Analysis and evaluation of circuit switched NoC and packet switched NoC

Shaoteng Liu
Royal institute of technology
liu2@kth.se

Axel Jantsch
Royal institute of technology
axel@kth.se

Zhonghai Lu
Royal institute of technology
zhonghai@kth.se

***Abstract*-Circuit switched NoC has, compared to packet switching, a longer setup time, guaranteed throughput and latency, higher clock frequency, lower HW complexity, and higher energy efficiency. Depending on packet size and throughput requirements they exhibit better or worse performance. In this paper we designed a circuit switched NoC and compared that with packet switched NoC. By speculation and analysis, we propose that, as packet size increases, performance decreases for packet switched NoC, while it increases for circuit switched NoC. By close examination on the router architecture, we suggest that circuit switched NoC can operate at a higher clock frequency than packet switched NoC, and thus at zero load above a certain packet size circuit switched NoC could be better than packet switched NoC in packet delay. Experiment results support our intuitions and analysis. We find the cross-over point, above which circuit switching has lower latency, is around 30 flits/packet under low load and 60-70 flits/packet under high network load**.

I. Introduction

Packet switched (PS) NoCs (Network on Chip) have been studied more extensively and thoroughly. However, circuit switched NoC (CS) NoC could be preferable under certain traffic patterns and requirements[1]. Thus, it requires an analysis and comparison on PS and CS NoC to reveal their properties and limitations, and offer intuitions for people to make design decisions.

In order to commence such a study, on one hand, for PS NoC, we use an input-buffered virtual channel (VC) wormhole routed PS NoC for comparison. This kind of PS NoC is widely utilized in practice, eg. TILE64[2]. On the other hand, since no classical design exists for CS NoC, we build our own platform by inheriting and integrating the merits from state-of-art works. In addition, mesh topology is used for both NoCs because it is the most popular NoC topology, and it is scalable, easy to layout, and offers path diversity.

In this paper we will show the respective strengths and weaknesses of circuit and packet switched NoCs. Particularly, our work offers following contributions:

- We reveal the detailed mechanisms which make PS NoC not fit for large packets (Section III).
- We suggest that CS NoC with proper design should work at a higher clock frequency than PS NoC (Section VI).
- By analysis on zero load packet delay, we find that above a certain packet size, CS NoC delivers faster than PS NoC (Section VI).
- By experiments and evaluations, we represent the respective favorite working areas of PS NoC and CS NoC. We reveal that if packet size is very large, even with more VCs and buffers, PS NoC still suffers a performance loss (section VIII)

II. Related work

There are only a few papers on the analysis and comparison of CS NoCs and PS NoCs. In [3][4] some comparisons on area and power consumption are presented. Although [5][6] concern about the performance of PS and CS NoC in a ring topology, the influence of packet size on delay and throughput is not evaluated and compared. The pros and cons of a certain kind of NoCs are still poorly understood. In addition, even though some other works have been done on combining PS and CS NoC [7][8][9][10], but none of them provides a serious analysis and evaluation on their respective characteristics.

PS NoC has been intensively studied by researchers. There are several kinds of PS NoC [11][12][13][14]. Generally speaking, input-buffered virtual channel wormhole routed NoC is well accepted and utilized[2][15]. Dally's book [16] has covered almost every aspect of such a PS NoC. Moreover, many works have focused on optimizations on this kind of PS NoC [12]. Therefore, we will also choose input-buffered virtual channel wormhole routed PS NoC in our paper.

However, there is no well-accepted CS NoC architecture. According to the path search and setup methods, CS NoCs can be classified into two categories: dynamic setup methods [10][9][19][20][21] or static setup methods [22][23]. Static setup methods schedule paths at compilation time. As a result, they may not well support applications like H.264 [24] with requirements for dynamic communication setups. Therefore, we only focus on dynamic methods which search and setup paths at run time.

Dynamic methods can be further classified into centralized [10][9] or distributed methods [1][25][19][20] [21][7][9][8][26][27]. Generally speaking, centralized setup has two disadvantages: One is scalability and the other is dropping of failed setup requests [17][18]. Since retrying of failed requests causes the blockage of the following requests, failed setup requests are usually dropped in centralized setup methods[17], [18]. We focus on distributed setup.

`/13 $26.00 © 2013 IEEE
DOI 10.1109/DSD.2013.13

Distributed setup can be implemented by packet configuration [21][1][9][8][27] or by a probing search approach [25][19][20][26].

Packet configuration requires an additional separate PS (packet switched) NoC to deliver configuration messages like set-up, tear-down and Ack/Nack during a path setup procedure. In our view, this approach suffers from four major drawbacks. Firstly, the adding of an additional PS NoC is a unnecessary overhead. Secondly, since set-up, tear down and Ack/Nack packets of a path must be routed by deterministic routing algorithm to ensure them on the same path. However, deterministic 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. Besides, there is typically no delay guarantee for packets in the PS network, rendering the path set-up procedure unpredictable. Fourthly, this approach does not scale well. In PS NoC, each output port of a switch just allows to deliver one packet every time slot. However, if there are several sub-channels in the CS NoC each sub-channel requires a separate setup packet for path configuration, thus significantly increasing the number of setup packets [21].

