eSim: An Open Source EDA Tool for Mixed-Signal and Microcontroller Simulations

Rahul Paknikar, Saurabh Bansode, Gloria Nandihal,
Madhav P. Desai, Kannan M. Moudgalya
Indian Institute of Technology Bombay
Mumbai, Maharashtra, India
e-mail: kannan@iitb.ac.in

Ashutosh Jha
Vellore Institute of Technology
Chennai, Tamil Nadu, India
e-mail: jashutosh0@gmail.com

2021 4th International Conference on Circuits, Systems and Simulation (ICCSS) | 978-1-7281-6752-7/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICCSS51193.2021.9464198

***Abstract*—The ability to carry out simulations before making a PCB can save a lot of time, effort and cost. This work explains the creation of an open source mixed-signal simulation software eSim that will be of great help to students, hobbyists, the SME sector and startups. Analog and digital components are respectively modelled using SPICE and a hardware descriptive language in eSim. Inclusion of AVR based microcontroller as a part of the digital circuit is demonstrated through its instructions implemented as a C code library. This methodology could be used to provide support to other microcontroller families, such as PIC, STM and also more sophisticated controllers. These concepts are demonstrated through a few examples.**

Keywords-eSim, Circuit Design, Circuit Simulations, Mixed-Signal Simulation, Open Source, Microcontroller Simulation, EDA tools, Crowd-sourcing, Massive Training

I. INTRODUCTION

Most electronic components used in India are imported as their manufacture is almost nonexistent in India. Considering the size of India's economy and the vulnerability in this strategic area, the Indian Government has been making efforts to promote the hardware industry in general, with some success recently.

One of the important ingredients required for the success of a high tech industry is the trained manpower. About a million students enrol in engineering every year in India. About a half of them undergo at least one course in electronic circuits. Unfortunately, the attainment level of these students is low, owing to the lack of resources and suitable pedagogy.

The main objective of this work is to address the above mentioned requirement of teaching circuit simulation to millions. For a national level training programme such as this to succeed, we need the software to be open source and easy to learn and use. For it to be acceptable, the software should be reasonably powerful.

We state the following requirements for this software: (1) It should be capable of carrying out mixed-signal circuit simulation. (2) It should have flexibility to include microcontrollers, FPGA, etc. (3) It should have a GUI to create the schematics. (4) It should come with easy to learn instructional material suitable for the target population.

While it can do mixed-signal simulation, Qucs [1] cannot handle PCB design and also cannot simulate circuits with microcontrollers. Although it can handle microcontrollers, MCUSim [2] does not have a GUI to create the schematics. While it can carry out microcontroller based simulations, GUI to create schematics in ISOTEL [3] requires EAGLE, a proprietary tool.

Earlier efforts to develop the simulation software resulted in the development of Oscad [4], using open source software KiCad [5] and Ngspice [6]. It was difficult to do mixed-signal simulation in Oscad. Moreover, Oscad could not handle microcontrollers. These difficulties have been addressed in the improved version of Oscad, which is now called as eSim [7]. Use of GHDL [8] to simplify mixed-signal simulation is addressed in Sec. II. The procedure to accommodate microcontrollers through a C code is explained in Sec. III. The final section is devoted to conclusion and future work.

II. NGHDL: A MIXED-SIGNAL SIMULATOR

The open source simulator Ngspice supports mixed-signal simulation through XSPICE extension. When a circuit comprising both analog and digital components is being simulated, the analog and the digital parts of the circuits are segregated first, Fig. 1. The analog part is simulated by the Spice3 simulator and the digital part is simulated by the XSPICE code models. Digital and analog components in the netlist are interfaced with each other through ADC and DAC provided by XSPICE.

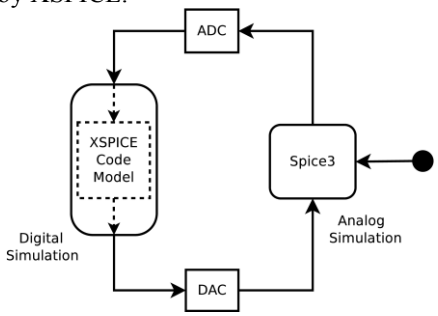


Figure 1. Logic of mixed-signal simulation in Ngspice

When an event occurs in a mixed-signal circuit, the analog data of the circuit is sent to the code model of the digital part of the circuit via an ADC. The resulting digital data is returned to the analog part via the DAC. This cycle repeats as long as events are generated within the stop time of the simulation. Digital components such as logic gates,

flip-flops and hybrid models such as ADC, DAC are defined using XSPICE's code modelling technique.

