High-Performance VLSI Processor for Robot Inverse Dynamics Computation

Somchai Kittichaikoonkit, Michitaka Kameyama and Tatsuo Higuchi

Department of Electronic Engineering
Tohoku University
Sendai, Japan 980

Abstract

This paper proposes a VLSI-oriented matrix multiply-addition processor (MMP) for minimum-delay-time inverse dynamics computation on a linear array structure. We show that the delay time of the inverse dynamics computation becomes minimum based on the concept of the "odd-even alternative computation". The MMP architecture is systematically designed by the layout evaluation of the odd-even alternative computation. It is demonstrated by the layout evaluation that the MMP can be easily implemented in a single chip using the current VLSI technology. The performance with regard to the delay time is the highest in the architectures reported until now.

1 Introduction

In robot dynamic control, inverse dynamics, namely the computation of the joint torques required to produce given joint motions of the robot arm, has until recently been a computational time bottleneck. A great amount of computation for these torques must be completed in a servo loop time. Moreover, to achieve desirable motions of the robot arm, the servo loop time must be as small as possible. Thus, development of high-performance VLSI processors computing the inverse dynamics with not only high throughput rates but also small delay time is very important [1],[2].

The inverse dynamics can be regularly computed based on linear recursive equations of 4x4 matrix multiply-additions. The linear array of n matrix multiply-addition VLSI processors (MMPs), is well suited to the inverse dynamics computation because of its computational time lower bound is O(n), where n is the number of degrees-of-freedom of the manipulator. The recursive Lagrange algorithm is therefore used in this paper. The recursive Lagrange formulation can be expressed as the following recursive equations for joint i (i = 1, ..., n) [3]:

\[ T_i = T_{i-1} A_i B_i \]
\[ T_i = (T_{i-1} + T_{i-1} Q_{1i}) A_i B_i \]
\[ W_i = H_i T_i + A_{i+1} B_{i+1} W_{i+1} \]
\[ \tau_i = \text{Trace}(T_{i-1} DA_i B_i W_i) - g T_{i-1} DA_i B_i w_i. \]

The set of equations for the ith joint torque can be decomposed into 19 tasks as shown in Table 1, where each task could be considered as a single 4x4 matrix multiply-addition. The natural means of performing the inverse dynamics computation is to use a MMP, each of which assigned to a manipulator joint to evaluate the corresponding joint torques. The recursive nature of the above formulation suggests that the communications between the adjacent MMPs are sufficient to perform the parallel processing. Thus, the linear array of the MMP is well suited to the inverse dynamics computation. Fig.1 shows a task graph for the inverse dynamics computation. The computation could be attributed to the serial sequence of the linear recursive equations as defined in Fig.2. Thus, it is important to minimize the delay time of the linear recursive equation.

3 Odd-Even Alternative Computation

For the regular data flow, assume that each MMP contains four multiply-adders (MAs). Then, the delay time for the linear recursive equation becomes minimum based on the following "odd-even alternative computation".

Consider first the computation of the MMP1. Let all the elements of the matrices B and P be available; and let the ith row elements of A, a_1, ..., a_4 (i = 1, ..., 4) arrive at step (4i-3). Then, the most efficient computation may be arranged as follows:
Fig. 1: Task graph.

Table 1: 19 Tasks.

<table>
<thead>
<tr>
<th>TASK</th>
<th>COMPUTATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$M_0 = A_0 B_0$</td>
</tr>
<tr>
<td>2</td>
<td>$T_1 = T_{i-1} M_0, T_{i-1} \rightarrow M_1$</td>
</tr>
<tr>
<td>3</td>
<td>$M_2 = Q_1 M_0$</td>
</tr>
<tr>
<td>4</td>
<td>$M_3 = M_1 M_2$</td>
</tr>
<tr>
<td>5</td>
<td>$T_1 = T_{i-1} M_0, T_{i-1} \rightarrow M_4$</td>
</tr>
<tr>
<td>6</td>
<td>$M_5 = Q_1 M_0$</td>
</tr>
<tr>
<td>7</td>
<td>$M_7 = M_1 M_5$</td>
</tr>
<tr>
<td>8</td>
<td>$M_8 = Q_2 M_0$</td>
</tr>
<tr>
<td>9</td>
<td>$M_9 = M_4 M_5 + M_3$</td>
</tr>
<tr>
<td>10</td>
<td>$T_1 = T_{i-1} M_0 + M_3, T_{i} \rightarrow M_4$</td>
</tr>
<tr>
<td>11</td>
<td>$M_0 = M_0 W_1$</td>
</tr>
<tr>
<td>12</td>
<td>$M_3 = M_4 M_5^2$</td>
</tr>
<tr>
<td>13</td>
<td>$w_1^i = w_1^{i+1}, M_3 + M_1, w_1^i \rightarrow M_1$</td>
</tr>
<tr>
<td>14</td>
<td>$M_5 = TR(M_4 M_5^2)$</td>
</tr>
<tr>
<td>15</td>
<td>$M_6 = M_1 M_5^2$</td>
</tr>
<tr>
<td>16</td>
<td>$\tau_i = M_0 g^i + M_1^i$</td>
</tr>
</tbody>
</table>

Fig. 2: Linear recursive equation.

