A Template to be used by Students for Typesetting FYDP Synopsis

Submitted by:

2019-FYP-xx

Maja 20xx-EE-xxx

Saja 20xx-EE-xxx

Basantay 20xx-EE-xxx

Supervised by: Prof. Rab Nawaz

Department of Electrical Engineering

University of Engineering and Technology Lahore

Contents

Abstract

This project focuses on the placement and routing of the UETRV-Pcore, a RISC-V-based system-on-chip (SoC). RISC-V is an open-source instruction set architecture (ISA). The core implements a pipelined architecture with essential stages such as instruction fetch, decode, execution, memory access, and writeback. The flow of the UETRV-Pcore follows the complete ASIC flow, starting from Register Transfer Level (RTL) design to the generation of a GDSII layout file for fabrication. Cadence Genus is used for logic synthesis, converting the RTL code into a gate-level netlist optimized for timing, area, and power. Cadence Innovus is employed for physical design, including key steps such as floorplanning, standard cell placement, power planning, clock tree synthesis, and routing. We use a 45nm generic process design kit (PDK) for implementing the design. This project highlights the challenges of modern VLSI design, including timing closure, power management, and layout optimization. It provides hands-on experience with industry-standard tools, offering valuable insights into the complexities of digital design flows. The final deliverable of this project is a fully verified GDSII file, which will be fabricated to produce the UETRV-Pcore chip. This project not only contributes to open-source hardware development but also serves as a foundation for future research in processor design and SoC development.

Chapter 1

Introduction

The UETRV-Pcore project is a groundbreaking initiative for Pakistan's academic and industrial sectors. So far, no student team has completed the entire RTL-to-GDS flow using Cadence tools for a complete SoC, which are widely used in the chip design industry. Our project, led by the Department of Electrical Engineering at UET Lahore, aims to fill this gap by developing a complete RISC-V-based application-class System-on-Chip (SoC). This effort is a first of its kind, as we are the pioneers in creating a GDSII (Graphic Data System) layout, the final stage of chip design required for fabrication. Although Pakistan has a strong foundation in digital design and microarchitecture, the field of physical design and chip fabrication remains largely unexplored. By working on the UETRV-Pcore, we hope to set a new benchmark and encourage engineering programs across the country to focus on physical design. This project will also create opportunities for local chip design innovation and help improve Pakistan's standing in the global semiconductor industry. If we successfully fabricate the UETRV-Pcore, it will not only validate our team's expertise but also inspire students, researchers, and industries to explore chip design and fabrication leading to a major shift in the country's technological landscape. The UETRV-Pcore is a RISC-V-based application-class System-on-Chip (SoC). It integrates a 32-bit RISC-V ISA core, supporting RV32IMAZicsr instructions, which include base integer operations (I), multiplication and division (M), atomic operations (A), and control/status registers (Zicsr). The core implements three privilege levels—User (U), Supervisor (S), and Machine (M). With instruction and data caches and an SV32-based MMU (Memory Management Unit), UETRV-Pcore is capable of running Linux. The design features peripherals like UART, SPI, CLINT and PLIC that are connected via shared data buses. The design flow for UETRV-Pcore follows a structured physical design methodology using Cadence tools like Genus and Innovus. The first step in the physical design flow is synthesis, where the high-level Register Transfer Level (RTL) code is converted into a gate-level netlist. This is done using Cadence Genus, which performs technology mapping by replacing RTL constructs with logic gates from the 45nm standard cell library. Once synthesis is complete, the next step is floorplanning. In this stage, the major blocks of the design such as the core processor, caches,

MMU, and peripherals are placed logically within the chip area. It ensures that sufficient space is allocated for the components while minimizing wire lengths for efficient routing. In the placement stage, the standard cells from the synthesized netlist are placed onto the physical layout according to the floorplan. The tool ensures that the layout has no overlaps and creates space for clock distribution and routing tracks. This stage plays a critical role in improving the timing closure of the design. Clock Tree Synthesis (CTS) is one of the most critical stages of the physical design process. The objective of CTS is to create a balanced clock tree that distributes the clock signal uniformly across the chip with minimal clock skew and latency. Clock skew refers to the difference in the arrival time of the clock signal at various points in the design, which can cause timing violations if not controlled. Cadence Innovus uses buffer insertion and gating techniques to build an optimized clock tree. The tool ensures that the clock reaches all registers simultaneously, enabling synchronized operations across the pipeline stages. In the routing stage, the design tool connects the placed cells and peripherals through metal layers to form a complete circuit. During this process, the tool ensures that the design meets electrical rules, timing constraints, and power requirements. Design Rule Checking (DRC) and Layout vs. Schematic (LVS) verification are performed to confirm that the routed design matches the original schematic and complies with the manufacturing rules of the 45nm process.

