DESIGN OF SHARED BUFFER ARCHITECTURE FOR CPU-GPU ON CHIP NETWORK

Deepika Pitliya

Department of Electronics and Communication Engineering,
R V College of Engineering, Bengaluru, India

Namita Palecha

Department of Electronics and Communication Engineering,
R V College of Engineering, Bengaluru, India

ABSTRACT

To achieve high-performance and energy optimized computing the GPU-CPU heterogeneous architectures are standout choice. Streaming multiprocessors (SMs) are increasing to boost throughput in GPUs. Design of on-chip interconnect for GPU-CPU diverse system is a challenge to make it scalable and efficient. Mesh network is being used in manycore CPUs but for GPU it consumes more area and power as well, due to traffic pattern of GPU. Crossbar is good fit, but it is not supporting communication among the SMs when numbers of SM are high. The motivation is to design the scalable crossbar which provide communication between SMs and SM to memory unit. The objective here to design two types of crossbar with shared buffer, crossbar local and global. Crossbar local provides communication among SMs and take all the input request which are going to the memory in coincide manner and pass it to crossbar global that divaricate these request to memory unit, Last-level cache as well as memory controllers. Sharing buffer give opportunity to all input for communication way efficient to achieve high throughput with reduce area and power. Compare to mesh network in Shared buffer crossbar network reduction in area 28% and power 32%. Scalability of design is verified by increasing the number of SMs.

Key words: Crossbar, Graphical processing unit, Buffers.

Cite this Article: Deepika Pitliya and Namita Palecha, Design of Shared Buffer Architecture for CPU-GPU on Chip Network. *International Journal of Electrical Engineering and Technology*, 11(4), 2020, pp. 368-376.

<https://iaeme.com/Home/issue/IJEET?Volume=11&Issue=4>

1. INTRODUCTION

Designing on chip network for GPU-CPU diverse system is sustainable challenge. CPUs enhance optimization which lessen latency but hardly queuing the resources. In GPUs there is lot of congestion occurs due to massive traffic that degrade performance of CPU. High number of SM create congestion in the connection of SM to memory controllers and last level

cache [1]. In CPUs system generally mesh, butterfly and clos used which provide communication among CPUs but not suitable for GPUs, because these topologies traffic pattern is many to few and few to many [2,3]. There is requirement of interconnect which provide communication among SMs and SMs to memory controller and last level cache.

Here motivation is to design shared buffer crossbar architecture for GPU-CPU heterogeneous system [4]. Buffers are being used on routers at input/output port to store the packet temporarily. Sharing buffer on network because every input port in the routers is not having input packets that is required to transfer at concurrently [5]. Routers, buffers and links are the important components of the Network on chip. All these elements connect the input and output ports of neighboring cores. For each router, the input buffers and the crossbar switch are the major components and consuming chip area and power. As the number of ports increases, the associated buffers, allocation logic and crossbar increase so area also increase. There is need to construct a scalable and efficient network for many core systems. which provide the connection between all the cores and all the input/output ports. Buffers occupy more area in network design, so sharing buffer gives opportunity to data packets to utilize the buffers when traffic is high by sharing them Because not all core transfer the data so if traffic at one core is high and other core is low then by sharing buffer space packet can be transfer. There is no data loss even traffic is high.

Here for crossbar designing two switching nodes, crossbar local and crossbar global rather using the conventional crossbar which is fully connected. The idea here is not all the processor wants to communicate to memory unit simultaneously for example SMs to MCs and LLCs [6,7]. Rather than connecting all SM to memory unit, crossbar local takes the request from SM and pass it to crossbar global in converged manner then these requests are diverging to memory unit by using crossbar global.

The whole thing can be analyzed by taking one example, fully connected type crossbar in which 40 SMs that are connected to 8 LLC means all 40 is connected to 8 LLC. If taking same example in shared buffer crossbar design there are 5 crossbars local, each one connects 8 SMs input side to 3 outputs that will be input to crossbar global. 5 crossbars local converged 3 output per port all combine 15 input for crossbar global, so these 15 requests coming from SM to memory unit through crossbar local and then crossbar global and passing to 8 LLC. So observation is rather connecting all 40 requests to 8 LLC here 15 requests to 8 LLC as not all processor wants to process memory at same time. All SMs are grouped through crossbar local so communication can happen among SMs with the help of crossbar local and crossbar global.

