POSSIBILITY OF DEMOCRATIZATION OF INTEGRATED CIRCUITS DESIGN FOR EDUCATIONAL AND SMALL PROJECTS

Mihailo Knezevic[1], Zeljko Jovanovic[1], Milentije Lukovic[1]
[1] *University of Kragujevac, Faculty of Technical Sciences, Čačak, Serbia*

Abstract

Since very early stages of software development, concept of open-source and free tools developed by community took important role in pushing software development technologies and techniques forward. Such idea just recently started its development in world of integrated circuits (IC) where proprietary and expensive software and tools still play huge part in industry, practically making it impossible for individuals and small companies to develop their ideas in world of ICs. With ever-rising need for Application Specific Integrated Circuits (ASIC) for budget constrained projects, need for open-source developing tools to support those processes has become very important, in fact, so important that some of the biggest technology companies took part in their development in hope of diversifying available tools for designing and producing ASICs.This paper aims to raise awareness on so far available tools (some of which have long ongoing history) with special focus on Tiny Tapeout, tool specially designed for students and beginners in ASIC digital systems design world to make it as easy as possible to tapeout their first design from scratch.

Keywords: VLSI, ASIC, IC, Tapeout, Open Source

INTRODUCTION

From the very beginning of the development of integrated circuit industry, there has been a need to expedite and facilitate their development and testing. For this purpose, various tools have been developed to speed up and simplify the process of developing complex integrated circuits [1]. These tools are typically categorized in a way that follows the different stages of integrated circuit development which are provided on figure 1 which presents all important stages of IC design. Depending on the manufacturing technology, the development phases can vary. It is also important to note that the development phases of systems differ between certain ASIC technologies and especially between ASIC and FPGA technologies [2]. Furthermore, it is essential to mention that FPGA system synthesis is mostly carried out using fully automated processes, while the design of ASIC systems still relies on careful engineering to achieve maximum performance [3].

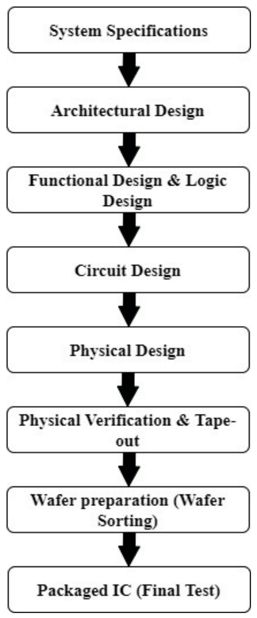

System Specifications
Architectural Design
Functional Design & Logic Design
Circuit Design
Physical Design
Physical Verification & Tape-out
Wafer preparation (Wafer Sorting)
Packaged IC (Final Test)

Fig. 1 *Integrated Circuit Design Flow*

As previously mentioned, with the increasing complexity of integrated circuits, the need for tools in various stages of development has grown. In this paper, we will focus on tools that cover Functional Design & Logic Design, Circuit Design and Physical Design stages of integrated circuit development since tasks related to those segments are taught by Digital Systems Design courses. In an industrial environment, these development stages are most commonly addressed using tools from companies like Synopsys and Cadence (Cadence Virtuoso and Synopsys Design Compiler among others), whose licenses prices can significantly impact the budgets of small and medium-sized (and sometimes even large) companies [4]. Additionally, the cost of licenses for these tools can make it impossible for startup companies or individuals to get into field of integrated circuits design. Inspired by the open-source ecosystem that has existed for decades and is developed, maintained and used by the community of software engineers, open-source hardware development tools have begun to emerge on much larger scale than ever. Ultimate goal of this paper is to present possibilities to teach engineering basics of chip design as early as possible and without usage of complex procedures and tools.

TOOLS FOR DESIGN

In this section, we will explore available free and open-source techniques and tools used for the previously mentioned design stages. Some of these tools are part of the widely accepted industrial toolset, while others, lesser-known tools, are currently emerging as alternatives to expensive but well-established tools.

