Multiplicative Complexity and Solving Generalized Brent Equations With SAT Solvers

Nicolas T. Courtois
University College London,
Gower Street, London, UK
Email: n.courtois@ucl.ac.uk

Daniel Hulme
University College London,
Gower Street, London, UK
Email: d.hulme@cs.ucl.ac.uk

Theodosis Mourouzis
University College London,
Gower Street, London, UK
Email: theodosis.mourouzis.09@ucl.ac.uk

***Abstract*—In this paper we look at the general problem of Multiplicative Complexity (MC) as an essential tool for optimizing potentially arbitrary algebraic computations over fields and rings in the general non-commutative setting. Our goal is to find optimizations in a fully automated way via algebraic formal coding and conversion to a SAT problem [1].**

We focus on the basic problems of minimizing the number of multiplications in Matrix Multiplication, complex number multiplication and also quaternion multiplication. Minimizing the number of multiplications in the Matrix Multiplication problem alone (and this for problems of fixed size some of which we were able to optimize [4]) is known to be able to lead to immediate improvements in countless other algorithms on formal languages, graphs, arbitrary finite groups, various real/complex/algebraic rings and fields of practical importance. Thus we may hope to translate our efforts to improve many high-profile applications in computer graphics, signal processing, cryptography, computational physics and chemistry, weather prediction, financial computing, Google page ranking, etc.

The classical tool to solve the Matrix Multiplication problem are the Brent Equations [3]. We have developed a methodology for solving these equations over small fields such as $\mathrm{GF}(2)$ with a conversion to a SAT problem and progressive lifting to larger fields and rings. We generalize the Brent Equations [3] and extend our method to similar algebraic optimizations and to tri-linear problems.

We have been able to obtain new results to decrease the MC of several well known operations in algebra, which to the best of our knowledge are new. For example we have obtained a new general 3×3 matrix multiplication method with 23 multiplications [4]. We also present new formulas for complex number multiplications and quaternion multiplications. Additionally, using our methodology we are able to produce highly optimized implementations of small circuits. We obtained exact lower bounds with respect to MC of two very well known block ciphers, such as PRESENT and GOST, known for their exceptionally low implementation cost. Our method is efficient for any sufficiently small circuit [5].

***Index Terms*—Linear Algebra, Fast Matrix Multiplication, Complex Numbers, quaternions, Strassen's algorithm, Multiplicative Complexity**

I. Introduction

The optimization of certain arbitrary algebraic computations over fields and rings in the general non-commutative setting is considered as one of the most important topics in theoretical computer science and mathematics. In this paper we study how the Multiplicative Complexity (MC) of certain arbitrary algebraic computations such as the Matrix Multiplication (MM), multiplication of complex numbers, multiplication of quaternions and of a general Boolean circuit can be reduced over small fields such as $\mathrm{GF}(2)$, the field of two elements, and then be progressively lifted to larger rings.

MC is the minimum number of AND gates that are needed if we allow an unlimited number of NOT and XOR gates. Informally, we are interested in reducing the number of multiplications involved in an arbitrary algebraic computation allowing unlimited number of additions.

Our method consists of three basic steps. In the first step we formally encode the problem by writing a system of equations which describe the problem and then we consider the problem over the finite field of two elements $\mathrm{GF}(2)$. In case of the MM problem and the complex or the quaternion multiplication problem we use the Brent Equations [3] in the encoding step while for circuit minimization we encode the problem formally as a straight-line representation problem, described by a quantified set of multivariate relations [5]. Then we proceed by converting the reduced modulo 2 problem to a SAT problem using the Courtois-Bard-Jefferson method [2] and then we progressively lift the solution to larger fields and rings using different heuristic techniques and other constraint satisfaction algorithms.

A. *Motivation for Low MC*

Matrix Multiplication:

One of the most famous problems in computer algebra is the problem of MM of square and non-square matrices, where the aim is to reduce the number of 2-input multiplications needed in order to compute the product of two matrices. A speed-up in MM will automatically result in a speed improvement of many other algorithms and techniques such as:

- Gauss Elimination algorithm for solving a system of linear polynomial equations
- Algorithms for solving of non-linear polynomial equations
- Recognizing if a word of length n belongs to a context-free language
- Transitive closure of a graph or a relation on a finite set
- Cryptanalysis

Circuit Complexity:

We refer to some reasons why circuits of low MC are very important especially for industrial reasons and for cryptography. For more analytic explanations, see [5].

