International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 5, May - 2014

The Next Generation SOC Processor for Emerging on-Chip Functionalities

[1]Savitha A,[2]Mrs Shailee.S
[1] CMR Institute Of Technology, Visvesvaraya Technological University, Bangalore, Karnataka, India.
[2] Asst.prof, Dept of ECE, CMR Institute Of Technology, Visvesvaraya Technological University, Bangalore, Karnataka, India

***Abstract*— The system on chip is an integrated circuit that integrate all components of electronic system into a single chip. It may contain a digital, analog, mixed-signal and often radio-frequency functions all on a single chip substrate. SOC is believed to be more cost-effective than SIP since it increases the yield of the fabrication and because its packaging is simpler. In this project we designed a SOC processor to offer high multithreaded performance, as well as high single threaded performance from the same chip. It consist of 8 banks and 8 cores and each core consist of 8 threads of which 2 can be executed simultaneously. We implemented the core, bank, memory controller, network interface, MESI protocol into a single SOC chip. By implementing different techniques to this three functional blocks, which significantly improves the throughput ,power performance and read/write abilities of the cores**

Keywords: multi core, power management, cache,system-on-chip (SOC),MESI protocol, FIFO, packet synchronization

I. INTRODUCTION

The processor combines optimized performance on single-threaded and cryptographic workloads with high throughput performance, creating a scalable system-on-a-chip ideal for a wide range of mission-critical applications. The SOC processor delivers a increase in single-thread performance over the previous generation, while maintaining the high multi-thread throughput performance of T-series system. The throughput based on the on-line transaction process(OLTP) benchmark, has doubled with increasing on chip thread count for each generation from T1 to T3. Here the processor is designed in such a way that, it automatically switches to single-thread mode when only a single thread is active, dedicating all resources to that thread's execution. By integrating encryption capabilities directly inside the instruction pipeline, the SOC processor eliminates the performance and cost barriers typically associated with secure computing and makes it possible to high security level without impacting the user experience. To bridge this increasing gap between the single-threaded and throughput performance processor introduces a new out-of-order ,dual-issue ,high frequency core. The single-thread performance in the processor results in better scalability to address a wider range of application and a faster response time for each thread.

II. PROPOSED ARCHITECTURE OF SOC PROCESSOR

The processor implements eight core to provide a total of 64 threads on a die. Each of the core communicates to the L2 cache through a shared memory bus. The L2 cache is divided into 8 16 way set-associative 0.5MB banks with a 64 –byte line size. All the cores typically share common bus for their memory access, the available bandwidth can be overwhelming and causes performance degradation. To alleviate this problem L1 caches are typically localized within each core, saving bus bandwidth and minimizing memory contention.T4 is a complete system-on-chip utilizing the well-established platform of its predecessor. Here we expose a new network interface (NI) with advanced networking functionalities can significantly improve the throughput and power performance of the interconnection on a chip. MESI coherency protocol which is again a lower power and improves the read/write abilities of the cores. On-chip memory controller(MC) works with area efficient FIFO based command and read/write controlling. We integrate all the above functionalities and test their architectural behavior in SOC processor and evaluate the performance.

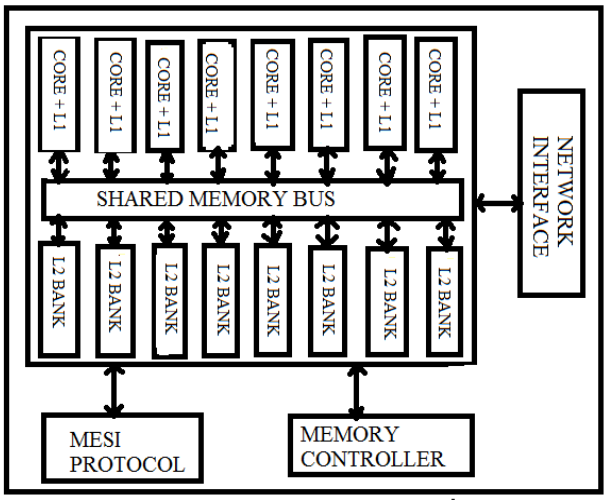

Figure 1- architecture of the SOC processor

A. *Network interface*

It has been long predicted that SOC grows in complexity with more and more processing blocks(like MPSOCs), the current bus based architecture will run out of performance and in, consuming far more energy than desirable to achieve the on-chip communication and bandwidth. The search for alternative technique as lead to the concept of network_on_chip. Redesigning interconnection network between cores is major focus of chip manufacturers. A faster network means a lower latency in inter-core communication and memory transaction. This above problem can be solved by exposing new network interface(NI) with advanced networking functionalities can significantly improve the throughput and power performance of the interconnection on a chip. A key element of network_on_chip(NOC) is the network interface(NI). NIs are the peripheral building blocks of NOC,Decoupling computation from communication. A NOC packet includes a header and data payload which are physically split into units called flits. All the flits of the packet are routed through the same path across the network. In top level view of NI we have request path and response path. In a request path the NI generates request transactions and receives responses, and in the request path it receives and elaborates the request and then send back proper response. Basically, the NI is in charge of traffic packetization/depacketization to/from the NoC: it provides protocol abstraction by encoding in the packets header all data to guarantee successful end to end data delivery between cores. The latest research on NI architecture design aims at integrating more features to directly support in hardware advanced networking functionalities. The challenge low as possible with respect to the connected cores.

