OLITS: An Ohm's Law-like Traffic Splitting Model Based on Congestion Prediction

Gaoming Du*, Yanghao Ou*, Xiangyang Li *, Ping Song *, Zhonghai Lu†and Minglun Gao*
*Institute of VLSI design, Hefei University of Technology, Hefei, China, 230009. Email: dugaoming@hfut.edu.cn
† School of ICT, KTH Royal Institute of Technology, Stockholm, Sweden. Email: zhonghai@kth.se

***Abstract*—Through traffic splitting, multi-path routing in Network-on-Chip (NoC) outperforms single-path routing in terms of load balance and resource utilization. However, uncontrolled traffic splitting may aggravate network congestion and worsen the communication delay. We propose an Ohm's Law-like traffic splitting model aiming for application-specific NoC. We first characterize the flow congestion by redefining a contention matrix, which contains flow parameters such as average flow rate and burstiness. We then define flow resistance as the flow congestion factor extracted from the contention matrix, and use the parallel resistance theory to predicate the congestion state for every target sub-flow. Finally, the traffic splitting proportions of the parallel sub-flows are assigned according to the equivalent flow resistance. Experiments are taken both on 2D and 3D multi-path routing NoCs. The results show that the worst-case delay bound of target flow is significantly improved, and network congestion can be effectively balanced.**

I. Introduction

Quality of Service (QoS) has become a hot topic since the birth of Network on Chip (NoC), where critical real-time applications require not only the average case performance but also the worst-case metrics. As NoCs are facing heavier workload, which leads to much more on-chip congestions, it becomes more important to address both worst case performance guaranteeing and congestion balancing.

For traditional simulation based approaches, adaptive routing is used to alleviate on-chip congestion. Two kinds of congestion are generally considered: channel congestion [1][2][3][4] and switch congestion [5]. As an example shown in Fig. 1(a), a typical latency model of NoC router is presented. Most of the previous works use only channel congestion to detect path congestion, but they are easy to fall into indetermination of congestion [6] when the candidate output channels are in the same congestion state.

Chang *et al.* [5] use both channel and switch congestion to identify path congestion, and achieves higher saturation throughput. However, they are easy to fall into the local optimal problem, as illustrated in Fig. 1(b). Assume that the target flow f(0,3) intends to send packets from source router R0 to the destination R3, and six contention flows (f1 to f6) are competing for the potential paths. At router R0, two candidate paths are considered, i.e., east channel (from R0 to R1) and south channel (from R0 to R2). When using the routing algorithm proposed in [5] to find different flow paths, the south channel will be selected, since it has less congestion than the other. However, when looking into this problem in a broader view, it may not be the global optimal strategy,

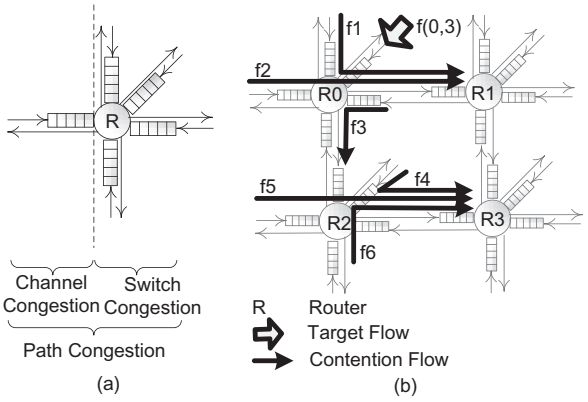

Fig. 1. A motivation example revealing the lacking of global information in current practice of congestion prediction, where (a) a latency model with channel congestion, switch congestion and path congestion, and (b) prediction only with local congestion information

because more congestions may exist in path R0→R2→R3 (four contention flows altogether) than in path R0→R1→R3 (only two). This is due to the intrinsic limitation of these kinds of routing algorithms, which can only make routing decision based on the local router or the nearest neighbours.

Besides the disability of global congestion prediction, the practice above only handles average delays in NoC, but rarely on worst-case. Recently, approaches based on network calculus [7][8] have been proposed to derive delay bound for application-specific NoCs [9]. Results have been obtained on wormhole routing NoCs [10] and credit based flow control router [11]. But congestion prevention problems are rarely discussed. This research gap motivates us to analyze the worst-case delay bound and globally balance the on-chip traffic workload to reduce delay both in average and maximum case.

