VeriNP: A FPGA-Based Verification Platform for General-Purpose Many-Core Network Processors

Jinfeng Huang, Hongbin Wu, Tao Li, Hui Yang and Zhigang Sun
School of Computer
National University of Defense Technology
Changsha, P. R. China
jinfeng_huang2014@163.com

***Abstract*—General-purpose many-core network processors have been widely used in network packet processing due to its high programmability and parallel processing ability. The design of general-purpose many-core network processors involves a lot of key technologies, including packet scheduling, inter-core communication, co-processing, etc. The verification of these technologies is essential before applied to the system. However, there are some of limitations in the current software-based verification platform, such as low simulation speed and fidelity. A FPGA-based verification platform for general-purpose many-core network processors (VeriNP) is proposed in this paper. The VeriNP supports verification of network processor for at least 16 real RISC CPU cores, and the frequent of cores can run at 100MHz or higher. Based on the VeriNP platform, two packet scheduling algorithms are studied and verified.**

Keywords—general-purpose many-core; network processor; verification platform; packet scheduling algorithm; FPGA

I. INTRODUCTION

With constantly enrich of network business, the requirement for flexibility of network equipment has been extremely increasing. Due to its inherent inflexibility feature, the traditional ASIC-based network equipment has been unable to meet the development of network business. And the software programming of network processor with customized cores is also a little bit difficult. The flexibility in programming of general-purpose multi-core and many-core makes them easy to adapt to different network business [1]. And as the number of cores increasing, the processing performance of general-purpose multi-core and many-core is becoming higher. The number of cores for many-core is more than multi-core, and correspondingly, its performance is higher than multi-core. It has become a significant trend to perform network processing based on general-purpose many-core [2, 3, 4]. General-purpose many-core network processors, such as TILE-Gx [5], NPS-400 [6], are widely used in network equipment such as router, switch, firewall, intrusion detection system, etc.

The design of general-purpose many-core network processor involves many key technologies, e.g. packet scheduling, inter-core communication, co-processing, lookup and classification, etc. Most of the verification platforms for these technologies are based on software simulation, such as NepSim [7], ComNetSim [8], PALAC [9], P-GAS [10], Graphite [11] etc. The software simulation verification platforms only occupy partial resources of CPU, and there are a lot of cores and hardware modules should be simulated, which makes the speed and throughput slow. To accelerate the process of simulation, the software simulation verification platforms usually reduce the simulation fidelity or simplify details of the running system.

With the development of programmable hardware, it is prevailing to perform packet processing based on FPGA (Field Programmable Gate Array). The processing performance of FPGA is much higher than software, and the FPGA is easy to be reconfigured. The advantages of FPGA make it possible for key technologies of general-purpose many-core network processors to be verified based on it. FPX [12] is a reprogrammable network packet processing system based on FPGA, it utilizes the configurability of FPGA to implement different network processing task; however, it does not integrate cores.

In this paper, the architecture of the verification platform for general-purpose many-core network processors (VeriNP) is introduced. The architecture has separated data SRAM and instruction SRAM for each core, and data SRAM can be directly accessed by packet scheduler. This architecture can match real network processor's memory system. By tailoring the components that are unrelated of network processing, VeriNP can integrate more than 16 C-programmable general-purpose CPU cores, and the CPU cores can run at more than 100 MHz. Two scheduling algorithms are studied and verified as a user case on VeriNP.

The rest of this paper is organized as follows. Section II and III introduce the background and related work. Section IV describes the design and implementation of the FPGA-based verifying platform. Section V studies and verifies two packet scheduling algorithms for general-purpose many-core network processors. Finally, Section VI concludes the paper and represents the future work.

II. BACKGROUND

General-purpose many-core network processors have the advantage of flexibility and performance; however they are some technological challenges, such as packet scheduling, software programming model, inter-core communication.

This work is supported by grants from Natural Science Foundation of Youth of China(No. 61202483 and No. 61202485).

- Packet scheduling: Packet scheduling performance seriously affects the performance of general-purpose many-core network processors. It is challenging to achieve high load balance between massive cores. The theoretically processing performance of general-purpose many-core network processors is the sum of each core processing performance, while the actual processing performance is largely reduced if loads are not balancing. The scheduling algorithms for general-purpose many-core network processors should be optimized designed and verified.
- Software programming model: There are two processing model in network processors, Run-to-Completion (RTC) and pipeline [14]. In RTC processing model, a task is processed by one core until completed, while in pipeline processing model, a task is divided into several phases, and each phase is processed by the corresponding core. The efficiency and complexity of software programming model depend on the processing model, and how to design optimized software programming model is a challenge.
- Inter-core communication: Communication and data transmission between cores of general-purpose many-core network processors is complex, and traditional on-chip bus and crossbar cannot support high-speed communication and data transmission among the CPU cores. Network-on-Chip (NoC) is a new interconnection methods, it is widely used in communication and data transmission between cores. However, the specific structure of NoC used in different general-purpose many-core network processors should be carefully designed and verified.

