Regular FPGA based on regular fabric

Chen Xun(陈迅)[1, 2, †], Zhu Jianwen(朱剑文)[2], and Zhang Minxuan(张民选)[1]

[1] School of Computer, National University of Defense Technology, Changsha 410073, China
[2] Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, M5S3G4, Canada

Abstract: In the sub-wavelength regime, design for manufacturability (DFM) becomes increasingly important for field programmable gate arrays (FPGAs). In this paper, an automated tile generation flow targeting micro-regular fabric is reported. Using a publicly accessible, well-documented academic FPGA as a case study, we found that compared to the tile generators previously reported, our generated micro-regular tile incurs less than 10% area overhead, which could be potentially recovered by process window optimization, thanks to its superior printability. In addition, we demonstrate that on 45 nm technology, the generated FPGA tile reduces lithography induced process variation by 33%, and reduce probability of failure by 21.2%. If a further overhead of 10% area can be recovered by enhanced resolution, we can achieve the variation reduction of 93.8% and reduce the probability of failure by 16.2%.

Key words: FPGA; layout automation; design for manufacturability; regular design fabric

DOI: 10.1088/1674-4926/32/8/085015 **EEACC:** 2570

1. Introduction

As feature size has decreased dramatically in recent years, backend designers not only need to make sure that the function of the layout is correct but also that they determine the kind of layout style that could be better printed on the wafer, in other words, they need to achieve small variation and low probability of failure. These new requirements bring a new challenge, known as design for manufacturability (DFM).

DFM has often been marketed by the field programmable gate array (FPGA) industry as an advantage over application-specific integrated circuits (ASICs). However, as leading users of advanced process nodes, FPGA companies themselves are not shielded from printability problems: even though FPGAs are largely constructed by repeating tiles, each tile is complex enough that DFM techniques have to be employed.

In a recent keynote paper by Pillage *et al.*[1] (henceforce referred to as the Carnegie Mellon University study or CMU study), the notion of macro-regularity (property of circuits that use a small, limited number of cells) and micro-regularity (property of circuits that use a limited number of layout constructs) is distinguished. It was empirically validated that various regular fabrics can be used to construct micro-regular circuits with comparable area as traditional circuits, with significantly better printability.

In this paper, the CMU study is extended and applies micro-regularity in the context of FPGA by reporting a micro-regular FPGA fabric. More specifically, we make the following contributions:

(1) We report an automatic tile generation flow in the same spirit of Refs. [2, 3], while targeting micro-regular fabric. Although not all techniques used by the reported flow are new, we believe that the synthesis and adaptation of these techniques to produce a competitive layout is critical to establish the credibility of the rest of the study.

(2) We quantify the “area overhead” of micro-regular fabric against that generated by previously reported tile generation approaches using the same design rules. Although it is understood that the use of micro-regular fabric enables design/process co-optimization, thus leading to the recovery or even better area by using more aggressive design rules, we leave the quantification of such a tradeoff to more resourceful readers. Nevertheless, we hope our reported overhead can help one assess if such further study is worthwhile.

(3) We quantify the benefits of micro-regular FPGA fabric on both process variation (relating to parametric yield) and functional yield. Note that these results have not been previously reported and thus are complementary to CMU study, which focuses on printability.

The remainder of the paper is organized as follows. Section 2 introduces the background and related work. Section 3 introduces our proposed micro-regular FPGA fabric generation flow, including grid selection, cell library building and tile generation. Section 4 details the evaluation methodology. Finally, we report our results and draw conclusions.

2. Background and related work

2.1. FPGA physical architecture

FPGAs are well known for their regular, repetitive logic architecture. This is reflected in their physical architecture, where a basic building block, known as a tile, is repeated many times. An example of classic island-style FPGA architecture is shown in Fig. 1, where each tile consists of basic logic elements (BLEs), BLE input connection blocks (ICBs), BLE output connection blocks (OCBs) and routing switch blocks (SBs). A BLE in turn consists of a look up table (LUT), LUT ICB, flip-flop (DFF) and output multiplexer. From this figure, it can be seen that an FPGA tile can be constructed from a small set

[†] Corresponding author. Email: xun.chen.1981@gmail.com

Received 20 February 2011, revised manuscript received 12 April 2011

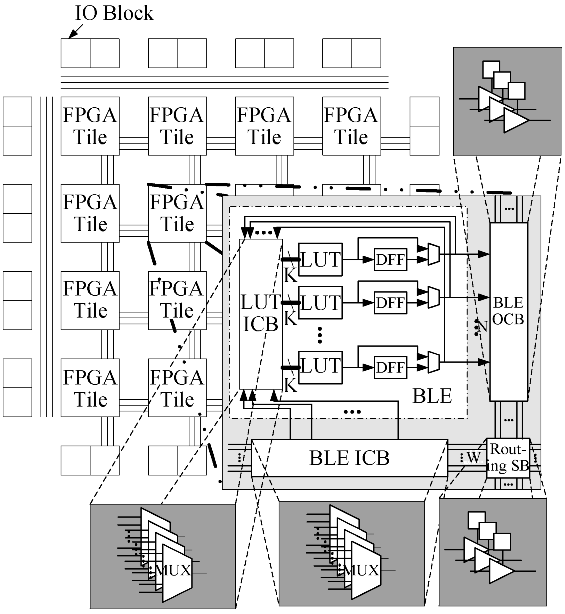


Fig. 1. FPGA tile architecture.

of library cells.

