*SoC-Now: An Open-Source Web based RISC-V SoC Generator*

Shahzaib Kashif   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of software engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[shazaib@students.uit.edu](mailto:shazaib@students.uit.edu)

Syeda Fizza Jaffery   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of software engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[sjaffery@students.uit.edu](mailto:sjaffery@students.uit.edu)

Farhan Ahmed Karim   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of computer science*  
*UIT University (UITU)*Karachi, Pakistan  
[faahmed@uit.edu](mailto:faahmed@uit.edu)  
Talha Ahmed   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of software engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[tahmed@students.uit.edu](mailto:tahmed@students.uit.edu)

Muhammad Yasir Khan   
*dept. of computer science*  
*UIT University (UITU)*Karachi, Pakistan  
[mykhan@uit.edu](mailto:mykhan@uit.edu)   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*

Dr Ali Ahmed  
*Micro Electronics Research Lab (MERL)*   
*dept. of electrical engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[aahmed@uit.edu](mailto:aahmed@uit.edu) Muhammad Shahzaib  
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of software engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[mshahzaib@students.uit.edu](mailto:mshahzaib@students.uit.edu)

Hadir Khan   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of computer engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[mhkhan@uit.edu](http://mhkhan@uit.edu)

Dr Roomi Naqvi   
*Micro Electronics Research Lab (MERL)* Karachi, Pakistan  
[rnaqvi@uit.edu](mailto:rnaqvi@uit.edu)Almas Ibrahim   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of software engineering*  
*UIT University (UITU)*Karachi, Pakistan  
[aibrahim@students.uit.edu](mailto:aibrahim@students.uit.edu)

Usman Zain   
*dept. of software development*  
*Micro Electronics Research Lab (MERL)*   
*dept. of computer science*  
*UIT University (UITU)*Karachi, Pakistan  
[uzuabedin@uit.edu](http://uzuabedin@uit.edu)

*Abstract*— *In the rapidly changing technological world of today, smart phones and other gadgets have taken over our daily lives. Every aspect of life is reliant on technology, or "smart devices." And these devices are called “smart” because they contain a computing system inside called as processor or system on chip (SoC). Development of SoC is a quite long process. It can take months to build and develop a SoC from scratch. Following the design stage, testing, and verification of it adds another layer of stress. Designers like to reuse a verified component in their design to save time. However, this doesn't save time because without adequate documentation, which is lacking in many contributions to open-source projects, it can be extremely challenging to grasp the functionality and usability of a component. The open-source semiconductor industry is therefore in need of software that can quickly produce or design a SoC rather than taking the designer years to complete the build or design. Alternately, any designer can also generate a component with sufficient documentation and plug and play support. SoC-Now is an open-source web-based SoC design solution in terms of generator that enables anyone to create a SoC with their own customized specifications and then further process that SoC to the Field Programmable Gate Array (FPGA) Emulation. Additionally, it provides the capability to produce any standalone, verified, and reusable SoC component (Core, Device, Bus). A completely automated Verification system is also included.*

Keywords—RISC-V, System on Chip (SoC), CHISEL, Open-Source, SoC Generator.

# Introduction

As technology evolves, we witness new developments every day. Every day we wake up we see a new smart gadget, or a device introduced. Each of these “smart” devices has a silicon chip (Processor or SoC) attached as its central processing unit. As of today, millions of products like cars, washing machines, smartphones, and more rely on computer (silicon) chips. The manufacturing and usage of so many smart devices powered by silicon chips has resulted in a global silicon shortage [1].

Basically, the SoC development process takes a year or more to complete. Individual workloads are decreased when entire teams are assigned to this purposeful activity, but there are also potential risks, such as coordination challenges and communication breakdowns. With the aim to reduce this problem in the semiconductor industry. SoC-Now comes in place. SoC-Now is a SoC Generator that will generate customized/configurable SoCs rapidly. The generated SoC will be tested and verified based on industry standards. Additionally, SoC-Now will also provide the functionality of generating just only the Core standalone. Also, any Device or any other SoC Component can also be generated separately.

Fig. 1. Web App Home page of SoC-Now

Diagram

Description automatically generated

Graphical user interface, website

Description automatically generated

SoC-Now is a web-based RISC-V SoC generator as shown in Fig. 1 created by UIT University (UITU) undergraduate software engineering students with assistance from the Micro Electronics Research Lab (MERL). It targets the users who want to generate custom configurable SoC, any re-usable SoC component, and Standalone RISC-V based Core.

