On a Fast Interconnections

Ravi Rastogi and Nitin*
Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat, Solan-173234, Himachal Pradesh, INDIA

*College of Information Science and Technology, Peter Kiewit Institute, University of Nebraska at Omaha, 6001 Dodge Street, NE 68182-0116, USA

Summary
In this paper, we have discussed tool called Fast Interconnection, which have been designed for developing fault-tolerant multi-stage interconnection networks. The designed tool is one of its own kind and will help the user in developing 2 and 3-disjoint path networks.
Key words:
Interconnection Network, Fault-tolerance, 2 and 3-disjoint path Interconnection Network.

1. Introduction and Motivation

Multi-stage Interconnection Networks (MINs) are widely used for broadband switching technology and for multiprocessor systems. Besides this, they offer an enthusiastic way of implementing switches used in data communication networks. With the performance requirement of the switches exceeding several terabits/sec and teraflops/sec, it becomes imperative to make them dynamic and fault-tolerant. A number of techniques have been used to increase the reliability and fault-tolerance of the MINs, a survey of the fault-tolerance attributes of these networks is found in [1, 2].

The typical modern day application of the MINs includes fault-tolerant packet switches, designing multicast, broadcast router fabrics while system on-chip and networks on-chip are hottest now days. Normally the following aspects are always considered while deigning the fault-tolerant MINs: the topology chosen, the routing algorithm used, and the flow control mechanism adhered. The topology helps in selecting the characteristics of the present chip technology in order to get the higher bandwidth, throughput, processing power, processor utilization, and probability of acceptance from the MIN based applications, at an optimum hardware cost. Soon, as the topology is freeze, the analytical bounds, which helps for measuring reliability and availability can be examine [3-6]. The topology helps in determining the throughput and latency of the MINs whereas the routing algorithm and flow control encourage in achieving the performance bounds. Whenever we want to design an interconnection network, we used to design them manually using the windows word. At present, we do not have any tool through which we can develop the interconnection networks tool or this remains out of limelight therefore in this paper; we have discussed a tool designed for developing fault-tolerant multi-stage interconnection networks. The designed tool is one of its own kind and will help the user in developing 2 and 3-disjoint path networks.

The rest of the paper is as follows: Section 2 discusses the algorithm of the codes, developed to design the case tool together with the screen shots followed by the conclusion and references.

2. Algorithms

In this section, we have provided the algorithm for the various components, which are part of Fast Interconnection tool.

2. 1 Auto Align Method

Function of this Method: This method is used to align the components on the canvas and is called when the auto align button is clicked.

ALGORITHM: AUTO_ALIGN

```
Step1: Divide whole of the canvas into a grid.
Step2: Assign each component to the respective column such that column width is less than twice the width of the component after making a copy of all the current coordinates.
Step3: Set the x coordinate of the components equal to the average x value of that column.
Step4: Set the x coordinates of the components in the following columns by
```

Manuscript received August 5, 2010
Manuscript revised August 20, 2010

adding the horizontal distance specified by the user.
Step5: Get the y coordinate of the first component in first column and determine the total y height of the first column.
Step6: Get the y heights of all the columns.
Step7: Determine the height differences of all the columns.
Step8: To the aligning y factor of every column add the average height difference.
Step9: Assign the newly calculated (x,y) coordinates to the respective components.
Step10: Revert alignment if overlapping occurs.

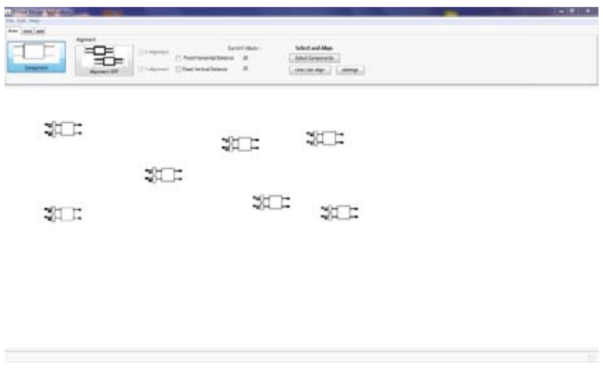

Figure 1. Shows the components to be aligned.

