Regular Fabrics In Deep Sub-Micron Integrated-Circuit Design

Fan Mo and Robert K. Brayton
University of California, Berkeley

Abstract — Several regular circuit structures are proposed. They provide alternatives to the widely used standard-cell structure and have better predictability and a simpler design methodology. A regular global routing scheme is developed. A design flow for use with of all these regular fabrics is discussed.

1. Introduction

Regularity is a feature, which can better guarantee that the layout designed by a CAD tool is exactly replicated in the fabrication. The limits of the mask making system are reached as the geometries get smaller. Accompanied by the optical phase shift, coherence etc., the actual masks may not look exactly the same as what a CAD tool produced [2]. For instance, the patterns after the lithography may have unexpected shapes such as collapsed edges and rounded corners. What a CAD tool can do to tackle the problem is to add some pre-distortion to offset the real distortions that happen during the fabrication. However the number of layout patterns generated by a conventional design flow might be too large to get this job done within a reasonable amount of time.

Another point is also related to the progress of the processing technology. Whenever a technology is migrated to an advanced one with smaller feature sizes, the whole cell library has to be rebuilt. The reuse of the old library is almost impossible, because factors like cell speed, cell, power etc. are not scaled proportionally. Building a library with hundreds of gates involves layout design, parameter extraction, SPICE analysis, design rule checking and documentations, which is expensive and time consuming. Regular circuit structures, or non-cell-based structures, need less library-rebuilding work.

The third point is the well-known timing closure problem, which arises from the fact that the design flow contains a set of sequential steps. Earlier steps have to predict what the later steps will do. Inaccurate prediction may lead to wrong decisions, which can only be recognized later. When this happens, design iteration is necessary. For instance, at the synthesis stage, the wire delay is not known. So the synthesizer has to estimate wire length based on probability or some other models. When the synthesized circuit is passed to the physical design tool, it might become clear some wire lengths are incorrectly estimated. Unfortunately this is not the end of the nightmare, because using the "exact" wire length just derived from the routed design to control a new round of synthesis does not guarantee the same layout, and thus not those wire lengths. To prevent this type of iteration is difficult, but it is possible to make the estimation more accurate by using special components such as the regular structures.

A Programmable Logic Array (PLA) is a regular structure, in the sense that its layout is composed of regular patterns. In addition, its area and delay are directly related to the logic functions it represents. The result of a Sum-of-Products (SOP) minimization can be mapped directly to a PLA [6,8]. Technology mapping, required in standard-cell designs, is not necessary; neither are placement and routing necessary for a single-PLA circuit. The PLA structure is also library free. To represent more complex logic, a multi-level structure is needed [5,7]. In multi-level logic minimization, the entire circuit is represented as a network of nodes where each node is a SOP logic function. A common approach is to optimize both the entire network as well as each node, and then transform the circuit to a network of library cells via technology mapping. A natural step is to build a network of PLAs (NPLA) from the minimized network without technology mapping [4]. Thus some desirable features of single-PLAs such as technology-mapping-free synthesis are preserved. But NPLAs require placement and routing and the placement of the PLAs is at the block level, and block level placement is not as well developed as gate level placement.

Three structures are presented, maintaining the regularity of the single-PLA and eliminating the routing irregularity of the NPLA: Whirlpool-PLA (WPLA), River-PLA (RPLA) and Four-River-PLA (FRPLA). All these structures share the following advantages:

