The ART behind IDEAS

Thomas Beth, Andreas Klappenecker,[*] Torsten Minkwitz,[**] and Armin Nückel

Institut für Algorithmen und Kognitive Systeme
Universität Karlsruhe, Fakultät für Informatik
D-76128 Karlsruhe, Germany
e-mail: EISS_Office@ira.uka.de

Abstract. The IDEAS design system is an innovative algorithm engineering environment. It can be employed to derive correct and efficient software or hardware implementations from algebraic specifications using the advanced formal method ART. Several case studies for some algorithm engineering tasks are discussed in this article to demonstrate the feasibility of this system.

1 Introduction

IDEAS is the acronym (*I*ntelligent *D*esign *E*nvironment for *A*lgorithms and *S*ystems) for an innovative algorithm engineering tool that has been built during a long term research project at the authors' institution, incorporating modern techniques of informatics, mathematics, computer algebra, and engineering. In contrast to traditional CAE-tools, like software compilers or hardware description languages (HDLs) or so-called Silicon Compilers and synthesis tools, in IDEAS the problem specification is stated in terms of mathematical expressions, which are the natural application descriptions by the user. Subsequently, using the methodology of *A*bstract *R*epresentation *T*heory (ART), a type refinement is carried out to take into account the requirements of a given implementation technology according to the semantics of the application. The basic idea is to generate and optimize implementations by well-chosen mathematical operations in relation to the algebraic problem specification. This leads to an alternative approach to algorithm engineering. In several case studies the feasibility of such an intelligent design environment, the strength of the approach and the power of the system IDEAS is demonstarated.

1.1 The IDEAS Design Flow

What is the basic difference between the traditional top-down or bottom-up approach and the sketched IDEAS design flow presented here? Following the

[*] This research was supported by DFG under project Be 887/6-2.
[**] This research was supported by DFG through the Graduiertenkolleg "Beherrschbarkeit komplexer Systeme".

IDEAS approach, the terms *problem, specification, algorithm,* and *implementation* become quite well-defined algebraic notions. Since user defined specifications are done using traditional mathematical data structures, such as number domains, vector spaces, matrix algebras, polynomial rings, permutation groups and ordered sets, modelling the data structure means specializing the ADT by introducing operational semantics in the specificated domain itself. Obviously, the specification may be changed during the design process. At first glance, this seems improper, because the specification is describing the problem. The unusual idea, however, is to introduce operations on the specification itself, i.e. combining it with the process of modeling ab initio. The engineering tool is used to generate algorithms, which are correct with respect to the specification in the semantics of the given model. Optimization then becomes a rewriting problem of the specification. Summarizing, one can quote several advantages of the algebraic algorithm design approach:

- availability of deep mathematical knowledge,
- correctness by automatic algebraic transformations,
- efficiency by modularization and parallelization,
- reusability through ADT's

The design environment IDEAS is to our knowledge the first fully operational implementation (cf. Fig. 1) of such a system. The strength of the environment relying on the fact that both the knowledge representation and the transformations on the specifications are provided by modern Computer Algebra Systems. The implementation of the features of IDEAS, making it an intelligent design environment, are briefly sketched in the following sections. They have become possible due to the increased capabilities of the underlying Computer Algebra packages during the last decade.

1.2 The Principle of ART

The theoretical background of this approach, ART (Abstract Representation Theory) is based on the thought that problems should be specified in the most adequate data structure. Elementary datatypes that have been studied thoroughly in the past are boolean logic and finite state automata. For problem specifications of this type, quite efficient tools exist to optimize the specification before the actual implementation. However, if a problem is best and most naturally stated in terms of linear equations, it doesn't seem to make sense to first refine down to the level of boolean logic and then try to optimize at this level, since important information is lost.

The development of the theory was initiated by the observation that algorithm engineering for restricted problem classes in signal processing is based on a limited number of data types and design principles [1]. It became clear that there is a strong relationship between symmetries and fast implementation of a system matrix A [20, 19].

The *cyclic shift property* of the Discrete Fourier Transform (DFT) can be described in these terms of symmetries. If P denotes the cyclic shift, which e.g.

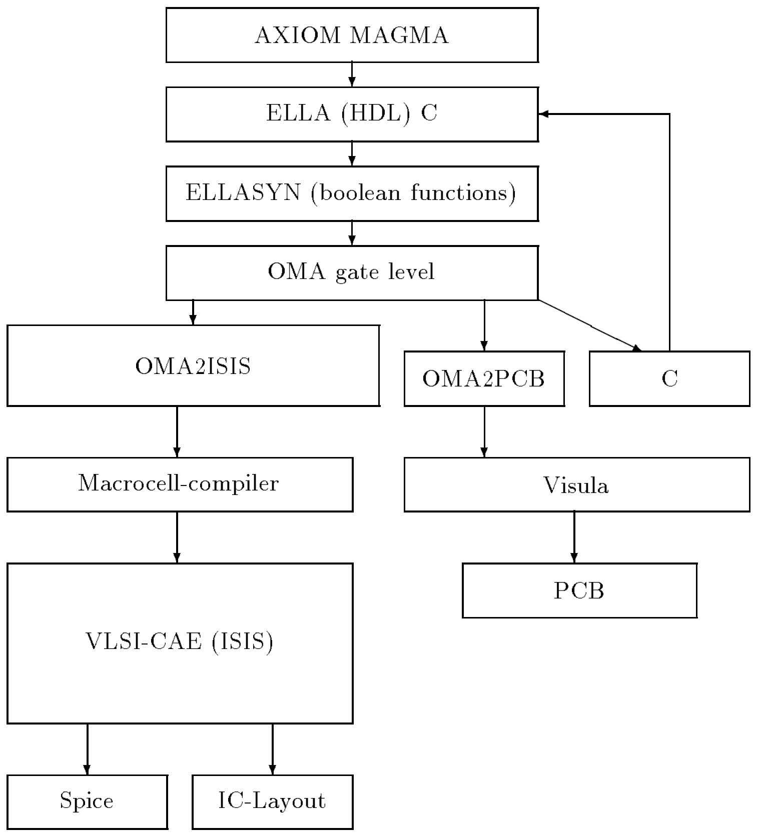


Fig. 1. The structure of IDEAS

naturally occurs as the phase shift operator in many models of specification (cf. section 4), and F denotes the Fourier tranfom matrix, then the following equation holds:

$$FP = QF,$$

where Q denotes the diagonal matrix $\mathrm{diag}(1, \omega, \omega^2, \ldots, \omega^{n-1})$. This property, which has many applications in position invariant pattern analysis, is an example of a *symmetry,* which offers itself to be reformulated in terms of representation theory.

