A SPECIFICATION AND PROTOTYPING ENVIRONMENT FOR SUPER-INTEGRATED DATA-DRIVEN PROCESSOR SYSTEMS

Yasuhiro Wabiko, Takuji Urata, and Hiroaki Nishikawa
Institute of Information Sciences and Electronics, University of Tsukuba, Tsukuba Science City, Ibaraki, Japan

ABSTRACT

The authors have studied multi-media networking environment for use in future infrastructure, and have proposed a reuse-oriented data-driven specification environment to achieve higher reusability of software components. It has been reported that the first version prototype is effective for not only supporting side-effects detection, but also so-called verification. At the same time, the authors have also studied a super-integrated data-driven processor to realize effective parallel/multiple processing essential in multi-media networking. From the experimental result, it has been shown that the data-driven processor has ideal multi-processing capability without any context-switching overheads, as long as hardware resources are not depleted and at the same time program allocation to multi-processors is optimized.

The authors propose a specification and prototyping environment ensuring not only well-behaviness in programs but also satisfaction of performance requirements in real-time processing. In other paper, the authors have shown that an interactive optimization method of initial program allocation by genetic algorithm (GA) can achieve as high throughput as program allocated by users. This paper focuses upon optimization facility of turnaround time. At the same time, in realizing interactive environment, simply describing or showing schemas is not enough in an interactive user/system interface. Since the audio media is essential to realize literally interactive environment, this paper also discusses incorporating audio media for user/system interface.

Keywords: specification, real-timeness, data-driven, user/system interaction via audio

INTRODUCTION

Many studies aiming at parallel/distributed systems to build future infrastructures have been broadly carried out (Nishikawa, 1997). The authors have studied data-driven processors to achieve multiple processing capability essential in realizing multi-media networking environment, and realized a super integrated data-driven processor CUE-p (Coordinating Users’ requirements and Engineering constraints - prototype) (Nishikawa and Aoki, 1998).

In a software lifetime, reuse is more dominant than anew development. From this point of view, the authors have studied a reuse-oriented specification environment in which users can be assisted to detect side-effects in updating software components when being reused. It has been reported that this environment is effective for so-called verification (Wallace et al., 1989) (Nishikawa and Wabiko, 1998). However, considering that the multi-media processing will be one of dominant targets for information systems, reusing software should avoid not only side-effects on well-behaveness but also side-effects for real-time constraints.

At the same time, user/system interaction is crucial in realizing an interactive environment. In order to realize a truly and literally interactive development environment, audio, including voiced messages, is essential since humans rely not only on visual information, but on audio as well or everyday communication.
In this paper, the authors first show that reducing the critical path in a program is essential for satisfying requirements on time constraints on our data-driven processor as long as hardware resources are not problems. Specification including requirements on turn-around time and throughput is then proposed. An algorithm for finding critical path in programs is then shown. The prototype of our interactive optimization facility of turn-around time is then described, and preliminary evaluation result is described, which shows possibility of easy tuning capability that our environment has. User/system interaction via audio is then discussed based on multi-modal user interface, in which realization by using pattern-matching method is described.

DATA-DRIVEN REAL-TIME PROCESSING SYSTEM AND ITS DEVELOPMENT ENVIRONMENT

(1) Specification and Prototyping Environment for Data-Driven Processor Systems

The authors have developed a prototype of a re-use-oriented data-driven specification environment to support detection of unexpected side-effects in reusing software components. The authors have pointed out that the difficulties in reusing software components are derived from difficulties in explicitly showing side-effects in programs onto specification, which has essentially caused by the gap between a specification and its program. To fundamentally resolve this problem, the unique representation both for specifications and programs is needed. The authors has focused upon a fact that data-driven schema (DDS), which is the program representation on data-driven processors, has broadly been utilized as specification tool, in SADT (Ross,1977) for example. Then the DDS has been adopted for both specification tool and program representation. To directly reflect the side-effects in a generated program onto a specification, consistency between the specification and the program is kept as long as data-dependence is concerned, which is realized by directly generating programs from a specification via data-dependence that is essential both in specifications and programs. The specification environment supports a prototyping method, which explicitly shows the result of prototyping onto a specification whenever a change is made in a generated program. The prototyping method is based on symbolic interpretation to provide comprehensive testing for domain of the data. Furthermore, the specification environment supports restructuring a hierarchical specification to keep understandability of the specification and to free users from conventional top-down or bottom-up constraints in building the hierarchy.

