**Design of Low Area Interconnect Architecture for CPU GPU Network On Chips (Nocs)**

 **N. NITHYA, M.E.,**

 **Department of VLSI Design,**

 **Tnks1906@gmail.com**

**Abstract** - The main objective of this paper is to develop a new approach for round robin CPU scheduling algorithm which improves the performance of CPU in real time operating system. The proposed Priority based Round-Robin CPU Scheduling algorithm is based on the integration of round-robin and priority scheduling algorithm. It retains the advantage of round robin in reducing starvation and integrates the advantage of priority scheduling. The proposed algorithm also implements the concept of aging by assigning new priorities to the processes. Existing round robin CPU scheduling algorithm cannot be implemented in real time operating system due to their high context switch rates, large waiting time, large response time, large turnaround time and less throughput. The proposed algorithm improves all the drawbacks of round robin CPU scheduling algorithm. The paper also presents the comparative analysis of proposed algorithm with existing round robin scheduling algorithm based on varying time quantum, average waiting time, average turnaround time and number of context switches. The field of study in CPU-GPU multicore is a hot topic in the semiconductor research and industry, however the challenges mentioned in the literature are still not been addressed yet. In this project, we aim to focus on designing a unique heterogeneous crossbar style network-on-chip (NoC) to connect heterogeneous CPU-GPU processors.

**1** INTRODUCTION

CPU-GPU heterogeneous systems are emerging as architectures of choice for high-performance energy-efficient computing. Designing on-chip interconnects for such systems is challenging; CPUs typically benefit greatly from optimizations that reduce latency, but rarely saturate bandwidth or queueing resources. In contrast, GPUs generate intense traffic that produces local congestion, harming CPU performance. While the level of heterogeneity in the system can introduce interconnection architecture designs with require relatively large circuit area and power for communication circuitry.

2 CROSSBAR NETWORK TOPOLOGY

Crossbar network a low any processor in the system to connect to any other processor or memory unit so that many processors can communicate simultaneously without contention. A new connection can be established at any time as long as the requested input and output ports are free.

 Fig.1.1Crossbar network ktopology

Crossbar networks are used in the design of high-performance small-scale multiprocessor in the design of routers for direct networks, and as basic components in the design of large-scale indirect networks. A crossbar can be defined as a switching network with N inputs and M outputs, which allows up to min {N, M} one-to-one interconnections without contention, shows an N × M crossbar network. Usually, M = N except for crossbars connecting processors and memory modules.

3 SYSTEMS-ON-A-CHIP

The Systems-on-a-Chip (SoCs) is a new category of systems which have emerged during recent years. In such systems processor cores and other system components available as intellectual properties (IPs) are integrated on a single chip. This IPs (CPUs, DSPs, memories, peripherals, etc.) incorporate more reusability and flexibility in the design since they can be quickly customized and integrated into multiple design projects. One SoC example is shown in Figure These heterogeneous IPs can be micro-processor, data memory, multimedia decoder and general peripherals. They mainly communicate with each other via an on-chip bus. Several industrial bus standards are available for SoCs, such as ARM AXI, IBM Core Connect and Wishbone.

 

Fig:2. Bus centric SoC Communication

Combining several programmable devices. Owing to the high number of cores that require communications between them it is not feasible to use a single shared bus or a hierarchy of buses. The multiprocessor system-on-chip (MP-SoC) uses multiple CPUs along with other hardware subsystems to implement a system. A wide range of MP-SoC architectures have been developed over the past decade. MP-SoCs incorporate an essential and distinct branch of multiprocessors.

4. BASE LINE ARCHITECTURE

CPU-GPU heterogeneous systems are emerging as architectures of choice for high-performance energy-efficient computing. Designing on-chip interconnects for such systems is challenging; CPUs typically benefit greatly from optimizations that reduce latency, but rarely saturate bandwidth or queuing resources. In contrast, GPUs generate intense traffic that produces local congestion, harming CPU performance. While the level of heterogeneity in the system can introduce interconnection architecture designs with require relatively large circuit area and power for communication circuitry. In baseline CPU-GPU systems, each tile contains a network router enabling NoC interconnections. Because of L1 concentration, GPU tiles incorporate 4 GPU cores and 4 L1s in each tile. The router hence enables inter communication for GPUs within tile and Intra communication with MCs. At same time router enables the communication for CPU tile with MC tiles.

