MULTI-MICROCOMPUTER ROBOT CONTROL SYSTEM USING A SPECIAL ROBOT-ARITHMETIC-PROCESSOR FOR ADVANCED APPLICATIONS OF ROBOT MANIPULATORS

K. Rathgeber and M. Schnell

Section Control Systems Theory, Institute of Control Engineering, Technical University of Darmstadt, Schlossgraben 1, D-6100 Darmstadt, Federal Republic of Germany

Abstract. The performance of complex and advanced applications of industrial robots requires large computational power. The control system presented in this paper is designed as a multi-microcomputer system. The main purpose of this system is to create a pleasant environment for present and further studies in robotic research.
The Advanced Robot Control System (ARC) uses multiple microcomputers based on iAPX 8086 processors. A local bus of each processor allows additional memory and input/output expansions for increased throughput. Nevertheless, the rising calculation effort for the kinematic and dynamic relations shows the insufficient computing power of standard microprocessors.
Therefore a fast floating-point processor has been developed. This Robot Arithmetic Processor (RAP) is built up with bipolar bit-slice components, special arithmetic modules and hardware control mechanisms for on- and off-line diagnostic functions.

Keywords. Special purpose computers; Microprocessors; Hierarchical Systems; Multiprocessing Systems; Microprogramming; Diagnostic on Chip; Computer Architecture; Robots; Closed Loop Systems; Real-Time Control; Cartesian Robot Control.

INTRODUCTION

The performance of complex and advanced applications for robot-manipulators, e.g. handling or assembly tasks with external sensor-feedback, requires large amount of computing power. The control-strategies for these applications are time-critical tasks. A multi-microcomputer system is very suitable for robotics control because it can provide the computing power required. The single components of such a control-system can fullfil different time-critical tasks. A real-time system can be designed by coordinating the single subsystems.
There are two opposite requirements in constructing such a multi-microcomputer control-system.

- The primary goal is to provide the computing power and speed for real-time calculations of cartesian and joint-based control loops available.
- On the other hand there should be a pleasant user environment for programming and coordinating the several processors of the control system. The system should offer the feasibility for users to create and test software and to analyse the results of the implemented control-strategies on the multiprocessor system or on a host-machine.

The system we designed is called ARC (Advanced Robot Control). ARC is a multi-microcomputer system with one master-unit and several slave processors.
Within this system we have implemented a special purpose processor, the Robot-Arithmetic-Processor (RAP) to meet the computational requirements of present and future robotic control applications. The development of the Robot-Arithmetic-Processor aims at very fast processing of kinematic and dynamic transformations for robot-manipulators. The algorithms implemented on RAP are kinematic independent to provide the control of different n-degrees-of-freedom manipulators.
There are two main goals of this paper:

- to discuss the architecture of the ARC-System including the Robot-Arithmetic-Processor and
- to present some principles for advanced applications with simultaneous position and force-control running on the Advanced-Robot-Control-System.

MULTIMICROPROCESSOR CONTROL ARCHITECTURE

The required performance for a robot control system can be obtained with a multi-microcomputer system. Figure 1 presents the system structure using 4 identical single board computers based on Intel iAPX 8086 processors. The 8086 boards run at 8 MHz and each board includes 64 Kbytes of ROM, 16 Kbytes local RAM, 16 Kbyte dualport RAM, interrupt controller and one serial and parallel interface. A 8087 floating-point co-processor can be added to enhance the performance of the system. The global multiprocessor bus - an AMS-M-Bus, compatible with Multibus I - connects all processors and the global memory for fast interprocessor communication. The dualport memory permits together with the local onbord memory a high degree of parallel processing. The local 16-bit private bus systems are used for I/O-modules, intelligent controllers, local memory expansions and for fast interfaces to local processor-subsystems. The system is expandible with additional processor boards at the global bus, but a limit results from the maximum transfer rate on the bus (10 Mbytes/second).
To create a pleasant and functional software environment, it is useful to have one master processor avoiding system overhead on the other processors, for example synchronization and multi-

