Volume : 2 | Issue : 1 | Jan 2013 • ISSN No 2277 - 8160

Research Paper | Engineering

Simulation of Cross-Talk Avoiding Algorithms in Optical Multistage Interconnection Network

Mr. Sandeep kumar — Sekhawati Engg. College, Dundlod, Dist. - Jhunjhunu (Raj.)

Mr. Noor Mohmmad — Sekhawati Engg. College, Dundlod, Dist. - Jhunjhunu (Raj)

ABSTRACT

Several types of interconnection networks have been proposed for parallel processing. One of them is OMIN's (Optical Multistage interconnection networks). Through optical networks we can achieve high performance and low latency also. With their great advantages over electronic networks, OMIN's also having their own demands and problems. One undesirable problem introduced by the Optical Multistage Interconnection network is a crosstalk that is caused by coupling two signals within a switching element. To avoid a crosstalk, many approaches have been proposed such as time domain and space domain approaches. Because the messages should be partitioned into several groups to send to the network, some methods are used to find conflicts between the messages. Window Method is used to find out which messages have conflict and should not be in the same group. In this paper, some methods to avoid crosstalk are compared and checked and Simulate some of them solution approaches.

KEYWORDS: Multistage Interconnection Networks (MIN), Optical networks, Crosstalk, Window methods.

I. INTRODUCTION

Multistage interconnection networks (MINs) consist of more than one stages of small interconnection elements called switching elements and links interconnecting them. A MIN normally connects N inputs to N outputs and is referred as an N × N MIN. The parameter N is called the size of the network.

Multistage interconnection network is actually a compromise between crossbar and shared bus networks of various types of multiprocessor interconnections networks.

II. OPTICAL MIN:

Optical interconnections have the potential of becoming an appealing alternative to electrical interconnections. For long and medium range distances (e.g., local area networks and telecommunication), optical technology (fibers) is the technology of choice, offering better performance and lower costs than electrical wires [2]. There is a trend for optics to replace electronics for shorter distances and larger connectivity applications.

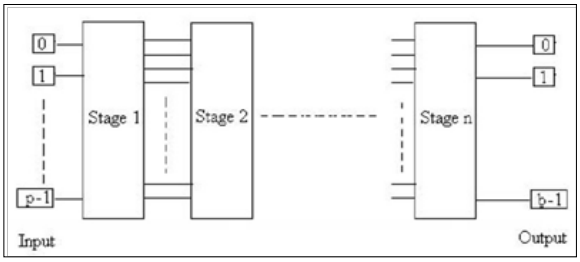


Figure 1: A Multistage Network

A. Path Dependent Loss

Path dependent loss means that optical signals become weak after passing through an optical path. In a large MIN, a big part of the path-dependent loss is directly proportional to the number of couplers that the optical path pass.

B. Optical Crosstalk

Optical crosstalk occurs when two signal channels interact with each other. To reduce the negative effect of crosstalk, various approaches which apply the concept of dilation in either the space or time domain have been proposed. With the space domain approach, extra SEs are used to ensure that at most one input and one output of every SE will be used at any given time. Window Methods are based on Time domain approach.

III. WINDOW METHOD

Window Method is a technique used to find which messages should not be in the same group because they introduce crosstalk in the network [2, 5]. WM can be described as follows:

For network size N*N, there are N source and N destination address. Each source and its corresponding destination address are combined to produce a combination matrix. From this matrix, the optical window size is M-1 where M= log 2N and N is the size of the network.

This window is used in the combination matrix from left to right except first and last column. If two messages have the same bit pattern, they will cause conflict in the network. Hence, they must be routed in different passes. To see how the WM works, refer to the following example. The network size is 8*8 and permutation is shown in Figure 2:

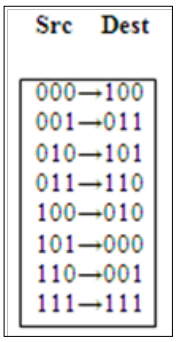
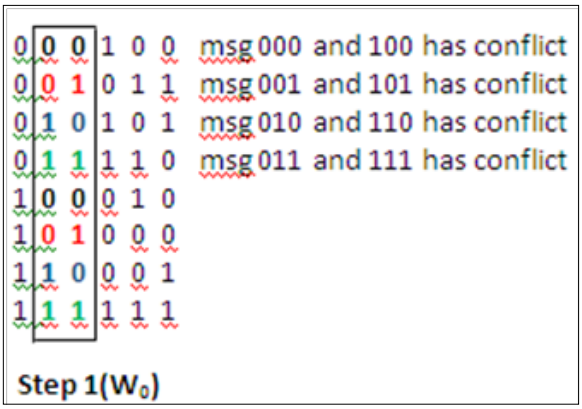


Figure-2: Permutation in binary format

Window method is demonstrated in Figure 3. The window size is M-1=2 (M= $\log_2 8$=3) and the number of windows is M=3 ($^{w}0$, $^{w}1$, $^{w}2$).



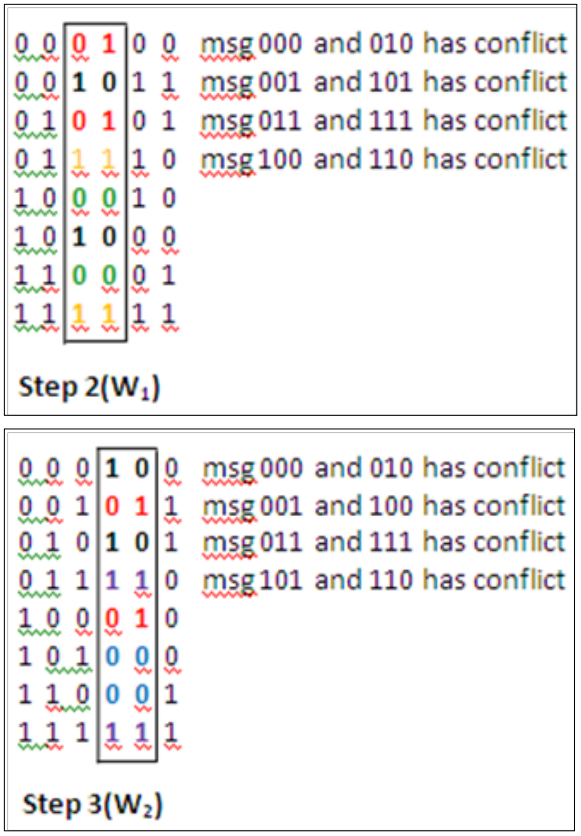


Figure 3: Optical Window Method

The pseudo code of WM is as shown in Figure 4.

```
Window _Method ()
For l= 1 to m+1          // m is the number of stages
For i=0 to N-1           // N is the network size
For k=1 to l+m-1 do
A [k-1] =Com_Matrix[i] [k]; End for;
For j=i+1 to N-1
For k=1 to l+m-1 do
B [k-1] =Com_Matrix[j] [k] End for;
If (A=B)
Conflict_Matrix[i][j]=1;          //conflict
End if; End for;
End for; End for;
```

Figure 4: The pseudo code of WM

Algorithm WM.

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size.
4. Make the window to see where the conflict in the data is.
5. Access the window to see where the conflict in the sent data is.
6. If any conflict occurs change that particular value in conflict matrix to 1.

IV. IMPROVED WINDOW METHOD

The number of windows in WM is equal to the number of stages (M= log 2N, M is the number of stages). In Improved WM (IWM), to find the conflicts among the messages in comparison with the first window is eliminated. To do this, the conflict matrix is initialized with 0. Then it is checked whether the node is one of the first half of the nodes or no (node<N/2, N is the size of the network), if it is, set the conflict matrix with finally execute WM for the other stages.

The pseudo code of IWM algorithm is as shown in Figure 5

```
Improved Window Method ()
Begin
Initialize conflict matrix with zero;
For (n=0 to N/2-1) do
Set conflict Matrix [n] [n+N/2] with 1
End for;
Window ()
End for;
End;
```

Figure 5: The Pseudo Code of Improved Window Method

Algorithm IWM

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size. Decrement window size 1 as there is one window less as was in window method.
4. Make the window to see where the conflict is there in data.
5. Access the window to see where the conflict is in sent data.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step no
8. Converts integer number into binary number to show the conflict.
9. Finally, display the source-destination and conflict matrix as output.

