FOSSSC: A Free, Open-Source Software Stack Cluster for Digital System Design

Tassadaq Hussain
Electrical Engineering Department
Namal University Mianwali
tassadaq@ucerd.pk

Amna Haider
Pakistan Supercomputing Center
PakASIC and UCERD Islamabad
amna@ucerd.com

Syed Uzair Ghous Ali
Centre for AI and Big Data
Namal University Mianwali
uzair.ghous2021@namal.edu.pk

Dawood Mazhar
Centre for AI and Big Data
Namal University Mianwali
embedded.researcher@namal.edu.pk

Eduard Ayguade
Barcelona Supercomputing Center Spain
eduard.ayguade@bsc.es

2024 19th International Conference on Emerging Technologies (ICET)

***Abstract*—Chip design tools play an important role in the semiconductor industry, affecting production costs, accessibility, and innovation. Traditional commercial chip design tools have high licensing fees which cause innovation barriers and slow down the progress of smaller entities. However, in the last few years, advancements have been made in open-source chip design tools, supported by initiatives such as RISC-V and DARPA funding. These chip design tools target important issues like secure and specialized hardware design and encourage cooperation among worldwide innovators to provide alternatives to traditional expensive tools. However, a cloud-based cluster that supports end-to-end chip design using open-source hardware and software solutions is necessary to realize its full potential. In this work, we propose a Free Open-Source Software Stack-based Cluster (FOSSSC) to support open-hardware-based chip design. The proposed system provides end-to-end chip design and software development solutions and gives accessibility to open-source tools using cloud platforms and high-performance computing. The FOSSSC provides a globally accessible open-source digital system design software stack including design, verification, simulation, and programming.**

I. Introduction

The foundation of the entire semiconductor industry [1] is standing on the Open-source chip design tools. These tools have a huge impact on the cost of semiconductor manufacturing, its accessibility, and its innovation potential. This demands the need for improved user-friendly technologies that are more efficient, effective, and easily accessible for chip designers and software developers.

In the past few years, standard chip design tools [2] have faced issues such as high cost, making them inaccessible to digital designers and small business owners. Due to their non-open-source nature and limited accessibility, these tools hinder innovation in the semiconductor industry, stifling smaller players and slowing the pace of progress. These issues direct the chip design tool to shift towards the open-source. Open-source tools are now more competitive, particularly with the success of projects such as RISC-V and sponsored through DARPA funding [3]. Open-software solutions such as GCC compiler and Linux have democratized design resources and innovation. This approach has opened the door for open-hardware tools to open new avenues for more international cooperation in the semiconductor industry. Open-hardware tools like Chisel [4] and OpenRAM [5] take on different methodologies that open up innovative ways for chip designers. It addresses the challenging issues like memory design [6] that are also part of the chip design domain.

The open-source tools need a High-Performance Computing (HPC) cluster [7] that can do multiple tasks in parallel for multiple users and can utilize resources more efficiently. In this work, we have proposed an HPC cluster for an open-source digital system design software stack called Free Open Source Software Stack Cluster (FOSSSC). The FOSSSC provides a free and globally accessible system that is an an open-source digital system design software stack including design, verification, simulation, and programming. Through this, the semiconductor industry can be forwarded to a new innovative era.

II. Related Work

In the globally wide semiconductor industry, open-source tools have provided improvements and evolution by offering cheaper solutions for chip design development software. Several tools are now available, especially for system development, simulation and verification:

Icarus Verilog (iverilog) [8] is a free and open-source Verilog synthesis and simulation tool. For digital designers using open-source solutions, iverilog is a very important tool. Since for assembling and simulating Verilog HDL designs [9], this tool is widely used.

Verilator [10] is an effective open-source tool for simulating Verilog HDL. Verilator, in contrast to conventional simulators, translates Verilog code into C++ or SystemC [11], allowing for high-performance digital design simulation and verification.

979-8-3503-7958-7/24/$31.00 ©2024 IEEE

GTKWave, [12] provides simulation support with Icarus Verilog and Verilator, with feature waveform viewer for VCD (Value Change Dump) and LXT (LXT2/LX2) files [12] for debugging and visualizing the simulated results.

Cocotb [13] is a Python-based digital logic verification framework, a coroutine-based cosimulation [14] library for writing testbenches in Python. With Cocotb, users may create testbenches in Python, providing an efficient and adaptable method of verification.