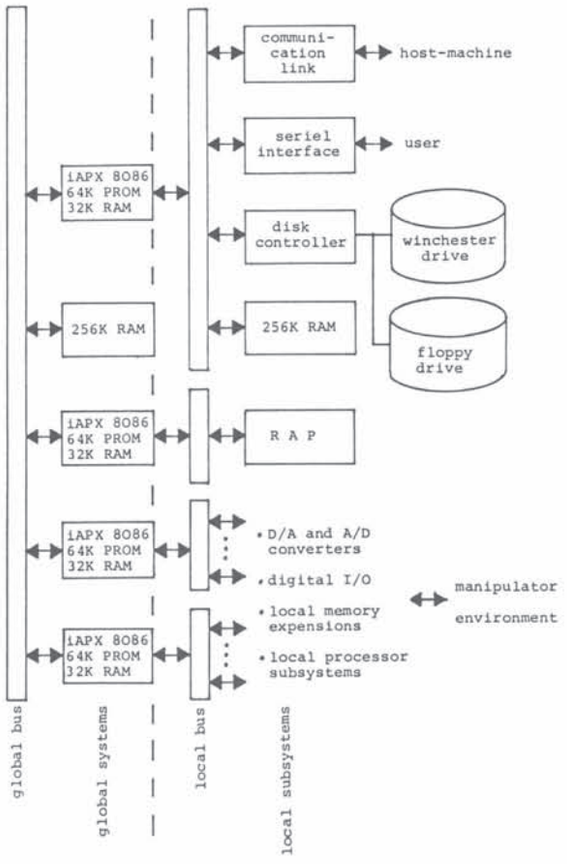

Fig. 1 Multi-microcomputer architecture

processor-supervision. These functions together with the high level tasks are implemented on the master unit using an Intel RMX multitasking operating system managing a winchester- and a floppy-disk, terminals and a communication link to a host-computer, at the moment a LSI 11/34 machine. The connection to the robotarm and the environment is realized with interface-components attached at the local bus system of two slave-processors. According to the requirements, the interfaces can be flexibly adapted to different robot tasks.
One processor of the system is qualified for arithmetic intensive tasks using a special processor described in the next paragraph.

RAP - ROBOT ARITHMETIC PROCESSOR

It is necessary for the real-time implementation of control tasks to compare different hardware-solutions with respect to the required computation performance for robot tasks. The most common realization would be an application with standard 16- or 32-bit microcomputers with hardware-arithmetic-support. The great advantage of these systems is the quantity of available hardware- and software-components and a good development support. Nevertheless, the arithmetic computation performance is today not high enough for robotic algorithms, although it is possible to use mathematic co-processors for the floating-point arithmetic (e.g. time for one 32-bit floating-point multiplication with Intel 80287 ≈ 11,9 µs, Motorala 68881 ≈ 5,5 µs, National 32081 ≈ 4,8 µs). Looking at the computation-speed, an application with signal-processors could provide good results, but they have not yet on-chip hardware support for 32-bit floating-point arithmetic. Transputers would have the required performance, but they are at the moment too expensive and difficult in handling. Nowadays, the realization is only possible with a mini-computer (too expensive) or with special hardware designed for a specific problem.
We decided us for a micro-programmed solution, because this leads to a flexible system structure. Modifications are not complicated, so it is very practicable for different test implementations. We designed an independent microprocessor subsystem using bipolar bit-slice components and special floating-point building-blocks.
Figure 2 shows the block diagram of the Robot Arithmetic Processor, RAP.