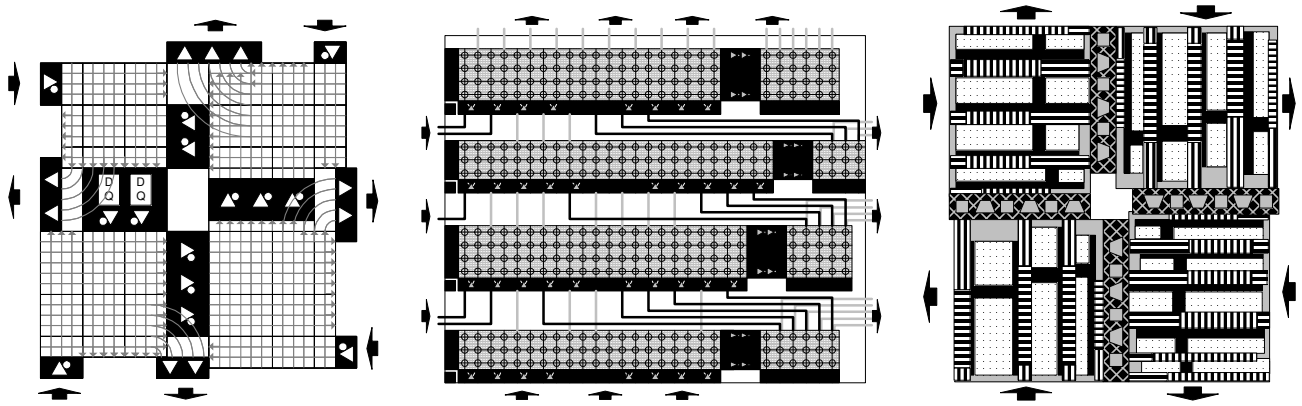
1. Regularity.
2. Well-buffered inputs and outputs.
3. Less metal layers as compared to standard-cell and NPLA implementations.
4. No placement and routing.

The difference between these structures is the size of circuit they can implement. WPLA can handle about 1K-gate circuits, RPLA can handle 10K-gate circuits, and FRPLA can handle 50K-gate circuits. For future Deep Sub-Micron (DSM) designs in which regularity may be a key issue [3], their regularity should be an advantage.

To implement even larger circuits, multiple regular blocks should be interconnected. Conventional routing is feasible but has poor wire delay predictability. Accurate delays of the global wires can only be derived after detail routing. Even at the early stages of routing, exact delays are not known. Our main idea is to pre-define the net topologies and maintain them throughout the design flow.

A regular global routing scheme called the Fishbone Routing Scheme (FBRS) is developed, which provides a regular and highly predictable routing structure. The critical nets are routed with the FBRS. In the two-pin net case, this reduces to an L-shape connection. To prevent one critical net from blocking the routing of another, a grid rule is enforced on the positions of the critical pins (pins connected to the critical nets are called the critical pins) such that no two critical nets will compete for the same routing track. When the floorplan is completed with the FBRS, all the critical nets are finalized. The feasibility of the FBRS is supported by the following observations. First, we focus on the critical nets and their quantity is limited. Second, not all pins of a block are critical, so the number of critical pins may not be dominated by Rent's rule. Third, the routing resources are sufficient, because global wiring density is not as high as gate-level wiring density. Therefore routability becomes less of an issue, while the critical wire delays are the most important. The FBRS gives us the ability to get accurate wire delays during the floorplanning; since the critical net delays planned by the floorplanner will be preserved in the design steps that follow, such as the routing of non-critical nets.

Buffer insertion is a powerful means for reducing wire delays [10], cross talk, and reduction of Miller effects. Unlike in the purely standard-cell designs where the buffers can be placed anywhere in the layout, at the block-level, there are two categories based on how the buffers are placed. The first is to place buffers in the gaps between the blocks. The second is to reserve buffers in the blocks, which are ready to serve any nets crossing them. Existing methods mostly belong to the first category and they are also a post-floorplanning process. However, buffers inserted between the blocks may increase the chip area, which may in turn change the wire lengths. Thus the wire delays are changed, making delay prediction



(a) Whirlpool PLA **(b) River PLA** **(c) Four-River PLA**

Figure 1. Three regular circuit structures.

more difficult. In addition, to insert a buffer, vias need to be introduced to bring the net, most probably running on higher metal layers, down to the substrate for buffering and bring the buffered signal back to higher layers [11]. The number of vias used to connect to the buffer, and thus the resistance, needs to be predicted. Further, the vias may cause routing congestion, which may change the topologies of other nets. Finally, it is not rare to see blocks with dimensions of 2mm or more, so any critical wire crossing such blocks may need buffering right within the block area. This means that in block-level buffer insertion, the second category becomes necessary. The FBRS provides a means of using blocks with built-in buffers and plans buffer insertion during the floorplanning. The built-in buffer scheme can be viewed as a guide to IP block designs, just like scan chain in the area of testing.