QFLOW [15] is an open-source framework for digital synthesis and layout, offering a comprehensive RTL-to-GDSII flow for chip design. It simplifies the processes of synthesis, placement, routing, and optimization which makes the creation of custom integrated circuits easy.

OpenLane/OpenROAD [16] [17] are open-source projects providing complete chip design solutions and cover synthesis, placement, routing, and optimization. These projects use open-source tools and methodologies to facilitate the design and production of integrated circuits.

EDA Playground [18] is an online platform that allows users to simulate and test hardware description languages (HDLs) like Verilog [19] and VHDL [20] directly. Quite Universal Circuit Simulator [Qucs-S] [21] is an open-source Qucs, briefly for Quite Universal Circuit Simulator, is a circuit simulator with a graphical user interface (GUI). The software supports all kinds of circuit simulation types like DC, AC, S-parameter [22], Harmonic Balance analysis [23], noise analysis, etc.

Tiny Tapeout [24] makes manufacturing ASICs (chips) affordable and accessible to students and makers. They offer an open-source ASIC toolchain and made a community of open silicon researchers who collaborate and offer different digital design projects.

Additionally, SkyWater Technology Foundry [25] provides an open-source Process Design Kit (PDK) that provides resources including standard cell libraries, IO libraries [26], SRAMs [27], and other essential components for designing integrated circuits. This PDK resource enables chip designers to develop custom chips using open-source tools and methodologies. LibreSilicon [24] focuses on developing a complete open-source PDK [25] and toolchain for IC design and manufacturing. Their aim is to provide tools for chip design, simulation, and fabrication.

eFabless [28] is an open-source tool that provides resources to chip design flow with access to open-source PDKs and design tools. These resources enable to creation of custom chips and prototypes [29] using open-source technology and methodologies.

Google and SkyWater Technology Foundry [25] collaborate on OpenROAD, an open-source digital RTL-to-GDSII flow [30]. This project serves as a complete chip design flow, including synthesis, placement, routing, and optimization, utilizing open-source tools and methodologies.

Raptor [31] simplifies and shares control of FPGA (Field-Programmable Gate Arrays) design and implementation, giving a user-friendly environment for FPGA development. It supports multiple FPGA architectures [32], integrates with popular open-source tools, and provides a flexible design flow. Raptor aims to empower the users to efficiently develop and deploy FPGA-based solutions across various applications, emphasizing accessibility and collaboration, providing documentation, tutorials, and community support to facilitate learning and knowledge sharing within the FPGA community.