Floating-Point Path

The main modules of RAP are two VLSI-circuits with a one-bus building-block architecture, a floating-point ALU and multiplier /1/. They can handle both single (32-bit) and double (64-bit) precision formats and operations, as well as full 32-bit two's complement integers. The floating-point operations and data format is conform to the requirements of the IEEE standard 754, version 10.0. Basic instructions - add, subtract, multiply, divide, conversion to and from 32-bit two's complement integers, absolut value, compare - are executed onchip. Transcendental functions are calculated by a microprogram with a table lookup for fast processing. The devices can represent infinity, NaN (Not a Number), denormalized and zero operands and are capable to operate with all rounding modes. The treatment of exceptions, such as divide-by-zero, overflow,

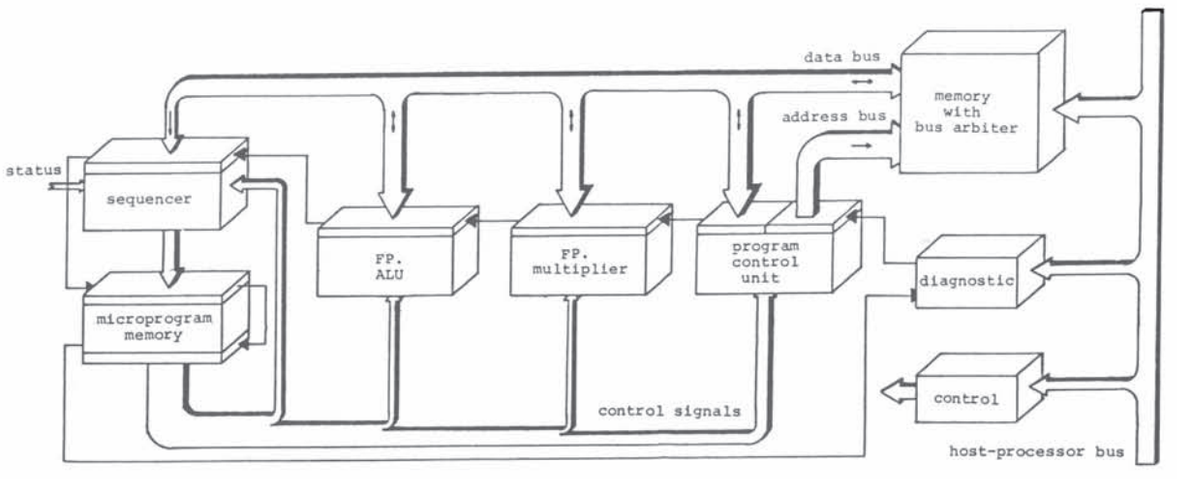

Fig. 2 Robot Arithmetic Processor

underflow, invalid and inexact operations are fully implemented. A very important feature for real-time control applications is a fast mode of operation which removes the time penalty of underflow exception handling by substitution of zero for denormalized numbers.
As the units use dedicated circuit arrays to perform the required functions, the arithmetic processing is faster than designs which rely solely on sequential, clocked logic. For example, the array flowthrough time is under 180 ns for single precision floating-point multiplication and addition. Data input and output transfer may occur at a rate of one transfer per clock cycle. One or both operands can be stored internally which allows repeated operations with a constant or multiple functions performed on two operands. Although the circuits had been designed for optimal flowthrough time, they have also the capability of pipeline-operations, because the next operands can be loaded into the input-registers during operation in the floating point array. Together with the possibility of parallel computation in the ALU and in the multiplier, vector- and matrix-arithmetic can be implemented very efficiently.

