Guiding the Standard-Cell Placements for Regular Layout Structure Using Virtual Net

Yongseok Yi, In-Cheol Park
Electrical Engineering and Computer Science Department, KAIST
Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
+82-42-869-4405
Email: yslee@ics.kaist.ac.kr, icpark@ee.kaist.ac.kr

Abstract —*In this paper, we present a new standard-cell placement technique to achieve a regular layout structure, mostly for datapath circuits. Virtual nets are introduced to connect the cells that the designer wishes to align in line to achieve a regular layout structure. We have incorporated the processing of virtual nets into the partitioning-driven placement. To apply the virtual nets, the cell-gain computing part of the original partitioning algorithm is modified without increasing the run-time. Experimental results prove that the virtual net is an effective method to shape the placement of standard cells.*

I. Introduction

The first objective of the placement problem is usually to minimize the total wire length, which leads to less area and better circuit performance. In turn, most of the algorithms concerning this problem try to gather the cells that are connected to a net closely.[1] The stronger the cells are connected, i.e. connected by more nets, the more probable are they placed closely. This phenomenon is quite desirable for general circuits, except for datapaths.

In datapaths, cells are connected to form bit-slices, paths for a bit of data to flow. Generally, cells that lie in a bit-slice are not connected tightly to the cells in other bit-slices. If the conventional placement algorithms are applied to this style of circuits, the cells in a bit-slice are usually placed closely to form a bundle of cells as illustrated in Figure 1, which bears certain problems, among which the time skew is the mostly encountered one. This is the most important reason why many designers want to draw the datapath layouts in full-custom style while entrusting the layout of other parts of the chips (control logics, etc.) to automated placement and routing procedures.

To solve this problem and exploit the benefits of the bit-sliced structure [2], many researches have been conducted. They can be classified into two parts: regularity extraction [3][4][5] and regular placement [2][6][7][8][9].

This paper concerns about the latter one while assuming the regularity of the circuit is already acquired. A drawback of the previous approaches is that they focus only on the datapath-styled circuits, not embracing other general circuits.

The proposed approach is to guide the placement of cells by letting the placement tool know which cells should be placed

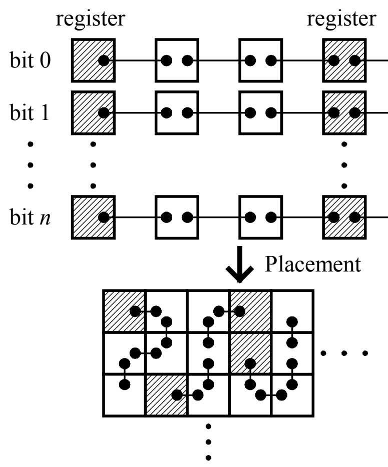


Figure 1. The placement of datapath cells using conventional placement procedures

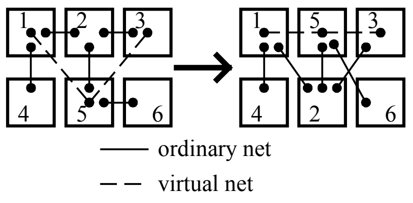


Figure 2. The role of the virtual net

in line. For this purpose, the concept of the *virtual net*, which does not have any electrical significance but only affects the placement procedure, is introduced.

The rest of the paper is organized as follows: After Section 2 introduces the virtual net and the regarding considerations, the application of the virtual net to a specific placement algorithm is explained in Section 3. The experimental results are described in Section 4.

II. Virtual Net

A *virtual net* is defined as a net that connects the cells that are desired to be aligned in one direction, on a single line, if possible. In Figure 2, a virtual net is introduced to align 3 cells {1, 3, 5}, horizontally. We also define a *guided cell* to designate the cell connected to virtual net. A virtual net does not have any electrical significance.

