Formal Analysis and Verification of a Communication Protocol

XIN BEN LI, DE CHAO SUN
Department of Computer Science and Technology
Zhejiang Wanli University
8 Qianhu South Road, Ningbo, Zhejiang, 315100
PEOPLE'S REPUBLIC OF CHINA

Abstract: - Reliability is one of the major concerns in the design of embedded systems. Formal verification by model checking is of a great advantage in verifying the correctness of computer system, whether they are hardware, software or a combination. The paper reports the design and development of an intelligent telephone alarm that is a single-chip computer system, and the formal verification of communication protocol is described based on the model checker NuSMV.

Key-Words: - Model check; Communication Protocol; NuSMV; Single-chip Processor

1 Introduction

Reliability plays an important role in the design of embedded systems or other electronic systems, especially for the safety-critical or mission-critical systems. Formal verification is one of the methods that can help the designer to test the properties of the system and debug the errors during the course of system development, so that the correctness of the design can be guaranteed.

As for formal verification techniques, Michael Huth[1], from logics' application point of view, has classified the approaches into proof-based and model-based, and pointed that model checking is an model-based property-verification approach and is used mainly for concurrent and reactive systems. W. Marrero and J. Wen [2,5] applied model checking to the analysis of security protocols that consist of a sequence of messages with encrypted parts. After analysis some applications of formal verification to functional requirements (specifications) of circuits, I. Pill etc. in [6] noted that the crucial activity of producing an implementation satisfying given properties is the quality enhancement of the specifications before the design phase, and presented techniques and guidelines to explore and assure the quality of a formal specification.

This paper reports the design and development of an intelligent telephone alarm that is based on the single-chip computer system, and the emphasis is placed on the formal analysis and verification of the communication protocol by using the model checker NuSMV.

2 Design Requirements of the Alarm

For most office or home used alarms, such as the smoke detector & alarm device, a common characteristic is that it can work quite well on the spot if something happens (such as the fire). But the situation occurred might not be handled immediately and properly if nobody is in, especially in the evenings. The reason is that such kind of alarms cannot give an alarm remotely. Based on the demand of the market, a new type of intelligent alarm and control system has been designed that functions as follows:

- It can sample the information from the different alarm sources, decision its actions and, if necessary, give the alarm signal on the spot and to the remote agent via the public telephone line.
- The alarm message delivered can be audio or digital signals depending on the user's choice. The four pre-set telephone numbers are calling one by one until one of them give the answer.
- User can remotely reset and set the calling numbers and safety password using his telephone if the telephone number has changed. User can also control the state of the alarm to open or close using his telephone.

The system consists mainly of the sensor signal process circuit, dual tone multi-frequency (DTMF) send/receive circuit, audio control & record/play circuit etc. Since most of functions for the communication and control are carried out by the DTMF send and receive circuit, we will focus on this part. On the top of the Fig.1 is the single-chip processor 89C51 that is the center of control and information process. The 5087 is a DTMF chip and play the sender' role, i.e., it receives the 89C51's 8-bit output signals (corresponding to one dial keypad number), transforms the dial number into the DTMF signals and transmits the signals to remote

telephones. The chip 8870, in the bottom left-hand corner of the Fig1, is the DTMF receiver, i.e., it receives the audio signals from remote telephones and decodes them into four bits for reading by 89C51. Note that the eight output ports of 74LS374 are connected into the input C1-C4 and R1-R4 of 5087 chip, which play the keypad's role of a telephone, and 89C51's 8-bits output is connected to the input ports of 74LS373, so that the CPU can drives the chip 5087 sending the audio signals to remote telephone. So, the sending path of the message is formed from 89C51, 74LS373, and 5987 to telephone via PSTN. Similarly the receiving path of message can be formed.

