Description of document:	**Three Selected JASON Defense Advisory Panel/MITRE Corporation Reports released by the Department of Defense (DoD), 1984-1985**
Requested date:	12-August-2008
Released date:	25-March-2010
Posted date:	19-April-2010
Title of documents:	AD-B149675, Radical Computing, May, 1984, JSR-82-701 AD-B099221, Report on the Workshop for Automated Software Programming, 1985 AD-B149872, Space Power System Study, May, 1984, JSR-82-801
Source of document:	Department of Defense Office of Freedom of Information 1155 Defense Pentagon Washington, DC 20301-1155

DEPARTMENT OF DEFENSE
OFFICE OF FREEDOM OF INFORMATION
1155 DEFENSE PENTAGON
WASHINGTON, DC 20301-1155

MAR 2 5 2010

Ref: 08-F-1844
FOIA 2008-44

This is our second interim response to your Freedom of Information Act (FOIA) request, FOIA 2008-44, dated August 12, 2008, which you originally submitted to the Defense Technical Information Center (DTIC) for the below listed reports. As DTIC responded separately to you concerning the documents numbered 4 and 5, this response does not address those two documents.

1. AD-526617, Acoustic Backscatter from Microstructure, December 1971
2. AD-526067, Comments on Sub-LF SATCOM Technology Development Program, December 1972
3. AD-0528668, Summary Report of the 1973 JASON Summer Study, October 1973 (referred to the Air Force for direct response)
4. AD-B076811, Pulsed Electric Discharge Laser Technology Development Program AVCO Everett Research Lab, September 1982
5. AD-B069354, Summary of Trapped Electron Data, McDonnell Douglas Astronautics Co., October 1982
6. AD-B149872, Space Power System Study
7. AD-B149873, Speech Research, May 1984
8. AD-B149675, Radical Computing, May 1984
9. AD-B099221, Report on the Workshop for Automated Software Programming, February 1986
10. AD-B103383, Some Surface Wave Modulation Mechanism Relating to the JOWIP and SARSEX Observations, May 1986

DTIC forwarded eight documents to this Office, identified above as numbers 1-3, and 6-10. Enclosed you will find six documents. We previously advised you in our interim letter dated July 22, 2009, that "AD-0528668 Summary Report of the 1973 JASON Summer Study" (document number 3) was referred to the Department of the Air Force for direct response to you. The document identified as number 2 is still under review by this Office.

Ms. Delores M. Nelson, an Initial Denial Authority for the Central Intelligence Agency (CIA), has determined that some of the information in the enclosed documents is exempt from release pursuant to 5 U.S.C. § 552(b)(3), which pertains to information exempted from release by statute, in this instance, 50 U.S.C. § 403-3 (c)(7), which permits the withholding of intelligence sources and methods, and 50 U.S.C. § 403 (g), the withholding of functions and information regarding the CIA. Ms. Diane M. Janosek, an Initial Denial Authority for National Security Agency (NSA), has determined that some of the information in the documents are exempt from release pursuant to 5 U.S.C. § 552(b)(3), which pertains to information exempted from release by statute, in this instance, 50 U.S.C. § 402 Sec 6, the withholding of functions and information regarding the NSA. Ms. Alesia Y. Williams, an Initial Denial Authority for Defense Intelligence Agency (DIA), has determined that some of the information in the documents are exempt from release pursuant to 5 U.S.C. § 552(b)(3), which pertains to information exempted from release by statute, in this instance, 50 U.S.C. § 424, the withholding of information regarding the protection of organizational and personnel for DIA, NRO, and NGA.

If you are not satisfied with this response, you may appeal to the appellate authority, the Director of Administration and Management, Office of the Secretary of Defense, by writing directly to the Defense Freedom of Information Policy Office, Attn: Mr. James Hogan, 1155 Defense Pentagon, and Washington, D.C. 20301-1155. Your appeal should be postmarked within 60 calendar days of the date of this letter, should cite to case number 08-F-1844, and should be clearly marked "Freedom of Information Act Appeal."

Sincerely,

[illegible]

Paul J. Jacobsmeyer
Chief

Enclosures:
As stated

doc. 8

OK DTIC

84-0330

1

Radical Computing

AD-B149 675

90 12 4 031

MITRE

Radical Computing

A. M. Despain
G. J. MacDonald
A. M. Peterson
O. S. Rothaus
J. F. Vesecky

May 1984

JSR-82-701

JASON
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, Virginia 22102

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS.......... v

LIST OF TABLES.......... vii

1.0 INTRODUCTION.......... 1

2.0 RESIDUE ARITHMETIC COMPUTERS.......... 6
2.1 Introduction to Residue Arithmetic.......... 6
2.2 Table Look-up Method for Modular Arithmetic.......... 15

3.0 FURTHER DEVELOPMENTS IN RESIDUE ARITHMETIC.......... 24
3.1 Residue Arithmetic.......... 24
3.2 Application of Residue Arithmetic to Complex Numbers.......... 26
3.3 Implementation of Residue Arithmetic Using Complex Numbers.......... 29
3.4 Further Comments on Reducing the Size of Look-up Tables.......... 31

4.0 USE OF GAUSSIAN RESIDUE ARITHMETIC.......... 34
4.1 Introduction.......... 34
4.2 Example of Gaussian Residue Arithmetic.......... 35

5.0 A SAMPLE PROCESSOR USING GAUSSIAN RESIDUE ARITHMETIC.......... 41

6.0 SYMBOLIC COMPUTING.......... 51
6.1 Introduction.......... 51
6.2 History.......... 52
6.3 Language Issues.......... 53
6.4 Curent Development of Symbolic Manipulation Programs.......... 54

7.0 CONCLUSIONS.......... 57
7.1 Residue Arithmetic.......... 57
7.2 Symbolic Manipulation.......... 58
7.3 Final Remarks.......... 58

TABLE OF CONTENTS
(Conculded)

	Page
APPENDIX A-1 FORTRAN PROGRAM PRODUCED BY MACSYMA..........	A-1
APPENDIX A-2 REPRINT OF "A BIBLIOGRAPHY OF SOVIET WORKS IN ALGEBRAIC MANIPULATIONS".........	A-9
APPENDIX A-3 AN EXAMPLE OF SOVIET WORK IN MACHINE SYMBOLIC MANIPULATION.............	A-13
REFERENCES..	B-1

LIST OF ILLUSTRATIONS

		Page
Figure 2.1	Residue Addition	9
Figure 2.2	Residue Subtraction	10
Figure 2.3	Residue Multiplication	10
Figure 2.4	Polynomial Transforms by Table Look-Up	12
Figure 2.5	Residue to Mixed Radix Conversion	13
Figure 2.6a	Adder or Subtractor	16
Figure 2.6b	Multiplier	16
Figure 2.7	Six and Five Bit Coding	17
Figure 2.8	Reduced Multiplication Table for Modulo 7 Multiplication	19
Figure 5.1	Despain Cascade	42
Figure 5.2	Residue FFT Cascade Processor (N = 64)	43
Figure 5.3	Binary to Residue Conversion	43
Figure 5.4	Residue Pipe Configuration	44
Figure 5.5	Residue Pipe Stage m	44
Figure 5.6	Residue to Binary Conversion for FFT Cascade	45

LIST OF TABLES

Page

Table 1.1 Computer Research Topics that Could Lead to a Radical Improvement in Computing............. 5

Table 2.1 Modulus 7 Multiplication.......................... 23

Table 2.2 Modulus 7 Addition................................ 23

Table 4.1 Residue Representation of Gaussian Integers.......... 36

Table 4.2 Gaussian Primes Less Than 150........................ 40

Table 5.1 Small Complex Integer Approximations to Rotations................................ 49

Table 5.2 Six Bit Approximations to $n\frac{\pi}{32}$ with Gain of 14...................................... 49

1.0 INTRODUCTION

During the 1982 JASON summer study we investigated certain aspects of both classical mathematics and advanced computing methods which, if applied, could lead to radical increases in the speed of at least certain computer calculations. In the past, such methods have included the fast Fourier transform (FFT), the simplex method, and the von Neumann machine architecture. The methods considered include theory, software, and machine organization, but not technological improvements [e.g., very high speed integrated circuits (VHSIC)]. This report contains both introductory material and new results. For example, ~~we~~ the authors discuss the basics of residue arithmetic in Chapter 2, while Chapter 3 reports some new applications of complex numbers to residue arithmetic and Chapter 5 lays out a design for an FFT processor using residue methods.

While the general goal of the study was to identify new approaches for computer system development, there was also the desire to develop a sensitivity to critical mathematical and computer concepts that might aid in tracking worldwide developments in computer systems. Developments within the closed areas of the USSR are of particular interest. Indeed, a general question is whether one can, by innovatively using mathematics, significantly

Keywords: residue arithmetic computers, Gaussian residue arithmetic, symbolic computing. (

simplify computational tasks, thus overcoming deficiencies in computer components and software.

There has been speculation that the development of computer systems in the Soviet Union emphasizes algorithm development and machine organization and that this emphasis has led to advances in computational efficiency which offset the well-known lag in Soviet development of very large scale integration (VLSI) technology. In particular, the Soviets may have compensated for their evidently poor integrated circuits with clever algorithms, innovative machine organizations, and so on. Several past Soviet achievements are often mentioned to support this speculation: First, the Soviet space program has required extensive orbital calculations that would have been beyond the "poor" computers thought to be available to them in the late 1950s. Their ICBM developments raise similar questions. Second, Soviet computers have beaten U.S. computers in chess, in the only two engagements that have been played (1978 and 1980). Third, the USSR has mounted a relatively large research and teaching effort in residue arithmetic. They have published far more articles and textbooks on this subject than the rest of the world. Could they be training students for radically different computing machines, or are the publications just a reflection of an academic "publish or perish" syndrome? Fourth, the Soviets have recently produced interesting and important results in Linear Programming Theory.

If one chooses to pursue this speculative scenario for a radical Soviet approach to computing, there seems to be at least two possible themes. The most evident theme is <u>residue arithmetic and number theoretic computation</u>. This approach to computing can compensate for poor VLSI circuits in that the computations can be mostly "table driven." Such tables could be implemented in optical memories, instead of VLSI circuits, for example. A second theme might be an emphasis on <u>symbolic rather than numerical computation</u>, i.e., manipulation of mathematical expressions as opposed to numbers. An example of symbolic computation is given in Appendix A.1. It is known that the Soviets have a special symbolic computer language "ANALITIK-71" that runs on the MIR-2 machine. The MIR-2, openly known to the West, is about 15 years old and cannot represent state of the art Soviet technology. Nevertheless, it has some intriguing features. It is microprogrammed (table-like control) to perform at least some <u>symbolic</u> operations. This has only recently occurred in the United States. It also has some form of pattern-matching control (details unknown). Appendix A.2 contains a bibliography of some Soviet work in symbolic computing. Appendix A.3 contains a specific example of Soviet work in machine symbolic manipulation.

Besides the two topics mentioned above, there are a number of rapidly developing areas of computer research that could lead to a radical increase in computing speed. These will only be listed here in Table 1.1 but may deserve further detailed study.

This report focuses on the two main topics discussed above: residue arithmetic and symbolic computing. Each topic is of current research interest in the West, and there are indications that these themes may be even more important to the Soviets.

TABLE 1.1

Computer Research Topics that Could
Lead to a Radical Improvement in Computing

Topic	Remarks
Floating Point Arithmetic with Embedded Symbols	Recently inserted into proposed IEEE standard study for Floating Point Arithmetic (NAN's)
Functional Programming	Work by J. Backus and others
Interval Arithmetic	Unknown utility. Some provision in IEEE F.P. Standard (directed rounding modes)
Recursive Machines	Work by Glushkov, Barton, Wilner, Davis, Mago et al.
Multiple Processor Concurrency	Extensive worldwide interest
Fast Recurrence Evaluation	Good potential but inconsistent with Soviet style
"Modern" DDA Machine	Soviet and Dutch interest
"Data-Flow" et al.	Extensive investigation in U.S.
Hyper-Column Cortex Model	Speculative
Symbolic Computing	A subject of this report
Residue Arithmetic and Number Theoretic Computing	A subject of this report

2.0 RESIDUE ARITHMETIC COMPUTERS

2.1 Introduction to Residue Arithmetic

In the residue number system a positive number is represented in the form of a set of residues with respect to a sequence of positive integers $m_1, m_2, \ldots, m_r$, each of which is called a modulus. The moduli must not have common factors; that is, the moduli must be relatively prime. The idea is to have several moduli $m_1, m_2, \ldots, m_r$ and to work indirectly with residues $x \bmod m_1, x \bmod m_2, \ldots, x \bmod m_r$ instead of directly with the number x. We will use the following notation for the residues:

$$x_1 = x \bmod m_1, \; x_2 = x \bmod m_2, \; \ldots, \; x_r = x \bmod m_r$$

We may regard $(x_1, x_2, \ldots, x_r)$ as a new type of internal computer representation, a modular representation of the integer x.

The residue of $x \bmod m_i$ is the least positive integer remainder of the division of x by m_i, where x is the number to be converted and m_i is the modulus. The range M is defined as the product of the moduli. Only between 0 and M - 1 are the results of the arithmetic operation unique. In the following examples the moduli chosen are 5, 7, and 8. Therefore, M = 280 and the range before overflow is from 0 to 279. If the sign is represented implicitly, then the range becomes -140 to 139.

The advantages of a modulus representation are that addition, subtraction, and multiplication are very simple to implement. The amount of time required to add, subtract, or multiply n-digit numbers is essentially proportional to n (not counting the time to convert in and out of modular representation). This is a considerable advantage with respect to multiplication, since the conventional method requires an execution time proportional to n^2 for repeated addition and shifting.

In addition, as pointed out by Knuth (1969), on a computer designed to allow many operations to take place simultaneously, modular arithmetic can be a significant advantage even for addition and subtraction. The operations with respect to different moduli can all be done at the same time, which results in a substantial savings in execution time. The same kind of decrease in execution time could not be achieved by conventional means, since carry propagation must be considered. A comment by Knuth is particularly interesting:

Perhaps some day highly parallel computers will make simultaneous operations commonplace, so that modular arithmetic will be of significant importance in "real-time" calculations when a quick answer to a single problem requiring high precision is needed. (With highly parallel computers, it is often preferable to run K separate programs simultaneously, instead of running a single program K times as fast, since the latter alternative is more complicated but does not utilize the machine any more efficiently; "real-time" calculations are exceptions which make inherent parallelism of modular arithmetic more significant.)

The <u>disadvantages of a modular representation</u> are that it is comparatively difficult to test whether a number is positive or negative or to test whether or not $(u_1, \ldots, u_r)$ is greater than $(v_1, \ldots, v_r)$. It is also difficult to test whether or not overflow has occurred as the result of an addition, subtraction, or multiplication, and it is even more difficult to perform division. When these operations are required frequently in conjunction with addition, subtraction, and multiplication, the use of modular arithmetic can be justified only if fast means of conversion into and out of the modular representation are available.

2.1.1 Residue Addition

	Moduli	5	7	8
19	→	4	5	3
87	→	+2	+3	+7
106	→	1	1	2

Figure 2.1. Residue addition.

In the example shown in Fig. 2.1, 19 is converted to 4 5 3 in the residue system. The 4 is the residue of 19 mod 5. In other words, 4 is the least positive integer remainder of the division of 19 by 5. Similarly, 5 results from modulus 7, and 3 from modulus 8. Eighty-seven is translated in a similar manner. Each column is independently added, and the sum is expressed as a residue of the associated modulus. In the first column, 4 + 2 = 6, but this is a 1 for modulus 5. The other columns are processed in a similar manner. The result 1 1 2 means that the actual result, when divided by 5, gives a remainder of 1; when divided by 7, a remainder of 1; and when divided by 8, a remainder of 2. These clues uniquely determine the number within the range of 0 to 279. As a quick check, the actual answer can be converted into residues to see if the digits correspond.

2.1.2 Residue Subtraction

		Moduli	5	7	8
106		→	1	1	2
− 99	→ 4 1 3	→	+1	+6	+5
7		→	2	0	7

Figure 2.2. Residue subtraction.

In the example shown in Fig. 2.2, the subtrahend is first converted into residues, and each residue digit is then complemented with respect to its particular modulus, 5 resulting in a 1. The other residues are complemented in a similar manner. The rest of the example proceeds as with addition.

2.1.3 Residue Multiplication

	Moduli	5	7	8
11	→	1	4	3
x 12	→	x 2	x 5	x 4
22				
+ 11				
132	→	2	6	4

Figure 2.3. Residue multiplication.

In the example shown in Fig. 2.3, the multiplier and multiplicand are converted into residues. The product of each column is expressed as a residue with respect to the corresponding modulus.

2.1.4 Modular Multiplicative Inverse and Residue Division

The multiplicative inverse, MINV, of a number A for a modulus m is the smallest positive number such that A*MINV = 1 mod m.

Division can be performed by multiplying the dividend by the multiplicative inverse of the divisor. This operation only works when the division is exact (no remainder) and when the divisor is not a multiple of any of the moduli. Long division, using a multiply, subtract, and test procedure, can be performed, but this is quite awkward since the test requires a mixed radix conversion (Szabo and Tanaba, 1967).

2.1.5 Conversion Using Chinese Remainder Theorem

A residue number may be converted into a decimal number with a procedure based on the Chinese Remainder Theorem. When applied to these particular moduli, the following conversion formula results:

$$\mathrm{mod}\ (56{*}R_5 + 120{*}R_7 + 105{*}R_8,\ 280)$$

where MOD(A, B) is the least positive remainder of A divided by B, and R_5, R_7, and R_8 are the respective residue digits. The result of the addition, 1 1 2, converts to 106 in decimal. If the sign is represented implicitly, then

If $X > 139$, then X is replaced by $X - 280$.

2.1.6 Evaluation of Polynomials by Table Look-up

$$P(X) = X^2 - X + 1$$

Residue	Moduli	5	7	8
0	....	1	1	1
1	....	→ 1	1	1
2	....	3	3	3
3	....	2	0	→ 7
4	....	3	→ 6	5
5	....		0	5
6	....		3	7
7	....			3

Figure 2.4. Polynomial transforms by table look-up.

Tables like that shown in Fig. 2.4 can be constructed to perform integer polynomial transforms. The tables for the transform $X^2 - X + 1$ are shown. To perform a transform, a number is first encoded into residues, and then each residue digit is used to index the appropriate table. If X is 11, then this is 1 4 3 in residues. The first digit is used to index the leftmost table. The result (noted by the arrow in Fig. 2.4) is 1. The rest of the digits are translated in a similar manner. The result, noted by the arrows, is 1 6 7, which, upon using the conversion formula, is 111.

2.1.7 Residue to Mixed Radix Conversion

Moduli	5	7	8
$a_o \rightarrow$	1	6	7
	− 1	− 1	− 1
	0	5	6
		x 3	x 5
$a_1 \rightarrow$		1	6
		− 1	− 1
		0	5
$a_2 \rightarrow$			x 7
			3

Figure 2.5. Residue to mixed radix conversion.

The conversion from residues to a mixed radix number system is simpler than the technique based on the Chinese Remainder Theorem. This conversion, as shown in Fig. 2.5, can be performed with residue arithmetic. It is based on Euclid's base conversion algorithm and involves a cast-off and divide procedure.

The operations in the mixed radix conversion can be performed by polynomial transforms of the form P(X,Y) = (X-Y) x C, where X is a modular result, Y is the modular result to be subtracted, and C is the multiplicative inverse indicated below.

If the residue number to be converted is 1 6 7, then 1 1 1 would be subtracted from this to give 0 5 6. The 0 indicates that this number, i.e., the weighting factor of a_0, is exactly divisible by 5. Rather than dividing by 5, the remaining modular results are multiplied by the multiplicative inverse of 5 for each of the remaining moduli. The modulus 7 digit 5 is multiplied by 3, which is the multiplicative inverse of 5 for modulus 7 since MOD(5*3,7) = 1. Similarly, 6 is multiplied by 5. The result is 1 6. 1 1 is subtracted from this to give 0 5. To divide this by 7, the 5 is multiplied by 7 which is the multiplicative inverse of 7 for modulus 8.

The numbers 3, 1, and 1 (a_2, a_1, and a_o) which were subtracted are the coefficients of the number in a mixed radix number system. The weighting factor associated with each coefficient is the product of 1 and all the previously zeroed moduli. In this case,

$$X = 1*5*7*a_2 + 1*5*a_1 + 1*a_o$$

Thus the residue number 1 6 7 is equivalent to 3, 1, 1 in mixed radix, which is equal to 3*35 + 5*1 + 1 = 111 in decimal.

If the sign is represented implicitly, then

If $a_2 > 3$, then X is replaced by X - 280.

2.2 Table Look-up Method for Modular Arithmetic

In table look-up arithmetic the binary representation of the two numbers x_i and y_i which are to be added, subtracted, or multiplied are used as addresses to a random access memory (RAM, ROM, etc.), and the word read from the selected address is the sum, difference, or product. For example, to add 2 numbers of four bits, each will require 8 address lines and will then address one of 2^8 words in the memory. Each word will need to be of 4 bits (5 if a carry bit is needed). Fig. 2.6 shows this operation diagramatically. A multiplier is essentially the same except that for ordinary binary arithmetic the product can be as large as 8 bits.

Memories of this size are readily available and operate at address to output delays less than 50 nsec. On the other hand, if 16-bit binary arithmetic is desired, the memory requires 32 address lines and 32-bit words to represent the product of 16-bit binary numbers. The multiplier in this case would require $2^{32} \times 32 = 2^{37}$ bits of memory ($\approx 1.4 \times 10^{11}$ bits). This would be totally unfeasible at the present time. Thus, for ordinary binary arithmetic, table look-up methods can rarely be used.

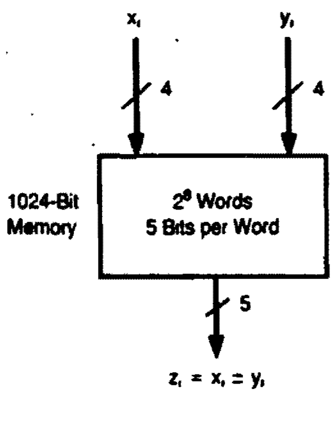


Figure 2.6a. Adder or subtractor

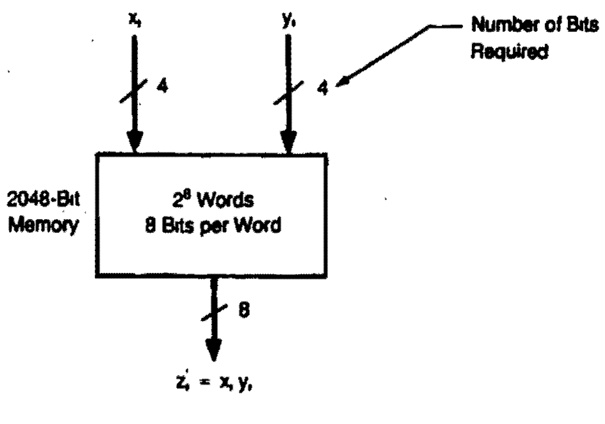


Figure 2.6b. Multiplier

On the other hand, using modular representation of numbers, table look-up methods become quite attractive. In this case, arithmetic is done independently for each of the moduli, and numbers within the moduli can be represented by binary codes of less than about 7 or 8 bits. For example, a modular processor might use moduli of decimal values $x_i = 101, 103, 107, 109, 113, 127$, each of which can be represented by 7-bit binary numbers. The range of numbers represented by the moduli listed above is

$$R = \prod_{i=1}^{r} x_i \approx 1.7 \times 10^{12}$$

Alternatively suitable selections of moduli representable by 6-bit and 5-bit binary coding are

6 bit $\quad x_i = 64, 61, 59, 57, 53, 49$

$$\text{Range} = \prod_{i=1}^{r} x_i \approx 3.4 \times 10^{10}$$

5 bit $\quad x_i = 32, 29, 27, 25, 23, 19$

$$\text{Range} = \prod_{i=1}^{6} x_i \approx 2.7 \times 10^{8}$$

Figure 2.7. Six and five bit coding.

If an increased range R must be represented, these 5- and 6-bit moduli sets can be increased to a large number of moduli.

Direct table look-up addition, subtraction, and multiplication using modular numbers representable by 5-, 6-, and 7-bit binary codes require table look-up memories of

5-bit codes, memory = $2^{10} \times 5$ bits, range $\approx 2.7 \times 10^{8}$
6-bit codes, memory = $2^{12} \times 6$ bits, range $\approx 3.4 \times 10^{10}$
7-bit codes, memory = $2^{14} \times 7$ bits, range $\approx 1.7 \times 10^{12}$

In the United States (and Japan), 5- and 6-bit arithmetic is readily feasible using existing ROMs, PROMs, and RAMs. Seven-bit implementations are still possible but somewhat more difficult with available LSI chips. VLSI is, of course, rapidly giving increased memory sizes which will ease the memory size problem. Soviet authors write that modular arithmetic makes table methods very attractive and, in particular, are partial to "optical holographic" digital memories for use in modular arithmetic computers.

Akushskiy (1968 & 1970) observes that the table for multiplication can be reduced in size by exploiting the symmetries that occur. For example, he notes that the occurrence of zero for either operand α_i or β_i with resultant zero output can be detected separately and that a renumbering of the remaining tables can be presented as shown in Fig. 2.8.

$$\begin{array}{cccc|ccc}
 & & & & & \alpha_1 & \\
 & & & & 6 & 5 & 4 \\
 & & & & 1 & 2 & 3 \\
\hline
 & 6 & 1 & & 1 & 2 & 3 \\
\beta_1 & 5 & 2 & & 2 & 4 & 6 \\
 & 4 & 3 & & 3 & 6 & 2
\end{array}$$

Figure 2.8. Reduced multiplication table for modulo 7 multiplication.

This modification reduces the table look up size by a factor of 4, as illustrated in Fig. 2.8. He also suggests that an additional factor of 2 can be achieved by exploiting the symmetries that occur about the principle diagonals of the multiplication tables.

Akushskiy also observed that for addition, a different approach involving a type of "ternary" encoding of the bits of α_i and β_i are the binary encoding of the modular numbers. He finds that instead of 2^{2n} table entries for n bit numbers, the number of table entries can be reduced to less than 3^n.

In order to use the table size reduction techniques for both multiplication and addition, a special encoding is described which facilitates the addressing of the reduced look-up tables. This encoding is illustrated below for modulo 7 encoding:

Digit	ν	α_i	Digit	ν	α_i
1	0	01	4	1	11
2	0	10	5	1	10
3	0	11	6	1	01

Recovery of the normal binary weighted code can be achieved by inverting the digits of α_i when $\nu = 1$ and adding an appropriately weighted number. The weight of ν is

$$G_\nu = p_i + 1 - 2^{n-1}$$

In our example, since $p_i = 7$ (modulus 7) and $n = 3$ for 3-bit binary,

$$G_\nu = 7 + 1 - 2^{3-1} = 4$$

Therefore, binary 4 is added to the inverted bits of α_i for those table entries in which $\nu = 1$.

digit 4 $\nu = 1, \alpha_1 = 11$, becomes binary 100

Thus, digit 5 $\nu = 1, \alpha_1 = 10$, becomes binary 101

digit 6 $\nu = 1, \alpha_1 = 01$, becomes binary 110

As a result of the special coding, it is easy to address the reduced multiplication table in table look-up operations. It is also easy to convert to normal binary coded numbers for use in implementing addition or subtraction. Much emphasis is given in Soviet writing on the suitability of table look-up methods for doing modular arithmetic which suggests that a lot of work has been done to improve the hardware implementation aspects.

Akushskiy also discusses methods for implementation of modular addition and multiplication using modifications of ordinary binary adders. In this method, the modular numbers are first added as in normal binary, and then a correction term is added to the result. The correction term when needed is conditional, based on the overflow bit from the binary adder. Once modular adders for the necessary moduli are available, repeated addition and correction operations will yield the modular products. It is difficult to tell whether this method has been implemented in the Soviet hardware, but it appears very interesting for implementation of modular arithmetic in VLSI.

Two other methods for reducing the size of tables needed in modular arithmetic units are discussed by Soviet authors. In one case, a so-called two-stage system is proposed in which, for example, if arithmetic modulo 127 is needed it can be carried through arithmetic using several smaller moduli; that is, moduli 5, 7, 8 would be sufficient for addition of numbers modulo 127, since the largest value of the sum would be $(2\ p_i - 2) = 252$ and the range of a system modulo 5, 7, 8 is $5 \times 7 \times 8 = 280$. If multiplication is necessary, the maximum range is $(p_i-1)^2 = (126)^2 = 15{,}876$ and it is therefore necessary to include more moduli for the second level; that is, moduli 5, 7, 8, 9, 11 which would give a range of $5 \times 7 \times 8 \times 9 \times 11 = 27{,}720$. This method for reduction of table size for table look-up hardware looks very efficient from the table-size point of view, since it reduces the required memory size by orders of magnitude. However, between each add or multiply it is necessary to correct the set of residues in the second level of moduli. This problem is discussed by Ahushkiy, but details of the implementation are not understood (by the authors of this report) at present.

Address	Cont	Address	Cont	Address	Cont
000000	X000	010101	X011	101010	X011
000001	X000	010110	X101	101011	X001
000010	X000	010111	XXXX	101100	X110
000011	X000	011000	X000	101101	X100
000100	X000	011001	X011	101110	X010
000101	X000	011010	X110	101111	XXXX
000110	X000	011011	X010	110000	X000
000111	XXXX	011100	X101	110001	X110
001000	X000	011101	X001	110010	X101
001001	X001	011110	X100	110011	X100
001010	X010	011111	XXXX	110100	X011
001011	X011	100000	X000	110101	X010
001100	X100	100001	X100	110110	X001
001101	X101	100010	X001	110111	XXXX
001110	X110	100011	X101	111000	XXXX
001111	XXXX	100100	X010	111001	XXXX
010000	X000	100101	X110	111010	XXXX
010001	X010	100110	X011	111011	XXXX
010010	X100	100111	XXXX	111100	XXXX
010011	X110	101000	X000	111101	XXXX
010100	X001	101001	X101	111110	XXXX
				111111	XXXX

Table 2.1: Modulus 7 Multiplication

Address	Cont	Address	Cont	Address	Cont
000000	X000	010101	X000	101010	X000
000001	X001	010110	X001	101011	X001
000010	X010	010111	XXXX	101100	X010
000011	X011	011000	X011	101101	X011
000100	X100	011001	X100	101110	X100
000101	X101	011010	X101	101111	XXXX
000110	X110	011011	X110	110000	X110
000111	XXXX	011100	X000	110001	X000
001000	X001	011101	X001	110010	X001
001001	X010	011110	X010	110011	X010
001010	X011	011111	XXXX	110100	X011
001011	X100	100000	X100	110101	X100
001100	X101	100001	X101	110110	X101
001101	X110	100010	X110	110111	XXXX
001110	X000	100011	X000	111000	XXXX
001111	XXXX	100100	X001	111001	XXXX
010000	X010	100101	X010	111010	XXXX
010001	X011	100110	X011	111011	XXXX
010010	X100	100111	XXXX	111100	XXXX
010011	X101	101000	X101	111101	XXXX
010100	X110	101001	X110	111110	XXXX
				111111	XXXX

Table 2.2: Modulus 7 Addition

3.0 FURTHER DEVELOPMENTS IN RESIDUE ARITHMETIC

The purpose of this section is to develop a background in the mathematics relevant to residue arithmetic and its application to computers.

3.1 Residue Arithmetic

As typically proposed, one selects a large list of distinct prime numbers $p_1, p_2, \ldots, p_k$, and in order to add or multiply two integers, one adds or multiplies their respective residues modulo each of those primes to get the description in terms of residue classes of the sum or product (for details see section 2.0). The latter uniquely characterizes the sum or product, up to the ambiguity of adding an integral multiple of $P = p_1, p_2, \ldots, p_k$. If P is adequately large, there is no particular difficulty at this juncture. If there are a very large number of arithmetic operations, then any ambiguity can be awkward and has to be resolved by some internal procedure sensitive to overflow.

