A Split-Foundry Asynchronous FPGA

Benjamin Hill, Robert Karmazin, Carlos Tadeo Ortega Otero, Jonathan Tse, and Rajit Manohar
Computer Systems Laboratory, Cornell University
Ithaca, NY, 14853, U.S.A.
{ben,rob,cto3,jon,rajit}@csl.cornell.edu

***Abstract*—We present the first published measurements of a complex digital integrated circuit fabricated in both standard and split-foundry processes. Our 1.3-million-transistor asynchronous FPGA operates at over 300MHz in 130nm. We discuss the challenges inherent in split design and our automated layout tools that address them.**

I. Introduction

The semiconductor industry relies heavily on access to cost-effective integrated circuit production at state-of-the-art semiconductor foundries, which have become a global resource providing service to a wide range of customers. For these customers, design intellectual property (IP) must leave their control for fabrication. This exposes the IP to certain risks such as reverse engineering [1], hardware piracy [2], and malicious modification [3]. Foundry customers, especially those involved in national defense applications, must take steps to mitigate these risks to protect their IP, economic position, and security interests.

Split-foundry fabrication has been proposed as a technique to enable use of state-of-the-art semiconductor foundries while minimizing the risks to IP or reducing production costs [4,5]. Split manufacturing separates a design into Front End of Line (FEOL) and Back End of Line (BEOL) portions for fabrication by separate foundries. An untrusted foundry performs FEOL manufacturing, then ships wafers to a trusted foundry for BEOL fabrication.

Split manufacturing introduces additional complexity to the design process, such as FEOL/BEOL mask alignment and unknown differences in electrical characteristics of structures made by two different foundries. We had a unique opportunity to evaluate these challenges by fabricating two versions of the same design: one manufactured normally in a single foundry and the other in a split-foundry process.

In this work we describe the challenges in designing for split manufacturing (Section II) and our split-foundry capable synthesis flow (Section III). We also provide quantitative analysis of the energy and performance impacts of the split manufacturing process we used (Section VI).

The design chosen for this case study is an asynchronous field programmable gate array (Section V). FPGAs can provide an additional layer of IP protection, since the application is not introduced until after the manufacturing process [6]. Our asynchronous design methodology (Section IV) is robust to gate and wire delays, accommodating any variation introduced by the split manufacturing process.

This work was supported by IARPA award N66001-12-C-2009

To the best of our knowledge, this work is the first to present measured results from a complex digital system fabricated in a split-foundry process.

II. Physical Design

We fabricated our FPGA design using two different 130 nm processes: a complete FEOL/BEOL fabrication at an untrusted foundry (Foundry A), and a split-foundry process in which the FEOL and first metal level were fabricated at Foundry A followed by BEOL manufacturing in a trusted foundry (Foundry B).

Fabrication in a split-foundry process brings a number of challenges, especially due to different design rules and electrical characteristics. While the exact details will depend on the choices of foundries and technology node, we present the challenges we faced as a case study of the design considerations inherent to split-foundry fabrication.

To select the FEOL Foundry A, a designer simply chooses the process with the most appropriate semiconductor device characteristics for the application. FEOL design rules are unaffected by any design rules from Foundry B.

However, any differences in BEOL design rules and actual BEOL implementations between Foundries A and B will result in different electrical characteristics. Common metallization stacks in state-of-the-art processes provide three to six *thin* wiring layers. Some foundries offer up to three *thick* metal layers, which have lower RC parasitics per unit length of wiring. To make the most efficient use of planar wiring resources, thin wiring layers are often well-suited for local, dense interconnect whereas the thick layers are better for long distance interconnect.

Oftentimes, most switching activity is confined to the lower level metals—typically comprised of the thin metallization layers. In our FPGA case study, the thicker, higher-level metals are used mostly for power distribution nets, so the effects of the thin BEOL characteristics dominate our performance measurements. For our processes, Foundry B offers 10% to 15% worse RC characteristics per unit area for thin wiring, but approximately 5% better RC characteristics for thick wiring.

