A FORMAL METHOD FOR VERIFYING THE IMPLEMENTATION OF SPW DATA-STROBE-ENCODING BY APPLYING THEOREM PROVING

Session: SpaceWire Test and Verification (Poster)

Long Paper

Liming Li

College of Information Engineering, Capital Normal University,

Beijing, China, 100048

Liya Liu

Dept. of Electrical and Computer Engineering, Concordia University,

Montreal, Quebec, Canada, H3H 1M8

Yong Guan, Yan Zhang

College of Information Engineering, Capital Normal University,

Beijing, China, 100048

Jie Zhang

College of Information Science & Technology, Beijing University of Chemical Technology, Beijing, China, 100029

Limin Tao

Beijing Engineering Research Center of High Reliable Embedded System

Beijing, China, 100048

E-mail: liliminga@126.vom, liy_liu@ece.concordia.ca,guanyxxxy@263.net, anythingroom@126.com, jzhang@mail.buct.edu.cn

ABSTRACT

For the requirement of building a highly reliable communication system, SpaceWire was applied in Space Solar Telescope (SST) project completed by National Astronomical Observatories, Chinese Academic of Sciences. This paper is based on part of work on SST project. In SpaceWire standard, Data-Strobe (DS) encoding was an encoding scheme for transmitting data in digital circuits. This study aimed to verify that the DS encoder circuit, which was developed for SST project, faithfully implemented the specification in SpaceWire standard. Equivalence checking was applied between the specification and implementation. With the aid of HOL tool, this

equivalence checking was carried out in a formal verification method, theorem proving. According to the requirements of the standard, a primitive recursive function was defined using Meta Language (ML) in order to specify the circuit. Then the components implemented in VHDL code were modeled with predicates. The result showed that the implementation is equivalent to the specification. It suggested that this DS encoder circuit implemented on FPGA can be applied reliably in the SST project.

1 INTRODUCTION

With the development of technology and science, the density of the circuits increases quickly. The extensive applications of the circuits are found in transportation systems, medical applications, defense systems, and so on, nearly all aspects. It is well known that the cost of a failure is unacceptably high [3], especially in aerospace safety domain. Any subtle error in the design might cause unexpected trouble under a particular set of conditions. It may mean the loss of life or serious impairment for many human beings, the disastrous consequences. Thus, to verify the correctness of the circuit becomes imperative.

Generally, designers try to ensure correctness through simulation and testing. However, it is impossible to test or simulate all the cases for large, complex systems. Furthermore, simulation and testing inputs are usually designed to detect only certain well-defined types of faults. Some corner cases might be ignored during the process of design, simulation and testing. And exhaustive simulation and testing are no longer possible because of the increasing complexity of the circuit. [3]

Formal methods have the capability of conducting precise system analysis and overcome the limitation described above. It is a technology to construct the system based on mathematical model formally. Theorem proving is one of the most commonly used formal methods. It allows to mathematically reason about system properties by representing the behavior of a system in higher-order logic in a general model. In this way, the specification and implementation are expressed as the general mathematical model so that all the cases are covered when they are proved to be equivalent.

In SST project, SpaceWire link addresses the handling of payload data and control information on boarding a spacecraft. DS encoder is the critical circuit part in SpaceWire standard for high speed data link. To verify that the design of this circuit is implemented reliably, theorem proving is applied. In chapter 2, the method is described in details. Then result is shown in chapter 3. Finally, it comes to the conclusion and presents the future work.

2 METHOD

For the purpose of verification, both the specification based on SpaceWire standard and the implementation based on the hardware are modeled abstractly. These two

models are called DS Encoding Specification and DS Encoding Implementation separately. Both of the models are defined formally utilizing ML (Meta Language) in higher-order-logic. The latter comes from the VHDL code and the gates are obtained based on their Boolean functionality. By applying the rules for reasoning in higher-order-logic, equivalence checking was applied between these two models.

2.1 DS ENCODING SPECIFICATION

In SpaceWire standard, Data-Strobe (DS) encoding is specified as the coding scheme which encodes the transmission clock with the data into Data and Strobe so that the clock can be recovered simply by implementing an exclusive OR operation on the Data and Strobe signals. It is illustrated in Figure 1. As it shows, Data are transmitted directly and the state of the Strobe signal changes whenever the data alters one data bit interval to the next. [1]

