Journal of Systems Architecture 61 (2015) 423–434

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

MultiCS: Circuit switched NoC with multiple sub-networks and sub-channels

Shaoteng Liu [a,*], Axel Jantsch [b], Zhonghai Lu [a]

[a] KTH Royal Institute of Technology, Sweden
[b] TU Wien, Vienna, Austria

ARTICLE INFO

Article history:
Received 26 August 2014
Received in revised form 9 June 2015
Accepted 27 July 2015
Available online 13 August 2015

Keywords:
NoC
Circuit switched
Multi-channel
SDM

ABSTRACT

We propose a multi-channel and multi-network circuit switched NoC (MultiCS) with a probe searching setup method to explore different channel partitioning and configuration policies. Our design has a variable number of channels which can be configured either as sub-channels (spatial division multiplexing channels) or sub-networks. Packets can be delivered on an established connection with one or multiple channels. An adaptive channel allocation scheme, which determines a connection width according to the dynamic use of channels, can greatly reduce the delay, compared to a deterministic allocation scheme. However, the latter can offer exact connection width as requested. The benefits and burden of using different number of channels and configurations are studied by analysis and experiments. Our experimental results show that sub-network configurations are superior to sub-channel configurations in delay and throughput, when working at the highest clock frequency of each configuration. Under reasonable channel partitioning, sub-networks with narrow channels can generally achieve higher throughput than the network using single wide channels.

1. Introduction

Compared to packet switched (PS) NoCs with TDM channels, circuit switched (CS) NoCs are preferable under certain requirements. For example, the advantages of using CS NoC with spatial division multiplexing (SDM) channels on streaming applications like 3D graphics have been demonstrated in [1].

With the increasing number of wires available on-chip, the number of possibilities to organize, use and allocate them grows combinatorially. We have more freedom to choose the number of channels and allocate wires for them, instead of giving all wires to only one channel. Thus, how to allocate wire resources and organize multiple channels becomes an interesting research question.

We propose a CS network with multiple channels and multiple networks (MultiCS) to explore the effects of different channel partitioning and configuration polices. We offer the following contributions:

- The proposed CS NoC combines spatial division multiplexing [1], which we call sub-channels, with sub-networks. Sub-channels divide the wires between two switches which then can be allocated separately and independently. Sub-networks are independent networks that connect to the same nodes. A connection between two nodes can utilize one or several sub-channels and one or several sub-networks (Section 3).
- We extend the parallel probing setup method described in [2] to allow a single communication to utilize several sub-networks or sub-channels to meet its bandwidth requirements. This setup method uses a minimal number of extra wires. If a connection, possibly spanning across multiple sub-channels and sub-networks, with sufficient bandwidth is available, it will be found in $3 * D + 4$ cycles, where D is the distance between source and destination (Section 3).
- We propose two schemes for building connections with multiple channels. One is called deterministic channel allocation (DCA), which imposes an exact width requirement on a connection. The other is called adaptive channel allocation (ACA), with which the width of a connection is allocated according to run-time channel usage of the network. ACA avoids the channel fragments and superfluous connections problems associated with DCA. However, it guarantees only a minimum connection width (Section 4).

* Corresponding author. Tel.: +46 722935320.
E-mail addresses: liu2@kth.se (S. Liu), axel.jantsch@tuwien.ac.at (A. Jantsch), zhonghai@kth.se (Z. Lu).

http://dx.doi.org/10.1016/j.sysarc.2015.07.013
1383-7621/

- We discuss the implementation cost of MultiCS. We also develop an analytical performance model to explain how the maximum throughput is related to packet size and the number of channels (Section 5).
- We evaluate five configurations of MultiCS under the two schemes. The five configurations vary the number of sub-networks and sub-channels. The experiment results comply with our previous model and analysis (Section 6).

2. Motivation

CS NoC has a fundamental limitation. When a channel between two switches is allocated to one connection, it cannot be used by any other connection. This inherent inflexibility limits the usefulness of CS. A solution is to partition the channel into multiple narrow channels and allocate only one or a few narrow channels to a given connection. An obvious question is, what would be the trade-off of number of narrow channels: as many as possible, or is there any limitation?

The second question is: what are pros and cons of different ways of organizing multiple channels? Generally speaking, there are two methods to configure channels of a node: by sub-channels or by sub-networks, as illustrated in Fig. 1. The term sub-network refers to several separate circuit switched networks working in parallel. A network interface has access to all available sub-networks, but once data has entered one particular sub-network, it cannot change to another sub-network. In contrast, sub-channels are parallel links between switches. Those sub-channels are organized as SDM inside a switch. Data can use different sub-channels for different hops in the network.

The third question is how to establish a connection with multiple channels. Traditionally, even if there are multiple channels, still each packet flow is delivered by building a connection with only one channel [3,4] per hop. However, multiple channels have offered us the possibility of building a wide connection by combining several channels together for data transfer. Thus, we need new schemes and guidelines for connection set-up.

In the following we answer these three questions by comparing alternatives with the same total wire resources and with the same path search and setup algorithm. We construct our platform by combining features and merits from earlier work [2,5,6], and modify them to support multiple sub-channels and multiple sub-networks. We use a mesh topology but arguable many of our main conclusions are also valid for other topologies.

3. Architecture of MultiCS using parallel probing setup

Our MultiCS NoC can have any number of sub-channels, sub-networks, or a combination of them. The term "sub" denotes the number of sub-networks used, and the term "ch" denotes the number of sub-channels used. Suppose k is the number of sub-channels of a switch and m is number of sub-networks (*subm_chk*), and given that the total number of wires of a switch is a constant.

3.1. Overview of a switch

As shown in Fig. 2, in a mesh topology every switch has five directions which are used for connecting to four neighbors and one local resource. Each direction may have multiple duplex channels. Each channel contains a data path, which is used for carrying the probe during the setup phase and for transmitting data when a connection has been established. Every data path is also associated with 3 bits control signals: an answer (ANS) signal consisting of 2 bits, which goes in the opposite direction to the data channel, and 1 bit for a request signal, which travels in the same direction as the data channel. When the request signal is '1', a probe search is running or data transfer is active. When request signal is '0', it denotes the idle state, and an established connection will be released. The usage of the ANS signal is listed in Table 1.

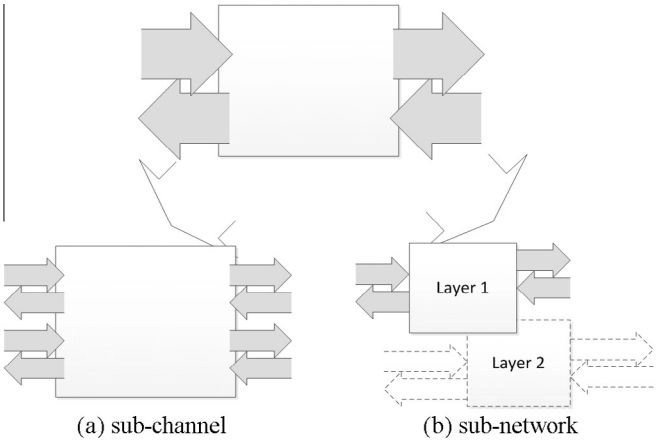


Fig. 1. Splitting wide channel into sub-channels or sub-networks.

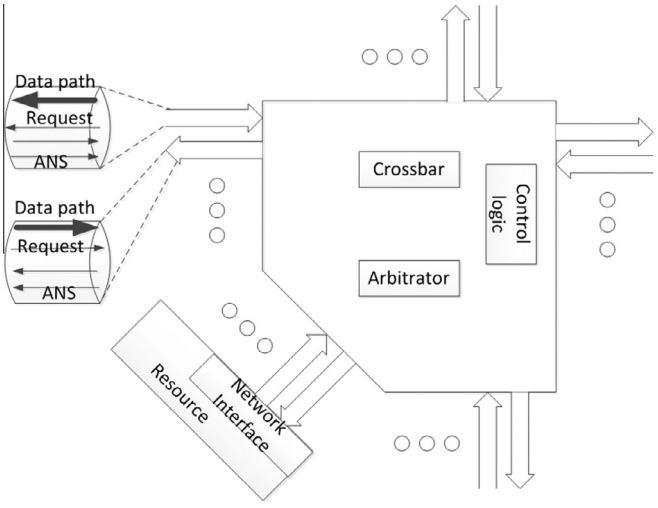


Fig. 2. Overview of a switch.

