FORMAL VERIFICATION OF A VHDL HARDWARE COMPONENT

Petr HLÁVKA, Master Degree Programme (5)
Dept. of Intelligent Systems, FIT, BUT
E-mail: xhlavk00@stud.fit.vutbr.cz

Supervised by: Dr. Tomáš Vojnar

ABSTRACT

The work presented here is a part of the efforts to apply formal verification in the process of developing an FPGA-based network device in the Liberouter project. We briefly characterise the verification framework introduced in the project and then describe in more detail the verification we propose for one interesting component – namely an asynchronous FIFO queue.

1 INTRODUCTION

The rising complexity of computer based systems together with the increasing stress on their correctness (linked to more and more critical tasks solved by the systems) lead to a search for better methods of ensuring error-freeness. Along with traditional methods like testing and simulation, formal verification can be used for checking system properties in every possible run. The most widely used automatic verification method is model checking [1], which is based on a complete state space search for reachable states violating desired properties often specified using temporal logics, e.g., LTL or CTL.

2 THE VERIFICATION PROCESS IN THE LIBEROUTER PROJECT

The aim of the Liberouter project [2] is to develop a multigigabit PC-based IP protocol router. In order to achieve a higher throughput, hardware accelerator cards with field programmable gate arrays (FPGA) are used. The most of the network functions (e.g., packet manipulation, forwarding or monitoring) are implemented using the VHDL hardware specification language directly in the hardware accelerators. Proving correctness of this hardware design is the task of the Liberouter formal verification group.

One of the verification approaches used in the Liberouter project is model checking on the VHDL code. The others are based on abstract models of some key functionalities of the considered devices which is, however, beyond the scope of this paper. There is no available model checking tool capable of reading model specification from the VHDL

subset used in the project. That is why the verification group proposed a translation [3] including synthesis into the Verilog hardware description language that can be used in connection with the Cadence SMV model checker. We developed also several tools supporting the whole verification process, which includes the conversion, specification of properties, iterative verification and generating verification reports.

3 THE ASFIFO_DIST COMPONENT

In this paper, we present the verification of one particular component – namely the asfifo_dist (Figure 1). This component is an implementation of an asynchronous FIFO queue using the basic cells (look-up tables) of the FPGA as the memory for queue items. The number of items and their width can be adjusted as generic parameters of the queue.

An interesting feature of the queue is that it provides, along with the standard `EMPTY` and `FULL` queue status signals, also 2-bit width `STATUS` information about the remaining FIFO capacity. The design is complexified due to presence of two asynchronous clock signals and thus the formal verification is really justified here. Depending on the read and write clock's shift, the delay of the queue status signals may differ within specified bounds [4], e.g., the remaining FIFO capacity signal has delay up to two write clocks.

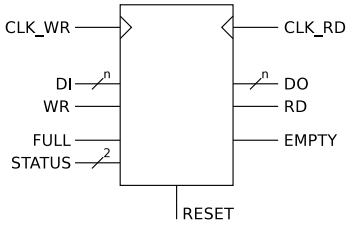

Figure 1: Asfifo_dist block scheme.

4 VERIFICATION OF ASFIFO_DIST

Due the ambiguous delay of the queue status signals, the temporal logic formulae describing the proper signal behaviour would be hard to construct. Instead of this, we have decided to create an observer process using the SMV language that models the desired minimal delay component behaviour. The observer is counting the number of items in the FIFO (variable `counter`) and stores its previous value for two writing clocks (variable `our_status`). Using these two process variables, temporal logic formulae comparing the behaviour of the observer and the component can be easily constructed.

The verified properties were chosen according to the particular signal's specification, and also general properties like the overflow/underflow in the number of items or the possibility to fill the FIFO up to the maximum number of items were added. Because of the state space complexity, only the control (not data) signals were verified. The complete list of the verified properties can be found in the verification report [5]. Here, we list examples of some of the verified properties including CTL or LTL formulae:

- It is not possible to read from an empty FIFO (underflow):
  ```
  G ( ~( counter <= -1 ) )
  ```
- It is possible to store the specified maximum number of items in the FIFO:
  ```
  AG EF ( counter = ITEMS - 1 )
  ```
- The specified correct behaviour of the `EMPTY` signal (i.e., it is set/unset properly) is divided into the set up, hold on, unset and hold off phases:

```
- G ( ( counter = ITEMS - 1 & ~full & ~writer_clk & X writer_clk ) -> X full )
- G ( full -> ( G ( counter = ITEMS - 1 ) & full ) |
              ( ( ( counter = ITEMS - 1 ) & full ) U ( counter ~= ITEMS - 1 ) ) )
- G ( ( ( counter ~= ITEMS - 1 ) & full & ~writer_clk & X writer_clk ) ->
            ( ( X X ~full ) | ( X ( writer_clk U
                                  ( ~writer_clk U ( writer_clk U ~full ) ) ) ) ) )
- G ( ~full -> ( G ( counter < ITEMS - 1 ) & ~full ) |
               ( ( ( counter < ITEMS - 1 ) & ~full ) U
               ( ( counter = ITEMS - 1 ) | gsr ) ) )
```

4.1 VERIFICATION RESULTS

As the number of items of the FIFO and its width can be parametrised, the verification should be run for every used combination in the project. We consider only the queue lengths (number of stored items) of power of two because the parameter specifies the bit width. We also omit the item width parameter because only the control (not data) signals are verified. The verification run durations for selected properties and queue lengths are shown in Table 1. Due to the state explosion problem, it is not possible to verify some of the `STATUS` signal properties for more than 32 items and some of the other signals properties for more than 64 items.

Property	$t_{ITEMS=8}$	$t_{ITEMS=16}$	$t_{ITEMS=32}$	$t_{ITEMS=64}$
FIFO underflow	0.49	3.88	26.65	143.64
Reach full	1.03	9.15	54.98	301.93
Empty set up	0.69	4.32	29.34	167.78
Empty hold on	3.18	14.74	53.67	242.33
Empty unset	1.84	10.71	66.55	282.07
Empty hold off	10.33	62.80	123.60	562.91
Status	6.53	81.38	2133.35	n/a

Table 1: Verification time [s] of selected properties - Intel Xeon 3.2 GHz, 2 GB RAM.

5 CONCLUSION

The work shows that the chosen verification process on the VHDL code is applicable even to asynchronous components, but it meets its boundary in verification of the complex `STATUS` signal properties. The component works according to the specification, all proposed properties were successfully verified. Due to the two clock delay, the `STATUS` signal should only be used in queues with more than 64 items where its value is more precise.

REFERENCES

[1] Clarke, E. M., Grumberg, O., Peled, D. A.: Model Checking, Cambridge, Massachusetts, The MIT Press 1999, ISBN 0-262-03270-8

[2] Liberouter: The Liberouter Project WWW pages, http://www.liberouter.org

[3] Kratochvíla, T., Řehák, V., Šimeček, P.: Verification of COMBO6 VHDL Design, Technical report number 17/2003, CESNET 2003

[4] Xilinx: FIFOs Using Virtex-II Block RAM, Xilinx Application Note XAPP258 (v1.4), Xilinx 2005, http://direct.xilinx.com/bvdocs/appnotes/xapp258.pdf

[5] Hlávka, P.: Asfifo_dist Verification Report (2005-11-17), http://www.liberouter.org/formal_verification/work/asfifo_dist_ver.html