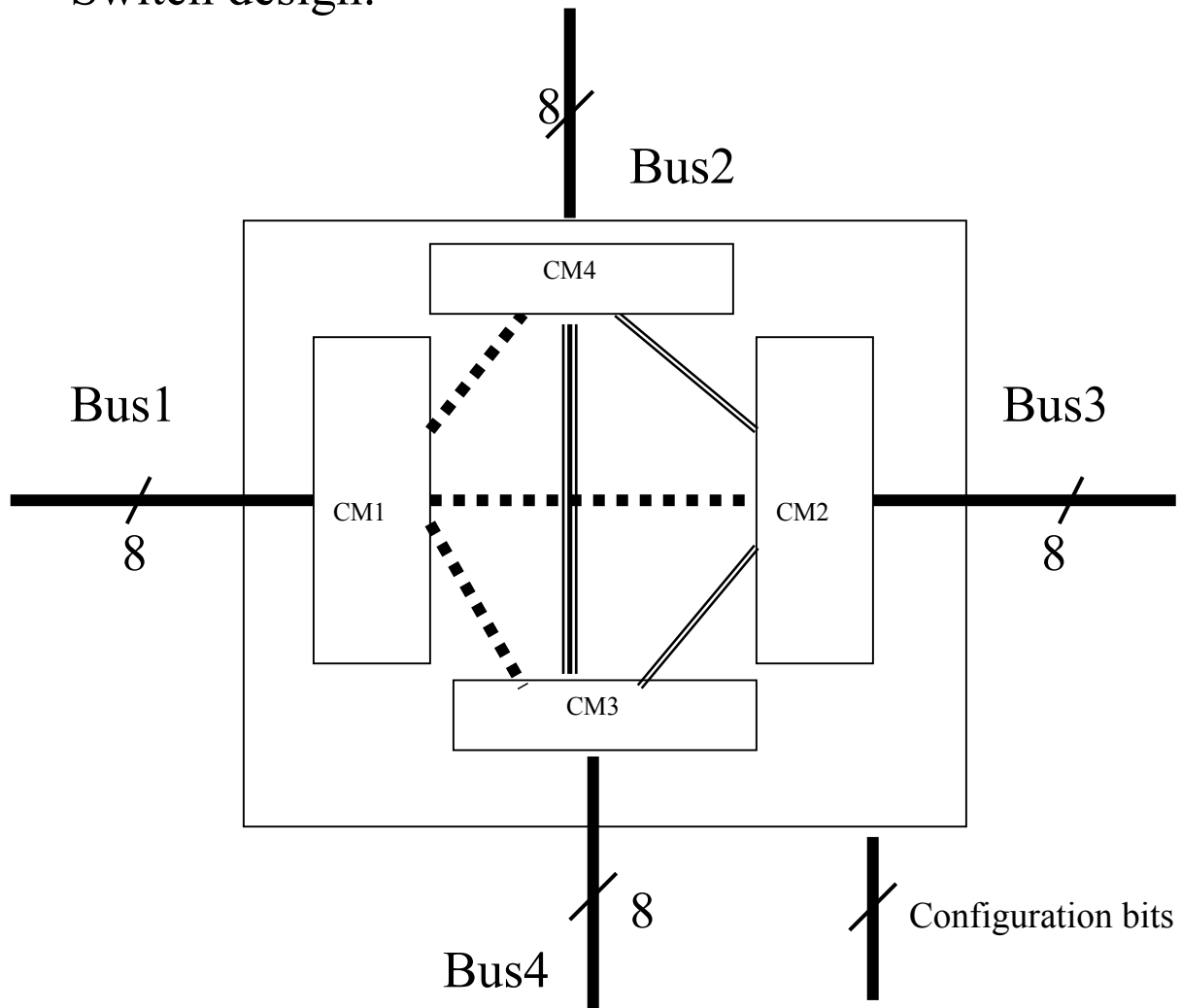
Design 1: PARBUS

Possible Configurations:

Design 2: RMESH

Example Configuration of the RMESH network:

Switch design:



Possible configurations:

1-2, 1-3, 1-4, 3-2, 3-4, 2-4, (1-3)+(2-4), (1-2)+(3-4), (1-4)+(2-3), No connection.

VHDL Implementation:

Tool used: asimut- Alliance tool.

Limitations:

- limited functionality
- needs others Alliance tools for advance design
- example: Bus design needs 'buseg' tool

Design:

Instead of buses this implementation has used bit-vectors and showed how connections will be established.