Du *et al.* have done similar work [12]. They propose a contention matrix to characterize the flow congestion in a global view. However, their limitations are: the contention matrix uses only the traffic splitting proportion to represent the traffic contention, which may result in inaccurate traffic congestion; how to utilize the contention matrix to predict congestion has not been well addressed, which may degrade the importance of the contention matrix based on congestion representation.

In this paper, we adopt the basic concept of contention matrix [12] and propose an *O*hm's *L*aw-l*i*ke *t*raffic *s*plitting model (OLITS) by redefining the elements of contention matrix. The main contributions are as follows.

978-3-9815370-6-2/DATE16/

- *Flow congestion characterization and prediction through a redefined contention matrix.* We characterize the flow congestion by redefining a contention matrix, which contains parameters of average flow rate and burstness. We define *flow resistance* as the flow congestion factor extracted from the contention matrix, and use the parallel resistance theory to predicate congestion state.
- *OLITS.* The proposed flow resistance reflects the global congestion map of an application specific NoC. Inspired by Ohm's law with parallel resistance, we assign the traffic splitting proportions of the parallel sub-flows according to the predicted congestion information. In this way, the traffic can be well-balanced and the worst-case delay can be effectively improved.

II. Related Work

Initially proposed by Cruz [7], network calculus [8] was successfully applied from traditional application, e.g., Internet, Intranet and real-time embedded systems to on-chip communication [9][13][14].

However, there is a long way to put the theoretic results into practical usage. Because the congestion in real application-specific NoC is hard to handle. Congestion-aware adaptive routing methods were proposed, e.g., regional congestion awareness (RCA) [1], proximity congestion awareness (PCA) [2], look-ahead congestion detection [3] and neighbors-on-path (NoP) selection [4], but they were lack of accurate path-congestion detection. Chang *et al.* [5] combined both the channel congestion and switch congestion for path congestion prediction, but it was easy to fall into local optimal results. Even more, it was lack of worst-case analysis. So systematic approaches are needed to improve both global path-congestion and worst-case performance.

Du *et al.* had a preliminary try [12]. They utilized network calculus in worst-case delay analysis, and proposed contention matrix to capture network congestion for application-specific NoC. However, the contention matrix was not accurate enough for real-life traffic congestion prediction. In this paper, we not only handle the traffic congestion prediction problem, but also optimize the contention matrix to fix problems mentioned above all .

Inspired by Ohm's law, we propose OLITS to redistribute traffic loads. Tornero *et al.* also proposed an Ohm's Law-like method in topological mapping [15]. Our approach is different in three aspects. Firstly, they use Ohm's law to compute the equivalent distance between any two nodes in NoC. We define the equivalent resistance as the congestion factors, i.e., our approach is not only related to the distance, but also to the flow parameters. Secondly, we use this concept for traffic redistribution. Finally, our approach can improve the worst-case end-to-end delay for NoC packet delivery.

III. Network Calculus Basic

A. Basic results of network calculus

Network calculus can be used for worst-case performance derivation for NoC, and the arrival curve and service curve are two important concepts. The former is usually defined as a linear arrival curve $\alpha(t) = rt + b$, where r is the average (sustainable) rate of generating packets, and b is the burstiness that means the maximum number of packets sent to network one time [7]. The service curve is a latency-rate function $\beta(t) = R(t-T)^+$, where R is the minimum service rate, and T is the maximum output delay of the network element. Here x^+ equals 0, when $x \leq 0$, else x. Then the upper delay bound $\bar{D}$ [16] can be expressed by

$$\bar{D} = \frac{b}{R} + T. \tag{1}$$

B. Per-flow delay bound analysis

For a flow going through series of routers (i.e., servers), the concept of *equivalent service curve* (ESC) can be used to calculate the delay bound by Equation 1. In ESC derivation, the main challenge is how to handle the contention problem. Three contention types were classified in [9], i.e., nested, parallel and cross contentions. Algorithms have been developed to automate the analysis for both single-path [9] and multi-path routing [12]. All the work above laid a solid basis of the per-flow end-to-end delay bound analysis for 3D multi-path routing studied in this paper.

