Supplemental Material for the Paper: "Insights from Bugs in FPGA High-Level Synthesis Tools: An Empirical Study of Bambu Bugs"

Zun Wang, He Jiang, *Member, IEEE,* Xiaochen Li, Shikai Guo, Xu Zhao, and Yi Zhang

***Abstract*—High-Level Synthesis (HLS) tools have been widely used in FPGA design to convert C/C++ code to Hardware Description Language (HDL) code. Unfortunately, HLS tools are susceptible to bugs which can introduce serious vulnerabilities in FPGA products, leading to substantial losses. However, the characteristics of these bugs (e.g., root causes and bug-prone stages) have never been systematically studied, which significantly hinders developers from effectively handling HLS tool bugs. To this end, we conduct the first empirical study to uncover HLS tool bug characteristics. We collect 349 bugs of a widely used HLS tool, namely Bambu. We study the root causes, buggy stages, and bug fixes of these bugs by applying a multi-person collaboration method. Finally, 13 valuable findings are summarized. We find 14 categories of root causes in Bambu bugs; most bugs (22.1%) are caused by incorrect implementation of IR processing; the front end of Bambu is more bug-prone; to fix these bugs, 2.27 files and 80.19 lines of code need to be modified on average. We also present the insights gained from 95 Vitis HLS bugs. From these findings, we suggest that developers could use an on-the-fly code generator configuration method to generate suitable testing programs for HLS tool bug detection and apply Large Language Models to assist in fixing HLS tool bugs.**

***Index Terms*—FPGA, high-level synthesis tool, software bugs, bug characteristics, empirical study.**

I. Comparison with similar studies done on software compilers

In the supplemental material, we have made a comparison with similar studies done on software compilers [1], [2], [3], [4], [5]. More specifically, Sun et al. [1] conducted the first systematic study to analyze GCC and LLVM bugs. However, they did not analyze the root causes, bug-prone stages, and bug symptom of compiler bugs. Zhou et al. [2] conducted the first study to characterize the optimization bugs of GCC and LLVM. Wang et al. [3] and Romano et al. [4] focused on analyzing the bug in the WebAssembly framework (i.e., WebAssembly compilers and runtime). Shen et al. [5] performed an exhaustive analysis of bugs in deep learning compilers. The details of the comparison are shown in the following subsections. Based on the comparison, we have shown that our study does not have a similar taxonomy and bug distribution compared to these studies.

He Jiang is with the School of Software, Dalian University of Technology, Dalian 116024, China, also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116024, China, and also with Dalian Key Laboratory of Artificial Intelligence, Dalian 116024, China, and also with DUT Artificial Intelligence Institute, Dalian 116024, China (corresponding e-mail: jianghe@dlut.edu.cn).

Zun Wang and Xiaochen Li are with the School of Software, Dalian University of Technology, Dalian 116024, China, and also with the Key Laboratory for Ubiquitous Network and Service Software of Liaoning Province, Dalian 116024, China (e-mail: wangzun_ssdut2020@mail.dlut.edu.cn; xiaochen.li@dlut.edu.cn;).

Shikai Guo is with the School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China (e-mail: shikai.guo@dlmu.edu.cn).

Xu Zhao and Yi Zhang are with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (e-mail: zhaoxu@bit.edu.cn; 3220195123@bit.edu.cn).

A. Root cause comparison

The root causes and their distribution of each study are shown in Table I. We have highlighted the top 3 root causes reported in each study. We exclude the studies from Zhou et al. [2], since they did not analyze the root causes in their studies. Romano et al. [4] found 8 root causes of WebAssembly compiler bugs, such as asyncify synchronous code, incompatible data types, memory model difference, and other infrastructures. The top 3 root causes are incompatible data types, other infrastructures, and emulating native environment. Wang et al. [3] concluded 16 root causes of WebAssembly runtime bugs (e.g., incorrect algorithm implementation, API misuse, and incorrect assignment). They found that incorrect algorithm implementation, memory, and incorrect exception handling are the top 3 root causes. Shen et al. [5] found 12 root causes of Deep Learning compiler bugs (e.g., type problem, API misuse). They stated that type problem is the most common root cause. Compared to these studies, we have 14 categories of root causes. There may be some overlap between our study and other studies, such as type problems and configuration problems. However, most root causes we found do not appear in previous studies (e.g., *Incorrect Implementation of Binding Processing*). For the distribution of root causes, we found that *Incorrect Implementation of IR Processing*, *Configuration File and Script Problem*, and *Incorrect Implementation of Code and File Generation* are the top 3 root causes of HLS tool bugs. Thus, the distribution of root causes in our study is also different from those found in other studies.

