Highly Parallel Collision Detection Processor for Intelligent Robots

著者	亀山　充隆
journal or publication title	IEEE Journal of Solid-State Circuits
volume	27
number	4
page range	500-506
year	1992
URL	http://hdl.handle.net/10097/46836

doi: 10.1109/4.126537

Highly Parallel Collision Detection Processor for Intelligent Robots

Michitaka Kameyama, *Senior Member, IEEE,* Tadao Amada, and Tatsuo Higuchi, *Fellow, IEEE*

***Abstract*—In intelligent robots capable of autonomous work, the development of a high-performance special-purpose VLSI processor for collison detection will become very important for automatic motion planning. Conventionally, this kind of processing is performed by general-purpose processors. In this paper, a first collision detection VLSI processor is proposed to achieve ultrahigh-performance processing with an ideal parallel processing scheme. A large number of coordinate transformations and memory accesses to the obstacle memory are fully utilized in the processing algorithm, so that direct collision detection can be executed with a VLSI-oriented regular data flow. The structure of each processing element (PE) is very simple because a PE mainly consists of a COordinate Rotation DIgital Computer (CORDIC) arithmetic unit for the coordinate transformation and memories for the storage of manipulator and obstacle information. When 100 PE's are used to make parallel processing, the performance is about 10 000 times faster than that of conventional approaches using a single general-purpose microprocessor.**

I. Introduction

Motion planning involves finding a path from an initial robot configuration to a given final configuration that avoids collisions with obstacles in the workspace. The task is essential for intelligent robots to move autonomously in the free workspace. The most fundamental problem in motion planning is collision detection between a robot manipulator and obstacles. There are many kinds of algorithms for solving collision detection problem [1]–[3]. Usually a manipulator and obstacles are modeled by polyhedra, cylinders, or spheres and collision is detected by examining the contact between those models. The use of these representations makes the memory capacity smaller than the complete representation, however, it does not approximate the complex shapes of a manipulator and obstacles well, and there is much of the collision free space that is occupied by parts of cylinders and spheres. In addition, many of the advanced algorithms in these fields require up to a few seconds of computational time even if a high-performance workstation is used.

From this point of view, the implementation of a collision detection processor using VLSI technology can meet the computational speed requirement. Therefore, the development of a high-performance special-purpose VLSI processor for collision detection is a very important subject.

Manuscript received August 28, 1991; revised December 16, 1991.
M. Kameyama is with the Department of Information Engineering, Tohoku University, Aoba, Aramaki, Aoba-ku, Sendai 980, Japan.
T. Amada and T. Higuchi are with the Department of Electronic Engineering, Tohoku University, Aoba, Aramaki, Aoba-ku, Sendai 980, Japan.
IEEE Log Number 9106387.

In this paper, we describe a newly developed collision detection VLSI processor using an ideal parallel processing scheme. If we use many obstacle memories which are distributed in all processing elements (PE's), the communication-free architecture can be constructed. Namely, the proposed processor is completely free from communication between PE's, because all the information on obstacles is obtained from a single PE itself and the computation can be localized. This ideal parallel processing architecture reduces the computational time of the collision detection linearly as the number of PE's increases.

A PE performs the computation of a large number of coordinate transformations and memory access control for collision detection. The coordinate transformation between the joint space and the Cartesian space of the robot is very important since robots are controlled in the joint space, whereas obstacles are located in the Cartesian space. However, several kinds of elementary operations are required for the coordinate transformation, which is very time consuming. For the solution, the COordinate Rotation DIgital Computer (CORDIC) algorithms [5]–[10] are efficiently employed for the coordinate transformation, because the elementary operations can be efficiently computed using two-dimensional (2-D) vector rotations [4].

A PE is designed using a VLSI CAD system and the performance of the VLSI processor is evaluated. The PE contains 320K transistors, and the chip size becomes 13.5 mm × 15.7 mm in 2-μm CMOS design rule. When 100 PE's are used to make parallel processing, the typical collision detection time becomes about 450.5 μs. This performance is estimated to be 10 000 times faster than conventional approaches using general-purpose microprocessors.

