

Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

**Computations in Prime Fields using Gaussian
Integers**

Examensarbete utfört i Datatransmission
vid Tekniska högskolan i Linköping
av

Adam Engström

LITH-ISY-EX--06/3836--SE

Linköping 2006



Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Computations in Prime Fields using Gaussian Integers

Examensarbete utfört i Datatransmission
vid Tekniska högskolan i Linköping
av


Adam Engström

LITH-ISY-EX--06/3836--SE

Handledare: **Mikael Olofsson**
ISY, Linköpings universitet

Examinator: **Mikael Olofsson**
ISY, Linköpings universitet

Linköping, 19 June, 2006

	Avdelning, Institution Division, Department Division Department of Electrical Engineering Linköpings universitet S-581 83 Linköping, Sweden		Datum Date 2006-06-19
	Språk Language <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English <input type="checkbox"/> _____	Rapporttyp Report category <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	ISBN _____ ISRN LITH-ISY-EX--06/3836--SE Serietitel och serienummer ISSN Title of series, numbering _____
URL för elektronisk version http://www.dtr.isy.liu.se http://www.ep.liu.se/2006/3836			
Titel Title Beräkningar i primtalskroppar med hjälp av gaussiska heltal Computations in Prime Fields using Gaussian Integers			
Författare Adam Engström Author			
Sammanfattning Abstract <p>In this thesis it is investigated if representing a field Z_p, $p \equiv 1 \pmod{4}$ prime, by another field $Z[i]/\langle a + bi \rangle$ over the gaussian integers, with $p = a^2 + b^2$, results in arithmetic architectures using a smaller number of logic gates. Only bit parallell architectures are considered and the programs Espresso and SIS are used for boolean minimization of the architectures. When counting gates only NAND, NOR and inverters are used.</p> <p>Two arithmetic operations are investigated, addition and multiplication. For addition the architecture over $Z[i]/\langle a + bi \rangle$ uses a significantly greater number of gates compared with an architecture over Z_p. For multiplication the architecture using gaussian integers uses a few less gates than the architecture over Z_p for $p = 5$ and for $p = 17$ and only a few more gates when $p = 13$. Only the values 5, 13, 17 have been compared for multiplication. For addition 12 values, ranging from 5 to 525313, have been compared.</p> <p>It is also shown that using a blif model as input architecture to SIS yields much better performance, compared to a truth table architecture, when minimizing.</p>			
Nyckelord Keywords gaussian integers, prime fields, arithmetic, logic minimization			

Abstract

In this thesis it is investigated if representing a field Z_p , $p \equiv 1 \pmod{4}$ prime, by another field $Z[i]/\langle a + bi \rangle$ over the gaussian integers, with $p = a^2 + b^2$, results in arithmetic architectures using a smaller number of logic gates. Only bit parallel architectures are considered and the programs Espresso and SIS are used for boolean minimization of the architectures. When counting gates only NAND, NOR and inverters are used.

Two arithmetic operations are investigated, addition and multiplication. For addition the architecture over $Z[i]/\langle a + bi \rangle$ uses a significantly greater number of gates compared with an architecture over Z_p . For multiplication the architecture using gaussian integers uses a few less gates than the architecture over Z_p for $p = 5$ and for $p = 17$ and only a few more gates when $p = 13$. Only the values 5, 13, 17 have been compared for multiplication. For addition 12 values, ranging from 5 to 525313, have been compared.

It is also shown that using a blif model as input architecture to SIS yields much better performance, compared to a truth table architecture, when minimizing.

Contents

1	Introduction	1
1.1	Background	1
1.2	Problem Definition	1
1.3	Reading Instructions	1
2	Mathematical Background	3
2.1	Groups and Rings	3
2.2	Ring Homomorphisms, Ideals and Fields	5
2.3	Description of $Z_p \cong Z[i]/\langle a + bi \rangle$	8
2.3.1	Representations	10
2.3.2	Generalization	12
3	Logic Circuits and Minimization	15
3.1	Minimization Algorithms	15
3.1.1	Heuristic Algorithms	20
3.2	Complexity Measures	24
3.3	Data Formats	25
3.4	Logic Circuits	27
4	Addition and Multiplication	31
4.1	Bit Representations	31
4.2	Addition in Z_p	32
4.3	Addition in $Z[i]/\langle a + bi \rangle$	32
4.4	Multiplication in Z_p	37
4.5	Multiplication in $Z[i]/\langle a + bi \rangle$	38
5	Conclusions	41
5.1	Addition	41
5.2	Multiplication	45
5.3	Further Research	45
	Bibliography	49

Chapter 1

Introduction

In this chapter the background and definition of the problem investigated in this thesis is described. A reading instruction for the chapters in this thesis is also provided.

1.1 Background

Error correcting codes and cryptography are often used for reliable and secure transmission of data. In both disciplines finite fields may be used. Often the field Z_2 is used since 0 and 1 are easy to represent in digital circuits.

Some problems are best solved in fields Z_p where p is a prime not equal to 2. For example use of the so called residue number system, cryptography using elliptic curves and discrete logarithms [5] and error correcting codes [3].

It is important that the calculations done in finite fields can be performed fast and with little power consumption and that the arithmetic operations can be implemented using as few logic gates as possible.

1.2 Problem Definition

In this thesis it will be investigated if it is possible to use less logic gates in a circuit by representing a finite field Z_p , where p is a prime, as another finite field, $Z[i]/\langle a + bi \rangle$ where $p = a^2 + b^2$ and $a, b \in Z$, over the gaussian integers. The finite field Z_p can only be represented in this way if $p \equiv 1 \pmod{4}$.

In this thesis only two arithmetic operations will be considered, addition and multiplication. All architectures considered will be bit parallel.

1.3 Reading Instructions

Chapter 1 describes the background and problem definition. The necessary mathematics about fields and the construction of the representation used in this thesis

is described in Chapter 2. The heuristic algorithms used for minimization of the architectures in this thesis are described in Chapter 3.

In Chapter 4 architectures for addition and multiplication in both Z_p and $Z[i]/\langle a + bi \rangle$ are presented. Gate counts for these architectures and estimates for the gate count is also presented. Chapter 5 contains a comparison between the two representations and some ideas for further research.

Chapter 2

Mathematical Background

The work in this thesis is built upon the following field isomorphism

$$Z_p \cong Z[i] / \langle a + bi \rangle \quad (2.1)$$

where p is a prime and $a^2 + b^2 = p$. This chapter will describe what is meant by this. For a more thorough description as well as important theorems and their proofs see [8].

2.1 Groups and Rings

In this section the definitions of groups and rings are stated as well as some examples.

Definition 2.1 (Group) A group G is a set A and an operation $*$ which is subject to the following axioms.

1. A must be closed under $*$, that is for all $a, b \in A$, it is the case that also $a * b \in A$.
2. Associativity, that is $(a * b) * c = a * (b * c)$ for all a, b and $c \in A$.
3. Existence of an identity element, that is $e * a = a * e = a$ for some $e \in A$ and for all $a \in A$.
4. Inverses that is for all $a \in A$ there exists an element $b \in A$ such that $a * b = b * a = e$. The inverse of an element a is written a^{-1} .

If additionally the commutative law also holds, that is for all elements a and $b \in A$ it is the case that $a * b = b * a$, the group is said to be Abelian or commutative.

To denote that an element $a \in A$, where A together with an operation $*$ forms a group G , $a \in G$ is used.

By introducing an equivalence relation \equiv the integers, Z , are partitioned in equivalence classes. Let a, b and $n \in Z$, then $a \equiv b \pmod{n} \Leftrightarrow a - b = mn$

for some $m \in Z$. This relation can be shown to be an equivalence relation. It partitions Z into equivalence classes denoted $[a] = \{\dots, -2n + a, -n + a, a, n + a, 2n + a, \dots\}$. In [8, p. 39] the relation is shown to satisfy a number of rules, like the associative and commutative laws for both addition and multiplication, the distributive law and the existence of identity elements, 0 and 1, for addition and multiplication respectively. There exists a multiplicative inverse to every class $[a]$ whenever $(a, n) = 1$.

The set of all equivalence classes mod n is denoted Z_n . This leads to an example of a group.

Example 2.1

The set $A = \{1, 2, 3, 4\}$ together with multiplication modulo 5 forms a group as can be seen from the table below. The associative law holds, the identity element is given by 1 and there exists an inverse for every element (for example $2 \cdot 3 \equiv 1 \pmod{5}$ so 2 and 3 are each others inverses). Since the table is symmetric, the group is also commutative.

\cdot	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

In a group only one operation is defined. The next definition is a ring, which is a structure with two operations.

Definition 2.2 (Ring) A ring R is a set A and two operations, $+$ and \cdot , which are subject to the following axioms.

1. The set A together with the $+$ operation forms an abelian group.
2. For all $a, b \in A$ also $a \cdot b \in A$.
3. For all a, b and $c \in A$ it holds that $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.
4. $a \cdot (b + c) = a \cdot b + a \cdot c$ and $(a + b) \cdot c = a \cdot c + b \cdot c$ for all a, b and $c \in A$

If additionally $a \cdot b = b \cdot a$ for all $a, b \in A$, the ring is said to be commutative. If there exists an element, f , such that $f \cdot a = a \cdot f = a$ for all $a \in A$, the ring is said to have a unit element.

Since the \cdot operation is an analogy of multiplication for ordinary multiplication in for example Z , $a \cdot b$ will hereafter be written ab with the \cdot omitted.

An example of a ring to illustrate the definition.

Example 2.2

The set $A = \{0, 1, 2, 3\}$, together with addition and multiplication modulo 4 forms a ring as can be seen from the tables below. The addition table shows that A together with addition modulo 4 forms an abelian group, where 0 is the identity element. The multiplication table shows that there exists a unit element 1 and that multiplication modulo 4 is commutative. The distribution laws can be verified by calculation. This ring, called Z_4 , is therefore a commutative ring with a unit element.

+	0	1	2	3	·	0	1	2	3
0	0	1	2	3	0	0	0	0	0
1	1	2	3	0	1	0	1	2	3
2	2	3	0	1	2	0	2	0	2
3	3	0	1	2	3	0	3	2	1

In the previous example $2 \cdot 2 = 0$ although $2 \neq 0$. This is an example of a zero divisor.

Definition 2.3 (Zero divisor) R ring, $a \in R$. If $b \cdot a = 0$ for some $b \neq 0$ then a is said to be a zero divisor.

Definition 2.4 (Integral domain) A commutative ring with unit element and no zero divisors is called an integral domain.

An example of an integral domain is Z , where the unit element is 1. The only invertible elements are -1 and 1.

2.2 Ring Homomorphisms, Ideals and Fields

The concepts of homomorphisms, ideals and fields are very important in showing the isomorphism in Equation 2.1.

Definition 2.5 (Ring homomorphism) Let R and S be two rings. A ring homomorphism is a map $\varphi : R \rightarrow S$, with the following properties.

1. $\varphi(a + b) = \varphi(a) + \varphi(b)$ for all $a, b \in R$.
2. $\varphi(ab) = \varphi(a)\varphi(b)$ for all $a, b \in R$.

The mapping $\varphi : Z \rightarrow Z_5$ given by $\varphi(a) = a \pmod{5}$ is an example of a homomorphism. There are several important special cases of homomorphisms.

Definition 2.6 (Surjective) Let $\varphi : R \rightarrow S$ be a homomorphism. It is said to be surjective if for every $a \in S$ there is at least one $b \in R$ such that $\varphi(b) = a$.

Definition 2.7 (Injective) Let $\varphi : R \rightarrow S$ be a homomorphism. Then φ is said to be injective if for all $a, b \in R$ such that $a \neq b \Rightarrow \varphi(a) \neq \varphi(b)$.

Definition 2.8 (Ring isomorphism) If a homomorphism between two rings R and S is injective and surjective it is called an isomorphism. This is written $R \cong S$.

All elements that are mapped to 0 by a homomorphism belongs to a set called the kernel of that homomorphism.

