HyDMA: low-latency inter-core DMA based on a hybrid packet-circuit switching network-on-chip

Zhenqi Wei\(^{1(a)}\), Peilin Liu\(^1\), Rongdi Sun\(^1\), Zunquan Zhou\(^1\), Ke Jin\(^1\), and Dajiang Zhou\(^2\)

\(^1\) Department of Electronic Engineering, Shanghai Jiao Tong University, 800 Dongchuan Rd, Minhang District, Shanghai 200240, China
\(^2\) Graduate School of Information, Production and Systems, Waseda University, 2–7 Hibikino, Kitakyushu, Fukuoka 808–0135, Japan

\(^a\) awaylovemusic@sjtu.edu.cn

Abstract: With a growing number of cores integrated in a single chip, the efficiency of inter-core direct memory access (DMA) transfers has an increasingly significant impact on the overall performance of parallel applications running on network-on-chip (NoC) processors. In this paper we propose HyDMA, a low-latency inter-core DMA approach based on a hybrid packet-circuit switching NoC. With dynamic setup and lengthening of circuit channels composing of bidirectional links, HyDMA can achieve both high flexibility of packet switching and low communication latency of circuit switching for concurrent DMA transfers. Experimental results prove HyDMA exhibits high efficiency with marginal hardware overhead.

Keywords: NoC, DMA, packet-circuit switching, circuit setup, bidirectional link

Classification: Integrated circuits

References

1 Introduction

As the number of processing cores integrated in a single chip keeps increasing, the efficiency of inter-core communication has a non-ignorable impact on the performance gain of parallel computing [1]. In applications with high degrees of data parallelism, large amounts of data stored in a contiguous address space are usually transmitted between cores. Compared to explicit data movements by software, direct memory access (DMA) is preferred because a processing core can run concurrently to data transfer after configuring a DMA controller and is thereby offloaded. Thus the latency of inter-core DMA transfers is vital to the whole performance of parallel applications running on many-core processors.

On-chip networks of many-core processors have been extensively studied, while most solutions are applied to general on-chip traffic, which offer limited improvements for DMA transfers. In this paper we propose HyDMA, a low-latency DMA approach for many-core processors based on a hybrid switching network-on-chip (NoC), which adopts packet switching (PS) and circuit switching (CS) in two sub-networks respectively. Following the deterministic routing paths of data flits transmitted in the PS sub-network (Pnet), circuit channels are dynamically allocated for DMA transfers in the CS sub-network (Cnet). In order to overcome
the drawbacks of CS, including long latency of circuit setup and blocking of other on-chip traffic, data requests of a DMA transfer can travel through Cnet channels starting from intermediate cores. The partially built channels are further lengthened when adjacent circuits are also reserved for the same DMA transfer. Pairs of bidirectional links connecting Cnet crossbars can change their directions to perform concurrent DMA transfers with overlapping transmission routes. Besides, the Cnet can also be used for non-DMA requests according to manual configuration on crossbar connections.

The remainder of this paper is organized as follows. Section 2 introduces related works. Section 3 gives the design details of HyDMA. Section 4 presents the performance evaluation along with the hardware cost of HyDMA from experimental results. Finally, we present our conclusion in Section 5.

2 Related works

Quite a few techniques have been put forwarded to optimize inter-core communication in NoCs. Zheng et al. [2] design a wormhole switched router with a deadlock-free routing algorithm to support tree-based multicast communication. Kumar et al. [3] use Express Virtual Channels (EVCs) to allow packets to skip at most three intermediate routers in one dimension. Krishna et al. [4] achieve Single-cycle Multi-hop Asynchronous Repeated Traversal (SMART) in a NoC with the presetting of static crossbar connections. Modarressi et al. [5] and Jiang et al. [6] respectively propose virtual point-to-point (VIP) paths and virtual circuit-switched (VCS) connections to bypass pipeline stages of traversed routers. However, both techniques need a root node to monitor network traffic and perform path allocation in runtime. Al Faruque et al. [7] apply centralized configuration on directions of 2X-Links to implement bidirectional transmission. Lan et al. [8] propose self-reconfigurable bidirectional channels with direction control protocol to satisfy bandwidth requirement. Qian et al. [9] further propose a flit-level speedup scheme by allowing flits of a packet to be transmitted in both inter-router channels simultaneously.