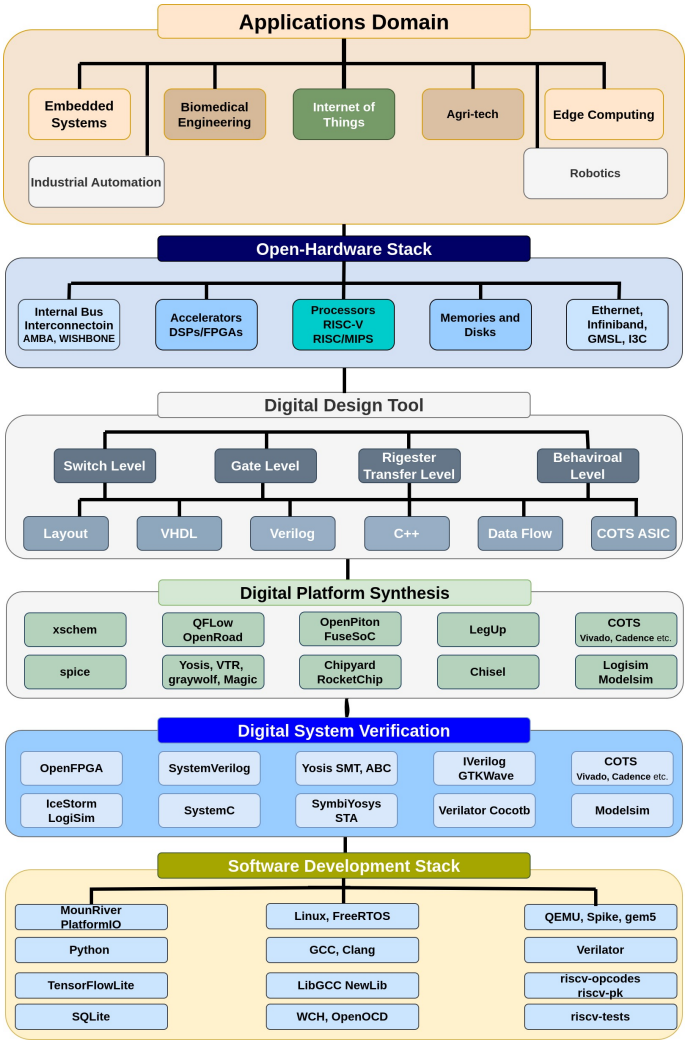

Fig. 1. Free Open-Source Software Stack Cluster (FOSSSC) for Chip Design: Visual Representation of Development Tools and Approaches for End-to-End Chip Design

III. FOSSSC: Free Open-Source Software-Stack Cluster for Digital System Design

The FOSSSC uses 10 Nodes CPU-GPU-based HPC cluster (shown in Figure 2) that handles the intensive computational simulation and compilation tasks. Each node has 128 gigabytes of RAM, 4070TI Nvidia TPUs, Intel 80 cores, and 2 terabytes of SSD storage. The nodes are networked with a robust 10 Gigabit Ethernet backbone, ensuring high-speed data transfer and low-latency communication across nodes. Operating on the Linux Ubuntu-Server Operating System (OS) the cluster provides performance and flexibility, with efficient user man-

Fig. 2. Photograph of High-Performance Computing Free Open Source Software Stack Cluster

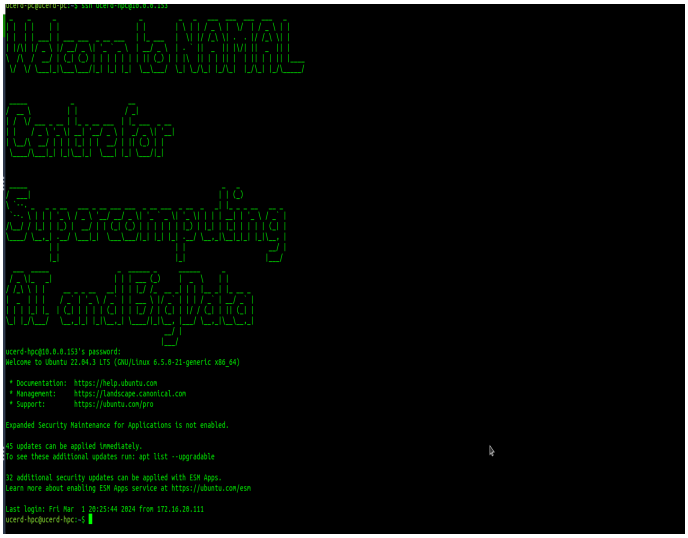

Fig. 3. Secure Shell Interface Connection for FOSSSC access

agement of diverse workloads and scientific computations. The section is further subdivided into the following subsections: a) Application Domain, b) Open-Hardware Stack, c) Digital Design Tools d) Digital Platform Synthesis e) System Verification, and f) Software Development Stack.

A. Application Domain

The application domain provides a web and secure shell interface to wide range of applications for digital system design shown in Figure 1. To manage application source code in FOSSSC, we used Git server systems. It prioritizes application development and management performance and accommodates large numbers of users and repositories efficiently the application development and management performance and accommodates large numbers of users and repositories. Git's decentralized architecture enables digital system projects with variable sizes to collaborate effectively on codebases of any scale. Its branching and merging capabilities facilitate parallel development workflows, allowing teams to work concurrently on different features or fixes without contention. The Application Version System integrates markdown and draw.io for text-based documentation, diagramming, and flowcharts respectively.

B. Open-Hardware Stack

The hardware stack facilitates digital system development, including bus systems, memory controllers [33], [34], memory manager [35]–[37], scheduler [38], [39], and heterogeneous processing cores [40]–[42]. Open hardware standards like RISC-V [43], Vector accelerator [44], [45] and AXI-bus system [46] are pivotal. RISC-V offers customizable processor designs, while AXI ensures efficient interconnects within System-on-Chip (SoC) architectures [47]. Memory accelerators and network components also improve system connectivity and performance. Resources for the development of applications are also provided by this stack.

C. Digital Design Tools

The FOSSSC provides multi-level digital design support to ensure that hardware designers have the required tools and support at each stage. The FOSSSC provides switch-level design, gate-level design, register-transfer level (RTL) design, and system-level design support.