- Lower the hardware implementation cost of a cipher in silicon
- Develop certain so called Bitslice parallel-SIMD software implementations of block ciphers such as in [16]
- In symbolic computing and numerical algebra, this kind of optimization can be applied recursively to produce asymptotically fast algorithms to solve very famous and important practical problems such as Gaussian reduction and MM
- Prevent Side Channel Attacks (SCA) on smart cards such as Differential Power Analysis (DPA) [15].

II. Methodology

We have fully automated the process as follows:

1) Form the Brent Equations (or write a quantified set of multivariate relations that describes the problem)
2) Consider only solutions in 0,1=integers modulo 2
3) Convert to SAT with Courtois-Bard-Jefferson method [2]
4) Lift the solution from $\mathrm{GF}(2)$ to the general bigger fields by another constraint satisfaction algorithm

A. Brent Equations

We use Brent Equations as a sort of "formal algebraic" method for encoding problems that optimize certain arbitrary algebraic computations. Our main idea is to encode such problems into a "language" which can be converted to a SAT problem and then we attempt to solve this hard problem using our portfolio of 500 SAT solvers.

Suppose we want to multiply a $M \times N$ matrix A by a $N \times P$ matrix B using T 2-input multiplications.

We solve the above problem by solving the following system of $(MNP)^2$ equations in $T(MN+NP+MP)$ unknowns, see [3]:

$$\{\forall i \forall j \forall k \forall L \forall m \forall n, \textstyle\sum_{p=1}^{T} \alpha_{ijp}\beta_{kLp}\gamma_{mnp} = \delta_{ni}\delta_{jk}\delta_{Lm}\}(1)$$

A solution to this set of equations implies that the coefficient entries c_{ij} of the product matrix $C = AB$ can be written as $c_{nm} = \Sigma_{p=1}^{T}\gamma_{mnp}q_p$ (2) where the products $q_1, q_2, ..., q_T$ are given by $q_p = (\Sigma\alpha_{ijp}a_{ij})(\Sigma\beta_{KLp}b_{KL})$ (3).

Thus, our aim is to form Brent-like equations for other problems such as complex multiplication and quaternion multiplication and then convert it to a SAT problem where we can apply our portfolio of SAT solvers to get the solution.

B. SAT Solvers

Satisfiability (SAT) is the problem of determining if the variables of a given Boolean formula can be assigned in a way as to make the formula evaluate to TRUE [13]. SAT was the first known example of an NP-complete problem. A wide range of other decision and optimization problems can be transformed into instances of SAT and a class of algorithms called SAT solvers can efficiently solve a large enough subset of SAT instances such as MiniSAT solver [23]. Our aim is to transform problems like MM into SAT problems.

C. Solving Brent Equations Modulo 2 and Lifting

In the first step we form the Brent Equations for our problem and we consider them over the field $\mathrm{GF}(2)$. We are interested only in simple solutions that work over small finite rings and fields. Then using the Courtois-Bard-Jefferson converter we convert this system of equations over $\mathrm{GF}(2)$ to a SAT problem and attempt to solve it. After obtaining the solution modulo 2 we begin again and try to lift the solution to a modulo 4 solution using very similar formal encoding.

D. Solving and Conversion

The system of equations is encoded algebraically and then converted to a SAT problem. We have implemented a method to convert this very hard problem to a SAT problem, and we have attempted to solve it, with our portfolio of some 500 SAT solvers and their variants.

III. Matrix Multiplication

Many attempts to solve the general MM problem in the literature work by solving fixed-size problems and applying the solution recursively. This leads to pure combinatorial optimization problems with fixed size. For square matrices the naive algorithm is cubic and the best known theoretical exponent is 2.376, due to Coppersmith and Winograd [14]. This exponent is quite low and it is conjectured that one should be able to do MM in so called "soft quadratic time", with possibly some poly-logarithmic overheads, which could even be sub-exponential in the logarithm. This in fact would be nearly linear in the size of the input.

In 2005 a team of scientists from Microsoft Research and two US universities established a new method for finding such algorithms based on group theory, and their best method so far gives an exponents of 2.41 [17], close to Coppersmith-Winograd result and subject to further improvement.