FUNCTIONAL DESIGN & LOGIC DESIGN

In the early days of design, schematic representations were used to implement integrated circuits. However, as the number of transistors on a single chip grew in accordance with Moore's Law, the need for a more efficient and easier way to implement the desired behavior emerged. This led to the development of Hardware Description Languages (HDL): **VHDL** and, subsequently, the **Verilog** hardware description languages, which allow the description of desired behavior through code. **VHDL** is developed by The United States Department of Defence for military needs while **Verilog** is developed by Phil Moorby and Prabhu Goel and it served as a proprietary hardware modeling language owned by Gateway Design Automation Inc [5]. Even though both started as proprietary languages, right now both are free and available for use by all individual and legal entities. In Figure 2, we can see a simple circuit implemented with schematics while on Figures 3 and 4 same circuit is implemented using **Verilog** and **VHDL** respectively.

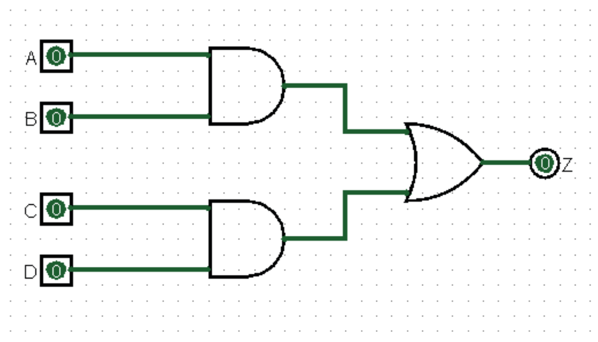
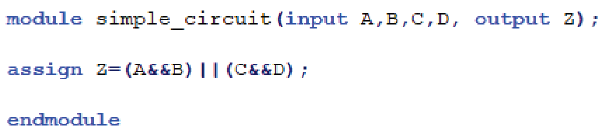
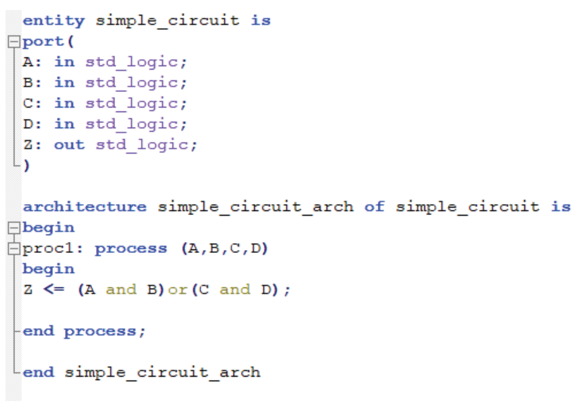


Fig. 2 *Simple Circuit*



```
module simple_circuit(input A,B,C,D, output Z);

assign Z=(A&&B)||(C&&D);

endmodule
```

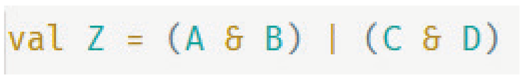
Fig. 3 *Simple Circuit in Verilog*



```
entity simple_circuit is
port(
A: in std_logic;
B: in std_logic;
C: in std_logic;
D: in std_logic;
Z: out std_logic;
)

architecture simple_circuit_arch of simple_circuit is
begin
proc1: process (A,B,C,D)
begin
Z <= (A and B)or(C and D);

end process;

end simple_circuit_arch
```

Fig. 4 *Simple Circuit in VHDL*

From the provided codes, we can conclude that the code written in Verilog is simpler, while VHDL is more structured.
A noticeable trend in hardware description languages is moving towards higher levels of abstraction, enabling more efficient descriptions of complex circuits with minimal code [6]. One such language is Chisel which is basically framework for Scala and as such has similar syntax. Althoough it is still in development, it is predicted that it will increase productivity of hardware engineers in future [7]. Representation of before mentioned circuit in Chisel can be seen on Figure 5.



```
val Z = (A & B) | (C & D)
```

Fig. 5 *Simple Circuit in Chisel*

Currently, there are few more hardware description languages which are in development and whose impact could be potentially evaluated in the upcoming years.

