
WRTP'88


# PREPRINTS

## 15th IFAC/IFIP Workshop on Real-Time Programming

Edited by

Juan A. de la Puente
Alfons Crespo

**25-27 May1988**
**Valencia, Spain**

UNIVERSIDAD POLITECNICA DE VALENCIA

# A REAL-TIME SYSTEM FOR ROBOT MANIPULATOR INVERSE DYNAMICS COMPUTATION

J. A. Tenreiro Machado, J. L. Martins de Carvalho and J. A. Silva Matos

Faculdade de Engenharia da Universidade do Porto, Dep. Eng. Electrotecnica e de Computadores, 4099 Porto Codex, Portugal

Antonio M. C. Costa

INESC-Instituto de Engenharia e Sistemas de Computadores, Rua Jose Falcao, 4000 Porto, Portugal

Abstract. A new robot manipulator inverse dynamics computational algorithm is announced. The novel feature resides in the computations which are a blend of ordinary and Boolean algebra. As such, this method may also be interpreted as a dedicated compiler that optimizes the on-line computing time at expenses of the off-line stage. Nevertheless, the high off-line requirements are alleviated, through the derivation of some general rules that stem from the structure of the robot manipulator equations. Moreover, the on-line computing time is further optimized through the use of Binary Decision Diagrams. The results show a considerable computational improvement on a conventional sequential machine. Furthermore, they clearly point out new computational parallel architectures, without scheduling problems, and where performance improvement is proportional to the number of processors. Finally, it is observed that the proposed algorithm is not restricted to robot inverse dynamic computations, but is also applicable to many other real-time computing structures.

Keywords. Computational methods; Boolean algebra; computer architecture; parallel processing; robots; compilers.

## INTRODUCTION

In the last two decades robot manipulator inverse dynamics, that is, the task of computing the required joint torques for a desired set of positions, velocities and accelerations, has been a topic of research and development. In the period since then, the area of computational aspects of robot manipulator dynamic modelling has achieved high results, namely in the numerical recursive methods, using either Newton-Euler (Luh, Walker and Paul, 1980) or Lagrangian (Hollerbach, 1980) algorithms; some improvement over these purely numerical procedures was achieved by Horak (1984) which developed a mixed algorithm, having both numerical and symbolic computation. More recently, a further step towards computational improvement was attained with the the automatic generation of the dynamic symbolic formulae, using LISP-based computer algebra systems (Leu and Hemati, 1986; Koplic and Leu, 1986). Computation through these schemes achieves better performances than the previous ones; moreover, it allows physical insight into the manipulator dynamics and provides strategies for the manipulator design, that, as pointed out by Yang and Tzeng (1986), can reduce the calculation burden. More recently still, Neuman and Murray (1987a, 1987b, 1987c) developed the concept of customized computational robot dynamics, showing considerable performance improvement over the previous methods.

One may think of these last methods as an "investment" in the off-line computational time so that the on-line calculation time is abbreviated. The present tradeoff between off-line vs. on-line computing time can be pushed further if we bear in mind the following considerations:

-Both the numerical recursive and symbolic algorithms are converted to the computer internal code through a high level language compiler like Fortran or C.

-The resulting object code imposes a complex burden, as far as a microprocessor is concerned, due to the high number of arithmetic and transcendental floating point calculations.

-The floating point calculations correspond to a large number of microprocessor's machine code instructions.

-The floating point calculations must be performed with high precision in order to reduce problems associated with finite precision arithmetic.

-Floating point calculations with high precision require a large word length.

Clearly, this type of implementation is far from satisfactory. But further reasons can be put forward:

-The computer internal numerical representation has a much higher accuracy than the manipulator hardware (A/D, D/A, etc.), usually with only 8 to 16 bit precision.

-If computations could be performed with a precision of the same magnitude as the ones used in the manipulator hardware, calculation time would decrease.

-The use of 8 to 16 bit computing accuracy without finite precision problems implies that the arithmetic operations, as well as

the transcendental functions, can not be executed in the ordinary way.
An alternative method must be simple enough for any microprocessor to perform, i.e. it should be well adapted to the microprocessor's machine code instruction set.