A. *Framework for NGHDL [9]*

Simulating complex digital circuits through XSPICE requires a thorough learning of its code modelling techniques [10] and Ngspice macros, and involves a steep learning curve. Using GHDL for digital circuits, as given in Fig. 2, reduces this complexity considerably. The part of this figure enclosed by dashed lines is identical to the entire Fig. 1. The only difference is that the computational part of XSPICE is bypassed in Fig. 2: these calculations are now executed through GHDL. One can also see that the infrastructure of Ngspice, in the form of ADC and DAC, is continued to be used in the new implementation, called NGHDL. Ngspice and GHDL work together through a client-server architecture, making NGHDL a type of glued-mode simulator. GHDL's foreign language interface, VHPIDIRECT, has been utilised to facilitate this Inter-Process Communication.

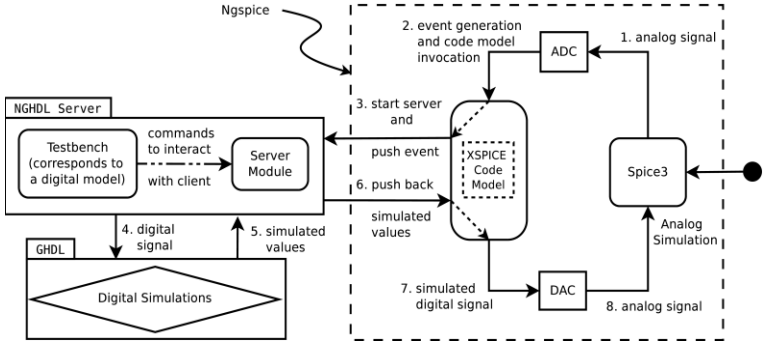


Figure 2. Workflow of NGHDL

The NGHDL interface in eSim allows a user to upload their digital model defined in VHDL language, auto-generate a code model corresponding to that digital model, and register it with Ngspice's XSPICE extension. Furthermore, it creates a schematic component for this digital model to be used in KiCad, based on the entity declaration in this VHDL model.

Using an efficient and synchronised method of creating sockets for client-server interaction, error-free iterations for the simulation are executed. By exploiting a large cache of loopback IP addresses, each code model can have its own NGHDL Server. As a result, the simulation speed is not affected and the previous state of an event is retained when the next event arrives. It is also possible to use the same NGHDL digital model multiple times in a netlist. At the end of a simulation, Ngspice terminates the NGHDL Server and releases the acquired resources for this simulation.

B. *Example 1: 7-Stage Ring Oscillator*

The 7 inverters in this oscillating configuration are cascaded such that they form a feedback loop [11] as shown in Fig. 3. The first 6 inverter gates, labelled from *X1* to *X6* in Fig. 3, are in CMOS logic created as a subcircuit and the 7th inverter is a digital model described in VHDL language.

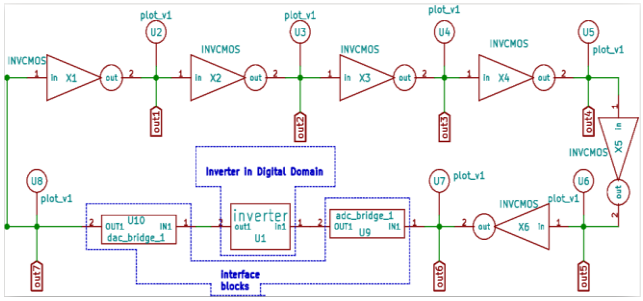
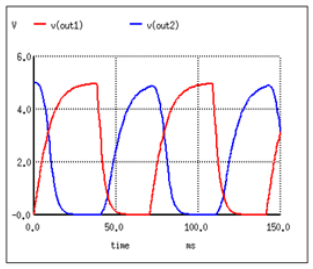
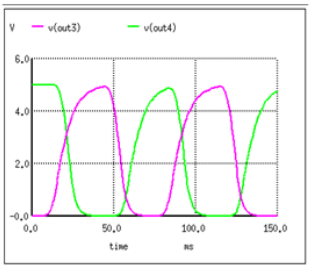