The specification environment has been applied to actual software development and its ability to support detecting side-effects was evaluated. As a result, some side-effects that prevented the program from executing were detected. Furthermore, incorrectness where the specification didn’t correctly reflect user’s requirements were also detected, despite the fact the generated program from the specification was executable. Conventionally such incorrectness has been said to be hard to detect. From this evaluation, it has shown that the specification environment has not only side-effect detection capability, but also so-called Verification (Wallace et al., 1989) capability.

(2) Super-Integrated Data-Driven Processor: CUE-p

The authors have studied not only homogeneous processor systems but also heterogeneous ones because heterogeneous systems can achieve totally higher functionality than homogeneous ones. In particular, the authors have studied VLSI super-integration of heterogeneous processing elements (PEs). As shown in Fig.1, the authors have realized a super-integrated dynamic data-driven processor CUE-p (Nishikawa and Aoki, 1998) using circular elastic pipelines (Nishikawa and Terada et al.,1987). CUE-p consists of two super-integrated data-driven processors (DDPs). Four PEs are super-integrated in a DDP. In contrast to conventional processors, the circular elastic pipeline structure does not depend on a central system clock. As shown in Fig.2, the pipeline realizes each PE and executes self-timed data transfer by local handshaking, in which each data packet flows autonomously. The self-timed data transfer mechanism makes it possible to join two elastic pipelines. The routers connecting PEs within a CUE-p are also realized by the elastic pipeline structure. Furthermore, inter-chip communications are also realized by handshakes between two routers. Thus, multi-processor system of CUE-p is organized by multiple pipelines maintaining their elastic
nature in total.

The instruction-set of the CUE-p is designed to make the CUE-p stream-oriented. It has been reported that the data-driven processor that is super-integrated within CUE-p has sufficient performance to process Japanese high-definition standard for processing moving picture known as MUSE decoding (Yoshida et al., 1995).

Furthermore, the CUE-p has multi-processing capability without any context-switching overhead, which is derived from the elastic pipeline and data-driven principle. That is, there is no interference among independent processes running multiple/concurrently on CUE-p as long as hardware resources are not problems. It has already reported that CUE-p could realize TCP/IP (Transmission Control Protocol / Internet Protocol) protocol handling in 135Mb/s that the ATM (Asynchronous Transfer Mode) requires (Nishikawa and Aoki, 1998).

(3) Assuring Real-timeness on Multiple CUE-p Processors

In multi-media information processing, execution free from side-effects is not enough; it is essential to assure performance requirements such as throughput and turn-around time. In multi-processor systems, performances of a program depend on its allocation to multi-processors. The optimization problems of program allocation to multi-processor system are known as NP-hard or NP-complete; generally no algorithm can solve the problems at useful computation cost. Therefore the authors proposes a method in which a user interactively optimizes initial allocation that was generated by genetic algorithm (GA).

The GA-based allocation method has been realized by one of the authors, and applied to actual software. As a result, it has been shown that the GA-based method could achieve as high throughput as the user optimized whole of the program, but could not achieve turn-around time as short as one by user (Nishikawa, Ishii et al, 1998). Thus this paper focuses upon assurance of turn-around time.

Because of the CUE-p’s multi-processing capability without any context-switching overhead, only the critical path in the program determines the turn-around time of a process as long as hardware resources are not problems. For example, the critical path of the IP handling process in TCP/IP protocol handling resides in manipulating header of an IP datagram. On the other hand, transferring body of the IP datagram does not affect turn-around time of the IP handling process, regardless of the size of the IP datagram as long as hardware resources are not problems. Because of this feature, followings are needed to assure the turn-around time on CUE-p:

a) The critical path should be executed during the term specified by users, and
b) Processes executed along with the critical path should not make execution of the critical path delayed.

