Automatic Obfuscated Cell Layout for Trusted Split-Foundry Design

Carlos Tadeo Ortega Otero, Jonathan Tse, Robert Karmazin, Benjamin Hill and Rajit Manohar
Computer Systems Laboratory, Cornell University
Ithaca, NY, U.S.A.
{cto3,jon,rob,ben,rajit}@csl.cornell.edu

***Abstract*—We present *split-cellTK*, a tool that automates the obfuscation of split-foundry layout, in which an untrusted foundry fabricates the devices (FEOL) and a trusted foundry completes the design with metalization (BEOL). *split-cellTK* does not alter the design netlist for obfuscation—it accepts an arbitrary transistor-level netlist as input. By obfuscating the the organization, placement and connectivity of devices in the FEOL, *split-cellTK* increases the difficulty of a reverse engineering effort. We evalute two FEOL obfuscation schemes: uniform layout and layout with random spacing. We extend the set of metrics previously defined in the literature to capture the benefits of obfuscation at the cell level. We use these metrics to evaluate the degree of obfuscation provided by our schemes and present the power, area, and throughput overheads for our obfuscation schemes. Finally, we present measured results for an asynchronous FPGA fabricated in a 65 nm split-foundry technology using our tool.**

I. Introduction

Cost-effective access to state-of-the-art semiconductor manufacturing has become a global commodity. In most cases, designers' intellectual property (IP) must leave their control for fabrication, exposing it to risks such as hardware piracy [1], reverse engineering [2], or even modification with malicious intent [2]. A number of techniques, including obfuscated 3D integrated circuits and split-foundry fabrication, have been proposed to mitigate these risks [3-6].

This work primarily concerns the split-manufacturing process, in which only the Front End of Line (FEOL) is sent to an untrusted foundry, limiting exposure of the IP. The unfinished wafers are then transferred to a trusted foundry to manufacture the Back End of Line (BEOL) and complete the fabrication process. Preliminary work in this area suggests that split-foundry fabrication is viable and does not itself present significant overheads in performance [7].

However, split-foundry fabrication is not a complete security solution. Malicious agents with access to a finished device can make use of a wide array of reverse engineering techniques to extract the complete netlist of a design [2]. Even guaranteeing the physical security of the completed die, i.e. only exposing the FEOL, is not a sufficient countermeasure against reverse engineering efforts [8]. Designers must take additional steps to obfuscate the FEOL and increase the effort required to obtain the design netlist.

A typical ASIC flow, which relies on a standard cell library, increases the amount of information available to the attacker, especially if the cell library is known. To reduce information leakage, designers can map their design to a restricted library or introduce dummy (decoy) cells. Generalizing this approach, one can map the design to a set of isomorphic cells, i.e. cells which share identical FEOL and have functionality defined by the BEOL connections [9]. These approaches result in a significant difference between the original and obfuscated gate-level netlists. In order to obtain part or all of the gate netlist of an ASIC circuit, an attacker needs to find the cell boundaries, determine intra-cell connectivity and infer the design functionality.

We present an alternative approach to obfuscate layout that places no restrictions on cell libraries. The input to our toolflow is an arbitrary, transistor-level netlist from which we create standard-cell-like units on-demand. This avoids mapping the design to a fixed or static standard cell library, which an attacker might have access to, as well as avoiding any restrictions on the design netlist. Our toolflow affords us control at the device level to disrupt an attacker's ability to infer the netlist. Note that this is not a full-custom flow, which also provides this level of control, but with increased design time and cost.

Our tool, *split-cellTK*, automates FEOL obfuscation against reverse engineering, providing an additional level of trustworthiness for a split-foundry process. For a given netlist, *split-cellTK* can produce multiple FEOLs using different obfuscation strategies, described in Sec. III. Our strategies are aimed at increasing the difficulty of a reverse engineering attempt to discover cell boundaries, functionality, and interconnectivity. Using an asynchronous FPGA netlist as a testbed, we evaluate the throughput, energy, and area overheads of our obfuscation techniques in Sec. IV. FPGAs can provide protection for the application IP, since the application is not introduced until after the manufacturing process [10], but the FPGA design itself is still vulnerable.

We compare our obfuscated implementations against an implementation of the original design netlist, as opposed to starting with a netlist that has been mapped to a restricted cell library. To our knowledge, we are the first to take this approach to split-foundry obfuscation. Sec. IV presents measured results from a 65 nm split-foundry test-chip as well as a first order evaluation of the trustworthiness of our design.