II. Algorithm

A. Collision Detection

To perform exact collision detection, the representation of a manipulator and obstacles located in workspace is important. The most direct way to represent a manipulator is discrete representation of its surface, as shown in Fig. 1. A manipulator is represented by a set of discrete points covering the surface. Let the vector $Q_{ij} (= (x_{ij}, y_{ij}, z_{ij}))$ be

0018-9200/92$03.00 © 1992 IEEE

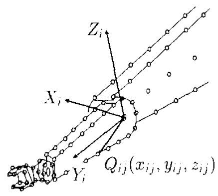


Fig. 1. Manipulator representation.

coordinates of the discrete points on the ith link surface, where $i = 1, 2, 3, \cdots, l$, $j = 1, 2, 3, \cdots, m_i$, the value l is the most distal link, and m_i is the number of discrete points on the ith link surface.

Similarly, obstacles are represented by a three-dimensional (3-D) image composed of cubic discrete pixels as shown in Fig. 2. Let $P(x_e, y_e, z_e)$ be one of these pixels in the coordinate system defined in the workspace.

When the joint angles are specified, the following algorithm gives the result of collision detection:

begin $i = 1; j = 1;$ *Flag* $= 0;$
  while $(i \leq l)$ **do**
  begin
    while $(j \leq m_i)$ **do**
    begin
      a) For Q_{ij}, perform the coordinate transformation using the given joint angles. Let the result thus obtained be Q'_{ij}.
      b) Check conflict between the obstacle pixels and the discrete point on the manipulator surface Q'_{ij} by means of memory access.
      If Q'_{ij} conflicts with an obstacle pixel **then** *Flag*$=1$ and **goto result**.
    increment j
    end;
  increment i
  end;
result:
If *Flag*$=1$ **then** collision is detected;
else (*Flag*$=0$) collision free;
end

In the execution of the above algorithm, a large number of coordinate transformations and memory access control are involved in the while loop. However, steps a) and b) can be executed in parallel for every Q_{ij}. The proposed collision detection processor utilizes the nature of this parallelism.

B. Coordinate Transformation

The coordinate transformation requires the computation of several elementary operations: multiplications, additions, divisions, and computation of trigonometric functions. However, these elementary operations are efficiently computed by the use of 2-D vector rotations [4]. For an example, let us consider the computation of (1):

$$x' = (x \cos \alpha - z \sin \alpha) \cos \beta - y \sin \beta. \tag{1}$$

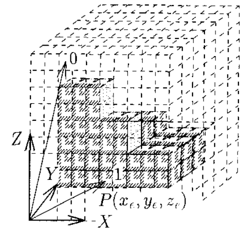


Fig. 2. Obstacle representation.

$x_0 \rightarrow$ [X] $\rightarrow K(x_0 cos z_0 - y_0 sin z_0)$
$y_0 \rightarrow$ [Y] $\rightarrow K(x_0 sin z_0 + y_0 cos z_0)$
$z_0 \rightarrow$ [Z] $\rightarrow 0$
ROTATION

$x_0 \rightarrow$ [X] $\rightarrow K\sqrt{x_0^2 + y_0^2}$
$y_0 \rightarrow$ [Y] $\rightarrow 0$
$z_0 \rightarrow$ [Z] $\rightarrow tan^{-1}(y_0/x_0) + z_0$
VECTOR

$x_0 \rightarrow$ [X] $\rightarrow x_0$
$y_0 \rightarrow$ [Y] $\rightarrow y_0 + x_0 z_0$
$z_0 \rightarrow$ [Z] $\rightarrow 0$
MULTIPLICATION

$x_0 \rightarrow$ [X] $\rightarrow x_0$
$y_0 \rightarrow$ [Y] $\rightarrow 0$
$z_0 \rightarrow$ [Z] $\rightarrow z_0 + y_0/x_0$
DIVISION

$$K = \prod_{i=1}^{n-1} (1/cos\theta_i)$$

Fig. 3. CORDIC functions.

Equation (1) can be computed by the following successive steps.