To ease implementation, we used the union of the most restrictive design rules from Foundry A and B to ensure that our design would pass DRC in both foundries. Table I shows examples of the BEOL design rule differences between Foundries A and B, as well as our composite rule set.

TABLE I: Design rules normalized to Foundry A dimensions

Design Rule	Foundry B	Composite
Manufacturing grid	2.00	2.00
M1-M1 spacing	0.90	1.00
M6 min width	2.00	2.00
M7-M7 min spacing	0.80	1.00

Unlike metal wiring, via cuts have exact size and shape requirements which differ for each foundry. Our composite ruleset for vias implements the most strict rules for metal overhang, but we use a placeholder cell for via cuts. When emitting the final layout, we simply substitute the appropriate foundry's via cut geometry.

III. CELLTK: NON-STANDARD CELL LAYOUT

Our split-foundry toolflow is based on *cellTK* [7]. *cellTK* is an on-demand cell generator that transforms a transistor-level netlist into a design mask layout by clustering transistors into cells and producing the physical layout for each cell. Each of these "non-standard" cells is generated using two-step process: a transistor placement phase followed by an intra-cell routing phase. The cells produced by *cellTK* are compatible with standard cell place and route tools, which we use to assemble the final design.

Split manufacturing alone may not be enough to guarantee security. Attackers can use device proximity and standard cell pin placement information from the FEOL to infer BEOL connectivity [8]. *cellTK* does not use standard library cells, so it is less vulnerable to this approach. Further, it allows for full designer control over placement and routing of FEOL devices and metallization if additional obfuscation is required.

cellTK gracefully handles differences in the manufacturing grid by aligning geometry to the most conservative grid. An off-grid design complicates or outright prevents alignment of the FEOL/BEOL geometry when assembling the final wafer. Common hierarchical file formats such as GDS-II can exacerbate the grid problem if different grids are used throughout the hierarchy.

Our approach to our dual-foundry fabrication was to generate layout using geometry intended for Foundry A, then replace the BEOL with that of Foundry B. This requires layer translation steps as well as the instantiation of the appropriate vias as described earlier. During the conversion, we verify that all portions of the design satisfy our composite DRC ruleset and that all geometry is on the most conservative grid.

IV. ASYNCHRONOUS DESIGN METHODOLOGY

Our FPGA was designed using Martin synthesis [9], a procedure that decomposes a sequential program description into fine-grained parallel hardware processes. These processes are Quasi Delay-Insensitive (QDI) [10]: they operate correctly in the presence of arbitrary wire or gate delays[1]. As a result, QDI circuits are intrinsically robust and tolerant of variations in fabrication process, voltage and temperature. For example, reducing the operating voltage simply causes processes to operate more slowly. This adaptability makes self-timed circuits a good fit for BEOL variations introduced by split manufacturing.

Instead of using a global clock to synchronize data transfers, asynchronous processes communicate by passing tokens over delay-insensitive, point-to-point channels using a handshaking protocol. Processes are naturally event-driven, waiting in a quiescent state with no switching activity until a data token arrives. This is the equivalent of perfect clock-gating in a synchronous system. Inactive processes consume only leakage current.

Finally, the local synchronization behavior of self-timed circuits enables average-case system performance. Synchronous systems define their clock period by the slowest pipeline stage (even if it is rarely exercised), throttling the entire system. In asynchronous systems, system throughput is determined only by *active* hardware units. Thus, each execution path runs at maximum local throughput, yielding average-case performance over all possible paths.

[1]With the exception that the relative delay on certain wire forks must be bounded

V. ASYNCHRONOUS FPGA

A. Architecture

Our FPGA is based on the one described in [11] and [12]. It has an "island-style" architecture, as shown in Figure 1. The FPGA fabric is organized as a symmetric 2D array with "islands" of logic in a "sea" of programmable routing.

