

A Glitch-Free Clock Multiplexer for Non-Continuously Running Clocks

FOSSEE Semester-Long Internship — Project Report

Design, Simulation and Verification using eSim

Name: Bhargavi Ghanshyam Hulke

Institute / College: VIT Bhopal University

FOSSEE Project Title: A Glitch-Free Clock Multiplexer for Non-Continuously Running Clocks

Submission Date: 23 August 2026

1. Introduction

This project implements, in KiCad schematic capture and eSim/ngspice simulation, the novel glitch-free clock multiplexer architecture proposed by Zeidler, Schrape, Breitenreiter and Krstić, presented at the 2020 23rd Euromicro Conference on Digital System Design (DSD).

2. Theory / Description

2.1 Motivation

Modern systems-on-chip frequently contain blocks that must be triggered by more than one clock source depending on system state (for example, switching between a system clock and an SPI clock). A clock multiplexer (clock switch) selects between such clock sources, but naive switching can produce glitches — narrow, unintended pulses at the output that can cause metastability or erroneous behaviour in the downstream logic. Conventional glitch-free multiplexer architectures (based on cross-coupled flip-flop synchronizer chains) solve the glitch problem, but they share a critical limitation: they require both candidate clocks to be actively running throughout the switch. If the currently selected clock stops toggling before the switch is completed, the conventional multiplexer gets permanently stuck, and the output never reflects the newly selected clock.

2.2 Conventional Reference Design

The conventional (reference) glitch-free multiplexer is built from two cross-coupled two-flip-flop synchronizer chains. Each chain is clocked by its own input clock and produces an enable signal (`en_clk0` / `en_clk1`). Combinational logic (NOR/AND gates) generates the D-inputs `sel_clk0` and `sel_clk1` from the selector signal `sel` and the opposite chain's enable output, ensuring that a chain is only allowed to enable its clock once the other chain has confirmed its own clock is disabled — the classic break-before-make interlock. The two gated clocks (`clk_0 AND en_clk0`, `clk_1 AND en_clk1`) are finally combined with an OR gate to produce the multiplexer output `clk_out`.

2.3 Novel Contribution: Watchdog Timers

The paper's key contribution is the addition of two watchdog timer blocks, `Timer_clk0` and `Timer_clk1`, layered on top of the conventional multiplexer. Each timer is itself a small synchronizer (at minimum two flip-flops) with asynchronous reset. `Timer_clk0` is clocked by `clk_1`, has its data input tied to `sel`, and its asynchronous reset tied to `clk_0`; its output is the signal `disable_clk0`. `Timer_clk1` is the mirror image: clocked by `clk_0`, data input tied to the inverted selector, and reset tied to `clk_1`; its output is `disable_clk1`.

As long as both clocks keep toggling, each timer is continuously held in reset by its own observed clock, so `disable_clk0` and `disable_clk1` remain at logic-0 and the circuit behaves exactly like the conventional reference design. The novel behaviour appears when the currently active clock stops before a switch is