Step 1: If the coordinate (x, z) is rotated by α, we can obtain $(x \cos \alpha - z \sin \alpha)$ as follows:

$$\begin{bmatrix} \cos \alpha & -\sin \alpha \\ \sin \alpha & \cos \alpha \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} x \cos \alpha - z \sin \alpha \\ x \sin \alpha + z \cos \alpha \end{bmatrix}.$$

Step 2: Then, let us consider the rotation of $(x \cos \alpha - z \sin \alpha, y)$ by β. As a result, we can obtain x' according to the following equation:

$$\begin{bmatrix} \cos \beta & -\sin \beta \\ \sin \beta & \cos \beta \end{bmatrix} \begin{bmatrix} x \cos \alpha - z \sin \alpha \\ y \end{bmatrix} = \begin{bmatrix} (x \cos \alpha - z \sin \alpha) \cos \beta - y \sin \beta \\ (x \cos \alpha - z \sin \alpha) \sin \beta + y \cos \beta \end{bmatrix}.$$

To compute (1), four multiplication, two subtraction, and four trigonometric functions are required. Using 2-D vector rotations, only two 2-D vector rotations are sufficient. From this point of view, the coordinate transformation is performed by using only 2-D vector rotations which can be efficiently computed using the CORDIC algorithms.

Fig. 3 summarizes the basic functions in the CORDIC algorithms. The coordinate transformation for a manipulator with three degrees of freedom can be described as

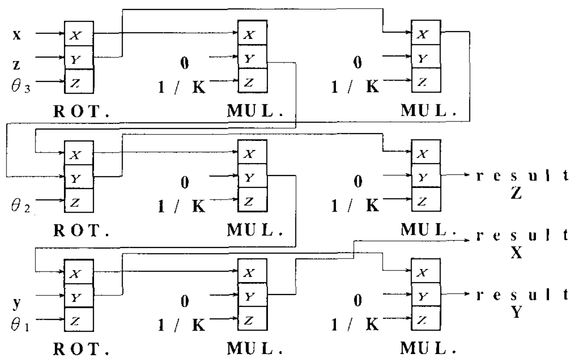


Fig. 4. Computational flow diagram for the coordinate transformation.

follows:

$$\begin{bmatrix} x_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} \cos\theta_3 & -\sin\theta_3 \\ \sin\theta_3 & \cos\theta_3 \end{bmatrix} \begin{bmatrix} x \\ z \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ Z \end{bmatrix} = \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 \\ \sin\theta_2 & \cos\theta_2 \end{bmatrix} \begin{bmatrix} x_2 \\ z_2 \end{bmatrix}$$

$$\begin{bmatrix} X \\ Y \end{bmatrix} = \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 \\ \sin\theta_1 & \cos\theta_1 \end{bmatrix} \begin{bmatrix} x_1 \\ y \end{bmatrix}.$$

It can be efficiently computed using only nine functions as shown in Fig. 4. The 2-D vector rotations can be applied to any other solution for the coordinate transformations.

III. Architecture

A. Highly Parallel Architecture

As indicated in the preceding section, the algorithm consists of the coordinate transformations of Q_{ij}'s and the memory access control. These two kinds of operations are repeated for all Q_{ij}'s, so that they are an iterative process which can be executed in parallel. The tremendous computational power for the collision detection can be provided by the ideal parallel architecture.

In conventional general-purpose processors, it takes a few seconds to perform collision detection. However, a special parallel architecture enables us to perform collision detection at high speed. The use of many memories instead of a shared memory is rather important, because recent advances in VLSI technology make it possible to include a large capacity of memory into a single chip and the communication between PE's is unnecessary. The parallelism exhibits the potential for the attainment of massively parallel processing.

The parallel feature of the collision detection algorithm makes it possible to construct a parallel structure as shown in Fig. 5. The collision detection processor consists of n identical PE's which perform collision detection in parallel without any communication between PE's. At the beginning of collision detection, joint angles are broadcasted to each PE. Then, each PE performs the coordinate

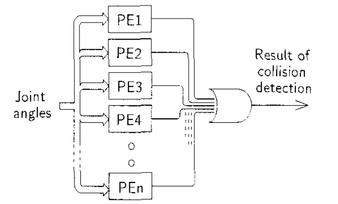


Fig. 5. Block diagram of the collision detection processor.