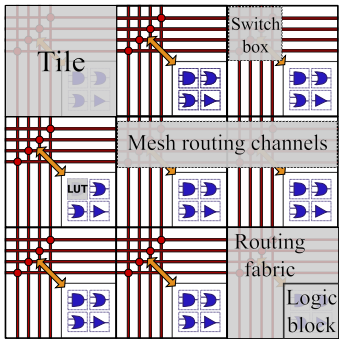


Fig. 1: "Island-style" FPGA architecture

Each tile in the array contains both a logic block and routing fabric. The FPGA is asynchronous, so all logic is implemented as QDI processes and all communication occurs via point-to-point delay-insensitive channels, as described in Section IV.

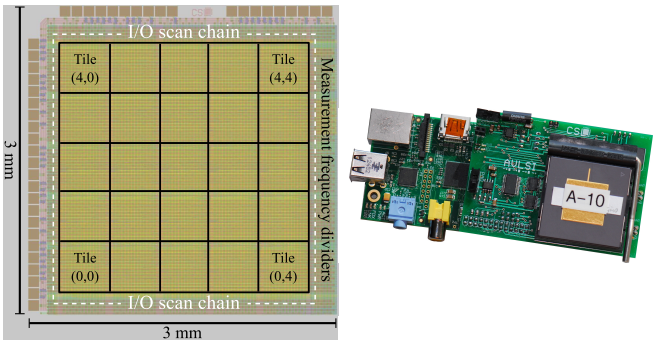
The logic block contains four lookup tables (LUTs), each of which can be programmed to implement any four-input logic function. LUT outputs can be used as inputs to other LUTs in the same tile without using any mesh routing resources. The logic block also includes units to source and sink data tokens.

Within the routing fabric, a programmable switch box statically connects inter-tile channels in a 2D mesh network. This FPGA has 32 channels/tile in each direction. Each tile also has local routing connecting the logic block to the mesh. Unlike a synchronous FPGA, the routing fabric is pipelined to

support high throughput. The asynchronous channel protocol permits this pipelining without retiming problems.

B. Physical implementation

For this chip we fabricated a 5x5 array of tiles, shown in Figure 2a. This FPGA fabric is large enough to allow us to characterize the split-foundry process.



(a) (b)

Fig. 2: FPGA layout floorplan (a) and test platform (b)

Because of pin limitations on this test chip, we use a scan chain at the array periphery to pass data to and from the FPGA. This allows us to test far more functionality than would otherwise be possible, albeit at greatly reduced throughput.

The FPGA is programmed by writing to a configuration memory, implemented as a linear scan chain. In addition to asynchronous data flow graphs, we can also map synchronous designs to the fabric using a simple conversion tool. Design place-and-route can be performed using VTR [13] (formerly VPR) or specified manually for the greatest control.

Each tile uses approximately 52k transistors. The configuration memory consumes 58% of those, routing fabric another 32%, and logic block 10%. This routing-to-logic ratio is typical of FPGAs and reflects the high cost of reconfigurability. Die area for the chip is $9\,\text{mm}^2$, with 1.33 million transistors.

C. Using Asynchronous FPGAs for Measurement

Self-timed circuits naturally run at highest possible throughput for a given environmental condition (Section IV). This behavior allows us to experimentally characterize each process technology by simply measuring the performance of a given benchmark.

Our FPGA was designed with measurement in mind. There are on-die frequency taps placed throughout the routing fabric that observe the switching activity on the mesh channels and pass it off-chip for measurement. For example, Figure 3 shows an example configuration for measuring the maximum FPGA operating frequency. Tokens are generated in one logic block, routed through a channel where they are measured, and consumed in another tile. When measuring frequency, we do not use the periphery scan chain to avoid artificially throttling the FPGA.

In addition, the programmable FPGA fabric allows for highly detailed experimentation. It is possible to choose the number of transistors active at a given time and their location on the die. This allows us to characterize both performance and power at a fine granularity.

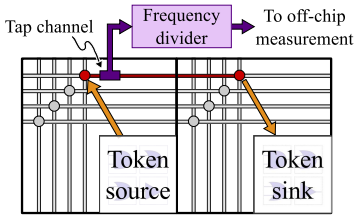


Fig. 3: Test to determine maximum FPGA operating frequency

VI. Evaluation

To evaluate the impact of split manufacturing on functionality, power, and performance, we compared split-foundry chips with those fabricated entirely by Foundry A.

