Collision Detection VLSI Processor for Highly-Safe Intelligent Vehicles Using a Multiport Content-Addressable Memory

Masanori HARIYAMA and Michitaka KAMEYAMA

*Department of Computer and Mathematical Sciences,
Graduate School of Information Sciences, Tohoku University,
Aoba 05, Aramaki, Aoba-ku, Sendai 980-8579, Japan*

Received June 14, 1999; final version accepted September 1, 1999

High-speed collision detection plays an essential role in a collision warning system for highly-safe vehicles. In collision detection, high computational power is required to perform matching operation between discrete points of obstacles and a vehicle. This paper presents design of a collision detection VLSI processor using content-addressable memories (CAMs) for parallel matching operation. The VLSI processor consists of identical CAMs and processing elements (PEs) for coordinate transformation. If they are fully utilized and the fixed computation time of the VLSI processor is given as a constraint, area minimization of the VLSI processor is attributed to minimization of the area-time product of the CAM and the PE. To reduce the area-time product of the CAM, a multiport CAM (MCAM) is proposed. Each word stored in the MCAM is shared between multiple matching units so that matching operation can be performed in parallel with multiple input words without increasing memory capacity. It is shown that the area of the VLSI processor using 4-port CAMs can be reduced to 43% in comparison with the area of the VLSI processor without them under a time constraint.

KEYWORDS: area-time product minimization, parallel processing, VLSI-oriented algorithm, high-level synthesis, path planning

1 Introduction

High-speed collision detection plays an essential role in a collision warning system for highly-safe vehicles. However, its computational requirement exceeds the computational power of present-day general-purpose processors. One efficient way to overcome this problem is development of special-purpose VLSI processors. From this point of view, a VLSI-oriented algorithm based on regular iterations of coordinate transformation and matching operation has been proposed [1]–[3]. This paper presents design of a collision detection VLSI processor based on a systematic methodology for area minimization under a time constraint. The VLSI processor consists of multiple identical clusters, each of which has a CAM for parallel matching operation and processing elements (PEs) for coordinate transformation. Therefore, the area of the VLSI processor is given by the product of the area of the cluster and the number of the clusters. If the clusters are fully utilized and fixed computation time of the VLSI processor is given as a constraint, the number of the clusters is in proportion to the computation time of the cluster. Therefore, area minimization of the VLSI processor is attributed to minimization of the area-time product of the cluster. The concept of area-time-product minimization is also used to design the CAM and the PE in the cluster.

A multiport CAM (MCAM) and the PE based on a bit-serial pipelined architecture can be efficiently employed for area-time-product minimization. Each word stored in the MCAM is shared between multiple matching units to perform matching operation in parallel with multiple input words. To achieve the same performance as the MCAM, multiple single-port CAMs with the same memory capacity as the MCAM are necessary. Therefore, under a time constraint, the area of the MCAM becomes smaller than the total area of the single-port CAMs. In the PE, the bit-serial pipelined architecture for multiplication and addition are employed to achieve the full utilization of the PE in the gate level.

It is shown that the area of the VLSI processor using 4-port CAMs and the bit-serial pipelined PE becomes 43% of the area of the VLSI processor without them.

2 Algorithm

2.1 Representation of a Vehicle and Obstacles

As shown in Fig. 1, a vehicle is represented by a union of rectangular solids. Let us assume that each surface of a rectangular solid is parallel to the xy -plane, the yz -plane, or the zx -plane. Then, a rectangular solid can be expressed by a pair of the coordinates, (x_s, y_s, z_s) and (x_e, y_e, z_e) as shown in Fig. 2. A discrete point $P = (x, y, z)$ contained in a rectangular solid satisfies the following three conditions:

$$x_s \leq x \leq x_e, \quad y_s \leq y \leq y_e, \quad z_s \leq z \leq z_e. \quad (1)$$

Fig. 1 Rectangular solid representation of a vehicle.

