



Sucking up the time

Conventional flows perform a single pass of a physical design, generating a floorplan and corresponding physical implementation. Changes to the design – including late engineering change orders (ECOs) – take place all the way up to tapeout, so it is often necessary to respin large portions of the physical implementation. This takes weeks and it is not uncommon to respin the entire design, including creating a new floorplan.

With this flow, physical design engineers don't engage in the process until the RTL is 90% complete; if they engage earlier, anything they do will almost certainly end up being scrapped. Once they engage, it typically takes six weeks to generate the initial floorplan and corresponding physical implementation.

By the time the RTL is 95% complete, much of the original physical implementation will have been modified and upgraded; it typically takes another four weeks for the floorplan to be frozen. Even after the RTL is complete, it usually takes at least nine weeks before the final physi-

Taking a different approach to layout and verification allows the process to be automated. By **Yutki Rao**.

cal implementation is ready for tapeout.

An alternative approach is to take the design RTL and perform the key tasks in parallel with the RTL coding. This enables quick feedback to the RTL designer.

Dividing the physical implementation tasks automatically across multiple processors means a complete RTL to GDSII pass

can be performed in two days, irrespective of design size. Following the final pass, automated sign off verification prior to tapeout can be performed in two hours.

This approach means physical implementation can now start earlier and the faster turnaround for a full physical implementation provides predictability and early feedback on achievable performance.

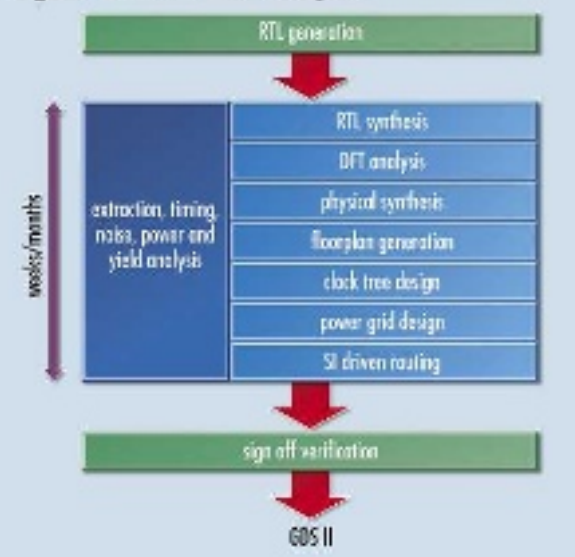
This approach also means that only one or two engineers are needed for the physical implementation of each design, reducing overall design costs by up to 50%, as well as giving better performance in terms of timing, area, power consumption and signal integrity. However, to work effectively, this approach needs a unified database with closely coupled layout, placement and analysis tools.

Illustration: Don Seed





Figure 1: The conventional design flow



When it comes to partitioning, the automated design flow decides which portions of the logical hierarchy are assigned to corresponding portions of the physical hierarchy. Whilst partitioning, this flow minimises block related pin counts, simplifies timing budgeting and prevents any top level closure issues, all without altering the original design hierarchy.

This is important for functional verification – in some cases, the contents of a logical block containing glue logic (or similar cell) may be distributed throughout the physical hierarchy.

Soft macro shaping is automatic, with the macros shaped and placed simultaneously using sophisticated congestion and timing aware algorithms. A similar quality of results (QoR) is achieved when compared to flat processing of the chip, but with a run time of hours as opposed to days or weeks. Similarly, hard macro placement is congestion and timing driven.

The algorithms, which must be integrated with the block shaper algorithms, minimise the perimeter of standard cell regions and shape them to avoid congestion problems, while all of the logic cones are analysed and the connectivity extracted to determine memory block interrelationships. This data is then used to achieve optimal placement of these blocks.

An important aspect of the overall process is the idea of relative placement

constraints, where blocks can be positioned relative to each other. In the very simple example shown in figure 2, block B2 is defined as being relative to the upper left corner of the chip. The left hand edge of B4 is defined as being relative to the right hand edge of block B3 and so on.

Unlike flows that use fixed X/Y coordinates, this flow provides incremental change capabilities, because relative placement constraints extracted from one iteration are used in the next. This means subsequent iterations build on earlier ones.

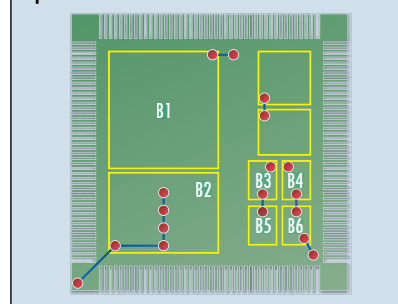
Another key feature of the automated approach is power network generation. The IR drop across the chip is analysed to provide non uniform power grids so areas with a higher IR drop will be provided with increased power grid densities. This is optimised on a layer by layer basis. The power mesh is updated as more detailed information becomes available on the power consumed by the clock tree and signal densities following routing.

This is all implemented within the power grid at the full chip level and then pushed down into the physical blocks, so when the blocks are implemented, there is already a good understanding of the power distribution.

Design freezes

Often, the clock tree has to wait until the rest of the design is frozen, causing the whole design timescale to slip, but all aspects of clock tree generation can be automated by using a prototype inside each block. These prototypes are eventually discarded, but the data is used to create the clock tree automatically in the final implementation.

Figure 2: Using relative placement constraints




In a conventional flow, each instantiation is treated as a unique entity and implemented individually. Whilst this can give slightly better area utilisation, it means that any changes to the parent have to be implemented in each child instantiation. An automatic approach ensures that repeated blocks have identical shapes, power meshes, signal and clock pin assignments. This allows any downstream ECOs to be propagated quickly.

Parallelised and distributed concurrent processing technology makes all of this possible. Synthesis and implementation are partitioned automatically and intelligently distributed across multiple processors and the results from the various partitions are integrated automatically at the end of the process.

When a design featuring two large blocks and two small blocks is implemented in a three processor environment, the two small blocks would run in series on one of the processors, which would run in parallel with the other two processors. These processors each would be assigned one of the larger blocks.

Although the user can provide input (such as specifying minimum/maximum partition sizes), all they need to specify is the number of machines to be used. The larger the design, the more computing resources can be applied.

It's important to note that QoR is maintained, irrespective of the number of processors used for the physical implementation. In one 8m gate test chip, implementation on one processor resulted in the design running at 523MHz. The same implementation distributed over three processors resulted in the design running at 524MHz.

A fully automated approach should not mean throwing away the existing design flow. Instead, it can be used immediately following RTL creation, with the integrated synthesis engine making decisions that ensure repeatability and predictability. 

Author profile:

Yutki Rao is product manager for Magma Design Automation's design implementation business unit.