If, in advance, we have a rough idea of the magnitude of the end quantity we are seeking, the overflow in the course of a computation is no problem at all—it all comes out in the wash at the end. So for ordinary arithmetic, where there is no concern about overflow, residue arithmetic is a perfectly satisfactory procedure.

By ordinary arithmetic, we mean addition and multiplication. As for subtraction, it is just multiplication by -1 followed by addition. Division, in general, is not easily possible, and questions about the relative magnitude of two numbers are quite difficult to handle when the numbers are described by their residue classes. There is a scheme to handle this, within the capacity of a computer, that does only residue arithmetic, called "conversion to mixed radix system" (see section 2.0 and Szaba & Tanaba, 1967). This conversion is relatively costly in terms of hardware implementation, but with its availability, general division, scaling, and overflow detection can be handled.

Western computing technology has grown up being driven largely by a fixation with ordinary or Archimedean arithmetic. Given the enormous strides in chip and related technologies, it will probably persist in these directions, largely because of the capital investment already in place. While there has been a substantial theoretical investigation (Knuth, 1969) of the principles of modular arithmetic as applicable to computer design, the decisions not to go down that avenue were probably inevitable a long time ago (Nussbaumer, 1981).

This is by no means to suggest that a technologically informed society, starting out afresh on the problems of computer design and conversant with the principles of residue arithmetic, might not succeed in overcoming some of the serious problems attendant on its use, and develop in whole or in part, a perfectly acceptable format for computation along these altogether different lines.

Residue arithmetic does have some aspects which are peculiar to its setting. These appear to be the object of significant investigation by the Soviets, and in the remainder of this section we will attempt to describe what we think they are up to, with the appropriate mathematical underpinnings.

3.2 Application of Residue Arithmetic to Complex Numbers

One of the Soviets' major concerns seems to be the applicability of the ideas of residue arithmetic to complex numbers. With ordinary residue arithmetic in hand, one can, of course, carry out residue arithmetic on complex numbers simply by treating the real and imaginary parts separately, as we conventionally do. There is, however, a much better way of proceeding which enables us to regard the complex number as a single entity, rather than made up of real and imaginary parts. This other way depends on the theory of factorization of complex "integers," developed long ago by Gauss (MacLane, 1980) and perhaps not as well known in the computer community as it should be.

The objects with which we shall be dealing are the Gaussian integers m + ni, where m and n are ordinary integers and $i^2 = -1$. The theory of factorization of Gaussian integers parallels, in the main, the corresponding statements for ordinary integers. Thus, every Gaussian integer may be factored uniquely into a product of Gaussian primes, up to the ambiguity of the Gaussian units ± 1, $\pm i$. And if a Gaussian integer is divisible by two relatively prime, i.e., having no common factor, Gaussian integers, it is divisible by their product. Some Gaussian integers and their residue representations are listed in Table 4.1 of section 4.0 below.

The first question, then, is what are the Gaussian primes? They are described as follows:

(1) A rational prime p of the form 4n + 3 is also a Gaussian prime.

(2) A rational prime p of the form 4n + 1 factors $p = (a + bi)(a - bi)$, and a + bi and a − bi are distinct Gaussian primes not differing by multiplication of a Gaussian unit. Note that $(b + ai) = (a - bi)i$, so b + ai differs from

$a - bi$ by only a unit multiplication. We note, for example, that $5 = (2 + i)(2 - i)$.

(3) The rational prime 2 splits, $2 = (1 + i)(1 - i)$, but $(1 - i) = (1 + i)i$, so we get only one new Gaussian prime, $1 + i$.

This completes the description of Gaussian primes. A list of some Gaussian primes is given in Table 4.2 in section 4.0 below.

For a rational prime, say 7, the residue classes are typically described by 0, 1, 2, . . . , 6. What then, is a description of the residue classes for a Gaussian prime? For a Gaussian prime that is an ordinary prime of the form $p = 4n + 3$, there are p^2 residue classes, one for each choice of the residue class of real and imaginary parts. Multiplication of residue classes then involves the usual separation into real and imaginary parts, with the attendant extra hardware for computer implementation.

For a Gaussian prime of the form $a + ib$, where $a^2 + b^2 = p$, an ordinary prime of the form $4n + 1$, the residue classes are easily seen to be described by the numbers $0, 1, 2, . . . , p - 1$, with addition and multiplication of residue classes takes modulo p as for the usual real residue arithmetic described in section 2.0. The residue classes for the prime a - ib are the same, so the same tables used for arithmetic look-up mod (a + ib) suffice for look-up mod (a - ib). The pairing of the primes a + ib and a - ib has several advantages beyond the one just noted. If we know, for example, the residues of the Gaussian integer (m + in) modulo both of a + ib and a - ib, then we automatically know the residues of (m - in), the complex conjugate. Hence, we know the residues of 2m and 2n, as well as the residue of $m^2 + n^2$. Additionally, if we know the residues of 2m and 2n, we can find easily the residue of m and n, the real and imaginary parts.

3.3 Implementation of Residue Arithmetic Using Complex Numbers

Actual implementation of residue arithmetic for complex numbers might be achieved as follows. Pick a suitably large set of rational primes $p_1, p_2, \ldots, p_k$, all congruent to 1 modulo 4. Set $P = p_1, p_2, \ldots, p_k$. Set $p_v = (a_v + ib_v)(a_v - ib_v)$. We will carry out residue arithmetic with the set of Gaussian primes $a_v + ib_v$ and $a_v - ib_v$. For a given Gaussian integer it is relatively straightforward to find its residues for each of the Gaussian primes in our list. The procedure is as follows. Given a Gaussian prime a + ib, let a^{-1} denote the inverse of "a" modulo $p = a^2 + b^2$. Then the residue of m + in modulo a + ib is the residue modulo p of $m + a^{-1}bn$. Each Gaussian integer lying in the square centered at the origin of side P - 1 is uniquely identified by its residue modulo each of the Gaussian primes in our list. P must be suitably large enough so that overflow into adjacent boxes is not a problem. It is worth noting, however, that if the answer we seek is known, in advance, to lie in the square above, then overflow in the course of a long computation does not affect the validity of the answer. With the residues of (m + in) all known, there is a Chinese Remainder Theorem enabling us to compute (m + in), but this calculation must be carried out in Archimedean arithmetic. However, by using the device known as the mixed radix representation, a large part of the calculation can be performed on the residue arithmetic computer (see section 2.1.7).

It is not as well known as it should be that the mechanisms of circular convolution and fast Fourier transform can be carried out at the level of residue arithmetic. The available literature, such as Nussbaumer (1981), seems to suggest that these devices are exploitable only for very special choices of primes and length of circle. The truth of the matter is, however, that these computational ploys may be exploited with only mild restrictions for arbitrary primes and circles, both for real and complex residue arithmetic. This fact strongly suggests the broad applicability of residue arithmetic to signal processing.

3.4 Further Comments on Reducing the Size of Look-up Tables

The Soviet literature also introduces a clever device for carrying out real residue arithmetic modulo a large prime in terms of residues modulo a collection of smaller primes. Their device is particularly effective in reducing look-up table sizes if one is concerned only with addition. In principle, the device also works for multiplication, but we have not seen how a really substantial savings in table look-up size can be realized in this case (see section 2.0). We do want to note in passing that the device for real residue addition works equally well for complex residue addition.

Perhaps because Soviets are aware of the problem with multiplication, or perhaps because they are exploring all possibilities, they have described an alternative procedure for saving memory space in case of multiplication, which we now consider. The basic idea is simply that even in residue arithmetic there is a satisfactory notion of logarithm. Let p be an ordinary prime. For our purposes, we first suppose we have a satisfactory procedure for labeling zero, and doing multiplication of a residue class by zero. This is, after all no great task. So we will concentrate on multiplication of two non-zero residue classes modulo p. The essential fact is that the non-zero residue classes modulo p form a cyclic group. That is, there is a residue α (not unique) such that every residue class is uniquely of the form α^k for some integer k, $0 \le k \le p - 1$. α shall be picked once and for all for the prime p and then every non-zero residue class is uniquely labeled by its logarithm k. If one residue has a logarithm k and a second k' then their product has logarithm $k + k' \pmod{p - 1}$. Multiplication is thus reduced to addition modulo $p - 1$. And the addition can be handled by factoring $p - 1$ into relatively prime factors and adding up residues modulo each of these factors. If some of the factors are large, which would require a large look-up table, then the second-stage residue device alluded to earlier can additionally be applied.

The procedure for multiplying described just above works equally well for complex residue arithmetic and, in fact, gives us some additional freedom we did not have previously. Let p be a ordinary prime of form $4n + 1$. Then p factors $p = (a + ib)(a - ib)$, and the residue classes of Gaussian integers modulo $(a + ib)$ may be selected as residue classes of integers modulo p; that is, $0, 1, 2, \ldots, p - 1$. Then as was the case for real residue arithmetic, the non-zero residues have a logarithm, and multiplication is reduced to addition modulo $p - 1$.

But additionally, if p is an ordinary prime of form $4n + 3$, then p is a Gaussian prime; and the residue classes modulo p fall into p^2 classes, the residues separately of real and imaginary part. If we throw away zero, the remaining $p^2 - 1$ residue classes form a cyclic group, so there is a residue class α (of Gaussian integers) modulo p so that every non-zero residue class is uniquely of form α^k, $0 < k < p^2 - 1$, k now playing the role of logarithm. The log of a product is the sum of the logs modulo $p^2 - 1$, so multiplication is reduced to addition modulo $p^2 - 1$. Moreover, the use of logarithms does not require us to multiply by considering the real and imaginary parts separately. Hence, from this point of view, primes of the form $4n + 3$ are just as good as primes of the form $4n + 1$, and for the former type, a large number of factors of $p^2 - 1$ might make them especially desirable.

4.0 USE OF GAUSSIAN RESIDU ARITHMETIC

4.1 Introduction

It is not difficult to find residues of a given Gaussian integer $(m + ni)$ for each Gaussian prime $(a + bi)$. Let the residue be x, and let the symbol % represent "modulo." Now $x = (m + ni) \% (a + bi)$ is the desired result. Thus,

$$x(a - bi) = (m + ni)(a - bi) \% (a + bi)(a - bi)$$
$$xa - xbi = [(am + bn) + (an - bm)i] \% p$$

where $p = a^2 + b^2$. Let $a^{-1}a \% p = 1$, that is, a^{-1} is the multiplicative inverse of a, taken module p. Then $xaa^{-1} = x = (aa^{-1}m = a^{-1}bn) \% p$, or $x = (m + a^{-1}bn) \% p$. For any given p and particular choice of root, $a^{-1}b$ is, of course, unique, so it may be designated as $a^{-1}b = k_p^+$ (and for the other root, let $k_p^- = a^{-1}b$), so that $x = (m + k_p^{\mp}n) \% p$. Conversion from the Gaussian numbers to their residue representation is therefore very simple. For example, we compute $(1 + 2i) \% (3 + 2i)$:

$a = 3$

$b = 2$

$p = 13$

$a^{-1} = 9$ (since $3.9 \% 13 = 27 \% 13 = 1$)

$k_{13}^{+} = 9.2 \% 13 = 18 \% 13 = 5$ and $k_{13}^{-} = 8$

then

$x^{+} = (m + 5n) \% 13$ and $x^{-} = (m + 8n) \% 13$

$x^{+} = (1 + 10) \% 13 = 11$ and $x^{-} = (1 + 16) \% 13 = 4$

$$\underline{x^{+} = 11 \text{ and } x^{-} = 4}$$

The conversion from Gaussian residue to Gaussian integer form is accomplished in the usual way using the Chinese Remainder Theorem or via a mixed radix representation. It is much more difficult than the conversion into residue form (see section 2.0).

4.2 Example of Gaussian Residue Arithmetic

To illustrate Gaussian residue arithmetic, consider representing Gaussian integers in residues modulo (2 + i), (2 - i), (3 + 2i), and (3 - 2i). The conversion is illustrated in Table 4.1. It is derived as described above. Now using the table, addition is demonstrated.

TABLE 4.1

Residue Representation of Gaussian Integers

		Gaussian Residue Representation			
	Gaussian Integers	p = 5		p = 13	
	(real + imag i)	%(2 + i)	%(2 - i)	%(3 + 2i)	%(3 - 2i)
	m + ni	(m + 3n)%5	(m + 2n)%5	(m + 5n)%13	(m + 8n)%13
	0 + 0i	0	0	0	0
	0 + 1i	3	2	5	8
+	0 + 2i	1	4	10	3
	0 + 3i	4	1	2	11
	0 + 4i	2	3	7	6
	0 + 5i	0	0	12	1
	0 + 6i	3	2	4	9
	0 + 7i	1	4	9	4
	0 + 8i	4	1	1	12
	0 + 9i	2	3	6	7
	0 + 10i	0	0	11	2
	0 + 11i	3	2	3	10
	0 + 12i	1	4	8	5
	0 - 12i	4	1	5	8
	0 - 11i	2	3	10	3
	0 - 10i	0	0	2	11
	0 - 9i	3	2	7	6
	0 - 8i	1	4	12	1
	0 - 7i	4	1	4	9
	0 - 6i	2	3	9	4
	0 - 5i	0	0	1	12
	0 - 4i	3	2	6	7
	0 - 3i	1	4	11	2
	0 - 2i	4	1	3	10
	0 - 1i	2	3	8	5
	1 + 0i	1	1	1	1
+	1 + 1i	4	3	6	9
+	1 + 2i	2	0	11	4
	•	~	~	~	~
	•	~	~	~	~
	•				
	-3 -2i	1	3	0	7
	-3 - i	4	0	5	2
	-2 0i	3	3	11	11
+	-2 +1i	1	0	3	6
	-2 +2i	4	2	8	1
	•				
	•	~	~	~	~

TABLE 4.1 (Cont'd)

Gaussian Integers (real + imag i) m + ni		Gaussian Residue Representation p = 5 %(2 + i) (m + 3n)%5	%(2 - i) (m + 2n)%5	p = 13 %(3 + 2i) (m + 5n)%13	%(3 - 2i) (m + 8n)%13
-2	-2i	2	4	1	8
-2	-1i	0	1	6	3
-1	0i	4	4	12	12
-1	1i	2	1	4	7
-1	2i	0	3	9	2
→ -1	3i	3	0	1	10
•					
•		~	~	~	~
-1	-2i	3	0	2	9
-1	-1i	1	2	7	4

Gaussian Residue Addition:

	Module	(2 + i)	(2 - i)	(3 + 2i)	(3 - 2i)
(1 + 2i)	→	2	0	11	4
+ (-2 + i)	→	+1	+0	+3	+6
= (-1 + 3i)	→	3	0	1	10
		%5		%13	

Similarly, a multiplication example is illustrated below:

$$\begin{array}{r c cccc}
 & \text{Module} & (2+i) & (2-i) & (3+2i) & (3-2i) \\
\hline
(1+i) & \rightarrow & 4 & 3 & 6 & 9 \\
\underline{\times(1+i)} & \rightarrow & \times 4 & \times 3 & \times 6 & \times 9 \\
\hline
=(0+2i) & \rightarrow & 1 & 4 & 10 & 3 \\
 & & \multicolumn{2}{c}{\underbrace{\qquad\qquad}_{Z5}} & \multicolumn{2}{c}{\underbrace{\qquad\qquad}_{Z13}}
\end{array}$$

Assume we wish the residue of the magnitude squared of a Gaussian integer:

$$\begin{array}{r c cccc}
 & \text{Module} & (2+i) & (2-i) & (3+2i) & (3-2i) \\
\hline
(1+i) & \rightarrow & 4 & 3 & 6 & 9 \\
\underline{\times(1-i)} & \rightarrow & \times 3 & \times 4 & \times 9 & \times 6 \\
\hline
=(2+0i) & \rightarrow & 2 & 2 & \times 2 & \times 2
\end{array}$$

Note that since we know <u>a priori</u> that the imaginary part is zero, we need only do one of the indicated modular multiplications since the results will be identical for all the modular multiplications.

In a similar manner, extracting the real or imaginary parts is a straightforward task:

$$\begin{array}{r c cccc}
(1+i) & \rightarrow & 4 & 3 & 6 & 9 \\
\underline{+(-1+i)} & \rightarrow & 2 & 1 & 4 & 7 \\
\hline
(0+2i) & \rightarrow & 1 & 4 & 10 & 3
\end{array}$$

Multiplying by $(2i)^{-1} \rightarrow (1, 4, 4, 9)$,

$(0 + 2i)$	1	4	10	3
$\times\ (0 + 2i)^{-1}$	x1	x4	x4	x7
$(1 + 0i)$	1	1	1	1

produces the desired imaginary part.

Archimedean and Gaussian primes up to 149 are listed in Table 4.2.

TABLE 4.2

Gaussian Primes Less Than 150

Archimedean Primes		Gaussian Primes	
1	(1)		(i)
2		(1 + i)	
3	(3)		(3i)
5	(2 + i)		(2 - i)
7	(7)		(7i)
11	(11)		(11i)
13	(3 + 2i)		(3 - 2i)
17	(4 + i)		(4 - i)
19	(19)		(19i)
23	(23)		(23i)
29	(5 + 2i)		(5 - 2i)
31	(31)		(31i)
37	(6 + i)		(6 - i)
41	(5 + 4i)		(5 - 4i)
43	(43)		(43i)
47	(47)		(47i)
53	(7 + 2i)		(7 - 2i)
59	(59)		(59i)
61	(6 + 5i)		(6 - 5i)
67	(67)		(67i)
71	(71)		(71i)
73	(8 + 3i)		(8 - 3i)
79	(79)		(79i)
83	(83)		(83i)
89	(8 + 5i)		(8 - 5i)
97	(9 + 4i)		(9 - 4i)
101	(10 + i)		(10 - i)
103	(103)		(103i)
109	(10 + 3i)		(10 - 3i)
113	(7 + 8i)		(7 - 8i)
127	(127)		(127i)
131	(131)		(131i)
137	(11 + 4i)		(11 - 4i)
139	(139)		(139i)
149	(10 + 7i)		(10 - 7i)

5.0 A SAMPLE PROCESSOR USING GAUSSIAN RESIDUE ARITHMETIC

One of the most intensively used algorithms in signal processing is the Fast Fourier Transform (FFT). In order to explore the potential of the new Gaussian residue methods discussed above, we have sketched out a design of a cascade FFT processor in the style of Despain (1974) but using Gaussian residue arithmetic.

The FFT calculates the discrete Fourier transform

$$Ar = \sum_{k=0}^{N-1} B_k W_N^{kr}$$

This expression can be converted to the FFT form of Cooley & Tukey (1965). Despain has worked out an efficient pipeline organization of the FFT that is well suited for our purposes.

The pipeline structure shown in Figures 5.1-5.6 is derived by Despain (1974) and consists of only the three modules shown in Fig. 5.1.

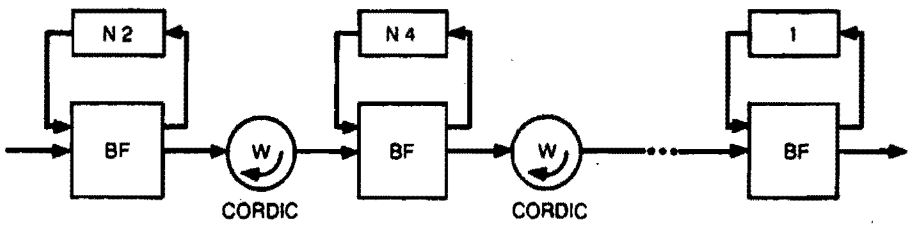


Figure 5.1. Despain Cascade

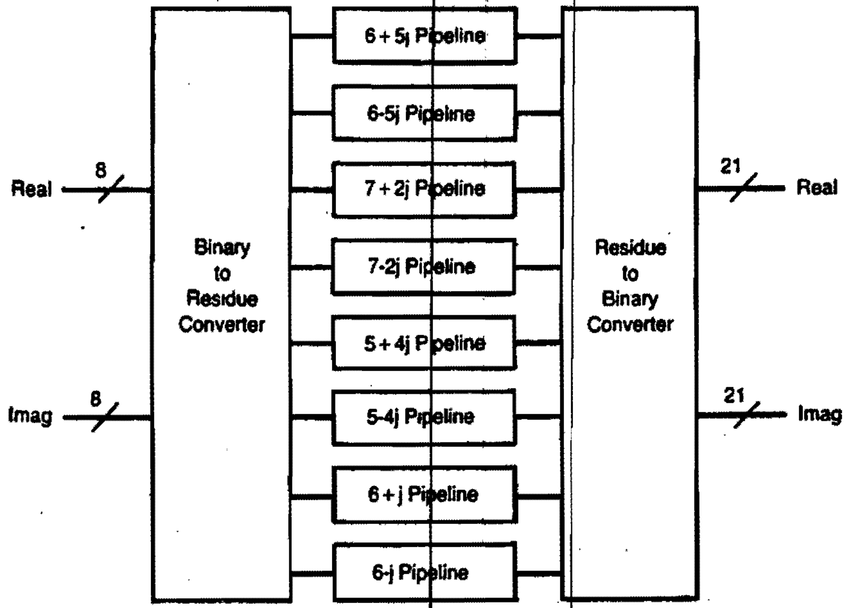


Figure 5.2. Residue FFT Cascade processor (N = 64)

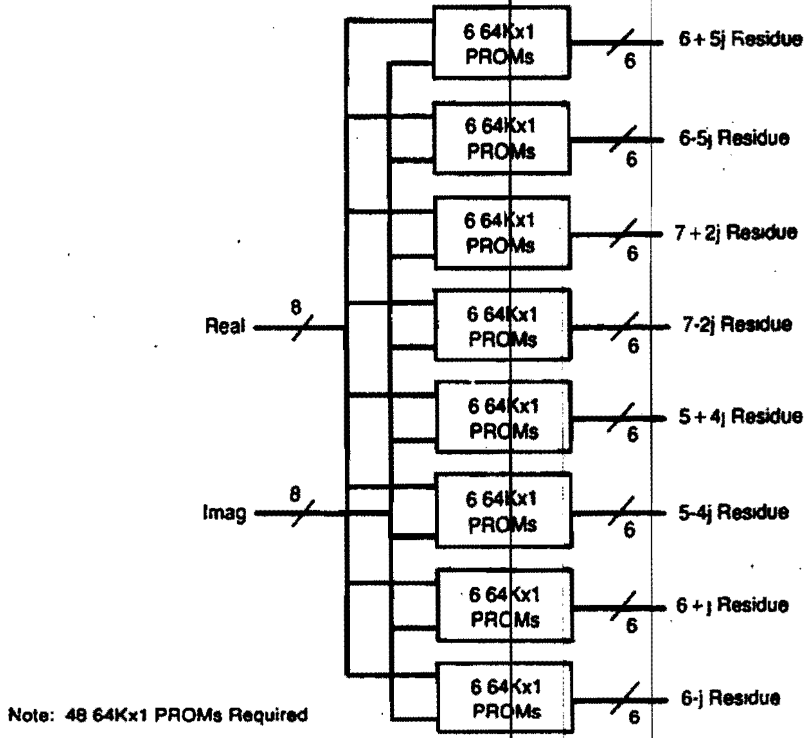


Figure 5.3. Binary to residue conversion

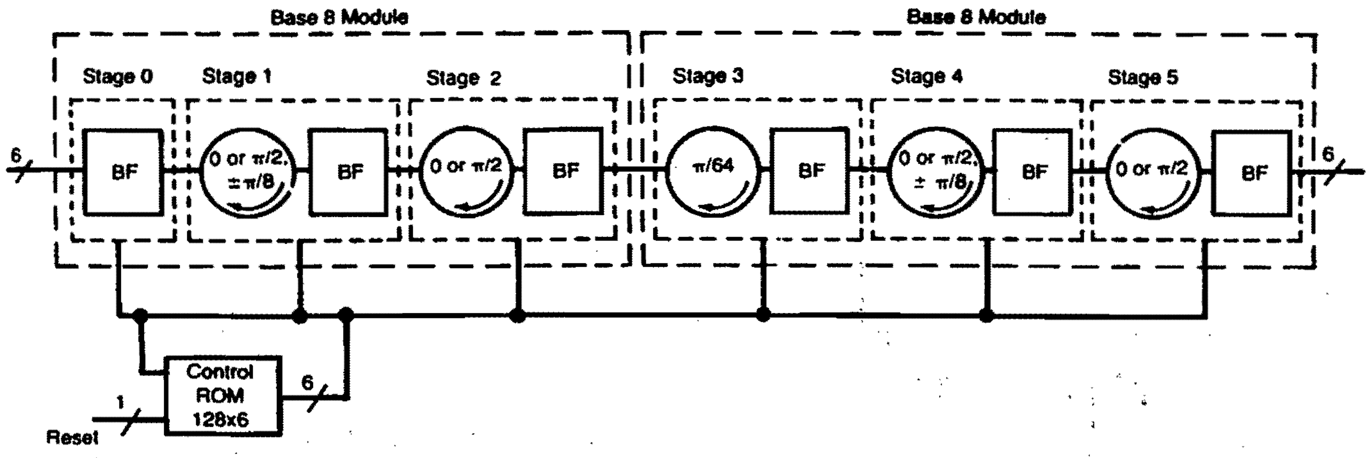
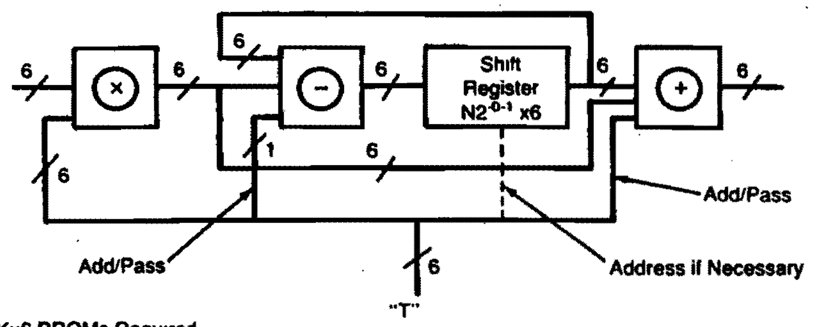
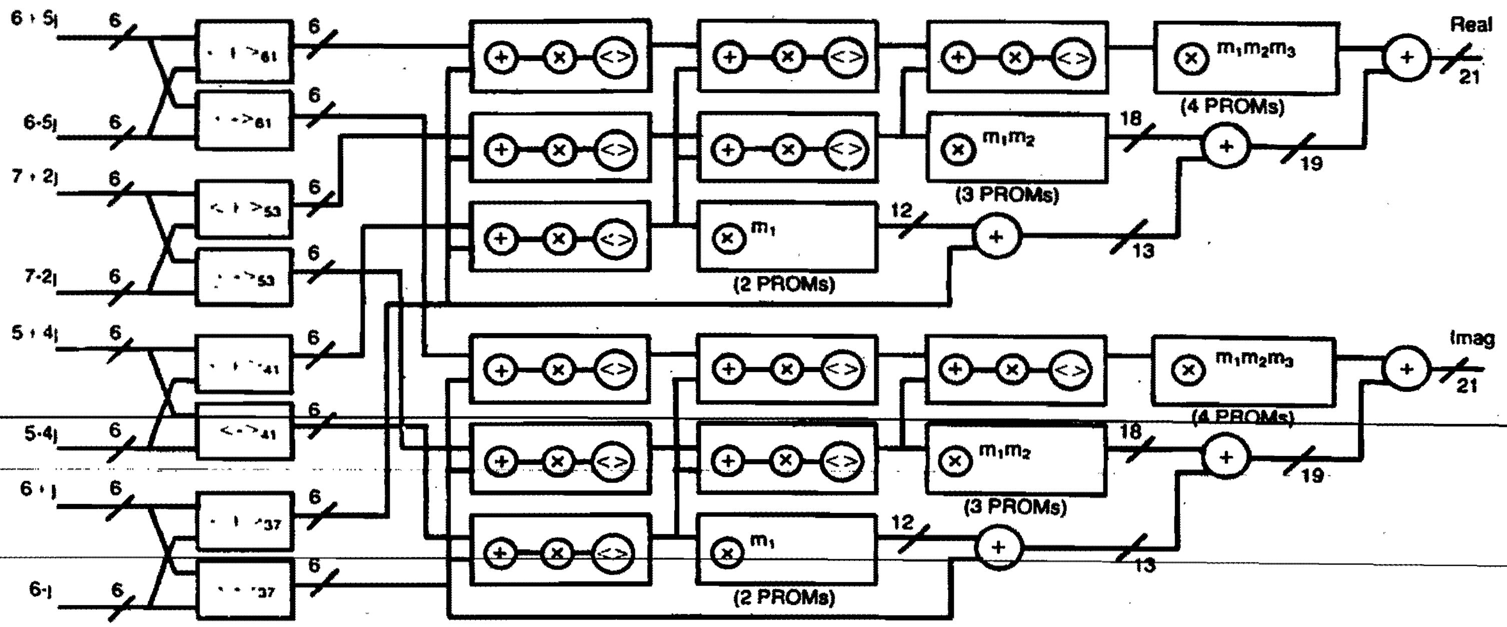


Figure 5.4. Residue pipe configuration



Note: 3 8Kx6 PROMs Required

Figure 5.5 Residue pipe stage m



Note: 38 4Kx6 PROMs and 6 Adders Required

Figure 5.6. Residue to binary conversion for FFT Cascade

Basically, the operation of the processor is as follows: The first butterfly (BF) module allows the input α_i to pass unchanged into the shift register of length 2^{n-1} until it is full. At that point, the incoming data and the data in the shift register are combined in a 2-point DFT:

$$\begin{aligned} b_i &= \alpha_i + \alpha_{i+\frac{N}{2}} \\ b_{i+\frac{N}{2}} &= \alpha_i - \alpha_{i+\frac{N}{2}} \end{aligned} \qquad i = 0, \ldots, \frac{N}{2} - 1$$

The b_i are sent to the CORDIC rotator module which applies the proper rotations (twiddle factors), while the $b_{i+\frac{N}{2}}$ are sent back into the shift register. When all $\frac{N}{2}$ 2-point DFT's have been computed, the $b_{i+\frac{N}{2}}$ are allowed to pass out of the shift register into the CORDIC rotator. The operation of the next butterfly module is similar, except that each input datum is combined with one $\frac{N}{4}$ away. The entire computation is completed after $n = \log_2 N$ stages, where one stage consists of the butterfly module, CORDIC rotator, and shift register.

The Gaussian residue technique is well suited to this calculation as it consists of only complex multiplications and additions. The input need only be converted into Gaussian residue form; all calculations can then be performed in this system. Then, if necessary, the numbers can be converted from the Gaussian residue to binary or decimal representation.

Because we wish to minimize multiplications because they lead to long results, we will convert the FFT cascade into a base -8 FFT cascade. This changes only the CORDIC rotators by specializing them. To obtain a FFT processor for $N = 64$, we then need a pair of base -8 pipelines, with a CORDIC rotator capable of performing a multiplication by all 64 roots of unity. This multiplication doubles the required number of bits needed to represent the signal. Assume we start with 8 bits. Since all of the computations in the residue representation must be done as integer operations, and since scaling is impossible, we must carry enough bits of precision so that we can represent our answers exactly, even though there is only a 1 bit increase in real precision per stage, which means that at most $8 + \log_2 64 = 14$ bits would be meaningful. Of course, we would right away realize that we could not represent the 64th roots of unity exactly, and would round these off at 9 bits (since they are applied after the 3rd stage). However, this would imply that the central twiddle factor multiply and the two non-trivial twiddle multiplies internal to the base 8 pipelines would require an extra $9 + 11 + 12 = 32$ bits, bringing the total to $14 + 32 = 46$ bits. This would be inefficient and costly.