To conclude, we observe that there is a need for a "special" algebra that provides a better management of the existing hardware/software resources. By other words, we need to find a new compiler that generates a more efficient object code in the manipulator hardware/software environment. Such compiler may be called a dedicated compiler, contrasting with general purpose compilers such as Fortran or C. Boolean algebra satisfies all the above requirements; nevertheless, we are now faced with the problem of translating the ordinary arithmetic and transcendental formulae to Boolean algebra. Although formally independent, there is a way of doing it. The robot manipulator inverse dynamic algorithms correspond to multivariable functions, and, as such, can be tabulated. If both input and output variables are quantified and converted to a suitable binary code, then the resulting table may be viewed as a truth table where one can use standard Boolean function-simplifying techniques, having as input variables the bits of the quantified binary coded <position + velocity + acceleration> vector, and output Boolean functions the bits of the quantified binary coded required torques. Unfortunately, upon further examination this ideal situation is not feasible in practice as it poses critical obstacles to its computation due to the necessity of a huge Boolean table. Nevertheless, it points to a methodology for a realizable implementation that, with modifications, can achieve a remarkable reduction of the on-line computing time. This realizable implementation is developed in the next section.

## A HYBRID COMPUTATIONAL ALGORITHM

Since it has been found that the full Boolean-based computation is impractical, there is the necessity of a compromise between realizability and computational improvement. Such compromise may be achieved through the use of a hybrid computation, having both ordinary arithmetic sums and Boolean algebra.

The manipulator dynamic equations are of the form

$$T = J(q)\ddot{q} + C(q, \dot{q}) + G(q) \qquad (1)$$

where $J(q)$ is the n x n inertial matrix, $C(q,\dot{q})$ is the n dimentional Coriolis/centripetal vector, $G(q)$ is the n dimentional gravitational vector and $q$, $\dot{q}$ and $\ddot{q}$ are the n dimentional vectors of link positions, velocities and accelerations, respectively.

A natural way of achieving our purpose is to split each link torque in its terms. Then each term can be calculated by Boolean algebra, and the final result, i.e. the link torque, by an ordinary arithmetic sum of these terms. In this case, for each joint torque there are several truth tables, requiring a maximum of

[n(position) + 1(acceleration)]
i.w.v. for the inertial terms

[n(position) + 2(velocities]
i.w.v. for the Coriolis terms

[n(position) + 1(velocity)]
i.w.v. for the centripetal terms

n(position)
i.w.v. for the gravitational terms

where input word variable (i.w.v.) means the appropriated binary-coded representation of each input variable. The Boolean computation becomes alleviated as we have a maximum of (n+2) input word variables, contrasting with the requirement of 3n input word variables for the fully Boolean-based computation.

One of the more serious restrictions imposed by the coexistence of two different algebras, is that arithmetic summation degrades the overall precision, hence resulting in the need for m extra bits, given by

$$m = int[\log_2(p+1)] \qquad (2)$$
p=total number of terms

in each term in order to achieve the desired accuracy; nevertheless, this problem is of minor influence since m increases much more slowly than p. Therefore, for a desired output resolution of r bits we must have a term accuracy of

$$W_{INITIAL} = r + m \qquad (3)$$

bits.

In conclusion, we may say that the off-line "compilation" (i.e. truth table simplification) requirements, either in computing time or in computer memory space, are considerably alleviated, as we pass from one huge Boolean table with 3n input word variables, to several truth tables which have a much smaller number of input word variables, and this without altering significantly the on-line computing time.

## IMPLEMENTATION OF THE NEW COMPUTATIONAL ALGORITHM

To illustrate the implementation of our algorithm, we consider a 2R (Fig. 1) robot manipulator described by the following dynamic equations (Paul, 1980; Brady and others, 1982)

$$j_{11} = (m_1+m_2)r_1^2 + m_2r_2^2 + 2r_1r_2m_2C_2 + J_1 \qquad (4a)$$
$$j_{12} = j_{21} = m_2r_2^2 + r_1r_2m_2C_2 \qquad (4b)$$
$$j_{22} = m_2r_2^2 + J_2 \qquad (4c)$$
$$c_{122} = -r_1r_2m_2S_2\dot{q}_2^2 \qquad (4d)$$
$$c_{112} = -2r_1r_2m_2S_2\dot{q}_1\dot{q}_2 \qquad (4e)$$
$$c_{211} = m_2r_1r_2S_2\dot{q}_1^2 \qquad (4f)$$
$$g_1 = m_1gr_1C_1 + m_2g(r_1C_1 + r_2C_{12}) \qquad (4g)$$
$$g_2 = m_2gr_2C_{12} \qquad (4h)$$