Chapter 2

Problem Statement

Unmet need or problem, what is the unmet need or problem the FYDP is aiming to solve? How significant is the problem? Quantify as much as possible. In case of a research problem, show the significance of the unsolved problem. Who needs it? List the type of customers who will be interested in the solution of the problem. For each type of customer, indicate the potential market size. In case of a research problem, identify its scope. (1 page)

Chapter 3

Literature Review

Huang et al. (2024) present a novel RTL-to-GDS automation flow specifically designed for adiabatic quantum-flux-parametron (AQFP) superconducting circuits. The authors detail how their custom approach optimizes design tasks at each stage of the flow, including synthesis, placement, and routing, tailored to the unique electrical characteristics of AQFP technologies [2]. Although the focus is on a different circuit type, the core principles of customizing the design flow can be adapted to the UETRV-Pcore project. By analyzing the automation techniques described, the project can enhance its current flow using Cadence Genus and Innovus, potentially leading to better energy efficiency and performance in the RISC-V architecture. This paper serves as a critical reference for exploring custom automation in design flows, encouraging the adaptation of similar strategies for specific project needs. Acharya and Mehta (2022) conducted a performance analysis comparing the open-source tool Qflow with the commercial tool Cadence Encounter for the RTL to GDS-II flow of a Synchronous FIFO design. They highlighted the accessibility of Qflow for students and researchers, allowing them to engage in projects without the financial burden of expensive commercial tools. Conversely, Cadence Encounter is noted for its efficiency and accuracy, making it a preferred choice in industrial applications. The study revealed that the number of standard cells required when using Qflow was 1.5 times greater than that of Cadence Encounter, resulting in an area requirement over 2.6 times larger. These findings underscore the trade-offs involved in selecting design tools, especially concerning area, power, and operating frequency. This work serves as a critical reference for understanding the implications of tool selection in the RTL to GDS-II process, providing valuable insights that will inform the design decisions for the UET-RV Pcore.

Dwight Hill and Andrew B. Kahng explore the complex journey of chip implementation from RTL (Register Transfer Level) description to GDSII (Graphic Data System II) data, essential for the tape-out process in chip design. They argue that chip implementation encompasses several critical stages, notably logic synthesis, placement, and routing (SP&R), which have long been supported by advanced commercial tools. These

tools have evolved significantly over the years, enabling design teams to refine their approaches through a spiraling methodology that enhances timing estimation, device placement, and accuracy in parasitic extraction. The authors highlight that much of the RTL-to-GDSII work is rooted in industrial practice rather than academic research. This trend is attributed to the complexity of the design process, which requires competitive technology across various software platforms—capabilities that are often beyond the scope of typical graduate projects. They note that the chip design flow involves multiple representations and thus is often referred to as physical synthesis. The process necessitates various libraries, including timing libraries that describe cell delay properties and physical libraries that define the geometry of logic cells and I/O buffers. Additionally, the authors emphasize that the complexity of timing constraints plays a vital role in the design implementation process, reflecting the critical need for timing closure. They discuss the inadequacies of traditional static timing analysis, which is typically conducted at the RTL handoff and mask sign-off milestones. In modern design flows, embedded timing analysis has become integral, driving the need for accurate timing abstractions throughout the entire implementation process [1]. Hill and Kahng address the challenges posed by the non convergence of traditional flows, particularly as designs grow larger and more intricate. They identify various factors, such as crosstalk, substrate coupling, and thermal effects, that influence circuit timing and signal integrity. This complexity necessitates deeper integration between synthesis, analysis, and specification, which the authors argue is crucial for achieving predictable outcomes in chip design. The article presents three main categories of prediction methods aimed at improving design predictability. The first is statistical prediction, which, while quick, often lacks the required accuracy due to its reliance on average metrics. The second is constructive and iterative prediction, which involves real-time estimates based on previous design iterations. Lastly, the authors discuss the importance of enforced assumptions in design properties, which facilitate consistent outcomes throughout the design process [1].

