McMahon: Minimum-cycle Maximum-hop network

Gokul Subramanian Ravi
University of Wisconsin - Madison
gravi@wisc.edu

Tushar Krishna
Georgia Tech
tushar@ece.gatech.edu

Mikko Lipasti
University of Wisconsin - Madison
mikko@engr.wisc.edu

Abstract

McMahon: Minimum-cycle Maximum-hop network proposes to extend the SMART [8] NOC to enable data to traverse multiple hops in minimum possible cycles in an efficient manner. McMahon uses transparent latching based flit traversal to enable flits to potentially travel the entire route from start node to end node in a single pass. McMahon improves performance in the NOC further by (a) avoiding hop quantization and (b) estimating the accumulated slack over the course of flit traversal in the interconnect network and recycling this total slack.

Preliminary evaluation shows that McMahon can reduce flit latency beyond SMART by almost 50% under favorable NOC and workload characteristics. This paper also introduces new design features which are in development.

1 Introduction

Single-cycle Multi-hop Asynchronous Repeated Traversal (SMART) NoC [8] exploits the observation that global repeated wires are fast enough to send signals across 10+ mm within 1ns. SMART NoCs augment mesh routers with a bypass mux (that acts as a repeater) and enable flits to traverse multiple routers asynchronously in one cycle before getting latched. The maximum number of hops that can be traversed in a cycle is a design time parameter known as HPCMAX (maximum hops per cycle), which depends on (a) the underlying repeated wire delay at the particular technology node, (b) the clock frequency, and (c) the tile size. The authors in SMART observed a HPCMAX of 9 to 11 at 45nm at 1GHz with 1mm * 1mm tiles.

Under designs enjoying high HPCMAX, SMART has high potential for performance speedup. On the other hand, if HPCMAX is low (discussed in Section 2), benefits might be lesser and the SMART design presents opportunities for improvement.

In this proposal we design McMahon, an NOC design inspired by SMART, which is able to achieve the potential of high HPCMAX even in designs wherein HPCMAX is low. McMahon uses transparent latching based flit traversal to enable flits to travel the entire route from start node to end node in a single pass (if without conflicts) - covering multiple hops over minimum number of cycles. McMahon improves performance in the NOC further by (a) avoiding the hop quantization constraint of the SMART design and (b) estimating the accumulated slack over the course of flit traversal in the interconnect network and recycling this total slack (note that in this preliminary work we only estimate slack qualitatively).

AISTECS, January 2019, Spain

.

2 Background and Motivation

2.1 SMART: Multiple hops in a single cycle

A comparison of the SMART router pipeline and a baseline 2 cycle per hop design are shown in Fig.1 (a) and (b) respectively. SMART enables flits to traverse multiple routers asynchronously in one cycle before getting latched. A SMART-hop starts from a start router, where flits are buffered. Unlike the baseline router, Switch Allocation in SMART occurs over two stages: Switch Allocation Local (SA-L) and Switch Allocation Global (SA-G). SA-L is identical to the SA stage in the conventional pipeline: every start router chooses a winner for each output port from among its buffered (local) flits. In the next cycle, instead of the winners directly traversing the crossbar (ST), they broadcast a SMART-hop setup request (SSR) via dedicated repeated wires (which are inherently multi-drop) up to HPCMAX. The SSR carries the length (in hops) up to which the flit winner wishes to go. For instance, SSR = 2 indicates a 2-hop path request. Each flit tries to go as close as possible to its ejection router, hence SSR = min(HPC_{max}, $H_{remaining}$). SA-G is performed after an SSR arrives at a router. During SA-G, all inter routers arbitrate among the SSRs they receive which guarantee that only one flit will be allowed access to any particular input/output port of the crossbar. In the next cycle (ST+LT), SA-L winners that also won SA-G at their start routers traverse the crossbar and links up to multiple hops. Thus flits spend at least 2 cycles (SA-L and SA-G) at a start router before they can use the switch. Flits can end up getting prematurely stopped (i.e before their SSR length) depending on the SA-G results at different routers.

