

Reverse Process Migration From 65nm to 130nm in Under Three Months

May 2010

Authors Introduction

Bob Lefferts

R&D Group Director,
Analog and
Mixed-Signal IP,
Synopsys Inc.

Neel Gopalan

AMS
Corporate
Applications Engineer,
Synopsys Inc.

The decades-old march to smaller process geometries has rarely been questioned. Normally, a design team will tackle a new project on a new, smaller-geometry process and realize the benefits of increased performance and lower cost per chip. This white paper addresses the reverse of this situation, in which a functioning 65nm analog and mixed-signal design is “blown up” to a 130nm process to help mitigate the higher mask costs of the smaller geometry.

Synopsys’ Solutions Group was asked by a customer to create a 130nm version of an existing 65nm analog and mixed-signal IP block. While it is unusual to migrate backwards to 130nm from 65nm, the customer had strong economic reasons for doing so.

The interface standard for their application was in a state of flux and the customer was expecting many respins of the design before the standard settled down. Since the development of their chip was running simultaneously with the finalization of the specification, the choice for stepping back to 130nm was primarily economic in nature. The customer’s desire to save money was based on the higher cost of 65nm masks and the higher-than-normal expected number of respins.

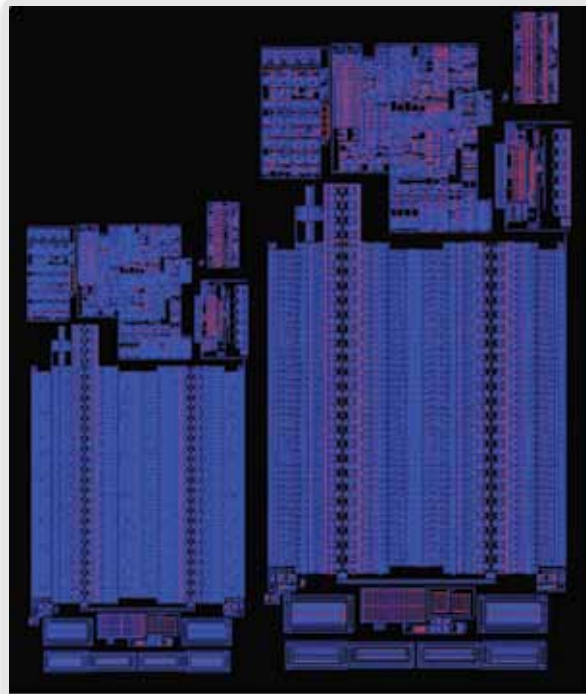


Figure 1: Layout before and after scaling from 65nm to 130nm

Background

The customer was also working with other constraints on the project that made automation a must for completing the project:

1. To meet their market window they needed the IP in three months. Earlier estimates for redoing the block from scratch put the effort at 6-8 months. It was clear that manually redoing the design was not a viable option.
2. To allow concurrent design of their chip while the IP was being ported, they needed an accurate LEF pinout up front.

The schematics for this design were already in Synopsys' Galaxy Custom Designer™ but the transistor sizes were tuned for the 65nm process, as was the matching layout. We needed to address four issues:

1. Update the schematic to the new process device sizes
2. Scale the layout and preserve the interconnect and device positions
3. Replace the design's transistors, vias and guardrings to P-Cells from the 130nm PDK
4. Verify that the scaled design was still working to spec

Scaling the Schematic

Verifying that the new design would still work meant that netlists for the new scaled devices were needed up front to allow the team to begin simulations in parallel with the layout work.

The schematic underwent scaling in Custom Designer with a TCL script that was used to adjust the widths and lengths according to the changes that would also be made in the layout. Like the other scripts used in this project, this script was relatively simple in nature. Basically, the script iterates through all of the device instances in the schematic and adjusts the parameter values for widths and lengths. The first version of the script did a wholesale update of both width and length by a scale factor of 1.4x.

Scaling the schematic first allowed us to immediately begin simulations to determine if the newly-scaled ideal design would still operate according to specification. This process was conducted using multiple HSPICE® and CustomSim™ runs utilizing the test benches from the original 65nm design. The results of these new runs were then compared to the original runs to verify proper operation.

Scaling the Layout

Moving the layout to the new process was relatively straightforward. First, the 65nm design was streamed out to GDSII and streamed into a new database, preserving the layout's hierarchy and creating a new design with no P-Cells.

The process of selecting the appropriate scaling factor was not as simple as just multiplying by two to change the 65nm device to 130nm. The original 65nm design was laid out with enough room to fit 100nm devices, but a scale factor of 1.3x times the 100nm would have resulted in many places where significant manual rework would have been required. Eventually, a scale factor of 1.4x was used and the transistors were then biased back down accordingly to bring their polysilicon critical dimension to the required 130nm. This combination resulted in a design that met both the size requirements of the design, as well as the process rules with minimal manual intervention.