2.2. FPGA tile layout automation flow

In commercial FPGA design, the FPGA core tile repeats in the order of 10,000 times[4]. It is thus essential to achieve high layout efficiency in tile design. The transistor-level and layout design of a tile is particularly labour intensive and time consuming, with the latter reportedly taking from 9 months to a year[4] to develop.

Motivated by reducing the tile design time, several efforts to automate tile layout generation has been attempted. These efforts vary with different design flows.

(1) Standard cells with standard tools (SS). In this design flow, the FPGA tile architecture is directly written in RTL, and then synthesized and laid out using a standard cell library and design flow[5−7]. This method makes it easier to use the FPGA in embedded context while the area overhead is significant.

(2) Custom cells with custom tools (CC). In this design flow, the cells are custom designed, then they are used by a custom designed placer and router to generate the final layout[2−4, 8−11]. The area of the generated layout was reported as within a factor of two of the manually designed tiles, 33% smaller than the tile generated by SS flow.

(3) Custom cells with standard tools (CS). Since the standard cell library cannot implement FPGA tiles efficiently, a remedy to this is to judiciously introduce custom cells, in addition to standard cells[7]. By adding custom multiplexer cells that reduce waste of the nwell area, an improvement of 42% was achieved over the SS flow.

2.3. Regular design fabric

The regular design fabrics are proposed[12, 13] precisely to reduce the limit of the layout patterns in a design. The fabrics in general obey the following rules:

(1) All non-contact layers are restricted to one orientation. Given an orientation of a layer, we can distinguish those edges along the designated direction as the line edges, and the perpendicular edges as the end edges.

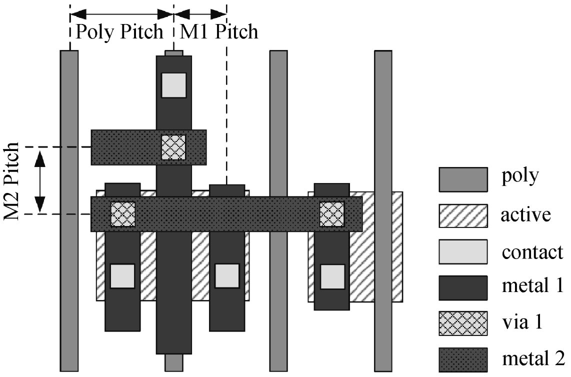


Fig. 2. Regular design fabric.

(2) All non-contact layers are restricted to be placed on the multiple of a minimum horizontal or vertical fabric grid. The exact multiple for a layer defines its layer grid. Accordingly, the contacts and vias are allowed on the grid intersection points of the layers that they connect.

(3) The layer pitch of a non-contact layer is defined as the minimum distance between corresponding line edges of adjacent, parallel rectangles. Note that the layer pitch of a layer is not necessarily equal to its layer grid.

The different configuration of layer orientation, grid and pitch then defines the different regular fabrics. The number of possible layout patterns is implicitly controlled by these configurations. Figure 2 shows one such configuration adopted by this paper, called the front-end-of-line (FEOL)-limited fabric. A limited number of exceptions can be made to allow Metal1 running horizontally to reduce the number of non-redundant vias. These are called wrong-way Metal.

3. Micro-regular FPGA

Micro-regular design involves first deciding the layout parameters for the chip layout. These include factors such as the cell heights and the placement grid size. Next, a cell library is created for each netlist component found in the FPGA architecture, such as SRAM cells or multiplexers. This library is then used as the basic building blocks to create our FPGA where an FPGA architecture description is technology mapped to the basic building blocks. The resulting netlist is then place-and-routed to generate the final FPGA tile layout.

3.1. Designing the grid

To better leverage the standard placement and routing tools, we decide to build cells on the fabric using the same cell height (with the exception of SRAM and multiplexer cells) but variable width. The remaining parameters to decide are the fabric grid size, as well as the cell height. The grid size is chosen as the minimal value that can avoid the design rule variation and the cell height is chosen considering the routability and area. In our work, the height is set to be 9 VGrid.