Fig. 2. System level Block Diagram of SoC-Now

# Overview of RISC-V

The open-source Instruction Set Architecture (ISA) known as RISC-V acronym of Reduced Instruction Set Architecture (RISC) and V represents its fifth version was created in 2010 at the University of California, Berkeley. It comprises of unprivileged [2] and privileged ISA [3]. It is academically usable, deployable without any royalty in any hardware and software design, and it doesn’t require any non-disclosure agreement. It has computing units that are 32, 64, and 128 bits. It has an RV32I base architecture with a variety of optional extensions, such as M for "Multiplication and Division" and K for "Cryptography” etc.

To deploy agile hardware methodology, CHISEL, an agile enabling language, will be required for RISCV based SoCs [4].

## CHISEL

CHISEL “Construction Hardware in SCALA Embedded Language” is an open-source language which is used for hardware construction (HCL). Its purpose is to construct the digital electronics design at the register transfer level (RTL) that helps advanced circuit synthesis for both ASIC and FPGA designs [5].

Fig. 3. Features of SoC-Now Web App

Graphical user interface, application, website, Teams

Description automatically generated

Since CHISEL is integrated into SCALA, it offers designers the most recent or high-level programming concepts, such as Object-Oriented Programming and Functional Programming, and others. It makes it possible for designers to use intricate, parameterized circuits that can be synthesized in Verilog. These ideas make it possible to advantage from reusability and agile development [4].

# SoC-Now

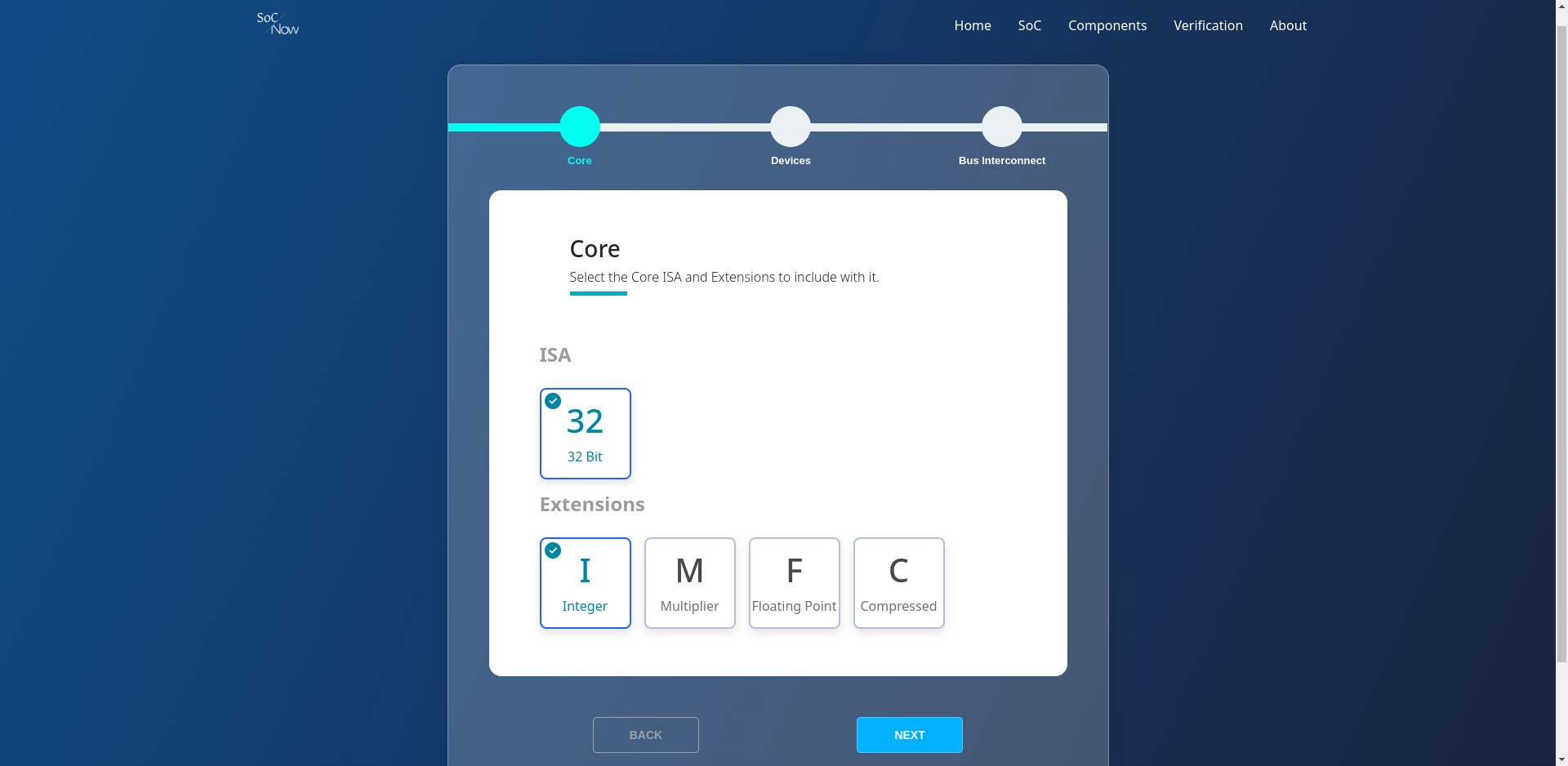
SoC-Now is a tool for quickly producing customized/configurable SoCs. To achieve the factor of reusability, it is built and developed on CHISEL HCL and Functional Programming paradigm [6].

