Title of the experiment:

Simulation of SPI Multi-Slave Communication Protocol Using eSim

Theory:

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol widely used in embedded systems for short-distance, high-speed data transfer. It operates in a Master-Slave architecture where the Master generates the Serial Clock (SCLK) and controls data flow using active-low Chip Select (CS) lines to address individual Slave devices. Unlike I2C, SPI uses separate dedicated lines for data transmission — MOSI (Master Out Slave In) and MISO (Master In Slave Out) — enabling full-duplex communication.

This experiment simulates a 1-Master 2-Slave SPI bus using eSim's Mixed-Signal environment. The SPI Master is implemented as a Verilog RTL block compiled using NgVeri, while the Clock and Reset stimuli are provided as analog pulse sources converted to digital logic levels via ADC bridge components. The Master's Finite State Machine (FSM) sequentially addresses Slave 1 and Slave 2 using independent chip select lines CS1 and CS2, transmitting an 8-bit data byte to each slave in SPI Mode 0 (CPOL=0, CPHA=0) with MSB transmitted first.

Schematic Diagram:

The mixed-signal schematic was designed in eSim's KiCad editor. It consists of two analog pulse sources (CLK and RESET), two ADC bridge components for analog-to-digital conversion, one SPI Master NgVeri block (U3), and two SPI Slave NgVeri blocks (U4 and U5). The Master and Slaves are interconnected via the SPI bus lines — MOSI, MISO, SCLK, CS1, and CS2.

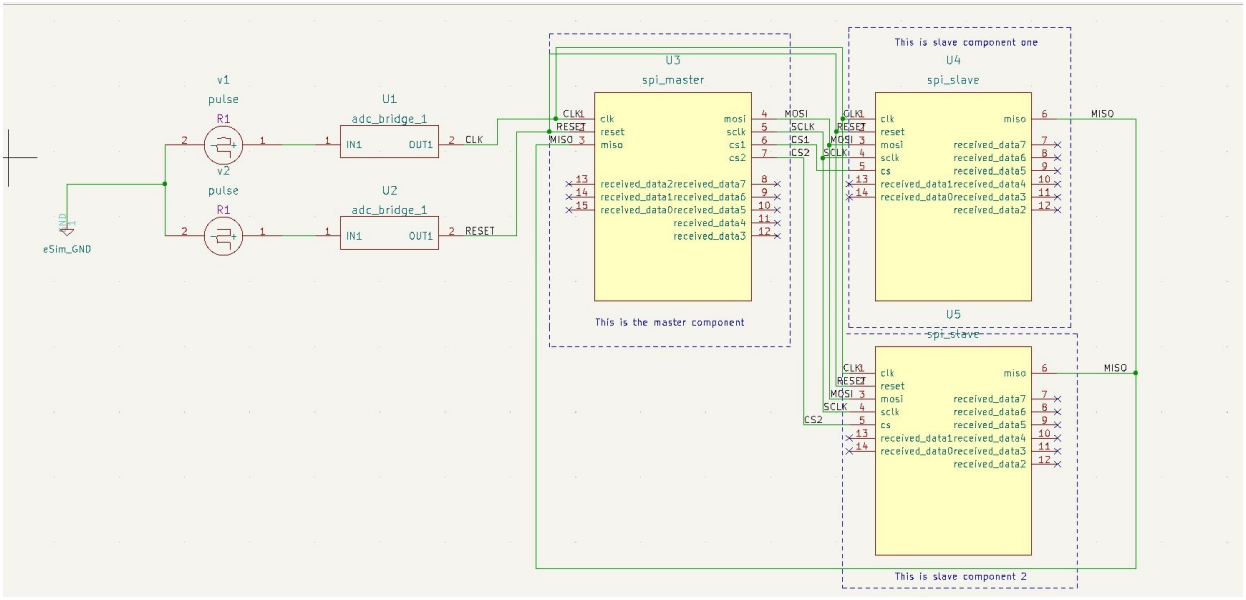

Figure 1: Mixed-Signal Schematic of SPI Multi-Slave System in eSim KiCad

Nomenclature:

Master Side:

