ECE/CS 5745/6745: Testing and Verification of Digital Circuits
Hardware Verification Using Symbolic Computation

Prepared by *Priyank Kalla*

Fall 2023, Homework # 5

Due Date: Nov 10, by midnight. Please complete the HW by the due date, as I will give you next and last HW right after. This is a short and relatively straight-forward HW, the next one will require programming in Singular.

- (Reading Assignment) Please read the book chapter I've provided on Galois fields and hardware design, uploaded on the class website.

- (To make your Mathy-life easy) For your benefit, a LaTeX tarball is also uploaded on the class website, under the 'CAD tools and other resources' section. If you aren't a LaTeX expert, you can now get started with using LaTeX for typesetting technical documents and manuscripts. I hope you will be brave enough to typeset this HW and subsequent HWs also in LaTeX ☺. Download the tarball, and go through the file latex-for-class.pdf. It is self-explanatory.

- This HW assignment gives you some practice with the following concepts: (i) Some basic concepts of Galois fields (GF); (ii) design of GF circuits; and (iii) setting up verification problems in Singular. The theory of equivalence checking using algebraic techniques will follow in the next two assignments. For all computations, you are strongly encouraged to use Singular.

- Singular is installed on the CADE lab machines under `'/usr/local/bin/Singular'`. Notice that the 'S' in Singular is upper-case. Feel free to download the latest version on your own personal computers for use.
- Along with this HW, I am uploading some example Singular files, particularly the ones that I used to give you a demo in class.

- On a unix terminal, the way to load a singular script file is as follows:

```
prompt>> Singular
                        SINGULAR                                 /
 A Computer Algebra System for Polynomial Computations   /   version 3-1-1
                                                       0<
     by: G.-M. Greuel, G. Pfister, H. Schoenemann        \   Feb 2010
FB Mathematik der Universitaet, D-67653 Kaiserslautern    \
> < "finite-field-demo.sing";
```

Okay, so the HW questions are as follows:

1) **(Finite fields – 20 points.)** In class, we generated the field $\mathbb{F}_{16}$ as $\mathbb{F}_2[x] \pmod{x^4+x^3+1}$, where x^4+x^3+1 is a primitive polynomial. Now you are asked to generate $\mathbb{F}_{16} = \mathbb{F}_2[x] \pmod{P(x)}$ where $P(x) = x^4+x^3+x^2+x+1$, and let $P(\alpha) = 0$. Identify a *primitive element* of the field. In other words, find an element β such that $\beta = a_3\alpha^3 + a_2\alpha^2 + a_1\alpha + a_0$ for $a_i \in \{0,1\}$, and that $\mathbb{F}_{16} = \{0, 1 = \beta^{15}, \beta, \beta^2, \ldots, \beta^{14}\}$.

2) **(Understanding Vanishing Polynomials and field containment – 10 points)**. In class, we have seen that for any field $\mathbb{F}_q$, we have that $x^q = x$ or $x^q - x = 0$ for all $x \in \mathbb{F}_q$. We call $x^q - x$ as the vanishing polynomial (or the field polynomial) of $\mathbb{F}_q$, as every element of $\mathbb{F}_q$ is a root of $x^q - x$. Note that any element outside of $\mathbb{F}_q$ may not satisfy $x^q - x = 0$. Now, let $\mathbb{F}_{q1}$ and $\mathbb{F}_{q2}$ be two finite fields such that $\mathbb{F}_{q1} \subset \mathbb{F}_{q2}$. Then for all elements $x \in \mathbb{F}_{q1}$, we will have $x^{q1} = x$. Similarly, for all $y \in \mathbb{F}_{q2}$, we will have $y^{q2} = y$. However, we may or may not have $y^{q1} = y$. You will confirm this with the following experiment:
   In my book chapter, consider Ex. 1.1 and 1.2, along with Fig. One.1 in Section IV. Here we construct $\mathbb{F}_{16} = \mathbb{F}_2[x] \pmod{P(x) = x^4+x^3+1}$, with $P(\alpha) = 0$. We found that $\alpha^5, \alpha^{10} \in \mathbb{F}_4 = \mathbb{F}_2[x] \pmod{x^2+x+1}$. Show that α^5, α^{10} satisfy (are the roots of) $x^4 - x$, whereas any element <u>outside</u> of $\mathbb{F}_4 = \{0, 1, \alpha^5, \alpha^{10}\}$ does not satisfy $x^4 - x$, but it should satisfy $x^{16} - x$. You may use Singular for this computation.