However, there are ways around this problem. For this example, we will set our sights a little lower and derive a processor which will guarantee 6 bit precision in the output. The methods derived in Despain (1979) allow us to use small integer approximations to the rotations at the cost of introducing a constant complex gain into the final results. We achieve this by using a $\pm \frac{\pi}{8}$ rotation internal to the base 8 module. To minimize the number of bits which need to be carried through the computation, we need to find complex integers of the form $x + yj$ whose arguments approximate the angles we wish to rotate by to the accuracy desired, and for which x and y are small. Multiplication by an $x + yj$ will perform the desired rotation and will introduce a known, correctable gain. To rotate by the same angle in the opposite direction, we multiply by $x - yj$. A computer program was written to search for these integers and produced the results shown in Table 5.1. Some small integer approximations to other useful angles are also included. From the table, we choose $2 \pm j$ for the $\pm \frac{\pi}{8}$ rotators for 6 bit accuracy.

For the central rotation between the base 8 sections, we must find small integer approximations to the rotations $n \frac{\pi}{32}$ $n = 0, 1, \ldots, 8$ with the added constraint that the gain introduced by all of these rotations must be a constant to within 6 bit accuracy. Again, a computer program was written to search for integers with these characteristics. The results are shown in Table 5.2.

TABLE 5.1

Small Complex Integer Approximations to Rotations

angle	x	y	accuracy(bits)
$\pi/8$	2	1	6.4
	5	2	9.0
	12	5	11.5
	29	12	14.0
	70	29	16.6
$\pi/16$	4	1	7.0
	5	1	12.5
	111	22	13.1
	136	27	14.0
	156	31	15.0
	171	34	16.2
$\pi/32$	1	0	6.0
	7	1	7.1
	9	1	8.9
	10	1	12.0
	51	5	13.7
	61	6	15.5
	132	13	19.8

TABLE 5.2

6 bit approximations to $n\frac{\pi}{32}$ with gain of 14

n	x	y
0	14	0
1	14	2
2	14	2
3	13	5
4	13	5
5	12	7
6	12	7
7	11	9
8	10	10

To determine the number of bits we now require to avoid overflow, we now need to find the maximum gain due to multiplying by the set of numbers we have just chosen. For the $\pm \frac{\pi}{8}$ rotations, the gain of $2 \pm j$ is 2.236. Thus the dynamic range requirement due to multiplies is given by the product of this gain and the gain for the central multiply. In our case, this product is equal to 70, which corresponds to a 6.2 bit gain. Thus, we need a dynamic range of $6 + \log_2 64 + 6.2 = 18.2$ bits in all. From Table 4.2 we choose the primes 61, 53, 41, and 37 whose product is 4.9×10^6 which allows a dynamic range of 22 bits. We need the primes to be less than 2^6 since our PROMs have only 13 bits of address and two data inputs are required for an add.

This design requires approximately 200 chips to realize a 6 bit, 64 point FFT processor with a processing rate of 40 million complex samples per second. Due to the high chip count, this is not an especially attractive design. However, despite the large chip count, the latency is small and thus the design could be useful in critical applications.

6.0 SYMBOLIC COMPUTING

6.1 Introduction

The basic idea of symbolic computing is to compute "smart" rather than "fast." A computing problem can generally be solved in a number of ways. At one extreme, a simple algorithmic formulation will be employed by the programmer, and the machine will compensate with massive numerical calculations. At the other extreme, the simple formulation will be subjected to extensive symbolic reduction in order to reduce all numerical calculation as much as possible. What is being suggested here is that the computing machine can and will be employed to do the symbolic reduction that is usually manually performed. This has great potential in very complex situations, although human manual analysis often produces superior results in less complex cases. An example of this phenomena occurs in everyday computer programming practice. It is well known that any clever human programmer can produce more efficient machine code than can any optimizing compiler, provided that the programming task is not too complex. When the complexity of the programming task is large, however, optimizing compliers produce superior results.

The critical idea is this: For large, complex calculations, machine symbolic manipulation of the programming task might be able to radically improve the program. In fact, what is envisioned is that the statement of the computing task would be submitted in mathematical form to a symbolic manipulating system. The system would then process this statement of the computing problem, formulate it, reduce it, and create an efficient program. The program then would be compiled, optimized, and executed in the usual fashion. Because extensive symbolic manipulation has been applied, radical speed improvements might result. At present, there are no such systems (at least in the West).

6.2 History

The idea of employing an automatic machine to manipulate symbols was credited by Pavelle et al. (1981) to Babbage and Lovelace. Of course, Babbage was the first to propose, in 1812, automatic numerical calculation. While Babbage did build and demonstrate numerical computers, he and Lady Lovelace only vaguely speculated on the possibility of building a machine to manipulate symbols and to play chess. (Machine chess requires, of course, symbolic manipulations.) A simple machine, constructed in about 1890, to mechanically play chess endgames was probably the first machine to automatically manipulate symbols. However, its symbol-manipulating capabilities were extremely limited.

The use of a computer to automatically and symbolically manipulate a numerical computer program to improve it was probably first employed in compilers. Compilers optimize programs by rearranging formulas, making use of common subexpressions, and so on. The result of even quite simple optimizations is sometimes dramatic even though only very simple symbolic operations are employed.

Computer programs designed specifically to manipulate and solve mathematical equations began to appear in the early 1960s. For example, Slagle (1963), in his doctoral dissertation work, created a program named SAINT to solve elementary symbolic integration problems at approximately the level of college freshmen. There are now many symbolic manipulation systems. The most developed is MACSYMA (Martin & Fateman, 1971). None of these systems is yet capable of providing the radical improvements in performance we seek.

6.3 Language Issues

Classical mathematical notation is not the ideal form to express the original calculation problem. Current work in computer language theory indicates that an ideal language would be more formal and precise than mathematical notation but would retain the idea of statements with well-defined mathematical properties.

Backus (1978), the father of FORTRAN, has extensively examined this problem and has proposed the new language "fp." Current research indicates "fp" will be advantageous for many kinds of programming tasks, especially symbolic manipulation, but "fp" has a significant disadvantage in the awkwardness of its expression.

PROLOG is a language born in France, raised in England, and adopted by Japan for its "Fifth Generation" computer development program (Kowalski, 1979). It was created to be efficient in "logic programming." As such, it is well adapted to symbolic manipulation, especially for theorem proving, and so on. It seems too weak in the numerical calculation area to be suitable for the type of "radical computing" we have in mind.

It appears that some new computer language will be needed. It might include, for example, the familiar expressive style of classic mathematical notation as accepted by MACSYMA, the formal properties of the expression of "fp," and the logical calculus features of PROLOG. There is much work to be done in this area.

6.4 Current Development of Symbolic Manipulation Programs

Modern symbolic manipulation programs have impressive capabilities. They can solve complex mathematical systems that are well beyond the capabilities of almost all programmers. They do not yet compete with first-class mathematicians except in their ability to handle simple, but massively tedious symbolic calculations, for example, manipulations of large polynomials. As time goes on, more and more mathematical knowledge is being incorporated into these systems, so significant improvements are expected.

One of the most exciting developments in symbolic manipulation programs is the capability of these systems to produce, automatically, computer programs themselves, whenever extensive numerical calculations are needed. Appendix A.1 contains an example of such a program produced by MACSYMA to calculate numerical answers to a partial differential equations problem that had been formulated, manipulated, and reduced symbolically. The form of the program is a FORTRAN function that represents the heart of the calculation that will provide the numerical results. Note that this function is non-trivial and would have required many man-hours of programming effort if it had been produced by hand. A further feature is that programmer-produced errors are avoided, and debugging is no longer necessary.

During the next few years, one can expect new approaches to symbolic manipulation systems which will displace the older systems that have grown obsolete. SAINT was replaced by MACSYMA, for example. One such newly developed system is SMP (Cole and Walham, 1981). It is just now starting to be distributed to outside users, so it is too early to see if it is sufficiently innovative to displace MACSYMA. SMP is written in the language "C," a very popular language used for an exceptionally large class of applications, and available on all sized machines from microcomputers to maxicomputers. MACSYMA, on the other hand, is written in a particular variant of LISP that is not generally available. As mentioned above, symbolic manipulation programs are beginning to produce programs themselves. SMP produces numerical evaluation programs written in this same language "C." MACSYMA produces its program in FORTRAN, a language efficiently compiled and executed on almost all computers, but which is less expressive than "C" and is more divorced from the hardware architecture of modern computers.

7.0 CONCLUSIONS

7.1 Residue Arithmetic

It would appear that some parts of the Soviet computational mathematics community are setting out with dogged tenacity and considerable skill to make residue arithmetic a working basis for computer architecture. As we have noted before, we in the West are probably already committed to Archimedean arithmetic. They, on the other hand, are starting out afresh, without the same intellectual or capital framework, to find a different path. It is not known just how interested the rest of the Soviet scientific community is in this alternate method. However, the reams of publications and tutorial material which enthusiasts are promulgating are a clear indication of their determination. They are obviously bringing to the attention of their engineering and computer communities a circle of ideas from number theory and algebra, albeit mathematically elementary, with which our computer scientists are not generally acquainted.

Some, though not all by any means, of the Soviet material shows considerable ingenuity in applying number theory ideas to computer design. It is quite conceivable that after a period of time, by dint of working hard and creatively on these concepts, that they will come up with computers every bit as good as ours, perhaps better for some purposes, poorer for others, but most strikingly, vastly different than ours. Indeed, they may have already achieved significant advances for certain classes of signal processing such as filtering or FFT.

7.2 Symbolic Manipulation

Symbolic manipulation of user "programs," or more accurately user specifications, could turn out to be the key to radical improvements in computing power. The field is rapidly developing, with major developments expected in both the East and the West.

The relative magnitude of the Soviet effort is impressive. Appendix A.2 demonstrates the extent of their work. Appendix A.3 contains a sample of this effort. Symbolic manipulation by computer fits well with the Soviet style of computer science, and mathematical traditions (e.g., Mordukhai-Bottovskoi's work--for reference, see item 78 in Appendix A.2).

7.3 Final Remarks

It seems unlikely that the Soviets have managed to find a new, radical approach to computing. However, they do seem to be deeply interested in the two critical subjects discussed in this report, residue arithmetic and symbolic computing. These two areas are of intense, current research interest, and it is likely that one or both will be an integral part of the next revolution in computing, with the Soviets actively participating in this revolution.

APPENDIX A.1

FORTRAN PROGRAM PRODUCED BY MACSYMA

The following FORTRAN was produced by MACSYMA. It represents the 'interloop' of a program to calculate the numerical solution of a set of partial differential equations.

```
      subroutine setmat(il,jl,kl,ild,jld,s,x,y,z,h1,h2,h3,cf112,cf121,c
     1   f122,cf123,cf132,cf211,cf212,cf213,cf221,cf222,cf223,cf231,cf23
     2   2,cf233,cf312,cf321,cf322,cf323,cf332)
c     ild,jld,kld are the dimensions of all arrays.
c     il,jl,kl give the size of the problem.
c     s is the input array of densities.
c     h1,h2,h3 are input differences.
c     x,y,z are the input arrays of coordinates.
c     quantities of the form cf followed by integers are
c     the output arrays.  Here we use the convention that
c     the difference form of the differential equation is
c     written as a sum over i1, j1 and k1 where
c     these indices run from 1 to 3 and each term in the sum
c     is of the form:
c           cfi1j1k1[i,j,k]*g[i-i1+2,j-j1+2,k-k1+2]
      real s(ild,jld,kl),x(ild,jld,kl),y(ild,jld,kl),z(ild,jld,kl),cf112
     1   (ild,jld,kl),cf121(ild,jld,kl),cf122(ild,jld,kl),cf123(ild,jld,
     2   kl),cf132(ild,jld,kl),cf211(ild,jld,kl),cf212(ild,jld,kl),cf213
     3   (ild,jld,kl),cf221(ild,jld,kl),cf222(ild,jld,kl),cf223(ild,jld,
     4   kl),cf231(ild,jld,kl),cf232(ild,jld,kl),cf233(ild,jld,kl),cf312
     5   (ild,jld,kl),cf321(ild,jld,kl),cf322(ild,jld,kl),cf323(ild,jld,
     6   kl),cf332(ild,jld,kl)
      real jac,jac1,jac2,j
     2   ac3,x1,x2,x3,y1,y2,y3,z1,z2,z3,x11,x12,x13,x22,x23,x33,y11,y12,
     3   y13,y22,y23,y33,z11,z12,z13,z22,z23,z33,s0,s1,s2,s3,a11,b11,c11
     4   ,a12,b12,c12,a13,b13,c13,a22,b22,c22,a23,b23,c23,a33,b33,c33
      real tt1,tt2,tt3,tt4,tt5,tt6,tt7,tt8,tt9,tt10,tt11,tt12,tt13,tt14
     1   ,tt15,tt16,tt17,tt18,tt19,tt20,tt21,tt22,tt23,tt24,tt25,tt26,tt
     2   27,tt28,tt29,tt30,tt31,tt32,tt33,tt34,tt35,tt36,tt37,tt38,tt39,
     3   tt40,tt41,tt42,tt43,tt44,tt45,tt46,tt47,tt48,tt49,tt50,tt51
      real tt52,tt53,tt54,tt55,tt56,tt57,tt58,tt59,tt60,tt61,tt62,tt63,
     1   tt64,tt65,tt66,tt67,tt68,tt69,tt70,tt71,tt72,tt73,tt74,tt75,tt7
     2   6,tt77,tt78,tt79,tt80,tt81,tt82,tt83,tt84,tt85,tt86,tt87,tt88,t
     3   t89,tt90,tt91,tt92,tt93,tt94,tt95,tt96,tt97,tt98,tt99,tt100,tt1
     4   01,tt102,tt103,tt104,tt105,tt106
      integer ild,jld,il,jl,kl,i1,j1,k1
      i1 = il-1
      j1 = jl-1
      k1 = kl-1
      do 1 k=2,k1
      do 2 j=2,j1
      do 3 i=2,i1
      s0 = s(i,j,k)
      s1 = (s(i+1,j,K)-s(i-1,j,K))/h1/2.0
      x1 = (x(i+1,j,K)-x(i-1,j,K))/h1/2.0
      y1 = (y(i+1,j,K)-y(i-1,j,K))/h1/2.0
      z1 = (z(i+1,j,K)-z(i-1,j,K))/h1/2.0
      s2 = (s(i,j+1,K)-s(i,j-1,K))/h2/2.0
      x2 = (x(i,j+1,K)-x(i,j-1,K))/h2/2.0
      y2 = (y(i,j+1,K)-y(i,j-1,K))/h2/2.0
      z2 = (z(i,j+1,K)-z(i,j-1,K))/h2/2.0
      s3 = (s(i,j,K+1)-s(i,j,K-1))/h3/2.0
      x3 = (x(i,j,K+1)-x(i,j,K-1))/h3/2.0
      y3 = (y(i,j,K+1)-y(i,j,K-1))/n3/2.0
      z3 = (z(i,j,k+1)-z(i,j,K-1))/h3/2.0
      x11 = (x(i+1,j,K)-2*x(i,j,K)+x(i-1,j,K))/h1**2
      y11 = (y(i+1,j,K)-2*y(i,j,K)+y(i-1,j,K))/h1**2
      z11 = (z(i+1,j,K)-2*z(i,j,K)+z(i-1,j,K))/h1**2
      x12 = (x(i+1,j+1,K)-x(i+1,j-1,K)-x(i-1,j+1,K)+x(i-1,j-1,K))/(h1*h2
     1   )/4.0
      y12 = (y(i+1,j+1,K)-y(i+1,j-1,K)-y(i-1,j+1,K)+y(i-1,j-1,K))/(h1*h2
     1   )/4.0
      z12 = (z(i+1,j+1,K)-z(i+1,j-1,K)-z(i-1,j+1,K)+z(i-1,j-1,K))/(h1*h2
     1   )/4.0
      x22 = (x(i,j+1,K)-2*x(i,j,K)+x(i,j-1,K))/h2**2
      y22 = (y(i,j+1,K)-2*y(i,j,K)+y(i,j-1,K))/h2**2
      z22 = (z(i,j+1,K)-2*z(i,j,K)+z(i,j-1,K))/h2**2
      x13 = (x(i+1,j,K+1)-x(i+1,j,K-1)-x(i-1,j,K+1)+x(i-1,j,K-1))/(h1*h3
     1   )/4.0
      y13 = (y(i+1,j,K+1)-y(i+1,j,K-1)-y(i-1,j,K+1)+y(i-1,j,K-1))/(h1*h3
     1   )/4.0
      z13 = (z(i+1,j,K+1)-z(i+1,j,K-1)-z(i-1,j,K+1)+z(i-1,j,K-1))/(h1*h3
     1   )/4.0
      x23 = (x(i,j+1,K+1)-x(i,j+1,K-1)-x(i,j-1,K+1)+x(i,j-1,K-1))/(h2*h3
     1   )/4.0
      y23 = (y(i,j+1,K+1)-y(i,j+1,K-1)-y(i,j-1,K+1)+y(i,j-1,K-1))/(h2*h3
     1   )/4.0
      z23 = (z(i,j+1,K+1)-z(i,j+1,K-1)-z(i,j-1,K+1)+z(i,j-1,K-1))/(h2*h3
     1   )/4.0
      x33 = (x(i,j,K+1)-2*x(i,j,K)+x(i,j,K-1))/h3**2
      y33 = (y(i,j,K+1)-2*y(i,j,K)+y(i,j,K-1))/h3**2
      z33 = (z(i,j,k+1)-2*z(i,j,K)+z(i,j,K-1))/h3**2
      jac = x1*y2*z3-x2*y1*z3-x1*y3*z2+x3*y1*z2+x2*y3*z1-x3*y2*z1
      jac1 = x11*y2*z3+x1*y12*z3-x2*y11*z3-x12*y1*z3-x11*y3*z2-x1*y13*z2
     1    +x3*y11*z2+x13*y1*z2+x1*y2*z13-x2*y1*z13-x1*y3*z12+x3*y1*z12+x2
     2    *y3*z11-x3*y2*z11+x12*y3*z1-x13*y2*z1+x2*y13*z1-x3*y12*z1
      jac2 = x1*y22*z3+x12*y2*z3-x2*y12*z3-x22*y1*z3+x1*y2*z23-x2*y1*z23
     1    -x1*y3*z22+x3*y1*z22-x12*y3*z2-x1*y23*z2+x3*y12*z2+x23*y1*z2+x2
     2    *y3*z12-x3*y2*z12+x22*y3*z1+x2*y23*z1-x3*y22*z1-x23*y2*z1
      jac3 = x1*y2*z33-x2*y1*z33+x1*y23*z3+x13*y2*z3-x2*y13*z3-x23*y1*z3
     1    -x1*y3*z23+x3*y1*z23-x1*y33*z2-x13*y3*z2+x3*y13*z2+x33*y1*z2+x2
     2    *y3*z13-x3*y2*z13+x2*y33*z1+x23*y3*z1-x3*y23*z1-x33*y2*z1
      c11 = z1**2
      b11 = y1**2+c11
      a11 = x1**2+b11
      c12 = z1*z2
```

```
 b12 = y1*y2+c12
 a12 = x1*x2+b12
 c13 = z1*z3
 b13 = y1*y3+c13
 a13 = x1*x3+b13
 c22 = z2**2
 b22 = y2**2+c22
 a22 = x2**2+b22
 c23 = z2*z3
 b23 = y2*y3+c23
 a23 = x2*x3+b23
 c33 = z3**2
 b33 = y3**2+c33
 a33 = x3**2+b33
 tt1 = 2*c12*c33-b12*c33-2*c13*c23+b13*c23+b23*c13-b33*c12+2*b12*b3
1   3-2*b13*b23
 tt2 = -2*c12*c23+b12*c23+2*c13*c22-b13*c22-b22*c13+b23*c12-2*b12*b
1   23+2*b13*b22
 tt3 = -z2*z33+z23*z3-y2*y33+y23*y3
 tt4 = -z22*z3+z2*z23-y22*y3+y2*y23
 tt5 = z1*z33-z13*z3+y1*y33-y13*y3+2*b33/h1
 tt6 = z12*z3-2*z1*z23+z13*z2+y12*y3-2*y1*y23+y13*y2-4*b23/h1
 tt7 = z1*z22-z12*z2+y1*y22-y12*y2+2*b22/h1
 tt8 = c12*z2*z33-c22*z1*z33-c12*z23*z3+c13*z22*z3+c22*z13*z3-c23*z
1   12*z3-c13*z2*z23+2*c23*z1*z23-c33*z1*z22-c23*z13*z2+c33*z12*z2-
2   c12*y2*y33+b12*y2*y33+c22*y1*y33-b22*y1*y33+c12*y23*y3-b12*y23*
3   y3-c13*y22*y3+b13*y22*y3-c22*y13*y3+b22*y13*y3+c23*y12*y3-b23*y
4   12*y3+c13*y2*y23-b13*y2*y23-2*c23*y1*y23+2*b23*y1*y23+c33*y1*y2
5   2-b33*y1*y22+c23*y13*y2-b23*y13*y2-c33*y12*y2+b33*y12*y2-b12*x2
6   *x33+b22*x1*x33+b12*x23*x3-b13*x22*x3-b22*x13*x3+b23*x12*x3+b13
7   *x2*x23-2*b23*x1*x23+b33*x1*x22+b23*x13*x2-b33*x12*x2+(-4*c22*c
8   33+2*b22*c33+4*c23**2-4*b23*c23+2*b33*c22-4*b22*b33+4*b23**2)/h
9   1
 tt9 = 2*c22*c33-b22*c33-2*c23**2+2*b23*c23-b33*c22+2*b22*b33-2*b23
1   **2
 tt10 = -2*c12*c33+b12*c33+2*c13*c23-b13*c23-b23*c13+b33*c12-2*b12*
1   b33+2*b13*b23
 tt11 = 2*c12*c23-b12*c23-2*c13*c22+b13*c22+b22*c13-b23*c12+2*b12*b
1   23-2*b13*b22
 tt12 = b23*jac3-b33*jac2
 tt13 = b23*jac2-b22*jac3
 tt14 = b33*jac1-b13*jac3
 tt15 = b12*jac3+b13*jac2-2*b23*jac1
 tt16 = b22*jac1-b12*jac2
 tt17 = -2*c12*c23*jac3+b12*c23*jac3+2*c13*c22*jac3-b13*c22*jac3-b2
1   2*c13*jac3+b23*c12*jac3-2*b12*b23*jac3+2*b13*b22*jac3+2*c12*c33
2   *jac2-b12*c33*jac2-2*c13*c23*jac2+b13*c23*jac2+b23*c13*jac2-b33
3   *c12*jac2+2*b12*b33*jac2-2*b13*b23*jac2-2*c22*c33*jac1+b22*c33*
4   jac1+2*c23**2*jac1-2*b23*c23*jac1+b33*c22*jac1-2*b22*b33*jac1+2
5   *b23**2*jac1
 tt18 = 2*c12*c23-b12*c23-2*c13*c22+b13*c22+b22*c13-b23*c12+2*b12*b
1   23-2*b13*b22
 tt19 = -2*c12*c33+b12*c33+2*c13*c23-b13*c23-b23*c13+b33*c12-2*b12*
```

```
     1    b33+2*b13*b23
      tt20 = 2*c11*c23-b11*c23-2*c12*c13+b12*c13+b13*c12-b23*c11+2*b11*b
     1    23-2*b12*b13
      tt21 = z2*z33-z23*z3+y2*y33-y23*y3+2*b33/h2
      tt22 = -z1*z33+z13*z3-y1*y33+y13*y3
      tt23 = z12*z3+z1*z23-2*z13*z2+y12*y3+y1*y23-2*y13*y2-4*b13/h2
      tt24 = -z11*z3+z1*z13-y11*y3+y1*y13
      tt25 = z11*z2-z1*z12+y11*y2-y1*y12+2*b11/h2
      tt26 = -c11*z2*z33+c12*z1*z33+c11*z23*z3-c12*z13*z3-c13*z12*z3+c23
     1    *z11*z3-c13*z1*z23+2*c13*z13*z2-c33*z11*z2-c23*z1*z13+c33*z1*z1
     2    2+c11*y2*y33-b11*y2*y33-c12*y1*y33+b12*y1*y33-c11*y23*y3+b11*y2
     3    3*y3+c12*y13*y3-b12*y13*y3+c13*y12*y3-b13*y12*y3-c23*y11*y3+b23
     4    *y11*y3+c13*y1*y23-b13*y1*y23-2*c13*y13*y2+2*b13*y13*y2+c33*y11
     5    *y2-b33*y11*y2+c23*y1*y13-b23*y1*y13-c33*y1*y12+b33*y1*y12+b11*
     6    x2*x33-b12*x1*x33-b11*x23*x3+b12*x13*x3+b13*x12*x3-b23*x11*x3+b
     7    13*x1*x23-2*b13*x13*x2+b33*x11*x2+b23*x1*x13-b33*x1*x12+(-4*c11
     8    *c33+2*b11*c33+4*c13**2-4*b13*c13+2*b33*c11-4*b11*b33+4*b13**2)
     9    /h2
      tt27 = -2*c12*c33+b12*c33+2*c13*c23-b13*c23-b23*c13+b33*c12-2*b12*
     1    b33+2*b13*b23
      tt28 = 2*c11*c33-b11*c33-2*c13**2+2*b13*c13-b33*c11+2*b11*b33-2*b1
     1    3**2
      tt29 = -2*c11*c23+b11*c23+2*c12*c13-b12*c13-b13*c12+b23*c11-2*b11*
     1    b23+2*b12*b13
      tt30 = b33*jac2-b23*jac3
      tt31 = b13*jac3-b33*jac1
      tt32 = b12*jac3-2*b13*jac2+b23*jac1
      tt33 = b13*jac1-b11*jac3
      tt34 = b11*jac2-b12*jac1
      tt35 = 2*c11*c23*jac3-b11*c23*jac3-2*c12*c13*jac3+b12*c13*jac3+b13
     1    *c12*jac3-b23*c11*jac3+2*b11*b23*jac3-2*b12*b13*jac3-2*c11*c33*
     2    jac2+b11*c33*jac2+2*c13**2*jac2-2*b13*c13*jac2+b33*c11*jac2-2*b
     3    11*b33*jac2+2*b13**2*jac2+2*c12*c33*jac1-b12*c33*jac1-2*c13*c23
     4    *jac1+b13*c23*jac1+b23*c13*jac1-b33*c12*jac1+2*b12*b33*jac1-2*b
     5    13*b23*jac1
      tt36 = -2*c11*c23+b11*c23+2*c12*c13-b12*c13-b13*c12+b23*c11-2*b11*
     1    b23+2*b12*b13
      tt37 = z22*z3-z2*z23+y22*y3-y2*y23+2*b22/h3
      tt38 = -2*z12*z3+z1*z23+z13*z2-2*y12*y3+y1*y23+y13*y2-4*b12/h3
      tt39 = -z1*z22+z12*z2-y1*y22+y12*y2
      tt40 = z11*z3-z1*z13+y11*y3-y1*y13+2*b11/h3
      tt41 = -z11*z2+z1*z12-y11*y2+y1*y12
      tt42 = -c11*z22*z3+2*c12*z12*z3-c22*z11*z3+c11*z2*z23-c12*z1*z23+c
     1    13*z1*z22-c12*z13*z2-c13*z12*z2+c23*z11*z2+c22*z1*z13-c23*z1*z1
     2    2+c11*y22*y3-b11*y22*y3-2*c12*y12*y3+2*b12*y12*y3+c22*y11*y3-b2
     3    2*y11*y3-c11*y2*y23+b11*y2*y23+c12*y1*y23-b12*y1*y23-c13*y1*y22
     4    +b13*y1*y22+c12*y13*y2-b12*y13*y2+c13*y12*y2-b13*y12*y2-c23*y11
     5    *y2+b23*y11*y2-c22*y1*y13+b22*y1*y13+c23*y1*y12-b23*y1*y12+b11*
     6    x22*x3-2*b12*x12*x3+b22*x11*x3-b11*x2*x23+b12*x1*x23-b13*x1*x22
     7    +b12*x13*x2+b13*x12*x2-b23*x11*x2-b22*x1*x13+b23*x1*x12+(-4*c11
     8    *c22+2*b11*c22+4*c12**2-4*b12*c12+2*b22*c11-4*b11*b22+4*b12**2)
     9    /h3
      tt43 = 2*c12*c23-b12*c23-2*c13*c22+b13*c22+b22*c13-b23*c12+2*b12*b
```

```
1    23-2*b13*b22
 tt44 = -2*c11*c23+b11*c23+2*c12*c13-b12*c13-b13*c12+b23*c11-2*b11*
1    b23+2*b12*b13
 tt45 = 2*c11*c22-b11*c22-2*c12**2+2*b12*c12-b22*c11+2*b11*b22-2*b1
1    2**2
 tt46 = b22*jac3-b23*jac2
 tt47 = -2*b12*jac3+b13*jac2+b23*jac1
 tt48 = b12*jac2-b22*jac1
 tt49 = b11*jac3-b13*jac1
 tt50 = b12*jac1-b11*jac2
 tt51 = -2*c11*c22*jac3+b11*c22*jac3+2*c12**2*jac3-2*b12*c12*jac3+b
1    22*c11*jac3-2*b11*b22*jac3+2*b12**2*jac3+2*c11*c23*jac2-b11*c23
2    *jac2-2*c12*c13*jac2+b12*c13*jac2+b13*c12*jac2-b23*c11*jac2+2*b
3    11*b23*jac2-2*b12*b13*jac2-2*c12*c23*jac1+b12*c23*jac1+2*c13*c2
4    2*jac1-b13*c22*jac1-b22*c13*jac1+b23*c12*jac1-2*b12*b23*jac1+2*
5    b13*b22*jac1
 tt52 = -2*b22/h3**2-2*b33/h2**2
 tt53 = -2*b11/h3**2-2*b33/h1**2
 tt54 = -2*b11/h2**2-2*b22/h1**2
 tt55 = (4*c11*c22-2*b11*c22-4*c12**2+4*b12*c12-2*b22*c11+4*b11*b22
1    -4*b12**2)/h3**2+(4*c11*c33-2*b11*c33-4*c13**2+4*b13*c13-2*b33*
2    c11+4*b11*b33-4*b13**2)/h2**2+(4*c22*c33-2*b22*c33-4*c23**2+4*b
3    23*c23-2*b33*c22+4*b22*b33-4*b23**2)/h1**2
 tt56 = -z22*z3+z2*z23-y22*y3+y2*y23+2*b22/h3
 tt57 = 2*z12*z3-z1*z23-z13*z2+2*y12*y3-y1*y23-y13*y2-4*b12/h3
 tt58 = z1*z22-z12*z2+y1*y22-y12*y2
 tt59 = -z11*z3+z1*z13-y11*y3+y1*y13+2*b11/h3
 tt60 = z11*z2-z1*z12+y11*y2-y1*y12
 tt61 = c11*z22*z3-2*c12*z12*z3+c22*z11*z3-c11*z2*z23+c12*z1*z23-c1
1    3*z1*z22+c12*z13*z2+c13*z12*z2-c23*z11*z2-c22*z1*z13+c23*z1*z12
2    -c11*y22*y3+b11*y22*y3+2*c12*y12*y3-2*b12*y12*y3-c22*y11*y3+b22
3    *y11*y3+c11*y2*y23-b11*y2*y23-c12*y1*y23+b12*y1*y23+c13*y1*y22-
4    b13*y1*y22-c12*y13*y2+b12*y13*y2-c13*y12*y2+b13*y12*y2+c23*y11*
5    y2-b23*y11*y2+c22*y1*y13-b22*y1*y13-c23*y1*y12+b23*y1*y12-b11*x
6    22*x3+2*b12*x12*x3-b22*x11*x3+b11*x2*x23-b12*x1*x23+b13*x1*x22-
7    b12*x13*x2-b13*x12*x2+b23*x11*x2+b22*x1*x13-b23*x1*x12+(-4*c11*
8    c22+2*b11*c22+4*c12**2-4*b12*c12+2*b22*c11-4*b11*b22+4*b12**2)/
9    h3
 tt62 = -2*c12*c23+b12*c23+2*c13*c22-b13*c22-b22*c13+b23*c12-2*b12*
1    b23+2*b13*b22
 tt63 = 2*c11*c23-b11*c23-2*c12*c13+b12*c13+b13*c12-b23*c11+2*b11*b
1    23-2*b12*b13
 tt64 = -2*c11*c22+b11*c22+2*c12**2-2*b12*c12+b22*c11-2*b11*b22+2*b
1    12**2
 tt65 = b23*jac2-b22*jac3
 tt66 = 2*b12*jac3-b13*jac2-b23*jac1
 tt67 = b22*jac1-b12*jac2
 tt68 = b13*jac1-b11*jac3
 tt69 = b11*jac2-b12*jac1
 tt70 = 2*c11*c22*jac3-b11*c22*jac3-2*c12**2*jac3+2*b12*c12*jac3-b2
1    2*c11*jac3+2*b11*b22*jac3-2*b12**2*jac3-2*c11*c23*jac2+b11*c23*
2    jac2+2*c12*c13*jac2-b12*c13*jac2-b13*c12*jac2+b23*c11*jac2-2*b1
3    1*b23*jac2+2*b12*b13*jac2+2*c12*c23*jac1-b12*c23*jac1-2*c13*c22
```

```
4    *jac1+b13*c22*jac1+b22*c13*jac1-b23*c12*jac1+2*b12*b23*jac1-2*b
5    13*b22*jac1
 tt71 = -2*c11*c23+b11*c23+2*c12*c13-b12*c13-b13*c12+b23*c11-2*b11*
1    b23+2*b12*b13
 tt72 = -z2*z33+z23*z3-y2*y33+y23*y3+2*b33/h2
 tt73 = z1*z33-z13*z3+y1*y33-y13*y3
 tt74 = -z12*z3-z1*z23+2*z13*z2-y12*y3-y1*y23+2*y13*y2-4*b13/h2
 tt75 = z11*z3-z1*z13+y11*y3-y1*y13
 tt76 = -z11*z2+z1*z12-y11*y2+y1*y12+2*b11/h2
 tt77 = c11*z2*z33-c12*z1*z33-c11*z23*z3+c12*z13*z3+c13*z12*z3-c23*
1    z11*z3+c13*z1*z23-2*c13*z13*z2+c33*z11*z2+c23*z1*z13-c33*z1*z12
2    -c11*y2*y33+b11*y2*y33+c12*y1*y33-b12*y1*y33+c11*y23*y3-b11*y23
3    *y3-c12*y13*y3+b12*y13*y3-c13*y12*y3+b13*y12*y3+c23*y11*y3-b23*
4    y11*y3-c13*y1*y23+b13*y1*y23+2*c13*y13*y2-2*b13*y13*y2-c33*y11*
5    y2+b33*y11*y2-c23*y1*y13+b23*y1*y13+c33*y1*y12-b33*y1*y12-b11*x
6    2*x33+b12*x1*x33+b11*x23*x3-b12*x13*x3-b13*x12*x3+b23*x11*x3-b1
7    3*x1*x23+2*b13*x13*x2-b33*x11*x2-b23*x1*x13+b33*x1*x12+(-4*c11*
8    c33+2*b11*c33+4*c13**2-4*b13*c13+2*b33*c11-4*b11*b33+4*b13**2)/
9    h2
 tt78 = 2*c12*c33-b12*c33-2*c13*c23+b13*c23+b23*c13-b33*c12+2*b12*b
1    33-2*b13*b23
 tt79 = -2*c11*c33+b11*c33+2*c13**2-2*b13*c13+b33*c11-2*b11*b33+2*b
1    13**2
 tt80 = 2*c11*c23-b11*c23-2*c12*c13+b12*c13+b13*c12-b23*c11+2*b11*b
1    23-2*b12*b13
 tt81 = b23*jac3-b33*jac2
 tt82 = b33*jac1-b13*jac3
 tt83 = -b12*jac3+2*b13*jac2-b23*jac1
 tt84 = b11*jac3-b13*jac1
 tt85 = b12*jac1-b11*jac2
 tt86 = -2*c11*c23*jac3+b11*c23*jac3+2*c12*c13*jac3-b12*c13*jac3-b1
1    3*c12*jac3+b23*c11*jac3-2*b11*b23*jac3+2*b12*b13*jac3+2*c11*c33
2    *jac2-b11*c33*jac2-2*c13**2*jac2+2*b13*c13*jac2-b33*c11*jac2+2*
3    b11*b33*jac2-2*b13**2*jac2-2*c12*c33*jac1+b12*c33*jac1+2*c13*c2
4    3*jac1-b13*c23*jac1-b23*c13*jac1+b33*c12*jac1-2*b12*b33*jac1+2*
5    b13*b23*jac1
 tt87 = 2*c11*c23-b11*c23-2*c12*c13+b12*c13+b13*c12-b23*c11+2*b11*b
1    23-2*b12*b13
 tt88 = -2*c12*c33+b12*c33+2*c13*c23-b13*c23-b23*c13+b33*c12-2*b12*
1    b33+2*b13*b23
 tt89 = 2*c12*c23-b12*c23-2*c13*c22+b13*c22+b22*c13-b23*c12+2*b12*b
1    23-2*b13*b22
 tt90 = z2*z33-z23*z3+y2*y33-y23*y3
 tt91 = z22*z3-z2*z23+y22*y3-y2*y23
 tt92 = -z1*z33+z13*z3-y1*y33+y13*y3+2*b33/h1
 tt93 = -z12*z3+2*z1*z23-z13*z2-y12*y3+2*y1*y23-y13*y2-4*b23/h1
 tt94 = -z1*z22+z12*z2-y1*y22+y12*y2+2*b22/h1
 tt95 = -c12*z2*z33+c22*z1*z33+c12*z23*z3-c13*z22*z3-c22*z13*z3+c23
1    *z12*z3+c13*z2*z23-2*c23*z1*z23+c33*z1*z22+c23*z13*z2-c33*z12*z
2    2+c12*y2*y33-b12*y2*y33-c22*y1*y33+b22*y1*y33-c12*y23*y3+b12*y2
3    3*y3+c13*y22*y3-b13*y22*y3+c22*y13*y3-b22*y13*y3-c23*y12*y3+b23
4    *y12*y3-c13*y2*y23+b13*y2*y23+2*c23*y1*y23-2*b23*y1*y23-c33*y1*
5    y22+b33*y1*y22-c23*y13*y2+b23*y13*y2+c33*y12*y2-b33*y12*y2+b12*
```

```
6    x2*x33-b22*x1*x33-b12*x23*x3+b13*x22*x3+b22*x13*x3-b23*x12*x3-b
7    13*x2*x23+2*b23*x1*x23-b33*x1*x22-b23*x13*x2+b33*x12*x2+(-4*c22
8    *c33+2*b22*c33+4*c23**2-4*b23*c23+2*b33*c22-4*b22*b33+4*b23**2)
9    /h1
  tt96 = -2*c22*c33+b22*c33+2*c23**2-2*b23*c23+b33*c22-2*b22*b33+2*b
1    23**2
  tt97 = 2*c12*c33-b12*c33-2*c13*c23+b13*c23+b23*c13-b33*c12+2*b12*b
1    33-2*b13*b23
  tt98 = -2*c12*c23+b12*c23+2*c13*c22-b13*c22-b22*c13+b23*c12-2*b12*
1    b23+2*b13*b22
  tt99 = b33*jac2-b23*jac3
  tt100 = b22*jac3-b23*jac2
  tt101 = b13*jac3-b33*jac1
  tt102 = -b12*jac3-b13*jac2+2*b23*jac1
  tt103 = b12*jac2-b22*jac1
  tt104 = 2*c12*c23*jac3-b12*c23*jac3-2*c13*c22*jac3+b13*c22*jac3+b2
1    2*c13*jac3-b23*c12*jac3+2*b12*b23*jac3-2*b13*b22*jac3-2*c12*c33
2    *jac2+b12*c33*jac2+2*c13*c23*jac2-b13*c23*jac2-b23*c12*jac2+b33
3    *c12*jac2-2*b12*b33*jac2+2*b13*b23*jac2+2*c22*c33*jac1-b22*c33*
4    jac1-2*c23**2*jac1+2*b23*c23*jac1-b33*c22*jac1+2*b22*b33*jac1-2
5    *b23**2*jac1
  tt105 = -2*c12*c23+b12*c23+2*c13*c22-b13*c22-b22*c13+b23*c12-2*b12
1    *b23+2*b13*b22
  tt106 = 2*c12*c33-b12*c33-2*c13*c23+b13*c23+b23*c13-b33*c12+2*b12*
1    b33-2*b13*b23
  cf112(i,j,k) = (s0*tt1-a12*b33*s0+a13*b23*s0+a23*b13*s0-a33*b12*s0
1    )/(h1*h2*jac**2)/2.0
  cf121(i,j,k) = (s0*tt2+a12*b23*s0-a13*b22*s0-a22*b13*s0+a23*b12*s0
1    )/(h1*h3*jac**2)/2.0
  cf122(i,j,k) = (s1*tt9+s0*tt8+a33*s0*tt7+a23*s0*tt6+a22*s0*tt5+a13
1    *s0*tt4+a12*s0*tt3+s3*tt11+s2*tt10-a12*b23*s3+a13*b22*s3+a22*b1
2    3*s3-a23*b12*s3+a12*b33*s2-a13*b23*s2-a23*b13*s2+a33*b12*s2-a22
3    *b33*s1-a33*b22*s1)/(h1*jac**2)/2.0+(s0*tt17+a33*s0*tt16+a23*s0
4    *tt15+a22*s0*tt14+a13*s0*tt13+a12*s0*tt12)/(h1*jac**3)/2.0+a23*
5    b23*s1/(h1*jac**2)
  cf123(i,j,k) = (s0*tt18-a12*b23*s0+a13*b22*s0+a22*b13*s0-a23*b12*s
1    0)/(h1*h3*jac**2)/2.0
  cf132(i,j,k) = (s0*tt19+a12*b33*s0-a13*b23*s0-a23*b13*s0+a33*b12*s
1    0)/(h1*h2*jac**2)/2.0
  cf211(i,j,k) = (s0*tt20-a11*b23*s0+a12*b13*s0+a13*b12*s0-a23*b11*s
1    0)/(h2*h3*jac**2)/2.0
  cf212(i,j,k) = (s0*tt35+a33*s0*tt34+a23*s0*tt33+a13*s0*tt32+a12*s0
1    *tt31+a11*s0*tt30)/(h2*jac**3)/2.0+(s3*tt29+s2*tt28+s1*tt27+s0*
2    tt26+a33*s0*tt25+a23*s0*tt24+a13*s0*tt23+a12*s0*tt22+a11*s0*tt2
3    1+a11*b23*s3-a12*b13*s3-a13*b12*s3+a23*b11*s3-a11*b33*s2-a33*b1
4    1*s2+a12*b33*s1-a13*b23*s1-a23*b13*s1+a33*b12*s1)/(h2*jac**2)/2
5    .0+a13*b13*s2/(h2*jac**2)
  cf213(i,j,k) = (s0*tt36+a11*b23*s0-a12*b13*s0-a13*b12*s0+a23*b11*s
1    0)/(h2*h3*jac**2)/2.0
  cf221(i,j,k) = (s0*tt51+a23*s0*tt50+a22*s0*tt49+a13*s0*tt48+a12*s0
1    *tt47+a11*s0*tt46)/(h3*jac**3)/2.0+(s3*tt45+s2*tt44+s1*tt43+s0*
2    tt42+a23*s0*tt41+a22*s0*tt40+a13*s0*tt39+a12*s0*tt38+a11*s0*tt3
3    7-a11*b22*s3-a22*b11*s3+a11*b23*s2-a12*b13*s2-a13*b12*s2+a23*b1
```

```
     4   1*s2-a12*b23*s1+a13*b22*s1+a22*b13*s1-a23*b12*s1)/(h3*jac**2)/2
     5   .0+a12*b12*s3/(h3*jac**2)
      cf222(i,j,K) = (s0*tt55+a33*s0*tt54+a22*s0*tt53+a11*s0*tt52)/jac**
     1   2+4*a12*b12*s0/(h3**2*jac**2)+4*a13*b13*s0/(h2**2*jac**2)+4*a23
     2   *b23*s0/(h1**2*jac**2)
      cf223(i,j,K) = (s0*tt70+a23*s0*tt69+a22*s0*tt68+a13*s0*tt67+a12*s0
     1   *tt66+a11*s0*tt65)/(h3*jac**3)/2.0+(s3*tt64+s2*tt63+s1*tt62+s0*
     2   tt61+a23*s0*tt60+a22*s0*tt59+a13*s0*tt58+a12*s0*tt57+a11*s0*tt5
     3   6+a11*b22*s3+a22*b11*s3-a11*b23*s2+a12*b13*s2+a13*b12*s2-a23*b1
     4   1*s2+a12*b23*s1-a13*b22*s1-a22*b13*s1+a23*b12*s1)/(h3*jac**2)/2
     5   .0-a12*b12*s3/(h3*jac**2)
      cf231(i,j,K) = (s0*tt71+a11*b23*s0-a12*b13*s0-a13*b12*s0+a23*b11*s
     1   0)/(h2*h3*jac**2)/2.0
      cf232(i,j,K) = (s0*tt86+a33*s0*tt85+a23*s0*tt84+a13*s0*tt83+a12*s0
     1   *tt82+a11*s0*tt81)/(h2*jac**3)/2.0+(s3*tt80+s2*tt79+s1*tt78+s0*
     2   tt77+a33*s0*tt76+a23*s0*tt75+a13*s0*tt74+a12*s0*tt73+a11*s0*tt7
     3   2-a11*b23*s3+a12*b13*s3+a13*b12*s3-a23*b11*s3+a11*b33*s2+a33*b1
     4   1*s2-a12*b33*s1+a13*b23*s1+a23*b13*s1-a33*b12*s1)/(h2*jac**2)/2
     5   .0-a13*b13*s2/(h2*jac**2)
      cf233(i,j,K) = (s0*tt87-a11*b23*s0+a12*b13*s0+a13*b12*s0-a23*b11*s
     1   0)/(h2*h3*jac**2)/2.0
      cf312(i,j,K) = (s0*tt88+a12*b33*s0-a13*b23*s0-a23*b13*s0+a33*b12*s
     1   0)/(h1*h2*jac**2)/2.0
      cf321(i,j,K) = (s0*tt89-a12*b23*s0+a13*b22*s0+a22*b13*s0-a23*b12*s
     1   0)/(h1*h3*jac**2)/2.0
      cf322(i,j,K) = (a12*s0*tt99+s0*tt104+a33*s0*tt103+a23*s0*tt102+a22
     1   *s0*tt101+a13*s0*tt100)/(h1*jac**3)/2.0+(s3*tt98+s2*tt97+s1*tt9
     2   6+s0*tt95+a33*s0*tt94+a23*s0*tt93+a22*s0*tt92+a13*s0*tt91+a12*s
     3   0*tt90+a12*b23*s3-a13*b22*s3-a22*b13*s3+a23*b12*s3-a12*b33*s2+a
     4   13*b23*s2+a23*b13*s2-a33*b12*s2+a22*b33*s1+a33*b22*s1)/(h1*jac*
     5   *2)/2.0-a23*b23*s1/(h1*jac**2)
      cf323(i,j,K) = (s0*tt105+a12*b23*s0-a13*b22*s0-a22*b13*s0+a23*b12*
     1   s0)/(h1*h3*jac**2)/2.0
      cf332(i,j,K) = (s0*tt106-a12*b33*s0+a13*b23*s0+a23*b13*s0-a33*b12*
     1   s0)/(h1*h2*jac**2)/2.0
    3 continue
    2 continue
    1 continue
      return
      end
```