All attempts to solve the MM problem in the literature rely on solving certain fixed size problems, which can be the recursively applied to produce asymptotically fast algorithms that can be used for more general cases. In 1969 Victor Strassen established a first asymptotic improvement to the complexity of MM algorithm, by proving that two matrices 2×2 can be multiplied by using seven instead of eight multiplications [22]. Later in 1975 Laderman published a solution for multiplying 3×3 matrices with 23 multiplications [9]. Since then this topic generated very considerable interest and yet to this day it is not clear if Laderman's solution in case of 3×3 multiplication can be further improved.

As in many previous attempts to solve the problem we proceed by solving the so called Brent equations [3]. This approach has been tried many times before, see [[3],[8],[10],[12],[13],[11]].

We write the coefficients of each product as three 3×3-matrices for each multiplication $A^{(i)}, B^{(i)}$ and $C^{(i)}$, $1 \leq i \leq r$, with $r = 23$ where A will be the left hand side of each product, B the right hand size, and C says to which coefficient of the result this product contributes.

The Brent equations are as follows:

$\forall i \forall j \forall k \forall l \forall m \forall n \; \sum_{i=1}^{r} A_{ij}^{(i)} B_{kl}^{(i)} C_{mn}^{(i)} = \delta_{ni}\delta_{jk}\delta_{lm}$ (4)

For 3×3 matrices we get exactly 729 cubic equations. Then using our methodology we obtained the following solution for the case of 3×3 matrices. Our solution in non-isomorphic to any of the existing solutions:

$P01 := (a_{23}) * (-b_{12} + b_{13} - b_{32} + b_{33});$
$P02 := (-a_{11} + a_{13} + a_{31} + a_{32}) * (b_{21} + b_{22});$
$P03 := (a_{13} + a_{23} - a_{33}) * (b_{31} + b_{32} - b_{33});$
$P04 := (-a_{11} + a_{13}) * (-b_{21} - b_{22} + b_{31});$
$P05 := (a_{11} - a_{13} + a_{33}) * (b_{31});$
$P06 := (-a_{21} + a_{23} + a_{31}) * (b_{12} - b_{13});$
$P07 := (-a_{31} - a_{32}) * (b_{22});$
$P08 := (a_{31}) * (b_{11} - b_{21});$
$P09 := (-a_{21} - a_{22} + a_{23}) * (b_{33});$
$P10 := (a_{11} + a_{21} - a_{31}) * (b_{11} + b_{12} + b_{33});$
$P11 := (-a_{12} - a_{22} + a_{32}) * (-b_{22} + b_{23});$
$P12 := (a_{33}) * (b_{32});$
$P13 := (a_{22}) * (b_{13} - b_{23});$
$P14 := (a_{21} + a_{22}) * (b_{13} + b_{33});$
$P15 := (a_{11}) * (-b_{11} + b_{21} - b_{31});$
$P16 := (a_{31}) * (b_{12} - b_{22});$
$P17 := (a_{12}) * (-b_{22} + b_{23} - b_{33});$
$P18 := (-a_{11} + a_{12} + a_{13} + a_{22} + a_{31}) * (b_{21} + b_{22} + b_{33});$
$P19 := (-a_{11} + a_{22} + a_{31}) * (b_{13} + b_{21} + b_{33});$
$P20 := (-a_{12} + a_{21} + a_{22} - a_{23} - a_{33}) * (-b_{33});$
$P21 := (-a_{22} - a_{31}) * (b_{13} - b_{22});$
$P22 := (-a_{11} - a_{12} + a_{31} + a_{32}) * (b_{21});$
$P23 := (a_{11} + a_{23}) * (b_{12} - b_{13} - b_{31});$

$c_{11} = P02 + P04 + P07 - P15 - P22;$
$c_{12} = P01 - P02 + P03 + P05 - P07 + P09 + P12$
$+P18 - P19 - P20 - P21 + P22 + P23;$
$c_{13} = -P02 - P07 + P17 + P18 - P19 - P21 + P22;$
$c_{21} = P06 + P08 + P10 - P14 + P15 + P19 - P23;$
$c_{22} = -P01 - P06 + P09 + P14 + P16 + P21;$
$c_{23} = P09 - P13 + P14;$
$c_{31} = P02 + P04 + P05 + P07 + P08;$
$c_{32} = -P07 + P12 + P16;$
$c_{33} = -P07 - P09 + P11 - P13 + P17 + P20 - P21;$

Lemma 1: : Our new solution is neither equivalent to the Ladermans solution [9] nor equivalent to any of the solutions given in [1].