In IWM, the number of windows is M-1(M is a number of stages) after eliminating the first window. In the network size 8×8, the number of windows is only two because of eliminating the first window (w0). Messages 000,001,010 and 011 pass same switch with messages 100,101,110 and 111 respectively in the first stage. The execution time is reduced approximately by 1/M compared to the time of standard WM where M is the number of stages [3]. Figure 6 shows the conversion of WM to IWM by an example.

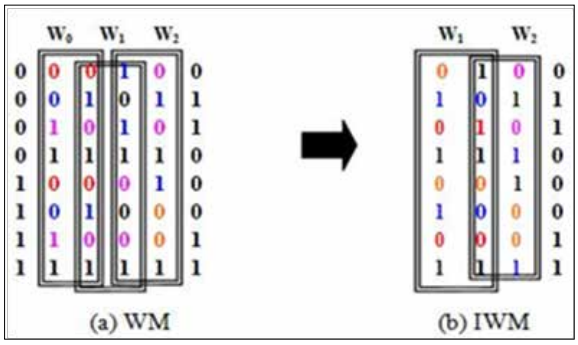


Figure 6: Conversion WM to IWM

V. REASONS FOR USING BITWISE WINDOW METHOD

WM has already proved to be efficient in many researches (Miao, 2000, Shen, 2001, Katangur, 2002 and Munir, 2005) but except for a few attempts, the authors have not found any improvement of WM [3]. Nevertheless, in this research a large amount of time is used to find conflicts among the messages. In Table 1, the execution time of WM and routing algorithm is demonstrated.

Net Size	WM	Routing & WM	Only Routing
8	0.031	0.032	0.001
16	0.063	0.078	0.015
32	0.204	0.219	0.015
64	1.000	1.031	0.031
128	4.656	4.797	0.141
256	24.187	25.329	1.142
512	108.906	110.750	1.844
1024	499.046	519.922	20.876

Table 1, Execution time of WM and routing algorithm

From Table 1, it is clear that a large amount of time is used in WM to find conflicts. For instance when the network size is 256, 24.656 milliseconds is used to find conflicts between the messages and only 1.141 milliseconds is used to route the messages. If the time of WM decreases, consequently the execution time will be decreased to route the messages. As a result, the focus of this paper is in reducing the execution time of WM by applying bitwise operations.

VI. BITWISE COMBINATION MATRIX

In WM with N*N size of the network, there are 2*n (n= log2N) columns to produce a combination matrix but in bitwise WM, there are only n columns. To produce this matrix, all binary bits of single rows in each window are converted to decimal. When all of these are converted to decimal, the number of columns is reduced to n. As a result, when comparing the messages to find a conflict, only n columns is compared instead of 2*n columns. By this method, time is reduced approximately by 10 times. This is a very effective method especially when the network size is large. For example the bitwise combination matrix for 8*8 network size is demonstrated in Figure 6. The number of columns in WM is 6 (Ci, i= 2*n) and for bitwise WM is 2 (Ci, i=n).

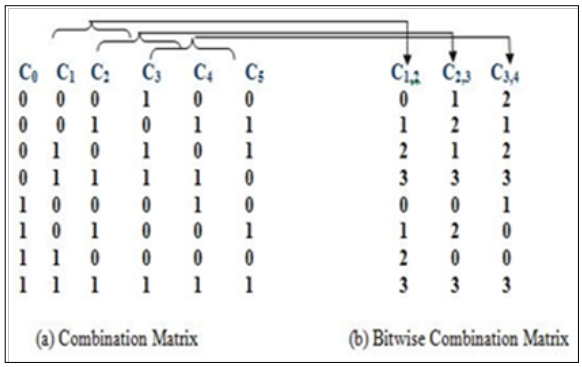


Figure 7: Bitwise combination matrix 8*8

The pseudo code of bitwise combination matrix is shown in Figure 8.

```
Com_ Matrix ()
Shright = n;                //n is the number of stages
Shleft = (2pow (n-1))-1 << n
For (i = 0 to n-1)
For (j = 0 to N-1)          // N is the network size
Temp = j*N+out(j);
Com_ Matrix (j) (i) = temp & shleft; Com_Matrix(j)(i)=Com_Matrix(j)
(i)>>shright;
 End for;
Shleft / = 2;
Shright --;
End for;
```

Figure 7: Pseudo Code of Bitwise Combination Matrix

In the new approach, the decimal numbers of the combination matrix are made by doing bitwise operations on source and destination addresses.