Figure 3. Schematic of the 7-Stage Ring Oscillator

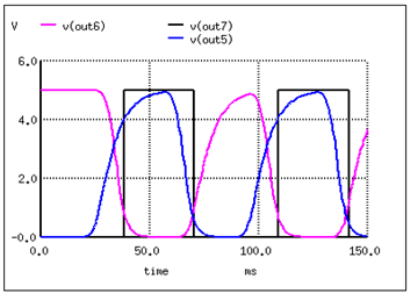
This circuit is self-oscillating and produces continuous waveforms. The difference in the continuous waveforms at the consecutive gates is due to the propagation delay when the signal passes from one gate to another. This propagation delay has a cumulative effect on the inverter gates put together in a ring configuration.



(a) Output of nodes: out1 and out2



(b) Output of nodes: out3 and out4



(c) Output of nodes: out5, out6 and out7

Figure 4. Output at each node of 7-Stage Ring Oscillator

Fig. 4 shows the output of this circuit at all 7 nodes which is in accordance with [11]. From Fig. 4(a), 4(b) and 4(c) respectively, we can observe the propagation delay between the *out1*, *out3* and *out5* nodes. Similar observations can be made for the *out2*, *out4* and *out6* nodes. The relation

between the *out7* and *out1* nodes can be observed as well through Fig. 4(a) and 4(c).

C. *Example 2: Thermometer to Binary Converter*

We present another example, essentially being a *Thermometer to Binary Converter* for a 2-bit flash ADC [12], whose schematic is shown in Fig. 5. The *Thermometer to Binary Encoder* block, which is highlighted as a digital block in the schematic, is implemented in VHDL language. This block takes input values from 3 comparators and converts them into a 2-bit binary number. These comparators are realised using an LM741 subcircuit, and a resistor ladder is used for setting the reference voltages. The input voltage, a piecewise linear source (PWL) signal varies from 0V to 3V in 100ms, 25 ms at each voltage level of 0V, 1V, 2V and 3V. This PWL signal is fed to the comparators, which generates either a high or a low output based on the comparison of reference voltages and PWL (input) signal values.

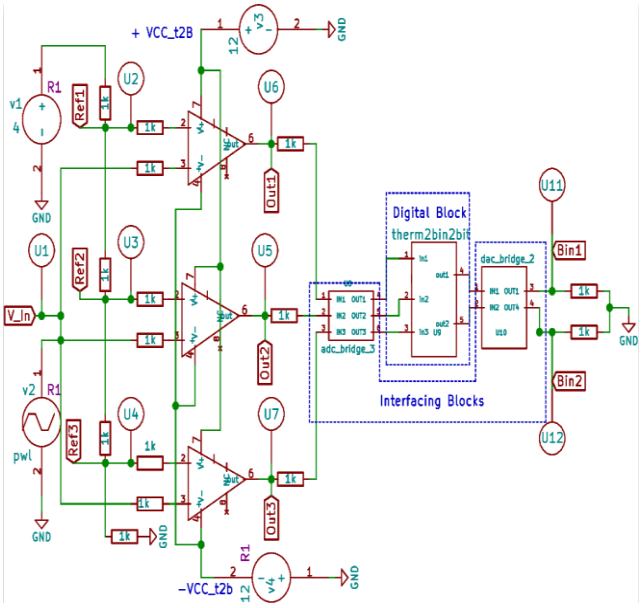
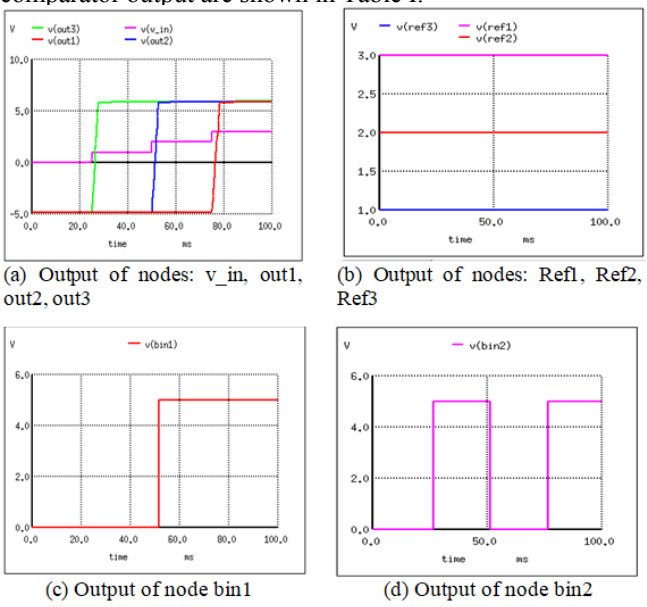