Compared with above mentioned shortcomings of a packet configuration approach, probing search is the superior choice because of its efficiency in wire resource utilization and path setup procedure. The concept of the probing search was firstly proposed by [26]. In [19][20], Pham et al. developed a backtracking routing algorithm, which reportedly has better performance than [26]. Another contribution of [19][20] is that a source synchronized data transfer mechanism is introduced into CS NoCs, so that separate clocks can be applied to path setup and data transfer. [28] 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 [19] by experiments.

III. Intuitions and Conjectures

According to our considerations, large packets are detrimental to PS NoC in two ways:

Firstly, large packets will cause burst injection of flits. Bursty traffic may cause massive contentions in a short period, and thus prolongs the average flits delivering delay.

Secondly, large packets will incur unbalanced usage of channels. As illustrated in Fig. 1, we consider the traffics inside an input buffered virtual channel wormhole router by assuming that there is only two virtual channel queues (FIFO queues) and two output directions. Each packet desires either output A or output B for delivering. We suppose that packets with different desires are uniform randomly distributed. However, flits belong to one packet must have the same desired output direction and queued by the same VC. Round-robin allocator is used for allocation. Since it is possible that at a certain cycle both flits from VC1 and VC2 want the same output channel, leaving the other output channel an idle cycle.

As Fig. 1 suggests, when each packet just contains one flit, the span of idle periods are just made up of a few cycles and equally distributed between output flow A and B. However, when packet size increases, the average span of idle periods grows wider and wider.

Although from statistical view, in a very long term, output flow A and B should have the same number of idle cycles; no matter what the packet size is, the total idle cycles should be equal. However, within a short period, we observe that with large packets, output flow A is quite busy, while output flow B is almost idle. Such unbalanced usage of physical channels is destructive to throughput.

In PS NoC, we deal with the burstiness by increasing the buffer depth of a VC, and balance the traffics by adding more VCs. Long packets are detrimental to PS NoC since they increase the needs on both sides. The longer the packets, the more VCs and buffers are required to compensate the performance loss. Unfortunately, both VC and buffer are expensive, especially that adding virtual channels will lower down the clock frequency. Thus, we can imagine that, if packets are long enough, they are almost impossible to be well handled by PS NoC with acceptable cost.

Nevertheless, CS NoCs favor large packets. In CS NoC, after a path has been established, data transfer will be launched. So that the channel utilization ratio for CS NoC is $= \frac{t_{trans}}{t_{setup}+t_{trans}}$.

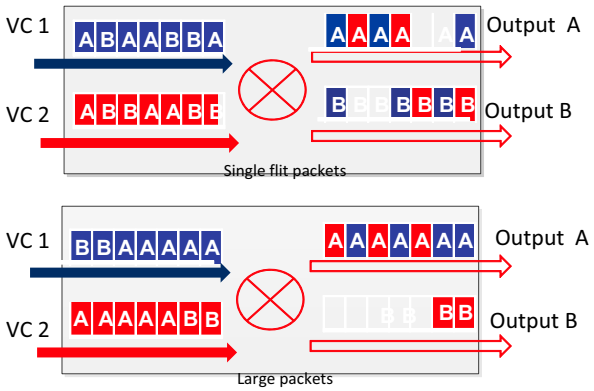


Fig. 1 Unbalance traffic in PS NoC caused by large packets

As packet size increases, t_{trans} goes up, so the channel utilization ratio U increase. The per-node throughput, which equals to channel utilization ratio multiplies bandwidth, increases as well.

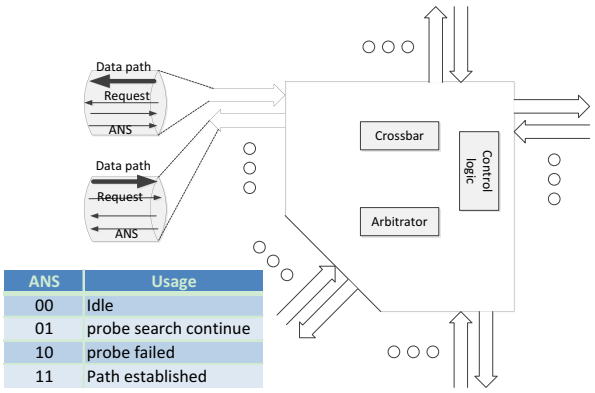
We may conjecture that, as the packet size increases, the performance curve of CS NoC and performance curve of PS NoC might have a cross point, since one rises and the other falls. However, we need practical parameters such as clock

frequency, as well as experiment results, to justify the existence of such a point.