The rest of the paper is organized as follows. In Section 2, the regular circuit structures are described. In Section 3, the regular global routing scheme and the corresponding buffer scheme are presented. A discussion is given in Section 4.

2. Regular Circuit Structures

2.1. Whirlpool-PLA

A WPLA is a cyclic four-level programmable array, as shown in Figure 1(a). The cascaded NOR (or NAND, if the intermediate buffers/inverters are included) structure allows binary inputs to each plane, extending the conventional SOP form. The four programmable planes, labeled 0, 1, 2 and 3, are organized in a loop. In each plane, input signals consist of external inputs as well as outputs from the preceding plane. Placing latches between planes 3 and 0 breaks any combinational loops. WPLA circuits consume only 2 metal layers.

The area of a WPLA is totally determined by the embedded logic functions. The delay formulation shows that smaller area usually means smaller delay, partially because the number of levels is fixed, and uniform buffering is used in WPLAs. In a standard-cell design, collapsing nodes on a critical path can reduce the logic levels in the hope of reducing delay, essentially introducing more parallelism. However, real gates have limited driving capacities. Increasing parallelism means larger loading; hence buffers are inserted, or driving gates are duplicated or resized. Inserting buffers may introduce additional delays; duplicating gates actually shifts the load burden backwards. In addition, such timing optimization may trigger an unexpected blow-up in area. Placement and routing factors may further complicate the problem. When a standard-cell implementation does not meet delay requirements, it is difficult to decide whether and how to continue collapsing the circuit. The WPLA synthesis approach has no such scenario. An algorithm called *Doppio-ESPRESSO* is developed to synthesize logic into WPLAs. The basic idea of WPLA synthesis is to minimize a pair of NANDs at a time, and iterate for different pairs until no further improvement. The possible pairs in the WPLA are 0-1, 1-2 and 2-3. The minimization of a pair of NANDs differs from the conventional SOP minimization [6], since the WPLA structure allows negative products. *Doppio-ESPRESSO* uses this extra structural flexibility for further optimization.

Experiments show that standard-cells (SC) generally have larger areas than WPLAs, but SCs can provide smaller delays if more area is allowed. NPLAs are just the opposite; they can provide smaller (raw) areas, but usually are slower. Comparing the level=4 cases, on average, WPLAs are 37% smaller than SCs and 0% smaller than NPLAs, but only 5% and 3% slower than SCs and NPLAs. However, in these studies, the areas of SCs and NPLAs only account for the raw area of the logic components and use more metal layers. After placement, the areas of both are expected to grow, especially NPLAs. Comparing the areas of SC, NPLA and WPLA with similar delays (may have different number of levels), we find that WPLA is on average 19% larger than NPLA (raw area), and 26% smaller than SC.

2.2. River-PLA

A RPLA consists of a stack of multiple-output PLAs, as shown in Figure 1(b). For each PLA in the stack, the input signal to its AND-plane come from the bottom and left side, while its output signals contains those generated in the OR-plane and those by-passed through the AND-plane. The output signals feed the next PLA in the stack or exit the RPLA at the right side. All connections between PLAs are made via two-layer river routing.

The area and delay of a RPLA are both directly expressed in terms of the PLA contents, so the objective function in the optimization can be evaluated with good accuracy. The height of a RPLA is the sum of the heights of the PLAs, and the width is the largest width. The non-uniformity of the PLA widths results in "white" space on the right side of those PLAs whose widths are smaller than the RPLA width. Since buffers isolate the PLAs in the stack, the delay computation is just a summation of the delays of all the PLAs. In delay analysis, accuracy depends only on the delay model, because given a synthesized PLA, there is nothing unknown such as fanout number etc. The delays of the river-routing wires are small because they only form the local connections. So as the circuit is synthesized, the wire delay can be safely ignored. The design flow for the RPLA contains three steps: multi-level logic minimization,