1) Switch-Level Design: Switch-level design approach offers comprehensive digital circuit development while targeting on transistor-level modeling. FOSSSC offers switch-level design by utilizing the Verilog design approach and Layout Design Approach. It allows digital designers to model circuits at low levels for detailed electrical characteristics, behavior, and analysis of the transistor using SPICE (Simulation Program with Integrated Circuit Emphasis) tool. This approach utilizes tools like Verilog for HDL-based design and simulation, Magic for layout creation and editing, and KLayout for layout visualization and design rule checking. SPICE is used for circuit simulation. For memory components, OpenRAM automates the generation of SRAM layouts and integrates with other tools like Magic and SPICE for a smooth design process. Alliance CAD System provides a suite of tools, including schematic capture, layout, and verification, supporting the complete flow of transistor-level design. This open-source ecosystem enables detailed switch-level design, ensuring high reliability between the digital model and the physical circuit.

2) Gate-Level Design: FOSSSC provides a comprehensive framework for Gate-level design using Verilog as the main hardware description language (HDL), enabling designers to define and simulate digital circuits at the logic gate level. Additionally, FOSSSC supports VHDL, offering a strong alternative that is particularly valued for its rigor and detailed design syntax.

3) Register-Transfer Level (RTL) Design: FOSSSC supports RTL design by offering a versatile range of hardware description languages including Verilog, VHDL, and Sys-

temVerilog. It allows designers to describe the flow of data between registers and the logical operations that occur on that data, capturing the behavior of digital circuits in a way that closely resembles the final hardware's operation.

4) System-Level Design: In system-level design, FOSSSC targets support through SystemVerilog and high-level synthesis (HLS) tools. This enables designers to model and develop complex systems with higher abstraction, facilitating quicker design iterations and validations. FOSSSC offers a behavioral-level design approach using SystemC to model and simulate complex digital systems, where designers can define the behavior, data flow, and control logic of the entire system without needing low-level hardware specifics. Tools like TLM (Transaction-Level Modeling) within SystemC facilitate high-level simulation and verification, making it easier to explore different architectural configurations. Additionally, High-Level Synthesis, an open-source library (HLSLib), allows to convert C++ or SystemC descriptions into hardware descriptions in Verilog or VHDL [48], [49]. This method ensures that system-level behavior is captured and optimized early in the design process, making the development process more efficient and flexible.

D. Digital Platform Synthesis

This section below discusses software stacks that are used to develop digital system-based chip design.

Simulation: The VLSI Chip Design process begins with simulation tools helping in validation and testing the digital design's concept and functional specifications. The tools like Verilator, Icarus, and GTKWave are used to ensure that the code written in Hardware Description Language (HDL) is accurate and functional. These tools ensure that our designs work correctly before moving on to the next stages of chip development.

System Integration: These tools and frameworks provide efficient development, testing, and integration of complex semiconductor components. We integrate FuseSoC for IP core management and integration for complex FPGA/ASIC digital design. LiteX serves as a comprehensive SoC builder framework, capable of generating complete SoC designs tailored to diverse FPGA platforms. For research and innovation, the OpenPiton platform is integrated. It provides support for exploring and experimenting with many-core processor architectures, contributing to the advancement of SoC development methodologies.

OpenPDK: Open Process Design Kits (OpenPDKs) include standard cell libraries, technology files and design rules are essential for physical design and tape-out. OpenPDKs ensure compatibility with foundry processes and facilitate the generation of design files ready for tape-out.

E. System Verification

The System Verification section provides tools for functional and formal verifications.

1) Functional Verification: For the initial simulation phase, we employ tools like Cocotb and SVUnit for further verification of our chip designs. These tools help us set up test benches, which are essentially environments where we can test how our chip behaves under different conditions. Cocotb allows us to write test cases in Python and simulate how the chip responds, while SVUnit helps us verify specific functionalities of our design in SystemVerilog. In simpler terms, these tools help ensure that our chip performs as expected and meets all the requirements before it goes into production.

2) Formal Verification: The formal verification tools provide an extra layer of assurance that hardware design behaves correctly and consistently across various stages of the chip development process. This process helps validate the hardware design down to a low level of detail. We utilize tools such as SymbiYosys and Formality for this purpose. SymbiYosys is an open-source tool that helps in formal verification by analyzing the design for correctness specifications. The formality tool also provides equivalence checking and property verification. It ensures the functionality of the design by doing the pre-and post-synthesis.