Figure 2b shows our custom-designed automated testing platform, which loads configurations into the FPGA, runs test procedures, and collects data. All current measurements were performed on ceramic-packaged die at room temperature, using a separate high-precision source-measure unit.

A. Functionality/Yield

We constructed a benchmark exercising the entire configuration memory and all LUTs within the FPGA. All die[2] from both processes were found to be completely functional. This suggests that at least for our particular FEOL/BEOL pair, split manufacturing is a viable option.

B. Static power

To measure static power, we loaded the FPGA with an empty configuration. Due to the event-driven nature of self-timed circuits, the absence of data inputs guarantees there will be no switching activity.

Static power consumption depends only the FEOL transistors[3], so we should expect identical results for both fabrication techniques. Figure 4a shows that this is indeed the case: measured leakage current is closely matched across the full range of supply voltage. The average static power consumption of Foundry A chips was $1.57\,\text{mW}$ at $1.5\,\text{V}$ and $2.09\,\mu\text{W}$ at $0.7\,\text{V}$, versus $1.58\,\text{mW}$ and $2.00\,\mu\text{W}$ for the split-foundry.

C. Dynamic power and performance

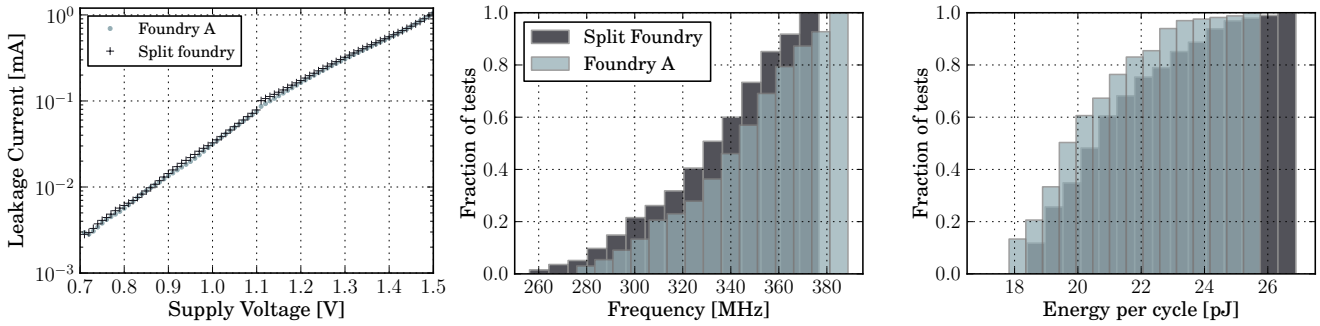
Maximum operating frequency[4] for the FPGA was measured using the configuration shown in Figure 3. We instantiated 15 of these test configurations at different locations within the fabric, in order to capture intra-die variation.

Unlike static power, maximum operating frequency depends strongly on the BEOL wiring, which has 10-15% higher parasitic capacitance for thin wiring in Foundry B. It takes

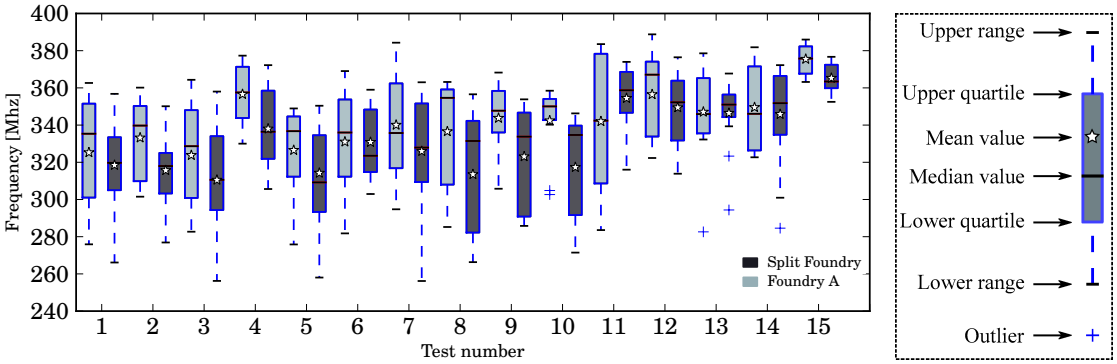
[2] 12 chips from Foundry A alone; 13 split-foundry chips

[3] Assuming power networks are adequate to handle the very small current

[4] More precisely throughput, measured in asynchronous tokens per second. This has units of Hz and is roughly equivalent to synchronous frequency.