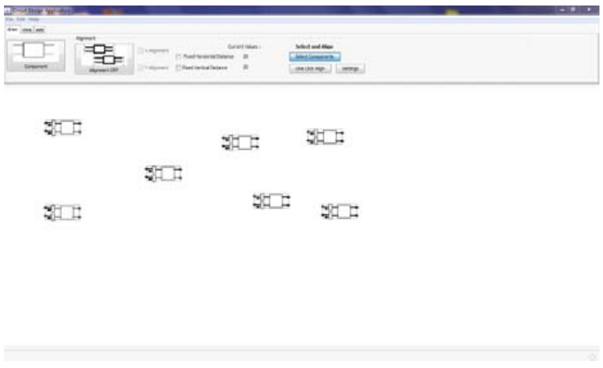

Figure 2. Shows that the elements to be aligned have been selected.

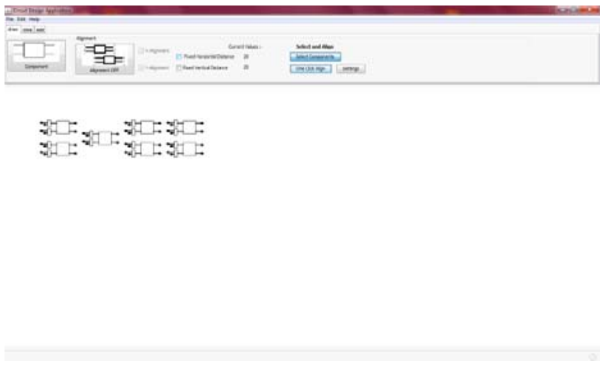

Figure 3. Shows that the elements have been aligned.

2.2 Draw Method

Function of this method: This method is used to draw the components on the canvas. It is called when the canvas is refreshed, a new component is added to the circuit, when any movement occurs, or when a wire connection is made.

ALGORITHM: DRAW

```
Declare four points;
Get the graphics object for the canvas;

Initialize a for loop from 0 to the total 'number of components' - 1
Begin FOR loop
Draw each component according to the 'type of the component' variable stored in the component class;
End loop
Initialize a for loop from 0 to the total 'number of lines' - 1
Begin FOR loop
IF line to be drawn is a normal line
get start point;
get end point;
draw line;
ELSE
retrieve the top and bottom points of the component
IF line is to be drawn from upper point THEN
the line clearance will be 1.5 times the width of the component;
ELSE
the line clearance is 2 times
End IF block
IF the start point is before the final point draw line 1.5 times of the width from right;
ELSE
draw line 1.5 times of the width from
```

```
left;
End IF block
End IF block
End FOR loop
```

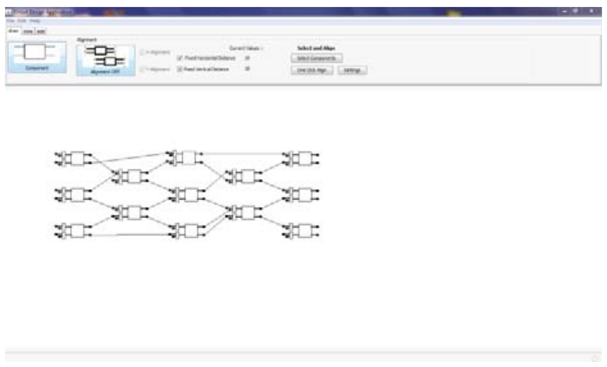

Figure 4. Shows that the components drawn using the draw method.

2.3 Intersection Check

Function of this method: This method detects overlapping of the components. This method is called when any component is moved across the canvas or a new component is added to the circuit or when the components are aligned.

