**Advances in Computer Science and Engineering**
Volume 5, Number 2, 2010, Pages 213-230
Published Online: November 23, 2010
This paper is available online at http://pphmj.com/journals/acse.htm
© 2010 Pushpa Publishing House

# QUALITY-DRIVEN FLOORPLANNING USING GENETIC ALGORITHM ACCELERATOR

**MASAYA YOSHIKAWA and HIDEKAZU TERAI**

Department of Information Engineering
Meijo University
1-501 Shiogamaguchi Tenpaku Nagoya, 468-8502, Japan
e-mail: evolution_algorithm@yahoo.co.jp

Department of VLSI System Design
Ritsumeikan University
Japan

## Abstract

With recent advance technologies, the floorplanning becomes an essential design step in VLSI layout design. In previous work, we proposed the bus-oriented floorplanning technique, which was based on Genetic Algorithm (GA). However, GA required a lot of calculation time, because it is the population-based algorithm. On the other hand, the reducing calculation time is one of the most important points for VLSI EDA tools. In this paper, we propose a new GA-based floorplanning technique using the dedicated hardware accelerator which synchronizes with software processing. It enables to reduce the calculation time while keeping the quality of solution. Experiments using benchmark data prove the validity of the proposed floorplanning technique using the hardware accelerator.

## 1. Introduction

With increasing circuit integration and downsizing, Large-scale Integrated circuits (LSIs) are currently designed on a nano-scale. In LSIs, interconnect delay, rather than gate delay, has become a constraint for the system's performance [1].

Keywords and phrases: floorplanning, accelerator, genetic algorithm.

Received April 25, 2010

Although the exact interconnect delay cannot be clarified unless the wiring is completed, the interconnect delay can be accurately estimated by designing a floorplan (floorplanning) that determines the outline placement of functional blocks. Therefore, floorplanning's importance has increased in the design of LSIs, system LSIs, in particular, to which many functions are mounted [2]. The conventional floorplanning technique determines the blocks' optimal positions in order to minimize the area and wire length. In a system LSI, however, multiple bus wirings with different bit widths generally connect several blocks. Therefore, bus wiring greatly affects the blocks' placement. Moreover, in bus wiring, short wires with few bends are desirable from the viewpoint of timing and electrical parasitic capacitance. In general, floorplanning uses two types of blocks: a hard block, in which the height and width are specified; and a soft block, the area of which is given and the shape of which is variable. Moreover, blocks with placement constraints such as interface (I/F) circuits also exist.

In previous work [13], we proposed the floorplanning technique, which synthetically considered not only the area and wire length, but also bus wiring, soft blocks, and placement constraints. As its method of representing solutions, the proposed technique adopted the sequence-pair (S-pair) [3], which can represent even a non-slicing structure [4]; and it used the Genetic Algorithm (GA) [5, 6] to search for solutions. GA is based on technologically modeling biological evolutionary process, and it has a powerful searching ability for combinatorial optimization problems. However, GA has an inherent time problem, because it is a population-based algorithm. On the other hand, reducing calculation time, that is, "quick turn around time" is one of the most important points for VLSI EDA tools.

In this paper, we propose a new dedicated hardware accelerator to overcome the inherent time problem of GA. The dedicated hardware accelerator processes the portion of data with a large processing load (computational amount); and software processes the portion of data with a small processing load. In other words, the proposed technique achieves interlocking processing by using both hardware and software.

Regarding related studies, floorplanning techniques using GA and S-pair have been reported by Shigehiro et al. [7], Nakaya et al. [8, 9] and others. Shigehiro et al. [7] proposed the crossover method of inheriting a parent individual's characteristics, and investigated the quality of solutions obtained using their methods. Nakaya et al.

[8, 9] proposed the evaluative technique using an elite degree and an adaptive crossover technique, and achieved excellent performance. These reports dealt with only hard blocks, and sought to minimize only the area and the wires' length. Imai et al. [10] and Graham and Nelson [11] have studied dedicated hardware to shorten the GA's processing time. Imai et al. created the architecture of the parallel GA for solving simple problems and, by converting the step number, they expected high-speed processing to be realized in the dedicated hardware. This dedicated hardware's processing speed could be several dozen times higher than that of software. Graham and Nelson [11] implemented the simple GA on a Field Programmable Gate Array (FPGA) to solve small-scale problems; and consequently, the dedicated hardware's processing speed was several times that of the software. However, no studies have been reported on an interlocking processing system using both hardware and software to solve practical problems such as VLSI floorplanning.

