CUE-v3: Data-Driven Chip Multi-Processor for Ad hoc and Ubiquitous Networking Environment

Hiroaki Nishikawa∗ Hiroshi Tomiyasu∗ Masanobu Okamoto∗
Masayoshi Sugiyama∗ Hiroyuki Uchida∗ Osamu Mizuno†
Hirosi Ishii‡ Makoto Iwata§

Abstract To realize secure and safe information communication environment, the authors are carrying out CUE (Coordinating Users’ requirements and Engineering constraints) project and started to develop a data-driven chip-multiprocessor CUE-v3 for ad hoc and ubiquitous communication environment available even in the case of emergency. Since CUE-series data-driven processors were designed to be an embedded programmable component as well as a multiprocessor element, particular design considerations were taken to achieve pipelined stream processing and real-time processing for multi-media communication environment. In CUE project, an emulation/simulation facility RESCUE (Real-time Execution System for CUE-series data-driven processors) was also built to develop scalable chip multiprocessors in self-evolutional manner. Through evaluations by RESCUE, the latest version named CUE-v2 was developed. CUE-v2 is build as a hybrid processor enabling simultaneous processing of data-driven and control-driven threads to achieve higher performance for parallel processing and to avoid bottlenecks in sequential parts of real-time programs frequently encountered in time-sensitive applications. After discussing requirements to a platform for the ad hoc and ubiquitous communication environment, data-driven chip multiprocessor CUE-v3 has been developing by integrating 4 CUE-v2’s. In this paper, the CUE-v3 architecture will be finally addressed showing effectiveness of the CUE-v3 architecture with special emphasis on scalability and multi-processing capability.

Keywords: chip-multi-processor, dataflow, ad hoc, ubiquitous, real-time

1 Introduction

Through advances in recent data transmission technology, potential bandwidth of network links has been boosted exponentially. To accommodate the next generation broadband multimedia networking infrastructure, there appears to be a new breed of processors called network processor[1]-[3].

In contrast to these approaches, authors are carrying out CUE project to realize a safer ad hoc and ubiquitous communication environment workable even in the case of emergent condition, and have been developing CUE series networking-oriented data-driven processors as a platform utilizing their parallel processing capabilities[4].

Network processors are intensively optimized to extract inherent parallelism in packet processing. Although their detailed optimization methodologies differ considerably, it seems to establish the wide consensus for the extraction of TLP (Thread-Level Parallelism) due to the abundance of explicit packet-level parallelism in the network workloads. Furthermore, network processors provide multithreading and low latency context-switching to hide the long latencies of memory references. Presently, most of network processors don’t exploit ILP (Instruction-Level Parallelism), since the packet processing done in current routers is simple and limited for header fields. In the advanced future network scenario, for example, real-time media trans-coding becomes common, it is expected that the flexible exploitation of various levels of parallelism becomes more essential due to the abundance of ILP in media processing for payload fields.

SMT (Simultaneous Multi-Threaded)[5] is claimed to be suited for the scenario. SMT is designed to exploit both ILP and TLP, thereby aug-
menting a fixed number of state storages of each thread (program counters and register files) to a conventional superscalar processor. Hence, SMT must have a large number of state storages to fully utilize the abundant TLP in the network processing. Consider the cost and the latency of multiple large register files, SMT is not a scalable solution.

To flexibly and naturally exploit various levels of parallelism in the future network workloads, authors have studied networking-oriented data-driven processors unlike SMT, a data-driven processor contains threads, thereby dynamically allocating a packet holding an instruction and an operand into a matching memory at instruction level. Generally, the greater number of matching memory entries than that of register files is feasible. Hence, a data-driven processor is more scalable and flexible than SMT in view of extraction of parallelism. These features lead to the fair multi-processing at instruction level without context switching overhead.

To investigate this, authors previously evaluated CORBA protocol handling\[4\] and real-time video compression\[6\] using CUE series data-driven processors. These evaluations gave us several issues to streamline the execution of the applications for multimedia networking. One of the problems is the unavoidable inefficiency on serial codes. To avoid this inefficiency, the latest version named CUE-v2 developed as a data-driven processor was designed as a hybrid processor enabling simultaneous processing of data-driven and control-driven threads to achieve higher performance for parallel processing and to avoid bottlenecks in sequential parts of real-time processing\[8\]. In this paper, preliminary performance evaluations, and future directions of the CUE-v3 will briefly introduced.

