TOOLSET FOR TEST AND VERIFICATION OF IP-BLOCKS WITH SPACEWIRE INTERFACE

Session: SpaceWire Test and Verification

Short Paper

Elena Suvorova

St. Petersburg State University of Aerospace Instrumentation

67, Bolshaya Morskaya st. 190 000, St. Petersburg RUSSIA

E-mail: suvorova@aanet.ru,

ABSTRACT

Toolset for test and verification of RTL, post-synthesis and post-implementation models is usually based on BFM. The BFM (Base Formal Model) typically is used for testing and verification of interfaces that correspond to different standards. (Other term BFM – Base Functional Model is typically system level model of a particular device or IP-block). A toolset obligatory includes also generators of test sequences (or a prepared set of test sequences) and modules for monitoring and processing of test results. Often the toolset also includes an expanded set of components for simulation and performance estimation of the device in context of a real system. This expanded set could include switches, memory blocks and other devices. We suggest to use this approach for SpaceWire toolset development. We specify the SpaceWire BFM as a multilayer structure, every layer of which corresponds to a layer of the SpaceWire standard. It allows to use the BFM for test and verification of SpaceWire controllers with support of different SpaceWire layers and at different stages of design. For example, if we plan to test IP-block, that includes layers from character to packet we will use only corresponding layers of a BFM in test shell. In many cases the detailed test of physical level is very important problem for SpaceWire product designers. BFM includes physical level also.

The important task of coordination between test tools and tested device or IP-block settings is considered.

1 BFM STRUCTURE

Suggested BFM of SpaceWire is multilevel structure. Levels of BFM correspond to layers of SpaceWire standard. All levels of BFM exclude signal and physical are described on SystemC. These two down layers could be described on SystemC, VHDL or Verilog (dependently on design tools used for simulation, and on Unit Under Test (UUT) description language)

Let's consider the BFM structure. Figure 1 illustrated information flows between levels of model.

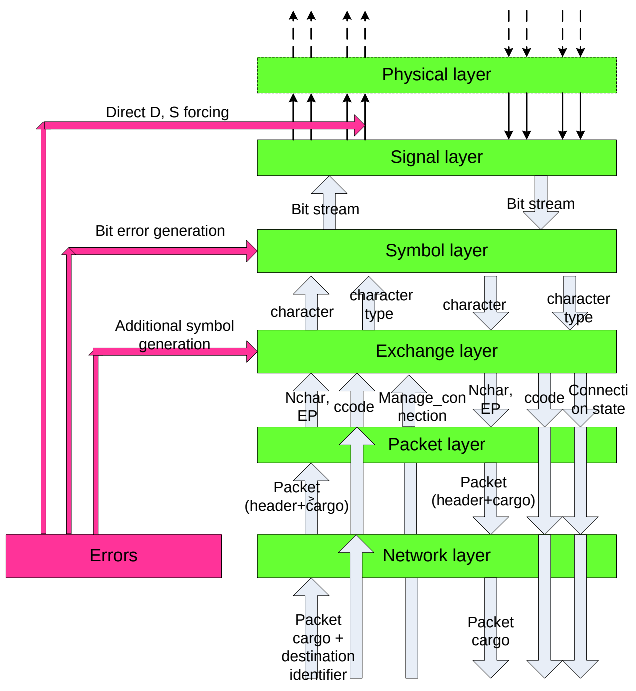

Figure 1. – The information flows in BFM model

Main information flows defined in SpaceWire standard are shown by grey arrows. One function of BFM is verification of UUT behavior when external errors appear. Therefore BFM include error situation generation possibilities. Suggested BFM allow generation errors of exchange, symbol and signal level.

The exchange level of BFM includes features for data flow control errors generation. It could send to channel Nchar symbols that are not credit. The BFM allow sending to channel FCT symbols that credit more than 56 Nchars. Also BFM allow sending to channel different symbols independently from current state of state machine. For example it allows to send Nchar symbols in states ErrorWait, Ready, Started, Connecting.

Possibility of modifying (inversion) of any bits of symbols exists on symbol level BFM (after generation of true bit representation for the symbol).

On signal level BFM allow change values of D and S lines in any time moments. As result we can simulate situations when signals D and S changed together or when time interval between D and S change is very short.

These possibilities allow testing system behavior in case of different types of external faults and noise. On figure 1 the control flows for error generation are marked by red arrows.

Information exchange between neighbor levels of BFM is going via access points that are logical ports. Logical port is a class that includes set of methods for information exchange. For example let's consider interconnection between packet level and exchange level. This interconnection includes three access points for transmission to exchange level: for data and end packet symbols, for control symbols (time, interrupt and acknowledge codes), for interconnection management. The access point (logical port) for control codes transmission includes next methods:

```
int send_Ccode(t_code Ccode_);
bool ready_to_send_Ccode();
t_code receive_Ccode();
bool received_Ccode();
```

The test generator and results controller could be connected to different levels of BFM dependently on set of SpaceWire layers includes in UUT. Also different levels of BFM could be connected to corresponding levels of UUT. The special wrappers are used between BFM and UUT interface because the interface of UUT is specified in terms of signals (class sc_signal).

