NTPT: On the End-to-End Traffic Prediction in the On-Chip Networks

Yoshi Shih-Chieh Huang[†], Kaven Chun-Kai Chou[†], Chung-Ta King[†], Shau-Yin Tseng[‡]

[†]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[‡]SoC Technology Center, Industrial Technology Research Institute, Hsinchu, Taiwan
[†]{yoshi, kavenc, king}@cs.nthu.edu.tw, [‡]tseng@itri.org.tw

ABSTRACT

Power and thermal distribution are critical issues in chip multiprocessors (CMPs). Most previous studies focus on cores and on-chip memory subsystems and discuss how to reduce their power and control thermal distribution by using dynamic voltage/frequency scaling. However, the on-chip interconnection network, or *network-on-chip* (NoC), is also an important source of power consumption and heat generation. Particularly, the traffic flowing through the NoC affects directly its power and thermal distribution. Unfortunately, very few works discuss the dynamism of NoC. A key technique for NoC management is to capture its traffic patterns and predict future behaviors. In this paper, we propose a table-driven predictor called *Network Traffic Prediction Table* (NTPT) for recording and predicting traffic in NoC. The most unique feature of NTPT is its ability to predict end-to-end traffic, rather than switch-to-switch traffic. Thus, more application behaviors can be captured and monitored. Evaluations on Tilera's TILE64 show that NTPT has very high prediction accuracy. Analyses also show that it incurs a low area overhead and is very feasible.

Categories and Subject Descriptors

B.4 [**Input/output and data communications**]: Processors

General Terms

network-on-chip, many-core, end-to-end traffic prediction, power management

1. INTRODUCTION

With the progress of VLSI technology, the number of cores on a chip multiprocessor (CMP) keeps increasing. To interconnect the many cores on the chip, perhaps to the thousands, a Network-on-Chip (NoC) is essential. For example, Tilera's TILE64 uses a 2D mesh network to interconnect 64 general-purpose tiles with supports for explicit tile-to-tile communication. As NoC becomes one of the most critical components on a CMP, its design and behavior affect every aspect of the CMP, from design complexity and chip area to run-time performance, power consumption, and thermal distribution. Particularly, NoC is becoming a major source of power consumption and heat generation on chip [4, 8]. When packets go through a switch of the NoC, power will be consumed by the switch and heat will be generated. If it is possible to predict the traffic flowing through a specific switch in the following time interval, we can adjust its power mode accordingly [7] to save power while maintaining the performance.

Predicting traffic for NoC is challenging, especially from the perspective of each individual switch. For one thing, traffic generated by one core may be consumed by a core at the other end of the NoC, going through several switches in between. The intermediate switches must know the end-to-end communications to correctly predict the through traffic for the next time interval. This in turn requires knowledge of the running application. Previous works on switch-to-switch traffic prediction [6] are limited in that they do not know any application information and make predictions based only on the states of the switches, e.g., the buffer status. The predictions are at most passive and reactive with a local view. It is difficult to anticipate changes in traffic behaviors, such as phase changes in the application or bursts in the traffic, and react accordingly in time. In this paper, we focus on the end-to-end (E2E) traffic of the NoC.

Since most prediction techniques are based on past history, given that there is no knowledge of future behavior, another challenge in predicting traffic of NoC is to capture the past traffic behaviors with the minimal resources. In this work, we propose to use a small table in the network interface (NI) of each tile to record the traffic and help prediction. The proposed *Network Traffic Prediction Table* (NTPT) can predict the end-to-end traffic in a NoC. Inspired by the 2-level branch predictor [10], we use a local predictor and a global predictor. In the local predictor, prediction is based on the last value. On the other hand, the global predictor uses a table to record transmission patterns, which are then used to index the prediction value.

