Research on the Embedded Heterogeneous Multi-core Design Method for 100GbE Network Processor

Xiangyun Zeng\textsuperscript{a}\textsuperscript{*}, Lianfeng Zhao\textsuperscript{a}, Dong Bian\textsuperscript{b}

\textsuperscript{a}Xidian University, Shaanxi, Xian 710126, China
\textsuperscript{b}Shandong University, Shandong, Jinan, 250100, China

Abstract

We proposed a design method of heterogeneous multi-core processor on chip. In our design flow, we structured a computing model mapped onto the processor’s micro architecture, and structured a work-load model mapped onto the system architecture. The design method of heterogeneous multi-core processor means that the core types, core numbers, and core interconnections varied along with the various applications. In other words, the multi-core architecture is application-specific. At last, a design of an embedded heterogeneous multi-core 100GbE network processor is used as an application example.

© 2011 Published by Elsevier Ltd. Selection and/or peer-review under responsibility of Harbin University of Science and Technology Open access under CC BY-NC-ND license.

Keywords: Embedded heterogeneous multi-core design method; Network processor; Manage core; Base core; Special core

1. Introduction

The integrated circuit processing technology and the processor architecture technology are two important factors in the development of processor chips [1].

The development of processor architecture technology makes it possible to design more complex and flexible systems on a chip. At the same time, more complex and flexible systems on a chip brings about more complicated design process and higher costs.

There are two essential requirements for the embedded processor design. One is that the processor should have application-specific architecture. The other is that the power consumption should be low.

\* Corresponding author. Tel.: +86-15029980878.
\textit{E-mail address:} xbwzxy2000@163.com

The project is supported by the State Key Laboratory of High-end Server & Storage Technology. The project ID: 2009SHSSA11
Researches have shown that heterogeneous multi-core SoC architecture design method will be one concentration for the embedded processor’s design in the future.

2. Design method of the multi-core processor[1][2] on chip overview

2.1. Multi-core processor instruction set

Dynamic instruction set: several heterogeneous core architectures are integrated on a chip, and different instruction systems are required by heterogeneous cores to meet the needs of diverse tasks’ parallel processing.

MISC (Multi-core processor Instruction Set Computer) is introduced to implement task allocation, load balancing and system management by the manage core. Parallel multi-tasks computing model is implemented by instruction pipelining. Specific micro-architecture can be used in the special core, to implement I/O parallel processing, because of the special core’s specific applications. Each core can execute its own instruction system.

2.2. Main tasks for the heterogeneous multi-core processor design

There are three main tasks for the multi-core processor on chip design.

• We designed the micro architecture. First, we built the application-specific computing model, and then we mapped it onto the processor computing architecture model to structure the micro architecture.
• We designed the system architecture. First, we built the parallel multi-task execution model, and then mapped it onto the multi-core processor’s system architecture, such as core types, core numbers, and core interconnections.
• We designed the interconnect structure, mainly focusing on types of local storage and shared storage needed, the interconnection inside between cores and storages, I/O Interfaces.

3. Design method of heterogeneous multi-core processor on chip

To overcome the processor design barrier, the design method of heterogeneous multi-core processor is proposed, which means cores focusing on different functions are integrated together.

Considering the need for cores with different architectural feature handling separately functions, multi-core architecture comprises the following three programmable cores:

• Manage core. The manage core is used mostly for task scheduling, system management and load balancing, characterized by a high clock frequency, complex out-order issue and instruction pipeline, as well as high computing performance.
• Base core. It has been designed specifically for the highly extensible parallel task program with low clock frequency, simple instruction pipeline.
• Special core. It is optimized towards specific tasks such as network processor, video processor, and motor control processor for intelligent robot’s elbow, with special processing power and I/O interface.

Employing different architectural strategies, all three cores have been specifically optimized to meet specific tasks. It is not only the manage core, the base core, but also the special core that can be integrated in the heterogeneous multi-core architecture. Application-oriented heterogeneous multi-core architecture design method makes multi-core SoC processor design concept change from general purpose to application-specific purpose, from homogenous to heterogeneous, and from focusing on computing model to giving consideration to I/O processing. The design method and application mode for general purpose processor will be replaced by the design method and application mode for application-specific processor.
When considering multi-core architecture, we no longer integrate large numbers of homogenous general-purpose cores and simply interconnect them with each other, but shift our focus to application-specific heterogeneous multi-cores.

4. The design flow for heterogeneous multi-core processor

The typical implementation flow for application-specific multi-core processor consists of three steps. First, determine the processor’s micro architecture by establishing the multi-task computing model for specific application. Second, establish the thread execution module for multi-tasking, or mapping the parallel multi-task execution model onto parallel multi-core architecture, such as core types, core numbers, and core connections. Third, build the interconnection structure of the multi-core processor on chip.

4.1. The micro architecture design

Considering the specific application, such as net processing and video processing, establish the multi-task computing model, or model for the instruction-level parallelism and the thread-level parallelism. On the basis of the computing model, determine the micro architecture of the heterogeneous multi-core processor.