Recall that for any finite group G, a linear representation of G in a vector space V is a homomorphism ρ from the group G into the group of invertible matrices of automorphisms $GL(V)$ of V. A *symmetry* of a matrix A is defined by a pair of representations ρ, σ with:

$$A\rho(g) = \sigma(g)A \qquad \forall g \in G,$$

where ρ is a permutation representation and σ is fully decomposed (thus of block diagonal form). If A is invertible, then $A\rho(g)A^{-1} = \sigma(g)$, and it is called a *decomposition matrix* of the permutation representation.

If a matrix has a symmetry, then the transform has obviously strong structural properties, which can be used to derive fast algorithms by modularization (block diagonal structure) or parallelization as a consequence of an induced tensor structure. The objective is the computation of a matrix describing a change of base, such that A can be rewritten as product of factors in a sparse form. As we shall show below, representation theory is a powerful tool to construct such suitable base change matrices. In order to explain the construction mechanism, some basic notions from representation theory are needed.

Recall that a representation $\rho : G \to GL(V)$ is called *irreducible,* if V is non-zero and if no vector subspace of V is stable under G, except 0 and V. Here, a vector space is called *stable,* if it is invariant under the action of G. For two irreducible representations $\rho : G \to GL(V)$ and $\sigma : G \to GL(W)$ Schur's Lemma [22] states in particular that a linear mapping $f : V \to W$ intertwining these representations, i.e. satisfying $\rho \circ f = f \circ \sigma$, is zero, if σ and ρ are not isomorphic, or otherwise is a scalar multiple of the identity.

Now if T_ρ is any decomposition matrix of a permutation representation, then an extension of Schur's Lemma shows that there is a matrix E with $ET_\rho = A$, such that E is block diagonal (up to a permutation) with blocks corresponding to homogeneous components of ρ. A homogeneous component of a representation ρ consists of all irreducible subrepresentation of ρ from one isomorphism class of irreducible representations of the group G. This enforces modularization and tensor product factorization. With a constructive method to compute a factorized decomposition matrix T_ρ with sparse factors at hand [20], using the equation $ET_\rho = A$, IDEAS constructs a fast algorithm to multiply with A. To give an impression of the notions used in this constructive process, the next section will provide a specific case study.

2 Case Study: Fast Cosine Transform

Here, in an application, we demonstrate that we have a method to produce a factorization of the transform matrix into sparse factors as an improved implementation from a specification given by the linear transform and a symmetry. A detailed explanation would require a thorough understanding of representation theory.

2.1 Preliminaries.

The Discrete Cosine Transform (DCT) is well known from signal processing applications. Especially, this transform on eight points is at the heart of the well-known JPEG/MPEG compression family, the construction of a fast algorithm to carry out the DCT is an important task. An IDEAS generated solution improving the straight forward implementations is given in this case study. The DCT is given by:

$$x(n) = C_o(n)\sqrt{\frac{2}{N}}\sum_{k=0}^{N-1} C(k)\cos\left(\frac{k(n+1/2)\pi}{N}\right), \qquad n = 0,\ldots,N-1,$$

where $C_o(n) = 1/\sqrt{2}$, if $n = 0$, and 1 otherwise.

The DCT matrix of size 8×8 turns out to be a decomposition matrix of the dihedral group D_8 on 8 points (thus it has a symmetry), which can be written as a permutation group in the following form:

$$\mathrm{D}_8 \cong \langle (1,3,5,7,8,6,4,2), (1,2)(3,4)(5,6)(7,8) \rangle,$$

resembling the invariance of the DCT with respect to 2-step reflected shifts and local swapping. As mentioned before, another decomposition matrix for the group D_8 can be constructed that differs only by a block diagonal matrix E as another factor. A factorized base change matrix is constructed using a recursive procedure along a chain of normal subgroups (cf.[2, 19]).

2.2 A Fast Discrete Cosine Transform Algorithm

The dihedral group in its usual permutation group presentation leads naturally to a permutation representation. In representation theory a key notion is that of an induced representation. Roughly speaking, the induction operation extends a 'small' representation of a subgroup to a 'bigger' representation of the whole group. The dimension of the representation vector space increases by a factor of $|\,\text{group}\,|/|\,\text{subgroup}\,|$, producing a block structure of this order. Permutation representations are always induced from representations of smaller groups, namely from the 1-representation of the stabilizer. More formally, this can be stated as follows:

$$\langle (1,3,5,7,8,6,4,2), (1,2)(3,4)(5,6)(7,8) \rangle \cong \mathrm{Ind}_{\mathtt{Stab}(D_8,1)}^{D_8}(1).$$

First Step. The dihedral group D_8 contains a normal subgroup D_4 of prime index that contains the stabilizer of D_8 :

$$D_4 \cong \langle (3,2)(5,4)(7,6), (1,5)(7,2)(8,4) \rangle$$

Now a procedure called *induction recursion* can be applied, which is based on the identity:

$$\mathrm{Ind}_{\mathtt{Stab}(D_8,1)}^{D_8}(1) \cong \mathrm{Ind}_{D_4}^{D_8}\left(\mathrm{Ind}_{\mathtt{Stab}(D_8,1)}^{D_4}(1)\right).$$

This means that the induction operator can be factored, which leads to a factorization of the decomposition matrix. There are always four factors, typically of the form:

$$T = (T_1 \otimes 1)B(1 \otimes T_2)P,$$

where T_1 is a decomposition matrix of the factor group D_8/D_4, B is a block diagonal matrix with images of irreducible representations of the group D_8 as blocks, T_2 is a decomposition matrix of $D_4 \cong \langle(1,2,3,4),(1,4)(2,3)\rangle$, and P is simply a permutation.

