A Functionally Distributed Responsive Micro Controller for Distributed Real-time Processing

Nobuyuki Yamasaki and Toshihiro Matsui
{yamasaki, matsui}@e1.go.jp
Electrotechnical Laboratory
Agency of Industrial Science and Technology
Ministry of International Trade and Industry
1-1-4, Umezono, Tsukuba, Ibaraki, 305 JAPAN

Abstract

In this paper, we propose a functionally distributed responsive (reactive in real-time) micro controller for distributed real-time processing, which can control personal robots, home automation, office automation, etc. A responsive controller is composed of a mother module and an I/O card customizable for its specific application. A common mother module is composed of a processor, a memory system, an I/O card bus interface for its daughter card, and high speed real-time links called responsive links. In order to realize hard real-time communication needed for robot control, sensing, etc., a responsive link consists of a pair of full-duplex data lines and two-way event lines. Many kinds of systems can be flexibly composed by connecting any number of the responsive controllers using the responsive links.

keywords: Responsive, Real-time, Parallel, Distributed, Control, Responsive link

1 Introduction

Currently, the design of electronic parts for robots, factory automation, home automation, etc., is done in an ad hoc manner, because there is no standard platform. However many parts have similar designs including large overheads. It is unusual that only one processor controls everything in current electronic systems, because it can not provide all the processing power or fulfill different requirements to various functions of sensors, actuators, and integration. In order to realize distributed control, real-time communication between processors is required. But real-time network interface and protocol are currently not standardized. So it is also difficult to connect different systems, and to make large scale systems.

In this paper, we propose a functionally distributed responsive micro controller for distributed real-time processing, which is available to control some kinds of robots, home automation, office automation, factory automation, etc. We call this controller a responsive controller. We want to standardize the interface and protocol of the responsive link so that the responsive controller will be a standard platform which is available to many kinds of applications.

We designed and implemented a prototype of a responsive controller to prove its effectiveness. The prototype was developed to evaluate its performance so as to design the next practical version of a responsive controller, which will be an ASIC integrating whole functions of the prototype board into a chip. That is to say, the prototype was designed as a board, but the next practical version will be an ULSI chip.

2 Responsive System

Embedded control systems including robots must respond to external inputs (sensors, user inputs, etc.) and operate according to the inputs. The external inputs (interrupts) may occur frequently and continuously, so there is a need for a quick response time and a short processing time in order to process accurately. Also, many kinds of devices (motors, sensors, etc.) must be controlled within a specified time limit.

Therefore responsiveness is the most important factor for the operation of the responsive controller. Responsiveness is defined as a combination of responsiveness and real-time. We propose that embedded and control systems should be designed as responsive systems. The word reactive and real-time used herein are defined as follows:

Reactive : A characteristic in which a reflexive action occurs against an external input [1][2]

Real-time : A characteristic in which the exactness of the system (calculation or operation) depends not only on the result but also the time it took to achieve the result [3]
In a responsive system, both the real-time processing for controlling low level devices and the reactivity for emergency processing are required. Reponsiveness can not be achieved by just hardware, and it also can not be achieved by just software. It can only be achieved when both hardware and software are suitably designed, giving careful consideration to each other.

3 Design

3.1 Overview

We let link connection and multi-stage interconnection network (MIN) have responsiveness, and combine them. Then we propose and design an architecture of a functionally distributed parallel computer [4] which enables flexible connection forms. Basically, the whole system is controlled by several responsive controllers connected to each other through responsive links which we designed.

Each responsive controller consists of two components as shown in Figure 1. One is a mother module composed of a processor, a memory system, several responsive links, and an I/O card bus interface for function specific I/O card. The other is an I/O card that controls small grain I/Os and absorbs its system dependency. We designed the responsive controller as small as possible so that it can be arranged close to its I/O.

![Responsive Controller](image)

Figure 1: Responsive Controller

3.2 Responsive Link

Inter-module communication is basically realized by a point-to-point link connection. Each responsive controller is connected by a high speed special link called a responsive link. The responsive link consists of a pair of full-duplex data lines and two-way event lines as shown in Figure 2. Here, it is important to place the event line beside the data communication line. This architecture realizes the responsiveness of communication at the hardware level. A responsive link cable and a responsive link connector are designed so that link connection topology can be changed flexibly.

![Responsive Link](image)

Figure 2: Responsive Link

Each mother module of a responsive controller is equipped with 4 responsive link interfaces, which is sufficient for normal use. Assuming a module needed to communicate with many other modules frequently (e.g., main module), we designed a MIN module which corresponds to the backbone so as to realize flexible and efficient data transmission. The MIN module itself, which is similar to a switching hub, is designed as an I/O card for the responsive controller. Since the MIN card is equipped with 4 responsive links, the MIN module has 8 (4 for the mother module + 4 for the MIN card) responsive links.

3.3 System Configuration

In order to build a system using the responsive controllers, first a function-specific I/O card is inserted into a mother module to compose the function-specific responsive controller. Then, several kinds of function-specific responsive controllers are connected by the responsive link cables to build the system.