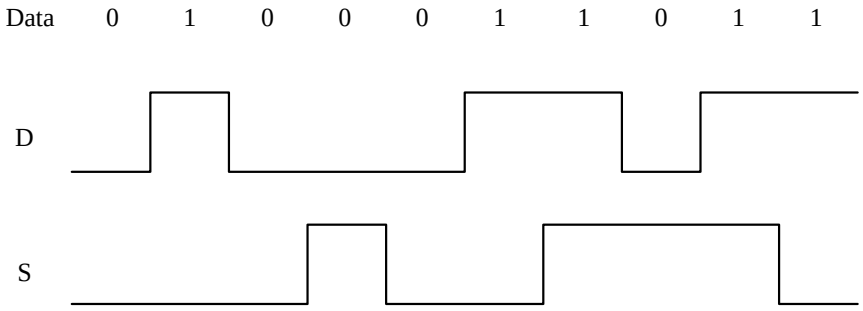

Figure 1 Data-Strobe (DS) Encoding

The major feature of DS encoding is that the Data are transmitted directly and the Strobe signal shall vary state whenever the Data does not change from one bit to the next. In order to demonstrate the signals in a general expression, in following definition, t denotes clock, and X t denotes the signal level at t time. Obviously, X (t+1) means the signal at t+1 time. In the process of reasoning, if the signal is "1", it equals to *True* (T); if it is "0", it equals to *False* (F). This feature can be modeled in abstract as the following properties.

Property 1:
If *reset* is T at time t, then the value out at *dataout* and *strobe* at time t are both F, i.e.
$\forall t.\ reset\ t \Rightarrow (dataout\ t=F) \wedge (strobe\ t=F)$ (1)
where $\forall t$ *means for all t;* $\Rightarrow$ *means implication.*

As described in SpaceWire standard , the data values are transmitted directly, the relation between data and strobe signal presents the fact that the output of the data signal should be 3 clock cycles later than the input. Therefore, this can be expressed in property 2:

Property 2:
If *reset* is T at time t+1 or t+2 or t+3, then the value output on *dataout* at time t+3 is F,

otherwise it is equal to the value input at time t on *datain*, i.e.

$\forall t.\ dataout\ (t+3) = if\ reset\ (t+1) \vee reset\ (t+2) \vee reset\ (t+3)\ then\ F\ else\ datain\ t$ (2)

Property 3:

If reset is T at time t+1, then the value output at *strobe* at time t+1 is F, otherwise, if *dataout* is equal at time t and time t+1, the value output at *strobe* at time t+1 is equal to the negation of the value at time t on itself otherwise the value at time t on itself, i.e.

$\forall t.\ strobe\ (t+1) =$

$if\ reset\ (t+1)\ then\ F\ else$

$(if\ dataout\ t = dataout\ (t+1)\ then\ \neg(strobe\ t)\ else\ strobe\ t)$ (3)

where ¬ means negation.

To state the specification property 3 in an intuitive logic expression, Xor operation is defined.

Definition 1: $\forall a\ b.\ Xor\ a\ b = \neg a \wedge b \vee a \wedge \neg b$

Therefore, (3) can be expressed using Xor as follows:

$\forall t.\ strobe\ (t+1) =$

$if\ reset\ (t+1)\ then\ F\ else$

$Xor\ (strobe\ t)\ \neg Xor\ (dataout\ (t+1))\ (dataout\ t))$ (4)

2.2 DS Encoding Implementation

The SpaceWire Encoding Circuit is designed with VHDL code. Only the process related to encoding is considered. For example, instead of *TX_ShiftReg*, *Datain* signal indicates the data transmitted. The simplified code is given in Figure 2:

```
IF (Clock'EVENT AND Clock = '1') THEN
    IF (Reset = '1') THEN
        D1          <= '0';
        D2          <= '0';
        DataOut     <= '0';
        St          <= '1';
        Strobe      <= '0';
    ELSE
        D1          <= Datain;
        D2          <= D1;
        DataOut     <= D2;
        St          <= St XOR NOT (D1 XOR D2);
        Strobe      <= St;
      END IF;
END IF;
```

Figure 2 Simplified code based on VHDL code

The approach to translating VHDL code into logic expression is not so straight. In addition, the proofs depend on the proper modeling and analysis of complex timing behavior. The tactics for modeling VHDL code in abstract have never been found in any referable paper or book. The method to model the VHDL code in behavior level is based on functions. Each function or any operation is modeled with predicates. All predicates are joined together with AND operator as shown in Figure 3. In such a way, the entire circuit can be modeled.

The predicate ONE is that for all times t the value of *out* is T.[2]
Definition 2: $\forall out.\ ONE\ out = \forall t.\ out\ t = T$

As known, has constructs to handle the parallel behavior designs, the value of the signal variable in the right-hand-side of the assignment statement is the one at the preceding time. For the purpose of expressing this behavior, REG is defined. The value of the output at time t+1 is that of the input at the previous time t, except at time 0. Since Time 0 refers to the exact time point that the device is power on, REG outputs F. [2]
Definition 3: $\forall inp\ out.\ REG\ (inp,out) = \forall t.\ out\ t = if\ t = 0\ then\ F\ else\ inp\ (t - 1)$