The use of virtual net is not restricted to a specific placement technique because the incorporation of virtual net exploits the algorithm's effort to minimize the wire lengths via minimal modification. The modification increases the probability of guided cells to move, so that they are aligned to the desired direction by overcoming the effects of other real nets incident to the guided cells.

A. Connection of Virtual Nets

This subsection discusses the necessary considerations about connecting the guided cells with virtual nets. The supporting comparison results are presented in Section 4.

Direction In the case of datapath circuits, assuming that the data flow horizontally and the control signals flow vertically, one may think of two different ways to guide the cell placement using the virtual nets: (1) connecting the equivalent cells of distinct bit-slices, in which the virtual net lies vertically, or (2) connecting the cells in a bit-slice, in which the virtual net lies horizontally.

Position Determining on which cells the virtual nets be connected is closely related to the capability of the used placement algorithm. For example, assuming a set of pipeline registers corresponding to a certain pipeline stage are guided by a virtual net, it is hard to guarantee that the registers corresponding to the adjacent stage are also aligned in a similar way because the unguided cells between the two sets of registers are placed more irregularly as the logic distances from the guided register cells increase. However, imposing virtual nets to cells overly may lead to less possibility to take advantage of the virtues of the original placement mechanism.

B. The Intensity of Forcing

As mentioned above briefly, a virtual net imposes an excess value of cost factor, the amount of improvement when a cell moves. The cost factor could be the cell gain in partitioning-based placement, the temperature and moving distance in simulated annealing-based placement, or the attractive and/or repulsion constant in force-directed placement.

As will be illustrated in Section 3 and Section 4, the amount of the excess cost depends on how the processing of virtual net is incorporated into the placement algorithm. Experiments reveal that the effect is saturated on a certain value of the excess cost factor, so we can determine the value empirically.

III. Application to Placement

This section describes how to apply the virtual net technique to a specific placement algorithm. We chose the partitioning-based placement, in which the min-cut partitioning is done using the FM algorithm [10] combined with the min-cut shuffle [11] technique.

Unlike the conventional min-cut placement whose purpose is to minimize the half-perimeter length, since the purpose of the virtual net is to force the connected cells' positions to a narrow vertical region, the virtual net is given a tuple of two weights, (vertical weight, horizontal weight). To fulfill the purpose, it is necessary to confine the guided cells in one partition block only when the cut line is vertical. This is achieved by adding excess gains (henceforth referred to as *vn_weight*) to the guided cells that are candidates to cross the vertical cut line, thus increasing the possibility of moves. The calculation of cell gains should be different from that of the original FM algorithm because, in the original algorithm, a cell is given a gain only when it has a critical net. In contrast, regardless of whether a virtual net is critical or not, the connected cells on either side must move.

In the beginning of each level, the gain of a guided cell *i* with respect to the corresponding virtual net *n* is computed as follows.

$$gain(i) = gain(i) + \left(T(n) - F(n)\right) \times vn\_weight$$

, where *T*(*n*) is the number of cells on a virtual net *n* in the "to" block and *F*(*n*) is the number of cells on *n* in the "from" block. Note that it is assumed that *gain*(*i*) due to the non-virtual nets has already been computed, and the value of *vn-_weight* should be used selectively according to the partition mode (vertical or horizontal). The equation above is indeed the majority rule: the cells belonging to a side with fewer neighbors are forced to move.

Updating a guided cell's gain after a tentative move is also different from that of a non-guided cell. When a guided cell moves across the cut line, the gain of all other free neighbors with respect to the corresponding virtual net is updated as follows.

$$gain(i) = \begin{cases} gain(i) + 2 \times vn\_weight, \text{ if cell } i \text{ is on the "from" side} \\ \\ gain(i) - 2 \times vn\_weight, \text{ if cell } i \text{ is on the "to" side} \end{cases}$$

The complexity is $O(n)$, where *n* is the number of cells, and $O(1)$ for gain computation and gain update, respectively. Thus it does not deteriorate the complexity of the original FM algorithm.