with $C_i=\cos(q_i)$ and $S_i=\sin(q_i)$. In this paper we use data similar to other sudies (Young, 1978, Morgan and Ozguner, 1985, Machado and Carvalho, 1988), namelly:

$$m_1=0.5\ Kg\ ;\ m_2=6.25\ Kg\ ;\ r_1=1\ m\ ;\ r_2=0.8\ m$$
$$J_1=5\ Kgm\ ;\ J_2=5\ Kgm \qquad (5)$$

and assume the amplitude of each link variable within the following ranges (i=1,2)

$$-\pi \text{ rad} \leq q_i \leq \pi \text{ rad} \quad (6a)$$

$$-1 \text{ rad/s} \leq \dot{q}_i \leq 1 \text{ rad/s} \quad (6b)$$

$$-1 \text{ rad/s}^2 \leq \ddot{q}_i \leq 1 \text{ rad/s}^2 \quad (6c)$$

For these ranges and for the payload referred in (5) we have

$$-151.5 \text{ Nm} \leq T_1 \leq 152.5 \text{ Nm} \quad (7a)$$

$$-69.1 \text{ Nm} \leq T_2 \leq 69.1 \text{ Nm} \quad (7b)$$

Nevertheless, motivated by the natural assumption that the torque computation shall be a block of a larger control structure, the feedback loop may demand higher torques, and therefore we assume

$$-200 \text{ Nm} \leq T_1 \leq 200 \text{ Nm} \quad (8a)$$

$$-100 \text{ Nm} \leq T_2 \leq 100 \text{ Nm} \quad (8b)$$

From these considerations it results that equations (6) and (8) give the quantization ranges for the input and output ($\delta T_i$ and $\delta T_2$) word variables, respectively.

Now, for the multivariable function (4), we may find the appropriated numerical table and the corresponding bynary truth table. Finaly, from this table the Boolean based code can be generated. Nevertheless, if some general rules are previously implemented, we may alleviate the aforementioned procedure. These rules stem from considerations like (Machado and others, 1987, 1988):

a) The quantization ranges (8) must be the same for all terms of each equation. As usual, amplitude range of each term only covers part of the quantization interval, and the remainder becomes "unused". If the term amplitude fits within a new quantifying amplitude range $\delta T_{NEW}$ related to the initial $\delta T_{INITIAL}$ one by

$$\delta T_{NEW}=\delta T_{INITIAL}/2^K \quad (9)$$

then, for that term, we may drop out the K most significant bits of each input word variable.

b) The accuracy compensating extra number of bits m, as represented in equation (3), varies in discrete steps, and much more slowly than p. This means that situations may arise where m permits a higher fictitious number of summing terms $p_{FICTITIOUS}$ (i.e. $p_{FICTITIOUS}$=2 or 4 or 14 or ...) then the actual one (i.e. p). We may take advantage of this property, using these extra $p_{FICTITIOUS}$-p-1 possible sums by subdividing some sum terms and, therefore, enabling the use of rule a).

c) Bearing the physical (mathematical) robot manipulator requirements (specifications) in mind, it would appear that the Gray code is the more suitable and this, indeed, was confirmed by experimentation.

d) For even arithmetic functions of a single input variable, it was observed that the most significant bit of the corresponding input word variable could be dropped out if the Gray code was used.

e) On terms which have several input variables, the required final precision implies a similar accuracy on each input word variable.

The application of these rules to $T_2$ for the 2R robot manipulator inverse dynamic hybrid algorithm gives the chart depicted in Fig. 2. From these charts the source code (i.e. the truth tables) can be built and the object code (i.e. the Boolean based code) generated. Boolean code can be further optimized, through the use of Binary Decision Diagrams (Matos, 1983; Matos and Oldfield, 1983); with this method Boolean functions are computed using IF_THEN_ELSE structures. For v (single bit) input variables, we need a maximum of v evaluations, resulting a much more efficient code, namely for complex Boolean formulae. Figure 3 shows a histogram of the required computing time for the term $T_{34}=r_1 r_2 m_2 S_2 \dot{q}_1^2$, comparing the conventional arithmetic method with the new algorithm. Both codes were written in Turbo Pascal V 4.0, and running on a 8086 8 MHz machine, under MSDOS V3.2. Notice the remarkable improvement achieved with the new algorithm.