node placement (node leveling)[1] and signal ordering. The multi-level logic minimization uses SIS [5]. Node level-placement is possible because some nodes have flexibility in their levels. However the flexibilities of different nodes may be correlated, because of the fanin/fanout relations. Simulated-annealing is suitable for the node placement task, since the evaluation of area and delay is straightforward. The objective function is chosen as a weighted sum of the area and delay. In the area and delay computation mentioned above, every variable can be trivially derived from the configuration, except the number of product terms at each level. A threshold is set for the sum of the individual product terms of all the nodes in the cluster, above which we use it to approximate the number of terms in the minimized PLA. Otherwise a SOP minimization is performed. After the level of the flexible nodes is determined, the nodes on the same level are assigned to a multiple-output PLA and further optimized with a SOP minimizer. The net ordering algorithm is detailed in [9].

Experimental results show that, compared to SCs and NPLAs, RPLAs provide similar delays. The RPLAs implemented with 2 metal layers consume on average 23% more area than SCs with 3 layers, however on average 22% of the area overhead is white space, which might be utilized by other circuits.

Extension to sequential circuits is easy for RPLAs. The latches can be placed at the top of the RPLA, and a third metal layer can be used to build the feedback signal lines. The feedback wires are expected to use river routing as well.

2.3. Four-River-PLA

A circuit having too many levels may result in a thin and tall RPLA, which is problematic. Or one single RPLA is not big enough to accommodate the given circuit. It is possible to organize four RPLAs (FRPLA) in a ring, as illustrated in Figure 1(c). This is somewhat similar to WPLA, but the building block is now a RPLA. Multiplexers and latches can be inserted between two adjacent RPLAs. Non-adjacent RPLAs (on two ends of the diagonal) can be directly connected through river routing on another metal layer. So far, we have not experimented with a synthesis algorithm for this structure.

3. Fishbone: A Regular Global Routing Scheme

3.1 Description

Assume there are B blocks on the chip, and the routing pitch is 1. An integer N, the grid cycle, is determined (will be clarified later), and the global routing grids are given indices labeled 0,1,2,..,N–1, repeatedly, in both X and Y directions. Each block will be given an output/input pin offset (G_{XO}, G_{YI}) such that its output pins are exactly on grids ($sN+G_{XO}$, tN)[2], where s and t are integers and G_{XO}=1,2,..,N–1. Also all output pins are within the block boundary and no two output pins have the same X coordinate. Similarly, the input pins are on grids (uN, $vN+G_{YI}$) with modulus N, where u and v are integers and G_{YI} = 1,2,..,N–1, as long as they keep the input pins within the block boundary, and guarantee no two input pins have the same Y coordinate. The rule implies that there will be no conflict between input and output pins, because the output pins have 0 Y-indices, and the input pins have Y-indices ranging from 1 to N–1. A symmetrical implication exists for the X direction. The number of the output pins that can be accommodated in a block is W/N, where W is the width of the block. Similarly we have H/N for the input pins, where H is the height of the block. If N is reasonably controlled, this accommodation can be achieved; otherwise the block needs to expand. In this paper, we only use a single N for both X and Y directions, although different N's are possible. In fact, if the total number of the critical output pins is not greater than the critical input pins, we may use a smaller N_{XO} and a larger N_{YI}.

Suppose two metal layers are used for the vertical and horizontal routing of these critical nets, respectively. It is obvious that if all the blocks in the same row (meaning that their horizontal projections overlap) have different G_{YI}, and all the blocks in the same column have no common G_{XO}, all the pin-to-pin connections can be implemented by an L-shaped routing, where a via occurs at the bending point. Multiple fanout nets result in a fish-bone topology, in which, the output pin connects to a vertical trunk, and all input pins connect to the trunk by horizontal branches. The net topology is fully determined by the positions of the net's pins. An example is illustrated in Figure 2(a).

Without loss of generality, assume the output pins are on the lower metal layer[3], and on this layer the routing direction is vertical, and the input pins are on the higher metal layer and this layer uses the horizontal direction. For an input pin to connect to the higher metal layer, a via is needed. This means that the grid on the lower metal layer above the input pin is pre-occupied. This is the reason that a 0 Y-index is reserved for the output pins. The X (output pins) and Y (input pins) grid offsets of block b are denoted by $G_{XO}(b)$ and $G_{YI}(b)$. Consider $G_{XO}(b)$. Two rules are given below, each of which alone guarantees the routability of the critical nets with the FBRS.

