Roshni Ramesh
18-341 End of Semester Project

LFSR-Based Arbiter Verification

1. INTRODUCTION

Arbiters that use Linear Feedback Shift Registers (LFSRs) rely on pseudo-random priority, but this randomness makes verification difficult. An 8-bit LFSR has 255 distinct seeds, and each seed generates a different sequence of priorities. Verifying a bounded liveness property such as "every request must be granted within k cycles" requires checking all seeds, which quickly becomes infeasible using simulation alone.

To explore scalable verification, I implemented two techniques inspired by Auerbach et al. [1]:

1. **Property strengthening with underapproximation**: a simulation-based bug-finding method that identifies “bad seeds” extremely quickly.
2. **Bounded Model Checking (BMC)**: a formal method that symbolically examines all 255 seeds simultaneously and can prove or disprove the property.

The goal was to understand how these techniques complement each other when verifying an intentionally unfair LFSR-based arbiter. ***Code implementations and instructions to run it can be found at https://github.com/rrameshh/lfsr-arbiter-verification.***

2. BACKGROUND

Property Strengthening

Property strengthening [1] addresses the verification challenge by checking easier properties first. Instead of immediately testing "grant within 100 cycles," the technique tests strengthened properties like "grant within 25 cycles" or "grant within 50 cycles." When a strengthened property fails, the counterexample reveals a specific LFSR seed that causes problems. This seed is then used to create an underapproximation: the arbiter is fixed to use only that specific seed, creating a simpler model that is faster to verify.

Underapproximation & Bounded Model Checking

Underapproximation reduces the state space by constraining some inputs. While unsound for proving correctness (checking one seed does not prove all 255 are correct), it is sound for bug finding: any bug found in the underapproximation exists in the original system. This makes it ideal for rapid debugging during development.

BMC systematically checks all possible states up to a set number of cycles. For our LFSR arbiter, we let the SMT solver treat the seed as a symbolic variable, basically saying, “the seed could be anything from 1 to 255.” This way, the solver doesn’t test each seed one by one; it reasons about all of them at once. If it finds a problem, it points to the exact seed that causes starvation and produces a trace showing exactly how the sequence plays out.

Roshni Ramesh
18-341 End of Semester Project

4. EXPERIMENTAL RESULTS

4.1 Property Strengthening (Algorithm 1)

Platform: Synopsys VCS T-2022.06 on CMU ECE Linux machines. Runtime: 0.72 seconds

Algorithm 1 was configured with a full bound of k=50 cycles. An initial random test using seed 0x54 confirmed that the arbiter satisfies the specification under typical conditions, granting after 46 cycles. The binary-search phase then attempted progressively tighter bounds, selecting a new random seed for each test. For smaller bounds, most seeds granted the request within the limit, quickly passing the test. However, at certain bounds, some seeds failed to produce a grant within the reduced cycle limit. Specifically, seeds 0x43 and 0xd8 failed both the reduced and full-bound tests, indicating that they could cause starvation beyond 50 cycles.

4.2 Bounded Model Checking

Platform: SymbiYosys with Boolector SMT solver on my local machine. Method: Symbolic execution with 'anyconst' seed modeling. Runtime: 39 seconds.

BMC explored depths 0 through 50 systematically. The assertion failed at step 50 (the boundary) I had specified as: seed 0x45 causes starvation at exactly k=50 cycles. A counterexample trace (trace.vcd) was generated showing the LFSR sequence from seed 0x45, req[0] remaining active continuously, grant[0] never being asserted, and the starvation counter reaching 50.

5. Lessons Learned / Takeaways

Through these five iterations, the strengthened-property approach efficiently identified the problematic seeds without exhaustively simulating all 255 possibilities. By focusing on strategically selected bounds and using underapproximation, the method reduced the number of tests by roughly 51x while still capturing all observed failing seeds, demonstrating its power for rapid bug detection in LFSR-based arbiters. Bounded Model Checking, on the other hand, lets you prove correctness for all seeds at once. Using symbolic ‘anyconst’ seeds, the SMT solver explored all 255 possibilities and produced a counterexample for the failing seed, giving confidence that no other seeds would break the specification. Together, these methods show a nice balance: fast bug-hunting with strengthened properties, and full coverage with BMC when you need it.

7. REFERENCES

[1] J. Auerbach, P. Chitteti, G. Feygin, P. Hoover, S. Moalemi, "Formal Verification of Arbiters using Property Strengthening and Underapproximations," Formal Methods in Computer-Aided Design (FMCAD), 2010.

[2] SymbiYosys Documentation, YosysHQ, https://symbiyosys.readthedocs.io/