The messages required for connection configuration are simplified by using these control signals: tear-down message is carried by the request signal, and Ack/Nack messages are delivered by ANS signal. They are free of contention.

In this switch architecture, the overhead of each channel is just the 3-bit control signals. We believed this is the minimum requirement for supporting probing based setup. Although it may be argued that the 1 bit request signal can be omitted, this would increase the logic inside a switch.

The probe format is also compact. It contains source address, destination address and the channel id of the network interface of the source (Table 2). In an 8×8 mesh with 4 sub-channels/sub-networks, the minimum width of a probe is 14 bits. Each probe is one flit in length. Thus the width of a data path is ranged from 14 bits to any bits.

3.2. Path searching algorithm

Generally speaking, our platform is not bound to a particular path searching algorithm, meaning that any algorithm can be

Table 1
Usage of ANS signal.

ANS	Usage
00	Idle
01	Probe search continue
10	Probe failed
11	Path established

applied. The performance comparison results of different path searching algorithms are shown in Section 6.2.1.

The parallel probing algorithm [2] is chosen as our default path searching algorithm in this paper because of its high performance.

Parallel probing is an adaptive path searching algorithm. The fundamental idea was proposed by us in [2]. We illustrate this idea in Fig. 3. Suppose node 1 want to set-up a path to the destination node 9. In the beginning, a setup probe carrying information such as source address, destinations address and channel id is generated by the network interface and enters into node 1. Then, node 1 sends out two probes to the neighboring nodes 2 and 4. In the second hop each probe splits into two, and all the probes continue to move towards the destination along all the possible minimum paths.

Whenever two probes containing the same setup request meet, one of them is regarded as redundant and is canceled, and all channels used only by the canceled probe are released. For example, when node 5 receives two probes which are the same, one of them is canceled and all the channels it has booked before are released, (we have marked the released channel with a cross marker) as shown in Fig. 3b). Note that, the channel between node 1 and node 2 is not released, because it is still needed for the probe that has traveled further to node 3.

The release process proceeds backwards hop by hop. The switch does a release based on the registered connectivity information that connects an input port to an output port. When a release signal (ANS = 10, "probe failed") arrives at an output port from a downstream switch, the corresponding input port is looked up, the connection is canceled, and the release signal is forwarded upstream to the input port. Applying this mechanism, if several possible paths exist, one and only one of them can be finally booked, just as desired.

In this way, a wavefront of probes travel through the network and reach the destination on a minimal path. The time is exactly 2D, where D is the distance in terms of hops, and it takes 2 cycles to traverse each hop. When a probe successfully reaches the destination, an acknowledgment is sent back to the source node, which takes 1 cycle per hop.

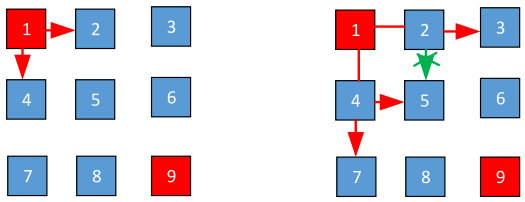
However, in our previous work [2], this algorithm is designed for circuit switched NoC with one wide link per direction. It applied a complicated priority based arbitration mechanism which is not applicable for multiple sub-channel usage. It requires a sorting component when applied in a multi-sub-channel switch, which is too costly to implement in hardware. Instead we use a round-robin based allocation. The set-up clock frequency is increased from 570 MHz [2] to 1.1 GHz (using SMIC 90 nm Tech).

3.3. Operation flow

Our circuit switched network has six operations which are explained in Fig. 4. In stage 1, a probe carrying setup information is sent out from source node and moving towards the destination

Table 2
Probe format.

6-Bit src. addr	6-Bit dest. addr	2-Bit channel id



(a) In each node a probe may double. (b) When two probes meet, one is cancelled.

Fig. 3. The overview of the parallel probing algorithm.

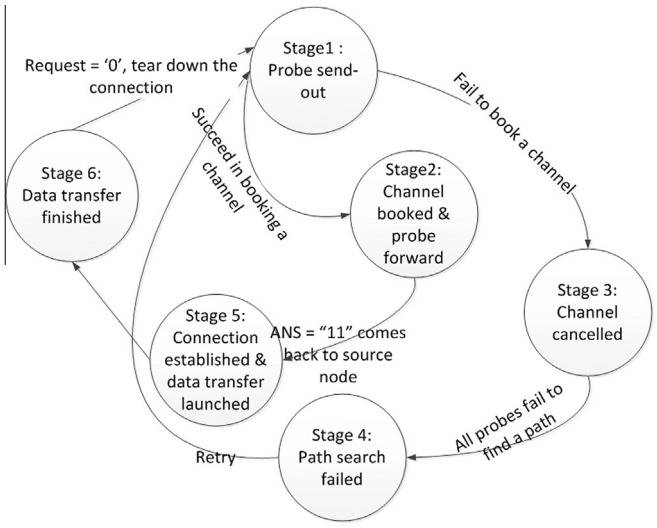


Fig. 4. Operation flow of parallel probing method.

according to the following algorithm. At each hop, if there are free channels, the probe will book one channel and move forward (stage 2). Otherwise, the probe is failed and it will use ANS signal (Nack) to clear all the channels it has already been booked (stage 3). When the source node gets notice that all the sent out probes have failed (stage 4), it will retry again. If a probe successfully reaches its destination, the Ack signal will be sent back, which means the connection has been established and data transfer can be launched (stage 5). After data transfer finished (stage 6), connection will be torn-down. It will then go back to stage 1 to wait and serve new setup request.

3.4. Detailed switch architecture

The internal structure of a switch is shown in Fig. 5. It is divided into two parts: control path and data path. The data path transfers data through the configured data crossbar. The control path is used to set up or tear-down a data path. The control path and data path share the same input and output wires.

We designed an allocator to do channel allocation for the probes arriving simultaneously at a switch. The principle of our single cycle maximal allocator with round-robin fairness is similar to a wave-front allocator [7], but it is smaller, fairer, faster and free of combinational loops. Detailed description of this allocator is described in our work [8].

It should be noted that the area of this allocator increase with $O(n^2)$, the critical path length scales $O(n)$, with n being the number of channels per direction. For example, in a 1-channel-per-direction switch, the allocator in charge of each output direction consists of only 4 tiles, while in a

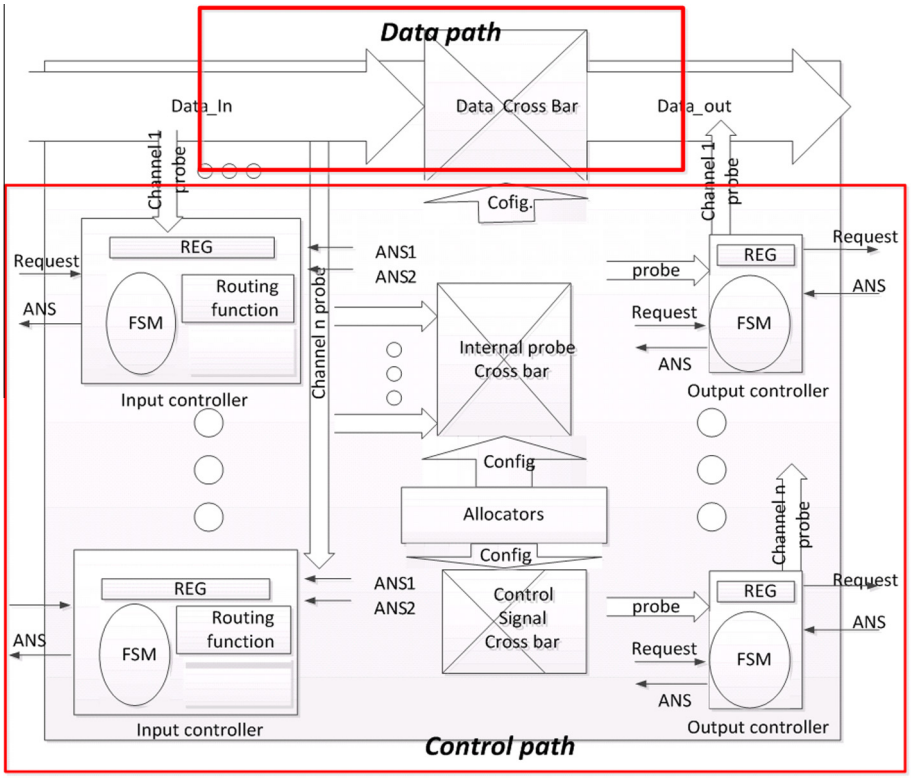


Fig. 5. Internal structure of a switch.