The 3D NoC router architecture has the following assumption. The routing protocol is wormhole routing, and the router arbitration strategy is first in first out (FIFO).

IV. Congestion Prediction

When flows are split into sub-flows, two major problems are faced: (i) how to choose the sub-flows; and (ii) how to assign the flow proportion to improve the worst-case transfer performance. Du *et al.*[17][12] have answered the first question. In this paper, we focus on the latter.

A. Dimension based Contention Matrix

Contention matrix was proposed to capture network contention [12]. Although it can balance the network congestion and decrease the worst-case delay, it has three limitations: lack of direction information of flows, too much memory consumption in run-time or inaccuracy.

For example, a 2D NoC with size of $n \times n$ has space complexity of $O(n^4)$. But for 3D NoCs, the space complexity becomes $O(n^6)$, which is unacceptable.

In order to capture the contention more accurately and decrease the space complexity, we propose a dimension based contention matrix. As shown in Fig. 2, a 2×2×2 based NoC is presented. So we define the dimension as 6. We first define the adjacency matrix as,

$$A = (a_{ij})_{v \times 6} \tag{2}$$

$$\text{where} \quad a_{ij} = \begin{cases} p_{ij}, & \text{global splitting propotion of a flow} \\ & \text{in } j \text{ direction at node } v_i \\ 0, & \text{else} \end{cases} \tag{3}$$

Let $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$, of which $a_{ij}, b_{ij} \in R^+$. Define the logic AND operation of two adjacent

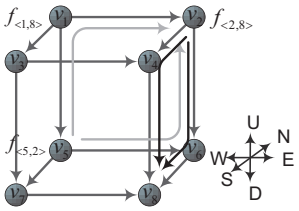

Fig. 2. A contention example in 3D NoC

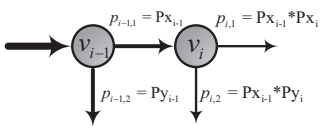

Fig. 3. Global traffic splitting proportion

matrices as follows,

$$A \wedge B = \begin{bmatrix} a_{11} \wedge b_{11} & a_{12} \wedge b_{12} \\ a_{21} \wedge b_{21} & a_{22} \wedge b_{22} \end{bmatrix}, \tag{4}$$

where $\wedge$ is the logic AND. It is defined as

$$a \wedge b = \begin{cases} a, \mathrm{b} \neq 0 \\ 0, \mathrm{b} = 0, \end{cases} \tag{5}$$

where $a \wedge b \neq b \wedge a$. In our application, we take a as the target flow traffic splitting proportion, and b the contention flow. Since the contention matrix is to represent the contention state of the target flow, so when b is 0, it means there is no contention affecting the target flow; else, there exist contentions.

The global traffic splitting proportion of node v_i is shown in Fig. 3. It considers the traffic flows injected into the node, so it is more reasonable than that only uses local traffic splitting Px_i or Py_i (Px_i and Py_i is the local traffic splitting proportion of node v_i). The calculation of the adjacency matrix A is shown in row 6 of Algorithm 1. Assume flow $f_{<1,8>}$ traverses from v_1 to v_8 in Fig. 2. The local splitting proportion at every node is equivalent in all traversal directions. The original proportion at every node is the sum of the splitting proportions of the flows injected into this node. But our adjacency matrix is

$$A_{(1,8)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{bmatrix} E & S & W & N & U & D \\ 0.33 & 0.33 & 0 & 0 & 0 & 0.33 \\ 0 & 0.17 & 0 & 0 & 0 & 0.17 \\ 0.17 & 0 & 0 & 0 & 0 & 0.17 \\ 0 & 0 & 0 & 0 & 0 & 0.33 \\ 0.17 & 0.17 & 0 & 0 & 0 & 0 \\ 0 & 0.33 & 0 & 0 & 0 & 0 \\ 0.33 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6}$$

From above we can see that v_1 has three directions, so the splitting proportion for every direction is 0.33. v_4 only has one transfer direction, and the sum of the splitting proportions of flows injected is 0.33, so the splitting proportion in the down direction is 0.33. Similarly, we can get other elements of this adjacency matrix.