A verification platform for general-purpose many-core network processor is thus required to provide the environment for the key technologies emulation and evaluation. The platform should be reconfigurable for different network processor architectures. Moreover, the platform should integrate at least 10 cores for many-core verification. Lastly, the more details of the technologies should be able to verified in the platform.

III. Related Work

There are mainly two kinds of the network processor verification platform, i.e., network processor software simulator-based verification platform and FPGA-based verification platform.

Yan et al. [7] propose NepSim network processor simulator, which is a network processor simulation environment based on Intel IXP1200 network processor architecture. NepSim can support at most four microengine simulation cores, the performance and power consumption of network processor components can be quantitatively analyzed at cycle-level. The optimization design of network processors could be made by analyzing the tradeoff between performance and power consumption. ComNetSim simulator [8] implements the abstract model of Cisco Toaster network processor by using object oriented programming in C++ environment, and the behavior of network processor components can be simulated and studied in it. PALAC [9] is a system developed by Stanford University to study packet classification algorithms. The system consists of traffic generator, packet classifier algorithms and statistics collection mechanisms. Packet classification algorithms can be designed and verified through the system.

Using the high performance and configurability features of FPGA, John et al. [12] propose FPX reprogrammable network packet processing system, which integrated packet receiving, sending, processing and other functions modules, and packet processing hardware of routers and firewalls can be rapid deployed into it by changing the reconfigurable function modules. Dan et al. [13] have studied IP packet compression technology based on network processor which is implemented on FPGA, while there is no specific structure of the FPGA-based network processor.

IV. Design and Implementation of VeriNP

A. Architecture of VeriNP

The VeriNP is designed based on tailored C*CORE [15] and the architecture of VeriNP is shown in Fig. 1.

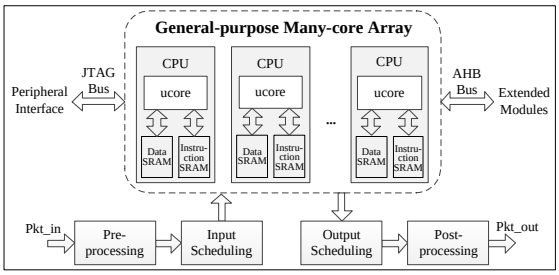

Fig. 1. The architecture of VeriNP

The VeriNP mainly includes: pre-processing module, post-processing module, input scheduling module, output scheduling module, general-purpose many-core array, peripheral interface and extended modules.

Pre-processing module performs pre-processing of incoming packets, e.g. flow control. Post-processing module performs post-processing of output packets, e.g. discard packets. Input scheduling module sends packets to cores according to specific packet scheduling algorithm for further processing. Output scheduling module fetchs packets from cores after processing is completed.

General-purpose many-core array is responsible for the packet processing functions. The array is constructed based on tailored C*CORE (i.e. ucore in Fig. 1). General-purpose many-core array can be programmed with C or C++ high-level language, and instructions can be loaded to its instruction SRAM through JTAG bus. Due to the separated instruction SRAM of the cores, both RTC and pipeline software programming model can be supported.

The functions of the general-purpose many-core array can be extended by utilizing high-speed AHB bus, which is important for hardware coprocessor verification for network processor.

B. Tailoring and Implementation of UCORE

To reduce the complexity of cores in general-purpose many-core array, the C*CORE is tailored for packet processing. The principle of tailoring is cutting out instructions and modules that are not related to or hardly used in network processing.

For instance, MAC instruction is seldom used in network processing but occupies comparatively large resources. Timers and some unavailable registers are also costly in resources and they should also be removed from the cores.

Table 1 illustrates the resources comparison between C*CORE and UCORE. LC combinational logics resource of a C*CORE and UCORE is respectively 9349 and 7576, the reducing ratio is 18.96%; LC registers resource of a C*CORE and UCORE is respectively 4485 and 3597, the reducing ratio is 19.80%.

TABLE I. RESOURCES COMPARISON BETWEEN C*CORE AND UCORE

Resources	C*CORE	After Removing MAC	After Removing Timers	After Removing Unavailable Registers (UCORE)
LC Combinationals	9349	8229	7805	7576
LC Registers	4485	4522	3885	3597

Multiple regression tests have been performed after tailoring. Large amount of use cases are been created to validate the tailored instruction set, and the execution results are checked for the consistency. The result of the regression tests proves that the UCORE has the same correct functionality as the C*CORE.