The organization of this paper is as follows: Section 2 briefly explains the floorplanning problem as preliminaries. Section 3 describes the base algorithm for floorplanning and Section 4 explains the GA accelerator. Section 5 reports the experiments using benchmark data. Section 6 summarizes and concludes this study.

## 2. Preliminaries

### 2.1. Floorplanning problem

The floorplanning problem in this study has $n$ pieces of rectangular blocks, $B_1, B_2, ..., B_n,$ and they consist of hard blocks and soft blocks. The hard block has the height and width which are fixed, and the soft block has the area which is given and the shape which is variable. The smallest rectangle which encloses all the blocks is called a *chip*. Moreover, each block has multiple terminals in its perimeter and inside; and is wired using the wiring layer on the block, based on the net list given by an input data. Here, the net represents a set of terminals connected by one signal. The wire length of the net is evaluated by the half perimeter of the smallest rectangle surrounding all the terminals.

### 2.2. Sequence-pair

S-pair is the method of representing the blocks' placement using sequence-pair $(\Gamma+, \Gamma-)$ consisting of the blocks' names. In S-pair, a block exists in sequence-pair $(\Gamma+, \Gamma-),$ and the relative locations of two blocks are specified based on the order

of $(\Gamma+, \Gamma-)$. The actual placement can be obtained by placing a block according to the relative location obtained from the S-pair. Figure 1 shows a floorplan and the relative placement of blocks. When the blocks' relative placement is determined, the horizontal constraint graph and the vertical constraint graph are created based on the relative placement of all the blocks, as shown in Figure 2. In these graphs, each block's width and height are given as the weight of the edge, respectively, and the longest route from the starting point to the end point is obtained. The totals of the weights of the edges of blocks' widths and heights on the longest route are equal to the width and height of the chip, and the chip's area is obtained by the product of its width and height. Thereby, the blocks' placement and each terminal's location are determined.

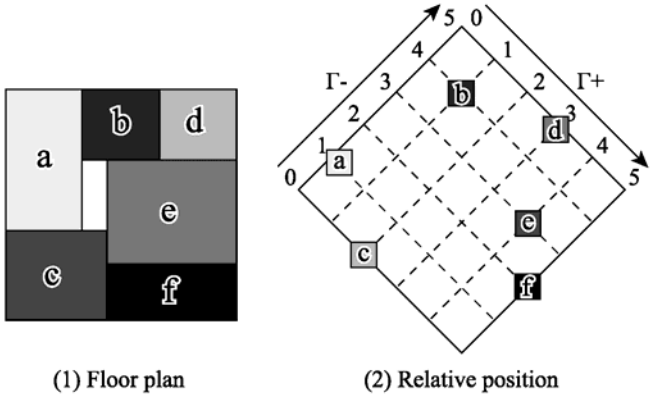



**Figure 1.** Example of sequence-pair for floorplanning.

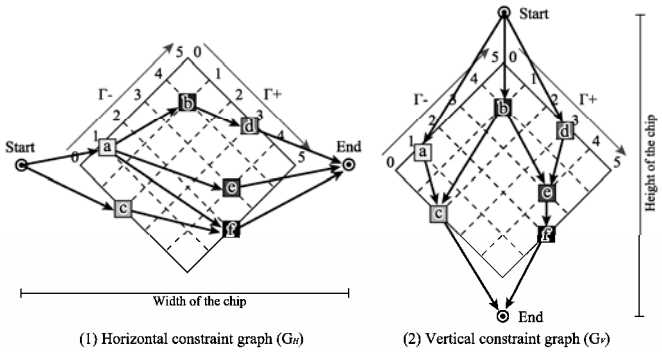



**Figure 2.** Example of the horizontal and vertical constraint graphs.

## 3. Floorplanning Algorithm

### 3.1. Coding

It needs to model for adopting GA to the optimization problem. This study expresses a soft block's shape by the aspect ratio (which is between 1.0 and 5.0). Therefore, an individual consists of S-pair $(\Gamma+, \Gamma-)$, representing the block's direction, and the block's aspect ratio (*ap*), as shown in Figure 3. That is, when the number of blocks is *n*, the gene length is 4*n* (S-pair, 2*n*; the direction θ, *n*; and the ap4, *n*). Here, four directions of a block exist (a block is rotated in 90 degree units), and a hard block's *ap* is set to 0. Thus, this coding enables to address the floorplanning in which hard and soft blocks are mixed.

### 3.2. Selection and evaluation

The selection operation is developed based on the idea of natural selection. That is, individuals with higher evaluation (higher fitness) have more descendants in the next generation than those with lower evaluation. The proposed algorithm introduces new evaluation function that considers both the bus wiring and the block with constraints, in addition to the area and total wire length, which already have been used to evaluate the conventional floorplanning.