> **Grid Rule:** $G_{XO}(b_i)\neq G_{XO}(b_j)$, if $LL(b_i)<RR(b_j)$ and $LL(b_j)<RR(b_i)$, where $LL(.)$ is the left edge coordinate of the block, and $RR(.)$ is the right edge coordinate of the block.

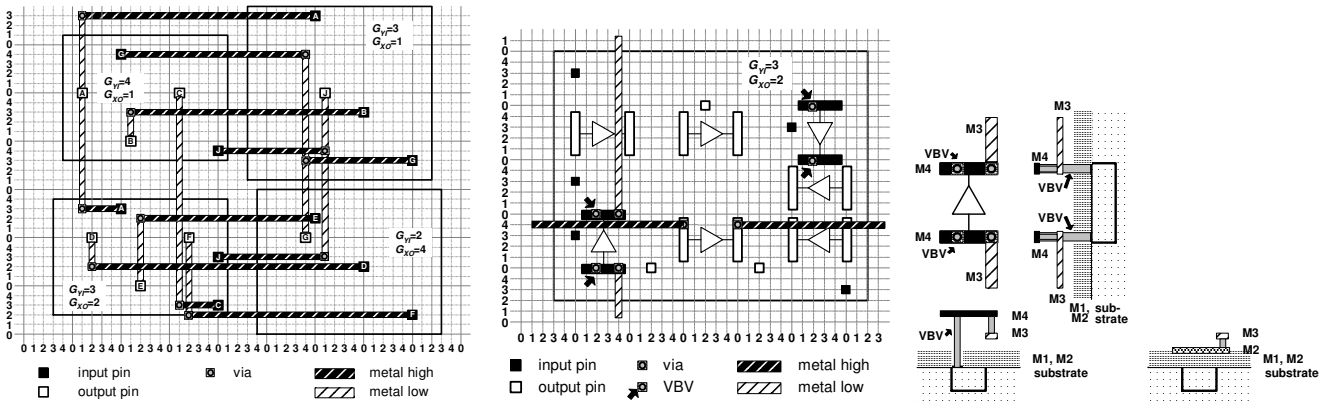
A symmetrical criterion exists for the $G_{YI}(b)$. We pre-define N and seek an assignment of $G_{XO}(b)$ and $G_{YI}(b)$ that satisfies the rule. A reasonable N can be chosen as $N=\lceil B^{1/2}\rceil+\rho$, where ρ is a small integer. In case the number of blocks is too large, to get a reasonably small N, a hierarchical routing scheme might be necessary, where the chip can have more than one level of FBRS, for example, using metal 3 and 4 for the lower level FBRS and metal 5 and 6 for the higher level FBRS.

For soft blocks, the pins can be located based on the $G_{XO}(b)$ and $G_{YI}(b)$ assigned to the block. For hard blocks, we can have various versions of the layouts ready. The versions are only different in terms of $G_{XO}(b)$ and $G_{YI}(b)$. The variation of wire length due to the offsets of the pins is small. If the hard blocks have only one version, or their G_{XO}'s and G_{YI}'s are fixed, the floorplanning of them is less flexible because of the grid rule.

We only consider the case of hard blocks with non-inverted built-in buffers. To guarantee the buffer locations are valid for the FBRS, the buffers are arranged as shown in Figure 2(b). A buffer has the span of N wire pitches[4], and the input pin and output pins both have width of (N–1) wire pitches on a low or high metal. The pins of the horizontal buffers (facing left or right) vertically occupy grid-indices (0, 1 to N-1) on metal low. A horizontal wire (metal high) directed to the right with vertical grid $y\in\{1,2,..,N-1\}$, for instance, can use a horizontal buffer by connecting it through a via to the buffer input pin at (0,y,*low*) and a via to the output pin at (0+N,y,*low*). The rule for valid horizontal buffer locations is:

> **Horizontal Buffer Grid Rule**: Both input and output pins of a horizontal buffer are on metal low, are (N–1) wide, and must not overlap any input pin of the block (on metal high).

