TELKOMNIKA, Vol. 11, No. 2, February 2013, pp. 740~746
ISSN: 2302-4046

On the Model Checking of the SpaceWire Link Interface

Wei Hua*[1,2], Xiaojuan Li1,[1,2], Yong Guan[1,2], Zhiping Shi[1,2], Jie Zhang[1,2], Lingling Dong[1,2]
[1]Beijing Engineering Research Center of High Reliable Embedded System, College of Information Engineering, Capital Normal University, Beijing 100048 China
[2]College of Information Science & Technology, Beijing University of Chemical Technology, China
*Corresponding author, e-mail: begin_dream@126.com, jzhang@mail.buct.edu.cn

Abstract

In this paper we display a practical approach adopted for the formal verification of SpaceWire using model checking to solve state explosion. SpaceWire is a high-speed, full-duplex serial bus standard which is applied in aerospace, so its functions have very high accuracy requirements. In order to prove the design of the SpaceWire was faithfully implements the SpaceWire protocol's specification, we present our experience on the model checking of SpaceWire link interface using the Cadence SMV tool. We applied environment state machine to overcome state explosion and successfully verified a number of relevant properties about transmitter and controller of the SpaceWire in reasonable CPU time.

Keywords*: SpaceWire, formal verification, model checking, environment state machine, CTL, Cadence SMV.*

1. Introduction

With the increasing use of digital systems, design errors will cause serious failures, resulting in the loss of time and money. Especially when the error is discovered late in the design process, large amounts of effort are required to correct the error. So we need some approaches to discover errors and validate designs as early as possible. Conventionally, simulation has been the main debugging technique, but it is incomplete because it cannot involves all possible input values for the complex design. Therefore, there has been a recent surge of interest in formal verification [1]. One very successful formal verification approach is model checking [2], it is an automatic technique for verifying finite-state reactive systems, such as sequential circuit designs and communication protocols. The basic idea of the model checking is that it represents the state of the system transfer structure with finite state machine (FSM), and represents the properties of the system with CTL formula, then traverse FSM to check the correctness of the temporal logic formula. If temporal logic formula is not correct, the system will give an example for user to find the error.

In this paper we verified the SpaceWire link interface [3], the circuit design includes eight modules as Figure1: transmitter, controller, receiver, timer, recovery, credit counter, tx_baudratecounter and error notification. The main function of controller is to control the transformation between the states in the link, this module controls the transmitter to send null, FCT and normal-character (N-Char), also controls the reset of transmitter and receiver reset (the RX-Reset, the TX-Reset). The transmitter is responsible for encoding data and transmitting it using the DS encoding technique. The receiver is responsible for decoding the DS signals (Din and Sin) to produce a sequence of N-Chars (data, EOP, EEP) that are passed on to the host system. The recovery is used to get receive clock (RX_CLOCK) by simply XORing the received Data and Strobe signals together. The timer provides the After 6.4μs and After 12.8μs timeouts used in link initialization. The function of error notification is to deal with a variety of errors in the link. The credit counter is used to control the number of normal-characters (N-Char) received by receiver, avoiding input buffer overflow .The tx_baudratecounter is responsible for controlling the frequency of the signal sent to transmitter module.

Literature [4] already verified all eight modules separately, this paper we verified the compositional model of transmitter and controller, extracted relevant properties and used the Cadence SMV [5-7] tool to verify the interface between two modules, further ensure the correctness of the whole design. SMV is a formal verification system for hardware designs,

Received September 18, 2012; Revised December 28, 2012; Accepted January 13, 2013

based on a technique called symbolic model checking [8]. When to verify the compositional model of transmitter and controller, the size of the program or circuit will increase, and the number of state will increase exponentially and cause the state explosion, this paper will take the appropriate method of modeling and simplification techniques to solve this problem.