IV. Architecture of our CS NoC

A. Overview of a switch

In this paper, our CS NoC design adopted probing approach with parallel probing routing algorithm [28].



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

Fig. 2 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 has one duplex channel. The data path of a channel is used for carrying the probe during setup and for transmitting data when a connection has been established. Each probe is one flit in length. Every data path is associated with 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 path will be released. The usage of the ANS signal is listed in the table of Fig. 2.

B. Detailed switch architecture

The internal structure of a switch is shown in Fig. 3. 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.

There are five allocators, each of which is responsible for the channel allocation of one output direction. The principle of our single cycle maximal strong fairness allocator is similar to a wave-front allocator [13][14]. Detailed discussion of our allocator is beyond the scope of this paper.

Besides, every input or output channel has a controller that controls the value of the backward ANS signal and the forward probe. The FSMs of the input and output controllers are shown in Fig. 4.

C. Properties of our CS NoC

1) Source synchronized data transfer

Similar to Pham's work [19] and as in [28], the control path and data path can work at different clock frequencies but share the same wires without interference. This clock scheme takes advantage of the property of CS NoC that the setup phase never overlaps with the data transfer phase. During the path 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.

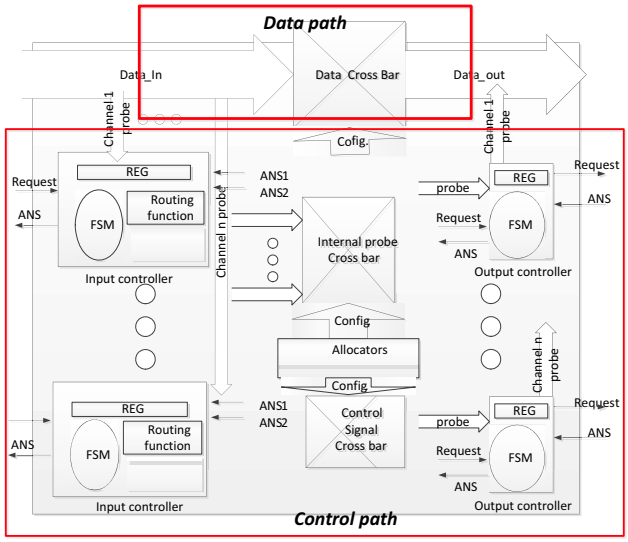
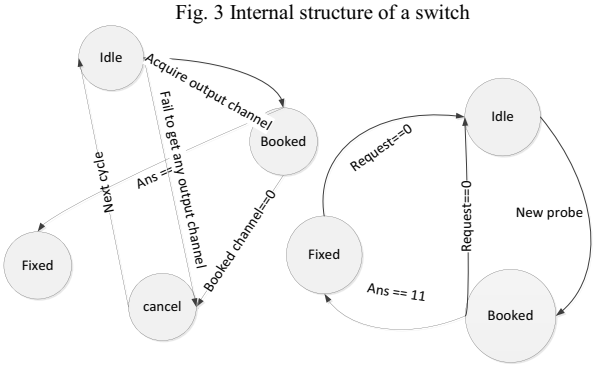


Fig. 3 Internal structure of a switch



Input controller FSM. b) Output controller FSM

Fig. 4 FSM of input controller and output controller

Therefore, we can utilize either source synchronous data transfer [29][30][19] or clock gating to realize this separation of data and control path clock schemes, so that the data transfer can be 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. [19][20].

We want to emphasize that source synchronized data transfer is a unique property of CS NoC. PS NoC cannot take advantage of such technique because its TDM channel sharing nature.

2) Predictable latency

One of the benefits of the probing set-up approach is predictable latency. In our design, each setup probe takes 2 clock cycles per hop, and the ANS signal takes 1 cycle per hop. So, it takes at most 3*D+6 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.

For data transfer, the head flit takes 2 cycles per hop, and the following flits are pipelined with 2cycles per hop.

V.PS NOC USED FOR COMPARISON

The PS NoC for comparison adopts a classical input-buffering virtual channel (VC) wormhole router architecture with the dimension order routing algorithm. The router's architecture is shown in Fig. 5. A router has 3 pipeline stages: (1) VC Allocation and speculative Switch Allocation Switch Traversal (ST) and Next Route Computation (NRC). (3) Link traversal[1].

The PS NoC adopts some advance techniques like speculative allocation and look-ahead route computation [16][31]. Speculative allocation enables head flits to bypass the VC allocation stage in the router pipeline by allowing them to bid for crossbar access at the same time they request an output VC. Look-ahead route computation, or next route computation, computes the route for the next router before a head flit is actually delivered to the next router. These two techniques can reduce the pipeline stages of a PS router.