In order to get more accurate network congestion state, flow rate r_i and burstiness b_i of the contention flows are added into the adjacency matrix, with the form of $(\varepsilon r_i + \beta b_i) A_{s_i,d_i}$, where ε and β are the impact factor of r_i and b_i of the target flow i, respectively (shown in line 7 of Algorithm 1). With this expression, different contention flows will bring different effects to target flow, for they have different r_i and b_i. Here we assume $f_{<1,8>}$ to be a contention flow, and $\varepsilon = \beta = 1$ for the convenience of discussion. If the arrival curve of flow $f_{<1,8>}$ is $\alpha = 0.2t+1$, then the improved version of adjacency matrix is

$$B_{(1,8)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{bmatrix} E & S & W & N & U & D \\ 0.396 & 0.396 & 0 & 0 & 0 & 0.396 \\ 0 & 0.204 & 0 & 0 & 0 & 0.204 \\ 0.204 & 0 & 0 & 0 & 0 & 0.204 \\ 0 & 0 & 0 & 0 & 0 & 0.396 \\ 0.204 & 0.204 & 0 & 0 & 0 & 0 \\ 0 & 0.396 & 0 & 0 & 0 & 0 \\ 0.396 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{7}$$

Definition 4.1: Contention Matrix Assume that there are *k* flows $f_{<s_1,d_1>}$, $f_{<s_2,d_2>}$ ... and $f_{<s_k,d_k>}$ in NoC, in which s_i and d_i are the source node and the destination node, respectively. Take $f_{<s_i,d_i>}$ as the target flow, so the others are contention flows. The dimension based contention matrix C_{s_i,d_i} is

$$C_{s_i,d_i} = [(\varepsilon r_1 + \beta b_1) A_{s_1,d_1} + (\varepsilon r_2 + \beta b_2) A_{s_2,d_2} + ... + (\varepsilon r_j + \beta b_j) A_{s_k,d_k} ... + (\varepsilon r_k + \beta b_k) A_{s_k,d_k}] \wedge A_{s_i,d_i}, j \neq i. \tag{8}$$

For simplicity,

$$C_{s_i,d_i} = (B_{s_1,d_1} + B_{s_2,d_2} + ... + B_{s_j,d_j} + ... + B_{s_k,d_k}) \wedge A_{s_i,d_i}, i \neq j \tag{9}$$

Still take Fig. 2 as an example. Let $f_{<1,8>}$ be the target flow. Two other contention flows $f_{<5,2>}$ and $f_{<2,8>}$ are added into NoC to compete for network resource with $f_{<1,8>}$. Assume that the arrival curve of $f_{<5,2>}$ is $\alpha = 0.2t + 1$, and $f_{<2,8>}$ is $\alpha = 0.3t + 2$. The contention flows are even split in every node on their paths. The dimension based adjacency matrices of the two contention flows can be obtained in the same way. Using Eq. 6 and Eq. 9, we obtain the contention matrix of $f_{<1,8>}$ as follows,

$$C_{(1,8)} = \begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{array} \begin{bmatrix} E & S & W & N & U & D \\ 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 & 0.6 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.6 \\ 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.6 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{10}$$

We develop an algorithm to calculate the dimension based contention matrix for each target flow, and the pseudo code is shown in Algorithm 1.

B. Algorithm Complexity

If there are n nodes in 3D network in Algorithm 1 , the calculation of cohesive elements of the adjacency matrices

Algorithm 1 Contention Matrix Generation Algorithm

Require: NoC with router $Rout\_j, 1 \leq j \leq X \times Y \times Z$. Target flow and contention flows $f\langle m_i, n_i\rangle$ with (r_i, b_i), $1 \leq i \leq FlowNum$. The proportion on X, Y and Z direction for the target flow and contention flows.

