Combined Centralized and Distributed Connection Allocation in Large TDM Circuit Switching NoCs

Yong Chen, Emil Matus, Gerhard P. Fettweis
Technische Universität Dresden, Vodafone Chair for Mobile Communications Systems
{yong.chen, emil.matus}@ifn.et.tu-dresden.de

Abstract—The centralized methods for connection allocation in a circuit-switched network-on-chip (NoC) based on time-division multiplexing (TDM) may pose serious performance and scalability issues in large-scale networks due to the 1) limited path search speed, 2) increasing allocation request rate at central unit and 3) the increasing communication cost between the central unit and NoC nodes. This paper tackles this problem by proposing combined centralized-distributed approach that splits the original NoC into multiple non-overlapping logical partitions, and each of them served by a dedicated NoC-Manager unit. NoC-Manager employs fast trellis-search shortest path algorithm enabling local path search inside the associated NoC partition, while a set of NoC-Managers jointly combine the partial results in distributed manner in order to find the most likely global path. This approach attempts to combine the benefits of distributed and centralized systems and the experimental results demonstrate its high potential regarding performance and scalability improvement.

I. INTRODUCTION

The Network-On-Chips (NoCs) can provide Guaranteed Services (GSs), such as minimum throughput or bounded latency. Circuit Switching (CS) is frequently adopted to support GS, which allocates exclusively links to form a dedicated connection for each flow. However, since the resources (link and buffer) is occupied exclusively during the entire lifetime of a connection, it is inflexible and may block other communications. Therefore, an extension, Time-Division Multiplex (TDM) CS, was proposed to share the resource. In TDM CS NoCs, the link is split into time slots and shared through TDM scheme, with the allocation information stored in a slot allocation table of particular router [1].

The TDM CS enjoys contention-free low latency communication with the cost of connection setup overhead, which has to find the shortest contention-free path between two given modules and allocate the time slots along the path. Due to the global knowledge of the system, the previous centralized approaches [2]–[5] for connection allocation (i.e. connection setup) provide good performance for small and moderate NoCs. However, the more NoC size grows, the more allocation requests are received by the central unit per cycle, and the more communication cost is risen between the central unit and NoC nodes. Consequently, the central unit can become a serious bottleneck in hotspot traffic. In this paper, we proposed a scalable mechanism to address the dynamic connection allocation problem in large systems. The dedicated hardware connection allocator, ‘NoCManager’, is presented to solve the allocation problem. The partitioned architecture (i.e. spatial partitioning technique) is used to overcome the scalability problem in traditional centralized systems. NoC is divided into small differentiated logical partitions contained local NoCManagers. This partitioning technique keeps the request load of the manager and manager-node communication overhead moderate. Inside each partition, the path search problem is solved by a local manager with trellis-search algorithm. To establish a path that crosses partitions, the managers communicate with each other in distributed manner to converge the global path. Hence, good scalability and high performance can be achieved at the same time.

II. BACKGROUND AND RELATED WORK

There are two categories of dynamic connection allocation techniques i) centralized [2]–[5] and ii) distributed allocation [1], [6]. In centralized system, a central manager is responsible for connection allocation. Since the central manager has the global knowledge of the system, it can achieve global optimal results. The central manager can be a embedded processor or a dedicated hardware accelerator. HAGAR [4] is a hardware accelerator, which supports basic CS and allows to speedup allocation by two orders of magnitude against Microblaze processor based software methods [3]. The TrElliS-Search Allocation (TESSA) approach [2] proposed a high speed dedicated allocator for TDM CS. In paper [5], a centralized hardware unit that uses breadth-first searching algorithm is proposed with excellent performance. Though the centralized system has the advantages of global knowledge and high performance, as the network grows with increasing allocation request rate at central unit, the central unit might become the bottleneck due to the drawbacks of centralism in computation and communication. This paper is motivated by TESSA, but we proposed the partitioned architecture that divides the large system into small logical partitions with multiple dedicated local managers to enhance the scalability.

III. SYSTEM MODEL

The system model of central allocator (i.e. NoCManager) based 2x2 mesh NoC architecture is illustrated in Fig. 1. The NoCManager (NoCM) is the dedicated unit that attempts to allocate the appropriate connections when it receives GS connection requests. The messages between NoCM and NoCs are delivered over a light-weight dedicated control network.

As soon as a module needs a connection to another module, it sends the request of connection reservation to the NoCM. When NoCM receives the request, it attempts to allocate the
appropriate path and slots according to the request requirements. When the allocation succeeds, NoCM sends the results on allocating information to the relevant routers in order to set up the connection. In the NoC, when the data transfer phase is finished, the source node sends deallocate flit to delete the connection, and informs NoCM to free the corresponding allocated links.