A BIBLIOGRAPHY OF SOVIET WORKS IN ALGEBRAIC MANIPULATIONS

Alfonso M. MIOLA
Istituto di Analisi dei Sistemi e Informatica
Via Buonarroti 12
00185 ROMA (ITALY)

In the June 1979 a Summer School on Programming has been organized by the Bulgarian Academy of Sciences in Primorsko (Bulgaria). Among other topics Symbolic and Algebraic Manipulations was covered with a few lectures by me and with some panel discussions. The lecturers were Professors Lavrov, Arato, Havel, Pottosin, Bauer, Pasula, Ershov, Andronico, Miola.
Recently Prof. Pottosin sent me a bibliography of the works done in Russia in Computer Algebra. I do think that this bibliography could be of interest of our community. Prof. Pottosin address is:
– 630090 Novosibirsk 90 – Computer Center – I.V. Pottosin.– USSR

1. S.A.Abramov. On Rational Functions Summing. J. Comput. Math. and Math. Phys., v. 11, No.4, 1971, pp. 1071-1075.
2. S.A.Abramov. On Some Algorithms for Algebraic Transformations of Functional Expressions. J. Comput. Math. and Comput. Mach., No.3, Kharkov, 1972, pp. 55-57.
3. I.R.Akselrod, L.F.Belous. Input Language for Automatic Programming System SIRIUS. In: "Automatization of Programming", No.3, Kiev, 1967.
4. I.R.Akselrod, L.F.Belous. Input Language for Automatic Programming System SIRIUS. Kharkov University, 1969.
5. I.R.Akselrod, L.F.Belous. On Reprocessing Literal-Analytical Information by Computer. J. Kibernetika, No.6, 1966.
6. I.R.Akselrod, L.F.Belous. On Symbolic Manipulation in Conversational Programming System SIRIUS. In: "Proc. 2-nd All Union Conference on Programming", Section H, Novosibirsk,1970.
7. I.R.Akselrod. Computation of Expressions in SIRIUS-System. In: "Automatization of Programming", No.2, Kiev, 1969.
8. I.R.Akselrod, L.F.Belous. Recursive Programs Organization in SIRIUS-System. In: "Automatization of Programming", No.3, Kiev, 1969.
9. I.R.Akselrod. On Syntax Analysis in SIRIUS-System. In: "Automatization of Programming", No.1, Kiev, 1969.
10. E.A.Arays, A.Shutenkov. Solution of Linear Algebra Problems in AVTO-ANALITIK-System of programming and Automatic Design, Tomsk, Tomsk University, 1971, pp. 191-196.
11. E.A.Arays, A.Shutenkov, G.V.Sibiriakov. Interpretation System for Solution of Large Problems. Issues of programming and Automatic Design, Tomsk, Tomsk University, 1971.
12. E.A.Arays, G.V.Sibiriakov. The AVTO-ANALITIK Programming System. J. Comput. Math. and Comput. Mach., No.3, Kharkov, 1972.
13. E.A.Arays, G.V.Sibiriakov. AVTO-ANALITIK. Novosibirsk University, 1973.
14. E.A.Arays, A.Shutenkov. The Realization of External Form with Cartan Method. Dokl. Akad. Nauk SSSR, 1974, 214, No.4, pp. 737-738.
15. I.O.Babaev. Some Extention of FORTRAN for Solving Celestial Mechanics Problems. Proc. 5-th Conf. on Math. and Mech., Tomsk, v.2, pp.145-146.
16. M.M.Bexhanova. On Some Aspects of Symbolic Manipulations. J. Comput. Math. and Comput. Mach., No.3, Kharkov, 1972, p. 60.
17. M.M.Bexhanova, I.V.Pottosin. Purpose of Difprocessor and its Input Language. Report of Programming Depart. of the Computing Center, Siberian Branch, USSR, Novosibirsk, 1966.
18. M.M.Bexhanova, N.I.Kostiukova, G.A.Plothikova, I.V.Pottosin. Outline of Difprocessor Algorithms. Report of Programming Department of the Computing Center, Siberian Branch, USSR, Novosibirsk, 1966.
19. M.M.Bexhanova, V.L.Katkov, I.V.Pottosin. Researchs on Symbolic Manipulation in the Computing Center, Siberian Branch, Academy of Sciences, USSR. J. Comput. Math. and Comput. Mach., No.3, Kharkov, 1972, p. 21.
20. L.F.Belous, I.R.Akselrod. On Realization of SIRIUS Automatic Programming System. In: "Automatization of Programming", No.3, Kiev, 1967.
21. L.F.Belous. Analytical Differentiation in SIRIUS-System. In: "Automatization of Programming", No. 2, Kiev, 1969.
22. L.F.Belous. Dynamic Storage Allocation in SIRIUS-System. In: "Automatization of Programming", No. 2, Kiev, 1969.

23. Yu.V.Blagoveshchensky, V.G.Bondarchuk, Yu.S.Fishman. On Efficiency of Problem Solving Analytical Methods by Computer. In: "Issues of Accuracy and Efficiency of Comput. Algorithms", Proc. of Symposium, v. 5, Kiev, 1969.

24. Yu.V.Blagoveshchensky, Yu.S.Fishman, V.A. Shcherbakov. The Program for Analytical Solving of Nonlinear Ascillation Equations on MIR-2 Computer with ANALITIK Input Language. J. Kibernetika, No.6, Kiev, 1971.

25. V.G.Bondarchuk, S.V.Pogrebinsky. On Basic Principles of ANALITIK-Language Implementation. In: "Theory of Automata", No.2, Kiev, 1968.

26. V.G.Bondarchuk, Yu.S.Fishman. Integration Algorithms on ANALITIK-Language. J. Kibernetika, No. 4, 1968.

27. V.A.Brumberg. Celestial Mechanics Methods for Literal Manipulations. Tomsk University, 1974.

28. V.A.Brumberg, L.A.Isakovich. AMS-System for Analytical Manipulations of Poisson Series on Computer. Celestial Mechanics Algorithms, No.1, Theoretical Astronomy Institute, Leningrad, 1974.

29. A.V.Vasilieva. ALITA-System for Analytical Manipulations of Poisson Series on Computer. Celestial Mechanics Algorithms, No.7, Theoretical Astronomy Institute, Leningrad, 1975.

30. L.I.Goncharova. A Contribution Toward the Problem of General Organization of Symbolic Manipulation Systems. J. Comput. Math. and Comput. Mach., No.3, Kharkov, 1972, p. 62.

31. V.P.Gerdt. On Application of Symbol Manipulation Systems for Feinman Integrals Computation (Review). In Proc. "International Conference on Programming and Mathematical Methods for Solving Physical Problems", Dubna, 20-23 Sept. 1977, IINI, D10, 11-11264 Dubna, 1978.

32. V.P.Gerdt, O.V.Tarasov, D.V.Shirkov. Analytical Computations in Physics and Mathematics. Preprint IINI P2-11547, Dubna, 1978.

33. V.M.Glushkov, V.G.Bondarchuk, T.A.Grinchenko, A.A.Dorodnitsyna, V.P.Klimenko, A.A.Letichevsky, S.B.Pogrebinsky, A.A.Stogny, Yu.S.Fishman. ANALITIK (Algorithmic Language for Description of Computation Processes with algebraic manipulation). J. Kibarnetika, No.3, 1971.

34. V.M.Glushkov, T.A.Grinchenko, A.A.Dorodnitsyna, A.M.Drakh, Yu.V.Kapitonova, V.P.Klimenko, L.H.Kress, A.A.Letichevsky, S.B.Pogrebinsky, A.A.Stogny, Yu.S.Fishman, N.P.Tsariuk. ANALITIK-74 Algorithmic Language (Informational Part) Preprint 77/27, Institute of Cybernetics, Kiev, 1977.

35. V.M.Glushkov, T.A.Grinchenko, A.A.Dorodnitsyna, A.M.Drah, Yu.V.Kapitonova, V.P.Klimenko, L.H.Kress, A.A.Letichevsky, S.B.Pogrebinsky, A.A.Stogny, Yu.S.Fishman, N.P.Tsariuk. ANALITIK-74, J. Kibernetika, No. 5, 1978, pp. 114-147.

36. T.A.Grinchenko. Internal Representation and Analytical Expression Computations : "Theory of Automata", No. 2, Kiev, 1968.

37. T.A.Grinchenko, A.A.Dorodnitsv [illegible] .P.Klimenko, Yu.S.Fishman. Symbolic Manipu[illegible]on Computer System for Engineering Calculations, MIR-2. J. Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, p. 26.

38. T.A.Grinchenko. Construction Principles and Computer Realization of APPLY-Operator. In: "Theory of Automata", No. 2, Kiev, 1968.

39. T.A.Grinchenko. Computer Realization Principles of Symbol Manipulation. J. Kibernetika, No. 1, 1968.

40. S.A.Ivanova. Language and Translator for Algebraic Manipulations with Polynomial from several Variables. Letvia annual, 1974, 17, pp. 230-237.

41. N.A.Kalinina. Some Algorithms and Methods in Symbol Manipulation Systems. Report of Programming Dept. of the Comput. Center, Siberian, USSR, Novosibirsk, 1972.

42. N.A.Kalinina. The ANALITIK System Program. J. Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, p. 33.

43. N.A.Kalinina. Symbol Manipulation Systems (Review). Report of Computing Center, Siberian Branch, USSR, Novosibirsk, 1972.

44. N.A.Kalinina, On Hierarchy in Symbol Manipulation Systems. J. Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, p. 70.

45. N.A.Kalinina. Some Aspects of Design of Symbol Manipulation Systems. In: "System and Theoretical Programming", Computing Center, Siberian Branch, URRS, Novosibirsk, 1973, pp. 103-123.

46. N.A.Kalinina. The Structure and Semantic Features of Symbol Manipulation Language. In: "Programming Problems", Computing Center, Siberian Branch, URRS, Novosibirsk, 1976, pp.8-34.

47. N.A.Kalinina, I.V.Pottosin. Architecture of General Purpose Symbol Manipulation Systems: Adaptability to Solved Problems and Interface with Programming System. In: "Theory and Practica of System Programming", Computing Center, Siberian Branch, URRS, Novosibirsk, 1977, pp. 5-12.

48. N.A.Kalinina. KANVA-Complex Analytical Evaluator. Organization and Key Algorithms. In: "Theory and Practice of System Programming", Computing Center, Siberian Branch, USSR, Novosibirsk, 1977, pp. 13-21.

49. L.V.Kantorovich. On Numeral and Analytical Computations on Computer. News of Academy Science of Armenia SSR, Section Phys. Math., 1957, No. 2.

50. L.V.Kantorovich. On a Mathematical Symbolism Suitable frr Carring out Calculations on Computers. Dokl. Akad. Nauk SSSR, 1957, n.113, No. 4.

51. L.V.Katkov, N.I.Kostiukova. The Processor KINO. In: "Dynamic of Continuous Medium", Novosibirsk, 1969, v. 1.

52. V.L.Katkov, N.I.Kostiukova. The KINO System for Construction of Analytical Solutions of differential Equations on Computer. Proc. 1st All Union Conference on Programming, Kiev, 1968.

53. V.L.Katkov, N.I.Kostiukova. Calculations of Group on Computer. "Some Problems of Computing and Applied Mathematics", Novosibirsk, 1975, pp. 257-267.

54. V.L.Katkov, M.D.Popov. Using Computer BESM-6 for Calculations of Group Admitted by Differential Equations System. Intern. Symposium "Theoretical-Group Methods in Mechanics",1978, p. 17.

55. V.P.Klimenko, S.B.Pogrebinsky, Yu.S.Fishman. A Contribution Toward the Problem Recognition of Functional Properties of Analytical Expressions on MIR-2 Computer. J. Kibernetika, No. 2, 1973, pp. 43-53.

56. N.I.Kostiukova. The Processor PASSIV. J.Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, p. 38.

57. G.P.Kozhenvikova. On Efficient Realization of Algorithmic Languages for Analytical Transformations. Proceedings of Symposium "Language Theory and Methods of Constructing Programming System", Kiev-Alushta, 1972, pp. 338-345.

58. G.P.Kozhenvikova. Computational Complexity of the Procedure "Compare" and "Differentiate" with Respect to the Languages of Lukashevicz and Kantorovich. J. Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, pp. 64-65.

59. G.P.Kozhevnikova. On the Estimation of Efficiency of Symbol Manipulations. J. Cont. Systems and Machines, Kiev, No. 1, 1974.

60. G.P.Kozhevnikova, A.A.Stogny. Representation of Analytical Expressions under Algebraic Manipulations Performing on Computer. J. Kibernetika, No. 4, Kiev, 1975.

61. L.T.Petrova. On Execution of Algebraic Manipulations on Computer. High School Reports. Mathematics, No. 5, 1958, pp. 95-104.

62. L.T.Petrova. Some Applications of Scheme Symbolism. J. Comput. Math. and Comput. Phys., v. 1, No. 3, M. 1961, pp. 513-522.

63. L.T.Petrova, I.A.Platunova. Realization of Calculations in Source Lists Class on Computer. Proc. Math. Inst. Ac. Sci. USSR, v.66, 1962.

64. G.A.Plotnikova. Analytical Transformations in Difprocessor. Report of Programming Dept. of the Computing Center, Siberian Branch, USSR, Novosibirsk, 1965.

65. S.B.Pogrebinsky, Yu.S.Fishman. Dialog System for Analytical Solution of Some Problems of Algebra. Proc. of Symposium on "Language Theory and Methods of Constructing Programming System", Kiev-Alushta, 1972, pp.329-337.

66. E.N.Paskhin. Analytical Differentiation on Computer. In: "Computing Methods and Programming", v. 9, Moscow State Univ., 1967.

67. V.I.Skripnichenko. Operations with Literal Decompositions on Computer Results of Science and Tech. Astronomy, v. 11, p. 131, All Union Inst. Sci. and Tech. Information, Moscow,1975.

68. T.N.Smirnova. Polynomial PRORAB and Carrying out Analytical Transformations on Computer. Ph. D. Thesis, Leningrad, 1963.

69. T.N.Smirnova. Carrying out Analytical Transformations for on M-20 Computer with PRORAB-Program. Leningrad, Nauka, 1967.

70. V.V.Tumasonis. The System of Equivalent Transformations Expressions. J. Comput. Math. and Math. Phys., v. 11, No. 5, 1971, pp. 1272-1281.

71. V.V.Tumasonis. ALDA-Conversational System of Equivalent Transformations on Expressions. J. Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, pp. 52-54.

72. V.F.Turchin, V.V.Serdobolsky. REFAL Languages and its Application for Algebraic Expressions Transformations. J. Kibernetika, No.3, Kiev, 1969.

73. Yu.S.Fishman. Integration of Functions by Computer Performing Analytical Transformations. In: "Theory of Automaton", v. 2, Kiev, 1968.

74. Yu.S.Fishman, A.T.Kotsiuba. The Realization of General-Purpose Symbol Integration Program on Computer. In: "Issue of Accuracy and Efficiency of Comput. Algorithms", v. 5, Kiev, 1969.

75. M.A.Chubarov. Polynomial Assembler. J. Comput. Math. and Comput. Mach., No. 3, Kharkov, 1972, pp. 42-44.

76. M.A.Chubarov. The ISP Interpretation System for Polynomial Manipulations. In: "Digital and Comput. Tech.", v. 5, Moscow, 1969.

77. V.A.Shurygin, N.N.Yanenko. On Realization of Algebraic Differential Algorithms by Computer. Problems of Cibernetics, v. 6, 1961.

78. D.Mordukhai-Bottovskoi, (trans. by Boris Korenblum and Myra Prelle), "A General Investigation of Integration in Finite Form of Differential Equations of the First Order"; Trans. in ACM SIGSAM Bulletin, v. 15, No. 2, pp. 20-32, May 1981.

APPENDIX A-3

AN EXAMPLE OF SOVIET WORK IN MACHINE SYMBOLIC MANIPULATION

KRUPNIKOV, ERNST DAVIDOVICH
POSTE RESTANTE
NOVOSIBIRSK 90
630090, SOVIET UNION

DOCTOR B.DAVID SAUNDERS
"SIGSAM BULLETIN" EDITOR
DEPARTMENT OF MATHEMATICAL SCIENCES
RENSSELAER POLYTECHNIC INSTITUTE
TROY, NEW YORK 12181, USA

March 5, 1982

DEAR DAVID,
THE INTEREST IN USING THE THEORY OF GENERALIZED HYPERGEOMETRIC FUNCTIONS IN COMPUTER ALGEBRA ALGORITHMS IS INCREASING. I OFFER A PROBLEM FOR CONSIDERATION BY READERS OF YOUR RESPECTABLE BULLETIN.