Physical Design and Timing Analysis: After functional and formal verification, the next stage is timing analysis, Physical Design Place, and Routing. Timing analysis ensures the timing behavior of the chip to ensure that signals propagate correctly and meet the required timing constraints. OpenSTA and OpenTimer are used to identify and address timing constraints that specify the maximum delay allowed for signals to propagate through various paths in the design.
Physical Design, often referred to as PnR, in which the logical representation of the chip is mapped into the physical layout of the IC. This includes processes like placing the various components of the design on the chip's surface and routing connections between them. Basically, the goal is to optimize factors such as area, power consumption, and signal integrity. In our work, the OpenRoad, QFlow, Yosys, and Magic are integrated.

Tapeout: Once the physical design is complete and timing constraints are met, the design is ready for the tape-out. When the design files like layout data and timing information are finalized, then its ready for submission to the foundry. If any errors or deficiencies occur during the earlier stages then it must be addressed before tape-out to minimize the risk of costly manufacturing process.

F. Software Development Stack

The Software Development Stack provides application development support for the digital system. The section further discussed the software stack that is used to develop applications for open-hardware.

1) IDEs (Integrated Development Environments): The FOSSSC IDE provides a software suite which is designed to streamline and optimize the software development process. The cluster uses three IDEs: a) Eclipse IDE with RISC-V plugin, b) Platform-IO IDE, and c) Visual Studio Code

with RISC-V extensions These IDEs encompass a range of tools and features such as source code editing, compiling, debugging, testing, file linking, and project management, all within a user-friendly Graphic User Interface (GUI). This integrated approach aims to enhance efficiency and productivity in embedded application development.

2) Compilers: It plays a crucial role in software development by translating programs written in a source language into equivalent programs in a target language, such as assembly language or absolute machine code. The translation process involves several phases including lexical analysis, syntax analysis, semantic analysis, intermediate code generation, code optimization, and code generation. Specific optimizations for embedded systems are also incorporated to generate executable files tailored for specific platforms. The HPC cluster provides a wide range of open-source compilers for RISCV-V processor architecture such as GCC (GNU Compiler Collection) for RISC-V: Supports C, C++, and other languages. It also provides support for LLVM-based RISC-V compilers: Clang, and LLVM-RISC-V.

3) System Libraries: The cluster includes pre-written code libraries (such as Newlib, Libgcc etc) that provide specific functionalities, enhancing efficiency, modularity, coherence, and re-usability of functions. In the realm of embedded systems, libraries are tailored to meet the requirements of microcontrollers and IDEs. These libraries provide functions for hardware peripheral interfacing, communication protocols, sensor interfacing, mathematical calculations, signal processing, and power management.

4) Debuggers: Debugging tools such as GDB (GNU Debugger) with RISC-V support and OpenOCD (Open On-Chip Debugger) are used for the debugging of RISC-V programs and to identify and rectify errors in software code during the development process

5) Real Time Operating System Support: Different Real-time Operating Systems (RTOS) such as Zephr, and FreeRTOS support is provided. These RTOS are designed specifically for applications with real-time operational requirements. They ensure correctness in logical computations within defined time constraints, with determinism and response time predictability being key characteristics. RTOS features include task scheduling, resource management, inter-task communication, interrupt handling, and fault tolerance.

6) Simulations: The FOSSSC deploys different simulators for systems-level applications testing which provide functional, instruction and cycle-level accuracy. Spike, QEMU, and GEM5 simulators are used for functional verification at the event, cycle-level and provide a modular framework that supports the RISC-V architecture. Spike simulator is used as it provides robust RISC-V ISA Simulation to emulate a RISC-V system standalone or running RTOS, offering developers a comprehensive environment for testing and validation. It offers flexibility and extensibility for simulating intricate computer systems. With these simulators, the software stack provides a rich toolkit to work, explore, experiment, and innovate in the realm of embedded systems development.