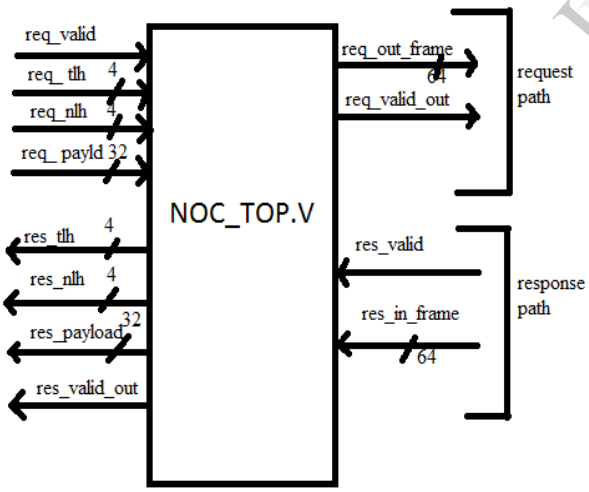

Figure 2 – top level view of NI

B. *MESI protocol*

Cache coherency is a major concern in a multicore environment because of distributed L1 and L2 cache. Since each core has its own cache, the copy of the data in that cache may not always be the most up-to-date version. For example, imagine a dual-core processor where each core brought a block of memory into its private cache. One core writes a value to a specific location, when the second core attempts to read that value from its cache it won't have the updated copy unless its cache entry is invalidated and a cache miss occurs. This cache miss forces the second core's cache entry to be updated. If this coherency policy wasn't in garbage data would be read and invalid results would be produced, possibly crashing the program. This above mentioned cache coherency problem is solved by using MESI protocol.

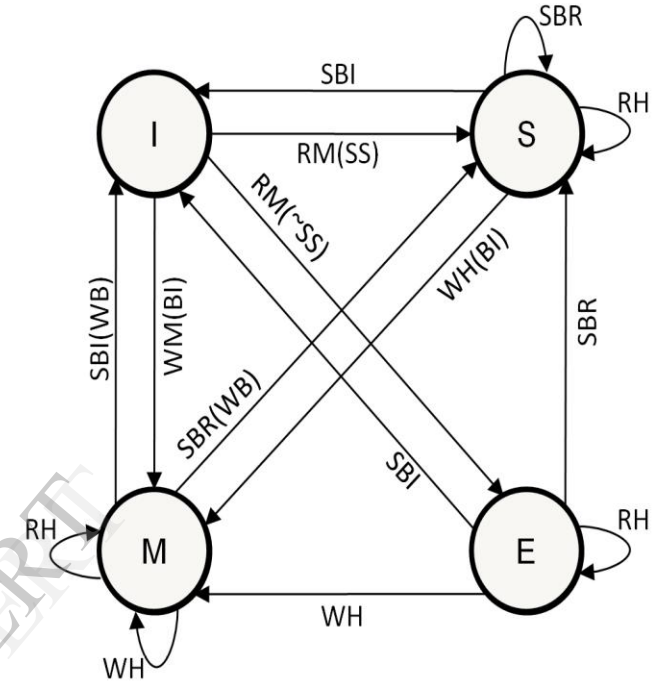

Figure 3 – MESI write-back state diagram

The MESI protocol is a widely use cache coherency and memory coherency protocol. It is the most common protocol which supports write-back cache. In a Modified(M) the cache line is present only in current cache, and is dirty, and it has been modified from the value in main memory at some time in the future, before permitting any other read of the main memory state. The write-back changes the line to the exclusive state. In Exclusive(E) state the cache line is present only in the current cache, but is clean it matches the main memory. It may be changed to the shared state at any time, in response to a read request. Alternatively, it may be changed to the modified state when writing to it. The Shared(S) state indicates that this cache line may be stored in other caches of the machine and is clean. It matches the main memory. the line may be discarded(changed to the invalid state) at any time. Lastly Invalid(I) state means no valid data is stored in that cache line. A cache may satisfy a read from any state except invalid. An invalid line must be fetched(to the shared or exclusive states) to satisfy a read. A write may only be performed if the cache line is in the Modified or Exclusive

state. If it is in the shared state, all other cached copies must be invalidated first, this is typically done by a broadcast operation. A cache that holds a line in the Exclusive state must also snoop all read transaction from all other caches, and move the line to shared state on a match. The Modified and Exclusive states are always precise. The shared state may be imprecise that if another cache discards a shared line, this cache may become the sole owner of that cache line, but it will not be promoted to exclusive state.