Suppose one application-specific program consisted of \( M \) instructions and has \( N \) threads. Given the degree of thread parallelism is \( T_{\text{para}} \) and every parallel thread is executed by one specific module in the processor, then under the maximum computing speed, the number of thread execution modules is

\[
T_{\text{number}} = N \cdot T_{\text{para}} + K_T
\]  

(1)

Where \( T_{\text{number}} \) is the maximum number of thread, \( K_T \) is the number of threads which are not parallel.

Suppose every thread consists of \( J \) instructions on average and the degree of instruction parallelism is \( I_{\text{para}} \). Under the condition that the waiting time of every parallel thread is zero when to be execute, the number of instruction pipelining for one thread can be calculated as

\[
I_{\text{number}} = J \cdot I_{\text{para}} + K_I
\]  

(2)

Where \( K_I \) is the number of serial instruction pipelining.

In the micro architecture of the multi-thread, multi-instruction pipeline processor, the number of parallel instructions can be calculated as

\[
V_{\text{max}} = T_{\text{number}} \cdot I_{\text{number}}
\]  

(3)

Generally, it can be assumed that in one machine cycle, one instruction is executed in the instruction pipelining design, or the instruction cycle equals to the machine cycle.

in the micro architecture of the multi-thread, multi-instruction pipeline processor, the parameter MIPS(Million Instructions Per Second) is

\[
MIPS = V_{\text{max}} \cdot f = T_{\text{number}} \cdot I_{\text{number}} \cdot f
\]  

(4)

formula (4) shows that under certain frequency \( f \), increasing the number of threads and instruction pipelining, or changing processor’s micro architecture can improve the computing power of the processor.

Considering specific application, next we are mapping the computing model onto the micro architecture of the processor. Under the constraints of \( T_{\text{para}} \) and \( I_{\text{para}} \), the micro architecture consists of \( T_{\text{number}} \) thread execution modules, with \( I_{\text{number}} \) instruction pipelinings in each execution module. So far, we have finished the design flow mapping the micro architecture from the computing model.
4.2 System architecture design: establish the work-load model for multi-tasking

The main task for system architecture design is the whole architecture design, or the hardware resource allocation and implementation strategy. The design’s basis is the work-load model for multi-tasking[4].

To meet the requirement of the multi-task parallelism and the load balance, and determine the parallel multi-core architecture, such as core types, core numbers, and core interconnections, the main focus lies in the partitioning of multi-tasks and mapping onto cores, the SoC communication concept, the task scheduling, the system management, and the resource allocation by the manage core.

Figure 2 shows the flow of the system architecture design.

Suppose one specific application consisted of $M$ pieces of tasks, comprising $N$ pieces of computing tasks, $J$ pieces of I/O processing tasks and $I$ pieces of network processing tasks[5], with the task parallel degree $N_{p}$, $J_{p}$, $I_{p}$, respectively. Here, $M=N+J+I$.

- Determination of core’s types
  
  Three types of cores are needed to meet the tasks’ requirement. First, a processor with high performance is needed to perform as the manage core. Second, a base core used to process computing tasks is required. Third, the special core is needed specifically for specific applications, such as the I/O processing tasks and network processing tasks.

- Determination of core’s numbers
  
  The maximum number of cores is determined by the task parallel degree. The minimum number of cores is determined by the cost, area, power consumption. The maximum number of base cores is determined by the computing tasks. It can be calculated as

$$P_{base} = N_{p}N_{p} + P_{npN}$$  \tag{5}

where $P_{npN}$ is the number of base cores needed for the serial task execution programs.

The maximum number of I/O special cores is determined by the I/O tasks. It can be calculated as

$$P_{io} = J_{p}J_{p} + P_{npJ}$$  \tag{6}

where $P_{npJ}$ is the number of special cores needed for the serial I/O task execution programs.

The maximum number of network processing special cores is determined by the network processing tasks. It can be calculated as

$$P_{net} = I_{p}I_{p} + P_{npI}$$  \tag{7}

where $P_{npI}$ is the number of special cores for the serial network processing task execution programs.
In conclusion, under the multitask condition, the number of cores for each task is determined by formula (5) (6) (7).

5. Application example: embedded heterogeneous multi-core 100GbE network processor

FPGA is used for the network task process such as network flow control, router control and switch control. In this application example, there are five cores in the project. The system architecture is shown in fig.3.(a), in which the manage core is named NetMcore, the special core is named NP0X, the 10Gbps transceiver and Ethernet MAC interface is named MAC 0X. The project’s compiling results are shown in fig.3.(b). The maximum throughput of the 10*10G transceiver channel is 100GbE.

6. Conclusion

Specific micro architecture is required for specific design projects with different tasks. The design method of application-specific processor differs greatly from the general-purpose processor design. In this paper, the design of the 100GbE Network processor works as an instance of the application-specific design method for the embedded heterogeneous multi-core processor.

7. Acknowledgements

We should be thankful a lot for Prof. Jueping Cai in Xidian University. He has given us a lot of helps.

8. References