Figure 5. Schematic of thermometer to binary converter

TABLE I. OBSERVATIONS FOR THERMOMETER TO BINARY CONVERTER

Input	Node out1	Node out2	Node out3	Node bin1	Node bin2
0V	0	0	0	0	0
1V	1	0	0	0	1
2V	1	1	0	1	0
3V	1	1	1	1	1

Fig. 6 shows the eSim produced waveforms at the various nodes present in the circuit. Fig. 6(a) shows the input voltage given to the comparators and the output signals of the 3 comparators as well. Fig. 6(b) shows the reference voltages fed to the comparators. Fig. 6(c) and Fig. 6(d) show the results obtained through the *Thermometer to Binary Converter* circuit after the comparator outputs are processed through it. The binary numbers for combinations of comparator output are shown in Table I.



(a) Output of nodes: v_in, out1, out2, out3

(b) Output of nodes: Ref1, Ref2, Ref3

(c) Output of node bin1

(d) Output of node bin2

Figure 6. Waveforms for thermometer to binary converter

III. MICROCONTROLLER SIMULATIONS

A. *Implementing Microcontrollers*

Fig. 7 shows the microcontroller peripherals defined in VHDL language and its instruction set architecture (ISA) emulated in C programming language. Leveraging the NGHDL framework, the core and the peripheral modules of the microcontroller are statically linked at compile time. Its implementation involves the following components:

The peripheral module: This block, defined in VHDL language, declares the type and functionality of ports for a given microcontroller, and acts as a mediator between the core module and analog environment.

The core module: This block, written in C language, emulates the ISA of the microcontroller. The firmware is provided by the user in the form of a HEX file and is loaded in the ROM.

The utility module: For the peripheral module to access a variable and/or a function defined in the core module and vice-versa, a helper block, called utility module is written to typecast variables defined in the core module.

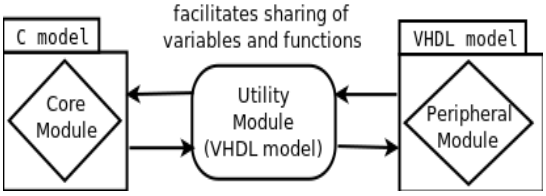


Figure 7. Block diagram of microcontroller implementation

As shown in Fig. 8, the workflow is similar to NGHDL with the minor change being, the entire microcontroller

framework is now simulated in GHDL. The digital signal received from the NGHDL Server is sent to the peripheral module of the microcontroller. At the first event of simulation, the core module loads the firmware in the modelled ROM. Based on the events present in the digital signal, the peripheral module shares the port values with the core module through the utility module. Once the core module starts executing the firmware, the changes of the affected ports' values are pushed back to the NGHDL Server through the peripheral module. These digital values are then converted to an analog signal, and an iteration of this simulation gets completed.

