БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ • BULGARIAN ACADEMY OF SCIENCES

ПРОБЛЕМИ НА ТЕХНИЧЕСКАТА КИБЕРНЕТИКА И РОБОТИКАТА, **69**
PROBLEMS OF ENGINEERING CYBERNETICS AND ROBOTICS, **69**

София • 2018 • Sofia

Hardware acceleration of a scheduling algorithm for crossbar switch node via decomposition of the connection matrix.

Kiril Kolchakov, Vladimir Monov

Institute of Information and Communication Technologies, 1113 Sofia, Bulgaria
Email: kkolchakov@iit.bas.bg

Abstract: *In this paper we propose an approach for improving the execution time of an algorithm for cross bar switch node with large dimension of the connection matrix. The algorithm is executed by diagonal connectivity matrix activation and it is optimal with respect to speed and common performance in the cases of small and medium dimensions of the connection matrix. In the case of large dimensions of the connection matrix, we apply a decomposition approach in order to accelerate the synthesis of a non-conflict schedule*

Key words*: network nodes, node traffic, crossbar switch, conflict elimination, packet messages.*

1. Introduction

The objective of this study is to improve the execution time of a scheduling algorithm for crossbar switch node with diagonal activation of the connection matrix. The algorithm is intended for CPU that controls the requests execution serving the traffic in the crossbar node. The crossbar node works in real-time, and the speed, total performance and memory required are of utmost importance.

One of the main problems in the operation of switching nodes is the presence of conflicts in the incoming requests which dramatically reduces the traffic. To eliminate conflicts, algorithms for scheduling the requests processing through a conflict-free schedule are used.

The problem with the conflicts associated with the operation of the switching nodes is as follows: the switches in the switching nodes are of the N x N size. N number of packet message sources are connected through the switch of the switch node with N number of receivers of these messages.

Conflicts occur in two cases:

- When a message source requests a connection to two or more message receivers.
- When a message receiver has a connection request from two or more message sources.

The switch state of a Crossbar node is represented by the so-called connection matrix. For a switch of N x N size , the connection matrix T is also N x N, with each member Tij = 1 if there is a request for a connection between the source of the message i and the receiver j. Otherwise Tij = 0.

A conflicting situation occurs when, in any raw of the connection matrix, the number of units is greater than one, this corresponds to the case when one source declare a connection to more than one receiver. The presence of more than one unit in any column of the T matrix is also an indication of a conflict situation and means that more than one source has declared a connection to one and the same receiver [1,3,4].

In this paper we consider an algorithm for synthesis of non-conflict schedule by diagonal connectivity matrix activation. The algorithm has been optimized with respect to speed and common performance in the cases of small and medium dimensions of the connection matrix. Here, our aim is to accelerate the synthesis of a non-conflict schedule in the case of large dimensions of this matrix. We compare optimized and existing algorithms in terms of speed, complex performance and memory required. For this purpose, their software models have been used for various characteristic types and sizes of the T-connection matrix. There are studies on modeling the traffic in crossbar switch nodes via the apparatus of generalized nets [6]. Here, we use software models written in Matlab language.

2. Description of the algorithm

The algorithm with diagonal connectivity matrix activation (ADA) uses the knowledge that requests placed diagonally are non-conflict [3]. Requests placed in any diagonal parallel to the main one are non conflicting with each other. In T connection matrix of N x N size, the number of diagonals with requests is 2N-1.

ADA activates the main diagonal of service requests and then sequentially activates the requests diagonals above and below it. For each iteration, only one requests diagonal is triggered, the first iteration being for the main diagonal. The iteration counter is organized to check that the number of iterations has not become 2N - 1, i.e. whether all requests diagonals have been enabled.

The downside of ADA is that it does not monitor the exhaustion of the requests. Whether at any stage of the algorithm's work, all requests are executed or not, 2N-1 of query diagonals are activated sequentially. Furthermore, the algorithm does not check for a zero-connection T matrix, an event that is very rare, but it is possible to happen. In this case, 2N - 1 number of zero diagonals will start to be activated sequentially. Arrived requests in previously activated null diagonals can not be processed until the last of the zero input matrix diagonals is not activated.