The main challenge of the design for NTPT is to reduce the size of the tables while maintaining high prediction accuracy. We will discuss in detail how this issue is addressed in our design. Our discussions are based on the TILE64 architecture, which supports explicit tile-to-tile communication through the iLib library. Traffic from implicit communications such as the shared L2-cache and coherence traffic are

DAC 2010, June 13-18, 2010, Anaheim, California, USA.

not considered. The proposed technique is very general and should be applicable to other CMP architectures. With the traffic prediction from NTPT, we can perform various operations to improve CMP performance and power consumption:

- End-to-end flow control by controlling the injection rate.
- Setting appropriate power modes of the switches.
- Modeling thermal distribution of a chip taking both cores and switches into consideration.
- Optimizing inter-core communications at runtime.

This paper is organized as follows. Section 2 discusses related works. In Section 3, we give a formal definition of our problem. In Section 4, we introduce the design of our traffic prediction scheme and several techniques for reducing the size of the tables. Section 5 presents evaluations of our design in terms of prediction accuracy and effects of design parameters. Finally, conclusions of the paper and suggestions for future works are given.

2. RELATED WORKS

A general strategy for predicting the future is to capture the past behaviors. Most existing works rely on simple counters for capturing past behaviors, due to the simplicity and low area overhead of such counters. In [3], counters and tables are used to identify whether running tasks are computation- or memory-bound. Tables have also been used widely in branch prediction and cache prefetching [5]. The tables can capture both temporal and spatial behaviors for accurate future predictions. Unfortunately, even though prediction tables have been applied successfully and widely to predict branches and cache accesses, there are very few works on predicting traffic in NoC. In [6], information exchanged between adjacent switches is used to predict the pressure of the traffic and make flow control decisions. The prediction is from the switch perspective and does not take account of the application behaviors. As far as we know, NTPT is the first work to address end-to-end traffic prediction for NoC from the perspective of applications.

3. PROBLEM FORMULATION

For simplicity of discussion, we assume an NoC with a 2D mesh topology. The size of the mesh network is $M \times M$. Note that our solution is general and independent of the underlying topology. We also assume that each core runs one task. Thus, the terms *task* and *core* will be used interchangeably in the following discussions. Again, this assumption can be easily relaxed by duplicating the NTPT. The application running on the CMP can be represented as a *task graph*, denoted as T. The task graph consists of a set of tasks, each runs on one core. Let $comm_i$ denote the set of tasks which task i may communicate to, i.e., $comm_i = \{j | i \rightarrow j\}$. Let $p_t(i,j)$ be the traffic predicted by NTPT to flow from task i to j at the t-th time interval, where $i, j \in T$ and $j \in comm_i$. Note that this is end-to-end traffic. Let $c_t(i,j)$ be the actual traffic from task i to j at the t-th time interval. The goal of traffic prediction is to make the difference between $p_t(i,j)$ and $c_t(i,j)$ as small as possible, i.e., the prediction is as accurate as possible.

4. SYSTEM DESIGN

4.1 The Design of NTPT

NTPT is a two-level table designed for predicting the traffic of NoC in the next time interval based on the history. The design is inspired by the branch prediction table. In this subsection, we first introduce the notations used and then describe the design of NTPT.

There are two predictors supported by the NTPT, *local predictor* and *global predictor*. The local predictor makes predictions based on the last activity saved in $counter_{i,j}$, where i and j are the source and destination task, respectively. The local predictor is usually accurate if the running application has a stable and consistent behavior. On the other hand, the global predictor makes predictions based on the communication history pattern. It makes different predictions for each tracked communication pattern. NTPT has a selector to decide which predictor to use at runtime.

Let $history_{i,j}$ be the record of the communication pattern between tasks i and j in NTPT. Let $table_{i,j}$ be the global prediction table, in which each entry is indexed by $history_{i,j}$ and contains a prediction value. The prediction value records the amount of data transmitted when this pattern was encountered last time. If the same pattern appears again, the recorded value is used as the prediction value. Let Δ denote the *sampling period*. The capacity of each link is W bits/s.