step 1 $x_{11} = a_{11} b_{11} + a_{12} b_{21} + a_{13} b_{31} + a_{14} b_{41} + p_{11}$
step 2 $x_{12} = a_{11} b_{12} + a_{12} b_{22} + a_{13} b_{32} + a_{14} b_{42} + p_{12}$
step 3 $x_{13} = a_{11} b_{13} + a_{12} b_{23} + a_{13} b_{33} + a_{14} b_{43} + p_{13}$
step 4 $x_{14} = a_{11} b_{14} + a_{12} b_{24} + a_{13} b_{34} + a_{14} b_{44} + p_{14}$

step 13 $x_{41} = a_{41} b_{11} + a_{42} b_{21} + a_{43} b_{31} + a_{44} b_{41} + p_{11}$
step 14 $x_{42} = a_{41} b_{12} + a_{42} b_{22} + a_{43} b_{32} + a_{44} b_{42} + p_{12}$
step 15 $x_{43} = a_{41} b_{13} + a_{42} b_{23} + a_{43} b_{33} + a_{44} b_{43} + p_{13}$
step 16 $x_{44} = a_{41} b_{14} + a_{42} b_{24} + a_{43} b_{34} + a_{44} b_{44} + p_{14}$

Fig. 3 shows the data-dependence graph and its mapping onto the MAS. Because the utilization of all the MASs is 100%, the delay time for the completion of the matrix multiply-addition becomes minimum. Only the communications between the adjacent MASs are sufficient for the computation.

Secondly, consider the computation of the MMP2. Let all the elements of the matrices $C$ and $Q$ be available. Because the data $x_{ij}$ from the MMP1 arrive at step $(4i+j-3)$ in a serial manner, the MMP2 must perform the following computation to make the MASs fully utilized:

step 1 $o_{11} = x_{11} c_{11} + q_{11}$
step 2 $o_{12} = x_{11} c_{12} + q_{12}$
step 3 $o_{13} = x_{11} c_{13} + q_{13}$
step 4 $o_{14} = x_{11} c_{14} + q_{14}$

step 13 $o_{41} = x_{41} c_{41} + q_{41}$
step 14 $o_{42} = x_{41} c_{42} + q_{42}$
step 15 $o_{43} = x_{41} c_{43} + q_{43}$
step 16 $o_{44} = x_{41} c_{44} + q_{44}$

Fig. 4 shows the data-dependence graph and its mapping onto the MASs. Because the utilization of all the MASs is 100%, the delay time for the completion of the matrix multiply-addition becomes minimum. Only the communications between the adjacent MASs are sufficient for the computation.

Similarly, for any odd-stage MMP, the input data arrive in the same manner as that of the MMP1. The delay time becomes minimum based on the the data-dependence graph in Fig.3. While, for any even-stage
STEP 1

STEP 2

STEP 3

STEP 4

STEP 5

STEP 6

Fig. 3: Data-dependence graph of the odd-stage MMP.

Fig. 4: Data-dependence graph of the even-stage MMP.

MMP, the input data arrives in serial manner as that of the MMP2. The delay time becomes minimum based on the data-dependence graph of Fig.4. Therefore, the total delay time required for the linear recursive equation becomes minimum using the data-dependence graphs of Figs.3 and 4 in the odd-stage and even-stage MMPs alternatively.

4 Minimum-Delay-Time Architecture

Fig.5 shows a block diagram of the MMP designed based on two type of data-dependence graphs in the “odd-even alternative computation”. The MMP consists of four MAS, local memories (LMs), and additional hardware to provide localized control and two parallel I/O ports. A set of four multiplexers and three switches are sufficient for performing data communications between MAS and LMs. The interprocessor communications between the MMPs are also implemented by simple data latches via registers. Thus, the structure of the MMP is simple and regular. The localized control with an instruction memory (IM) and a counter allows the MMP to down-load the inverse-dynamics programs for a rotational/translational joint torque of any manipulator.

5 Performance Evaluations

Since the cycle time of the MMP is determined by the execution time of the multiplier $T_{mpy}$, the total delay time $T_d$ to obtain all the joint torques is expressed as follows:

$$T_d = T_c + T_s = \begin{cases} (19 \times 16)T_{mpy} + (7n - 3)T_{mpy} & \text{if } n \text{ is odd} \\ (19 \times 16)T_{mpy} + (7n)T_{mpy} & \text{otherwise} \end{cases}$$

where $T_c$ and $T_s$ are the times defined in Fig.6. The scheduling for the inverse dynamics computation of a six degrees-of-freedom manipulator is shown in Fig.6. In the current VLSI technology, the cycle time of 10 nsec is achievable. The total delay time is, therefore, 3.4 $\mu$sec which is 1/1000 that of the ordinary multiprocessor systems [4]-[7]. Moreover, Fig.7 shows that the delay time is almost constant irrespective of $n$. Fig.8 shows the layout of the MMP with 16-bit precision. The effective size of the MMP chip is approximately 14000x12000 $\lambda^2$, where $\lambda$ is the minimum geometry size. In case of 1 $\mu$m CMOS design rule, the chip size will be 6x7 mm$^2$.

6 Conclusion

In this paper, we have presented a matrix multiply-addition processor (MMP) for minimum-delay-time inverse dynamics computation on a linear array structure. The MMP architecture based on the data-dependence graphs of the “odd-even alternative computation” is very VLSI-oriented. Thus, it is possible to implement the MMP which consists of four multiply-adders in a single chip. The delay time becomes minimum based on two special cases of computation in the “odd-even alternative computation”. Moreover, the delay time is almost constant irrespective of the degrees-of-freedom of the manipulator. For the inverse
dynamics computation of a typical manipulator having six degrees-of-freedom, the total delay time of 3.4 \( \mu \text{sec} \) can be achievable on the proposed architecture with the current VLSI technology. Thus, development of the MMP is very important to reduce the servo loop time that necessary to achieve the desired manipulator motions in robot control.

References