Network Interface Buffer Elimination

checkpoint 3

Jiong Xue Qilu Guo Jing Ji

Project overview

The project aims to eliminate the network interface buffer to reduce area for a NoC. We propose a solution where the packet delivered is preserved on the cache instead of the network interface. Thus, only the head and tail flits of the packet need be stored in the network interface. We modified the communication scheme that cache would directly transmit the data to and from the router. Both transmissions require interference with network interface.

A high-level architecture design is shown in Figure 1 below.

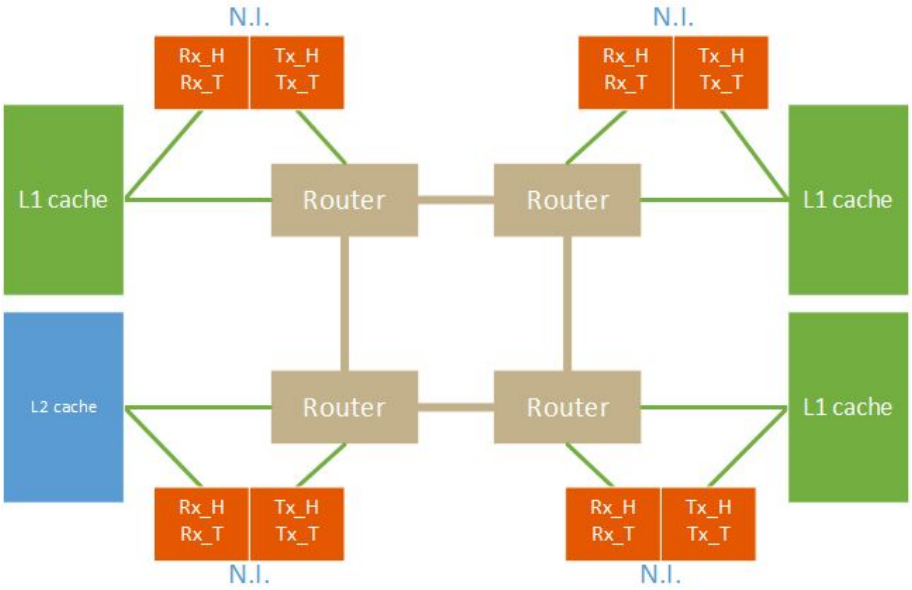

Figure 1 high-level architecture of our design

Progress so far

1. Completed the design and coding for the router, the network interface and the cache and assembled the system
2. Completed baseline design for network interface and built up the baseline
3. Thoroughly tested the network interface, the router and the cache for functional correctness
4. Generated memory access traces from 470 test cases and injected them into the testbench for performance evaluation
5. Synthesized the network interface and the router for area evaluation

Components Detail

Router

The flow control for the router is store and forward, which means that the head flit waits at the router until the entire packet is received. To support the new communication scheme between the cache and router, more states are added in the router.

Network Interface

We implement the network interface to only store two pairs of head and tail flits, corresponding to "send" and "receive" process respectively. The "send" process refers to sending data out of cache from this core to another, while the "receive" process is the opposite. The head flit containing adequate information to transmit one packet is composed of the (1) source and destination coordinates within the network, (2) the number of flits in one packet, and (3) the memory address of the leading data flit. The composition of head flit is shown in Figure 2. Data transmitted is assumed to be adjacent to each other and follow sequence order in address.

Figure 2. head flit in network interface composition

Cache

We implement the L1 cache as an N-way write-back, write-allocate cache. The L2 cache is implemented like a memory, meaning that it is capable of storing all the data coming from other caches and never needs eviction.

Data Transmission Protocol

When a cache miss happens on an invalid cache line, the cache will send an request to the processor to ask for the data from other caches. Since we do not implement the cache coherence protocol, the destination is determined by our simulation inputs. Then, the cache is blocked and it waits for the requested data to come back. When a cache miss happens on a valid cache line, eviction of that cache line is needed first. Hence, the cache sends an eviction request and the corresponding eviction address to the network interface. Then, the network interface will construct the head flit based on the eviction address. Once the head flit is sent to the router successfully, the network interface will signal the cache to start sending data flit by flit to the router. After the router receives all the data, it signals the network interface to send the tail to it. When the router gets the whole packet, it start to send the packet to the destination.