ALGORITHM: INTERSECTION_CHECK

```
Get (X, Y) coordinates of the item just pasted;
Initialize 'component selected' variable to zero;
IF component and move option is enabled
    Initialize 'intersect' variable to zero;
    Initialize 'no of selected components' variable to zero;
    WHILE 'no of selected components' is less than 'number of components'
        get next 'selected component' in a variable;
        get height, width, centre coordinates in respective variables;
        IF current (X, Y) coordinates are in range of the 'selected component' coordinates set 'intersect';
        End IF block
        IF 'intersect' is set and move or component options are enabled then
            draw rectangle around current component
            Break loop;
        End IF block
        increment ''no of selected components' by one;
    End WHILE loop
    IF 'intersect' not equal to one and move and component options are enabled THEN
        create new 'component' and a 'temporary component reference'
        Initialize 'counter' to zero;
        WHILE ‘counter' is less than 'number of components'
            store 'component[counter]' in 'temporary component reference';
            get width, height and centre coordinates in respective variables;
            IF component boundary overlaps with another component boundaries then
                get x and y coordinates of the 'temporary component reference'
            End IF block
            increment 'counter';
        End WHILE loop
        Initialize 'intsct' to zero;
        Initialize '2nd selected component number' to zero;
        WHILE ‘2nd selected component number' less than 'no of components'
            IF current component in boundary of existing component
                set intsct;
            End IF block
            IF 'intsct' is set then
                draw red rectangle;
            End IF block
            Break loop;
            Increment ‘2nd selected component number';
        End loop
        IF 'intsct' is zero
            draw component;
            increment the 'number of components';
            add new component in the list of existing components;
        End IF block
End IF block
```

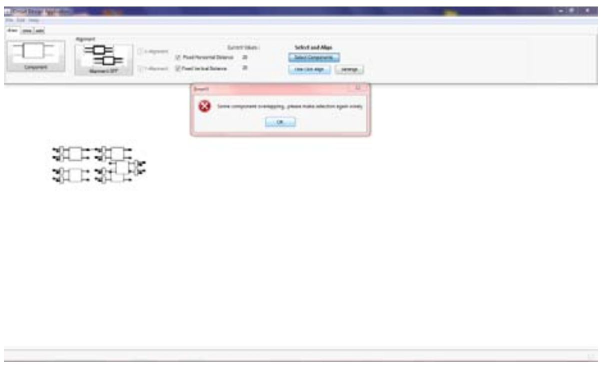

Figure 5. Shows the working of the intersection checking method at work.

2.4 Selecting Component

Function of this method: This method is used to select several or any one component from the circuit. This method is also called when the components have to be aligned selectively using the auto align functionality.

ALGORITHM: SELECTING_ONE_COMPONENT

```
IF select button selected
get (X, Y) coordinate of mouse pointer click location
Initialize variables for intersection check, component selected number, component increment counter
WHILE component number (temporary loop variable) is less than total number of components
get next component;
store the coordinates of the component;
IF the pointer coordinates are in the area of the component set an intersection flag;
IF the intersection flag is set then
draw a rectangle around the component;
Break the loop;
End IF block
ELSE
increment the component number so that when the loop runs the next time next component is selected;
End ELSE block
End WHILE loop
If intersection flag is set then in the selected components array store the selected component number;
refresh;
End IF block;
```

ALGORITHM: SELECTING_MULTIPLE_COMPONENT

```
IF the select component button is pressed and the CTRL key has been pressed down
Follow the exact same procedure to search for the one component being selected from several components drawn on the canvas;
Initialize a 'flag' variable to zero;
Begin
FOR loop initializing i to 0 running loop till all the 'selected components' are considered;
IF any of the 'selected components' value is equal to the most recently selected component
THEN
undraw the rectangle around the component;
decrement the' selected components' count;
Begin
FOR loop initializing j to i running loop till all the 'selected components' but one less components are considered;
shift every 'selected component' array element to a previous index;
//basically deleting the element from the array and maintaining the contiguous nature of the data
End FOR loop
set flag variable;
refresh the rectangles drawn;
Break loop;
End IF block;
End FOR loop;
IF flag is not equal to one then
increment the 'number of selected components' variable;
store the 'selected component number' in the 'selected component' array at the location referenced by
'number of selected components' variable;
End IF block
End Procedure
```

