

CHAPTER 7. OTHER SOVIET HIGH-PERFORMANCE COMPUTING PROJECTS

7.1 Introduction

In this chapter we discuss at greater length HPC projects introduced in chapter 3 which were not part of the case studies of the preceding chapters. We present additional detail of the technical characteristics of the machines, identifying major design influences and principles, give basic chronologies, and discuss factors which influenced their development and ultimate fate. The next chapter, chapter 8, will consist of a cross-cutting discussion of these systems and those of the preceding chapters, and the organizations in which they were developed.

7.2 Modular Pipeline Processor (MKP)

The Modular Pipeline Processor (MKP) is, besides the El'brus series, the second major high-performance computing project in progress at the Institute of Precision Mechanics and Computer Technology (ITMVT). The project is headed by A. A. Sokolov, who headed the AS-6 project and was directly involved in BESM-6 development [Ryab91b]. Like other Soviet projects funded in the mid-late 1980s, the MKP sought to break the one Gflops threshold. In contrast to the El'brus-3, the MKP was designed for a broad set of users who would not necessarily be able or willing to pay very large amounts of money for the fastest system available [Supe91, 16]. The design goals also included developing a general-purpose system with a full complement of systems software in a manner which would reduce the cost of hardware and maintenance. The system is called "modular" because it was designed to be attached to a number of different machines, in a number of different configurations, depending on the needs of the user. G. G. Ryabov felt very strongly that the system should be designed for use in universities and scientific centers

with a wide array of applications; here the chances for building a broad user base were the greatest. These considerations resulted in the following design goals and principles [Ryab91b]:

- achieve qualitatively new level of performance in domestic supercomputers (billions of floating-point operations per second);
- design hardware and software which is reliable, easy to use, and easy to maintain;
- follow international trends in the development of information technologies (UNIX, networks standards);
- involve scientific centers and universities in the initial stage of development in the creation of systems software and an algorithmic base.

The following brief description of the MKP is taken from [Byak90; Byak91].

7.2.1 MKP Architecture

Logically, the MKP consists of a number of functional blocks shown in figure 7-1. The two scalar blocks do preliminary processing on a stream of program instructions. This includes instruction decoding, address calculation, control command formulation, and interrupt processing. Instructions are then sent to other MKP blocks for final execution. The scalar blocks have independent scalar processing resources, but treat the vector and vector sorting blocks as a shared resource. Synchronization of the two streams is done explicitly using semaphores.

The vector processing block manages the execution of vector arithmetic-logic operations. It does not perform the computation itself, but sets up the computation by managing the distribution of vector resources, reading and writing vectors to memory, and link-

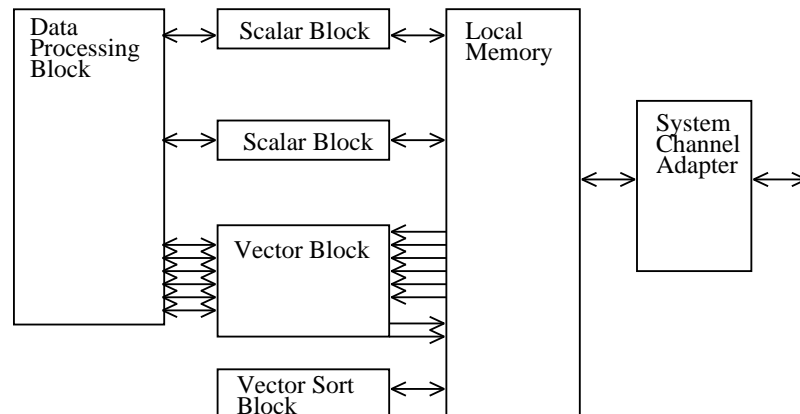


Figure 7-1 MKP Structure
Source: [Byak90; Byak91]

ing individual vector instructions. The vector sorting block provides a variety of sort options on vectors.

The scalar and vector blocks rely on the data processing block for the execution of real and whole-number arithmetic, logic operations, shifts, and a number of other operations on scalar and vector elements. Unlike traditional vector-pipelined (Cray-like) computers which have separate scalar and vector units with independent functional units, the MKP data processing block contains a number of pipelined functional units which can be used by both the vector and scalar blocks. This approach requires a more complex functional unit design, but considerably less hardware. In the standard MKP construction, there are eight pipelined functional units: two each of addition, multiplication, division, and logic. The precise mix was determined through an analysis of the applications of the primary customers (such as the Scientific Research Institute of Experimental Physics in Arzamas-16, a principal nuclear weapons design facility [Gig1930122, 2]), but the issue

was not a pressing one. In principle, the MKP can be built with a different mix of functional units to customize it to the needs of specific customers.

The two scalar blocks make it possible to execute two streams of instructions simultaneously. As they and the vector block set up the execution of individual instructions for the data processing block, the latter might simultaneously be executing vector and scalar operations from the same or different instruction streams.

In vector processing, the MKP allows for up to five vector operations to be chained together so that the results of some operations are used directly as arguments for others without an intermediate write to memory.

Scalar and vector operations are combined by buffering vector operators when they are passed to the data processing block. Scalar operations can be inserted arbitrarily between vector operators without upsetting the pipeline.

Table 7-1 lists the constituent parts of the MKP. Table 7-2 lists the operational parameters.

The earliest design work on the MKP began during the mid 1970s shortly after A. A. Sokolov's team finished work on the AS-6. The design of the machine was strongly influenced by the available component and construction technologies, the traditions of A. A. Sokolov's group, and the vector-pipeline concepts popularized by Cray Research, Inc. In keeping with ITMVT tradition, engineers sought to implement some of the basic pipeline ideas in a manner which was more appropriate to local conditions and design goals rather than try to duplicate the Cray architecture. According to one engineer [Li91]:

We always wanted to adapt the new ideas which we encounter to our capabilities. On the one hand, we set before ourselves the task of [building a machine with a peak performance of] billion operations per second. On the other hand, we ... don't want to place too high requirements [on the construction]. Therefore, we are obligated, when taking

<p>Data Processing Block 8 pipelined functional units operating on floating-point operands, whole numbers, bit sets, logical values</p>
<p>Scalar Block 4 64-bit adders 128 64-bit general-purpose registers 16 37-bit address-index registers 32 64-bit display registers 1 address functional unit operating on address pointers, indexes, descriptors, and semaphores</p>
<p>Vector Block 1 operator with 5 operations 2 operators with 6 operations 2 packed read vectors from the scalar block 2 vectors for reads to local memory</p>
<p>Vector Sort Block Simple reference packing/unpacking of vectors under direction of logic conditions scale and displacement vectors</p>
<p>Local Memory 2M or 8M 64-bit words 32 banks 12 ports</p>
<p>System Channel Adapter 32 parallel read/write operations</p>

Table 7-1 MKP Functional Blocks

Source:[Byak90, 9]

into account Western ideas, to find a means of implementing them with lower technological levels. We feel that such possibilities open up when we, on the one hand, incorporate the spirit of the new ideas, and on the other hand, try to draw from the most important, and not simply familiarize ourselves with the [Western] implementation and try to copy it. We try to find the realization of those ideas which we like in such a way that it's agreeable to us.... It appears to us that simply copying without having continual access

Word length	64 bits
Clock period	10 nsec
Local memory	
Size	16-64 Mbytes
Total throughput	8 words/cycle (6.4 Gbytes/s)
System adapter	
Total throughput	800 Mbytes/s (400 Mbytes/s in each direction)
Theoretical peak performance on fl. pt. operands	600 Mflops (one result per clock period in each of 6 fl. pt. pipelines)

Table 7-2 MKP Performance Characteristics
Source: [Byak90; Byak91]

to the literature and documentation, not having the ability to use the same components, that this is a dead-end approach. It absolutely leads to a situation in which we continually lag behind....

Designers sought to implement a pipelining scheme which could be supported by the technology available, provide higher performance than a Cray with a comparable clock period,¹ and at the same time use the hardware as economically as possible to reduce the cost of implementation. The clock period and the number of functional units were mostly determined by the number necessary to achieve the goal of one Gflops, given the component technology projected to be available when the MKP design was completed [Ryab90d, 1].² The chip and printed-circuit board technologies also placed an upper bound on the number of functional units and amount of local memory which the MKP

¹With a clock period of 9.5 nsec, the Cray X-MP has a peak performance of 210 Mflops per processor.

²The 1 Gflops threshold is actually achieved only in a configuration with two or more MKP.

could contain [Ryab90d, 3]. The final design discussed above reflects the tradeoff between these design goals.

Many of the architectural ideas for the interrupt system, monitoring of the system state, etc. reportedly grew out of the AS-6 project, although so little has been written about this machine that it is difficult to make a comparison.