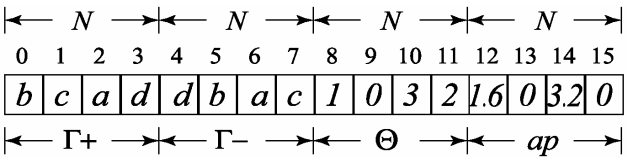



**Figure 3.** Example of coding.

#### 3.2.1. Constraints regarding bus wiring

Regarding bus wiring, it is important to shorten wire length and to reduce the number of bends, in consideration of the bit width of the bus wiring. These conditions regarding bus wiring are hereinafter referred to as the bus constraints. These bus constraints introduce new evaluative function which is based on the bus planning method [14]. The evaluative function divides the bus wiring into one trunk and several branches; and that evaluates the number of bends in bus wiring, as shown in Figure 4. A concrete evaluative procedure is shown below:

**Step 1.** The horizontal and vertical lengths of the smallest rectangle enclosing the blocks' terminals connected in bus wiring are represented by *X* and *Y*, respectively.

**Step 2.** The trunk corresponding to the bit width is drawn on all terminal's barycentric coordinates. The trunk is horizontal when $X \geq Y$, and vertical when $X < Y$. In the case shown in Figure 4, the trunk is drawn horizontally because $X \geq Y$.

**Step 3.** Each branch is drawn from each terminal to the trunk, and each branch's length $b_i$ ($1 \leq i \leq bm$, $bm$; $bm$ represents the total number of blocks connected to the particular bus) is obtained. The evaluative function $b_n$ for the number of bends utilizes the total number of branches other than 0.

**Step 4.** As Figure 4(2) shows, $l_i$ $(1 \leq i \leq bm - 1)$ represents the distance between blocks. However, when two blocks overlap, like $B_3$ and $B_4$ in this figure, $l_i$ represents the overlapped area's length.

**Step 5.** The evaluative function $bl(j)$ for the bus wiring $j$ is as follows:

$$bl(j) = \sum_{i=1}^{bm} b_i + \sum_{i=1}^{bm-1} l_i. \tag{1}$$

**Step 6.** After performing Steps 1 through 5 for all the bus wirings, the sum is obtained using the following equation. We determine the obtained sum as the bus wiring's evaluated value ($bs$).

$$bs = \sum_{j=1}^{k} bl(j). \tag{2}$$

Here, $k$ represents the total number of bus wirings.

### 3.2.2. Constraints for corner placement

Generally, I/F circuits, such as AD/DA and USB circuits, must be placed on the corner of a chip. The constraints for this corner placement are hereinafter referred to as placement constraints. New evaluative function $pl$ is introduced for these placement constraints, and it evaluates the distance between the chip's perimeter and the block with placement constraint. Specifically, as shown in Figure 5, when $m$ pieces of blocks with placement constraints exist, the evaluative function $pl$ is defined by equation (3):

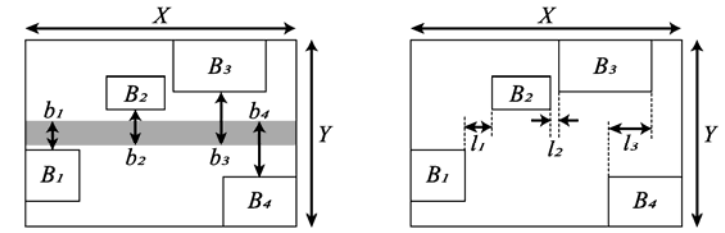
$$pl = \sum_{i=1}^{m} x(i) + y(i). \tag{3}$$

Here, $x(i)$ and $y(i)$ represent the distance between block $i$ (with the placement constraints) and the chip's perimeter in the $x$ and $y$ directions, respectively.

Moreover, the following linear sum of evaluative values is used for fitness $f(i)$ when the individual $i$ is actually selected.

$$f(i) = A + \alpha \times L + \beta \times bs + \gamma \times bn + \delta \times pl. \quad (4)$$

Here, $A$ represents the area; $L$ represents the wire's length; and $\alpha$, $\beta$, $\gamma$ and $\delta$ represent the real numbers' parameters.