2.2 Limitations under low HPCMAX

SMART enables flits to traverse multiple hops in a single cycle, with potential for significant reduction in flit latency when HPCMAX is large. But analysis performed in [9] shows that at higher NOC frequencies, the maximum number of hops that can be traversed per cycle falls super-linearly. Further, prior work [6] suggests that server processors are relatively fat with high clock frequency, and so a single-cycle multi-hop NOC can send a packet over just a few hops in a single cycle (e.g., two hops). Given that extra cycles are needed to set up a multi-hop path (details in Section 3), the net effect of SMART might potentially reduce under low HPCMAX.

Further, SMART can suffer some limitation from quantization at the end of a single-cycle multi-hop sequence. If the number of hops that can be covered per cycle is N integral hops + M fractional hop, the actual number is quantized to N. While the effect of quantization loss (M/N) might be insignificant under a high HPCMAX, it can be significant when HPCMAX is low. For example, if 2.5 hops can be theoretically covered per cycle, quantization to 2 hops/cycle results in a 20% wastage in potential.

2.3 NOC Slack

Next, the estimated latency of traversing a single hop is usually governed by the worst case delay characteristics across operating nodes (V/F), PVT variation and lengths of links and crossbar paths. This potentially results in a considerable percent of the switch/link traversal clock cycle being wasted as slack i.e. portion of clock cycle with no useful work.

On-chip networks are especially vulnerable to within-die parameter variations. Since they connect distant parts of the chip, they need to be designed to work under the most unfavorable parameter values in the chip [2]. Prior works have noticed more than 30% difference in the minimum voltage required for error-free functioning of routers across a 64-node NOC [2].

Further, circuit level analysis [7] has shown that there is considerable difference in the scaling of wire resistance and switching resistance with voltage, meaning that the available slack during link traversal can vary significantly with changing V/F operating points.

Finally, the latency of flit traversal is also dependent on the path it takes. The length of the path is determined by the connections traversed within the crossbar and the lengths of the links traversed. For example, in a torus based topology the wrap around link length is usually greater than the other link lengths. Similarly, the average delay through the crossbar is considerably lesser than the worst-case critical path delay. If link+switch traversal is single cycle, they would be timed by the worst case delay which results in considerable slack in the average scenario.

3 Proposal for McMahon

McMahon uses transparent latching based flit traversal which allows flits to travel the entire route from start node to end node in a single pass (if without conflicts). Only a single pass for the entire flit route instead of many SMART-style multi-hop passes provides two benefits: a) it avoids the overheads of extra setup cycles required by SMART for each pass and b) it avoids the integral hop quantization on each pass. McMahon further improves hops covered per cycle by estimating the slack accumulated over a flit traversal route and then recycling the total slack.

Example Timing Analysis:

We explain the proposal with an example described below and with timing diagram shown in Fig.1. Consider that a flit needs to travel five hops from source to destination. Assuming no conflicts at any router, the number of cycles to the destination in a standard 2-cycle/hop design, the SMART design, and two versions of McMahon are discussed below. Note that naming convention for operations in designs (b)-(d) are as followed in SMART.

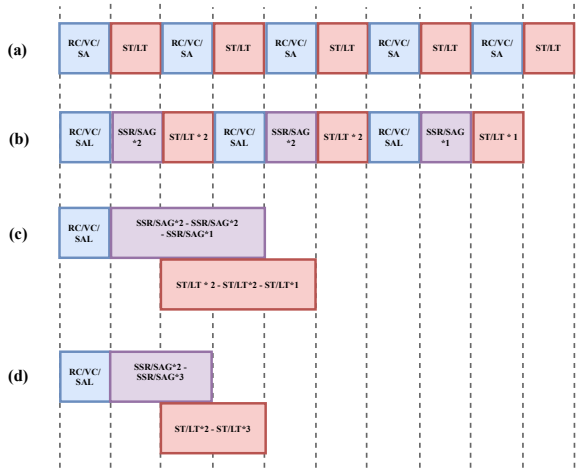

Figure 1. Timing Analysis of (a) a standard 1-cycle router + 1-cycle link traversal, (b) SMART [8], (c) McMahon:vanilla, w/o slack recycling, (d) McMahon:slack, with slack recycling

The baseline 2 cycle per hop design (Fig.1.a) would take 10 cycles to perform this route, assuming no conflicts at any router.