4 Implementation

4.1 Real-time Communication

In order for systems to be applicable to different configurations and applications, the communication speed and link length of the responsive link are adjustable. The responsive link connection is point-to-point, because the responsive communication can not be realized by serial bus connection which shares a channel
among modules. We designed a prototype of the link
cable made of copper cable, since its handling is eas-
ier than that of fiber cable. There are two types of
cables. One is an unshielded twist pair (UTP) cable
that is relatively thin and easy to bond. The other
is a shielded twist pair (STP) cable that is relatively
thick and slightly difficult to bend. The performance
of STP is better than that of UTP. Cables should be
chosen according to communication speed and length
requirements (Table 1):

<table>
<thead>
<tr>
<th>Length</th>
<th>Speed</th>
<th>Slow</th>
<th>Middle</th>
<th>Fast</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short</td>
<td>UTP</td>
<td>UTP</td>
<td>UTP</td>
<td>STP</td>
</tr>
<tr>
<td>Middle</td>
<td>UTP</td>
<td>STP</td>
<td>STP</td>
<td>(OF)</td>
</tr>
<tr>
<td>Long</td>
<td>STP</td>
<td>(OF)</td>
<td>(OF)</td>
<td>×</td>
</tr>
</tbody>
</table>

Slow: 160[Mbps], Middle: 480[Mbps], Fast: 960[Mbps]
Short: ~1[m], Middle: ~10[m], Long: ~100[m]

Since conventional real-time communication sys-
tems lacked hardware support, only soft real-time
could be realized. Physical layers of network com-
munication including ATM, IEEE 1394, Gbit Ether-
net, etc., are still growing. But they do not support
hard real-time communication because their designs
are aimed at inter-computer or inter-I/O communique-
tions.

On the other hand, considering carefully embedded
and control use, we designed a responsive communi-
cation architecture of a responsive controller. To re-
duce and fix communication latency, needed for a hard
real-time system, we designed a protocol and hardware
support at the same time. To keep high responsiveness
of data communication and event transmission, the re-
sponsive link has the capability of hard real-time on
events and that of soft real-time on data.

In general, the size of normal data is large, whereas
the size of an event is very small. Communication data
are divided into several packets. Since multi-task sys-
tems are required to exchange data with several other
systems at the same time, they concurrently send and
receive the packets using time slicing. Therefore there
are a large number of data packets and a small num-er of event packets, which are required to communi-
cate at the same time, in this way. Here latency of
event transmission is most important in real-time sys-
tems. But since event transmission time can not be
bound by the time slicing communication, or conven-
tional CSMA/CD through the shared communication
line, a hard real-time system can not be realized.

4.2 Event link
In order to realize a hard real-time system, the re-
sponsive link has a special event line together with a

normal communication line as shown in Figure 2. The
normal communication line is used to transmit normal
data. The event line is used to transmit only events.
Its characteristics are as follows:

- A link is point-to-point,
- The size of an event packet is constant and small
  (64 byte),
- The data path and the event path are separated,
- The number of events is very small compared with
  that of normal data.

Events can be routed to remote responsive controllers
with hardware support independently of data commu-
nication. Events can be immediately sent to the des-
tination controller in real-time through the responsive
link. The total latency of an event is simply calculated
as the number of hops x the latency per link, even if
the normal data line is congested, so that the capa-
bility of hard real-time is maintained for events. The
time needed for one hop is less than 1 μs in the worst
case. For example, if 5 responsive controllers are con-
ected serially, the worst latency between the farthest
nodes is calculated as follows:

\[4 \text{hops} \times 1 \mu s/\text{hop} = 4 \mu s\]

This is the most important characteristic of the re-
sponsive link. This event line is only used to transmit:

- Interrupts between modules,
- Synchronization between modules
  (Inter-processor synchronization); and
- Transmission of the status of responsive con-
trollers.

An event packet contains the:

- Address (source and destination);
- Interrupt level;
- Interrupt vector; and
- Status.

4.3 Data line
There are two types of data transmission modes. They
are:

- Real-time transmission mode; and
- Non-real-time transmission mode.
Real-time transmission mode is similar to isochronous transmission of IEEE 1394 or USB. It keeps a constant bandwidth, so that it can realize soft real-time on the data line. Data for real-time transmission mode is given higher priority than data for non-real-time transmission mode. If a real-time packet and a non-real-time packet with the same destination reach a responsive controller at the same time, the real-time packet takes priority over the non-real-time packet. But real-time packet has a limitation. When an error occurs (1 bit error per 8 bit), normally the packet is recovered by using FEC. When an unrecoverable error occurs, the non-real-time packet is re-sent, but the real-time packet is not re-sent.

4.4 Processor

As a prototype of a responsive controller, we use a RISC processor (the SPARC family) instead of conventional CISC processors or DSP. Since performance of CISC processors is very poor, we don't use CISC processors.

