Python-Based SPICE Circuit Simulation

Sheng-Lun Huang (黃聖倫)
Electrical Engineering,
National Central University
Taoyuan, ROC
shenglun5738@gmail.com

Chi-Chun Huang (黃啓鈞)
Electrical Engineering,
National Central University
Taoyuan, ROC
jim1743543332@gmail.com

Shiuan-Yi Lee (李軒毅)
Electrical Engineering,
National Central University
Taoyuan, ROC
lee.eason0524@gmail.com

Chun-Wei Chen (陳俊瑋)
Electrical Engineering,
National Central University
Taoyuan, ROC
a0935109965@gmail.com

Abstract*— SPICE is a tool used for simulating circuit behavior, and this project aims at using Python to make a SPICE ourselves. We firstly introduce the usage of SPICE and the motivation to implement it. Then the theories behind the SPICE tool and some numerical methods are illustrated. After the introduction of theories, the syntax and implementation of our SPICE are shown. Finally, the simulation results are presented. In the largest circuit, up to 234 MOSFETs were used in the transient simulation. Compared with ngspice, the average error of the simulation is found to be within 10^{-2} mV, showing perfect agreement.

Keywords—AC/DC/TRAN, BDF2, MNA, Python, Simulation, SPICE, Subcircuit.

I. INTRODUCTION

SPICE is the essential software for simulating and analyzing circuit behavior. It helps engineers design and verify analog and digital circuits, assessing factors like frequency and amplitude response. It detects faults and aids in optimization for specific performance needs. We could say that SPICE is an indispensable tool for modern electronics design.

Underlying SPICE are a number of mathematical theories and numerical techniques. For example, given a circuit *netlist*, we need a systematic way to transform the circuit into equations; to solve the responses of diodes and transistors, we need a root-finding method for system of nonlinear equations; and for transient analysis, numerical solution for ordinary differential equations (ODEs) must be utilized.

In this project, we write a SPICE ourselves in Python, from the very beginning of netlist parsing, to matrix equation stamping and ODE solving. Except the basic Numpy and Scipy libraries for linear algebra, no any third-party script is borrowed in our project. The goal is that, through this manner, we gain a deeper understanding of the principles and techniques in circuit analysis, as well as the various SPICE tools.

The reason for choosing Python for developing SPICE is because its syntax is simple and intuitive, and it provides a wide variety of string methods to implement a parser. Additionally, matrix processing and graphical plotting are very convenient in Python. Its excellent writability allowed us to check simulation results and revise code conveniently. Hence, using Python for SPICE development is an ideal choice.

This report will begin by introducing the theories in circuit analysis, followed by an explanation of netlist syntax, and conclude with some simulation results as validation.

II. METHODS AND THEORIES

A. Circuit Analysis

There are two important laws to form the circuit equations: Kirchhoff's current law (KCL) and Kirchhoff's voltage law (KVL), and the equations can be formed into a linear system to solve the voltages on the nodes and the currents on the branches.

First, consider Fig. 1 below:

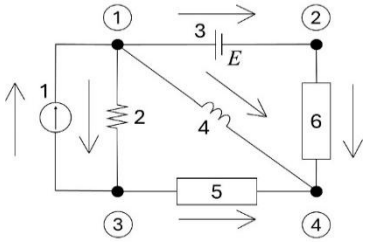


Fig. 1. The interconnection of an example circuit.

The blocks 1~6 represent the circuit elements in Fig. 1, while the arrows beside those blocks are the currents passing through, and the circles denote the node numbers. The KCL equation of each node can be formed as:

Node1: $$-i_1 + i_2 + i_3 + i_4 = 0 \quad (1)$$

Node2: $$-i_3 + i_6 = 0 \quad (2)$$

Node3: $$+i_1 - i_2 + i_5 = 0 \quad (3)$$

Node4: $$-i_4 - i_5 - i_6 = 0 \quad (4)$$

The i_1 to i_6 are decided by the voltage across the elements and the type of the elements. In this section, we only consider linear elements (such as voltage source, resistors, capacitors, inductors). For elements that can be described as: [1, pp. 59]

$$i = g \times v + s_1, \quad (5)$$

where g is the conductance, v the voltage across the elements, and s_1 the independent current source, we call them type-I elements. Examples are resistors, capacitors, and current sources. By (1) ~ (5), we can derive the Nodal Equation (NE) formulation