Next, assume that wire delay analysis (at design time) shows that 2.25 hops can be traversed per cycle. Assume that this constraint is therefore directly applicable for both the control bits (SSR) and the data (flit). Thus HPCMAX is (int) 2.25 = 2. The SMART design consumes 3 cycles for a maximum of HPCMAX hops, this can be broken down into: 1 cycle for RC/VC/SAL, 1 cycle for SSR/SAG to HPCMAX routers and 1 cycle for data flit traversal of to the HPCMAX routers. Thus SMART, as shown in Fig.1.b, would take 3 cycles for first 2 hops, 3 cycles for the next 2 hops and 3 cycles for the last hop, totalling 9 cycles.

The McMahon:vanilla proposal (Fig.1.c) aims to perform the 5 hops in only 5 cycles. This would involve 1 cycle of RC/VC/SAL, 3 cycles for SSR/SAG, pipelined with 3 cycles of switch/link traversal. The 5 hop traversal time of 3 cycles is the minimum possible since, the wire delay analysis showed that 2.25 hops can be traversed per cycle.

Note that the wire delay analysis above was assumed to be done at design time. Now assume availability of dynamic delay analysis. Let's say that under favorable PVT conditions and/or a favorable route consisting of shorter links, and/or favorable short input/output crossbar connections, the number of hops per cycle is 2.75. Running at only 2.25 hops per cycle would result in roughly 20% slack in such a system.

The McMahon:slack proposal (Fig.1.d) aims recycle this slack by completing the 5 hops in lesser cycles (4 cycles

is sufficient). This would involve 1 cycle of RC/VC/SAL, 2 cycles for SSR/SAG, pipelined with 2 cycles of switch/link traversal.

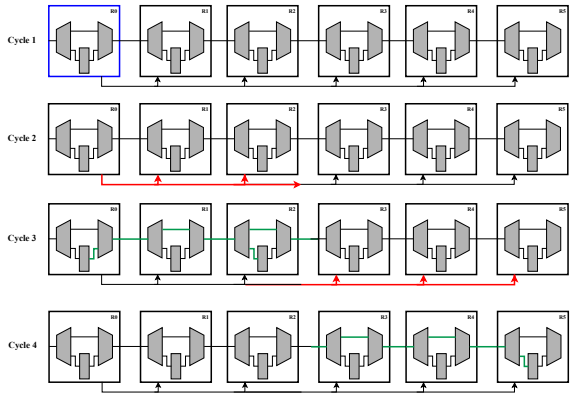

Figure 2. Design illustration without Conflicts

4 McMahon Illustrative Design

Fig.2 illustrates the cycle by cycle data/control transitions for McMahon:slack (was shown in Fig.1.d), but the described design is applicable to McMahon:vanilla as well.

Traversal without conflicts:

A flit is to be transmitted from R0 to R5, which is 5 hops away. In this example we assume that all switch allocations are won by our flit of interest. We use the wire delay estimate of 2.75 hops/cycle to apply to both data and control.

① In Cycle 1, R0 performs router computation, virtual channel allocation and local switch allocation - this is shown in blue. The flit wins locally and the SSR now needs to be transmitted out to the 5 routers.

② In cycle 2, the SSR signal reaches routers R1 and R2 and SAG is performed at these routers, indicated by red in figure. Moreover, the SSR signal is enroute to R3.

③ In cycle 3 there is both data flow and control flow. The data flit flows from R0 and is shown in green. It flows through R1 without being latched. At R2, it is allowed to flow through but is also latched in. This flit is latched because at this point it is not know if SSRs flowing to R3/R4 etc have won switch allocation. If not, we would need to restart flow from R2. The flit is also allowed to flow beyond R2 speculatively so as to use up the cycle completely. At the end of cycle 3, the flit is in flow between routers R2 and R3. Meanwhile, the SSRs reach routers R3, R4, R5 and win SAG.

④ In cycle 4, the data flit travels the remaining 2.25 hops, thus traveling through R3, R4, and gets latched at destination router R5.

Note that the links/routers are not being all held throughout the 4 cycles i.e. the usage is pipelined. For example, in cycle 3, routers R1 and R2 are free to receive new SSR requests and in cycle 4 they are free to get new flits.