4-channel-per-direction switch, the allocator per direction contains 64 tiles.

3.4.1. Source synchronized data transfer

The control path and data path can work at different clock frequencies and share the same wires without interference. This clock scheme takes advantage of the property of circuit switched NoC that the setup phase never overlaps with the data transfer phase. During the connection setup phase, the data path should have no active clock signal, thus it is idle. And the cross-bar of the data path can be configured under the control path clock (probe clock). During the data transfer phase, the control path ignores data variations on the shared wire links. It just listens to the request and ANS signals.

Therefore, we can utilize either source synchronous data transfer [9–11] or clock gating to realize this separation of data and control path clock schemes, so that the data transfer can benefit from a higher clock frequency. In this paper, we chose the former source synchronous data transfer. The usage of this technique on CS NoC has been justified by Pham et al. [11,6].

3.4.2. Predictable delay

One of the benefits of the probing set-up approach is predictable latency. In our design, each probe takes 2 clock cycles per hop, and the ANS signal takes 1 cycle per hop. So, it takes at most $3 * D + 4$ cycles for a probe to travel from source to destination and back the ANS signal (D is the hop distance between source and destination). 4 cycles is the overhead consumed in the source and destination nodes. Therefore, in an $n * n$ mesh the worst case for a single search takes $3 * (2 * n - 2) + 6$ cycles, no matter if the result is a success or a failure. There is no such bound for the packet configuration approach such as [12]. It is reported that it takes 76 cycles on average for 6 hop distances [12], while it is fixed to 22 cycles by using our probing approach.

For data transfer, the head flit takes two cycles per hop, and the following flits are pipelined.

3.4.3. Configurable sub-channels and sub-networks

Even though the total number of wires between switches is the same in different configurations, their costs and performances are different. Fig. 6 depicts configurations sub1_ch2 and sub2_ch1. Their intra-switch connection relationship of channels is unveiled by using a switch block diagram, in which a line denotes a bi-directional connection between two duplex channels. For example, in the multi-sub-channel sub1_ch2 case, an output channel can be connected to all input channels except to the ones of the same direction. Since each output channel needs to select from 8 input channels, which means an 8-to-1 multiplexer is required by each output channel, and thus the switching logic in total has ten 8-to-1 multiplexers. However, in the multi-sub-network sub2_ch1, a channel is restricted to connect to the channels of the same sub-network. Thus, each output channel just needs a 4-to-1 multiplexer and the entire switching logic is just made up of ten 4-to-1 multiplexer. As a result, for a given number of wires, although sub-channel configuration offers more switching flexibility [13] than sub-network, its switching logic is much more complicated.

ler até aqui

4. Connection building schemes in MultiCS

Since a network interface has access to all available sub-networks and sub-channels (Fig. 6), we have the flexibility to build a wide connection with one or more channels per hop to deliver a packet, rather than only one-channel-per-hop. For example, in a sub2_ch2 network, each resource node is connected to 4 input–output channel pairs, so at most it can use all its 4 output channels for one connection. In order to set up a connection with 4-channel width, the resource node has to send 4 setup probes

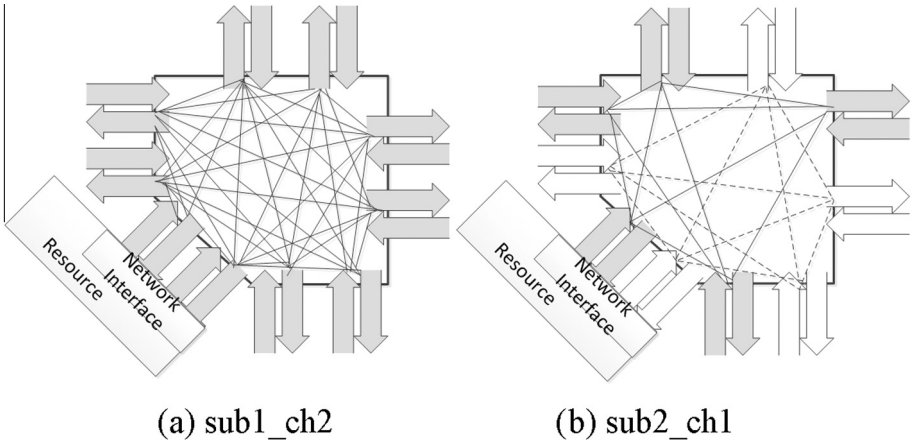


Fig. 6. Switch block diagram of sub1_ch2 and sub2_ch1.

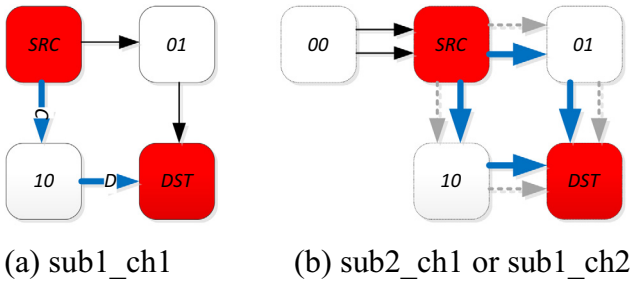


Fig. 7. Illustration for channel fragments.

out through its 4 output channels. Each probe will try to set up a narrow connection of 1-channel width to the destination. After the success of all 4 probes, a 4-channel connection is established.

We propose two schemes to explore the opportunities and challenges of building connections with multiple channels. We name the two schemes as deterministic channel allocation (DCA) and adaptive channel allocation (ACA), respectively.

- *DCA:* DCA imposes mandatory requirement on the connection width. For example, if a packet has a connection width requirement of 4 bytes, it is restricted to use two 2-bytes channels per hop, or four 1-byte channels per hop to build up a connection; any allocation below or above this figure is unacceptable.
- *ACA:* ACA scheme has no hard connection width requirement. During the setup phase, a setup request tries to utilize as many channels as possible to build a connection. However, the final connection width is determined by the number of successful setup probes, which depends on the run-time congestion situation of the network.

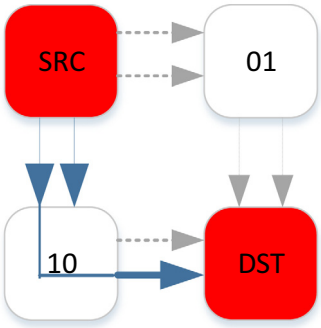


Fig. 8. Illustration for a superfluous connection.

The DCA scheme is intended to provide desired and predefined throughput and flit width for data transfer. It imposes strict requirement on the setup phase. DCA scheme is useful in the circumstances when an end-to-end transfer has exact predefined throughput or flit width requirement.

The ACA scheme is designed to achieve better performance, at the expense of only minimum throughput and flit width (1-channel width) guarantee. Depending on channel use, ACA adaptively sets up a connection, of which the width of a connection can vary from 1 to k channel-width, where k is the total number of output channels per direction. As a result, at low load connections are likely to be wide and thus data transfer time can be shortened; at high load connection width tends to be narrower because of contention. Thus, more requests can be served and high throughput can be reached.

The traditional one-channel-per-connection (OCPC) scheme is a special case of DCA scheme (DCA with 1-channel width requirement). In OCPC scheme, the width of every connection is restricted to 1-channel width.

The implementation of DCA may introduce additional steps during setup. For example, if a packet in a sub4_ch1 or sub1_ch4

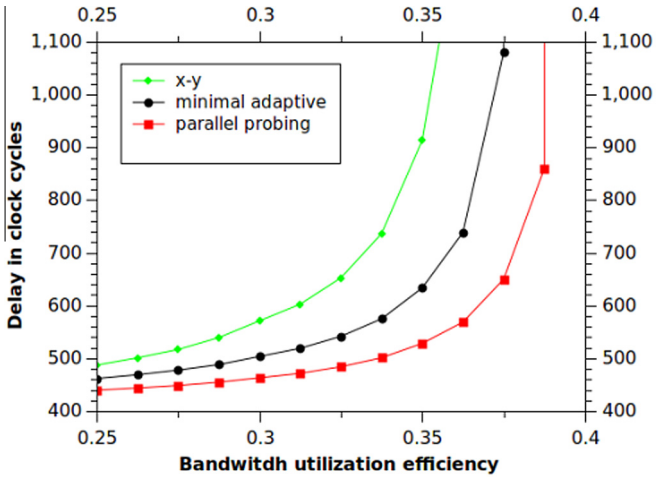


Fig. 9. Evaluation of path searching algorithm by using sub1_ch4 in ACA-FPS (packet size 5120 bytes).

configuration has a DCA requirement of 8-byte width, and each channel is 2-byte wide, the resource node has to wait until four 2-byte output channels of the network interface are available, and then four probes are sent out simultaneously. Only if all of the probes succeed, the data transfer can commence. Otherwise, the one-channel connections (*superfluous connections*) established will be released and all four probes will be re-sent again until all four succeed. The release of superfluous connections is adopted to avoid deadlock.