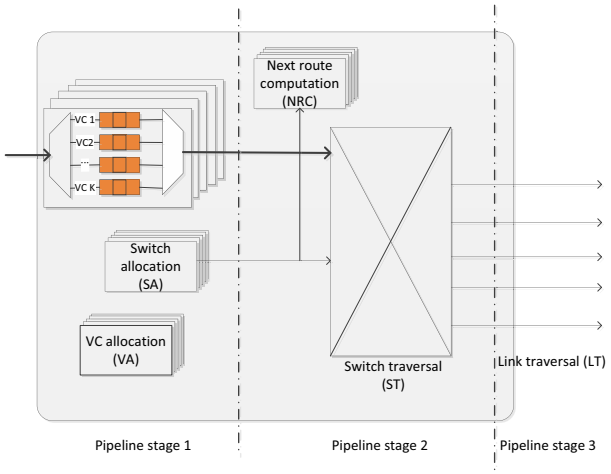


Fig. 5 Architecture of the PS NoC for comparison

[1] The link traversal stage is used to compensate wire delay.

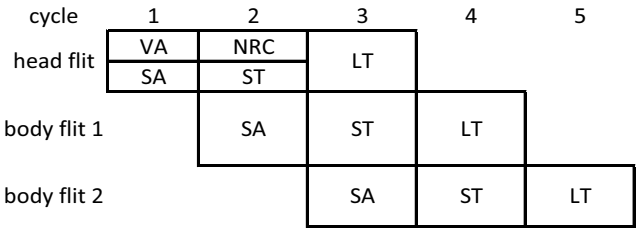


Fig. 6 Pipeline stages of the PS NoC

As suggested by Fig. 6, head flits and body flits experience 3 pipelines. Both virtual channel allocations and switch allocations are realized by simple separable-input-first round-robin allocator to reduce the critical path latency.

The PS NoC in this paper can have different number of virtual channels and buffer depths.

VI.ANALYSIS ON PS NOC AND CS NOC

A. Critical path latency analysis

Let us consider a CS NoC. For data path, the critical path latency is basically a 4-by-4 cross-bar latency. Using standard library, the critical path of the cross-bar is just a 4-to-1 multiplexer, which consists of 4 level and/or gates latency. Full custom design can further reduce such latency to one gate level. For control path, the critical path consists of a 4-channel allocator latency plus a 4-by-4 crossbar latency.

For PS NoC, the critical path latency comes from VC-allocator. A separable allocator with round-robin fairness consists of two consecutive steps of round-robin arbitration. Even each input direction just has one VC, still the latency of such 4-channel allocator is much larger than the 4-by-4 cross-bar latency inside CS NoC. With the most advanced design [32], the latency of a round-robin allocator scales up with $O(logq)$, where q is the number of virtual channels. Empirically, each input directions needs 4 or more VCs to reduce head-of-line blockings, so the VC allocator in practice is at least a 16-channel allocator.

Comparing CS NoC with PS NoC which has 4 VCs per input, we may deduct that the critical data path latency td of CS NoC should be much smaller than the critical path latency tp of PS NoC. The critical control path latency tc of CS NoC should be close to, or smaller than tp. This is because tc consists of one 4-channel allocator latency plus one 4-by-4 crossbar latency, while tp is basically the latency of a 16-channel allocator, which may double the latency of a 4-channel allocator.

B. Zero load analysis of CS vs. PS

Zero load analysis is useful, since in practice, compared to actual requirements, a NoC is usually over-designed. This means that in a lot of chances, an on-chip network is operating under very low traffic load.

Suppose a packet contains k flits, the clock period is tp for PS NoC, and tc for the probe path of CS NoC and td for data path. The average hops of a packet is D. Suppose

further that it takes *hp* cycles-per-hop for the head flits and 1 cycle per hop for the following flits in the PS NoC, and in the CS NoC it takes *hc* for the setup probes (including the travelling back of acknowledgement signal) and *hd* for the data. For simplicity, overheads in sender, receiver and network interface are neglected. Then, at zero load, the approximate average time of packet delivery is, PS NoC and CS NoC, respectively:

$$Tps = [D * hp + k] * tp \quad (1)$$

$$Tcs = D * hc * tc + D * hd * td + (k-1) * td \quad (2)$$

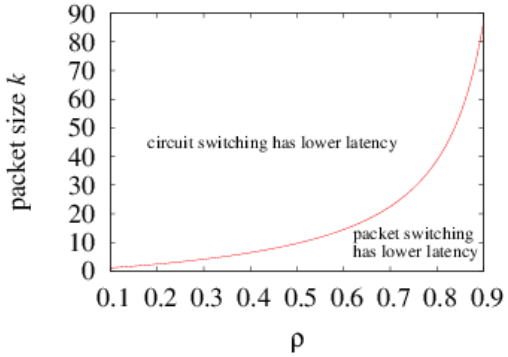


Fig. 7 Packet size k for break even points between PS and CS NoC

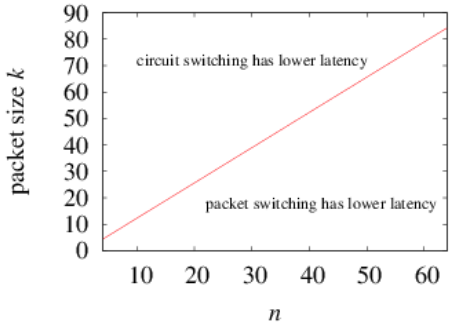


Fig. 8 Mesh size n for break even points between PS and CS NoC

To make a simple comparison, we assume that tp=tc, hd=hc=hp=2 cycles per hop. Then the breakeven point is defined by