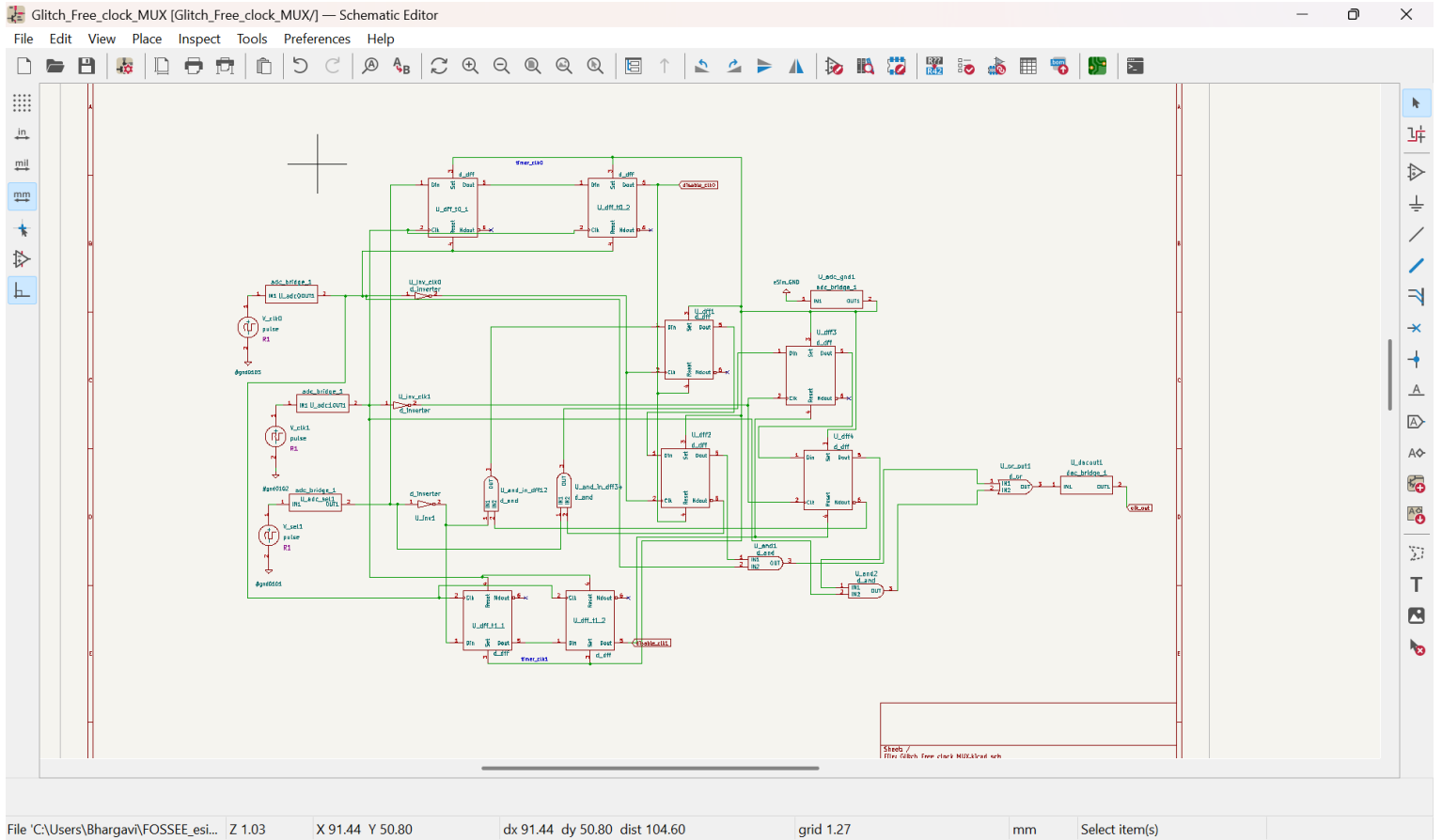
requested: because the corresponding timer is no longer being reset, the selector value propagates through the timer's flip-flops (clocked by the other, still-running clock) and, after at most two cycles of that clock, asserts the relevant disable signal. This disable signal is wired to the asynchronous reset of the conventional multiplexer's own flip-flops, forcibly clearing the stuck enable signal and allowing the new clock to be gated through to the output — something the conventional design alone cannot do.

2.4 Timer Dimensioning

The number of flip-flops in each timer chain must be sized to the ratio of the two clock frequencies to avoid a premature or unintended reset of the enable signal. At least two flip-flops are required (to guard against metastability); for larger frequency ratios between the fast and slow clock, additional stages (or a binary counter) are needed, following the dimensioning rule given in Section III-C of the reference paper.

3. Circuit Diagram(s)

The circuit was captured in KiCad using the eSim digital/hybrid symbol libraries (d_dff, d_and, d_or, d_inverter, adc_bridge_1, dac_bridge_1). The design comprises: the conventional cross-coupled multiplexer (four D flip-flops, two AND gates, one OR gate, three inverters), and the two watchdog timers (four additional D flip-flops), together with analog-to-digital and digital-to-analog bridges to interface the ngspice pulse sources (clk_0, clk_1, sel) with the XSPICE digital domain.