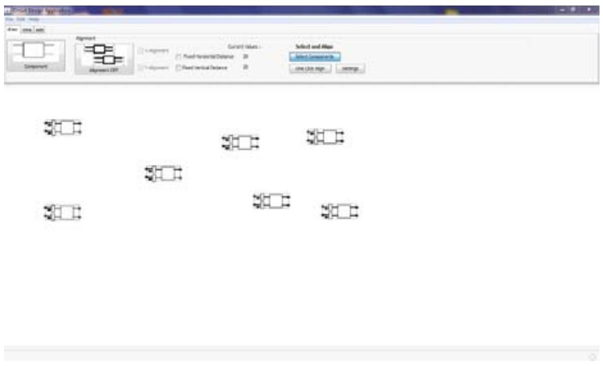

Figure 6. Shows the selection methods usage.

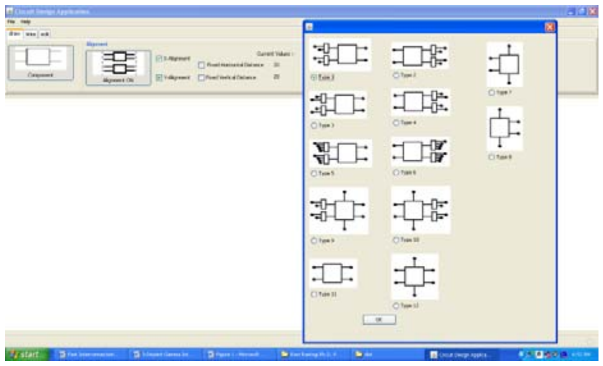

Figure 7. The front window case tool with the various components.

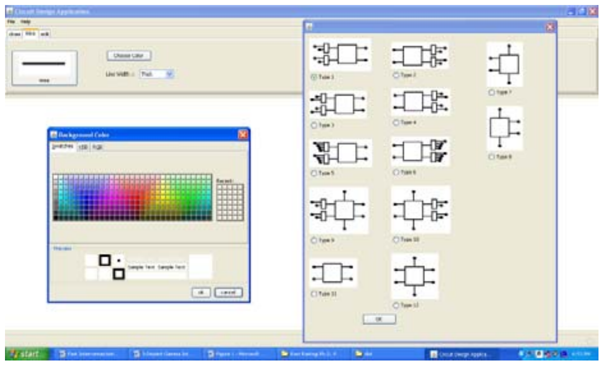

Figure 8. The front window with different widths of the wire and with different background color.

3. Conclusion

In this paper, we have discussed a case tool called as Fast Interconnections, which have been designed to develop the 2 and 3-disjoint path multi-stage interconnection network. We have provided the algorithm for the various functions, which we have created for the software.

References

[1] T.Y. Feng, A survey of interconnection networks, Computer, Vol. 14, 1981 December, pp. 12-27.

[2] G.B. Adams III, D.P. Agrawal, and H.J. Siegel, A survey and comparison of fault-tolerant multi-stage interconnection networks, IEEE Computer, Vol. 20, 1987 June, pp. 14-27.

[3] J.H. Patel, Performance of processor-memory interconnection for multiprocessors, IEEE Transactions on Computers, Vol. 30, 1981, pp. 771-780.

[4] J.P. Shen, Fault-tolerance analysis of several interconnection networks, Proceedings of International Conference on Parallel Processing, August, pp. 102-112.

[5] J. Duato, S. Yalamanchili, and L. Ni, Interconnection Networks: An Engineering Approach, IEEE Press, 1997.

[6] W.J. Dally and B. Towles, Principles and practices of interconnection networks, Morgan Kaufmann, 2004.