  FIG:3. Baseline Architecture

5. PROPOSED SYSTEM

The proposed architecture consists of three types of crossbars, a static crossbar, a local crossbar and a global crossbar. Static crossbar is used only for CPU based traffic and Local crossbar is used for GPU-GPU communication. Local crossbar converges input ports from the GPUs into so-called converged ports which offers routing path diversity. The global crossbar diverges these converged ports to the last-level cache (LLC) slices and memory controllers. Priority based Round-robin routing balance network traffic among the converged ports within a local crossbar and across crossbars. Transforms the computational power of a modern graphics accelerators shared pipeline into general-purpose computing power. The memory controller consists of three types of interface generation. This module is a primer for in-depth looks at the different interfaces used in the Power QUICC II processor. In completing this section, you will have a basic understanding of the three memory interfaces utilized by the memory controller: Standard chip select generation for SRAM, ROM and basic devices only requiring a chip select. An SDRAM controller specifically for providing the correct memory control signals, connections, and timing for SDRAM devices.



 Fig:4. Proposed Architecture

This module gives you a block diagram overview of the memory controller as well as the basic operation and finishes with an in depth look at the registers. Each of the interfaces are covered in subsequent section **CPU** is the abbreviation for central processing unit Sometimes referred to simply as the central processor, but more commonly called a processor, the CPU is the brains of the computer where most calculations take place. In terms of computing power, the CPU is the most important element of a computer system. Graphics Processing Units have been used for general purpose computation for more than a decade. GPU computing has made it possible to exploit massive degrees of parallelism.

6 PRIORITY BASE ROUND ROBIN ALGORITHM

The CPU scheduling also plays an important role in the real time operating system which always has a time constraint on computations. A real time system is the one whose applications are mission-critical, where real-time tasks should be scheduled to be completed before their deadlines. Most real-time systems control unpredictable environments and may need operating systems that can handle unknown and changing tasks. So, not only a dynamic task scheduling is required, but both system hardware and software must adapt to unforeseen configurations. There are two main types of real-time system. Hard Real-Time System, Firm or Soft Real-Time System. In Hard Real-Time System, it requires that fixed deadlines must be met otherwise disastrous situation may arise whereas in Soft Real-Time System, missing an occasional deadline is undesirable, but nevertheless tolerable. System in which performance is degraded but not destroyed by failure to meet response time constraints is called soft real time systems. First in first out Also called firs come, first serve(FCFS), this principle states that customers are served one at a time and that the customer that has been waiting the longest is served first. Last in first out This principle also serves customers one at a time, but the customer with the shortest waiting time will be served first. Also known as a stack. Processor sharing Service capacity is shared equally between customers. Priority Customers with high priority are served first in computing, scheduling is the method by which work is assigned to resources that complete the work. The work may be virtual computation elements such as threads, processes or data flows, which are in turn scheduled onto hardware resources such as processors, network links or expansion cards. A scheduler is what carries out the scheduling activity. Schedulers are often implemented so they keep all computer resources busy (as in load balancing), allow multiple users to share system resources effectively, or to achieve a target quality of service. Scheduling is fundamental to computation itself, and an intrinsic part of the execution model of a computer system; the concept of scheduling makes it possible to have computer multitasking with a single central processing unit (CPU). The long-term scheduler, or admission scheduler, decides which jobs or processes are to be admitted to the ready queue (in main memory); that is, when an attempt is made to execute a program, its admission to the set of currently executing processes is either authorized or delayed by the long-term scheduler. Thus, this scheduler dictates what processes are to run on a system, and the degree of concurrency to be supported at any one time – whether many or few processes are to be executed concurrently, and how the split between I/O-intensive and CPU-intensive processes is to be handled. The long-term scheduler is responsible for controlling the degree of multiprogramming.



 Fig:5**.** Queuing diagram for scheduling

7 SIMULATION RESULT

The Test Bench Waveform Editor View is the graphical editing environment in which you can display and edit your Test Bench Waveform (TBW). You can create a test bench that includes input stimulus, and test bench length. The values for your input stimulus can be seen and edited as waveforms. The default Waveform Editor View background is gray. This is to distinguish it from the simulation results in the Simulation View, which has a default black background.  You can also change the Test Bench Waveform Editor View color scheme. In the Waveform Editor View, you can view and edit waveform values, and delete and add back signals. This tab also provides helpful tools for examining the TBW. At any time, you, can view the HDL equivalent of your waveform using the View Generated Test Bench as HDL process in ISE. You can also view the HDL code of the test bench source file by double-clicking the file in the Sources window in ISE. The contents of the source file are displayed in ISE Text Editor.