C. *Memory controller*

Managing traffic among multiple cores, on chip memories and other mastering peripherals, and the configuration register is one of the challenging tasks in the next generation SOC designs. As a core on single die share the DRAM memory system, multiple programs executing on different cores can interfere with each others memory access request, thereby adversely affecting one another performance. In effect, the memory request of some thread can be denied service by the memory system for long period of time. Thus, an aggressive memory intensive application can severely degrade the performance of other threads with which it is co-scheduled(often without even being significantly slowed down itself) the problem caused by this memory performance will become much more severe as processor manufacturers integrate more cores on same chip in future. This problem can be solved by using area efficient FIFO in the memory controller. First-In First-out(FIFO) is a method of processing and retrieving data. In a FIFO system, the first items entered are the first ones to be removed. The memory controller is the part of the system that, well controls the memory. It generates the necessary signals to control the reading and writing of information from and to the memory, and interface the memory with the other major parts of the system.

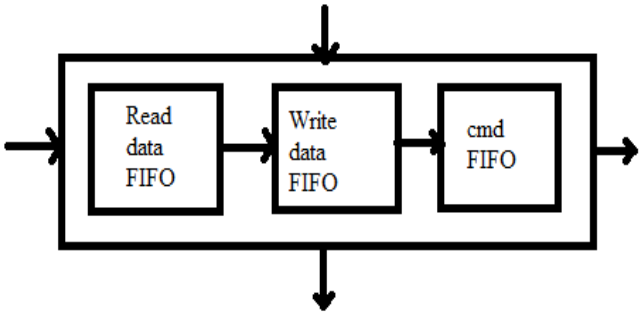

Figure 4 – proposed architecture of the memory controller

III. CONCLUSION

The trend of increasing a processor's speed to get a boost in performances is a way of the past. Mutlicore processor are the new direction manufacturers are focusing on. Using multiple cores on a single chip is advantageous in raw processing power. With additional cores, power consumption and heat dissipation become major concern. To over come the cache coherency problem we implement a MESI protocol that reduces the power consumption and improves the read/write abilities of the core. Network interface with advanced networking functionalities, significantly improved the throughput and power performance. Finally By using FIFO in the memory controller than a arbiter has obviously reduces the area of the SOC processor.

REFERENCES

[1] J. L. Shin et al., "The next generation 64b SPARC core in a T4 SoC processor," in IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech Papers, 2012, pp. 60–62.
[2] A. S. Leon et al., "A power-efficient high throughput 32-thread SPARC processor," IEEE J. Solid-State Circuits, vol. 42, no. 1, pp. 7–16, Jan. 2007.
[3] P. Kongetira et al., "A 32-way multithreaded SPARC processor," IEEE Micro, vol. 25, pp. 21–29, Mar. 2005.
[4] R. Golla et al., "T4: A highly threaded server-on-a-chip with native support for heterogeneous computing," in Hot Chips 23 Symp., Aug. 2011.
[5] J. L. Shin et al., "A 40 nm 16-core 128-thread CMT SPARC SoC processor," IEEE J. Solid-State Circuits, vol. 46, no. 1, pp. 131–144, Jan. 2011.
[6] A. Dash and P. Petrov. "Energy-efficient cache coherence for embedded multi-processor systems through application-driven snoop filtering". In DSD '06: Proc. 9th EUROMICRO Conference on Digital System Design, pages 79–82, Washington, DC, 2006
[7] M. Mamidipaka and N. Dutt. eCACTI: An enhanced power estimation model for on-chip caches. University of California, Irvine Center for Embedded Computer Systems Technical Report, TR-04-28, 2004.97
[8] C.-H. Chan, K.-L. Tsai, F. Lai, and S.-H. Tsai, "A priority based output arbiter for NoC router," in Proc. IEEE Int Circuits and Systems (ISCAS) Symp, 2011, pp. 1928–1931.
[9] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chipcommunication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches," ACM Trans. Des. Autom. Electron. Syst., vol. 12, pp. 23:1–23:20, May 2007.
[10] D. Matos, M. Costa, L. Carro, and A. Susin, "Network interface to synchronize multiple packets on NoC-based Systems-on-Chip," in Proc.18th IEEE/IFIP VLSI System Chip Conf. (VLSI-SoC), 2010, pp. 31–36.
[11] A. Radulescu, J. Dielissen, S. G. Pestana, O. P. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens, "An efficient on-chip NI offering guaranteed services, shared-memory abstraction, and flexible network configuration," vol. 24, no. 1, pp. 4–17, 2005.
[12] H. Kariniemi and J. Nurmi, "NoC Interface for fault-tolerant Message-Passing communication on Multiprocessor SoC platform," in Proc. NORCHIP, 2009, pp. 1–6.
[13] U. G. Nawathe, M. Hassan, L.Warriner,K.Yen, B. Upputuri, D. Greenhill, A. Kumar, and H. Park, "An 8-core 64-thread 64b power-efficient SPARC SoC," in IEEE ISSCC Dig. Tech. Papers, Feb. 2007, p. 108.
[14] S. Shastry, A. Bhatia, and S. Reddy, "A single-cycle-access 128-entry fully associative TLB for multi-core multi-threaded server-on-a-chip," in IEEE ISSCC Dig. Tech. Papers, Feb. 2007, p. 410.