[1] Nodes on the same level are clustered together and placed in the same PLA.
[2] The N modulus operations are abbreviated in the paper.
[3] A pin in this paper is considered to be on metal low (M3) or high (M4), so it consists of a piece of metal on that layer and the via(s) connecting it to metal 2 or lower.
[4] It is a reasonable size to accommodate a buffer if N is not too small.



(a) The FBRS (b) Built-in buffers (c) The vertical buffer (d) Pins on M2?

Figure 2. The Fishbone routing scheme for critical nets.

The vertical buffers, which buffer vertical (low) wires, have their pins on metal high and grids (1 to N–1, 0). But they have a more restrictive rule:

> **Vertical Buffer Grid Rule**: A vertical buffer (connecting to vertical wires (metal low)) has input and output pins on metal high, are (N–1) wide, and connects through a pair of vias, called vertical-buffer-vias (VBV), on grids (G_{XO}, 0) and (G_{XO}, 0+N) respectively. The connection to the vertical buffers below metal low must not be placed in any column of the block that has an output pin.

The configuration in Figure 2(b) uses this rule. The reason for the last part of this rule is that a vertical buffer's connection to metal high uses a double via to first connect to its pin at metal high, and then another via to connect from metal high to metal low at the horizontal index of the vertical wire being connected to, as illustrated in Figure 2(c). One may ask why we do not use a simpler scheme as illustrated in Figure 2(d), that is, laying the buffer pins on metal 2 (M2). The reason is that M2 and below are mainly used for the intra-block wiring which has high density. We cannot afford to place so many N-grid wide buffer pins on M2. Notice that although the VBV scheme seems to use more vias, the number of vias is exactly known, thus will not cause prediction error.

An alternative to this rule is to create a new vertical track to place the VBV's, but this would increase the horizontal grid cycle by one.

Since the blocks are designed prior to the floorplanning, there is no reason to use an uneven distribution of the buffers. In addition, the buffers should be arranged such that a wire crossing the block can reach at least one buffer. An integer M_{BD} denotes the density of the built-in buffers, such that the horizontal buffers are embedded at a grid of $NM_{BD}\times N$, and the vertical buffers are embedded in a grid of $N\times N\cdot M_{BD}$.

The floorplanning problem with FBRS is to determine the location and orientation for each block, and the shapes and pin positions for the soft blocks, such that no two blocks overlap, that one of the grid rules is obeyed, and some form of cost function is minimized.

3.2 Floorplanning

A simulated-annealing approach is adopted, and the sequence-pair method is employed as the layout representation [8]. A random move can be one of the following types: sequence-pair swapping, block shape adjustment if the block is soft, block orientation adjustment, and block grid adjustment if the block is soft or the hard block has more than one version of G_{XO} and G_{YI} available. Note that now a block only has four valid orientations, i.e., normal, X-flipping, Y-flipping and XY-flipping. G_{XO} and/or G_{YI} also change when the orientation is changed. The cost function is:

$$Cost = w_A A + w_{CWDa} CWDa + w_{CWDm} CWDm + w_{NCWL} NCWL + w_G V_G$$

in which A is the chip area, $CWDa$ is the average critical wire delay, $CWDm$ is the maximum critical wire delay, $NCWL$ is the total wire length of the non-critical nets, and V_G is the number of grid rule violations. Parameter w_A, w_{CWDa}, w_{CWDm}, w_{NCWL} and w_G are the corresponding weights. In the cost function, the explicit critical wire delays (CWD) replace the commonly used total critical wire lengths (CWL). The computation of the FBRS wire delays uses the Elmore model. With FBRS, terms in the Elmore model can be represented explicitly by the pin positions.