IV. Experimental Results

We implemented a placement tool that supports the virtual net processing. It behaves as a conventional placement tool when a virtual netlist is not supplied.

A. Connections

Direction Figure 3 (a) depicts the placement in which the vertical virtual nets are applied in orthogonal direction to the bit-slices whereas Figure 3 (b) depicts the contrary. The thick dotted lines represent the virtual nets and the solid lines represent parts of the real nets. In both figures, the virtual nets show good shapes as intended. However, compared to Figure 3 (a), in which the horizontal real nets are also relatively well aligned, Figure 3 (b) shows that the vertical real nets are not aligned desirably. This is because the non-virtual nets also play an important role in the placement.

Position Figure 4 (a) shows the increasing irregularity of cell positions as the distances of the cells from the guided cells increase. Setting another virtual net rectifies the irregularity as depicted in Figure 4 (b).

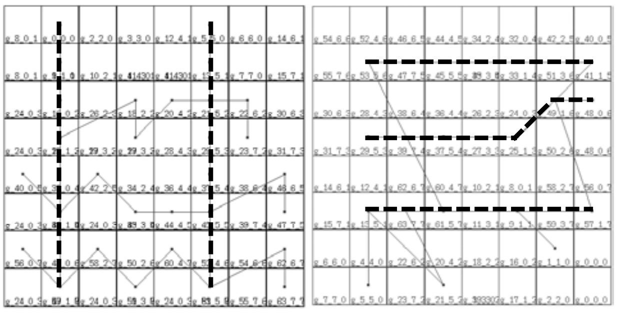

(a) Vertical virtual net processed (b) Horizontal virtual net processed

Figure 3. Different directions of virtual nets

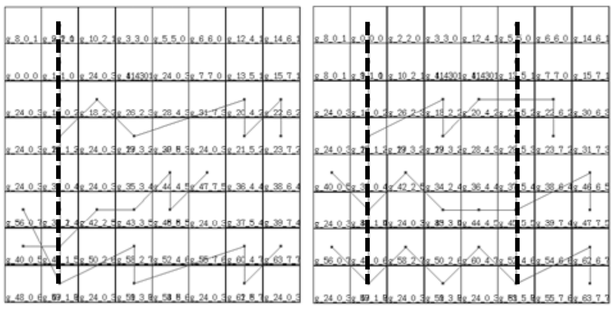

(a) One virtual net (b) Two virtual nets

Figure 4. The effect of virtual net positions

B. Amount of the Excess Gain

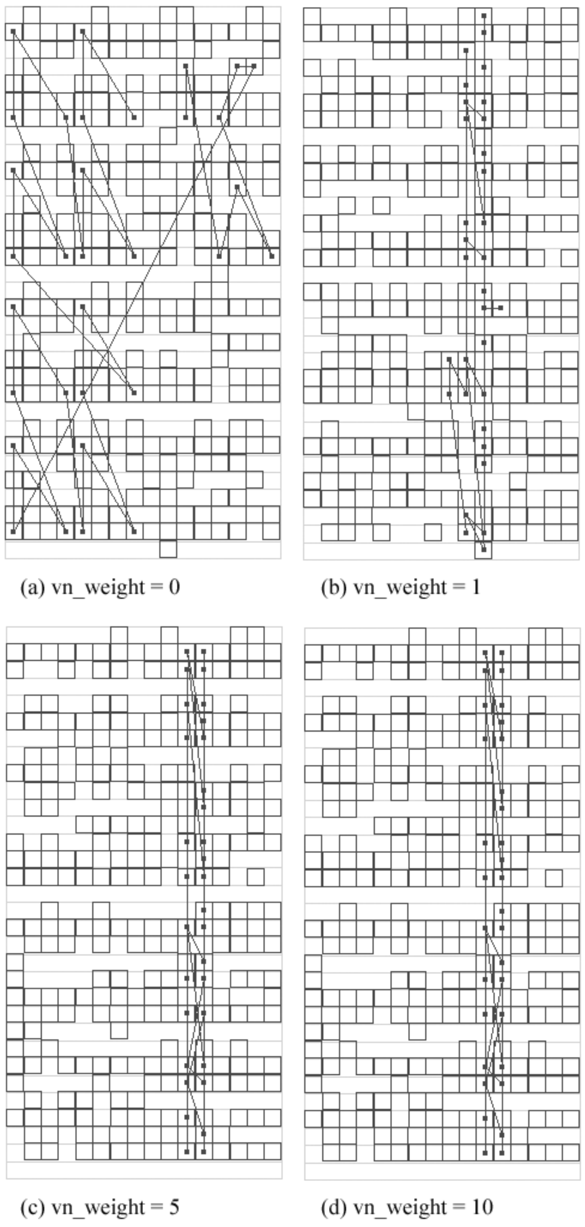
Figure 5 illustrates an example of how the cells' positions change as the vn_weight value changes. The circuit is a simple 32-bit datapath consisting of 287 cells and 383 nets. In the figures, the wires going back and forth upon the cells represent the virtual nets.