3. Optimized ADA

Considering the above mentioned drawbacks, an optimization of the algorithm ADA is implemented including the following steps.

1. Verify for a zero T connection matrix. At T = 0, the algorithm stops working[1].

2. Check for requests exhaustion for each iteration i.e. after activating a current requests diagonal, the number of requests in the queries are subtracted from the current amount of requests prior to its activation. It is checked that the current amount of requests are not reset, and if so, the algorithm stops working before all 2N - 1s of diagonals to be activated [1].

The examination of the optimized algorithm with diagonal connection matrix activation (ADAO) has been done through its SMADAO software model. Tests are performed with respect to speed, memory and complex performance of the algorithm. Typical types of connection matrices are used at different N values.

The comparison between the optimized version ADAO and the original algorithm ADA is also made through their SMADAO and SMADA software models. The software models are written in the Matlab programming language and tested on the Dell OPTIPLEX 745 computer system (Core 2 Duo E6400 2.13GHz, RAM 2048).

Figure 1 shows five types of characteristic connection matrices.

$$\begin{matrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{matrix} \qquad \begin{matrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{matrix}$$

2A 2B 2C

$$\begin{matrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \end{matrix} \qquad \begin{matrix} 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{matrix}$$

2D 2E

Fig. 1. Types of connection matrices.

The performance P of the non-conflict scheduling algorithms is calculated using formulas 1 and 2. R (v) denotes the number of null solutions (solutions when service requests are not realized), R (W) is the number of non-zero solutions, and R is the total number of solutions. P is the relation of non-zero solutions to the total number of solutions. When the number of zero solutions tends to zero, performance P approaches to 100% [1].

R=R(v) + R(w) **(1)**

P=(R(w)/R).100[%] **(2)**

Complex performance (CP) includes the speed and it is calculated by formula 3 [1].

CP = P.t , при **N = const. , t = 1/S** **(3)**

In Formula 3, S indicates the time in seconds to run the non-conflict schedule by using the respective software model. P is the performance. When the value of S is small, i.e the algorithm is faster, it is seen that t is larger and CP has a higher value.

4. ADAO examination and comparison with ADA at null connection matrix

The software models of the ADAO and ADA algorithms for zero input matrices for different N values are used in the study. Table 1 and Table 2 show the results.

Table 1. Speed and memory required

	SMADAO	SMADAO	SMADA	SMADA
N	S[Sec.]	M[KB]	S[Sec.]	M[KB]
50	0,27	20,62	0,71	101,03
100	0,31	81,22	2,02	402,03
150	0,48	181,82	3,9	903,03
200	0,74	322,43	7,24	1604,03
250	0,96	503,03	14,11	2505,03
300	2,34	723,63	18,88	3606,03
350	3,03	984,23	22,30	4907,03

Table 2 Complex Performance

	SMADAO	SMADAO	SMADA	SMADA
N	P[%]	CP	P[%]	CP
50	100	370,37	0	0
100	100	322,58	0	0
150	100	208,33	0	0
200	100	135,13	0	0
250	100	104,66	0	0
300	100	42,73	0	0
350	100	33,00	0	0

From Table 1 it is seen that SMADAO is 2.6 times faster than SMADA at N = 50, up to 14.6 times at N = 250.
The required memory on SMADAO is on average 5 times smaller than SMADA.

Table 2 presents the results of the study in terms of performance P and complex CP performance, calculated according to Formulas 2 and 3, respectively. For SMADAO, P is 100% for all sizes of N, while SMADA is 0%. CP for SMADAO decreases with N increase of 370.37 at N = 50, to 33 at N = 350, and CP at SMADA is zero for all N values[1].

5. ADAO and ADA comparison in typical connection matrices

Considering the typical types of connection matrices represented in Figure 1, it can be concluded that the type 2D corresponds to the most common traffic because the requests are randomly allocated. Larger connection matrix sizes have been selected for greater reliability in determining the trend of the results.

Tables 3 through 7 present the results of the study in terms of speed and memory required for the different types of connection matrices [1].