Various attempts have also been made in combining PS and CS in one on-chip network. Ou et al. [10] propose a double-layer NoC to transmit scattered data and control information in packet layer and bulk data in circuit layer, where routing is wholly determined by manual configurations in advance. Modarressi et al. [11] and Lin et al. [12] divide NoC links and switch allocators between PS and CS sub-networks using spatial and time division multiplexing respectively to reduce packet latency or power consumption. Chen et al. [13] and Abousamra et al. [14] both use PS requests stored in reservation queues to allocate crossbar connections for circuit paths. While the reservation of both forward and backward paths results in under-utilization of network resources. Jerger et al. [15] propose a hybrid network where data can be piggy-backed immediately behind a circuit setup request. When the setup request finds no available circuits, the least recently used circuit will be reconfigured as PS for the following data flits, which may stop or alter other CS transfers. Leveraging both hybrid switching method and bidirectional links, HyDMA can dynamically build and lengthen Cnet channels without blocking other
Cnet traffic, which achieves low-latency inter-core communication for concurrent DMA transfers performed in a NoC processor.

3 Design of HyDMA

![Diagram of a NoC processor using HyDMA](image)

Fig. 1. Architecture of a NoC processor using HyDMA

Without loss of generality, we consider a NoC-based many-core processor with 2D mesh-grid topology as our baseline processor, which is shown in Fig. 1(a). As Fig. 1(b) depicts, a DMA controller within each processing core is configured by a local instruction pipeline before the controller initiates a DMA transfer. The processor can also realize inter-core data sharing or message passing by transmitting non-DMA requests between globally addressable memory banks, mailboxes, or synchronization controllers included in on-chip cores. After communication requests have been packetized by network interfaces, generated flits are forwarded by PS routers composing of multiple pipeline stages (see Fig. 1(c)) along unidirectional links (see left part of Fig. 1(d)) in the Pnet. While CS crossbars (see Fig. 1(e)) are configured by passing Pnet flits to set up circuit channels dynamically, and then messages can be transmitted through bidirectional links (see right part of Fig. 1(d)) in the Cnet.

3.1 Cnet crossbar & port configuration

As shown in Fig. 2, along with Cnet messages, flits inputted to and outputted from a Pnet router are both fed to the coupled crossbar to configure directions of bidirectional crossbar ports, according to the type of flits and messages listed in Table I. Connections between crossbar ports are updated by sets of control registers, including an 8-bit register SRC saving the source core ID of received Pnet flits, a 1-bit register RET indicating whether a port is used to transmit returned data, a 1-bit register DIR deciding the direction of the port (1 for input and 0 for output), and two 1-bit registers HOLD and LOCK showing the port status. The selection between Pnet flits and Cnet messages of the two gray MUXs in the figure to configure sets of control registers for crossbar ports is explained later in this section (see Fig. 3 for detail). When two ports with different directions are both held by the same source core, the connection between them is established within the
crossbar. Besides, if one of the two held ports has already been locked, a lock message will be sent from the other port to set up a dedicated channel along previously traversed crossbars. By saving processing time of flits in pipelined routers of the Pnet, the Cnet driven by the same clock can transmit messages with a delay of single cycle per dimension in medium-scale 2D-mesh NoC, which is similar to [4].

![Cnet crossbar diagram](image)

**Fig. 2.** Cnet crossbar

When data flits with type ‘hold’ are initially transmitted in the Pnet, their source core ID is saved in crossbar ports of traversed cores. Pnet hold flits can always update the allocation of crossbar ports before they are locked, which avoids deadlock of building a Cnet channel for multiple source cores. To tackle with the potential competition of a crossbar bidirectional port between Pnet input and output hold flits, we apply a simple scheme based on the preference of crossbar ‘input’ and ‘output’ ports with opposite default directions for input and output flits respectively. Held crossbar ports change their directions to the opposite of Pnet flits to later transmit Cnet messages with type ‘lock’. The only exception is that the local port held by an output flit keeps output direction and is also locked to generate a lock message. On receiving the lock message, crossbar ports reverse their directions with LOCK registers turned on to transmit data requests in the Cnet. During the transmission of ‘free’ flits or messages, crossbar ports allocated to the same source core will be released. Fig. 3 shows the configuration of control registers for crossbar ‘input’ port [X][I], where X can be L (Local), N (North), E (East), S (South) and W (West). While a similar configuration method is applied to ‘output’ port [X][O]. With manual configurations on control registers of crossbar ports, inter-core communication for non-DMA transfers can also be performed in the Cnet.