2. DESIGN METHODOLOGY

The design methodology of network for many core systems is shown in Fig. 1 where multiple SMs grouped in crossbar local.

All crossbar local grouped and connected to crossbar global.

Figure 1 Design methodology for crossbar design for GPU-GPU system

2.1. Crossbar Local & Global

There are two types of crossbar, combination of two crossbar gives the complete functionality, communication between SMs and SMs to memory unit. These two types of crossbars give the opportunities to utilize the networks thoroughly. All SMs are grouped to crossbar local, so there are several numbers of crossbar local that again depends on the number of SMs and how grouping is done. Crossbar local provide the communication among SMs and if any request from the SM to memory then passes the request to crossbar Global in coincide manner. The key benefit here is converging the memory request from SMs to crossbar global through crossbar local.

The crossbar global provides the connection for the request coming from the crossbar local to memory unit, so crossbar global provide the communication between SMs to memory with the help of crossbar local. The crossbar global is connecting all local crossbar, MCs and LLCs. Hence gives the opportunity to all SMs to communicate each other.

2.2. Buffers

Buffers consume router area but to remove contention and no packet loss. There is need to store packet temporarily at input/output port. Wormhole switching technique applied here in which the credit-based flow being used, which provides the assurance for all the packets that there will be the space in buffers, so no packet loss.

2.3. Round Robin Algorithm

In all crossbar for routing algorithm Round – robin being used, it considers all of the packets which are sending to converge port hence contention minimized. Here for example in crossbar four inputs are sending packets and at output side two converge ports. First algorithm chooses two out of four packets considering first come and first serve method and send to at two converged port following round robin manner. It minimizes contention of flits. Fig. 2 shows flow chart of round robin microarchitecture. time quantum which is define as time amount share provide to each process in identical portion. There is cyclic order and it give opportunity to all the process considering time quantum and burst time.

Figure 2 Round Robin Algorithm

2.4. LLC & MC

GPU offers enormous thread-level parallelism. Earlier GPU and CPU used to connect through PCI bus. CPU offloads data into GPU where it explicitly copy to memory, but explicit copy significantly gives performance overhead. To improve programmability and reduce communication cost integrate both on same network. Sharing the Last Level Cache and memory controllers on network on chip.

2.5. Scalability

Scalability is another feature of the design. In case if the requirement is increment in the numbers of SM in the design there are two choices to scale the design. First option increase numbers of the SMs per crossbar local where output converge port remain same for crossbar local, only input number change as number of SM increases, so there is no scaling issue for crossbar global because number of crossbar local and output of crossbar local still same only input of crossbar local changes for increase in SMs. Second option if there is increase in SMs, increase number of crossbar local, in this case as increase in crossbar local which effect the crossbar global because input for crossbar global increases as number of crossbar local increases. So, the former choice is better option because in second choice scaling issue for crossbar global.

3. IMPLEMENTATION

3.1. GPU-GPU Homogeneous System

In GPU- GPU homogeneous system design with 24 SMs, 3 crossbars local with 8 SMs clustered per crossbar local. Crossbar local converging 8 inputs to 2 outputs at outputs port which are the inputs for crossbar global. Crossbar global is bridge between crossbar local and memory unit. In this design crossbar global is taking 6 inputs which are coming from crossbars local and diverging to 3 outputs to memory unit memory controller and last level cache.

In this Fig. 3 it is shown the SMs are connected through crossbar local and crossbar global being used to connect all crossbar local and memory unit. Routing algorithm round robin is giving prospect to each input for communication. Crossbar global is diverging the memory request which is coming from memory. So here rather passing all 24 request to all memory unit the design converge the 24 requests to 6 because each crossbar local output is 2 so all 3 crossbar local together gives 6 output that is again going through crossbar global hence output of crossbar global is 3. Hence out of 24 finally only 3 requests are going from SMs to memory unit.