CIRCUIT DESIGN

In stage of circuit design, design implemented with logic gates or HDL gets translated into mesh of transistors. For the purpose of circuit design many free tools are available (KiCad as an example) and while those tools do have great capabilities to be used for design of circuits on printed circuit board (PCB) which are consisted of discrete components they lack capabilities to port that design to silicon. Since in this paper we are dealing with digital design, importance of circuit design. Although very important in the design of analog integrated circuits, this step is significantly simplified in digital techniques by introducing standardized cells that perform specific desired logic functions.

PHYSICAL DESIGN

The design of the actual circuit on silicon is the next step in approaching the physical level design of the chip. There still isn't a complete replacement for the proprietary tools, especially when working on microwave circuitry. Since we are focusing on tools for low-budget and student projects, we can highlight the tool **Magic**, which is open-source and was created in the 1980s with the aim of allowing students to learn chip layout design. This tool can be used independently or as part of an automated process that translates a circuit described in a hardware description language into physical design [8]. Typical working space of **Magic** layout tool can be seen on Figure 6.
Physical design takes a lot of time and can be quite complicated and challenging and requires a lot of background knowledge of semiconductor fabrication process so this tool is not suitable for educational purposes on early courses of Digital Systems Design.

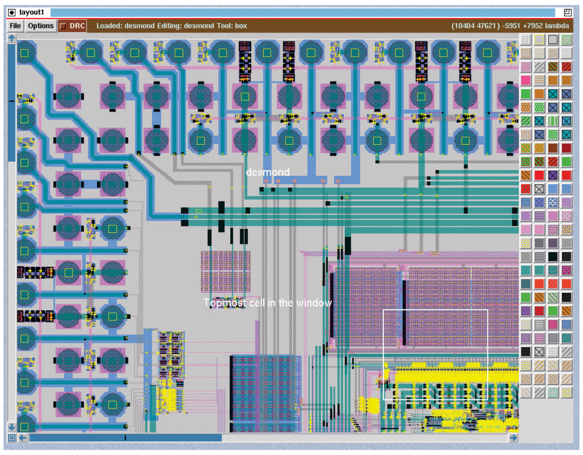

layout1
File
Options
DRC
Loaded: desmond Editing: desmond Tool: box
desmond
Topmost cell in the window

Fig. 6 *Magic VLSI Layout Tool Workspace*

TINY TAPEOUT

From previous sections it is evident that process of generating chip from idea is not straightforward, especially when it is taken into consideration how limited we are with specialized tools availability [9][10]. Luckily, Tiny Tapeout solves that problem by enabling students to implement their design in Wokwi graphical environment or HDL. For purposes of this paper, we will be dealing with Verilog HDL approach. Home page of Tiny Tapeout website where all necessary information could be found is provided on Figure 7.

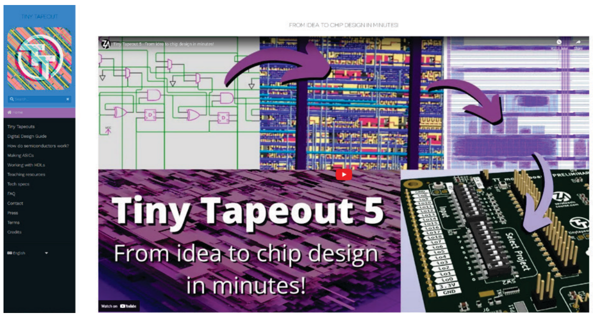
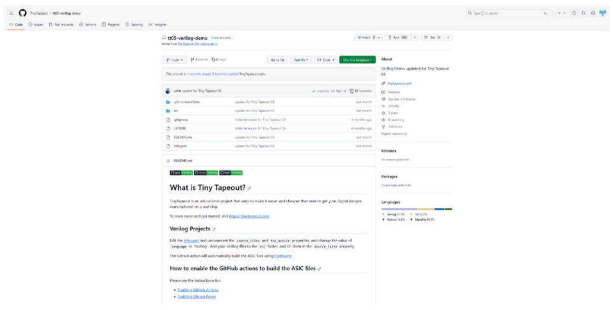

FROM IDEA TO CHIP DESIGN IN MINUTES
Tiny Tapeout 5
From idea to chip design in minutes!

