How much have you progressed?

Verification of Polynomial Division through BDDs

Team members: Fernando Araujo, Nathan Sartnurak, Kidus Yohannes

Project Idea:

Implement the polynomial division algorithm using BDDs. Given a circuit and its gate-level polynomial representation, apply polynomial reduction on the output polynomials with the entire set of polynomials to return a final polynomial made up of primary inputs that represent the circuit outputs. Under successful implementation, the output polynomial will match the initial specification and verification will be satisfied.

Objectives:

- Create several Singular scripts that apply polynomial division for different boolean functions.
- Provide a polynomial representation of a circuit (for each gate).
- Represent each polynomial as an individual BDD and apply a lexicographical ordering.
- Apply Multivariate Reduction Algorithm
  - Create a Leading Monomial function **(main challenge)**
  - Create a Cube Division function **(main challenge)**
  - Apply operations (XOR, AND) and verify algorithm correctness against Singular script

that's the main challenge.

this may already exist in the CUDD package- check!

Algorithm 1 Multivariate Reduction of f by $F = \{f_1, \ldots, f_s\}$

```
1:  procedure multi_variate_reduce(f, {f_1, ..., f_s}, f_i ≠ 0)
2:      u_i ← 0; r ← 0; h ← f
3:      while h ≠ 0 do
4:          if ∃i s.t. lm(f_i) | lm(h) then
5:              choose i least s.t. lm(f_i) | lm(h)
6:              u_i = u_i + lt(h)/lt(f_i)
7:              h = h − (lt(h)/lt(f_i)) f_i
8:          else
9:              r = r + lt(h)
10:             h = h − lt(h)
11:     return ({u_1, ..., u_s}, r)
```

Figure 1: Multivariate Reduction Algorithm [2]

Methodology:

1. Read the paper on Grobener Basis Reductions and multivariate reduction for understanding.
2. Create example circuits as their corresponding polynomials in a lexicographical representation and then perform polynomial division to find the remainder. Do this using

Try only BDD first.

Singular and by hand to get an understanding of the resulting remainder we are supposed to get.

3. Learn to represent polynomials as BDDs. Go through the documentation to become more familiar with BDD functions. Understand the difference between ZDDs from the paper. Use Graphviz to represent corresponding boolean functions.
4. Apply lexicographical ordering using ASCII value.
5. Explore and experiment with leading monomial examples. Look for any connections between the polynomial and BDD representation. Create some sort of BDD traversal algorithm to find the leading monomials We can recursive function call to traverse the monomial path until we reach a terminal node.
6. Come up with some sort of cube division algorithm. Understand examples and verify. Possibly keep track of each variable and its degree. Or if there's an existing library.
7. Apply addition using XOR because of modulo 2, apply multiplication using AND, implement the rest of the algorithm pseudocode reference, and verify against Singular.

Tools:

- Singular (Polynomials, reference)
- CUDD (BDD representation, verification)

References:

[1] Dr. Kalla's slides [GB Reduction]

[2] https://my.ece.utah.edu/~kalla/papers/TCAD-GBR-ZDD.pdf

Take poly: $x + yz$ in F_2 Let $x > y > z$

or $x \oplus yz$ in Boolean domain

Draw BDD using CUDD.

Can you look at the structure of BDD and recognize the leading term?

Print this BDD using cudd dumpdot() function from the package.