Since the factor group $\mathbb{Z}_2 \cong D_8/D_4$ is abelian, the decomposition matrix is simply given by the Fourier matrix (cf. section 3.1)

$$T_1 := \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

The matrix T_2 is determined by recursion, and the matrix B is given by a solution of a system of linear equations.

Second Step. The dihedral group D_4 contains $\mathbb{Z}_2 \times \mathbb{Z}_2 \cong \langle(2,4),(1,3)\rangle$ as a normal subgroup. Again, induction recursion can be applied. This time the identity

$$\mathrm{Ind}_{\mathtt{Stab}(D_4,1)}^{D_4}(1) \cong \mathrm{Ind}_{\mathbb{Z}_2\times\mathbb{Z}_2}^{D_4}\left(\mathrm{Ind}_{\mathtt{Stab}(D_4,1)}^{\mathbb{Z}_2\times\mathbb{Z}_2}(1)\right)$$

is used, leading again to a factorization of the form

$$T_2 = (T_1' \otimes 1)B'(1 \otimes T_2')P',$$

where the matrices are of size 4×4. The matrix T_2' is a decomposition matrix for $\mathbb{Z}_2 \times \mathbb{Z}_2$ with permutation representation $\langle(1,2)\rangle$ and is determined via recursion. The matrices T_1', B', and P' are like T_1, B, and P respectively execpt of smaller degree (cf. the similar approach in section 5).

Last Step. Since permutation groups of degree 2 always have the Fourier matrix DFT_2 as a decomposition matrix, this one is taken from a library. In the end, one has obtained a factorized decompostion matrix T. A fast algorithm to multiply with DCT_8 follows immediately. To learn about the details of computing B and B' consult [20].

2.3 VLSI Implementation

The previous section gave an impression of the "internal life" of IDEAS. The user from the outside is simply requested to provide a matrix together with its symmetry and gets a fast algorithm as a result of the ART approach. For the DCT_8 the system automatically produced an algorithm which uses 14 multiplications instead of 64. In view of the regularity, this compares preferably in gate count and chip size with the 13 multiplications of handoptimized algorithms [12], which economizes on multiplications at the cost of additions. The algorithm is stated as a list of matrices in a computer algebra system. The IDEAS system translates this representation to the high-level Hardware Description Language ELLA. Using the LOCAM synthesis tools a gate level description can be generated. The step from the gate level description to the geometry is done with the help of the VLSI-CAE system ISIS (cf. Fig. 1). Sequential programs (C-syntax) are generated by the use of ordinary software compilers for this purpose.

3 Case Study: Algebraic Discrete Fourier Transform

In this case study techniques for compiling and optimizing Algebraic Discrete Fourier Transforms (ADFTs) [1, 4] are dicussed. ADFT is based on the so-called Modular Polynomial Transformations (MPTs), providing a common generalization of the Diskrete Fourier Transform and Chinese Remainder Technique [8].

Both transformations are of high demand for high speed applications in coding theory [17, 6], cryptography [15, 13] and signal processing [1, 7].

3.1 Preliminaries

Denote by $a = (a_0, \ldots, a_{n-1})$ an input vector over a field $\mathbb{F}$. It will be convenient to repeat this vector as a signal polynomial, namely as $a(x) = \sum_{i=0}^{n-1} a_i x^i$. Assume that the characteristic of $\mathbb{F}$ does not divide the signal length n. The polynomial $x^n - 1$ can be decomposed in disjoint, irreducible, monic factors

$$x^n - 1 = \prod_{i=1}^{m} p_i(x) \ \in \mathbb{F}[x].$$

The *Modular Polynomial Transform* (MPT) to the factors $(p_i)_{i=1}^m$ of a signal polynomial $a(x)$ is defined as the set of polynomial remainders

$$MPT_{(\mathbb{F}, x^n-1)}(a(x)) = (B_1(x), \ldots, B_m(x)),$$

where $B_l(x) \equiv a(x) \bmod p_l(x)$. As a direct consequence of the Chinese remainder theorem it follows that the Modular Polynomial Transform is invertible. The intrinsic parallel structure of an MPT is illustrated in Figure 2, allowing a fast multiplication of degree n polynomials in $O(n^{1+\epsilon})$ or $O(n \, log \, n)$ time rather than $O(n^2)$ [8].

The Modular Polynomial Transform with respect to the factor basis $(x - \omega^i)_{i=1}^n$ is important, because it computes the Discrete Fourier Transform of the same length if the n-th roots ω^i of unity are contained in $\mathbb{F}$. Otherwise it implicitely computes the Discrete Fourier Transform via the Algebraic Discrete Fourier Transform. For this it is necessary to represent all results of the MPT in a common field extension.

The irreducible factors $p_i(x)$ of $x^n - 1$ can be viewed as minimal polynomials of several extension fields of $\mathbb{F}$. Each zero of the polynomials $p_i(x)$ is some root of unity. In the case where $\mathbb{F}$ is the field of rationals, all quotient fields $\mathbb{Q}[x]/\langle p_i(x)\rangle$ can be viewed as subfields of the cyclotomic field $\mathbb{Q}[x]/\langle \phi_n(x)\rangle$, where ϕ_n is the n-th cyclotomic polynomial.

In the case where $\mathbb{F}$ is of characteristic p the situation is somewhat different. The usual cyclotomic polynomial $\phi_n(x) \in \mathbb{Q}[x]$ may be reducible over $\mathbb{F}$, and the situation may appear that some quotient fields are isomorphic. In this case, it suffices to work with one of the isomorphic fields, saving further computations after a local change of base to so-called normal bases. Recall that an $\mathbb{F}_p$-basis