Control path
A control unit is required to build an independent processor-subsystem which can supply the arithmetic-chips with data at a high transfer rate utilizing the high performance of the floating point units. We have realized an universal micro-programmed control-unit to get a great flexibility for the implementation of a wide range of algorithms for robot control systems.
A micro-programmed machine uses a coherent sequence of micro-instructions to execute the machine instructions. Without any overlapping instructions, the throughput of the system would be to small. To improve the performance of the control unit, a pipeline technique is used with three stages of pipeline registers. We use the most efficient structure, an address-instruction-data based microprogram control architecture /2/, although this technique is the most difficult solution for the software development. The three required steps (determine microprogram address, fetch and execute micro-instruction) for a complete microcycle are overlapped by this technique which results in a shorter microcycle. This pipeline architecture offers significant speed improvements except in the case of conditional jumps where the pipeline must be cleared before execution.
In order to access instructions and data in an orderly manner, a Program Control Unit (PCU) is used to provide the most efficient mechanism for program control. The PCU is built up with four 4-bit ALU-slices. This 16-bit ALU has the capability of using its internal registers as the program counter, stack pointer and internal temporary registers for address calculations avoiding the inefficient directly produced memory addresses from the microword. This of course provides considerable flexibility in the architecture and also allows a much greater repertoire of instructions to be executed.
The 8086 host processor has direct access to the memory of RAP in a 8 Kbyte page addressing mode, because the address range of RAP is 32×2^{16} bit = 256 Kbyte which is a quarter of the address space of the 8086 processor. This is not a restriction of the data transfer rate, since it is not necessary to move more than 8 Kbyte of data in real-time.
RAP can interrupt the host processor for communication and synchronization and - in the other direction - the host processor can set a signal in the status register of RAP.
RAP is attached to the local bus of the 8086 - a 16-bit SMP-M bus - via one interface board which can be replaced to use RAP in connection with other bussystems, e.g. Multibus I or II, VME bus, Q-bus.

Diagnostic Path
We have implemented hardware control mechanisms in RAP to utilize the following functions:
- on- and off-line diagnostics
- microprogram download
- debugging during software-development

Diagnostic is used to verify the functionality of a component or of the whole system and to identify the location which failed. A processor like RAP is difficult to test comprehensively because the combinatorial logic is very complex and includes sequential elements. Due to the complexity the logic should be partitioned into several smaller logic blocks. The problem of observation and control of sequential logic is difficult, because the output is a function of both the inputs and the state stored in a register. The design of RAP avoids the lack of access to the state registers by breaking the feedback path in the sequential logic block. The state can be controlled by injecting test data and output can be observed by reading the registers. Now the system is easy to test because the sequential system is converted into a pseudo-combinatorial logic.
A very feasible realization is the Diagnostic-On-Chip (DOC) technique, which works without a considerable increasing hardware effort /3/. DOC uses a buried shift register, called the shadow register, to load and unload diagnostic data. It allows diagnostic testing in the background - transparent to the normal system operation - and on-line diagnostics. The system needs not to be taken into off-line mode during diagnostic which is beneficial for maintenance purpose. For proper work all registers in the control path, e. g. pipeline and instruction register, must have these diagnostic capability. All DOC-elements are connected by a serial ring-line for minimal expenditure of hardware, controlled by the 8086 host-processor (see Figure 2).
Another advantage of this technique is the capability of using RAM- instead of PROM-elements for the microprogram memory. This feature is called Writeable Control Store (WCS) and increases the flexibility of RAP with regard to the development of different robot control algorithms. The working method of RAP can be adapted according to the requirements of the algorithms. Additionally the basic functions can be changed during operation by loading new parts of microcode.
The hardware of RAP together with the host-processor allows an easy software development without necessity of an expensive bit-slice processor development system. The program development utilities are:
- Assembler
  We use a relocatable macro meta assembler running on an IBM-PC which differs from an ordinary microprocessor assembler in that way, that the user can define the instruction set and hardware configurations. This development step can be divided further in defining and assembling of microprogram, generation of machine instructuion addresses out of the microprogram, assembly of machine program and formatting of micro- and machine program.
- Download
  The DOC-technique together with the Writabe Control Store allows downloading of the formatted microprogram, whereas machine-code can be loaded directly into the memory of RAP.
- Test and Debug
  The user can run the programs and debug them

with additional hardware facilities of RAP and with a software control monitor implemented on the 8086 host processor. The off-line test is supported by start/stop and single-step functions. Hardware breakpoints are also useful for the real-time test of the system.