The scaling was accomplished by another relatively simple TCL script that ran through every object in the layout and applied the scale factor, shifting each object and aligning it to the manufacturing grid. This script preserved the interconnections and correctly located the flat transistors, albeit still with the scaled geometry rules of the 65nm process.

Below is the pseudo-code of the scaling script's top loops:

```
For each Cell in the Design {
  For each View in each Cell {
    For each Shape in each View {
      Switch on Shape Type {
        If Instance then scaleOrigin...
        If Polygon then scalePolygon...
        ...
      }
    }
  }
}
```

The script called a number of individual scaling function calls that operated independently on the layout objects (Polygons, Lines, Rectangles, Vias and other fundamental Open Access object types). For instance, the scalePolygon function loops through the individual points in the polygon, scaling each of them in turn and aligning them with the grid:

```
scalePolygon {oaPolygon} {
  For each Point {
    Scale this Point...
    Align to Manufacturing Grid
  }
}
```

The scaling factor of 1.4x was chosen to be conservative and to most closely map the scaled objects to the manufacturing grid in the new process while minimizing the amount of manual rework. Because both instance origin and instance feature sizes have to be scaled to the manufacturing grid, it is impossible to not end up inducing opens in the design as one object could round down (e.g., shift left) and one could round up (shift right) and leave at most a two-grid-sized gap as shown in Figure 2.

These errors were easily found and fixed by running DRC on the design, then stepping through the minimum spacing violations and manually repairing them.

At this point, the LEF-accurate pin locations (if the scaled-up design were known) and an abstract was provided to the digital design team for providing the final IP LEF to the customer for their chip design.

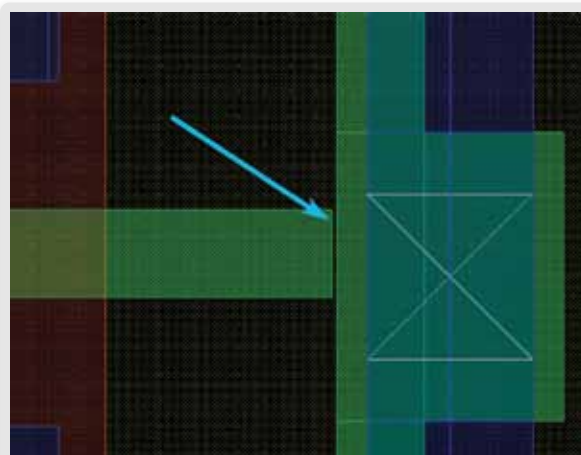


Figure 2: Gap left by manufacturing grid rounding (arrow)

P-Cell Replacement

The process of replacing the old devices with the new P-Cells was also fairly straightforward. The initial effort was done manually by opening each streamed-in transistor cell (the “\$\$” cells), removing its existing contents, and placing an appropriately-sized new P-Cell inside. This simplistic approach caused one extra layer of hierarchy that was later removed by flattening, leaving the new P-Cell in place^[1].

Verifying the New Design

As with any new version of a design, there remains the task of verifying that it meets specification. The existence of the schematic meant that both LVS and simulations could be run and the results compared to the previous verification runs of the 65nm design.

After achieving clean LVS and DRC runs with IC Validator and parasitic extraction with StarRC™, the simulations showed that simplistic scaling of all lengths and widths had a drawback—scaling the widths of all the transistors increased the power consumption without significantly affecting performance. In other words, the design was operating correctly but burning way too much power. It was clear that further optimizations in this area would be required.

Careful consideration of the simulation results and the knowledge of the circuit design highlighted a subset of transistors that actually did not require scaling of their widths. This subset was subject to a “reverse” scaling, reducing the widths to 75% of their original scaled values. While this effort was completed in short order and allowed the new schematic to be resimulated, it left disconnects in the layout as the devices pulled away from their interconnections. This handful of issues was resolved manually in approximately three days using LVS runs as a guide.

The sum of this final optimization effort, conducted over approximately one week, reduced the power consumption of the block by 18% and delivered the block well within the time constraint.

Summary

Developing a reverse process migration methodology proved to have significant positive results. Our team was able to complete this project in less than three months with a minimum amount of manual intervention. Armed with the methodology we developed and the scripts we created, we can easily perform this task again in much less time.

The use of scripting and other automation aids within Custom Designer contributed significantly to the productivity of our designers. Features like the flexible “Find and Replace” allowed us to easily manage attribute and name changes when necessary. Custom Designer’s “Edit In-Place” feature gave us new insight into the effects of scaling on the design. This feature allowed us to move up and down the hierarchy and clearly see the results of these operations.

We met our customer’s needs on time, and they were able to continue with their concurrent digital design due to early access to the LEF-accurate pin locations for the block. They also realized power savings due to the extra optimization efforts, leading to a more competitive final chip.

^[1] Eventually, this process was automated and a GUI crafted with Custom Designer’s OPAL subsystem was added to make it easier to use.