7.2.2 El'brus-3-1

The MKP constitutes only the computational component of a full system. Through the system adapter, the MKP is attached to a channel system which provides connections to:

- shared main memory modules
- other MKP
- a variety of front-end systems, such as ES mainframes, El'brus-2, BESTA workstations, El'brus-B, etc.
- Peripheral devices

Such a complex, shown in figure 7-2, is called an El'brus-3-1. By allowing a variety of systems to function as an MKP front-end, ITMVT decision-makers hoped to provide a migration path to multiple classes of users: the ES mainframe, BESM-6, and El'brus-2 camps. The MKP is software compatible with none of these machines for a number of reasons. First, no reasonable system could be compatible with all three classes of machines. Second, the user community for the previous machine developed by Sokolov, the AS-6, was small. Third, and most importantly, the MKP was developed under the ideology that machines should be built to achieve the highest performance possible, even if that meant that they would not be compatible with other systems. The one Gflops threshold could not be achieved with an architecture compatible with the machines most widely

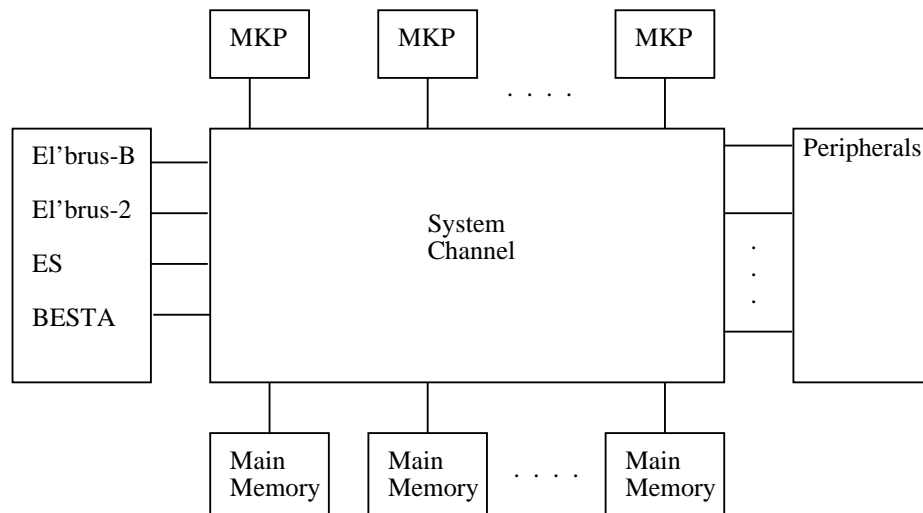


Figure 7-2 El'brus-3-1 Structure
Source: [Byak90; Byak91]

used in the early 1980s, the ES and BESM-6. Theoretically, compilers for El'-76 could be built to port El'brus applications to the new machine.

The El'brus-3-1 reflects the multi-machine philosophy that had developed during the construction of the AS-6. That machine can be thought of as a cluster of interacting systems connected by a network. The communication channel network of the El'brus-3-1 makes it possible to join a variety of different systems into a single complex.

7.2.3 MKP Development

The earliest design work on the MKP began not long after Sokolov's team finished working on the AS-6, in 1976. Like the BESM-6 and AS-6, the MKP was developed together with the Moscow SAM Plant. Prototype construction did not begin until the mid-1980s. Before then, work on ECL gate arrays with 1500 gates per chip had not progressed far enough. Also, this project was given significantly greater support from the di-

rectorate of ITMVT after G. G. Ryabov succeeded V. S. Burtsev as director in 1984 [Pent93c]. The first mock-up was built by 1988 and by December, 1990, the prototype was ready for the first test software routines [Alek91]. By the end of 1991, a total of four MKP had been built. The system passed state testing around the end of 1992.

The MKP uses much of the same technology—ICs, PCBs, CAD, cooling, etc.—that are used in the El’brus-3. Naturally, ITMVT could save considerable time, effort, and money by using the same technology in multiple projects [Supe91, 17].

Designing the functional units for both scalar and vector operations did add to the complexity of the hardware, but designers felt that the reduction in hardware and cost which this permitted made it a good tradeoff. Even so, the logic of the MKP is less complex than that in the El’brus machines. For example, the MKP does not use hardware tags, relying instead on descriptors, reducing the complexity of hardware control considerably. The use of less complex logic reportedly improved reliability [Supe91, 17].

In spite of efforts to design the MKP for use by a broad user base, and very active promotion by ITMVT, the market for the machine remained virtually non-existent through the beginning of 1993. At the October, 1991 Conference “Problems of the Development of High-Performance Computing Systems,” Ryabov and others made strong efforts to find new customers for the machine. Although less expensive than an El’brus, the MKP prototypes were not cheap; they cost on the order of 10-15 million rubles in 1991 [Supe91, 18].³ During 1992, no MKP units were manufactured; the total remained at four.

³ At this time, the ruble was valued at 20./\$.

7.3 Elektronika SSBIS (“Red Cray”)

In 1978 V. A. Mel’nikov left ITMVT to work at the Delta Scientific Production Association within the Ministry of the Electronics Industry (Minelektronprom) and took with him a number of members of the team that had developed the BESM-6 and the AS-6. In 1980 he began a project to build a vector-pipeline supercomputer based on the principles popularized by the Cray computers only a few years earlier. Mel’nikov continued many of Lebedev’s traditions, particularly that of trying to build the fastest machine possible on the “available” component base. The Elektronika SSBIS is noteworthy for a number of reasons. First, it is perhaps the only HPC system developed within Minelektronprom, a ministry with considerable experience in the development of microelectronics, but whose systems development was largely limited to mini- and personal computers. Second, to a greater degree than other Soviet HPC systems, it was originally patterned after Western computers, the Cray family. In this regard, Mel’nikov departed from the traditions of Lebedev. Engineers did implement a number of non-Cray features, particularly with regard to memory systems, however. Like other Soviet projects which stretched the limits of indigenous technologies, it had a development cycle of over ten years.

Ironically, the decision to develop a system with a Cray-like architecture rather than an indigenous one grew out of a desire to build a machine quickly. Leaders of the military-industrial complex and prominent politicians had become quite worried by the introduction of the Cray-1 into series production in 1976 [Sher92b]. The vector-pipeline architecture had been proven sound and effective by Cray Research Inc. and Control Data Corporation, and policy makers felt that by minimizing architectural innovation, the development cycle could be shortened considerably [Gig1921217, 1]. The El’brus computers would not be ready for use for a number of years, and in the early 1980s Western

export controls made it virtually impossible to acquire systems like the Cray, even if military users had wanted to run their code on a foreign system.

The Elektronika SSBIS was a high profile, high priority project. In 1984 the Delta Scientific Production Association became affiliated with the Institute of Cybernetics Problems (IPK), a new organization created as part of the Academy of Sciences Department of Informatics, Computer Technology, and Automation (OIVTA). This Department, headed by Academy Vice-President Ye. P. Velikhov, was designed to provide a forum within the Academy for coordinated computer policy-making. The development of a supercomputer was one of its main goals [Mikh84; Veli85]. V. A. Mel'nikov was named director of the IPK. Although it was subordinate to the Academy of Sciences, the IPK retained close contacts with Minelektronprom.

The system consists of 1-2 subsystems sharing a solid-state storage subsystem and peripheral storage. The basic system parameters are shown in table 7-3. The performance of 250 Mflops is achieved by generating one result in each vector functional unit per "synchronization period" (two clock periods) [Meln91].

The Elektronika SSBIS uses workstations or minicomputers as a front end and supports standard international network protocols (TCP/IP, Ethernet) and programming languages (FORTRAN 77, Pascal, C) [Elek91, 4-5]. The system is constructed using ECL gate arrays with 256 gates per chip. The 16-layer printed circuit boards, by some accounts of rather good quality, were designed and manufactured at IPK.

The Elektronika SSBIS is nearly compatible with the Cray at the level of assembler, but is not binary compatible. According to one individual intimately involved in development, developers had little information besides published descriptions of the Cray [Sher92b]. The memory system differs considerably from Cray's. Drawing on some ideas from the AS-6, developers designed an "intelligent memory" system. The solid-state

Number of CPUs	1-2	CPU speed CPU clock period vector performance scalar performance	13.5 nsec 250 Mflops 75 MIPS
Registers in each CPU address (24 bit) intermediate address (25 bit) scalar (64-bit) intermediate scalar (24-bit) vector registers (64x64)	8 64 8 64 8	CPU Technology	ECL
Instruction cache capacity number of instruction buffers	2 Kbytes 16	CPU Main memory word size (data + error correction bits) capacity number of banks bank cycle time max. transfer rate	90 bits 8-32 Mbytes 16 54 nsec 300 Mwords/sec
Number of CPU pipelines address scalar vector	2 7 7	Shared solid state storage word size (data + error correction bits) capacity number of banks bank cycle time max. transfer rate technology	90 bits 256 Mbytes 16 640 nsec 150 Mbytes/s CMOS/TTL
Word size scalar and vector operands address operands	64 bits 24 bits	Disk Storage Subsystem disks per subsystem number of subsystems max. capacity (317.5 Mbyte disks)	1-8 2-8 20 Gbytes

Table 7-3 Elektronika SSBIS Parameters
Source: [Elek91; Meln91]

memory subsystem has its own processor and instruction set which makes it more than a buffer between main and peripheral storage. It can perform complex memory retrieve functions (such as selecting the diagonal of a matrix) and send only the salient data to main memory. This increases the effective speed of the channels [Supe91c, 7].

The software is indigenous. Porting Cray code to the Elektronika SSBIS was never an issue: the hardware was not binary compatible, and system software developers had little desire to copy others' work. By developing their own systems software, they could improve the quality and maintainability of their code in future years.

Research on the Elektronika SSBIS began in 1980. Gate arrays designed for this machine with 256 gates/chip became available to developers in 1982-1983. Early specifications were ready by 1982 and by 1985 documentation for the prototype was completed and turned over to the factory to begin prototype construction [Gigl921217, 1]. The first prototypes neared completion in 1989 and underwent state testing in that year. By the end of 1991, approximately five prototypes had been constructed at the Kvarits Plant in Kaliningrad (Königsberg) on the Baltic Sea [Sher92, 9; Sher92b].

Although the machine incorporated a proven architecture, many of the same systemic factors which delayed development of the El'brus computers affected the Elektronika SSBIS. Having close contact with Minelektronprom helped developers get the necessary ECL gate arrays. Like the El'brus computers, the Elektronika SSBIS helped drive the development of many technologies. The monopolistic nature of the ministry and taut planning created a reluctance by factories to assimilate production of new technologies.