It generates SoC in such a way that it contains a user selected parametrized Core that connects with the Bus Interconnect on its host end which acts as the communication medium all over the SoC. All the selected devices and memories connect with that bus on the device/slave end. As depicted in Fig. 2 .

It has three different generic frameworks for using plug-and-play-enabled Bus Interconnects, Devices and Memories, and Core components, respectively. These APIs were developed for users that want to generate and use only standalone, verified components. Additionally, these precise frameworks are combined into the SoC-Now Generator, where they are all integrated to provide a high-level top wrapper that makes it simple to construct a whole SoC. All the frameworks along with the Generator are designed using CHISEL and Functional Programming approach.

The entire CHISEL-based SoC Generator is connected to the web using the free and open-source Python programming language and the Django open-source web framework. The Web App gives users the option to construct a SoC by simply selecting the configuration from the Graphical User Interface (GUI). After the SoC has been generated, users are also given the option to perform all FPGA-related configurations via GUI to produce the configured SoC's Bitstream. Additionally, there is a function that allows you to create a custom SoC component by just choosing it from the list of components. Additionally, a full automated verification mechanism is available as an add-on.

Fig. 4. Tilelink (TL-UL) Adapter implementation in Caravan



SoC-Now contains multiple different components that lie in different subcategories as Bus, Devices, and Core. These subcategories are implemented in separate frameworks as Caravan, Jigsaw, and NucleusRV respectively. A list of all the components and a brief description of their implementation is displayed in Table 1.