Definition 2.9 (Kernel) Let $\varphi : R \rightarrow S$ be a ring homomorphism. Its kernel is defined as the set $\ker \varphi = \{r \in R : \varphi(r) = 0\}$

Take the same homomorphism as above, that is $\varphi(a) = a \pmod{5}$. Its kernel are all $a \in Z$ such that $a = 5b$ for some $b \in Z$.

Definition 2.10 (Ideal) Let R be a ring and I a subring of R . If both rs and $sr \in I$ for every $r \in R$ and $s \in I$, then I is called an ideal in R .

In Z all ideals can be shown to be expressible in a certain form.

Definition 2.11 (Principal ideal) Let R be a commutative ring with identity and I an ideal. If I is of the form $\langle a \rangle = \{ar : r \in R\}$ then I is said to be a principal ideal.

Definition 2.12 (Principal ideal domain) A principal ideal domain is an integral domain where every ideal is a principal ideal.

Let R be a ring and I an ideal of R . Define a multiplication.

$$(r + I)(s + I) = rs + I$$

And an addition.

$$(r + I) + (s + I) = r + s + I$$

These definitions, together with the fact that I is a subring, give rise to a ring denoted by R/I . The elements of the ring are given by $r + I$ where $r \in R$.

Addition is well defined because $a \in r + I$ and $b \in s + I$ implies that $a = r + c$ and $b = s + d$ for some $c, d \in I$. Then $a + b = r + c + s + d = r + s + c + d$, $r + s \in R$ and $c + d \in I$. Since $r, s \in R$, addition is commutative and inverse to every element exists. The identity element is given by $0 + I = I$.

To see that this multiplication is well defined consider $a \in r + I$ and $b \in s + I$. This implies that $a = r + c$ and $b = s + d$ for some $c, d \in I$. Because $r, s \in R$ then $rs \in R$. Since $c, d \in I$ which is an ideal in R it is the case that $ab = (r + c)(s + d) = rs + rd + cs + cd \in I$. Next the axioms of a ring must be verified.

Let $r + I, s + I$ and $t + I \in R/I$.

$$\begin{aligned} ((r + I)(s + I))(t + I) &= (rs + I)(t + I) = rst + I \\ (r + I)((s + I)(t + I)) &= (r + I)(st + I) = rst + I \end{aligned}$$

Thus the associative law holds. Also the distributive law holds.

$$\begin{aligned}(r + I)(s + I + t + I) &= (r + I)(s + t + I) \\ &= r(s + t) + I = rs + rt + I = rs + I + rt + I \\ (r + I + s + I)(t + I) &= (r + s + I)(t + I) \\ &= (r + s)t + I = rt + st + I = rt + I + st + I\end{aligned}$$

These facts together implies that R/I , where R is a ring and I an ideal, is a ring. This ring is called a quotient ring.

Theorem 2.1 *If I is an ideal in a ring R then the map $\varphi : R \rightarrow R/I$ given by $\varphi(r) = r + I$ is a surjective ring homomorphism and $\ker \varphi = I$.*

Another theorem related to quotient rings is the following.

Theorem 2.2 *R, S are two rings. Let $\varphi : R \rightarrow S$ be a ring homomorphism and set $I = \ker \varphi$. Define $\psi : R \rightarrow R/I$ as $\psi(r) = r + I$. Then there exists an injective homomorphism $\pi : R/I \rightarrow S$ such that $\varphi = \pi\psi$.*

For proofs of both theorems see [8, pp. 254-255]. If φ in the above theorem is surjective then π is an isomorphism.

The next definition states the necessary properties needed for a ring to have a division algorithm and unique factorization.

Definition 2.13 (Euclidean domain) *An integral domain D is said to be a euclidean domain if there exists a function $\nu : D \setminus \{0\} \rightarrow \{a \in \mathbb{Z} : a \geq 0\}$ with the following properties.*

1. $\nu(a) \leq \nu(ab)$ for all $a, b \in D \setminus \{0\}$
2. If $a, b \in D \setminus \{0\}$ then there exists $q, r \in D$ such that $a = bq + r$ where r is either 0 or $\nu(r) < \nu(b)$

In [8, pp. 298-301] it is shown that every euclidean domain is a principal ideal domain and that every principal ideal domain is a unique factorization domain. It is also shown that the ring of gaussian integers $Z[i] = \{a + bi : a, b \in \mathbb{Z}\}$ (where $i = \sqrt{-1}$) is a euclidean domain. The valuation is given by $\nu(a + bi) = a^2 + b^2$. The ring of integers, \mathbb{Z} , is also a euclidean domain with the absolute value as valuation.

If a ring, R , is a euclidean domain it has a division algorithm and the euclidean algorithm can be used to compute the greatest common divisor, (a, b) , of two numbers $a, b \in R$.

Example 2.3

Division in $Z[i]$.

$$\begin{aligned}\frac{5 + i}{3 + 2i} &= \frac{17 - 7i}{13} = 1 + \frac{4}{13} - i + \frac{6i}{13} \Rightarrow \\ 5 + i &= (1 - i)(3 + 2i) + \frac{(4 + 6i)(3 + 2i)}{13} = (1 - i)(3 + 2i) + 2i\end{aligned}$$

Thus $5 + i \equiv 2i \pmod{3 + 2i}$.

Also the greatest common divisor can be calculated. For example to calculate $(5 + i, 3 + 2i)$ euclids algorithm is used.

$$\begin{aligned} 1 \cdot (5 + i) + 0 \cdot (3 + 2i) &= 5 + i \\ 0 \cdot (5 + i) + 1 \cdot (3 + 2i) &= 3 + 2i \end{aligned}$$

As the division suggests.

$$1 \cdot (5 + i) - (1 - i)(3 + 2i) = 2i$$

New division.

$$\frac{3 + 2i}{2i} = \frac{2 - 3i}{2} = 1 - i - \frac{i}{2} \Rightarrow 3 + 2i = (1 - i)2i + 1$$

Then the calculation is almost finished. Because $3 + 2i - (1 - i)2i = 1$.

$$-(1 - i)(5 + i) + (1 + (1 - i)(1 - i))(3 + 2i) = 1$$

Thus $(5 + i, 3 + 2i) = 1$. By the calculation it is also clear that $(3 + 2i)^{-1} \equiv 1 + (1 - i)^2 \equiv 1 - 2i \pmod{5 + i}$. And $(5 + i)^{-1} \equiv (2i)^{-1} \equiv -1 + i \pmod{3 + 2i}$.

The last main concept needed is the concept of a field.

Definition 2.14 (Field) A field is an integral domain, D , where all nonzero $a \in D$ has a multiplicative inverse.

An example of a field is Z_p where p is prime. If p is a prime $(p, 1) = 1, (p, 2) = 1, \dots, (p, p - 1) = 1$. Thus every $a \in Z_p, a \neq 0$ has a multiplicative inverse and hence Z_p is a field.

2.3 Description of $Z_p \cong Z[i] / \langle a + bi \rangle$

Let $a, b \in Z$ such that $(a, b) = 1$. Then there exists m, n such that $am + bn = 1$. This may be written in the following way.

$$\begin{aligned} (am + bn)am + bn &= 1 \Rightarrow \\ (am)^2 + (bn)^2 + abmn + bn - (bn)^2 &= 1 \Rightarrow \\ (a^2 + b^2)m^2 + b(amn + n - bm^2) &= 1 \end{aligned}$$

Thus a multiplicative inverse of $b \pmod{a^2 + b^2}$ exists such that

$$b^{-1} \equiv amn + n - bm^2 \pmod{a^2 + b^2}.$$

Then there is a solution to $x^2 \equiv -1 \pmod{a^2 + b^2}$.

$$\begin{aligned} a^2 + b^2 &\equiv 0 \pmod{a^2 + b^2} \Rightarrow \\ a^2 &\equiv -b^2 \pmod{a^2 + b^2} \Rightarrow \\ (\pm b^{-1}a)^2 &\equiv -1 \pmod{a^2 + b^2} \end{aligned}$$

Define a map $\varphi : Z[i] \rightarrow Z/\langle a^2 + b^2 \rangle$ by

$$\varphi(c + di) = c + (-b^{-1}a)d + \langle a^2 + b^2 \rangle$$

This map is a homomorphism because

$$\begin{aligned} \varphi((c + di)(e + fi)) &= \varphi(ce - df + (cf + de)i) \\ &= ce - df + (cf + de)(-b^{-1}a) + \langle a^2 + b^2 \rangle \\ &= (c + d(-b^{-1}a) + \langle a^2 + b^2 \rangle)(e + f(-b^{-1}a) + \langle a^2 + b^2 \rangle) \end{aligned}$$

and

$$\begin{aligned} \varphi(c + di + e + fi) &= \varphi(c + e + (d + f)i) \\ &= c + e + (-b^{-1}a)(d + f) \\ &= c + (-b^{-1}a)d + \langle a^2 + b^2 \rangle + e + (-b^{-1}a)f \\ &= \varphi(c + di) + \varphi(e + fi). \end{aligned}$$

The homomorphism φ is also surjective because

$$\begin{aligned} \varphi(0) &= 0 + \langle a^2 + b^2 \rangle \\ \varphi(1) &= 1 + \langle a^2 + b^2 \rangle \\ &\vdots \\ \varphi(a^2 + b^2 - 1) &= a^2 + b^2 - 1 + \langle a^2 + b^2 \rangle. \end{aligned}$$

Since

$$\begin{aligned} \varphi(a + bi) &= a + (-b^{-1}a)b + \langle a^2 + b^2 \rangle \\ &= a - a + \langle a^2 + b^2 \rangle = 0 + \langle a^2 + b^2 \rangle \end{aligned}$$

it is the case that $c + di \in \langle a + bi \rangle \Rightarrow c + di \in \ker \varphi$.

Now assume that $c + di \in \ker \varphi$. This means that

$$\begin{aligned} \varphi(c + di) &= 0 + \langle a^2 + b^2 \rangle = c + (-b^{-1}a)d + \langle a^2 + b^2 \rangle \\ bc - ad &= 0 + \langle a^2 + b^2 \rangle = ab^2c - a^2bd = 0 + \langle a^2 + b^2 \rangle \\ ac + bd &= 0 + \langle a^2 + b^2 \rangle \end{aligned}$$

Since

$$\frac{c + di}{a + bi} = \frac{ac + bd + (ad - bc)i}{a^2 + b^2}$$

$c + di \equiv 0 \pmod{a + bi}$ because $ac + bd$ and $ad - bc$ has been shown to have zero residue. Thus $c + di \in \ker \varphi \Rightarrow c + di \in \langle a + bi \rangle$. Then it has been shown that $c + di \in \langle a + bi \rangle \Leftrightarrow c + di \in \ker \varphi$.

Define a homomorphism

$$\psi : Z[i] \rightarrow Z[i]/\langle a + bi \rangle$$

by

$$\psi(c + di) = c + di + \langle a + bi \rangle .$$

According to Theorem 2.1 ψ is surjective. Theorem 2.2 says that there exists an injective homomorphism

$$\pi : Z[i]/\langle a + bi \rangle \rightarrow Z/\langle a^2 + b^2 \rangle$$

such that $\varphi = \pi\psi$. But φ is surjective so then also π must be surjective and thus π is an isomorphism.

Let p be a prime such that $p \equiv 1 \pmod{4}$. Then $p = a^2 + b^2$ for some $a, b \in Z$. For proof see [7, p. 152]. Since p is a prime then Z_p is a field then also $Z/\langle a + bi \rangle$ is a field due to the isomorphism.

2.3.1 Representations

The residue classes in $Z[i]/\langle a + bi \rangle$ can have different representatives. In this thesis two different representations will be used.