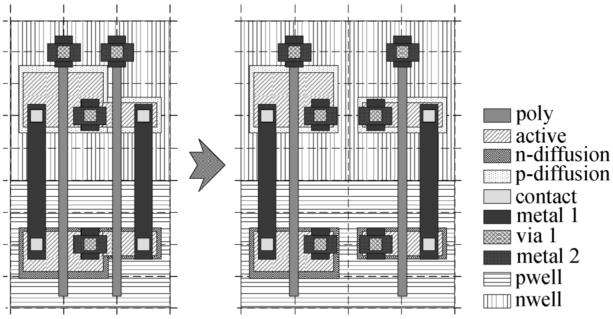


Fig. 3. Split transistor chaining to avoid diffusion rounding.

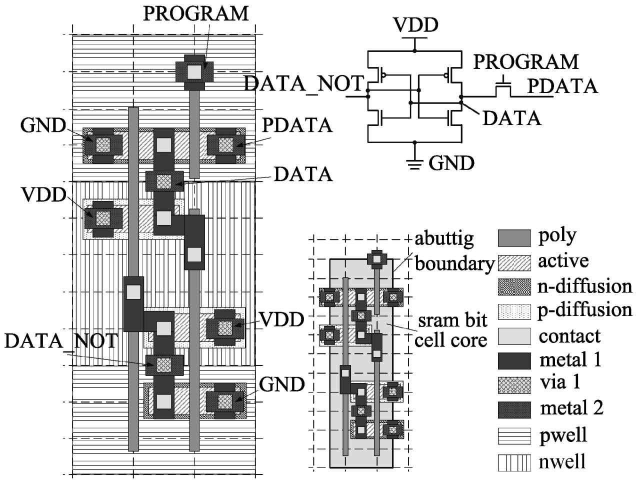


Fig. 4. The layout of SRAM bit mapped onto regular design fabric.

3.2. Designing cell library

Decades of research on the layout generation of CMOS logic cells has been reported[14−17]. The basic flow of these methods includes transistor pairing, folding, chaining and ordering. The key objective of the flow is to share the diffusion to save area.

We augment such a flow with the option of diffusion splitting due to a corner rounding problem that we will discuss in detail in Section 4.1. In a nutshell, whenever the widths of neighboring transistors differ, the printability problem leads to larger variation in transistor length. As a design trade off between area and variation, we allow the generation of cells without corner rounding by automatically splitting the transistor chain in such cases. As shown in Fig. 3, this costs 1 H in cell width.

All of the layout generation algorithms are implemented using the Pycell studio[18] of Ciranova[19]. Below we discuss in detail the design of SRAM and MUXes, which are found ubiquitous use in the tile and require special attention.

3.2.1. SRAM cell design

Figure 4 shows the SRAM bit layout mapped to the regular design fabric. Note that a layout topology different from traditional SRAM is chosen in order to satisfy the fabric constraints. The SRAM bit is designed to be 3 HGrid wide and 12 VGrid high (3 H × 12 V). Note in particular that the wrong-way metal is used to connect the cross coupled inverters in order to reduce cell area and non-redundant via count.

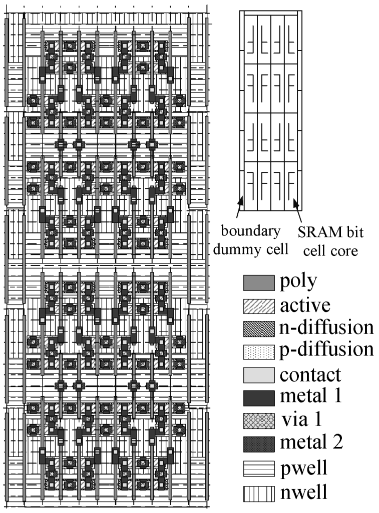


Fig. 5. Mirroring technology used to build MEM4 × 4 cell.

We use the grouping method introduced by Egier[3] to achieve further area reduction by the mirroring and abutting technique commonly found in SRAM cell array design. As shown in Fig. 5, the SRAM bit is first mirrored in horizontal and vertical direction, then abutted together. As shown in Figure 4, the abutting boundary is designed to be 2 H × 11 V, smaller than the cell bounding box. This enables the sharing of PDATA, VDD, GND and PROGRAM port. This technique can reduce the area of the SRAM bit group, and the more SRAM bits are grouped, the more active area that will be saved. For example, a 4 × 4 group needs only 9 H × 44 V dimension, as opposed to 12 H × 48 V dimension. On the other hand, larger grouping may result in more routing congestion.

We chose to group SRAM bits into a 4 × 4 bit array, and a single cell named MEM4×4 is introduced into the cell library. Since the MEM4×4 cell is placed with the rest of the cells in a standard cell form factor, special attention should be paid to the boundary of the SRAM bit group to maintain the layout regularity and avoid design rule violation. Dummy cells are inserted to maintain the continuity of pwell and nwell, and this takes one horizontal grid space at both horizontal sides of the SRAM bit group. In addition, the MEM4×4 cell height needs to be the multiple of chosen standard cell height. These considerations lead to the final area of MEM4×4 to be 11 H × 45 V.