Unlike the Ministry of the Radio Industry (Minradioprom), Minelektronprom did not have a well developed infrastructure for developing large-scale systems. Not only did new design, construction, and manufacturing technology have to be developed, but the inter-organizational contacts between IPK and participating factories and institutes had to be established. During the 1980s these ties were not horizontal, but ran through the ministerial hierarchy. Establishing this infrastructure took an enormous amount of time and was a major cause of delay.

By the end of 1992, two Elektronika SSBIS had been installed at user locations. The project had been sufficiently near completion that financing for development continued at least through the end of 1992 [Gigl921217, 1]. Volume production of the systems is unlikely. There are few users who can afford to pay millions of rubles for a computer which is still not mature. Four more systems, already constructed, stand idle, without customers to purchase them. Like the El'brus systems, the Elektronika SSBIS depends on products from plants throughout the former Soviet Union. Disruptions in supplies can hinder production. For example, connectors were manufactured in Armenia, but production in that state has come to a virtual standstill. The Kaliningrad plant was able to manufacture Elektronika SSBIS prototypes only because it had stockpiled connectors in advance [Gigl921217, 3].

Engineers at NPO Delta have designed a multiprocessor configuration with up to 16 processors, but prospects for funding and the development of the necessary component base are dim. A small-scale entry-level system is also being designed [Gigl921217, 4].

7.4 ES-1191

Until the mid-1980s, the Scientific Research Center for Electronic Computer Technology (NITsEVT) was involved in HPC only as a facilitator, providing general-purpose mainframe hosts for attached array processors (AAP) and other specialized processors. Around 1985, NITsEVT engineers began work on a general-purpose supercomputer containing vector-pipeline processors which were integrated into the system architecture from the outset, rather than added on, as AAPs. The design ideas of what was later called the ES-1191 were presented at conferences in 1986 [Lomo86; Valk87]. The basic approach of designing general-purpose machines with vector processors was adopted by IBM in the 3090 VF (Vector Facility) series, announced in 1985, but there are significant differences in the architectures of the 3090 VF and ES-1191 [Dpro86].

The ES-1191 grew out of the desire of many engineers and policy makers during the early-mid 1980s to reach the one Gflops threshold on 64-bit data. Three additional development goals were: 1) to design a system for which the nominal processing rate would be close to the peak performance; 2) to incorporate programming methods already in use on the general-purpose mainframes, preserving software compatibility and the investment in systems software; 3) to separate as clearly as possible the processing functions from the control functions in order to free up the high-powered computational units from routine tasks which don't require high-speed computing [Loma86, 60-61; Lomo91].

Recognizing that the key to high nominal performance in a vector processor is high scalar performance [Loma86, 61], NITsEVT engineers designed a system with a conventional ES mainframe used as a control system, plus a computational subsystem consisting of four scalar processors, a vector processor, a memory management device, a monitor processor, and expanded main memory [Loma86; Lomo91]. The scalar processors treat the vector processor as a shared resource. The basic scalar portion, with a performance of 15-60 MIPS depending on the number of scalar processors, is called an ES-1181. A configuration consisting of the ES-1181 and a vector processor is called an ES-1191. The control system is responsible for I/O, program compilation, data exchange between the computational subsystem and the control processor, etc. [Loma86, 64].

The vector processor contains sixteen pipelines; there are four each of addition, multiplication, division, and fixed-point logic [Loma86, 67]. An additional pipeline handles reads/writes from/to memory. The theoretical peak performance, 1.07 Gflops, is achieved when the sixteen pipelines each generate one result per 15 nsec clock period. The vector processor contains a reconfigurable register file with a total of 16K 64-bit elements. These can be organized as sixteen files of 1024 elements, 256 register files with 64 elements, etc. [Loma86, 68].

The ES-1191 is one of three Soviet systems (the others are the PS-2100 and Elektronika SSBIS) which incorporate solid-state external storage as an intermediate stage between main and peripheral storage. The 256 Mbytes of main memory are augmented by up to two Gbytes of so-called “expanded RAM” with a transmission rate close to one Gbyte/s [Lomo91].

Much of the systems software development was carried out at the Institute of Applied Mathematics in Moscow. This includes the development of parallel programming languages and an assembler-level language oriented towards the parallel system [Yush91].

The ES-1191 was originally scheduled for completion by 1989 [Valk87]. As the *perestroika* reforms progressed, the development time was drawn out. Although financing continued, it remained problematic. It was sufficient to support the development teams, but insufficient to get the customized integrated circuits built by the electronics industry. As users and factories began operating in self-financing modes, there were few who could afford to allocate the huge sums required for the development, manufacture, and purchase of such a machine. The design reportedly was scaled back to include four pipelines rather than sixteen. Others have reported that by the end of 1992, only a scalar version of the machine was being developed. But even individuals within NITsEVT remain skeptical about the prospects. “Who would have use for such a machine? No one,” commented one senior NITsEVT manager.

7.5 Other HPC at the Scientific Research Institute of Control Computers

7.5.1 PS-3000

Like the PS-2000, the PS-3000 grew out of a collaboration between the Institute of Control Problems in Moscow (IPU) and NIIUVM. During the mid-1970s, researchers at

IPU under V. V. Ignatushchenko worked on parallel processing ideas and developed ideas about pipelined processing and dynamic allocation of computing resources.

To overcome the limitations of a static architecture typical of SIMD machines in which one control unit has exclusive access to a fixed set of processing elements, Ignatushchenko designed a two-phase execution process in which control units and processor fields were decoupled from one another. A set of control units would operate independently and in parallel, feeding instructions into a single buffer. A set of so-called processing element blocks, each consisting of multiple processing elements, would independently and in parallel execute instructions. Any instruction could be initiated by any control unit and executed on any processor block. Simulations showed that such an approach could result in 22-36% greater loading coefficient than a rigid configuration [Igna84, 128-137].

The other key conceptual idea was pipelining, now a standard feature of supercomputers. The idea itself cannot now be considered a novelty, but Ignatushchenko claims that he developed it independently and rather fully before architectural details of the Cray-1 computer were disseminated in the Soviet Union. He claims that evidence that Cray Research incorporated pipelining ideas proved convincing justification for his ideas and played a large role in convincing authorities to support his project. The Cray-1 was not the first computer to implement pipeline processing, however. The CDC 7600, introduced in 1969, incorporated eight pipelined functional units [Hock88, 16] and the basic ideas of overlapped execution of different computer functions had been around considerably longer than that [Rama77]. We do not know to what degree Ignatushchenko was familiar with the CDC 7600, although although such information could have been available, given CDC activities in the Soviet Union during the 1970s.

In its implementation, the PS-3000 incorporated only the dynamic linkage between control units and processing elements. True pipelining of functional units was deemed too complex to implement. As an alternative, a quasi-SIMD vector processor consisting of multiple processing elements was created which in many respects mimicked the operation of a true vector-pipelined processor. Although its performance on various vector operations was more uneven than that of Western pipeline processors, it did perform reasonably well on many operations, given the technology with which it was constructed.

NIIUVM-IPU collaboration on the PS-3000 began in 1975/76, and construction of the prototype began around 1978. Although a prototype was completed sometime between 1982 and 1984, the machine never entered large-scale production. On the order of ten machines were manufactured.

The 32-bit PS-3000 multiprocessor was designed for use as the top level of complex hierarchical data processing and control systems, such as real-time systems in atomic energy plants, simulation systems, etc. Key requirements were high performance, high reliability, and compatibility with prior processor control systems developed at NIIUVM.

The PS-3000 consisted of two or four scalar processors where each pair of scalar processors was associated with a single vector processor. Scalar processors executed different tasks, or different branches of the same task, but used the shared vector processors to execute vector instructions [Impu85, 5; Iosh87, 114]. Of the approximately ten machines which were manufactured, no full configuration machines were built. Reportedly, all systems contained two scalar processors and one vector processor.

While Ignatushchenko designed the vector processors to use pipeline processing and the Cray experience demonstrated the viability of this approach, the PS-3000 vector processors did not use true pipelining. As explained by V. M. Borisenko, one of the chief NIIUVM engineers on the project, the representatives from the prototype development

factory objected to the complexity of a true pipeline implementation. They claimed that it was too difficult to manufacture a pipeline given the SSI and MSI chips available at the time. As a compromise, they implemented the vector processor as a set of eight processing elements operating in a quasi-SIMD mode [Iten85]. The vector processor consisted of a control unit and a processor field. The control unit received operations from the scalar processor, divided them into instructions and data and, when operands were available for a particular operation, placed the instruction and operands in a buffer of instructions to be executed. The processing elements operated independently of each other, fetching the next operation in a buffer for execution. In practice, vector operations involved the execution of the same instruction on multiple data items. Thus the operands were fed to the processing elements in a round-robin fashion. In contrast to a true SIMD architecture in which all processing elements operate in lock-step, the PS-3000 processors operated independently such that PEs could begin executing the next vector operation even if not all PEs had finished executing the current one [Iten85].

The scalar processor incorporated some pipelining at the system level, in the instruction processing functions. This concept, in which the fetching, decoding, operand fetch, and instruction execution of one instruction can be overlapped with that of another was first incorporated commercially by IBM in 1977 in its 3033 mainframe [Lusa78; Pras89]. It was not common practice when the PS-3000 was first designed.

All processors shared a field of four or eight Mbytes of memory, organized in units of two Mbytes each. The units had independent power supplies and independent links with CPUs and I/O processors. The shared memory and independent operation of modules not only facilitated inter-process communication and enabled a single, re-enterable copy of the operating system to serve all processors, but also made it possible to run the system in

a fully redundant mode to increase reliability [Impu85, 5; Iosh87, 114]. The system had an addressable virtual memory space of 256 Mbytes [Reza83; Impu85, 3].