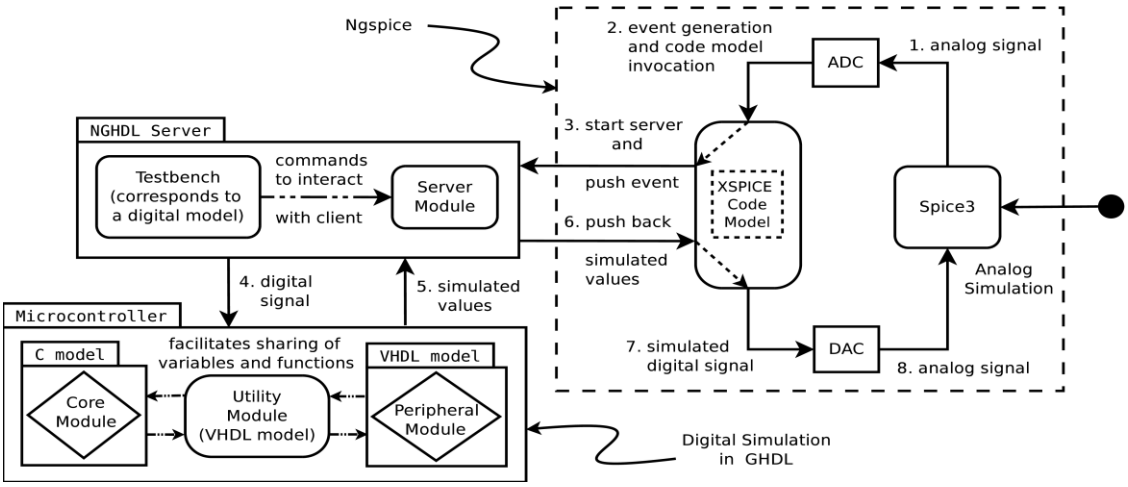


Figure 8. Workflow of microcontroller implementation

B. *Design Example: Pulse-Width Modulation (PWM) using ATtiny85 Microcontroller [13]*

The ATtiny85 microcontroller periphery is modelled in VHDL language, and the ISA modelled in C. The 2-bit binary number generated using the *Thermometer to Binary Converter* explained earlier, will be fed to the PB1 and PB2 pins of the ATtiny85 microcontroller as shown in Fig. 9. These inputs will control the PWM signal's duty cycle at PB0 pin. Fig. 10 shows the workflow for PWM signal control using the ATtiny85 microcontroller.

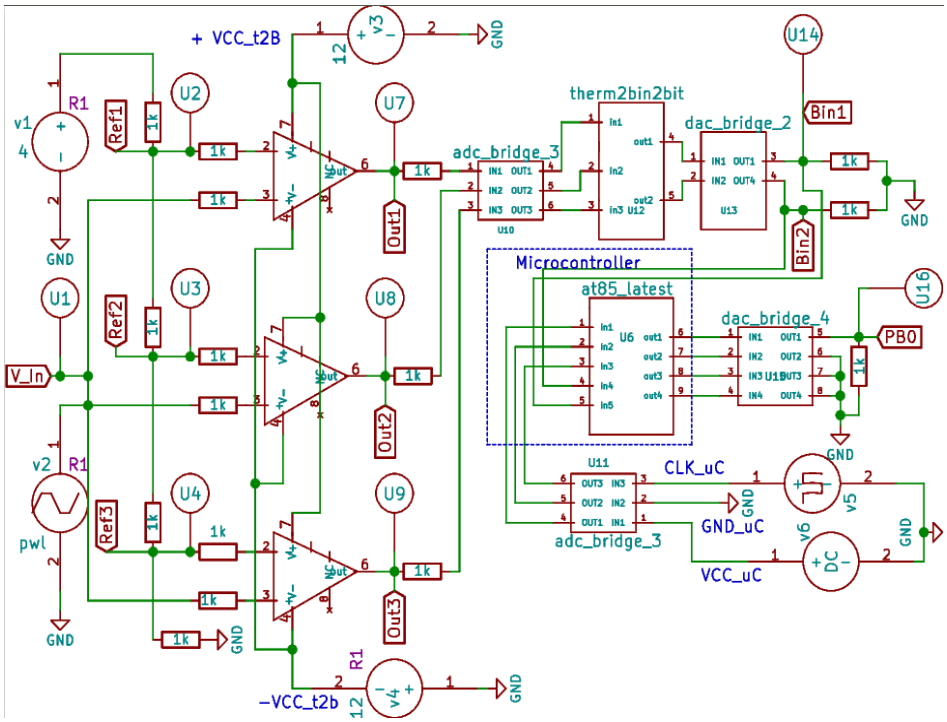


Figure 9. Schematic of PWM signal control using ATtiny85 microcontroller with inputs from thermometer to binary converter