Traversal with conflicts:

In Cycle 3, as described above, we noted that when the flit reaches router R2, it travels via both the transparent path and is latched at the router. The latching is required in case the flit is unable to transparently flow through to the next router due to inability to win SAG there. In that case, R2, where the flit is latched becomes the new initiating router and a multi-hop over multi-cycle traversal is attempted from R2 to the destination router.

Assumptions:

1. The design illustrated above assumes that either a) the wire delay analysis resulted in the number of hops per cycle being the same for the data (flits) and control (SSR) paths or b) the same hops per cycle for both data and control is enforced by clocking the system based on the slower of the two path.
2. The delay analysis and incorporating slack information into it are performed dynamically with appropriate hardware, inspired by prior work [3, 10].

5 Preliminary evaluation

We evaluated the benefit of the McMahon:vanilla design (with no slack recycling) and the McMahon:slack design (recycling an assumed and fixed slack of 40%) by implementing atop a SMART baseline on the GARNET2.0 [4] + Gem5 [5] simulator. We evaluate our design on a 16*16 mesh with XY-routing. We only evaluate with single-flit packets and the NOC runs at 1 GHz. We tested 3 synthetic workloads: Uniform Random, Bit Complement and Tornado. The results perform a sweep of HPCMAX (2, 4, 8) and varying injection rates.

Average flit latency of SMART, McMahon:vanilla and McMahon:slack are seen in Fig.3. For all 3 workloads, McMahon:vanilla enjoys a lower average flit latency than SMART and McMahon:slack's latency is even lower. The latency reduction compared to SMART is higher at lower HPCMAX (when SMART is less effective) and is significant across varying injection rates. Overall flit latency reduction for McMahon:vanilla and McMahon:slack in comparison to SMART go up to 23%/41%, 29%/49% and 26%/44% for Uniform Random, Bit Complement and Tornado respectively.

6 Future Work

We introduce new features which are in development.

6.1 Adding an SSR Network

SMART uses dedicated SSR links for signaling to downstream routers (that can be accessed in a hop) about the potential arrival of flits in the subsequent cycle. The dedicated SSR links suffer from 2 potential overheads.

First, the number of dedicated links requires grows as $O(HPCMAX^2)$, meaning a large amount of wires. Prior work [1] discusses potential wire overheads up to 75% and 1000% in SMART-1D/SMART-SD designs at a HPCMAX of 8. Further, the number of SSR signals to be sent for an N hop request grows as O(N), resulting in energy overheads.

Second, sending dedicated SSR signals to each router to indicate the future arrival of a flit may result in false negatives i.e. the flit may not arrive at an expectant router. This

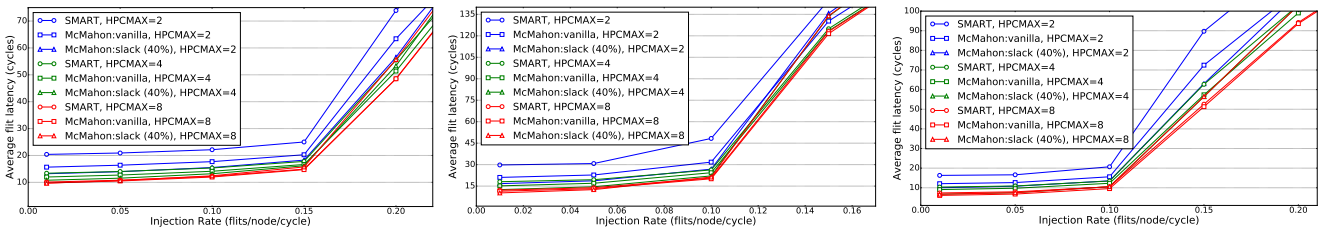

Figure 3. Latency analysis with synthetic traffic: (a) Uniform Random, (b) Bit Complement, (c) Tornado

can happen if that flit is forced to prematurely stop earlier due to some SSR interaction at prior inter routers that the current router is not aware of. Prior work [9] shows that false negatives can go up to 25-40% under certain schemes reducing network throughput.

