Designing HIPAOC : High Performance Architecture On Chip

By:-
Anvesh Polepalli
Prashant Ahir

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Background

- HIPAOC was proposed by Marta Beltr´an and Antonio Guzm´an , Rey Juan Carlos University, Spain.

- Goal was to propose and implement a HIPAOC model which is application independent and independent of platform, taking reconfigurable computing and different types of memory systems (shared and distributed) into account in the proposed design.

- The proposed model's results have not been presented in the paper, even though they claim to have implement the proposed model.

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Introduction

Field Programmable Gate Array (FPGA)

- Application and platform Specific optimised based on design purpose.
- Reconfigurable solutions with flexibility of general purpose microprocessors with high performance of ASIC devices at low cost.
- Increase in FPGAs configurable logic capacity and the decrease in their costs allow designers to incorporate varying numbers of processors, processor cores or processing units within a single FPGA design

Introduction

Following challenges faced during the design and utilization of these high performance systems on chip.

(i) lack of Unifying Model

(ii) lack of programs portability

(iii) lack of high level design and techniques that allows non expert designers to select this kind of systems as a solution.

Therefore, proposed HIGH PERFORMANCE ARCHITECTURE ON CHIP (HIPAOC)

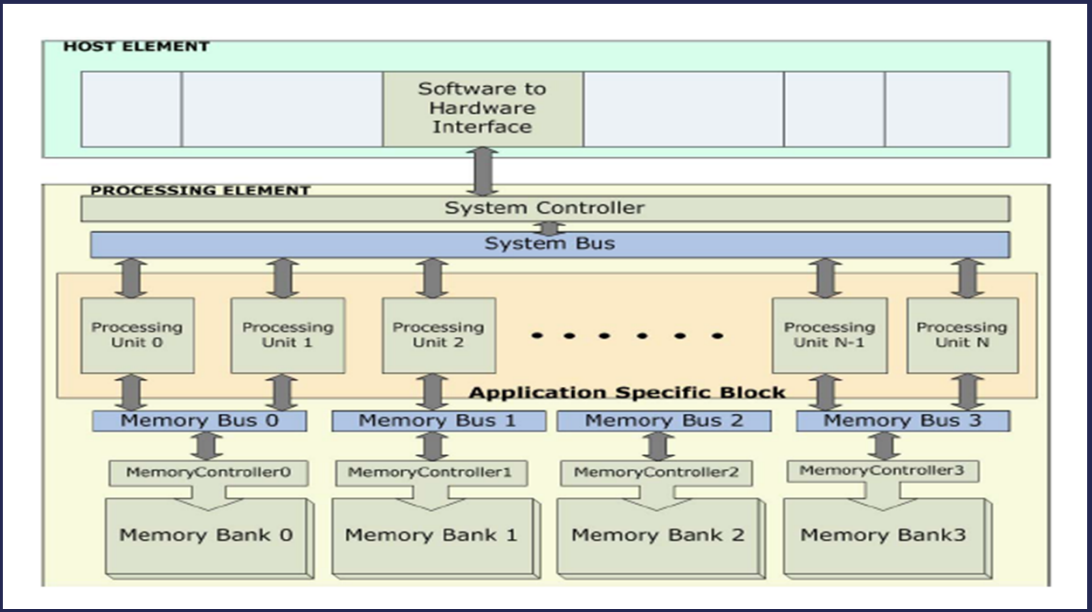
- General purpose reconfigurable high performance architecture on chip.
- Application independent architecture.
- Use of High Level Description Language.

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

HIPAOC Description

- Hardware-Software Codesign approach.



- General purpose System Controller (SC), Memory Controller(MC) & Interconnects. Independent of application.
- HDL Handel-C used for design implementation.
- Application dependent Processing Unit (PU).

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Host Design - Software element

- Host design cannot be general as it has strong relationship with application and FPGA platform.