As for the latter requirement, if the elastic pipeline falls into overloaded region and cannot achieve sufficient multi-processing capability at overloaded region, it would pose a problem. To satisfy this requirement, a pipeline level emulation facility has been studied by one of the authors, T. Urata.

Therefore this paper focuses upon reducing critical path of a program. Following chapter first describes specification including requirements on throughput and turn-around time. An algorithm to find critical path in a program is then shown. And the prototype which realizes interactive optimization method of program allocation of critical path is described with preliminary evaluation result that shows the effectiveness of this method.

IMPLEMENTATION OF SYSTEM DEVELOPMENT ENVIRONMENT

(1) Specification Including Requirements on Turn-around Time and Throughput

Fig.3 shows elemental notations in DDS. DDS consists of “block” (Fig.3 (a)) to describe specification or a part of specification, “node” (Fig.3 (b)) for functional element and its “input/output ports”, “source” (Fig.3 (c)) / “sink” (Fig.3 (d)) for input/output from/to outer block and “arc” (Fig.3 (e)) for data-dependence.

An example of described requirements for turn-around time and throughput is shown in Fig.4. In this example, turn-around time within 1000 nano seconds and 135Mb/s throughput is required for the “Compute Checksum” node. Turn-around time is the interval between the time at which a pair of input tokens are inputted and the time at which a pair of output tokens are generated. A node represents relations between input/output data. Thus the
most direct way of specifying turn-around time of a node is, to describe the turn-around time onto node. In the same way, throughput of input/output data can be specified on the input/output arcs that represent input/output dataflow.

(2) An Algorithm for Finding the Critical Path

Finding the critical path in executable programs is significant to verify real-timeness of programs on the CUE-p. The algorithm to find the critical path in a data-driven generated program is below:

- ID stands for identifier; e.g.) node ID is the identifier of the node. Route consists of its route ID, its turn-around time and a linear list of node IDs or source IDs that represent the route. Route list is a linear list of route IDs. Current focused point is either a node ID or a source ID. Focused points list is a list of candidates as the current focused point.
- Initialize the route list into empty. Register all of source IDs to focused points list.

0) If the focused points list is empty, go to 3). Otherwise, obtain a node ID or a source ID from the focused points list as the current focused point, and go to 1).

1) Do one of the following a) and b) against the each input port on which the current focused point lies:
   a) If more than one arc are connected to the port:
      compare the each turn-around time of the start points (copy or source) of each routes, delete the route(s) from the route list except the route(s) that has (have) the longest turn-around time. Go to 2).
   b) If only an arc is connected to the input port:
      do nothing. Go to 2).

2) Do one of the following a) and b) again the each output port on which the current focused point lies:
   a) If more than one arc is connected to the port:
      Generate new route IDs and give them to the new routes that consist of the current route and each output arc. Then register the new routes to the route list, and delete the current route from the route list. Then register the sum of the each amount of communication for each arc and the turn-around time of the current route as the turn-around time of the new routes corresponding to each arc. Add each destination of the each arc to the focused point list. Then return to 0).
   b) If only one arc is connected to the port:
      Add the amount of communication cost of the arc to the turn-around time of the current route. Then move the current focused point to the destination of the arc. Go to 1).

3) Find the route(s) that has (have) the longest turn-around time in the route list. The route(s) is (are) the critical-path.

(3) Interactive Optimizing Program Allocations

In realizing the interactive optimization of program allocation of critical path, it is needed to explicitly show users the critical path and hardware resource constraints. In order to support reducing turn-around time of critical path, it is essential to explicitly visualize inter-PE or inter-chip communications. Sequence charts have been broadly utilized to explicitly visualize communication (Tsai, 1996, Yamazaki et al., 1993). Therefore the authors are currently developing an interactive optimization environment using the sequence chart.

