Experience of Using OpenROAD Flow Scripts on RISCV32i

Siddesh Patil, Student.
Electronics Engineering, VJTI

Abstract— In this paper, we present our experience of using OpenROAD Flow Scripts (ORFS) on RISCV32i. ORFS is a set of integrated scripts that allow for RTL-to GDSII flow using open-source tools. The OpenROAD Flow project aims for automated, no-human-in-the-loop digital circuit design with 24-hour turnaround time. We discuss the benefits and challenges of using ORFS and provide recommendations for future projects.

Keywords— OpenROAD Flow Scripts, RTL-to-GDSII flow, open-source tools, automated design, no-human-in-the-loop.

I. INTRODUCTION

OpenROAD Flow Scripts is a full RTL-to-GDS flow built entirely on open-source tools. The project aims for automated, no-human-in-the-loop digital circuit design with 24-hour turnaround time. The flow typically starts with Verilog design file and files that define the properties of the target fabrication process, which then generates intermediate files and output files such as placed and routed GDSII file. The OpenROAD flow script is highly configurable and can be tailored to specific design requirements.

II. DESIGN FLOW OVERVIEW

Our experience of using ORFS on RISCV32i has been positive overall. Learning about the ORFS tools like Floorplan, TritonCTS 2.0, RePlace, OpenSTA, TritonRoute for achieving best performance and runtime, gave a very brief overview of how the OpenROAD Flow Script works from RTL to GDSII.

III. EXPERIENCE OF USING ORFS

Clock tree synthesis (TritonCTS):

Clock Tree Synthesis makes sure that the clock signal is circulated evenly to all sequential elements in design and is optimized. ORFS uses its own CTS tool called TritonCTS, based on the GH Tree paradigm of Han et al, where a global tree is created followed by a local tree which is then implemented based on the power network hierarchy. TritonCTS uses a hierarchical approach that divides the design into multiple levels and applies CTS at each level. The OpenROAD flow script includes several CTS-related parameters that can be tuned to optimize the clock distribution network for a given design, like the clock period, target skew, maximum capacitance, buffer cell library(-clk_period<value>,-target_skew<value>,-max_cap<value>,-buffer_list <file>).

Placement (RePlAce,TritonMacroPlacer,OpenDP):

Minimisation of total wire length,area optimization,routable design are some goals of placement.We will optimize our placement, i.e., calculate the capacitance and wire length between inserted pins and cells to ensure that they are as close to one another as possible, and if not, add buffers or repeaters between them.Initially, we utilize the RePlace tool for Global placement, which seeks to produce a rough placement solution while preserving a global view of the entire netlist.The goal is to reduce the length of the connection wires.TritonMAcroPlacer will then be used to place macros.Positioning and spacing of macros and adding halos,partial and permanent blockage are points into consideration.The OpenDP tool will be used for detailed placement, which iteratively improves the legalized placement.The goal is to complete conventional cell placement while also adhering to design restrictions like timing and congestion.

Routing (FastRoute,Antenna Checker,TritonRoute):

Routing aims to reduce wirelength and meet timing budgets. Global routing using FastRoute partitions the region into tiles, optimizes tile-to-tile paths for all nets, and attempts to optimize an objective function. Detailed routing is done using TritonRoute, which assigns actual tracks and vias for each net. Antenna Checker checks for antenna violations and generates a report. The Maze Algorithm is used for routing, where a "wavefront" progressively labels adjacent grid cells until the target node is reached, and a shortest path is retracted from target to source node. Lee's algorithm guarantees the shortest path between two places, even with obstacles, but is memory- and time-intensive. Popular optimization techniques include coding schemes, search algorithms, and search spaces. We may use one or more in combination to improve the algorithm while working with RISC-V32i.

IV. CONCLUSIONS AND RECOMMENDATIONS

In conclusion, our experience of using OpenROAD Flow Scripts on RISCV32i has been largely positive. The use of open-source tools and automation greatly simplifies the design process and allows for rapid exploration of design alternatives. However, careful consideration of tool compatibility and versioning, as well as careful analysis of design results, is necessary to avoid unexpected issues.

REFERENCES

[1] “INVITED: Toward an Open-Source Digital Flow: First Learnings from the OpenROAD Project”, T. Ajayi, Vidya A. Chhabria, Mateus Fogaça, S. Hashemi, Abdelrahman Hosny, A. Kahng, Minsoo Kim, Jeongsup Lee, U. Mallappa, Marina Neseem, G. Pradipta, S. Reda, Mehdi Saligane, S. Sapatnekar, C. Sechen, M. Shalan, W. Swartz, Lutong Wang, Zhehong Wang, M. Woo, Bangqi Xu

[2] ”Physical Design of 32-bit RISC-V Processor” Chethana H K1, Manish S2, Harshitha C S3, Keerthana B N4, Dr. C M Patil5

[3]http://cc.ee.ntu.edu.tw/~ywchang/Courses/PD_Source/EDA_placement.pdf