II. Background

The security or trustworthiness of split-foundry fabrication hinges on the assumption that the untrusted foundry manufacturing the FEOL mask cannot determine the function of the various circuits in an IC design. However, a motivated attacker could infer details of the netlist by inspecting the FEOL in isolation. The FEOL contains information about device parameters, device location and metalization stack via placement, which an attacker can use to infer BEOL connectivity. Because of this, Rajendran et al. suggest that split manufacturing is not secure [8]. However, the authors based their conclusion on the assumption that a non-obfuscated layout is used and that the untrusted foundries have access to the FEOL *and* the first few BEOL metal layers. In contrast, we obfuscate the FEOL and assume the untrusted foundries have no BEOL access, i.e. no information beyond via cuts for Metal 1 connectivity.

Researchers have previously demonstrated that split manufacturing as a process is viable for large digital systems [7,11] and for analog devices and SRAM cells [12]. To evaluate the trustworthiness of obfuscated designs, Jagasivamani et al. proposed four first-order metrics [9]:

1. Neighbor Connectedness (NC) measures the interconnectivity of neighboring cells, and tests the effort to find inter-cell connectivity and functionality of the design. For any cell i, $N(i,r)$ is the count of its neighbors in radius r, and $I(i,r)$ is the number of neighbors cell i is connected to in radius r. A low NC value for small r suggests either low inter-cell connectivity or a low degree of cell clustering, both of which hinder the ability of an attacker to infer inter-cell connectivity.

$$NC(r) = \frac{\sum_i I(i,r)}{\sum_i N(i,r)} \tag{1}$$

2. Composition of Hint Cells (CHC) is a coarse measure of how the cell population in a design might provide clues as to the design's functionality. For example, XOR-type cells might *hint* at a cryptographic function while adders suggest an arithmetic logic unit. U is the multiset of cells in the design, i.e. it contains all instances of cells in the design, allowing for multiplicity. H is the sub-multiset of all hint cells, and H' is the set of unique hint cells. Designs with high CHC indicate that despite lack of cell connectivity, FEOL layout leaks crucial information to reverse engineer a design.

$$CHC(H) = \frac{|H|}{|U|}\left(\sum_{i\in H'}\left|\frac{|i\cap H|}{|H|} - \frac{1}{|H'|}\right|\right) \tag{2}$$

3. Entropy (E), specifically the Shannon entropy, measures the level of disorder in the FEOL mask. $N = |U'|$ is the total number of cell types and $p_i = |i|/|U|$ is the fraction of the design composed of cell type i. Entropy estimates the overall order of the FEOL. A uniform design, utilizing fewer types of cells to implement all functions, results in low entropy and would leak little information about the design functionality. A designer should aim to minimize entropy, and thus increase the difficulty of any reverse engineering effort.

$$E = -\sum_{i=1}^{N} p_i \log(p_i) \tag{3}$$

4. Cell-Level Obfuscation, as proposed by Jagasivamani et al., is less of a measure and more of a practice. We propose several metrics in Sec. IV, to better quantify cell-level obfuscation. These metrics focus on the process of inferring cell boundaries, which is a common first step during reverse engineering. Typical techniques include identifying boundaries by repeated features in the FEOL, by transistor proximity, etc.

III. Obfuscation with *split-cellTK*

Our trusted split-foundry toolflow is based on *cellTK* [13]. Unlike a traditional ASIC flow, which relies on a predefined and characterized cell library, *cellTK* generates cells on-demand from any arbitrary transistor-level netlist. Note that there is no gate-mapping step—the cells which *cellTK* generates implement the circuits as described in the original transistor-level netlist. By default, transistors are placed to maximize both diffusion sharing as well as polysilicon gate sharing. The cells generated by *cellTK* are compatible with commercial place and route tools, which are used to assemble the full design [13].

In order to support various FEOL obfuscation strategies, we developed *split-cellTK* as an extension to the existing *cellTK* framework. *split-cellTK* can generate multiple physical layouts for each unique cell without modifying the circuit topology. It implements our obfuscation strategies by modifying the original transistor placement method in *cellTK*. These strategies, described in more detail below, are designed to combat the following bottom-up reverse engineering attack model:

1) Find standard cell boundaries.
2) Find intra-cell connectivity to infer cell functionality.
3) Infer design functionality though inter-cell connectivity, cell placement, and cell composition.