(1) Example of calculation for *bi* (2) Example of calculation for *li*

**Figure 4.** Example of bus constraints.

### 3.3. Crossover

In order to realize floorplanning mixing hard and soft blocks, the proposed technique adopts two crossovers. One is a common topology-preserving crossover (CTPX) [8, 9] used for S-pair, and the other is a blend crossover (BLX-$\alpha$) [12] used for the aspect ratio. When CTPX is used, the locations and directions of blocks that have a common order in the locations of the parent S-pair are directly inherited. The other blocks inherit the order and directions of the sequence of the crossover target's parent. BLX-$\alpha$ generates a child individual according to a uniform random number within the area obtained by extending $\alpha \times d$ from the parent individual's real number vector. Figure 6 shows an example of BLX-$\alpha$.

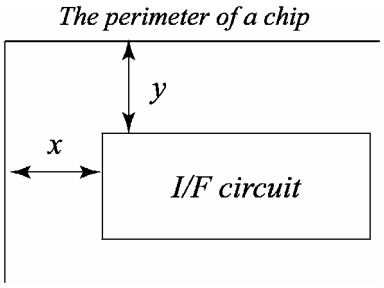



**Figure 5.** Example of placement constraint.

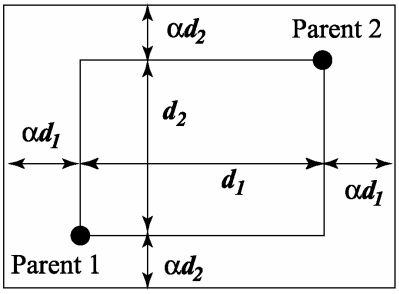



**Figure 6.** Example of BLX-α.

### 3.4. Mutation

Because mutation may recover lost alleles in an individual group (population), it is important for retaining the population's diversity. Therefore, the proposed technique introduces six mutations in order to realize effective mutation. Four kinds mutations (rotation, aspect ratio, Γ+ and Γ−) are introduced to apply to four elements which compose an individual; and two kinds mutations (Γ+ Γ− and pair swap) are introduced to greatly change the floorplan. In rotation mutation, a gene to represent the rotation is arbitrarily selected, and it is replaced by an allele. In aspect ratio mutation, the replacement similar to that used in rotation mutation is performed for the aspect ratio. In Γ+ mutation, two genes are exchanged for sequence Γ+, as shown in Figure 7.

In Γ− mutation, two genes are exchanged for sequence Γ−. In Γ + Γ− mutation, two genes are exchanged for both sequences Γ+ and Γ− . In pair swap mutation, two blocks are exchanged. Introducing the six kinds of mutations enables the lost gene's recovery in each locus. Here, the mutation operation is arbitrarily selected from these six kinds of mutations.

### 3.5. Hybrid search

Although the GA is superior in its ability to search globally to generate new search points, focusing on crossover operation, it is inferior in searching systematically at vicinity of the optimal solution. A local search is complementarily related to the GA. Therefore, a hybrid search, which combines a global search with a local search, can improve the searching ability. The proposed technique introduces a hybrid technique that combines the GA with three local search methods. These three concrete local search methods include a local search method for the relative locations

of blocks used in the paper [8, 9], one for rotation, and one for the aspect ratio. Then a local search is individually performed for each of the three elements (relative location, rotation and aspect ratio) that determine a floorplan. Regarding the local search for the aspect ratio, the aspect ratio of a soft block on the longest route in the graphs of horizontal and vertical constraints is changed to shrink the area. The local search for the aspect ratio is performed as follows:

**Step 1.** Graphs of horizontal and vertical constraints, $G_H$ and $G_V$, respectively, are generated for elite individual $x$; and a set of blocks on the longest route in each graph is represented by $B_H$ and $B_V$, respectively.

**Step 2.** Soft block $B_1 \in (B_H \cup B_V) - (B_H \cap B_V)$ is selected at random. The aspect ratio of $B_1$ is represented by $ap_{B_1}$. When $B_1 \in B_H$, equation (5) is used:

$$ap_{B_1} = ap_{B_1} + \eta. \tag{5}$$

When $B_1 \in B_V$, equation (6) is used. Using these equations generates new individual $x'$.

$$ap_{B_1} = ap_{B_1} - \eta. \tag{6}$$

Here, $\eta$ represents a real number parameter.

**Step 3.** When the evaluation value of newly generated individual $x'$ is greater than that of elite individual $x$, individual $x'$ replaces elite individual $x$.

**Step 4.** Operations described between Steps 1 and 3 are repeatedly.

If $x'$ does not improve during the predetermined number of rotations, then $\eta = \eta \times 0.5$ will be employed. Thus, shrinking the changeable part of the aspect ratio (i.e., shrinking the change in the block's shape) improves the performance of a local search.