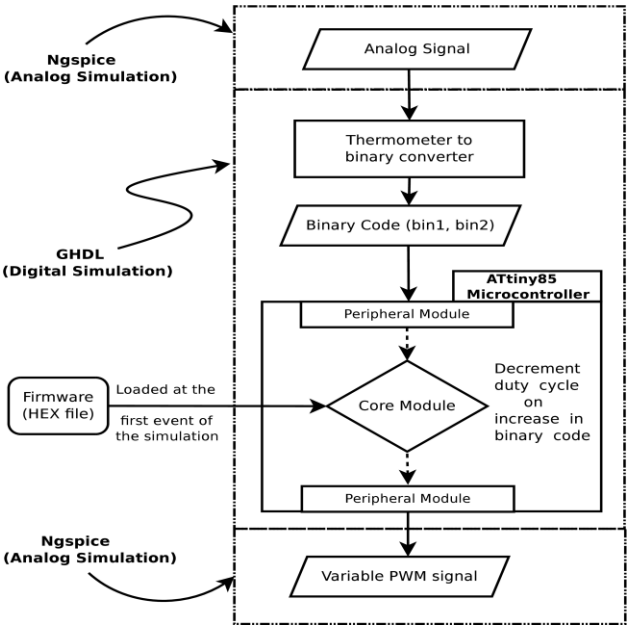


Figure 10. Workflow of PWM signal control through ATtiny85 Microcontroller

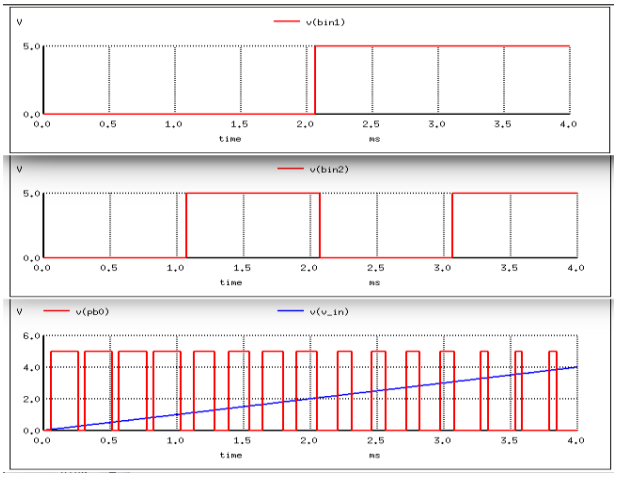


Figure 11. Output of the thermometer to binary converter in comparison with generated PWM signal

Input signals to the PB1 and PB2 pins of the ATtiny85 microcontroller are pushed to its peripheral module through the NGHDL Server. Note that the NGHDL Server for the microcontroller and the *Thermometer to Binary Encoder* are distinct. On the first event of the simulation, the firmware is loaded by the core module in the ROM. This core module, emulating the ATtiny85 microcontroller's ISA, decodes and executes the instructions as per the HEX file. Depending on the firmware, the values of the internal registers as well as the state of the General Purpose Input Output (GPIO) pins may get updated. In the latter case, the new state of GPIO pins is conveyed by the core module to the peripheral module using the utility module. The peripheral module, in turn, pushes back the simulated digital values for the output pin PB0 to the XSPICE client through its NGHDL Server. The duration, for which the PB0 pin of the microcontroller is ON, controls the duty-cycle of the PWM signal generated at PB0 pin and thus, the objective is achieved. As shown in Fig. 11, the duty cycle of the PWM signal changes based on values of input at PB1 and PB2 pins, and is summarised in Table II.

TABLE II. VARYING DUTY CYCLE FOR PWM SIGNAL

Pin PB1	Pin PB2	PWM signal duty cycle (%)
0	0	80
0	1	60
1	0	40
1	1	20

IV. CONCLUSION

In this paper, we have interfaced Ngspice with GHDL and arrived at NGHDL, which gives mixed-signal simulation capability to the open source EDA tool eSim. We have illustrated this capability with the help of two examples. By modelling the instruction set architecture in C language, an entire microcontroller can also be made available in eSim for mixed-signal simulations. This is illustrated through an 8-bit AVR microcontroller example. Through this method, many more microcontrollers can be accommodated in eSim, a task suitable for crowd-sourcing. It should also be easy to interface eSim with FPGA, which will be taken up in the future.