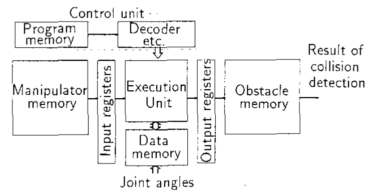


Fig. 6. Block diagram of PE.

Address	Data
0	x_{11}
1	y_{11}
2	z_{11}
3	x_{12}
•	•
•	•
•	x_{ij}
•	y_{ij}
•	z_{ij}
•	•
•	•
L	z_{im_i}

Fig. 7. Manipulator memory organization.

transformations for the assigned Q_{ij}'s at the given joint angles, and these data are used as the conflict check in the obstacle memory. The final result of collision detection is transmitted through the OR gate.

B. Structure of PE

A block diagram of the PE structure is shown in Fig. 6. A PE mainly consists of an obstacle memory, a manipulator memory, an execution unit (EU), and a control unit. Each of these major functional blocks is discussed below in detail.

1) Manipulator Memory: A 1-kb SRAM is provided for the storage of the manipulator discrete point coordinates. This capacity enables about 50 discrete points on the surface to be stored in a serial manner. Thus, $Q_{ij} (= (x_{ij}, y_{ij}, z_{ij}))$ is stored in the memory as shown in Fig. 7. At the beginning of collision detection for each discrete point, the data of its coordinates (x_{ij}, y_{ij}, z_{ij}) are transmitted to the EU from the manipulator memory.

2) Obstacle Memory: The obstacle pixels are represented by 1 or 0, where 1 and 0 designate obstacle occupying and free space, respectively. The 3-D coordinates of the obstacle pixels are linearly mapped into the addresses as shown in Fig. 8. The address of $P(x_e, y_e, z_e)$

Address	*Data*
0	$0\ (= P(0,0,0))$
1	$0\ (= P(0,0,1))$
2	$1\ (= P(0,0,2))$
3	$1\ (= P(0,0,3))$
•	•
•	•
•	•
$lmx_e + my_e + z_e$	$1\ (= P(x_e, y_e, z_e))$
•	•
•	•
•	•
$klm - 1$	$0\ (= P((k-1),(l-1),(m-1)))$

Fig. 8. Obstacle memory organization.

corresponds to $mlx_e + my_e + z_e$, where the workspace is divided into klm pixels. A large-capacity DRAM is essential to represent the obstacle information precisely. The 256-kb DRAM is provided to store the $64 \times 64 \times 64$ pixels of the 3-D obstacle image. If more precise representation is required, a DRAM with a larger capacity is replaced. Collision between the manipulator and obstacles is easily determined by reading the obstacle memory with the address being the result of the coordinate transformation.

3) Execution Unit: The execution unit (EU) performs the coordinate transformation very fast. The CORDIC algorithms are very suitable for computing the 2-D vector rotations. Using the CORDIC algorithms, the 2-D vector rotation is computed with iterative procedures involving only shift-and-add operations at each step. The EU based on the CORDIC algorithms becomes very simple and compact. In the CORDIC algorithms, 16-b fixed-point arithmetic operations are selected from the error analysis using simulation. The EU contains barrel shifters, adders, multiplexers (MUX's), input and output registers, accumulators, and a constant generator which generates arctangent radix constants as shown in Fig. 9. Input and output registers introduce pipelining into the data flow.

4) Control Unit: The control unit consists of three major blocks as shown in Fig. 10: a 4-b program counter, a decoder, and a program memory. Since all functions performed in the EU are executed in 16 clock cycles without conditional branching, the 4-b counter is sufficient to specify the program sequence. The 8-b program memory address counter (PAC) keeps track of the current program memory address.