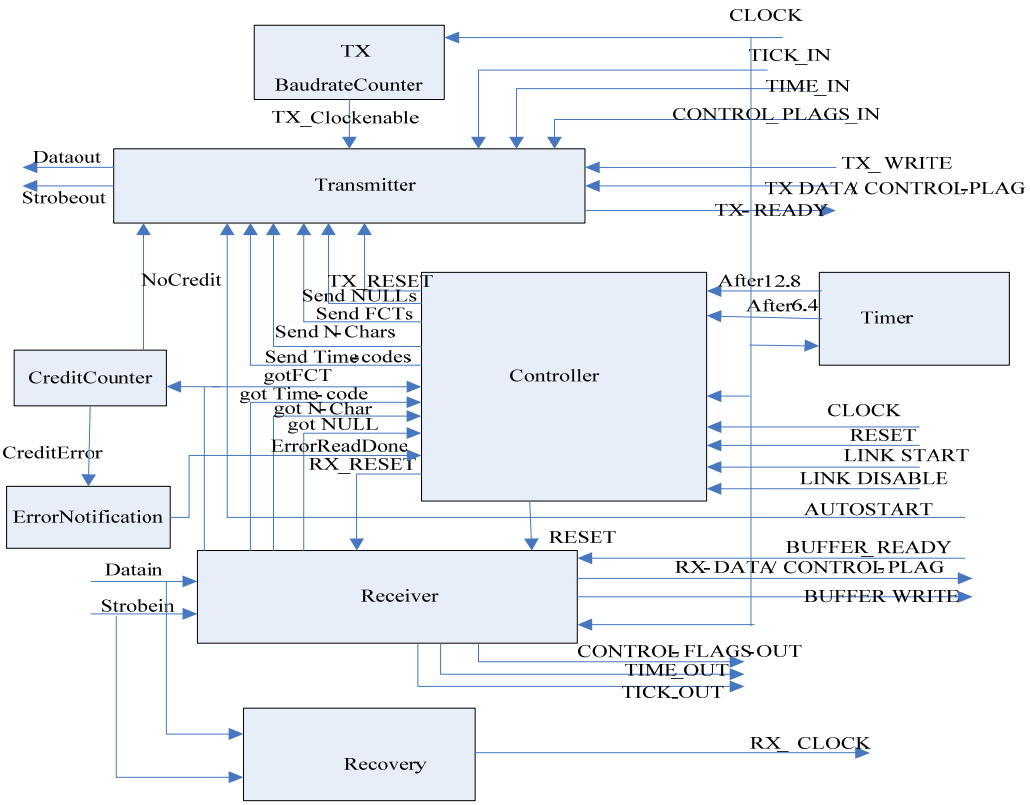

Figure 1. Structure of The Spacewire Link Interface

The rest of the paper is organized as follows. In section 2, we briefly introduce transmitter and controller of spacewire. In section 3, we present our experience on the model checking of transmitter and controller of SpaceWire using the Cadence SMV tool. In section 4, we use environment state machine to solve state explosion. Section 5 concludes the paper.

2. Transmitter and Controller Description

2.1. Transmitter

The transmitter is responsible for encoding data and transmitting it using the DS encoding technique. Figure 2 is the structure of the transmitter, it has six components. It receives 8 bits N-Chars from the host system, turn them into a serial data, then transmit it with StrobeOut. If there is neither a Time-Code, FCT nor an N-Char to transmit, the transmitter sends NULL. The transmitter sends N-Chars only if the host system at the other end of the link has room in its host receive buffer. This is indicated by the other link interface sending an FCT, showing that it is ready to accept another 8 N-Chars. If a link interface receives an FCT, it will transmit 8 N-Chars.

When the Tick_IN signal is asserted the transmitter sends out a Time-Code as soon as the transmitter has finished sending the current character or control code. The value of the Time-Code is the value of the Time_In and Control_Flag_In signals at the point in time when Tick_IN is asserted.