2 Design Philosophy of Networking-Oriented Data-Driven Processor

This section describes background and development history of CUE series data-driven processors. We have consistently built scalable VLSI-oriented data-driven processors with special cautions on real-time multi-processing without any runtime overheads and higher performance.

2.1 Pipelined Stream Processing Based on Data-Driven Scheme

A highly efficient multimedia stream processing scheme becomes essential to develop a ubiquitous information system environment fully utilizing the communication capabilities of broadband networks. Effective processing of multiple multimedia streams is characterized by magnitude as well as their variety in time constraints. In order to utilize wide variety of parallelism and achieve fast response time, the authors have been studying on data-driven chip multiprocessors named CUE.

The authors focused on dynamic data-driven principle which can naturally represent parallel processes in concurrency, pipelining and multi-processing. The stream processing emphasizes the receiving capability of the input stream without obstructing it rather than attempting to speeding up individual processing as have done in the past. Accordingly, in the system design of the first version prototype data-driven processor Q-p[9], it was first decided that the a VLSI-oriented processor has to be realized by extensive pipeline processing scheme since load sharing among spatially distributed modules tends to consume excessive chip area for distribution and collection of tokens or packets. With the pipeline scheme, receiving capability of the data volume can be enlarged by subdividing stages.

By dynamic data-driven scheme, streams are realized in a parallel executable format as tagged token queues which can be effective in avoiding starvation. Also, as long as sufficient data flow is supplied, processing delay is tolerable in the data-driven scheme which is an excellent feature that maximizes the pipelining efficiency. Furthermore, by the data-driven firing rule, processing is always in a forward direction and never requiring flushing. This means that every process in parallel processing can be organized just to do simple unidirectional or straight-forward pipeline processing without any runtime overheads.

2.2 Hybrid Processor CUE-v2

The previous studies by RESCUE[7] also gave us several issues to streamline the execution of the applications for multimedia networking. One of the issues is the inevitable inefficiency on serial codes, such as connection/port management in TCP and the serialization of parameters in video compression. Since a dataflow processor cannot exploit the locality of computation, it is not good at sequential processing\[6\][10][11]. This is the compensation
of exploiting fine-grained parallelism. To alleviate this issue with retaining the advantages of pure dataflow, we proposed an architecture which can simultaneously process dataflow and control-flow threads in common pipelines. The CUE-v2 is an extended architecture of our previous study. The CUE-v2 performs both as dataflow and as out-of-order superscalar in order to alleviate the bottlenecks caused by sequential processing.

The CUE-v2 simultaneously processes two different kinds of threads in common pipelines at instruction level. One kind, denoted “dataflow thread” can execute parallel codes efficiently by exploiting the maximum parallelism exposed by dataflow graph representations; the other, ”control-flow thread” can execute a serial code by exploiting locality as fast as von Neumann processors do. This paper defines dataflow thread (DT) and control-flow thread (CT) as follows: DT is the execution sequence of instructions triggered by tokens (packets) with identical colors. CT is the execution sequence of instructions fetched/issued based upon a PC (Program Counter). The CUE-v2 employs the instruction fetch policy which can minimize the interference between DT and CT. Under this policy, DT has priority to CT. That is, the CUE-v2 basically allocates empty slots, caused by the execution of DT, to CT. Note that minimal issue opportunity is guaranteed to CT to avoid blocking. The invocation of different kinds of thread and the communication between DT and CT are as follows: In case of the CT invocation from DT, a programmer first specifies the instruction which activates a PC by any value. Note that a programmer must specify the instruction which terminates instruction fetch based upon PC at the end of each CT. Then, a programmer specifies the instruction which stores the operand in a dataflow packet to a general purpose register. If a programmer wants to move a lot of data from DT to CT, pointer reference should be used. In case of the DT invocation from CT, a programmer specifies the instruction which generates a dataflow packet with any value of color.

2.3 Microarchitecture of CUE-v2

Fig. 1 shows the block diagram of the CUE-v2. The CUEv2 seems similar to a superscalar machine except its circulation path for DT and its thread management units, i.e., DiCount and CTQ. In fact, CT is processed like as a nonspeculative 2-issue out-of-order superscalar machine does. In view of a traditional dataflow machine, the CUE-v2 adds a program counter and architectural register to front-end pipes.