The general instruction encoding format is shown in Fig. 11. All the instructions are 8×5-b words in length. Each function of Fig. 3 can be described using one instruction. The OP field of the instruction specifies the function to be executed in the EU. The FML field specifies whether the instruction is the first, a middle, or the last one of the function in the coordinate transformation. The *XS*, *YS*, and *ZS* fields specify whether the input data are in the data memory or in the accumulators. These data are stored in the op-code register I (OPCRI) and the op-code register II (OPCR II), and the outputs of these registers are used as the control signal in the instruction cycle. The other fields specify the address of the data

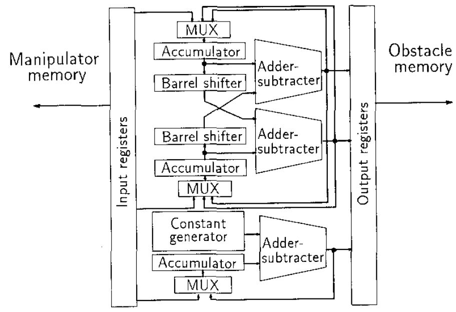


Fig. 9. Block diagram of EU.

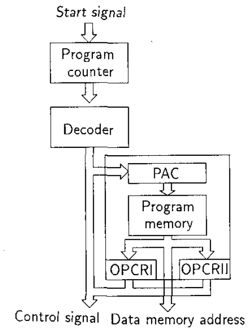


Fig. 10. Block diagram of the control unit.

	4 3 2 1 0	
0	Address of x0	
1	Address of y0	
2	Address of z0	
3	OP	XS | ZS
4	YS	FML
5	Address of xn	
6	Address of yn	
7	Address of zn	

Fig. 11. Instruction format.

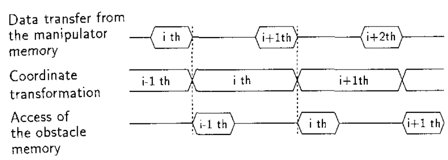


Fig. 12. Pipeline processing in collision detection.

memory when the memory is specified by the *XS*, *YS*, and *ZS* fields.

Fig. 12 shows a timing diagram of PE operation. To enhance the computational speed, the control signals allow the coordinate transformation and the memory access to be overlapped using pipelining. Since the propagation delay time of the decoder influences the clock period, the decoder is designed using hardwired circuitry.

IV. Evaluation of the VLSI Processor

A. Chip Layout

Fig. 13 shows some basic CMOS circuits using in the proposed VLSI processor. Based on these circuits, each

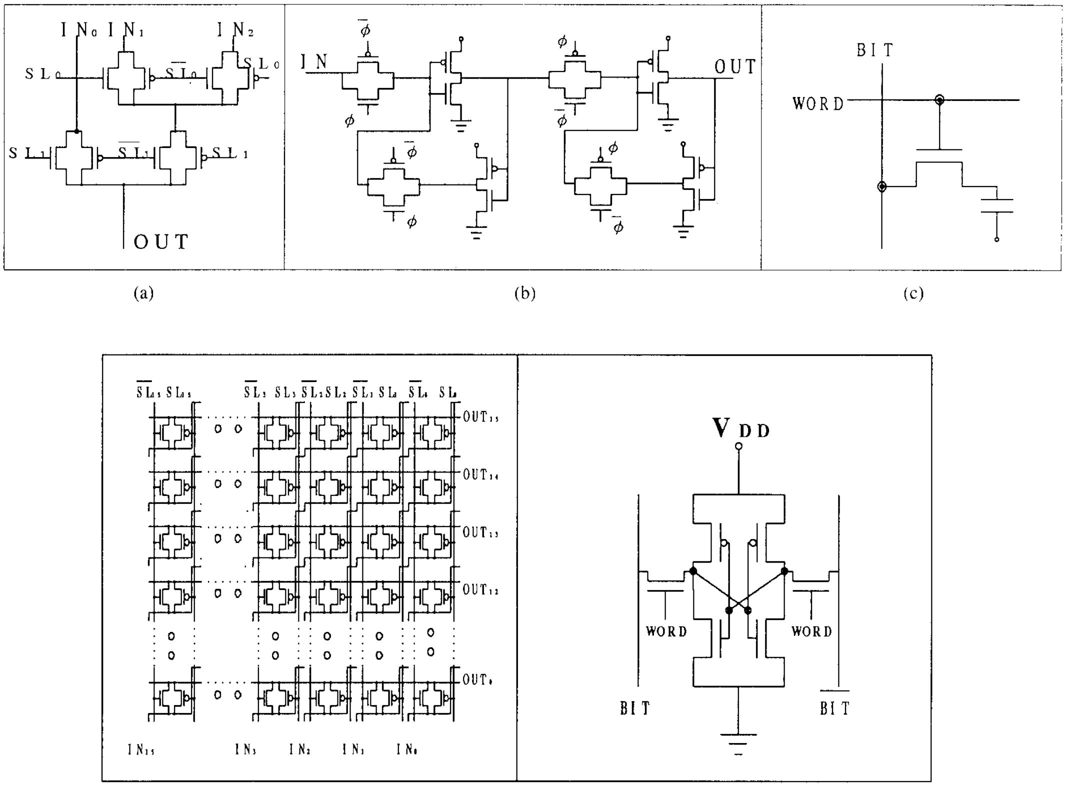
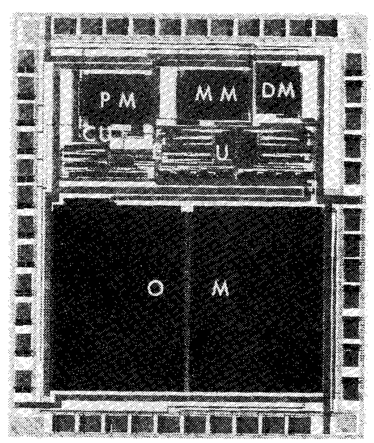