Fig. 2 Rectangular solid.

Fig. 3 Motion of the vehicle.

Fig. 4 Coordinate transformation for an obstacle discrete point.

The rectangular solid representation is efficient in storage because a large number of discrete points are usually contained in a rectangular solid. To represent an arbitrary shape of an obstacle, we introduce a discrete point representation for its surface. There are many methods such as the stereo vision [4] of obtaining discrete points of obstacles. A vehicle discrete point is contained in a vehicle rectangular solid as expressed by (1), if collision exists between them. Equation (1) is evaluated only by magnitude comparison in parallel for all the rectangular solids using a CAM [7]–[11].

2.2 Coordinate Transformation

As shown in Fig. 3, locations of a vehicle at the time t and $t + dt$ are described in the static three-dimensional (3-D) space. Frames $x_0y_0z_0$ and $x_1y_1z_1$ are attached to the vehicle at the time t and $t + dt$, respectively. An obstacle discrete point is described relative to $x_0y_0z_0$. Coordinate transformation is to change a description of an obstacle discrete point from $x_0y_0z_0$ to $x_1y_1z_1$ as shown in Fig. 4. This idea is extremely important. The vehicle pixel information is not changed, but remains the same relative to $x_0y_0z_0$ and $x_1y_1z_1$ one, respectively, so that the real-time transformation from the discrete point representation to the rectangular one of the vehicle is not required.

A discrete point $P = (x, y, z)$ is translated to $P' = (x', y', z')$ with the matrix operation as follows.

$$P' = R(P - T) \quad (2)$$

where R and T are a 3×3 matrix called a *rotation* matrix and a 3×1 vector called a *translation* vector, respectively. Let R and T be as follows:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix}$$

then, Eq. (2) can be rewritten as

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} R_{11}x + R_{12}y + R_{13}z - (R_{11}T_1 + R_{12}T_2 + R_{13}T_3) \\ R_{21}x + R_{22}y + R_{23}z - (R_{21}T_1 + R_{22}T_2 + R_{23}T_3) \\ R_{31}x + R_{32}y + R_{33}z - (R_{31}T_1 + R_{32}T_2 + R_{33}T_3) \end{bmatrix}. \quad (3)$$

When the same rotation matrix and the same translation vector are used for all the discrete points, that is, all the elements of R and T are constant, the coordinate transformation for a discrete point requires only nine additions and nine multiplications. In the case that a vehicle moves on a flat ground as shown in Fig. 3, R and T are given by

$$R = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad T = \begin{bmatrix} x_R \\ y_R \\ 0 \end{bmatrix}$$

where θ is the angle between the x_0 -axis and the main axis of the vehicle, and x_R and y_R are the coordinates of the midpoint R between the two rear wheels at the time $t + dt$ as shown in Fig. 3. Since R is located at the origin at time t as shown in Fig. 3, (x_R, y_R) represents the translation in the $x_0 - y_0$ and $x_1 - y_1$ planes [6].

2.3 Collision Detection Algorithm

First, a transformation matrix is calculated from (2). Next, the result of collision detection is obtained by executing the following two steps for all the discrete points.

Step 1) Coordinate transformation for an obstacle discrete point P is performed using (2). This result becomes P' .

Step 2) Matching operation is performed. That is, (1) is evaluated for P' and all the vehicle rectangular solids.

These sequential steps can be executed in parallel with all the discrete points. The execution time of the matching operation increases with the number of rectangular solids in a straightforward manner. A content-addressable memory can be employed to perform the matching operation in a constant time.

3 Processor Architecture

3.1 Optimal Design Based on Minimization of an Area-Time Product

Figure 5 shows a structure of the collision detection VLSI processor. It consists of identical clusters, each of which has processing elements (PEs) for the coordinate transformation and a multiport CAM (MCAM) for the matching operation. Obstacle discrete points are equally distributed to the PEs which perform the coordinate

Fig. 5 Structure of the collision detection VLSI processor.