WHAT IS A MINIMAL SET OF IDENTITIES APPLICATION OF WHICH FACTORIZE EACH OF THE FOLLOWING TEN FUNCTIONS INTO THE PRODUCT OF TWO HYPERGEOMETRIC FUNCTIONS WITH LEAST NUMBER OF PARAMETERS?

$${}_1F_2\left(\begin{matrix}1/2 \\ 1/2-a,\ 1/2+a\end{matrix};z\right), \qquad {}_2F_3\left(\begin{matrix}a-b,\ a+b \\ a,\ 1/2+a,\ 2a\end{matrix};z\right),$$

$${}_1F_1\left(\begin{matrix}1/2 \\ 1-a,\ 1+a\end{matrix};z\right), \qquad {}_2F_3\left(\begin{matrix}a,\ 1/2+a \\ 1/2+a-b,\ 1/2+a+b,\ 2a\end{matrix};z\right),$$

$${}_1F_2\left(\begin{matrix}a \\ 1/2+a,\ -1+2a\end{matrix};z\right), \qquad {}_3F_2\left(\begin{matrix}a,\ a-b,\ a+b \\ 1/2+a,\ 2a\end{matrix};z\right),$$

$${}_1F_2\left(\begin{matrix}a \\ 1/2+a,\ 2a\end{matrix};z\right), \qquad {}_3F_2\left(\begin{matrix}a,\ a-b,\ a+b \\ 1/2+a,\ -1+2a\end{matrix};z\right),$$

$${}_1F_2\left(\begin{matrix}a \\ 1/2+a,\ 1+2a\end{matrix};z\right), \qquad {}_3F_2\left(\begin{matrix}a,\ 1/2+a-b,\ 1/2+a+b \\ 1/2+a,\ 2a\end{matrix};z\right).$$

I BELIEVE IT WILL BE VERY INSTRUCTIVE TO BRING THE CORRESPONDING PROGRAMS IN REDUCE-2, MACSYMA, АНАЛИТИК-71, AND OTHER HIGH-LEVEL LANGUAGES INTO COMPARISION. IF YOU WISH I SHALL IMMEDIATELY AIRMAIL MY PROGRAM IN АНАЛИТИК-71 TO YOU.

HAVE YOU RECEIVED MY LETTER OF JANUARY 26?

WITH WARMEST REGARDS,

[illegible]

ERNST OF NOVOSIBIRSK
ALGEBRA PROGRAMMER

CARBON COPIES TO PROFESSORS ANTHONY CLEM HEARN AND RICHARD J.FATEMAN
ENCLOSURE: TEST RUN OUTPUT WITH MY COMMENTS.

computer : МИР-2
algorithmic language : АНАЛИТИК-71
turnaround time : 83 sec.
algebra programmer : Ernst Krupnikov

Novosibirsk, March 1982

```
F(U(1/2)*L(1/2-(A))*L(1/2+A),Z)→0
-1+2*F(L(1/2-(A)),1/4*Z)*F(L(1/2+A),1/4*Z)
```

$$ {}_1F_2\left(\genfrac{}{}{0pt}{}{1/2}{1/2-a,\,1/2+a};z\right)=-1+2\,{}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1/2-a};z/4\right){}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1/2+a};z/4\right); $$

```
F(U(1/2)*L(1-(A))*L(1+A),Z)→0
F(L(1-(A)),1/4*Z)*F(L(1+A),1/4*Z)
```

$$ {}_1F_2\left(\genfrac{}{}{0pt}{}{1/2}{1-a,\,1+a};z\right)={}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1-a};z/4\right){}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1+a};z/4\right); $$

```
F(U(A)*L(1/2+A)*L(-1+2*A),Z)→0
F(L(1/2+A),1/4*Z)*F(L(-1/2+A),1/4*Z)
```

$$ {}_1F_2\left(\genfrac{}{}{0pt}{}{a}{1/2+a,\,-1+2a};z\right)={}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1/2+a};z/4\right){}_0F_1\left(\genfrac{}{}{0pt}{}{—}{-1/2+a};z/4\right); $$

```
F(U(A)*L(1/2+A)*L(2*A),Z)→0
F(L(1/2+A),1/4*Z)↑2
```

$$ {}_1F_2\left(\genfrac{}{}{0pt}{}{a}{1/2+a,\,2a};z\right)=\left[{}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1/2+a};z/4\right)\right]^2; $$

```
F(U(A)*L(1/2+A)*L(1+2*A),Z)→0
F(U(A)*L(1+2*A),2*Z↑(1/2))*F(U(A)*L(1+2*A),-2*Z↑(1/2))
```

$$ {}_1F_2\left(\genfrac{}{}{0pt}{}{a}{1/2+a,\,1+2a};z\right)={}_1F_1\left(\genfrac{}{}{0pt}{}{a}{1+2a};2\sqrt{z}\right){}_1F_1\left(\genfrac{}{}{0pt}{}{a}{1+2a};-2\sqrt{z}\right); $$

A-14

```
F(U(A-(B))*U(A+B)*L(A)*L(1/2+A)*L(2*A),Z)→0
F(U(A-(B))*L(2*A),2*Z↑(1/2))*F(U(A-(B))*L(2*A),-2*Z↑(1/2))
```

$$ {}_2F_3\left(\genfrac{}{}{0pt}{}{a-b,\,a+b}{a,\,1/2+a,\,2a};z\right)={}_1F_1\left(\genfrac{}{}{0pt}{}{a-b}{2a};2\sqrt{z}\right){}_1F_1\left(\genfrac{}{}{0pt}{}{a-b}{2a};-2\sqrt{z}\right); $$

```
F(U(A)*U(1/2+A)*L(1/2+A-(B))*L(1/2+A+B)*L(2*A),Z)→0
F(L(1/2+A-(B)),1/4*Z)*F(L(1/2+A+B),1/4*Z)
```

$$ {}_2F_3\left(\genfrac{}{}{0pt}{}{a,\,1/2+a}{1/2+a-b,\,1/2+a+b,\,2a};z\right)={}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1/2+a-b};z/4\right){}_0F_1\left(\genfrac{}{}{0pt}{}{—}{1/2+a+b};z/4\right); $$

```
F(U(A)*U(A-(B))*U(A+B)*L(1/2+A)*L(2*A),Z)→0
F(U(1/2*A+(-1)/2*B)*U(1/2*A+1/2*B)*L(1/2+A),Z)↑2
```

$$ {}_3F_2\left(\genfrac{}{}{0pt}{}{a,\,a-b,\,a+b}{1/2+a,\,2a};z\right)=\left[{}_2F_1\left(\genfrac{}{}{0pt}{}{(a-b)/2,\,(a+b)/2}{1/2+a};z\right)\right]^2; $$

```
F(U(A)*U(A-(B))*U(A+B)*L(1/2+A)*L(-1+2*A),Z)→0
F(U(1/2*A+(-1)/2*B)*U(1/2*A+1/2*B)*L(1/2+A),Z)*F(U(1/2*A+(-1)/2*B)*U(1/2*A+1/2*B)*L(-1/2+A),Z)
```

$$ {}_3F_2\left(\genfrac{}{}{0pt}{}{a,\,a-b,\,a+b}{1/2+a,\,-1+2a};z\right)={}_2F_1\left(\genfrac{}{}{0pt}{}{(a-b)/2,\,(a+b)/2}{1/2+a};z\right){}_2F_1\left(\genfrac{}{}{0pt}{}{(a-b)/2,\,(a+b)/2}{-1/2+a};z\right); $$

```
F(U(A)*U(1/2+A-(B))*U(1/2+A+B)*L(1/2+A)*L(2*A),Z)→0
F(U(1/4+1/2*A+(-1)/2*B)*U(1/4+1/2*A+1/2*B)*L(1/2+A),Z)*F(U(-1/4+1/2*A+(-1)/2*B)*U(1/4+1/2*A+1/2*B)*L(1/2+A),Z)0
```

$$ {}_3F_2\left(\genfrac{}{}{0pt}{}{a,\,1/2+a-b,\,1/2+a+b}{1/2+a,\,2a};z\right)={}_2F_1\left(\genfrac{}{}{0pt}{}{(1/2+a-b)/2,\,(1/2+a+b)/2}{1/2+a};z\right){}_2F_1\left(\genfrac{}{}{0pt}{}{(-1/2+a-b)/2,\,(1/2+a+b)/2}{1/2+a};z\right); $$

MICROPROCESSORS WITH PROGRAMMABLE STRUCTURE AND MULTIPROCESSOR SYSTEMS WITH PROGRAMMABLE COMMUTATION

A.V. Kalyayev

Radioengineering Institute

Taganrog, U S S R

second slide (Patenting paper [illegible]

[illegible]

Microprocessors with Programmable Structure.

The development of LIC technology allows at present to implement new principles of microprocessor synthesis on a chip. These principles consist in application to microprocessors of high level languages based on a limited set of large operations rather than a large number of small elementary commands. Modern large integrated circuits give the opportunity to work out such microprocessor schemes that will allow programming of microprocessor structure but not the computation procedure thus performing adjusting for execution of one or another large operation from a given set of operations.

Consider design principles of microprocessors that instead of a set of microoperations $K=\{K_1,K_2,\ldots K_n\}$ execute large operations from a given set of macrooperations $O=\{O_1,O_2,\ldots O_n\}$. The set of macrooperations $O=\{O_1,O_2,\ldots O_n\}$ may be chosen sufficiently universal for solving any problem that may be solved using universal set of elementary commands K. A set of large operations must simplify the information exchange between microprocessor systems working in parallel. It is very important to choose the set of operations $O=\{O_1,O_2,\ldots O_n\}$ so that the adjusting of microprocessor for executing every particular operation be performed by the most simple method.

The author has developed the structural method of realization of large operations $O_i\in O$ in microprocessors (Fig.1).

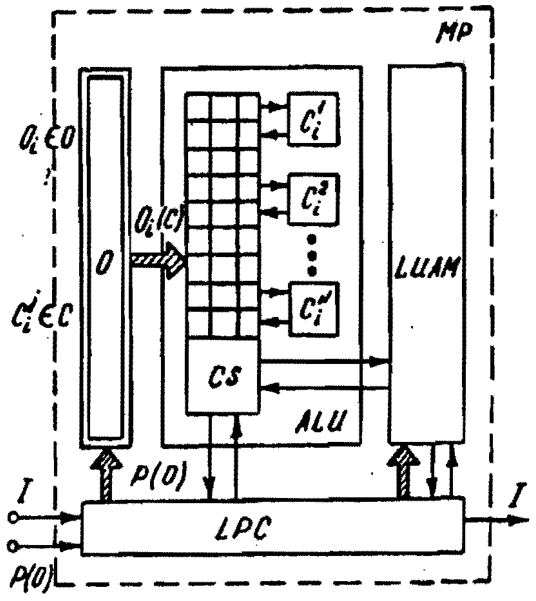


Fig.1

In this case every large operation O_i is performed due to the proper adjustment of the microprocessor internal structure. The microprocessor structure adjustment for performing large operation $O_i\in O$ is done by means of changing the internal commutation of separate structural portion of ALU of the microprocessor (Fig.1) due to the reconfiguration of special commutation structure being part of ALU. It gives the possibility to execute the large operation O_i as a whole, without splitting it in time into a sequence of elementary commands $K_j\in K$ All the necessary microoperations K_j comprised in the operation $O_i(K)$ are executed in parallel if one uses programming the microprocessor structure. As a result

1980 IEEE INT SYMP. CIR & SYS. PROCEEDINGS the execution time of the operation $O_i \in O$ sharply decreases. The information exchange betwsen the microprocessor and the memory beoomes simpler because in this case the programme $P(O)$ is written in a high level language and the loading of communication channels is less. Such a programme requires less memory size and makes the process of microprocessor programming simpler.

Sets of Large Operations of Microprocessors

To make the structure and the microprocessor adjustment for executing a concrete operation $O_i \in O$ simple the set of large operations of the microprocessor $O=\{O_1,O_2,\dots O_m\}$ must be minimized. On the other hand, in order to ensure the solution of any problems the set of macrooperations must be sufficiently universal. Besides, the set of large operations $O=\{O_1,O_2,\dots O_m\}$ must be of such a kind that the information exchange and problem distribution between microprocessor of a multiprocessor system be simple. It is also necessary that as far as possible the operations $O_i \in O$ should coincide with operations of common mathematical language. It will simplify the information exchange between the microprocessor structure and the programmer at most.

Surely one can form a set of large operations executed by microprocessors with programmable structure differently. Obviously, to obtain optimum variants of large operation sets is possible only on the basis of sufficiently profound theoretical investigations and extensive experimental work with multiprocessor systems using microprocessors with programmable structure. Right now however we can outline the ways of solution of this problem proceeding from the consideration of the most probable tasks and simulation objects realized in multiprocessor systems.

Tasks and simulated objects most often realized in multiprocessor systems include subsystems. The operators of these subsystems may be described by functional dependences; systems of algebraic and ordinary differential equations; partial differential equations; integral, vector and tensor transformations; logical transformations and dependences; matrix and recursive transformations etc.

Analysing the structure of the processors performing the large operations we may conclude that every large operation including functional transformation, integration of ordinary differential equations and partial differential equations, matrix, vector and tensor transformations, rotor, gradient and divergence operations as well as integration, differential and arithmetic operations are synthesized on the basis of some base system of operators which includes operators of summation Σ, multiplication M, differentiation D, integration I and the simplest logical operators L (comparison, sign etc.).

Any large operation synthesis comes to the realization of the corresponding combinations of the listed operators i.e. to the adjustment of the required operator commutation. Hence it is possible to construct a microprocessor with programmable structure with the help of some elementary processors realizing base operators and commutation system designed for connection of elementary processors (Fig.2).

Elementary processors necessary for realization of base operators may be synthesized differently. In the simplest case every elementary operator may realize only one concrete base operator. An elementary processor is more optimum, if it can realize any base operators and can be adjusted for realizing any concrete base operator from the given set

(Σ, M, I, D, L).

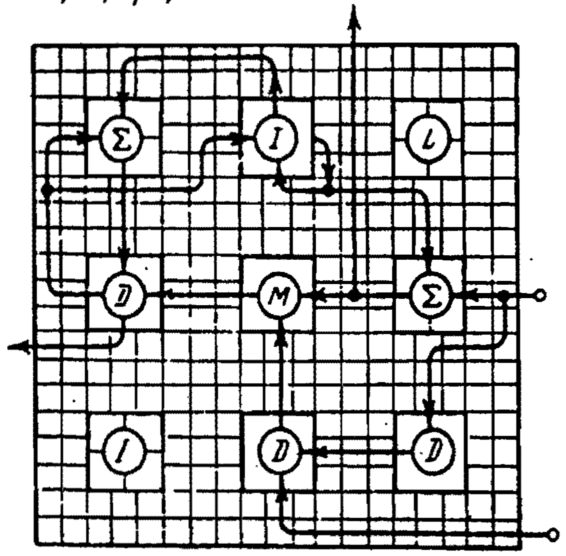


Fig.2

In the latter case we obtain a sufficiently universal microprocessor (Fig.2) whose structure may be programmed for performing a wide range of complex large operations considered partially in the previous section. Programming microprocessor structure for performing a large operation is realized in two stages - adjustment of elementary processors for performing the necessary base operators and adjustment of commutation structure for the formation of the necessary combinations of elementary processors.

Multiprocessor System with Programmable Architecture

Microprocessor with programmable structure provide us with ample possibility to construct multiprocessor structure that will enable us due to the parallel information processing to increase the computation speed, provide solution of many complex problems in real and speeding time scale, to raise reliability and vitality of computing systems and provide technology of their production.

Characteristics of multiprocessor computer structures depend on the basic principles of multiprocessor construction as well as on the structure architecture as a whole. Multiprocessor structure architecture is defined by methods of construction and reconfiguration of communication channels inside the structure and by the memory organization.

According to formation methods of communication channels multiprocessor structures may be divided into time shared bus structures, matrix structures, hierarchical structures and structures with universal commutation. Depending on the memory organization the mentioned multiprocessor structures may be also divided into multiprocessor structures with centred, common for all processors memory and into multiprocessor structures with distributed, individual for every processor memory.

The greatest speed and flexibility

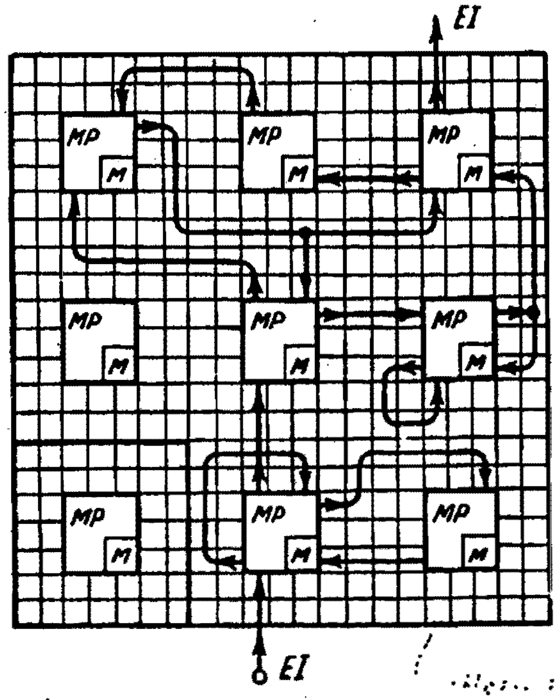


Fig.3

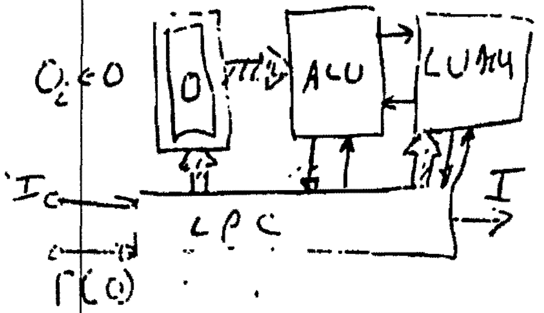
[illegible] Commutation system

result from synthesising multiprocessor structures with distributed memory and universal commutation. The processors of these structures work on the basis of a large operation set and structure programming. Fig.3 shows a microprocessor structure with distributed memory and universal communication. Information exchange between the microprocessors and the memory is realized through rigid straight communication channels. Information exchange between the microprocessors goes through straight flexible electronic communication channels synthesized by the programming method in a special homogeneous commutation structure.

The adjustment of a similar multiprocessor structure for solving one or another problem consists in multiprocessor structure programming for executing large operations and in programming communication channels between microprocessors in the commutation structure. As a result programming multiprocessor computing structure with distributed memory and universal commutation will be performed in the language of a sufficiently high level and this will result in the facilitation of the programming process. High speed of information processing and maximum simplicity of distributing separate tasks between microprocessors working in parallel are provided by multiprocessor structure with distributed memory and universal commutation and by realization large operations in microprocessors.

diff eq
partial " "
matrix

Input slide



1
2
3
4
5
6 [illegible] ops
7 arithmetic ops Σx_i, Πx_i, $\Sigma a_i x_i$, $\Sigma a_i x^i$
8 logical ops

REFERENCES

Agarwal, R. C., and C. S. Burrus, "Number Theoretic Transforms to Implement Fast Digital Convolution," *Proceedings of the IEEE*, 63, pp. 550-560, April 1975.

Akayev, A. A., S. A. Mayorov, Ye. A. Chernyavskiy, *Coherent Optical Computing Machines: Optoelectronic Table Processors*, Kogerentnyye opticheskiye vychislitel'nyye machiny. Leningrad, Izd'vo Mashinostronyeniye (Leningradskoye otdeleniye), 1977, 439p. [Translation]

Akushskiy, I. Ya. and A. Ya. Pyanzin. *Specialized computing device*. Opizaniye izobreteniya k avtorikomu svidetel'stvu. Patent No. 368597. Arithmetic to Residue Code Converter. [Translation]

Akushskiy, I. Ya., V. M. Amerbayev, V. S. Kokorin, L. G. Rykov, and D. I. Yuditskiy. *Residue class number system to polyadic code converter*. Opisaniye izobreteniya k avtorskomy svidetel'stvu. Patent No. 328448, 28 Apr 1972. [Translation]

Akushskiy, I. Ya., D. I. Yuditskiy. *Machine Arithmetic in Residual Classes*, Mashinnaya Arifmetika v Ostatochnykh Klassakh, Publishing House "Sovetskoye Radio," Moscow, 1968, pp. 1-439. English pages: 717. [Translation]

Akushskiy, I. Ya., V. M. Amerbayev, I. T. Pak. *Principles of the Machine Arithmetic of Complex Numbers*, Osnovy Mashinnoy Arifmetiki Kompleksnykh Chisel, Publishing House "Nauka," Alma-Ata, 1970, pp. 1-248. English pages: 347. [Translation]

Backus, John, "1977 ACM Turing Award Lecture: Can Programming be Liberated from the von Neumann Style? A Functional Style and its Algebra of Programs," *Comm. ACM*, 21, 8, pp. 613-641, Aug. 1978.

Cheney, P. W., "A Digital Correlator Based on the Residue Number System," *I.R.E. Trans. on Electronic Computers*, pp. 63-70, March 1961.

Cole, C. A., S. Walham, et al., *SMP: A Symbol Manipulation Program*, Version 1, California Institute of Technology, Pasadena, CA, July 1981.

Collins, S., J. Mckay, and C. Vick, *Digest of Papers of Compcon Spring 78*, IEEE Comp. Soc., San Francisco, IEEE, p. 198, 1978.

Cooley, J. W. and J. W. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Math. Comput.*, 19, pp. 297-301, April 1965.

Despain, Alvin M. "Very Fast Fourier Transforms; Algorithms for Hardware Implementation," *IEEE Trans. Comput.*, C-28, pp. 333-341, May 1979.

Despain, A. M., "Fourier Transform Computers Using CORDIC Iterations," *IEEE Transactions Computation*, C-23, pp. 993-1001, Oct. 1974.

Fateman, Richard J., "Symbolic and Algebraic Computer Programming Systems," *SIGSAM Bull.*, 15, 1, pp. 21-32, Feb. 1981.

Garner, H. L., "The Residue Number System," *I.R.E. Trans. on Electronic Computers*, EC-8, 2, pp. 140-147, June 1959.

Gauss, K. F., *Disquistiones Arithmeticae*, Translated by Arthur A. Clarke, Yale University Press, New Haven, Connecticut, 1966.

Huang, A., "Implementation of a Residue Arithmetic Unit Via Optical and Other Physical Phenomena," *Proceedings of the International Optical Computing Conference, Washington, D.C.*, April 1975.

Huang, A., Y. Tsunoda, J. W. Goodman, *Optical Computation Using Residue Arithmetic*, (Annual Technical Report No. 6422-1 for AFOSR-77-3219), Stanford Electronics Laboratories, Stanford, CA, March 1978.

Huang, A., "System for Processing Arithmetic Information Using Residue Arithmetic." United States Patent No. 4,107,783, August 1978.

Huang, A., Y. Tsunoda, J. W. Goodman, and S. Ishihara, "Optical Computation Using Residue Arithmetic," *Applied Optics*, 18, 2, pp. 149-162, January 1979.

Huang, A., J. Mandeville, J. W. Goodman, S. Ishihara, *System Considerations in the Design of Residue Processors*, (Annual Technical Report No. L722-2 for AFOSR-77-3219), Stanford Electronics Laboratories, Stanford, CA, March 1979.

Huang, C. H., and F. Taylor, "A Memory Compression Scheme for Molulo Arithmetic," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, ASSAP-27, 6, pp. 608-611, December 1979.

Huang, C. H., and F. Taylor, "High Speed DFT Using the Residue Number System," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, Denver, 1980*, pp. 238-242, April 1980.

Jenkins, W. K., and B. J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters," *IEEE Trans. Circuits Systems*, CAS-24, pp. 191-201, April 1977.

Jullien, G. A., "Residue Number Scaling and Other Operations Using ROM Arrays," *IEEE Trans. Computers*, C-27, pp. 325-337, April 1978.

Kantor, I. L., A. S. Solodovnikov, *Hypercomplex Numbers*, Giperkompleksnyye Chisla, Publishing House "Nauka," Moscow, 1973, pp. 1-144. English pages: 258. [Translation]

Kasper, T., "Integration in Finite Terms: The Liouville Theory," *SIGSAM Bull.*, 14, 4, pp. 2-8, Nov. 1980.

Knuth, Donald E., "The Art of Computer Programming," *Seminumerical Algorithms*, 2, Addison-Wesley, Reading, Mass.

Kowalski, R, *Logic for Problem Solving*, North Holland, N.Y., 1979.

Lockheed Missiles & Space Company, *Modular Arithmetic Techniques*, by R. I. Tanaka et al., 2-38-62-1A, ASD TDR 62-686, Sunnyvale, CA, Nov. 1962.

Lohmann, A. W., Private Conversations with N. Basov.

MacLane, S., "Mathematics," *Science*, 209, 452, pp. 104-110, 4 July 1980.

Martin, W. A. and R. J. Fateman, "The MACSYMA System," *Proc. Second Symp. Symbolic and Algebraic Manipulation*, S. R. Petrick, ed., American Chemical Society, Washington, D.C., pp. 59-75, 1971.

McClellan, J. H., and C. M. Rader, *Number Theory in Digital Signal Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1979.

Miola, A. M., "A Bibliography of Soviet Works in Algebraic Manipulations," *SIGSAM Bull.*, 15, 1, pp. 5-7, Feb. 1981.

Nussbaumer, H., *Fast Fourier Transform and Convolution Algorithms*, Springer-Verlag, N.Y., 1981.

Orlov, L. A., Yu. M. Popov, "Optoelectronic arithmetic unit using the residue class number system,*" *Avtometriya*, 6, 14-23, 1972. [Translation]

Pavelle, Richard, Michael Rothstein, and John Fitch, "Computer Algebra," *Sci. Am.*, 244, 12, Dec. 1981.

Rader, C. M., "Discrete Convolution via Mersenne Transforms," *IEEE Trans. on Computer*, C-21, pp. 1269-1273, December 1972.

Radio Corporation of America, Aerospace Systems Division, *Modular Arithmetic Computer Research*, AL TDR 64-86, Burlington, MA, April 1961.

Sin'kov, M. V., N. M. Gubareni, *Nonpositional Representations in Multidimensional Numerical Systems*, Nepozitsionnyye Predstavleniya v Mnogomernykh Chislovykh Sistemakh, Publishing House "Naukova Dumka," Kiev, 1979, pp. 1-137. English pages: 225. [Translation]

Slagle, James R., "A Heuristic Program That Solves Symbolic Integration Problems in Freshman Calculus," *Computers and Thought*, McGraw-Hill, New York, N.Y., 1963.

Soderstrand, M. A., "A High Speed Low-Cost Recursive Digital Filter Using Residue Number Arithmetic," *Proceedings IEEE*, 65, pp. 1065-1067, July 1977.

Stoner, W. and F. Horrigan, "Residue-Based Optical Processors," *Proceedings of the Conference on Optical Processing Systems*, SPIE, 185, Bellingham, WA, pp. 19-29, 1979.

Svoboda, A., "Computer Progress in Czechoslovakia," in *Digital Information Processors*, W. Hoffman, Editor, John Wiley and Sons, Inc., N.Y., 1962.

Szabo, N., and R. Tanaba, *Residue Arithmetic and its Applications to Computer Technology*, McGraw-Hill, New York, N.Y., 1967.

Tai, A., I. Cindrich, J. R. Fienup, and C. C. Aleksoff, "Optical Residue Arithmetic Computer with Programmable Computation Modules," *Applied Optics*, 18, 16, pp. 2812-2823, August 1979.

Tseng, B. D., G. A. Jullien, and W. C. Miller, "Implementation of FFT Structures Using the Residue Number System," *IEEE Trans. on Computers*, C-28, 11, Nov. 1979.

Vyshinskiy, V. A., Z. L. Rabinovich, "A method for substantially increasing the processing capacity of computers of accelerating matrix-vector operations," *Kiberneticka*, 1, 120-123, 1979. [Translation]

Winograd, S., "On the Time Required to Perform Addition," *Journal of the ACM*, 12, 2, pp. 277-285, April 1965.

Winograd, S., "On the Time Required to Perform Multiplication," *Journal of the ACM*, 14, 4, pp. 793-802, October 1967.

DISTRIBUTION LIST

Dr. Saul Amarel
25 White Pine Lane
Princeton, NJ 08540

Dr. Marv Atkins
Deputy Director, Science & Tech.
Defense Nuclear Agency
Washington, D.C. 20305

Prof. Peter M. Banks
23 Peter Coutts Circle
Stanford, CA 94305

Dr. Curtis G. Callan, Jr.
Department of Physics
Princeton University
Princeton, NJ 08540

Mr. Dean O. Carhoun [4]
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

Dr. Forrest L. Carter
Chemistry Division, Code 621-75
Naval Research Laboratory
Washington, D.C. 20375

Dr. Robert Cooper [2]
Director, DARPA
1400 Wilson Boulevard
Arlington, VA 22209

Defense Technical Information [2]
Center
Cameron Station
Alexandria, VA 22314

The Honorable Richard DeLauer
Under Secretary of Defense (R&E)
Office of the Secretary of
Defense
The Pentagon, Room 3E1006
Washington, D.C. 20301

Dr. Alvin M. Despain
1192 Grizzly Peak Boulevard
Berkeley, CA 94708

Director [2]
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note

CAPT Craig E. Dorman
Department of the Navy, OP-095T
The Pentagon, Room 5D576
Washington, D.C. 20350

CDR Timothy Dugan
NFOIO Detachment, Suitland
4301 Suitland Road
Washington, D.C. 20390

Dr. Larry Gershwin
NIO for Strategic Programs
P.O. Box 1925
Washington, D.C. 20505

Dr. S. William Gouse, W300
Vice President and General
Manager
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102

Dr. Edward Harper
SSBN, Security Director
OP-021T
The Pentagon, Room 4D534
Washington, D.C. 20350

Mr. R. Evan Hineman
Deputy Director for Science
& Technology
P.O. Box 1925
Washington, D.C 20505

(b)(3):50 USC §403(g) Section
CIA/DDS&T
P.O. Box 1925
Washington, D.C. 20505

The MITRE Corporation [25]
1820 Dolley Madison Blvd.
McLean, VA 22102
ATTN: JASON Library, W002

Dr. Robert E. Kahn [2]
DARPA/IPTO
1400 Wilson Boulevard
Arlington, VA 22209

Mr. Jack Kalish
Deputy Program Manager
The Pentagon
Washington, D.C. 20301

Mr. John F. Kaufmann
Dep. Dir. for Program Analysis
Office of Energy Research, ER-31
Room F326
U.S. Department of Energy
Washington, D.C. 20545

Mr. Edwin L. Key [2]
The MITRE Corporation
O.O. Box 208

Dr. George A. Keyworth
Director
Office of Science & Tech. Policy
Old Executive Office Building
17th & Pennsylvania, N.W.
Washington, D.C. 20500

MAJ GEN Donald L. Lamberson
Assistant Deputy Chief of Staff
(RD&A) HQ USAF/RD
Washington, D.C. 20330

Dr. Donald M. LeVine, W385 [3]
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102

Mr. V. Larry Lynn
Deputy Director, DARPA
1400 Wilson Boulevard
Arlington, VA 22209

Dr. Gordon MacDonald, W300
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, VA 22102

Dr. Joseph Mangano [2]
DARPA/DEO
9th floor, Directed Energy Office
1400 Wilson Boulevard
Arlington, VA 22209

Mr. John McMahon
Dep. Dir. Cen. Intelligence
P.O. Box 1925
Washington, D.C. 20505

Director
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note

Dr. Marvin Moss [3]
Technical Director
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

(b)(3):50 USC §403(g) Section 6
P.O. Box 1925
Washington, D.C. 20505

Director
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note
DDR-FANX 3

Prof. William A. Nierenberg
Scripps Institution of
Oceanography
University of California, S.D.
La Jolla, CA 92093

Dr. Nils J. Nilsson
EJ-247
Artificial Intelligence Center
SRI International
333 Ravenswood Avenue
Fenlo Park, CA 94025

Dr. Allen M. Peterson
SRI International, (EL-154)
333 Ravenswood Avenue
Menlo Park, CA 94025

Mr. Alan J. Roberts
Vice President & General Manager
Washington C^3 Operations
The MITRE Corporation
1820 Dolley Madison Boulevard
Box 208
McLean, VA 22102

Los Alamos Scientific Laboratory
ATTN: C. Paul Robinson
P.O. Box 1000
Los Alamos, NM 87545

Mr. Richard Ross [2]
P.O. Box 1925
Washington, D.C. 20505

Dr. Oscar S. Rothaus
106 Devon Road
Ithaca, NY 14850

Dr. Phil Selwyn
Technical Director
Office of Naval Technology
800 N. Quincy Street
Arlington, VA 22217

Dr. Eugene Sevin [2]
Defense Nuclear Agency
Washington, D.C. 20305

Dr. Joel A. Snow [2]
Senior Technical Advisor
Office of Energy Research
U.S. DOE, M.S. E084
Washington, D.C. 20585

Mr. Alexander J. Tachmindji
Senior Vice President & General
Manager
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

Dr. Vigdor Teplitz
ACDA
320 21st Street, N.W.
Room 4484
Washington, D.C. 20451

Dr. Al Trivelpiece
Director, Office of Energy
Research, U.S. DOE
M.S. 6E084
Washington, D.C. 20585

Dr. John F. Vesecky
Center for Radar Astronomy
Stanford University
Stanford, CA 94305

Mr. James P. Wade, Jr.
Prin. Dep. Under Secretary of
Defense for R&E
The Pentagon, Room 3E1014
Washington, D.C. 20301

Mr. Leo Young
OUSDRE (R&AT)
The Pentagon, Room 3D1067
Washington, D.C. 20301

Mr. Mark Zimmerman [5]
P.O. Box 1925
Washington, D.C. 20505

APPENDIX A-1

FORTRAN PROGRAM PRODUCED BY MACSYMA

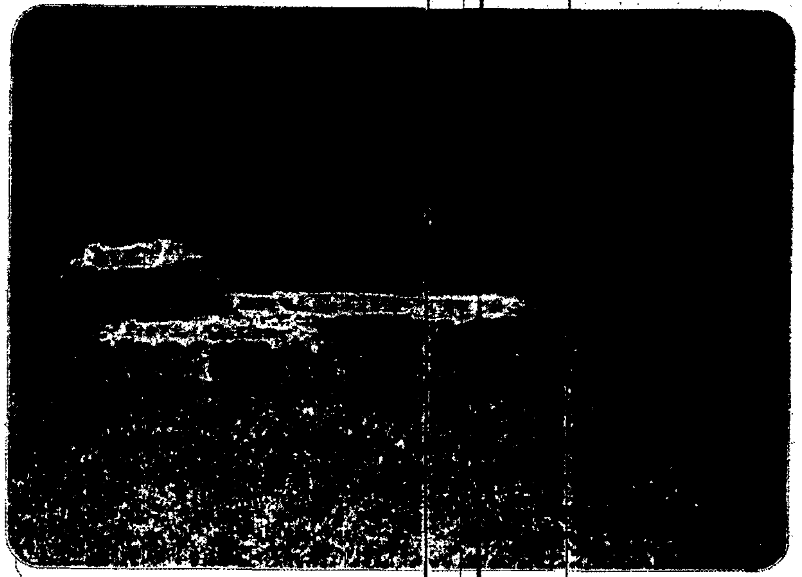
APPENDIX A-2

REPRINT OF "A BIBLIOGRAPHY OF SOVIET WORKS IN ALGEBRAIC MANIPULATIONS"

by

Alfonso M. Miola
[SIGSAM Bull., 15, 1, February 1981.]