Fig. 13. Basic CMOS circuits for the layout: (a) 3-to-1 MUX, (b) register cell, (c) DRAM cell, (d) barrel shifter, and (e) SRAM cell.

PE is designed as shown in Fig. 14, where DM, MM, PM, CU, OM, and EU are data memory, manipulator memory, program memory, control unit, obstacle memory, and execution unit, respectively. The features of the designed chip are summarized in Table I. The chip dimensions are 13.5 mm $\times$ 15.7 mm with 2-μm CMOS design rule with a double-layer metal. The number of transistors is about 320K. In the following sections, the performance is discussed in detail.



DM : Data memory
MM : Manipulator memory
PM : Program memory
CU : Control unit
EU : Execution unit
OM : Obstacle memory

Fig. 14. Layout of PE.

B. Speed Evaluation

Fig. 15 summarizes the execution timing. Let T_{ct} be the time for the coordinate transformation and t_a be the memory access time. Since the three manipulator data x_{ij}, y_{ij}, and z_{ij} are transmitted to the EU at the beginning of the operation for each manipulator discrete point, the time for data transfer from manipulator memory becomes $3t_a$. Let M be the number of the manipulator points assigned to each PE. Then, the time T for the collision detection becomes

$$T = MT_{ct} + 4t_a. \tag{2}$$

The coordinate transformation time T_{ct} is

$$T_{ct} = 16Nt_c \tag{3}$$

where 16 is the number of clock cycles for one instruction, N is the number of instructions required for the coordinate transformation, and t_c is the clock period determined by a delay for the critical path in the EU as shown

TABLE I
FEATURES OF PE
(2-μm CMOS design rule)

Obstacle memory	256-kb DRAM
EU	16-b fixed-point arithmetic
Chip size	13.5×15.7 mm^2
Number of transistors	320 000
Performance	9.5 μs/manipulator point

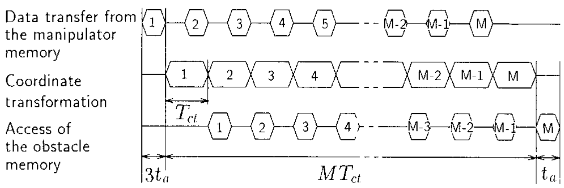


Fig. 15. Execution timing.