The first representation is to use the value with minimum valuation. This is simply achieved by using the division algorithm. Another variant is to represent the residue classes with numbers where both the real part and the imaginary part are positive and with minimal valuation. This representation is given by points in the complex number plane which covers a^2 points nearby the origin and b^2 points next to this square of points, here it is assumed that $a > b$. The geometric shape of this is shown in Figure 2.1 and the numbers are shown in the following table.

$$\begin{array}{llll} (a-1)i & 1+(a-1)i & \dots & a-1+(a-1)i \\ (a-2)i & 1+(a-2)i & \dots & a-1+(a-2)i \\ \vdots & & & \\ bi & 1+bi & \dots & a-1+bi \\ (b-1)i & 1+(b-1)i & \dots & a+b-1+(b-1)i \\ (b-2)i & 1+(b-2)i & \dots & a+b-1+(b-2)i \\ \vdots & & & \\ i & 1+i & \dots & a+b-1+i \\ 0 & 1 & \dots & a+b-1 \end{array}$$

This representation give the complete set of residues. Because as in the table the residue classes, $0, 1, \dots, p-1$, in Z_p can be partitioned as $0, 1, \dots, a+b-1$ and then $a+b, a+b+1, \dots, 2a+2b-1$ and so forth $b-1$ times with the last partition $(b-1)(a+b), (b-1)(a+b)+1, \dots, b(a+b)-1$. The next partition is given by $b(a+b), b(a+b)+1, \dots, b(a+b)+a-1$ and the next $b(a+b)+a, b(a+b)+a+1, \dots, b(a+b)+2a-1$ and so forth $a-b$ times with the last partition given by $b(a+b)+(a-b-1)a = a^2+b^2-a, a^2+b^2-a+1, \dots, a^2+b^2-1$ thus all p residue classes are given in this form.

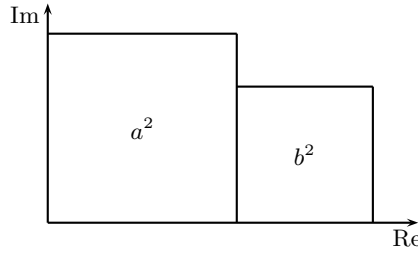


Figure 2.1. Geometry of the positive representation of residue classes in $Z[i]/\langle 2 + i \rangle$

Since p is a prime also a has an inverse in Z_p . Because

$$\begin{aligned}
 a + b &\equiv a - (-ba^{-1})a \\
 &\equiv a - (-b^{-1}a)^{-1}a \\
 &\equiv a - (-b^{-1}a)^3a \\
 &\equiv a - (b^{-1}a)a \\
 &\equiv (-b + a)(-b^{-1}a) \pmod{p}
 \end{aligned}$$

and $b(a + b) + ca \equiv b(a - b - c)(-b^{-1}a) \pmod{p}$ for $c = 1, 2, \dots, a - b - 1$ it is the case that.

1. The first partition is represented as it is via the isomorphism.
2. The next $b - 1$ partitions can be represented as $(a - b)i, (a - b + 1)i, \dots, (a - b + b - 1)i$ via the isomorphism.
3. The other $a - b$ partitions can be represented as $b(a - b)i, b(a - b - 1)i, \dots, b(a - b - a + b + 1)i = bi$.

Thus all residue classes in Z are represented by a residue class in $Z[i]$ in the same way as the table.

To illustrate the ideas, an example of an isomorphism and two of its representations, will be given.

Example 2.4

This example illustrate $Z_5 \cong Z[i]/\langle 2 + i \rangle$. The isomorphism is given by $\varphi(c + di) = c + 3d \pmod{5}$. First addition and multiplication tables for the minimum valuation representation are shown and then the positive real and imaginary parts tables are shown. In the tables it can for example be seen that $2 \equiv -i \pmod{2 + i}$ and that $-1 \equiv 1 + i \pmod{2 + i}$.

+	0	1	2	i	$i+1$	·	0	1	2	i	$i+1$
0	0	1	2	i	$i+1$	0	0	0	0	0	0
1	1	2	i	$i+1$	0	1	0	1	2	i	$i+1$
2	2	i	$i+1$	0	1	2	0	2	$i+1$	1	i
i	i	$i+1$	0	1	2	i	0	i	1	$i+1$	2
$i+1$	$i+1$	0	1	2	i	$i+1$	0	$i+1$	i	2	1

+	0	1	$-i$	i	-1	·	0	1	$-i$	i	-1
0	0	1	$-i$	i	-1	0	0	0	0	0	0
1	1	$-i$	i	-1	0	1	0	1	$-i$	i	-1
$-i$	$-i$	i	-1	0	1	$-i$	0	$-i$	-1	1	i
i	i	-1	0	1	$-i$	i	0	i	1	-1	$-i$
-1	-1	0	1	$-i$	i	-1	0	-1	i	$-i$	1

In Figure 2.2 the minimum valuation representation and the positive real and imaginary part representation are shown.

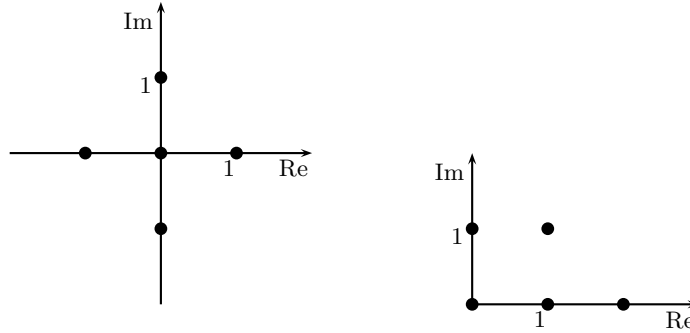


Figure 2.2. On the left minimum valuation representation of residue classes in $Z[i]/\langle 2+i \rangle$. On the right positive representation of residue classes in $Z[i]/\langle 2+i \rangle$.

2.3.2 Generalization

There is a generalization of the above isomorphism. According to [6, Ch. 12] $Z[s]$ is a euclidean domain for both $s = i$ and $s = e^{i\frac{2\pi}{3}}$. If $s = e^{i\frac{2\pi}{3}} = \rho$ the ring $Z[\rho] = \{a + b\rho : a, b \in Z\}$ is called the ring of eisensteinian integers. The valuation is given by $\nu(a + b\rho) = a^2 - ab + b^2$. A prime $p \in Z$ can be written as $a^2 - ab + b^2$ whenever $p \equiv 1 \pmod{3}$.

Let $p \equiv 1 \pmod{3}$ be a prime then $x^3 \equiv 1 \pmod{p}$ has a solution, $r \neq 1$. The solution is given by $r \equiv -b^{-1}a \pmod{p}$. Since $(x-1)(x^2+x+1) = x^3-1$ this is also a solution to $x^2 + x + 1 \equiv 0 \pmod{p}$. Thus $(-b^{-1}a)^2 \equiv -1 - (-b^{-1}a) \pmod{p}$. There is a similar expression for ρ since $\rho^3 = 1$ but $\rho \neq 1$. Then $\rho^2 + \rho + 1 = 0$ and thus $\rho^2 = -1 - \rho$.

Define a mapping

$$\varphi : Z[\rho] \rightarrow Z/\langle a^2 + b^2 - ab \rangle$$

by

$$\varphi(a + b\rho) = a + b(-b^{-1}a) + \langle a^2 + b^2 - ab \rangle.$$

Since

$$\begin{aligned}\varphi(c + d\rho + e + f\rho) &= \\ \varphi(c + e + (d + f)\rho) &= \\ c + e + (d + f)(-b^{-1}a) + \langle a^2 + b^2 - ab \rangle &= \\ c + d(-b^{-1}a) + e + f(-b^{-1}a) &= \varphi(c + d\rho) + \varphi(e + f\rho)\end{aligned}$$

and

$$\begin{aligned}\varphi((c + d\rho)(e + f\rho)) &= \\ \varphi(ce - df + (cf + de - df)\rho) &= \\ ce - df + (cf + de - df)(-b^{-1}a) + \langle a^2 + b^2 - ab \rangle &= \\ \varphi(c + d\rho)\varphi(e + f\rho) &= \\ (c + d(-b^{-1}a))(e + f(-b^{-1}a)) + \langle a^2 + b^2 - ab \rangle &= \\ ce - df + (cf + de - df)(-b^{-1}a) + \langle a^2 + b^2 - ab \rangle, &\end{aligned}$$

φ is a homomorphism.

Since

$$\varphi(a + b\rho) = a + b(-b^{-1}a) + \langle a^2 + b^2 - ab \rangle = 0 + \langle a^2 + b^2 - ab \rangle$$

it is the case that if $c + d\rho \in \langle a + b\rho \rangle \Rightarrow c + d\rho \in \ker \varphi$. Assume $c + d\rho \in \ker \varphi$. Then

$$\begin{aligned}\varphi(c + d\rho) &= c + d(-b^{-1}a) + \langle a^2 + b^2 - ab \rangle = \\ 0 + \langle a^2 + b^2 - ab \rangle &\Rightarrow \\ bc - ad + \langle a^2 + b^2 - ab \rangle &= 0.\end{aligned}$$

Multiplying with ba gives

$$ab^2c - a^2bd + \langle a^2 + b^2 - ab \rangle = 0,$$

then multiplying with b^{-2} gives

$$ac - (b^{-1}a)^2bd + \langle a^2 + b^2 - ab \rangle,$$

since

$$\begin{aligned}(-b^{-1}a)^3 &\equiv 1 \pmod{a^2 + b^2 - ab} \Rightarrow \\ (b^{-1}a)^3 &\equiv -1 \pmod{a^2 + b^2 - ab} \Rightarrow \\ (b^{-1}a)^2 &\equiv b^{-1}a - 1 \pmod{a^2 + b^2 - ab},\end{aligned}$$

and hence $ac - ad + bd + \langle a^2 + b^2 - ab \rangle = 0$. Thus the residue of $c + d\rho$ when divided by $a + b\rho$ is zero (see [6, p. 188]). This means that

$$c + d\rho \in \ker \varphi \Rightarrow c + d\rho \in \langle a + b\rho \rangle$$

Then it has been shown that $c + di \in \langle a + b\rho \rangle \Leftrightarrow c + di \in \ker \varphi$.

As was the case with the isomorphism in the previous section, the conclusion is that $Z[\rho]/\langle a + b\rho \rangle \cong Z_p$ when $p = a^2 + b^2 - ab$. When p is a prime this is a field isomorphism.

Chapter 3

Logic Circuits and Minimization

In the following chapter the general problem of logic minimization and the SIS program will be described. The complexity measures used in this thesis for comparison between different circuits will also be described, as well as the different formats to describe boolean functions and circuits. Some basic logic circuits will be defined finally.

3.1 Minimization Algorithms

A boolean function with n input signals and m output signals is a mapping $f : \{0, 1\}^n \rightarrow \{0, 1, *\}^m$ (* means don't care). This can be seen as m functions f_1, \dots, f_m of n variables each. If there are no don't cares the function is said to be completely specified and if there are don't cares it is incompletely specified.

There are three sets associated with each output function f_i of a boolean function f . First it is the on set, X_i^{on} , which is the set of input signal combinations $x = x_1, \dots, x_n$ such that $f_i(x) = 1$. Similarly the off set, X_i^{off} , is the set of input signal combinations, x , such that $f_i(x) = 0$. Finally the don't care set, X_i^{dc} , is the set of input signal combinations, x , such that $f_i(x) = *$.

An incompletely specified function $f : \{0, 1\}^n \rightarrow \{0, 1, *\}^m$ can be uniquely represented using three completely specified functions, called $r : \{0, 1\}^n \rightarrow \{0, 1\}^m$, $s : \{0, 1\}^n \rightarrow \{0, 1\}^m$ and $t : \{0, 1\}^n \rightarrow \{0, 1\}^m$. For r_i the on set is X_i^{on} and the off set is $X_i^{off} \cup X_i^{dc}$. For s_i the on set is X_i^{off} and the off set is $X_i^{on} \cup X_i^{dc}$. For t_i the on set is X_i^{dc} and the off set is $X_i^{on} \cup X_i^{off}$.