The schematic editor of KiCAD used in eSim makes it user friendly. To make it learner-friendly, we have created Spoken Tutorials [14] on eSim that are (1) suitable for self-learning, (2) dubbed into many Indian languages and (3) usable offline. Using this content, we have trained several thousand students and teachers on eSim. Students trained by us have migrated more than 100 examples from PSpice to eSim through a converter we have developed [15]. In the same link, another 100 circuits developed on eSim are also available. These circuits form an alternate documentation for eSim. They also demonstrate the versatility of eSim, which helps in promoting eSim. With this growth in the use of eSim, we believe that small and medium companies, at the minimum, will start using eSim, which in turn will improve the employment potential.

ACKNOWLEDGMENT

The authors would like to thank Prof. Pramod Murali, Department of Electrical Engineering, IIT Bombay and Mrs. Usha Viswanathan, FOSSEE, IIT Bombay for their guidance. We would also like to express our gratitude towards Powai Labs Technology Private Limited for their gratis contribution to the VHPIDIRECT package and Utility package of NGHDL. The FOSSEE project is funded by the National Mission on Education through ICT, Ministry of Education, Govt. of India.

REFERENCES

[1] M. Brinson and V. Kuznetsov, "Qucs-0.0.19S: A new open-source circuit simulator and its application for hardware design," *2016 International Siberian Conference on Control and Communications*

(SIBCON), Moscow, 2016, pp. 1-5, doi: 10.1109/SIBCON.2016.7491696.

[2] MCUSim. (2019). [Online]. Available: https://github.com/mcusim/MCUSim

[3] "Mixed Signal & Domain Simulation for Embedded Worlds". [Online]. Available: https://www.isotel.eu/mixedsim/index.html

[4] Y. Dilip Save *et al*., "Oscad: An open source EDA tool for circuit design, simulation, analysis and PCB design," *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, Abu Dhabi, 2013, pp. 851-854, doi: 10.1109/ICECS.2013.6815548.

[5] Jean-Pierre Charras and Fabrizio Tappero. KiCad. (2019). [Online]. Available: https://docs.kicad.org/4.0/en/kicad/kicad.pdf

[6] Holger Vogt, Marcel Hendrix and Paolo Nenzi. Ngspice User's Manual Version 31 (Describes ngspice release version). (2019). [Online]. Available: http://ngspice.sourceforge.net/docs/ngspice-31-manual.pdf

[7] eSim. (2021). FOSSEE, IIT Bombay. [Online]. Available: https://github.com/FOSSEE/eSim

[8] GHDL. (2017). Tristan Gingold. [Online]. Available: http://ghdl.free.fr/

[9] NGHDL. (2021). FOSSEE, IIT Bombay. [Online]. Available: https://github.com/FOSSEE/NGHDL

[10] F. L. Cox, W. B. Kuhn, J. P. Murray and S. D. Tynor, "Code-level modeling in XSPICE," *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, San Diego, CA, USA, 1992, pp. 871-874 vol.2, doi: 10.1109/ISCAS.1992.230083.

[11] V. Sikarwar, N. Yadav and S. Akashe, "Design and analysis of CMOS ring oscillator using 45 nm technology," *2013 3rd IEEE International Advance Computing Conference (IACC)*, Ghaziabad, 2013, pp. 1491-1495, doi: 10.1109/IadCC.2013.6514447.

[12] M. P. Ajanya and G. T. Varghese, "Thermometer code to Binary code Converter for Flash ADC - A Review," *2018 International Conference on Control, Power, Communication and Computing Technologies (ICCPCCT)*, Kannur, 2018, pp. 502-505, doi: 10.1109/ICCPCCT.2018.8574244.

[13] ATtiny85 - 8-bit AVR Microcontrollers. (2021). Microchip Technology Inc. [Online]. Available: https://www.microchip.com/wwwproducts/en/ATtiny85

[14] K. M. Moudgalya, "IT Skills Training through Spoken Tutorials for Education and Employment: Reaching the Unreached," CEC Journal of Digital Education, vol. 1, no. 1, pp. 19-62, 2017. [Online]. Available: http://spoken-tutorial.org/media/CEC.pdf

[15] "Completed PSPICE to KiCAD Circuits". [Online]. Available: https://esim.fossee.in/circuit-simulation-project/completed-circuits/pspice-to-kicad