On the other hand, the local search for rotation is performed similarly to one for the relative location. Specifically, Block $B_2 \in (B_H \cup B_V) - (B_H \cap B_V)$ is selected at random. The direction of $B_2$ is arbitrarily changed to generate new individual $x'$. When the evaluation value of newly generated individual $x'$ is greater than that of elite individual $x$, individual $x'$ replaces elite individual $x$.

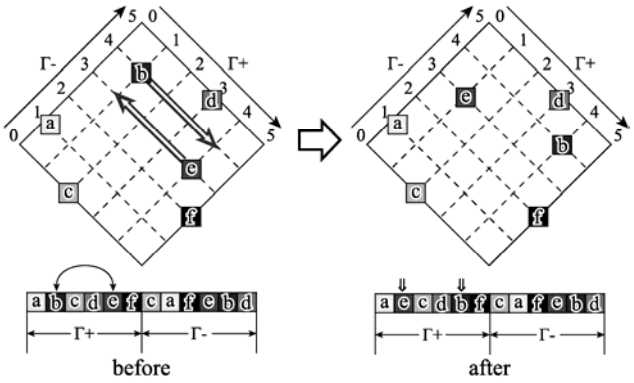



**Figure 7.** Example of Γ+ mutation.

## 4. Hardware Accelerator

### 4.1. Methodology

In order to achieve high-speed processing using a hardware (HW) accelerator, it is desirable that a portion of data with a large computational amount is processed using dedicated hardware. Preliminary experiments were performed to investigate the processing time in detail. Figures 8 and 9 show the detailed processing time obtained using two benchmark data. In both cases, the time required for evaluation exceeded 70% of the total time. Therefore, this study performs the evaluation using an HW accelerator. During evaluation, in order to calculate the wire length and placement constraints other than each individual's genetic information, the net list and the constraint conditions are required. Here, although each individual's genetic information used in the operation differs according to the generation because of the application of crossover and mutation, the same net list and constraint conditions are used in the operation.

The HW accelerator consists of an $I/F$ circuit, a memory controller, memory (SRAM), a calculation module, and an output control circuit. Here, the calculation module consists of *n* pieces of area operation circuits, *n* pieces of total wire length operation circuits, a bus constraints operation circuit, and a placement constraints operation circuit. The processing procedure can be explained as follows: (1) the data of the net list, the placement constraints, and the initial population sent from the software are stored in the HW accelerator's memory as pre-processing; (2) in each circuit in the operational block, the information required for the operation is read

from the memory, and the evaluation is performed; and (3) the obtained results are sent to the software through the output control circuit.

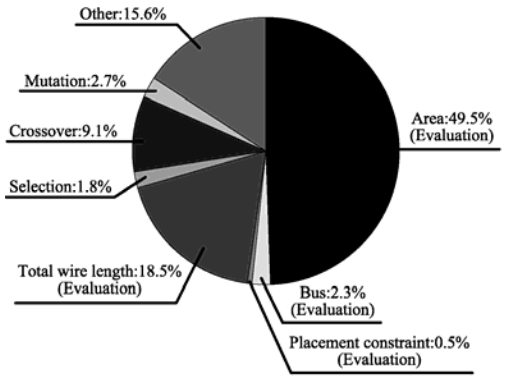



**Figure 8.** Example of detailed processing time of benchmark (ami33).

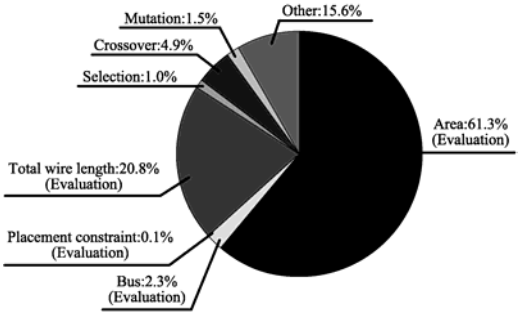



**Figure 9.** Example of detailed processing time of benchmark (ami49).

The communicative overhead grows as the data communication between the software and the HW accelerator increases. This communicative overhead increases the processing time. In order to reduce the communication traffic's volume, the net list and the constraint conditions, the values of which do not change through generations, are stored in the HW accelerator's memory as pre-processing. Thereby, the data communication between the software and the HW accelerator in each generation is performed only for an individual's genetic information; consequently, decreasing the communicative overhead. Figure 10 shows the processing flow using the HW accelerator.