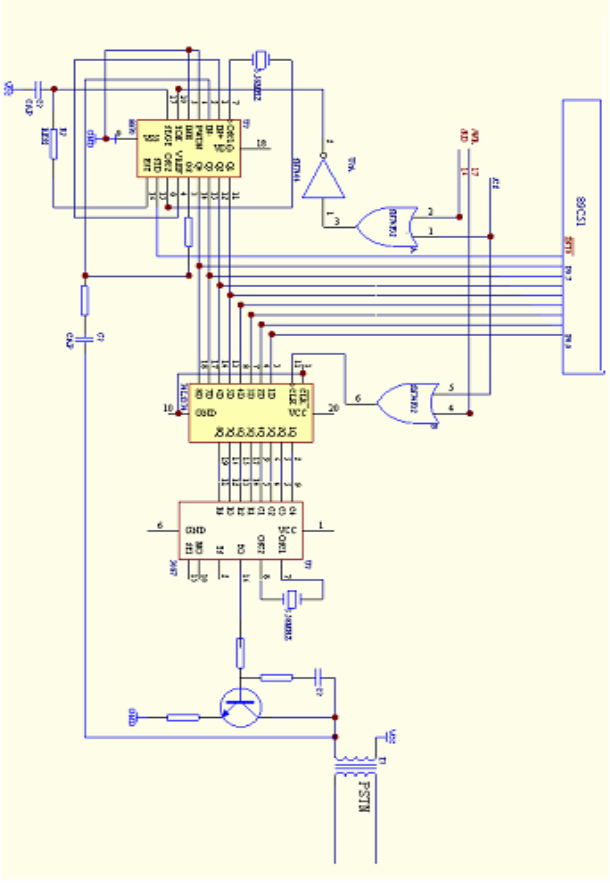

Fig. 1 Send and Receive Circuit

Due to the reliable requirements of the message delivery, a communication protocol is designed and its correctness needs to be verified by using some formal tool, such as NuSMV. In the following section, we will focus on the analysis and verification of the protocol.

3 Verification and Analysis of the Communication Protocol Model

3.1 Model checker NuSMV

Model checking is an automatic, model-based, property-verification approach for formal verification. In model checking, the models $\mathcal{M}$ are transition systems and the properties ϕ are formulas in temporal logic. To verify that a system satisfies a property, we must do three things: the first is to model the system using the description language of a model checker, arriving at a model $\mathcal{M}$; then to code the property using the specification language of the model checker, resulting in a temporal logic formula ϕ; and the third is to run the model checker with inputs $\mathcal{M}$ and ϕ. The model checker can output the answer 'yes' if $\mathcal{M} \models \phi$ and 'no' otherwise; In the latter case, most model checkers also produce a trace of system behavior that causes this failure. This automatic generation of such 'counter traces' is an important tool in the design and debugging of systems.

NuSMV stands for 'New Symbolic Model Verifier.' [3,4] It is a software tool for the formal verification and allows checking finite state systems against specifications in the temporal logic, including linear-time temporal logic (LTL) and computational tree logic (CTL). The input language of NuSMV is designed to allow the description of finite state systems that range from completely synchronous to completely asynchronous. The NuSMV language provides for modular hierarchical descriptions and for the definition of reusable components. The basic purpose of the NuSMV language is to describe (using expressions in propositional calculus) the transition relation of a finite Kripke structure. Following are the formal definitions about the syntax and semantics of one of temporal logic CTL, i.e., Computation Tree Logic.

Definition 3.1 The syntax of CTL formulas can be given in Backus Naur form as follow:

$$\phi ::= \vee \mid \mathrm{T} \mid p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid$$
$$(\phi \rightarrow \phi) \mid \mathrm{AX}\phi \mid \mathrm{EX}\phi \mid \mathrm{AF}\phi \mid \mathrm{EF}\phi \mid$$
$$\mathrm{AG}\phi \mid \mathrm{EG}\phi \mid \mathrm{A}[\phi \mathrm{U} \phi] \mid \mathrm{E}[\phi \mathrm{U} \phi]$$

Where p ranges over a set of atomic formulas.

Notice that each of the CTL temporal connectives is a pair of symbols. The first of the pair is one of A and E. A means 'along All paths' (*inevitably*) and E means 'along at least (there Exists) one path' (*possibly*). The second one of the pair is X, F, G, or U, meaning 'neXt state,' 'some Future state,' 'all future states (Globally)' and 'Until', respectively. Notice that AU and EU are binary. The symbols X, F, G and U cannot occur without being proceeded by an A or an E; every A or E must have one of X, F, G, and U to accompany it.

Definition 3.2 A transition system M =(S, →， L) is

a set of states S endowed with a transition relation → (a binary relation on S), such that every s∈S has some s'∈S with s→s', and a labelling function L:S →P(Atoms).

Transition systems are also simply called models. So a model has a collection of states S, a relation →, saying how the system can move form state to state, and, associated with each state s, one has the set of atomic propositions L(s) which are true at that particular state. Here P(Atoms) stands for the power set of Atoms, a collection of atomic descriptions.