### Table 1. Types of Pnet flits and Cnet messages

<table>
<thead>
<tr>
<th>Type</th>
<th>Pnet Flits</th>
<th>Cnet Messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hold</td>
<td>Hold crossbar ports</td>
<td>Transmit in Cnet channels</td>
</tr>
<tr>
<td>Lock</td>
<td>Manually configure crossbar ports</td>
<td>Lock crossbar ports</td>
</tr>
<tr>
<td>Free</td>
<td>Release crossbar ports and remove Cnet channels</td>
<td></td>
</tr>
</tbody>
</table>

© IEICE 2016
DOI: 10.1587/elex.13.20160529
Received May 24, 2016
Accepted June 8, 2016
Publicized June 28, 2016
Copyedited July 25, 2016

IEICE Electronics Express, Vol.13, No.14, 1–12
3.2 DMA transfers using HyDMA

Fig. 4 shows the update of crossbar control registers during the setup and release of a Cnet channel for a DMA transfer between two adjacent cores 00 and 10. As depicted in Fig. 4(a), the source core ID 00 is stored into SRC registers of core 00’s \([L][I]\) and \([L][O]\) ports, and core 10’s \([W][I]\) and \([L][O]\) ports when Pnet hold fits within the DMA transfer are fed to these ports. Since core 10’s \([L][O]\) port is locked to output direction, a lock message is issued from core 10’s \([W][I]\) port to lock previously held ports and reverse their directions, which is shown in Fig. 4(b). Then Cnet hold messages converted from Pnet hold fits travel through the newly built circuit channel, which is demonstrated in Fig. 4(c). As shown in Fig. 4(d), when the last DMA request with type free traverses the Cnet, control registers of configured ports are all reset and thus the dedicated channel is removed. In order to dynamically build and lengthen Cnet channels following the routes of Pnet fits, we apply deterministic routing algorithm (eg. XY routing) in the Pnet to lead the transmission of both forward fits with data requests and backward fits with returned data.

![Fig. 4. Update of control registers during a DMA transfer](image-url)
In Fig. 5, three DMA transfers adopting HyDMA are concurrently performed in a $3 \times 3$ NoC with the mostly used XY routing, and each core is numbered with a unique ID. The data sending transfer from core 12 to core 01 and the data fetching transfer between cores 10 and 20 are both performed in the Cnet, while core 00 initiates another DMA transfer in the Pnet to fetch data from core 22. For comparison with HyDMA, we also demonstrate the setup of circuit channel for a DMA transfer sending data from core 01 to core 21, which uses a previously proposed hybrid PS-CS method [13]. Not all ports are displayed in the figure for clarity purpose.

As shown in Fig. 5(a), core 00 initiates a DMA transfer to fetch data from core 22 by sending hold flits in the Pnet (see ③). Crossbar ports of traversed cores according to the XY routing are allocated to core 00 with opposite directions to those of the flits. However, core 10’s east ports and core 20’s west ports are still locked by the fetching transfer between the two cores. After one of the destination core 22’s south ports is held, a Cnet lock message is issued from the port and travels along the same route of the Pnet hold flits (see ②). Reversing the direction of allocated crossbar ports, the lock message is finally blocked at core 20 and is thus discarded in the Cnet. A dedicated Cnet channel is set up for core 00’s DMA transfer to bypass router stages of cores 21 and 22 (see ④). The source core ID 00 is saved in traversed cores’ unlocked crossbar ports with their RET registers being turned on. Since only one 12-02-01 channel is occupied by core 12’s sending transfer, a Cnet lock message sent from core 00 successfully reaches core 22 (see ⑤). As depicted in Fig. 5(c), the returned data from core 22 can be sent back along the complete 22-00 channel in the Cnet with much lower latency than in the Pnet (see ⑥).

Meanwhile, the DMA transfer between cores 10 and 20 ends up with Cnet free messages. After being released by free messages, crossbar ports between cores 10 and 20 are re-held by core 00’s hold flits (see ⑦). Then a lock message generated by core 20’s west port traverses the Cnet heading to the source core 00, which extends the 21-22 channel to the full length between cores 00 and 22 (see ⑧). Eventually, core 00’s DMA transfer is wholly performed in the Cnet (see ⑨ and ⑩), which is shown in Fig. 5(d).
For the DMA transfer with the hybrid PS-CS method between cores 01 and 21, on the other hand, a Pnet request is first sent from core 01 to reserve crossbar ports along the 01-11-21 route (see \( C_1 \)), which is demonstrated in Fig. 5(c). Only on successful building of both forward and backward channels, an acknowledgement message is sent to the source core along the backward channel (see \( C_5 \)). As shown in Fig. 5(d), the DMA transfer starts with both channels occupied (see \( C_2 \)), regardless of whether the backward one is needed for returned data, which results in underutilization of network resources. In HyDMA, by contrast, only one channel is required for a DMA sending transfer, such as the transfer between cores 12 and 01.