Some DSPs including TM320C4X, etc., are very fast for specific applications such as image processing, voice recognition, etc. Also, TM320C4X has communication ports to realize multi-processing on board easily. But the communication port of TM320C4X is 8bit bus, so it is not easy to connect among processor modules through the port, because its link length is restricted. Moreover it is very difficult to get peak performance out of these DSPs, because a program should be written and tuned by assembly language to do so. We think that DSPs are not suitable for general purpose.

A transputer is considered as another choice. It has also serial links. It is easy to connect among processors by using its serial links. But its serial link doesn't have responsiveness, because its serial links is a shared line for both data and events. Its processing performance is not so high. Some transputers are discontinued. So we don't use a transputer.

We use a SPARC processor as a MPU. Generally RISC processors run fast when instructions and data are fed linearly and continuously through the pipeline and cache. However, many interrupts occurring in embedded applications force frequent context switching which causes pipeline stalls and saving many registers to memory. Moreover, current thread-based operating systems frequently execute context switching between threads, causing the same effects. Therefore RISC processors are handicapped for embedded and control applications. Thus RISC processors are rarely used to control robots.

To avoid the above problems, we chose a SPARC processor to take advantages of its register windows to dramatically increase the computing performance compared with CISC. At the same time, we maintained high interrupt processing performance.

![Figure 3: Register Windows of SPARC](image)

In the SPARC family, register windows (general purpose registers that are cyclically overlapped) are provided to execute procedure calls at high speed (Figure 3) as follows:

1. Output parameters are assigned to the out registers.
2. The SAVE operation is executed.
3. The CWP is decremented. (The current window is moved down to the neighboring window.)

The input parameters are held at the in registers of the new current window and can be used by the new current window. Likewise, the procedure return is processed as follows:

1. The return value is assigned to the in register.
2. The RESTORE operation is executed.
3. The CWP is incremented. (The current window is moved up to the neighboring window.)
We apply the register windows to interrupt processing and context switching, and try to reduce the penalty caused by interrupts, context switching, and so on.

Preparation for interrupts requires more than just making available an empty register window. When an interrupt occurs on a SPARC processor, the CWP is decremented by the hardware, and then a register window is prepared for the interrupt (Figure 3). That is to say, interrupt processing can be performed only by switching the register windows. So it is basically not necessary to save registers to the stack, which is an overhead. Therefore, having an operating system provide a suitable register windows schedule allows frequent interrupts and context switching to be processed at high speed.

4.5 Mother Module

A diagram of a responsive controller is shown in Figure 4.

We use SPARCLite (MB86934) as a processor, because it has built-in vector registers that are efficient at floating point operations (over 60Mflops), which are especially effective in image processing.

In designing its memory system, we use 64 bits synchronous DRAM with an access speed of 10 ns, so as to ensure sufficient memory bandwidth to realize real-time image processing.

Communication ports are mainly designed and implemented by emitter coupled logic (ECL) with a latency of 170 ps to realize very fast communication. We use differential interfaces for tolerant to electronic noise, as shown in Figure 2.

4.6 I/O Card

We designed the following I/O cards to implement various applications:

- MIN card;
- General purpose motor control card;
- Sensor card (A/D converter card); and
- PCMCIA card.

Personal computer I/Os including SCSI, Ethernet, etc., are available using PCMCIA cards.

4.7 Applications

It is easy to re-configure a system because only connecting the responsive link cables among responsive controllers makes the system. Therefore many kinds
of robots including humanoids (Figure 5), cart robots, etc., can be composed and controlled by the responsive controllers.

We will apply the responsive controllers to the next version of ET1-Humanoid [5] and an office conversant robot Nyo-1 [6].

Since the responsive controller is very small and easy to connect, it can be easily embedded in the wall, so that it is available as the controller of office automation, home automation, factory automation, etc.

5 Conclusion

In this paper, we proposed a functionally distributed responsive micro controller for distributed processing, which could control responsive robots and home automation, etc. A common mother module was composed of responsive links, a processor, a memory system, and an I/O card bus interface for its daughter card. A responsive controller was composed of the mother module and the I/O card that is specific for its application. Many kinds of systems can be flexibly composed of responsive controllers connected by responsive links.

In fact, there are some problems with constructing a practical system, such as a humanoid, using current responsive controllers. For example, the size of a responsive controller is not small enough, and two kinds of power resources (23V) are needed. Therefore, we are now researching and developing an ASIC into which can be integrated almost all the functions in Figure 4. That is to say, we are integrating a MPU(SPARC) core, responsive links, SDRAM I/Fs, an interrupt controller, A/D converters, D/A converters, timers, counters, PWM generators, SIOs, PIOs, etc., into an ULSI chip called a responsive processor, as shown in Figure 6.

Acknowledgments

We thank to Motoki Suwa, the head of information science division, Nobuyuki Ohsu, the head of machine understanding division, and Kazuhito Ohmaki, the head of COE lab. They gave us a chance to study a responsive controller and supported our study.

References