Ensure: Determine Contention Matrix $ConMatrix$

1: Generate the adjacency matrix $A_i(j,d), 1 \leq j \leq X \times Y \times Z, 1 \leq d \leq 6$ of each flow:
2: **for all** $f\langle m_i, n_i\rangle$ **do**
3: **for all** $Rout\_j, 1 \leq j \leq X \times Y \times Z$ **do**
4: Calculate the sum of proportion $Pin(Rout\_j)$ injected to $Rout\_j$ of $f\langle m_i, n_i\rangle$
5: **for all** $d \in east, west, south, north, up, down$ **do**
6: $A_i(j,d) = Pro(d) \times Pin(Rout\_j)$
7: $B_i(j,d) = (\varepsilon r_i + \beta b_i) \times A_i(j,d)$//Adding the flow's information to the adjacency matrix $A_i(j,d)$;
8: **end for**
9: **end for**
10: **end for**

should be taken for $n \times 6$ times. If there are m flows, the calculation should be taken for $m \times n \times 6$ times. It will take $n \times 6 \times (m-1)$ times to calculate the elements in contention matrix. Therefore, $(2m-1) \times n \times 6$ times of calculation are needed in total. The time complexity of the whole contention matrix is $O(n)$, as m is small and can be ignored.

C. Path Congestion Factor

Through above, we can obtain the local contention state of routers. But we still need to derive the global contention for target flow, moreover, to reveal its relationship against the worst-case delay (or delay bound). So we define the path congestion factor as follows.

Definition 4.2: Path Congestion Factor. We denote by λ the path congestion factor, i.e., the total contention on any sub-flow path, as the sum of contention matrix elements that the flow traverses through, i.e., $\lambda = \sum_{i=1}^{N} C(node\_i, direction\_j)$, where N is the total number of nodes in the sub-flow path, $node\_i$ the *i*-node through the path, and $direction\_j$ the direction next to $node\_i$.

For example, the sub-flow path $v_1(E) \rightarrow v_2(D) \rightarrow v_6(S) \rightarrow v_8$ is one of the sub-flow of $f_{<1,8>}$, so we can calculate the path congestion factor as $\lambda = 0.6 + 0.6 + 0.6 = 1.8$.

V. OLITS

A. Equivalent Resistance

In order to redistribute traffic splitting strategy and improve delay bound, we regard the congestion factor as the resistance of the current sub-flow, and the congestion factors of all target sub-flows as a series of parallel resistances. The current in the network refers to the traffic distributed on its path. The traffic of every sub-flow is redistributed before sending packets, based on the shunt principle in parallel circuit.

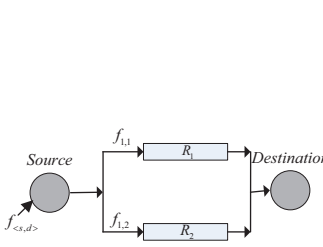

Fig. 4. Ohm's Law-like traffic splitting with two flows

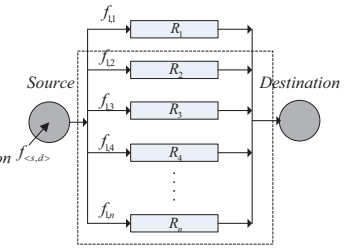

Fig. 5. Ohm's Law-like traffic splitting with n flows

As shown in Fig. 4, there are only two sub-flows from the source to the destination. R_1 is the congestion factor on path $f_{1,1}$, and R_2 of path $f_{1,2}$. The flow $f_{<s,d>}$ should be distributed according to the congestion factor obtained from contention matrix when the flow reaches the two sub-paths. According to the principle in parallel circuit, it is easy to get the traffic proportion of the sub-flows: $f_{1,1}$ is $R_2/(R_1+R_2)$, while $f_{1,2}$ is $R_1/(R_1+R_2)$.

In general, there will be lots of sub-flows from the source node to the destination node, as shown in Fig. 5. Firstly the congestion factor of each sub-flow is obtained from contention matrix, then the congestion factors of all sub-flows are used to redistribute the flow traffic. For example, the total resistance except R_1, the contention factor on the sub-path $f_{1,1}$ should be firstly calculated. In other words, the remaining paths can be regarded as the equivalent parallel resistance R_1^*:

$$R_1^* = 1/(1/R_2 + 1/R_3 + 1/R_4 + ... + 1/R_n). \quad (11)$$

So the proportion $R_1^*/(R_1+R_1^*)$ of flow $f_{<s,d>}$ is captured on path $f_{1,1}$. The traffic on other sub-paths can be redistributed in the same way. We present the algorithm in the next section.