$$Tps - Tcs = k * tp - (k-1) * td - D * 2 * td \quad (3)$$

$\rho = td/tp$ expresses how much faster the circuit switching logic can be compared to packet switching. $\rho = 1$ means both run at the same frequency. However, according to our previous analysis, $\rho < 1$. When $\rho < 1$, circuit switching exhibits lower packet delay for packets above a certain size *k*. This value of *k* is plotted in Fig. 7 as a function of ρ and for (D=6).

Does the critical packet size *k* depend on the size of the network? We consider this by using uniform random traffic in an nxn mesh, so that $D = \frac{2}{3}n$. For $\rho = 0.5$, we vary *n* between 4...64, and find the break even k grows from 5 to 85, as shown in Fig. 8. Thus, with larger networks the packet size has to be larger as well for CS NoC to become faster than PS NoC.

Mathematical analysis of network contention under load is complex and has to be based on many assumptions. Therefore, we study the networks under various loads by simulation where we also have more realistic values for *tc*, *td*, *tp*, *hc*, *hd*, and *hp*.

VII. Experiment Settings

In section 3, we have made our conjecture on the performance curves of CS NoC and PS NoC. Now, we will check whether experiment results are in accordance with our conjecture. All experiments are based on PS NoCs and CS NoCs with 8x8 mesh topology. Uniform random traffic with Poisson arrival time distribution is used in our experiments for evaluation purpose.

A. Simulation method

As in Fig. 9, inside each resource node a request generator generates set-up requests according to a certain probability distribution and pushes them into a queue. An FSM (Finite State Machine) pops a request out of the queue and sends it out when the output channel is available. Then the FSM waits for the ANS signals to decide what to do next.

We have implemented an HDL model for synthesis and for evaluation. Any data point that is shown in the figures comes from a simulation of 250 million cycles, of which the first 250000 cycles are discarded as warm up period.

Since PS NoC and CS NoC are operating at different clock frequencies, performances in the unit of clock cycles are not persuasive. Thus we evaluate the delay in the unit of nano-second, and the throughput and injection rate in MB/s.

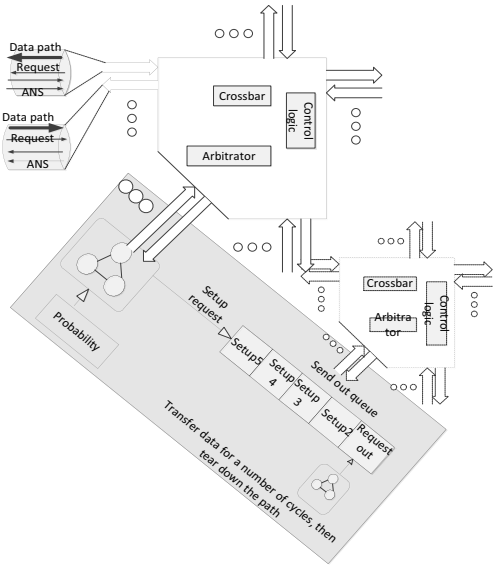


Fig. 9 Experiment setup

VIII. Evaluation and Comparison

A. Evaluations on baseline candidates

The baseline candidates of the PS and CS NoC are listed in Tab. 1, both of which are synthesized by Synopsys

Design Compiler (DC) with SMIC 90 nm library. Both candidates have the same channel width (8 bytes).The power and area per switch reported by DC is calculated at each one's maximum clock frequency.

The synthesize results obey our previous analysis. The data path of CS NoC is more than twice faster than PS NoC. The control path of CS NoC is also about 1.3 times faster than PS NoC.

The header flits in PS NoC take 3 cycles per hop. And for CS NoC, the probes take 2 cycles per hop and 1 cycle for backward ANS signal, under the control of probe clock. The data transfer takes 2 cycles per hop, under the control of the data clock.

Therefore, in our experiments, according to Tab. 1, tp=1.2 ns, tc=0.9 ns, td=0.56 ns, hp=3, hc=2, hd=2, $D = \frac{2}{3}n = 5.3$.