The implementation of ACA takes advantage of the predictable set-up delay of MultiCS. For each setup request, a resource sends out probes through all free output channels of the network interface, and each probe tries to build up a one-channel connection. After one probe succeeds, the sender will wait a short period ($<(3*D+4)$ cycles) for all outstanding probes to return their results, then combine all the established one-channel connections together to form a wide connection for transfer.

5. Cost and performance analysis

5.1. Implementation cost

Suppose k is the number of sub-channels of a switch and m is number of sub-networks (*subm_chk*), and given that the total number of wires of a switch is a constant.

The critical latency of the data path of a switch is chiefly decided by the crossbar latency, which scales with $O(\log k)$, and it is independent of m.

The area of a data cross-bar scales $O(k^2)$ with sub-channels, and it is again independent of m. The registers inside the data path take a large part of area, but their number is independent of k and m.

To the latency of the control path, the allocator contributes $O(k)$ latency and the cross-bar contributes $O(\log k)$. Combining both we see that latency scales $O(k)$ with sub-channels. Again, the latency is independent of the number of sub-networks m.

The area of control path scales $O(m)$ with sub-networks and $O(k^2)$ with sub-channels. This is because using sub-networks causes a linear increase of certain components, e.g. FSMs. However, using sub-channels will cause certain components, e.g. allocator, have a k^2 increase.

The synthesis results of a few configurations are listed in Table 3 with SMIC 90 nm library. The number of wires per-direction of each configuration is the same, i.e. 8 bytes. The total power and area per node is reported by Design Compiler and calculated at each one's maximum clock frequency.

Generally speaking, synthesis results are in accordance with our expectation. For example, sub1_ch1 has the smallest area and power consumption, and can work at the fastest clock frequency. Sub4_ch1 has the same frequency as sub1_ch1, while sub1_ch4 consumes the largest area because it has an $O(k^2)$ increase in area, and works at the slowest clock frequency due to its $O(k)$ latency scale factor.

5.2. An analytical performance model

We propose a model for per-node maximum throughput analysis. We assume that every node inside a network has the same behavior and the network achieves the maximum throughput when a node is always busy in requesting connection setup or transferring data. This means that there is no idle time.

Suppose t_0 is the time used for data transfer when a connection has been established, t_1 is the time consumed for a single search (it has a bounded value in our approach), α is the failure rate ($1-\alpha$ is the success rate). Suppose further that the intensity (average number of certain events per time unit) of successful transfers of a node is $\lambda(A)$, the intensity of a single search is $\lambda(B)$. Based on the conclusions of Palm Calculus [14], we have

$$\frac{\lambda(A)}{\lambda(B)} = 1-\alpha \tag{1}$$

The average time between two single searches is $1/\lambda(B)$, which is equal to

$$\frac{1}{\lambda(B)} = (1-\alpha)(t_0+t_1)+\alpha t_1 \tag{2}$$

As there is always either a search or a data transfer going on, $\lambda(A)t_0+\lambda(B)t_1=1$.

The average time between two successful searches is $1/\lambda(A)$, and, combining (1) and (2), we obtain

$$\frac{1}{\lambda(A)} = t_0+\frac{t_1}{1-\alpha} \tag{3}$$

According to (3), maximum normalized throughput (bandwidth utilization rate) is

$$THN = \frac{t_0}{t_0+\frac{t_1}{1-\alpha}} \tag{4}$$

Suppose B is the bandwidth of a resource node, then the maximum throughput of each resource is

$$TH = THN*B = \frac{Bt_0}{t_0+\frac{t_1}{1-\alpha}} \tag{5}$$

This simple model can explain the following intuitions:

I. As the packet size increases, t_0 goes up and thus TH goes up, i.e. CS NoC becomes more efficient with larger packets.
II. Assume a fixed packet size of M (bytes) and a total bandwidth B of a resource node. If each node just has one channel, then the required time for data transfer is $t_0 = M/B$, and from (4) we obtain the normalized throughput THN. However, if we allocate the total bandwidth into two channels, with each one $B/2$ the bandwidth, then the data transfer time for a packet become $t_0' = 2t_0$, and the normalized throughput becomes

$$THN' = \lambda(A)^* t_0' = \frac{2t_0}{2t_0+\frac{t_1}{1-\alpha}} \tag{6}$$

Thus, splitting a wide channel into narrow channels increases the maximum throughput. This conclusion is less expected but can intuitively be explained by the fact, that if the channel bandwidth is smaller, the penalty of not using this bandwidth during setup is also smaller.

This model is a simplification. For example, in reality, α in formula (4) is not a constant. It depends on a number of factors such as t_0, topology, sub-network/sub-channel configuration, connection build-up schemes and so forth. Although the model is fairly simple we will see in the subsequent simulation results, that the main conclusions are confirmed.

5.3. Analysis of connection building schemes

The behavior of ACA in general follows our previous intuitions, as we will see in Section 6, e.g. Fig. 10.

For DCA, however, contrary to our intuition II, under certain circumstances, the throughput of multiple narrower channels is inferior to a single wide channel. Two phenomena degrade the performance of DCA.

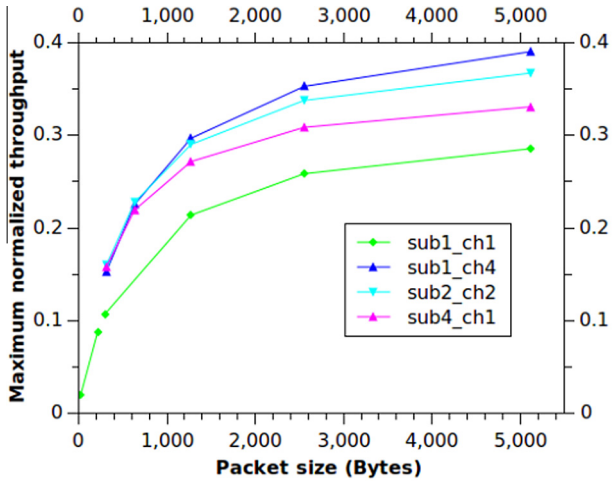


Fig. 10. Influences of packet size on maximum throughputs.

5.3.1. Channel fragments

As Fig. 7 illustrates, in multiple channel configurations such as sub2_ch1 or sub1_ch2, a DCA requirement demands a 2-channel connection and such a connection is possibly built by using channels from two distinct routes[1] (Fig. 7b)). This kind of channel allocation will generate channel fragments. Although there is a free channel between nodes SRC and 01, it cannot be utilized by node 00 to build a connection with 2-channel requirement to node 01. In comparison, single channel configuration sub1_ch1 has the double channel width of sub2_ch1 or sub1_ch2. Thus, it can build a one-channel connection to satisfy the same DCA requirement. There is no fragment in sub1_ch1 under such a DCA requirement. Channel fragments are the result of increased flexibility when given a higher number of narrower channels. But for certain traffic scenarios this increased flexibility is doing more harm than good, as we will see in Fig. 15.

5.3.2. Superfluous connections

Connections that are setup but cannot be used, are superfluous. Suppose we have a CS NoC as sub2_ch1, and each channel is 2-byte wide. For example, as Fig. 8 suggests, a connection with 4 bytes width requirement needs to send out 2 probes and set up two one-channel connections simultaneously with DCA scheme. However, since some of the channels have already been occupied, only one one-channel connection can be established. As a result, since the DCA requirement cannot be satisfied, data transfer cannot be launched and thus the established one-channel connection becomes superfluous. The superfluous connection will be released and then a new setup is attempted. However, the reserve and release of these superfluous connections inside a CS NoC puts a heavy burden on the network and degrades performance.