To enable the central processing units to devote more time to computational tasks, I/O tasks were off-loaded to peripheral I/O processors which handled I/O dispatching, I/O program execution, and queue control. The I/O processors were attached to the scalar processors and provided exchange of data not only between peripherals and main memory or the scalar processors, but also between configuration subsystems [Impu85, 5-6; Iosh87, 114]. The latter included a data processing subsystem based on a PS-2000 [Impu85, 5]. Each configuration subsystem was attached to two I/O processors, primarily to provide redundancy and improve system reliability [Impu85, 6].

The PS-3000 served as the basic computation engine for so-called regional geophysics computing systems (RGVK). Three RGVK versions were planned, the K143-12, K143-13, and K143-14, shown in table 7-4. The three differed primarily in the number of processors and mix of peripheral devices.

Why did the PS-3000 never enter series production? Individuals give different answers, but a common thread between the explanations is a lack of effort on the part of developers to see their project through. Some claim that by the time the machine was ready for series production, the component base had become completely obsolete; developers felt that their energies were better spent designing the next generation than investing the time, energy, and resources necessary to both see the machine through to production and establish the support network necessary to maintain it in the field. Others claim that the project was terminated for economic reasons. The Severodonetsk Instrument-Building Factory was reluctant to manufacture the PS-3000. In addition, there was little demand for the machine. As the Soviet Union opened to a flood of imported high-end personal computers in the late 1980s, many potential customers preferred to purchase

	K143-12	K143-13	K143-14
Number of scalar processors	2	4	2
Max. performance on fixed-point add (MIPS)	6	12	6
Number of vector processors	1	2	1
Max. performance on fixed-point vector adds (MIPS)	10	20	10

Table 7-4 PS-3000 Configurations
Source: [Impu85]

personal computers which, although not parallel machines, compared not unfavorably with the PS-3000 in terms of performance, memory, and reliability. Still others lay the responsibility solely at the feet of the developers, claiming that they simply did not make the effort to push the project through to completion. Each of these perspectives undoubtedly has merit and the final answer probably lies in a combination of these reasons.

7.5.2 PS-3100

The successor to the PS-3000, the PS-3100, was at the scientific-research stage (*nauchno-issledovatel'skaya razrabotka*) in 1990. It differs from the PS-3000 in a number of ways. First, the processing elements within the vector processor will contain a small cache, accessible to the programmer. The number of vector registers will be increased, and the physical dimensions will be decreased. In addition, a number of changes to the basic approach are being made to make the system more marketable. These include implementing Unix as the basic operating system, writing a C compiler to increase the amount of software available, and designing the vector processor so that it can interface with a number of processors besides the system's own scalar processor. Given a standard

set of interfaces, the PS-3100 vector processor could become a high-performance attachment to existing machines. Given the lack of funding for the project, however, it is not likely that it will be completed.

7.6 HPC at the NII of Computing Systems (NIIVK)

7.6.1 M-10

During the 1970s a multiprocessor called the M-10 was developed at the Scientific Research Institute of Computing Systems (NIIVK) under the direction of M. A. Kartsev. Kartsev had worked on some of the earliest Soviet systems, including the M-2 which was prototyped in 1952 [Tadz77, 6-7; Yers80b]. The M-2, a medium-sized, general-purpose machine running at 2000 operations per second and emphasizing economy and efficiency over size, was not considered a “high-performance” system [Kuzn61; Rudi70; Yers80].

We do not know what Kartsev, who had close ties to the military, worked on during the 1960s, but during the 1970s he focused on signal and image processing systems [Supe91, 27]. The M-10 was a vector-oriented multiprocessor for such applications. The machine had multiple types of processors for various functions which shared main memory. The main processor section consisted of two “lines” (sets) of eight 16-bit processors, each set operating in SIMD mode. An additional specialized processor performed logic operations on boolean variables in parallel with the main processor lines. These could provide conditional control functions and masks for regulating the main processors. A further specialized processor performed index operations. It had an average performance of five MIPS, and five Mbytes of main memory [Kart79; Kart81; Sots790913]. The M-10 operating system could support multi-tasking with up to 48 batch and interacting jobs [Bely80].

The distinguishing feature of the M-10 was its ability to process conveniently in parallel data of different formats, dynamically changing the clustering of processor to match the format of the data at hand. Data could be fetched from memory in units of 2-64 bytes and the eight 16-bit processors could execute identical instructions on two 64-bit, four 32-bit, or eight 16-bit numbers [Kart79].

Of the five Mbytes of ferrite-core main memory, one Mbyte was reserved for the resident portions of the operating system and four Mbytes were available to users [Kart80; Gakh82]. Each process could access up to four Mbytes of virtual memory. The system used peripherals designed for the ES mainframes [Kart79; Sots790913; Grin82].

It was possible to execute different instructions on different pieces of data (MIMD), but higher processing rates were obtained in SIMD mode. The processors each executed instructions at an effective rate of one per 1.8 microsecond machine cycle [Kart79; Kart81b].

The M-10 was used as a platform for the development of early parallelizing compilers. The earliest efforts, such as a compiler for Algol-60 completed in 1974, focused on parallelizing sequential programs. Later efforts on FORTRAN compilers for the machine placed the burden of indicating parallelism on the programmer [Berk85; Prog86; Varc86].

The M-10 was operational by 1979, but possibly several years earlier [Sots790913; Varc86]. We do not know how many M-10s were manufactured, but published reports seemed to indicate that this was a serious industrial machine. A number discuss actual use in modeling seismic activity and plasma behavior [Bere80; Same85]. The M-10 continued to be used well into the 1980s [Golo85; Golo87].

7.6.2 M-13

Kartsev died in 1983 and was succeeded as director of NIIVK by A. A. Novikov [Kart83]. Under Novikov's direction an M-10 successor, the M-13, was developed and passed state testing in 1990 [Supe91, 27].

The M-13 was designed primarily for real-time applications. Building on Kartsev's work, it incorporates multiple heterogeneous processors operating on shared memory. The three basic processor types are the central control processor, the central processor for vector operations, and specialized processor for executing specialized functions [Grin90, 4]. The 32-bit vector processor reportedly runs at 50 MIPS [Supe91b, 9]. It contains six vector registers each holding up to 256 vector elements [Grin90, 13]. The vector processor has eight multi-format arithmetic units. Each can function as one 64-bit unit, two 32-bit units, four 16-bit units, or eight 8-bit units [Supe91b, 8].

The most significant features of the architecture, according to [Grin90], are the organization of memory using a memory controller and the address protection mechanism. The M-13 implements capability-based addressing in which each process has access, potentially, to the entire virtual memory space. This concept, developed in a number of Western systems such as the Cambridge CAP during the late 1970s, does not limit each process to sharply delineated portions of virtual memory, as is conventional in most commercial systems and in the M-10 as well [Grin90, 15-18]. Although capability-based addressing allows more flexible sharing of data between processes, it places greater demands on the system which must guard against unsanctioned memory access. The M-13 incorporates a number of memory protection features in hardware.

The M-13 has a 300 nsec cycle time and is built using quite conservative Logika-2 TTL technology, the same used in the El'brus-1 during the late 1970s [Supe91b, 9].

7.6.3 El'brus-M14E

Another high-performance project carried out at NIIVK during the late 1980s was called the El'brus-M14E. This machine was an effort to implement a system having the same basic architecture as the El'brus-3 at ITMVT, but on a smaller scale [Baba89]. Naturally, there was close cooperation between ITMVT and NIIVK on this project. This 64-bit system was designed to have 2-8 processors in the maximum configuration. Each processor, built using ECL technology was to have a clock period of 25 nsec and a peak performance of 160 Mflops. The full configuration would have a theoretical peak performance of 1.28 Gflops [Baba89, 879]. This project had effectively ended by 1991 because of a lack of financing.

7.7 Dynamic Architecture Machines

Valeriy A. Torgashev and others have conducted research on dynamic architecture machines (MDA) since the late 1960s, first at the Leningrad Institute of Aviation Instrument Building (LIAP) and later at the Leningrad Institute of Informatics and Automation of the USSR Academy of Sciences (LIIA). Earlier references to these machines use the term “recursive architecture machines.” The earliest presentation of some embryonic ideas is found in [Glus74]. Arguing that the von Neumann architecture—characterized by low-level machine language representations, sequential execution, and linear storage in memory—required a burdensome amount of complex software to bridge the distance between the hardware and the applications to be executed, the authors proposed a so-called recursive architecture solution. Such a machine would be characterized by a recursive internal language, a recursive parallel control method, a recursive memory organization, and recursive internal and external architectures. The internal language would allow a programmer to define a set of low-level primitives and then construct additional levels of language elements that themselves would be viewed as entities, but which would be de-

composed during execution. Control would be based on dataflow principles in which operators (at any level) would execute as soon as all the necessary operands were available. Memory would be stored and referenced as objects, rather than memory locations; the logical-to-physical mapping would be done at run-time. The internal architecture, the set of connections between processors, would dynamically change in response to the structure of the task being executed. Finally, the machine would consist of a hierarchy of physical switches that would allow one to cluster processors physically together. These ideas, expressed in very general terms in [Glus74] have, to varying degrees, been incorporated into Torgashev's later work on dynamic architecture machines.

Torgashev's work falls into the general category of non-von Neumann architectures that are classified as data-driven and demand-driven [Trel82]. In dataflow machines, the availability of operands triggers the execution of the operation to be performed. In demand-driven (reduction) computers, the requirement for a result triggers the operation that will generate it. Two primary motivations behind these developments are the desire to improve performance by uncovering and utilizing the greatest amount of parallelism possible in a program, and to support declarative languages, particularly functional languages which have desirable properties for programmability and execution [Huda89; Vegd84].