To receive a packet, the router first sends the head flit to the network interface. The network interface then deconstructs the head flit to obtain the memory address and signals the cache to receive the data from the router. Once all the data is received by the cache, the router will send the tail to the network interface to signal the end of the receiving.

Testbench Development

Since we don't include core in the design, it is difficult to run real tests from binary executable. Instead, we chose to generate memory access traces from 470 pipeline, and inject these traces into our testbench following timing constraints. To ease the burden, we assume every memory access depends on its preceding memory accesses. Thus, the delay time between memory accesses are kept the same as that in the 470 pipeline.

Evaluation

We injected memory access traces of same tests to both our design and the baseline. We measured the total execution cycles and synthesized the design for overall area. The results are shown in Figure 3.

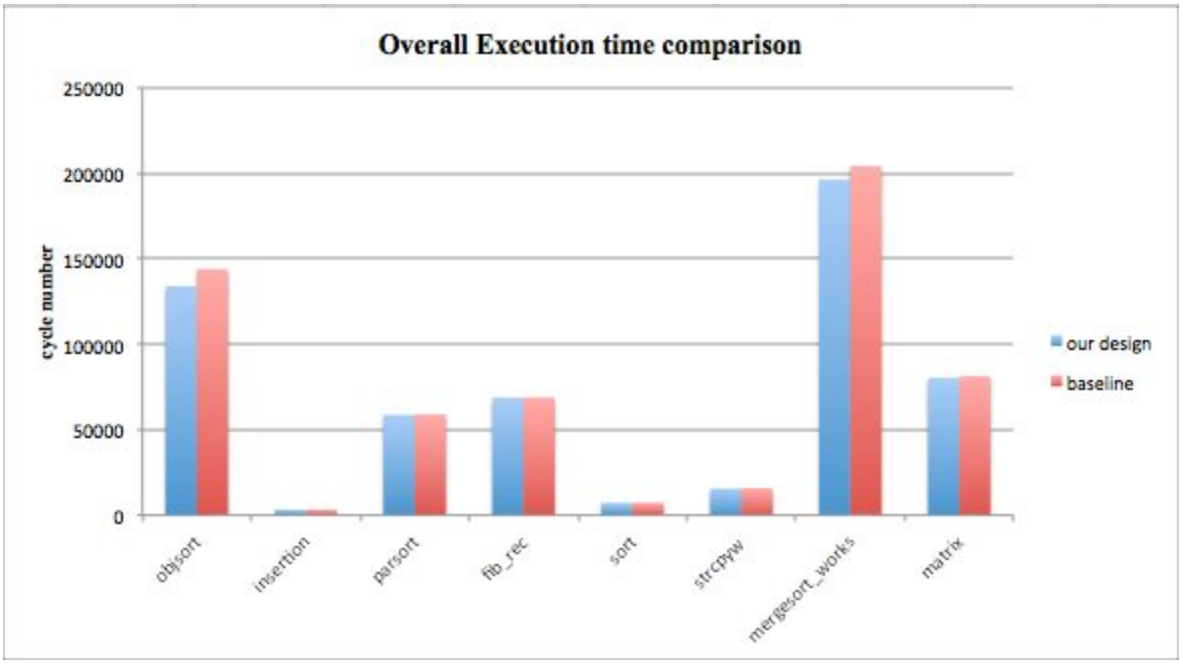

Figure 3. Overall Execution cycle time comparison between our design and the baseline

We can see that our design actually achieves better performance for all of these test cases and the maximum execution time reduction is up to 6%. This is because we did not include acknowledgement in the routing scheme. Thus by directly connecting cache to router, all data flits are transferred directly through router and they leave and enter the cache as soon as the transfer completes. On the contrary, in baseline design, all data flits need to go through network interface that introduces delay.

For area measurement, we expect to see an area reduction in our design compared to the baseline design. We can also see this result in Table 1.

Table 1. Area Analysis between our design and the baseline in um^2

Area Analysis			
module	**our design**	**baseline**	**percentage**
router	517620	512025	1.1%
network interface	36269	179558	-79.8%

The area of network interface decreases around 80% for eliminating data flit buffers, while the cache area does not change. The area of router increases 1.1% compared to the baseline design because in our design, the router need more ports to connect with the cache.

Next steps:

1. Add any design modifications and conduct more tests
2. Preparing for posters

Issue

1. Do we need to add acknowledgement in our routing scheme?
2. Do we need to run tests for three cores simultaneously for performance measurement?