VII. BITWISE WINDOW METHOD

In WM based on bitwise operations, the source and destination address is decimal format. Thus, from combination matrix, the optical window size is only one for a different network size and the number of window is Log2N. In other words, there are only one decimal number in each row and each window for comparison and finding a conflict. Figure 8 shows the pseudo code of WM based on bitwise operations. From Figure 8, it can be realized that the proposed algorithm is very simpler and faster compared to WM.

```
Finding Conflict Matrix;
For (i=0 to n-1)            // n is the number of stages
For (j=0 to N-1) // N is the size of the network
For (k=j+1 to N-1)
If (Comb Matrix (j)(i) = Com Matrix (k)(i)) Set conflict Matrix;
End if; End for; End for;
End for;
```

Figure 9: Pseudo Code of bitwise WM

For example for bitwise permutation in Figure 6 (b) , in W0, the first message is zero, to find a conflict between zero and other messages in W0, zero is compared with all other messages. We can realize that zero has a conflict with four because of same message; message one has a conflict with five, two with six and message three with seven. For the second window (W1), it is clear that message zero has a conflict with message two, one with four and message three with seven and so on. Finally in the last

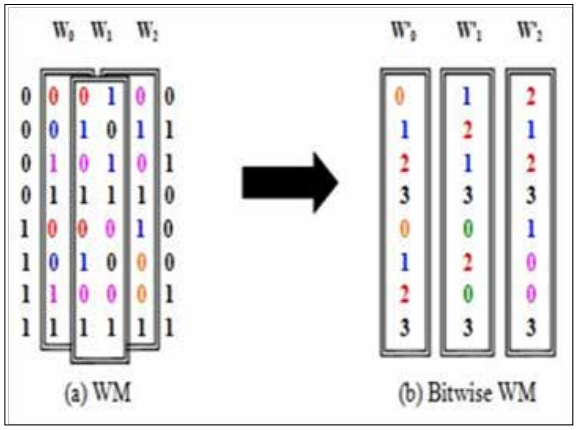


Figure 10: Window method based on Bitwise operations

Algorithm BWM.

1. Initialize the conflict matrix with value 0.If there is conflict between data sent then corresponding value is changed to 1 in conflict matrix.
2. Input or initialize the number of source-destination pairs from the user.
3. Calculate the no of windows and window size. Decrement window size 1 as there is one window less as was in window method.
4. Here the window size will be smallest i.e. window size is of one column size.
5. Access the window to see where the conflict in the sent data is.
6. If any conflict occurs change that particular value in conflict matrix to 1.
7. Go to step 4 if any more window is there else go to step no 8.
8. Finally, display the source-destination and conflict matrix as output.

VIII. COMPARATIVE ANALYSIS

This section discusses experiments done to evaluate the proposed methods. This is execution time of bitwise WM proposed to be compared with WM and IWM. In the proposed combination matrix to produce a conflict matrix, there are only n (n=Log 2N) columns instead of 2*n columns, which means that the columns is reduced to half. The window size also decreases to one. To compare the messages, there is only one bit instead of n-1 bits in the window. The comparison time of other bits is eliminated. Hence totally the execution time is decreased roughly 10 times. This reduction becomes more obvious when the network size is large. The execution time of WM, IWM and bitwise WM is shown in Table 2.