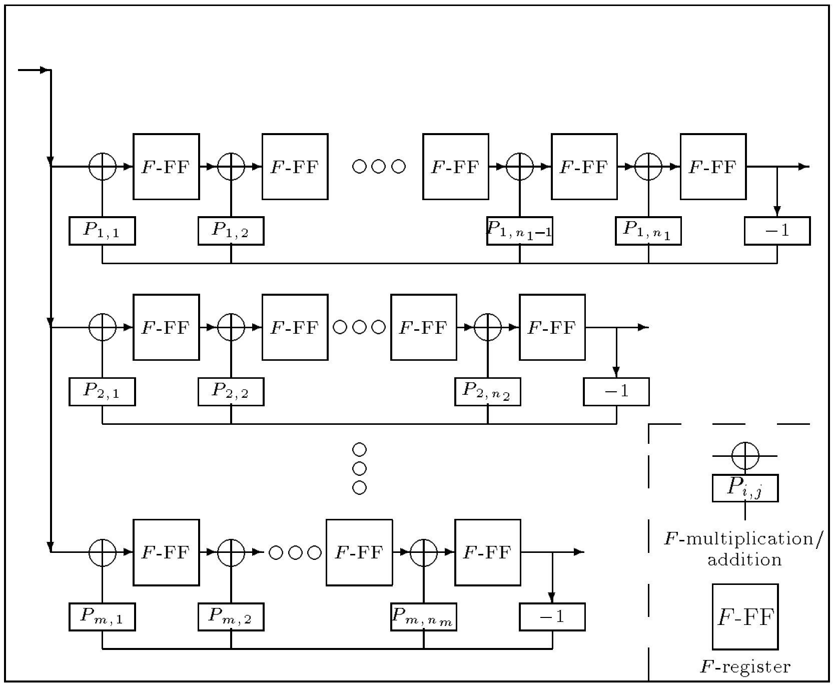


Fig. 2. The structure of Modular Polynomial Transforms over finite fields

of the form $(\alpha, \alpha^{p^1}, \alpha^{p^2}, \ldots, \alpha^{p^{n-1}})$ is called a *normal basis* of $\mathbb{F}_{p^n}$. Two normal bases $(\alpha, \alpha^{p^1}, \ldots, \alpha^{p^{n-1}})$ and $(\beta, \beta^{p^1}, \ldots, \beta^{p^{n-1}})$ are called *dual* iff

$$\sum_{i=0}^{n-1} (\beta_i \alpha_j)^{q^i} = \delta_{i,j}$$

holds. Moreover, if the two dual bases coincide, then the normal basis is called a *self-dual*.

3.2 The Algebraic Discrete Fourier Transform

Suppose that $\mathbb{F}$ is a field with a characteristic not dividing n. Denote by ω a primitive n-th root of unity and by $\mathbb{E}$ the Galois extension $\mathbb{F}(\omega)$. Furthermore, let DFT_N^n be the DFT-matrix (ω^{ij}) of size $n \times n$, where each coefficient is represented with respect to a normal base N of the field extension $\mathbb{E}/\mathbb{F}$. The ADFT is defined by the matrix that is obtained from the (ω^{ij}) matrix by projecting each coefficient ω^{ij} onto its first component with respect to the normal base B.

In this section an algorithm to compute the ADFT via the MPT is described. In a first step the $MPT_{(\mathbb{F}, x^n - 1)}(a(x))$ is used to compute the remainders B_i. Then the different fields $\mathbb{F}/\langle p_i(x) \rangle$ are injected into the common field $\mathbb{E}$. Afterwards, a second base change is applied to represented the results with respect to a joint normal base N. It can be shown that the resulting transform computes the ADFT spectrum up to a permutation. Therefore, it is possible to use the MPT in order to compute the ADFT.

The multiplications required in the feedback shift registers realizing the MPT can be visualised by so-called multiplication matrices. Let $\beta \in \mathbb{F}_{p^n}$ be a primitive element of $\mathbb{F}^*_{p^n}$, then there is an isomorphism (the so-called dlog) $\phi : \mathbb{F}^*_{p^n} \rightarrow [1, \ldots, p^n - 1]$, mapping each element $b \in \mathbb{F}^*_{p^n}$ to a value $i \in \mathbb{Z}_{p^n-1}$ with the property: $b = \beta^i$. Let M_β denote the associated matrix with the property $b \cdot M_\beta = b \cdot \beta$ for all $b \in \mathbb{F}$, where $b \cdot \beta$ is regarded as a vector.

With the knowledge of M_β, gate level networks for fixed elements may be compiled. If $p(x)$ is the defining polynomial of $\mathbb{F}$, then M_β is of the form:

$$M_\beta = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & \ldots & \vdots \\ \vdots & & \ddots & 1 & 0 \\ 0 & \ldots\ldots & & 0 & 1 \\ -p_{n-1} & \ldots\ldots & & -p_1 & -p_0 \end{pmatrix} \tag{1}$$

Using this knowledge, it is clear how to generate VLSI layouts based on the method of algebraic compilation: The diagonal of 1's represent a shift operator followed by a feedback set of taps p_i.

3.3 VLSI Implementations

All the outputs of the computer algebra system, for which in this case AXIOM [16] was used, are sent to a parser and an HDL code generator. Together with commercial synthesis tools, a CMOS layout has been generated. Figure 3 shows an MPT layout for MPT($\mathbb{F}_{2^4}, x^7 - 1$) reflecting the factorization of $x^7 - 1$ which is $(x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$. In the next case study it is shown that architectural information is already available at the algebraic level and can be used to improve geometries.

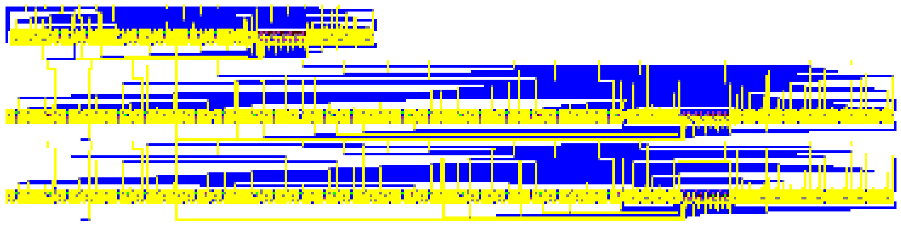


Fig. 3. Layout of the MPT of the polynomial $x^7 - 1$ is characterized by the factorization $(x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$.