After introducing the notations, we are now ready to describe the design of NTPT by illustrating the mechanisms for monitoring and predicting the traffic produced by a task. Consider a task i and the task j with which i may communicate, i.e., $j \in comm_i$. The prediction function can be defined as follows:

$$p_t(i,j) = \begin{cases} c_{t-1}(i,j), & s(i,j) = 0; \\ table_{i,j}(history_{i,j}), & s(i,j) = 1. \end{cases}$$

Recall that $p_t(i,j)$ is the predicted traffic from task i to task j at the t-th interval. The prediction either comes from the local predictor or the global predictor, determined by a selector function s. The selector function can be designed according to the system requirements. In this paper, we use a 2-bit saturating counter.

Each entry in the NTPT has a counter for recording the total size of transmitted data in the current time interval, a shift register for tracking the history of traffic pattern, and a flag indicating whether there is an outgoing traffic in the current time interval. The traffic pattern is updated with the latest traffic behavior (i.e., the transmission flag) at the end of each time interval. The traffic pattern is used to index a global history table. The global history table stores the prediction for each different traffic pattern.

When task i generates an outgoing traffic to task j, the NTPT records the size of the transmitted data and the destination, and sets the transmission flag indicating that there is an outgoing traffic in the current time interval. At the end of the current time interval, the NTPT shifts the transmission flag into the traffic pattern shift register and makes a traffic prediction for the next time interval either by the local predictor or the global predictor. The prediction for the size of the data to be transmitted is a pure last-value prediction, i.e., the size is the same as in the last time interval.

Besides, the resolution for the size of the transmitted data

can be set by the quantizer G bits/unit according to the application. It follows that the most coarse-grained record is to set G to W. In this way, $\lceil c_t(i,j)/(G \cdot \Delta) \rceil$ is either 0 or 1 for some task i and j at t-th time period, indicating the link either transmits or not at all. When we are going to do fine-grained prediction, we can set a smaller G for finer adjustments.

4.2 Practical Implementation Details

NTPT can be implemented as a two-level hierarchical table. The first-level table is indexed by the id of the destination core. Each entry contains the number of packets transmitted most recently and the recent traffic patterns. The second-level table is indexed by the history of the traffic patterns, and each entry contains the predicted transmission rate.

Due to space limitation, the technical details for reducing the table size and estimating the area overhead are omitted here. Main techniques used are listed as follows and interested readers are referred to the technical report [2]. (1) Reducing the size of the first-level table, (2) Sharing the second-level table, (3) Quantizing the values in the second-level table, (4) Quantizing the size of transmitted data, and (5) Entry replacement in the second-level table.

Assume that the NoC has a topology of 5×5 mesh. Each network interface thus needs to maintain a table with 5^2 entries. After applying all the reduction strategies above and using LRU replacement in the 2nd-level table, we can show that the table can be reduced to 56 bytes.

5. EVALUATION

In this section, we evaluate the accuracy of our NTPT traffic prediction method using different time interval settings, and compare our method with local and global predictors.

5.1 Methodology

We use Tilera's TILE64 as the evaluation platform [1]. For each NTPT in the system, we use a dedicated tile to simulate its behavior, such as tracking outgoing traffic and updating the communication pattern register. In other words, if we are going to evaluate NTPT on a CMP with n tiles, we will use n extra tiles in TILE64 for simulating the behavior of corresponding NTPTs. We define *WORKER* as the set containing the tiles which are running the application and *UPDATER* as the set containing the tiles which are simulating the behavior of NTPT. The relation between the tiles in *WORKER* and the tiles in *UPDATER* is defined as follows: $ut_i \in UPDATER$ is updating the NTPT of $wt_j \in WORKER$, if $i = j$.

Our evaluation uses a modified blocked LU decomposition kernel from the SPLASH-2 suite [9] as the benchmark. The blocked LU decomposition kernel from SPLASH-2 is a data-parallel shared-memory program. For our evaluation, we made two modifications to the benchmark. First, in the initialization stage, we make each tile in *WORKER* to allocate and cache its own portion of the data memory space. Second, when wt_i needs data which belongs to wt_j, we use the iLib tile-to-tile message passing API to transmit the data from wt_j to wt_i. Meanwhile, a notification containing the information of this transmission will be sent to ut_j for NTPT simulation.