$$\mathbf{YV} = \mathbf{J}, \quad (6)$$

where **Y**, **V** and **J** in (6) are matrix of the admittance, vector of nodal voltages, and vector of independent

currents flowing into the node, respectively. Elements of **J** are:

$$J_n = \sum i_n^{\text{in}} - \sum i_n^{\text{out}}. \tag{7}$$

Take element 2 in Fig. 1 for example. According to (5), we have:

$$i_2 = g_2 \times (v_{\text{node 1}} - v_{\text{node 3}}). \tag{8}$$

By (6) we can form a part of the NE: [1, pp. 52]

$$\begin{array}{c} \\ \text{node1} \\ \\ \text{node3} \\ \\ \end{array} \begin{array}{c} \begin{array}{cc} \text{node 1} & \text{node 3} \end{array} \\ \begin{bmatrix} \vdots & \vdots \\ \cdots \quad +g_2 & -g_2 \quad \cdots \\ \vdots & \vdots \\ \cdots \quad -g_2 & +g_2 \quad \cdots \\ \vdots & \vdots \end{bmatrix} \end{array} \begin{bmatrix} \\ v_{\text{node 1}} \\ \\ v_{\text{node 3}} \\ \\ \end{bmatrix} \tag{9}$$

By convention, currents flow out of the nodes are taken as positive at the left-hand side of NE.

For the independent current source flowing from node 3 to node 1, we only need to add the value of the current source in the right-hand side (RHS) vector since it is independent of nodal voltages: [1, pp. 53]

$$\boldsymbol{J} = \begin{bmatrix} 0 \\ +I \\ \vdots \\ -I \\ 0 \end{bmatrix} \begin{array}{l} \\ \text{node 1} \\ \\ \text{node 3} \\ \\ \end{array} \tag{10}$$

NE is formed with stamp approach, like (9) and (10). Additionally, it is necessary to choose a node to be the reference node (i.e., ground). Then, the row and column of the reference node in the matrix equation will be eliminated. In Fig. 1, node 4 is taken as ground. The column and row of node 4 will be eliminated, and the stamps of type-I elements will be modified accordingly.

However, for elements described by [1, pp. 59]

$$v = z \times i + s_2, \tag{11}$$

where z, i and s_2 are impedance, branch current, and independent voltage source, respectively, we call them type-II elements, and their branch current cannot be expressed as a linear or differential function of node voltages. Examples are independent voltage sources ($z = 0$), and inductors (in the time domain). Thus, we need to use the modified nodal analysis (MNA) to form the matrix equation.

The concept of MNA is to append the nodal equations with the characteristic equations of the current-controlled (type-II) elements, and its stamp often shows as below: [1, pp. 60]

$$\begin{array}{c} \mathbf{A} \\ \begin{bmatrix} & & \vdots & \\ & \text{G} & \vdots & \text{B} \\ & & \vdots & \\ \cdots & \cdots \quad \cdots & \vdots & \cdots \\ & \text{C} & \vdots & \text{D} \end{bmatrix} \end{array} \begin{array}{c} \mathbf{x} \\ \begin{bmatrix} v_{\text{node1}} \\ v_{\text{node2}} \\ \vdots \\ \cdots \\ i \end{bmatrix} \end{array} = \begin{array}{c} \mathbf{b} \\ \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \cdots \\ \vdots \end{bmatrix} \end{array}, \tag{12}$$

where part G, B, C, D of the matrix A are formed by the nodal equation, branch current of type-II elements flowing in or out of the node, the nodes to which the type-II elements are connected, and the impedance of the type-II elements, respectively. In vector **x**, elements above the dotted line represent node voltages, and those below represent branch currents. In vector **b**, elements above the dotted line are independent current sources, and those below are independent voltage sources.