3.3. MUX cell design

To simplify the generation of high fan-in MUXes, we first map 4-input MUX (4-MUX) onto regular design fabric and then use it as a basic block. The layout of 4-MUX is shown in Fig. 6. The traditional diffusion sharing technology is used to save diffusion area, and transistor pairing is used to share the transistor gate. The output of the first level pass transistor is fed to the second level pass transistor through the metal routing. The area of 4-MUX is 6 H × 9 V.

Similar to SRAM bit grouping, special attention should be paid to the boundary of MUXes. Since there are no PMOS transistors in the MUX, the dummy cells are added at the boundary to maintain the continuity of pwell and nwell. For example,

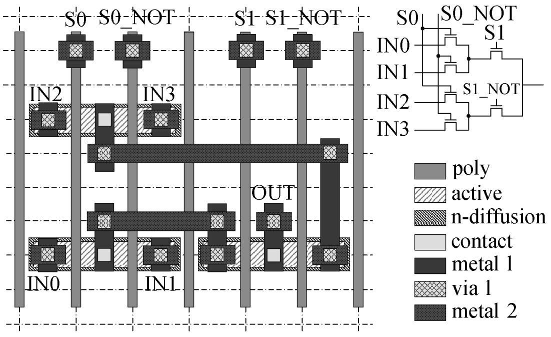
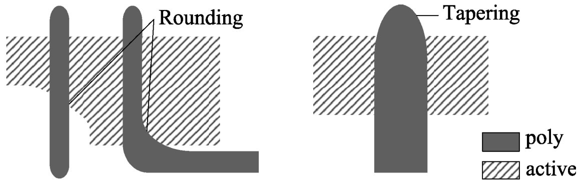


Fig. 6. The layout of 4-input MUX mapped onto regular design fabric.



(a) Diffusion and poly rounding (b) Line-end tapering

Fig. 7. Demonstration of transistor variation.

The 4-input LUT is implemented using a 16-input MUX, and its dimension is 17 H × 18 V.

4. Evaluation metric

The primary evaluation metrics to evaluate our micro-regular FPGA fabric is process variation and probability of failure.

4.1. Variation

There are two major sources of variation during manufacturing which affect device performance: lithography variation and doping fluctuation. Since doping cannot be possibly influenced by layout design, and past research[20] shown that the magnitude of lithography variation is comparable to doping fluctuation, we therefore focus on quantifying lithography variation of FPGA tiles.

As described in Refs. [21, 22], the imperfect layout printing affects the transistor gate dimension, which in turn affects the drive and off current of the transistor, and finally the performance of the circuit. Since the performance of FPGA circuit is undefined before an application circuit is defined, we report on the variation of gate dimension. There are three dominant sources of lithography variation causing gate-dimension variation: (1) diffusion and poly corner-rounding, (2) line-end tapering under overlay error and line-end pullback, (3) critical dimension (CD) variation.

Diffusion and poly corner-rounding: As shown in Fig. 7(a), the transistor width is affected by the diffusion rounding caused by the mismatch between neighboring chaining transistor width, and transistor length is affected by the poly rounding caused by the poly routing. The variation caused by corner rounding is modeled using the same method in Ref. [21].

Line-end tapering under overlay error and line-end pullback: As shown in Fig. 7(b), the erosion of the poly line-end causes the line-end tapering and pullback. Under overlay error, the transistor length may vary. The line-end tapering can be modeled using the method proposed by Gupta in Ref. [23], where the tapering shape is modelled as a super-ellipse.

CD variation: CD uniformity (CDU) is another major contributor to the change in transistor length. CDU is usually described by a normal distribution, which captures the dependency on exposure dose and focus variations.

After determining all terms from different sources, the total variation is calculated by Eq. (1).

$$\Delta\left(\frac{W}{L}\right) = \frac{\sum_{\text{allgates}} \left|\Delta\left(\frac{W}{L}\right)_i\right|}{\frac{W_{\text{tot}}}{L_{\text{ideal}}}}. \tag{1}$$

4.2. Probability of failure

Probability of failure (POF) is used to characterize the IC functional yield. There are three major sources of IC failure: (1) contact hole failure, (2) overlay error coupled with lithographic line-end shortening, (3) random particle defects. They can be characterized using the method described below.

Contact hole failure: The contact hole failure can be reasonably approximated by multiplying the number of non-redundant contacts in the layout and the contact hole failure rate.

Overlay error coupled with lithographic line-end shortening[22]: The overlay error between poly and diffusion layer may cause the line-end shortening. It can be calculated by measuring the probability of the poly line end shifted down to the diffusion edge.

Random particle defects: Critical area analysis[24] is used to capture the failure caused by random particle defects (RPD). The critical area can be breakdown to the open and short area in poly, contact and metal1 layers.