6. Experiments and evaluations

In this section, we will check whether experiment results are in accordance with our design goals, intuitions and analysis. All experiments are based on 8×8 mesh topology. Uniform random traffic with Poisson arrival time distribution is used for evaluation purpose.

In addition to the four configurations in Table. 3, configuration sub2_ch1 is also used in our experiments. This configuration has two sub-networks, each of which has a channel width of 4 bytes, so that the total channel width per-direction is also 8 bytes. The clock frequencies of sub2_ch1 are the same as sub4_ch1.

We use two scenarios which use ACA and DCA, respectively, to compare the performance of different number of sub-channels and sub-networks, as well as the path searching algorithms. In both scenarios we include several test cases, such as fixed packet size case (FPS) (all packets have the same size), variable packet size case (VPS) (all packets have random packet sizes). However, since FPS and VPS have similar results, we just show the results of FPS.

When nothing else is specified, we use the parallel probing algorithm by default.

6.1. Simulation method and metrics

Inside each resource node a generator generates setup requests according to a probability and pushes them into a queue. An FSM pops a request from the queue and sends it out when sufficient output channels are available. Then the FSM waits for a success or failure notification. Then it either retries the request or commences the data transfer.

We have implemented an HDL model for synthesis and for area and power evaluation. We have also built a cycle accurate SystemC simulator which can run 10–30 times faster than the HDL model. Any data point that is shown in the figures comes from simulation of 250 million cycles, of which the first 250 thousand cycles are discarded as warm up period.

Suppose β is the packet generation probability and M is the packet size (in bytes), and clk_{freq} is the data path clock frequency of a configuration, B is the bandwidth of a resource node, then the injection rate per node (IR) is defined as

$$IR = \beta * M * clk_{freq}$$

Besides throughput and delay, we use *bandwidth utilization efficiency* (Eb), also called *normalized throughput*, as one of the metrics. It is defined as

$$Eb = \frac{Throughput\ (per\ node)}{Bandwidth\ (per\ node)}$$

In our simulations each configuration operates at its maximum frequency.

6.2. ACA (adaptive channel allocation) evaluation

6.2.1. Evaluation of path searching algorithms

Three path searching algorithms are compared, which are x–y [15], minimal adaptive [15] and parallel probing [2]. The results in Fig. 9 suggest that parallel probing is the best path searching algorithm for ACA scheme. E.g. at offered load 0.35, the average packet delay of parallel probing is only 83% of minimal adaptive, and 57% of x–y algorithm. We also have evaluated algorithms in different channel number and configurations. Their results suggest the same ranking of algorithms. Consequently, we choose parallel probing as our default path searching algorithm.

6.2.2. Influences of packet size on maximum throughputs

The influence of packet size on maximum normalized throughput is shown in Fig. 10, which suggests that as packet size increases, the maximum normalized throughput for each configuration also goes up. This result complies with intuition 1 from our model. Thus, we may safely conclude that CS NoC is suitable for delivering large packets. This is the reason why throughout this paper we prefer large packets for evaluations. This conclusion implies that applications that generate large bulk of data for communication, like task allocation and migration on MPSoC, or page

[1] In this situation, each flit will be split into two phits at the source, with each route simultaneously delivering only one phit. At the receiver side, it will restore a flit by combining the two phit together.

Table 3
Per-node synthesis results of different CS NoCs with 8 bytes of wires per-direction.

Configuration	Sub1_ch4	Sub2_ch2	Sub4_ch1	Sub1_ch1
Channel width	2 Bytes	2 Bytes	2 Bytes	8 Bytes
Num. sub-network	1 (Sub1)	2 (Sub2)	4 (Sub4)	1 (Sub1)
Num. sub-channel	4 (Ch4)	2 (Ch2)	1 (Ch1)	1 (Ch1)
Max. probe freq. (MHz)	556	740	1111	1111
Max. data freq. (GHz)	1.116	1.397	1.786	1.786
Total area (um^2)	150214.5	86777.5	57599.5	30874.9
Probe path area	85791.8	53161.3	35632.8	8908.2
Data path area	64422.7	33616.2	21966.7	20133.7
Power@max. freqs. (mW)	50.1	45.0	49.5	26.8

based virtual memory management, benefit from CS NoCs, while applications with mostly short messages may prefer PS NoCs, as concluded in [16].

6.2.3. Evaluation of different number of channels

The experiment results of splitting a wide channel into narrow sub-channels is shown in Figs. 11 and 12. The packet size is fixed to 5120 bytes in this evaluation.

As shown in Fig. 12, sub4_ch1 provides higher throughput than sub2_ch1, which in turn is better than sub1_ch1. E.g. the maximum throughput of sub4_ch1 is about 17% higher than sub1_ch1. The increase in maximum throughput complies with our intuition II. We can imagine that some packets in the sub4_ch1 configuration use only 1, 2, or 3 of the subnetworks. However, e.g. using 1 subnetwork with ¼ the channel width compared to the sub1_ch1 configuration means that the packet consists of 4 × the number of flits. As we studied in the last section, larger packet sizes lead to higher maximum throughput in CS NoC. Thus, using narrow sub-links will achieve higher maximum throughput because the average packet size counted in flits is also larger.

Regarding delay, as Fig. 11 suggests, sub1_ch1 has better packet delay results only when the injection rate is low. This is due to the connection setup delay contributes little to the total packet delay because of low contention probability. In this situation, data transfer delay dominates the total packet delay. Sub1_ch1 has shorter data transfer delay due to its wider channel. However, at high injection rate, sub4_ch1 outperforms sub1_ch1. For example, at injection rate 3500 MB/s, the average packet delay of sub4_ch1 is 20% less than sub1_ch1.

If throughput is our main concern, the number of channels should be maximized. However, in our design, the minimum channel width is decided by probe format, which is about 14 bits. Narrower than this value complicate the probe delivering process.

6.2.4. Evaluation of different configurations (Fig. 13)

Although sub1_ch4 has more switching flexibility than sub4_ch1, this advantage is compensated by its slower clock frequency. As a result, sub-network configuration (sub4_ch1) outperforms sub-channel configurations (sub2_ch2 and sub1_ch4) in delay and throughput. Sub1_ch1 has lower maximum throughput than the other multi-channel configurations. However, it presents better latency at low load for the same reasons explained above.

Bandwidth utilization efficiency discounts the difference in frequency and gives a performance comparison under the assumption that the networks operate at the same frequency. Sub1_ch4 has the best bandwidth utilization efficiency under ACA scheme. For example, we observed 30% higher efficiency than sub1_ch1 in ACA scheme. Sub2_ch2 and sub4_ch1 fall in between.

Bandwidth utilization efficiency may also be useful because in certain situations the maximum clock frequency differences of configurations are not sharp. For example, as reported in [25], when implemented in FPGA, a CS NoC with SDM channels roughly has the same maximum clock frequency no mater if 1 or 4

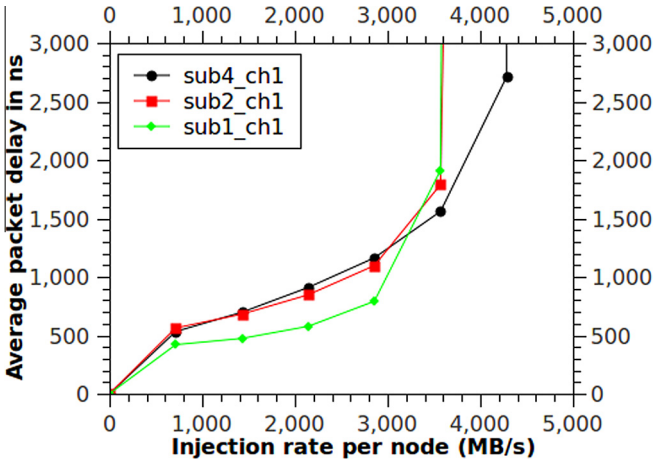


Fig. 11. Delay influence of dividing a wide channel into narrow sub-channels (packet size 5120 bytes).