CTL formulas are interpreted over transition systems. Let M=(S, →, L) be such a model, s∈S and Φ a CTL formula. The definition of whether M, s|= Φ holds can be given formally as following definition.

Definition 3.3 Let M =(S, →, L) be a model for CTL, s in S, Φ a CTL formula. The relation M, s|= Φ is defined by structural induction on Φ:

(1) M, s|=T and M, s|≠⊥
(2) M, s|=p iff p∈L（s）
(3) M, s|=∅ Φ iff M, s|≠ Φ
(4) M, s|= $\Phi_1 \wedge \Phi_2$ iff M, s|= Φ1 and M, s|= Φ_2
(5) M, s|= $\Phi_1 \vee \Phi_2$ iff M, s|= Φ1 or M, s|= Φ_2
(6) M, s|= $\Phi_1 \rightarrow \Phi_2$ iff M, s|≠ Φ1 or M, s|= Φ_2
(7) M, s|=AX Φ iff for all s1 such that s→s1 we have M, s1|= Φ. Thus, AX says: 'in every next state.'
(8) M, s|=EX Φ iff for some s1 such that s→s1 we have M, s1|= Φ. Thus, EX says: 'in some next state.' E is dual to A --- in exactly the same way that $ is dual to " in predicate logic.
(9) M, s|=AG Φ holds iff for all path s1→s2→s3 →…, where s1 equals s, and all si along the path, we have M, si|= Φ. That is, for ALL computation paths beginning in s the property Φ holds Globally, and 'along the path' include the path's initial states s.
(10) M, s|=EG Φ holds iff there is a path s1→s2 →s3→…, where s1 equals s, and all si along the path, we have M, si|= Φ, i.e., there Exists a paths beginning in s such that Φ holds Globally along the path.
(11) M, s|=AF Φ holds iff for all path s1→s2→s3 →…, where s1 equals s, there is some si such that M, si|= Φ. That is, for ALL computation paths beginning in s there will be some Future state where Φ holds.
(12) M, s|=EF Φ holds iff there is a path s1→s2→ s3→…, where s1 equals s, and for some si along the path, we have M, si|= Φ, i.e., there Exists a computation path beginning in s such that Φ holds in some Future state;
(13) M, s|=A[Φ1U Φ2] holds iff for all paths s1 →s2→s3→…, where s1 equals s, that path satisfies Φ1U Φ2, i.e., there is some si along the path, such that M, si|= Φ2, and for each j<I, we have M, sj|= Φ1.
(14) M, s|=A[Φ1U Φ2] holds iff there is a path s1 →s2→s3→…, where s1 equals s, and that path satisfies Φ1U Φ2, as specified in (13).That means there Exists a computation path beginning in s such that Φ1 Until Φ2 holds on it.

3.2 NuSMV description of the Protocol

The protocol transmits the alarm messages along the sending path to the remote telephones. Since the four remote telephones may have no person near by to answer and result in the loss of messages, the message will be send again and again until one of telephone gives a answer. This means that the protocol guarantees the communication between the sender and the receiver being successful.

The protocol works as follows. There are four entities, or agents: the sender, the receiver, the message path and the acknowledgement path. The sender transmits the first part of the message together with the 'control' code *. If, and when, the receiver receives the message, it sends code * along the acknowledgement path. When the sender receives this acknowledgement, it sends the next packet with the control bit #. If and when the receiver receives this, it acknowledges by sending a code # on the acknowledgement path. By alternating the control code, both receiver and sender can guard against duplicating messages and losing messages (i.e., they ignore messages that have the unexpected control code).

If the sender doesn't get the expected acknowledgement, it continually resends the message, until the acknowledgement arrives. If the receiver doesn't get the message with the expected control code, it continually resends the previous acknowledgement.

Fairness is also important for the protocol. It comes in because, although we want to model the fact that the path can lose messages, we want to assume that, if we send a message often enough, eventually it will arrive. In other words, the channel cannot lose an infinite sequence of messages. If we did not make this assumption, then the channel could lose all messages and, in that case, the protocol would not work.