[This page is intentionally left blank.]



doc. 9

2

Report on the Workshop for Automated Software Programming

MITRE

[This page is intentionally left blank.]

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

19. KEY WORDS (Continued)
20 ABSTRACT (Continued)

DD FORM 1 JAN 73 1473 (BACK)

EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

WORKSHOP PARTICIPANTS

Despain, Alvin M. (Workshop Leader)	JASON/University of California, Berkeley
Abarbanel, Henry	JASON/University of California, San Diego
Adams, Duane	Self
Amarel, Saul	Rutgers University
Banks, Peter	JASON/Stanford University
Brown, Richard	MITRE
Callan, Curt	JASON/Princeton
Case, Ken	JASON/The Rockefeller University
Cuadrado, John	I.D.A.
Druffel, Larry	Rational
Eardley, Doug	JASON/University of California, Santa Barbara
Frank, Geoffrey	R.T.I.
Goldberg, Allen	Kestral
Hammer, Dave	JASON/Cornell University
Jarvis, Theodore	MITRE
Jullig, Richard	Kestrel
Leveson, Nancy	University of California, Irving
MacDonald, Gordon	JASON/MITRE
Nierenberg, Bill	JASON/Scripps Institute of Oceanography
Offutt, James	S.D.I.O.
Peterson, Allen	JASON/Stanford University
Probert, Thomas	I.D.A.
Riddle, Bill	Self
Rothaus, Oscar	JASON/Cornell University
Vesecky, John	JASON/Stanford University
Waldinger, Richard	Stanford Research Center

TABLE OF CONTENTS

Page

LIST OF ILLUSTRATIONS vii

EXECUTIVE SUMMARY ix

1.0 INTRODUCTION 1-1

1.1 Description of Workshop 1-1
1.2 The Software Problem 1-2
1.3 Signal Processing 1-6
1.4 Related NASA Space Station Activities 1-10

2.0 ISSUES 2-1

2.1 Introduction 2-1
2.2 ADA 2-2
2.3 Software Engineering 2-4
2.4 Current Programming Paradigm 2-5
2.5 Bottom-up Design 2-8
2.6 Conclusions 2-8

3.0 AUTOMATED PROGRAMMING 3-1

3.1 Introduction 3-1
3.2 A Computer-Aided Specification System 3-3
3.2.1 A New Language 3-4
3.2.2 Conversion to English 3-7
3.2.3 Specialized Editors 3-7
3.2.4 Expert System for Specific Application Areas 3-8
3.3 Automatic Program Generation 3-9
3.3.1 Program Development Support 3-10
3.3.2 Algorithm Design 3-11
3.3.3 Automatic Program Analysis 3-12
3.3.4 Program Optimization 3-12
3.4 Conclusions 3-13

4.0 CONCLUSIONS AND RECOMMENDATIONS 4-1

REFERENCES R-1

APPENDIX A: AGENDA A-1

LIST OF ILLUSTRATIONS

Figure		Page
1	Layered model of battle management [From Fletcher Panel Report]	1-5
2	Data flow for a single platform [From Fletcher Panel Report]	1-8
3	Current programming paradigm [Balzar, 1985]	2-6
4	Automation based paradigm [Balzar, 1985]	3-2
5	Automatic specification system [proposed]	3-5

Table		
1	Some complex functions to be implemented by software [Probert, 1985]	1-3
2	SDI battle management system (BMS) [Probert, 1985]	1-7

EXECUTIVE SUMMARY

The Workshop for Automated Software Programming, was sponsored by the Strategic Defense Initiative Organization (SDIO), under the auspices of JASON. The purpose of the workshop was to examine the key issues surrounding the generation of software for the SDI system and in particular the merits of automating part of this process.

The participants included representatives from government, academia, and industry with experience in SDI problems, software system development and automated programming.

First, SDI concepts were examined. Next, issues in software systems were considered. Then automated programming was examined.

It was discovered that there are only very rough estimates of just how big the software problem is, except that it is likely to far exceed in size and cost any programming project yet attempted. It is doubtful that all [illegible] software can be built with only today's technology.

Current research and development suggests that programmer productivity might be greatly improved by a combination of new technologies.

We make the following recommendations:

1. Develop a Computer-Aided Specification System (CASS) containing an expert system core that can be adapted to various applications.

2. Develop a Very High Level Language (BOOLE) similar to the way ADA was developed.

3. Develop an Automated Programming System (APS) in cooperation with the DARPA program.

4. Establish Conferences, Summer Studies, "Schools", etc., to train more professionals in advanced programming techniques and especially in the techniques of building automatic specification and automatic programming systems.

5. Develop a library (SDIL) of re-useable subprograms, both independently and in cooperation with the STARS Program.

6. Develop Software Standards for SDI (SDISS)
   a. ADA
   b. IEEE Floating Point Arithmetic

c. Communication Protocols

d. Data Base Formats & Organization

e. Graphics Display

f. System Calls

g. Network Protocols

7. Begin top-level specif'-ation of software soon so that more realistic estimates of the true size of the software problem can be made.

8. Track and cooperate with the Space Station software work.

1.0 INTRODUCTION

1.1 Description of the Workshop

A workshop for Automated Software Programming, sponsored by the Strategic Defense Initiative Organization (SDIO) and under the auspices of JASON, was conducted 1-3 July 1985 in La Jolla, CA. The purpose of the workshop was to examine the key issues surrounding the generation of software for the SDI system and in particular the merits of automating part of this process.

The participants included a number of JASON's interested in this problem and a diverse set of representatives from various government, academic, and industrial laboratories. About 1/3 of the participants had worked on some aspect of SDI problems, about 1/3 had experience with large software system development and about 1/3 were doing active research in automated programming.

First, as can be seen from the workshop agenda (see Appendix A), the SDI concepts and especially how these related to the SDI software problem were examined. Next, experiences with large software systems were considered. Then the state of automatic programming research was presented. At the end, we discussed what conclusions we could come to and what recommendations to SDIO we could agree on.

1.2 The Software Problem

The SDI battle management system (estimated to be about 20% of the total software required for the SDI program) will consist of a number of defense layers, each of which must handle between 10^4 and 10^6 objects moving in space. Computers (with software) must provide real-time management of most of the information processing and decisions in a rapidly evolving battle. All of this must be accomplished in the face of unreliable systems and the unpredictable loss of system components due to battle damage. Despite this, the software must be designed to maintain system integrity and effectiveness during the battle.

The number of functions and the complexity of each is very large. Table 1 illustrates some of the functions that must be performed by the software. It can be seen that the design of the software poses an unprecedented challenge.

How "large" is this challenge? We have only very rough estimates. For example, Bell Telephone Laboratories, in solving a much smaller, but related set of problems, in the Safeguard BMD system, generated some two million lines of code. This required about 1,200 people over a period of six years (approximately one line of code

TABLE 1

SOME COMPLEX FUNCTIONS TO BE IMPLEMENTED BY SOFTWARE
[Probert, 1985]

COMMUNICATIONS NETWORK

- Data relay - packet assembly/disassembly
- Data error detection/correction
- Security
- Protocol conversion
- Load management
- Hardware/software reconfiguration
- Cross-talk and hand-over algorithms
- Diagnostics

SURVEILLANCE PLATFORMS

- Target recognition and discrimination algorithms
- Sensor control and sensor data processing
- Target tracking algorithms
- Autonomous action rules
- Electronic warfare algorithms
- Mission execution algorithms

COMMAND AND CONTROL

- Battle control algorithms
- Target assignment (multiple assignments)
- Object tracking calculations (global view)
- Weapon selection
- Data base management
- Autonomous action rules
- User/machine interfaces

WEAPON SYSTEMS

- Navigation
- Propulsion
- Terminal control
- Sensor control
- Electric power management
- Self defense
- Attitude control
- Fire control
- Target acquisition/track
- Beam control

per person each work day, [3]. The full set of SDI software will be much larger due to the much greater complexity of the SDI architecture as compared to the BMD terminal defense software.

It is possible to see some of this complexity from the outline for battle management as presented in the Fletcher Panel Report [3]. Figure 1 illustrates a layered model of battle management. The BMD software referred to above represents only a fraction of the box labeled "Terminal Battle Management" because the BMD software was for *isolated* terminal defense, but not *coordinated*, *multilayered*, defense as shown in the model.

Each of the boxes in Figure 1 represents *unique* functionality corresponding to the environment, etc., in which it operates. Hence *unique* software is required for each box. Now a very crude estimate can be made. There are six boxes, each of greater than 2×10^6 lines of code or a total of more than 10^7 lines of code.

The effort to program large complex systems does not, unfortunately, grow linearly with the size of the problem. It grows *much* faster. This phenomena is explained in the famous book by Fred Brooks "The Mythical Man-Month" [28], in which he illustrates that if twice as much code is to be produced it requires, not twice the "Man-Months", but many more.

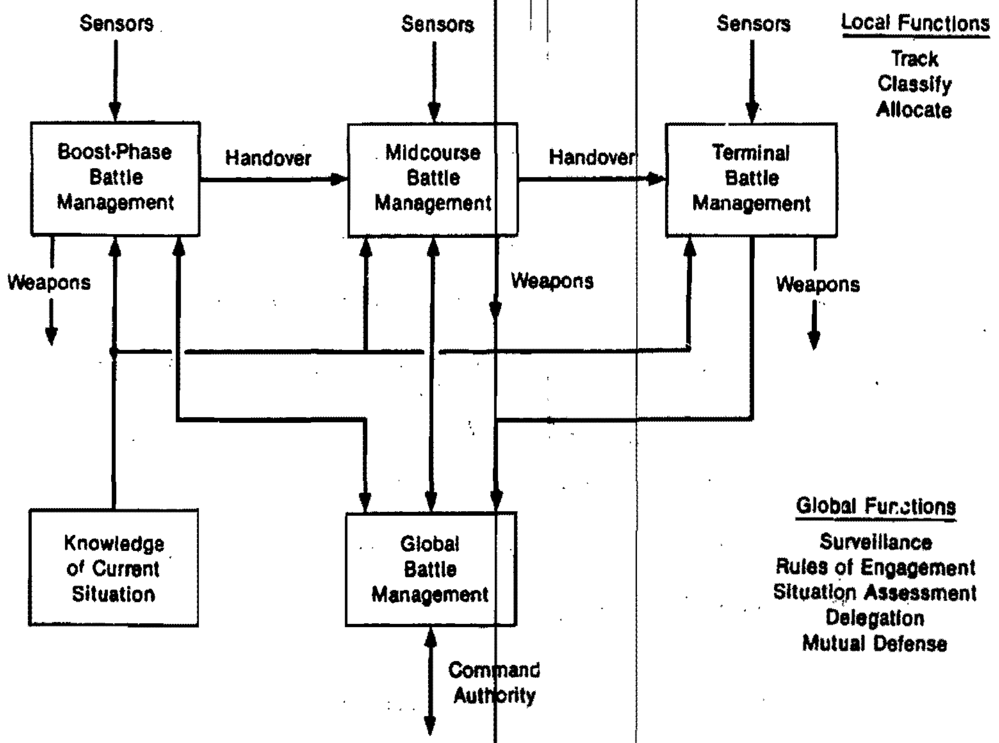


Figure 1. Layered model of battle management. [From Fletcher Panel Report]

The problem of modelling the programmer time needed to create software has been studied for more than 20 years. While we have little faith in the predictive power of any of these models, we have little else to rely on. Recently a group at IDA, attempted to estimate the programming effort required for the SDI using the best of these models [1]. The results as illustrated in Table 2, indicate 14 to 28 years will be required to produce the software. These are very crude estimates that can only indicate the seriousness of the software problem, not provide any reliable estimate for planning.

1.3 Signal Processing

Figure 2 is a schematic of the data flows in a computer system typical of a single platform. It provides some idea of the computer processing requirements. In examining the software programming requirements to accomplish this processing, it is soon evident that while the bulk of the data processing is needed in the sensor signal processing, that the problem of programming this, while a large effort, is not the critical part of the software problem and can most likely be handled using current technology. The remainder of the software supports very complex decisions, etc., and this type of software is notorious for its difficulty.

TABLE 2

SDI BATTLE MANAGEMENT SYSTEM (BMS)
[Probert, 1985]

- Best case - 19 million lines of code
  - -- 45,480 person years of effort
  - -- 14 years minimum time to deliver

- Worst case - 35 million lines of code
  - -- 94,666 person years of effort
  - -- 18 years minimum time to deliver

- Best case with support software - (3:1)
  - -- 118,858 people years over 22 years

- Worst case with support software
  - -- 247,402 people years over 28 years

- Cost (with support software)
  - -- Best case - $12 - 18 billion
  - -- Worst case $25 - 37 billion

- Target allocation function of BMS alone is 20% of total SDI

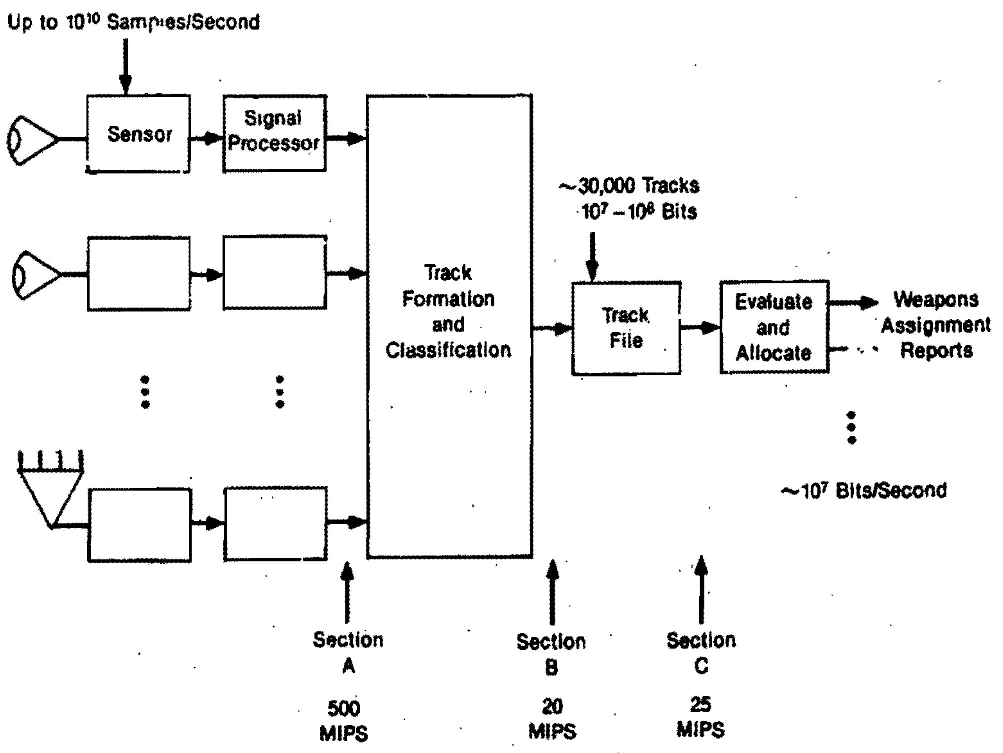


Figure 2. Data flow for a single platform. [Fletcher Panel Report]

From the brief examination we made of the SDI software problem we were able to conclude the following things:

1. It is evident that no one understands, in any detail, just how big the software development problem is. It is recommended that SDIO immediately focus on a more detailed estimate of the SDI software requirements.

2. It is highly unlikely that the software for SDI can be created using today's technology. Thus some improvement in programmer productivity will need to come from research breakthroughs, probably in the area of automatic programming and especially in the area of computer-aided requirements specification.

3. Due to the apparent need to develop new software technology and to expand the base of programming professionals, SDIO should consider building the base of programming professionals by sponsoring conferences, workshops, "schools", summer studies, etc., in generic software technologies critical to the SDIO task.

1.4 Related NASA Space Station Activities

NASA's Space Station program is now well into the planning phase with a separate (code S) administrative group at Headquarters. Although on a much smaller scale, many issues concerning Space Station are analogous to issues in SDI, including software. The current Space Station concept includes not only a manned station in a low inclination orbit, but also one or more satellites in polar orbit at higher altitudes and possibly an unmanned platform co-orbiting with the manned platform. This constellation of satellites must handle data at rates in the hundreds of megabits per second. The analogy with SDI is clear.

Some areas of common interest between SDI and Space Station are listed below. Many more are likely to emerge as the programs progress.

1. Control of multiple, interrelated spacecraft

2. High data rate communications between multiple spacecraft and multiple ground stations

3. Control of observation and response with inputs and outputs at multiple nodes

4. Flexible decision making keyed to a series of observations

5. Intensively networked computer systems

These issues involve massive software development as does SDI. Hence we recommend that SDI software programs track and cooperate with analogous NASA Space Station software efforts.

A part of the space Station program with particular interest for SDI is what is being called "Telescience". The idea is that an experimenter on the Earth or possibly the manned Space Station Platform remotely conducts an experiment on one of the space station platforms. Thus observations are made, appropriate data conveyed to the remote experimenter and the experimenter then alters the experiment parameters in accordance with observations as they are made. To support this activity NASA is developing the necessary communications and data management tools. Although on a different scale than SDI, Telescience will generate many of the same software problems as SDI. Hence exchange of information and possible cooperation with NASA would seem fruitful. The Telescience program is now using the Space Shuttle as a testbed. Information arising out of this activity should be of direct interest to SDI and software development in particular.

2.0 ISSUES

2.1 Introduction

The next part of the workshop considered the past and current approaches to large software systems, with the goal of identifying what the critical issues might be for the SDI software. There was considerable discussion of the development, usage, utility and deficiencies of the DOD developed language "ADA" [5]. Then the STARS [6] program, designed to provide an order of magnitude improvement in defense software, was considered. Next a series of new approaches to the development of complex software systems was considered. Some issues of software safety were also explored.

The following issues were identified and discussed:

1. What is the proper role of Software Engineering?

2. Is real-time, distributed, problem solving even possible?

3. How can we come to trust the software?

4. How can such large, complex, operational software systems ever be tested?

5. Can the software be made safe and secure?

6. How can performance of the software be monitored?

7. Can developments in Software Engineering and related technologies be expected to lead to large improvements in productivity?

2.2 ADA

ADA is a computer language recently developed by DOD [5] to achieve a large improvement in the cost of creating and maintaining DOD software systems. The workshop group included participants that had used ADA themselves, managed programmers that used ADA, analyzed ADA effectiveness, and managed the developement of an ADA programming environment. It would be fair to characterize the workshop as being dominated by ADA fans as opposed to ADA critics [6]. Still, less than 1/2 of the participants employed ADA in their own programming work. It was reported at the workshop that more than a million and one-half lines of ADA code had been completed in creating the ADA support environment. This was done by highly competent, small teams, in a period of about four years. It appears that use of ADA probably

2.4 Current Programming Paradigm

The construction of software is foremost a design process. The programmer is an integral member of the over-all system design team. The general process of design begins with a high level, usually somewhat vague set of statements of what is desired of the system to be designed. For SDIO this is President Reagan's call to "Eliminate the threat posed by nuclear ballistic missiles". The system designers must understand what general sorts of component systems can be brought to bear to help achieve this goal. An important part of the design process is then to first generate a set of informal requirements, which if they are satisfied, will achieve the goal. As studies are conducted, design alternatives explored, etc., these requirements become more formal and eventually become specifications. Part of these requirements become the input for the next level of design, perhaps for a major software system. The programmer is then faced with the task of converting his given level of informal requirements into a computer program, which is the ultimate formal specification of computer behavior. This design goes by several names such as "step-wise refinement." Figure 3 illustrates this process (for programming).

It was mentioned earlier that programmers only achieve about one line of production code per hour. It only takes about 10 seconds to

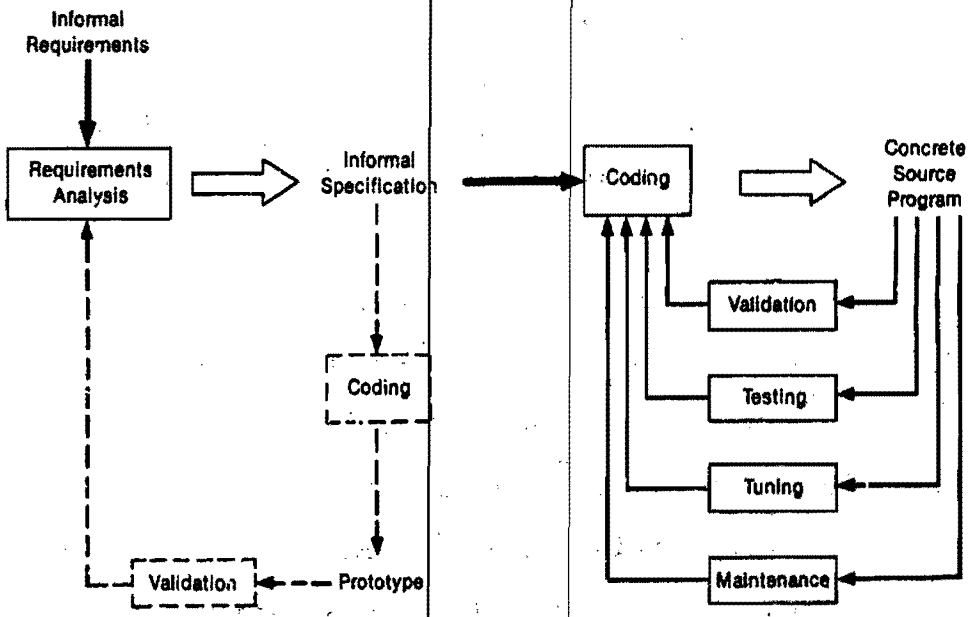


Figure 3. Current programming paradigm. [Balzar, 1985]

actually type such a line of code. What happens for the remainder of the hour? As for any intense intellectual activity there is no simple answer. However much of the time is spent trying to understand, from the vague requirements, what is really needed. Figure 3 illustrates the current programming paradigm. What is clear from this observation is that the SDI software problem is not dominated by problems of code generation from formal requirements, but the problem of developing a *final*, *correct*, *formal* specification of the software function. As is always the case in large complex system design, the errors and misunderstandings mostly occur at the interfaces between independently developed modules.

SDIO thus needs to develop a very high level specification language (BOOLE) and an associated computer-aided specification system. Such a system should allow both non-programmers (but technically sophisticated engineers) and programmers to accept informal requirements and turn these into a set of formal specifications. Later, as much of this process is automated, it may be possible for the system to automatically generate specifications for component sub-systems, given its knowledge data base and a formal input specification. This is a critical task, as the architecture of the SDI system is likely to be re-designed many times as new technologies, etc., come to light.

2.5 Bottom-up Design

An alternative design method is "bottom-up design" which proceeds by building larger components from smaller ones, until the desired system appears. In terms of software, this is the process of building libraries of reusable subroutines such as the Collected Algorithms of ACM or the IMSL Library. Since much of the low level software for SDI will no doubt be common across many programming modules, this is a good design strategy for getting started. To make this approach work, it is very important to establish standards before the libraries are started. Later the library subroutines are provided to a top-down design system as components, and are incorporated into the evolving design. The STARS program is beginning to develop many such reusable sub-routines in ADA, and many of these should be useful to the SDI.

2.6 Conclusions

It can be concluded that software engineering techniques will be needed for the SDI software task but will not be sufficient. It is recommended that SDI develop more powerful software engineering tools by automating many of the activities that human programmers now perform.

The most serious problem in generating massive, but reliable software will be getting the specifications straight and in a formal form. A Computer-Aided Specification System (CASS) to aid engineers and programmers should be developed.

ADA should be adopted as a standard language by SDI but there should also be a very high level specification language (BOOLE) as well. SDI should sponsor the development of BOOLE.

SDIO should work with the STARS program to establish standards for and libraries of reusable subroutines. SDIO should also begin development of libraries of subroutines peculiar to its own mission.

3.0 AUTOMATED PROGRAMMING

3.1 Introduction

While it would be nice if it were possible to *completely* automate the production of software, this is certainly not a feasible goal for SDI. It does appear that much of the software development effort could be automated, however. With suitable research and development support, it is highly likely that SDI could develop programming environment systems that could sufficiently automate the programming task.

The current programming paradigm is illustrated in Figure 3. It is very labor intensive with many points into which human errors can be injected. In an automated programming system, the focus is on first obtaining a formal specification, and then employing an automatic system to transform it into a computer program. How can this be accomplished? First, observe that a programmer is an expert in a narrow technical domain. S/he employs a series of well-known, generic methods to solve programming problems. Much of this work can probably be captured using an expert-systems (also called knowledge-based) approach. Figure 4 illustrates one concept for automatic programming. Note that the first part of the system is essentially a computer-aided specification system we proposed

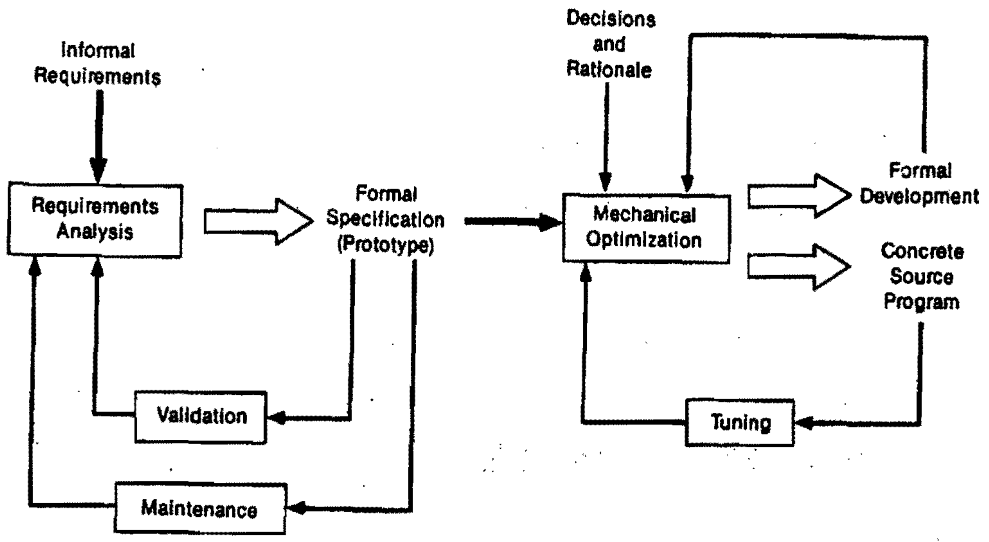


Figure 4. Automation based paradigm. [Balzar, 1985]

earlier. The second half represents the more specialized program synthesis function. We will discuss this later. We will now discuss the structure of a generic computer-aided specification system that could be the general structure of a system to use at several levels of the specification process, as well as the specification part of the automated programming process.

3.2 A Computer-Aided Specification System

There are a number of components that are needed for a computer-aided specification system:

1. A new, high-level, formal language at the specification level (BOOLE).

2. A translator program to convert BOOLE programs into English language reports.

3. A specialized screen editor tailored for BOOLE.

4. A series of testing, debugging, display and housekeeping tools tailored for BOOLE.

5. An Expert System (Knowledge Based System) specialized fo· the particular specification task at hand. For the automatic programming system, this would be an automatic program generator that accepts BOOLE and produces ADA and other conventional code.

6. A personal workstation, designed for specification and program development, using the above software systems, with a high resolution screen, high performance processor, computer network interface, UNIX compatible operating system, ADA compilers, etc.

The logical structure of such a system could be like that shown in Figure 5.