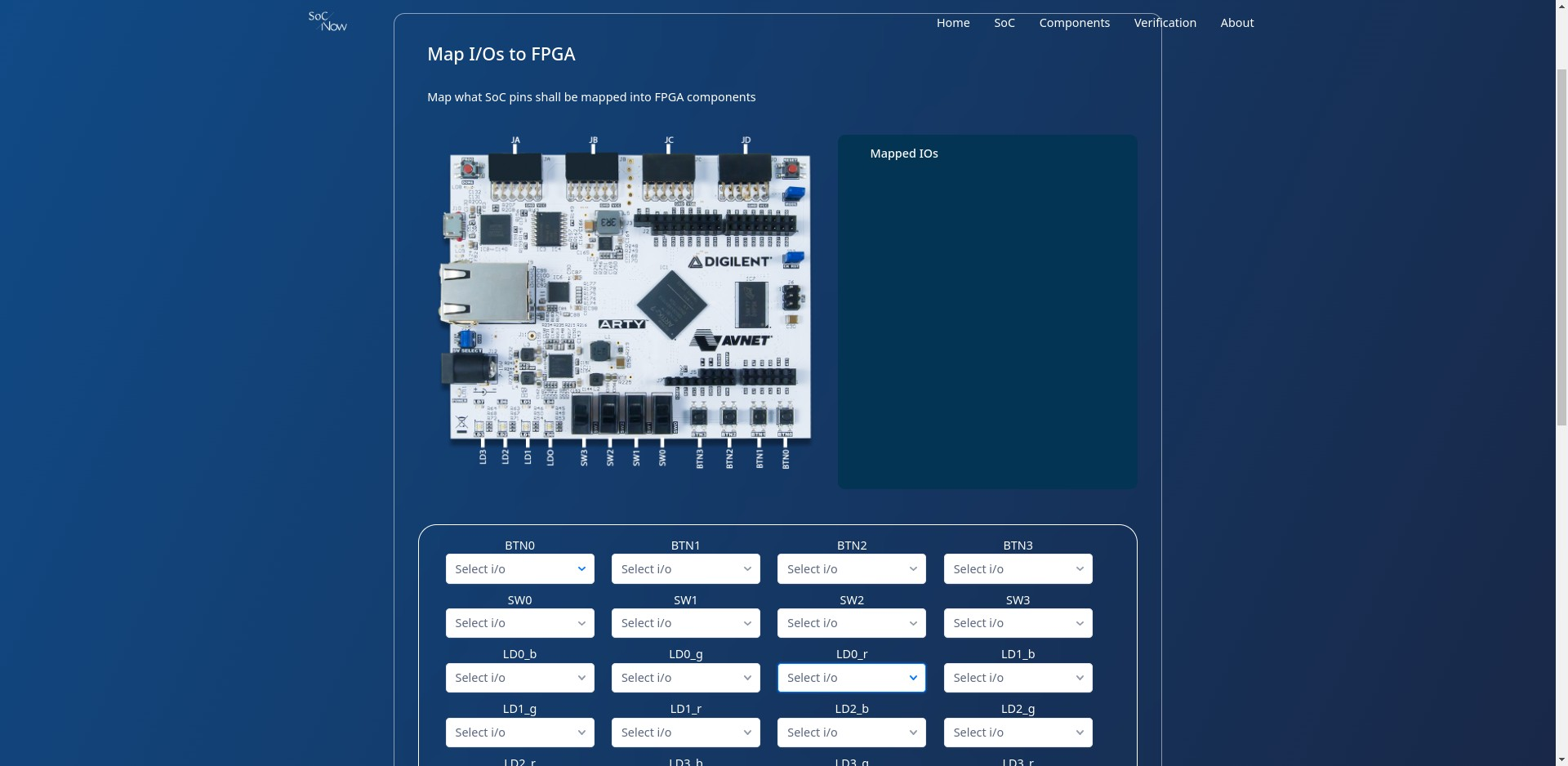


Fig. 5. SoC Configuration in SoC-Now

## Generic Frameworks

Following are the CHISEL based Generic Frameworks which all combined makes up the SoC-Now Generator.

* *“Caravan” intends to be equipped with a fully-fledged API for easily creating open-source bus protocols in CHISEL based designs. The SoC-Now is mainly depends upon the Generic Interconnect in terms of request and response. All the bus protocols are translated into generic interface of SoC-Now then it is used wherever it is needed.* *An illustration is provided in Fig. 4 the form of a block diagram interpreting the Implementation of the Tilelink Uncached Lightweight (TL-UL) Adapter [7] as present in the Caravan Framework. The request coming into the adapter from the left and going out into the device from the right is actually an generic interface. Anyone using any bus from Caravan Framework only has to poke values into this request and all the Tilelink or any other bus related functionality will execute by itself. Same scenario for response coming into adapter from the right and going out from the left.*

Fig. 6. FPGA I/O Mapping in SoC-Now

* *“Jigsaw” aims to be a helpful utility that provides the designers with pre-made useful peripherals including other IP blocks that provides re-usability and agile development of SoCs. All the peripherals are using generic bus interface comprises by Caravan that helps it to attach with any Caravan integrated SoC by simply plug-n-play.*
* *The open-source, RISC-V based “NucleusRV” core presently supports parameterizable I, M, F, and C extensions. It is utilized alongside the SoC-Now, which communicates with peripherals and memory via a generic interface.*

Diagram, schematic

Description automatically generated with medium confidence

## Features

SoC-Now Web App provides the following listed features as shown in Fig. 3. It uses the CHISEL based SoC Generator along with all the dependent Frameworks on the backend.