After determining all terms from different sources, the total POF is calculated through Eq. (2).

$$\begin{aligned}\text{POF}_{\text{total}} &= 1 - (1 - \text{POF}_{\text{contact}}) \\ &\quad \times (1 - \text{POF}_{\text{overlay}})(1 - \text{POF}_{\text{RPD}}).\end{aligned} \tag{2}$$

5. Experimental results

5.1. Methodology and settings

5.1.1. FPGA tile architecture

Although lacking some features of modern FPGA architectures, the well-documented, publicly available academic FPGA named POWELL developed at University of Toronto[3] was chosen as the yard stick for fair, repeatable comparisons. Industry readers are encouraged to add their own grain of salt in interpreting the results. The architecture setting of POWELL is shown in Table 1. All the routing tracks use bidirectional buffered switches.

5.1.2. Comparison methodology

We would like to evaluate the relative merits of proposed micro-regular tile against what was previously reported by SS,

Table 1. FPGA tile architecture parameters.

Parameter	Value
LUT input count (K)	4
Number of LUT in BLE (N)	3
Number of BLE input (I)	8
Number of tracks (W)	20
Number of tracks connected to BLE input ($F_{\text{c,input}}$)	12
Number of tracks connected to BLE output ($F_{\text{c,output}}$)	20/3
Routing tracks length	4

Table 2. Different implementation of FPGA tile.

Tile	Technology	Purpose
180_NCR	0.18 μm	area
180_CR	0.18 μm	area
180_CC	0.18 μm	area
45_NCR	45 nm	manufacturability
45_CR	45 nm	manufacturability
45_SS	45 nm	manufacturability

CS, and CC flows (refer to Section 2.2). Given the lack of access to actual tools used by previous studies and given that misleading conclusions might be drawn if naively citing the published data obtained often by different experiment settings, we adopted the following comparison methodology to compare process variation and the probability of failure.

(1) Since we are given full access to the original POWELL chip produced by GILES, which is developed on TSMC 0.18 μm technology, we compare the proposed method against the CC flow on the same technology to evaluate area overhead. We believe this comparison is representative for both CC and CS flow since they reportedly produce comparable area.

(2) Since process variation and probability of failure are more relevant in more advanced technology, we evaluate them on the publicly accessible 45 nm FreePDK technology[25] with the 45 nm Open Cell Library V1.3[26]. Since we do not have access to the previous GILEs tool, we produce the results of the SS flow using standard commercial CAD tools. This provides a fair area comparison between our approach and the previous work in Ref. [2].

(3) Given the difficulty, we cannot reproduce variation and functional yield results for CC and CS methodology on FreePDK 45 nm technology. However, we believe this is unnecessary since they should be very similar to those of the SS, which we report in detail.

We compare six tile implementations of the same POWELL architecture. In Table 2, 180_NCR and 180_CR are the proposed micro-regular fabric technique using no corner rounding and corner rounding respectively. These are using the TSMC 0.18 μm technology, 180_CC is the original GILES generated tile, 45_NCR and 45_CR is the proposed micro-regular fabric on the 45 nm FreePDK technology and 45_SS is produced by standard cell methodology (using custom SRAM cell design).

An RTL model for POWELL is written in Verilog to implement the SS flow. The same types of cells, transistor sizes, and the cell netlist (kindly provided by POWELL authors) are used to generate micro-regular tiles.

Table 3. DRE layout style parameters for 45_SS evaluation.

Parameter	Value
Power-straps	metal 1
Cell height	1400 nm
Cell unit-width	190 nm
Fixed gate-pitch	no
1D-poly	no
Limited poly routing	no
Discrete cell width	yes
Input pin access requirement on M1	yes
Output pin access requirement on M1	yes

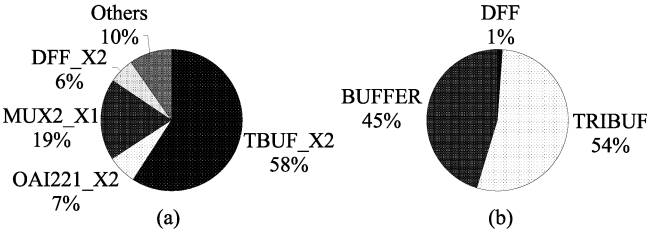


Fig. 8. Variation analysis. (a) 45_SS corner rounding. (b) 45_CR corner rounding.

5.1.3. Parameter settings for variation and probability of failure evaluation

The variation and POF is strongly affected by the choice of process control parameter. We choose the same process control parameters as reported in Ghaida's paper[27], which are original derived from projected values from ITRS technology road map. We also used the DRE tool developed by the same authors to evaluate the variation and POF for 45_SS. Because the layout style parameters of DRE will affect the calculation, we list them in Table 3.