Fig. *7 Tiny Tapeout Home Page*

Design is done in GitHub template (Figure 9) which can be found under “Working with HDLs” section.



tt05-verilog-demo Public template

This branch is 1 commit ahead, 6 commits behind TinyTapeout:main.

.github/workflows	update for Tiny Tapeout 05	last month
src	update for Tiny Tapeout 05	last month
.gitignore	Initial template for Tiny Tapeout 04	4 months ago
LICENSE	Initial template for Tiny Tapeout 04	4 months ago
README.md	update for Tiny Tapeout 05	last month
info.yaml	update for Tiny Tapeout 05	last month

About
Verilog Demo, updated for Tiny Tapeout 05
tinytapeout.com

README.md

What is Tiny Tapeout?

TinyTapeout is an educational project that aims to make it easier and cheaper than ever to get your digital designs manufactured on a real chip.

To learn more and get started, visit https://tinytapeout.com.

Verilog Projects

Edit the info.yaml and uncomment the `source_files` and `top_module` properties, and change the value of `language` to "Verilog". Add your Verilog files to the `src` folder, and list them in the `source_files` property.

The GitHub action will automatically build the ASIC files using OpenLane.

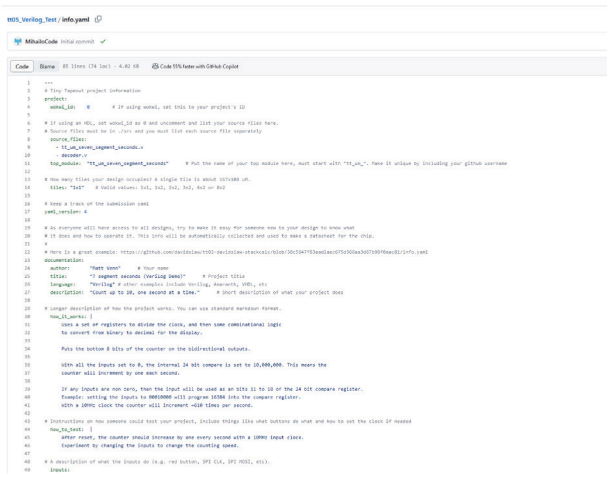
How to enable the GitHub actions to build the ASIC files

Please see the instructions for:

- Enabling GitHub Actions
- Enabling GitHub Pages

Fig. 7 *Template for HDL Design for Tiny Tapeout*

All required necessities to create ASIC project and to prepare it for tapeout is provided on Tiny Tapeout home page (Figure 7). All source files should be contained inside of *src* folder. One of the most important files in main directory is *info.yaml* which is used for configuring design. Snippet of this file is shown on Figure 8.