Boolean functions can be specified as a truth table with r rows where each row consists of an input $x \in \{0, 1\}^n$ and an output $y \in \{0, 1, *\}^m$. For a completely specified function the number of rows must be equal to 2^n and $y \in \{0, 1\}$ instead. In [1, p. 19] an algorithm to form an algebraic description of f from a truth table description is given. The algebraic description is in sum of product form. It is given in pseudo code below.

```

for each  $y_i$ 
  for each row
    if  $y_i = 1$  on current row
      product = 1
      for each  $x_j$  on current row
        if  $x_j = 1$ 
          product = product ·  $x_j$ 
        else
          product = product ·  $x'_j$ 
        end if
      end for
       $f_i = f_i + product$ 
    end if
  end for
end for

```

Every uncomplemented or complemented variable is called a literal. Since the equations only include the terms under which the function is 1 all other terms give the value 0 and thus eventual don't care conditions are not visible in this representation.

Example 3.1

x_2	x_1	x_0	y_1	y_0
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	0
1	1	1	0	0

The truth table gives an example of an incompletely specified boolean function with three input signals and two output signals. Its algebraic representation is given by $y_1 = x'_2x_1x_0 + x_2x'_1x'_0$ and $y_0 = x'_2x'_1x'_0 + x'_2x'_1x_0 + x'_2x_1x'_0 + x_2x'_1x'_0$

Definition 3.1 (Cube) Let p be a product term in the algebraic representation of a boolean function f with n input signals and m output signals. Then the cube associated with p is a pair of row vectors of dimensions n and m respectively, such that the i :th entry, $1 \leq i \leq n$, in the first vector is 0, 1 or * depending on if x_i appears in p complemented, uncomplemented or does not appear at all, and the j :th entry, $1 \leq j \leq m$ in the second vector is 0 or 1 depending on if p does not occur in f_j or if it does.

The two vectors in this definition is referred to as the input part and the output part respectively.

Example 3.2

From example 3.1 an example of a cube is.

$$(011), (11)$$

Another example is the following.

$$(0 **), (01)$$

Definition 3.2 (Minterm) A minterm, e^i , associated with a product term e , is a cube where the input part is not * anywhere and where the output part contain 1 on entry i and 0 everywhere else.

The intersection, e , of two cubes c and d is written $e = c \cap d$. The i :th entry of the input part of e is given by 0 if both corresponding entries of c and d are 0 or if one entry is 0 and the other one *. The entry is 1 if both c and d are 1 or if just one of them is 1 and the other *. The entry is given by * if both c and d are *. When one of c and d is 0 but the other one is 1 then e is said to be the empty cube. The i :th entry of the output part of e is given by 1 if both the corresponding entries of c and d are 1. If one of c and d is 0 then e is 0. If the output part of e consists only of zeroes then e is the empty cube.

Example 3.3

The intersection between the cubes given in Example 3.2 is the following.

$$(011), (01)$$

Definition 3.3 (Implicant) Given a boolean function f represented as completely specified functions r , s and t as described on page 15. Then an implicant of f is a cube p which has an empty intersection with the cubes of a representation of s .

A cube c contains another cube d , written $d \subseteq c$ if for every entry in the input part, the entries of c and d are equal or if the entry in c is * then the entry in d can be either 0 or 1. For the output part the entries can either be equal or if a entry is 1 in c it can be 0 in d .

Example 3.4

Taking two cubes from Example 3.1, it is the case that

$$(100), (10) \subseteq (1 * 0), (10).$$

Definition 3.4 (Prime implicant) Let p and q be two implicants such that $p \subseteq q$. If this implies that $p = q$ for every such implicant q , then p is called a prime implicant.

Definition 3.5 (Cover) A cover for a boolean function $f : \{0, 1\}^n \rightarrow \{0, 1, *\}^m$ is a set of cubes where for every output function f_i the set of input parts for f_i includes X_i^{on} but contains no element from X_i^{off} . It may of course contain elements from X_i^{dc} .

Example 3.5

A cover for the boolean function in Example 3.1 is

$$\begin{aligned} &(0 * *), (01) \\ &(011), (10) \\ &(1 * 0), (10) \end{aligned}$$

Definition 3.6 (Prime cover) A cover where all cubes are prime implicants is called a prime cover.

Definition 3.7 (Irredundant cover) A cover, c , for a boolean function f is said to be irredundant if there is no $d \subset c$ such that d is also a cover for f .

The minimization problem is now solved by generating all prime implicants and all minterms and arrange them in a matrix, A , called the prime implicant matrix. Each implicant corresponds to a column and each minterm corresponds to a row in A . An element, a_{ij} , in the matrix is 1 if the i :th minterm is covered by the j :th implicant and 0 otherwise. Then the minimized function is given by the following expression where x is a binary vector with dimension equal to the number of implicants and $\mathbf{1}$ is also a vector with the dimension equal to the number of minterms and with 1's in every position.

$$Ax \geq \mathbf{1} \tag{3.1}$$

When this expression is minimized then each minterm is covered by at least one prime implicant. Which implies that f is minimized. The boolean expression is deduced by just taking each prime implicant belonging to the cover and write it out as an expression.

Example 3.6

In the previous example the r , s and t sets are given by the following tables.

r	x_2	x_1	x_0	y_1	y_0
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	0

s	x_2	x_1	x_0	y_1	y_0
0	0	0	0	1	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	1	1

t	x_2	x_1	x_0	y_1	y_0
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	0	0

The implicants are the following.

$(0 **), (01)$
 $(011), (11)$
 $(01*), (01)$
 $(10*), (10)$
 $(101), (11)$
 $(110), (11)$

The minterms.

(000), (01)
 (001), (01)
 (010), (01)
 (011), (01)
 (011), (10)
 (100), (10)

The prime implicant matrix is the following.

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

For example $a_{64} = 1$ since the implicant (10^*) , (11) contains the minterm (100) , (10) . By inspection of A it is seen that the first implicant must be included in the cover since the two first minterms are only covered by that implicant. Also the second implicant must be included since the fifth minterm is only covered by this one. Finally the fourth implicant must be included since the sixth minterm is only covered by that implicant. In matrix form this can be seen to fulfill inequality 3.1.

$$A \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 1 \\ 1 \end{pmatrix}$$

Thus the first, second and fourth (from left) prime implicants are sufficient to cover the boolean function. Hence by just converting the included implicants to algebraic form the minimized boolean function is $y_1 = x'_2x_1x_0 + x_2x'_1$ and $y_0 = x'_2$.

3.1.1 Heuristic Algorithms

Since the problem of minimizing boolean functions is NP-complete [1, p. 8] the time to solve an instance of the problem grows exponentially with the number of input signals. Therefore heuristic methods are used instead. They are methods that in polynomial time finds a solution that may be far away from the minimal solution but performs reasonably well in practice.

Espresso

The first of the heuristic algorithms used in this thesis is Espresso [1]. Espresso takes as input the on set and the don't care set or the off set of a boolean function

f . Then it tries to minimize three metrics, the number of product terms in the cover, the number of literals and the number of outputs. Finally it returns the minimized cover. The basic algorithm in espresso is a loop which iterates until none of the three metrics has decreased since the last iteration. Espresso uses a number of algorithms called expand, essential primes, irredundant cover, reduce and last gasp in its main loop.

In what follows a short description of each of algorithms will be given. For more details see [1, Ch. 4] and [9, pp. 304-318].

- Expand

An expansion of a cube p is a new cube q where some 0 in the input part has been changed to a 1. Thus q contains p but it also covers more cubes than p . That is $p \subset q$. In expanding a cube it is important that the new cube still is a cube. Thus every expansion requires an intersection test with the off set of the boolean function in question.

Given a cover c of the boolean function f , expand produces a new cover, g which is a prime cover. This is done by, for every cube $p \in c$ expand p so that it covers as many other cubes as possible and subsequently delete these other cubes. Thus every cube will be a prime implicant and therefore g will be a prime cover. The prime cover also has the property that $a \not\subseteq b$ for all cubes $a, b \in g$ such that $a \neq b$. A cover with this property is said to be minimal with respect to single cube containment.

Expand is a heuristic algorithm, and the number of cubes, contained in g , depends on in which order the cubes of c are expanded.

- Essential primes

A prime implicant, p , is called an essential prime implicant if there is a minterm which is only contained in p and no other cube covering f . All essential prime implicants must be contained in a cover of f . Thus they need not be processed in the other algorithms. Hopefully saving some memory and computation time.

- Irredundant cover

Given a cover c which is the output from the expand algorithm, the irredundant cover algorithm produces a cover g which is irredundant and where $g \subseteq c$.

The irredundant cover g should also contain as few cubes as possible. Therefore the cubes in c are partitioned into two classes. One class where removing one cube results in a set which is not a cover. These cubes are called relatively essential. The second class consists of the other cubes, called redundant cubes. The redundant cubes can be partitioned further into totally redundant and partially redundant cubes. The totally redundant cubes can be removed since they are covered by the relatively essential cubes. Finally a subset is chosen from the partially redundant cubes such that a cover of f is obtained. This cover should contain as few cubes as possible but the selection is done heuristically.

- Reduce

Given a cube p a reduction of that cube is a new cube p' which is contained in p . That is $p' \subset p$.

Given a prime cover c of a boolean function f , reduce produces a new cover, g which is not necessarily a prime cover by reducing each cube $p \in c$.

The purpose of running reduce is, if possible, to find a smaller cover than the one already found. Thus moving from one cover, that may or may not be the optimal cover, to a cover from which another perhaps better cover can be found by using expand.

- Last gasp

Last gasp is run as a final step when the cover size of f is no longer decreasing when running reduce, expand and irredundant cover. The algorithm use modified versions of expand and reduce to be able to minimize f further.

SIS

The second of the heuristic algorithms used in this thesis is SIS. Its predecessor MIS is described in [2] and SIS is described in [11]. The first paper describes many of the algorithms used both in SIS and MIS while the second paper is more of a user guide to SIS. SIS is an interactive system in which you can run several different algorithms and also read and write different circuit and logic description formats, do technology mapping with different circuit libraries and simulate the behavior of circuits. SIS includes the espresso program and uses it for minimization and uses other commands to do common subexpression elimination (missing in espresso) and do other operations related to logic simplification like removing redundancy and do substitutions.

Many of the algorithms in SIS use an algebraic model where some of the properties of a boolean algebra are removed. A boolean function is considered as an ordinary polynomial in these algorithms. To illustrate this restriction consider $(a + b)(a + d)$, which would equal $aa + ad + ab + bd$ as an algebraic expression but as a boolean expression it would equal $a + bd$. Such a restricted representation makes it possible to run different search algorithms in a reasonable time and yet with an acceptable result. An example of the searches done is searching for common subexpressions. See [2] and [9, Ch. 8] for more information.

In SIS a boolean function f is represented as a boolean network. A boolean network is a directed acyclic graph where each node corresponds to a boolean expression or variable. There is an edge from a node u to a node v if the expression associated with u occurs in the expression associated with v .

Scripts with certain combinations of commands are included in SIS to be used for minimization. In this thesis the script *rugged* is used. The contents of *rugged* is the following.

1. sweep; eliminate -1
2. simplify -m nocomp

3. eliminate -1
4. sweep; eliminate 5
5. simplify -m nocomp
6. resub -a
7. fx
8. resub -a; sweep
9. eliminate -1; sweep
10. full_simplify -m nocomp

Each of the algorithms above are described in the following.

- Sweep

Eliminates all constant nodes (no incoming edges) and all nodes that do not have any effect on the output signal.

- Eliminate

When a node in a boolean network is used in more than one place, i.e. it has more than one outgoing edge, it saves literals in the network. Eliminate removes nodes which do not save more literals than a threshold. When removing a node its boolean expression replaces the occurrence of the node in other nodes to which this node is connected by an outgoing edge. The threshold value is given as a parameter to the algorithm. For example the eliminate command on line 4 in *rugged* will eliminate all nodes which do not save more than 5 literals. The call on line 1, where the threshold parameter is -1, means that all nodes are eliminated. Thus the boolean function is written in its sum of products form. No common subexpressions are reused.