The number of arrays in the LU decomposition kernel is set as 50, and each of them is a 512×512 64-bit floating point array generated randomly at initialization. Each array is separated into 1024 16×16 subarrays and dispatched to 16 tiles for parallel processing.

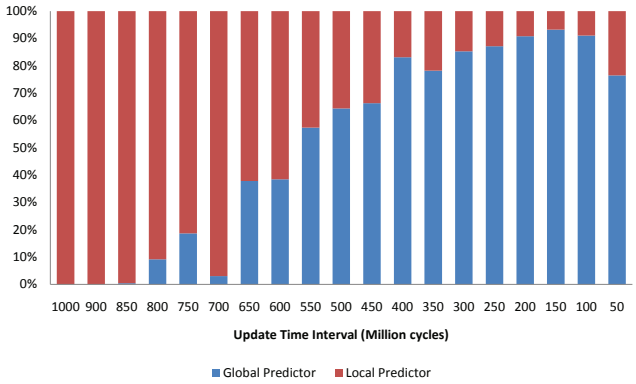

Figure 1: The ratio of using global predictor and local predictor under different time intervals.

5.2 The Impact of the NTPT Update Time Interval

Update time interval setting is a critical factor affecting the prediction accuracy and the applicability of the prediction results. In this experiment, we try to find a *suitable* time interval for our benchmark program. A *suitable* time interval should be able to capture the communication behavior embedded in the application logics and predict the network traffic with acceptable accuracy. We have tested the time interval settings ranging from 1,000,000,000 cycles to 50,000,000 cycles, as shown in Figure 1.

While using larger time interval settings (over 800,000,000 cycles), the prediction error rate is relatively low, but the global predictor is rarely selected. Although we can get a high prediction accuracy by using large time interval settings, the low global predictor usage means that NTPT does not observe much variation in the communication behavior. This indicates that it misses the dynamism of the NoC traffic and thus the prediction is not applicable. Similar results can be observed if applying relatively small time interval settings (under 50,000,000 cycles).

For our benchmark, with the time interval settings ranging from 100,000,000 cycles to 400,000,000 cycles, the global predictor is frequently selected. This indicates that NTPT observes a varying communication behavior, and the prediction can tell us whether the NoC is busy or not. In summary, the time interval settings ranging from 100,000,000 cycles to 400,000,000 cycles allow NTPT to capture the communication behavior inherited in our benchmark program.

5.3 Comparison between Local and Global Predictors

In this subsection, we evaluate the performance of our NTPT-based predictor by comparing with a local predictor and a global predictor. The prediction error rates of the three different predictors with different update interval settings are shown in Figure 2. The local predictor (diamond dotted line) suffers from high error rates using update

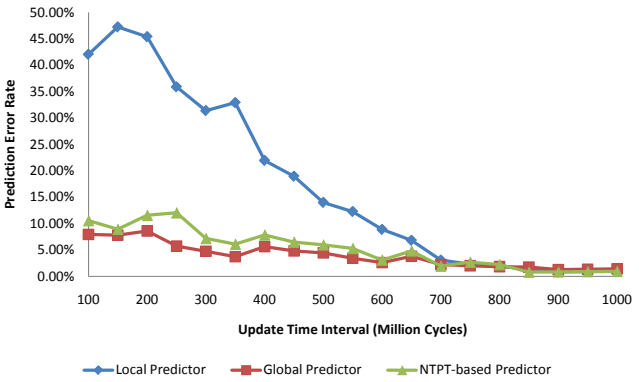

Figure 2: Prediction error rate of the three types of predictors.

time intervals ranging from 150,000,000 to 650,000,000 cycles, because the communication traffic with these time interval settings varies widely as monitored by NTPT. While using larger time intervals, the local predictor has a good prediction accuracy since the communication patterns are almost the same, again as shown in each NTPT update.