- **CLK:** System clock input to the SPI Master, generated by an analog pulse source (100 kHz)
- **RESET:** Active-low synchronous reset signal. FSM is held in reset while RESET is LOW (first 200us). FSM begins operation when RESET goes HIGH.
- **MOSI:** Master Out Slave In — serial data line carrying 8-bit data from Master to Slaves
- **SCLK:** Serial Clock generated by the Master, synchronizes data shifting between Master and Slaves
- **CS1:** Active-low Chip Select for Slave 1. Driven LOW by Master during Slave 1 transaction
- **CS2:** Active-low Chip Select for Slave 2. Driven LOW by Master during Slave 2 transaction
- **MISO:** Master In Slave Out — serial data line from Slaves back to Master

Slave Side:

- **MOSI:** Serial data input received from Master
- **SCLK:** Clock input from Master for synchronizing data sampling
- **CS:** Active-low chip select input. Slave operates only when CS is LOW
- **MISO:** Serial data output driven by Slave back to Master

Data Flow and Transmission Sequence:

1. At t=0, RESET is LOW. The Master FSM is held in reset state, clearing all internal registers including shift registers, bit counter, and state register.
2. At t=200us, RESET goes HIGH. The Master FSM begins operation from the IDLE state.
3. The FSM transitions IDLE → LOAD_S1. CS1 is driven LOW (Slave 1 selected), CS2 remains HIGH. The 8-bit transmit data 10101010 (0xAA) is loaded into the shift register.
4. The FSM enters TRANS_S1 state. For each rising edge of CLK, SCLK toggles. On the falling edge of SCLK, MOSI is driven with the current MSB of the shift register. On the rising edge of SCLK, the next bit is loaded. This repeats for 8 clock cycles, transmitting all 8 bits serially.
5. The FSM transitions to DONE_S1. CS1 is deasserted HIGH, SCLK returns LOW, and MOSI is cleared.
6. The FSM transitions DONE_S1 → LOAD_S2. CS2 is driven LOW (Slave 2 selected), CS1 remains HIGH. The 8-bit transmit data 11001100 (0xCC) is loaded into the shift register.

7. The FSM enters TRANS_S2 state and repeats the same serial transmission process for Slave 2.
8. The FSM transitions to DONE_S2. CS2 is deasserted HIGH, completing the full multi-slave transaction cycle. The FSM returns to IDLE.

Simulation Results (eSim NgSpice):

The transient simulation was performed for 1ms with a step size of 1us. All key SPI signals were observed and verified in NgSpice. The following plots show the input stimulus and output SPI signals. Note: The MISO line did not produce observable activity in this simulation. As the MISO net is shared between both slaves, further investigation into tri-state driving conditions is required to fully verify the MISO response.

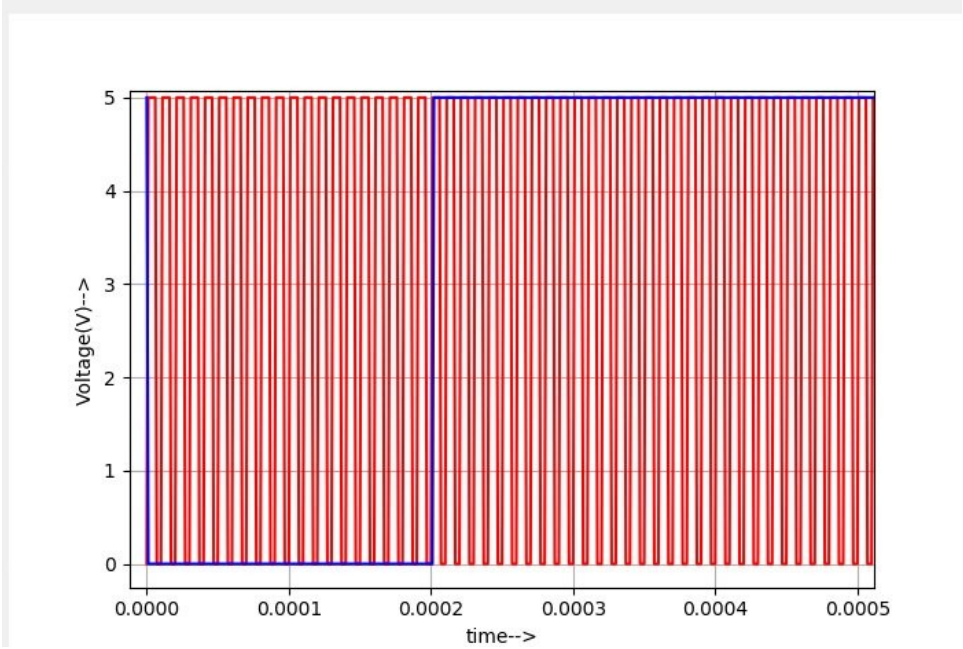

Figure 2: CLK (red) and RESET (blue) — 100 kHz system clock and 200us reset signal