A typical interface between the host system and the transmitter comprises TX_Ready, TX_Write and TX_Data. When the transmitter is ready to receive another N-Char from the host

system, it asserts the TX_Ready signal. When the host system has an N-Char to transmit and the TX_Ready signal is asserted it may put the N-Char onto the TX_Data lines and assert the TX_Write signal. When the transmitter has registered the N-Char data it de-asserts the TX_Ready signal.

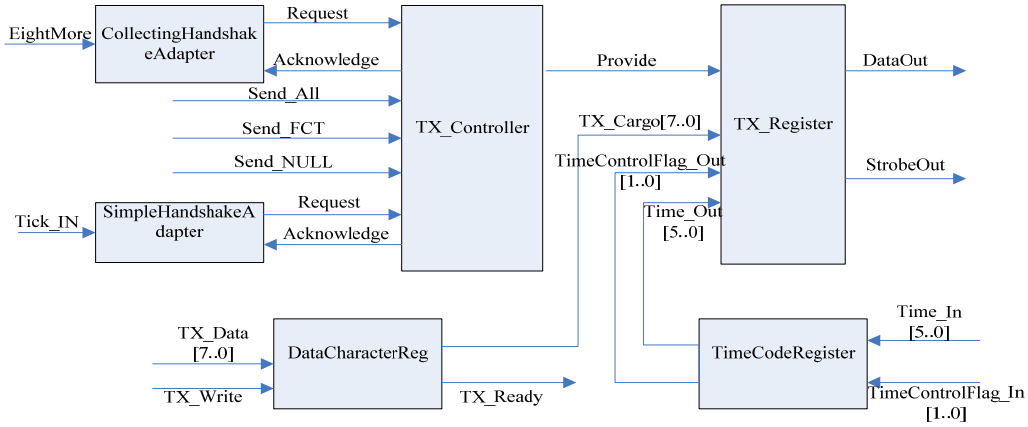

Figure 2. The Transmitter Structure

2.2. Controller

Controller has seven working conditions, it controls the change of the states of the transmitter. Its operation is shown in Figure 3. Controller converts between the seven states depending on different triggers, in different work condition allows transmitter to send different data type, so as to control the transmitter.

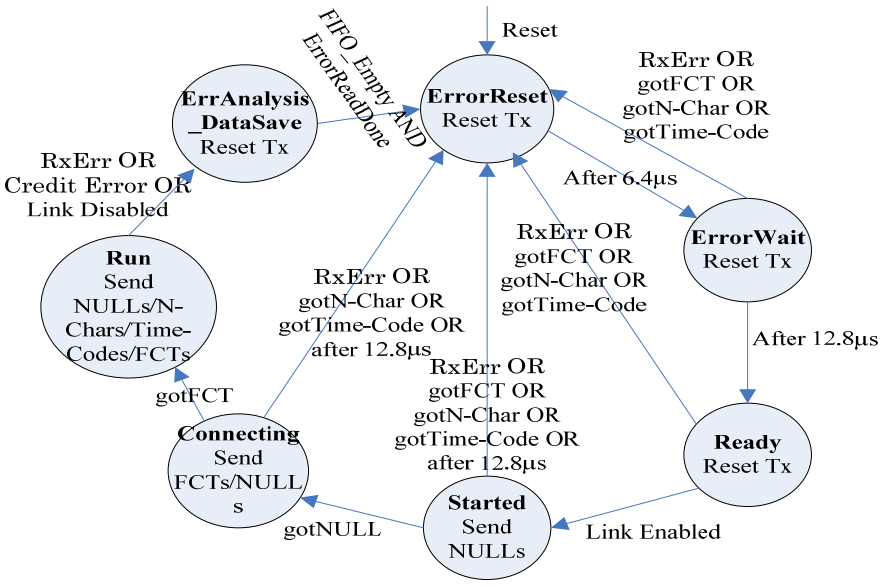

Figure 3. State Diagram for Controller

The ErrorReset state shall be entered after a system reset, after link operation is terminated for any reason or if there is an error during link initialization. In the ErrorReset state the Transmitter shall be reset. When the reset signal is de-asserted, the ErrorReset state shall be left unconditionally after a delay of 6.4µs (nominal) and the state machine shall move to the