5.2. Process variation and probability of failure result

Table 4 to 6 list all cells used to build FPGA tile and their individual variation and POF parameters. Here, the row labeled "Accumulated" gives the accumulated value for these parameters. The row labeled "Total" gives the total variation and POF values. This final row is equivalent to the variation and POF of the tile since each cell and process affect is treated as an independent event. The column labeled "Cell count" shows the number of cells used in the tile and the "Transistor width" column shows the total transistor length in the cell which will be used to calculate "Accumulated" value for the variation.

More information is given in Figs. 8 and 9, which give the individual contributions of different cell types for the target variation and POF parameters. Figures 8(a) and 8(b) refer to corner rounding caused variation. Since CDU caused variation is uniform for all the cells and line-end tapering caused variation is small, we do not plot them here. Figures 9(a), 9(b) and 9(c) show the RPD caused failure, and Figures 9(d), 9(e) and 9(f) show the contact caused failures.

From the tables, we can see the total tile variation when using our approach with no corner rounding (45_NCR) is reduced by 93.8% when compared to 45_SS and the POF reduces by 16.2%. When using corner rounding, our approach reduces variation by 33% and POF by 21.2%.

Table 4. Area, variation and POF of 45_SS.

Cell	Cell count	Cell area (μm^2)	Transistor width (μm)	Variation (%)			POF		
				CDU	Corner rounding ($\times10^{-4}$)	Line-end tapering	RPD ($\times10^{-10}$)	Contact ($\times10^{-10}$)	Overlay ($\times10^{-8}$)
AOI21_X1	8	1.064	9.35	0.783	0.355	2.75	1.45	4.00	1.12
AOI21_X2	8	1.064	1.87	0.783	0.353	1.37	1.51	4.00	1.12
AOI22_X1	16	1.330	1.30	0.783	0	2.64	1.98	5.20	1.12
AOI221_X1	20	1.596	1.91	0.783	0.347	2.25	2.33	6.00	1.12
AOI222_X1	8	1.862	2.34	0.783	0	2.20	2.97	8.00	1.12
AOI222_X4	8	3.724	9.36	0.783	0	1.16	6.40	1.28	1.12
TBUF_X2	111	1.596	1.17	0.783	2.16	1.52	0.749	2.20	1.12
INV_X1	63	0.532	0.225	0.783	0	3.81	0.659	2.00	1.12
INV_X4	2	0.532	0.900	0.783	0	0.953	0.563	2.00	1.12
MUX2_X1	46	1.862	1.980	0.783	0.938	2.60	3.23	8.40	1.12
NAND2_X1	136	0.798	0.530	0.783	0	3.23	0.992	2.80	1.12
NAND3_X1	8	1.064	0.915	0.783	0	2.81	1.35	3.60	1.12
NOR2_X1	49	0.798	0.570	0.783	0	2.64	0.993	2.80	1.12
NOR2_X2	4	0.798	1.14	0.783	0	2.64	1.01	2.80	1.12
OAI21_X2	3	1.064	1.83	0.783	0.517	1.41	1.53	4.00	1.12
OAI211_X1	8	1.596	1.34	0.783	0.371	2.56	1.85	4.80	1.12
OAI22_X1	20	1.330	1.30	0.783	0	2.64	1.85	4.80	1.12
OAI22_X2	8	1.330	2.60	0.783	0	1.32	1.85	4.80	1.12
OAI221_X1	16	1.596	1.77	0.783	0.563	2.43	2.35	6.00	1.12
OAI221_X2	16	1.596	3.53	0.783	0.536	1.22	2.47	6.00	1.12
DFF_X2	3	5.586	4.77	0.783	1.85	2.52	8.56	0.220	1.12
SRAM_bit	241	2.128	0.58	0.783	0	1.65	0.810	6.00	1.12
Accumulate				0.783	0.555	2.18	1010	3640	1.12
Total	802	1205.51	820.68	1.338			4.76×10^{-7}		

Table 5. Area, variation and POF of 45_SS.

Cell	Cell count	Cell area (μm^2)	Transistor width (μm)	Variation (%)			POF		
				CDU	Corner rounding	Line-end tapering	RPD ($\times10^{-10}$)	Contact ($\times10^{-10}$)	Overlay ($\times10^{-13}$)
BUFFER	128	1.305	1.07	0.083	2.21	0	0.932	4.80	4.18
DFF	3	3.915	1.74	0.083	0.71	0	2.52	0.124	4.18
INVX1	26	0.653	0.45	0.083	0	0	0.387	2.00	4.18
INVX2	86	0.653	0.45	0.083	0	0	0.387	2.00	4.18
INVX4	13	0.979	0.90	0.083	0	0	0.556	3.20	4.18
LUT4	3	11.09	7.20	0.083	0	0	6.36	32.4	4.18
LR	10	0.979	0.60	0.083	0	0	0.733	3.20	4.18
MEM4x4	16	17.95	9.28	0.083	0	0	11.3	60.8	4.18
MUX12	20	9.135	5.76	0.083	0	0	5.91	2.60	4.18
MUX2	3	0.979	0.18	0.083	0	0	0.518	2.40	4.18
AND	111	0.979	0.60	0.083	0	0	0.572	2.80	4.18
TRIBUF	111	1.958	1.25	0.083	1.89	0	1.17	6.00	4.18
Accumulate				0.083	0.816	0	697	3050	4.18
Total	530	1106.31	694.41	0.899			3.75×10^{-7}		

Table 6. Area, variation and POF of 45_SS.