Using these utilities, a wide range of algorithms for robot tasks can be implemented.

Computation Performance

The resulting computation performance is listed in Table 1 for 32- and 64-bit floating-point operations. Together with the number of mathematic operations of the control algorithms it is possible to estimate the attainable sampling periods of the control loops. The table shows two different calculation methods, a pipelined operation and an operation with memory-to-memory transfer which corresponds to the maximal execution time without using the parallel datapaths and pipeline facilities. Further reduction of the execution time can be obtained through parallel use of the two floating-point units. This method is very suitable for vector- and matrix-operations. An example of a vector-vector multiplication is described in the appendix. The computation performance is between 3 and 5,5 MFLOP (330 - 180 ns per operation) depending on the vector dimensions. If the whole algorithm is implemented on the microprogram level, the result would be the shortest sampling period, but the size of the microprogram is limited to addressable memory.

TABLE 1 RAP: EXECUTION TIMES, SINGLE FUNCTION

operation		computation of memory-to-memory transfer	pipelined operation
multiply	32 bit	720 ns	450 ns
	64 bit	1170 ns	900 ns
add substract compare	32 bit	720 ns	450 ns
	64 bit	990 ns	720 ns
divide	32 bit	2250 ns	1980 ns
	64 bit	4230 ns	3960 ns

For real-time applications it must be taken into account that data transfer between RAP and the host processor requires in proportion to the RAP arithmetic a large time. Nevertheless, the ratio between arithmetic and transfer operations of RAP is less than the ratio of standard microprocessors. Therefore the estimation of the computation time cannot be restricted to the number of mathematic operations. The worst-case execution time - memory-to-memory transfer - is a first evaluation if parts of the algorithms are microprogrammed and data transfer is avoided as far as possible.

APPLICATIONS

Over the last few years several applications in robotics have been investigated. The achievement of robot-assembly and handling tasks requires not only the use of additional sensor-systems, called external sensors in oppsite to the internal sensors like joint-position sensors, but also complex control-strategies and algorithms. Examples for external sensor systems are wrist-mounted force/torque-sensors or proximity-sensors. The control strategies, like simultaneous motion and force control in contour-finding and -tracking tasks or the performance of the peg-in-hole problem require large amounts of computationl power to realize good performance of the applications.

Control Hierarchy

The performance of robot tasks can be stressed out in a hierarchical control structure, which is presented in figure 3. The robot tasks can be devided into three levels:

- Configuration level
  The specification of the desired trajectories in the configuration space is based on the application dependent formulation of the robot task.
- Cartesian control level
  Depending on the desired trajectories and the specified sensor-configuration the cartesian control algorithms and the cartesian sensordata (e.g. force/torque signals) are processed on this level.
- Joint control level
  The joint torques are calculated as functions of the desired trajectories in joint coordinate space and the sensor feedback from joint-based sensor systems like joint-position sensors.

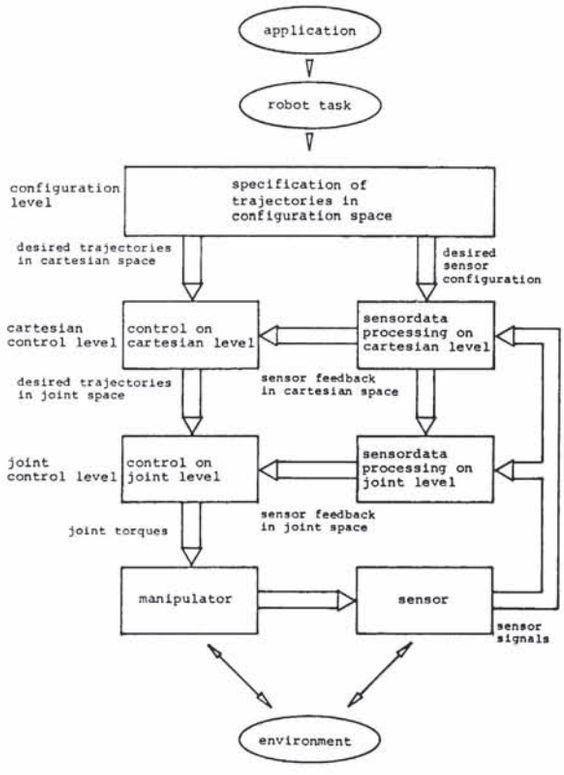

Fig. 3 Control hierarchy

General Requirements

From this hierarchical point of view we have worked out some general requirements concerning robot applications:

- Servo rates
  The realization of a high task-performance requires short servo cycles on cartesian level. In spite of the complexity of the control schemes there are two posibilities to meet the requirements: the use of a control system with very large computation power or the split of the system's servo rate. Most control strategies allow the splitting of the servo rate into two levels. At the high level the calculation of the control algorithms and the servo equa-

tions takes place. The lower level computes the configuration dependent parameters at a slower servo rate.
- Sensor data processing
  The utilization of external, in most cases complex sensor-systems necessitates additional computional power on the control system.
- Storage capability
  In order to evaluate and compare control structures in a quantitative way the system must provide the capability to store large amounts of data in real-time.
- Evaluation of experimental results
  The fast evaluation of test runs is enabled by storage and display facilities on the control system or on a host machine.
- Universal algorithms
  The design of a robot control system as a testbed for robotics applications requires the implementation of configuration independent universal algorithms to make quite sure that the control strategies are well guided for adaptation to different tasks, changing sensor configurations and distict kinematics of robot manipulators.
- Programming language
  The program development process is supported by using appropriate programming languages: Highlevel languages, like PL/M or FORTRAN on task configuration level and machine dependent languages (assembler- and microprogramming) on control and sevo level.

Referring to these requirements the control system must provide the following facilities and attributes:
- Modular system architecture
  The robot control system must be divided into modular subsystems to extract the parallelisms in the control algorithms. Several time critical tasks can be processed on such a modular system architecture.
- Large computational power
  The computation of the control loop equations and the sensor data processing in real-time tasks requires a multiprocessor system to provide the large amounts of computing power.
- Large amount of memory
  The system must have the facility to store large amounts of data in real-time. For later analysis data such as joint positions or sensor signals (e.g. measured contact forces) must be stored in every servo cycle.
- Intelligent subsystems
  The complex sensor data preprocessing is achieved by intelligent subsystems. These subsystems calculate the sensory feedback for the control algorithms. This concept reduces data transfer to other processors of the control system.

ARC-System for Robot Control

We have taken these requirements into account when designing the Advanced Robot Control System as a testbed for our further studies in robotics research. The distribution of the computational burden on the system's processor units is shown in figure 4.
- The master processor with the multitasking operating-system processes all high-level tasks and manages the global state memory.
- The time-critical numerically-intensive tasks - in particular kinematic and dynamic calculations - are executed by one 8086-processor with attached RAP. Data transfer to other processors reduces to a minimal value.
- The third processor uses two 8088-processor subsystems connected via dual-port memory at the local bus. The flexibility and the performance increases through sensor data preprocessing and local gripper control.