3.2.1 A New Language

Herein we have proposed that new language "BOOLE" be developed. There was general agreement at the workshop that such a language for specification be based on mathmatical logic. This is because the best of the current languages used for program specification are generally based on logic. Examples include the languages Prolog (PROgramming in LOGic), Setl (mostly set theory based, but

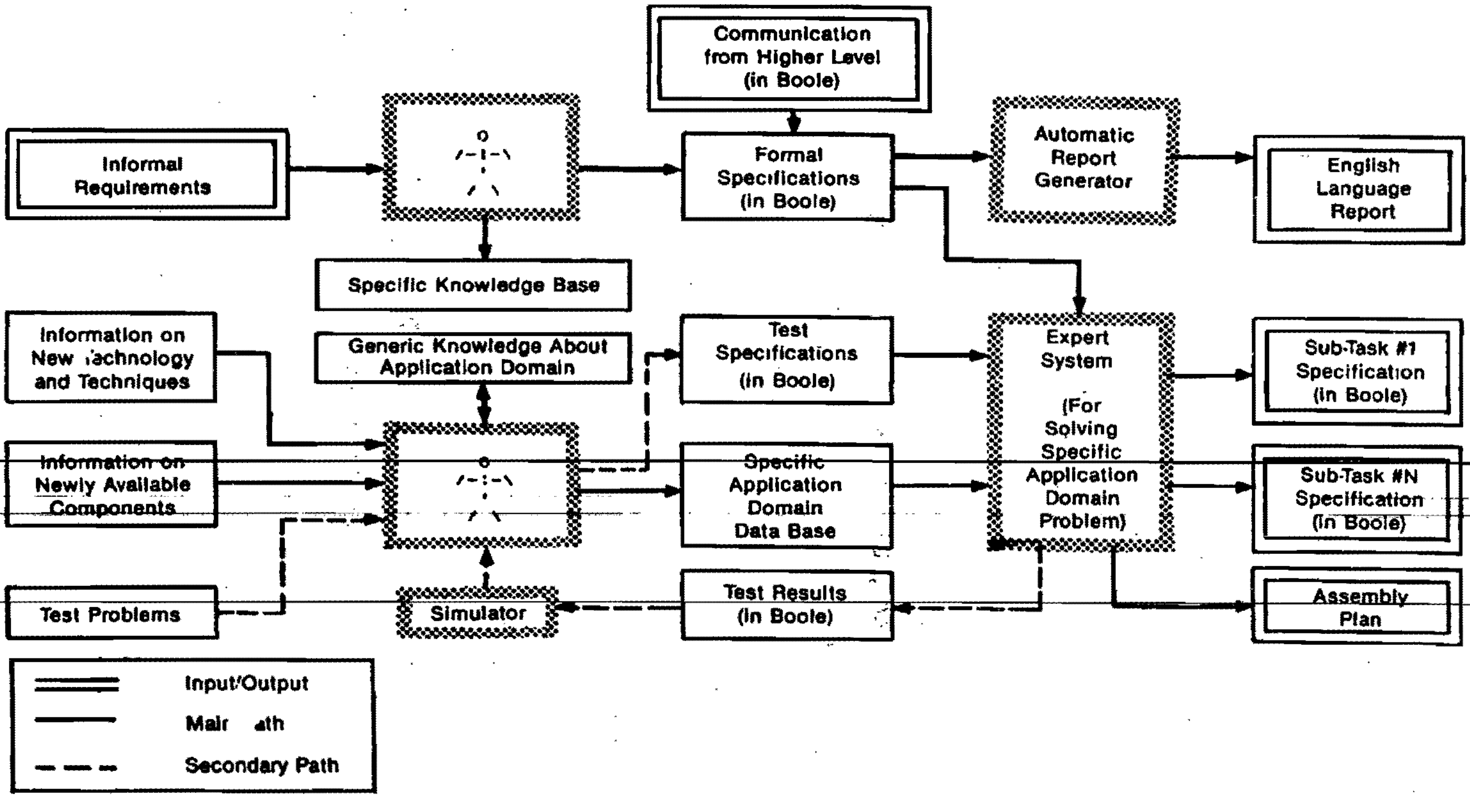

Communication from Higher Level (in Boole)
Informal Requirements
Formal Specifications (in Boole)
Automatic Report Generator
English Language Report
Specific Knowledge Base
Information on New Technology and Techniques
Generic Knowledge About Application Domain
Test Specifications (in Boole)
Expert System (For Solving Specific Application Domain Problem)
Sub-Task #1 Specification (in Boole)
Information on Newly Available Components
Specific Application Domain Data Base
Sub-Task #N Specification (in Boole)
Test Problems
Simulator
Test Results (in Boole)
Assembly Plan
Input/Output
Main Path
Secondary Path

Figure 5. Automatic specification system. [proposed]

entries in the symbol table, (entered so far) and warning of the use of an undefined variable, etc.

3. A 'pretty printer-displayer' that formats the source material into a pleasing form.

3.2.4 Expert System for Specific Application Areas

The ultimate goal of each specification node will be the automatic conversion of a given specification into an assembly plan and a set of component specifications. This is a classic engineering design task requiring much generic engineering knowledge as well as specific application knowledge. If the design process is to be automated, it will require the transfer of this knowledge from engineers to the system. This is now a reasonably well understood process (for narrow technical domains) that is called "Knowledge Engineering" and the result is an "Expert System (or Knowledge Based System)." Most Expert Systems have extensive facilities for English-like dialog. While such facilities may be helpful, the primary input and output, in this case, is the formal language "BOOLE."

At first, in attempting to solve the SDI problem, much of the design work will be done by humans. As the expert system is built, more and more knowledge will be incorporated, until it can accept formal specifications (in its area of expertise) and automatically generate an assembly plan (in BOOLE) and a set of component specifications (in BOOLE).

At the end of the chain, the final specification must be converted into purchase orders for hardware or into conventional low-level computer instructions for software.

Later, as either the higher level specifications change, or new technology, techniques or components become available, the system should be capable of automatically resolving the design problem. Human supervision will still be needed at all levels to review the decisions made by the automatic system.

3.3 Automatic Program Generation

Above we proposed a general approach for many different types of design. Here we will expand these ideas in the context of automatically generating programs.

There are several features that could make up a future automatic programming system:

1. Program development support
2. Algorithm design

3. Automatic program analysis
4. Program optimization

We will discuss each of these in turn.

3.3.1 Program Development Support

There are now several examples of experimental program development support systems (sometimes called "programming environments") that have potential for greatly increasing programmer productivity [27]. Much professional programmer effort generally goes toward documentation and control of versions of program modules. This can be automated to a large degree. Some of these are:

1. Documenting the development history
2. Code installation
3. Distribution changes
4. Maintaining an agenda of pending work
5. Static (data-flow) analysis
6. Bug reporting and disposition
7. Restoration of the system to an earlier state
8. Linking of component development

9. Coordination of source code-editing and object code maintenance
10. Checking on compatibility of updates
11. Preparation of user documentation

There are a number of examples of programming aids that do these tasks individually. For example, under UNIX, the "make" system can support automatic compiling to maintain object code when source code is edited. There are also several systems in either research or development that try to handle all of the above tasks. Three were discussed at the workshop:

1. The Rational ADA programming environment by Rational Inc.
2. The FDS Knowledge based programming system by USC-ISI.
3. The CHI knowledge based programming system by Kestrel Institute, and its successor REFINE by Reasoning Systems Inc.

3.3.2 <u>Algorithm Design</u>

Algorithm design is one of the most intellectually challenging tasks that face a programmer. There are excellent textbooks in the subject such as the three volume series "The Art of Computer

Programming" by Knuth [29]. Much of this knowledge can be captured by an expert system or knowledge based system. One of the most advanced systems that has a powerful algorithm design capability is the CHI system developed at the Kestrel Institute.

3.3.3 Automatic Program Analysis

Automatic program analysis is probably the best understood process of all the topics we are considering. Today's compilers can provide an extensive analysis of programs submitted to them. What we have in mind here is an extension of this, with facilities to identify potential performance problem areas, etc.

3.3.4 Program Optimization

Modern compilers have powerful optimization facilities. It is envisioned that these would be extended modestly for the proposed system.

Programs written at a high level are improved by applying program optimization techniques such as the classical techniques used in optmizing compilers:

1. Global flow analysis
2. Live variable analysis

3. Available expression analysis
4. Use/def links
5. Interprocedural analysis

Additional optimizing techniques proven in research systems for automating programming include:

1. Reduction in strength (finite differencing)
2. Loop fusion
3. Copy optimization
4. Recursion removal
5. Algebraic manipulation
6. Alternate selection of data structures

3.4 Conclusions

It was the concensus of the Workshop that an automated programming paradigm offered a good approach to solving part of the software generation problem. There are now several examples of very powerful programming environments:

1. The Rational ADA programming environment by Rational Inc.
2. The FDS knowledge based programming system by USC-ISI

3. The CHI knowledge based programming system by Kestrel Institute, and it successor REFINE by Reasoning Systems Inc.

Each of these systems is currently being used to produce software, and yet each is continuing to be developed into a yet more powerful tool. While there is no system available today that would be suitable for SDI, it is likely that such a system could be developed from these and other examples of the state of the art.

A SDI automatic programming system would not be completely automatic, but would greatly enhance the productivity of programmers by automating many of the tasks that programmers are forced to do today.

It is recommended that SDIO begin the design process for BOOLE. Also that SDIO should start the development of an computer-aided specification system. Finally SDIO should begin the development of an automatic programming system with BOOLE as the main language, but with facilities to generate ADA and low-level (machine) code.

4.0 CONCLUSIONS AND RECOMMENDATIONS

It is doubtful that all the software that will be needed can be built with only today's technology for a reasonable cost (a fraction of the projected SDI budget). There are a number of directions in current research and development that suggest that, over the next 5-10 years, large gains in programmer productivity could be achieved. A factor of five to ten would be enough to make the task practical. The workable approaches seem to be:

1. Develop a Computer-Aided Specification System (CASS) containing an expert system core that can be adapted to various applications. Multiple proposals should be sought and evaluated.

2. Develop a Very High Level Language (BOOLE) similar to the way ADA was developed. Multiple proposals should be sought and evaluated.

3. Develop an Automated Programming System (APS) in cooperation with the DARPA program. Multiple proposals should be sought and evaluated.

4. Establish Conferences, Summer Studies, "Schools", etc., to train more professionals in advanced programming techniques and especially in the techniques of building automated specification and programming systems.

5. Develop a SDI library (SDIL) of reuseable subprograms, both independently and in cooperation with STARS.

6. Develop Software Standards for SDI (SDISS)

   a. ADA
   b. IEEE Floating Point Arithmetic
   c. Communications Protocols
   d. Data Base Formats & Organization
   e. Graphics Display
   f. System Calls
   g. Network Protocols

7. Begin top-level specification of software soon so that more realistic estimates of the true size of the software problem can be made.

8. Track and cooperate with the Space Station software work.

REFERENCES

1. Thomas Probert, John Soliski, and Audrey Hook, "Estimated Evaluation of Software Requirements for SDI Battle Management System," Internal Memo, Institute for Defense Analyses (1985).

2. Larry Druffel, "Rational's Software Engineering Experience," *Workshop for Automatic Programming* (1-3 July 1985).

3. James C. Fletcher (Study Chairman) and Brockway McMillan (Panel Chairman), *Eliminating the Threat Posed by Nuclear Ballistic Missiles*, Department of Defense (February 1984).

4. Samuel T. Redwine, Jr., Louise Giovane Becker, Ann B. Barmor-Squires, R. J. Martin, Sarah H. Nash, and William E. Riddle, "DoD Related Software Technology Requirements, Practices, and Prospects for the Future," IDA Paper P-1788, Institute for Defense Analyses (June 1984).

5. Karen A. Frankel, "Toward Automating the Software-Development Cycle," *Communications of the ACM* Vol. **Volume** 28 (Number 6) June 1985).

6. Richard M. King, Thomas C. Brown, and Cordell Green (Principal Investigator), "Research on Synthesis of Concurrent Computing Systems," Final Techical Report, Kestrel Institute, Palo Alto, CA (September 1982).

7. Douglas R. Smith, Gordon B. Kotik, and Stephen J. Westfold, "Research on Knowledge-Based Software Environments at Kestrel Institute," *IEEE Transactions of Software Engineering*, Kestrel Institute (August 1985).

8. Stephen Westfold, *Very High-Level Programming of Knowledge Representation Schemes*, Kestrel Institute, Palo Alto, CA (August 1984).

9. Cordell Green (co-chairman), David Luckham (co-chairman), Robert Balzar, Thomas Cheatham, and Charles Rich, *Report on a Knowledge-Based Software Assistant*, Kestrel Institute, Palo Alto, CA (August 1984).

10. Gordon Kotik, *Knowledge-Based Compilation of High-Level Data Types*, Kestrel Institute, Palo Alto, CA (March 1984).

REFERENCES (Cont'd.)

11. Saul Amarel, "Program Synthesis as a Theory Formulation Task-Problem Representations and Solution Methods," CBM-TR-135, Computer Science, Rutgers University, New Brunswick, NJ (June 1983).

12. Jorge Phillips, *Self-Described Programming Environments*, Kestrel Institute, Palo Alto, CA (March 1983).

13. T. Winograd, "Breaking the Complexity Barrier (Again)," *Proceedings of the SIGIRSIGPLAN Conference* (November 1973).

14. S. Knobe Haradhvala and N. Rubin, "Expert Systems for High Quality Code Generation," *Proceedings of the 1st Conference on Artificial Intelligence Applications*, IEEE Computer Society (December 5-7 1984).

15. D. Barstow, *Knowledge-Based Program Construction*, 1979.

16. C. C. Green, R. P. Gabriel, E. Kant, B. Kedrierski, B. P. McCune, J. Phillips, S. Tappel, and S. J. Westfold, "Results in Knowledge-Based Program Synthesis," *Proceedings of the Sixth International Joint Conference on Artificial Intelligence* (August 20-23 1979).

17. C. C. Green and S. J. Westfold, "Knowledge-Based Programming Self-Applied," *Machine Intelligence*, John Wiley & Sons (1982).

18. C. C. Green, R. Balzer, and T. Cheatham, "Software Technology in the 1990's Using a New Paradigm," *IEEE Computer* (November 1983).

19. E. Kant, *Efficiency Considerations in Program Synthesis: A Knowledge-Based Approach, Ph.D. Dissertation, Computer Science Dept.*, Stanford University (1970).

20. Z. Manna and R. J. Waldinger, "A Deductive Approach to Program Synthesis," *ACM TOPLAS* Vol. **Vol. 2, no. 1** (1980).

21. R. Paige and S. Koenig, "Finite Differencing of Computable Expressions," *TOPLAS 4, 3* (July 1982).

REFERENCES (Concl'd.)

22. H. Partsch and R. Steinbruggen, "Program Transformation Systems," *Computing Surveys* Vol. **Vol 15, no. 3** (1983).

23. W. L. Scherlis and D. S. Scott, "First Steps Towards Inferential Programming," *Proc. IFIP-83* (1983).

24. J. Schwartz, "On Programming: An Interim Report of the SETL Project," *Technical Report*, New York University (1975).

25. J. Schwartz, "Automatic Data Structure Choice in a Language of Very High Level," *CACM* Vol. **vol. 18** (1975).

26. S. Tappel, "Some Algorithm Design Methods," *Proceedings of AAAI-80* (1980).

27. P. Zave and W. Schell, "The PAISLey Software Tools: An Environment for Executable Specification," *Proc. Workshop on Software Engineering Environments for Programming-in-the-Large* (1985).

28. F. Brooks, Jr., "The Mythical Man-Month; Essays on Software Engineering," *Addison-Wesley, Reading, Mass.* (1975).

29. Donald E. Knuth, "The Art of Computer Programming," **Vol.'s 1, 2 and 3,** *Addison-Wesley, Reading, Mass.* (1975).

DISTRIBUTION LIST

Dr. Henry Abarbanel
University of California
San Diego, CA

LTGEN James Abrahamson
Director
Strategic Defense Initiative
Organization
Office of the Secretary of
Defense
The Pentagon, Rm. 3E1034
Washington, DC 20301

Mr. Duane Adams
2325 N. Richmond Street
Arlington, VA 22207

Mr. Saul Amarel
Director
DARPA/IPTO
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Marv Atkins
Deputy Director, Science & Tech.
Defense Nuclear Agency
Washington, D.C. 20305

MAJ David Audley
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

LTCOL Joseph Bailey
Executive Office of the President
Office of S&T Policy
Room 5026
New Executive Office Building
Washington, DC 20500

National Security Agency [2]
Attn R5: (b)(3):50 USC §402 Note
Ft. George G. Meade, MD 20755

Dr. Peter Banks
Stanford University
Stanford, CA

Dr. Richard Bleach
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

Mr. Walter L. Brothers
Senior Weapons System Analyst
ANSER
1215 Jefferson Davis Highway
Suite 800
Crystal Gateway 3
Arlington, VA

Mr. Richard Brown
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

Dr. Solomon J. Buchsbaum
Executive Vice President
Bell Laboratories
Crawfords Corner Road
Holmdel, NJ 07733-1988

Dr. Curtis Callan
Department of Physics
Princeton University
Box 708
Princeton, NJ 08544

Dr. Ken Case
The Rockefeller Univ.
New York, NY 10021

DISTRIBUTION LIST (Cont'd.)

Dr. Bijoy Chatterjee
Advanced Technology Directorate
ESL
495 Java Drive
Sunnyvale, CA 94088-3510

Dr. Danny Cohen
USCISI
4676 Admiralty Way
Marina del Ray, CA 90292

Dr. Eugene E. Covert
58 Fresh Pond Place
Cambridge, MA 02138

Ms. Anita L. Crossland
Editor
ANSER
1215 Jefferson Davis Highway
Suite 800, Crystal Gateway 3
Arlington, VA 22202

Mr. John Cuadrado
Institute for Defense Analysis
1801 N. Beauregard St.
Alexandria, VA 22311

Mr. James W. Cusack
USAF/RADC
RADC/OCSP
Griffin AFB, NY 13440

Mr. John Darrah
Sr. Scientist and Technical
Advisor
HQ Space Cmd/XPN
Peterson AFB, CO 80914

Defense Technical Information [2]
Center
Cameron Station
Alexandria, VA 22314

Dr. Alvin M. Despain
University of California
Berkeley, CA

CAPT Craig E. Dorman
Department of the Navy, OP-095T
The Pentagon, Room 5D576
Washington, D.C. 20350

Mr. Larry Druffel
1501 Salado Drive
Mtn. View, CA 94043

CDR Timothy Dugan [2]
NFOIO Detachment, Suitland
4301 Suitland Road
Washington, D.C. 20390

Dr. Robert C. Duncan
Director
DARPA
1400 Wilson Boulevard
Arlington, VA 22209

Dr. Doug Eardley
University of California
Santa Barbara, CA

Dr. David D. Elliott
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Mr. James C. Elms
112 Kings Place
Newport Beach, CA 92663

Mr. John Entzminger
Director
DARPA/TTO
1400 Wilson Blvd.
Arlington, VA 22209

DISTRIBUTION LIST (Cont'd.)

Mr. Robert E. Everett
President
The MITRE Corporation
P. O. Box 208
Bedford, MA 01730

Mr. Daniel J. Fink
8016 Matterhorn Court
Potomac, MD 20854

Dr. J. Richard Fisher
Assistant BMD Program Manager
U.S. Army
Strategic Defense Command
P. O. Box 15280
Arlington, VA 22215-0150

Dr. James C. Fletcher
James Fletcher Associates
7925 Jones Branch Drive
McLean, VA 22101

(b)(3):50 USC §403(g) Section 6 [2]
P.O. Box 1925
Washington, D.C. 20505

Director [2]
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note, A05

Dr. John S. Foster, Jr.
TRW Inc.
1900 Richmond Road
Cleveland, OH 44124

Mr. Bert Fowler
Senior Vice President
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

Mr. Geoffrey Frank
RTI
P.O. Box 12194
Research Triangle Park
North Carolina 27709

Dr. Edward Frieman
Executive Vice President
SAI Inc.
1200 Prospect Street
P. O. Box 2351
La Jolla, CA 92038

Mr. John Gardner
Director, Systems
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

MAJ Glenn S. Geary
Strategic Defense Initiative
Organization/SLKT
Washington, DC 20301-7100

Dr. Larry Gershwin
NIO for Strategic Programs
P.O. Box 1925
Washington, D.C. 20505

Mr. Allen Goldberg
Kestral
1801 Page Mill Road
Palo Alto, CA 94304

Dr. S. William Gouse, W300
Vice President and General
Manager
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102

DISTRIBUTION LIST (Cont'd.)

COL Jim Graham
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

Dr. Dave Hammer
109 Orchard Place
Ithaca, NY 14850

Dr. William Happer
559 Riverside Drive
Princeton, NJ 08540

Dr. Edward Harper [2]
SSBN, Security Director
OP-021T
The Pentagon, Room 4D534
Washington, D.C. 20350

CAPT David Hart
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

Mr. David Heebner
SAI Inc.
P. O. Box 1303
McLean, VA 22101

COL George Hess
Director
Survivability, Lethality
and Key Technologies
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

Dr. Donald A. Hicks [2]
Under Secretary of Defense (R&E)
Designee
Office of the Secretary of
Defense
The Pentagon, Room 3E1006
Washington, D.C. 20301

Mr. R. Evan Hineman
Deputy Director for Science
& Technology
P.O. Box 1925
Washington, D.C 20505

Dr. James Ionson
Director
Innovative Science and
Technology Office
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

Mr. Theodore Jarvis
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

Mr. Richard Jullig
Kestrel
1801 Page Mill Road
Palo Alto, CA 94304

Mr. Ed Key
Vice President
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

LTCOL Gene Kluter
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

Dr. Ostap S. Kosovych
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

MAJ GEN Donald L. Lamberson
Assistant Deputy Chief of Staff
(RD&A) HQ USAF/RD, Rm. 4E334
Washington, D.C. 20330

DISTRIBUTION LIST (Cont'd.)

Dr. Richard Lau
Scientific Officer
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217-5000

(b)(3):10 USC §424
Defense Intelligence Agency
The Pentagon
Washington, DC 20301-6111

MAJ Roger Lenard
SDIO/KE
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

COL Alfonso Lenhardt
Exeuctive Officer
Strategic Defense Initiative
Organization
Washington, DC 20301-7100

Ms. Nancy Leveson
Univ. of California
ICS Department
Irving, CA 92717

The MITRE Corporation [3]
1820 Dolley Madison Blvd.
McLean, VA 22102
ATTN: JASON Library, W002

MAJ Rodney Liesveld
Special Assistant
Strategic Defense Initiative
Organization/DD
Washington, DC 20301-7100

Dr. Richard J. Lipton
Professor
Dept. of Computer Science
Princeton University
Princeton, NJ 08540

Dr. Gordon MacDonald
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102

Mr. William G. MacLaren
MAJGEN USAF (Ret.)
Vice President
V. Gerber Associates
Arlington, VA 22210

Mr. Charles Mandelbaum
Mail Stop ER-32/G-226 GTN
U.S. Department of Energy
Washington, D.C. 20545

Mr. Robert Manners
Office of Research and
Development
P.O. Box 1925
Washington, DC 20505

Dr. Hans Mark
Chancellor, University of Texas
601 Colorado Street
Austin, TX 78712

Dr. Louis Marquet
Director
Directed Energy Weapons
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

Dr. David Martin
Director
External Affairs
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

Mr. John McMahon
Dep. Dir. Cen. Intelligence
P.O. Box 1925
Washington, D.C. 20505

DISTRIBUTION LIST (Cont'd.)

LTCOL Robert McMains
Executive Office of the President
Office of S&T Policy
Room 5026
New Executive Office Building
Washington, DC 20500

Dr. John D. McTague
Director (Acting)
Office of Science & Tech. Policy
Old Executive Office Building
17th & Pennsylvania, N.W.
Washington, D.C. 20500

Dr. Allan T. Mense
Special Assistant to Chief
Scientist
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

Mr. Mewson
HQ SAC/NRI
Offutt AFB
Nebraska 68113-5001

Dr. Harlan D. Mills
IBM Fellow
IBM Federal Systems Division
6600 Rockledge Drive
Bethesda, MD 20817

Dr. David Mizell
Scientific Officer
Office of Naval Research
1030 East Green St.
Pasadena, CA 91106-2485

Dr. Richard Montgomery
1398 Avenida de Cortez
Pacific Palisades, CA 90272

Dr. Marvin Moss [2]
Technical Director
Office of Naval Research
800 N. Quincy Street
Arlington, VA 22217

(b)(3):50 USC §403(g) Section 6
P.O. Box 1925
Washington, D.C. 20505

Director
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note
DDR-FANX 3

Dr. Al Newton
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

Prof. William A. Nierenberg
Scripps Institution of
Oceanography
University of California, S.D.
La Jolla, CA 92093

Dr. Robert Norwood [2]
Office of the Assistant Secretary
of the Army
(Research Development
& Acquisition)
The Pentagon
Room 2E673
Washington, D.C. 20310-0103

BGEN Malcolm O'Neill
Director
Kinetic Energy Office
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

DISTRIBUTION LIST (Cont'd.)

CDR James Offutt
Strategic Defense Initiative
Organization/SY
Washington, DC 20301-7100

Dr. Nicholas Osifchin
Director, C^3I
Defensive Systems Center
AT&T Bell Laboratories
25 Lindsley Drive
Room 2C214
Morristown, NJ 07960

COL Robert Parker
Director
Resource Management
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

Mr. Albert J. Perrella
Institute for Defense Analyses
1801 N. Beauregard Street
Alexandria, VA 22311

Dr. Allen M. Peterson
Stanford University
Stanford, CA

Mr. Thomas Probert
Institute for Defense Analysis
1801 N. Beauregard Street
Alexandria, VA 22311

Dr. Simon Ramo
One Space Park
Mail Stop E2-11000B
Redondo Beach, CA 90278-1001

The MITRE Corporation
Records Resources
1820 Dolley Madison Blvd.
Mail Stop W971
McLean, VA 22102

Mr. Richard Reynolds
Director
DARPA/DSO
1400 Wilson Blvd.
Arlington, VA 22209

Mr. William Riddle
P.O. Box 3521
Boulder, CO 80303

Mr. Alan J. Roberts
Vice President & General Manager
Washington C^3I Operations
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, VA 22102

Dr. Marshall Rosenbluth
University of Texas at Austin
Institute for Fusion Studies,
RLM 11.218
Austin, TX 78712

Dr. Oscar Rothaus
Cornel University
New York

COL Garry Schnelzer
Director, Sensors
Strategic Defense Initiative
Organization/OSD
Washington, DC 20301-7100

GEN Bernard A. Schriever
Schriever & McKee, Inc.
1899 L Street, N.W., Room 405
Washington, DC 20036

Dr. Charles L. Seitz
Professor of Computer Science
California Institute of
Technology
Pasadena, CA 91125

DISTRIBUTION LIST (Cont'd.)

Dr. Frederick Seitz
The Rockefeller University
1230 York Avenue
New York, NY 10021-6399

(b)(3):10 USC §424
Defense Intelligence Agency
The Pentagon
Washington, DC 20301-6111

Dr. Phil Selwyn [2]
Technical Director
Office of Naval Technology
800 N. Quincy Street
Arlington, VA 22217

Dr. Eugene Sevin [2]
Defense Nuclear Agency
6801 Telegraph Road
Room 244
Alexandria, VA 22310

Mr. Shen Shey
Special Assistant for
Directed Energy
DARPA
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Sidney Singer
Executive Office of the President
Office of S&T Policy
Room 5026
New Executive Office Building
Washington, DC 20500

Dr. Alan J. Smith
Associate Professor
Computer Science Division
University of California
Berkeley, CA 94701

Dr. Joel A. Snow [2]
Senior Technical Advisor
Office of Energy Research
U.S. DOE, M.S. E084
Washington, D.C. 20585

Dr. James W. Somers
Defense Nuclear Agency
6801 Telegraph Road
Alexandria, VA 22310-3398

Dr. Robert L. Sproull
201 Bausch and Lomb Hall
University of Rochester
Rochester, NY 14627

Dr. William R. Stout
Senior Mathematician
ANSER
1215 Jefferson Davis Highway
Suite 800
Crystal Gateway 3
Arlington, VA 22202

COMO William Studeman
Director of Naval Intelligence
Office of Naval Intelligence
Navy Department
Washington, D.C. 20350

Mr. Richard Sybert
Special Assist. to the Secretary
Executive Secretariat/OSD
The Pentagon, Room 3E880
Washington, DC 20301-7100

Mr. Alexander J. Tachmindji
Senior Vice President & General
Manager
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

DISTRIBUTION LIST (Concl'd.)

Dr. Edward Teller
Lawrence Livermore National Lab
P. O. Box 808
Livermore, CA 94550

Dr. Vigdor Teplitz
ACDA
320 21st Street, N.W.
Room 4484
Washington, D.C. 20451

Mr. Tony Tether
Director
DARPA/STO
1400 Wilson Blvd.
Arlington, VA 22209

Dr. Al Trivelpiece
Director, Office of Energy
Research, U.S. DOE
M.S. 6E084
Washington, D.C. 20585

Dr. John Vesecky
Center for Radar Astronomy
Electrical Engineering Dept.
Stanford University
Stanford, CA 94305

Mr. Richard Waldinger
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Dr. Albert D. Wheelon
Sr. VP and Group President
Space and Communications Group
Hughes Aircraft Company
P. O. Box 92919
Los Angeles, CA 90009

LTCOL Simon Peter Worden
Strategic Defense Initiative
Organization/OSD
Washington, D.C. 20301-7100

Dr. Gerold Yonas [2]
Strategic Defense Initiative
Organization
Office of the Secretary of
Defense
The Pentagon
Washington, DC 20301-7100

Mr. Leo Young
OUSDRE (R&AT)
The Pentagon, Room 3D1067
Washington, D.C. 20301

Mr. Charles A. Zraket
Executive Vice President
The MITRE Corporation
P.O. Box 208
Bedford, MA 01730

APPENDIX A

AGENDA

1 July 1985

Time	Speaker	Topic
0900	Despain	Introduction
0930	Offutt	SDI Systems Overview
1030	Coffee Break	
1100	Offutt	Battle Management C^3 Program
1200	Lunch Break	
1330	Adams	Second Look at the Software Requirements from the Fletcher Report
1430	Coffee Break	
1500	Probert	The Software Problem
1600	Druffel	Review of the STARS Program

AGENDA

2 July 1985

Time	Speaker	Topic
0900	Riddle	Support for Development of Large Software Systems
0945	Druffel	Strategies for Large Software Systems
1030	Coffee Break	
1045	Leveson	Software Reliability and Safety
1215	Lunch Break	
1330	Jarvis	Considerations in Battle Management/Command and Control for Strategic Defensa Systems
1415	Cuadrado	ADAS: An Architecture and Design Assessment System
1500	Coffee Break	
1515	Frank	The Future of ADAS
1600	Waldinger	Program Synthesis: The Deductive Approach

AGENDA

3 July 1985

Time	Speaker	Topic
0900	Amarel	AI and Program Synthesis
0945	Despain	Transforming Weak Program into Strong Programs
1030	Coffee Break	
1045	Balzar	FSD: A Specification Based Computing Environment
1130	Jullig	Current Research into Knowledge-Based Software Tools
1215	Lunch Break	
1330	Goldberg	Outlook on Automated Assistance for Software Development and Maintenance
1500	Coffee Break	
1515	Discussion	Summary of Critical Issues

ADB099221

Report on the Workshop for Automated Software Programming

MITRE CORP MCLEAN VA

FEB 1986

Encl 5

[This page is intentionally left blank.]

ADB149872

doc. 6

Space Power System Study

MITRE CORP MCLEAN VA

MAY 1984

84- 0300

AD-B149 872

Space Power System Study

MITRE

90 11 30 100

Space Power System Study

H. W. Lewis
A. M. Peterson

Accesion For	
NTIS CRA&I	☐
DTIC TAB	✓
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
7	5?/25

May 1984

JSR-82-801

JASON
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, Virginia 22102

TABLE OF CONTENTS

		Page
	LIST OF ILLUSTRATIONS	iv
	LIST OF TABLES	iv
1.0	INTRODUCTION	1
2.0	PHOTOVOLTAIC POWER SYSTEMS	3
2.1	Introduction	3
2.2	Projected Performance Capabilities	4
2.3	Hardening of Solar Power Systems	8
2.4	Batteries	9
3.0	NUCLEAR SPACE POWER	11
3.1	Introduction	11
3.2	Types of Reactor	13
3.3	Types of Conversion	14
3.4	Heat Rejection	17
3.5	Power Conditioning	18
3.6	Shielding	18
3.7	Safety	19
4.0	SUMMARY	21
	BIBLIOGRAPHY	25
	DISTRIBUTION LIST	D-1

LIST OF ILLUSTRATIONS

Page

Figure 1 Baseline systems deployed configurations............. 5

LIST OF TABLES

Table 1 Pertinent characteristics of NiCd and NiH_2 storage systems..................................... 10

1.0 INTRODUCTION

Space systems have become an important part of our military force structure. U.S. military space systems include communications, navigation, weather observation, mapping and geodesy, attack warning and threat surveillance. In general, there is a trend towards larger spacecraft which stresses survivability, autonomy, and extended life.