Table 3. Speed and Memory required in connection matrix 2A

	SMADAO 2A		SMADA 2A	
N	S[Sec.]	M[MB]	S[Sec.]	M[MB]
100	0,23	0,240832	1,36	0,402456
500	8,41	6,004032	42,92	10,012056
1000	34,51	24,008032	105,63	40,024056
1500	80,10	54,012032	183,46	90,036056
2000	95,65	96,016032	199,15	160,040056

Table 4. Speed and memory required in the connection matrix 2B

	SMADAO 2B		SMADA 2B	
N	S[Sec.]	M[MB]	S[Sec.]	M[MB]
100	1,74	0,409608	1,79	0,409632
500	33,52	10,048008	33,11	10,048032
1000	103,93	40,096008	101,05	40,096032
1500	253,57	90,144008	255,74	90,144032
2000	358,99	160,192008	269,60	160,184032

Table 5. Speed and Memory required in connection matrix 2C

	SMADAO 2C		SMADA 2C	
N	S[Sec.]	M[MB]	S[Sec.]	M[MB]
100	2,14	0,199608	3,83	0,199632
500	30,18	4,798008	32,71	4,798032
1000	102,74	19,096008	109,54	19,096032
1500	258,31	42,894008	253,99	42,894032
2000	403,88	76,192008	263,14	76,184032

Table 6. Speed and memory required in the connection matrix 2D

	SMADAO 2D		SMADA 2D	
N	S[Sec.]	M[MB]	S[Sec.]	M[MB]
100	1,65	0,402432	1,78	0,402456
500	30	10,012032	34,10	10,012056
1000	97,89	40,024032	97,96	40,024056
1500	236,86	90,036032	265,88	90,036056
2000	369,90	160,048032	284,18	160,040056

Table 7. Speed and memory required in a connection matrix 2E

	SMADAO 2E		SMADA 2E	
N	S[Sec.]	M[MB]	S[Sec.]	M[MB]
100	1,69	0,482432	1,81	0,482456
500	34,52	12,012032	33,78	12,012056
1000	112,13	48,024032	112,26	48,024056
1500	253,85	108,036032	264,14	108,036056
2000	431,03	192,048032	302,66	192,040056

Table 8. Complex Performance

Type Matrix	SMADAO CP	SMADA CP
2A	11, 89	0,15
2B	0,59	0,60
2C	2,65	2,44
2D	2,66	2,34
2E	1,54	1,57

From Tables 3 to 7 it is seen that the required memory is almost the same for SMADAO and SMADA in connection matrices 2B, 2C, 2D and 2E. For matrix type 2A, the memory required for SMADAO is about 1.66 times less than SMADA for all N.

Table 8 presents the results obtained for the complex performance of SMADAO and SMADA with different input connection matrices for N = 500 [1].

We have also studied the speed of execution of algorithms ADA and ADAO for large sizes of the connection matrix. The results obtained by means of the respective software models SMADA and SMADAO are shown in Table 9. From these results, it can be seen that in the cases of large sizes of the connection matrix, SMADAO speed compared to the speed of SMADA, generally decreases. In particular, for N = 2000, with exception of 2A, for all other types of characteristic input matrices, the speed of SMADAO is less than that of SMADA.

Table 9. Speed for N=2000 for typical input connection matrices

Matrix type	SMADAO N = 2000	SMADA N = 2000
2A	95,65	199,15
2B	358,99	269,60
2C	403,88	263,14
2D	369,90	284,18
2E	431,03	302,66

In view of the above results, our aim is to improve the execution time of ADAO at large N-sizes. For this purpose, we apply a decomposition of the connection matrix, so that the task being solved is reduced to solving two tasks with smaller dimensions.

6. Acceleration of the optimized algorithm with large dimensions of the connection matrix

Figure 2 shows the decomposed connection matrix. Sub-matrices α, α **+,** β and β**+** are the same and of **N/2 x N/2** dimensions. Knowledge is used that diagonally located service requests are non- conflict. Sub-matrices α and β are non- conflict to each other and α + and β+ are also non-conflict because they are diagonally located (Figure 2). The time to obtain a nonconflict schedule in the decomposed matrix is denoted by **Q** and it is given by

Q = 2.q, **(4)**

where q is the time to get a non-conflict schedule in a sub-matrix .