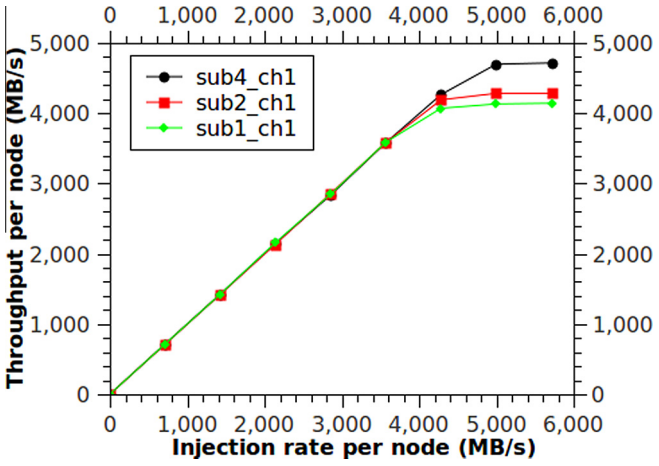


Fig. 12. Throughout influence of dividing a wide channel into narrow sub-channels (packet size 5120 bytes).

sub-channels are used. In situations like this, sub1_ch4 could offer better performance than other configurations.

We also tested under ACA scenarios with variable size of packets. The comparison among different configurations basically shows consistent results and is thus omitted here.

6.2.5. Comparison between ACA scheme and one-channel-per-connection (OCPC) scheme

OCPC is compared with ACA by using configuration sub4_ch1, as Fig. 14 suggests, at low load ACA offers much better average packet delay. E.g. at load 0.02, average packet delay with ACA is 170 probe clock cycles, while it is 490 cycles with OCPC. At very high load, OCPC represents slightly higher bandwidth utilization efficiency, and its maximum bandwidth utilization efficiency is 0.283, while for ACA it is 0.271.

The comparison result obeys our design goals of ACA in Section 4. At low load when data transfer delay dominates, ACA can significantly shorten the delay since the majority of packets are delivered by wide connections. At high load, due to contention, the probability of building a connection containing multiple channels is low and the majority of connections contain one channel only. Thus, the average packet length in flits by using ACA at high load is just slightly smaller than using one-channel-per-connection

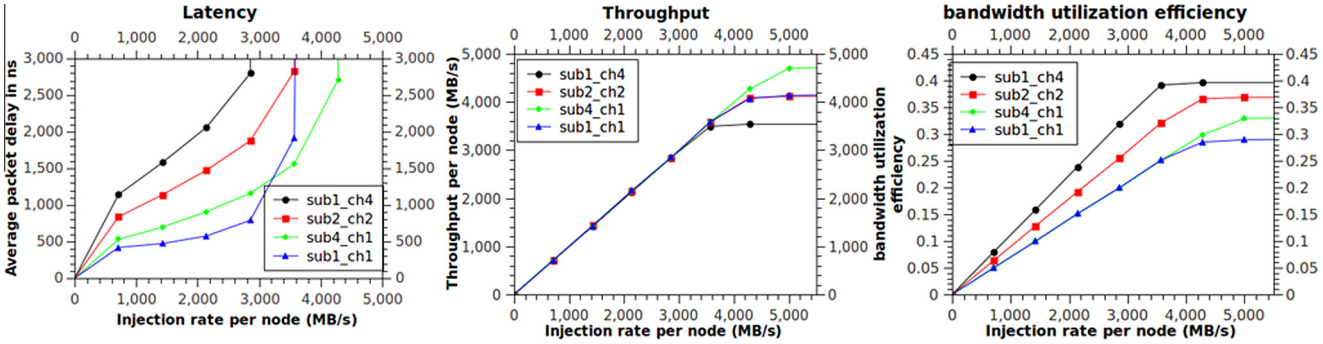


Fig. 13. Performance results of scenario ACA–FPS (packet size 5120 bytes).

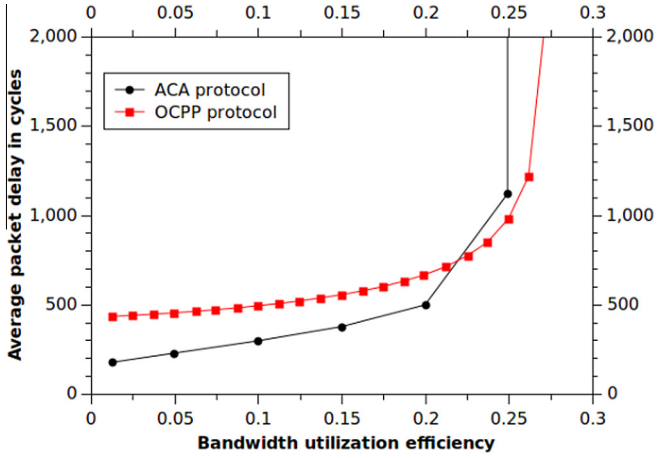


Fig. 14. Comparison between ACA scheme with traditional OCPC scheme (packet size 1280 bytes).

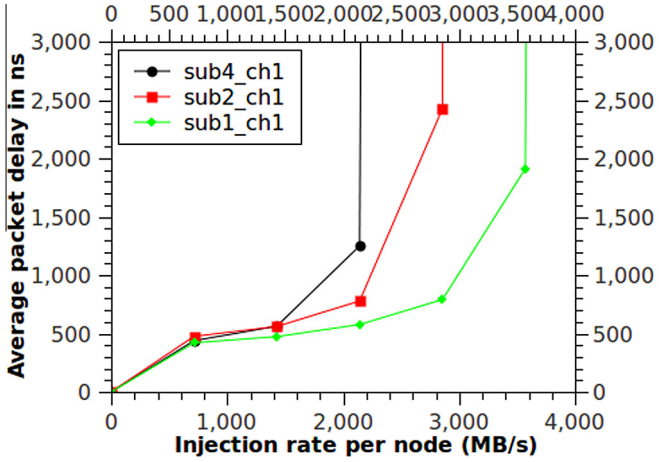


Fig. 15. Influence of dividing a wide channel into narrow sub-links for DCA transfer (packet size 5120 bytes).

scheme. As suggested by intuition II, the maximum bandwidth utilization of ACA drops slightly (smaller than 5% in this experiment).

6.3. DCA (deterministic channel allocation) evaluation

As mentioned before, ACA puts more emphasis on performance, while DCA focuses on desired throughput and flit width.

6.3.1. Evaluation of different number of channels

Previously, we showed that ACA benefits from the usage of multiple channels. However, the following example with DCA tells a different story.

Fig. 15 shows the result of a simulation with the exact connection width requirement of 8 bytes and a fixed packet size of 5120 bytes. In this case, as the latency curves suggest, more channels lead to higher delay.

This observation complies with our analysis in Section 5.3. Sub4_ch1 and sub2_ch1 perform worse than sub1_ch1, since sub1_ch1 generates neither channel fragments nor superfluous connections. Sub4_ch1 performs worse than sub2_ch1 because it generates more superfluous connections.

In addition, it is worth noting that, if less than half of the bandwidth can be utilized, splitting the wide channel into sub-links seems to be beneficial, even without special care on channel fragments and superfluous channels. In Fig. 16, the exact throughput requirement is 4 bytes/cycle. Because only half of the channel width in sub1_ch1 can be utilized for data transfer, sub1_ch1 is inferior to the other multi-channel configurations. This result also suggests that, according to the connection width requirement, proper channel partitioning could still be beneficial.

6.3.2. Evaluation of different configurations

For delay and throughput, both Figs. 16 and 17 demonstrate that sub-network configuration sub4_ch1 outperforms sub-channel configurations sub2_ch2 and sub1_ch4. These results are similar to those under ACA. However, in Fig. 17, the channel utilization efficiency of sub1_ch4 is worse than sub2_ch2, which is worse than sub4_ch1. This observation opposes the result with ACA and is not quite expected. It seems that switching flexibility becomes a handicap in this DCA case.

The reason for this phenomenon is that sub1_ch4 and sub2_ch2 are more likely to generate superfluous connections. Due to the increased switching flexibility, sub1_ch4 and sub2_ch2 have higher chances to set up a one-channel connection, which leads to a higher burden on the network due to set up and release of superfluous connections.

Generally speaking, as the comparison of Figs. 16 and 17 with Fig. 13 suggests, we may conclude that ACA offers better performance than DCA scheme. However, as mentioned before, DCA offers exactly predefined connection width and throughput.

We also tested DCA scenarios with variable size of packets. The comparison among different configurations basically shows consistent results and is thus omitted here.