Similarly, *`IF…THEN…ELSE…`* can be modeled with predicate MUX. The input *sw* selects which of the other two inputs are to be connected to the output *out*.
Definition 4: $\forall sw\ in1\ in2\ out.\ MUX\ (sw,in1,in2,out) = \forall t.\ out\ t = if\ sw\ t\ then\ in1\ t\ else\ in2\ t$

The NOT gate is used to denote the situation that the value of *out* is always the negation of the value of *inp*.
Definition 5: $\forall inp\ out.\ NOT\ (inp,out) = \forall t.\ out\ t = \neg inp\ t$

Xor operation is modeled with a predicate XORING. This predicate states the case that the value of *out* is always *in1* XOR *in2* having no delay.
Definition 6: $\forall in1\ in2\ out.\ XORING\ (in1,in2,out) = \forall t.\ out\ t = Xor\ (in1\ t)\ (in2\ t)$

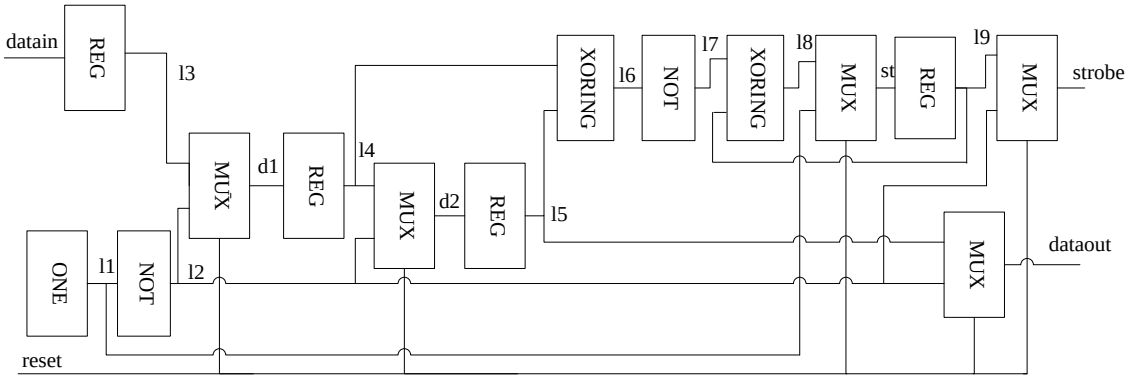

Figure 3 the connected predicates

Then *reset, datain, dataout, strobe, d1, d2* and *st* represent *Reset, Datain, Dataout, Strobe, D1* and *D2* in VHDL code, respectively. For the convenience of expression, *l1,*

l2, l3, l4, l5, l6, l7, l8 and *l9* are assumed as the temporary signal variables. The connected predicates are illustrated in Figure 3.

3 RESULT

Although the DS encoding circuit is small, it is worth doing verification on it because of its widely applications (The DS encoding scheme is also used in the IEEE Standard 1355--1995[4] and IEEE Standard 1394--1995 (Firewire) [5]). By applying the corresponding tactics and tacticals in HOL tool, the result showed the goal is proved, which suggested that the implementation of this circuit in FPGA hardware can imply the specification of this encoding scheme. So the design of the circuit can implement the behavior specified in the standard.

On the other hand, the efforts of the interactive verification work in a system might be huge. Our work compromises the merits of formal verification, simulation and testing. It suggests that a feasible scheme for verifying a safety system is applying theorem proving method on the critical and model checking method on the other parts.

4 CONCLUSION AND FUTURE WORK

By means of theorem proving with the aid of HOL tool, SpaceWire encoding circuit was verified and the result shows that the implementation is equivalent to the specification. Describing the behavior of the circuit with a general abstract model, it overcomes the limitation of simulation and testing. Furthermore, a new approach is proposed and proved to be usable in our work for modeling the VHDL code in abstract. More important, formal verification is expected be introduced into the process of design, since it is more helpful to ensure the design at the early stage of the product.

The successful verification on this small circuit is the first step of our investigation on applying the formal methods on the safety system. In the future, to verify some more circuits designed based on SpaceWire standard is our main aim so as to providing strong support to the correctness of our design in SST project.

Another plan for our future research is that more efforts will be done on investigating a compromise scheme for verifying the whole design applying formal methods in order to ensure the design in a critical security completed system.

5 REFERENCES

1. SpaceWire - Links, Nodes, Routers and Networks, ECSS Standard ECSS-E-ST-50-12A

2. The HOL System TUTORIAL (For HOL Kananaskis-4) , January 2, 2007

3. Michael C. McFarland, "Formal Verification of Sequential Hardware: A Tutorial", IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, VOL 12, NO. 5, May 1993

4. IEEE Std 1394-1995, IEEE Standard for a High Performance Serial Bus, 1995, ISBN 1-5593-7583-3

5. IEEE Std 1355-1995, Standard for Heterogeneous InterConnect (HIC), 1995, ISBN 1-55937-595-7