R-Verify™: Deep Checking of Embedded Code
Reservoir Labs, Inc.

R-Verify Design

Deep Bit-Level Checking

SAT-Based Static Assertion Checking

Definition: Given a Boolean formula $F$, decide if there is an assignment to the variables in $F$ such that $F$ evaluates to true.

Example: $F = (\neg x_1 \lor \neg x_2) \land (x_1 \lor x_2 \lor \neg x_3) \land (x_1 \lor x_3)$

Solution: $F$ evaluates to true (is satisfied) if $x_1 = 0$, $x_2 = 1$, $x_3 = 1$.

Application to VSIPL

How does it work?
1. Model the program (P) and the verification conditions (Q) using constraints.
2. Use Alef SAT solver to look for a solution to: $P \land \neg Q$
3. Decode an Alef solution into a counter-example

Development & Verification Cycle

R-Verify is built on three core Reservoir Labs technologies:
- **R-Stream™** — a high-level compiler which contains a state-of-the-art framework that permits a wide-range of program analyses
- **Salt™** — a constraint translator which efficiently generates optimized constraint systems from high-level program descriptions
- **Alef™** — a multiprocessor Boolean satisfiability (SAT) solver which contains novel algorithms that allow it to take advantage of HPC hardware

“R-Verify can quickly and efficiently find deep software errors that are invisible to superficial abstraction-based analysis. These types of errors are common in low-level embedded code and can easily go undetected during the normal software testing process. These undiscovered errors can be fatal in deployed software.”

The Satisfiability Problem