Configuration:	Instruction set:
1-2:	00110
1-3:	01110
1-4:	10110
3-2:	11000
3-4:	11100
2-4:	11111
(1-3)+(2-4):	01111
(1-2)+(3-4):	00100
(1-4)+(2-3):	10000
No connection:	11110

```

-- description generated by Pat driver v107
--           date : Mon Apr 22 22:38:01 2002
--           sequence : node
-- input / output list :
in      left1 (7 downto 0) B;
in      bottom1 (7 downto 0) B;
in      right1 (7 downto 0) B;
in      controlbits (4 downto 0) B;
out     bottom2 (7 downto 0) B;
out     right2 (7 downto 0) B;
out     top2 (7 downto 0) B;
out     left_top B;
out     left_right B;
out     left_bottom B;
out     right_top B;
out     right_bottom B;
out     bottom_top B;
begin
-- Pattern description :
# last 6 OUT bits are used to indicate if there is any connection
between 2 buses for example if left_top is 0, it means the switch
connects left bus ( BUS1) to top bus ( BUS2 )
#
# other OUT bit vectors are used to verify if the connection between
buses are made
#
# since this implementation is not using actual buses but bit vectors
there should
# be some way to verify if the connection has been established
# for example if input left1 appears at bottom2 along with a 0 for
left_bottom bit
# it means buses 1 and 4 are connected
#
p1: 01010101 11100111 10101010 00100 ?10101010 ?00000000 ?01010101 ?0
?1 ?1 ?1 ?0 ?1;
p2: 01010101 11100111 10101010 01110 ?00000000 ?01010101 ?00000000 ?1
?0 ?1 ?1 ?1 ?1;
p3: 01010101 11100111 10101010 10000 ?01010101 ?00000000 ?10101010 ?1
?1 ?0 ?0 ?1 ?1;
p4: 01010101 11100111 10101010 11110 ?00000000 ?00000000 ?00000000 ?1
?1 ?1 ?1 ?1 ?1;
p5: 01010101 11100111 10101010 11111 ?00000000 ?00000000 ?11100111 ?1
?1 ?1 ?1 ?1 ?0;
p6: 01010101 11100111 10101010 01111 ?00000000 ?01010101 ?11100111 ?1
?0 ?1 ?1 ?1 ?0;
p7: 01010101 11100111 10101010 00110 ?00000000 ?00000000 ?01010101 ?0
?1 ?1 ?1 ?1 ?1;
p8: 01010101 11100111 10101010 10110 ?01010101 ?00000000 ?00000000 ?1
?1 ?0 ?1 ?1 ?1;
p9: 01010101 11100111 10101010 11100 ?10101010 ?00000000 ?00000000 ?1
?1 ?1 ?1 ?0 ?1;
p0: 01010101 11100111 10101010 11000 ?00000000 ?00000000 ?10101010 ?1
?1 ?1 ?0 ?1 ?1;

end;

```

Summary of Results:

1. Meshes with buses are better than Meshes with only links
2. Meshes with Reconfigurable buses are still better than Meshes with buses.
3. Problem under consideration is best solved in Reconfigurable Meshes
4. Using the Simulator the algorithm mentioned can be verified
5. The hardware switch obeys the test patterns and hence can be used to build reconfigurable meshes

Future directions:

1. This paper ignored Queue size and other problems. This paper can be augmented by considering these
2. Switch proposed is not optimized with respect to instruction set and hence can be optimized. Also some algorithms require bi-directional buses that can be implemented.
3. Complex algorithms can be simulated in the Simulator
4. VHDL implementation can be done with buses instead of bit vectors.

REFERENCES:

[1] R.Miller, V.K.P.Kumar, D.Reisis, and Q.F.Stout, Parallel Computations on Reconfigurable Meshes, IEEE Transactions on Computers, 42 (1993), 678-692

[2] S.Rajasekaran, Mesh-connected computers with fixed and reconfigurable buses: Packet routing, sorting and selection. In Proc. 1st Annual European Symposium on algorithms, September 1993, 309-320

[3] S.Rajasekaran and T.McKendall, Permutation Routing and sorting on the reconfigurable mesh, Technical Report MS-CIS, Dept of Computer and Information Sciences, Univ. of Penn, May1992

[4] Y.Ben-Asher, D.Peleg, R.Ramaswami, A.Schuster, The power of reconfiguration, 18th International colloquium on Automata, Languages and Programming, July 1991, Madrid.

[5] G.F. Lev and N.Pippenger and L.G.Valinat, A Fast Parallel Algorithm for Routing in Permutation Networks, IEEE Trans. Comput. 30 (1981) 93-100

[6] J.Y.Leung and S.M.Shendu, On multidimensional packet routing for meshes with buses, J. Parallel and Distrib. Systems (1990)

[7] H.Li and Q.Stout, Reconfigurable massively parallel Computers, Prentice-Hall, 1991

[8] H. Stone, High performance computer Architecture

[9] B.Wang and G.Chen, Constant time algorithms for the transitive closure and some related graph problems on processor arrays with reconfigurable bus systems, IEEE Trans. Parallel. and Distrib. Systems. 1 (1990)

[10] J. Sibeyn, Solving Fundamental Problems on Sparse-Meshes, IEEE Transactions on Parallel & Distributed Systems. 11(12): 1324-1332, 2000 Dec

[11] T.Suel, Permutation routing and sorting on meshes with row and column buses, 8th International symposium on parallel computing

Reference websites:

[a] <http://bwrc.eecs.berkeley.edu/Classes/CS252/Notes/67>

[b] http://www.krellinst.org/UCES/archive/classes/ASC/tableofcontents2_1.html

[c] <http://www-unix.mcs.anl.gov/dbpp/text/node1.html>

[d] <http://www.mv.com/org/rmesh/Tutorial.html>

[e] www.rsise.anu.edu.au/annrep/1998/researchcs1.html

[f] www.iti.fh-flensburg.de/lang/papers/trans/transcl.htm

[g] www.cs.umass.edu/~weems/CmpSci635/635lecture14.html