It is important to note that for each pair of sub-matrices a non-conflict schedule is made at the same time.

From Table 6 we can calculate Q for connection matrix 2D and N = 2000. Thus after decomposition with four sub-matrices with sizes N = 1000, formula 4 gives:

Q = 2q = 2. 97.89 = 195.78 sec.

In case the connection matrix is not decomposed, the time for the non-conflicting schedule is S = 369.90sec. This result indicates a considerable improvement of the execution time of SMADAO with a relatively large sizes of the connection matrix.

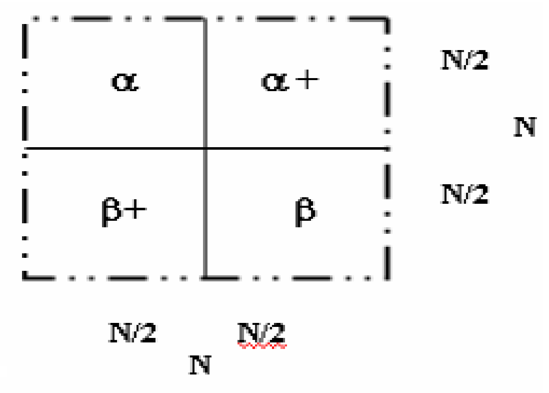

Fig. 2. Decomposed connection matrix

A hardware implementation of the proposed decomposition is shown in Fig. 3. To select sub-matrices in pairs, multiplexing of the buses leading to the packet receivers is used. Two-input multiplexers and only two N/2xN/2 commutators α^* **and** β^* are used. For **X = 0, Y = 1,** $\alpha^* = \alpha$, $\beta^* = \beta$, while for **X = 1, Y = 0,** $\alpha^* = \alpha^+$, $\beta^* = \beta^+$.