B. Bug-prone stages comparison

The bug-prone stages and their distribution of each study are shown in Table II. We exclude the studies from Wang et al. [3] and Zhou et al. [2], since they did not analyze the bug-prone stages in their studies. For Romano et al. [45], they found build time, program compile time, and runtime bugs in WebAssembly compilers. The program compile time and

TABLE I
ROOT CAUSES COMPARISON

Compilers	Root causes	Notes
WebAssembly runtimes [3]	1. ***Incorrect Algorithm Implementation (25.49%)***. 2. ***Memory (17.07%)***. 3. ***Incorrect Exception Handling (10.96%)***. 4. API Misuse. 5. Type Problem. 6. Incorrect Condition Logic. 7. Incorrect Assignment. 8. Incorrect Configuration. 9. Missing Condition Checks. 10. Concurrency. 11. Environment Incompatibility. 12. External API Incompatibility. 13. Dependent Module Issue. 14. Logical Order Error. 15. Incorrect Numerical Computation. 16. Others.	
WebAssembly compilers [4]	1. Asyncify Synchronous Code. 2. ***Incompatible Data Types (23).*** 3. Memory Model Difference. 4. ***Other Infrastructures (25)***. 5. ***Emulating Native Environment (23)***. 6. Supporting Web APIs. 7. Cross-Language Optimizations. 8. Runtime Implementation Discrepancy.	The author did not present the top 3 root causes in the paper. The number behind the root cause represents the number of bugs related to that specific root cause. According to the number of bugs, we have highlighted the top 3 root causes.
Deep Learning compilers [5]	1. API Misuse. 2. Incompatibility. 3. ***Type Problem (19.23%)***. 4. ***Tensor Shape Problem (13.27%)***. 5. Incorrect Numerical Computation. 6. Incorrect Assignment. 7. Incorrect Exception Handling. 8. Misconfiguration. 9. Concurrency. 10. ***Incorrect Code Logic (16.92%)***. 11. Typo. 12. Others.	
HLS tools (our study)	1. Incorrect Compiler Usage. 2. Incorrect Implementation of Compiler Plug-ins. 3. Incorrect Implementation of Tool Parameter Processing. 4. Type Problem. 5. ***Configuration File and Script Problem (14.6%)***. 6. Incorrect Implementation of Specific Structure Data Processing. 7. ***Incorrect Implementation of IR Processing (22.1%)***. 8. Incorrect Implementation of Allocation Processing. 9. Incorrect Implementation of Scheduling Processing. 10. Incorrect Implementation of Binding Processing. 11. Incorrect Implementation of Architecture and Interface Processing. 12. ***Incorrect Implementation of Code and File Generation (13.2%)***. 13. Incorrect Implementation of Tool Integration. 14. Other Problems.	

runtime are relatively more buggy than the build time. For Shen et al. [5], they found three bug-prone stages in deep learning compilers, including high-level IR transformation stage, low-level IR transformation stage, and model loading stage. They presented that the high-level IR transformation stage (44.92%) and the low-level IR transformation stage (33.64%) are relatively more buggy than the model loading stage (21.44%) on average. In our study, we found 5 bug-prone stages, such as build, configuration, front end, HLS process, and back end. We found the front end, HLS process, and back end are more bug-intensive stages in HLS tools. We think our analysis results are different from those reported in other studies due to differences in architecture between HLS tools and other compilers. For example, there is no HLS process in deep learning compilers.