Proof:

Following [1], the Ladermans solution has exactly 6 matrices of rank 3 (which occur in products $P01, P03, P06, P10, P11, P14$). At the same time in all new solutions presented in [1], at most 1 matrix will have rank 3. In our solution we have exactly 2 matrices of rank 3 (which occur in products $P18$ and $P20$, there are 2 and not more such matrices, both being on the left hand size namely $A^{(18)}$, in $A^{(20)}$). This proves that all these solutions are distinct.

Remark: This result demonstrates that the space of solutions to Ladermans problem is larger than expected, and therefore it becomes now more plausible that a solution with 22 multiplications exists. If it exists, we might be able to find it soon just by running our algorithms longer, or due to further improvements in the SAT algorithms.

IV. Complex number multiplication

In order to compute the product $(a + b\mathrm{i}) * (c + d\mathrm{i}) = (ac - bd) + (ad + bc)\mathrm{i}$ (5) we need 4 multiplications using the naive algorithm. Gauss was the first to prove that the multiplication of two complex numbers $(a+b\mathrm{i})*(c+d\mathrm{i})$ can be done using 3 multiplications instead of 4. We obtained the same result using our methodology. We can translate this complex multiplication problem to a MM problem using the isomorphism between the set of complex numbers $\{a + b\mathrm{i} : a, b \in \mathbb{R}\}$ and the 2 dimensional sub-algebra of $\{\begin{pmatrix} a & b \\ c & d \end{pmatrix} : a, b, c, d \in \mathbb{R}\}$, given by:

$\{\begin{pmatrix} a & -b \\ b & a \end{pmatrix} : a, b \in \mathbb{R}\}.$

In the first step we form the 3-dimensional Brent Equations for multiplying two 2x2 matrices A and B and then using SAT solvers and lifting techniques we obtain the seven following Strassen-like products, which can be used to compute the entries $\{c_{11}, c_{12}, c_{21}, c_{22}\}$ of the matrix $C = AB$.

$P1 = (a_{12} + a_{22}) * (b_{12} + b_{22});$
$P2 = (a_{11}) * (b_{11});$
$P3 = (a_{21}) * (b_{11} + b_{12} + b_{21} + b_{22});$
$P4 = (a_{12}) * (-b_{21});$
$P5 = (-a_{11} + a_{12} - a_{21} + a_{22}) * (b_{12});$
$P6 = (-a_{21} + a_{22}) * (b_{21} + b_{22});$
$P7 = (-a_{12} + a_{21} - a_{22}) * (b_{12} + b_{21} + b_{22});$
$c_{11} = P2 - P4;$
$c_{12} = P4 - P5 - P6 - P7;$
$c_{21} = P3 + P4 - P1 - P7;$
$c_{22} = P1 + P6 + P7 - P4;$

Now if we consider these products over the 2-dimensional sub-algebra of matrices defined before we get that $\mathrm{Span}\{P_1, .., P_7\} = \mathrm{Span}\{P_1, .., P_4\}$ since we have $P_5 = 2P_4, P_7 = P_3 - P_2$ (6) and $P_6 = 2P_2 - P_3 - P_1$ (7). This suggests that four products are enough to compute the product of two complex numbers as the naive multiplication. However, if we also consider the set of entries $\{c_{11}, c_{21}\}$ over the new set of products we have that $c_{11} = P_2 - P_4$ (8) and $c_{21} = P_2 + P_4 - P_1$ (9). As we see, our method gives three multiplications in total as proposed by Gauss.

A. Multiplication of three complex numbers

We provide an exceptionally good solution which exists over GF(2) in the non-homogenous case for the problem of multiplying three complex numbers. Multiplication of three complex numbers is a trilinear problem as we aim to minimize the number of multiplications needed to represent the map $f : (V, V, V) \to V$.

Using our method we show that multiplication of three complex numbers $(a + b\mathrm{i}) * (c + d\mathrm{i}) * (e + f\mathrm{i})$ can be achieved using five multiplications at most.

Lemma 2: : $MC((a+bi)*(c+di)*(e+fi)) \leq 5$ over GF(2).