B. OLITS Algorithm

To redistribute the traffic of the target flow for better traffic balance and better worst-case delay, we first calculate the congestion factor λ_{sub_i} of each target sub-flows (line 1-3 in Algorithm 2). Then we compute the parallel resistance except λ_{sub_i} (line 7 to 12). Finally, we calculate the traffic splitting proportions on the $i-th$ sub-path in sequence (line 6 to 14).

If there are n sub-flows in Algorithm 2, the calculation process in row 11 would be carried out for n^2 times. So the complexity of the Ohm's Law-like traffic splitting algorithm is $O\left(n^2\right)$.

VI. Experiments and Results

To verify the effectiveness of OLITS, we perform experiments in the following configurations: (a) comparing with uniform traffic splitting using synthetic patterns under different congestion environment; (b) comparing with general traffic splitting in industrial case.

A. Synthetic Pattern

According to the experimental configuration in Fig. 2, the traffic splitting proportion of three contention flows is adjusted in order to build different congestion environment. Uniform

Algorithm 2 OLITS

Require: Contention Matrix $ConMatrix$, sub-flow's $path\_i, 1 \leq i \leq SubflowNum$ of $f\langle m_{tag}, n_{tag}\rangle$.

Ensure: Determine splitting traffic for each sub-flow of $f\langle m_{tag}, n_{tag}\rangle$.

```
Calculate the congestion factor of each sub-flow
for all f_sub_i do
    λ_sub_i = ConindexGet(path_i, ConMatrix);
end for
Calculate splitting proportion Pro_i on each sub-flow;
for all f_sub_i do
    for all λ_sub_j do
        if i == j then
            continue;
        else
            Temp_i = Temp_i + 1/λ_sub_j;
        end if
    end for
    Pro_i = Temp_i/ (λ_sub_i + Temp_i)
end for
```

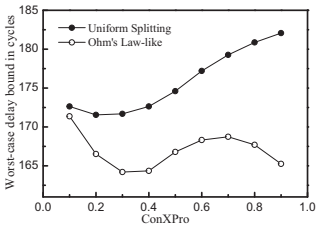

Fig. 6. Worst-case delay comparison between uniform traffic splitting and OLITS

splitting and OLITS are used to get the delay bounds in different environment, as shown in Fig. 6. In this experiment, we change the splitting proportionfrom 0.1 to 0.9 (as the X-axis shown), the Y and Z directions get half of the rest. Fig. 6 shows that the optimization is not obvious when the traffic splitting proportion of flows $f_{<7,19>}$, $f_{<4,20>}$ and $f_{<13,9>}$ in X direction is 0.1. The optimization effect becomes more significant as the traffic splitting proportion in X direction increases and till the maximum. The average improvement of OLITS is 5%, and the maximum gets to 9.24%.

B. Industry Pattern

We map the industry pattern DVOPD [18] in both 2D NoC and 3D NoC, and the results are shown in Fig. 7. The experiment of 2D NoC is conducted with the algorithm proposed in [12]. All the flows are split with uniform splitting, i.e., the traffic splitting proportion is 0.33 in all directions. Both the service curves of the router in 2D and 3D are $\beta = 0.33(t-4)^+$, where 0.33 is flow rate, meaning that the router can transfer one packet in three cycles. Meanwhile, 4 is burstiness, which means that the flow generator sends at most four packets in sequential four cycles.

The worst-case delay of each flow in the two mapping

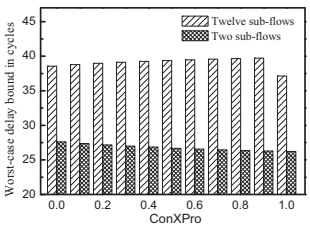

Fig. 8. Comparison of two-flow and full splitting.

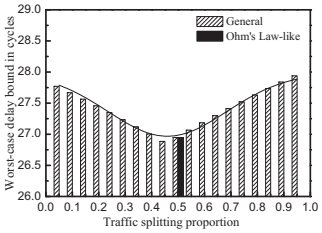

Fig. 9. Improvement effect between OLITS and the contention flows in uniform splitting.

cases are shown in Fig. 7. Flow numbers in X-axis represent the flows in DVOPD. The worst-case delay bounds of most flows in 3D mapping are better than in 2D, with a maximum improvement of 26.06%. Only 7 out of 44 flows in 3D mapping are worse than in 2D mapping. However, the delay bound of the 12th flow $f_{<26,32>}$ in 3D NoC is the maximum one, and is far more than that in 2D mapping.