Net Size	WM	IWM	BWM
8	0.079	0.032	0.014
16	0.172	0.078	0.023
32	0.422	0.25	0.059
64	1.219	1.109	0.184
128	6.235	5.766	0.672
256	30.128	29.297	2.83
512	148.875	148.252	12.156
1024	676.73	515.025	53.205
Average	107.92	87.47	8.64

Table 2: Execution time for WM, IWM and Bitwise WM

In Table 2, the proposed bitwise WM is compared with WM and IWM, which is used to find the conflict matrix in OMIN. The average execution time for bitwise WM is 8.64 milliseconds while the average execution time for WM is 107.92 and IWM is 87.47 milliseconds. Considering the Table 2, IWM has significant influence when the network size is less

than 64. When the network size is bigger than 64, the results for IWM and WM converge

In Table 2, the proposed bitwise WM is compared with WM and IWM, which is used to find the conflict matrix in OMIN. The average execution time for bitwise WM is 8.64 milliseconds while the average execution time for WM is 107.92 and IWM is 87.47 milliseconds. Considering the Table 2, IWM has significant influence when the network size is less than 64. When the network size is bigger than 64, the results for IWM and WM converge.

Nevertheless there is a huge difference between execution time of bitwise WM and both IWM and WM. Realizing Table 2 and Figure 10, this difference becomes more significant when the network size gets bigger. The difference ranges from 5.5 times for network size 8 to 11 times faster for network size 1024.

In WM and IWM, the process to find conflicts between the messages is very time consuming because all messages should be compared bit by bit by 2*n columns. But in WM based on bitwise operations,
the comparison is done for n columns and in decimal format. As a result, WM based on bitwise operations is much more efficient and faster regarding the execution time.

IX. CONCLUSION AND FUTURE WORKS

In this research, we use WM based on bitwise operations to improve the performance of finding a conflict in OMIN. In comparing the performance of bitwise WM to find a conflict with previous WM, the results are consistent with our intuition. The bitwise WM can improve the time nearly more than 10 times special when the network size is large. Efficient message routing algorithms directly affect the performance of communication networks.

REFERENCES

A. Verma and C.S. Raghvendra, "Interconnection Networks for Multiprocessors and Mul-ticomputers: Theory and Practice", IEEE Computer Society Press, Los Alamitos, California, 1994. | | [2] Shen, X., Yang, F., and Pan, Y. (2001). "Equivalent permutation capabilities between time division optical omega networks and non-optical extra-stage omega networks". IEEE/ACM Trans. Netw. Vol. 9, No. 4, pp. 518 -524. | [3] Munir, A, Mohamed, O., and Rozita,J. (2005). "An efficient approach to avoid crosstalk in optical Omega Network". | [4] Katangur, A. K. Pan, Y., and Fraser, M.D. (2002). "Message Routing and Scheduling in Optical Multistage Networks Using Simulated Annealing". InternationalProceeding of the Parallel and Distributed Processing Symposium (IPDPS). | [5] Pan, Y. Qiao, C. and Yang, Y. (1999). "Optical Multistage Interconnection Networks: New ch alle nge s and Approaches", IEEE Communications Magazine, Feature Topic on Optical Networks, Communication Systems and Devices, Vol. 37, No. 2, pp. 50-56. | [6] Yang Y., Wang, J. and Pan, Y. (2000). "Permutation Capability of Optical Multistage Interconnection Networks", Journal of Parallel and Distributed Computing (JPDC), Vol.60, No. 1, pp.72-91. | [7] Pan, Y. Ji, C. Lin, X. Jia, X. (2002). "Evolutionary Approach for Message Scheduling in Optical Omega Networks", Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP). | [8] Mohammad, A.A. Mohamed, O., Rozita, J. and Shamala, S. (2005). "New Algorithm to avoid Crosstalk in Optical Multistage Interconnection Networks", Proceeding of IEEE International Conference on Network (MICC-ICON 2005) Vol. 1,pp.501-504. | [9] Siu-Cheung,C. and Tiehong,X. (2004). "NewAlgorithm for Message Routing and Scheduling in Optical Multistage Interconnection Network", Optical Communications Systems and Networks, Proceeding International. | [10] Qiao, C. (1994). "A time domain approach for avoiding crosstalk in optical blocking multistage interconnection networks", Journal of Lightwave Tech, vol 12, No 10, pp