IV. NOCMANAGER ARCHITECTURE

The NoCMANAGER has to search and allocate connections between two modules. In our system, the connection allocation problem is solved as the shortest path problem in a trellis graph description of the NoC.

A. TESSA Approach

The TESSA (non-partitioned TESSA) was initially presented in [2] but without analysis on the performance in large systems. The successive NoC traversal from source to destination node for path searching can be represented by trellis graph (Fig. 2). In general, the path search in trellis graph comprises two steps:

- **Forward search** i.e. traverse the NoC from source to find the best free path to target node.
- **Backtracking** i.e. when target node is reached, the backtrack starts from target node to read out the stored survivor path hop by hop.

The trellis graph in Fig. 2a is defined to represent a traversal through the network of single hop. Normally, iterative traversals are necessary as the distance between the source and destination nodes is more than one hop. Hence, there is ‘iteration link’ from second stage to first stage and additional registers in order to hold the values of intermediate results(Fig. 2b). Each iteration consumes a clock cycle. The search will be stopped in two cases: i) either the target node has been reached or ii) after certain iterations (i.e. $2N - 2$ iterations for NxN mesh NoC, which equals the longest minimal path in the network). Hence, the livelock is avoided. As in Fig. 2b, the forward search starts from Src (node 0), and activates node 2 and node 1. At the next cycle, node 2 travels back to its first stage, and activates node 3. Then backtrack starts from Des (node 3) to select path sequence as {0, 2, 3}. The link 1 → 3 is assumed already occupied by previous allocation.

B. Partitioned TESSA

To address the scalability issue, we proposed the partitioned architecture that divides the original system into multiple small logical partitions with a local dedicated NoCM per partition. Each local manager only has the state of nodes in its region, and is responsible for searching path in its local region. The NoCMs are connected with each other in 2D-mesh topology via dedicated links. The dedicated link width is $2 \cdot \log_2 M + N \cdot S + 2$ bits, where $M$ is the NoC size (for Des and Src address), $N \cdot N$ is partition size (for border nodes) and $S$ is slot table size, and 2 bits for control signals. These NoCMs can work simultaneously.

1) **Partitioned TESSA search algorithm:** If the source and destination nodes of the request are both at the same partition, this request will be handled by the local manager only. Otherwise, as illustrated in Fig. 3, firstly, the local NoCM (NoCM_A) starts to search path in its trellis graph from source node (Src) to reach the border nodes (node A, B and C), and then forwards the search message to its neighbor NoCMs (NoCM_B and NoCM_C), continually until reaches
the destination (NoCM_D). When destination (Des) node is reached, backtrack starts from Des to select the survivor path.

The basic search idea among NoCMs is similar to parallel probe search [6], as shown in Fig. 4. The Src NoCM requests search to its neighbor NoCMs, which can lead closer to the destination. The reached neighbor performs trellis search in its partition, and then forwards the message to its reached neighboring NoCMs. Hence, the search is forwarded to the Des along all possible minimal paths. If two searches meet at one NoCM, then one of them will be canceled based on RoundRobin arbitration. If the downstream NoCM is not available, then that probe search will be canceled immediately without waiting.

2) Detailed partitioned NoCM architecture: The block diagram of the NoCM is shown in Fig. 5 and comprises trellis graph and control modules (routing module and control logic).

The ‘Search comes in’ indicates a new probe search comes in, and Nack/Ack are backward answer signals (Ans). The detailed control signals usage during probe search is illustrated in Fig. 6. For each NoCM, if it accepts a search, it will become busy and reject any later search until becomes free again. Since each search may have two productive output directions and may send out two searches, a counter (search_cnt) is used to record the number of sent out searches. Hence, the value of the counter will be decreased when it receives an Ans signal from the downstream NoCM. The NoCM sends Ack to its upstream NoCM as long as a Ack signal from downstream NoCM is sent back, while sending out Nack only when received all Ans signals (search_cnt=0) and they are all Nack. The busy NoCM will become free again in two cases: i) either receives the Ack or ii) receives all Ans signals (search_cnt=0).

Control trellis graph for search in each partition: If the NoCM is reached by a probe search, it will start the path search in its trellis graph to search the path inside its region. The path backtrack in trellis will start in two cases: i) either an Ack is sent back, or ii) the Des node is reached in this trellis graph. In forward probe search, message of the reached border nodes will be sent to the corresponding downstream NoCMs; and in backtrack, only that of the backtracked border node needs to be sent to the upstream NoCM. Each NoCM sets up the connection in its region.

