Annals of Nuclear Energy 46 (2012) 56–62

Contents lists available at SciVerse ScienceDirect

Annals of Nuclear Energy

journal homepage: www.elsevier.com/locate/anucene

Study on deterministic response time design for a class of nuclear Instrumentation and Control systems

Chang-Kuo Chen [a,*], Yi-You Hou [b], Cheng-Long Luo [a]

[a] Institute of Nuclear Energy Research, No. 1000, Wenhua Road, Chiaan Village, Longtan Township, Taoyuan County 32546, Taiwan, ROC
[b] Department of Electrical Engineering, Far East University, No. 49, Zhonghua Road, Xinshi District, Tainan City 74448, Taiwan, ROC

ARTICLE INFO

Article history:
Received 2 November 2011
Received in revised form 5 March 2012
Accepted 11 March 2012
Available online 12 April 2012

Keywords:
Deterministic response time
Computer-based systems
Petri nets
Sequence diagrams
Linear logic

ABSTRACT

This study is concerned with a deterministic response time design for computer-based systems in the nuclear industry. In current approach, Petri nets are used to model the requirement of a system specified with sequence diagrams. Also, the linear logic is proposed to characterize the state of changes in the Petri net model accurately by using symbolic time representation for the purpose of acquiring deterministic behavior. An illustrative example of the bistable processor logic is provided to demonstrate the practicability of the proposed approach.

1. Introduction

In recent years, computer-based systems have been widely applied to safety-critical and safety-related systems for the purpose of control, protection and monitoring of Nuclear Power Plants (NPPs) (Zio and DiMaio, 2009). In such systems, the response time is the most important ingredient for safety requirements. Furthermore, the target reliability figures for these systems (e.g. deterministic behavior) should be fulfilled in accordance with the specific property of a program because this guarantees that the system will be completely executed within a specific time frame. Therefore, the deterministic response time becomes one of the most significant issues for computer-based systems (IEEE, 1991; USNRC, 1997). In order to achieve this target, a timing budget should be allocated to the system architecture for demonstrating the feasibility of design timing, and timing requirements should be satisfied by design commitments so that the whole system can fulfill its timing requirements.

On the other hand, the qualification of a generic Programmable Logic Controller (PLC) platform was given by EPRI TR-107330 (EPRI, 1996). It is especially important for the determination of the response time to be dealt with taking into account the effect of input filtering, the I/O data acquisition time, the input access time for bus transfer to the main processor, twice the execution cycle time for an application program, the output access time for bus transfer to I/O boards, and the I/O data distribution time.

Normally, the behavior of computer-based systems can be depicted by means of dynamic diagrams. The most widely used dynamic diagrams are in Unified Modeling Language (UML), a sequence diagram which is based on the sequence of actions that occur in a system. It expresses the interaction of objects according to a time sequence and shows how objects interact with others in a particular use scenario (Bernardi and Merseguer, 2007). However, UML has its limitations when it is used for specifying computer-based systems, such as occurs in the simultaneous execution of several scenarios. Consequently, the limitations are often addressed by combining UML with a formal method.

Petri nets (PNs) are a prevailing formal model which has long been confirmed to be suitable for describing and analyzing discrete-event systems with significant advantages. For example, their graphical nature can visualize firing sequences via tokens passing over the net, and they have well-established formal methods for modeling and analysis of systems as well as the mathematical base for analyzing structural and dynamic behaviors of a system. Therefore, the theoretical analysis and realization of PNs have been thoroughly developed and studied (Murata, 1989; Giua and DiCesare, 1994; Zhou and Jeng, 1998; Lee and Seong, 2004; Németh et al., 2009).

Because some structural concurrencies of Petri net fragments cannot be expressed, linear logic is applied with the equivalence being between reachability in a Petri net and the provability of some linear logic sequents. Linear logic was proposed by Girard

* Corresponding author. Tel.: +886 3 4711400x6312; fax: +886 3 4711415.
E-mail address: changkuochen@iner.gov.tw (C.-K. Chen).

0306-4549/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.anucene.2012.03.010

(1987) as a refinement of first-order logic to take into account the concept of producing and consuming resources. It has been defined in the framework of sequent calculus particularly to deal with resources as logical propositions. Atoms corresponding to logic propositions may be counted, produced and consumed exactly like tokens in Petri net places (Brown, 1989; Engberg and Winskel, 1990; Girault et al., 1997; Pardin-Chézalviel et al., 1999; Nicolas et al., 2001).

In order to meet the timing requirements of the entire systems, such as deterministic behavior, an efficient design procedure is proposed in this study. Petri nets are used to model the requirements of a system initially specified with sequence diagrams. In order to achieve deterministic behavior, linear logic is proposed to describe state changes of the PN model precisely with symbolic time representation. Finally, the practicability of the proposed approach is demonstrated by an illustrative example of the bistable processor logic.