ErrorWait state. In the ErrorWait state the transmitter shall be reset. The ErrorWait state shall be left unconditionally after a delay of 12.8μs and the state machine shall move to the Ready state. In the ErrorWait state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move back to the ErrorReset state. In the Ready state the Transmitter shall be reset. The state machine shall wait in the Ready state until the Link Enabled becomes true and then it shall move on into the Started state. If, while in the Ready state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move to the ErrorReset state. In the Started state the transmitter shall be enabled and the transmitter shall send NULLs. The state machine shall move to the Connecting state if the gotNULL condition is set. If, while in the Started state, a disconnection error is detected, or if after the gotNULL condition is set, a parity error or escape error occurs, or any character other than a NULL is received, then the state machine shall move to the ErrorReset state. In the Connecting state the transmitter shall be enabled to send FCTs and NULLs. If an FCT is received the state machine shall move to the Run state. If a disconnect error, parity error or escape error is detected, or if any character other than NULL or FCT is received, or the 12.8μs timeout then the state machine shall move to the ErrorReset state. In the Run state the transmitter is enabled to send Time-Codes, FCTs, N-Chars and NULLs. If the link interface is disabled, or if a disconnect error, parity error, escape error or credit error is detected, while in the Run state, then the state machine shall move to the ErrorReset state. An ErrAnalysis_DataSave state was added in order to improve the error analysis and process ability. When an error occurs or the link is disabled in the run state, FSM enter into ErrAnalysis_DataSave state. In the same time, FSM save and analyze the error and the data. If the data has been saved and the error has been read, then the FSM enter into ErrorReset state, or still in the ErrAnalysis_DataSave state.

3. Model Checking

We abstracted the controller module code and transmitter module code from our whole design, generated a SMV model from Verilog in the Cadence SMV checking tool.

3.1. Properties Description

According to the functions of the transmitter and controller described in Section 2, we give 9 properties. In the following CTL(Computational Tree Logic) expressions, “*”, “->” and“^” mean logical “and”, “imply” and “xor”, respectively.“AG” and “AX” are CTL operators meaning for all paths in all states, and for all paths in the next state, respectively.

Property1: After link error the Data signals shall be set to zero. The CTL expression is the following:

Property1: SPEC AG(DisconnectionError=1 -> AF DataOut=0);

Property2: After link error the Strobe signals shall be set to zero. The CTL expression is the following:

Property2: SPEC AG(DisconnectionError=1 -> AF StrobeOut=0);

Property3:In the ErrorWaite state, the Data signals shall be set to zero. The CTL expression is the following:

Property3:SPEC AG (SpacewireControllerCurrentState = 0 &!Reset & After64 -> AF DataOut=0);

Property4:In the ErrorWaite state, the Strobe signals shall be set to zero. The CTL expression is the following:

Property4: SPEC AG (SpacewireControllerCurrentState = 0 &!Reset & After64 -> AF StrobeOut=0);

Property5:In the Ready state, the Data signals shall be set to zero. The CTL expression is the following:

Property5:SPEC AG (SpacewireControllerCurrentState = 1 &!Reset & After128 -> AF DataOut=0);

Property6:In the Ready state, the Strobe signals shall be set to zero. The CTL expression is the following:

Property6: SPEC AG (SpacewireControllerCurrentState = 1 &!Reset & After128 -> AF StrobeOut=0);

Property7: After Reset the Data signals shall be set to zero. The CTL expression is the following:

Property7:SPEC AG (Reset -> AF DataOut=0);

Property8: After Reset the Strobe signals shall be set to zero. The CTL expression is the following:

Property8: SPEC AG (Reset -> AF StrobeOut=0);

Property9: In the Connecting state, if the transmitter sends a FCT, the Data and Strobe signals meet DS encoding rule. The CTL expression is the following:

Property9:SPEC AG(AX (Provide_FCT)& SpacewireControllerNextState=4 & !Reset -> AF(((AX AX DataOut)^(AX DataOut)) ^ ((AX AX StrobeOut) ^ (AX StrobeOut))));

3.2. Experimental Results

All the properties were checked on a Microsoft Windows XP(2.93GHz/2GB). Table 1 summarizes the experimental results including the correctness of properties, CPU time in seconds, the number of BDD nodes generated and the number of states reached.

Table 1. Experimental Results

Properties	results	CPU time （seconds）	BDD Nodes Allocated	States Reached
property 1	true	273.140625	20204995	6.33826e+029
property 2	true	194.765625	13071242	2.5353e+030
property 3	true	106.037791	9803754	6.33826e+029
property 4	true	263.113372	11295352	2.5353e+030
property 5	true	369.553416	9974210	6.33826e+029
property 6	true	86.000727	11443055	2.5353e+030
property 7	true	415.265625	10377115	3.16913e+029
property 8	true	69.925872	11784565	1.26765e+030
property 9	------	------	------	------

Property 1 to property 8 were verified in reasonable CPU time. But when we verified the property 9, the computer run about three hours and did not give any result, producing state explosion. The scale of the model is exponential both in the number of variables and the number of parallel execution system components, this means: For example, if you add a boolean variable in the program, double the complexity of its property verification [9]. When we verify the compositional model of transmitter and controller, the size of the program will increase, the number of variables will increase and in transmitter the data which is transmitted is 8 bits, so cause the state explosion.

4. Environment State Machine

In order to overcome state space explosion, we adopted compositional reasoning method to cut the whole system into small parts, then use the small parts to verify the properties. But because the components depend on each other, we have to assume the behavior of the other components when we verify the properties of one component. The reasoning process of this method as follow [10]:

$$\frac{<> N < f >}{\frac{< f > M < g >}{<> M||N < g >}}$$

Here, we are asserting that if:

1. N satisfies f
2. if the environment of M satisfies f, then M satisfies g then the composition of M and N will satisfy g.

The advantage of doing the verification in this manner is that we never have to examine the composite state space of M and N. Instead, we check if using just N, and then check g using only M and the assumption g which is an environment of M.

Based on the above compositional reasoning method, we establish environment state machine to express the behavior of the controller, only abstracting away the behavior of the controller which involves the property 9. Because property 9 means in the connecting state, if the transmitter sends a FCT, the Data and Strobe signals meet DS encoding rule. So we ignore the errors in the link and some variables which do not involve the property 9. Then we model the abstracted controller in SMV instead of original controller module code in circuit design.

Figure 4 is the abstracted environment state machine and represents the behavior of the controller. States S0 to S6 correspond with ErrorReset, ErrorWait, Ready, Started, Connecting, Run and ErrAnalysis_DataSave , respectively. TX_Reset, Send_NULL, Send_FCT and Send_All are the interfaces with transmitter. They are valid only in certain states.

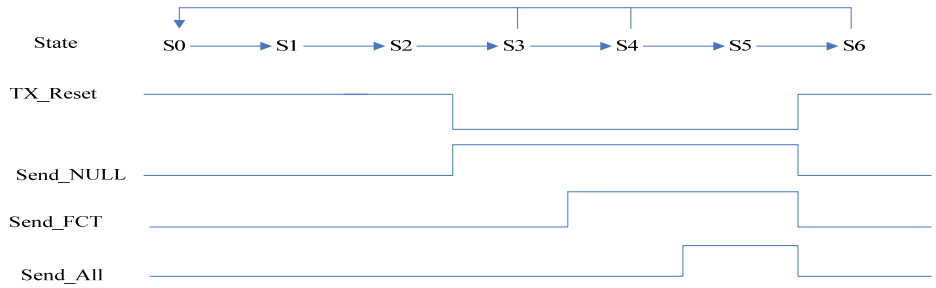

Figure 4. Environment State Machine

The environment state machine is composed with the transmitter, as shown in Figure 5. We use environment state machine to controls the change of the states of the transmitter.