Fig. 6 Timechart of the coordinate transformation of the PE and access of the MCAM.

transformation in parallel. There is no need for communication between the clusters because the result of collision detection for a discrete point does not depend on the other discrete points. Therefore, the collision detection time for all the discrete points is in proportion to the number of the clusters. In the cluster, the matching operation and the coordinate transformation are overlapped by pipelining. If the setup time in Fig. 6 of pipelining can be ignored, that is, the number of discrete points is large, not only the spatially-parallel structure but also the pipeline one is efficient to reduce to the total collision detection time.

When the access time of the MCAM is larger than coordinate transformation time of the PE, utilization of the MCAM becomes 100%, while that of the PE does not become 100%. Let T_{cam} and T_{pe} be the access time of the MCAM and the coordinate transformation time of the PE, respectively. When $T_{cam} = K T_{pe}$ for an integer K (> 0), the PE can be shared between K input ports of the MCAM as shown in Fig. 6 to achieve 100% utilization of the PEs. Let N_{pe} be the number of PEs. Since the total number N_{port} of the input ports of the MCAM is given by

$$N_{port} = K N_{pe}, \quad (4)$$

the access time of the MCAM can be rewritten as

$$T_{cam} = \frac{N_{port}}{N_{pe}} T_{pe}. \quad (5)$$

Let us consider minimization of the total area of the VLSI processor, assuming that the total number N_{ob} of obstacle discrete points and total collision detection time T_t for all the discrete points are given. When the setup time of pipelining can be ignored, T_t is expressed as

$$T_t = T_{pe} \frac{N_{ob}}{N_{cl} N_{pe}}, \quad (6)$$

where N_{cl} is the number of clusters. The term $N_{ob} / (N_{cl} N_{pe})$ represents the number of discrete points which are assigned to the PE. The total area A_t of the VLSI processor is given by

$$A_t = N_{cl} \times (N_{pe} A_{pe} + A_{cam}), \quad (7)$$

where A_{pe} and A_{cam} is the areas of the PE and the MCAM, respectively. Using (5) and (4), this equation is rewritten as

$$\begin{aligned} A_t &= \frac{N_{ob} T_{pe}}{T_t N_{pe}} (N_{pe} A_{pe} + A_{cam}) \\ &= \frac{N_{ob}}{T_t} \left(A_{pe} T_{pe} + \frac{T_{cam}}{N_{port}} A_{cam} \right) \\ &= \frac{N_{ob}}{T_t} (A_{pe} T_{pe} + A_{cam} T_{camw}), \end{aligned} \quad (8)$$

where $T_{camw} = T_{cam} / N_{port}$ is the access time per input word of the MCAM. Since N_{ob} and T_t are given, A_t is minimized when $A_{pe} T_{pe}$ and $A_{cam} T_{camw}$ are minimized.

This design methodology can be applied to unit selection in high-level synthesis. Since multiple types of functional units with different characteristics (e.g., area and delay) are usually available, one important task in high-

Fig. 7 MCAM with $s \times m$ input ports.

level synthesis is to determine the number and the types of functional units for optimal design. Let us consider the design of VLSI processors that execute algorithms based on regular iterations of sequential processings. Then, functional units of the type with minimum area-time products are selected as optimal ones for each processing, and the number of functional units can be determined for the total processing time and the number of data items that are given as constraints.

3.2 Structure of the MCAM

To minimize the area-time product of the CAM, the MCAM is proposed. Figure 7 shows a block diagram of the MCAM which consists of a memory array with m words and a matching-unit array with $m \times s$ matching units, where m and s are the number of rectangular solids and the number of input words of the MCAM, respectively.