Tab. 1 Parameters and synthesis results of PS NoC and CS NoC (Per switch)

Packet Swiched NoC	Circuit Switched NoC
Virtual channels: 4 (v4)	Channel per direction: 1
Buffer size: 4 (b4)	
Flit width : 8 bytes	Channel width : 8 bytes
Max. Freq. : 833 MHz	Max. Data Freq.: 1.786GHz
	Max. Probe Freq.: 1.11GHz
Area: 144572 um^2	Area: 30874. um^2
Power: 21.7 mW @ 833MHz	Power: 26.8 mW @1.786/1.11GHZ

The channel bandwidth and the clock frequency of PS and CS NoC can be found in Tab. 1. The influence of delivering packets of variable size (from 4 flits to 160 flits) in both PS NoC and CS NoC is evaluated and shown in Fig. 10 and Fig. 11. The trends in these two plots are opposite to each other. PS NoC (Fig. 10) decreases in performance under load as the packet size gets larger, while CS NoC (Fig. 11) improves in performance with larger packets. To better illustrate this phenomenon we take snapshots at injection rates 2000 MB/s and 1142 MB/s, and plot packet size against average packet delay (Fig. 12.).

For a given injection rate CS NoC improves performance as packets get larger, while the performance of PS NoC deteriorates. Consequently, there are cross-points between the circuit-switching and the PS NoC curves in Fig. 12. At injection rate 2000 MB/s, the cross-point is 62 (496 bytes) flits and at injection rate of 1142 MB/s the cross point is 40 flits. In general, the cross point shifts towards larger packets as the injection rate increases.

This observation can be explained by three phenomena:

- For medium packets (20-60 flits in this case) and large packets (above 60 flits in this case) at low load, CS NoC performs better than PS NoC. This is in agreement with our zero load analysis.
- PS NoC handles small (below 20 in uniform traffic case) and medium packets at high load better than CS NoC. A given number of VCs and buffers can handle burstiness and unbalance below a certain level well. If in a PS NoC small packets compete for a resource, the only cost is the waiting time of one packet for a few cycles. If contention occurs in a CS NoC during the setup phase, one probe has to go back, tear down all allocated resources and start over again. Thus, the penalty is more sever because the delay due to contention is higher and many resources are allocated unnecessarily.
- As packet size increases, the contention overhead increases for PS and it is constant (decreases in relative terms) for CS NoC. In PS the delay incurred by contention is proportional to packet size. Thus, the larger the packets the higher the penalty of contention. For CS the cost of contention is relatively independent of packet size and is thus better amortized over large packets at high load.

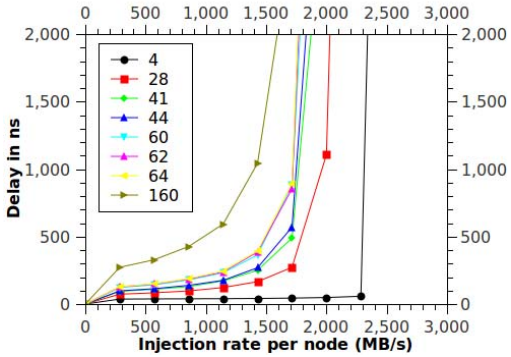


Fig. 10 Delay for PS NoC with different packet size in flits

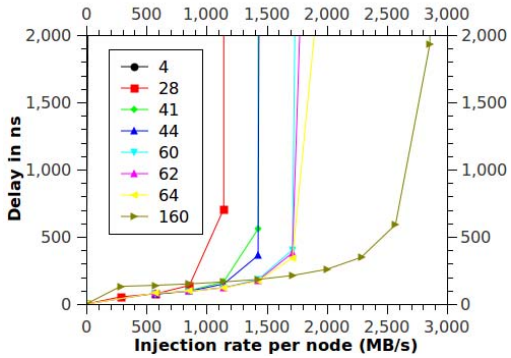


Fig. 11 Delay for CS NoC with different packet size in flits

We also studied the influence of packet size on maximum throughput, as suggested by Fig. 13. As the packet size increases, the maximum throughput decreases in PS NoC, while CS NoC has a contrary trend. For example, for PS NoC with packets of 4 flits (32 bytes), the maximum throughput reaches 2614 MB/s. However, when packet size grows up to 640 flits (5120 bytes), the maximum throughput is 1950 MB/s. For CS NoC, the maximum throughput is 279 MB/s with 4-flit packets. However, as the packet size grows to 640 flits, the throughput goes up to 3479 MB/s. Again, there is a cross over point which lies around 62 flits/packet.

Thus, when packets are large enough, CS NoC is superior to PS NoC in both latency and throughput.