- Two Staged operation
  - i) Configuration/reconfiguration stage:
    - Configures/reconfigures the system initially.
    - Decides number and architecture of PU's.
    - Decides Memory Controller and Memory model i.e. shared or distributed memory.
  - ii) Execution stage:
    - Generate tasks for PU
    - Send them to SC
    - Gather results

- Three different threads implemented: One to execute software part, one to send tasks to HIPAOC system and one to gather results.

Host Design - Software element

- Features to be considered

a. Application features: Kind of application, alternatives to break the overall functionality into several tasks or processes, granularity of these tasks, dependencies between these tasks, memory accesses pattern, concurrency and parallelism strategies that can be applied, etc.

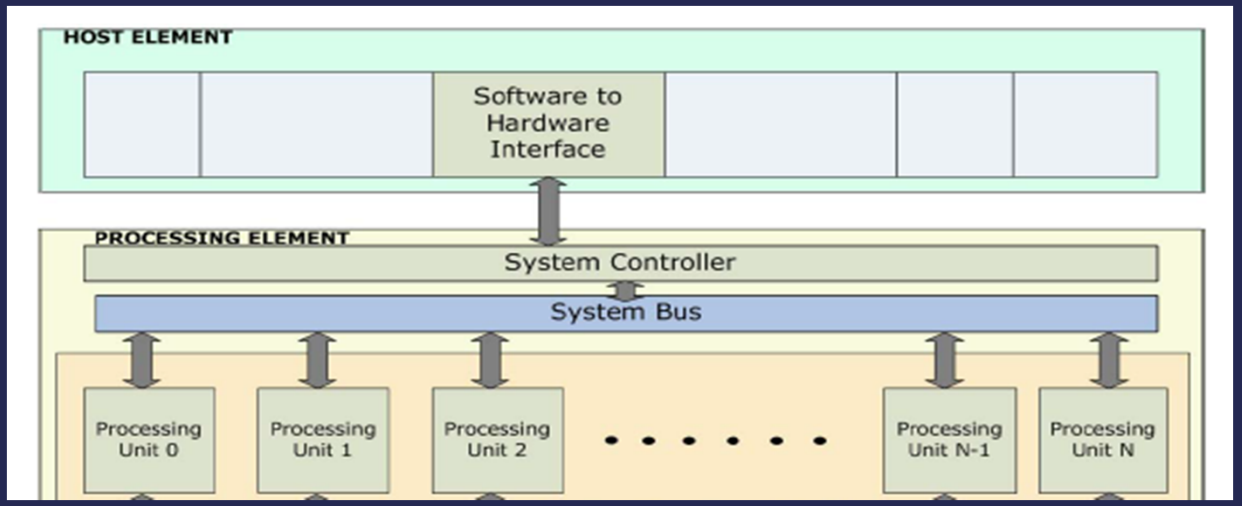
a. Platform features: Libraries for the communication with the software part, hardware memory resources (type, size and number of memory banks), access to the hardware resources (protocols and bandwidth restrictions), etc.

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

HIPAOC Design - Hardware element

System Controller (SC) controls HIPAOC operation



Performs three main function

a. Monitor PU for below two events and when they are produced informs host -
   i. PU finished work and results are to be collected
   ii. PU free to begin new work

HIPAOC Design - Hardware element

b. Manage communication with host using blocking interconnection. This interconnection is used by the host to send tasks to SC.

c. When new task arrives to the hardware, the SC decides which PU is going to execute it and due to the blocking connections, the host only access the hardware resources when SC tell it how to do the access.

Example, SC specifies the memory space that is to be read from or written to by the host depending on the selected PU and memory model.