4 Experimental results

We finish the hardware implementation of HyDMA based on a NoC processor, which integrates 32-bit RISC cores with a five-stage pipeline and two-stage routers [16] with two 16-entry VCs for each port. The baseline PS NoC applies the XY routing algorithm to transfer packets along pairs of unidirectional 64-bit links. The original network is separated into a Pnet and a Cnet with same width of unidirectional and bidirectional links respectively, and thus the size of Pnet flits and Cnet messages are both 32 bits. After synthesizing the NoC processor in SMIC® 40 nm technology at a working frequency of 400 MHz for the both sub-networks, we get circuit area cost and average runtime power consumption of HyDMA when performing DMA transfers. A crossbar switch for Cnet communication consumes 5.1% area (13.0 K GC (Gate Count)) and 11.4% power (0.50 mW) of a single on-chip core, while a PS router along with a network interface occupies 11.5% area (29.3 K GC) and 36.5% power (1.61 mW) per core. Besides, a single-channel DMA controller with minor modification to add a type field in data requests only takes 2.9% area (7.4 K GC) and 5.6% power (0.25 mW). Overall, HyDMA imposes about 2% area and 7% power overhead compared to the baseline.

Besides the RTL implementation, we also build a cycle-accurate NoC processor simulator using Synopsys® Processor Designer to perform simulation of at most 256 cores. Experiments of single DMA transfer, synthetic traffic workloads, and parallel applications are run in the simulator to evaluate the performance of HyDMA.

4.1 Single DMA transfer

First we evaluate the transmission latency of a single DMA transfer between two cores located at two diagonal corners of a many-core processor integrating 4 × 4 or 16 × 16 cores on its 2D-mesh on-chip network. Data size of the DMA transfer ranges from 16 to 256 words, and no other traffic is involved in the network. As described in [4], at a clock frequency of 1 GHz in 45 nm technology, the maximum number of hops that can be traversed in a cycle is 11 for a data path with one crossbar at every hop. Thus Cnet messages should be latched at intermediate crossbars in the 16 × 16-sized NoC, which results in a delay of two cycles per dimension for HyDMA. Fig. 6 compares the cycles consumed to send or fetch data streams using HyDMA and other DMA approaches based on various NoCs,
including a conventional PS-NoC, a PS-NoC with two pairs of inter-router links (PS-NoC-2X) to provide better performance than BiNoC [8], and a CS-NoC with runtime configuration in a PS sub-network [13]. Considering the effect of routers’ processing latency on the performance of DMA transfers, we also add HyDMA-SC and PS-NoC-SC using single-cycle routers to the performance comparison. Besides HyDMA, other approaches also adopt the deterministic XY routing in PS-NoCs for the fairness of comparison.

The diagonal distance in the 4×4 NoC is six hops, while it increases to 30 hops for the 16×16 NoC, where it takes longer time to communicate between source and destination cores. As the size of total sent or fetched data within a single transfer doubles, it needs less time to transmit a word on average because the gradually increased total latency is split by more data words. To send data from the source core’s local memory to the destination core, HyDMA consumes 10.6% and 18.8% fewer cycles of total transmission time than CS-NoC in 4×4 and 16×16 NoCs respectively, since the Cnet channels built by HyDMA can already be used before the source core receives a lock message. With double bandwidth for inter-core communication, PS-NoC-2X saves 12.6% of total cycles in the sending cases compared to PS-NoC. While apparently they both require over 20% more time than HyDMA or CS-NoC due to the processing delay of Pnet flits in pipelined routers. Leveraging Cnet channels, HyDMA-SC still outperforms PS-NoC-SC by 4.4% when using single-cycle routers.

In fetching cases running on the two 2D-mesh NoCs with 4×4 and 16×16 cores, CS-NoC consumes only 2.0% more cycles than HyDMA. The Cnet channel for returned data is built after DMA requests reach the destination core, which is later than the concurrent establishing for both forward and backward channels in CS-NoC. In our baseline processor, to ensure data correctness during DMA transfers, a fetch request can be issued from a core only when returned data of previous
requests have been successfully received. Therefore, PS-NoC-2X and PS-NoC both consume plenty of time waiting for returned data during DMA fetching transfers, regardless of their different on-chip bandwidths. Without the support of Cnet channels to transmit fetching requests or returned data with low latency, PS-NoC-SC has much worse performance than HyDMA-SC and HyDMA.