Figure 5 (a) is an example when the virtual net technique is not applied. We bind 32 cells with a virtual net for this experiment. Increasing the vertical vn_weight by only 1 drastically affects the cell positions (Figure 5 (b)). Setting vn_weight to 5 makes the region occupied by the virtual net narrower (Figure 5 (c)), which does not improve even if the value is doubled (Figure 5 (d)).

C. Real Application

The implemented tool has been applied to the datapath circuit of a 32-bit RISC processor with 14,831 cells and 14,908 nets. An abstract block diagram of the datapath is depicted in Figure 6, in which the positions of the virtual nets are indicated using dotted lines. The virtual nets have been set to the output mux cells of the register files, 2 sets of pipeline registers and the output cells of the barrel shifter. The vertical vn_weight is set to 5 and horizontal vn_weight to 0.

Figure 7 illustrates the corresponding result. The vertical virtual nets are accentuated using thick dotted lines and the real nets with thick solid lines. The displayed real nets represent the identical series of cells for bit-slices corresponding to even bits (bit 0, bit 2, …, bit 30). As can be seen in the figure, the bit-slices of the datapath are well aligned horizontally, and the guided cells connected to a vertical virtual net are aligned vertically as intended.


(a) vn_weight = 0 (b) vn_weight = 1

(c) vn_weight = 5 (d) vn_weight = 10

Figure 5. Effects of varying vertical excess gains

V. Conclusion

We presented a technique that employs the virtual net to achieve a regular layout structure in standard cell placement. The virtual net is used to guide the connected cells to be placed in either vertical or horizontal narrow regions by imposing excess costs to the cells. The presented technique has been incorporated into the min-cut placement, in which the gain computation part has been modified to reflect the excess gains. The effect of the virtual net has been illustrated based on experimental results. The virtual net technique, however, can be incorporated into other placement algorithms as well.