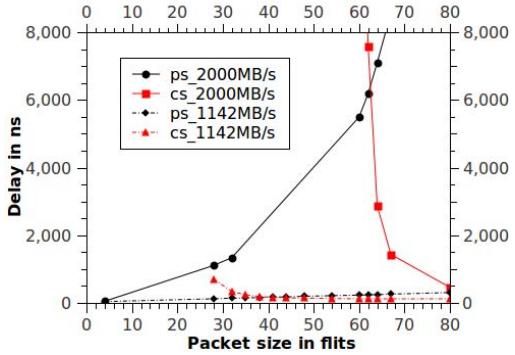


Fig. 12 Latency snapshot of injection rate 2000 MB/s and 1142 MB/s

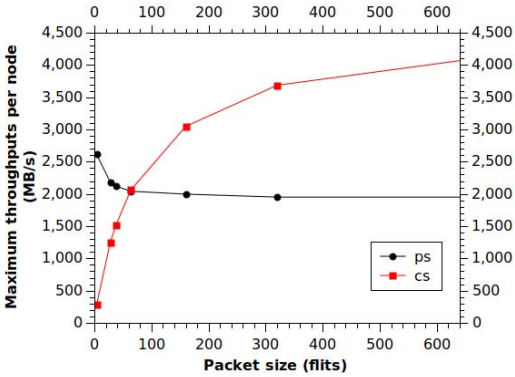


Fig. 13 Maximum throughput comparison.

B. Comparisons between more candidates

In this section, we compared more configurations of PS NoC. Throughput comparison results are shown in Fig. 14. The packet size is in the unit of byte.

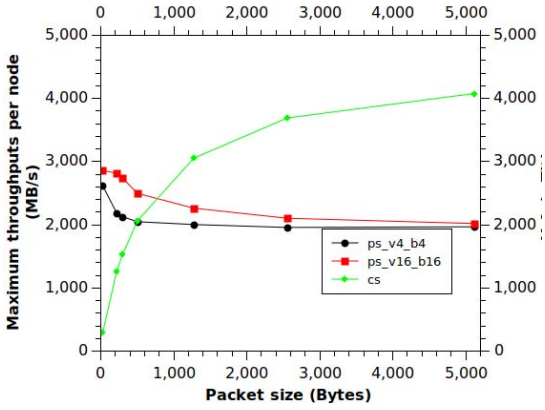


Fig. 14 Maximum throughput comparison between PS NoC and different configurations of CS NoC

For PS NoC, as we expected, we can see that ps_v16_b16 (16 virtual channels and each contains 16 buffers) can enhance the maximum throughput when packets are not very large.

However, for large packet size, eg. each packet contains 640 flits (5120 bytes), 16 VCs with 16 stages of buffers are still not enough. The maximum throughput of ps_v16_b16 at such a packet size is reduced to that of ps_v4_b4. Since large packet weakens the effects of VCs and buffers, we can conjecture that if packets are large enough, the maximum throughput of configurations which contain more VCs and buffers will shrink to that of the baseline candidate ps_v4_b4.

IX. Conclusions

Our general conclusions are shown in Tab. 2, detailed conclusions are listed as followings:

Tab. 2 The favorite working areas of CS and PS NoC

	small packets	medium packets	large packets
Low load	PS is better	CS is better	CS is better
High load	PS is better	PS is better	CS is better

- At very low injection rate, CS NoC is better than PS NoC above a certain packet size (see equation (3)). The critical packet size, above which CS NoC outperforms PS NoC, grows linear with the size of the network.
- For small packet size, PS NoC handles congestion better than CS NoC. Hence, when increasing injection rate, PS NoC shows better performance. This explains why for smaller size of packets (below 62 flits), PS NoC outperforms CS NoC at high injection rate.
- Growing packet size increases the congestion penalty for PS NoC, while it is relatively packet size independent for CS NoC. This means for a given injection rate, CS NoC performs better and better with growing packet size, while PS NoC is getting worse.
- Increasing virtual channels and buffers of PS NoC can enhance throughput for small packets. But this has little influence on very large packets.

References

[1] A. Leroy, D. Milojevic, D. Verkest, F. Robert, and F. Catthoor, "Concepts and Implementation of Spatial Division Multiplexing for Guaranteed Throughput in Networks-on-Chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1182 –1195, Sep. 2008.

[2] 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*, 2008, pp. 88–598.

[3] K.-C. Chang, J.-S. Shen, and T.-F. Chen, "Evaluation and design trade-offs between circuit-switched and packet-switched NOCs for application-specific SOCs," in *2006 43rd ACM/IEEE Design Automation Conference*, 2006, pp. 143 –148.

[4] P. T. Wolkotte, G. J. M. Smit, G. K. Rauwerda, and L. T. Smit, "An Energy-Efficient Reconfigurable Circuit-Switched Network-on-Chip," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, 2005, p. 155a.

[5] N. Chin-Ee and N. Soin, "Qualitative and quantitative evaluation of a proposed circuit switched network-on-chip," in *2010 IEEE*

International Conference on Semiconductor Electronics (ICSE), 2010, pp. 108–113.

[6] N. Chin-Ee and N. Soin, "A study on circuit switching merits in the design of network-on-chip," in *2010 International Conference on Computer and Communication Engineering (ICCCE)*, 2010, pp. 1–5.