- Simplify

Run the espresso algorithm on each node in the network. Simplify does not provide the whole don't care set as input to espresso.

- Resub

Let f, g be two expressions seen as algebraic expressions. If q is the largest set of cubes such that $f = qg + r$ then q is called the quotient and r the remainder of f/g .

The algorithm resub performs resubstitution of every node in the network by using an algebraic model of the boolean expressions. This is done by running an algorithm called algebraic divide where the q and r values are found. By the resub algorithm new nodes are created containing such common subexpressions.

- `fx`

The `fx` algorithm is similar to the resubstitution algorithm. `Resub` finds common nodes in a network by dividing the expression in one node by expressions in other nodes. `Fx` finds expressions common to one or two cubes, such cubes are divided by a common divisor.

- `full_simplify`

Almost the same algorithm as `simplify` but it uses the don't care set as well.

Technology mapping is the process of transforming a logic expression into hardware. Hardware is described on gate level in different library files. The technology mapping algorithm tries to minimize both area and delay. For technology mapping in this thesis the gate library called *minimal* is used. Minimal consists of only three gates: NAND, NOR and inverter. Information about every gate is given in the genlib format described in [11]. Each gate has information about the area occupied by a gate, the time it takes for the output signal to change when a input signal has changed, the maximal output load driven and the load of the input pins. The values used in this particular gate library can be seen in Table 3.1. According to [11] the area given is a relative cost measure and the rise and fall times are given in nanoseconds.

Gate	Area	Load		Rise time		Fall time	
		Input	Output	Input	Output	Input	Output
Inverter	1	1	999	0.9	0.3	0.9	0.3
NAND	2	1	999	1.0	0.2	1.0	0.2
NOR	2	1	999	1.0	0.3	1.0	0.2

Table 3.1. Technology parameters in gate library *minimal*.

The `map` command is used to do the actual mapping. The result of the mapping may be seen by the `print_gates -s` command. Different technology mapping algorithms are described in [9, Ch. 10].

Simulating the behavior of a circuit in SIS is done by the `simulate` command. The input signal values are specified by the user and the output signals are calculated and printed on the screen.

3.2 Complexity Measures

To compare different circuits and different representations of the elements of a prime field as described in chapter 2 a complexity measure is introduced. In this thesis the gate count, as reported by SIS, is used as a complexity measure.

3.3 Data Formats

Circuits can be described in many different ways. For example as boolean equations, truth tables or in the blif format.

An example of a truth table.

Example 3.7

The following truth table describes a full adder. The columns in order is input bit one, input bit two, input carry, output sum and finally output carry. The *type* command is used for telling SIS that both the on set and off set are given.

```
.i 3
.o 2
.type fr
000|00
001|10
010|10
011|01
100|10
101|01
110|01
111|11
```

A circuit is described in a format called Berkeley Logic Interchange Format (blif), see [11] for a full description. A model is described as a simple text format which describes the structure of a logic circuit by models and one model can be used in another. Each model has output and input signals. It is possible to connect models to each other by specifying how the input and output signals are connected. Each signal can have a symbolic name and intermediate signals can be used. A specification of the conditions for a 1 output, for each output signal, is given as a truth table. It is also possible to specify the don't care conditions as a separate truth table.

Example 3.8

A blif model describing a three bit adder (given in Figure 3.1) where no values greater than 5 are expected as input.

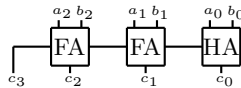


Figure 3.1. Three bit adder circuit.

```
.model adder
.inputs a2 a1 a0 b2 b1 b0
.outputs c3 c2 c1 c0
```

Here the name of the circuit model is given and also the input and output signals of the circuit.

```
.subckt half_adder p=a0 q=b0 c=carry0 r=c0
.subckt full_adder p=a1 q=b1 ci=carry0 c=carry1 r=c1
.subckt full_adder p=a2 q=b2 ci=carry1 c=c3 r=c2
```

The second part describes the structure of the circuit. How it is composed of one half adder connected to a full adder which is connected to another full adder. The input and output signals to and from the sub circuits are specified and whenever a name is not an input or output signal it is an intermediate signal. For example carry0 is an intermediate signal connecting the carry out signal from the half adder to the carry in signal at the first full adder.

```
.exdc
.names a2 a1 a0 c0
110 1
111 1
.names a2 a1 a0 c1
110 1
111 1
.names a2 a1 a0 c2
110 1
111 1
.names a2 a1 a0 c3
110 1
111 1
.exdc
.names b2 b1 b0 c0
110 1
111 1
.names b2 b1 b0 c1
110 1
111 1
.names b2 b1 b0 c2
110 1
111 1
.names b2 b1 b0 c3
110 1
111 1
.end
```

In this part the don't care set is given. Since no values greater than 5 are expected, these values as input give a don't care output. In the .names construction

only the last is an output signal, when specified as 1 inside a .exdc construction it means don't care. The .end command marks the end of a model specification.

```
.model half_adder
.inputs p q
.outputs r c
.names p q r
01 1
10 1
.names p q c
11 1
.end

.model full_adder
.inputs p q ci
.outputs r c
.names p q ci r
001 1
010 1
100 1
111 1
.names p q ci c
011 1
101 1
110 1
111 1
.end
```

In the last part the sub circuits used earlier are specified. Here by the use of truth tables.

3.4 Logic Circuits

Graphical representations of the basic logic gates, NAND, NOR and inverter, are shown in Figure 3.2. Frequently, more advanced circuits such as full and half adders, full and half subtracters will be used. They are defined by truth tables in Table 3.2. Their graphical symbols are shown in Figure 3.3.

Multiplexers will also be used in the architectures, its graphical symbol is shown in Figure 3.4.

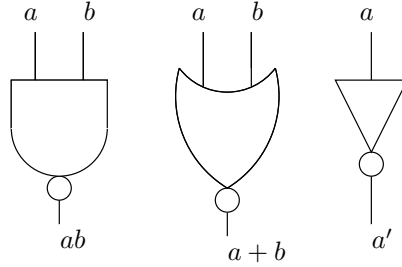


Figure 3.2. AND gate, OR gate and inverter

a	b	c_{in}	s	c_{out}	a	b	s	c_{out}
0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0
0	1	0	1	0	1	0	1	0
0	1	1	0	1	1	1	0	1
1	0	0	1	0				
1	0	1	0	1				
1	1	0	0	1				
1	1	1	1	1				

a	b	$borrow_{in}$	d	$borrow_{out}$	a	b	d	$borrow_{out}$
0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	1	1	1
0	1	0	1	1	1	0	1	0
0	1	1	0	1	1	1	0	0
1	0	0	1	0				
1	0	1	0	0				
1	1	0	0	0				
1	1	1	1	1				

Table 3.2. Truth tables defining from left to right, full and half adder, full and half subtracter.

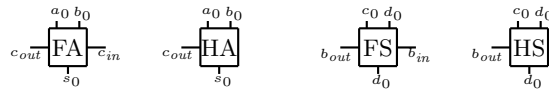


Figure 3.3. Basic circuits for addition and subtraction.



Figure 3.4. A multiplexer.

Chapter 4

Addition and Multiplication

In this chapter architectures for the two arithmetic operations, addition and multiplication in Z_p and $Z[i]/\langle a + bi \rangle$ will be given.

Gate counts of the different architectures will be given both as diagrams and tables. The results have been calculated using SIS, following the steps below.

1. Load an architecture.
2. Run *full_simplify* which minimizes the architecture using a variant of the espresso algorithm.
3. Run *source script.rugged*.
4. Load a gate library with only inverters and nand and nor gates. Using the command *read_library minimal.genlib*.
5. Run the technology mapping with *map*.
6. Count the gates with *print_gate -s*.

The gate count for the unminimized circuits are calculated by omitting step 2 and 3 in the list above.

Throughout the chapter it is assumed that the input signals are always reduced mod p and mod $a + bi$ before entering the circuit.

The estimates given for some of the architectures, use AND, OR and inverters as gate count.

4.1 Bit Representations

There are different ways of representing the elements of a field using binary values. In this thesis the normal binary representation is used. Thus in Z_p , each residue $0, \dots, p - 1$ is represented as a number in base 2. For example 5 is represented by 101. The complex numbers of $Z[i]/\langle a + bi \rangle$ are represented as two components, the real part and the imaginary part, each using base 2. For example $3+2i$ is

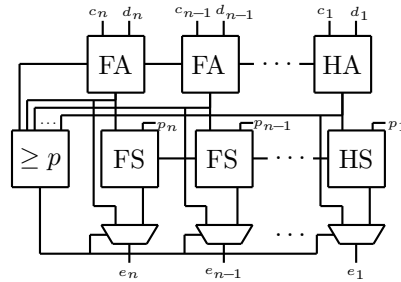


Figure 4.1. Architecture for addition in Z_p

represented as 11, 10. Negative numbers are represented using two complement representation see [4, p. 11]. For example -5 is represented as 011 using 3 bits and as 1011 using 4 bits.

4.2 Addition in Z_p

When adding two numbers $c, d \in Z_p$ the result $c + d \leq 2p - 2$. Therefore it is sufficient to test if $c + d \geq p$ and in that case subtract p . The resulting number e will always be less than p . In figure 4.1 this architecture is shown as a circuit diagram.

Let n be the number of bits needed to represent the p residue classes then, $n = \lceil \log_2 p \rceil$. One full adder and one two-input multiplexer, with 15 and 4 gates respectively, is needed per bit. Then in worst case a full subtractor with 4 gates per bit. Finally there should be a compare circuit, this takes approximately n gates. Thus an estimate on the number of gates in this architecture is given by $24n$.

When minimizing this architecture in SIS there are two alternative methods of representing the circuit.

1. Build a truth table.
2. Build a blif model.

The first alternative has not the structure of the architecture above while the second follow the architecture. Minimization with SIS for the two different approaches show very different results in gate count as shown in Figure 4.2. There are also great differences in running time of the minimization, alternative 1 is much more time consuming. Numerical results are shown in table 4.1.

