Proposal of CSP based Network Design and Construction

Session:Network and Protocol
Short Paper

Kazuto Tanaka, Satoshi Iwanami, Takeshi Yamakawa, Chikara Fukunaga*
Tokyo Metropolitan University, Hachioji, 192-0397 Japan

Kazuto Matsui
Prominent Network Inc., Tokyo, 104-0032 Japan

Takashi Yoshida
Smartscape Inc., Tokyo 151-0051 Japan

Abstract

We have designed a network router suited to be used in a SpaceWire network. The design was based on a formal method using CSP (Communicating Sequential Processes). The performance of the router has been verified by implementing it in a network with several processors [1]. In this paper we discuss the reason why we use CSP as a formal design method for the router-network system, actual design and refinement procedure. Discussions are emphasized on necessity of a formal method like CSP to construct and develop a secure system such as devices used in SpaceWire network system.

1 CSP model

As bit serial interconnect protocol adopted by SpaceWire allows us to form flexible network topology, we can construct the best optimized network system for a particular space mission. We can make also a fault-tolerant system on top of that by adding several duplicated network path rather than minimal topology. Many processors in the system should process and analyze data inputted concurrently from the front-end devices, and accurate information as a result of data analysis should be feed-backed to the front-end in a real time manner. Since a network comprises several active processors as well as measurement devices , the network system itself can be regarded as a parallel (concurrent) system with many processes. We have to, therefore, design the hardware system and application system with extremely strict care to avoid resource conflicts among processes.

*Correspondong author:`fukunaga@tmu.ac.jp`

Formal (design) methods are known to be very effective to establish a model based on the specification in a mathematical way and to verify the model meets the specification consequently. With a formal method we try to express a system specification in algebraic, symbolically logic or graphically theoretical way. Although each method uses different way to design a model, but usually every method has some algorithm to develop the model (namely to express the model in a different expression but consistently in a mathematical manner). The model developed in this way is used for the refinement or design verification. If we describe a model using a some formal method, and the model is verified equivalent as the original specification, then the implemented model after some refinement is thought to be verified in the design description level.

Communicating Sequential Processes (CSP) [2] is one of formal methods, and is known to be effective for description of a concurrent (parallel) system in which many processes (processors) work in parallel. We can express synchronization and concurrent processing in a simple way with CSP. In a parallel system, however, each process is described in a sequential manner, and a parallel processing between processes is achieved with synchronized communication between them with a concept of channels. Processes are shared their commonly used variables through the channel communication but they never keep shared memory. The real time condition of the system is also satisfied because of the inherent event driven nature embedded in CSP as a channel communication.

There is a verification tool to point out the existence of some failures (livelock, deadlock and/or divergence) in a system described in CSP, which is called FDR2 (Failure Divergence Refinement Version 2) [3]. With this refinement application, we can validate a model description automatically from the view point of existence of non-determenistic parts, namely livelocks or deadlocks. FDR2 is, hence, a relevant tool for system designers who use CSP for system modeling. In the following section we demonstrate the usage of the tool for our network design.

2 Crossbar switch example

There are algebraic notations (concepts) of CSP description. Among them relevant concepts are introduced here in order to follow the discussions given below;

1. parallel processing : $p \,|[\, a, b \,]|\, q$ which means processes p and q are running in parallel in which common parameters for two processes are exchanged using channels a, b,

2. channel input $c?x : a \to p$ means variable x of type a is inputted through channel c, then proceeds to process p, and $c!v \to q$ means variable v is output through channel c, and then proceeds to process q,

3. sequential processing : $p \ ; \ q$ means a sequential processing, namely process q is followed after the completion of p,

4. $p \mid\mid\mid q$ means two processes p and q are running in parallel without channel communication, this is called interleave, and $\mid\mid\mid_{i \in T} p_i$ means the extended version of interleave in which all processes $p_i, i \in T$ runs independently in parallel, and

5. data series is expressed as $s = \langle a, b \mathinner{..} n \rangle$, and two functions; *head* and *tail* work as $tail(s) = \langle b \mathinner{..} n \rangle$ while $head(s) = a$.