Initialize all input ports at simulation time zero, but do *not* drive expected stimulus until after 100 nanoseconds (ns) simulation time. During timing simulation, a global set/reset signal is automatically pulsed for the first 100 ns of simulation. To keep the test bench consistent for both timing and functional simulation, it is recommended that you hold off input stimulus until the global set/reset has completed.

8 CONCLUSION

The proposed architecture effectively utilizes the Network Interface (NI) unit. It acts as intermediary system between the routers and processing elements and is responsible for generating, transmitting, and receiving of data packets amongst IP cores at same time working as a channel for by pass switch during low traffic. In this project, I consider the problem of designing Interconnect architecture for CPU-GPU based heterogeneous many core systems. Efficient on chip Interconnection strategy to placing CPUs and GPUs on an integrated platform and sharing the common network resources that can decrease the data transfer latency for both cores. A need for choosing an optimal network topology to determine the optimal power-area-latency of the NoC. A better arbitration algorithm to tackle network latency, since at higher core counts, latency becomes a fundamental limitation as the packet has to wait for longer arbitration cycle. To make sure that this proposal will be useful to support modern Multicore chips and to verify that this approach enables high performance while preserving area- and power-efficiency.

REFERENCE

 1. NVidia. NVIDIA GP100 Pascal Architecture. White paper. 2016. [Online]. Available: http://www.nvidia.com/object/pascalarchitecture-whitepaper.html

2. Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz mesh interconnect for a teraﬂops processor,” IEEE Micro, vol. 27, no. 5, pp. 51–61, Sep. 2007.

 3. R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski, “Catnap: Energy proportional multiple network-on-chip,” in Proc. Int. Symp. Comput. Archit., Jun. 2013, pp. 320–331.

4. NVidias. NVIDIA Tesla V100 GPU Architecture the World Most AdvancedDataCenterGPU.Whitepaper.2017

5. S. Borkar, “Thousand core chips: A technology perspective,” in Proc. Des. Autom. Conf., Jun. 2007, pp. 746–749.

 6. B. K. Daya, C. H. O. Chen, S. Subramanian, W. C. Kwon, S. Park, T. Krishna, J. Holt, A. P. Chandrakasan, and L. S. Peh, “SCORPIO: A 36-core research chip demonstrating snoopy coherence on a scalable mesh NoC with in-network ordering,” in Proc. Int. Symp. Comput. Archit., Jun. 2014, pp. 25–36.

7. A. Bakhoda, J. Kim, and T. M. Aamodt, “Throughput-effective on-chip networks for manycore accelerators,” in Proc. Int. Symp. Microarchitecture, Dec. 2010, pp. 421–432.

 8. G. Chen, M. Anders, and H. Kaul, “Scalable crossbar apparatus and method for arranging crossbar circuits,” U.S. Patent 9,577,634, Feb. 21, 2017.

 9. G. Passas, M. Katevenis, and D. Pnevmatikatos, “VLSI microarchitectures for high-radix crossbar schedulers,” in Proc. Int. Symp. Netw. -, May2011, pp.217–224.

10. K. Sewell, R. G. Dreslinski, T. Manville, S. Satpathy, N. Pinckney, G. Blake, M. Cieslak, R. Das, T. F. Wenisch, D. Sylvester, D. Blaauw, and T. Mudge, “Swizzle-switch networks for many-core systems,” IEEE J. Emerging Sel. Topics Circuits Syst., vol. 2, no. 2, pp. 278–294, Jun.2012.

11. J. Lee and H. Kim, “TAP: A TLP-aware cache management policy for a CPU-GPU heterogeneous architecture,” in Proc. Int. Symp. High Perform. Compute. Archit., Feb. 2012, pp. 1–12.

12. A. K. Ziabari, J. L. Abellan, Y. Ma, A. Joshi, and D. Kaeli, “Asymmetric NoC architectures for GPU systems,” in Proc. Int. Symp. Netw.-Chip, Sep. 2015, pp. 25:1–25:8.

13. C. Sun, C. H. O. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. S. Peh, and V. Stojanovic, “DSENT - A tool connecting emerging photonics with electronics for opto-electronic networkson-chip modeling,” in Proc. Int. Symp. Netw. -Chip, May 2012, pp. 201–210.

14.. H. Kim, J. Kim, W. Seo, Y. Cho, and S. Ryu, “Providing costeffective on-chip network bandwidth in GPGPUs,” in Proc. Int. Conf. Comput. Des., Sep. 2012, pp. 407–412.

15. X. Zhao, S. Ma, Y. Liu, L. Eeckhout, and Z. Wang, “A low-cost conﬂict-free NoC for GPGPUs,” in Proc. Des. Autom. Conf., Jun. 2016, pp.34:1–34:6.