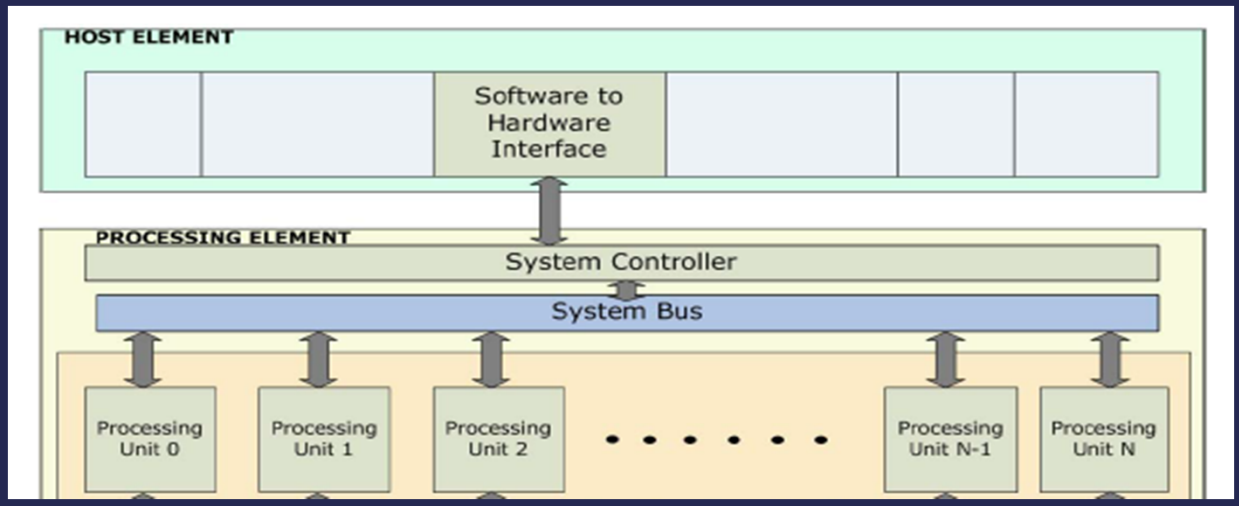
Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

SC/PU Interconnection

- Composed only of Control signals for synchronization of SC -PU.
- Two blocking signals used:

a. *START Signal* - Tells PU to begin work on the new allocated task.
b. *EVENT Signal* - Informs SC about events produced by the PUs.



Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Processing Unit

- PU is the only reconfigurable module in the HIPAOC design, because it is application-dependent.
- The host system decides the number of PU's needed in the system and reconfigures the architecture of these modules to perform the desired functionality.
- There are two main restrictions for the implementation of these modules:
  - They must be homogeneous, i.e. , the design can only contain multiple instances of the same PU.
  - The PU's description must be compatible with the Handel-C description of the rest of the HIPAOC system.

Processing Unit

- Each PU contains a small ROM to store a unique processing unit identifier (PUID) needed for all the HIPAOC operations.

- The following protocol must be used.
  - When the SC activates the *Start* channel of one PU, this PU begins to work on its assigned memory space. During the execution of the new task, all the memory accesses are performed using the appropriate MC, depending on the selected memory model.
  - When PU finishes its work, it notifies this event to the SC using the *Event* channel. Then, when the SC informs the host of this change and the host gathers the PU results, the PU uses again the *Event* channel to inform the SC of its availability.

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

PU/MC Interconnection

- An efficient interconnection architecture has been implemented combining both blocking and non-blocking channels.
- The non-blocking interconnection has been implemented with a bus composed by three shared registers(Address, Data, Check).
- These allow each PU to ask for memory accesses to the correspondent MC depending on the selected memory model.
- The *Address* and *Data* registers are used to perform memory access. The *Check* register has control functions and allows a PU to know when its requested memory access has been performed.