Buffers are used for network interface in each network so packet can be store temporarily until the previous data transfer not complete. In this way there is minimum packet loss in the network. In routers more than 60% of area consume by buffers and if buffer less routers used in network then poor performance. So, buffer play vital role in networks. Generally, in the network not all the port having data transfer simultaneously, some ports sending more packets and while others are not sending single one, in this condition there are higher chances of packet loss, since at the port where traffic is high there is probability of packet loss.

Figure 3 Crossbar design for GPU-GPU system

3.2. GPU-CPU Heterogeneous System

GPU-CPU heterogeneous system is most promising design in today world. GPU does intensive computation on huge data and CPU does the task in serial manner. Here in this design there are three crossbars local in which two crossbars local is for GPUs and one crossbar local for CPU. There are 16 SMs total in two crossbars local and 8 processor equivalent (PEs) for CPU crossbar local. In GPU parallel communication is there among SMs through crossbar local and in CPU serial communication among processor.

There are three crossbars local where two crossbar local functioning is same as the previous design gives the communication among SMs and converge the request for memory to crossbar global which is coming from SMs. Third crossbar local functionality used to provide the serial communication in between PEs. This design provides parallelism of GPU and the serialization of CPUs. Here integration of GPU and CPU done with sharing memory controller and last level cache.

3.3. Mesh Topology GPU-GPU Homogeneous System

In 5 x 5 Mesh topology design shown in Fig. 4. In $m \times m$ mesh topology m^2 routers are there and per router there are 5 input/output ports. In mesh design there is unique traffic pattern, where data traverse from source node to destination in unique pattern either horizontal or vertical direction. Router design based on virtual flow base control.

Figure 4 Mesh 5 x 5 network for GPU-GPU system

In mesh network the data injected from the processing nodes and route based on deterministic routing algorithm. Each router has five input/output port and corresponding buffers, crossbars and switches. In mesh network the data transfer also in unique fashion, its work in vertical and in horizontal direction.

4. RESULT & DISCUSSION

The shared buffer crossbar network design functionality shown in Fig. 5. There are various stages in network on chip. Data packet injects from source and crossbar routes packet based on round robin routing algorithm. At input/output units of network, buffers are placed to store data temporary.

Figure 5 Design functionality of Shared buffer crossbar network

In this section discussion on functional verification of shared buffer crossbar architecture. Crossbar local contains 8 SMs at input side and 2 converged output. The input/output signals of crossbar local shown in Fig. 6 where packet injected at SM1 in flit format. The injected packet is requesting for communication with memory. So, crossbar local according to round robin routing it checks the availability and grant the signal. The output is available for further transmission at global crossbars.

Figure 6 Output of crossbar Local

Other crossbars local function in similar fashion and converged output of all crossbar goes to crossbar global. Here 3 crossbars local designed and each crossbar local has 2 converged output so total 6 outputs. Crossbar global 6 input and these inputs are again diverging to memory unit last level cache and memory controller. Functionality of crossbar global shown in Fig. 7 there are 6 input data coming from the crossbar local and after passing various intermediate stages like buffers and routing function. Crossbar global grant the upcoming signal and pass it to memory unit. In this crossbar global connect all the crossbar local and all the memory controllers and last level cache. As not all SMs requests for memory unit. In GPU the traffic pattern is many to few so only one or two out of 8 SMs requests for memory that's the reason to use the crossbar global and crossbar local so rather doing more connections between all nodes requests are converging here.

Figure 7 Output of crossbar Global

Power Analysis of various network shown in Fig. 8. Where the shared buffer crossbar and mesh topology for 24 SMs and 8 SMs calculated. Network power for mesh network is more than shared buffer crossbar because in mesh network there are more input/output ports and more buffers, so numbers of links increases hence power become high compare to shared buffer crossbar.

Figure 8 Power comparison of the networks

Area report for shared buffer crossbar and mesh network shown in table 1 and table 2 for 24 SMs and 8 SMs respectively. Area of shared buffer crossbar network is less compared to mesh network because in shared buffer crossbar, rather using natural crossbar there are two crossbars with shared buffer. Crossbar are converging and diverging the requests so there are less input/output units and less buffers which is also shared so very less area consumes in comparison with mesh network.