It is possible to choose one of the most suitable normal bases with respect to the number of operations necessary to compute the base changes, refer also [14]. Figure 4 gives some results for different MPTs and bases. Thus only by some change of bases, a major reduction of the hardware was achieved. A similar transformation as described in the example can be computed with 4 additions and 2 multiplications. This completes the section on MPT, describing the use of computer algebra for compiling and optimizing these kinds of algorithms.

transform		additions		multiplications	
field	length	min	max	min	max
$\mathbb{F}_2$	3	1	1	-	-
$\mathbb{F}_4$	7	6	6	-	-
$\mathbb{F}_3$	4	2	2	1	4
$\mathbb{F}_3$	8	4	4	2	7
$\mathbb{F}_3$	16	24	24	7	14
$\mathbb{F}_5$	6	2	4	1	7
$\mathbb{F}_5$	12	4	8	3	14

Fig. 4. Some results of optimization.

4 Case Study: Fast Wavelet Transform

Wavelets are special orthogonal functions ψ which are localized having some oscillations. As such they can be chosen adapted to a certain waveform, giving high correlations with a signal. However, under the modelling paradigm of 1.1 such waveforms, such as noises or cracks on records, or ultrasound signatures, will be squeezed or spread and delayed in signal space. Therefore, the wavelet base of the model signal space is generated by dilation and translation from the *mother wavelet* ψ.

4.1 Preliminaries

Wavelet bases are special orthonormal bases of the space of square integrable functions $L^2(\mathbb{R})$ given by a family of wavelets of the form

$$\psi_{j,n}(x) = 2^{-j/2}\,\psi(2^{-j}x - n), \qquad j, n \in \mathbb{Z}.$$

A closer look at the construction of wavelet bases reveals an elegant way to derive the Fast Wavelet Transform (FWT) algorithm [10, 18].

A natural starting point for the construction is a *multiresolution analysis,* that is, in particular, a ladder of approximation subspaces

$$\{\,0\,\} \subset \cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \cdots \subset L^2(\mathbb{R}),$$

where for each $j \in \mathbb{Z}$ the subspace V_j is spanned by an orthonormal basis $\left\{2^{-j/2}\,\varphi(2^{-j}x - n) \mid j, n \in \mathbb{Z}\right\}$. The space V_0 is invariant under integer translations. The function φ is called *scaling function.*

The different approximation spaces are connected by the following dilation property:

$$f(x) \in V_j \Longleftrightarrow f(2x) \in V_{j+1}.$$

Therefore, the scaling function φ fulfills a dilation equation:

$$\varphi(x) = \sqrt{2} \sum_{n \in \mathbb{Z}} h_n\, \varphi(2x - n), \qquad \text{with suitable} \qquad h_n \in \mathbb{C}. \tag{2}$$

An orthonormal wavelet can be derived from the scaling function φ as follows:

$$\psi(x) = \sqrt{2} \sum_{n \in \mathbb{Z}} g_n\, \varphi(2x - n), \tag{3}$$

where $g_n = (-1)^n\, \overline{h_{k-n}}$, and k is an odd integer. The equations (2) and (3) play a key role in the construction of the FWT, which will be explained in the next section.

The hierarchic structure of a multiresolution analysis resembles that of the subgroup chain when deriving linear systems with symmetry. The techniques of the first case study are not applicable however, because the underlying vector spaces are not finite dimensional. Nevertheless, structural properties are used again to derive the fast algorithm.

4.2 The Fast Wavelet Transform Algorithm

Assume that the input signal s is already in an approximation space, say V_{-1}. Then this signal s can be represented as a sequence of "sample values"

$$(\ldots, \langle\, \varphi_{-1,n-1} \mid s \,\rangle, \langle\, \varphi_{-1,n} \mid s \,\rangle, \langle\, \varphi_{-1,n+1} \mid s \,\rangle, \ldots), \tag{4}$$

where $\varphi_{j,n}(x) = 2^{-j/2}\,\varphi(2^{-j}x - n)$, with respect to the orthonormal basis of V_{-1}. Denote by W_0 the orthogonal complement of V_0 in V_{-1}. The space W_0 comprises the "details" of V_{-1} missing in V_0. It can be shown that the integral translates of ψ constitute a basis for W_0.

The FWT realizes the change of base from V_{-1} to $V_0 \oplus W_0$. Substituting the integral translates of the scaling function $\varphi(x)$ in equation (2) and of the wavelet $\psi(x)$ in the equation (3), leads to an elementary decomposition step of the FWT:

$$\begin{aligned}
\langle\, \varphi_{0,m} \mid s \,\rangle &= \sum_{n \in \mathbb{Z}} \overline{h_{n-2m}}\, \langle\, \varphi_{-1,n} \mid s \,\rangle, \\
\langle\, \psi_{0,m} \mid s \,\rangle &= \sum_{n \in \mathbb{Z}} \overline{g_{n-2m}}\, \langle\, \varphi_{-1,n} \mid s \,\rangle.
\end{aligned}$$

In other words, an elementary decomposition step can be visualized in matrix form as follows:

$$\begin{pmatrix} \ddots & \ddots & \ddots & \ddots & & & & \\ & \cdots \overline{h}_0 & \overline{h}_1 & \overline{h}_2 & \overline{h}_3 \cdots & & & \\ & & & \cdots \overline{h}_0 & \overline{h}_1 & \overline{h}_2 & \overline{h}_3 \cdots & \\ & & & & & \ddots & \ddots & \ddots \; \ddots \\ \ddots & \ddots & \ddots & \ddots & & & & \\ & \cdots \overline{g}_0 & \overline{g}_1 & \overline{g}_2 & \overline{g}_3 \cdots & & & \\ & & & \cdots \overline{g}_0 & \overline{g}_1 & \overline{g}_2 & \overline{g}_3 \cdots & \\ & & & & & \ddots & \ddots & \ddots \; \ddots \end{pmatrix} \tag{5}$$

The design process for a VLSI implementation of an elementary decomposition step is sketched in the following sections.

4.3 VLSI Implementation

Input signals in signal processing applications are often associated with the rational sequence data type. From an algorithmic point of view, the FWT convolves in one decomposition step the rational input sequence with the sequences $(\bar{h}_{-n})_{n\in\mathbb{Z}}$ and $(\bar{g}_{-n})_{n\in\mathbb{Z}}$ and then drops every second sample in the resulting two sequences. Consider the sequence (h_n) corresponding to the Daubechies wavelet of order two [9]:

$$h_0 := \frac{1+\sqrt{3}}{8}, \; h_1 := \frac{3+\sqrt{3}}{8}, \; h_2 := \frac{3-\sqrt{3}}{8}, \; h_3 := \frac{1-\sqrt{3}}{8}. \tag{6}$$

The objective of this section is to derive an architecture for an elementary decomposition step of the FWT using internally symbolic arithmetic and delivering fixed point arithmetic.