7. Related work

The usage of sub-network and sub-channel in PS NoC has been studied during the past. For example, the cost and effect of intro-

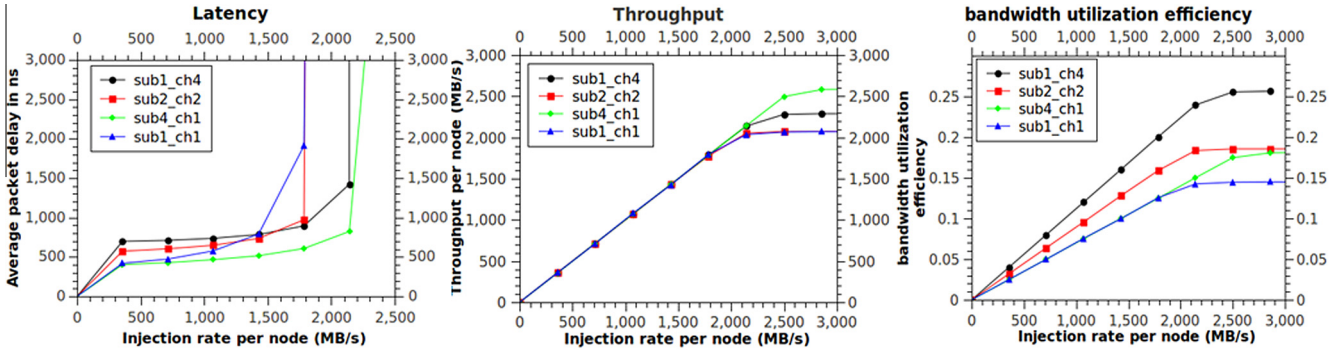


Fig. 16. Performance results of scenario DCA-FPS (packet size 2560 bytes, connection width requirement is 4 bytes).

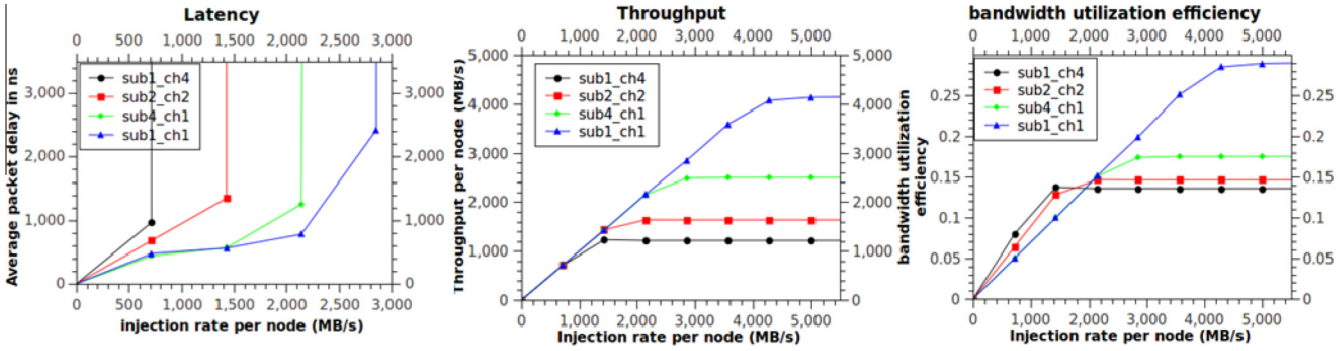


Fig. 17. Performance results of scenario DCA-FPS (packet size 5120 bytes, connection width requirement is 8 bytes).

ducing sub-networks into PS NoC has been studied by Yoon et al. in [17,18]. The pros and cons of using sub-channels in PS NoC has also been investigated in [3]. Besides, work [19,20] intend to increase the switching flexibility between virtual channels and the output ports of PS NoCs. In [19], several separate cross-bars are used inside one router, so that each virtual channel (VC) can choose between multiple crossbars to reach an output. In [20], a new switching layer is introduced at each input port, so that multiple VCs of an input port can be connected to different outputs at the same time.

However, compared with PS NoC, the usage of sub-network and sub-channel in CS NoC is not fully exploited and evaluated. Actually, in the past, the CS NoC architecture assumed in many papers (e.g. [2,11,21]) has just one duplex-channel between every two neighboring nodes. They did not consider the situation when a CS NoC can have multiple physical channels between two nodes.

Although some works [4,22] design CS NoCs with multiple channels and organize them in a sub-channel (SDM) way, the consequences of applying multiple channels in CS NoCs are still not well studied. For example, although [4,22] have multiple channels, packets are still delivered by following connections with only 1-channel width.

Another import aspect about CS NoC is connection setup, since a CS NoC requires a connection should be established before data transfer begins. According to the connection search and setup method, CS NoCs can be classified into two categories: dynamic setup or static setup. Static setup methods schedule connections at compilation time. As a result, they [23,24] may not well support applications like H.264 [25] with requirements for dynamic communication setups. Therefore, in this paper, we only focus on dynamic methods which search and setup connections at run time.

Dynamic methods can be further classified into centralized or distributed methods. Generally speaking, centralized set-up like [21,26] has two disadvantages. Firstly, the central schedule node needs to receive setup/release requests and distribute allocation decisions from/to the entire network. Such multiple-to-one and one-to-multiple traffic pattern is likely to become the system bottleneck which the number of nodes inside a NoC grows [27]. Secondly, since retrying of failed requests causes the blockage of the following requests, failed setup requests are usually dropped in centralized setup methods. Thus, we focus on decentralized setup.

Distributed setup can be implemented by sending configuration packets [4,28,29] or by a probing search approach [1,11,6,5].

Sending configuration packets requires a separate PS (packet switched) NoC to deliver configuration messages like set-up, tear-down and Ack/Nack during a connection setup procedure. In our view, this approach suffers from four major drawbacks. Firstly, using an additional PS NoC for connection set-up is an unnecessary overhead. Secondly, set-up, tear down and Ack/Nack packets of a connection must be routed by pre-determined routing algorithm to ensure them on the same connection. For example, [4,29] use deterministic routing algorithm, and in [28], source based routing information has to be carried by each configuration packet. However, such pre-determined routing algorithm is a sub-optimal choice among routing algorithms. Thirdly, compared with probing search, tear-down and Ack/Nack signals have to be sent in the form of packets. These packets will contend with set-up packets inside the PS NoC. There is typically no delay guarantee for configuration packets in the PS network, rendering the connection set-up procedure unpredictable. Fourthly, this approach does not scale well. The auxiliary PS NoC has fixed throughput, since each output port of a switch just allows to deliver one setup packet at a time. However, if there are many sub-channels in a CS NoC, and since each sub-channel requires a separate setup packet for connection configuration, this will significantly increase the number of setup packets as observed in [4].

Compared with above mentioned shortcomings of a packet configuration approach, probing search is the superior choice because of its efficiency in wire usage and connection setup procedure. The concept of the probing search was first proposed in [5]. Pham et al. [11,6] developed a backtracking path searching algorithm, which reportedly has better performance than [5]. Another contribution of [11,6] is that a source synchronized data transfer mechanism is introduced into CS NoCs, so that separate clocks can be applied to connection set-up and data transfer. [2] developed a parallel probing method for CS NoC. It can complete a search over all possible paths within $O(n)$ time complexity where n is the geometric distance between source and destination. They demonstrated superior performance of this parallel probing algorithm compared to Pham's backtracking algorithm [11] by experiments. But their channel allocation mechanism [2] is too complicated for multi-sub-channel usage.

The probing search approaches in all aforementioned works [1,11,6,5] are only implemented on CS NoC with a single channel between two neighboring nodes.

In this paper, we extend the parallel probing search method [2] to multiple sub-channels and sub-networks and study cost and performance of several configurations with sub-channels and sub-networks among 1 and 4.

8. Conclusion and future work

We have implemented MultiCS, a CS with multiple sub-channels and sub-networks with a parallel probing setup algorithm to study the consequences of splitting a wide channel into narrow channels. The design space of multi-channel CS NoC is explored from two angles: the channel number, and channel configurations. We have reached the following main conclusions:

A. Given a number of wire resources for each node inside a CS NoC, with ACA scheme, the thinner the channel width with more channels, the higher the throughput. However, the latency for data transfer also increases by using thinner channels. Sub-channels (SDM channels) consume much more resources than sub-networks. When splitting a wide channel into n narrow channels, organizing those channels into sub-networks gives an $O(n)$ increase in area, and the critical latency is unchanged. However, organizing those channels in sub-channels increases the area by $O(n^2)$, and the delay by $O(n)$. Furthermore, our experiments suggest that sub-networks offer better performance than sub-channels. Although sub-channels can achieve better channel efficiency due to higher switching flexibility, this is only useful in special situations. Thus, in general sub-networks are more efficient than sub-channels.