[7] J. Lim, E. Hunt Siow, Y. Ha, and P. K. Meher, "Providing both guaranteed and best effort services using Spatial Division Multiplexing NoC with dynamic channel allocation and runtime reconfiguration," in *Microelectronics, 2008. ICM 2008. International Conference on*, 2008, pp. 329 –332.

[8] M. Modarressi, H. Sarbazi-Azad, and M. Arjomand, "A hybrid packet-circuit switched on-chip network based on SDM," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 3001 Leuven, Belgium, Belgium, 2009, pp. 566–569.

[9] A. K. Lusala and J.-D. Legat, "Combining sdm-based circuit switching with packet switching in a NoC for real-time applications," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, 2011, pp. 2505 –2508.

[10] A. K. Lusala and J.-D. Legat, "Combining SDM-Based Circuit Switching with Packet Switching in a Router for On-Chip Networks," *International Journal of Reconfigurable Computing*, vol. 2012, pp. 1–16, 2012.

[11] M. Millberg, E. Nilsson, R. Thid, S. Kumar, and A. Jantsch, "The Nostrum backbone-a communication protocol stack for Networks on Chip," in *17th International Conference on VLSI Design, 2004. Proceedings*, 2004, pp. 693–696.

[12] S. Park, T. Krishna, C.-H. Chen, B. Daya, A. Chandrakasan, and L. Peh, "Approaching the theoretical limits of a mesh NoC with a 16-node chip prototype in 45nm SOI," in *2012 49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2012, pp. 398 –405.

[13] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *Proceedings of the conference on Design, automation and test in Europe* , 2000, pp. 250–256.

[14] F. Moraes, N. Calazans, A. Mello, L. Möller, and L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip," *the VLSI Journal*, vol. 38, no. 1, pp. 69–93, Oct. 2004.

[15] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software, 2009. ISPASS 2009*, 2009, pp. 33–42.

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

[17] M. Winter and G. P. Fettweis, "A Network-on-Chip Channel Allocator for Run-Time Task Scheduling in Multi-Processor System-on-Chips," in *11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools, 2008. DSD '08*, 2008, pp. 133 –140.

[18] M. Winter and G. P. Fettweis, "Guaranteed service virtual channel allocation in NoCs for run-time task scheduling," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, 2011, pp. 1 –6.

[19] P.-H. Pham, J. Park, P. Mau, and C. Kim, "Design and Implementation of Backtracking Wave-Pipeline Switch to Support Guaranteed Throughput in Network-on-Chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 2, pp. 270 –283, Feb. 2012.

[20] P.-H. Pham, P. Mau, J. Kim, and C. Kim, "An On-Chip Network Fabric Supporting Coarse-Grained Processor Array," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –5, 2012.

[21] A. K. Lusala and J.-D. Legat, "Combining SDM-Based Circuit Switching with Packet Switching in a Router for On-Chip Networks," *International Journal of Reconfigurable Computing*, vol. 2012, pp. 1–16, 2012.

[22] R. Stefan, A. Molnos, and K. Goossens, "dAElite: A TDM NoC Supporting QoS, Multicast, and Fast Connection Set-up," *IEEE Transactions on Computers*, vol. PP, no. 99, p. 1, 2012.

[23] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal network on chip: concepts, architectures, and implementations," *IEEE Design Test of Computers*, vol. 22, no. 5, pp. 414 – 421, Oct. 2005.

[24] N. Ma, Z. Lu, and L. Zheng, "System design of full HD MVC decoding on mesh-based multicore NoCs," *Microprocess. Microsyst.*, vol. 35, no. 2, pp. 217–229, Mar. 2011.

[25] A. Leroy, P. Marchal, A. Shickova, F. Catthoor, F. Robert, and D. Verkest, "Spatial division multiplexing: a novel approach for guaranteed throughput on NoCs," in *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, New York, NY, USA, 2005, pp. 81–86.

[26] D. Wiklund and D. Liu, "SoCBUS: switched network on chip for hard real time embedded systems," in *Parallel and Distributed Processing Symposium, 2003. Proceedings. International*, 2003, p. 8–pp.

[27] C. Hilton and B. Nelson, "PNoC: a flexible circuit-switched NoC for FPGA-based systems," *IEE Proceedings of Computers and Digital Techniques,* vol. 153, no. 3, pp. 181–188, 2006.

[28] S. Liu, A. Jantsch, and Z. Lu, "Parallel probing: Dynamic and constant time setup procedure in circuit switching NoC," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, 2012, pp. 1289 –1294.

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

[30] D. Schinkel, E. Mensink, E. Klumperink, E. van Tuijl, and B. Nauta, "Low-Power, High-Speed Transceivers for Network-on-Chip Communication," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 17, no. 1, pp. 12 –21, Jan. 2009.

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

[32] H. J. Chao, C. H. Lam, and X. Guo, "Fast ping-pong arbitration for input–output queued packet switches," *International Journal of Communication Systems*, vol. 14, no. 7, pp. 663–678, 2001.