Full top-level schematic (KiCad) — conventional multiplexer + Timer_clk0 + Timer_clk1



Block diagram reference (Figure 2 of the paper): the conventional clock-switch block receives `clk_0`, `clk_1` and `sel` and produces `clk_out`; `Timer_clk0` and `Timer_clk1` observe `clk_0/clk_1` and `sel` to produce `disable_clk0` and `disable_clk1`, which reset the conventional block's internal flip-flops.

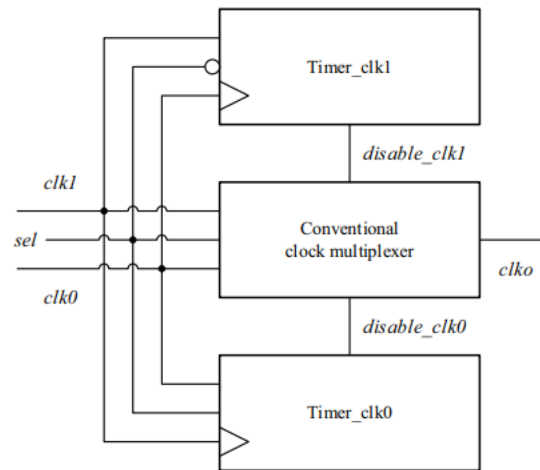


Figure 2. Block diagram of the multiplexer

Block diagram: conventional multiplexer + timers (reference Figure 2)

4. Results / Output

4.1 Simulation Setup

Simulation was carried out in eSim with clk0 and clk1 modelled as periodic PULSE sources of unequal period, and sel modelled as a PULSE source toggling the selected clock partway through the run.

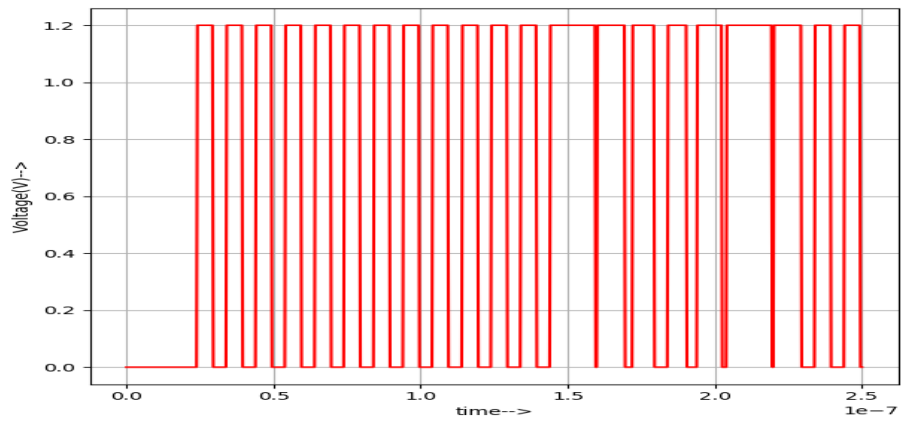
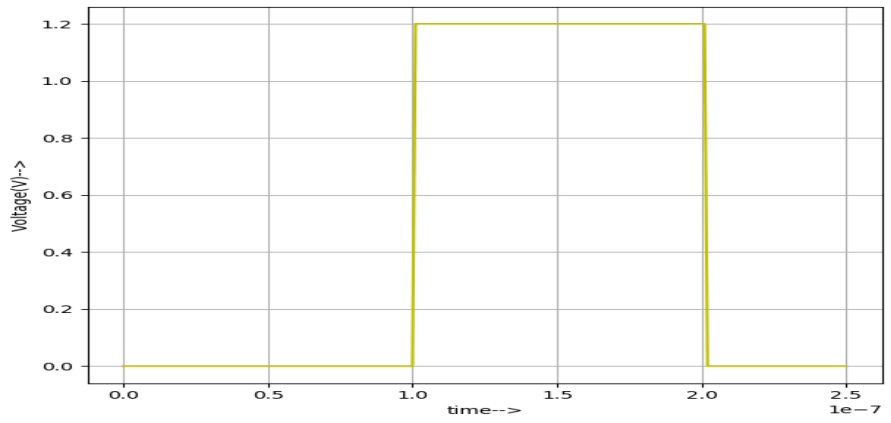
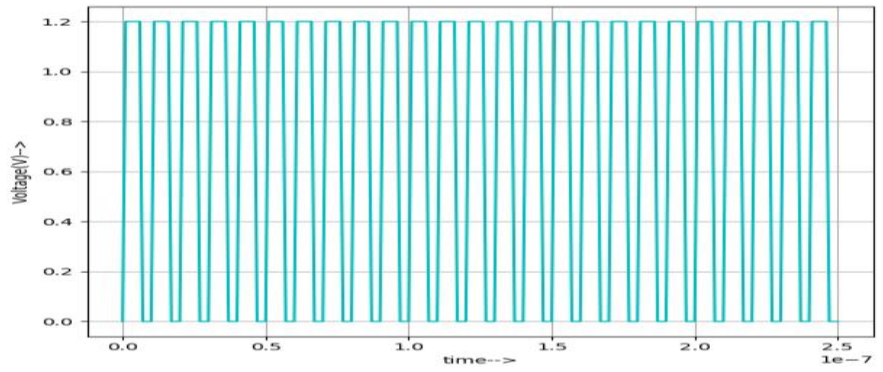
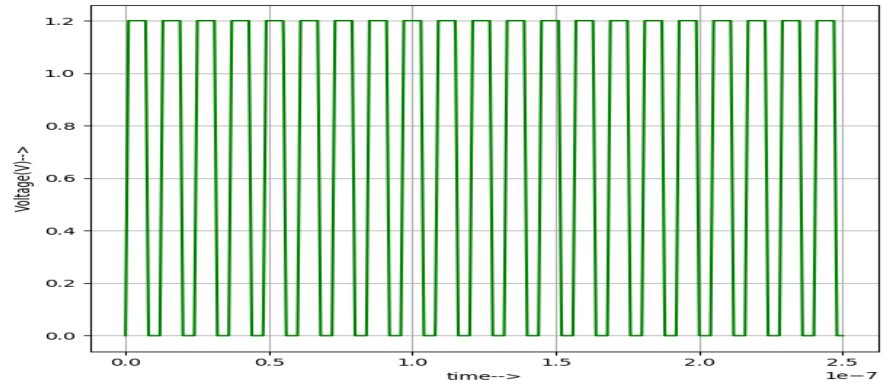
clk_0 period: 10ns