Proof:
In GF(2) we can do 5 multiplications total!
$P1 := (a+b+e+f)*(c+d+e+f)$;
$P2 := (a+e)*(d+e)$;
$P3 := (c+f)*(b+f)$;
$Im := P4 := (P1+P2+P3+a+d+e)*(P1+e+f)$;
$Re := P5 := (P1+e+f)*(P1+P4+a+b+c+d+1)$;

V. Quaternion Algebra

Quaternions are a number system that extends the complex multiplication that were introduced by the Irish Mathematician Sir William Rowan Hamilton, who defined a quaternion as the quotient of two directed lines in a three-dimensional space or equivalently as the quotient of two vectors [7]. It can also be seen as the sum of a scalar and a vector. They are widely used in both theoretical and applied mathematics, especially for calculations involving three-dimensional rotations such as three-dimensional computer graphics and computer vision and in real-time symmetric cryptography [6].

As a set, the quaternions are equal to $\mathbb{R}^4$ and every element can be represented as:

$a1+bi+cj+dk$, where i,j,k satisfy the following relations;

$$i^2=j^2=k^2=ijk=-1,\ ij=k, ji=-k,\ jk=i, kj=-i$$
$$\text{and } ki=j, ik=-j \ (10).$$

The Hamilton product of two quaternions:
$a_1+b_1i+c_1j+d_1k,\ a_2+b_2i+c_2j+d_2k$ is given by
$(a_1a_2-b_1b_2-c_1c_2-d_1d_2)+(a_1b_2+b_1a_2+c_1d_2-d_1c_2)i$
$+(a_1c_2+b_1d_2+c_1a_2+d_1b_2)j+(a_1d_2+b_1c_2-c_1b_2+d_1a_2)k$
(11).

Our aim is to compute the minimum number of 2-input multiplications needed to compute the product of two quaternions. Using the naive multiplication method we need 16 multiplications but this number of multiplication can be reduced using the Gauss method to 12. Using our software we obtain the 12 products that are needed to compute the product of two quaternions over the general non-commutative setting. Additionally, we further investigate the number of 2-input multiplication needed over GF(2) and we surprisingly get eight. Below we provide the encoding of quaternion multiplication problem into Brent Equations and the next *Lemmas* provide the result obtained by our software.

Encoding q_1*q_2 into Brent Equations:
Suppose $\{a_1,a_2,a_3,a_4\}$, $\{b_1,b_2,b_3,b_4\}$ are non-commutative variables and σ_{ijk} is a given three-dimensional array of numbers from the set $\{-1,0,1\}$, and we want to compute the 4 sums of 2-input products: $a_1b_1-a_2b_2-a_3b_3-a_4b_4, a_1b_2+a_2b_1+a_3b_4-a_4b_3, a_1b_3+a_2b_4+a_3b_1+a_4b_2, a_1b_4+a_2b_3-a_3b_2+a_4b_1$.

Then our aim is to find the least possible T and scalars $\alpha_{it},\beta_{jt},\gamma_{kt}$ such that from the T products of the form

$p_t=(\sum_i \alpha_{it}a_i).(\sum_j \beta_{jt}b_j)$ (12) for $1\leq t\leq T$, we can form the q_k as linear combinations of the p_t as

$q_k=\sum_{t=1}^{T}\gamma_{kt}p_t$ (13) for $1\leq k\leq K$.

Combining these two results we formulate the problem of finding the minimum number of 2-input multiplications for multiplying two quaternions $a_1+a_2i+a_3j+a_4k,\ b_1+b_2i+b_3j+b_4k$ as follows:

Quaternion multiplication problem: Find constants $\alpha_{it},\beta_{jt},\gamma_{kt}$ and least T (where $T\leq 12$) such that the following system of 64 equations in $12*T$ unknowns hold:

$$\sum_{t=1}^{T}\alpha_{it}\beta_{jt}\gamma_{kt}=\sigma_{ijk} \ (14),$$
$$\text{for } 1\leq i\leq 4,\ 1\leq j\leq 4,\ 1\leq k\leq 4$$

,where σ_{ijk}: $\sigma_{111}=1$, $\sigma_{122}=1$, $\sigma_{133}=1$, $\sigma_{144}=1$, $\sigma_{212}=1$, $\sigma_{221}=-1$, $\sigma_{234}=1$, $\sigma_{243}=1$, $\sigma_{313}=1$, $\sigma_{324}=-1$, $\sigma_{331}=-1$, $\sigma_{342}=1$, $\sigma_{414}=1$, $\sigma_{423}=1$, $\sigma_{432}=-1$, $\sigma_{441}=-1$ and zero elsewhere.