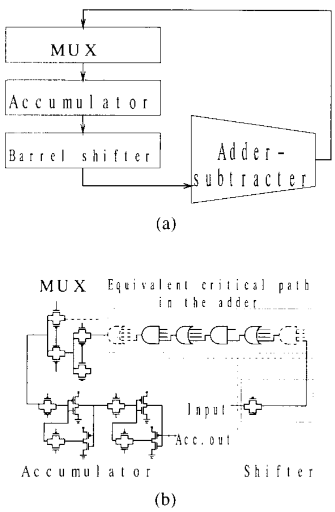


Fig. 16. Critical path in EU. (a) Block diagram. (b) Cascade chain of CMOS circuit in the critical path.

in Fig. 16(a). The delay time t_c for the critical path corresponds to the sum of the add time, the shift operation time, the multiplexer propagation time, and the accumulator setup time. Fig. 16(b) shows the cascade chain of the CMOS circuits in the critical path. Fig. 17 shows SPICE2 analysis of the delay times. The propagation delay times of input to the shifter, the adder, the multiplexer, and the accumulator are shown in Fig. 17(a), (b), (c), and (d), respectively. The propagation delay time for the critical path is about 62.0 ns from Fig. 17(d). Therefore, the chip can operate at the frequency of 16 MHz.

Consider an example of the case where nine CORDIC functions are required for the coordinate transformation as shown in Fig. 4. Let the number of manipulator points required be 5000. Also, assume that the manipulator coordinates are shared with 100 PE's. Then, each PE must perform 50 coordinate transformations. Since the clock period t_c in the EU is 62.5 ns and memory access time is 125.0 ns, the time for the collision detection becomes 450.5 μs from (2) and (3). Conventionally, it takes a few seconds using a general-purpose microprocessor. This performance is about 10 000 times faster than that of conventional approaches using a single microprocessor. If the collision detection is executed 1000 times in the search

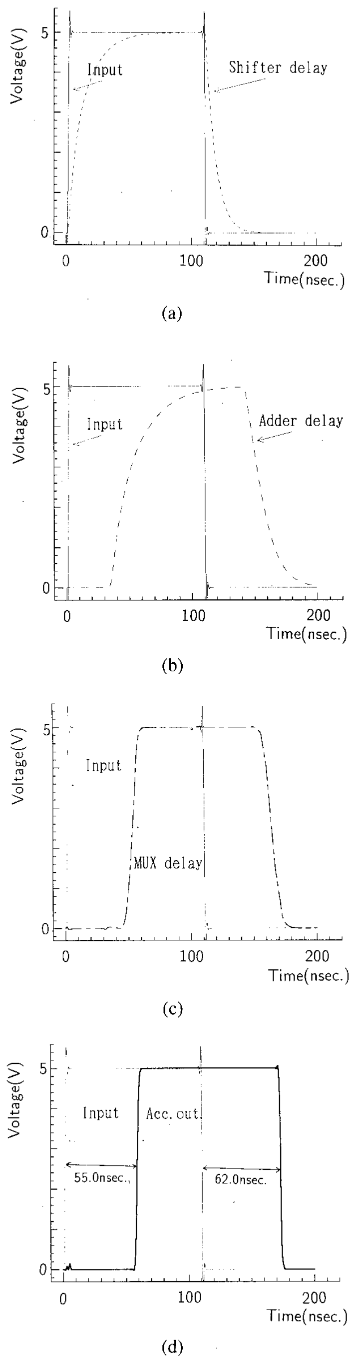


Fig. 17. Propagation delay times in EU: (a) delay time of the shifter, (b) delay time of the adder, (c) delay time of the multiplexer, and (d) delay time of the accumulator.

for a collision-free path, the total time will be 450.5 ms. This performance is enough for intelligent behavior of robots.

V. Conclusion

A high-speed collision detection processor has been presented. To perform collision detection at high speed, a parallel architecture is considered. The parallelism does not require any communication between PE's. Moreover, the 2-D vector rotations are utilized for the coordinate transformation, which is the key processing in the collision detection algorithm. Each PE is designed using 2-μm CMOS design rule with a double-layer metal. On the basis of these two concepts, the performance is 10 000 times faster than that of the conventional approaches using a single microprocessor. It is expected that the collision detection processor will be applied to practical robot motion planning.

References

[1] E. G. Gilbert and S. M. Hong, "A new algorithm for detecting the collision of moving objects," in *Proc. 1989 IEEE Int. Conf. Robotics Automat.*, May 1989, pp. 8-14.

[2] S. Bonner and R. B. Kelley, "A novel representation for planning 3-D collisin-free paths," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 6, pp. 1337-1351, Nov./Dec. 1990.