Cell	Cell count	Cell area (μm^2)	Transistor width (μm)	Variation (%)			POF		
				CDU	Corner rounding	Line-end tapering	RPD ($\times10^{-10}$)	Contact ($\times10^{-10}$)	Overlay ($\times10^{-13}$)
BUFFER	128	1.631	1.07	0.083	0	0	1.11	5.60	4.18
DFF	3	4.568	1.74	0.083	0	0	2.87	14	4.18
TRIBUF	111	2.284	1.25	0.083	0	0	1.35	6.80	4.18
Accumulate				0.083	0	0	741	3250	4.18
Total	530	1186.24	694.41	0.083			3.99×10^{-7}		

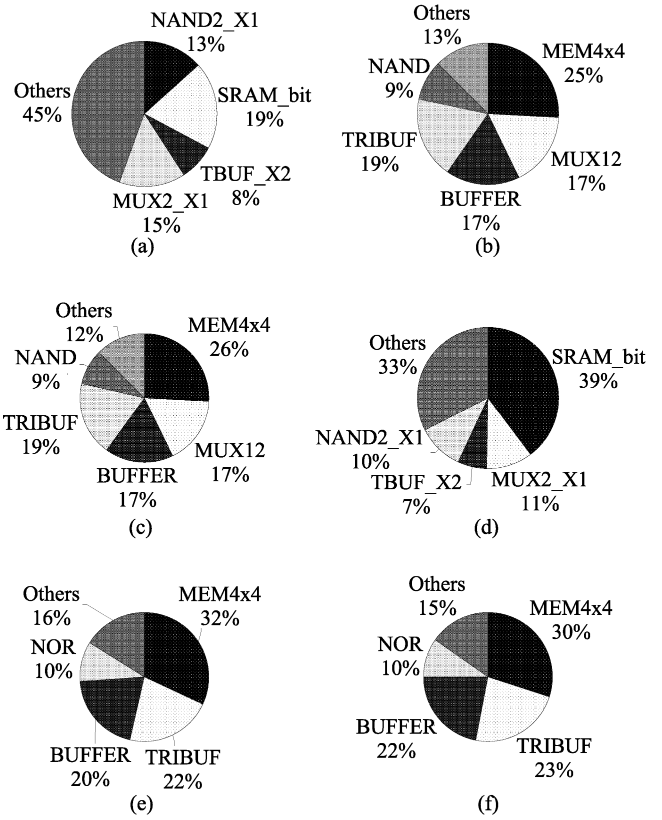


Fig. 9. Probability of failure analysis. (a) 45_SS RPD. (b) 45_CR RPD. (c) 45_NCR RPD. (d) 45_SS contact. (e) 45_CR contact. (f) 45_NCR contact.

Let us discuss in more detail on how the reduction on variation is achieved. By using regular design fabric, we can avoid the variation caused by line-end tapering because the minimal poly to diffusion extension is set to be 70 nm, larger than the 55 nm required by design rule. In addition, the fixed poly pitch naturally improves CDU's 3σ value. For 45_NCR, by carefully choosing the transistor width, we avoid the diffusion rounding in MEM4 × 4, and for other cells, chainings are split when the neighboring transistors' width are different, this is a trade off between area and variation. For 45_SS, the MUXes are implemented using other logic cell in standard cell library, leading to corner rounding problem. This is not the case for MUXes in 45_CR and 45_NCR. However, the buffers (include buffers and tri-state buffers) are the major contributors to the corner rounding variation for both 45_CR and 45_SS, as shown in Fig. 8.

Figures 9(a), 9(b), and 9(c) also give more insight on how improvement on functional yield is achieved. The RPD caused failure rate strongly relies on layout style and area: the sparser the layout style and the smaller total area of the layout, the lower failure rate it has. 45_NCR, 45_CR have sparser layout than 45_SS. This is evidenced by the different implementation of INVX4 cell, whose failure rate is 5.63×10^{-11} for 45_SS while 5.56×10^{-11} for 45_CR (45_NCR). And 45_CR(45_NCR) also has smaller total area than 45_SS because of the custom designed MUX and SRAM cells. The custom designed cell also lead to the contact number reduction, and then reduce the failure rate. From Fig. 9(d), 9(e) and 9(f), we can see the major contribution of contact caused failure is from configuration SRAM bit.

Table 7. Area comparison with GILES on POWELL.