At first glance, an obvious solution is to approximate the coefficients (h_n) numerically, for example as follows:

$$h_0 := 0.3415064, \; h_1 := 0.5915064, \; h_2 := 0.1584936, \; h_3 := -0.0915064.$$

Integral coefficients are obtained after a suitable normalization. The multiplication with integer constants can be realized with shifts, additions, and subtractions. When IDEAS is provided with these coefficients, the system generates an elementary decomposition step with 26 adders and subtractors in total for a twenty bit resolution.

Following our comments about application driven re-modelling (cf. 1.1), alternatively, the specification can be re-written to take advantage of the specific arithmetic structure of the coefficients in (6). It is possible to specify the coefficients as elements of the field extension $\mathbb{Q}(\sqrt{3})$ instead of the rational domain.

Using the isomorphism $\mathbb{Q}(\sqrt{3}) \cong \mathbb{Q}[x]/\langle x^2 - 3\rangle$, the coefficients (6) can be expressed as follows:

$$(h_0,\, h_1,\, h_2,\, h_3) = \left(\frac{1+x}{8},\ \frac{3-x}{8},\ \frac{3+x}{8},\ \frac{1-x}{8}\right) \qquad \bmod\ (\, x^2 - 3\,).$$

At first sight nothing is gained. However, conjugacy can be used to reduce hardware, using similar techniques as in the ADFT in section 3.2.

The automorphism $\sqrt{3} \mapsto -\sqrt{3}$ of the Galois group $\mathrm{Gal}(\mathbb{Q}(\sqrt{3})/\mathbb{Q})$ maps the coefficient sequence (h_n) onto the "mirrored" sequence (h_{3-n}). Up to change of sign, this conjugated sequences coincides with the sequence (g_n), which is given by $g_n = (-1)^n\, h_{3-n}$. Thus, the redundancy involved in the symbolic polynomial computation can be used to avoid effort for the computation of the convolution with (g_n). In order to transform the results from $\mathbb{Q}[x]/\langle x^2 - 3\rangle$ to the fixed point format, numerical approximations to the roots of the minimal polynomial $x^2 - 3$ are subsituted for x.

Using the methods of the preceeding case study, IDEAS optimizes the polynomial bases to minimize the necessary operations. Depending on the algebraic structure, geometric information is derived in order to place and route the basic cells, implementing additions, subtractions, and circuits multiplying signal values with constants, etc. The current implementation of IDEAS produced a layout for one decomposition step based on 15 adders and subtractors. A part of the layout is shown in Figure 5. The placement and routing information is detected on the algebraic level. Due to the close integration of Computer Algebra Systems, ELLA, and ISIS, it is possible to use this geometric information for the generation of architectures.

5 Future Technology: Quantum Gates

The newly emerging fields of Quantum Computing as described in the contribution by Brassard in this volume presents a most natural area of applications for the IDEAS design tool, as we shall show in this last case study.

5.1 Preliminaries

The state space of a Quantum Computer can be viewed as a Hilbert space $\mathcal{H}$ of kets $|\,\psi\,\rangle$, the dual space of which is formed by the bras $\langle\,\varphi|$ with respect to the hermitian form $\langle\,.|.\,\rangle$. Since proper computational states are those kets $|\,\psi\,\rangle$ for which $\langle\,\psi|\,\psi\,\rangle = 1$, the dynamics of the Quantum Arithmetical Logical Unit (QALU) is necessarily given by unitary transforms on $\mathcal{H}$, which for practical reasons as well as for complexity aspects afford representations of small degree. Without loss of generality we therefore assume that the state transition functions of a QALU are matrices $U \in \mathcal{U}(N)$ the unitary group of degree N. A computation c links an initial state $|\,\psi\,\rangle \in \mathcal{H}$ with the final state $|\,\varphi\,\rangle = U_c|\,\psi\,\rangle$ weighted with the amplitude $a_{\varphi\to\psi} = \langle\,\varphi|\,\psi\,\rangle$ via the state transition matrix U_c associated with c. If U_c can be written as a product $U_c = U_{t_l}\ldots U_{t_1}$ with $U_{t_i} \in \mathcal{G}$ from a given

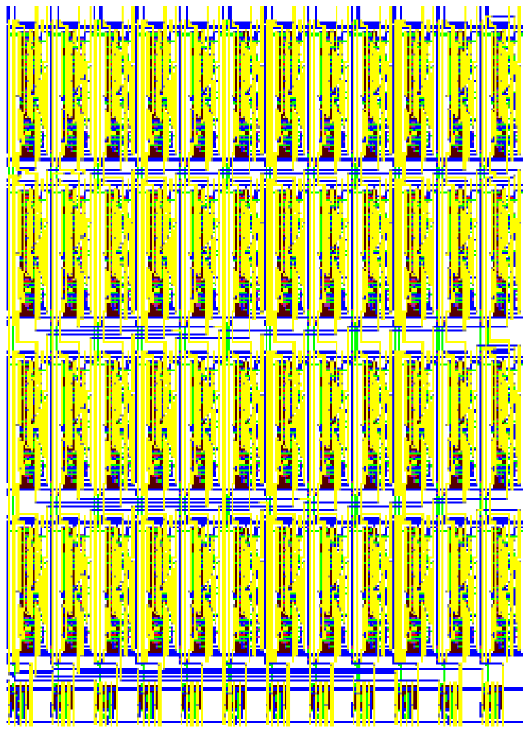


Fig. 5. Fast Wavelet Transform in a 12 bit realization. A part of the layout is shown. The full circuit is based on 4806 transistors and needs 1.544 mm^2 area with respect to a 1μ dual metal CMOS process.

set $\mathcal{G}$ of computational primitives of so-called quantum gates, the number l is the length of the computation. In this sense complexity issues can canonically be related to word problems in $\mathcal{U}(N)$ w.r.t. the generating set $\mathcal{G}$.