4.3 Addition in $Z[i] / \langle a + bi \rangle$

Different representations of the residue classes in $Z[i] / \langle a + bi \rangle$ give rise to different architectures. When $b = a - 1$ (as for example $5 = 4 + 1$ and $13 = 16 + 9$) an

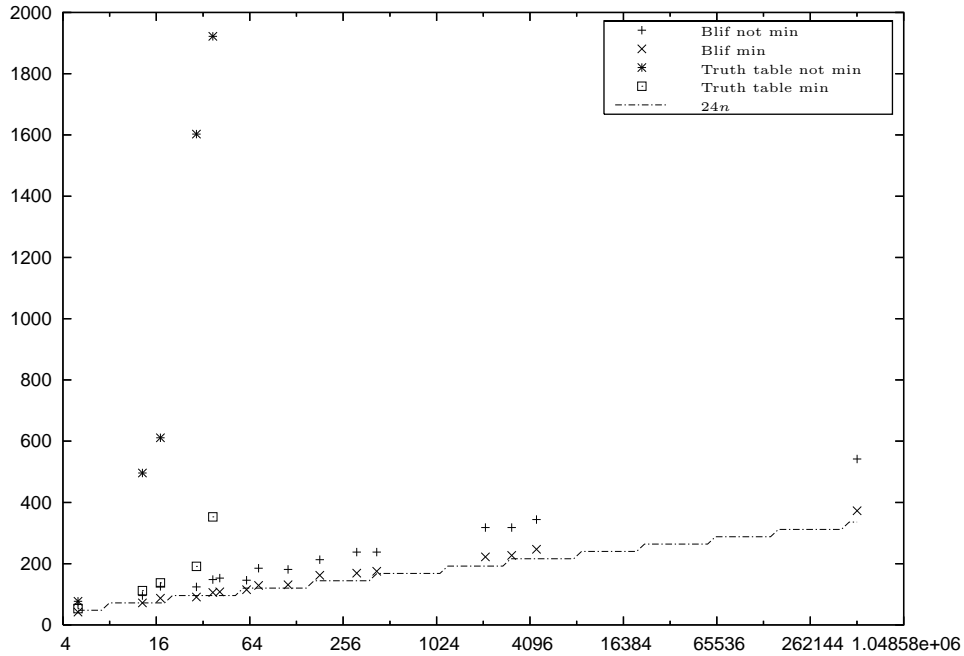


Figure 4.2. Gate count for addition in Z_p . The p values are on the x-axis and the gate count is on the y-axis.

p	Gate count			
	Minimized	Not minimized	Minimized	Not minimized
5	42	68	53	77
13	72	96	112	496
17	87	126	137	611
29	91	124	191	1603
37	106	148	353	1922
41	108	153		
61	115	146		
73	129	185		
113	131	181		
181	162	213		
313	169	238		
421	175	238		
2113	222	318		
3121	227	318		
4513	247	344		
525313	373	542		

Table 4.1. Gate counts for different architectures for addition in Z_p . On the left is data for blif models and on the right truth tables has been used.

architecture with three reductions can be devised when using the positive representation. Adding two numbers in this representation will give the elements shown below.

$$\begin{array}{ccccccc}
 (2a-2)i & \dots & 2a-2 & + & (2a-2)i & & \\
 (2a-3)i & \dots & 3a-3 & + & (2a-3)i & & \\
 \vdots & & & & & & \\
 (a+1)i & \dots & 3a-3 & + & (a+1)i & & \\
 \mathbf{ai} & \dots & \mathbf{3a-3} & + & \mathbf{ai} & & \\
 (a-1)i & \dots & a-1 & + & (a-1)i & \dots & \mathbf{4a-4} & + & (a-1)i \\
 (a-2)i & \dots & 2a-2 & + & (a-2)i & \dots & \vdots & & \\
 \vdots & & & & & & & & \\
 i & \dots & 2a-2 & + & i & \dots & & & \\
 0 & \dots & 2a-2 & & & \dots & \mathbf{4a-4} & &
 \end{array}$$

The elements in bold face are elements that must be reduced to the original representation. Since

$$\begin{aligned}
 2a-1 &\equiv 2a-1 - (a + (a-1)i) + (a + (a-1)i)i \\
 &\equiv 2a-1 - 2a+1 + i \\
 &\equiv i \pmod{a + (a-1)i}
 \end{aligned}$$

the first reduction is given by adding $1-2a+i$ to all elements, x , with $\operatorname{Re} x \geq 2a-1$. No new elements that needs to be reduced are added because, there are no element with an imaginary part greater than $2a-3$, and a real part greater or equal to $2a-1$. Thus adding i yields an imaginary part which is at most $2a-2$. Since there are no element with real part greater than $3a-3$ being reduced, this gives a real part after reduction less or equal to $2a-3$. Which is less than $2a-2$ in the upper right corner. This means that all elements left has a real part less or equal to $2a-2$.

The next reduction is to subtract $a + (a-1)i$ from all elements x such that $\operatorname{Re} x \geq a$ and $\operatorname{Im} x \geq a-1$. Since $\operatorname{Re} x \leq 2a-2$ and $\operatorname{Im} x \leq 2a-2$, the element with greatest valuation reduced will be reduced to $a-2 + (a-1)i$. This is inside the residue set because, $a-2 < a-1$ and $a-1 = a-1$. The element with least valuation reduced will be reduced to 0.

Now the only elements left that must be reduced are elements, x , such that $\operatorname{Re} x \geq a-1$ and $\operatorname{Im} x \geq a$. Since all elements x where $\operatorname{Im} x \geq a$ already have a real part less or equal to $a-1$, the first condition is unnecessary. By adding $a-1-ai$ these elements are reduced to elements, x , such that $a-1 \leq \operatorname{Re} x \leq 2a-2$ and $0 \leq \operatorname{Im} x \leq a-2$. This is due to that no element have imaginary part greater than $2a-2$ and the real part of the elements reduced were less than a . It has been shown that all elements, $x+y$, where x, y are in the residue set, are reduced to elements in the residue set by these three reductions.

Example 4.1

Here follows an example of the above reasoning for the field $Z[i]/\langle 3+2i \rangle$.

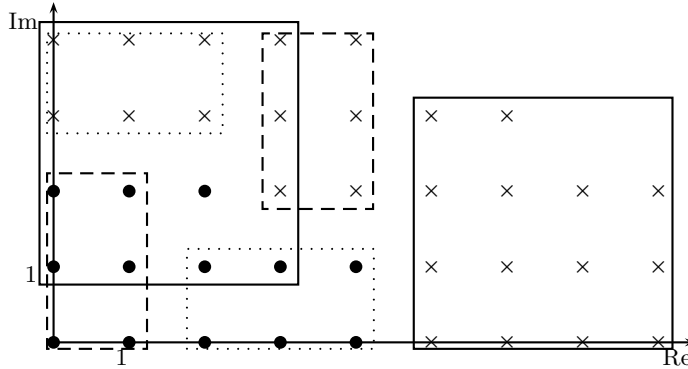


Figure 4.3. Positive real and imaginary part representation of residue classes in $Z[i]/\langle 3 + 2i \rangle$.

In Figure 4.3 the three reductions are illustrated. The dots represent the gaussian integers used to represent the residue classes in $Z[i]/\langle 3 + 2i \rangle$ and the crosses represents the gaussian integers that must be reduced when adding two of the numbers represented as dots. The first reduction is indicated by a rectangle with solid lines. The second reduction is indicated by a rectangle with dashed lines. Finally the third reduction is indicated by a rectangle with dotted lines.

An architecture using the above reductions is shown in Figure 4.4. It has almost the same structure as the circuit for addition in Z_p . Addition of i in the first reduction, is implemented by letting the first adder of the complex addition, be a full adder. With its carry set to the first compare of the added real part.

Let n_{re} , be the number of bits needed to represent the real part and n_{im} , be the number of bits needed to represent the imaginary part. Given by $n_{re} = \lceil \log_2(2a - 1) \rceil$ and $n_{im} = \lceil \log_2 a \rceil$ respectively. Then there are $n_{re} + n_{im}$ full adders with 15 gates in the architecture. In the first reduction step there are n_{re} full subtracters and multiplexers, 8 gates all together. In the second reduction step there are $n_{re} + n_{im} + 1$ full subtracters, where one input is constant, and multiplexers. In the third reduction step there are n_{re} full adders with one constant input, 7 gates, and multiplexers. There are also $n_{im} + 1$ full subtracters with one constant input and multiplexers, 8 gates in total. Finally there are four compare circuits, two with n_{re} gates and two with $n_{im} + 1$ gates. There is also one AND gate in the second reduction step. Thus an estimate on the number of gates required is $33(n_{re} + n_{im}) + 7n_{re} + 19$.

Minimization is done in the same way, using SIS, as for addition in Z_p . Two representations of the circuit are used, truth table and blif model. Gate count before and after minimization, for the blif model, is shown in Figure 4.5 together with the estimate for the architecture in Figure 4.4.

Blif models has only been developed for the case when $p = a^2 + (a - 1)^2$ (for example 5, 13 and 41). In Table 4.2 numerical data is shown. The differences in

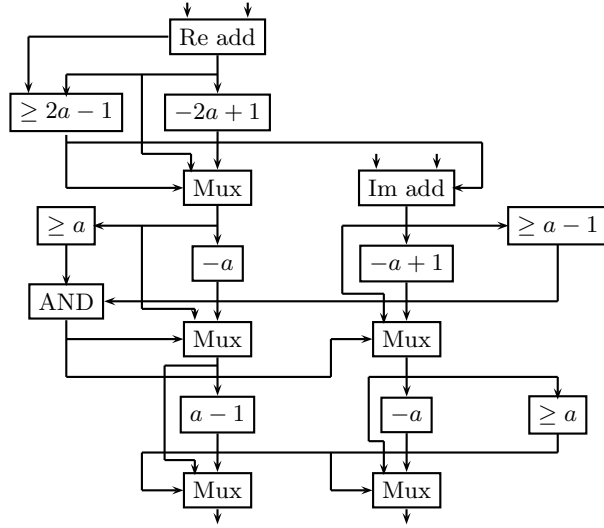


Figure 4.4. Architecture for addition in $Z[i]/\langle a + bi \rangle$

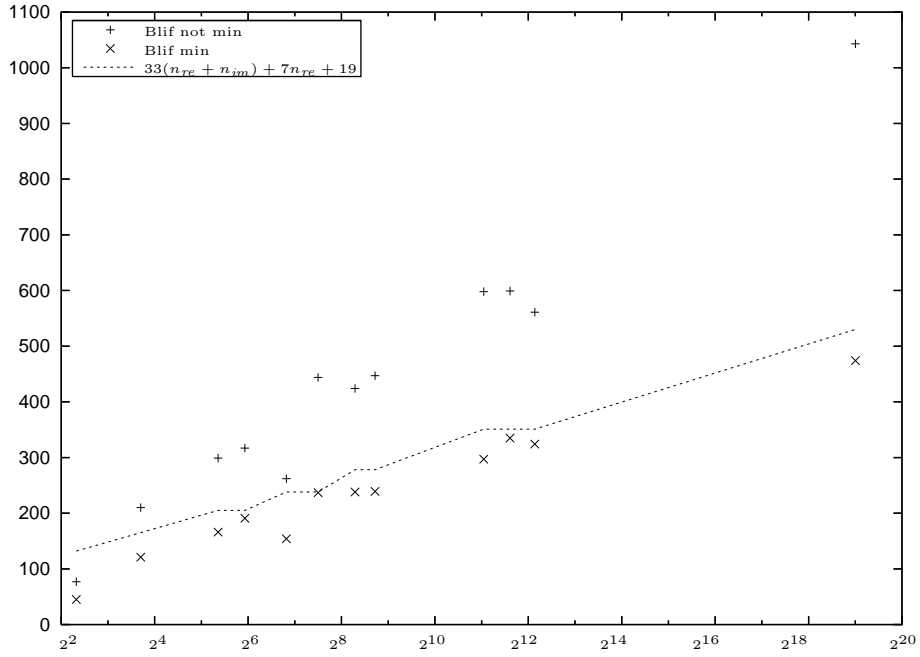


Figure 4.5. Gate count for addition in $Z[i]/\langle a + (a - 1)i \rangle$. The p values are on the x-axis and the gate count is on the y-axis.

p	$a + bi$	Gate count			
		Minimized	Not minimized	Minimized	Not minimized
5	2+i	45	77	53	77
13	3+2i	121	210	257	638
17	4+i			307	895
41	5+4i	166	299		
61	6+5i	191	317		
113	8+7i	154	262		
181	10+9i	237	444		
313	13+12i	238	424		
421	15+14i	239	447		
2113	33+32i	297	598		
3121	40+39i	335	599		
4513	48+47i	324	561		
525313	513+512i	474	1043		

Table 4.2. Gate counts for different architectures for addition in $Z[i]/\langle a + bi \rangle$. On the left is data for blif models and on the right truth tables has been used.

number of gates in the minimized architecture, between the two representations of the circuit, are very large.

4.4 Multiplication in Z_p

Architectures for multiplication in Z are given in [10, p. 589-591], one of these architectures is called the array multiplier. The array multiplier uses the same method as the algorithm used when multiplying by hand. There exists versions both for unsigned and signed numbers. The proposed architecture for multiplication in Z_p is based on the array multiplier but with some reduction steps added. The array multiplier is shown in Figure 4.6.