For our case of the communication protocol, we assume that the text to be sent consists of a telephone number with the alarm message, which are sent sequentially. The variable message1 is the current number of the message being sent, whereas message2 is the control signal. The definition of the module Sender is given In Fig. 2.

```
MODULE sender(ack)
VAR
   St          :{sending, sent};
   Message1 :boolean;
   Message1 :boolean;
ASSIGN
   Init(st)   := sending;
   Next(st) :=
         case
            Ack = message2 & !(st=sent) :sent;
            1                                  :sending;
         esac
   next(message1) :=
         case
            st = sent :{0,1};
            1          :message1;
         esac
   next(message2) :=
         case
            st = sent : !message2;
            1          :message2;
         esac
FAIRNESS running
SPEC AG ! (sender.st = sent → EF receiver.st=received)
```

Fig.2 the sender in NuSMV

This module spends most of its time in st=sending, going only briefly to st=sent when it receives an acknowledgement corresponding to the control code of the message it has been sending. The variables message1 and message2 represent the actual data being sent and the control code, respectively. On successful transmission, the module obtains a new message to send and returns to st=sending. The new message1 is obtained non-deterministically (i.e., from the sensors); message2 alternates in value. We impose FAIRNESS running, i.e., the sender must be selected to run infinitely often. The SPEC tests that we can always succeed in sending the current message.

The module receiver is programmed in a similar way. Another two modular is about the two paths. The acknowledgement path is an instance of the chan1. Its lossy character is specified by the assignment to forget. The value of input should be transmitted to output, unless forget is true. The sending channel chan2, used to send messages, is similar. Again, the non-deterministic variable forget determines whether the current code is lost or not. Either both parts of the message get through, or neither of them does (the channel is assumed not to corrupt messages).

Finally, we tie the four modular together with the modular main whose role is to connect together the components of the system, and giving them initial value of their parameters, see Fig. 3.

```
MODULE main
VAR
   s: process sender(ack-chan.output);
   r: process receiver(msg-chan.output1, msg-chan.output2);
   msg-chan :process chan2(s.message1, s.message2);
   ack-chan :process chan1(r.ack);
ASSIGN
   init(s.message2)   := 0;
   init(r.expected)   := 0;
   Init(r.ack)          := 1;
   Init(msg-chan.output2)   := 1;
   Init(ack-chan.output)     := 1;

SPEC AG ! (sender.st = sent → EF receiver.st=received)
```

Fig.3 The main in NuSMV

The paths have fairness constraints that are intended to model the fact that, although paths can lose message, we assume that they infinitely often transmit the message correctly.

After programmed all the source code of the protocol's model and its key properties into the file dtmf.smv, we can run NuSMV and debug the protocol. At last the protocol satisfies with the specifications what we expected, although several failures occurred during the beginning of the verification.

4 Conclusion

As a model-based formal verification tool, NuSMV is quite powerful and useful in the verification of the systems' design deficiency. The paper reports the design and development of an intelligent telephone alarm that is a single-chip computer system, and the formal verification of communication protocol is described based on the model checker NuSMV. Our experience showed that modeling system properly and specifying the properties to be verified precisely are key steps for designers, and the results demonstrated that the design of our communication protocol is proper and the system functions correctly.

References:

[1] Michael Huth and Mark Ryan. *Logic in Computer Science---Modeling and Reasoning about Systems*, Cambridge University Press, 2004.

[2] W. Marrero, E. Clarke, S Jha. *Model Checking for Security Protocols*. In DIMACS Workshop on Design and Formal Verification of Security Protocols, 1997.

[3] A. Cimatti, E. M. Clarke, E. Giunchiglia, etc.. *NuSMV 2: An Open Source Tool for Symbolic Model Checking*, In Proceeding of International Conference on Computer-Aided Verification (CAV 2002). Copenhagen, Denmark, July 27-31, 2002.

[4] R. Cavada, A. Cimatti, C. A. Jochim, etc.. NuSMV 2.4 User Manual. http://nusmv.irst.itc.it.

[5] J. Wen, M. Zhang, X. Li. *New Approach for Formal Analysing Encryption Protocols*, Computer Applications, 25(5): 135-145, May 2006.

[6] I. Pill, S. Semprini, R. Cavada, M. Roveri, R. Bloem, A. Cimatti. *Formal Analysis of Hardware Requirements*. International Journal of Computer Systems, Science and Engineering, 20(1): 19-35, January 2005.

[7] H. Zheng, C. Myers, D. Walter, s. Little, T. Yoneda. *Verification of timed Circuits with Failure Directed Abstractions*. IEEE Transactions of Computer-Aided Design of Integrated Circuits and Systems, Vol.25, No.3, pp403-412 (2006).