The open literature does not provide a great amount of detail about Torgashev's machines and it is difficult to see through Torgashev's terminology to make a clear comparison with Western work. Comparisons made here are, therefore, somewhat tentative. Torgashev's machines are described in [Pono83; Torg84; Pono87; Torg88; Torg89].

The underlying model of computation is the dynamic automata network (DAN). In such a network, the division of a problem into algorithms and data (typical in traditional programming) is minimized. Each automaton represents an object of a given problem,

which can be data, operations, relations, references (pointer-like entities), or physical machine resources. Automata can be complex, consisting of other, simpler automata. The execution of a problem, represented by a DAN, consists of applying transformations to the DAN until no further transformations are possible. The final structure represents the solution. An important point is that not only data can be transformed, but also other types of automata, such as relations and operations.

Treleaven et al. describe the program organization as the way machine code programs are represented and executed in a computer architecture. Two basic categories of mechanisms are the data mechanism that defines the way a particular argument is used by a number of instructions, and the control mechanism, which defines how one instruction causes the execution of another [Trel82]. It appears that the MDA use at least two of the three data mechanisms described by Treleaven—by value and by reference. It also appears that the MDA make provisions for using each of the three control mechanisms described in that article: sequential, where a single thread of control signals passes from one instruction to another; parallel, where control signals the availability of arguments and an instruction is executed when all its arguments are available (as seen in dataflow); and recursive—where control signals the need for arguments and an instruction is executed when one of the output arguments it generates is required by invoking the instruction.

The lowest level of automata are represented by so-called program elements that consist of code fragments to execute individual instructions (operator automata), to provide simple or complex memory access (data automata), to verify a relationship (relation automata), and to manage the assignment of program elements to physical resources (resource automata).

The ES-2704 and ES-2727 are two implementations of Torgashev's ideas. The structure of the ES-2704 is shown in figure 7-3. The machines consist of two basic compo-

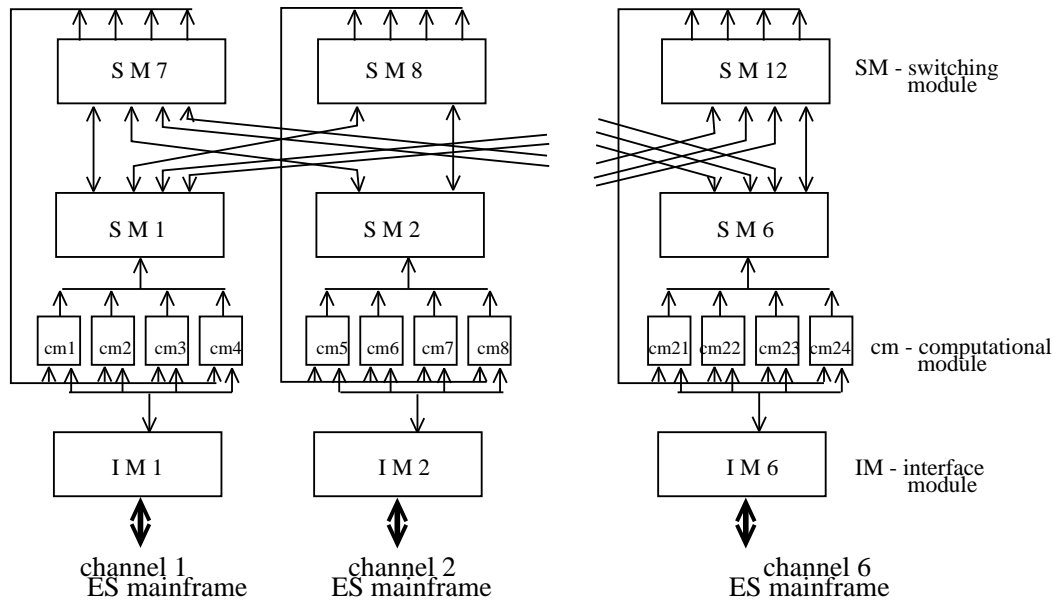


Figure 7-3 ES-2704 Structure
Source: [Pono87; Torg88]

nents: a set of computational modules, and a set of switching modules. The computational modules consist of a set of processors—administrative, execution, control, and three switching units, in the case of the ES-2704—together with shared memory. Automata exist in memory and are queued to the execution processor. If the execution processor becomes too heavily loaded, automata are queued to the switching processors that send them to other computational modules to be executed. The ES-2704 consisted of up to 24 computational modules, 12 switching modules, and six interface modules that provided access to external host computers, which provided access to external data stores. Several prototype models were built using the ECL IS-500 series of chips, Soviet analogs to Motorola's MECL 10K series [Torg88, 182]. With an 80 nsec clock period, the peak performance was 100 MIPS on 16-bit, fixed-point data [Pono87; Lomo91, 20]. The ES-2727 consists of 72 computational modules, 36 switching interface modules arranged in a

two-tiered hierarchy. The theoretical peak performance of this machine is said to be 750-1000 MIPS [Torg89].

In comparison to Western work on data-driven and demand-driven machines, the MDA have a greater “real-world” flavor. First, although the mathematical foundation for the MDA is not clear in the literature, it is not clear that it is as theoretically pure as that underlying many Western machines of this class. For example, it is not clear that the MDA are free of side-effects, a property which makes functional programming attractive. Second, unlike most Western work which uses either control flow, dataflow, or reduction as the computational model, the MDA appear to provide constructs that can be used to specify any of these modes. Third, the MDA apparently place significant emphasis on performance, perhaps compromising theoretical purity. Resources must be explicitly managed by a programmer. Automata can be assigned to run on specific physical processors or be stored in specified portions of memory. The systems also integrate uniprocessor and multiprocessor ideas. Each computational module can manage the execution of a number of automata concurrently, so it is possible to balance the cost of communications overhead with the benefit of distributed processing. Fourth, the MDA implementation emphasizes reliability. The switching network consists of switching processors that maintain “indicator sets” that record the status of the physical resources in the system. When processors or memory fail, automata can automatically be re-initiated on operating resources without the intervention of the user.

The MDA have had a long and difficult history. Torgashev persistently promoted his machine, trying to get resources to support it, for two decades. Although Glushkov was the lead author in the 1974 paper presented at IFIP '74 in Stockholm, he had little to do with the project. To build a broader base of support, Torgashev had invited him, a prominent Academician very influential in computing, to be a co-author together with V. A.

Myasnikov who for many years was the head of the Main Administration of Computer Technology and Management Systems at the State Committee on Science and Technology (GKNT) [Prav740508].

The project attracted the attention of Control Data Corporation, but inter-governmental difficulties prevented any joint projects.

The first mock-ups were completed by 1979, consisting of only four modules. It took several more years to overcome problems with the machine's control system and develop a complete version, however [Samo89]. An ES-2704 neared completion in 1987 [Marc87] and a number of operational units had been installed by 1989 [Ekon89; Przh89].

The ES-2704 prototype was constructed at the NITsEVT prototyping facility in Moscow using the same technology used to build the ES mainframes. Financing for this stage was orchestrated by NITsEVT and lasted through approximately 1986 or 1987. Since he reportedly followed NITsEVT construction guidelines, building it with only a half dozen different types of boards, the machine could be relatively easily accommodated by ES mainframe factories. In order to get the system into series production, however, Torgashev needed to find a critical mass of customers willing to pay for the system. Although some organizations expressed an interest, most users were dissatisfied with the machine. It used its own programming language, Ryad, which was unlike any used by existing HPC users. Furthermore, around 1989, the language had not even been fully implemented, and coding had to be done largely in an assembler-level language. In short, programming the machine was very difficult and discouraged potential users. Also, because of its reliance on the mainframe I/O channels and disks, data could not be moved in and out of the processor fast enough to support high processing rates. Complicating mat-

ters, as *perestroika* progressed, potential customers became less and less willing and/or able to spend money on unproven technology.

A colleague of Torgashev's at LIAP, M. B. Ignat'yev, continued work on a related project called System-3M after Torgashev moved to LIIA [Myas84; Igna86; Valk87]. He later proposed the creation of a highly parallel architecture for a personal supercomputer based on the recursive concepts [Igna89].

7.8 Macro-pipeline Processors

Around 1976, inspired by multiprocessor projects within the Soviet Union and abroad, a number of researchers at the Institute of Cybernetics of the Ukrainian Academy of Sciences in Kiev (IK AN UkSSR) including V. M. Glushkov, A. A. Letichevskiy, Yu. V. Kapitonova, and I. N. Molchanov worked on the mathematical, software, and hardware foundations of a multiprocessor "macro-pipelined" computer [Glus78]. In this approach, which in its rough form resembles coarse-grain systolic computation, the data for a given algorithm propagate across a network of computational components [Mikh86].

An important element of this approach is that the computational units executed by individual processors are large relative to the amount of data transmission between processors so that communications overhead is minimized [Leti85]. Engineers were particularly interested in improving the distribution of tasks through the machine. They wanted to be able to distribute not only arithmetic and logical computation, but also control.

Work on the construction of a machine was initiated in 1979. NITsEVT helped arranged for GKNT funding from 1980 through 1986 or 1987 [Prav850827]. The macro-

pipeline processor concepts were implemented during the early 1980s in a system called the ES-2701. The design allowed for variable-sized configurations shown in table 7-5.⁴

The ES-2701 consisted of computation processors, control processors, a switching network, and a systems monitor based on a minicomputer. The computation processors executed a subset of the ES mainframe instruction set on 32-, 64-, and 128-bit operands [Mikh88, 157]. The switching system was a two-level network. Each switching processor in the first level was linked to 16 other system entities. These could be computational or control processors, system monitors, or channels to the mainframe host. Switching processors of the second level were linked to all first-level switching processors.