The functionality of each pipeline stage is stated as follows: IF0 stage selects a thread type to be fetched and calculates addresses stored in an instruction memory (Inst Mem). The calculated address is forwarded to IF1 stage. IF1 stage fetches instructions and issues to ID0 stage. ID0 stage decodes instructions and reserves a matching memory field. Here, we represent the address of this field as MMA (Matching Memory Address). Note that FC stage and WB stage release the MMA in DT and in CT, respectively. ID1 stage dispatches instructions to the appropriate firing control unit (FC) according to the MMA. In addition, the ID1 stage accesses the register file and performs register renaming in case of CT. If the value of the source register is not available yet, the MMA of a producer instruction is given to a consumer instruction. FC stage is responsible for out-of-order scheduling. That is, incoming instructions in both DT and CT wait until their source operands become available in FC stage. INT/LS stage executes an incoming instruction in both DT and CT. In case of DT, it transfers to SW stage for circulating a dataflow packet after calculating a next instruction number (address); in case of CT, it transfers to WB stage and broadcasts a pair of a result operand and its tag (MMA) to matching memories in FC stage. BR stage is used only when executing CT. It executes branch instructions belonging to CT and sends a branch direction to branch predictors and FC stage. The CUE-v2 equips BTAC
(Branch Target Address Buffer) and BHT (Branch History Table) for branch prediction.

3 Architecture of Data-Driven CMP: CUE-v3

We are developing the first version of CUE-v3 as a chip multiprocessor. As shown in fig.2, CUE-v3 contains 4 processors connected with double ring networks on a 5 mm × 5 mm 90 nm CMOS chip. Each processor is based on CUE-v2, and slightly revised for inter communication. This chapter describes the first stage of development project on data-driven chip multiprocessor CUE-v3 realizing the ad hoc and Ubiquitous Communication Environment.

First, the communication scheme between CUE-v2 processor elements on CUE-v3 CMP is described. Second, implementation of intercommunication network connect each processor element is shown. Preliminary performance evaluations and an interconnection scheme of CUE-v3 are then shown. Future development plan will finally be introduced briefly.

3.1 Inter-Processor Communication Scheme in CUE-v3

To utilize previous research products, each processor of CUE-v3 has similar construction to CUE-v2 except inter processor communication mechanism. The authors extend thread activation instructions of CUE-v2, by explicitly assigning destination processor number. This allows remote thread calls. Since CUE-v2 execute simultaneously executes two types threads, each thread should be able to activate both types of threads. Fig.3 shows examples of extended thread activation instructions. Extended processor numbers are assigned to reserved fields, which represent up to 16. With this simple and tractable extension, we can handle remote thread calls in similar scheme as CUE-v2.

Figure 2: CUE-v3

Figure 3: Remote thread activation instructions

3.2 Interconnection Network of CUE-v3

Fig.4 shows the interconnection network of CUE-v3. In a data-driven circler pipeline, each data-driven token is a packet which consists of an operand, destination, node, address, and other flags. These packets are 62 bit width in CUE-v2. In further implementation of CUE-v series, we are planning to integrate additional CUE-v2s as chip multiprocessor cores. Hence wiring contention and area for interconnection network are can be critical. To alleviate this problem and to keep sufficient throughput, we are designing two simple unidirectional ring networks for interconnection.

A traditional data-driven circler pipeline has a switch which forwards result packets to instruction fetch unit. In CUE-v2, this switch also has a link to I/O. To construct ring networks, we extend an
additional link to the neighboring core. Routers shown in Fig.4 are switch with I/O port which can be connected another CUE-v3. This uni-direction ring network applies a simple routing scheme. If the destination of packet is proper, each switch forwards the packet to own instruction fetch unit; otherwise forwards the packet to the next node. Furthermore, this interconnection network has deadlock free structure with virtual channel buffering.

Conventional shared memory model is effective and flexible for inter-processor communication. However, loading execution environment time could be serious overhead at passing short messages. In CUE-v3, processors communicate with both data-driven short message and shared memory. CUE-v2 is based on hybrid execution scheme which can simultaneously execute both data-driven and control-driven instructions in a pipeline. Hence CUE-v2 has instructions which can invoke both types of thread in very short steps. Usually, data-driven scheme can invoke new operation with only one token. To extend these instructions for inter-processor communications, we append Processor Element number to destination fields. Since these new instructions are very similar to existing thread invoking instructions, a processor core of CUE-v3 has almost same construction as CUE-v2. We just extend a few bits in destination field and forwarding path width.

4 Performance Prediction for CUE-v3

For estimating elemental performance of CUE-v3 which is currently been designing as chip multiprocessor version based on CUE-v2, we selected MPLS(Multi-Protocol Label Switching) header handling as a benchmark. In this performance prediction, we first simulated CUE-v2 in RT level, then assume performance of CUE-v3 consist of multiple CUE-v2s were interconnected through double ring network. Fig.5 and fig.6 show turn around time as multiple header handlings. As previously studied results[8], when CUE-v2 has enough resource, turn around time is nearly flat. Thus “data driven”–“control driven” hybrid execution scheme, can achieve low latency response and wide range of pipeline utilization.