C. Implementation of VeriNP

The UCORE array is running at different clock domain with input and output scheduling, in order to provide high performance. VeriNP has been implemented on FPGA, and the FPGA device is Stratix IV EP4SE530F43C4, including 424960 ALUTs and 21233664 memory bits. The resources utilization of VeriNP and UCORE array is shown in Table 2. It can be seen that UCORE array occupies main resources of the VeriNP platform.

TABLE II. THE RESOURCES UTILIZATION OF THE VERINP PLATFORM AND UCORE ARRAY

Resources	The VeriNP Platform	UCORE Array
LC Combinationals	133178(31.34%)	128028(30.13%)
LC Registers	69236(16.29%)	60788(14.30%)
Block Memory Bits	2154624(10.15%)	2097152(9.88%)

V. USE CASE OF PACKET SCHEDULING ALGORITHMS VERIFICAITON

The main goal of packet scheduling is to properly schedule packets to different cores, maximally guaranteeing load balance of the system [16]. Two round-robin packet scheduling algorithms have been studied and verified in VeriNP.

Algorithm 1. Round-robin scheduling based on doorbell
Input: *Packet*
Output: *CPUID*
Processing:

```
for(int i=0;i<cpuNum; i++){
   RX[i] = 0;
}
int currCPU = 1;
while(when packet arrived){
   if(RX[currCPU]==0){
      CPUID = currCPU;
      RX[currCPU] = 1;
      if(++currCPU>=cpuAmount)
         curCPU = 0;
   }else
      DiscardPacket();
}
```

The first round-robin scheduling algorithm is based on doorbell mechanism. This scheduling method maintains two state registers RX and TX for each core in UCORE array. When the scheduler is initialized, all RX and TX registers are set to 0. A core's RX register is set to 1 when a packet processing task is assigned to it, and the core cannot be assigned packet processing tasks before the RX register resetting to 0. TX register is set to 1 after the packet is processed, indicating the packet can be output. After the processed packet being output, RX and TX registers are cleared, and the core can be assigned packet processing task again. The algorithm of round-robin scheduling based on doorbell is shown in algorithm 1.

The second round-robin scheduling algorithm is based on mailbox mechanism. The structure of this scheduling method is shown in Fig. 2.

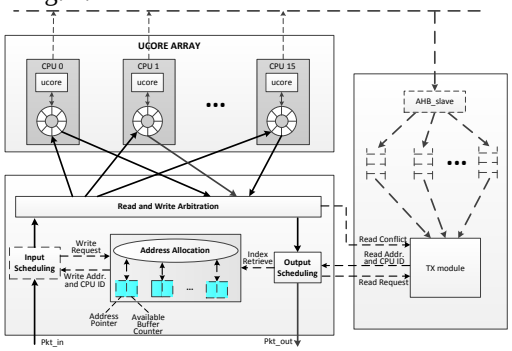

Fig. 2. The structure of round-robin scheduling based on mailbox

In this scheduling method, the data buffers of each core are organized as a ring. The address pointer register and available buffer counter register are maintained for each data buffer ring. The address pointer register indicates the write address of the incoming packet. The available buffer counter register indicates the number of available buffers of a core. When a packet is received, the address allocation module checks whether there is an available buffer block. And, the packet is dispatched to the next available core. The algorithm of round-robin scheduling based on mailbox is shown in algorithm 2.

Algorithm 2. Round-robin scheduling based on mailbox
Input: *Packet*
Output: *WriteAddr* and *CPUID*
Processing:

```
for(int i=0; i<cpuAmount; i++){
   Pointer[i] = HeadAddr;
   Quantity[i] = BlockAmount;
}
int currCPU = -1;
while(when packet arrived){
   currCPU = BufferAvailabeCPU(currCPU);
   if(currCPU == NoCPUAvaliable)
      DiscardPacket();
   else{
      WriteAddr = Pointer[currCPU];
      CPUID = currCPU;
      Quantity[currCPU]--;
      Pointer[currCPU] += BlockSize;
      if(Pointer[currCPU]>=TailAddr)
         Pointer[currCPU] = HeadAddr;
   }
}
```

The two scheduling methods have been implemented in VeriNP, and the resources utilization is shown in table 3. The resources utilization of round-robin scheduling based on mailbox is higher than that based on doorbell. However both resources consumption is small with respect to the FPGA device.