The matching units at a column are connected to an input port by a single bus, and the same input word is broadcasted to them. Each of them are also associated with a word of the memory array so that the matching operation can be performed in a word-parallel manner. In other words, (1) is evaluated in parallel with all the stored words. If (1) is satisfied, the corresponding response signals are set to 1, and others are set to 0. If at least one response signal is 1, collision exists. Otherwise, there is no collision. Therefore, the result of collision detection is obtained by ORing all the response signals. For an area-efficient design, dynamic NORs are used for ORing them.

The area-time product of the MCAM is small in comparison with that of a single-port CAM (SCAM) where each stored word is connected to a single MU. Let us consider the R-port CAM and the SCAM. In order to achieve the same performance as the R-port CAM, R SCAMs are necessary. The total memory capacity of the R SCAMs is larger than that of the MCAM since each of the SCAMs stores the same vehicle information. Consequently, the total area of the R SCAMs is larger than that of the MCAM under a condition of the same access time.

Figures 8 and 9 show the detailed structure of a word of the memory array and the MU, respectively. The six comparators are connected to memory cells which store X_s , X_e , Y_s , Y_e , Z_s and Z_e , respectively. Magnitude comparison is performed in a bit-serial manner from the most-significant bit to the least-significant bit. In our algorithm, the memory content is not changed because the vehicle pixel information is predetermined in advance. Therefore, a ROM cell is used as a memory cell so that the area and the access time can be reduced in comparison with those of the SRAM-type CAM [2].

3.3 Structure of the PE

Figure 10 shows the structure of the execution unit of the PE, which consists of 9 bit-serial pipelined multipliers shown in Fig. 11, 9 full adders and 15 pipeline registers. A bit-serial pipelined architecture is employed to minimize the area-time product of the PE by achieving the full utilization in the gate level.

4 Evaluation

Table 1 shows the comparison between a 60-Kbit 4-port CAM (4PCAM) and a 60-Kbit SCAM [3]. Table 2

Fig. 8 Word of the memory array.

Fig. 9 Matching unit.

BSPM: Bit-serial pipelined multiplier

Fig. 10 Structure of the PE.

FA: Full adder

Fig. 11 Structure of the bit-serial pipelined multiplier.

Table 1. Comparison between the CAMs.

	Area (mm ²)	Access time per input port (ns)	Area-time product
SCAM	3.0 × 2.4	48	346
4PCAM	4.9 × 2.4	12	141

Table 2. Comparison between the PEs.

	Coordinate transformation time (ns)	Area (mm × mm)	Area-time product
Serial	164	0.4 × 0.9	59
Bit-serial pipelined	40	0.7 × 0.95	27
Array	17.2	1.1 × 1.65	31

Table 3. Comparison between total areas of CAMs under the condition of the same access time per input word.

	Number of CAMs	Total area (mm ²)
SCAM	4	28.8
4PCAM	1	11.8

Table 4. Comparison between total areas of PEs under the condition of the same coordinate transformation time.

	Number of PEs	Total area (mm ²)
Serial	14	5.04
Bit-serial pipelined	4	2.66
Array	2	3.63

Table 5. Features of the VLSI processor.

Technology	0.8- μ m CMOS double-metal double-polysilicon
Chip size	$2.4 \times 6.3 \text{ mm}^2$
Clock frequency	300 MHz
Power supply voltage	5 V
Performance	12 ns/point
Number of transistors	140 000

shows comparison between PEs based on a serial multiplier, a bit-serial pipelined multiplier and an array multiplier. These results show that the 4PCAM and the PEs based on the bit-serial pipelined architecture are useful to minimize the area-time products.