(a) Leakage current for full FPGA (b) Test performance cumulative histogram (c) Test energy cumulative histogram



(d) Frequency distribution for individual test sites across all die

Fig. 4: Measured static and dynamic characteristics for each manufacturing process

longer to charge and discharge this capacitance, so we would expect the split process to be somewhat slower. Figure 4d shows the frequency distribution for each test site across all chips. The split-foundry chips are generally slower overall, but the variation between die is too large to allow a definitive conclusion.

The trend is more evident in a cumulative histogram of all tests, pictured in Figure 4b. The distribution for Foundry A chips is shifted higher in frequency, with a mean of 342 MHz versus 311 MHz for the split-foundry[5].

Figure 4c shows a similar shift, with split-foundry chips consuming more energy per operation on average[6]. This too follows from what we know about the BEOL, since each signal transition in the split process must charge and discharge a larger capacitance.

VII. Summary

In this paper we presented a comparative study of an asynchronous FPGA fabricated in both a standard 130 nm process and a split manufacturing process. We also described our methodology for creating split layout, as well as the challenges posed by split-foundry design.

Measurements found all split-foundry FPGAs to be fully functional, with a mean peak throughput over 300 MHz. Compared to chips from the standard process, they showed a 10% decrease in frequency and a 5% increase in energy, both likely attributable to higher capacitance BEOL wiring. Due to the inherent variation tolerance of asynchronous circuits, we were able to use the same netlist without modification for both processes. Our results demonstrate that split manufacturing is a viable technique.

[5]Foundry A: n=165, s=28.0 MHz; split-foundry: n=195, s=29.2 MHz

[6]Foundry A: $\bar{x}$=20.3 pJ, s=1.71 pJ; split-foundry: $\bar{x}$=21.2 pJ, s=1.97 pJ

References

[1] R. Torrance and D. James. "The state-of-the-art in semiconductor reverse engineering." *IEEE DAC*, 2011.

[2] J. A. Roy, *et al.* "EPIC: ending piracy of integrated circuits." *IEEE DATE*, 2008.

[3] R. S. Chakraborty and S. Bhunia. "HARPOON: An Obfuscation-Based SoC Design Methodology for Hardware Protection." *IEEE TCAD*, 2009.

[4] "IARPA Trusted Integrated Circuits (TIC) program announcement." https://www.fbo.gov/utils/view?id=b8be3d2c5d5babbdffc6975c370247a6.

[5] R. Jarvis and M. G. McIntyre. "Split manufacturing method for advanced semiconductor circuits." *US Patent 10/305,670*, 2007.

[6] S. Trimberger. "Trusted Design in FPGAs." *IEEE DAC*, 2007.

[7] R. Karmazin, *et al.* "cellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells." *IEEE ASYNC*, 2013.

[8] J. Rajendran, *et al.* "Is Split Manufacturing Secure?" *IEEE DATE*, 2013.

[9] A. Martin. "Compiling Communicating Processes for Delay-Insensitive VLSI Circuits." *Distributed Computing*, 1986.

[10] A. Martin. "The Limitations to Delay-Insensitivity in Asynchronous Circuits." *6th MIT Conference on Advanced Research in VLSI*, 1990.

[11] J. Teifel and R. Manohar. "An asynchronous dataflow FPGA architecture." *IEEE Computers*, 2004.

[12] R. Manohar. "Reconfigurable Asynchronous Logic." *IEEE CICC*, 2006.

[13] J. Rose, *et al.* "The VTR project: Architecture and CAD for FPGAs from Verilog to routing." *ACM FPGA*, 2012.