Lemma 3: $MC(q_1*q_2 : q_i\in\mathbb{H})\leq 12$

Proof: Using the complex representation of q_1 and q_2 we need to compute four entries of the form:

1) $(q_1*q_2)_{11}=(a+bi)*(e+fi)+(c+di)*(-g+hi)$
2) $(q_1*q_2)_{12}=(a+bi)*(g+hi)+(c+di)*(e-fi)$
3) $(q_1*q_2)_{21}=(-c+di)*(e+fi)+(a-bi)*(-g+hi)$
4) $(q_1*q_2)_{22}=(-c+di)*(g+hi)+(a-bi)*(e-fi)$

Using Gauss formulaes we can obtain the first two entries $\{(q_1*q_2)_{11},(q_1*q_2)_{12}\}$ using 12 multiplications. Using this methodology we have obtained the following terms $ae-bf, be+af, ce+fd, ed-fc, ag-bh, bg+ah, -cg-hd, ch-dg$. However the other entries $\{(q_1*q_2)_{21},(q_1*q_2)_{22}\}$ can be computed using these terms multiplied by -1. Using our software we obtained the following formulas for the quaternion multiplication using 12 multiplications which can be also directly verified using MAPLE computer algebra software:

$P01 := (a_4)*(b_2)$;
$P02 := (a_1)*(b_1+b_2+b_4)$;
$P03 := (a_1)*(b_3)$;
$P04 := (-a_1+a_2)*(b_1)$;
$P05 := (-a_2)*(b_1-b_2)$;
$P06 := (a_2)*(b_3)$;
$P07 := (a_2)*(b_4)$;
$P08 := (a_3)*(b_1)$;
$P09 := (a_1+a_3-a_4)*(b_1+b_2)$;
$P10 := (a_3+a_4)*(-b_3)$;
$P11 := (a_1-a_3+a_4)*(b_4)$;
$P12 := (-a_4)*(-b_3+b_4)$;
expand$(-P04-P05+P10+P12-a_1*b_1+a_2*b_2+a_3*b_3+a_4*b_4)$;
expand$(P02+P04-P11-P12-a_1*b_2-a_2*b_1-a_3*b_4+a_4*b_3)$;
expand$(P01+P03+P07+P08-a_1*b_3-a_2*b_4-a_3*b_1-a_4*b_2)$;
expand$(-P01+P02+P06+P08-P09-a_1*b_4-a_2*b_3+a_3*b_2-a_4*b_1)$;

Additionally, we obtain a result over the field GF(2) and our results are summarized in the next lemma. Obtaining results over the field of two elements is very useful as binary encoding is employed in many areas such as cipher design in cryptography and circuit design for either software or hardware implementations.

Lemma 4: $MC(q_1 * q_2 : q_i \in \mathbb{H}) \leq 8$ over $\mathrm{GF}(2)$.

Proof: Using our automated software we obtained the following solution which can be directly verified with MAPLE computer algebra software:

$P01 := (a_2 + a_3) * (b_1 + b_2 + b_4);$
$P02 := (a_1 + a_2 + a_3) * (b_1 + b_2 + b_3 + b_4);$
$P03 := (a_1 + a_2) * (b_2 + b_3 + b_4);$
$P04 := (a_1 + a_3) * (b_1 + b_2 + b_3);$
$P05 := (a_3 + a_4) * (b_1);$
$P06 := (a_1 + a_2 + a_3 + a_4) * (b_2);$
$P07 := (a_2 + a_4) * (b_4);$
$P08 := (a_1 + a_4) * (b_3);$

$\mathrm{expand}(P01 + P02 + P03 + P07 - a_1 * b_1 + a_2 * b_2 + a_3 * b_3 + a_4 * b_4) mod 2;$
$\mathrm{expand}(P02 + P03 + P04 + P08 - a_1 * b_2 - a_2 * b_1 - a_3 * b_4 + a_4 * b_3) mod 2;$
$\mathrm{expand}(P01 + P02 + P03 + P04 + P06 - a_1 * b_3 - a_2 * b_4 - a_3 * b_1 - a_4 * b_2) mod 2;$
$\mathrm{expand}(P01 + P02 + P04 + P05 - a_1 * b_4 - a_2 * b_3 + a_3 * b_2 - a_4 * b_1) mod 2;$

VI. Exact Circuit Complexity Optimization

In case of circuit complexity we employed the heuristic proposed by Boyar and Peralta [18] based on the notion of MC and consists of the following steps:

1. (Step 1) First compute the MC.