On the other hand, the global predictor (square dotted line) performs well in all the evaluated update time intervals. The global predictor can work as a local predictor when the two entries of **00000000** and **11111111** have been filled in the 2nd-level table, which stand for the communication patterns of *no transmission* and *transmit at all times*. This results in high accuracy predictions in low-variance communication patterns. The global predictor, by its nature, is also capable of predicting accurately in high-variance communication patterns.

However, the global predictor uses more memory space than the local predictor. The accuracy of the global predictor is determined mainly by the number of patterns that can be tracked in the 2nd-level table. Currently, we have not limited the size of the 2nd-level table, and therefore all the patterns can be tracked and used to make predictions. But in a practical implementation, the 2nd-level table may not be large enough to keep all the patterns throughout the application execution. The accuracy will be affected greatly by the replacement policy. When a longer global history is needed, the number of total tracked patterns (the difference between the global predictor and the NTPT-based predictor) will increase as Table 1 shows.

To summarize, the NTPT-based predictor (triangle dotted line) adaptively selects a local predictor in low-variance traffic and a global predictor for high-variance traffic. Moreover, the experiments show that the NTPT uses fewer entries than the global predictor and thus incurs a lower space overhead.

6. CONCLUSIONS

In this paper, we propose a two-level table design called Network Traffic Prediction Table (NTPT) for predicting end-to-end traffic of the NoC. The design is introduced and the area overhead is analyzed. For evaluation, we port the LU decomposition in SPLASH-2 with NTPT simulation to the TILE64 platform. The prediction accuracy of the proposed

Table 1: Total patterns tracked by pure global predictor and NTPT-based predictor by running a modified SPLASH-2 blocked LU decomposition benchmark with 16 tiles and 600,000,000 update time interval.

Global History Length	Global Predictor	NTPT
4	92	26
6	132	37
8	218	60
10	233	75
12	275	87
14	346	95
16	407	167

NTPT method is then evaluated based on different time interval settings and by comparing with pure local and global prediction. The results show that NTPT performs well. In the future, we will apply NTPT to adjusting the link frequencies for reducing the power consumption in NoC.

Acknowledgements

This work is funded by the Industrial Technology Research Institute and National Science Council grant NSC 98-2220-E-007-019.

7. REFERENCES

[1] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, L. Bao, J. Brown, M. Mattina, C.-C. Miao, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, and J. Zook. Tile64 - processor: A 64-core soc with mesh interconnect. In *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, pages 88–598, Feb. 2008.

[2] Y. S.-C. Huang, K. C.-K. Chou, and C.-T. King. Ntpt: On the end-to-end traffic prediction in the on-chip networks. Technical report, 2010.

[3] C. Isci, G. Contreras, and M. Martonosi. Live, runtime phase monitoring and prediction on real systems with application to dynamic power management. In *MICRO 39: Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 359–370, Washington, DC, USA, 2006. IEEE Computer Society.

[4] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In *Proc. DATE '09. Design, Automation. Test in Europe Conference. Exhibition*, pages 423–428, Apr. 20–24, 2009.

[5] K. J. Nesbit and J. E. Smith. Data cache prefetching using a global history buffer. In *Proc. 10th International Symposium on HPCA-10 High Performance Computer Architecture*, page 96, Feb. 14–18, 2004.

[6] U. Y. Ogras and R. Marculescu. Prediction-based flow control for network-on-chip traffic. In *Proc. 43rd ACM/IEEE Design Automation Conference*, pages 839–844, 2006.

[7] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proc. Ninth International Symposium on High-Performance Computer Architecture HPCA-9 2003*, pages 91–102, Feb. 8–12, 2003.

[8] L. Shang, L.-S. Peh, A. Kumar, and N. K. Jha. Temperature-aware on-chip networks. 26(1):130–139, Jan. 2006.

[9] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The splash-2 programs: characterization and methodological considerations. In *ISCA '95: Proceedings of the 22nd annual international symposium on Computer architecture*, pages 24–36, New York, NY, USA, 1995. ACM.

[10] T.-Y. Yeh and Y. N. Patt. Alternative implementations of two-level adaptive branch prediction. In *Proc. 19th Annual International Symposium on Computer Architecture*, pages 124–134, May 1992.