The stamp of the independent voltage source in Fig. 1 is given below, wherein the RHS vector's element above the E are the **J** in (6). [1, pp. 62]

$$\begin{array}{c} \begin{array}{ccc} \text{node 1} & \text{node 2} & i_{\text{v}} \end{array} \\ \begin{bmatrix} \vdots & \vdots & \vdots \ +1 \\ \vdots & \vdots & \vdots \ -1 \\ \cdots \quad \cdots & \cdots \quad \cdots & \vdots \ \cdots \\ +1 & -1 & \vdots \ 0 \end{bmatrix} \end{array} \begin{bmatrix} v_{\text{node 1}} \\ v_{\text{node 2}} \\ \cdots \\ i_{\text{v}} \end{bmatrix}, \begin{bmatrix} \cdots \\ E \end{bmatrix} \tag{13}$$

Compared with the Nodal Equation, we add a new column to the matrix which corresponds to the current flowing over the voltage source. The newly added row states that the voltage across the source is E.

Another example is the inductor in Fig. 1: [1, pp. 62]

$$\begin{array}{c} \begin{array}{ccc} \text{node 1} & & I_{\text{L}} \end{array} \\ \begin{bmatrix} \vdots & \vdots & \vdots \ +1 \\ \vdots & \vdots & \vdots \ \vdots \\ \cdots \quad \cdots & \cdots \quad \cdots & \vdots \ \cdots \\ +1 & & \vdots \ -Z_{\text{L}} \end{bmatrix} \end{array} \begin{bmatrix} v_{\text{node 1}} \\ \vdots \\ \cdots \\ I_{\text{L}} \end{bmatrix}, \begin{bmatrix} \cdots \\ 0 \end{bmatrix}, \tag{14}$$

where the last row is the inductor's characteristic equation. Compared with (13), we don't fill −1 in the matrix, since we have picked node 4 as reference.

After filling the stamp of type II elements, we can finally solve

$$\mathbf{Ax} = \mathbf{b}, \tag{15}$$

where **A** is the stamp matrix, **x** is the vector containing node voltages and branch currents to be solved, and **b** is the vector composed of **J** in (6) and the stamp of type-II elements.

B. Analysis Mode

We have three analysis modes in our SPICE: AC, DC, and transient analysis. This section introduces the algorithm and principle of these analysis methods.

(i). AC Analysis

The user specifies the frequency's range and step in the netlist. The stamps of all elements presented in the netlist can be filled into matrix according to frequency. Frequency dependent elements are inductors and capacitors. Inductors are type-II elements, and its stamp is (14) with Z_{L} equal to $j\omega L$. Capacitors are type-I elements, and its stamp is in the form of (9) with the conductance g equal to $j\omega C$. For frequency-independent elements, we may directly fill in their stamps outside the for-loop for ω. If the elements are nonlinear, we need to run the DC bias (to be discussed in the next section) before starting the AC analysis, to determine their equivalent linearized model. Fig. 2 is the flow chart of AC Analysis:

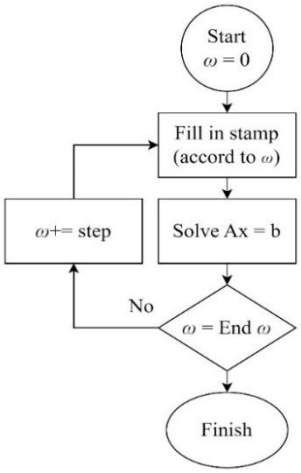


Fig. 2. Flow chart of AC analysis.

Since we have complex numbers in the **A** matrix, the solution **x** may also include imaginary numbers. For each frequency, we store the complex quantities; after the whole solution process, we can plot the curve of the magnitudes against frequency. This is known as AC sweep.

(ii). DC Analysis

DC Analysis is not as simple as AC analysis; it requires iteration to find the solution for the nonlinear elements. Therefore, it will take more steps to find the real solution compared to AC analysis. In DC analysis, frequency-dependent elements are treated as short or open circuits.

Newton method is applied to solve DC. To introduce this method, firstly consider the question of finding the solution to $f(x) = 0$. In the Newton method, nonlinear functions are approximated at each iteration point by the first two terms of the Taylor series expansion: [1, pp. 196]

$$f(x) \approx f(x^{(k)}) + \left[\frac{\partial f}{\partial x}\right]_{x=x^{(k)}} (x - x^{(k)}). \tag{16}$$

And the linear system we are going to solve is

$$J^{(k)}(x - x^{(k)}) = -f(x^{(k)}), \tag{17}$$

where J is the Jacobian matrix given by

$$J_{ij}^{(k)} = \left[\frac{\partial f_i}{\partial x_j}\right]_{x=x^{(k)}}. \tag{18}$$

After solving (17), $x^{(k+1)}$ is obtained and compared to $x^{(k)}$ to finding its error. If the error does not meet convergence condition, we set the new $x^{(k)}$ equal to $x^{(k+1)}$ and solve for the new $x^{(k+1)}$ until the solution converges.