2 THE STRUCTURE OF SUGGESTED TOOLSET

In this article we suggest method and toolset for test and verification of IP-blocks and devices that includes SpaceWire controllers some levels of protocol SpaceWire. One of important tasks for test development is control of UUT regime. For example we need test UUT in link start and auto start regimes, change transmission rate. If UUT includes network layer, then we need manage routing table. Correspondingly suggested test tools include not only BFM, test generators, test controllers and wrappers, but also special components for UUT configuration. Configuration process is typically specific for every concrete type of devices or IP-blocks. Interfaces for IP-block configuration could be differ for different blocks. Also the time between writing of new setting and real change could be varied essentially. We divide all UUT to two groups. First group includes UUT that are fully configurable via external interface. For example into this group included coder-decoder SpaceWire IP-block and SpaceWire controller IP-block for systems with internal processor.

For such UUT configuration our toolset include special component that allows transfer of configuration information via typical parallel synchronous memory interface. This component also includes functions for translation to/from signal based interface. Then user could describe wrapper from this interface to interface of his IP-block on any hardware description language. The example of test shell for such IP-blocks based on suggested toolset is represented on figure 2. On this figure components from suggested toolset are green.

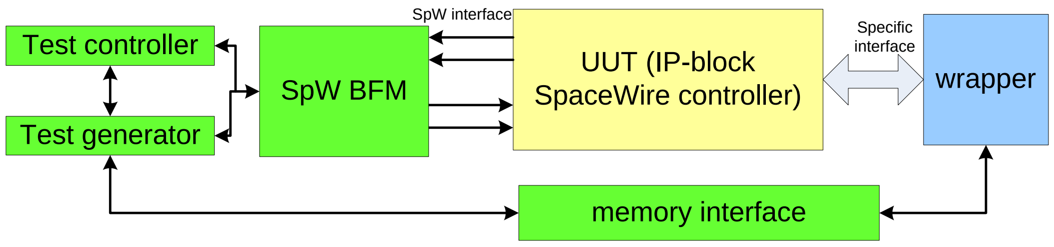

Figure 2. – The example of test shell structure for IP-blocks

Additionally toolset includes two special interface components. First of them intends for connecting to coder/decoder SpaceWire with typical interface represented on figure 19 of SpaceWire standard [1]. Second interface component includes two sub interfaces FIFO for data packets sending and receiving. The data line width is parameterized (number of bit lines s 2^n). Data packets are aligned to words boundaries. Interface includes byte valid lines. This special interface also includes sub interface parallel synchronous memory interface for working with state and regime registers. The control codes also could be sending and receiving via this interface.

Second group includes devices (UUT) with internal processor or automata. For these devices coordination between internal settings and test program in test shell typically is very difficult.

These devices could be divided to two subgroups. First subgroup includes devices configured only via SpaceWire interface. Second subgroup includes devices that could be controlled via other interface, for example interface with external memory.

In this article we consider possible decision of configuration problem for UUT with internal processor. The test program could be loaded to such UUT. But in this case appear the problem of synchronization between test program in UUT and test program in test shell. Therefore we suggest other approach. In frame of this approach the control of testing process is made by test program placed in test shell. This program when need could observe state of UUT and set regime of UUT with using of special commands. These commands are written one by one to fixed addresses of external memory model that accessible by internal processor of UUT. The simple program (written on C) that read and executes these commands is loaded to UUT. The interface component of test shell implement monitoring of reading commands by UUT and define time moments when next command could be written to memory. The program loaded to UUT is universal, for porting it to other device we need only correct base address of external memory. This scheme provides small time interval between start settings and activation new regime. This feature remove problem of synchronization between test program in test shell and test program in UUT.

Usually for connection of external memory is used parallel synchronous or asynchronous memory interface or sequential interfaces such as I2C, SPI or USB. Correspondingly suggested toolset includes components for interconnection external memory via these interfaces.

The example of test shell based on suggested toolset for device with internal processor is represented on figure 3.

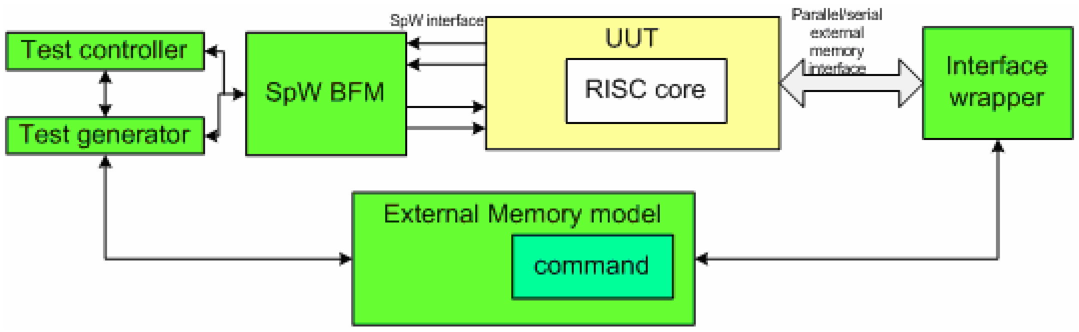

Figure 3. – The example of test shell for device with internal processor.

Thus suggested toolset includes next components: SpaceWire BFM, test generators and test controllers and wrappers for all levels of BFM, model of external memory and interface components for control of UUT regimes.

3 Conclusion

Suggested toolset could be user for test and verification of SpaceWire controllers IP-blocks that support different levels of protocol, of devices includes support some levels of protocol, for example, SpaceWire switches, special processors includes SpaceWire controllers.

4 References

1. ECSS E50 12A, “SpaceWire. Links, nodes, routers and networks”, 24 January 2003