TABLE III. THE RESOURCES UTILIZATION OF TWO SCHEDULING METHODS

Resources	Round-robin scheduling based on doorbell	Round-robin scheduling based on mailbox
LC Combinationals	1369(0.32%)	3588(0.84%)
LC Registers	775(0.18%)	2126(0.50%)
Block Memory Bits	0(0.00%)	1536(0.01%)

The throughputs of two scheduling methods are also measured on VeriNP. We set the packet size to 16 bytes for the maximum performance, and send packets every 24ns. The software running on UCORE array forwards the packets directly to the output scheduling module after it receives the packets. The performance of the two algorithms is shown in Fig. 3.

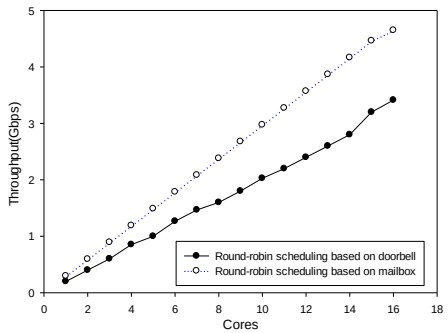

Fig. 3. The throughput of two scheduling methods

As we can see from the figure, the throughput of the two algorithms increase linearly with the core numbers. It is worthy noted that the throughput of round-robin scheduling based on mailbox is higher than that of doorbell. That means that the scalability of round-robin scheduling algorithm is better than the mailbox one.

VI. CONCLUSIONS AND FUTURE WORK

General-purpose many-core network processor is an important trend for network processing due to its high programmability and processing performance. The design of general-purpose many-core network processor involves a lot of key technologies. A FPGA-based platform VeriNP is designed to provide a high-performance and trust-worthy platform for the different technologies. VeriNP integrates separated data SRAM and instruction SRAM for each CPU core, and it can support both RTC and pipeline software programming model. The resource utilization and the performance of the two packet scheduling algorithms are studied and verified as a use case of VeriNP.

REFERENCES

[1] Diaz, Javier, Camelia Munoz-Caro, and Alfonso Nino. "A survey of parallel programming models and tools in the multi and many-core era." Parallel and Distributed Systems, IEEE Transactions on 23.8 (2012): 1369-1386.

[2] Singh, Amit Kumar, et al. "Mapping on multi/many-core systems: survey of current and emerging trends." Proceedings of the 50th Annual Design Automation Conference. ACM, 2013.

[3] Miyamori, Takashi, et al. "Development of low power many-core SoC for multimedia applications." Proceedings of the Conference on Design, Automation and Test in Europe. EDA Consortium, 2013.

[4] Borkar, Shekhar. "Thousand core chips: a technology perspective." Proceedings of the 44th annual Design Automation Conference. ACM, 2007.

[5] Tilera Corporation, "TILE-Gx72 Processor Product Brief", www.tilera.com/files/drim__TILE-Gx8072_PB041-04_WEB_7683.pdf.

[6] EZChip Corporation, "NPS-400 Product Brief", www.ezchip.com/files/drim__NPS-400_brief_Apr2015_7691.pdf.

[7] Luo, Yan, et al. "NePSim: A network processor simulator with a power evaluation framework." Micro, IEEE 24.5 (2004): 34-44.

[8] Suryanarayanan, Deepak. "A Methodology for Study of Network Processing Architectures." (2001).

[9] J. Balkman P. Gupta, "The Packet Lookup And Classification (PALAC)", http://klamath.stanford.edu/~pankaj/research.html.

[10] Lv, Huiwei, et al. "P-GAS: Parallelizing a cycle-accurate event-driven many-core processor simulator using parallel discrete event simulation." Proceedings of the 2010 IEEE Workshop on Principles of Advanced and Distributed Simulation. IEEE Computer Society, 2010.

[11] Miller, Jason E., et al. "Graphite: A distributed parallel simulator for multicores." High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on. IEEE, 2010.

[12] Lockwood, John W., et al. "Reprogrammable network packet processing on the field programmable port extender (FPX)." Proceedings of the 2001 ACM/SIGDA ninth international symposium on Field programmable gate arrays. ACM, 2001.

[13] Munteanu, Dan, and Carey Williamson. "An FPGA-based network processor for IP packet compression." Proc. of SCS SPECTS (2005): 599-608.

[14] Karras, Kimon, Thomas Wild, and Andreas Herkersdorf. "A folded pipeline network processor architecture for 100 Gbit/s networks." Proceedings of the 6th ACM/IEEE Symposium on Architectures for Networking and Communications Systems. ACM, 2010.

[15] China-core Corporation. http://www.china-core.com/.

[16] Lakshmanan, Karthik, Shinpei Kato, and Ragunathan Rajkumar. "Scheduling parallel real-time tasks on multi-core processors." Real-Time Systems Symposium (RTSS), 2010 IEEE 31st. IEEE, 2010.