Solar photovoltaic power supplies together with battery (nickel-cadmium cells) have in the past provided outstanding service for systems requiring average powers up to about 10 kw. During our study, presentations by industry and government laboratories indicate that improved solar-cell technology (use of GaS cells) and new batteries (nickel hydrogen) should result in mission effective average power levels approaching 25 kw.

However, there appear to be a number of important military applications which will require power levels in the tens to hundreds of kilowatts. Such missions include the following:

- sSpace based radars to track ships, aircraft, missiles, and satellites;
- Infrared trackers for missiles, aircraft, and satellites;
- Space based communication and radar jammers; and
- Jam resistant communications satellites. (RH)

This is a very incomplete list but serves to identify a class of applications which we believe will require long duration power systems beyond the capabilities of even improved photovoltaic-battery systems.

Nuclear reactor power plants appear to provide a particularly attractive and cost effective solution for the class of applications just discussed.

Another class of applications which is frequently discussed but poorly defined at present includes weapons of the high energy laser and particle beam types. Except for housekeeping functions, nuclear reactor supplies do not appear to be well matched to these applications which appear to require short to medium duration power at levels in the tens to hundreds of megawatts. Fuel cells or turbine driven generators using stored fuels appear more suitable for these applications, though not well developed at present.

2.0 PHOTOVOLTAIC POWER SYSTEMS

2.1 Introduction

Solar cell arrays have supplied nearly all of the primary power for space craft flown to date. Where continuous power is required through eclipse periods, battery storage is required; systems to date have utilized nickel cadmium (NiCd) cells.

To date only silicon solar cells have been used in operational spacecraft power systems. However, Gallium Arsenide cells would appear to be the preferred cell type for future long duration, high power space applications. Ga-As solar cells not only yield somewhat higher basic conversion efficiencies but more importantly will have greater immunity to natural and man made radiation.

For silicon solar cells the "end of life" (EOL) output power density which is achievable appears to be only about 0.3 to 0.6 of the basic solar cell efficiency. This reduced performance occurs because of "initial losses" as a result of array assembly, operating temperature, packing factor, etc; and "mission losses" which result from natural space radiation, thermal cycling, and array orientation. The net result is that so-called 15% silicon cells yield an EOL output power density which runs between 60 w/m^2 and 120 w/m^2. Thus an array operating at 60 w/m^2 would require

≈ 1700 m^2 to give 100 kw EOL power output and ≈ 420 m^2 area to provide 25 kw EOL power. A pictorial comparison between possible solar arrays for 25 and 100 kw and the proposed SP100 nuclear system is given in Fig. 1. As can be seen, a 100 kw EOL solar array would be about half the area of a football field.

2.2 Projected Performance Capabilities

The Space Shuttle will provide the space transportation for lifting payloads from Earth to Low Earth Orbit (LEO) and when an orbit transfer vehicle (OTV) is developed, payloads will be transferred from LEO to Geosynchronous Orbit (GEO). Considering the Shuttle's limited payload capability, photovoltaic power systems for GEO-bound payloads must have high specific power (w/kg). In order to maximize the life of a photovoltaic system which is subjected to the space radiation environment, there must be radiation damage control, especially for exposure to electrons and protons. In addition to providing economical power for the practical use of near-Earth space, there is a need for high power levels at low cost.

The need for high specific power is driven not only by the weight constraints of Shuttle, but by OTV capability of transferring an assembled spacecraft or platform from LEO to other orbits for operational use. The beginning-of-life specific power of solar arrays has increased from the value of approximately 2 w/kg for

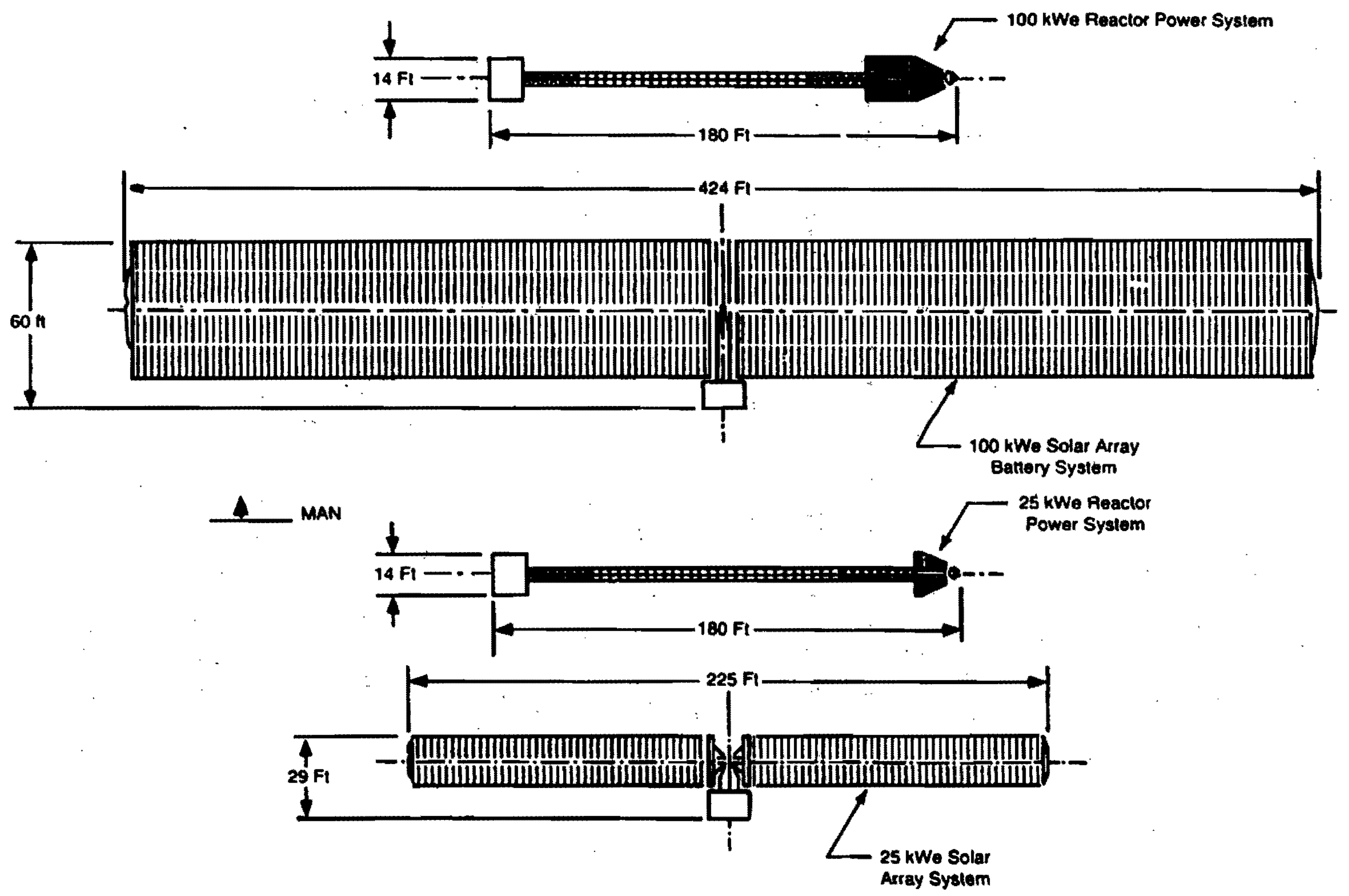

100 kWe Reactor Power System
14 Ft
180 Ft
424 Ft
60 ft
100 kWe Solar Array Battery System
MAN
25 kWe Reactor Power System
14 Ft
180 Ft
225 Ft
29 Ft
25 kWe Solar Array System

Figure 1. Baseline systems deployed configurations

Vanguard to the 66 w/kg level of the NASA solar electric power unit (SEP).

Barthelmy has written that present LEO solar orbit power systems are relatively heavy J(2-3 w/lb) due primarily to the energy storage (battery) weight. Typically 40% of the overall power system weight is attributed to the battery while the remaining weight percentages are divided approximately equally between the solar array and the power distribution and control subsystems. For geosynchronous orbit applications, the battery, solar array, and power distribution and control subsystem weight percentages are approximately equal. Solar power system energy density for present geosynchronous applications are in the range of 6-7 w/lb.

To meet the specific power requirements, both planar and concentrator configurations are being considered. The planar configuration will require advances in thin cells, encapsulants, substrates, coatings, and applicable production process in addition to major advances in array structures. Similar tasks are required for a concentrator configuration with the emphasis needed on high efficiency cells, elevated temperature operation, and more complex system considerations.

Another major need is to increase the operating lifetime of solar cells and arrays for use in space. The limited operating lifetime is principally due to the degraded performance of solar cells when exposed to charged particle radiation, although degraded performance has been attributed to a lesser extent to ultra-violet exposure and high temperature operation. Space applications in GEO and transfer of spacecraft from LEO to GEO through the Van Allen radiation belts require arrays which must withstand ionizing radiation. Mid-altitude systems must be designed for maximum EOL performance after exposure to 5 x $10^{16}/cm^2$ 1 mev electron equivalent radiation dose. The lifetime of space solar cells has been increased significantly since the original introduction of the P/N silicon solar cell. Improvements have resulted from the use of higher resistivity, shallower junction, and N/P silicon cells. Additional improvements are anticipated by the use of vertical junction cells, 50 µm silicon cells and Gallium Arsenide (GaAs) cells. Low degradation can be obtained when the cells are protected with suitable covers; however, weight is added to the structure. The degraded performance is due to radiation-induced defects. What is needed is radiation damage control; that is, the ability to minimize or eliminate both the defect formation and the effects of the damage.

It has been shown that the radiation damaged solar cells may recover to nearly initial performance levels by means of either thermal or laser annealing. A thorough understanding of the fundamental mechanisms is required to control radiation damage, not only in existing solar cells, but those yet to be developed, such as the multibandgap cells. It should be noted that the practical implementation of an array annealing has not been fully demonstrated.

However, the possibility has been suggested that GaAs cells operating in a concentrator configuration at temperatures near 200^{o}C might be continuously "self-annealing" and thus relatively immune from particle radiation damage. Silicon cell efficiency degrades rapidly as temperature increases and thus does not appear suitable for such applications.

2.3 Hardening of Solar Power Systems

The current solar power systems are designed to meet JCS Nuclear Survivability Design Criteria with relatively modest (~ 10%) weight and cost hardening penalties. Hardening penalties to envisioned laser threats appear to be much higher (> 50%).

Solar array hardening developments are currently focused on laser hardening, with maintenance of the current capability against nuclear radiation threats. The laser hardening activity includes (1) increasing the temperature capability of all array components by use of welded interconnects, integral coverglass, and high temperature adhesive, and (2) minimizing energy absorption using reflection and filtering or avoidance. Most efforts in the past dealt with silicon cell technology, but as the GaAs cell technology matures, it must be hardened also. Therefore, advanced cell hardening, integral covers, and high temperature contact areas address both GaAs and silicon. DOD is also evaluating concentration concepts as a means of increasing system hardness. Most schemes to date involve a reduction in efficiency and an increase in weight to achieve hardness. If demonstrated, the self-annealing performance of GaAs cells together with concentration might become a particularly attractive option for hardening.

2.4 Batteries

Space power systems based on solar cell technology require batteries to store energy for use during periods of spacecraft eclipse or when higher than normal peak loads are needed. To date, this requirement has been met by the use of nickel-cadmium (NiCd) storage cells. It, however, appears the preferred storage system by the latter part of the decade will make use of nickel-hydrogen (NiH_2) cells. It is anticipated NiH_2 battery systems will be lighter for a given power output and will have a longer service lifetime. Other storage cells involving lithium, sodium, etc., show promise of even better performance than NiH_2 cells, but their operational availabiltiy and cost cannot be predicted with confidence at the moment. It should be noted that although the weight per unit energy stored for NiH_2 cells is less than for NiCd cells, the volume for a given energy storage is somewhat greater.

A number of pertinent characteristics of NiCd and NiH_2 storage systems are listed in the table below.

TABLE 1

Pertinent Characteristics of
NiCd and NiH_2 Storage Systems

	NiCd	NiH_2
Watt-Hour/Kg	24 WH/Kg	40 WH/Kg
Watt-Hour/m^3	61 kwh/m^3	70 kw hr/m^3
Watt/Kg	10 W/Kg	20 W/Kg
Cost $/kw	≈ $500/Watt*	≈ $500/Watt*

*based on battery system costs for systems of a few kw

The numbers in this table suggest that a 100 kw NiH_2 storage battery system would cost ≈ $50 million dollars, would weigh ≈ 5000 kg and would be several cubic meters in volume. It should be noted that a production facility does not appear to be available at present for producing such a system.

3.0 NUCLEAR SPACE POWER

3.1 Introduction

As mentioned earlier, solar photovoltaic systems (which are already at a higher power density than RTGs) deliver power for approximately 50 w/kg, not including the necessary batteries. Depending upon the specific system design, the addition of the batteries brings the figure of merit down to about 10-25 w/kg, so that a 100 kwe mission, powered by solar photovoltaics, will require power supply weight of something of the order of 5000 kg, which is clearly prohibitive. In addition, such a mission would require more than 1000 m^2 of solar panels, which would have to be packaged in the STS shuttle and deployed in space. Most people with whom we have discussed this believe that the upper practical limit for solar photovoltaics is in the region of 25-40 kw.

By contrast, a reactor system of <u>current</u> design is expected to supply electricity at about 25-50 w/kg, without the battery requirement, and the figure of merit improves as the power level increases. Thus, the balance of preference shifts at about 25 kw mission requirement, and continues to favor nuclear power at still higher levels. We will discuss some special features of the nuclear option below.

Among the missions with requirements in this range are space-based radars and space-based laser communication systems, although each can be operated at somewhat (but not dramatically) lower power levels, by trading other features. There are, of course, more conjectural missions requiring even higher power, like directed-energy weapons and manned military bases in space, but we will ignore these for the moment. Established missions requiring between 10 and 20 kw already exist, so the eventual need for higher power is not illusory.

Before the U.S. program was stopped in the early 1970s, we had flown only one reactor, SNAP 10-A, which operated at a power level of 560 watts, at an efficiency of 1/4%, as a technical demonstration. The Soviets have flown many reactors at relatively low power levels (the most notorious being COSMOS 954), presumably in a quest for the lower drag at low orbital altitudes that is another feature of a concentrated power supply. COSMOS 954 illustrated the existence of safety issues we will discuss below.

Before getting down to cases, it is well to remind ourselves of some basic numbers that are useful in orientation. The central one is, of course, the energy concentration of fissionable material. A kilogram of U^{235} contains about 2.6 megawatt-years of fission energy, so that, if one thinks of a critical mass of the order of ten times that, and thinks of possibly recovering 10% of that energy, one is thinking of a few megawatt-years of electricity stored in a reacto' core. Although other system features (like heat rejection, to be discussed below) become more challenging at higher power levels, the fact is that nuclear power is well-suited to power requirements in the region between 100 kwe and 1 Mwe. Lower powers are wasteful of energy available.

In the following sections we will go briefly through some of the considerations and options in the selection of a nuclear electric power system for a spacecraft.

3.2 Types of Reactor

Among the types that have been discussed are the following:

- gas-cooled reactors
- reactors cooled by pumped liquid metals
- reactors cooled by heat pipes
- reactors cooled by other pumped liquids
- direct-conversion-cooled reactors

Each of these reactors has its advocates and its special features. For example, some people are concerned about the durability of any kind of rotating machinery (pumps) in space, although existing systems using moving parts seem to function reliably. Micrometeorite puncture of coolant lines is another issue affecting both gas- and liquid-cooled reactors. As will be seen below, we think there is time to resolve these questions.

3.3 Types of Conversion

Among the means of converting fission energy to electricity that have been investigated are the following:

- thermoelectric conversion
- thermionic conversion (direct and indirect)
- thermophotovoltaic conversion
- magnetohydrodynamic conversion
- dynamic conversion
  - Rankine cycle
  - Stirling cycle
  - Brayton cycle

With the exception of magnetohydrodynamic conversion, all of these are possible candidate conversion technologies to be mated to a reactor heat source. The first two have been most extensively studied, and SNAP 10-A was thermoelectric, with the principal concern about the various dynamic technologies having to do with moving fluids and moving parts. The Soviets appear to have used the first two. We discuss the first three in slightly greater detail.

Thermoelectric conversion is the most developed form, has been widely used (even in space), but suffers from some disadvantages. It is relatively inefficient, typically a few percent, so that the radiated power requirements are high for a given amount of electricity. It operates at a relatively low temperature, which compounds the radiation problem, though the search for high-temperature thermoelectric materials continues. It requires less extrapolation of the current state-of-the-art than the other options, and was the choice for the SP-100, to be discussed below.

Thermionic conversion (direct conversion through a plasma diode) had a burst of development in the sixties, but has not prospered since. It is in many ways the most promising option because the efficiency of conversion is high, typically 15% or more, and the heat rejection temperature at the anode is high, which relieves the radiation problem even more. Thus, it is particularly well suited to direct conversion systems with in-pile thermocouples, and such systems were in fact tested at Los Alamos in the early sixties. On the other hand, the cell spacing has to be small to achieve high efficiency, and the geometry has to be preserved under difficult environmental conditions, so that there remain unresolved materials problems that involve technical risk. Even so, this is a promising technology.

Thermophotovoltaic conversion was a sleeper--we had never heard of it before--and it is by no means clear that it is appropriate to the space nuclear application. Nonetheless, it is so appealing that we give it some space here.

Recall that photovoltaic conversion in a normal silicon solar cell proceeds by the absorption of a photon in the p-n junction region of a cell, with the electron-hole pair subsequently separated by the intra-junction field. The optimum photon energy required to create the pair is, of course, just equal to the energy gap between the valence and conduction bands in silicon, approximately 1.1 volts. Any photon energy over this contributes to kinetic energy of the electron and hole, and is subsequently lost into heat. Thus, a major source of inefficiency in a normal silicon solar cell derives from the fact that the solar spectrum is of Planck form, with a peak in the green at about 2 volts, so that fully half the energy of each photon is wasted in this way. (Of course, there are other inefficiencies in the cell.) There has been a great deal of research effort expended on the search for materials with larger band gaps, to more closely match the solar spectrum, but the beauty and simplicity of the thermophotovoltaic idea is that of matching the solar spectrum to the cell, rather than the other way around. This is possible because the second law of thermodynamics permits one to cool radiation with 100% efficiency. Efficiencies greater than 30% have in fact been achieved in the laboratory for photoelectric conversion with silicon cells. Unfortunately, this technique, though it would work just as well in space, remains limited to silicon cells, which in turn lose efficiency dramatically at temperatures over about 100°C, so the heat rejection problem is still severe until new high-temperature photovoltaic cells are discovered. Until then, this is only (and it is) a very promising earth-bound technology. _If_ higher temperature materials can be found, the potential for driving the cells directly with a reactor, rather than with a degraded solar spectrum, is rea..

3.4 Heat Rejection

There is no alternative to heat rejection by radiation in space, except for very short missions in which one is willing to squander working fluid, and such missions are of no apparent interest. For high power systems, the weight of the radiator begins to dominate the system weight, unless advantage is taken of the fact that thermal radiation from a black body increases as T^4.

For orientation, consider the following numbers. A black sphere in sunlight in space, at a distance from the sun equal to that of Earth, absorbs sunlight on an area of πR^2, while radiating from an area of $4\pi R^2$, and comes to equilibrium at a temperature of about 280^oK, while a flat black collector, like a solar cell, comes to equilibrium at about 330^oK. Thus, <u>any</u> system whose rejection temperature is as cool as a warm solar cell must have the same radiating area as a solar array. There is an enormous advantage to higher radiating temperatures, and a factor of three in radiating temperature (typical, for example, of thermionic systems) translates to a factor of eighty in radiating area. It is worth pursuing, despite the inevitable materials problems.

Apart from more visionary direct-conversion thermionic schemes, most radiator plans are one or another form of fin-ancradiator system, with heat transfer from the reactor accomplished by a working fluid, or by direct conduction at the lower power levels. The working fluid may play other roles, or may be in a heat pipe, but the principle is the same. All such schemes are limited as described above.

3.5 Power Conditioning

We have very little to say here, except to note that nuclear electrical supplies are appropriate to missions requiring steady supplies of power. Missions requiring large bursts of pulsed power (such as the illusory directed-energy weapons) require power conditioning in a form we have not discussed. Whether this should take the form of expendable fuels or storage via batteries, flywheels, inductive stores, capacitors, or whatever, is beyond our scope.

3.6 Shielding

Shielding is an integral part of any spacecraft design, to protect sensitive parts of the payload from the ravages of nuclear radiation, but the design of the shielding depends critically upon details. Some military payloads, already hardened against nuclear attack, may require little help, while others may need a great deal of protection. We see no way to design an all-purpose shield (shields are heavy), and believe it wasteful to put the entire burden of shielding on the power supply designer, and none on the payload designer. Yet the designs we have seen tend to incorporate a "shadow shield" designed to do just that.

Given the increasing hardness of modern electronics, we think it wise to divide the burden, and to reduce the weight of whatever shadow shield may be required by providing special protection, at the payload, of its sensitive parts. We call this the jockstrap concept.

3.7 Safety

No military reactors will be launched, in the present and foreseeable climate, unless there is careful attention to both the real and the apparent safety of the system and procedures. To be sure, there is safety inherent in the fact that one contemplates a "cold" launch, with the reactor activated once it has achieved a stable orbit. Thus, an aborted launch brings no fission products back to Earth. On the other hand, once a reactor has been operating, it is a repository for fission products, which will then need to be stored for at least several hundred years. The Soviets accomplish this by an end-of-mission boost into a higher and therefore longer-lived orbit, but the reliability of that system was illustrated by the failure of COSMOS 954.

There exists an interagency system for the approval of space nuclear launches, but we have no information on its effectiveness. We do think that this subject warrants real attention in DOD.

4.0 SUMMARY

In summary, we believe that there really is an impending need for nuclear power in space, and that there is a surfeit of ways to get there from here. The most highly developed (in the sense of design) concept is the Los Alamos SP-100, a 100 kwe design incorporating heat pipe cooling, thermoelectric conversion, and a shadow shield that constitutes over 25% of its weight of nearly 3000 kg. It is a point design, and serves, in our view as a clear demonstration that one can design a nuclear electric system in this power range.

On the other hand, although Solar/Battery systems as large as 100 kw cannot be ruled out on technical grounds, operational and cost considerations appear to make use questionable for Air Force systems at this power level.

The specific cost ($/W) of solar arrays has decreased over the past twenty years, principally because the size of the arrays has increased. An informal industry survey (Randolph, 1981) puts the cost of solar arrays between $500 and $1000/W for a 1981/82 time frame. However, recent NASA studies suggest design concepts for achieving planar or concentrator solar array at costs below $100/W. The technological feasibility and verification of economic payoff remain to be demonstrated for such concepts, and hardening considerations have not been considered in these low cost options.

A study completed in 1982 by General Electric for DARPA concludes that the Solar/Battery power system costs for the 25 kw and 100 kw power levels will continue in excess of $1000/W in the 1990 time frame. This study also compares the R&D costs for both Solar/Battery systems and reactor systems at the 100 kw level and the 25 kw level. It concludes that at the 100 kw level for both systems the R&D costs leading to a first flight unit will be about $600 M. The cost of follow on units are estimated to be about $200 M for the Solar/Battery system and $60 M for the reactor system. At the 25 kw level, R&D costs for the reactor system are about the same as for a 100 kw system while R&D costs for the solar system are estimated to be less than $100 M. Follow on units at the 25 kw level are estimated to be about $40 M for either system.

Because of R&D schedules and the necessity to develop solar-cell and battery production facilities for a 100 kw solar system or thermoelectric cell production capability for a reactor system, it is concluded that an immediate program start is necessary if 100 kw systems are needed in the 1990 time frame. It should also be noted that the weight and volume requirements for a 100 kw Solar/Battery are believed to be beyond the planned capability for shuttle/OTV launch. At the 100 kw power level the very large solar panels do not appear desirable for survivability, mechanical, and maneuverability reasons.

Because of the large R&D costs associated with 100 kw power systems, it would be desirable to find ways to spread this buy in cost over several programs, since no one space program will likely be able to afford the total cost. Unfortunately, space systems appear to be the only viable users of these specialized systems. The reactors and high temperature radiator cooling systems appear to be only useable in space and therefore not useable in other Air Force prime power applications. Reactors might, however, be combined in some way to provide power for maneuverability or even orbital transfer as well as supplying electrical power for the space systems.

It appears that the Air Force shoulu develop a Statement of Operational Need (SON) for spacecraft use of nuclear power systems. We believe, however, that the wide variety of technical options we have outlined above suggest that some technical competition is in order before the nation plunges on a specific system. We know all too well that the better is often the enemy of the good, but we also believe that the need for nuclear electricity in space, while real, is not so proximate that one need forego an orderly choice of technology. We are not advocating a dilatory selection procedure as a means of avoiding the issue, but think that we can afford the luxury, in this case, of developing the mission requirements concurrently with the technical selection. After all, they do influence each other. It is also essential that the relevant Government agencies, DOD, NASA, and DOE, all contribute their expertise and resources toward a timely resolution of the technical and mission questions. This will require modest R&D expenditures (a reasonable number of millions of dollars a year for the next few years) to assure a meaningful outcome, but the price of entry for such a nuclear electric system is going to be of the order of a half-billion to a billion dollars anyway, and a reasonable number of tens of millions spent to do it right would seem to be prudent.

BIBLIOGRAPHY

Anspaugh, B. E., and R. G. Downing, "Damage Coefficients and Thermal Annealing of Irradiated Silicon and GaAs Solar Cells," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 499-505.

Baraona, C. R., C. K. Swartz, and R. E. Hart, Jr., "Comparative Radiation Testing of Solar Cells for the Shuttle Power Extension Package," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 237-238.

Barthelemy, R. R., "Photovoltaic Outlook from the Department of Defense Viewpoint," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 14-16.

Buden, D., and J. Stooky, "SP-100 Functional Requirements Nuclear Space Power System," Los Alamos National Laboratory Report No. SP-100-1, 19 Jan. 1982.

Buden, D., "The Acceptability of Reactors in Space," Los Alamos National Laboratory Report No. LA-8724-MS, April 1981.

Buden, D., G. A. Bennett, K. Cooper, K. Davidson, D. Koenig, L. B. Lundberg, R. Malenfant, H. Martz, W. Ranken, R. E. Riley, and F. Schilling, "Selection of Power Plant Elements for Future Reactor Space Electric Power Systems," Los Alamos National Laboratory Report No. LA-7858, September 1979.

Chai, An-Ti, "Back Surface Reflectors for Solar Cells," Conference Record of the Fourteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1980), pp. 156-160.

Conway, E. J., G. H. Walker, and J. H. Heinbockel, "A Thermochemical Model of Radiation Damage and Annealing Applied to GaAs Solar Cells," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 38-44.

Görgens, B., H. Bebermeier, and G. Behrens, "Design Approach Toward 100 kw Flexible Solar Arrays," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 544-549.

Heinböckel, J. H., E. J. Conway, and G. H. Walker, "Simultaneous Radiation Damage and Annealing of GaAs Solar Cells," Conference Record of the Fourteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1980), p. 1085.

Hsu, Lan, "A Pentahedral Pyramidal Concentrator Design for Space Solar Array," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 560-564.

Jefferies, K. S., "Analysis of GaAs and Si Solar Cell Arrays for Earth Orbital and Orbit Transfer Missions," Conference Record of the Fourteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1980), pp. 1164-1168.

Keddy, E. S., and H. E. Martinez, "Development of High-Temperature Liquid Metal Heat Pipes for Isothermal Irradiation Assemblies," Los Alamos National Laboratory Report No. LA-UR-82-1273.

Loo, R., G. S. Kamath, and R. C. Knechtli, "Radiation Damage in GaAs Solar Cells," Conference Record of the Fourteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1980) pp. 1090-1097.

Loo, R., R. C. Knechtli, and C. S. Kamath, "Enhanced Annealing of GaAs Solar Cell Radiation Damage," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 33-37.

Randolph, L. P., "Photovoltaic Outlook from the NASA Viewpoint," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 10-13.

Scott-Monck, J., and D. Rockey, "Technology Requirements for GaAs Photovoltaic Arrays," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 539-543.

Weinberg, I., and C. K. Swartz, "Reduced Annealing Temperatures in Silicon Solar Cells," Conference Record of the Fifteenth IEEE Photovoltaic Specialists Conference, (IEEE, New York, 1981), pp. 490-494.

DISTRIBUTION LIST

Dr. Marv Atkins
Deputy Director, Science & Tech.
Defense Nuclear Agency
Washington, D.C. 20305

Dr. Robert Cooper [2]
Director, DARPA
1400 Wilson Boulevard
Arlington, VA 22209

The Honorable Richard DeLauer
Under Secretary of Defense (R&E)
Office of the Secretary of
Defense
The Pentagon, Room 3E1006
Washington, D.C. 20301

Director [2]
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note

CDR Timothy Dugan
NFOIO Detachment, Suitland
4301 Suitland Road
Washington, D.C. 20390

Dr. S. William Gouse, W300
The MITRE Corporation
Washington Operations
1820 Dolley Madison Blvd.
McLean, VA 22102

Dr. Edward Harper
OPNAV-021T
The Pentagon, Room 4D544
Washington, D.C. 20350

(b)(3):50 USC §403(g) Section 6
P.O. Box 1925
Washington, D.C 20013

(b)(3):50 USC §403(g) S
P.O. Box 1925
Washington, D.C. 20505

MAJ GEN Donald L. Lamberson [3]
Assistant Deputy Chief of Staff
(RD&A) HQ USaf/RD
Washington, D.C. 20330

The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102
ATTN: JASON Library, W002

Mr. John F. Kaufmann
Dep. Dir. for Program Analysis
Office of Energy Research
(ER-31, A1-4000)
U.S. Department of Energy
Washington, D.C. 20545

Dr. George A. Keyworth
Director
Office of Science & Tech. Policy
Executive Office of the President
Washington, D.C. 20500

Dr. Donald M. LeVine, W385 [3]
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102

Dr. Verne L. Lynn
Deputy Director, DARPA
1400 Wilson Boulevard
Arlington, VA 22209

Dr. Joseph Mangano [2]
DARPA/DEO
1400 Wilson Boulevard
Arlington, VA 22209

Dr. Michael May
P.O. Box 808
Lawrence Livermore National Lab
Livermore, CA 94550

Distribution List
Concluded

Mr. John McMahon
Dep. Dir. Cen. Intelligence
Washington, D.C. 20505

Director
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note

(b)(3):50 USC §403(g)
P.O. Box 1925
Washington, D.C. 20013

Director
National Security Agency
Fort Meade, MD 20755
ATTN: (b)(3):50 USC §402 Note
DDR-FANX 3

Prof. William A. Nierenberg
Scripps Institution of
Oceanography
University of California, S.D.
La Jolla, CA 92093

Dr. Allen M. Peterson
SRI International, (EL-154)
333 Ravenswood Avenue
Menlo Park, CA 94025

Mr. Alan J. Roberts
Vice President & General
Manager
Washington C^3 Operations
The MITRE Corporation
1820 Dolley Madison Boulevard
Box 208
McLean, VA 22102

(b)(3):50 USC §402 Note
P.O. Box 1925
Washington, D.C. 20505

Dr. Eugene Sevin [2]
Defense Nuclear Agency
Washington, D.C. 20305

Dr. Joel A. Snow
Senior Technical Advisor
Office of Energy Research
U.S. DOE, M.S. E084
Washington, D.C. 20585

Mr. Alexander J. Tachmindji, W300
Vice President & General Manager
Washington C^3I Operations
The MITRE Corporation
1820 Dolley Madison Boulevard
McLean, VA 22102

Dr. Al Trivelpiece
Director, Office of Energy
Research, U.S. DOE
M.S. 6E084
Washington, D.C. 20585

Dr. James P. Wade, Jr.
Prin. Dep. Under Secretary of
Defense for R&E
The Pentagon, Room 3E1014
Washington, D.C. 20301

Mr. Leo Young
OUSDRE (R&AT)
The Pentagon, Room 3D1067
Washington, D.C. 20301

[This page is intentionally left blank.]

[redacted]