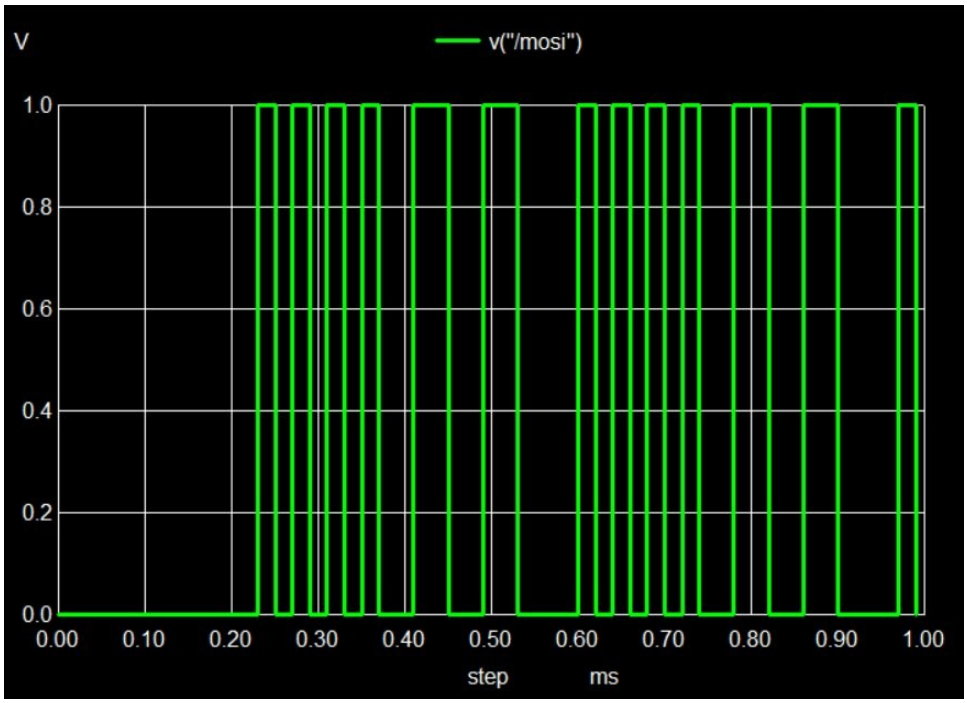

Figure 3: MOSI Signal — serializing 10101010 (Slave 1) then 11001100 (Slave 2), repeating each cycle