clk_1 period: 12ns

sel timing (delay / pulse width / period): 100ns/1n/200ns

4.2 Baseline Switching Behaviour (both clocks continuously running)

With both clk_0 and clk_1 toggling throughout the run, the disable_clk0/disable_clk1 timer outputs remain at logic-0, and the circuit reproduces the conventional reference design's behaviour: clk_out follows clk_0 while sel selects clock 0, holds at its current level (no truncated pulse) during the hand-off, then follows clk_1 once the switch has propagated through the interlock — and mirrors this on the return transition.

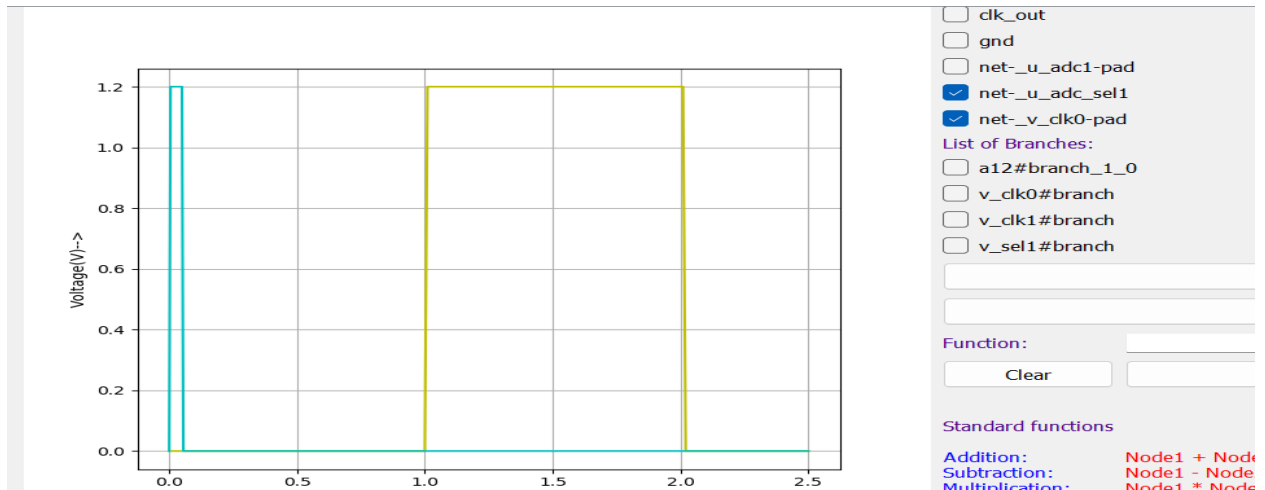


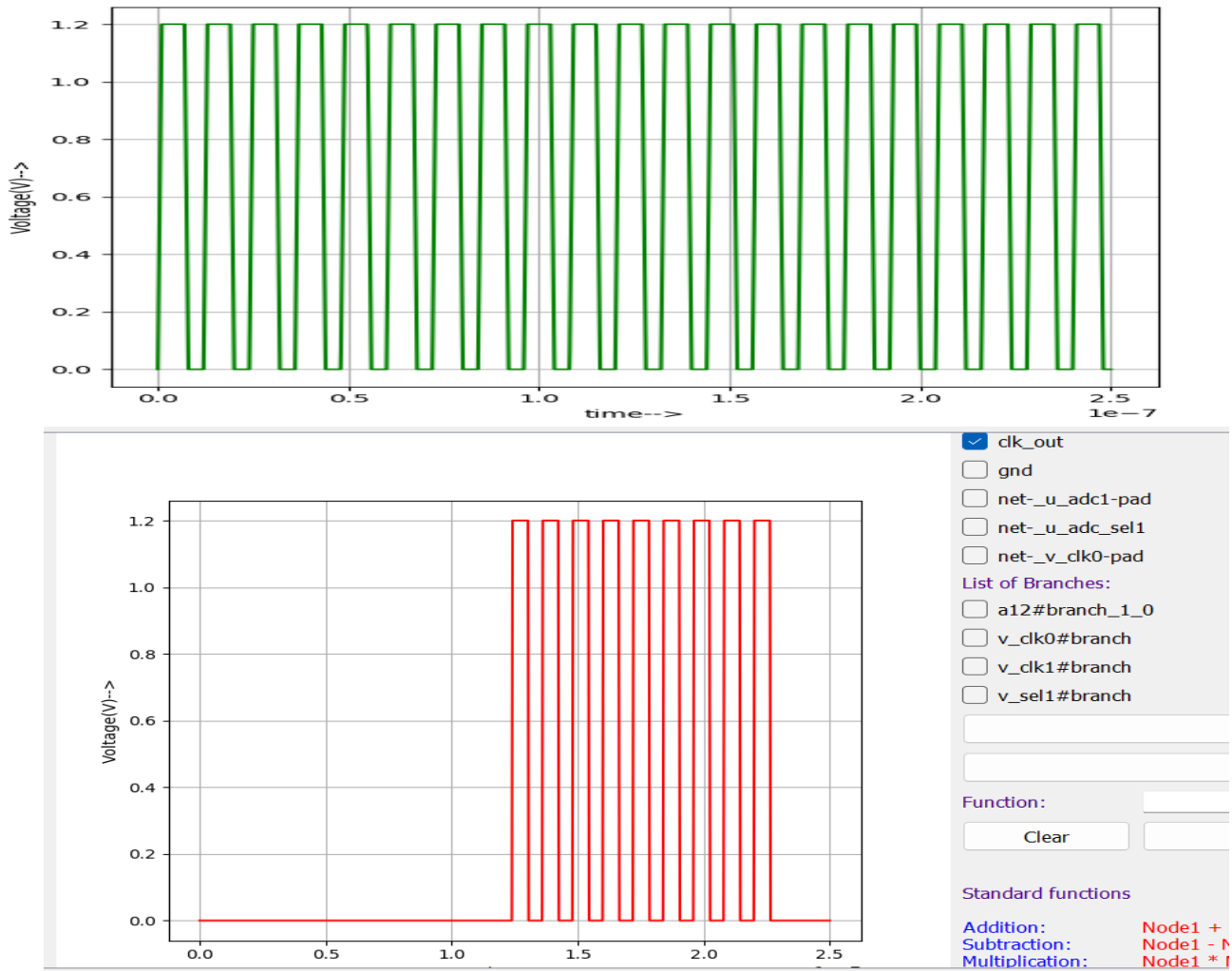
eSim plot: clk_0, clk_1, sel and clk_out respectively — both clocks continuously running