3) **(A finite fields challenge for ECE/CS 6745 students – 20 points. ECE/CS 5745 students are not required to solve it, but they can attempt it for extra credit.)** Let $\alpha_1, \alpha_2, \ldots, \alpha_t$ be arbitrary elements in

$\mathbb{F}_{2^k}$. Prove that:

$$(\alpha_1 + \alpha_2 + \cdots + \alpha_t)^{2^i} = \alpha_1^{2^i} + \alpha_2^{2^i} + \cdots + \alpha_t^{2^i}$$

for $i = 1, 2, \ldots$ [Hint: You could first try to prove $(a+b)^2 = a^2 + b^2$ in $\mathbb{F}_{2^k}$, then generalize the result using induction. Alternatively, you can also try to exploit the binomial expansion theorem: $(a+b)^p = \sum_{k=0}^{p} \binom{p}{k} a^{p-k} b^k$, set $p = 2$ and reduce the coefficients modulo $p = 2$, and then continue on with induction.]

4) **(GF multiplier design & Miter construction – 30 points)** In my slides and in my book chapter on finite fields, I have shown you how to design a Mastrovito multiplier circuit that performs multiplication $Z = A \cdot B \pmod{P(x)}$, where $A = \{a_{k-1}, \ldots, a_0\}, B = \{b_{k-1}, \ldots, b_0\}$ are 2 k-bit inputs, $Z = \{z_{k-1}, \ldots, z_0\}$ is the k-bit output and $P(x)$ is the given primitive polynomial. In the slides, I have given you a design of a 4-bit circuit, as well as that of a 2-bit circuit. In addition, the book chapter also shows a circuit schematic for a 4-bit Mastrovito multiplier. Study these multiplier design concepts carefully, and then:

a) Design a 3-bit Mastrovito multiplier over the Galois field $\mathbb{F}_8 = \mathbb{F}_2[x] \pmod{P(x)}$ using $P(x) = x^3 + x + 1$.

b) In the lecture slides on GF, I have also shown you how to construct a miter between a polynomial spec and a circuit implementation (towards the end of that slide-set).

c) Moreover, on the class webpage, along with this HW, I have also uploaded a file '2-bit-multiplier.sing' that shows: i) how to create/write an algebraic miter in Singular; ii) declare the ideal J generated by the miter's polynomials; and iii) compute the Gröbner basis 'G = groebner(J)'. Study this file and execute it in Singular to interpret the result. [I will explain in the class what the algebraic constructs *ideal* and *Gröbner bases* are.]

d) Create a similar 'algebraic miter' between your 3-bit GF multiplier and the spec $f_{spec} : Z + A \cdot B$. Describe the ideal J generated by the miter's polynomial in Singular and compute it's Gröbner basis 'groebner(J)'. What do you observe?

e) Can you ascertain whether or not your design is indeed a correct (bug-free) implementation? Explain.

f) Now purposely introduce a bug in the design – say, by changing some gate – and then observe the output of 'groebner(J)'.

g) Submit your circuit schematic, the polynomial ideal, the singular file, and its output.

5) **Lagrange Interpolation (20 points)** Consider the function (mapping) $f : \mathbb{B}^3 \rightarrow \mathbb{B}^3$ shown in the truth-table below.

$A = \{a_2 a_1 a_0\}$	$\mapsto$	$Z = \{z_2 z_1 z_0\}$
000	$\mapsto$	000
001	$\mapsto$	001
010	$\mapsto$	111
011	$\mapsto$	111
100	$\mapsto$	101
101	$\mapsto$	011
110	$\mapsto$	101
111	$\mapsto$	101

Interpret this function $f : \mathbb{B}^3 \rightarrow \mathbb{B}^3$ as a function $f : \mathbb{F}_{2^3} \rightarrow \mathbb{F}_{2^3}$. Recall from the lecture slides on finite fields (also see Sec VI in my book chapter on finite fields), any function $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$ is a polynomial function; i.e. there exists a polynomial $Z = \mathcal{F}(A)$ that describes this function.

a) Using the Lagrange's interpolation formula (Eqn. One.5, Sec VI in my book chapter), derive a unique, minimal, canonical polynomial representation of the function as $Z = \mathcal{F}(A)$ over $\mathbb{F}_{2^3}$. In other words, derive an expression for $\mathcal{F}(A) \in \mathbb{F}_{2^3}[A]$. Once again, you should use Singular to compute $\mathcal{F}(A)$.

b) Using Singular, evaluate the derived polynomial expression $\mathcal{F}(A)$ for all inputs, and show that your polynomial correctly models the given function. You are essentially performing simulation-based verification here! [Hint: see the `subst` command in Singular].

c) In the next homework, you will design a circuit corresponding to this function and perform formal equivalence checking using Gröbner bases.