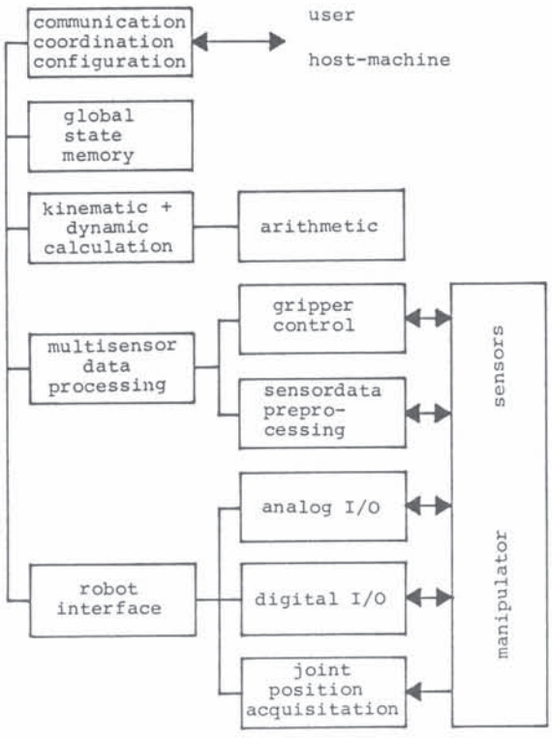

Fig. 4 Robot Control System

- The servo loops of the manipulator are connected via local interfaces to the fourth 8086-processor using analog and binary I/O-modules and a joint position acquisitation system. This processor executes all manipulator specific tasks (e.g. robot supervision, joint control) except the kinematic and dynamic calculations.

REAL-TIME IMPLEMENTATION - AN EXAMPLE

The ARC-System is designed as a testbed for real-time experiments in manipulator position and sensor control. As an example for implementation of robot control schemes the cartesian closed loop control with prescribed dynamics /4/ is discussed.
The control strategy is based on the idea to maintain a prescribed motion along desired cartesian trajectories and to get well defined responses in each direction of the task coordinate system. The manipulator being controlled with this strategy is a MANUTEC r3 with a 6-link-arm.
The real-time control algorithms must achieve the following steps:
- Measurement and preprocessing of the joint position and of the signals from a wrist mounted force/torque sensor.
- Computation of the end-effector position (forward solution of the kinematic problem), calculation of the cartesian velocity and acceleration based on the determination of the Jacobian matrix and of the time derivative of the Jacobian.
- Calculation of the desired joint velocity based on the cartesian control loop.
- Computation of the inverse dynamics of the system and calculation of the compensation-torque.

The computational burden of the control scheme is spread across the processor units of the ARC-system. The number of mathematic operations can be used to estimate the computation performance re-

quired in real-time control of the MANUTEC r3 arm.
The total number of operations for the calculation of the kinematic transformations, the inverse dynamics and the cartesian controller is approximately 3500 floating-point operations. The maximum servo-rate of the system with prescribed dynamics would be about 4 msec. To reach even a higher performance within this system, the sampling period is divided into two different cycles:

- Control level
  On the high-rate control level the system realizes the cartesian controller, computes the desired joint velocity and calculates the compensation torque resulting from the inverse dynamics of the manipulator.
- Parameter level
  On the parameter level, which works on lower rate, the control system refreshes the vectors and matrices of the inverse system.

Taking the splitted servo rates into account the system reaches a control level cycle rate of 2,5 msec and a parameter level rate of 5 msec. These cycle times are quiet feasable both for tasks with high speed requirements and for sensor-guided handling and assembly tasks in robot applications. The implementation of the proposed control scheme on the Advanced Robot Control System is under progress.

CONCLUSION

This paper has presented the structure of the Advanced Robot Control System (ARC). Within this system we have integrated a special purpose Robot Arithmetic Processor (RAP). The ARC-System is designed as a testbed for studies in robotic research.
The architecture of the control system meets the requirements arising from hierarchical control schemes of advanced robot applications. The computation performance of the system is sufficient for real-time control of industrial robots, using universal and kinematic independent algorithms. The control system offers the feasibility to create and test control schemes and to analyse the results on a host-machine. Applications like the cartesian closed loop control with prescribed dynamics can be achieved in real-time.
A later version of RAP with reduced hardware effort could be a processor for industrial control systems. This RAP has to be programmed only with the specific robot parameters. The kinematic and dynamic relations would be computed for a n-degrees-of-freedom manipulator in a 'black-box' system.