We propose two FEOL synthesis techniques to improve design obfuscation over an unaltered *Baseline* configuration. Both techniques disallow diffusion sharing and group transistors into nMOS/pMOS pairs, which may or may not share a gate. The first technique, *Uniform*, implements uniform transistor sizing and placement pitch. The second technique, *Random*, retains the design netlist sizing, but randomizes transistor placement spacing. We illustrate the effects of each technique on the 3-input Muller C-Element, a non-combinational gate with a feedback inverter used as a keeper, in Fig. 1a.

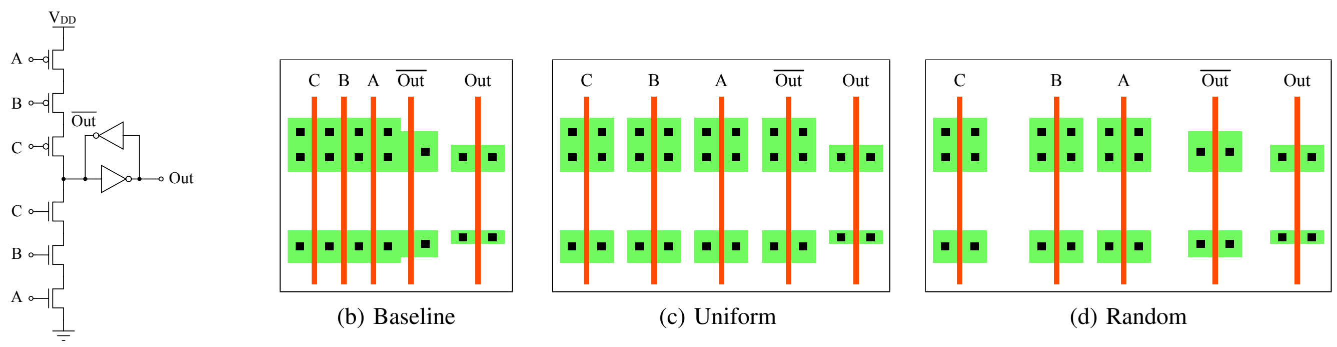
A. Baseline

By default, *cellTK* attempts to compact cell area by maximizing diffusion and polysilicon gate sharing among transistors, as is typically done in a commercial cell library. Fig. 1b shows the maximal diffusion- and gate-sharing configuration generated by *cellTK*. While not strictly an obfuscation technique, *Baseline* is slightly less vulnerable than a comparable ASIC flow. Asynchronous circuits, which are primarily composed of custom state-holding gates, typically do not match common standard cell commercial libraries and thus are difficult for an attacker to identify and reverse engineer to a netlist. *cellTK* generates custom cells on demand from the original netlist, further increasing the difficulty of identifying cells and their functionality.

B. Uniform

An attacker could identify cell boundaries by looking for breaks in diffusion, use that information to infer a set of cells, and guess cell functionality by identifying a set of hint cells. To thwart this approach, the *Uniform* strategy aims to erase identifiable cell boundaries from the FEOL using three techniques:

1) Uniform Sizing and Spacing — The original transistor-level netlist is modified such that all transistors have exactly the same dimensions, save those implementing keepers in dynamic gates. This sizing difference is the only netlist change, making the gate-level netlist otherwise identical to the original design netlist. Furthermore, all transistors are aligned to the global routing and cell placement grid. Thus, there is no discernible boundary between abutting cells.



(a) Schematic (b) Baseline (c) Uniform (d) Random

Fig. 1: 3-Input C-Element FEOL

2) Dummy Transistors — We add dummy transistors into cells to equalize the number of nMOS and pMOS devices. The source and drain of these transistors are connected to V_{DD} and *GND* as appropriate, and the gate is typically shared with the other transistor in the pair. The size of dummy transistors can take only two values: *logic transistor size* or *minimum transistor*. We use minimum transistor size to build weak-feedback keepers ($\leq 0.5\,\%$ transistors). The addition of these transistors further complicates guessing at both intra- and inter-cell connectivity and can cause aliasing problems when trying to infer cell boundaries.

3) Dummy Cells — After cell placement, any gaps of empty space which could potentially reveal information about cell boundaries must be filled with dummy cells. *split-cellTK* implements this by adding enough dummy inverter chains to fill the gaps in a post-placement step. As an added measure, the power grid pitch remains uniform across the die.

Fig. 1c shows the effects of uniform sizing and spacing. The feedback inverter is sized differently, but this sizing is consistent across all such inverters. Fixing transistor sizing has the side effect of altering the delay of each gate with respect to the original design specification. However, since our circuits are Quasi Delay-Insensitive (QDI) [14], they are tolerant to variations in delay. QDI circuits allow us to implement aggressive obfuscation techniques by automating netlist transformation and cell generation with *split-cellTK*. There are overhead costs to throughput, area, and energy from obfuscation, but correctness is preserved without additional designer effort such as retiming.