B. We can build a connection consisting of multiple channels with different schemes. The DCA offers desired and predefined throughput and flit width, but channel fragments and superfluous connections are two obstacles for DCA. Because of this, under certain width requirements, the performance of using multiple channels is even worse than using one single wide channel. ACA generally offers better performance than DCA. However, although ACA provides minimum connection width guarantee (one channel width), the actual width of a connection by applying ACA cannot be known beforehand. The connection width is decided by the success probability of setup probes, which depends on the dynamic channel use.

Our future work will study techniques to avoid channel fragments and superfluous connections, in depth evaluation of multi-channel CS NoC, and implementation and evaluation of mixed packet and circuit switched NoCs.

References

[1] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, D. Verkest, Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs, in: Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis, 2005, pp. 81–86.

[2] S. Liu, A. Jantsch, Z. Lu, Parallel probing: dynamic and constant time setup procedure in circuit switching NoC, in: proceedings of Design, Automation Test in Europe Conference Exhibition (DATE'12), 2012, pp. 1289–1294.

[3] C. Gomez, M. E. Gomez, P. Lopez, J. Duato, Exploiting wiring resources on interconnection network: increasing path diversity, in: Proceedings of Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'08), 2008, pp. 20–29.

[4] A.K. Lusala, J.-D. Legat, Combining SDM-based circuit switching with packet switching in a router for on-chip networks, Int. J. Reconfigurable Comput. 2012 (2012) 1–16.

[5] D. Wiklund, D. Liu, SoCBUS: switched network on chip for hard real time embedded systems, in: Proceedings of Parallel and Distributed Processing Symposium, 2003, p. 8.

[6] P.-H. Pham, P. Mau, J. Kim, C. Kim, An on-chip network fabric supporting coarse-grained processor array, IEEE Trans. Very Large Scale Integr. VLSI Syst. 21 (99) (2013) 178–182.

[7] D.U. Becker, W.J. Dally, Allocator implementations for network-on-chip routers, in: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, 2009, pp. 52:1–52:12.

[8] S. Liu, A. Jantsch, Z. Lu, A fair and maximal allocator for single-cycle on-chip homogeneous resource allocation, IEEE Trans. Very Large Scale Integr. VLSI Syst. 22 (10) (2014) 2229–2233.

[9] D. Walter, S. Hoppner, H. Eisenreich, G. Ellguth, S. Henker, S. Hanzsche, R. Schuffny, M. Winter, G. Fettweis, A source-synchronous 90 Gb/s capacitively driven serial on-chip link over 6 mm in 65 nm CMOS, in: proceedings of Solid-State Circuits Conference Digest of Technical Papers (ISSCC'12), 2012, pp. 180–182.

[10] D. Schinkel, E. Mensink, E. Klumperink, E. van Tuijl, B. Nauta, Low-power, high-speed transceivers for network-on-chip communication, IEEE Trans. Very Large Scale Integr. VLSI Syst. 17 (1) (2009) 12–21.

[11] P.-H. Pham, J. Park, P. Mau, C. Kim, Design and implementation of backtracking wave-pipeline switch to support guaranteed throughput in network-on-chip, IEEE Trans. Very Large Scale Integr. VLSI Syst. 20 (2) (2012) 270–283.

[12] A.K. Lusala, J.-D. Legat, A SDM-TDM-based circuit-switched router for on-chip networks, ACM Trans. Reconfigurable Technol. Syst. 5 (3) (2012) 15:1–15:22.

[13] J. Rose, S. Brown, Flexibility of interconnection structures for field-programmable gate arrays, IEEE J. Solid-State Circuits 26 (3) (1991) 277–282.

[14] J.Y. Le Boudec, Performance Evaluation of Computer and Communication Systems, Epfl Press, 2011.

[15] W.J. Dally, B. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann, 2003.

[16] S. Liu, A. Jantsch, Z. Lu, Analysis and evaluation of circuit switched NoC and packet switched NoC, in: Proceedings of Euromicro Conference on Digital System Design (DSD'13), 2013, pp. 21–28.

[17] Y. J. Yoon, N. Concer, M. Petracca, L. Carloni, Virtual channels vs. multiple physical networks: a comparative analysis, in: Proceedings of IEEE Design Automation Conference (DAC'10), 2010, pp. 162–165.

[18] Y.J. Yoon, N. Concer, M. Petracca, L.P. Carloni, Virtual channels and multiple physical networks: two alternatives to improve NoC performance, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 32 (12) (2013) 1906–1919.

[19] S. Noh, V.-D. Ngo, H. Jao, H.-W. Choi, Multiplane virtual channel router for network-on-chip design, in: Proceedings of First International Conference on Communications and Electronics (ICCE'06), 2006, pp. 348–351.

[20] F. Gilabert, M.E. Gómez, S. Medardoni, D. Bertozzi, Improved utilization of NoC channel bandwidth by switch replication for cost-effective multi-processor systems-on-chip, in: Proceedings of the ACM/IEEE International Symposium on Networks-on-Chip (NOCS'10), 2010, pp. 165–172.

[21] M. Winter, G.P. Fettweis, Guaranteed service virtual channel allocation in NoCs for run-time task scheduling, in: Proceedings of Design, Automation Test in Europe Conference Exhibition (DATE'11), 2011, pp. 1–6.

[22] A. Leroy, D. Milojevic, D. Verkest, F. Robert, F. Catthoor, Concepts and implementation of spatial division multiplexing for guaranteed throughput in networks-on-chip, IEEE Trans. Comput. 57 (9) (2008) 1182–1195.

[23] R. Stefan, A. Molnos, K. Goossens, dAElite: a TDM NoC supporting QoS, multicast, and fast connection set-up, IEEE Trans. Comput. PP (99) (2012) 1.

[24] K. Goossens, J. Dielissen, A. Radulescu, AEthereal network on chip: concepts, architectures, and implementations, IEEE Des. Test Comput. 22 (5) (2005) 414–421.

[25] N. Ma, Z. Lu, L. Zheng, System design of full HD MVC decoding on mesh-based multicore NoCs, Microprocess. Microsyst. 35 (2) (2011) 217–229.

[26] M. Winter, G.P. Fettweis, A network-on-chip channel allocator for run-time task scheduling in multi-processor system-on-chips, in: Proceedings of EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD'08), 2008, pp. 133–140.

[27] S. Liu, A. Jantsch, Z. Lu, Parallel probe based dynamic connection setup in TDM NoCs, in: Proceedings of the Conference on Design, Automation & Test in Europe (DATE'14), 2014, pp. 239:1–239:6.
[28] J. Lim, E. Hunt Siow, Y. Ha, P.K. Meher, Providing both guaranteed and best effort services using spatial division multiplexing NoC with dynamic channel allocation and runtime reconfiguration, in: Proceedings of International Conference on Microelectronics (ICM'2008), 2008, pp. 329–332.
[29] A.K. Lusala, J.-D. Legat, Combining sdm-based circuit switching with packet switching in a NoC for real-time applications, in: Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'11), 2011, pp. 2505–2508.

Shaoteng Liu received the B.Sc. degree from Fudan University, Shanghai, China, in 2006. He received his M.Sc. degree from Royal Institute of Technology (KTH), Stockholm in 2010. He is currently a PHD student at KTH. His current research interests include system modeling, performance analysis, embedded operating system, reconfigurable computing, network-on-chip and software defined network.

Axel Jantsch received the Dipl.Ing. and Dr.Tech. degrees from the Technical University of Vienna, Vienna, Austria, in 1988 and 1992, respectively. He was a professor of electronic system design with the Royal Institute of Technology, Stockholm, Sweden, from December 2002 to September 2014. He is currently a professor in system on chip with TU Wien, Vienna, Austria. His current research interests include VLSI design and synthesis, system-level specification, modeling and validation, HW/SW co-design and co-syntheses, reconfigurable computing, and networks-on-chip.

Zhonghai Lu received the B.Sc. degree from Beijing Normal University, Beijing, China, in 1989, and the M.Sc. and Ph.D. degrees from KTH Royal Institute of Technology, Stockholm, Sweden, in 2002 and 2007, respectively. He is currently an Associate Professor with KTH. His research interests include Network-on-Chip, Embedded Systems, Computer Architecture, and Internet-of-Things. He has published over 130 peer-reviewed papers in transactions, journals and international conferences in these areas.