IV. Results and Discussion

In this section, we discuss our results while using FOSSSC for various processor-based digital system design labs and projects. The cluster is accessible to 100 users via Secure Shell shown in Figure 3.
$StaticIPAddress$: 10.0.0.153
$PublicIPAddress$: 119.156.30.83

We have used FOSSSC for different labs, hands-on workshops, and have given access to chip developers to design and develop digital systems. We have conducted the following labs:

1) **VLSI Design Lab**:
   - Utilized OpenLane and QFlow GUI interface for GDS-II development, targeting OpenPDK libraries.
   - OpenPDK process design kits for chip fabrication processes, aiding in VLSI design.
2) **FPGA-based System Lab**:
   - Employed OpenFPGA boards for FPGA development and testing.
3) **Digital System Design Lab**:
   - Used tools like Icarus Verilog (iverilog), Verilator, GTKWave, and Cocotb for digital system designs.
   - These tools facilitate simulation, verification, and debugging of digital circuits and systems.
4) **Embedded System Design Lab**:
   - The lab focuses on industrial-grade 32-bit general-purpose RISC-V MCU-CH32V003 embedded systems for real-time and IoT applications.
   - It involves the implementation of the core functionalities of CH32V003 microcontroller like GPIO (General Purpose Input/Output) pins control, analog to digital conversion using 10-bit built-in ADC (Analog to Digital Converter), standard communication interfaces like USART (Universal Synchronous Asynchronous Receiver Transmitter), I2C (Inter-Integrated Circuit) communication and SPI (Serial Peripheral Interface) communication with sensors (e.g. MCP9808 high-accuracy I2C temperature sensor) and other devices.

V. Conclusion

In this work, we propose FOSSSC, a Free, Open-Source Software Stack-based HPC Cluster for Processor Digital System Design. The FOSSSC cluster provides chip design, simulation, verification, and embedded applications development support by giving open-source software access using a cloud cluster. The cluster is used for different projects such as VLSI design, Embedded applications, Computer Architecture, and digital system design.

VI. Acknowledgment

The research leading to these results has received findings from Unal Color of Education Research and Development (UCERD) Private Limited Islamabad and PakASIC. The authors would like to thank the Barcelona Supercomputing

Centre, Pakistan Supercomputing Centre and Centre for AI and BigData, Namal University Mianwali for the support.

REFERENCES

[1] John A Mathews. A silicon valley of the east: Creating taiwan's semiconductor industry. *California Management Review*, 39(4):26–54, 1997.

[2] David Atienza, Federico Angiolini, Srinivasan Murali, Antonio Pullini, Luca Benini, and Giovanni De Micheli. Network-on-chip design and synthesis outlook. *Integration*, 41(3):340–359, 2008.

[3] Tutu Ajayi1 Khalid Al-Hawaj Aporva, Amarnath1 Steve Dai2 Scott Davidson, Paul Gao4 Gai Liu2 Anuj Rao, Austin Rovinski1 Ningxiao Sun4 Christopher Torng, Luis Vega4 Bandhav Veluri4 Shaolin Xie, and Chun Zhao4 Ritchie Zhao. Experiences using the risc-v ecosystem to design an accelerator-centric soc in tsmc 16nm. In *1st Workshop on Computer Architecture Research with RISC-V (CARRV 2017)*, 2017.

[4] Amit Zoran, Roy Shilkrot, Pragun Goyal, Pattie Maes, and Joseph A Paradiso. The wise chisel: The rise of the smart handheld tool. *IEEE Pervasive Computing*, 13(3):48–57, 2014.

[5] Matthew R Guthaus, James E Stine, Samira Ataei, Brian Chen, Bin Wu, and Mehedi Sarwar. Openram: An open-source memory compiler. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–6. IEEE, 2016.

[6] Syed Anas Alam, Jakob Furbo Enevoldsen, Andreas Alkjaer Eriksen, Niels William Hartmann, Ulrik Helk, Jϕrgen Kragh Jakobsen, Christa Skytte Jensen, Nicolai Dyre Bülow Jespersen, Karl Herman Krause, Mads Rumle Nordstrϕm, et al. Open-source chip design in academic education. In *2022 IEEE Nordic Circuits and Systems Conference (NorCAS)*, pages 1–6. IEEE, 2022.

[7] Marco AS Netto, Rodrigo N Calheiros, Eduardo R Rodrigues, Renato LF Cunha, and Rajkumar Buyya. Hpc cloud for scientific and business applications: taxonomy, vision, and research challenges. *ACM Computing Surveys (CSUR)*, 51(1):1–29, 2018.