C. Random

To reduce the performance overheads of the *Uniform* strategy, we developed the *Random* obfuscation technique. Unlike *Uniform*, we do not modify transistor dimensions, which preserves the drive strengths intended by the designer. The key obfuscation is still related to obscuring cell boundaries. However, instead of erasing all cell boundaries, as per the *Uniform* strategy, *Random* introduces "fake" cell boundaries. As in *Uniform*, we disallow diffusion sharing and add dummy transistors to equalize the number of nMOS and pMOS devices. To create fake cell boundaries, *split-cellTK* randomly chooses the spacing, m, between each pair of nMOS and pMOS transistors from the following distribution: $m \in \{1, 1, 1, 1, 2, 2, 3, 3, 4, 4, 5\}$. $m = 1$ is the minimum spacing spacing allowed by the DRC rules, whereas $m = 4$ specifies four wiring tracks of spacing. By adjusting the given spacing distribution, one can control the average size of the cells, thereby controlling the area overhead compared to *Baseline* and *Uniform*. A sample physical implementation of our *Random* strategy is shown in Fig. 1d.

The *Random* technique does offer a layer of obfuscation over *Baseline* in that it creates decoy cell boundaries. However, an attacker can leverage the non-uniform transistor dimensions—unchanged relative to *Baseline*—to infer and reverse-engineer a set of cells.

IV. Evaluation

We evaluated our three FEOL synthesis techniques by synthesizing the island-style asynchronous FPGA (*AFPGA*) developed by Hill et al. [7] in a 65 nm split-foundry process. Results presented below represent measurements of our fabricated *Baseline* design and simulations of our *Uniform* and *Random* designs unless otherwise indicated. Our evaluation focuses on a *single AFPGA* tile, as they are all identical. The intra-tile space can be filled with dummy cells in a way that the layout mask hides the regularity of an FPGA structure. Our split-foundry process has the same FEOL design rules for both the untrusted and the trusted foundry, but *cellTK* can also use a DRC deck composited from the most restrictive rules from the two processes to increase yield and reliability [7]. From our initial 52k transistor netlist, *cellTK* generated 147 unique cells, which are then used in a commercial place and route flow to assemble a single FPGA tile. Of these 147 cells, 17 were universal gates (NOR, NAND, etc.). Thirty were non-combinational, state-holding gates that would not be found in a standard cell library. While researchers have augmented standard cell libraries with C-elements [15], which are common in asynchronous logic, 24 of the 30 non-combinational cells were not C-elements.

A. Performance Evaluation

Three main factors impact the performance of *Random* and *Uniform* relative to *Baseline*: 1) increased area, 2) increased wire length, and in the case of *Uniform*, 3) altered gate drive strength due to sizing changes.

A single *AFPGA* tile implemented with the *Baseline* technique occupies $58\,800\,\mu\mathrm{m}^2$. In contrast, the area of the *Random* and *Uniform* tiles were both approximately $94\,800\,\mu\mathrm{m}^2$, a 1.6x increase. Fig. 2 shows the area overhead per cell. The bars represent cell area overhead normalized to *Baseline* and are sorted in ascending order left-to-right by the *Baseline* cell area. For small cells (cell ids 1 to 50 in Figure 2), the overheads of

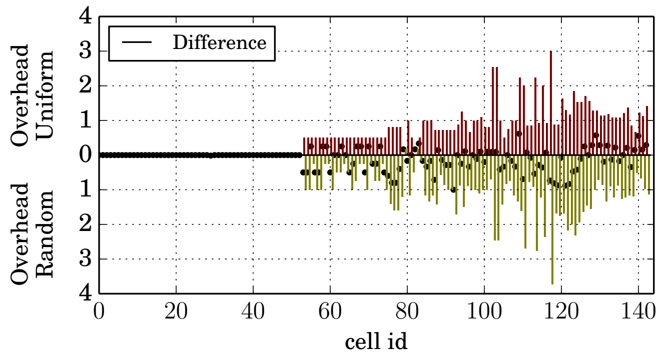
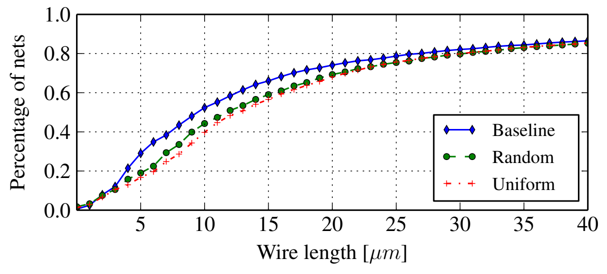
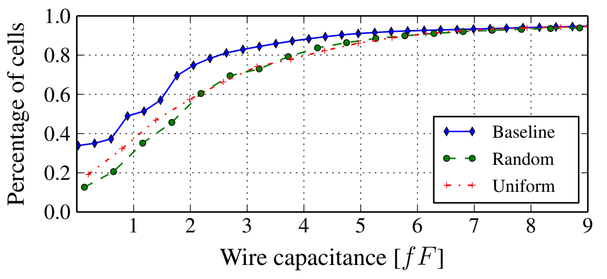