A multiplication of two numbers $x, y \in Z_p$ can be written as

$$xy = \sum_{i=0}^{n-1} 2^i \sum_{j=0}^{n-1} x_i y_j.$$

Every partial product

$$\sum_{j=0}^{n-1} x_i y_j$$

is less than or equal to $p - 1$. Thus, the following holds

$$xy \leq (2^{n-1} + 2^{n-2} + \dots + 2^1 + 1)(p - 1).$$

Then by subtracting $2^i p$ for $0 \leq i \leq n - 1$ from xy when $xy \geq 2^i p$, the product is reduced to a number in the complete residue set $0 \leq xy \leq p - 1$.

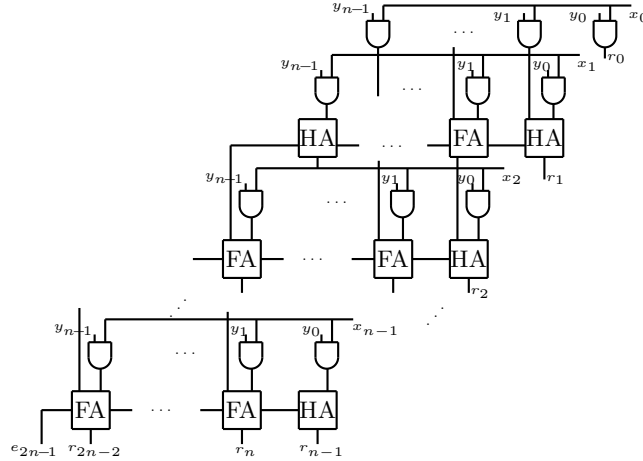


Figure 4.6. Architecture for the array multiplier.

The architecture consists of an array multiplier and n reduction steps with subtractors, multiplexers and compare circuits. An n bit array multiplier has n^2 AND gates for partial product generation. For addition of partial products it has $n(n-1)$ adders with 15 gates in each. The n reduction steps has $2n$ subtractors, with one constant input, and multiplexers. In total 8 gates. Finally there are $n-1$ compare circuits with $2n$ gates in each. Thus an estimation on the number of gates is given by $34n^2 - 17n$.

In Figure 4.7 the gate count from the blif model both before and after minimization using SIS is shown. There are large differences between the blif model and the truth table also in this case. In Table 4.3, numerical data is shown.

4.5 Multiplication in $Z[i]/\langle a + bi \rangle$

For multiplication, no general architecture for any class of primes, has been found. Architectures have only been developed for each of the three special cases, 5, 13 and 17. All three implementations have been done for the least valuation representation. Negative numbers have been represented as bit arrays using 2-complement representation. For the case of $p = 5$ the truth table representation had the smallest gate count. While a blif model was appropriate for $p = 13$ and $p = 17$.

Since a complex multiplication can be written as

$$(c + di)(e + fi) = ce - df + (cf + de)i,$$

the blif architecture implements four ordinary real valued signed multiplications. Some multiplications result in numbers already in the residue set but some multiplications need to be reduced. This is done by comparing each case and reducing

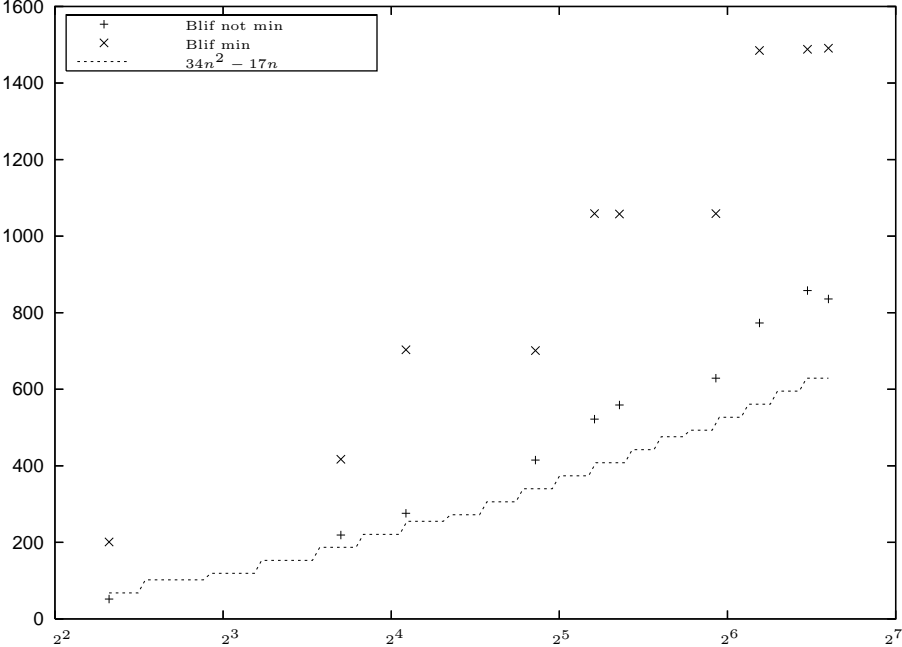


Figure 4.7. Gate count for multiplication in Z_p . The p values are on the x-axis and the gate count is on the y-axis.

p	Gate count			
	Minimized	Not minimized	Minimized	Not minimized
5	52	201	36	63
13	219	417	296	450
17	276	703	542	867
29	415	701	1682	2529
37	522	1059		
41	559	1058		
61	629	1059		
73	773	1485		
89	858	1488		
97	836	1491		

Table 4.3. Gate counts for different architectures for multiplication in Z_p . On the left is data for blif models and on the right truth tables has been used.

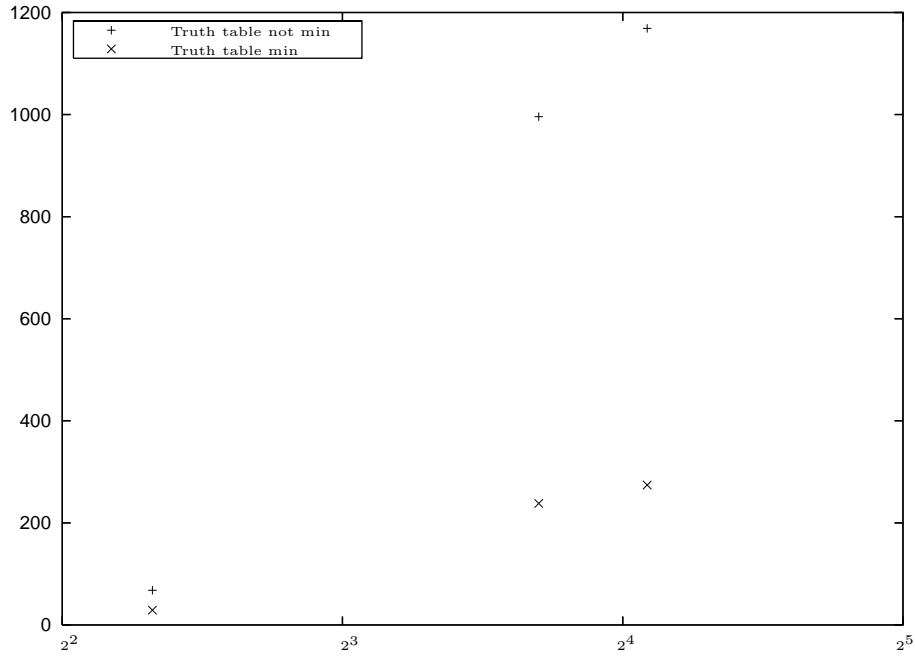


Figure 4.8. Gate count for multiplication in $Z[i]/\langle a + bi \rangle$. The p values are on the x-axis and the gate count is on the y-axis.

it individually for $p = 13$ and in clusters of four values when $p = 17$. For $p = 13$ there were 8 reductions and for $p = 17$ there were 5 reductions.

The gate counts are shown in figure 4.8 and in table 4.4. Both before and after minimization.

p	$a + bi$	Gate count			
		Minimized	Not minimized	Minimized	Not minimized
5	$2+i$			29	68
13	$3+2i$	238	996	335	687
17	$4+i$	274	1169	551	948

Table 4.4. Gate counts for different architectures for multiplication in $Z[i]/\langle a + bi \rangle$. On the left is data for blif models and on the right truth tables has been used.

Chapter 5

Conclusions

In the following chapter the conclusions of this thesis and some ideas for further work are described.

It has been shown that the representation of elements in Z_p as elements in $Z[i]/\langle a + bi \rangle$ results in hardware with a gate count comparable to the gate count in Z_p . It has also been shown that boolean function minimization depends to a great extent on the structure of the input boolean function.

5.1 Addition

For addition the Z_p representation is generally better. In Table 5.1 the gate count (both before and after minimization) for both representations are displayed along with the difference between the two representations. The differences are also shown in Figures 5.1 and 5.2.

For $p = 5$ and $p = 113$ the two representations have the smallest difference. For 5 even the truth table architecture is not too far away from the blif model architecture. Probably the $p = 5$ case is small enough to be minimized well in both cases. It can also be noted that the difference between the two representations are less in the blif model architecture than when using a truth table. For example when $p = 13$ the difference is 145 in the truth table architecture but only 49 using a blif model. When considering the not minimized architectures this difference is not so large, 114 and 142.

The difference between blif model and truth table is shown in Table 5.2. It can be seen that using a structured architecture yields a great difference in minimized gate count. For example it can be seen that the unminimized blif model architecture has less gates than the minimized truth table architecture for all cases but $p = 5$.

p	Gate count					
	Minimized			Not minimized		
	Z_p	$Z[i]/\langle a + bi \rangle$	Difference	Z_p	$Z[i]/\langle a + bi \rangle$	Difference
5	42	45	3	68	77	9
13	72	121	49	96	210	114
17	87			126		
29	91			124		
37	106			148		
41	108	166	58	153	299	146
61	115	191	76	146	317	171
113	131	154	23	181	262	81
181	162	237	75	213	444	231
313	169	238	69	238	424	186
421	175	239	64	238	447	209
2113	222	297	75	318	598	280
3121	227	335	108	318	599	281
4513	247	324	77	344	561	217
525313	373	474	101	542	1043	501

p	Gate count					
	Minimized			Not minimized		
	Z_p	$Z[i]/\langle a + bi \rangle$	Difference	Z_p	$Z[i]/\langle a + bi \rangle$	Difference
5	53	53	0	77	77	0
13	112	257	145	496	638	142
17	137	307	170	611	895	284
29	191			1603		
37	353			1922		

Table 5.1. Comparison for addition, upper table shows blif model and lower table shows truth table architecture.

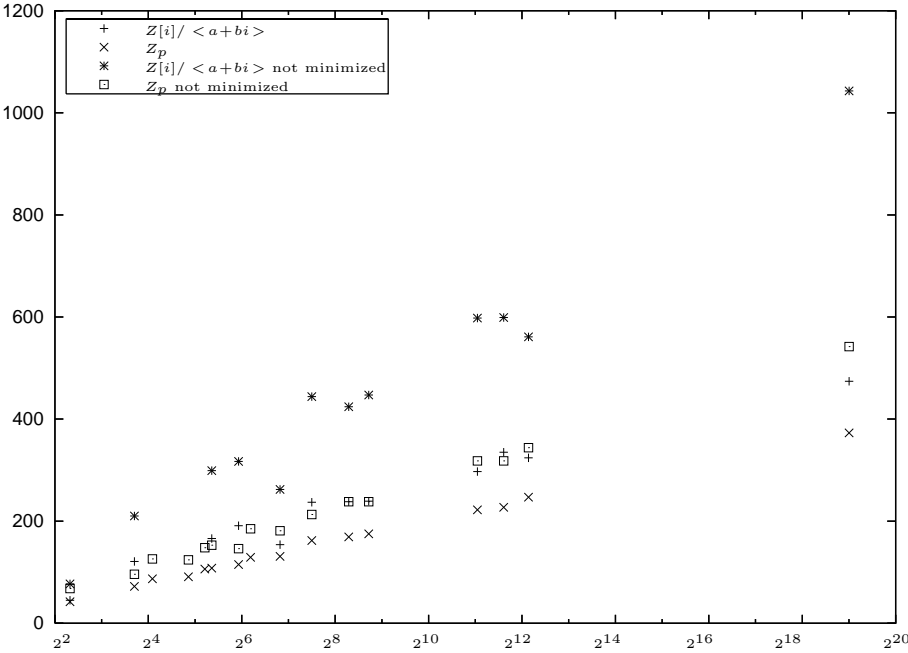


Figure 5.1. Blif architectures for addition. The p values are on the x-axis and the gate count is on the y-axis.