5.2 Quantum parallelism

What is more, not only sequential computations can be expressed in this manner by forming products, but the concept of true parallelism comes very natural by considering the tensor space $\mathcal{H} \otimes \mathcal{H}'$ which is physically constructed by quantum interference coupling of the two separate QALU's with state spaces $\mathcal{H}$ (resp. $\mathcal{H}'$) by entanglement (*Verschränkung,* so-called by E. Schrödinger). Its state vectors are given by the tensors $|\,\psi, \xi\,\rangle := |\,\psi\,\rangle \otimes |\,\xi\,\rangle$ with $\psi \in \mathcal{H}$, $\xi \in \mathcal{H}'$. On this space the unitary transforms of the form $W = U \otimes V \in \mathcal{U}(NN')$ with $U \in \mathcal{U}(N), V \in \mathcal{U}(N')$ resemble fully parallel computations as for the Kronecker product of transforms we have $U \otimes V = (U \otimes I_{N'}) \cdot (I_N \otimes V)$, cf. section 2.2. The the meaning in quantum micro code of e.g. $U \otimes I \in \mathcal{U}(NN')$ is: "Apply the unitary transform to all kets $|\,\psi, \xi\,\rangle \in \mathcal{H} \otimes \mathcal{H}'$ simultaneously by altering the left part amplitudes according to U and leave the right part unchanged."

5.3 Superposition at its best

With this trick we can automatically generate an algorithm for producing an coherent equidistribution of all 2^n states in one quantum register consisting of n entangled 2-state systems $\mathcal{H}_i$, $i \in [1:n]$, whose base in given by the kets $\{|\varepsilon_i\rangle \mid \varepsilon_i \in \{0,1\}\}$. The n-fold tensor space $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_n$ of dimension 2^n consists of all kets $|\psi\rangle = \sum_{i=1}^{n} \sum_{\varepsilon_i=0}^{1} \alpha_{\varepsilon_1,\varepsilon_2,\ldots,\varepsilon_n} |\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\rangle$. Starting from the initial ket $|0,\ldots,0\rangle$ the n-th Hadamard-Walsh transformation $W_n = W_1 \otimes W_1 \otimes \ldots \otimes W_1$ with (cf. equation 2.2)

$$W_1 = \frac{1}{\sqrt{2}} T_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{7}$$

produces the desired final state $|\psi_{equ}\rangle = 2^{-n/2} \sum_{i=1}^{n} \sum_{\varepsilon_i=0}^{1} |\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n\rangle$ in n parallel steps by applying W_1 successively to each of the n two-state systems $\mathcal{H}_i$.

5.4 Conditional Commands

Having produced a coherent superposition of all possible 2^n states equally "amply" computation can start on these simultaneously. For this different transforms U_k may have to be applied to different orthogonal states φ_k for $U \in [1:n]$. The transform $U_{cond} = \sum_k |\varphi_k\rangle\langle\varphi_k| \otimes U_k$ resembles the conditional CASE-operator. The matrix U_{cond} has the form of a block matrix $\mathrm{diag}(U_1, U_2, \ldots, U_k, \ldots)$, therefore representing a decomposition matrix for a subgroup of $\mathcal{U}(N)$ thus closing the loop to section 1.2 invoking the principles of ART in a most natural way.

5.5 Applying IDEAS: Automatic compilation and implementation

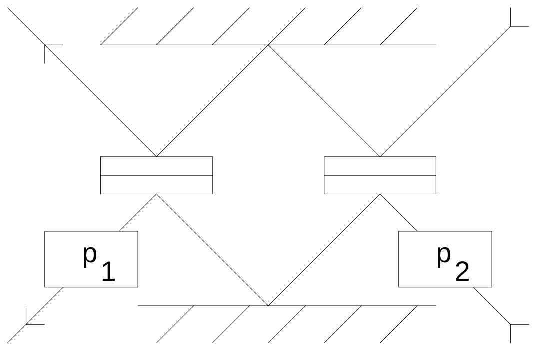
With these computational primitives, i.e., sequential, parallel, conditional composition, translated into unitary groups and therefore to the theory of Lie algebras, the IDEAS tools and paradigm provide a natural solution to the engineering task transforming the problem specification given by an initial /final state relationship $\langle\varphi|\psi\rangle$ for all $\varphi, \psi \in \mathcal{H}$ into the group-theoretic problem: To factor the matrix $U = (\langle\varphi|\psi\rangle)_{\varphi,\psi}$ with respect to the generating set $\mathcal{G}$. In other words, automatic compilation from a problem specification to a computation process will be possible. While a most general gate has been proposed by several authors [21, 11] resembling the basic primitive of a Boolean Horner scheme $(a, b, c) \to (a, b, (a \cdot b) \oplus c))$ on basis of which any Boolean polynomial presented in Ring-Normalform (i.e. the often called RM-Normalform) can be implemented, the IDEAS environment can be made to translate such unitary computations and compile them from a physically and practically feasible set $\mathcal{G}$ of generating customised gates adapted to efficient computational tasks as investigated recently by Beth, Blatt, Zoller & Weinfurter [3].

5.6 From Unitary Transforms to Layout: IDEAS all along

In implementing the unitary transform W_n, given as example in section 5.3, we have shown that is suffices to apply the basic transform

$$W_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{8}$$

on each Qbit $|\,\varepsilon_1\,\rangle$. A “Quantum” standard cell for this purpose has been known since more than a century, the so-called “Mach-Zehnder interferometer” acting on photons [21]. It can easily be build from beam splitters, phase shifters and mirrors cf. Figure 6.



$$\begin{pmatrix} 1 & 0 \\ 0 & -i \end{pmatrix} \cdot \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{i}{\sqrt{2}} \\ \frac{i}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Fig. 6. The Mach Zehnder configuration: A standard cell for Quantum Computing ($p_1 = -\frac{\pi}{2}, p_2 = \frac{\pi}{2}$) and the corresponding $\mathcal{U}_2$ description of the experimental configuration is shown