[3] T. Lozano-Pérez, "A simple motion-planning algorithm for general robot manipulators," *IEEE J. Robotics Automat.*, vol. RA-3, no. 3, pp. 224-238, June 1987.

[4] M. Kameyama, T. Matsumoto, H. Egami, and T. Higuchi, "Implementation of a high performance LSI for inverse kinematics computation," in *Proc. 1989 IEEE Int. Conf. Robotics Automat.*, May 1989, pp. 757-762.

[5] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, pp. 330-334, Sept. 1959.

[6] J. S. Walther, "A unified algorithm for elementary functions," in *Proc. 1971 AFIPS Spring Joint Comput. Conf.*, pp. 379-385.

[7] G. L. Haviland and A. A. Tuszynski, "A CORDIC arithmetic processor chip," *IEEE Trans. Comput.*, vol. C-29, pp. 68-79, Feb. 1980.

[8] T. W. Curtis, P. Allison, and J. A. Howard, "A CORDIC processor for laser trimming," *IEEE Micro*, vol. 6, pp. 61-71, June 1986.

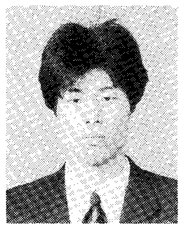
[9] C. S. G. Lee and P. R. Chang, "A maximum pipelined CORDIC architecture for inverse kinematic position computation," *IEEE J. Robotics Automat.*, vol. RA-3, no. 5, pp. 445-458, Oct. 1987.

[10] M. D. Ercegovac and T. Lang, "Redundant and on-line CORDIC: Application to matrix triangularization and SVD," *IEEE Trans. Comput.*, vol. 39, no. 6, pp. 725-740, June 1990.

Michitaka Kameyama (M'79-SM'92) received the B.E., M.E., and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1973, 1975, and 1978, respectively.

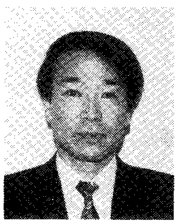
He is currently a Professor in the Department of Information Engineering, Tohoku University. His general research interests include robot electronics, VLSI systems, highly reliable digital systems, and multiple-valued logic systems.

Dr. Kameyama received the Outstanding Paper Awards at the 1984, 1985, 1987, and 1989 IEEE International Symposiums on Multiple-Valued Logic (with T. Higuchi *et al.*), the Technically Excellent Award from the Society of Instrument and Control Engineers of Japan in 1986 (with T. Higuchi *et al.*), the Outstanding Transactions Paper Award from the Institute of Electronics, Information and Communication Engineers of Japan in 1989 (with T. Higuchi *et al.*), and the Technically Excellent Award from the Robotics Society of Japan in 1990 (with T. Higuchi *et al.*).



Tadao Amada received the B.E. and M.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1990 and 1992, respectively.

He joined Fujitsu Limited in 1992. His research interests and activities include VLSI processors for robots.



Tatsuo Higuchi (M'70-SM'83-F'92) received the B.E., M.E., and D.E. degrees in electronic engineering from Tohoku University, Sendai, Japan, in 1962, 1964, and 1969, respectively.

He is currently a Professor in the Department of Electronic Engineering, Tohoku University. His general research interests include the design of 1-D and 2-D digital filters, multiple-valued logic systems, fault-tolerant computing, and VLSI computing structures for signal processing and image processing.

Dr. Higuchi received the Outstanding Paper Awards at the 1984, 1985, 1987, and 1989 IEEE International Symposiums on Multiple-Valued Logic (with M. Kameyama *et al.*), the Outstanding Transactions Paper Award from the Society of Instrument and Control Engineers of Japan in 1984 (with M. Kawamata), the Technically Excellent Award from the Society of Instrument and Control Engineers of Japan in 1986 (with M. Kameyama *et al.*), the Outstanding Transactions Paper Award from the Institute of Electronics, Information and Communication Engineers of Japan in 1989 (with M. Kameyama *et al.*), and the Technically Excellent Award from the Robotics Society of Japan in 1990 (with M. Kameyama *et al.*).