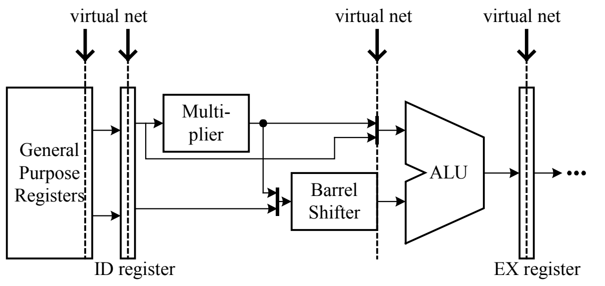


Figure 6. The block diagram of the 32-bit RISC datapath

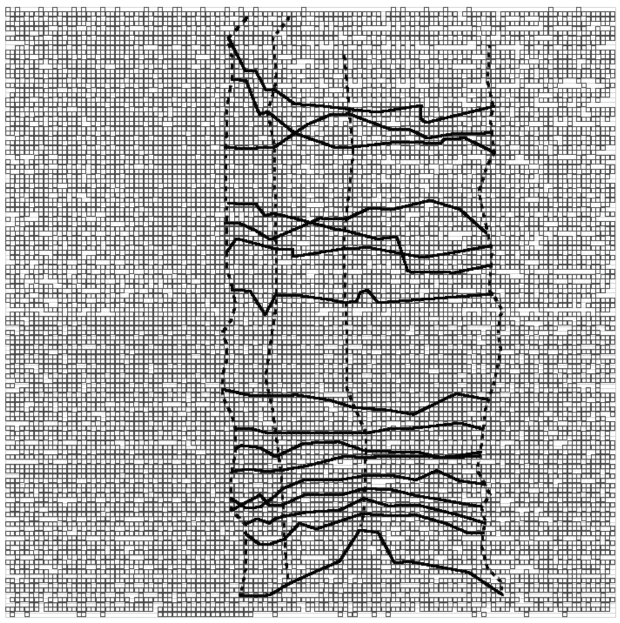


Figure 7. The effects of virtual nets on 32-bit RISC datapath circuit

ACKNOWLEDGMENT

This work was supported by the Korea Science and Engineering Foundation through the MICROS center, by the Ministry of Science and Technology and the Ministry of Commerce, Industry, and Energy through the project System IC 2010, and by IC Design Education Center (IDEC).

REFERENCES

[1] K. Shahookar and P. Mazumder, “VLSI Cell Placement Techniques,” ACM Computing Surveys, Vol. 23, No.2, June 1991, pp. 143-219.

[2] T. T. Ye and G. De Micheli, “Data Path Placement with Regularity,” Proceedings of the IEEE International Conference on Computer-Aided Design, 2000, pp.264-270.

[3] S. R. Arikati and R. Varadajan, “A Signature Based Approach to Regularity Extraction,” Proceedings of the IEEE International Conference on Computer-Aided Design, 1997, pp.542-545.

[4] R. X. T. Nijasen and C. A. J. van Eijk, “Regular Layout Generation of Logically Optimized Datapaths,” Proceedings of the International Symposium on Physical Design, 1997, pp.42-47.

[5] S. Hassoun and C. McCreary, “Regularity Extraction via Clan-based Structural Circuit Decomposition,” Proc. IEEE Int. Conf. Computer-Aided Design, 1999, pp. 414-418.

[6] W. K. Luk and A. A. Dean, “Multistack Optimization for Datapath Chip Layout,” IEEE Trans. Computer-Aided Design, Vol. 10 1, Jan. 1991, pp. 116-129.

[7] H. Cai, S. Note, P. Six and H. de Man, “A Data Path Layout Assembler for High Performance DSP Circuits,” Proc. ACM/IEEE Design Automation Conf., 1990, pp.306-311.

[8] N. Buddi, M. Chrzanowska-Jeske and C. L. Saxe, “Layout Synthesis for Data-path Designs,” Proc. European Design Automation Conf., 1995, pp. 86-90.

[9] J. –S. Yim and C. –M. Kyung, “Data Path Layout Optimisation Using Genetic Algorithm and Simulated Annealing,” Proc. IEEE Computers and Digital Techniques, Vol. 145 2, Mar. 1998, pp. 135-141.

[10] C. M. Fiduccia and R. M. Mattheyses, “A Linear-Time Heuristic for Improving Network Partitions,” Proc. ACM/IEEE Design Automation Conf., 1982, pp. 175-181.

[11] I. Bhandarl, M. Hirsch and D. Siewiorek, “The Min-Cut Shuffle: Toward a Solution for the Global Effect Problem of Min-cut Placement,” Proc. ACM/IEEE Design Automation Conf., 1988, pp. 681-68.