REFERENCES

/1/ B.Sackett, G.White (1985).
Solution For a High Performance Floating Point Co-processor Design.
Application Note Weitek Corporation, 15 March 1985

/2/ J.Mick, J.Brick (1980).
Bit-Slice Microprocessor Design.
McGraw-Hill, 1980

/3/ J.Birkner, V.Coli, F.Lee.
Shadow Register Architecture Simplies Digital Diagnosis.
Monolithic Memories Application Note AN-123

/4/ E.Ersü, K.Rathgeber, M.Schnell, W.Neddermeyer (1985)
A Robot Arithmetic Processor Concept for Cartesian Closed-loop Control with Prescibed Dynamics.
Proc. of the 1st IFAC Symposium on Robot Control, November, 6-8, 1985, Barcelona

APPENDIX

An example of a vector-vector multiplication in a 32-bit floating-point format illustrates the working method of the Robot Arithmetic Processor (see figure 5).

$$d = d_n = \underline{a} \cdot \underline{b} = \sum_{i=1}^{n} a_i \cdot b_i$$

with $d_i = d_{i-1} + c_i$, $d_0 = 0$, $c_i = a_i \cdot b_i$

For fast processing, the microprogram uses the pipeline facilities and parallel computation in the floating-point ALU and multiplier. For example, three parts are executed during step 10:

- load vector-element b_4 from the main memory into one input register of the multiplier (memory addressed through a register of the Program Control Unit; multiplication will be carried out in step 14 and 15)
- multiply a_3 and b_3, first step
- add c_1 and c_2, second step

The status signals of the floating-point-chips can be tested by the sequencer for exeption-handling.
For a multiplication of 2 vectors with the dimension n, 4n+4 steps are required (1 step = 90 ns). Computation performance P of a computer is expressed through the number of floating-point operations per seconds (FLOP). The number of operations is 2n-1 for this example , the performance of RAP is:

$$P(n) = \frac{2n-1}{4n-3} \cdot \frac{100}{9} \text{ MFLOP}$$

An estimation is possible using the two boundaries:

$$P_{max} = P_{(n \to \infty)} = 5{,}5 \text{ MFLOP}, \quad P_{min} = P_{(2)} = 3 \text{ MFLOP}$$

The result for a matrix-vector multiplication (matrix-dimension n x m) is:

$$P_{(n,m)} = \frac{2nm - m}{4nm - 2m + 5} \cdot \frac{100}{9} \text{ MFLOP}$$

$$P_{max} = 5{,}5 \text{ MFLOP}, \quad P_{min} = 3{,}9 \text{ MFLOP}$$

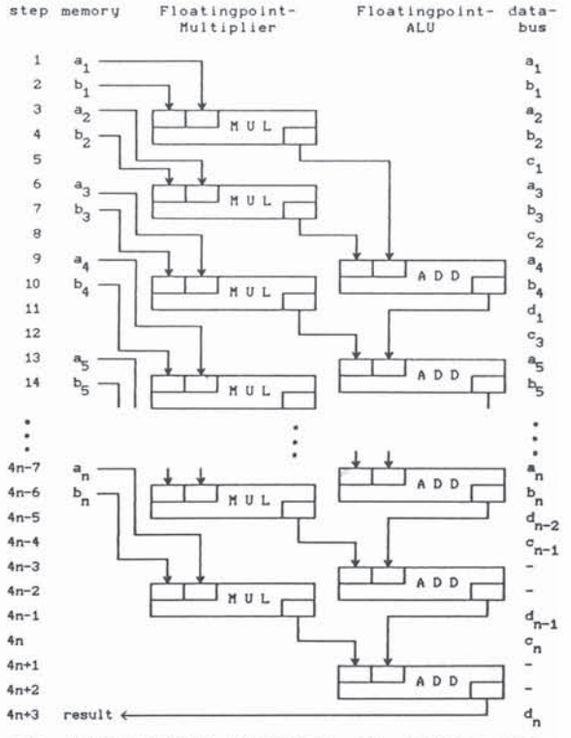

Fig. 5 Computation Structure of vector-vector multiplication