### 4.2. Circuit implementation

The HW accelerator evaluates the area, total wire length, bus constraints and placement constraints. As Figures 8 and 9 show, the load of each processing step of

the evaluation operation varies widely. Although the processing loads of the area and the total wire length are large, those of the bus constraints and placement constraints are small. Each item is calculated using the circuit dedicated to that item. For example, the area with a large processing load is calculated using the circuit dedicated to that area. Therefore, the utilizing of hardware's characteristics increases the number of circuits dedicated to the item with a large processing load; several circuits for both the area and the total wire length are incorporated into the hardware. Thus, introducing the parallel operation prevents a stall in the evaluation, so high-speed processing can be realized. Figure 11 shows a circuit block diagram.

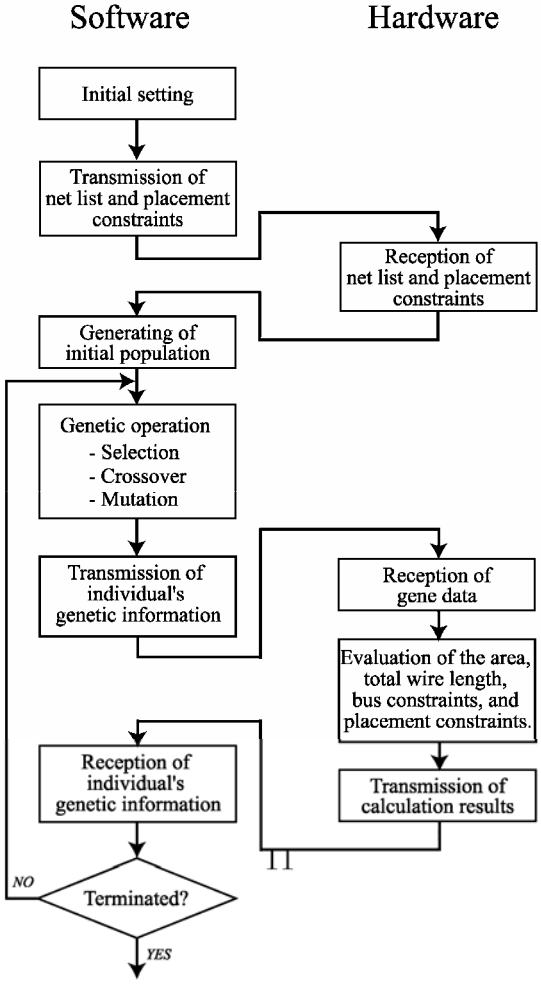



**Figure 10.** Processing flow using the HW accelerator.

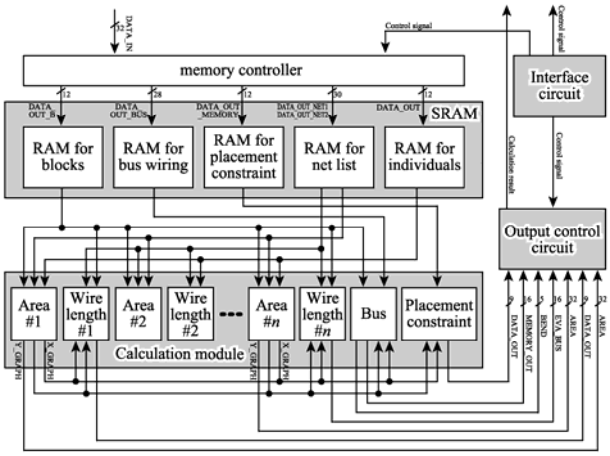



**Figure 11.** Circuit block diagram.

## 5. Experiments and Discussion

### 5.1. Experimental conditions

In order to verify the validity of the technique proposed in this paper, several comparative experiments were performed using the Microelectronics Center of North Carolina's (MCNC's) benchmark data, which have been used widely. Table 1 shows the benchmark data used in the experiments. Table 2 shows the additional data regarding soft block, bus wiring, and placement constraints. In Table 1, "Limit" expresses the area's lower limit. In Table 2, "#const." expresses the number of blocks to which the placement constraints were applied. Table 3 shows the parameters' values. For each parameter, the most effective value was set based on the processing time and the accuracy of the solutions obtained by the preliminary experiments. Moreover, software processing was described in C language and the HW accelerator was described in Verilog-HDL. Regarding the execution platform, Pentium IV was used for software processing and Virtual Turbo PCI with Virtex-IV was used for the HW accelerator.