tt05_Verilog_Test / info.yaml

MihailoCode Initial commit

85 lines (74 loc) · 4.02 KB

```yaml
---
# Tiny Tapeout project information
project:
  wokwi_id:    0        # If using wokwi, set this to your project's ID

# If using an HDL, set wokwi_id as 0 and uncomment and list your source files here.
# Source files must be in ./src and you must list each source file separately
  source_files:
    - tt_um_seven_segment_seconds.v
    - decoder.v
  top_module:  "tt_um_seven_segment_seconds"      # Put the name of your top module here, must start with "tt_um_". Make it unique by including your github username

# How many tiles your design occupies? A single tile is about 167x108 uM.
  tiles: "1x1"    # Valid values: 1x1, 1x2, 2x2, 3x2, 4x2 or 8x2

# Keep a track of the submission yaml
yaml_version: 4

# As everyone will have access to all designs, try to make it easy for someone new to your design to know what
# it does and how to operate it. This info will be automatically collected and used to make a datasheet for the chip.
#
# Here is a great example: https://github.com/davidsiaw/tt02-davidsiaw-stackcalc/blob/38c5647f83aad2aec675d566aa3d67b98f0aac81/info.yaml
documentation:
  author:       "Matt Venn"      # Your name
  title:        "7 segment seconds (Verilog Demo)"      # Project title
  language:     "Verilog" # other examples include Verilog, Amaranth, VHDL, etc
  description:  "Count up to 10, one second at a time."      # Short description of what your project does

# Longer description of how the project works. You can use standard markdown format.
  how_it_works: |
      Uses a set of registers to divide the clock, and then some combinational logic
      to convert from binary to decimal for the display.

      Puts the bottom 8 bits of the counter on the bidirectional outputs.

      With all the inputs set to 0, the internal 24 bit compare is set to 10,000,000. This means the
      counter will increment by one each second.

      If any inputs are non zero, then the input will be used as an bits 11 to 18 of the 24 bit compare register.
      Example: setting the inputs to 00010000 will program 16384 into the compare register.
      With a 10MHz clock the counter will increment ~610 times per second.

# Instructions on how someone could test your project, include things like what buttons do what and how to set the clock if needed
  how_to_test:  |
      After reset, the counter should increase by one every second with a 10MHz input clock.
      Experiment by changing the inputs to change the counting speed.

# A description of what the inputs do (e.g. red button, SPI CLK, SPI MOSI, etc).
  inputs:
```

Fig. 8 *Info.yaml file*

In *info.yaml* file most important parameters can be set: name of top level file, number of tiles used by design (in case of larger design, user can reserve more tiles, each tile provides area of approximately 160x100µm) and name of used HDL language. After writing HDL code and setting up *info.yaml* file we can generate GDS file (file used for chip fabrication) using GitHub actions. View of 3D model of simple circuit we discussed earlier is provided in Figure 9.

Since this specific design is very small and most of silicon area is empty, on Figure 10 is presented zoomed section of chip with removed fill to show cells in detail.

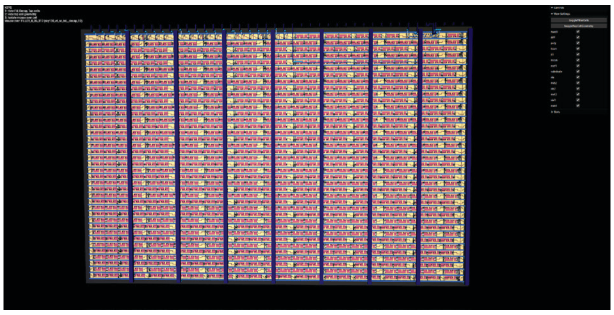


Fig. 9 *Whole chip render*

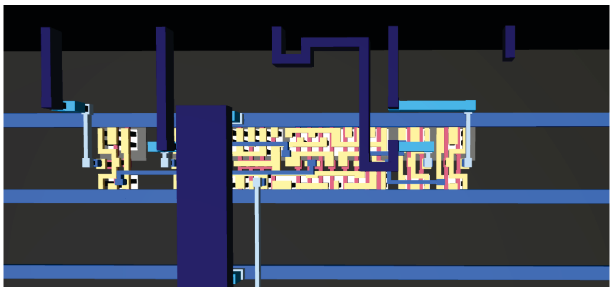
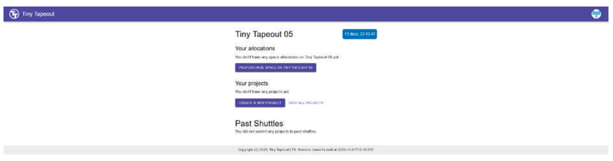


Fig. 10 *Zoomed in section of interest*

After design is finished, it can be submitted for tapeout using form on Figure 11.



Tiny Tapeout

Tiny Tapeout 05 — 13 days, 23:12:47

Your allocations

You don't have any space allocations on Tiny Tapeout 05 yet.

PREPURCHASE SPACE ON TINY TAPEOUT 05

Your projects

You don't have any projects yet.

CREATE A NEW PROJECT — VIEW ALL PROJECTS

Past Shuttles

You did not submit any projects to past shuttles.