## PARALLEL COMPUTATION

We have shown that the proposed hybrid computational algorithm for robot manipulator inverse dynamics, offers considerable performance improvement over purely arithmetic alternatives, even in a monoprocessor sequential computing environement. In this section we will show that the algorithm also leads naturally to simple parallel architectures allowing unlimited speed-up in the on-line computations, without accuracy restrictions.

The operations involved in the on-line computations required by our algorithm allow simple distribution of the computational load amongst several processors. Indeed, this can be achieved without any of the complex scheduling problems, common to arithmetic manipulator inverse dynamics parallel computing structures (Luh and Lin, 1982; Nigam and Lee, 1985; Liu and Chen, 1986; Watanabe and others, 1986). Parallel architectures like the ones shown in Fig. 4, are a natural consequence of the calculation decoupling allowed by our algorithm. Part a) shows a parallel structure in which a number (k) of sequential processors can be used. The result of the off-line computation, i.e. the object code, is stored in the processor memories in a way that assures optimum computational load distribution. Part b) shows an alternative parallel pipelined solution.

The improvement in the on-line computing time is proportional to the number of processors, and it is not subject to any theoretical limit. The importance of this fact must be emphasized since the other manipulator inverse dynamics parallel computing structures achieved a limited improvement, in the sense that the resulting speed-up is <u>not</u> proportional to the number of processors, and, is in fact, <u>restricted</u> to a maximum of n (a condition that is achieved only if some errors are allowed (Binder and Herzog, 1986)).

Finally, it should be noted that the Boolean formulae are bit oriented instead of word oriented. Consequently, general

purpose microprocessors with 8, 16 or 32 data buses, and large instructions sets (unused in this algorithm), that require several clock cycles, are of little use in improving the overall computing performance. Much more promising seems to be the use of single bit, reduced instruction set and special purpose microprocessors. The design of a dedicated processor based on Binary Decision Diagrams, is being investigated.

## DISCUSSION

In the previous sections the new hybrid algorithm was developed and implemented for the dynamic equations of the 2R robot manipulator. This strategy has been used by other investigators, due to the fact that for many aspects of robot system research, two or three d.o.f. manipulator models are suficient, as long as they include all the configuration dependent inertial, Coriolis/centripetal and gravitational torques that may appear in manipulators with more d.o.f..

At this point one should note, that the new method is not restricted to the present case study. In fact, the first step towards this algorithm, is the generation of a binary coded numerical table corresponding to the system behaviour. Clearly, this procedure is not confined to the tabulation of the robot dynamic equations, but is also applicable to many other system descriptions. Moreover, the table generation may be obtained from experimental data instead of being model based. This is of utmost importance, since situations appear where mathematical models are inaccurate or difficult to derive. Therefore, our algorithm is well suited for the computation of a large set of physical phenomenae, the only restriction being the size of the binary tables, which as shown, depend solely on the required precision and the number of input variables.

## CONCLUSION

A new computational algorithm for robot manipulator inverse dynamics was presented. This algorithm is very efficient because it takes full advantage of both hardware and software capabilities of the robot manipulator system. Another important consequence of the proposed calculation method is the natural appearance of simple, yet powerful, parallel computing structures. Finally, it is observed that the dedicated compiler philosophy is not restricted to robot manipulator computations (i.e. kinematic, dynamic or control), but can be successfully generalized to many other computing structures, as long as we can redefine the management of the corresponding "environmental resources". This may lead to optimization procedures, having implications on either sequential or parallel computing system structures.

## REFERENCES


Binder, E.E. and Herzog, J.H. (1986), Distributed computer architecture and fast parallel algorithms in real-time robot control, IEEE Trans. Syst., Man, Cybern., 16, 543-549.

Brady, M., Hollerbach, J.M., Johnson, T.L., Lozano-Perez, T. and Mason, M.T. (1982), Robot Motion: Planning and Control, MIT Press, Cambridge. Chap. 1, pp. 1-50.

