TSUNAMI: An Integrated Timing-Driven Place And Route Research Platform

Christophe Alexandre, Hugo Clément, Jean-Paul Chaput, Marek Sroka, Christian Masson, Remy Escassut

TSUNAMI: An Integrated Timing-Driven Place And Route Research Platform

Christophe Alexandre[1], Hugo Clément[1], Jean-Paul Chaput[1], Marek Sroka[1],
Christian Masson[1,2], Rémy Escassut[3]

[1]University Paris VI, LIP6/ASIM laboratory, [2]Bull SA, [3]Silvaco

Abstract

In this paper, we present an experimental integrated platform for the research, development and evaluation of new VLSI back-end algorithms and design flows. Interconnect scaling to nanometer processes presents many difficult challenges to CAD flows. Academic research on back-end mostly focuses on specific algorithmic issues separately. However one key issue to address also is the cooperation of multiple algorithmic tools. TSUNAMI, our platform, is based on an integrated C++ database around which all tools consistently interact and collaborate. Above this platform a fixed die standard cell timing-driven placement and global routing flow has been developed.

1. Introduction

The advent of nanometer silicon technologies has introduced new challenges in physical design CAD, introducing an almost intractable interdependence between the tasks of synthesis, placement, global and detailed routing, timing optimization and noise avoidance. Therefore new design flows are being worked out with major objectives : avoid iterations between levels of design, enable early assessment of chip area and performance, reduce design uncertainty on the feasibility of later design steps and provide scalable tools against complexity increase.

The key issue is to let multiple algorithmic tools cooperate through an integrated database providing a unified view of the ongoing state of the design, in order to concurrently refine all interacting design facets. This issue has been addressed by proprietary solutions into CAD industry. Recently the OPEN-ACCESS initiative [3] has proposed an open-source standard with the intent to improve interoperability of CAD tools.

The TSUNAMI project is one of the first academic attempts to develop a back-end platform where all algorithmic engines operate on an integrated C++ database (HURRICANE) around which they consistently interact and collaborate. This ongoing project currently addresses the timing-driven placement and global routing of fixed-die standard cell blocks.

In the following sections, we will briefly present the HURRICANE database, the TSUNAMI flow currently implemented and the main features of its algorithmic engines.

2. HURRICANE: the C++ database

HURRICANE is a lightweight C++ object oriented database and programming platform which provides a unified and consistent modeling of hierarchical VLSI layouts through all the design steps from logic description down to detailed layout. It also consistently manages parasitic data (RC trees) and the timing graph. For that purpose :

- It provides a powerful API for fast access and incremental update which fully relieves the application programmer from memory management issues.
- It allows the seamless forward or backward transformation of net-list into a global routing or a detailed layout (or a mix of those states), ensuring built-in connectivity invariance.
- It represents a hierarchical layout as a ”folded” memory data model (as usual), but provides a ”virtually unfolded” view to the tools tracing, annotating or displaying its content. For that purpose it manages the concept of ”occurrences” which virtually refer items anywhere within the ”unfolded” design hierarchy.
- User defined properties and relations can be attached to any database object but also to occurrences (without the need to ”unfold” or ”flatten” the design hierarchy). This provides elegant ways to design algorithms for visiting, extracting and annotating hierarchical designs.
- It provides a rich (extensible) set of powerful query objects (”collections”) for visiting database items or occurrence items.
- It embeds high performance 2D region query facilities, a high speed graphical display engine and a graphical ”data structure inspector”, significantly simplifying the development and debugging of layout algorithms, editors and user interfaces.

HURRICANE was developed by BULL S.A. in close co-

Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'05)
1530-1591/05 $ 20.00 IEEE

operation with UPMC/LIP6 and later with the support of SILVACO. It has been focused on the fast development of integrated RTL to silicon flows, full-custom layout generators and technology migration tools for highly hierarchical layouts (it has been used for the migration of a 40 M transistors CPU IC from 120 nanometer 6 M layers to 90 nanometer 9 M layers CMOS process).