The rest of this study is organized as follows: In Section 2, the basic concepts of system modeling are presented. Section 3 describes the deterministic response time design. Section 4 is an illustrated example of the bistable processor logic which demonstrates the practicability of the proposed approach. At last, it has a brief conclusion in Section 5.

2. System modeling

2.1. Basic Petri nets

A Petri net is defined as a particular kind of bipartite directed graph using three kinds of objects: places, transitions, and directed arcs connecting places to transitions and transitions to places. The places and transitions are pictured using circles and bars/boxes, respectively. A basic Petri net is formally defined as

$$PN = (P, T, I, O), \tag{1}$$

where $P = \{p_1, p_2, \ldots, p_n\}$, $n > 0$ is a finite set of places;

$T = \{t_1, t_2, \ldots, t_m\}$; $m > 0$ is a finite set of transitions with $P \cup T \neq \mathrm{O}$ and $P \cap T = \mathrm{O}$; $I{:}P \times T \to Z^+$ is an input function defined the set of directed arcs from P to T in which Z^+ is a set of nonnegative integers.

$O{:}P \times T \to Z^+$ is an output function defined the set of directed arcs from T to P.

In addition, either none or a positive number of tokens is potentially held in each place as drawn by small solid dots which are referred to as the *marking*. A Petri net which contains tokens is called a *marked Petri net*, and its transitions can be enabled and fired immediately. In a basic Petri net, an enabled transition t removes tokens from its input places and deposits them on its output places when it fires at a marking M; the new marking M' is given as

$$M'(p_i) = M(p_i) + O(p_i, t) - I(p_i, t) \quad \text{for } i = 1, 2, \ldots, n, \tag{2}$$

where $M{:}P \to Z^+$ is a marking in which the ith component represents the number of tokens in the ith place. A transition $t \in T$ is enabled, if and only if $M(p) > 0$ when $I(p, t) = 1$, $\forall p \in P$.

Several properties of a Petri net permit the system designer to identify the presence or lack of the specific functional properties under system design in the framework of the modeled system. Those properties, including reachability, boundedness, conservativeness, liveness and reversibility, can be analyzed by some validation methods, for example a reachability graph, invariant analysis, or reduction and simulation (Zhou and Jeng, 1998).

2.2. Transformation of sequence diagrams into Petri nets

The sequence diagram is one of Unified Modeling Language (UML), which is used primarily to describe the interactions between objects in a sequential order. Also, one of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. In order to obtain a model available for requirement validation, the sequence diagrams are translated into Petri nets, which allows the performance of a formal check of desirable properties in different kinds of systems.

The method and abstract data of inter-object communication give a high level view of the system requirements. Therefore, the transformation of sequence diagrams into Petri nets mostly depends on the message that links up the objects to each other in such a context (Pettit and Gomaa, 2006; Soares et al., 2008). In addition, the behavior of each operation in an object can be described in a similar way. More detailed justification is presented in Chen (2011). Specifically, three types of interpretations of a UML message in inter-object communications are described at sub-system levels and are represented as follows.

2.2.1. Asynchronous communication

The transformation of an asynchronous communication between the sequence diagram and the Petri net model is shown as Fig. 1. In the Petri net model, a communication is made up by means of a shared place that is referred to as an *outcome place* from the sender object (*object A*) and an *income place* from the receiver object (*object B*). The sender asks for an operation to the receiver by the call "pa" that puts a token in a shared place and after that continues with other operations.

2.2.2. Highly synchronous communication

The well-known remote procedure protocol which is equivalent to the highly synchronous message in which two shared places are implemented, one for the call "ps" and another for the return "pr" as shown in Fig. 2. The return "pr" is equivalent to the result return or acknowledgment.

2.2.3. Loosely synchronous communication

As shown in Fig. 3, a loosely synchronous message is similar to one with more synchronicity with a slight difference. The sender just needs the acknowledgment of the request, and then its own thread of control is continued while the receiver performs its operation. It does not need the result of the operation executed by the receiver.

2.2.4. Reduction rules for Petri nets

Because some Petri nets places which constructed from the sequence diagrams are redundant, in the sense that their removal does not change the behavior of the original model. It is possible to eliminate some of the communication and waiting places by using the reduction rules in Petri net theory after the translation from a sequence diagram into a PN model (Murata, 1989).

Due to the unpredictable conditions such as deadlock, a PN model obtained through the construction above does not represent a proper model from which a final state cannot be approached. Therefore, the behavior of PN model should be included to be further refined and restricted by the PN analysis (Zhou and Jeng, 1998).

3. Deterministic response time design

3.1. Response time design with linear logic

In order to design and analyze the deterministic response time of finite operations of any Petri net, linear logic is proposed. Based