Hollerbach, J. M. (1980), A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity, IEEE Tra. Syst., Man, Cybern., 10, 730-736.

Horak, D.T. (1984), A simplified modeling and computational scheme for manipulator dynamics, ASME J. Dynamic Syst. Meas. Contr., 106, 350-353.

Leu, M.C. and Hemati, N. (1986), Automated symbolic derivation of dynamic equations of motion for robotic manipulators, ASME J. Dyn. Syst. Meas. Contr., 108, 172-179.

Liu, C.-H and Chen, Y.-M (1986), Multi-microprocessor-based cartesian-space control techniques for a mechanical manipulator, IEEE J. Robotics and Automation, 2, 110-115.

Luh, J.Y.S., Walker, M.W., Paul, R.P. (1980), On-line computational scheme for mechanical manipulators, ASME J. Dynamic Syst., Meas., Contr., 102, 69-76,.

Luh, J.Y.S. and Lin, C.S. (1982), Scheduling of parallel computation for a computer-controlled mechanical manipulator, IEEE Trans. Syst., Man, Cybern., 12, 214-234.

Koplic, J. and Leu, M.C. (1986), Computer generation of robot dynamic equations and the related issues, J. Robotic Systems, 3, 301-319.

Machado, J.A.T., Costa, A.M.C., Carvalho, J.L.M, (1987), Robot manipulator dynamics-towards better computational algorithms, INESC Internal Report.

Machado, J.A.T. and Carvalho, J.L.M. (1988), A smooth variable structure control algorithm for robot manipulators, IEE Control'88, Oxford, UK.

Machado, J.A.T., Carvalho, J.L.M, Costa, A.M.C., Matos, J.S. (1988), Dedicated computer system for robot manipulators IFAC 3rd Int. Symposium on Systems Analysis and Simulation, Berlin, GDR.

Matos, J.S. (1983), The binary decision diagram: a tool for logic design and implementation, PhD Dissertation, Syracuse University, Syracuse, N. Y.

Matos, J.S. and Oldfield, J.V. (1983), Binary decision diagrams: from abstract representations to physical implementations, 20th IEEE/ACM Design Automation Conference, Florida, USA.

Morgan, R.G. and Ozguner, U., (1985), A decentralized variable structure control algorithm for robotic manipulators, IEEE J. Robotics and Automation, 1, 57-65.

Nigam, R. and Lee, C.S.G. (1985), A multiprocessor-based controller for the control of mechanical manipulators, IEEE J. Robotics and Automation, 1, 173-182.

Paul, R. P. (1981), Robot manipulators: mathematics, programming and control, MIT Press, Cambridge. Chap. 6, pp. 157-195.

Yang, D.C. and Tzeng S.W. (1986), Simplification and linearization of manipulator dynamics by the design of inertia distribution, J. Robotics Research, 5, 120-128.

Young, K.-K (1978), Controller design for

a manipulator using theory of variable structure systems, IEEE Trans. Syst., Man, Cybern., 8, 101-109.

Watanabe, T., Kametani, M., Kawata, K. and Tetsuya, K. (1988), Improvement in the computing time of robot manipulators using a multimicroprocessor, ASME J. Dynamic Syst., Meas., Contr., 108, 190-197.

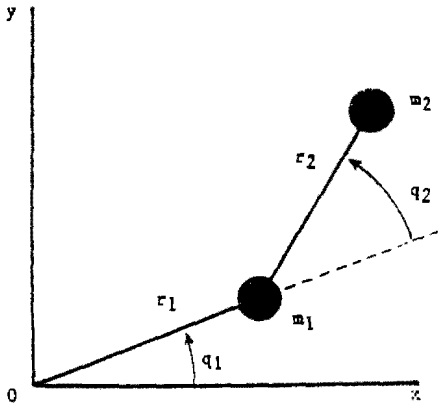



Fig. 1. The 2R robot manipulator.