In addition to the standard system startup and monitoring functions, the systems monitor had the ability to restructure the ES-2701 dynamically in the event of a hardware or software failure without disrupting the execution of the task [Mikh88, 157].

The ES-2701 processor design was influenced by previous work at the IK AN Uk-SSR. In particular, the control processors used a high-level interpreted programming language [Pogr87]. This feature had its roots in the MIR computer, developed during the 1960s. The MIR, oriented towards scientific computation, included a high-level interpreted language designed for easy coding of scientific equations [Rudi70; Moro75; Litv81]. Letichevskiy and Molchanov both worked on the software for this machine [Ikan90]. In the ES-2701, the use of a high level language reduced the amount of control information that had to be transmitted through the interconnect network and simplified dynamic allocation of parallel branches of the task [Pogr87; Mikh88, 153].

⁴The figures for tables such as these vary considerably. For example, [Vnesh89] lists processing rates for the three configurations as 25, 50, and 100 MIPS, while [Vnes89b] gives rates of 133, 266, and 533 MIPS. The amount of main memory distributed throughout the configurations is 2.4M, 4.8M, and 9.6M words according to [Molc85]; [Vnes89] reports 25, 25, 100 Mbytes; and [Vnes89b] states 96, 192, and 384 Mbytes. These figures reflect differences between the first prototype (1984) and the second (1986). The latter used the same component base, but vector operations were added to improve performance on vector operations.

	ES-2701.01	ES-2701	ES-2701.02
Number of processors			
computational	48	96	192
control	8	16	32
switching	8	16	32
System monitors	2	2	2
Main memory per processor	.5-1	.5-1	.5-2
Performance (MIPS)	133	266	533

Table 7-5 ES-2701 Parameters
Sources:[Molc85, 106; Vnes89; Vnes89b]

To enable programming of the system at various levels, developers created the programming language MAYaK (a Russian acronym for ‘Macro-pipeline Language’ which translates as ‘lighthouse’ or ‘beacon’). MAYaK is actually a family of languages with three parts: MPP (multi-modular programming) and two subsets, Prostor and YaDRO [Mikh86]. Other subsets, mentioned in [Goro84; Goro84b] apparently were not fully implemented. YaDRO is a low-level language for applications programming similar to standard programming languages. The primitives include process forking and joining and communication through shared memory. Prostor is used for assembling parallel modules into a parallel computation, implementing a given macro-pipelining scheme. MMP includes the other two languages and is often nearly synonymously with MAYaK.

This language has been categorized by one Western observer as a ‘coarse-grain composition language’ [Dong92b]. Not widely researched in the West, such languages are designed to provide mechanisms for allowing coarse-grain processes programmed in other languages into execute concurrently as one program. The benefit of MAYaK for tradi-

tional high-performance computing users is that existing FORTRAN code, for example, can be packaged with MAYaK headers and run as a parallel process on the macro-pipeline processors. None of the existing code needs to be changed. For example, in one case a 15,000-line FORTRAN written for an ES-1066 computer was converted to run on the ES-2701. The only changes to the code were the addition of MAYaK control statements. The program reportedly ran 33 times faster on a 41-processor system than on a single-processor ES-1066.

The first, 16-processor, ES-2701 prototype was completed 1984, in close cooperation with NITsEVT [Prav850827; Ikan90]. It was constructed using standard (but old) TTL ES mainframe components to speed the development time. The ES-2701 was designed as a special-processor attached to a mainframe host which would provide all interaction with peripheral storage. The host was an ES-1060 for the first prototype. A second prototype, completed in 1986, had an ES-1066 host and the complex was named the ES-1766. Such a configuration passed state testing in 1987 [Elec87; Kale87; Lari87; Przh89, 36]. This 48-processor unit reportedly had a performance of 133 MIPS [Przh89, 36; Vnes89b].

The prototypes were constructed at the VEM Plant in Penza [Ikan90]. Only three ES-2701 prototypes were ever constructed, and none in the full configuration.

Relationships with the factory were problematic. In spite of the support of NITsEVT and efforts to build the machine using standard ES equipment and components, the factory in Penza had little desire to build a machine designed in the Academy of Sciences, with unproven performance and a very small market. Although the machine was reasonably programmable because large portions of existing code could be used, the system was extremely constrained by the I/O bottleneck of the mainframe hosts which, in these configurations, limited to 2-3 Mbytes/sec. Whatever high performance could be achieved on

the multiprocessors was compromised by the mainframe and the peripherals. The time required to load a task was often a large percentage of the time needed to solve it.

Matters were further complicated when Minradioprom decided to move production of several machines, including the ES-2701 from Penza to Minsk, even though the production documentation for the latter had already been based on the equipment in Penza. The reasons for the move are not clear to us, although the most likely reason is that the ministry wanted to make the production facilities at Penza available for some other type of production. The end result was that the ES-2701 production documentation would have to be reworked, delaying matters considerably.

As the possibilities for direct negotiations with factories grew, engineers negotiated with others to try to get the system into production. Negotiations have not been successful, and no other systems along these lines were built.

Nevertheless, the machine continues to appear in articles. Design work on a successor consisting of up to 340 arithmetic processors and a peak performance of one GIPS, the ES-1710, is reported in [Przh89; Lomo91], but the prospects for such a machine are currently very poor.

7.9 Multiprocessor Computing Systems with Programmable Architecture (MCS PA)

The Scientific Research Institute of Multiprocessor Computer Systems (NIIMVS) is home to a large body of research on multiprocessor systems with programmable architecture. The Institute was founded in 1973 in the city of Taganrog on the coast of the Sea of Azov through the efforts of A. V. Kalyayev.

Kalyayev had been involved during the 1960s in the development of digital differential analyzers and around 1964 had built such a machine consisting of 100 parallel digital integrators. During the 1970s his work underwent a transition from integrators which

were designed to model analog machines digitally to computing systems with a more general-purpose orientation.

Kalyayev wanted to draw together into a single organizational framework a number of different scientific organizations which would support the research cycle from theory to production, and provide a training ground for students whose talent could be drawn on by the research institutes. NIIMVS is the research institute in a complex including an institution of higher education, a design bureau, a pilot production plant, and several associated laboratories in the industrial ministries [Mvs89]. This research complex was formed within the Ministry of Higher Education (MinVUZ), and is the leading example of high-performance computing in that ministry.

Strongly influenced by the work of Yevreinov and others described in section 3.3.2.1, work conducted at NIIMVS during the 1970s was labeled “homogeneous computational structures.” This was later changed to “multiprocessor computing systems with programmable architecture” to minimize confusion with other research on homogeneous computing systems carried out elsewhere in the country. An early implementation of the ideas described below was called the OVS-80, consisting of 16 general-purpose (non-NIIMVS) processors. It was begun in 1980 and the first units were completed in 1982.

7.9.1 General Characteristics

A computer’s performance depends on how well its architecture is suited to a particular algorithm (or how well the algorithm can be adapted to a given architecture). This is particularly true in the field of parallel systems [Hock88]. Ideally, a system with a programmable architecture can be adapted to fit the structure of data flows of a particular algorithm. The programmable architecture concepts permeate nearly everything done at NIIMVS, which has divisions devoted to architecture of multiprocessor systems with pro-

grammable architecture, signal processing, robotics, neuro-like computing structures, microelectronics, and others.

A second design philosophy in effect at NIIMVS is that the machines should reflect the “natural” language of the users, primarily engineers. The systems are characterized by high-level instructions (called “macro-operations”) which execute such functions as solving systems of linear equations, differential equations, fast Fourier transforms, calculation of eigenvalues of matrices, and many more. These are executed on “macro-processors.” Effort has been made to define a set of macro-operations which is on the one hand general enough to solve many numerical problems and on the other hand minimalist to keep the structure and tuning of macro-processors simple.

The ideological consistency of the many projects is noteworthy. The adherence to crossbar style interconnects, processors which execute complex operations, and the ability to tune the interconnects has remained nearly absolute for many years. This uniformity reflects A. V. Kalyayev’s strong control over the research agenda.

NIIMVS has developed a number of special- (such as for signal processing) and general-purpose systems which share at least the following features:

- A number of macro-processors which can operate asynchronously. The macro-processors consist of multiple “microprocessor sections,” which in turn consist of an elementary processor performing basic arithmetic and logical functions, a microswitch controlling links with other microprocessor sections, local memory, a switch memory unit storing switching patterns, and an elementary operations memory unit storing the computational instructions for individual macro-operations.
- A programmable cross-bar switching system. This switch provides point-to-point links between any two macroprocessors.

- A distributed, programmable memory system, enabling access of complex memory objects.
- An internal user-oriented high-level machine language based on a set of macro-operations and macro-switches.
- An internal machine language oriented towards the distribution of macro-operations and macro-switches in a parallel multiprocessor configuration.
- A macro-dataflow model of computation. Individual macro-operations execute when all operands are available, but the high-level organization of computation is more centralized than in true dataflow machines. A centralized control manages the allocation of macro-operations, but individual macro-processors have exclusive control over their execution.

To support systems development, one NIIMVS division works on components, building specialized processor, memory, and switch components for the programmable architecture machines. Products include the K1815 family of chips for general-purpose and digital signal processing, the SK 1509 KP1 16x16 single-chip switch, and many others [Niim90; Kaly88b; Kalo86; Bobk86; Niim89b]. Thanks to Kalyayev's personal contacts with individuals in the main administration for science in Minelektronprom, he was able to "push through" three chips—a macro-processor, a switch, and a memory unit—into series production during the early 1980s [Vadi84].