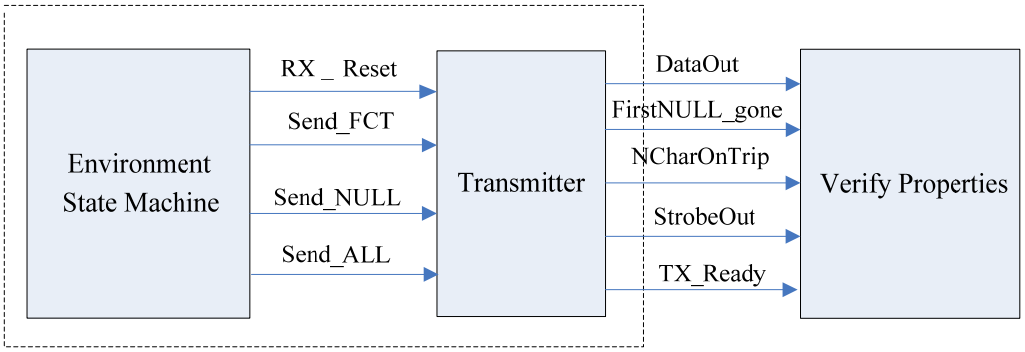

Figure 5. Combined State Machine

Then we verified Property 9:SPEC AG(AX (Provide_FCT)& state=S4 -> AF(((AX AX DataOut)^(AX DataOut)) ^ ((AX AX StrobeOut) ^ (AX StrobeOut)))); The result shows as Table 2. By this method, Property 9 was verified in SMV with a reasonable CPU time.

Table 2. Experimental Results After Establishing Environment State Machine

Property	result	CPU time (seconds)	BDD Nodes Allocated	States Reached
property 9	true	391.847020	60928438	4.95176e+027

5. Conclusion

In this paper, we applied model checking to verify transmitter and controller of SpaceWire. However, we encountered state space explosion problem. This has been solved by establishing environment state machine which we reduced unrelated design details when verifying a property , In this work, model checking of all the properties are done with a reasonable time.

References

[1] C Kern, M Greenstreet, *Formal Verification in Hardware Design: A Survey.* ACM Trans. on Design Automation of Electronic Systems. 1999; 4: 123-193.
[2] EM Clarke, O Grumberg, DA Peled. *Model Checking.* MIT Press. 2000.
[3] ECSS Standard ECSS-E-ST-50-12C. SpaceWire – Links. Nodes, Routers and Networks.
[4] Dai Zhiquan. *Formal verification for the system of harsh environment and high-speed bus based on model checking*. Beijing: Capital Normal University. 2011.
[5] K McMillan. *Getting started with SMV*. SMV Reference Manual. Cadence Berkeley Labs: Berkeley, 1999.
[6] K McMillan. *The SMV language.* SMV Reference Manual. Cadence Berkeley Labs: Berkeley. 1999.
[7] K McMillan. *The Model Checking System.* SMV Reference Manual. Cadence Berkeley Labs: Berkeley. 2002.
[8] K McMillan. *Symbolic Model Checking; Kluwer Academic Publishers*. Boston: Massachusetts. 1993.
[9] Michacl Huth, Mark Ryan. *Logic in Computer Science*. China Machine Press. 2007.
[10] Pnueli A. *In Transition for Global to Modular Temporal Reasoning About Programs In KR Apt ed Logics and Models of Concurrent System*. NAT0 ASI 13. Springer. 1984.