Fig. 2: Per-Cell Area Overhead, Sorted by *Baseline* Cell Size

using *Random* or *Uniform* is negligible. However, for larger cells, obfuscation overhead is significant. The maximum area overheads for *Uniform* and *Random* cells were 3× and 3.5× respectively. Overall, *Random* has more negative impact on the cell size compared to *Uniform* layout due to the increase in transistor spacing.



(a) Wire Length Cumulative Distribution Function



(b) Wire Capacitance Cumulative Distribution Function

Fig. 3: *AFPGA* Tile Wire Overheads

Wires shorter than 5 µm typically implement intra-gate nets and are largely unaffected by our obfuscation schemes. Medium length connections implementing local inter-cell connectivity range from 5 µm to 27 µm. Local interconnect in digital design often has the most switching activity, so an increase in wire length here increases wiring capacitance and thus switching energy. Fig. 3a and Fig. 3b illustrate how *Random* and *Uniform* techniques increase the length and capacitance of short and medium interconnect.

We evaluated the performance of our obfuscated circuits by simulating an adder within the logic block of our *AFPGA* tile. Extracted simulations of the full AFPGA tile are intractable, and the performance is heavily dependent on benchmarks—

TABLE I: Simulated Overhead of Obfuscation in an Adder

Technique	Area [µm^2]	Power [mW]	Energy [pJ]	Perf. [MHz]
Schematic		0.147	0.262	560
Baseline	462	0.146	0.257	568
Uniform	717	0.149	0.307	486
Random	760	0.164	0.303	542

most of which require more than one tile to implement. Results are shown in Table I. "Schematic" refers to a SPICE simulation of the circuit with an estimated lumped wire capacitance. *Baseline*, *Uniform*, and *Random* use parasitics extracted from layout, including coupling capacitance.

Uniform and *Random* designs have 55 % and 64 % area overhead respectively compared to the *Baseline*. *Uniform* has a 17 % throughput degradation while *Random* has only a 5 % decrease, but a 12 % increase in power consumption. The decreased performance of *Uniform* is a direct result of the fixed transistor size, which negatively effects the drive strength of some series transistor stacks.

We fabricated and tested a 10 by 10 tile *AFPGA* test chip using our *Baseline* technique in both a 65 nm split-foundry process and a traditional monolithic process. Our *AFPGA* has on-die frequency taps that measure the switching activity in the routing fabric. We evaluated performance by programming the *AFPGA* with a token loop, which yields the maximum operating frequency. For our self-timed, i.e. clock-less, QDI circuits, performance scales automatically with voltage, allowing us to easily characterize our foundry process at different voltages. Measurement of the *Baseline* design for the split- and monolithic-foundry processes shows a close match in performance, as shown in Fig. 4. At a nominal V_{DD} of 1.2 V, the monolithic-foundry chips[1] operates at an average of 524 MHz while the split-foundry dies[2] operate at 545 MHz.

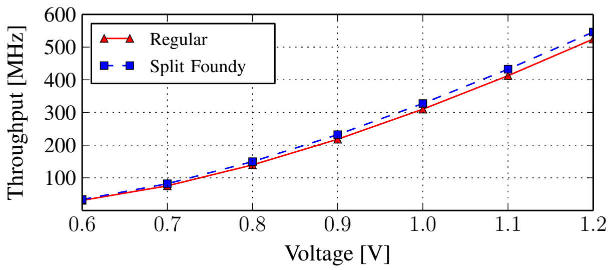


Fig. 4: Measured *Baseline* Test-Chip *AFPGA* Throughput

B. Trustworthiness

Our attack model assumes that an attacker only has access to the FEOL, i.e. no metalization geometry aside from the diffusion/polysilicon contacts to Metal 1. Thus, the only information available to an attacker would be the placement of transistors, any polysilicon routing, contact location and physical dimensions, and transistor characteristics such as width, length, and doping profile. From these features alone, the attacker must extract cell boundaries, determine the connectivity of the transistors in the cell to determine cell