Chapter 4

Project Overview and Objectives

The primary objective of this project is to complete the design and physical implementation of the UETRV-Pcore, a RISC-V-based System-on-Chip (SoC), using a structured ASIC flow. The specific objectives are:

- End-to-End ASIC Flow Implementation: To implement the UETRV-Pcore from Register Transfer Level (RTL) to a GDSII layout file, covering all key stages of physical design.
- Optimization for Timing, Area, and Power: To synthesize the RTL code into a gate-level netlist optimized for timing, area, and power using Cadence Genus.
- Physical Layout and Routing: To perform physical design steps including floorplanning, placement, clock tree synthesis, and routing using Cadence Innovus.
- Fabrication-Ready GDSII: To generate a verified GDSII layout file that meets design rule checks (DRC) and layout vs. schematic (LVS) checks, making the design ready for fabrication.
- Contribute to Open-Source Hardware: To contribute to the global RISC-V and open-source hardware community by creating a fabricatable, open-source processor design.

Chapter 5

Project Development Methodology/Architecture

The proposed solution for achieving the objectives is to implement the UETRV-Pcore by following a detailed ASIC design flow using industry-standard tools:

- RTL Design: A RISC-V 32-bit processor core implementing the RV32IMAZicsr instruction set architecture (ISA) will be designed. It will include a pipelined architecture with essential stages like instruction fetch, decode, execution, memory access, and writeback.
- Synthesis: The RTL design will be synthesized using Cadence Genus, which will optimize the design for area, timing, and power using the 45nm standard cell library.
- Floorplanning and Placement: The major blocks, such as the processor core, caches, MMU, and peripherals (UART, SPI, CLINT, and PLIC), will be floorplanned and placed logically within the chip area. The placement of standard cells will be optimized to minimize wire lengths and improve timing.
- Clock Tree Synthesis and Routing: A balanced clock tree will be created using Cadence Innovus to ensure uniform clock distribution with minimal skew and latency. The tool will then route the design to connect the cells and peripherals, ensuring it meets electrical, timing, and power requirements.
- Verification and Sign-Off: The routed design will undergo Design Rule Checking (DRC) and Layout vs. Schematic (LVS) verification to ensure compliance with manufacturing rules and match the original schematic.
- GDSII Generation: The final verified design will be exported as a GDSII layout file, ready for fabrication.

Chapter 6

Project Milestones and Deliverables

Clear milestones should be defined at the start of the project in the form of a Gantt chart. It is recommended to use excel or some equivalent software to make a Gantt chart. $(1-2$ pages)

Chapter 7

Block Diagram

Draw a block diagram of your project and explain it briefly. (1 page)

Chapter 8

Flow Chart

Include a flow chart of the project/sub-divisions/member wise activities. (1 page)

Chapter 9

Work Division

Clear work division among group members must be indicated. (1 page)

Chapter 10

Costing

Make a table of major required components with estimated prices. (1 page)

Appendix A

Introduction to Latex (Do not include this in the final version of your synopsis)

The material provided in this appendix is taken from
http://www.sunilpatel.co.uk/thesistemplate.php

A.1 Learning LaTeX

LaTeX is not a WYSIWYG (What You See is What You Get) program, unlike word processors such as Microsoft Word or Corel WordPerfect. Instead, a document written for LaTeX is actually a simple, plain text file that contains *no formatting*. You tell LaTeX how you want the formatting in the finished document by writing in simple commands amongst the text, for example, if I want to use *italic text for emphasis*, I write the '`\emph{}`' command and put the text I want in italics in between the curly braces. This means that LaTeX is a "mark-up" language, very much like HTML.

A.1.1 A (not so short) Introduction to LaTeX

If you are new to LaTeX, there is a very good eBook – freely available online as a PDF file – called, "The Not So Short Introduction to LaTeX". The book's title is typically shortened to just "lshort". You can download the latest version (as it is occasionally updated) from here:
http://www.ctan.org/tex-archive/info/lshort/english/lshort.pdf

It is also available in several other languages. Find yours from the list on this page:
http://www.ctan.org/tex-archive/info/lshort/

It is recommended to take a little time out to learn how to use LaTeX by creating several, small 'test' documents. Making the effort now means you're not stuck learning the system when what you *really* need to be doing is writing your thesis.

A.1.2 A Short Math Guide for LaTeX

If you are writing a technical or mathematical thesis, then you may want to read the document by the AMS (American Mathematical Society) called, "A Short Math Guide for LaTeX". It can be found online here:
http://www.ams.org/tex/amslatex.html
under the "Additional Documentation" section towards the bottom of the page.