[8] Jun Wang and Carl Tropper. Nicarus: A distributed verilog compiler. In *Workshops on Mobile and Wireless Networking/High Performance Scientific, Engineering Computing/Network Design and Architecture/Optical Networks Control and Management/Ad Hoc and Sensor Networks/Compil*, pages 514–519. IEEE, 2004.

[9] Shinya Takamaeda-Yamazaki. Pyverilog: A python-based hardware design processing toolkit for verilog hdl. In *Applied Reconfigurable Computing: 11th International Symposium, ARC 2015, Bochum, Germany, April 13-17, 2015, Proceedings 11*, pages 451–460. Springer, 2015.

[10] Wilson Snyder. Verilator and systemperl. In *North American SystemC Users' Group, Design Automation Conference*, volume 79, pages 122–148, 2004.

[11] Preeti Ranjan Panda. Systemc: a modeling platform supporting multiple design abstractions. In *Proceedings of the 14th international symposium on Systems synthesis*, pages 75–80, 2001.

[12] I Petrescu, IB Pavaloiu, M Raducanu, and DA Mitrea. Distance and online learning for programmable electronic systems with fpga. In *INTED2022 Proceedings*, pages 9303–9310. IATED, 2022.

[13] Benjamin John Rosser. Cocotb: a python-based digital logic verification framework. In *Micro-electronics Section seminar. CERN, Geneva, Switzerland*, 2018.

[14] Alejandro Amutio Duarte. *Evaluation and application of new Python-based frameworks for the verification of digital integrated circuits*. PhD thesis, Universitat Politècnica de València, 2024.

[15] K Sripath Roy, K Abhiram, M Arun Sumanth, Jaishree Jaishankar, P Abhishek, B Nikhil Prabhat, and L Gnana Teja. Development of graphical user interface for open source vlsi digital synthesis tool qflow. *International Journal of Engineering and Technology*, 2018.

[16] Mohamed Shalan and Tim Edwards. Building openlane: a 130nm openroad-based tapeout-proven flow. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–6, 2020.

[17] Andrew B Kahng and Tom Spyrou. The openroad project: Unleashing hardware innovation. In *Proc. GOMAC*, 2021.

[18] Debajit Saha. Evaluation of open-source eda tool "eda playground". 2023.

[19] Donald Thomas and Philip Moorby. *The Verilog® hardware description language*. Springer Science & Business Media, 2008.

[20] Peter J Ashenden. *The designer's guide to VHDL*. Morgan kaufmann, 2010.

[21] Mike Brinson and Vadim Kuznetsov. Qucs-0.0. 19s: A new open-source circuit simulator and its application for hardware design. In *2016 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–5. IEEE, 2016.

[22] William R Eisenstadt and Yungseon Eo. S-parameter-based ic interconnect transmission line characterization. *IEEE transactions on components, hybrids, and manufacturing technology*, 15(4):483–490, 1992.

[23] Rowan J Gilmore and Michael B Steer. Nonlinear circuit analysis using the method of harmonic balance—a review of the art. part i. introductory concepts. *International Journal of Microwave and Millimeter-Wave Computer-Aided Engineering*, 1(1):22–37, 1991.

[24] Oliver Child. Beyond prototyping boards workshop position paper: How open silicon could transform the prototyping board landscape.

[25] R Timothy Edwards. Google/skywater and the promise of the open pdk. In *Workshop on Open-Source EDA Technology*, 2020.

[26] Ayse Bagbaba. A comparative study of mpi-io libraries for offloading of collective i/o tasks. In *2021 International Conference on Engineering and Emerging Technologies (ICEET)*, pages 1–6. IEEE, 2021.

[27] PE Dodd and FW Sexton. Critical charge concepts for cmos srams. *IEEE Transactions on Nuclear Science*, 42(6):1764–1771, 1995.

[28] Anand Rajgopalan. Placement & routing of digital core ic to pads using cloud based eda tool.

[29] Carlye A Lauff, Daria Kotys-Schwartz, and Mark E Rentschler. What is a prototype? what are the roles of prototypes in companies? *Journal of Mechanical Design*, 140(6):061102, 2018.

[30] Het Suthar, Shubham Tomar, and Rutu Parekh. Rtl to gdsii: Fully digital indirect time of flight soc. In *Sustainable Technology and Advanced Computing in Electrical Engineering: Proceedings of ICSTACE 2021*, pages 889–898. Springer, 2022.

[31] Rapid Silicon. The first and only commercial open-source fpga eda suite with rapid-gpt integration, March 2024.