**Table 1.** Benchmark data

| Data | # blocks | # nets | Limit |
|---|---|---|---|
| ami33 | 33 | 123 | 1.156 |
| ami49 | 49 | 408 | 35.445 |

**Table 2.** Additional data for benchmark data

| Data | # soft blocks | # bus routing | # const. |
|---|---|---|---|
| ami33 | 17 | 4(4bit × 2, 8bit, 16bit) | 2 |
| ami49 | 30 | 5(4bit × 2, 8bit × 2, 16bit) | 2 |

**Table 3.** Parameters

| Name | Value | Name | Value |
|---|---|---|---|
| Population size | 50 | $\alpha$ | 0.001 |
| Generation | 10,000 | $\beta$ | 0.01 |
| Tournament size | 2 | $\gamma$ | 2 |
| Crossover ratio | 0.6 | $\delta$ | 0.05 |
| Mutation ratio | 0.05 | $\eta$ | 0.1 |

### 5.2. Evaluation of the basic algorithm's performance

In order to evaluate the performance of the algorithm used for floorplanning, comparative experiments were performed using software. Tables 4 and 5 show those experimental results. The value in both tables is the average value of 10 trials. Moreover, techniques (a) and (b) in both tables were used to evaluate only the area and the area and bus constraints, respectively. The comparison between techniques (a) and (b) reveals that the evaluation function used for the bus constraints shortened the bus wiring and reduced the number of bends. Similarly, the comparison between technique (b) and the proposed technique reveals that the proposed technique systematically improved four evaluations; area, bus constraints, placement constraints and total wire length.

### 5.3. Evaluation of the HW accelerator's performance

In order to evaluate the HW accelerator's performance, the accelerator was implemented on a FPGA. Figures 12 and 13 show comparisons of the processing time between the software and the HW accelerator.

**Table 4.** Experimental result of ami33 (software processing)

| Technique | (a) | (b) | Proposed |
|---|---|---|---|
| Area | 1.23 | 1.36 | 1.52 |
| Bus length | 1398.8 | 43.5 | 72.2 |
| # bends | 9.6 | 0.2 | 0.3 |
| Place const. | 698.3 | 486.1 | 0 |
| Total wire length | 113.8 | 84.2 | 65.7 |
| Time (s) | 48.8 | 50.3 | 63.3 |

**Table 5.** Experimental result of ami49 (software processing)

| Technique | (a) | (b) | Proposed |
|---|---|---|---|
| Area | 38.0 | 41.2 | 42.3 |
| Bus length | 9825.2 | 325.4 | 238.5 |
| # bends | 11.8 | 0.4 | 0.7 |
| Place const. | 3524.2 | 3233.6 | 56.8 |
| Total wire length | 1916.6 | 1933.5 | 1422.4 |
| Time (s) | 154.3 | 165.6 | 205.3 |

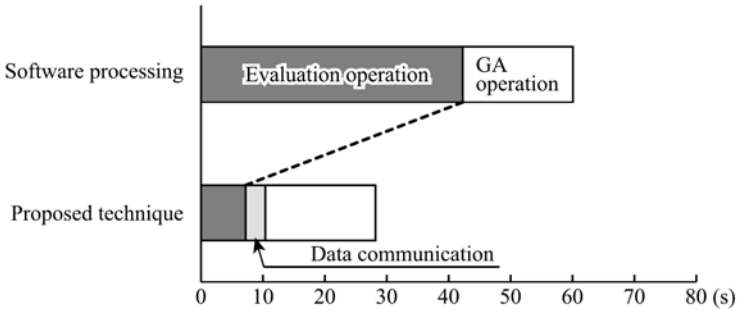



**Figure 12.** Comparison of processing time on ami33.

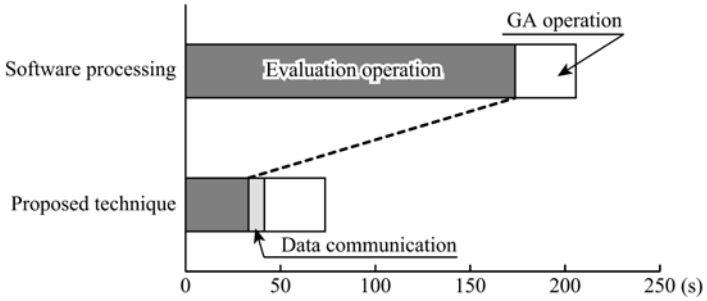



**Figure 13.** Comparison of processing time on ami49.

Although the HW accelerator's communication time was longer than the software's, the HW accelerator's total processing time decreased by about 50% and 70% compared to the software's, when ami33 and ami49 were used, respectively. Although the HW accelerator used an operating frequency of 33 MHZ because of the FPGA's constraints, using a faster FPGA could speed up the processing. Table 6 shows the results of the floorplanning obtained using the HW accelerator. Comparing these results to those obtained using the software demonstrates that the

HW accelerator achieves floorplanning with an accuracy equivalent to that obtained using software in all the evaluations of the area, total wire length, bus constraints and placement constraints. Figure 14 shows the layout results obtained using the proposed technique. As this figure demonstrates, the proposed technique achieves not only bus wiring with no bends and a short wire, but also the perimeter placement of blocks with the placement constraints.