Tile	Active area	Routed area	Normalized
180_CC	12921	14782	1.00
180_CR	14077	16104	1.09
180_NCR	15257	17454	1.18

5.3. Area result

The area results of tiles for TSMC 0.18 μm are shown in Table 7. From the result, we can see the micro-regular tile incurs 9% overhead against GILES. If corner rounding is not permitted, the effect of diffusion splitting introduces another 9% overhead.

6. Conclusions

In this paper, we validate the micro-regular design fabric, recently proposed to improve the printability of logic circuit implementation, in the context of FPGAs. By developing an automated micro-regular FPGA tile generator, and evaluating it on the academic FPGA called POWELL, we find that the area overhead of the micro-regularity is below 10%. On the other hand, significant advantage on transistor variation reduction of 33%, and probability of failure reduction of 21.2%, can be achieved in addition to better printability. If further overhead of 10% can be recovered by enhanced resolution, we can achieve variation reduction of 93.8%, and probability of failure reduction of 16.2%. We therefore conclude that this is a promising direction that warrants further commercial effort.

References

[1] Jhaveri T, Rovner V, Liebmann L, et al. Co-optimization of circuits, layout and lithography for predictive technology scaling beyond gratings. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29(4): 509

[2] Kuon I C. Automated FPGA design, verification and layout. Master Thesis of Applied Science and Engineering, University of Toronto, 2004

[3] Egier A C. Enhancing and using an automatic design system for creating FPGAs. Master Thesis of Applied Science and Engineering, University of Toronto, 2005

[4] Padalia K, Fung R, Bourgeault M, et al. Automatic transistor and physical design of FPGA tiles from an architectural specification. FPGA, 2003: 164

[5] Phillips S, Hauck S. Automatic layout of domain-specific reconfigurable subsystems for system-on-a-chip. FPGA, 2002: 165

[6] Wu J C H, Aken'Ova V, Wilton S J E, et al. SoC implementation issues for synthesizable embedded programmable logic cores. IEEE Custom Integrated Circuits Conference, San Jose, CA, 2003: 45

[7] Aken'Ova V C, Lemieux G, Saleh R. An improved "soft" EFPGA design and implementation strategy. Proceedings of the IEEE on Custom Integrated Circuits Conference, 2005: 179

[8] Padalia K. Automatic transistor-level design and layout placement of FPGA logic and routing from an architectural specification. Bachelor of Applied Science and Engineering Thesis, University of Toronto, 2001

[9] Fung R. Optimization of transistor-level floorplans for field programmable gate arrays. Bachelor Thesis of Applied Science and Engineering, University of Toronto, 2002

[10] Chan A B Y. Automating transistor resizing in the design of field programmable gate arrays. Bachelor of Applied Science and Engineering Thesis, University of Toronto, 2003

[11] Kuon I, Egier A, Rose J. Design, layout and verification of an FPGA using automated tools. FPGA '05, 2005: 215

[12] Jhaveri T. Regular design fabrics for low cost scaling of integrated circuits. Doctor Thesis of Philosophy, Carnegie Mellon University, 2009

[13] Jhaveri T, Rovner V, Pileggi L, et al. Maximization of layout printability/manufacturability by extreme layout regularity. Journal of Micro/Nanolithography, MEMS and MOEMS, 2007, 6(3):031011

[14] Cheng E Y C, Sahni S. A fast algorithm for transistor folding. VLSI Design, 2001, 12(1): 53

[15] Uehara T, vanCleemput W M. Optimal layout of CMOS functional arrays. Piscataway, NJ, USA, DAC, 1979: 287

[16] Hwang C Y, Hsieh Y C, Lin Y L. A fast transistor-chaining algorithm for CMOS cell layout. IEEE Transactions on CAD of Integrated Circuits and Systems, 1990, 9(7): 781

[17] Chen Xun, Zhu Jianwen. Transistor permutation for better transistor chaining. IEEE 8th International Conference on ASIC, 2009: 1276

[18] http://www.ciranova.com/products/pycell_studio.php

[19] http://www.ciranova.com/

[20] Zhai B, Hanson S, Blaauw D, el al. Analysis and mitigation of variability in subthreshold design. San Diego, CA, USA, ISLFED, 2005: 20

[21] Ghaida R S, Gupta P. A framework for early and systematic evaluation of design rules. ICCAD, 2009: 615

[22] Chan T B, Gupta P. On electrical modeling of imperfect diffusion patterning. 23rd International Conference on VLSI Design, 2010: 224

[23] Gupta P, Jeong K, Kahng A B, et al. Electrical metrics for lithographic line-end tapering. Proc SPIE, 2008, 7028: 70283A

[24] Stapper C H. Modeling of integrated circuit defect sensitivities. IBM Journal of Research and Development, 1983, 27: 549

[25] http://www.eda.ncsu.edu/wiki/freepdk45

[26] Nangate open cell library v1.3. http://www.si2.org/openeda.si2.org/projects/nangatelib

[27] International technology roadmap for semiconductors. http://www.itrs.net/