2. (Step 2) Then optimize the number of XORs separately, see [[19],[21]].

3. Optional Step 3: At the end do additional optimizations to decrease the circuit depth, and possibly additional software optimizations, see [[18],[20]].

We encode the problem formally as a straight-line representation problem, described by a quantified set of multivariate relations and we convert it to SAT with the Courtois-Bard-Jefferson tool [2]. Our method on how we compute the MC of the circuit is found in [5].

As a proof of concept we consider the following S-box with 3 inputs and 3 outputs, which have been generated at random for the CTC2 cipher [5] and is defined as $7, 6, 0, 4, 2, 5, 1$. We have tried to optimize this S-box with the well known software Logic Friday (based on Espresso min-term optimization developed at Berkeley) and obtained 13 gates. With our software and in a few seconds we obtained several interesting results, each coming with a proof that it is an optimal result (cannot be improved anymore). We get:

Lemma 5: The Multiplicative Complexity (MC) is exactly 3 (we allow 3 AND gates and an unlimited number of XOR gates).

The Bitslice Gate Complexity (BGC) is exactly 8 (allowed are XOR,OR,AND,NOT).

The Gate Complexity (GC) is exactly 6 (allowing NAND,NOR,NXOR).

The NAND Complexity (NC) is exactly 12 (only NAND gates and constants).

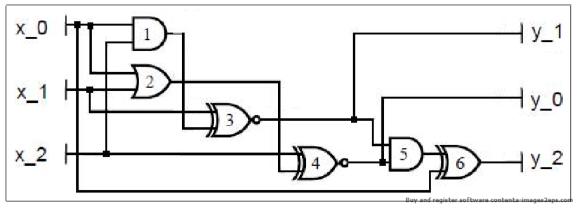

Fig. 1. Our provably optimal implementation of CTC2 S-box with 6 gates.

Proof: Unlike the great majority of circuit optimizations, needed each time a given cipher is implemented in hardware, our results are exact. They are obtained by solving the problem at a given gate count k, the SAT solver outputs SAT and a solution, and if for k-1 gates the *SAT* solver is good enough and fast enough, it will output *UNSAT* and we obtain a proven lower bound, a rare thing in complexity, see [5].

VII. Conclusion

In this paper we study the notion of Multiplicative Complexity (MC) which minimizes the number of elementary non-linear operations (AND gates) at the cost of linear operations. We used MC as an essential tool for optimizing potentially arbitrary algebraic computations over fields and rings in the general non-commutative setting.

We employed an automated method for obtaining new formulas for Matrix Multiplication (MM), complex number and quaternion multiplication based on SAT solvers. We extensively used the notion of Brent Equations [3] as a formal encoding of these problems and then we consider solutions of the corresponding system of equations over the field of two elements. After we algebraically encode the problem we convert it into a SAT problem using the Courtois-Bard-Jefferson [2] and then using our portfolio of 500 SAT solvers we try to solve the problem over $\mathrm{GF}(2)$. Starting from scratch we try to lift the solutions modulo 2 to solutions modulo 4 and also to bigger fields. We lift the solutions using another constraint satisfaction algorithm and some heuristics discovered during our simulations that reduces the complexity of our lifting technique even more.

We have been able to obtain new results in decreasing the MC of several well known operations in algebra, which to the best of our knowledge are new. For example we have obtained a new general 3×3 MM method with 23 multiplications [4]. We also derived new formulaes regarding the multiplication of three complex numbers using 5 multiplications over $\mathrm{GF}(2)$ and for multiplying two quaternions using 8 multiplications over $\mathrm{GF}(2)$. We also derived efficient implementations regarding the MC of some ciphers such as PRESENT, GOST and CTC2 [5].

So far our method works efficiently for obtaining compact representations of algebraic computations or circuits over the field of two elements. In some cases we are able to lift our solutions from GF(2) to the general non-commutative setting. However, our lifting technique sometimes is not efficient and is not able to lift the solutions. As future work we will improve our lifting techniques so that we will be able to obtain similar compact representations which hold over arbitrary non-commutative rings.

Acknowledgment

We would like to thank the anonymous referees of this paper who helped us a lot to improve it.