We explore the opportunity in an SSR signalling network (instead of dedicated point-point SSR wiring). The SSR network consists of links between adjacent routers in X-Y directions (similar to the flit-network). SSR signals are sent to the appropriate routers similar to flits but arrive at routers 1 or more cycles before the flits. This will provide with three benefits - it will have low wiring/area overheads, reduces energy spent on signalling and also prevents false negatives at routers which improves NOC throughput. The SSR network will also carry slack information.

6.2 Flexible design with slack tracking

The key idea is that wire delays experienced by the data flit traversal network and the SSR signals will most likely be different. Thus, in this transparent flow based design, the arrival of the SSR signal at the router would not necessarily mean the arrival of the flit one cycle later.

To account for this, we continue to use the SSR network described earlier, but incorporate a slack/delay tracking mechanism. The delay tracking mechanism makes note of the types of links / crossbar paths in use, along with the local operation point and parameter variations and accumulates delay accordingly. It accumulates at each link what it estimates as the lag between the data flit and the SSR signal.

As and when the SSR signal reaches a router a decision is made on when to perform SAG. In the scenario that the data flit can potentially reach the router in the next cycle after the SSR arrival, the SAG is immediately performed once the SSR arrives - this is the same scenario as was seen in Fig.2 and section 4. In the scenario that the data flit lags behind the SSR signal by N cycles (N > 1), the SSR signal is buffered in for N-1 cycles (in a shift register). In this way the SAG is attempted only in the cycle prior to the data flit potentially arriving and not any earlier.

6.3 Maximizing throughput via link speedup

The proposed preliminary design will be unable to maximize latency gains for back-back flits - the issue to be solved is how to have data/signals flowing through back to back links in the same cycle, if the latch between the links is transparent. The potential hazard is that if there are two units of data along a path with no opaque boundary separating them, it is possible that some bits of the 2nd unit of data are able to race ahead of the 1st unit of data (a short path) and thus corrupting the 1st unit of data.

We are exploring a simple solution called "link speedup": duplicating links between routers. If every link between a pair of routers is duplicated, adjacent units of data can alternate between the pair of links. This way no adjacent data units will be using the same link, preventing any racing of data. At the input to a router, a 2:1 mux selects which link of the pair is set to pass via the router in any particular cycle. At the output of a router a 1:2 demux selects which link of the pairs should carry the data out in the current cycle. This allows highest throughput/latency to be achieved on adjacent data units at the cost of extra link wiring. In designs where the extra wires does not add substantially to overheads, this can be used to maximizing throughput.

7 Conclusion

McMahon extends the SMART [8] NOC to enable data to traverse multiple hops in minimum possible cycles in an efficient manner. Moreover, McMahon can further reduce flit latency by recycling slack in the interconnect network. The efficient data flow in McMahon is achieved via intelligent control of latches / flip-flops based on analysis of each flit's traversal time. We envision that rigorous evaluation and design optimization of McMahon along with implementation of the future work directions can enable a significant reduction in on-chip network latency.

References

[1] X. Chen and N. K. Jha. 2016. Reducing Wire and Energy Overheads of the SMART NoC Using a Setup Request Network. *TVLSI* (2016).
[2] A. Ansari et al. 2014. Route-oriented dynamic voltage minimization for variation-afflicted, energy-efficient on-chip networks. In *HPCA*.
[3] C. Lefurgy et al. 2011. Active management of timing guardband to save energy in POWER7. In *MICRO*.
[4] N. Agarwal et al. 2009. GARNET: A detailed on-chip network model inside a full-system simulator. In *ISPASS*.
[5] N. Binkert et al. 2011. The Gem5 Simulator. *SIGARCH Comput. Archit. News* (2011).
[6] P. Lotfi-Kamran et al. 2017. Near-Ideal Networks-on-Chip for Servers. In *HPCA*.
[7] S. Hanson et al. 2008. Exploring Variability and Performance in a Sub-200-mV Processor. *JSSC* (2008).
[8] T. Krishna et al. 2013. Breaking the on-chip latency barrier using SMART. In *HPCA*.
[9] Tushar Krishna. 2014. *Enabling dedicated single-cycle connections over a shared network-on-chip.* Ph.D. Dissertation. MIT.
[10] G. Ravi and M. Lipasti. 2018. Aggressive Slack Recycling via Transparent Pipelines. In *ISLPED*.