* *The “Custom SoC Generation” Feature is the core feature of SoC-Now as displayed in Fig. 5. This enables users to generate a customized SoC based on their provided configurations/ parameters. The software first requests configuration for the core only, such as which extensions should be added to the core, then which device(s) to be included in the SoC and then finally which Bus Interconnect to be used as the communication medium between the Core and the Devices.*

Fig. 7. RISC-V Compliance Test Selection and Report in SoC-Now

Graphical user interface

Description automatically generatedA screenshot of a computer

Description automatically generated with medium confidence

* *After successful SoC Generation user is then provided with the option to “Generate Bitstream” of the generated SoC. The software then lets user select which Field Programmable Gate Array (FPGA) he wants to select and after that it lets user to map the SoC’s inputs/outputs (I/O) onto the FPGA via GUI as visualized in Fig. 6. After I/O mapping. The user will then be prompted by the software to create a program that will be placed in the SoC's instruction memory and will be executed in FPGA emulation. Following the configuration stage, the bitstream generation process starts, which is handled by the open-source tool F4PGA.*
* *F4PGA (FOSS Flows For FPGA), which was Symbiflow before, is an open-source toolchain for programming FPGA. It is a CHIPS Alliance project workgroup. For HDLs, it offers the Bitstream FPGA synthesis solution. Only the Xilinx 7-Series, Lattice iCE40, and Lattice ECP5 FPGAs are presently supported. Its process is composed of three sections called Description, Frontend, and Backend, which are, respectively, HDL, Synthesis tools, and FPGA/ASIC tools. A circuit description is converted into a bitstream that is supported by a particular FPGA in 4 steps; Description of resource availability in FPGA chip, Synthesis with the help of Yosys, Implementation by placing and routing blocks and connect them from the synthesis design in specific chip location, and the final design is translated into features available for FPGA in FASM format, further it is translated into bitstream [8].*
* *Soc-Now also offers “Generate Reusable Component” feature displayed in Fig. 8. This enables the user to create any standalone, reusable SoC component, including any combination of cores, any device, or any bus interconnect. The generated component will support plug-and-play, so you may connect it to any of your designs and it will work independently. Each component has a generic Request and Response handshaking interface. Additionally, the component has already been verified, eliminating any possibility of a malfunction or error.*