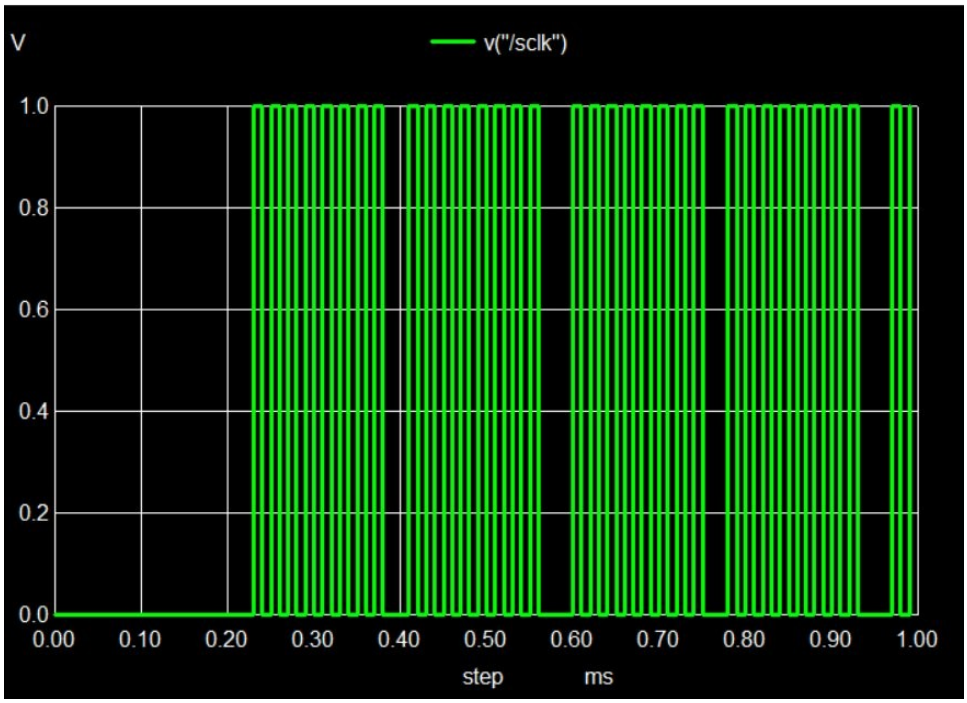

Figure 4: SCLK Signal — toggling synchronously with system clock during both transactions

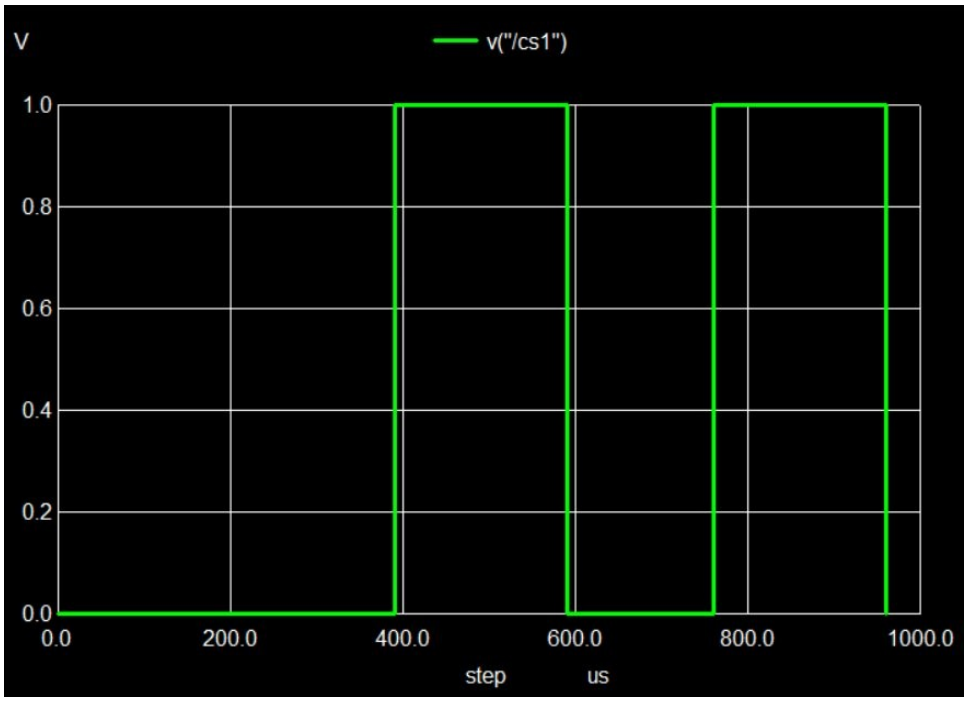

Figure 5: CS1 Signal — asserts LOW during Slave 1 transaction, deasserts HIGH after 8 bits

