**Why? What problem are the author(s) trying to solve?**

While there are many formal methods for verifying VHDL, there is no widely used VHDL verification solution. SO The practice presented in this paper is about transforming VHDL into Verilog models, and then verifying the Verilog by using model-checking tools, such as CadenceSMV and NuSMV to detect FPGA VHD software deadlocks.

**How? What method has been used to answer the question?**

We present a semi-automatic way to verify FPGA VHDL software deadlocks, especially those that reside in automata.

This domain is defined to represent the VHDL modules that will be verified; these modules will be transformed into Verilog models and be verified by SMV tools. By analyzing the verification results of SMV, deadlocks can be found; afterS looking back to the VHDL code, the deadlocking code is located and the problem is solved.

Bounded modelchecking tools, such as SAT, are first used to perform time-saving verification and then the verification result is analyzed to find deadlocks. . After finding these deadlocks, corresponding code lines are located in the VHDL sources. By debugging these source lines, the deadlock problem is solved. Finding deadlocks in verification results is currently the only manual procedure in this approach.

**What is the result from the method?**

The results are analyzed to find loops and loops found in the verification results are usually ones that cause deadlocks. So Analyzing the results is simple, the loops can be found in the output files of SMV.

After finding loops, we go back to the VHDL code to find deadlocks. The deadlocks reside in the automata A corresponding to the verification model V.

The verification solution works. As shown in the result, we found several deadlocks in the MVBC project. Some of them are confirmed to be impossible in the actual runtime environment, but the cmf ssc combine module is found to be the root cause of the issue.

**What is the validity of the result according to the author?**

Though confirmed as “impossible”, the deadlocks are not what the hardware team thinks they are. The deadlocks may not happen if all the hardware is in good condition, but they may occur in the future if some part of the circuit is not working, e.g., some pin is not soldered well or worn out. So it is still required that all the deadlocks be removed.

It is the hardware team’s responsibility to make sure the circuit works perfectly, but the software team is also obliged to prevent potential problems. Only in this way will the verification solution be meaningful. At the same time, verification is currently limited to locating deadlocks in VHDL automata.

**Does the research solve the problem/answer the question?**

**YES.** The safety properties were tested well in the development stage, but experienced a breakdown during the long-term software testing stage, which was mainly caused by deadlocks in the VHDL software. In this special case, we managed to locate the VHDL deadlocks and solve the problem by the FPGA deadlock detection approach provided in this paper, which demonstrates that our solution works well.

**What are to be done next? future work? and why?**

However, many other parts of the code can be troublesome, so we plan to extend our verification to those parts in the future. We currently use SAT and bounded-model checking to verify these automata, so we have to specify the depth of the SAT solver manually, according to our individual experience. This is not reliable and may vary from person to person. As it is more time consuming to design the overall verification strategy and depth dependency and to solve this problem, we also plan to use Binary Decision Diagram (BDD), in addition, to verify the existing automata and we believe that this will make it more convincing and convenient.

**And lastly something from you. What do you think and why?**

I think this practical approach is really straight forward since you will most likely find all the deadlocks in the corresponding code lines located in the VHDL sources and then all you need to do is unbuugin them. So yeah this all from me I hope you leaned something new and cya.