3. The TSUNAMI platform and flow

Above HURRICANE, the TSUNAMI platform provides general services: input/output LEF/DEF interfaces, cell library timing data inputs and utilitarians for building GUIs above the Hurricane display engine.
It also provides a interpretative PYTHON interface both as an extension language to HURRICANE API and as an encapsulation facility for the algorithmic engines in order to build and experiment different optimization flows and easily integrate new engines.
Within this environment, each algorithmic tool is an engine (a C++ object with its PYTHON wrapper) whose task is to analyze or process the current state of the design. Are currently implemented:

- A space manager, which plays a central communication role. It manages the recursive division of the design area into bins, the fences separating them and the pseudo-pins for nets crossing fences.
- A global placer, based on the **hmetis** multi-level net-list quadri-partitioner [1], which refines cell location into bins.
- A global router which refines or rebuilds the steiner-tree topology of nets. It can operate both within placement refinement steps and after placement finalization.
- A parasitics estimator which evaluates RC according to the level of precision of the routing and a delay evaluator which computes and stores Elmore delays.
- A static timing analyzer which, from interconnect delays and library cell delays, determines critical paths and valuates nets criticality to be fed back to placer and router.
- A detailed placer which finalizes and legalizes cell locations in each terminal bin.

And those under development:

- A gate sizing and buffer placement tool.
- A detailed router driven by the global router directives.

The standard cell place and route flow developed and under experimentation (figure 1) is a top-down progressive refinement process which proceeds by a succession of interleaved phases of quadri-partitioning, global routing and net-list timing optimizations:

- The entry point of a refinement loop is the geometric quadri-partitioning of all bins with more than 100 instances. Then each net-list of those bins are quadri-partitioned (but taking into account pseudo-pins and net criticalities, if already available from a previous iteration).
- Then the global router (re)builds or refines the steiner-trees of all nets whose cells have changed location. It has multiple algorithmic tactics tailored for different net configuration and timing criticalities, and tries to minimize both wire length and congestion on fences.
- Then the RC trees are (re)evaluated and a new static timing analysis is processed in order to compute updated critical paths, slacks and criticality value on each arc of the timing graph [2].This provides tighter directives to the next placement and global routing step.
- At this step, data is available to proceed (in the future) to gate sizing and buffer planning (virtual insertion in the timing graph, not in the net-list).

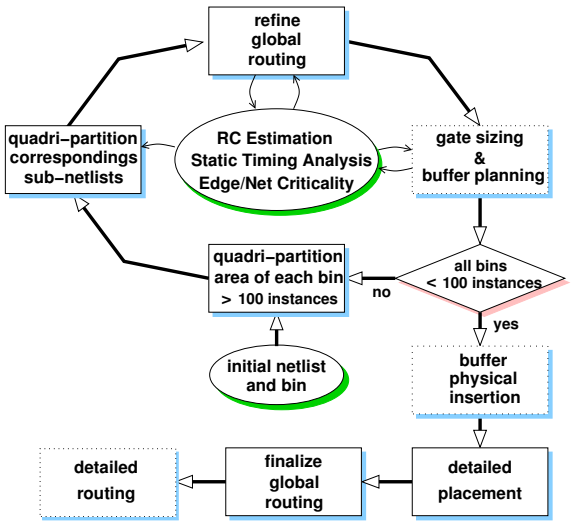

Figure 1. Overview of the place&route flow

At the end of the refinement loop (after buffers physical insertion) the simulated annealing detail placement of each bin is completed. Global routing is then refined, taking into account pin locations and obstructions. The resulting global routing directives and net criticalities will be fed to the detailed router under development.

References

[1] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th ACM/IEEE conference on Design automation*, pages 343–348. ACM Press, 1999.

[2] T. T. Kong. A novel net weighting algorithm for timing-driven placement. In *Proceedings of the 2002 IEEE/ACM international conference on Computer-aided design*, pages 172–176. ACM Press, 2002.

[3] D. Mallis and D. Cottrell. *OpenAccess: The Standard API for Rapid EDA Tool Integration*. Silicon Integration Initiative, Inc., 2003.