4.3 Non-Continuous Clock Behaviour (novel case)

To exercise the paper's actual novel contribution, one of the two clocks (e.g., clk_0) is held flat (stopped, at its inactive level) for a period spanning a sel transition. This activates the corresponding timer, which is no longer being reset by the stopped clock; after the required number of the other clock's cycles, the timer asserts its disable signal, forcibly resetting the stuck enable flip-flops and allowing the new clock through to clk_out — reproducing the behaviour shown in Figure 5 of the reference paper.

To exercise this behaviour, clk0 was replaced with a piecewise-linear (PWL) source rather than a periodic PULSE source, since a strictly periodic source cannot be made to stop. clk_0 was defined to complete a single 10 ns cycle — rising at $t = 0.5$ ns, held high until $t = 5$ ns, falling at $t = 5.5$ ns — and then held permanently low for the remainder of the simulation, modelling a clock that has stopped in its inactive state. clk_1 was left as an ordinary, continuously-running PULSE source with a 12 ns period, deliberately different from clk_0's 10 ns period so the two remain visually distinguishable. sel was defined to stay low until $t = 100$ ns — well after clk_0 had already stopped at $t = 5.5$ ns and then transition high for the rest of the run, requesting a switch to clk_1 while clk_0 is already confirmed inactive. With these values, Timer_clk0 is no longer being reset once clk_0 stops, so the sel = 1 request propagates through its two flip-flops on clk1's rising edges: disable_clk0 is expected to assert roughly two clk1 cycles after the sel transition (around $t = 128$ ns), after which the interlock releases and clk_out is expected to resume toggling at clk1's rate shortly after (around $t = 140\text{--}160$ ns).





eSim plot: clk_0 & sel, clk_1, clk_out

4.4 Observations

- clk_out exhibits no truncated / runt pulses at either switching boundary.
- The output correctly gates to the newly selected clock even when the previously active clock has stopped toggling.
- Timer outputs (disable_clk0 / disable_clk1) remain de-asserted whenever both clocks are continuously running, matching the paper's stated equivalence to the conventional design in that case.
- The Set pins on all eight flip-flops (both the conventional multiplexer's four and the two timers' four) were tied permanently inactive and left unused in this design, since only Reset-based asynchronous clearing was required by the architecture. This was a deliberate design choice made explicit here, not an oversight.

5. References

- [1] S. Zeidler, O. Schrape, A. Breitenreiter and M. Krstić, "A Glitch-free Clock Multiplexer for Non-Continuously Running Clocks," 2020 23rd Euromicro Conference on Digital System Design (DSD), 2020, pp. 11-15, doi: 10.1109/DSD51259.2020.00013.
<https://conferences.computer.org/dsd/pdfs/DSD2020-3gccs2DX4TPUFhzwStqjoz/953500a011/953500a011.pdf>
- [2] R. Mahmud, "Techniques to make clock switching glitch free," EE Times, Jun. 2003.
https://www.eetimes.com/document.asp?doc_id=1202359
- [3] R. Ginosar, "Fourteen Ways to Fool Your Synchronizer," Proc. 9th Intl. Symposium on Asynchronous Circuits and Systems, IEEE Computer Society, 2003.
- [4] eSim — Open Source EDA Tool, FOSSEE, IIT Bombay. <https://esim.fossee.in/>
- [5] ngspice User's Manual — XSPICE Digital / Mixed-Signal Extensions.
<http://ngspice.sourceforge.net/docs.html>
- [6] KiCad EDA Documentation. <https://docs.kicad.org/>