6 Conclusion

We have demonstrated the power and applicability of the modern algorithm and system design environment IDEAS, which resembles the new representation theoretic approach ART to the problem of automatic engineering algorithm giving correct and efficient implementation for computations in hardware and software. It has been shown that the concept of manipulating Abstract Data Types by proper tools of Computer Algebra is feasible, especially if the categories of mathematical structures presented by the user have a rich and intrinsic structural theory, such as those many areas which can be described by the classical language of today's modern math. By detecting symmetries in the problems presented and formalising congruence classes of patterns in the abstract data types, the algebraic approach not only allows the automatic generation of fast algorithms for solving problems in the model domains. What is more, it provides the opportunity of re-modeling the semantics of the model algebra according to symmetries detected in the algorithm development process. A surprising result of this type has been obtained when the IDEAS approach was applied to the design of molecular robotics [5], as a consequence of which also an improvement on the control of an ultra-large milling machine was achieved. The latter example shows in a nutshell that the approach sketched here has opened a completely new and promising road to solve wider classes of problems by the use of problem adapted description languages. The ART behind IDEAS therefore has developed beyond the Art of Computer Programming towards an art of integrated problem solving.

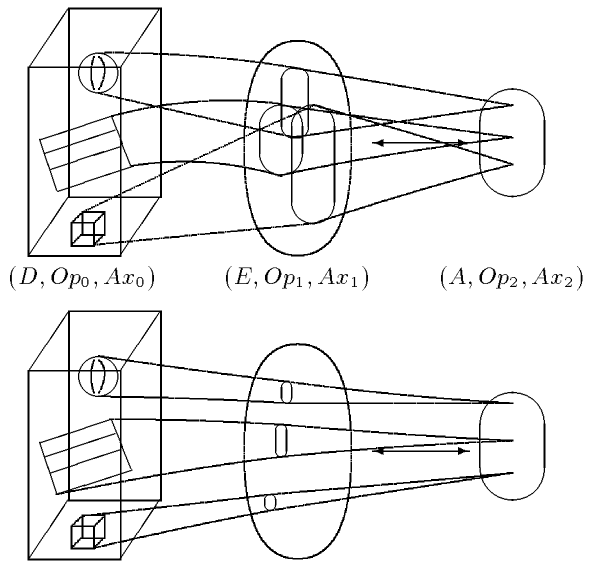


Fig. 7. Unsuitable description languages (upper picture) induce bad class formation in the model datatype due to inproper semantics. While in the lower picture a problem adapted modelling by a suitable semantic comes with “good” description languages.

Acknowledgements

We thank Volker Baumgarte, Sebastian Egner, Rüdiger Ganz, Willi Geiselmann, Dieter Gollmann, Martin Indlekofer, Kai Rohrbacher, Michael Vielhaber, and Otto Wohlmuth for their contributions to the IDEAS pilot system. The support by DFG, MWF (formally MWK) Baden-Würtemberg, WM Baden Würtemberg, INMOS, DRA (formerly RSRE), and Racal-Redac is gratefully acknowledged, which made the implementation of the IDEAS pilot system possible.

References

1. T. Beth. *Verfahren der schnellen Fourier-Transformation.* Teubner-Verlag, 1984.
2. T. Beth. Generating Fast Hartley Transforms – another application of the Algebraic Discrete Fourier Transform. In *Proc. Int. Sym. Signals, Systems, and Electronics ISSSE 89*, pages 688–692. Union Radio Scientifique International, 1989.
3. T. Beth, R. Blatt, P. Zoller, and H. Weinfurter. Engineering quantum gates. In preparation, 1995.
4. T. Beth, W. Fumy, and R. Mühlfeld. Zur Algebraische Diskreten Fourier Transformation. *Arch. Math.*, 40:238–244, 1983.
5. T. Beth, J. Müller-Quade, and A. Nückel. IDEAS - Intelligent Design Environment for Algorithms and Systems. In M. E. Welland and J. K. Gimzewski, editors, *Ultimate Limits of Fabrication and Measurement*, pages 49–57. Kluwer, 1995.
6. R. Blahut. *Theory & Practice of Error-Control Codes.* Addison-Wesley Publ. Comp., Reading, MA, 1983.
7. R. Blahut. *Fast algorithms for Digital Signal Processing.* Addison-Wesley Publ. Comp., Reading, MA, 1985.
8. B. Buchberger, Collins G. E., and Loos R. *Computer Algebra.* Springer-Verlag, Berlin, 1982.
9. I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41:909–996, 1988.
10. I. Daubechies. *Ten Lectures on Wavelets.* CBMS-NSF Reg. Conf. Series Appl. Math., SIAM, 1992.
11. D. P. DiVincenzo. Two-bit gates are universal for quantum computation. *Physical Review A*, 51(2), 1995.
12. E. Feig and S. Winograd. Fast algorithms for the Discrete Cosine Transform. *IEEE Trans. on Signal Processing*, pages 2174–2193, 1992.
13. W. Fumy and H. P. Rieß. *Kryptographie.* Oldenburg, 1988.
14. W. Geiselmann. *Algebraische Algorithmenentwicklung am Beispiel der Arithmetik in endlichen Körpern.* Dissertation, Universität Karlsruhe, 1993.
15. D. Gollmann. *Algorithmenentwurf in der Kryptographie.* Bibliographisches Inst., Mannheim, 1994.
16. R. D. Jenks and R. S. Sutor. *Axiom: the scientific computation system.* Springer-Verlag, Berlin, 1992.
17. F. J. MacWilliams and N. J. A. Sloane. *The Theory of Error-Correcting Codes.* North-Holland Publ. Comp., Amsterdam, 1977.
18. S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(7):674–693, Juli 1989.
19. T. Minkwitz. *Algorithmensynthese für lineare Systeme mit Symmetrie.* Dissertation, Universität Karlsruhe, 1993.
20. T. Minkwitz. Algorithms explained by symmetries. In E. W. Mayr and C. Puech, editors, *STACS 95, Proc. 12th Annual Symposium*, volume 900 of *Lecture Notes in Computer Science*, pages 157–167. Springer-Verlag, Berlin, 1995.
21. M. Reck and A. Zeilinger. Experimental realization of any discrete unitary operator. *Physical Review Letters*, 73(1), 1994.
22. J. P. Serre. *Linear Representations of Finite Groups.* Grad. Texts in Math. **42**. Springer-Verlag, 1977.