A.1.3 Common LaTeX Math Symbols

There are a multitude of mathematical symbols available for LaTeX and it would take a great effort to learn the commands for them all. The most common ones you are likely to use are shown on this page:
http://www.sunilpatel.co.uk/latexsymbols.html

You can use this page as a reference or crib sheet, the symbols are rendered as large, high quality images so you can quickly find the LaTeX command for the symbol you need.

A.1.4 Figures

There will hopefully be many figures in your thesis (that should be placed in the 'Figures' folder). The way to insert figures into your thesis is to use a code template like this:

```
\begin{figure}[htbp]
  \centering
    \includegraphics[width = 1.5in]{./Figures/uet_logo.pdf}
    \rule{35em}{0.5pt}
  \caption{The UET Laore logo.}
  \label{fig:uet_logo}
\end{figure}
```

Also look in the source file. Putting this code into the source file produces the picture of the UET logo that you can see in the figure below.

FIGURE A.1: The UET Lahore logo.

Sometimes figures don't always appear where you write them in the source. The placement depends on how much space there is on the page for the figure. Sometimes there

is not enough room to fit a figure directly where it should go (in relation to the text) and so LaTeX puts it at the top of the next page. Positioning figures is the job of LaTeX and so you should only worry about making them look good!

Figures usually should have labels just in case you need to refer to them (such as in figure A.1). The '`\caption`' command contains two parts, the first part, inside the square brackets is the title that will appear in the 'List of Figures', and so should be short. The second part in the curly brackets should contain the longer and more descriptive caption text.

The '`\rule`' command is optional and simply puts an aesthetic horizontal line below the image. If you do this for one image, do it for all of them.

The LaTeX Thesis Template is able to use figures that are either in the PDF or JPEG file format. It is recommended that you read this short guide on how to get the best out of figures in LaTeX, available here:
http://www.sunilpatel.co.uk/texhelp5.html

Though it is geared more towards users of Mac and OS X systems, much of the advice applies to creating and using figures in general. It also explains why the PDF file format is preferred in figures over JPEG.

A.1.5 Typesetting mathematics

If your thesis is going to contain heavy mathematical content, be sure that LaTeX will make it look beautiful, even though it won't be able to solve the equations for you.

The "Not So Short Introduction to LaTeX" (available here) should tell you everything you need to know for most cases of typesetting mathematics. If you need more information, a much more thorough mathematical guide is available from the AMS called, "A Short Math Guide to LaTeX" and can be downloaded from:
ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf

There are many different LaTeX symbols to remember, luckily you can find the most common symbols here. You can use the web page as a quick reference or crib sheet and because the symbols are grouped and rendered as high quality images (each with a downloadable PDF), finding the symbol you need is quick and easy.

You can write an equation, which is automatically given an equation number by LaTeX like this:

```
\begin{equation}
E = mc^{2}
  \label{eqn:Einstein}
\end{equation}
```

This will produce Einstein's famous energy-matter equivalence equation:

$$E = mc^2 \tag{A.1}$$

All equations you write (which are not in the middle of paragraph text) are automatically given equation numbers by LaTeX. If you don't want a particular equation numbered, just put the command, '`\nonumber`' immediately after the equation.

A.2 Sectioning and Subsectioning

You should break your thesis up into nice, bite-sized sections and subsections. LaTeX automatically builds a table of Contents by looking at all the '`\chapter{}`', '`\section{}`' and '`\subsection{}`' commands you write in the source.

The table of Contents should only list the sections to three (3) levels. A '`\chapter{}`' is level one (1). A '`\section{}`' is level two (2) and so a '`\subsection{}`' is level three (3). In your thesis it is likely that you will even use a '`\subsubsection{}`', which is level four (4). Adding all these will create an unnecessarily cluttered table of Contents and so you should use the '`\subsubsection*{}`' command instead (note the asterisk). The asterisk (*) tells LaTeX to omit listing the subsubsection in the Contents, keeping it clean and tidy.

References

[1] D. Hill and A. B. Kahng. Guest Editors' Introduction: RTL to GDSII—From foilware to standard practice. *IEEE Design & Test of Computers*, 21(1):9–10, 2004.

[2] L. Huang et al. SuperFlow: A Fully-Customized RTL-to-GDS Design Automation Flow for Adiabatic Quantum-Flux-Parametron Superconducting Circuits. *arXiv preprint arXiv:2407.18209*, 2024.