7.9.2 ES-2703

The ES-2703 is one general-purpose, high-performance MCS PA. It was one of the machines funded by the State Committee on Science and Technology in conjunction with NITsEVT. The 32-bit prototype system, shown in figure 7-4, was linked to an ES-1061 mainframe host. It incorporated sixteen macroprocessors, a switching system, a control

unit, an exchange processor. The operating system is resident on the control unit and manages the control unit and field of node processors (macroprocessors). Each task consists of a control module running on the control unit and a processing module running on a set of one or more node processors. All node processors participating in a given processing module execute the same program, but on different data. Multiple processing modules can execute simultaneously on the field of node processors. Consequently, node processors operate in uniprogram mode, while the control unit supports multitasking [Babe91].

Although direct connections with peripheral storage were designed for later machines, the ES-2703 prototype accessed all peripheral storage through the host (contrary to that shown in figure 7-4). The machine was designed primarily for gas-dynamic applications, which influenced the selection of the word-length, the macro-operations, and computational methods, and prompted designers to build a machine in which the processing elements could be grouped into arbitrary configurations [Babe89; Baba90]. Design began in 1981 and the technical statement of work was worked out in 1982. Between 1982 and 1985 the prototype was constructed. The machine passed state testing in 1986 [Mvs89]. Only two of the machines reportedly were built.

The ES-2703 macroprocessors were constructed using Soviet analogs to the AMD 2900 bit-sliced chips, called the 1804 series. These chips were used because during system design NIIMVS' own chips had not entered production and were not ready for use. Since the focus of this project was on systems architecture, it was felt that the development process could be simplified and speeded up by using traditional components, at least for the first version of the machine. Each macroprocessor had 64 or 256 Kbytes depending on the capacity of the memory chips [Babe89].

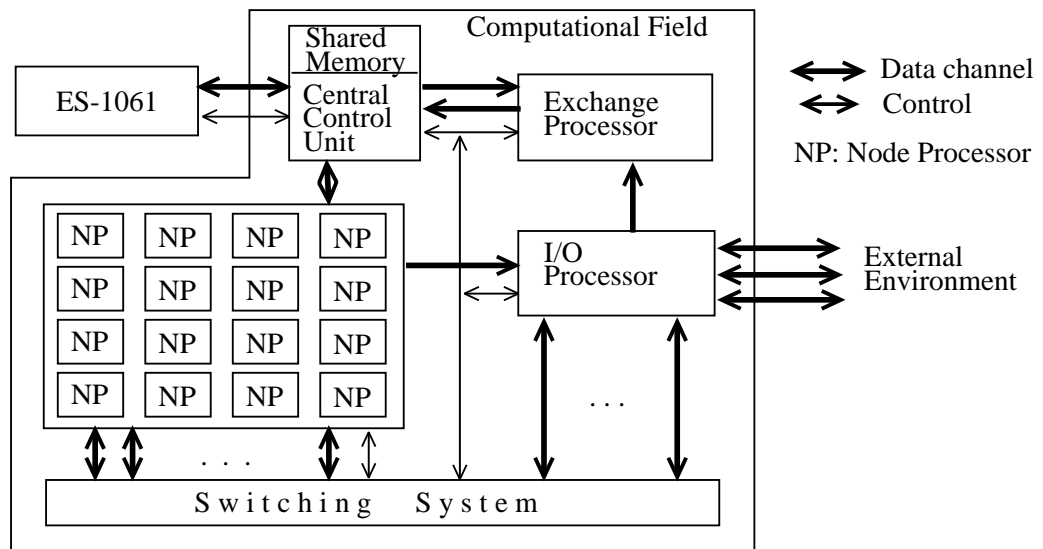


Figure 7-4 ES-2703 Structure

Source: [Kaly88, 162]

The published literature gives inconsistent information about system performance. Figures of 30-50 MIPS and 125+ MIPS are given in [Babe88; Kaly88; Lomo91]. Performance measurements reported in [Babe88] show a curious anomaly. Table 7-6 shows the speed-up in moving from one processor to four for a number of basic computations. These data seem to indicate that the system does not scale well even though the tasks are easily parallelizable.

7.9.3 ES-2703 Successors

Following completion of the ES-2703, design work underwent a bifurcation. The quest for high performance drove the design of a system with 128 processors and a theoretical peak performance of 1 Gflops on 64-bit data. The realities of the late 1980s—the difficulty of finding sponsors for large-scale work, the growing reluctance of industry to assimilate the production of unproven machines with weak markets—drove the develop-

Task	Speed-up (4 proc/1 proc)
Addition of two 12x12 matrices	1.26
Multiplication of two 12x12 matrices	2.80
Scalar product of two 132 element vectors	1.19

Table 7-6 Speed-up On ES-2703
Source: [Babe88, 8]

ment of small-scale macroprocessor systems which could be used as scientific co-processor attachments to a personal computer [Niim89; Niim89b].

7.9.4 Special-purpose Systems

Besides general-purpose systems, NIIMVS researchers have developed a series of special-purpose systems for real-time signal and image processing [Niim89c]. These include the PVK-460 which is built using the NIIMVS switching and macroprocessor chips manufactured by Soviet industry. The PVK-460 includes vector and scalar modules which can be combined in a variety of configurations, depending on user requirements. The machine has a word-length of 20 bits. Each vector module is equipped with 32 macro-operations (expandable to 512), and executes at 50-100 MIPS. Containing up to a total of 1024 processors and a theoretical peak performance of 460 MIPS, the PVK-460 was completed in 1988. A successor constructed during the early 1990s, the PVK-1600, has a theoretical peak performance of 1600 MIPS. A similar system has been built for medical image processing.

7.9.5 Research Trends

Two trends have characterized developments during the 1990s: an orientation towards smaller and/or specialized systems, and an orientation towards integrated systems. Both preserve the essential elements of the programmable architecture philosophy which has dominated NIIMVS research, and both reflect the changing nature NIIMVS' funding environment. The overall level of funding declined in real terms, and the portion of income from contract work increased.⁵

Although declining, funding for fundamental research at higher-education institutions from the state budget remained relatively good in comparison with other branches of Soviet science. NIIMVS was also able to win some proposal competitions held by the GKNT and the State Committee on Education. Such funding supported fundamental research in systems which integrate aspects of artificial intelligence, knowledge-bases, and neural-computing into a single, unified system [Niim90]. It also supported a continuation of research on architecture, integrated circuits, and neurocomputers.

The next generation of macro-processors have been called "super-transputers." The shift in terminology reflects public-relations efforts more than a fundamental shift in the research program. The features of distributed memory and hardware control of inter-processor communications arguably put the macro-processors in the same general class as transputers, and the higher-level orientation of the macro-processors leads to the "super" designation. Transputers are also being used as an alternative hardware platform for systems developers [Gig1930122, 3].

The deteriorating financial state of NIIMVS meant that other funding had to be found, through contract work in particular. Overall funding through military organizations has

⁵Reportedly, to 70% of the budget in 1990.

declined, but through 1990, at any rate, contracts for image and signal processing machines remained stable and in some cases reportedly increased. Under *khozraschet*, customers became increasingly interested in getting a real product for their money and accountability increased. Reportedly this improved the quality of work performed.

During the late 1980s and early 1990s it became harder to establish contact with Minelektronprom plants. Kalyayev lost important contacts within this ministry. Efforts to enlist factories in manufacturing a new generation of NIIMVS-designed chips have been unsuccessful. Such arrangements now have to be done on an economic basis, rather than administratively. NIIMVS does not have sufficient funds to pay the factory to develop new chips, and the factories are reluctant to support such work with their own funds.

NIIMVS also made changes to its organizational structure to improve its financial position. The Scientific Research Center for Super- and Neurocomputers (NITsSN) was created at NIIMVS in 1991 under the All-World Laboratory, an international organization headed by vice-president of the USSR Academy of Sciences Velikhov [Nits91]. The latter was established in 1989. There were no restrictions on the wages fund in centers affiliated with the All-World Laboratory, so workers could be paid more than was otherwise permitted. As a result, the majority of the contract work done at NIIMVS was carried out within the context of NITsSN. NITsSN therefore has a floating employee pool. Groups are formed on a temporary basis to work on specific contracts; when a task is complete, the group breaks up and is replaced by other groups. The charter of the All-World Laboratory also gave daughter centers considerable tax breaks, and the waiving of customs duties.

NIIMVS proper also experienced some minor reorganization. As of 1988 the individual divisions operated on *khozraschet* principles and gained the right to reorganize themselves as they saw fit, subject to the approval of the director. At least one division, de-

voted to neurocomputing, had reorganized itself on the basis of temporary, rather than permanent laboratories. Reportedly such changes were not encouraged by the institute's leadership, however, since there was a feeling that they threatened the cohesion of the work.

7.10 ES-2702

Like other ES-270x systems, the ES-2702 is a specialized processor attached to an ES mainframe host which provides monitoring facilities and access to peripheral storage, etc. While not a high-performance system, we mention it briefly for the sake of completeness. The ES-2702 is a machine designed for symbolic processing. Its input language is REFAL, a symbolic processing language with a long history of development in the Soviet Union [Fauc68; Turc86]. Work on a REFAL-oriented hardware platform began at the Institute of Applied Mathematics during the late 1970s under A. N. Myamlin [Myam86, 301]. Starting with the basic processor of the widely manufactured ES-1035 mainframe, the researchers implemented their own microcode to support symbolic processing operations. The resulting ES-2702 was connected via standard I/O channels to a general-purpose ES mainframe. Like most of the other ES-270x projects, the ES-2702 work was supported by NITsEVT [Myam86, 316-317].