[1]$n = 25$, σ=12.8 MHz
[2]$n = 25$, σ=21.7 MHz

function, and then determine the inter-cell connectivity. This type of reverse engineering attempt will reveal the original design netlist and thus the designer intent. We assume an attacker has access to all production EDA and CAD tools, including commercial and academic software such as *split-cellTK*. In this section, we examine the relative increase in reverse engineering difficulty presented to the attacker by our various obfuscation techniques.

Cell proximity and connectivity are measured by Neighbor Connectedness (NC), as described in Sec. II. NC is a measure of the overall clustering behavior of cells and not a direct attack, as calculating NC requires BEOL connectivity information that is unavailable to an FEOL-only attacker. For all our *AFPGA* implementations, the NC for small radii is significantly lower than previous results published by Jaqasivamani et al., who report a NC ranging 28 % to 50 % [9]. We attribute this difference in NC to the relatively high complexity of *cellTK* cells (1-50 transistors), which reduces overall inter-cell connectivity. Cell size is also larger than a traditional standard cell flow, so the degree of clustering within an absolute radius is less. The NC values for our *AFPGA* with and without obfuscation are shown in Fig. 5. A *Baseline* tile has a NC of 30 % for a small radius ($r \leq 3$ µm), and all our designs have an NC less than 5 % for $r \leq 20$ µm. If we include dummy transistors while computing the NC, our *Uniform* design has a maximum NC of 14 % for a radius of 2 µm. This reduction in NC is in part because of the fine granularity of dummy cells and the relatively low connectivity that the dummy stacks have.

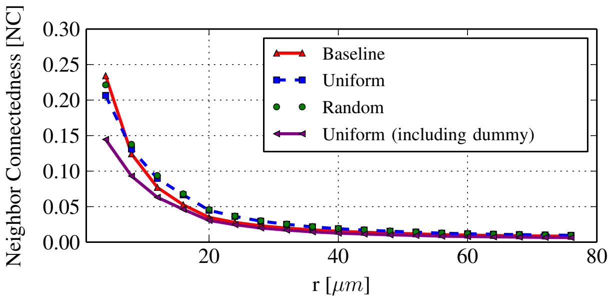


Fig. 5: Neighbor Connectedness (NC)

CHC is a rough estimate of the amount of information available to an attacker given a particular cell composition—a high XOR cell count might suggest a cryptographic accelerator, for example. However, CHC is extremely design-dependent and requires in-depth design knowledge to compute. At the point which an attacker could calculate CHC, the obfuscation of cell functionality has already been compromised. To that end, isomorphic cells or techniques such as *Uniform* or *Random* that increase obfuscation of cell functionality are appropriate countermeasures. Note that CHC remains the same between *Baseline*, *Uniform*, and *Random*, as the cell functionality does not change. We report CHC values for our design here for completeness, but it is difficult to make cross-design comparisons for a metric so dependent on the design.

Routing circuitry is a large fraction of an *AFPGA* design—the muxes which implement our routing fabric account for 38 % of all cell instances. The cells implementing the *AFPGA* latches and configuration memory comprise 30 % of all cell instances. Our *AFPGA* uses a scan-chain-based configuration memory, so these cells are mostly latches. Arithmetic and other logic cells (ADD, NOR, NAND, NOT, etc.) comprise only 18 % of the total number of instances. We not only compute CHC for particular cells as in [9], but also for multiple hint cell types an attacker could use. Table II shows the CHC for multiple hint cell combinations. Combination 1 is the one used in [9].