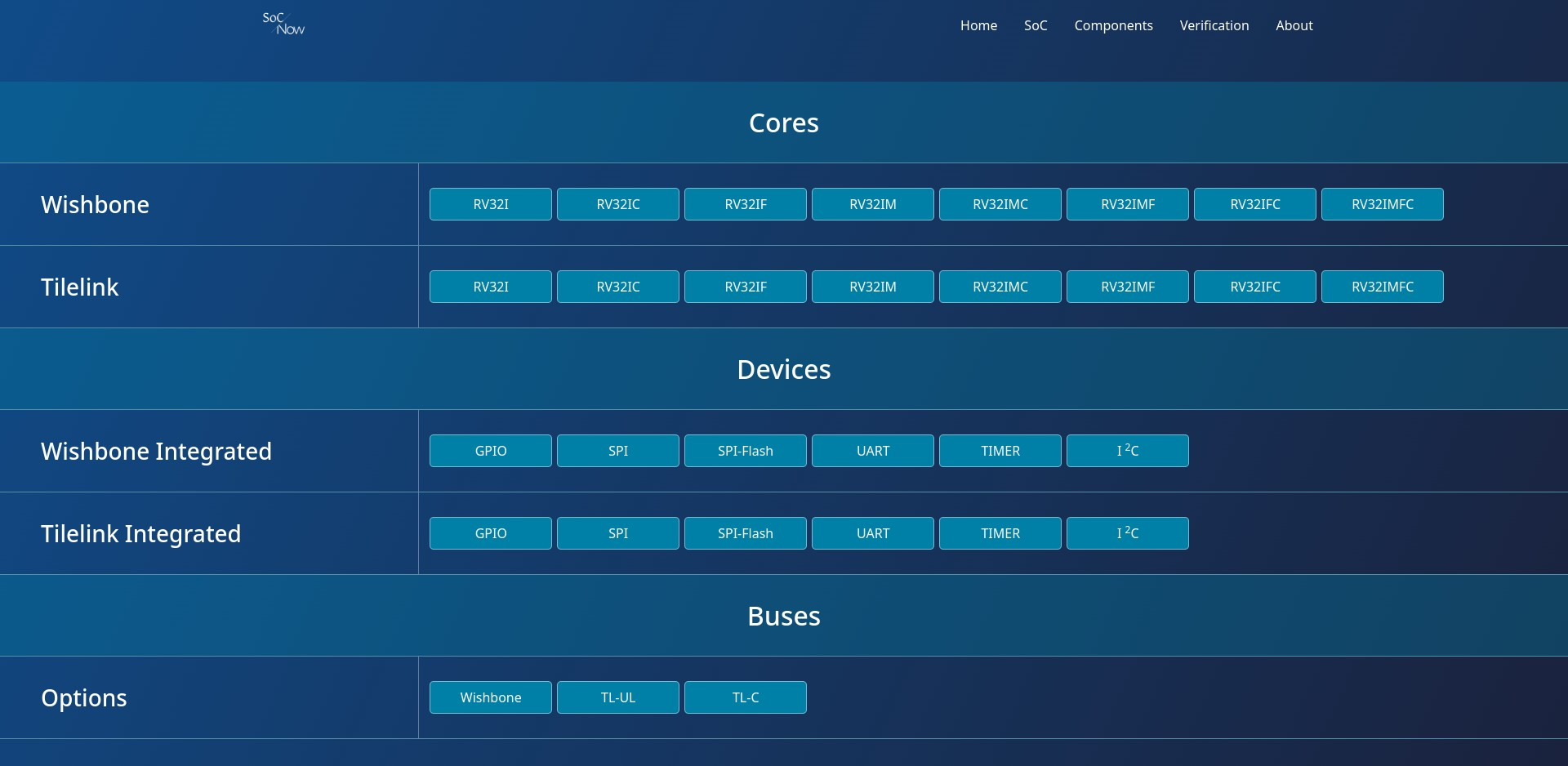


Fig. 8. Custom Component Selection of SoC-Now

* *SoC-Now also comes with a Verification Module, that on the web supports verification of Core by use of RISC-V standardized Compliance Tests [9]. User is prompted to select which instructions to be verified on which Core combination. A report is then generated outlining which instructions passed and failed the compliance tests. As shown in Fig. 7.*
* *Additionally, the Verification Module has an Addon called "Burq Suite." This provides the SoC-Now generated RISC-V based Core combination automated verification. Regression tests can be executed by users using testcases from the list that is provided. A RISC-V Instruction Set Simulator (ISS) like Spike [10] or OVPSim [11], which provides a golden model for any RISC-V Core, is used to simulate these tests first and a log file is dumped from it. After that, the Core is subjected to the same test, and the Core's log is also dumped. Once the Core logs and the ISS logs have been compared to see whether Core is up to standard, a report is maintained to keep track of which testcases passed on Core and which didn't, along with an explanation of what went wrong, and it is presented at the end. Additionally, the user can execute his or her own unique C-written test cases on the core. Another option in the verification suite is to use the randomly generated instructions by use of “RISC-V DV” provided by Google [12].*

Table 1 SoC-Now Component Specification Table

| Sr. | Components Specifications | | | |
| --- | --- | --- | --- | --- |
| Component | Framework | Description |
| 1. | Integer (i) Extension | NucleusRV | Integer (i) extension is the base extension of RISC-V ISA and shall be included must in any RISC-V Core. | |
| 2. | Multiplier (m) Extension | NucleusRV | Multiplier (m) extension is for Integer Multiplication and Division hardware support in core. | |
| 3. | Floating Point (f) Extension | NucleusRV | Floating Point (f) extension is for Single Precision Floating Point hardware support in core. | |
| 4. | Compressed (c) Extension | NucleusRV | Compressed (c) extension is for Compressed (low computing and memory resource consumption) hardware support in core. | |
| 5. | General Purpose Input Output (GPIO) | Jigsaw | GPIO for controlling devices by accepting input and sending output. | |
| 6. | Serial Peripheral Interface (SPI) | Jigsaw | SPI interface for communication with small peripherals or sensors. | |
| 7. | SPI-Flash | Jigsaw | SPI protocol specified for Flash memories. | |
| 8. | Universal Asynchronous Receiver Transmitter (UART) | Jigsaw | UART for asynchronous device-to-device communication. | |
| 9. | Timer | Jigsaw | The purpose of a timer is to notify the CPU when a certain amount of time has passed by raising an interrupt. | |
| 10. | Wishbone | Caravan | Bus Interface Crossbar implemented in accordance with Wishbone Bus Interface Protocols | |
| 11. | Tilelink Uncached Lightweight (TL-UL) | Caravan | Bus Interface Crossbar implemented in accordance with TL-UL Bus Interface Protocols by SiFive inc | |
| 12. | Tilelink Cached (TL-C) | Caravan | Bus Interface Crossbar implemented in accordance with TL-C Bus Interface Protocols by SiFive inc | |