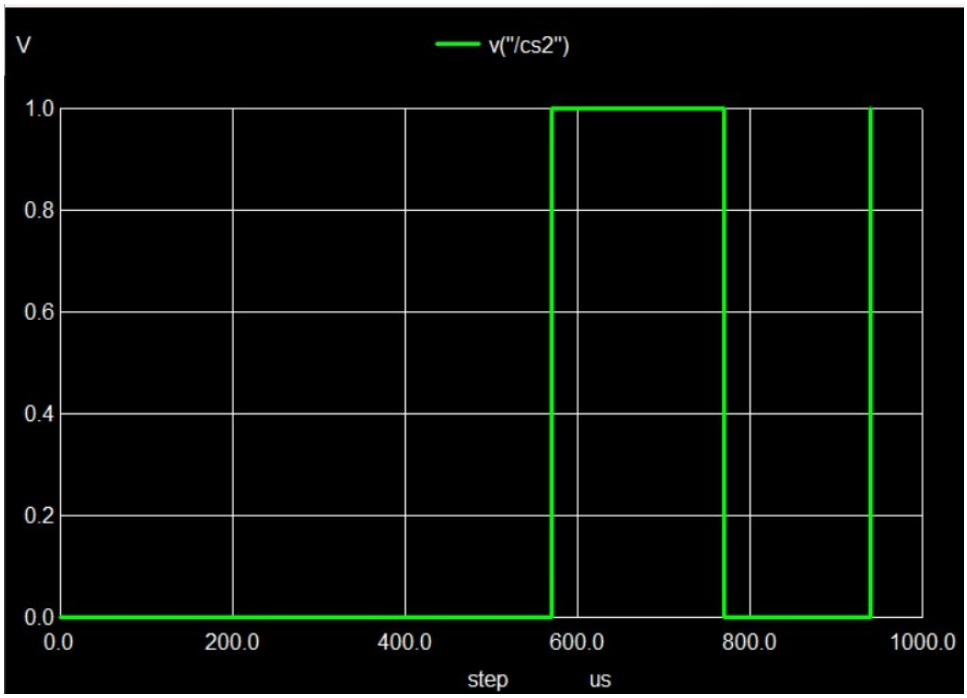

Figure 6: CS2 Signal — asserts LOW during Slave 2 transaction, deasserts HIGH after 8 bits

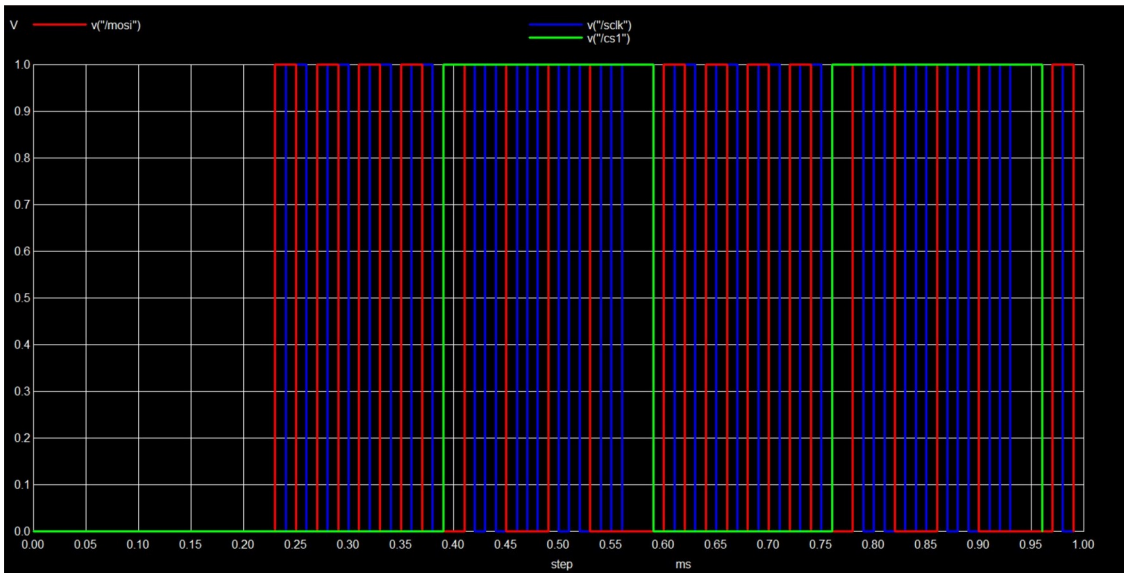

Figure 7: Combined — MOSI (red), SCLK (blue), CS1 (green) showing Slave 1 transaction timing