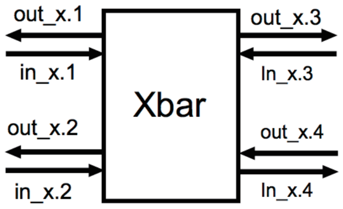

Figure 1: Four port crossbar switch with input/output channel naming

Based on these building blocks, we can design, for example, a primary crossbar switch (process xbar) that is indicated in fig. 1. In order to keep discussion simple, we assume the number of bi-directional ports is restricted to four($Tag = \{1, 2, 3, 4\}$) and the destination port number is stored in the header part of data packet ($packet = \langle port\ number, data \rangle$).

$$xbar(in, out) = \mid\mid\mid_{i \in Tag} INOUT(i, in, out), \tag{1}$$

$$\begin{aligned} INOUT \quad & (i, in, out) = in.i?packet \rightarrow \\ & out.head(packet)!tail(packet) \rightarrow INOUT(i, in, out) \end{aligned} \tag{2}$$

Then we implement the actual system model as

$$SYSTEM = xbar(in\_x, out\_x), \tag{3}$$

where both $in\_x(in)$ and $out\_x(out)$ are arrays of range $Tag = \{1, 2, 3, 4\}$ and the k-th element is specified like $in.k$. The model has been checked with FDR2 and was found neither deadlock, divergent failure (livelock) nor non-deterministic process.

As the next step of the development, we have added a broadcast besides one-to-one switching. The broadcast means the same data(message) from a particular channel (i) is relayed to all the ports except the source ($Tag \setminus \{i\}$). This broadcast can be achieved in CSP by modifying (2) as

$$\begin{aligned}
&\mathit{INOUT} && (i, in, out) = in.i?packet \rightarrow \\
&\quad \text{if} && head(packet) == 0 \text{ then} \\
& && |||_{j \in Tag \setminus \{i\}}\ out.j!tail(packet) \rightarrow \mathit{Skip};\ \mathit{INOUT}(i, in, out) \\
&\quad \text{else} && \\
& && out.head(packet)!tail(packet) \rightarrow \mathit{INOUT}(i, in, out),
\end{aligned} \tag{4}$$

where *Skip* indicates that it reaches (successful) termination. We introduce the port number zero for the broadcast. We bring an "*if* ... *else*" statement to judge one-to-one or broadcast switching. In this way (4) has been naturally derived from (2). We have also confirmed the validity of this model with FDR2.

If it is necessary not to send the same message to all the port (broadcast), but to transfer it to some selected channels, we should use a cascade link of two crossbar switches as shown in fig. 2. Even if we specify the broadcast connection in the second router, we send the message to three out of five channels. The CSP description of this model can be written from (3) by simply taking into account the parallel processing of two crossbar switches as

$$\mathit{SYSTEM} = xbar(in\_x, out\_x) \,[\![\, Left\_ch, Right\_ch \,]\!]\, xbar(in\_x', out\_x') \tag{5}$$

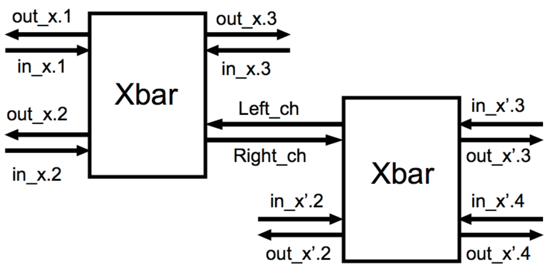

Figure 2: Cascade link of two crossbar switches. We can make one-to-many switching to the channels x'.2, x'.3 and x'.4 from any other channels with this circuit.

References

[1] K.Tanaka et al., "The design and performance of SpaceWire Router-network using CSP", presented in this conference.

[2] C.A.R.Hoare, "Communication Sequential Processes", Prentice-Hall Inc., 1985

[3] Formal System Inc., "Failure Divergence Refinement (FDR) 2", http://www.fsel.com