Fig.5 shows the interactive optimization windows that are currently under development. Fig.5 (a) shows the interactive optimization operation window with critical path viewer. The X-axis shows processors and the Y-axis shows data-dependence. An inter-processor communication is represented as a polyline between two nodes that are allocated in different processors. Small circles on the line means just passing through the processor. Fig.5 (b) shows an example of configuration of a processor network. Users can reduce time spent in inter-PE communications in the critical path, by moving the nodes to another processor by drag&drop operations. System computes the turn-around time of the program and shows it to user whenever program allocation has been changed by user.

As a preliminary evaluation, the authors applied their method to the message transmission part of data-driven implementation of IIOP (Internet Inter-ORB Protocol) / GIOP (General Inter-ORB Protocol) adopted in CORBA (Common Object Request Broker Architecture). Its executable program consisted of about 160 nodes. The ideal turn-around time without considering hardware resource constraints was about 10.6µsec. The turn-around time was estimated about 12.6µsec when GA initially allocated
the program to multi-processor system consisting of 36 PEs. Its critical path consisted of about 25% of whole program. The allocations of nodes corresponding to about 5% of whole program were optimized using the interactive optimization window. Then the turn-around time of the re-allocated program was improved to about 11.3 µsec. That is, most of the overhead involved within the initial allocation by GA was reduced by our interactive method. This evaluation shows the possibility of easy tuning capability in our data-driven system.

USER / SYSTEM INTERACTION VIA AUDIO

(1) Audio for User/System Interaction

From our research on development environment, the authors have obtained a perspective of realizing an environment that enables users reuse software components with prototyping via user/system interaction. Therefore what user/system interface are provided becomes more and more crucial issue for the environment. Considering that human beings use several media in communication, at least it is too strong constraint that available input media are only the mouse and keyboard and available output media is only screens. Based on such discussions, so-called multi-modal user interface has been broadly studied (Martine, 1997, Takeda et al., 1997).

For example, in reusing a software component, a user may want to change the names described in the components. In changing names represented by character strings, conventionally the user first specifies which name is being changed by clicking the name on the screen and then inputs the new name by typing keyboards. However, it may be easier to read aloud like “Rename A with B.”. Furthermore, in embedding a block to a node, conventionally user chooses the block from menu. However, reading like “Embed A in B.” may be easier. As for output media, considering that sirens in railroad crossing in real life, audio is effective for warnings for side-effects or guidance to complete specification. Therefore this paper focuses upon these kinds of operations, warnings and guidance via audio.

(2) Implementation of User/System Interaction via Audio

Approaches to realize user interface via audio for input media can be roughly classified into two kinds: one is natural language processing such as morphological analysis and the other is keyword extraction using pattern matching. The former targets on flexible recognition of ordinary spoken language; e.g. artificial intelligence (A.I.), and the latter is for a particular situation where available sentence patterns are limited; e.g. data-base search system or flight reservation system. For our development environment, because users’ operations are limited and flexible recognition of ordinary spoken language is not necessary, the latter approach has been adopted. In the subsection a. below, first, available operations are listed in a typical case, and then the regular expression for pattern matching is shown for the example case.

As for audio as output media, approaches can be classified into two: one is the audio synthesis and the other is the playback pre-recorded sentences. The former can read aloud any sentences, but the latter can read aloud only prepared sentences. In our environment, sentence patterns in warnings or guidance can be designed to be limited by representing the warnings or guide messages like “A side-effect possibility has been detected at this node.”, so that playback approach can be adopted. However, one of the most important features of audio is that user doesn’t have to look at display if enough information has already been obtained from audio such as “A side-effect possibility has been detected at ‘Compute Checksum’ node.”. (Of course, he can look at display if he wants to.) To avoid eliminating the merit of audio for user/system interaction, the audio synthesis approach has been adopted in our study. More detailed realization is described in subsection b. below.

a. Operation via Audio

In our development environment, user’s operations in reusing are follows: first, if the desired software component is not described as a block, extract the component as a block using restructuring hierarchical specification. Then user binds the component as implementation for a node. Source names or port names in the component may be changed to be easily understood by the user.