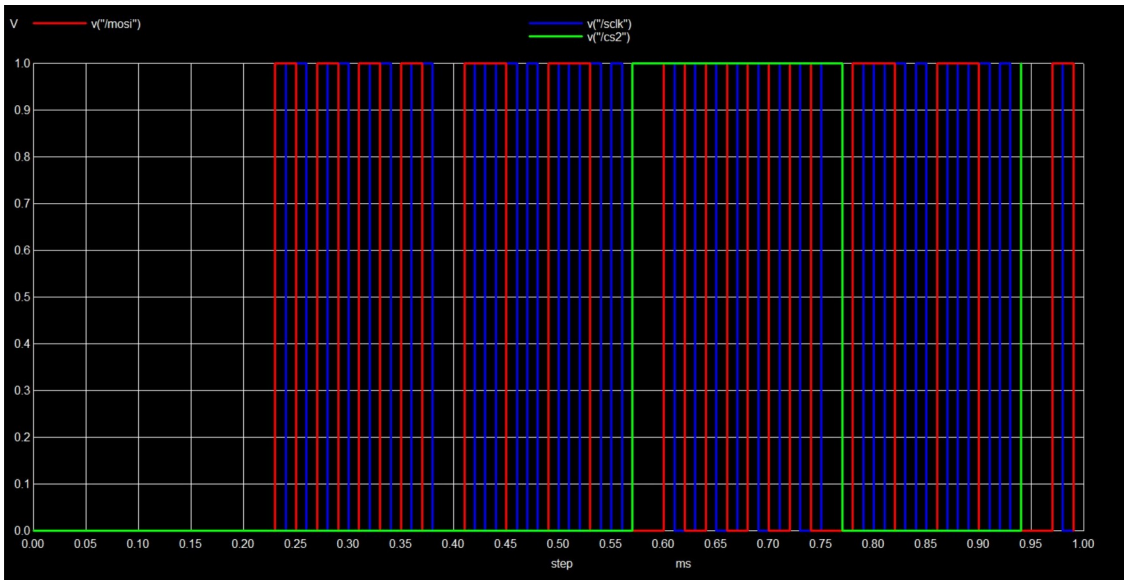

Figure 8: Combined — MOSI (red), SCLK (blue), CS2 (green) showing Slave 2 transaction timing

The following table summarizes the observed simulation results:

Signal	Observed Behaviour
CLK (Input)	100 kHz square wave toggling continuously throughout simulation
RESET (Input)	LOW for first 200us — FSM held in reset. Goes HIGH at 200us — FSM begins operation
SCLK	Toggles synchronously with system CLK throughout both transactions, starts at ~240us

Signal	Observed Behaviour
CS1	Asserts LOW at ~240us, deasserts HIGH at ~400us after 8-bit Slave 1 transaction
MOSI (Slave 1)	Alternating pattern from ~240us to ~400us confirming 10101010 (0xAA) transmission
CS2	Asserts LOW at ~570us, deasserts HIGH at ~760us after 8-bit Slave 2 transaction
MOSI (Slave 2)	Double-pulse pattern from ~400us to ~570us confirming 11001100 (0xCC) transmission

Inference:

The simulation confirms correct sequential multi-slave SPI operation. CS1 and CS2 assert independently with no overlap, demonstrating correct slave arbitration. The CLK signal runs continuously at 100 kHz throughout the simulation window, and RESET correctly holds the FSM inactive for the first 200us. MOSI correctly carries two distinct 8-bit data patterns — an alternating pattern for Slave 1 (10101010) and a double-pulse pattern for Slave 2 (11001100) — verifying that the Master FSM operates as designed and transmits data accurately to each slave in sequence. The MISO line, which carries return data from Slave to Master, requires tri-state driving logic on the shared net for observable output, and remains a scope for further verification.

Conclusion:

The SPI Multi-Slave communication system was successfully designed and simulated in eSim using a Mixed-Signal approach. The simulation verified correct SPI Mode 0 (CPOL=0, CPHA=0) protocol operation — sequential slave selection via independent active-low chip select lines CS1 and CS2, and accurate 8-bit serial data transmission on the MOSI line for both slave transactions. The Master FSM correctly cycles through all seven states (IDLE, LOAD_S1, TRANS_S1, DONE_S1, LOAD_S2, TRANS_S2, DONE_S2), demonstrating reliable multi-slave arbitration and data integrity. This project uses only eSim built-in XSPICE components — specifically adc_bridge_1 (eSim_Hybrid library) and analog pulse voltage sources (eSim_Sources library) — which are native code models compiled into the NgSpice binary and do not require external .lib or .sub files. The SPI Master and Slave modules are implemented using NgVeri, for which the Verilog source files spi_master.v and spi_slave.v serve as the complete source definition in accordance with NgVeri project submission guidelines.

References:

9. FOSSEE eSim User Manual, IIT Bombay — https://esim.fossee.in
10. Ngspice Reference Manual, Mixed-Mode Simulation — http://ngspice.sourceforge.net/docs/ngspice-manual.pdf