4.2 Synthetic traffic workloads

Second, we simulate inter-core DMA transfers of sending or fetching data in a 4 × 4-sized mesh NoC under three types of synthetic traffic, including transpose, random and hotspot. According to transpose traffic, cores are grouped into pairs and core (i,j) always communicates with core (j,i). Under random traffic, destination cores of concurrent DMA transfers are randomly selected, and thus one core may receive multiple data requests simultaneously. In hotspot traffic cases, the most centric core is used to collect data from or distribute data to other on-chip cores, which leads to the most serious network congestion among the three workloads. The performance results of synthetic traffic workloads adopting various DMA approaches are shown in Fig. 7. The data size of DMA transfers still ranges from 16 to 256 words.

Fig. 7. Performance of synthetic traffic workloads

The reduction in total time costs of HyDMA over CS-NoC is 57.7%/43.0%, 44.0%/32.1%, and 54.1%/21.2% for sending/fetching data in transpose, random, and hotspot traffic workloads respectively, which is more significant than it for the single transfer cases. In CS-NoC, a DMA transfer cannot be initiated until a complete circuit channel has been successfully allocated, which results in considerable time overhead to perform concurrent DMA transfers with overlapping routes. However, by realizing low-latency transmission along circuit channels,
CS-NoC provides better performance even than PS-NoC-2X as data size increases. The twofold communication bandwidth provides PS-NoC-2X with a performance improvement of 10%∼30% compared to PS-NoC in sending cases, while the improvement is decreased to less than 9% for fetching cases due to the one-by-one fashion of issuing DMA requests. Integrating one-cycle routers in the Pnet, HyDMA-SC can also benefit from dynamically allocated Cnet channels. The total time reduction of HyDMA-SC over PS-NoC-SC is 34.4%/31.0%, 30.2%/40.0% and 25.5%/15.0% for sending/fetching cases of transpose, random and hotspot traffic workloads respectively.

4.3 Parallel applications

We also evaluate the performance of parallel applications running on a 4 × 4-sized NoC processor using HyDMA or other DMA approaches for data movements between all on-chip cores. The selected applications include a 256 × 256-point 2D-FFT, a 256 × 256-point 2D-convolution with a 9 × 9-point convolution kernel (2D-Conv), and a multiplication of two 256 × 256-sized matrices (Matrix Mult).

Fig. 8(a) demonstrates the 2D-FFT task flow adopting multi-core parallelization, and for clarity only three cores are depicted in the figure. Raw data stored in a master core are first distributed to slave cores to concurrently perform 1D-FFTs by row, and then FFT results are collected by the master core. Since the processing time of 1D-FFT may vary among participating cores, barrier synchronization [17] is performed to ensure the integrity of the collected results before the master core distributes them again for column FFTs. The 2D-FFT is completed after all cores have passed through a second barrier for gathering column FFT results. In HyDMA, non-DMA requests such as barrier arrival and release messages are also transmitted in the Cnet with explicit configuration on crossbar connections.

![Fig. 8. Parallel applications](image-url)

Fig. 8(b) presents the performance improvements of HyDMA, PS-NoC-2X and CS-NoC over PS-NoC for the three applications, and cycles consumed by each approach are also shown in the figure. With over 85% data of each DMA transfer transmitted in the Cnet, HyDMA achieves speedups of 1.38x, 1.25x and 1.44x for 2D-FFT, 2D-Conv and Matrix Mult respectively, which is higher than those of PS-NoC-2X (1.13x/1.17x/1.18x) and CS-NoC (1.08x/1.15x/1.13x). For 2D-
Conv, computing within each core occupies most execution time of the application due to its comparatively lower traffics of inter-core communication. Therefore, the performance gap between HyDMA and its competitors, which is 6% for PS-NoC-2X and 8% for CS-NoC, is not as significant as it for the other two applications.

5 Conclusion

This paper proposes HyDMA, a low-latency inter-core DMA approach based on a hybrid PS-CS NoC for many-core processors. According to the type of DMA requests transmitted in the two sub-networks, the dynamic setup and lengthening mechanism of Cnet channels for DMA transfers solves the main drawbacks of CS, including long latency of circuit setup and blocking of other network traffic. Moreover, bidirectional links in the Cnet allow HyDMA to perform concurrent DMA transfers with overlapping routes. Evaluation results of single DMA transfer, synthetic traffic workloads and parallel applications using various DMA approaches are gained from a cycle-accurate NoC simulator. The time consumption of inter-core data movements is greatly reduced by HyDMA at the cost of insignificant hardware overhead under a 40 nm technology.

Acknowledgments

This research is partially supported by the Equipment Pre-Research Foundation of China (No. 9140A08010414JW03025).