| r=8 | r=8 , m=2 => $W_{INITIAL}$=10 bits | | | | PRECISION |
|---|---|---|---|---|---|
| $T_{ik}$ | $T_{21}^{m}=T_{22}^{m}$ | $T_{23}$ | $T_{24}$ | $T_{25}^{m}=T_{26}^{m}$ | TERM |
| 8 | 6 | 7 | 6 | 8 | $W_{FINAL}$ |
| 40 | 11 | 7 | 11 | 16 | TOTAL NUMBER OF INPUT BITS |
| 0.76 $10^{11}$ | 1474 | 92 | 1638 | 64226 | USED STATES |
| 0.34 $10^{11}$ | 574 | 36 | 410 | 1310 | UNUSED STATES |
| 100 | 6.25 | 12.5 | 6.25 | 25 | QUANTIZATION AMPLITUDE RANGE |
| 69.1 | 4.5 | 9 | 9 | 24.5 | TERM AMPLITUDE RANGE |

Fig. 2. $T_{ik}$ hybrid method source code chart, after using the relevant rules applicable for each summing term. $T_{ik}=(T_{21}^{m}+T_{22}^{m})+T_{23}+T_{24}+(T_{25}^{m}+T_{26}^{m})$
$T_{21}^{m}=T_{22}^{m}=(4+5C_2)\ddot{q}_1/2$; $T_{23}=5\ddot{q}_2$; $T_{24}=5S_2\dot{q}_1^2$; $T_{25}^{m}=T_{26}^{m}=49C_{12}/2$