TABLE II: Composition of Hint Cells (CHC)

	Hint Cell 1		Hint Cell 2		Hint Cell 3		CHC
1	0%	XOR	14%	Latch	1%	ADD	0.18
2	38%	MUX	14%	Latch	12%	Conf A	0.28
3	38%	MUX	30%	Latch and Mem.	18%	Arith/Logic	0.21

Instead of using Entropy metric proposed by Jagasivamani et al. [9], we use a Normalized Entropy ($\bar{E}$) because our basic symbol for representing information varies. To compute $\bar{E}$, we first reduce the physical layout to *layout units*, each of which represents the FEOL characteristics of the minimum distinguishable group of transistors. Groups are defined by shared diffusion or aligned polysilicon gates and by transistor stack proximity. For *Baseline*, there is a one-to-one correspondence between groups and cells as cell boundaries are easily extracted from the layout. Since *Uniform* and *Random* are designed to obfuscate cell boundaries, groups are simply pairs of nMOS/pMOS transistors. Each layout unit reduces these transistor groups into a tuple of defining characteristics such as transistor/via count, size, placement as well as information about diffusion and polysilicon connectivity—any identifying physical feature of the group. These tuples are then mapped onto an alphabet of *layout signatures*. We seek to minimize the value of the Normalized Entropy. Our normalized entropy measures the disorder of the FEOL, designs with low normalized entropy yield little information about the cell boundaries and the design functionality, increasing reverse engineering effort required for an attacker.

Although *split-cellTK* is not a standard cell flow, it does place the cells it generates into rows. Each cell in the row has a string comprised of *layout signatures*. We can then concatenate all strings in a row to obtain a *row signature*, and a *design signature* is simply the collection of all row signatures. $\bar{E}$ is defined in Eq. 4, with E as the Shannon entropy of the layout signatures. S is the multiset of all layout signatures, and S' is the set of all unique layout signatures.

$$\bar{E} = \frac{|S'|}{|S|} E \tag{4}$$

Table III shows the characteristics of the multiple *design signatures*. For this analysis, we omit *Random* as the results would be similar to that of the more obfuscated *Uniform*. For *Baseline*, the set of cells is the same as the set of layout signatures, which means that for *Baseline* the $\bar{E}$ is the same as the entropy proposed by Jagasivamani et al. [9]. For *Uniform*, a single cell contains multiple signatures, so $|S|$ is significantly higher than that of *Baseline*. To account for this, we use the normalization factor $|S'| / |S|$ to directly compare *Baseline* and *Uniform* designs. This is a direct consequence of *Uniform* disallowing diffusion sharing, resulting in a maximum group size of two transistors.

Our *Uniform* implementation has an $\bar{E}$ of 1.24—considerably lower than *Baseline*. By examining the layout signatures of our *Uniform* implementation, we found that the location of polysilicon contacts leaks a significant amount of information. If we align the contacts to wire-track pitch, we can reduce the $\bar{E}$ even further to 0.35 (Uniform-TP). Further reduction is possible by only allowing contacts above the pMOS, below the nMOS, or between the pair (Uniform-F). Uniform-F is the most "secure," but over-constrains the router and prohibitively increases the cell failure rate.

TABLE III: Design Signature Characteristics

Design	Instances	Cells	$\lvert S\rvert$	$\lvert S'\rvert$	E	$\bar{E}$	CTMFP
Baseline	8752	147	147	147	3.81	3.81	5k
Uniform	35360	150	814	236	4.31	1.24	29k
Uniform-TP	35360	150	814	88	3.32	0.35	190k
Uniform-F	35360	150	814	54	2.34	0.15	810k

An attacker with access to *split-cellTK* and knowledge about our design methodology would be able to generate a cell library similar to that used in our designs. By matching the layout signature of these generated cells or templates against the design signature, an attacker might be able to identify cells within the design. To measure the effectiveness of this approach, we match cells from the actual library used in the design against the design signature and measure the false positive rate, or Template Matching False Positives (TMFPs). A high false positive rate reduces the effectiveness of this matching attack on the design signature. TMFP is defined below, and Fig. 6 shows TMFP for cells with different transistor counts. Please note that our uniform layout tool does not produce cells with odd number of transistors and hence their TMFP is 0.

$$TMFP = |N_{\text{matched}}(c) - N_{\text{real}}(c)| \tag{5}$$

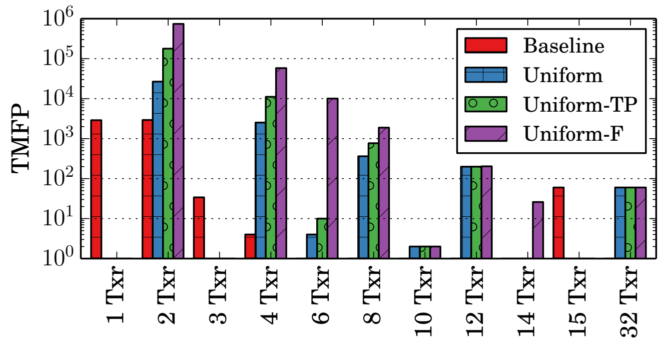


Fig. 6: Template Matching False Positives (TMFP)