There may be several factors to cause this happen, such as over traffic splitting and lack of proper splitting proportion. As to the former, we will perform experiments by limit the sub-flow number to two. As to the latter, we will improve the worst-case performance by using OLITS.

$f_{<26,32>}$ is the flow from node 26 to 32 in 3D mapping, and 12 sub-flows are created by full traffic splitting. Under uniform splitting, we change the traffic splitting proportion in X direction from 0 to 1 with step of 0.1, then the delay bounds of the 12 flows are obtained in each type of conflict case. Through this configuration, the conflict can be characterized, and the two optimal sub-flows can be picked out. Then keeping the same conflict environment above, the delay bounds of the two sub-flows can be obtained in each type of conflict scenario. The comparison result is shown in Fig. 8. Compared with full splitting, we can see that the delay bound is significantly improved, when the number of the sub-flows of $f_{<26,32>}$ decreases to 2 in 3D mapping. The results show that the average optimization ratio is 31.42%, and the maximum optimization ratio reaches 33.86%. It is obvious that the impact on network delay brought by over-splitting is great.

Then $f_{<26,32>}$ with only two sub-flows is optimized again by using OLITS. We set the traffic splitting proportion of the first sub-flow filtered to increase from 0.05 to 0.95 by step of 0.05. Meanwhile, the second sub-flow is set to decrease from 0.95 to 0.05 by step of 0.05. The delay bounds of the two sub-flows are respectively calculated in case of different traffic splitting proportions with the contention flows in uniform splitting. The calculated result of traffic splitting proportion using OLITS is 0.500165, and the delay bound is 26.9485, as shown in Fig. 9.

The optimal delay bound can be obtained when traffic splitting proportion of sub-flow1 is 0.45 under this congestion environment. The traffic splitting proportion got by OLITS is very close to it. Likewise, the network delay under this traffic splitting proportion is close to the optimal solution. The

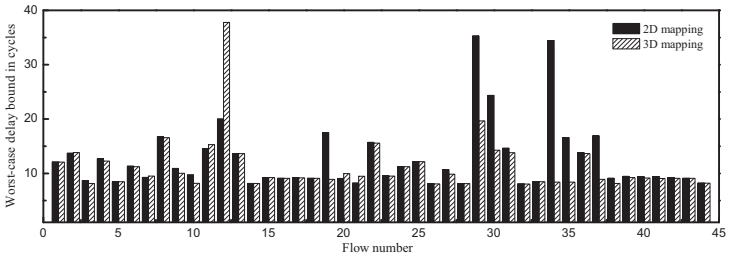

Fig. 7. Comparison results of DVOPD

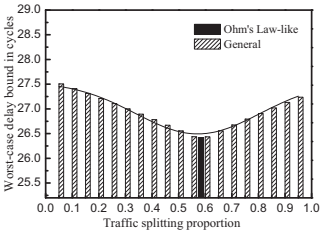

Fig. 10. The optimization effect of Ohm's Law-like splitting with $ConXPro = 0.6$ for contention flows

network delay can be improved by 3.54%, compared with that in the worst split case. The average improvement is 1.7%.

Then the traffic splitting proportion of contention flows turn into 0.6 in X direction, and 0.2 in both Y and Z direction. The above experiment is repeated and another set of results is obtained in the new congestion state (Fig. 10). The split ratio obtained by OLITS is quite close to the best traffic splitting proportion. The delay bound calculated by OLITS gets a maximum improvement of 4%, and average improvement of 1.9%. This traffic splitting proportion basically approaches the optimal traffic splitting proportion.

VII. Conclusions

To improve worst-case performance, we propose OLITS, a congestion-predicting method, to control the traffic over-splitting problem in multi-path routing NoCs. We first redefine contention matrix based on global information and flow parameters such as average rate and burstiness. We then define congestion factor, a variable that describes the congestion between sub-flows, for global congestion prediction. Finally, we propose OLITS to balance global NoC congestion and decrease the target flow delay bound. Experiments show that OLITS achieves an average improvement of 5% and a maximum of 9.24%, compared with uniform traffic splitting. With industrial case, compared with full traffic splitting, OLITS can achieve an average improvement of 31.42% and a maximum of 33.86%. It shows that our OLITS for congestion prediction plays an effective role in worst-case performance analysis.