Let us minimize the area of the VLSI processor based on the area-time minimization. Assumed that the collision detection time T_t and the number N_{ob} of obstacle discrete points are 120 μ s and 10,000 in (6), respectively. If the SCAM is used, the access time T_{camw} per input word and the number N_{cl} of clusters become 48 ns and 3, respectively. While, $T_{camw} = 12$ ns and $N_{cl} = 1$ using the MCAM. Since N_{cl} also represents the number of CAMs in the VLSI processor, the total area of CAMs is equal to the product of N_{cl} and the area of a CAM. The result of this evaluation is shown in Table 3. Similarly, the comparison result of the total areas of PEs is shown in Table 4. When the SCAM and the PE based on the serial multiplier are used, the total area of the VLSI processor becomes $28.8 \text{ mm}^2 + 5.04 \text{ mm}^2 = 33.84 \text{ mm}^2$ and maximal. While, when the 4PCAM and the PE based on the bit-serial multiplier are used, the total area of the VLSI processor becomes 14.46 mm^2 and minimal. This result shows that the total area of the VLSI processor based on the area-time minimization can be reduced to 43% in comparison with that of the VLSI processor without it under the condition of the same collision detection time. The layout of the VLSI processor and its features are shown in Fig. 12 and Table 5, respectively. The total area in the table is increased by 4% because of the interconnection and the empty area.

5 Conclusion

The systematic methodology for the design of a parallel VLSI processor is proposed based on minimization of area-time products, and it is applied to the collision detection VLSI processor. The keys to the minimization are the MCAM and the bit-serial pipelined PE.

The design methodology can be efficiently applied to all the VLSI processors for implementing the algorithms based on regular iterations of sequential processings. Moreover, the VLSI processor is also efficient in path planning for robot manipulators and related applications.

REFERENCES

- [1] M. Hariyama and M. Kameyama, "A Collision Detection Processor for Intelligent Vehicles," IEICE Trans. Electron., Vol. E76-C, No. 12, pp. 1804–1811, 1993.
- [2] M. Hariyama and M. Kameyama, "Design of a CAM-Based Collision Detection VLSI Processor for Robotics," IEICE Trans., Vol. E77-C, No. 7, pp. 1108–1115, 1994.
- [3] M. Hariyama and M. Kameyama, "A Collision Detection VLSI Processor Based on A ROM-Type Content-Addressable Memory for Intelligent Vehicles," IEICE Trans., Vol. J79-C-II, No. 11, pp. 698–705, 1996 (in Japanese).
- [4] S. Lee, M. Hariyama, and M. Kameyama, "A Three-Dimensional Instrumentation VLSI Processor Based on a Concurrent Memory-Access Scheme," IEICE Trans. Electron, Vol. E80-C, No. 11, pp. 1491–1498, 1997.
- [5] M. Hariyama and M. Kameyama, "Design of a Collision Detection VLSI Processor Based on Minimization of Area-Time Products," Proc. IEEE International Conference on Robotics and Automation, pp. 3691–3696, 1998.
- [6] J. C. Laatombe, "Robot Motion Planning," pp. 404–440, Kluwer Academic Publishers, 1991.
- [7] T. Yamagata, M. Mihara, T. Hamamoto, Y. Murai, T. Kobayashi, M. Yamada, and H. Ozaki, "A 288-kb Fully Parallel Content Addressable Memory Using a Stacked-Capacitor Cell Structure," IEEE J. Solid-State Circuits, Vol. 27, No. 12, pp. 1927–1933, 1992.
- [8] M. Yamada and H. Ozaki, "A 288-kb Fully Parallel Content Addressable Memory Using a Stacked-Capacitor Cell Structure," IEEE J. Solid-State Circuits, Vol. 27, No. 12, pp. 1927–1933, 1992.
- [9] I. Okabayashi, H. Kotani, and H. Kadota, "A Proposed Structure of 4 Mbit Content-Addressable and Sorting Memory," 1990 symp. on VLSI Circuits, pp. 109–110, 1990.
- [10] C. D. Stormon et al., "A General-Purpose CMOS Associative Processor IC and System," IEEE Micro, Vol. 12, No. 6, pp. 68–78, 1992.
- [11] D. Tavangarian, "Flag-Oriented Parallel Associative Architecture and Applications," IEEE Computer, Vol. 27, No. 11, pp. 41–52, 1994.