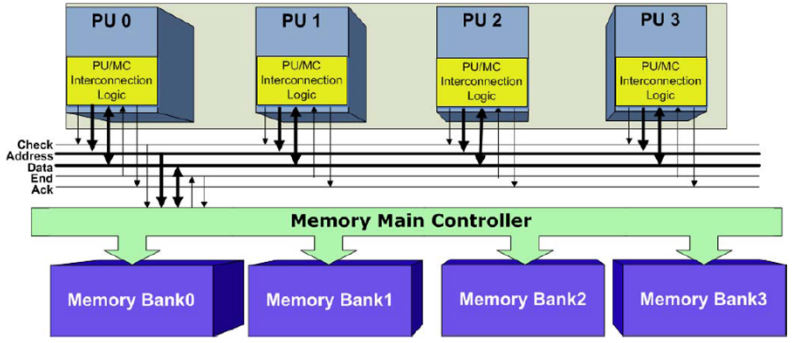
PU/MC Interconnection

- Two blocking channels (‘End’, ‘Ack’) have been added to perform protocols and to ensure synchronization.
- ‘End’ is activated by a MC when it has finished a memory access and it has been notified to the PU’s writing in the *Check* register the PUID of the requesting PU.
- ‘Ack’ is used by the PU which sees its PUID in the *Check* register to notify the MC that the memory access has been checked.

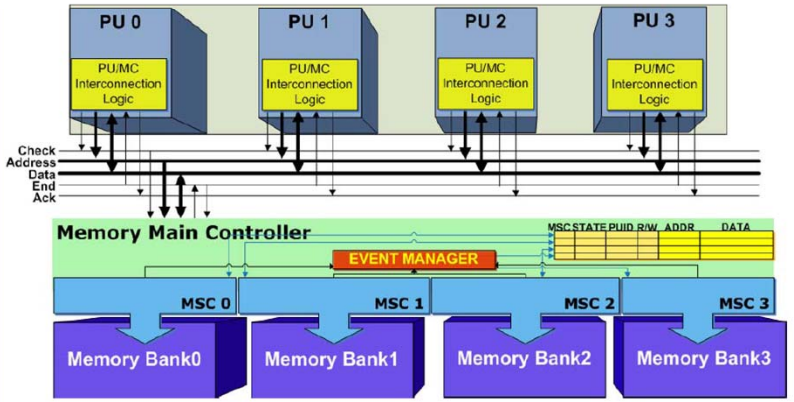
Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Memory Controllers (MC) - Shared



Shared memory model with one memory controller: the MMC



Shared memory model with several memory controllers: one MMC and the MSCs

Memory Controllers (MC) - Shared

- There is only one memory address space common to all the PU's.
- This model has one or more memory controllers to perform all their memory accesses and the communication between the PU's occurs through their shared memory space.
- Two implementation alternatives are being tested.
- One memory controller to perform the memory accesses following the traditional scheme, therefore, sequentially. It is called Main Memory Controller (MMC).

Memory Controllers (MC) - Shared

- MMC takes advantage of the hardware platform features and allows parallel memory accesses, it is connected to as many controllers (Memory Secondary Controllers, MSC) as parallel memory accesses can be performed.
- The function of the MMC is to deliver the PU's petitions to the adequate MSC, but it does not perform the real memory accesses, it only distributes the work among the MSC to exploit the accesses concurrency when it is possible.
- This implementation is more efficient but it spends more hardware resources.
- With shared memory - cache coherence problem if the PU's contain one or more cache levels is not solved.