C. Bug symptoms comparison

The bug symptoms and their distribution of each study are shown in Table III. Zhou et al. [2] found 3 bug types of compiler optimization bugs, including crash, misoptimization, and performance. They stated that misoptimization is the most common bug symptom in both GCC and LLVM, accounting for 57.21% and 61.38%, respectively. Romano et al. [4] found 11 types of bug symptoms and compile error is the most common bug symptom. Wang et al. [3] et al. summarized 6 symptoms (i.e., crash, incorrect functionality, build error, bad performance, hang, and unreported). They found that crash is the most prevalent bug symptom. Shen et al. [5] found 6

TABLE II
BUG-PRONE STAGE COMPARISON

Compilers	Root-prone stages	Notes
WebAssembly compilers [4]	1. ***Program runtime (299)***. 2. ***Program compile time (301)***. 3. Build time (95).	The author did not present the top 3 root causes in the paper. The number behind the root cause represents the number of bugs related to that specific root cause. According to the number of bugs, we have highlighted the top 3 root causes.
Deep Learning compilers [5]	1. ***High-level IR transformation stage (44.92%)***. 2. ***Low-level IR transformation stage (33.64%)***. 3. Model loading stage (21.44%).	The author only presented these 3 stages in the paper.
HLS tools (our study)	1. Build. 2. Configuration. 3. ***Front end (39.3%)***. 4. ***HLS process (19.8%)***. 5. ***Back end (20.3%)***.	

TABLE III
BUG-PRONE STAGE COMPARISON

Compilers	Root-prone stages	Notes
Optimization bugs in GCC and LLVM [2]	1. ***Crash (39.33% and 34.08%)***. 2. ***Misoptimization (57.21% and 61.38%)***. 3. Performance.	The author only presented these 3 bug symptoms (types) in the paper.
WebAssembly runtimes [3]	1. ***Crash (56.86%)***. 2. ***Incorrect Functionality (22.15%)***. 3. ***Build Error (12.92%)***. 4. Bad Performance. 5. Hang. 6. Unreported.	
WebAssembly compilers [4]	1. ***Build Error (95)***. 2. ***Compile Error (264)***. 3. Linker Error. 4. Code Bloating. 5. Crash. 6. Data Corruption. 7. Fail to Instantiate. 8. Performance Drop. 9. Hang. 10. ***Other Runtime Error (117)***. 11. Incorrect Functionality.	The author did not present the top 3 bug symptoms (types) in the paper. The number behind the bug symptom (type) represents the number of bugs related to that specific bug symptoms (types). According to the number of bugs, we have highlighted the top 3 bug symptoms (types).
Deep Learning compilers [5]	1. ***Crash (59.37%)***. 2. ***Wrong Code (25.04%)***. 3. Bad Performance. 4. Hang. 5. ***Build Failure (8.29%)***. 6. Unreported.	
HLS tools (our study)	1. ***Incorrect Results (38.9%)***. 2. ***Tool Failure (33.7%)***. 3. ***Incorrect GUI (9.5%)***. 4. Crash. 5. Hang. 6. Unreported.	

symptoms (i.e., crash, wrong code, bad performance, hang, build failure, and unreported) and crash is the most common bug symptom. In our study, we also found 6 types of bug symptoms, including incorrect results, tool failure, incorrect GUI, crash, hang, and unreported. Crash, hang, and unreported are common bug symptoms reported in many studies, as well in ours, but we also found tool failure, that is rarely mentioned in other studies. Meanwhile, we found that the symptom 'incorrect results' is the most common bug symptom in HLS tools, which is different from other studies where crash is the most common bug symptom.

REFERENCES

[1] C. Sun, V. Le, Q. Zhang, and Z. Su, "Toward understanding compiler bugs in gcc and llvm," in *Proc. 25th Int. Symp. Softw. Test. Anal.*, 2016, pp. 294–305.

[2] Z. Zhou, Z. Ren, G. Gao, and H. Jiang, "An empirical study of optimization bugs in gcc and llvm," *J. Syst. Softw.*, vol. 174, pp. 110 884–110 884, 2021.

[3] Y. Wang, Z. Zhou, Z. Ren, D. Liu, and H. Jiang, "A comprehensive study of webassembly runtime bugs," in *Proc. IEEE Int. Conf. Softw. Anal., Evol. Reengineering*. IEEE, 2023, pp. 355–366.

[4] A. Romano, X. Liu, Y. Kwon, and W. Wang, "An empirical study of bugs in webassembly compilers," in *Proc. 36th IEEE/ACM Int. Conf. Autom. Softw. Eng.* IEEE, 2021, pp. 42–54.

[5] Q. Shen, H. Ma, J. Chen, Y. Tian, S.-C. Cheung, and X. Chen, "A comprehensive study of deep learning compiler bugs," in *Proc. 29th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2021, pp. 968–980.