When built-in buffers are available, an algorithm based on van Ginneken's method [15] solves the buffer insertion problem at each iteration of the simulated-annealing. Now the legal locations of the buffers are given by the locations of the hard blocks and how their built-in buffers are arranged. After each random move of simulated-annealing, the following steps are executed:

> **Step 0**: Initialize a buffer indicator array, based on the current floorplan, which shows where a built-in buffer is available.
> **Step 1**: Collect the most critical nets. The "most criticality" can have different definitions. Here we want to minimize the maximum wire delays; hence a net is the most critical if it has at least one wire that exceeds a given delay threshold. Other definitions such as negative slacks can also be adopted. For run time concerns, the threshold is set such that only a small portion of nets, which are certainly the most critical, are collected.
> **Step 2**: Pick up a net from the collected set in descending order of criticality. Determine from the buffer indicator array all the available buffers for this net. Apply van Ginneken's algorithm to minimize the delays by using the available buffers. The used buffers will be labeled "non-available" in the buffer indicator array to prevent occupation by subsequent nets.
> **Step 3**: Iterate step 2 until all collected nets are tried.

After the simulated-annealing is complete, the remaining critical nets are tried with the same algorithm to further reduce their wire delays with the left-over buffers.

The resulting floorplan has the critical wires determined and fixed. The buffer assignment is also determined. Then the layout is passed to a router to finish routing the non-critical nets

3.3 Wiring Experiments

In our experiment, three different routing models are compared to FBRS: Single-Star (SS) [13], Half-Perimeter-and-Manhattan (HPM) and a simplified version of Rectilinear-Steiner-Arborescence (RSA) [14]. RSA is used only for predicting the critical wire delays after the floorplan is produced by HPM because it is too expensive to be included in the simulated-annealing approach. After floorplanning, the designs are passed to the Cadence Warp Router to finish the routing. All the delays are computed from the extracted parameters of the final layouts. For FBRS, there is no need to route the critical nets, because they are already determined by the floorplanning. The experimental results show that, on average, the area increase by using FBRS is small, compared to SS and HPM. As more blocks become soft, or hard blocks have more versions of G_{XO} and G_{YI} available, the area penalty will diminish. The average and maximum critical wire delays (CWD) produced by SS are both worse than those by FBRS. HPM gives 5% and 7% better results in the average and maximum CWD's. As mentioned, the values of the CWD's are not considered the most important; instead we are interested predictability, measured in terms of the errors between the predicted values and the real values. FBRS gives zero errors. In terms of average errors, SS overestimates the CWD; HPM has both negative and positive errors; and RSA underestimates. However average errors are better understood if their deviations are given. SS and HPM have deviations of about 40% to 30% respectively, of the prediction errors. The maximum errors also give a direct feeling of how inaccurate the predictions can be; SS and HPM can produce maximum errors of over 100%. Note that sometimes only one wrong prediction is enough to cause a complete new iteration in the design flow. RSA usually gives better predictions than SS and HPM. However, its prediction errors still exist, and due to its computational complexity, the floorplanner can't use it in a simulated-annealing approach.

The experiments with built-in buffers show that when M_{BD}=10, the average chip area increase is about 6%, the average CWD decreases 9%, and the maximum CWD decreases 10%. When M_{BD}=30, the area increase is on average about 1%. Meanwhile, the corresponding delay reductions are similar to the M_{BD}=10 case. This indicates that a good value for M_{BD} exists for a circuit such that delays can be improved without wasting too much chip area.

4. Discussion

Three regular circuit structures and a regular global routing scheme are presented in this paper. To complete the flow, we still need to solve the following problems: First, how to partition the whole system into a set of blocks that can be efficiently implemented with the regular structures and the regular global wires? Second, how to build the connections from critical pins to the internal logic? Third, the regular global routing scheme solves the wire delay uncertainty problem, but to optimize the wire delay and the logic delay may still require iterations between logic and physical design.

Some other remaining problems are worth discussion. (1) The PLA structures discussed in this paper are static, which consume quiescent DC power because they use pull-up/down devices. Dynamic PLA structures are more power efficient and are faster than static PLAs [12]. (2) PLAs can also be re-sized to get different area/performance characteristics. Also the characterization of a set of PLA parameters is much faster than that of a library of hundreds of gates. (3) Engineer Change Orders (ECOs) for standard cell implementations involve both synthesis and physical design modifications. However for the WPLA, it is mainly a synthesis problem. (4) A programmable version of the RPLA, called the Glacier PLA, is reported in [9]. A programmable version of the WPLA should be developed, and experiments done to show if it is a good alternative to the LUT-based structures.

