Bug Injection Report

Team 3 – Anurag Ranga, Yashaswi Katne
Naveen Reddy T, Maheshwar Reddy K

Bug Injection Strategy

The provided code defines a synchronous FIFO (First-In-First-Out) memory with bug injection points. The bugs are encapsulated using conditional compilation directives `ifdef` and `else`. We can selectively enable/disable bug injection during simulation or synthesis by defining `INJECT_THE_BUG` before compiling the code.

Types of Bugs Injected

Two bugs have been introduced in the code:

1. **Data Corruption Bug:** This bug resides in the `fifo_mem` module. When `INJECT_THE_BUG` is defined, the code corrupts the data being read from the FIFO by XORing it with 0xFF. This will cause the retrieved data to be completely different from the data that was written.
2. **Logic Bug:** This bug resides in the `synchronizer` module. When `INJECT_THE_BUG` is defined, the code assigns zero to the `q1` register instead of the data input `d_in`. This effectively freezes the value of `q1` to zero, and since `d_out` is assigned the value of `q1` on every clock cycle, it will always be zero regardless of the actual input. This bug can lead to various issues depending on how the synchronized data is used in the design.

Impact of Bugs

The data corruption bug will cause the FIFO to output incorrect data. This can lead to unexpected behavior or crashes in the overall system that utilizes the FIFO.

The logic bug in the synchronizer module can have a variety of effects depending on how the synchronized data is used. In the case of the provided code, the bug affects the calculation of the empty flag in the `rptr_handler` module. This can lead to the FIFO appearing to be empty even when it still contains data. This could cause the system to attempt to read from an empty FIFO, resulting in undefined behavior.

Utilizing Bug Injection for Verification

By injecting bugs into the design, we can verify that the design can tolerate these faults and behave as expected. This helps to improve the robustness of the design against unexpected errors that might occur during real-world operation. For instance, the data corruption bug can be used to verify that the data processing unit downstream from the FIFO can handle corrupted data gracefully. Similarly, the logic bug can be used to verify that the system can recover from an unexpected empty flag condition.

Conclusion

Bug injection is a valuable technique for design verification. By strategically injecting bugs into the code, we can identify potential weaknesses in the design and ensure that it is robust against unexpected errors. The provided code demonstrates how to use conditional compilation directives to easily enable and disable bug injection for verification purposes.