Arithmetic Verification Problems Submitted to the SAT Race 2019

Daniela Kaufmann Manuel Kauers Armin Biere
Johannes Kepler University Linz

David Cok
Safer Software Consulting

Multiplier Miters

In the benchmark description of our arithmetic challenge problems [1] submitted to the SAT Competition 2016, we have mentioned that there is another source of multiplier designs, which we could not retrieve back then. These circuits described in [2] were used in [3] and then synthesized and translated to AIGs in our related work [4]. Furthermore, the corresponding web-service "Arithmetic Module Generator" for generating the circuits (in Verilog) became recently available again at https://www.ecsis.riec.tohoku.ac.jp/topics/amg/. For the SAT Race 2019 we generated AIG miters and encoded them into CNF for interesting bit-widths 10, 12 and 14, where SAT solvers not using algebraic reasoning start to have a hard time. These benchmarks compare pairwise several multipliers with different architectures and characteristics. We also considered unsigned multipliers and a few signed multipliers (these are all $n \times n$ inputs to $2n$ bits outputs multipliers where signedness makes a difference). We compare two signed architectures "2cbpwtcl" and "2csparrc" with prefix "eq2..." which gives 6 signed benchmarks for bit-widths 10,12,14. The 12 unsigned multiplier architectures we compare are

bparcl, bparrc, bpctbk, bpdtlf, bpwtcl, bpwtrc,
sparcl, sparrc, spctbk, spdtlf, spwtcl, spwtrc

for bit-widths 10,12 and 14, which gives $396 = 3 \cdot 12 \cdot 11$ unsigned benchmarks (all with "eq..." but without "eq2", "btor" nor "ktsb" in their name).

Karatsuba Multiplication

As crafted benchmark we generated a bit-vector implementation of a single recursive step of the well-known Karatsuba multiplication algorithm. The implementation is then compared against a full multiplier of the same architecture (BTOR). We submitted only the three benchmarks "eqbtor10ktsb{10,12,14}*.cnf" for bit-widths 10,12 and 14.

The Crux of Multiplier Verification

During our work on multiplier verification we came across the issue that within a single column of a multiplier circuit (producing a certain output bit) the sum of the partial products can be permuted in an arbitrary order. Since adding up these partial products within a column needs adders of logarithmic size this summation requires bit-vector reasoning. In different multipliers these adders are ordered and grouped differently, which we conjecture to be the "crux" of multiplier verification on the bit-level.

To capture this problem we generated benchmarks which add up n bits with two input adder trees in a random order and grouping. The input bits are zero extended to m bits, which is the minimum number such that $2^m > n$. Then we generate two different random adder trees. Each tree consists of $n-1$ adders of bit-width m. The outputs of the two trees are compared, which is getting hard for standard SAT solvers on the CNF level at around $n = 30$ bits. We used 10 different seeds for $n = 20, \ldots, 32$ and thus submitted 130 benchmarks "cruxmiters{20,...,32}seed[0-9].cnf".

Integer Overflow Check

In program analysis of code similar to the following C program, the overflow check might yield hard bit-vector problems:

```
void *calloc (size_t a, size_t b) {
  if (((size_t)-1) / a < b) return NULL;
  return memset (malloc (a*b), 0, a*b);
}
```

Here is a corresponding SMT formula for this check

```
(set-logic QF_BV)
(declare-fun a () (_ BitVec 32))
(declare-fun b () (_ BitVec 32))
(assert
  (not (=
    ((_ extract 63 32)
     (bvmul ((_ zero_extend 32) a)
    ((_ zero_extend 32) b)))
    (_ bv0 32))))
(assert
  (bvuge (bvudiv (bvnot (_ bv0 32)) a) b))
```

This is for a 32-bit machine. We generated 29 instances for bit-widths 20 to 48 called "davidcokchallenge{20,...,48}.cnf". This problem is getting hard around 36 bits.

Supported by FWF, NFN Grant S11408-N23 (RiSE)

References

[1] A. Biere, "Collection of combinational arithmetic miters submitted to the SAT Competition 2016," in *SAT Competition 2016*, ser. Department of Computer Science Series of Publications B, T. Balyo, M. Heule, and M. Järvisalo, Eds., vol. B-2016-1. Univ. Helsinki, 2016, pp. 65–66.

[2] N. Homma, Y. Watanabe, T. Aoki, and T. Higuchi, "Formal design of arithmetic circuits based on arithmetic description language," *IEICE Transactions*, vol. 89-A, no. 12, pp. 3500–3509, 2006.

[3] A. Sayed-Ahmed, D. Große, U. Kühne, M. Soeken, and R. Drechsler, "Formal verification of integer multipliers by combining Gröbner basis with logic reduction," in *DATE*. IEEE, 2016, pp. 1048–1053.

[4] D. Ritirc, A. Biere, and M. Kauers, "Column-wise verification of multipliers using computer algebra," in *FMCAD*, D. Stewart and G. Weissenbacher, Eds. IEEE, 2017, pp. 23–30.