7.11 ES-2705

The ES-2705 was an analog-digital multiprocessor developed at Riga Polytechnical Institute in Latvia. It was designed for solving field theory boundary problems which are used extensively in seismic prospecting, aero-/hydrodynamics, weather prediction and other applications [Spal90]. The machine was prototyped as a special-processor for an ES mainframe during the early 1980s, but more recently a version has been built which can be attached to a personal computer [Komp91]. Although data transmission is done digi-

tally, computation is performed in analog by representing equations as transformations on voltage levels within each processing element. Input voltages are introduced into the system and the solution is represented by the voltage states in the individual processing elements when the system state has stabilized.

7.12 Attached-Array Processors

During the 1970s and 1980s attached array processors (AAP) were developed at ES R&D facilities in the USSR, Bulgaria, and East Germany. These are specialized processors attached to general-purpose mainframes, often through selector channels. The host CPU accesses them using standard I/O commands.

Like the associated mainframes, these systems were strongly modeled after Western systems. Although they did not have very high performance rates when measured against Western developments in high-performance computing, they were one of the few HPC alternatives available to many Soviet users. Thanks to the long development times of other systems such as the El'brus, the use of the series-produced attached array processors became rather widespread, particularly in the oil and gas industries [Mair81; Ivan88]. Manufactured with the same component base and technology as their mainframe counterparts, they were assimilated into production with relatively little difficulty.

7.12.1 ES-2335

The ES-2335 passed state testing in 1979 [Niko79]. Designed for use with the ES-1035 mainframe, this system came equipped with a library of specialized routines which could be invoked through program calls from the mainframe host. These included routines for matrix and vector processing, solution of differential equations, signal processing routines such as fast Fourier transform, etc. [Niko81]. The ES-2335 had a performance of 10 MIPS (5 Mflops) on 32-bit data [Niko81; Niko82; Niko82c; Kezl86]. It was

constructed using the series production technology of ES mainframes. The unit was manufactured in Bulgaria [Niko82].

7.12.2 ES-2345

The ES-2345 was the first completely Soviet attached array processor. Developed at the Scientific Research Institute of Mathematical Machines (YERNIIMM) in Yerevan, Armenia, this unit was introduced in 1978 for use with the ES-1045 and passed state testing 1979 [Meli79; Seme82]. The ES-1045 was also developed at YERNIIMM. Both systems were manufactured at the Kazan' Computer Plant. The ES-2345 uses pipelined computation in addition and multiplication blocks to achieve a performance of approximately 30 MIPS (6.4 Mflops) on 32-bit data [Kuch81, 178; Seme84; Kuch85]. Hundreds of ES-2335 and ES-2345 AAP were used in the Soviet Union.

7.12.3 MAMO 1-M

Developed at approximately the same time as the ES-23x5 systems, the MAMO 1-M (Matrix Module) is an attached array processor designed by the East German Robotron computer manufacturer for use with the ES-1055 and ES-1057 mainframes [Merk80]. The ES-1055M was an upgraded version of the ES-1055, often equipped with a MAMO processor [Grue81; Muen81]. Unlike the Russian attached array processors, MAMO is connected to the host computer not via I/O channels, but with direct connections to memory. The host computer interacts with the AAP through a specialized set of instructions rather than through I/O instructions [Przh89, 36]. The MAMO was designed with a set of array processing instructions which could be invoked directly from within the user program on the host computer. In other words, when an array processing instruction is encountered, an interrupt to MAMO is invoked. On the Soviet machines, the AAP is invoked to process a library routine, i.e., a pre-programmed set of instructions. The MAMO

therefore functions well when a program involves the execution of isolated array processing instructions, but has high overhead when the equivalent of a library subroutine of densely packed array operations is used. The MAMO has a performance of 5-10 Mflops [Rajm88; Robo88]. It was also used in a number of Soviet installations for seismic processing and other applications [Robo88; Pois90, 2].

7.12.4 ES-2700

During the mid-1980s YERNIIMM introduced a successor to the ES-2345, the ES-2700 [Seme85; Musa86]. Unlike the ES-2345 and ES-2335, this system could be used with a variety of ES models, including the ES-1045 and ES-1068 [Musa86; Tass88]. It has a performance of 100 MIPS (30 Mflops) on 32-bit data [Seme88]. Although production was rather easily assimilated at the Kazan' Computer Plant, few units reportedly were built because the market for them was weak. Many users apparently preferred the ES-2706 (described below) because it was cheaper, more compact, and more reliable than the ES-2700.

7.12.5 ES-2706

Development of the ES-2706 began in 1980 at the Central Institute for Computers and Computer Technology in Sophia, Bulgaria. It entered series production in 1984 at the Computing Machinery Works in Sophia. By 1990, approximately 400 units had been manufactured of which reportedly 90% had been exported to the Soviet Union for use in seismic exploration [Prat90]. More than 50 installations reportedly contain more than one processor [Maly91; Tcha92]. It is a functional duplicate of Floating Point Systems' AP-190L and can run the latter's software [Coop88, 2; Prat90]. The ES-2706 is an improvement over the AP-190L. It removes the AP-190L's page restriction, increased memory size, and has re-designed connectors [Prat90]. Like the ES-2700 it can be used on a

variety of ES mainframes. Its performance was less than the ES-2700. The peak performance is 60 MIPS (12 Mflops) on 38-bit data [Mark86; Bere87]. Unlike the ES-2335 and ES-2345, it can be attached directly to external storage using a dedicated I/O processor which speeds up access considerably [Sagd87].

Unlike the ES-2335 and ES-2345, the ES-2706 was built using Schottky TTL technology, most of it imported, off-the-shelf technology from the West [Prat90]. For this reason, it was reportedly more reliable than its Soviet counterparts.

During the early 1990s, Bulgarians developed a number of successors to the ES-2706. The ES-2706M and ES-2709 both have a peak performance of 18 Mflops due to a second buffer in data memory and a second adder [Prat90]. We do not know of any shipments of these systems to the Soviet Union.

7.12.6 Loosely-coupled Array Processor Systems

The attached array processors provided an alternative path for Soviet users to high performance in many application domains. Several organizations worked on incorporating multiple attached array processors in a configuration with one or more mainframe hosts. Multiple user jobs could be distributed dynamically among the attached array processors, or different parts of the same job could run concurrently on them. In the West, E. Clementi, working for IBM, demonstrated that this approach could be used effectively on applications with coarse-grain parallelism [Bern84; Clem84]. The Soviet researchers did not duplicate his approach exactly, but were familiar with his work and even communicated with him [Anan87; Coop88].

A number of these systems, called the IZOT 1703E, were built by the Bulgarians, using the ES-2706 together with the ES-1037 host [Rabd88]. A system with four ES-2706s

was installed at the Institute of Space Research in Moscow in 1986. It was later upgraded to 10 ES-2706s [Sagd87].

A similar approach was used by Soviets in creating systems based on the ES-1066 or ES-1068 mainframes. The ES-1066.20 and ES-1066.60 are configurations manufactured at the Minsk Production Association of Computer Technology with four and eight ES-2706 processors, respectively [Mpov91, 33]. The ES-1068.17 consists of an ES-1068 mainframe with two ES-2700 and eight ES-2706 attached array processors [Mche88b; Tass88]. The first ES-1068.17 configuration was tested in 1988 [Przh89, 37]. Such a system was installed at the Computer Center of the Siberian Department of the USSR Academy of Sciences in Novosibirsk (VTsSOANSSSR) where it reportedly ran on some geophysics applications at 96 Mflops [Przh89, 37].

Researchers at the VTsSOANSSSR under N. N. Mirenkov developed significant portions of the systems software for both the Bulgarian and ES-1068.17/ES-1066.x0 which enables such loosely-coupled configurations to function. They have also built a large hierarchical, heterogeneous multiprocessor called the Sibir' to support the execution of so-called assembly-line parallel programming systems in which parallel programs are developed that can be dynamically and efficiently adjusted to the available resources of the multiprocessor. Typical applications include image processing, seismic data analysis, solution of systems of differential equations using difference methods, etc. The Sibir', completed in 1988, consists of three ES-1066 mainframes, eight ES-2706s, a STARAN-like associative processor, and a PS-2000. A programming system called Inya has been developed to integrate the software of all the components [Maly91].

7.13 Special-Purpose High-Performance Computers

Today's supercomputers, often costing millions of dollars, are cost effective because they are general-purpose, able to solve problems in a wide range of application domains.

It is possible, however, to design machines that cost significantly less, yet out-perform conventional supercomputers on a very narrow set of problems [Poll90]. Since 1989, Soviet researchers have published a handful of designs of descriptions of application-specific processors and multiprocessors. The most fully developed is a processor built at the Landau Institute of Theoretical Physics that is tailored to the Monte Carlo method for the Ising model with random links on a lattice containing 256x256 spins [Tala90]. This machine, a prototype of which was completed in December, 1989 [Tala90b], is capable of executing 4×10^6 elementary Monte Carlo steps per second. Upgraded versions of this machine, using a Western component base, are reportedly under development in the West.

Vyzhikovskiy and Kanevskiy of the Kiev Polytechnic Institute have published designs for a systolic specialized processor for solving systems of linear equations using the Gaussian method [Vyzh90]. Specialists at Moscow State University have designed a specialized processor for magneto-hydrodynamics problems capable of running at one Gflops [Bakh89]. Although the article makes it clear that such a machine has not yet been built, it presents an analysis of the requirements, showing that it could be built using currently available CMOS components. Donskov et al. at the Institute of High-Energy Physics in Protvino, describe a special-purpose processor for particle sampling by momentum in experiments studying central hadron collisions [Dons90].