# Conclusion and Future Work

##### SoC-Now is designed for those who want to quickly and simply build a SoC based on RISC-V. It allows the user to specify their requirements and have a SoC built, which they can then use for bitstream emulation. SoC generation is made possible by generic bus interfaces which is basically a generalized wrapper upon ports of bus protocols in terms of “Request” and “Response”. Additionally, it enables users to simply create any SoC component standalone for use in their design by implementing these generalized ports in their design. By using RISC-V compliance tests on their created designs, users can perform automated verification using the Verification component. Users can also perform manual verification by creating custom testcases and downloading VCDs. Future work for SoC-Now is expected to involve the integration of the GDS Backend workflow as well as new devices, bus interfaces, UVM verification environment, and RISC-V Core Extensions along with other FPGA support.

##### Acknowledgment

SoC-Now is a Final Year Project (FYP) of Undergraduate Software Engineering Students from UIT University (UITU) under the supervision of visionary mentors and with support and assistance of Micro Electronics Research Lab (MERL). The Authors would like to acknowledge the support of Open-Source Technologies and Community for providing with such revolutionary tools as RISC-V, CHISEL, F4PGA, Python and many more.

##### References

[1] G. Mastalerczuk, B. Borawska-Jarmułowicz, P. Dąbrowski, E. Szara, A. Perzanowska, and B. Wróbel, "Can the application the silicon improve the productivity and nutritional value of grass–clover sward in conditions of rainfall shortage in organic management?," Agronomy, vol. 10, no. 7, p. 1007, 2020.

[2] R.-V. Foundation. "The RISC-V Instruction Set

Manual, Volume II: Privileged Architecture, version

1.10." <https://content.riscv.org/wp-content/> uploads/2017/05/riscv-privileged-v1.10.pdf (accessed.

[3] R.-V. Foundation. "The RISC-V Instruction Set Manual, Volume I: User-Level ISA, document version

20191213. ." <https://riscv.org/specifications/> (accessed.

[4] Y. Lee et al., "An agile approach to building RISC-V microprocessors," ieee Micro, vol. 36, no. 2, pp. 8-20, 2016.

[5] J. Bachrach et al., "Chisel: constructing hardware in a scala embedded language," in DAC Design automation conference 2012, 2012: IEEE, pp. 1212-1221.

[6] M. Käyrä and T. D. Hämäläinen, "A survey on system-on-a-chip design using chisel HW construction language," in IECON 2021–47th Annual Conference of the IEEE Industrial Electronics Society, 2021: IEEE, pp. 1-6.

[7] X. Guo, D. Bates, R. Mullins, and A. Bradbury, "Muntjac multicore RV64 processor: introduction and microarchitectural guide," University of Cambridge, Computer Laboratory, 2022.

[8] K. E. Murray, M. A. Elgammal, V. Betz, T. Ansell, K. Rothman, and A. Comodi, "SymbiFlow and VPR: An open-source design flow for commercial and novel FPGAs," IEEE Micro, vol. 40, no. 4, pp. 49-57, 2020.

[9] V. Herdt, D. Große, and R. Drechsler, "Towards Specification and Testing of RISC-V ISA Compliance⋆," in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020: IEEE, pp. 995-998.

[10] M. S. Louis et al., "Towards deep learning using tensorflow lite on risc-v," in Third Workshop on Computer Architecture Research with RISC-V (CARRV), 2019, vol. 1, p. 6.

[11] F. Campos, L. Jellema, M. Lemmen, L. Müller, D. Sprenkels, and B. Viguier, "Assembly or optimized C for lightweight cryptography on RISC-V?," in International Conference on Cryptology and Network Security, 2020: Springer, pp. 526-545.

[12] V. Herdt, D. Große, E. Jentzsch, and R. Drechsler, "Efficient cross-level testing for processor verification: a RISC-V case-study," in 2020 Forum for Specification and Design Languages (FDL), 2020: IEEE, pp. 1-7.