Ensure the destination only be activated once for one request: The searches belong to the same request but along different directions might reach the Des at different time. The request ID can guarantee the Des is activated only by the first search. Each Des NoCM has a table that stores the ID of last request from each Src NoCM. So when the Des NoCM receives a search, the ID of the received search is compared to the ID of last search from that Src NoCM. If these two IDs are the same, it means they belong to the same request and the newly received one will be rejected. Otherwise, the new search is accepted and the ID table is updated.

Since the forward probe search is along minimal paths and does not wait, the livelock and deadlock is avoided.

V. Performance Evaluation

A. Synthesis Results

Using Synopsys DesignCompiler, the NoCM was synthesized with TSMC 65 nm technology. For partitioned TESSA, in 18x18 mesh, it is divided into 4 partitions (four 9x9 mesh partitions) or 9 partitions (nine 6x6 mesh partitions); in 16x16 and 20x20 meshes, it is divided into 4 partitions or 16 partitions.

The area consumption is illustrated in Fig. 7. It might be surprising that the total area of partitioned TESSA is less than non-partitioned TESSA. The reason is that in non-partitioned TESSA, single trellis graph contains the whole network, so its trellis is much larger than that of partitioned architecture, which requires more effort for wiring and more flip-flops to distinguish different nodes. On the other side, in partitioned TESSA, the more partitions we have, the more control logic is required, so using more partitions brings the increase of logic area. Therefore, the partitioned TESSA with 4 partitions cost the least area.
B. Simulation Results

The allocation speed and success rate of partitioned NoCMs are compared to the centralized and distributed allocation techniques under uniform random traffic. We re-implemented a distributed parallel probe connection setup approach according to Liu’s work [6] for comparison. The request queue sizes of non-partitioned NoCM, single 9-partition NoCM and single 4-partition NoCM are 64, 7 and 16, respectively.

For evaluation several performance metrics are used:

- **success rate** denotes the ratio of successful requests that established paths to the total requests. The incoming requests will be discarded as failure if the request queue is full.
- **total allocation time** denotes the number of clock cycles for single request that the algorithms need to set up the path or to determine the allocation is impossible, additionally the waiting time for a request in the queue.
- **GS offered load** refers to the ratio of the GS traffic each master offered to its maximum capacity.

In the simulation, half nodes are GS masters and half are slaves. These nodes are uniformly randomly distributed in the system. The connection lifetime, i.e. the number of flits that each connection delivers, is set as 2000 flits, 3000 flits and 4000 flits. The retry deadline of parallel probe search [6] is set to 3000 cycles.

1) Comparison of allocation speed: In parallel probe search, multiple trials might be needed before success of search due to investigating single slot at a time. On the contrary, in TESSA, all slots are searched in parallel. As shown in Fig. 8, compared to parallel probe search, the partitioned TESSA offers 7X to 71X higher allocation speed, and it offers 48% to 72X higher speed against non-partitioned TESSA. We can see in non-partitioned TESSA with low request rate, e.g. low offered load (offered load from 0.1 to 0.2) or longer connection lifetime (4000 flits), the allocation time is reduced significantly. In high request rate, the reason is as follows. The non-partitioned NoCM is too busy that the incoming requests have to wait longer in the queue in order to get their turn. However, in partitioned TESSA, since there are multiple NoCMs working in parallel, the requests are usually processed in time without wait, thereby the allocation speed is much higher.

2) Comparison of Success Rate: The simulation results of success rate is illustrated in Fig. 9. For partitioned TESSA, the systems that are divided into 4 partitions achieve the best success rate than 9 or 16 partitions. Compared to parallel probe search, the 4 partitions TESSA can offer up to 55% higher success rate in 16x16 mesh. In non-partitioned TESSA with high offered load, many incoming requests must be rejected as failure immediately because of the full request queue even though there might be free paths for these requests. Hence, compared to non-partitioned TESSA, the 4-partition TESSA can provide up to 85% higher success rate in 16x16 mesh and up to 75% higher in 20x20 mesh. However, when offered load is very low, e.g. lower than 0.3 in 16x16 mesh, the single NoCM can handle all the requests in time, so the success rate of non-partitioned TESSA becomes the best.

VI. Conclusion

To address the scalability problem of traditional centralized systems for CS connection allocation, we proposed the partitioned architecture, which divides the original system into multiple partitions with multiple local managers. Since the managers work simultaneously, the computation capacity is increased. As the NoC nodes only communicate with their local managers, the communication overhead is mitigated. The evaluation results also suggest how to divide the system, that as long as the managers can handle the requests in time, the system with less partitions (i.e. larger size per partition) will offer better performance.

ACKNOWLEDGMENT

This research work was funded by EuroServer project (FP7-ICT-610456) from the European Union 7th Framework Program and partially supported by China Scholarship Council.

REFERENCES