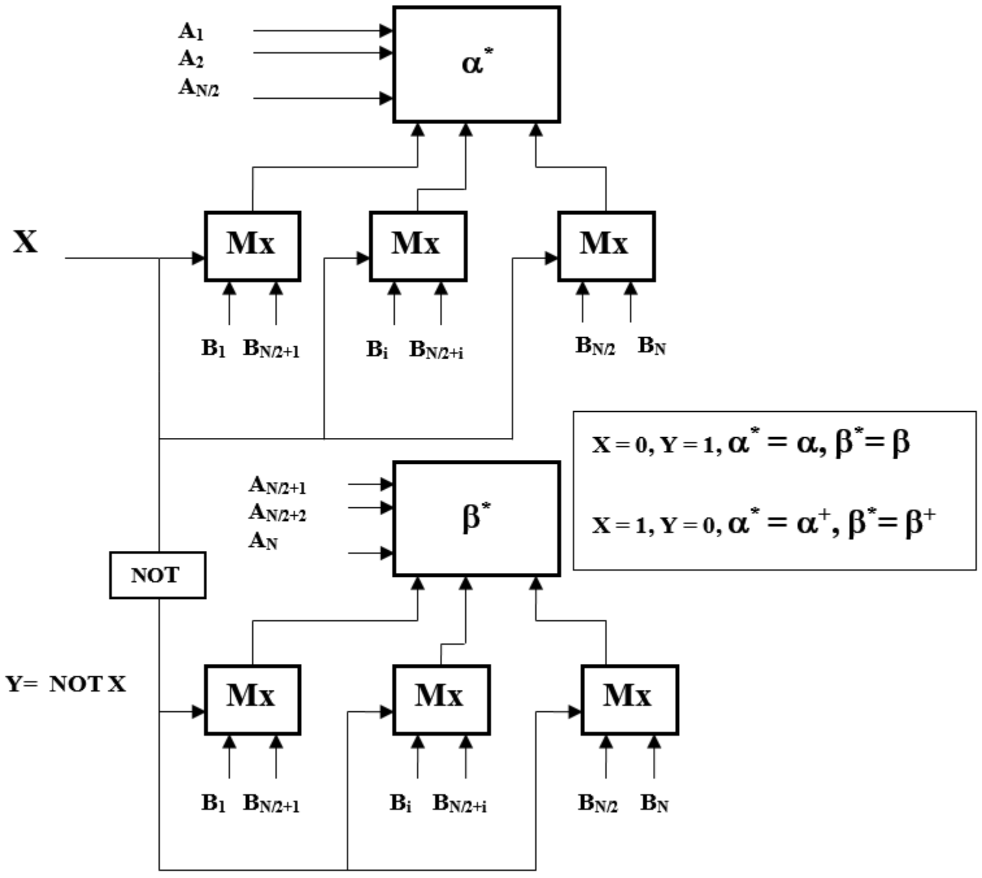

Fig. 3. Block scheme of the proposed decomposition

Conclusion

The research shows that with zero input matrix, the ADAO optimized algorithm has advantages over ADA in terms of speed, memory required, and complex performance for N from 50 to 350. With the input matrix of connections of type 2A, ADAO is 5.9 times faster for N = 100 to 2 times faster for N = 2000. The complex performance CP of ADAO for 2A is 11.89 vs. 0.15 for ADA.

For input connection matrix of type 2D (closest to the usual traffic), ADAO is faster than ADA for N = 100 to N = 1500 including. The trend is that for large N sizes for 2B, 2C, 2D and 2E, the algorithm's performance is aligned, whereas for N = 2000 ADAO is slower than ADA for all types of input matrices except 2A. However, by using the approach of connection matrix decomposition for sizes N= 2000, we have obtained a substantial acceleration of the optimized ADAO algorithm. Thus, for N = 2000 and decomposition with two switches of dimensions N = 1000, we obtained 1. 89 times faster conflict-free schedule.

References

1. Kolchakov K., V. Monov. An approach for algorithm optimization of non-conflict Schedule by diagonal connectivity matrix activation. Proceedings of the International Conference AUTOMATICS AND INFORMATICS`2017 John Atanasoff Society of Automatics and Informatics, Bulgaria, Sofia 04.10-06.10.2017., pp. 161 – 164, Proceedings ISSN 1313-1850, CD ISSN 1313-1869.
2. Kolchakov K., V. Monov. Comparative Analysis of a Class of Algorithms for Traffic Management in a Crossbar Commutator with Respect to Complex Performance, Speed and Memory, Problems of Engineering Cybernetics and Robotics, Vol.66, pp.53- 62, Sofia, 2015, ISSN 0204-9848. Bulgarian Academy of Sciences.
3. Kolchakov K., “An Algorithm Synthesis of Non-Conflict Schedule by Diagonal Connectivity Matrix Activation” Proceedings of the International Conference AUTOMATICS AND INFORMATICS`11, John Atanasoff Society of Automatics and Informatics, Bulgaria, Sofia 03.10-07.10.2011., pp. B-247 – B251, Proceedings ISSN 1313-1850, CD ISSN 1313-1869.
4. Tashev, T. , Atanasova, T. Computer Simulation of MIMA Algorithm for Input Buffered Crossbar Switch. International Journal "Information Technologies & Knowledge", Volume 5, Number 2, 2011. ITHEA® , Sofia, Bulgaria. Pages 183-189.
5. Wanjari P., A. Chonhari, Implementation of 4x4 crossbar switch for Network Processor, International Journal of Emerging Technology and Advanced Engineering, Website: www.ijetae.com (ISSN 2250 – 2459, Volume 1, Issue 2, December 2011).
6. Tashev T. Computering simulation of schedule algorithm for high performance packet switch node modelled by the apparatus of generalized nets. Proceedings of the 11th International Conference CompSysTech’2010, 17-18 June 2010, Sofia, Bulgaria. ACM ICPS, Vol.471, pp.240-245.

Аппаратное ускорение алгоритма расписания для кросбар переключателя через декомпозиции матрицы связи

Кирил Колчаков, Владимир Монов

Резюме: *Алгоритм выполняется путем диагональной активации входной матрицы и является оптимальным по скорости и общей производительности в случае малых и средних размеров матрицы. В случае больших размерностей входной матрицы применяем подход аппаратной декомпозиции, чтобы ускорить синтез бесконфликтного расписания.*