Copyright (C) 2023, Tiny Tapeout LTD. Revision [illegible] built at 2023-10-21T12:16:33Z

Fig. 11 *Submission Form*

For 100$, design with one tile can be fabricated and delivered to designer in 6 to 9 months.

CONCLUSION

This paper gave an overview process of integrated circuit design with special mentioning of free tools and presentation of possibilities for simple approach on chip design specially designed for students. The primary advantage of this approach is its accessibility to students, as the entire infrastructure is free and hosted on GitHub. Additional benefits of the Tiny Tapeout tool include the ability to go through and learn the entire process of generating an integrated circuit in a simple and user-friendly manner, resulting in a physical chip that can be tested and incorporated into student and educational projects. On the other hand, commercially available tools offer more capabilities for advanced projects that may not have much application in educational contexts and as such do not justify high expenses of software licensing.

ACKNOWLEDGEMENT

The research in this paper was supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, as part of the Project grant no. 451-03-47/2023-01/200132 with University of Kragujevac - Faculty of Technical Sciences Čačak.

REFERENCES

[1] Y. Lee and A. Waterman, "Managing Chip Design Complexity in the Domain-Specific SoC Era," 2020 IEEE Symposium on VLSI Circuits, Honolulu, HI, USA, 2020, pp. 1-2, doi: 10.1109/VLSICircuits18222.2020.9162812.

[2] Qahtan, Mays & Salih, Muataz & Yousif, Omar & Mohammed, Nada. (2016). THE VANTAGE OF UTILIZING FPGA IN THE DESIGN OF AN EMBEDDED MULTIPROCESSOR. ARPN Journal of Engineering and Applied Sciences. 11. 13815-13827.

[3] M. -H. Ho et al., "Architecture and Design Flow for a Highly Efficient Structured ASIC," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 3, pp. 424-433, March 2013, doi: 10.1109/TVLSI.2012.2190478.

[4] A. H. Farrahi, D. J. Hathaway, M. Wang and M. Sarrafzadeh, "Quality of EDA CAD tools: definitions, metrics and directions," Proceedings IEEE 2000 First International Symposium on Quality Electronic Design (Cat. No. PR00525), San Jose, CA, USA, 2000, pp. 395-405, doi: 10.1109/ISQED.2000.838903.

[5] Y. Chu et al., "Three decades of HDLs. I. CDL through TI-HDL," in IEEE Design & Test of Computers, vol. 9, no. 2, pp. 69-81, June 1992, doi: 10.1109/54.143147.

[6] P. Bellows and B. Hutchings, "JHDL-an HDL for reconfigurable systems," Proceedings. IEEE Symposium on FPGAs for Custom Computing Machines (Cat. No.98TB100251), Napa Valley, CA, USA, 1998, pp. 175-184, doi: 10.1109/FPGA.1998.707895.

[7] Truong, L., & Hanrahan, P. (2019). A Golden Age of Hardware Description Languages: Applying Programming Language Techniques to Improve Design Productivity. In B. S. Lerner, R. Bodík, & S. Krishnamurthi (Eds.), 3rd Summit on Advances in Programming Languages (SNAPL 2019) (Vol. 136, p. 7:1-7:21). Schloss Dagstuhl – Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.SNAPL.2019.7

[8] Dally, W. J., & Chang, A. (2000). The Role of Custom Design in ASIC Chips. Proceedings of the 37th Annual Design Automation Conference, 643–647. https://doi.org/10.1145/337292.337604

[9] A. B. Kahng, "Open-Source EDA: If We Build It, Who Will Come?," 2020 IFIP/IEEE 28th International Conference on Very Large Scale Integration (VLSI-SOC), Salt Lake City, UT, USA, 2020, pp. 1-6, doi: 10.1109/VLSI-SOC46417.2020.9344073.

[10] Rovinski, A., Ajayi, T., Kim, M., Wang, G., & Saligane, M. (2020). Bridging Academic Open-Source EDA to Real-World Usability. Proceedings of the 39th International Conference on Computer-Aided Design. https://doi.org/10.1145/3400302.3415734