Based on this simulation, the authors estimated performance of CUE-v3. In this estimation, inter processor thread activation codes are inserted into benchmark program described above. And communication delays are included to execution time. This communication delays are derived from RT level simulations. Estimated performance of CUE-v3 is shown fig.7. CUE-v3 will achieve almost linear performance improvement with multiplying processors.

Thus, chip multiprocessor systems based on CUE-v2 will provide low latency and high throughput protocol handling system as long as inter-processor communication overheads do not result in a bottleneck.
5 Future Directions

We are planning two stages of VLSI developments of CUE-v3. First, to build and evaluate interprocessor communication scheme, we will implement 4 processor cores chip with very similar CUE-v2 pipeline and simple shared embedded memory system. In the second stage, followings will be main subjects.

(i) Deeper and wider pipeline
To reduce implementation cost, the first version will use almost same pipeline as CUE-v2. CUE-v2 is basic implementation to demonstrate hybrid execution scheme. Hence CUE-v2 core still have room for improvement. Because data-driven instruction should send operand through pipeline, CUE-v2 has loosely connected front and backend pipe. In a near future implementation, we will clearly separate front-end pipe and backend pipe with operand renaming similar to register renaming, see fig. 8. We will also extend additional ALU and FPU. These improvements will achieve higher throughput and clock rate.

(ii) High band width multi-bank shared memory system
In high performance computing system, high band memory system is key technology. Because data-driven programs are insensitive to memory latency, CUE-v2 has small embedded memory and no cache. However in a higher throughput implementation such as CMP, this can be bottleneck. We are planning to extend CUE-v3 memory system to multi-bank shared memory. Fortunately, we can choose throughput oriented memory system. This memory system still has no cache and simple arbitration scheme, however it will fit an data-driven and control-driven multithreading which hides memory latency.

(iii) Further integration
We will study feasibility of further integration. The first implementation of CUE-3 will be 4 cores CMP because of limitation of available shuttle service. Because inter-processor communication scheme of CUE-v3 will meet further integration, we will study feasibility. In the viewpoint of system scalability, bandwidth and latency of interconnection network is critical. In physical implementation, wire intensity of reservation station which contains matching memory limits integration frequently. We will estimate these problems with the above extensions.

Figure 8: Extended CUE-v2 pipeline

6 Conclusion

This paper proposed and discussed a novel data-driven chip multiprocessor architecture to achieve both sufficient throughput and efficient real-time multiprocessing essentially needed in ad hoc and ubiquitous communication environment.

Needless to say, each host and terminal freely moves in the environment using wireless communication capability. Hence, the power feeding is one of the most crucial issues to be resolved. Besides of increase of battery capacity, necessary condition is how to reduce power consumption. The DDP does not operate without any input and operates at once by giving input. Waiting power consumption is rather smaller (ideally zero) in DDP than in existing Von Neumann processors assuming polling and/or preemption processes. By use of this virtue of DDP, there is high possibility to
adopt DDP as the most suitable processor for the environment. Especially, for power consumption, the authors have already examined that an elastic pipeline scheme based on completely distributed and self-timed design is effective through VLSI realizations of CUE-p and CUE-v1 and would like to discuss in another opportunity.

Considering scalability and expandability, the data-driven chip multiprocessor CUE-v3 in the CUE project will be one of the most promising realizations in the chip multiprocessor architecture for fully utilizing future VLSI progress.

The authors strongly feel that sub-100nm era is an appropriate time to shift from conventional machine first approaches to user- or demand-centered paradigm.

Although the networking-oriented data-driven chip multiprocessor CUE-v3 is still the first stage of its realization, the CUE project promises to provide outstanding platform to build a safer networking infrastructure; ad hoc and ubiquitous communication environment.

7 Acknowledgement

Although it is impossible to give credit individually to all those who organized and supported the CUE project, the authors would like to express his sincere appreciation to all the colleagues in the project. This research is partially supported by SCOPE (Strategic Information and Communications R&D Promotion Programme), Ministry of Internal Affairs and Communications, Japan, the Scientific Grant of Japan Society for the Promotion of Science and Semiconductor Technology Academic Research Center.

And, this work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc and Cadence Design Systems, Inc.

References