Outline

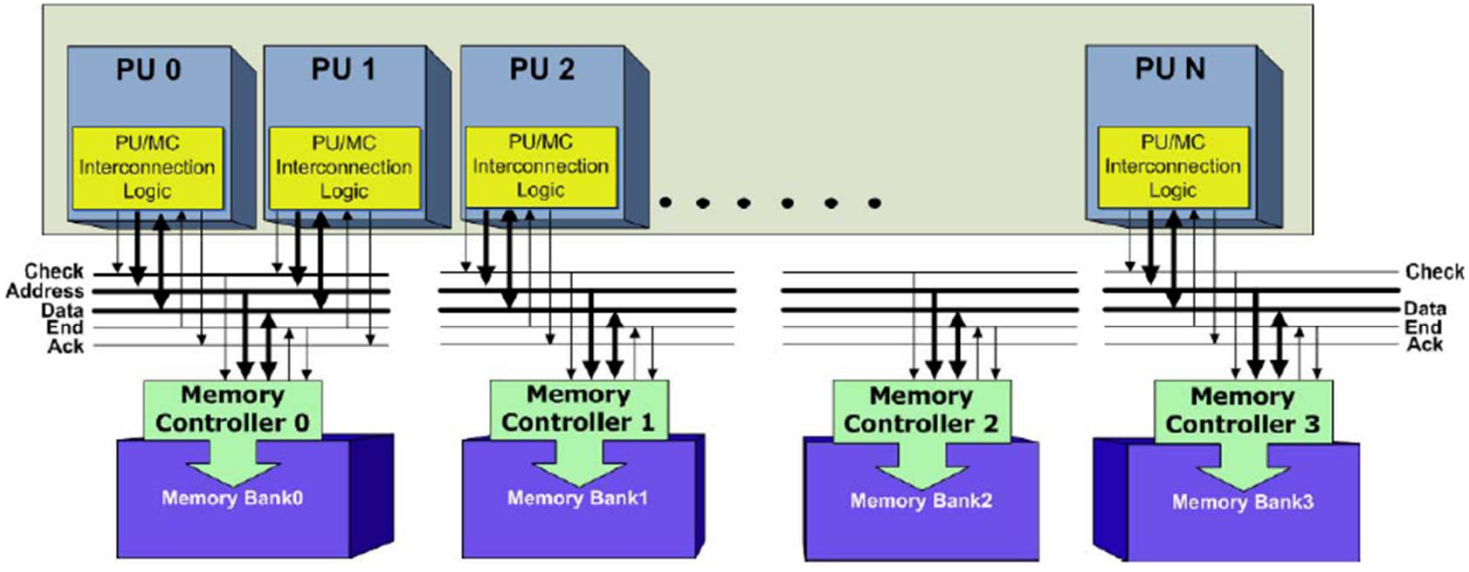
- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Memory Controllers (MC) - Distributed

- There are as many memory controllers as memory accesses can be performed in parallel.
- Number of parallel accesses equals number of memory banks in the platform.
- Each PU has its own memory address space, to perform its memory accesses it has to ask to its correspondent MC.
- Each PU is only connected to its MC there is no possible confusion about this correspondence.

Memory Controllers (MC) - Distributed

- If a PU needs to perform an access out of its memory space, it has to ask explicitly for it to the correspondent PU establishing a PU-PU interconnection with some kind of message passing (application-dependent therefore, depending on the PU's design) which has not been implemented.



Distributed memory model

Outline

- Background
- Introduction
- HIPAOC Description
- Host Design
- HIPAOC Design
  - System Controller (SC)
  - SC/PU Interconnection
  - Processing Unit
  - PU/MC Interconnection
  - Memory Controller (MC)
    - Shared
    - Distributed
- Conclusion

Conclusion

- The proposed mode seems very promising, in terms of performance flexibility (both application and platform independent).
- Two main decisions must be made: the PU's architecture and the memory model.
- But no analysis can be made as no results have been presented.

References

Beltran, M.; Guzman, A., "Designing HIPAOC: High Performance Architecture On Chip," Industrial Embedded Systems, 2008. SIES 2008. International Symposium on , vol., no., pp.233,236, 11-13 June 2008

Clive Maxfield. The design warrior's guide to FPGA devices, tools and flows. Elsevier, 2004.

A. Hung; W. Bishop and A. Kennings. Symmetric multiprocessing on programmable chips made easy. In Proceedings of the Design, Automation and Test in Europe Conference and Exhibition, volume 1, pages 240–245, 2005.

R.Dimond, O.Mencer, and W. Luk. Application-specific customisation of multi-threaded soft processors. In IEEE Proceedings on Computers and Digital Techniques, volume 153, pages 173–180, 2006.

Thank You