Several open problems remain for FBRS. Buffer insertion between the blocks, might provide more opportunities for delay reduction. One of its advantages is that it can make use of the white space there. In case the blocks are too small to hold enough built-in buffers, buffers between the blocks are helpful. The other question is how to define net criticality. The criticality of a net may change with the floorplan. In this paper, we assume the areas of the soft blocks are known. But in real designs, they are not, because "soft" implies there is some remaining synthesis work and thus the area is not yet determined.

Acknowledgements

This work was supported by GSRC (grant from MARCO/DAPPA 98DT-660, MDA972-99-1-0001). The authors are grateful to Philip Chong, Qingjian Yu and members of the Constructive Fabrics Group of GSRC for many enlightening discussions. Also we gratefully acknowledge support from the California Micro program and our industrial sponsors, Cadence and Synplicity.

References

[1] C. Papachristou and A. Pandya, "A design scheme for PLA-based control tables with reduced area and time-delay cost", IEEE Transactions on Computer Aided-Design, vol. 9, no. 5, May 1990, pages 453-472.

[2] "Silicon VLSI Technology", Chapter 5, Lithography, edited by J.D. Plummer, M.D. Deal and P.B. Griffin, Prentice Hall, 2000.

[3] R. Bryant, K-T. Cheng, A. Kahng, K. Keutzer, W. Maly, R. Newton, L. Pileggi, J. Rabaey, A. Sangiovanni-Vincentelli, "Limitations and challenges of computer-aided design technology for CMOS VLSI", Proceedings of the IEEE, vol. 89, issue 3, Mar 2001, pages 341-365.

[4] S. Khatri, R. Brayton and A. Sangiovanni-Vincentelli, "Cross-talk immune VLSI design using a network of PLAs embedded in a regular layout fabric", Proceedings of International Conference on Computer aided Design, Nov 2000, pages 412-418.

[5] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton and A. Sangiovanni-Vincentelli, "SIS: A system for sequential circuit synthesis", Tech. Rep., UCB/ERL M92/41, Electronics Research Lab, University of California, Berkeley, May 1992.

[6] R. Rudell and A. Sangiovanni-Vincentelli, "Multiple-valued minimization for PLA optimization", IEEE Transactions on Computer Aided-Design, vol. 6, Sep 1987, pages 727-750.

[7] R. Brayton, G. Hachtel and A. Sangiovanni-Vincentelli, "Multi-level logic synthesis', Proc. of IEEE, vol. 78, Feb. 1990.

[8] R. Brayton, G. Hachtel, C. McMullen and A. Sangiovanni-Vincentelli, "Logic minimization algorithms for VLSI synthesis", Kluwer Academic Publishers, 1984.

[9] F. Mo and R.K.Brayton, "River PLAs: A Regular Circuit Structure", to be published, Design Automation Conference, Jun 2002.

[10] J. Cong, T. Kong and Z. Pan, "Buffer Block Planning for Interconnect Planning and Prediction", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 9, no. 6, Dec 2001, pages 929-937.

[11] R. Ho, K.W. Mai and M.A.Horowitz, "The Future of Wires", Proceedings of the IEEE, vol. 89, no.4, Apr 2001, pages. 490-504.

[12] Y.B. Dhong and C.P. Tsang, "High Speed CMOS POS PLA using Predischarged OR Array and Charge Sharing AND Aray", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 39, no. 8, Aug 1992, pages 557-564

[13] F. Mo, A. Tabbara and R.K. Brayton, "A Force-Directed Macro-Cell Placer", International Conference on Computer Aided Design, Nov 2000, pages. 177-180.

[14] J. Cong, A.B. Kahng and K-S. Leung, "Efficient Algorithms for the Minimum Shortest Path Steiner Arborescence Problem with Applications to VLSI Physical Design", IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, vol. 17, no. 1, Jan 1998, pages 24-39.