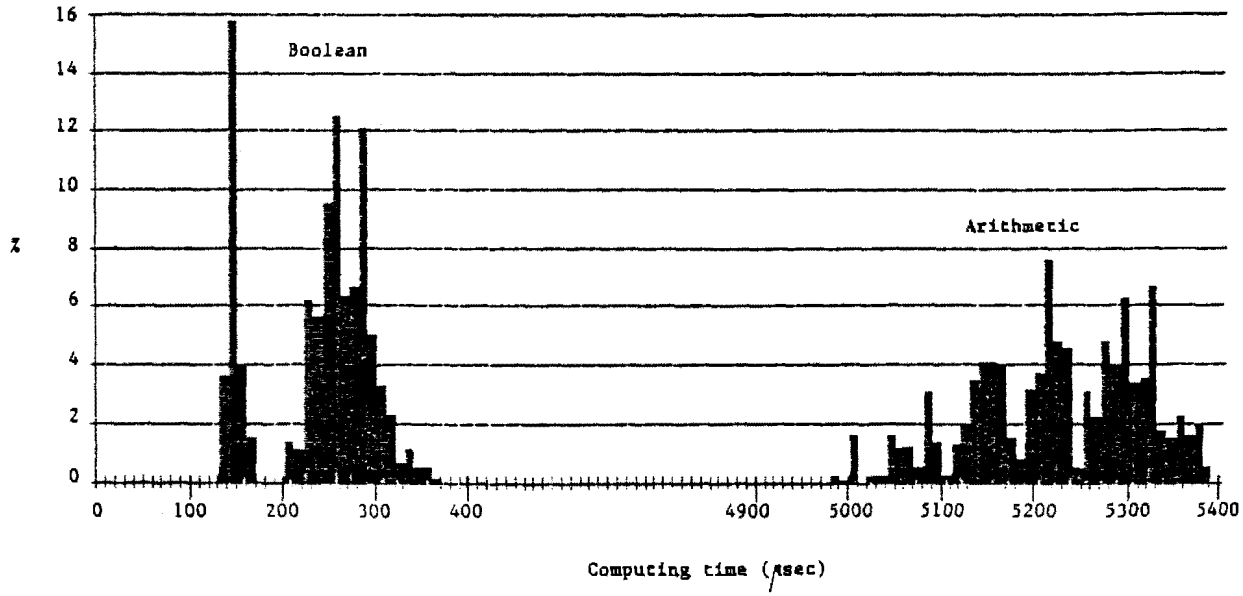
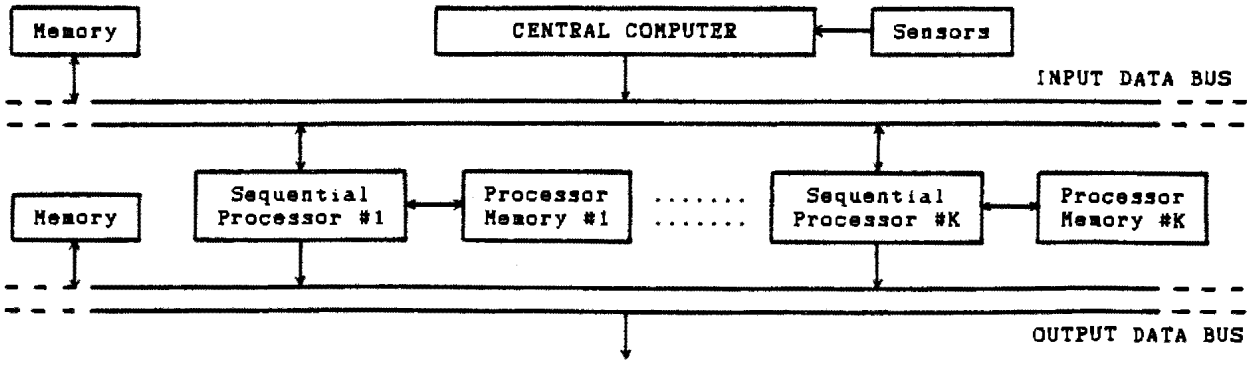
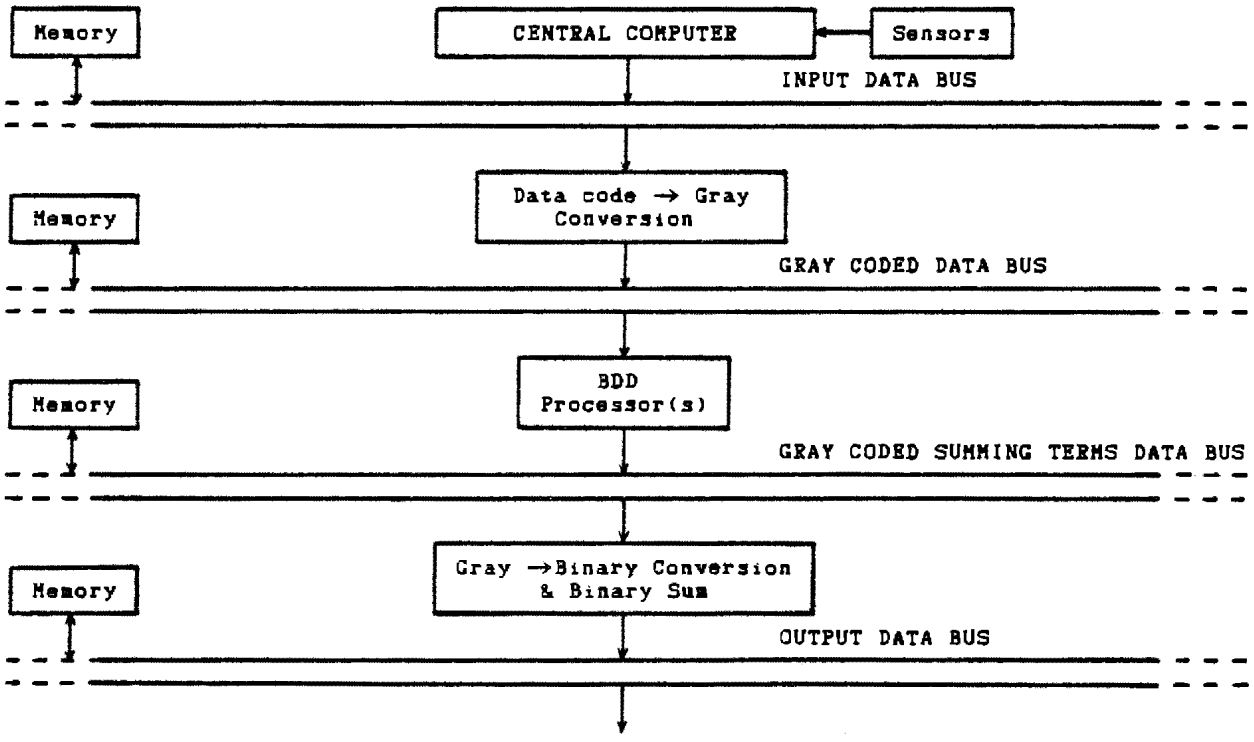



Fig. 3. Histogram of the computational time required by the term $T_{24}=r_1r_2m_2\dot{q}_1^2$.




a)




b)

Fig. 4. Possible parallel computer structures, suggested by the new hybrid algorithm.
a) Sequential/Parallel processor.
b) Pipeline/Parallel processor.