For example, the audio operation form for restructuring is as follows: (D means a name of input or output
data. \( D^+ \) means \( k \)Ds in the form of \( \{ D_0, D_1, \ldots, D_{k-2} \text{and } D_{k-1} \} \).)

a) “Focus on following data.”

b) “\( D^+ \) as input.”

c) “\( D^+ \) as output”

d) “Restructure.”

In the same way, audio operation form for interactive optimization of program allocation follows: (\( R \) means the name of a primitive, and \( X \) means the name/number of a processor.)

e) “Move \( R \) to Processor \( X \).”

As mentioned at the beginning of this chapter, pattern matching has been adopted to recognize which operation the user has specified. For example, a sentence pattern “\( D^+ \) as input.” is represented by regular expression as follows:

\[
(\cdot+)(\cdot+,\cdot+)(\text{and}\cdot+)? \text{ as input.}
\]

where “\( . \)” means any one character, “\( + \)” means 1 or more times, “\( * \)” means 0 or more times, “\( ? \)” means 0 or 1 times, and “( )” means the order of priority.

b. Warnings and Guidance via Audio

Following items are some examples of representative warnings or guidance message forms: (\( N \) means a name of node. \( P \) means a name of port.)

1) Warnings
   a) “A side-effect possibility has been detected at the node \( N \).”
   b) “Phantom tokens have been detected at the input port \( P \).”

2) Guidance
   a) “Specify the implementation of the node \( N \).”
   b) “Specify the data-dependence of the input port \( P \).”

System generates these sentences and transfers these messages to audio synthesizer.

Though the first version of the prototype audio operation system is currently under development, the authors have confirmed that our audio operation methods are at least realizable.

CONCLUSION

In this paper, the authors proposed a specification and prototyping environment that supports not only well-behaved execution but also real-time execution.

This paper first showed that reducing critical path is crucial to satisfy real-time constraints. The algorithm for finding critical path was then described. And, a specification method that includes requirements on turn-around time and throughput were described. Interactive optimization facility of turn-around time was then described, with the preliminary evaluation result that shows the possibility of our environment with interactive optimization method.

At the same time, the realization of the user/system interaction via audio was discussed to realize so-called multi-modal user interface. In this paper the authors showed that input via audio could be realized based on the pattern matching method using regular expressions. Then the realization of output via audio was shown using audio synthesizer.

Further works include realization of cooperation with the emulation facility to support virtual instructions is needed. Prototype multi-media networking environment will be constructed and the authors will apply their system development environment to meet two kinds of requirements that consists of one from end users such as low cost and flexibility, and the other from network service providers such as high performance and high reliability. Also, the effectiveness of the development environment will need to be further demonstrated.

ACKNOWLEDGEMENTS

Although it is impossible to give credit individually to all those who organizes and supports the CUE project, the authors would like to express their sincere appreciation to all the colleagues in the project. One of the authors, H.Nishikawa, is very grateful to Prof. Arvind and Prof. Jack B. Dennis of Laboratory for Computer Science, MIT for their fruitful discussions on data-driven paradigm.

This research is partially supported by the Scientific Grant of the Japanese Ministry of Education.

REFERENCES


FIGURES

(a) Overview

(b) Block Diagram of CUE-p

(c) Processing Element (PE)

Fig.1 Super-Integrated Data-Driven Processor: CUE-p

Data Latch Packet

INT MUL

Router

GNT TBL

DDP: super-integrated Data-Driven Processor
INT: Integer & Logical Operation
GNT: Generation Manipulation
TBL: Lookup Table Reference
MUL: Multiplier & Static Accumulator

Self-Timed Transfer Control Circuit

Handshaking

Self-Timed Transfer Control Circuit

Elastic Pipeline

Circular Elastic Pipeline

Router

DDP

DDP

COMBINATORIAL LOGIC CIRCUIT

Packet

CLK

Self-Timed Transfer Control Circuit

Fig. 2 Elastic Pipeline
Fig. 3  Elementary Notations in DDS

Fig. 5  Graphical User Interface for the Interactive Optimization Scheme on Program Allocation