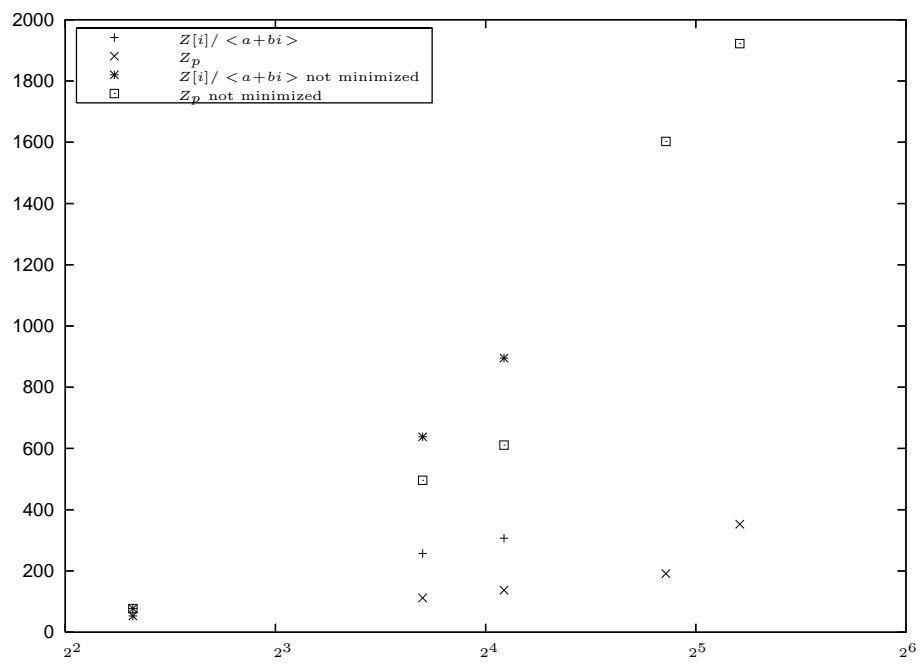


Figure 5.2. Truth table architectures for addition. The p values are on the x-axis and the gate count is on the y-axis.

5.2 Multiplication

The least valuation representation of the residue classes in $Z[i]/\langle a + bi \rangle$ use less gates than the corresponding architecture for Z_p for $p = 5$ and $p = 17$. Since only the cases 5, 13 and 17 have been investigated more cases need to be done before any conclusion about which representation is the best can be made.

In [5] different multiplication architectures for Z_p based on different encodings of the bit vectors representing the residue classes are investigated. Their architectures have gate counts similar to the truth table architectures in this thesis. For example in the case $p = 17$ their architecture give a gate count of 542, in case of $p = 13$ their architecture give a gate count of 301.

For integers on the form $m = 2^n + 1$, a special architecture for multiplication in Z_m is given in [12]. This architecture has 184 gates for the case $p = 17 = 2^4 + 1$ compared to 274 gates in this thesis. For the case $p = 5 = 2^2 + 1$ the architecture in [12] has 60 gates compared to 29 gates in this thesis.

In Table 5.3 all the data is summarized and in Figures 5.3 and 5.4 the differences are illustrated.

5.3 Further Research

Many aspects of the work in this thesis can be further investigated and better understood. Here some examples on interesting questions are listed.

The isomorphism between Z_p and $Z[i]/\langle a + bi \rangle$ for p prime such that $p \equiv 1 \pmod{4}$ is only one of many isomorphisms between Z_p and other fields. Another example mentioned in Chapter 2 is the isomorphism $Z_p \cong Z[\rho]/\langle a + b\rho \rangle$ for p prime such that $p \equiv 1 \pmod{3}$. It would be interesting to investigate architectures based on this and other isomorphisms.

Only normal binary representation and two complement has been used in this thesis to represent the residue classes in $Z[i]/\langle a + bi \rangle$. It is possible that other binary representations give better architectures.

Residue classes may be represented in many ways, here only two different complete residue systems have been used. It is possible that other complete residue systems yields better architectures.

Minimization with the algorithms in SIS show very data dependent behavior.

p	Gate count					
	Blif	Truth table	Difference	Blif	Truth table	Difference
5	42	53	11	68	77	9
13	72	112	40	96	496	400
17	87	137	50	126	611	485
29	91	191	100	124	1603	1479
37	106	353	247	148	1922	1774

Table 5.2. Difference between, minimized to the left and unminimized to the right, blif model and truth table.

p	Gate count					
	Minimized			Not minimized		
	Z_p	$Z[i]/\langle a+bi \rangle$	Difference	Z_p	$Z[i]/\langle a+bi \rangle$	Difference
5	52			201		
13	219	238	19	417	996	579
17	276	274	-2	703	1169	466

p	Gate count					
	Minimized			Not minimized		
	Z_p	$Z[i]/\langle a+bi \rangle$	Difference	Z_p	$Z[i]/\langle a+bi \rangle$	Difference
5	36	29	-7	63	68	5
13	296	335	39	450	687	237
17	542	551	9	867	948	81

Table 5.3. Comparison for multiplication, upper table shows blif model and lower table shows truth table architecture.

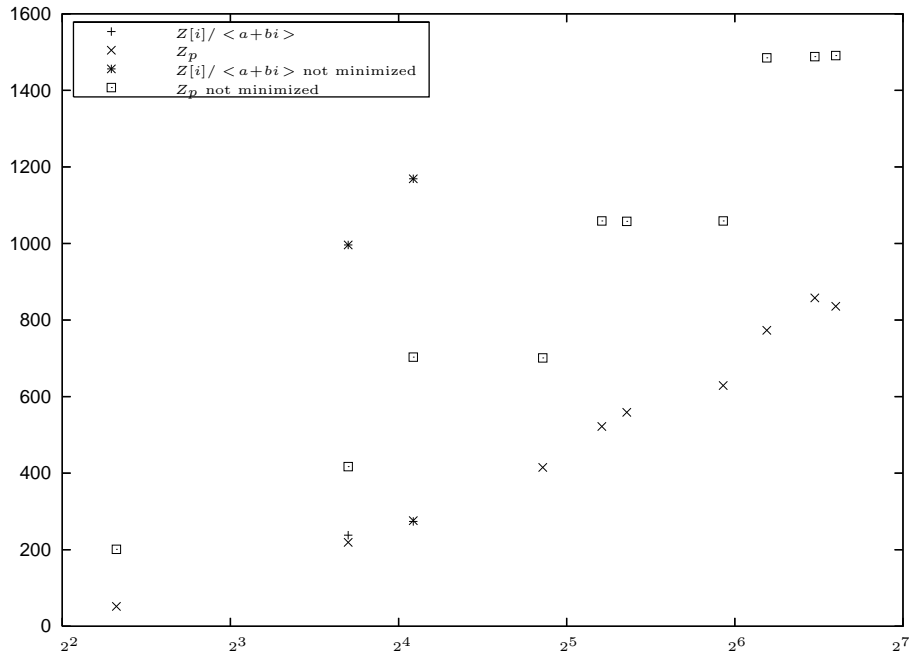


Figure 5.3. Blif architectures for multiplication. The p values are on the x-axis and the gate count is on the y-axis.

It would be interesting to see exactly in what way it is better to use a blif model than a truth table, and also how a model should be constructed to best utilize the minimization algorithms.

There are other minimization tools than espresso and SIS. Use of other algorithms more suited for arithmetic circuits may give better results. Also new use of the algorithms in SIS could be investigated. In this thesis only standard script has been used and tested, maybe using other algorithms in different order give better results.

For multiplication no general structure could be found. If a structure could be found more data on gate count could be generated to see if multiplication really has lower gate count in $Z[i]/\langle a + bi \rangle$.

Also for addition only a subset of the primes were covered by the architecture proposed. There can be similar or better architectures covering all interesting primes.

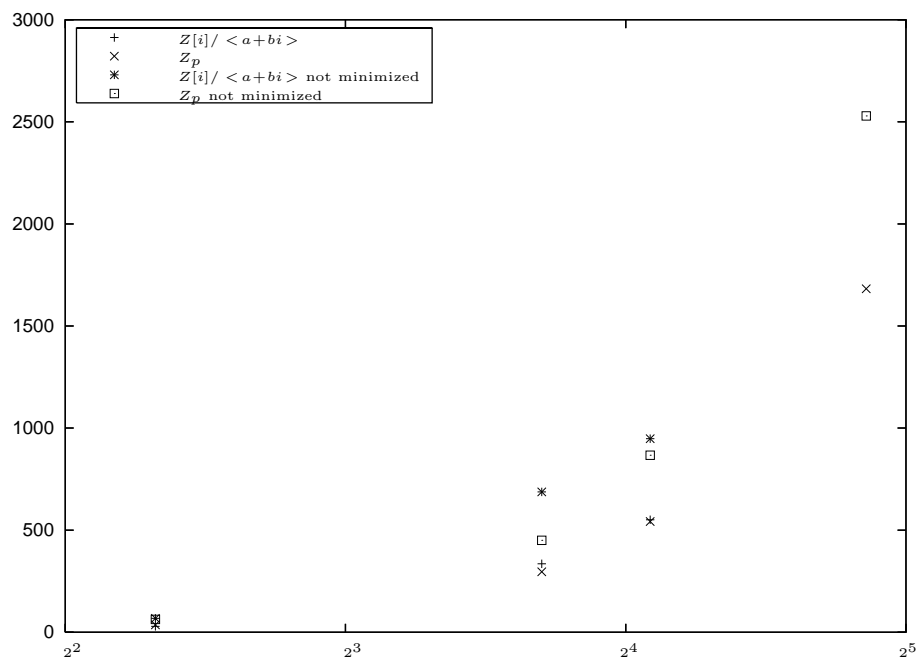


Figure 5.4. Truth table architectures for multiplication. The p values are on the x-axis and the gate count is on the y-axis.

Bibliography

- [1] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. Sangiovanni-Vincentelli. *Logic minimization algorithms for VLSI synthesis*. Kluwer Academic Publishers, Boston, 1984.
- [2] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. Mis: A multiple-level logic optimization system. *IEEE Transactions on computer aided design*, 1987.
- [3] M. de Noronha-Neto and B.-F. Uchoa-Filho. Space-time trellis codes over $GF(p)$ for p -PSK modulation. In *Communications, 2004 IEEE International Conference on*, 2004.
- [4] M. D. Ercegovac and T. Lang. *Digital arithmetic*. Morgan Kaufman, San Francisco, 2003.
- [5] J. Guajardo, T. Wollinger, and C. Paar. Area efficient $GF(p)$ architectures for $GF(p^m)$ multipliers. In *Circuits and Systems, 2002. MWSCAS-2002. The 2002 45th Midwest Symposium on*, 2002.
- [6] G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Clarendon, Oxford, 5 edition, 1979.
- [7] I. N. Herstein. *Topics in algebra*. John Wiley, New York, 2 edition, 1975.
- [8] T. W. Judson. *Abstract algebra: theory and applications*. PWS Publishing company, Boston, 1994.
- [9] G. De Micheli. *Synthesis and optimization of digital circuits*. McGraw-Hill, New York, 1994.
- [10] J. M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital integrated circuits : a design perspective*. Prentice Hall, Upper Saddle River, N.J., 2 edition, 2003.
- [11] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. Sis: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, EECS Department, University of California, Berkeley, 1992.

- [12] Leonel Sousa and Ricardo Chaves. A universal architecture for designing efficient modulo 2^n+1 multipliers. *IEEE Transactions on circuits and systems*, 2005.

Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>