[32] Umer Farooq, Zied Marrakchi, Habib Mehrez, Umer Farooq, Zied Marrakchi, and Habib Mehrez. Fpga architectures: An overview. *Tree-Based Heterogeneous FPGA Architectures: Application Specific Exploration and Optimization*, pages 7–48, 2012.

[33] Tassadaq Hussain, Miquel Pericas, and Eduard Ayguadé. Reconfigurable memory controller with programmable pattern support. In *5th HiPEAC Workshop on Reconfigurable Computing (WRC), Heraklion Crete 2011*, volume 67, page 95, 2011.

[34] Tassadaq Hussain, Oscar Palomar, Adrian Cristal, Osman Unsal, Eduard Ayguade, and Mateo Valero. Advanced pattern based memory controller for fpga based applications. In *International Conference on High Performance Computing & Simulation (Acceptance Rate 38.41%)*, page 8. ACM, IEEE, 2014.

[35] Tassadaq Hussain, Oscar Palomar, Osman S. Ünsal, Adrian Cristal, and Eduard Ayguade. Memory controller for vector processor. *Journal of Signal Processing Systems*, 2016.

[36] Tassadaq Hussain, Oscar Palomar, Adrian Cristal, Osman Unsal, Eduard Ayguade, and Mateo Valero. Ammc: Advanced multi-core memory controller. In *International Conference on Field-Programmable Technology FPT*. IEEE, 2014.

[37] Tassadaq Hussain, Amna Haider, Shakaib Gursal, A., and Eduard Ayguade. Amc: Advanced multi-accelerator controller. *Publisher : Journal of Parallel Computing*, 2014.

[38] Tassadaq Hussain, Amna Haider, and Eduard Ayguade. Pmss: A programmable memory system and scheduler for complex memory patterns. *Journal of Parallel and Distributed Computing*, 2014.

[39] Tassadaq Hussain. Memory resources aware run-time automated scheduling policy for multi-core systems. *Microprocessors and Microsystems*, 57:32–41, 2018.

[40] Tassadaq Hussain. A novel hardware support for heterogeneous multi-core memory system. *Journal of Parallel and Distributed Computing*, 106:31–49, 2017.

[41] Tassadaq Hussain, Amna Haider, and Abdelmalik Taleb-Ahmed. A heterogeneous multi-core based biomedical application processing system and programming toolkit. *Journal of Signal Processing Systems*, 91:963–978, 2019.

[42] Tassadaq Hussain. Hmmc: A memory controller for heterogeneous multi-core system. *Microprocessors and Microsystems*, 39(8):752–766, 2015.

[43] Samina Yasmin, Tassadaq Hussain, Muhammad Asim Naveed, Ayguade, and Eduard. Integration of bit manipulation extensions in the swerv eh1 core of a risc-v processor and its functional verification. In *21st*

International Bhurban Conference on Applied Sciences & Technology, Bhurban, 2024.

[44] Tassadaq Hussain, Oscar Palomar, Adrian Cristal, Eduard Ayguade, and Amna Haider. Mvpa: An fpga based multi vector processor architecture. In *The 13th International Bhurban Conference on Applied Sciences & Technology*. IEEE, 2016.

[45] Tassadaq Hussain, Amna Haider, Adrian Cristal, and Eduard Ayguadé. Emvs: Embedded multi vector-core system. *Journal of Systems Architecture*, 87:12–22, 2018.

[46] Tassadaq Hussain, Oscar Palomar, Adrian Cristal, Eduard Ayguade, and Amna Haider. Access pattern based multi-layer bus virtualization controller. In *The 13th International Bhurban Conference on Applied Sciences & Technology*, 2016.

[47] Tassadaq Hussain et al. A novel access pattern-based multi-core memory architecture. 2014.

[48] Saqib Amin, Usman Zabit, Olivier D Bernal, and Tassadaq Hussain. High resolution laser self-mixing displacement sensor under large variation in optical feedback and speckle. *IEEE sensors journal*, 20(16):9140–9147, 2020.

[49] Tassadaq Hussain, Miquel Pericas, Nacho Navarro, and Eduard Ayguadé. Implementation of a reverse time migration kernel using the hce high level synthesis tool. In *International Conference on Field-Programmable Technology (FPT) (28.7 % Acceptance Rate : Full Paper)*, pages 1–8. IEEE, 2011.