References

[1] R.W. Johnson and A.M. McLoughlin, *Noncommutative Bilinear Algorithms for 3 x 3 Matrix Multiplication* In SIAM J. Comput., vol. 15 (2), pp.595-603, 1986.

[2] G.V. Bard, N.T. Courtois and C. Jefferson, *Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers* Presented at ECRYPT workshop Tools for Cryptanalysis, 2007.

[3] R. Brent, *Algorithms for matrix multiplication* Tech. Report Report TR-CS-70-157,Department of Computer Science, Stanford, 52 pages, 1970.

[4] N.T. Courtois, G.V. Bard2, and D. Hulme, *A New General-Purpose Method to Multiply 3x3 Matrices Using Only 23 Multiplications* At http://arxiv.org/abs/1108.2830, 2011.

[5] N.T. Courtois, D. Hulme, and T. Mourouzis, *Solving Circuit Optimisation Problems in Cryptography and Cryptanalysis* Appears in electronic proceedings of 2nd IMA Conference Mathematics in Defence, UK, Swindon, 2011.

[6] R. Anand, G. Bajpai and V. Bhaskar, *Real-Time Symmetric Cryptography using Quaternion Julia Set* IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.3, 2009

[7] W.R. Hamilton, *On quaternions, or on a new system of imaginaries in algebra* Philosophical Magazine. Vol. 25, n 3. p. 489495, 1844

[8] G. Bard, *New Practical Approximate Matrix Multiplication Algorithms found via Solving a System of Cubic Equations* A draft paper submitted to a journal, can be found at: http://www-users.math.umd.edu/ bardg/

[9] J.D. Laderman, *A Non-Commutative Algorithm for Multiplying 3x3 Matrices Using 23 Multiplications* ull. Amer. Math. Soc. Volume 82, Number 1, 1976

[10] W. Smith, *Fast Matrix Algorithms And Multiplication Formulae* Available at:https://math.cst.temple.edu/ wds/matgrant.ps.

[11] N. Burr, *An investigation into fast matrix multiplication* done under supervision of Nicolas T. Courtois, and submitted as a part of BSc Degree in Computer Science at Univesity College London, 2010

[12] G. Bard, *New Practical Approximate Matrix Multiplication Algorithms found via Solving a System of Cubic Equations* A draft paper submitted to a journal, can be found at: http://www-users.math.umd.edu/ bardg/

[13] G. Bard, *Algorithms for Solving Linear and Polynomial Systems of Equations over Finite Fields with Applications to Cryptanalysis* Submitted in Partial Fulfillment for the degree of Doctor of Philosophy of Applied Mathematics and Scientific Computation, 2007

[14] D. Coppersmith and S.Winograd *On the asymptotic complexity of matrix multiplication* SIAM Journal Comp., 11, pp 472-492 , 1980

[15] E. Prouff, C. Giraud, and S. Aumonier *Provably Secure S-Box Implementation Based on Fourier Transform* In CHES 2006, Springer LNCS 4249, pp: 216-230, 2006

[16] M. Albrecht, N.T. Courtois, D. Hulme. and G. Song *Bit-Slice Implementation of PRESENT in pure standard C* , 2011

[17] H. Cohn, R. Kleinberg, B. Szegedyz and C. Umans *Grouptheoretic Algorithms for Matrix Multiplication* In FOCS05, 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 379, 2005

[18] J. Boyar and R. Peralta *A New Combinational Logic Minimization Technique with Applications to Cryptology* In SEA 2010: 178-189, 2009

[19] J. Boyar, P. Matthews and R. Penalta, *On the Shortest Linear Straight-Line Program for Computing Linear Forms* In MFCS, 2008

[20] J. Boyar and R.Peralta *A depth-16 circuit for the AES S-box* http://eprint.iacr.org/2011/332

[21] C. Fuhs and P. Schneider-Kamp *Synthesizing Shortest Linear Straight-Line Programs over GF(2) Using SAT* In SAT 2010, Theory and Applications of Satisfiability Testing, Springer LNCS 6175, pp. 71-84, 2010
Volker Strassen, ,

[22] V. Strassen *Gaussian elimination is not optimal* Numerische Mathematik 13 pp. 354-356, 1969

[23] N. Sorensson and N. Een *Minisat v1. 13-a sat solver with conflict-clause minimization* SAT journal pp. 53, 2005