To obtain N_{matched}, we perform a sliding window match of cell signatures against the design signature. N_{real} represents the actual number of instances of each one of the cells. Table III shows the cumulative TMFP (CTMFP): $CTMFP = \sum_{c \in U'} TMFP$. Table III shows almost a 6x increase in the CTMFP from *Baseline* to *Uniform*. Constraining contact placement results in a 38x and a 162x increase over *Baseline* for Uniform-TP and Uniform-F, respectively. The high value of CTMFP reinforces the hypothesis that finding cell borders is significantly difficult for an attacker operating on *Uniform*. This increase is in part due to the fact that we add dummy transistors to ensure an nMOS:pMOS ratio of 1:1 in *Uniform* cells.

V. Conclusion

We presented *split-cellTK*, a tool designed to automate the obfuscation of an arbitrary transistor-level netlist. *split-cellTK* does not place any restrictions on the netlist or force mapping to a restricted or isomorphic set of cells. Instead, our tool generates an on-demand cell library with restrictions on the placement and sizing of the transistors within each cell, including the addition of dummy transistors ensure a 1:1 ratio of nMOS to pMOS transistors in each cell. *cellTK* and *split-cellTK* have been used in the fabrication of FPGA test chips in 65 nm and 130 nm.

Using *split-cellTK*, we implemented two FEOL obfuscation schemes, *Uniform* and *Random*, and compared their performance to a *Baseline* implementation using traditional VLSI metrics such as throughput, power, and area. We extend the metrics in the literature to evaluate the trustworthiness of our obfuscation techniques. We propose 4 metrics to evaluate the obfuscation of the cells layout and global place-and-route: Neighbor Connectedness (NC), Composition of Hint Cells (CHC), Normalized Entropy ($\bar{E}$), and Template Matching False Positive Rate (TMFP). Note that these metrics do not provide a binary test as to whether or not a design is secure. Rather, they provide designers with quantitative metrics to integrate into their own evaluations of design security.

Our recommendation to designers is that if the split-foundry model is viable, i.e. attackers will only have access to the FEOL and be denied access to a completed die, additional effort should be spent on obfuscating the design FEOL. We present quantitative results that show obfuscation of the FEOL at the cell level significantly increases the difficulty of an attack, albeit at the cost of a 55-65% increase in area and up to 17% decrease in performance.

References

[1] J. A. Roy, *et al.* "EPIC: ending piracy of integrated circuits." "IEEE DATE," ACM, 2008.
[2] R. Torrance and D. James. "The state-of-the-art in semiconductor reverse engineering." "IEEE DAC," 2011.
[3] IARPA. "IARPA Trusted Integrated Circuits program announcement." http://www.iarpa.gov/index.php/research-programs/tic.
[4] R. W. Jarvis and M. G. McIntyre. "Split manufacturing method for advanced semiconductor circuits." US Patent Office, 2007.
[5] F. Imeson, *et al.* "Securing Computer Hardware Using 3D Integrated Circuit (IC) Technology and Split Manufacturing for Obfuscation." "USENIX Security Symposium," 2013.
[6] J. Valamehr, *et al.* "A 3-D Split Manufacturing Approach to Trustworthy System Development." *Computer-Aided Design of Integrated Circuits and Systems*, 2013.
[7] B. Hill, *et al.* "A split-foundry asynchronous FPGA." "IEEE CICC," 2013.
[8] J. Rajendran, *et al.* "Is Split Manufacturing Secure?" *IEEE DATE*, 2013.
[9] M. Jagasivamani, *et al.* "Split-fabrication obfuscation: Metrics and techniques." "IEEE HOST," 2014.
[10] S. Trimberger. "Trusted Design in FPGAs." "IEEE DAC," 2007.
[11] K. Vaidyanathan, *et al.* "Building trusted ics using split fabrication." "IEEE HOST," 2014.
[12] K. Vaidyanathan, *et al.* "Efficient and secure intellectual property (IP) design with split fabrication." "IEEE HOST," 2014.
[13] R. Karmazin, *et al.* "cellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells." "IEEE ASYNC," 2013.
[14] A. Martin. "The Limitations to Delay-Insensitivity in Asynchronous Circuits." "MIT Conference on Advanced Research in VLSI," 1990.
[15] T. Y. Wuu and S. B. K. Vrudhula. "A design of a fast and area efficient multi-input Muller C-element." *IEEE VLSI*, 1993.