References

[1] P. Gratz, B. Grot, and S. W. Keckler, "Regional congestion awareness for load balance in networks-on-chip," in *IEEE 14th International Symposium on High Performance Architecture*, 2008, pp. 203–214.

[2] E. Nilsson, M. Millberg, J. Oberg, and A. Jantsch, "Load distribution with the proximity congestion awareness in a network on chip," in *Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 1126–1127.

[3] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan, and C. R. Das, "A low latency router supporting adaptivity for on-chip interconnects," in *Proceedings of the 42nd annual Design Automation Conference*, 2005, pp. 559–564.

[4] G. Ascia, V. Catania, M. Palesi, and D. Patti, "Implementation and analysis of a new selection strategy for adaptive routing in networks-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 6, pp. 809–820, 2008.

[5] E.-J. Chang, H.-K. Hsin, S.-Y. Lin, and A.-Y. Wu, "Path-congestion-aware adaptive routing with a contention prediction scheme for network-on-chip systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 1, pp. 113–126, 2014.

[6] Y.-H. Kuo, P.-A. Tsai, H.-P. Ho, E.-J. Chang, H.-K. Hsin, and A.-Y. Wu, "Path-diversity-aware adaptive routing in network-on-chip systems," in *IEEE 6th International Symposium on Embedded Multicore SoCs*, 2012, pp. 175–182.

[7] R. L. Cruz, "A calculus for network delay, part I: Network elements in isolation," *IEEE Transactions on Information Theory*, vol. 37, no. 1, pp. 114–131, January 1991.

[8] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Number 2050 in LNCS, Springer-Verlag, 2004.

[9] Y. Qian, Z. Lu, and W. Dou, "Analysis of worst-case delay bounds for on-chip packet-switching networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 29, no. 5, pp. 802 – 815, May 2010.

[10] ——, "Analysis of worst-case delay bounds for best-effort communication in wormhole networks on chip," in *Proceedings of the 3rd ACM/IEEE International Symposium on Networks-on-Chip*, 2009, pp. 44–53.

[11] Y. Qian, Z. Lu, W. Dou, and Q. Dou, "Analyzing credit-based router-to-router flow control for on-chip networks," *IEICE Transactions on Electronics, Special Section on "Hardware and Software Technologies on Advanced Microprocessors"*, vol. E92-C, no. 10, pp. 1276–1283, Oct. 2009.

[12] G. Du, C. Zhang, Z. Lu, A. Saggio, and M. Gao, "Worst-case performance analysis of 2-D mesh NoCs using multi-path minimal routing," in *CODES+ISSS*, 2012, pp. 123–132.

[13] Z. Lu, M. Millberg, A. Jantsch, A. Bruce, P. van der Wolf, and T. Henriksson, "Flow regulation for on-chip communication," in *DATE*, 2009, pp. 578–581.

[14] M. Bakhouya, S. Suboh, J. Gaber, T. A. El-Ghazawi, and S. Niar, "Performance evaluation and design tradeoffs of on-chip interconnect architectures," *Simulation Modelling Practice and Theory*, vol. 19, no. 6, pp. 1496–1505, 2011.

[15] R. Tornero, J. M. Orduna, M. Palesi, and J. Duato, "A topology-independent mapping technique for application-specific networks-on-chip," *Computing and Informatics*, vol. 31, no. 5, pp. 939–970, 2013.

[16] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea, "Tight end-to-end per-flow delay bounds in fifo multiplexing sink-tree networks," *Perform. Eval.*, vol. 63, no. 9-10, pp. 956–987, 2006.

[17] G. Du, M. Li, Z. Lu, M. Gao, and C. Wang, "An analytical model for worst-case reorder buffer size of multi-path minimal routing NoCs," in *8th IEEE/ACM International Symposium on Networks-on-Chip*, 2014, pp. 49 – 56.

[18] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60–76, 2013.