Table 1 Area report of Shared buffer crossbar and mesh networks for 24 SMs.

Network topology	No. of Slice	No. of FF.	No. of Luts
Mesh Network	634	335	1090
Shared buffer crossbar	497	252	840

Table 2 Area report of Shared buffer crossbar and mesh networks for 8 SMs.

Network topology	No. of Slice	No. of FF.	No. of Luts
Mesh Network	211	110	364
Shared buffer crossbar	165	84	283

5. CONCLUSION

The work proposes shared buffers crossbar design in which for crossbar there are two switching nodes as crossbar local and crossbar global with shared buffer. Compare to natural crossbar which is fully connected the shared buffer crossbar design gives the flexibility for connection between SMs and the memory unit. Here crossbar local converging the requests coming from SMs and then diverging through crossbar global with sharing the buffers, so packets can share buffer space and maximum utilization of buffers. In shared buffer crossbar architecture reduction of area is 28% and power 32% in compare to mesh network. Mapped Scalability analysis of design in which more SMs can be added in existing design. Shared buffer crossbar architecture is highly scalable, flexible, high-performance optimized network in terms of area and power compare to others network

REFERENCES

- [1] Xia Zhao, Sheng Ma, Zhiying Wang, CD-Xbar: (2019) A Converge-Diverge Crossbar Network for High-Performance GPUs IEEE Transactions on Computers (Volume: 68 , Issue: 9 , Sept. 1)
- [2] Lulwah Alhubail and Nader Bagherzadeh, (2019) "Power and Performance Optimal NoC Design for CPU-GPU Architecture Using Formal Models" Design, Automation & Test in Europe Conference & Exhibition
- [3] V'ictor Garc'ia; Juan G'omez-Luna Thomas Grass (2016) "Evaluating the Effect of Last-Level Cache Sharing on Integrated GPU-CPU Systems with Heterogeneous Applications" in IEEE International Symposium on Workload Characterization (IISWC)
- [4] U. Milic, O. Villa, E. Bolotin, A. Jaleel, A. Ramirez, and D. Nellans, (2017) "Beyond the Socket: NUMA-aware GPUs," in Proceedings of the International Symposium on Microarchitecture (MICRO), pp. 123–135.

- [5] Xia Zhao, Sheng Ma, (2016) “A Heterogeneous Low-Cost and Low-Latency Ring-Chain Network for GPGPUs” in IEEE 34th International Conference on Computer Design (ICCD)
- [6] A. Bakhoda, J. Kim, and T. M. Aamodt, (2010) “Throughput-Effective On-Chip Networks for Manycore Accelerators,” in Proceedings of the International Symposium on Microarchitecture (MICRO), pp. 421–432.
- [7] Anh T. Tran, and Bevan M. Baas, (2013) “Achieving High-Performance On-Chip Networks with Shared-Buffer Routers” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems 1 DOI10.1109/TVLSI.2013.2268548
- [8] Khalid Latif, Tiberiu Seceleanu, Hannu Tenhunen, (2010) “Power and Area Efficient Design Of Network-OnChip Router Through Utilization Of Idle Buffers” Department Of Information Technology, University of Turku, Finland. IEEE.
- [9] A. T. Tran and B. M. Baas, (2011) “RoShaQ: High-performance on-chip router with shared queues,” in Proc. ICCD, Oct. pp. 232–238.
- [10] L. M. Ni and P. K. McKinley, (1993) “A Survey of Wormhole Routing Techniques in Direct Networks,” Computer, vol. 26, pp. 62–76.
- [11] Yash Sarvankar, Rohan Ranshinge and Shubham Tikare, (2016) Eye Controlled Wheelchair Using ARM Cortex-A7 CPU. International Journal of Computer Engineering and Technology, 7(4), pp. 73–80
- [12] M. Prashanth Reddy, C. Labesh kumar and T. Vanaja, VVSH Prasad and Dr. K Ashok Reddy, (2017) Thermal Modeling and Manufacturing of Heat Sink for Cooling CPU, International Journal of Mechanical Engineering and Technology 8(9), pp. 502–509