## 6. Conclusion

This study proposed the floorplanning technique based on the GA. In order to shorten the processing time, a new HW accelerator was developed. The HW accelerator realized the interlocking processing using software and dedicated hardware, and it also achieved high-speed processing while retaining the high quality of solutions. Furthermore, the HW accelerator was implemented on the FPGA, and experiments using benchmark data verified its validity.

**Table 6.** Experimental result by HW accelerator

| Data | ami33 | ami49 |
|---|---|---|
| Area | 1.46 | 44.7 |
| Bus length | 58.7 | 138.2 |
| # bends | 0.3 | 0.7 |
| Place const. | 0 | 132.7 |
| Total wire length | 68.9 | 1367.4 |

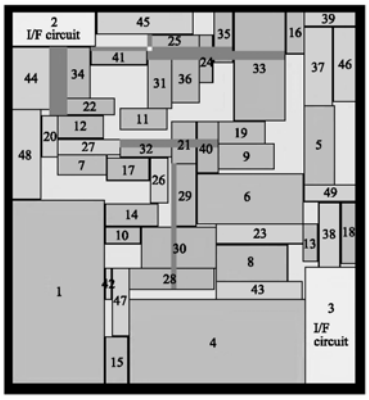



**Figure 14.** Layout result by HW accelerator (ami49).

Our future work is to develop the parallel operation of the accelerator and software processing. The interlocking system proposed in this paper does not perform software processing while the accelerator performs calculations. We will develop a new parallel interlocking system, which performs software processing while calculation is performed in the accelerator. Moreover, we will develop a design platform for systematic floorplanning, including power supply wiring.

## References

[1] J. Cong, Z. Pan, L. Hei, C. K. Koh and K. Y. Khoo, Interconnect design for deep submicron ICs, Dig. Tech. Papers ICCAD, 1997, pp. 478-485.

[2] K. Bazargan, S. Kim and M. Sarrafzadeh, A floorplanner of uncertain designs, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 18(4) (1997), 389-397.

[3] H. Murata, K. Fujiyoshi, S. Nakatake and Y. Kajitani, VLSI module placement based on rectangle-packing by the sequence-pair, IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems 15(12) (1996), 1518-1524.

[4] S. M. Sait and H. Youssef, VLSI Physical Design Automation, IEEE Press, 1995.

[5] J. Holland, Adaptation in Natural Artificial Systems, 2nd ed., MIT Press, University of Michigan Press, 1992.

[6] D. E. Goldberg, Genetic Algorithms in Search Optimization, and Machine Learning, Addison Wesley, 1989.

[7] Y. Shigehiro, S. Yamaguchi, M. Inoue and T. Masuda, A genetic algorithm based on sequence-pair for floorplan design, T. IEE Japan 121-C(3) (2001), 601-607.

[8] S. Nakaya, T. Koide and S. Wakabayashi, An adaptive genetic algorithm for VLSI floorplanning based on sequence-pair, Proc. IEEE International Symposium on Circuits and Systems 3 (2000), 65-68.

[9] S. Nakaya, T. Koide and S. Wakabayashi, A VLSI floorplanning method based on an adaptive genetic algorithm, IPSJ J. 43(5) (2002), 1361-1371.

[10] T. Imai, M. Yoshikawa, H. Terai and H. Yamauchi, Scalable GA-processor architecture and its implementation of processor element, Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing 3 (2002), 3148-3151.

[11] P. Graham and B. Nelson, Genetic algorithms in software and in hardware - a performance analysis of workstation and custom computing machine implementations, FPGAs for Custom Computing Machines, 1996, pp. 216-225.

[12] L. J. Eshelman and J. D. Schaffer, Real-coded genetic algorithms and interval schemata, Foundations of Genetic Algorithms 2 (1993), 187-202.

[13] M. Yoshikawa and H. Terai, Bus-oriented floorplanning technique using genetic algorithm, WSEAS Trans. on Circuits and System 2(6) (2007), 253-258.

[14] Hua Xiang, Xiaoping Tang and M. D. F. Wong, Bus-driven floorplanning, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems 23(11) (2004), 1522-1530.