For one-dimensional case, Newton method can be illustrated as in Fig. 3. Suppose x_1 is initial guess. We draw the tangent line on (x_1, y_1), and use it to find the next guess, which is the point where the tangent line intersects the $y = 0$ axis.

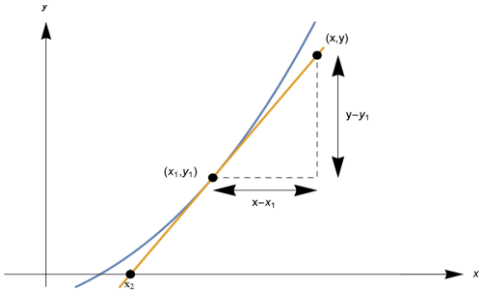


Fig. 3. Instance of the Newton iteration (figure taken from [5]).

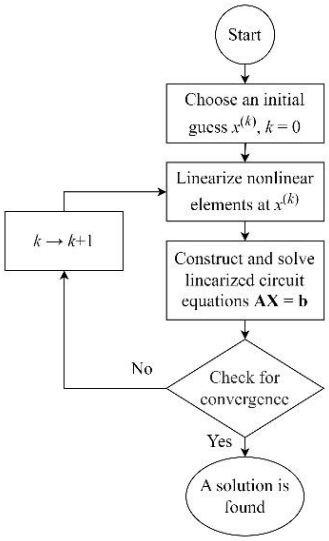


Fig. 4. Flow chart of DC analysis.

After each iteration, the difference between the former guess and the newer guess will be checked. If the difference is small enough, then we regard the newer guess as the solution and quit the iteration. If the function has two or more variables, it is necessary to check the convergence of all variables.

There are two modes of DC analysis in our SPICE:
(1) DC bias: DC bias is to determine the initial values of all node voltages in the circuit, enabling the calculation of the equivalent resistance of nonlinear elements and the voltages across capacitors, which are crucial for transient analysis. Without performing DC bias prior to transient analysis, the results would be inaccurate and convergence to a solution would be difficult. To solve DC bias, we adopt a method known as source-stepping, wherein all DC sources increase from zero to their nominal values incrementally, with a step of 1% of their nominal voltages/currents. After solving the equation in the first step, we take the result of the first step as the new initial guess to calculate the value of the next step. This process is continued until the nominal value of all sources is reached. This makes the iteration converges easier since there is no significant change to all the elements' values.
(2) DC sweep: The voltage of one object source is changed to observe the I-V characteristic of the circuit. It is similar to source-stepping; however, the start

voltage, end voltage, and voltage step are specified by the user in the netlist.

The flow chart of DC analysis is shown in Fig. 4.

(iii). Transient Analysis

Transient Analysis, or time-domain analysis, of electronics is the most widely applied analysis in circuit simulation. It is used to observe the behavior of the circuit over time with given initial values. Fig. 5 is the flow chart of transient analysis.

For certain elements like capacitors and inductors, their characteristic functions include a differential term like (19) and (20), respectively.

$$C\frac{\mathrm{d}v}{\mathrm{d}t} = i_{\mathrm{C}} \tag{19}$$

$$\frac{d\phi}{dt} = L\frac{\mathrm{d}i}{\mathrm{d}t} = v_{\mathrm{L}} \tag{20}$$

In general, the equation in the form of (19) and (20) can be described as

$$u'(t) = f(u), \tag{21}$$

where $u'(t)$ is the derivative of $u(t)$.

It is impractical to solve these equations by directly integrating them in programming. Hence, we need to use integration formulas for those terms. The numerical methods we applied are the Backward Euler method (BE) and the 2-step Backward Differentiation Formula (BDF2) method. The BE method [2, pp. 120] can be described as:

$$U^{n+1} = U^n + kf(U^{n+1}), \tag{22}$$

where k is the time step, U^n the value of the last time step, $f(U^{n+1})$ the tangent line's slope at U^{n+1} , and U^{n+1} is the value that we are going to find.

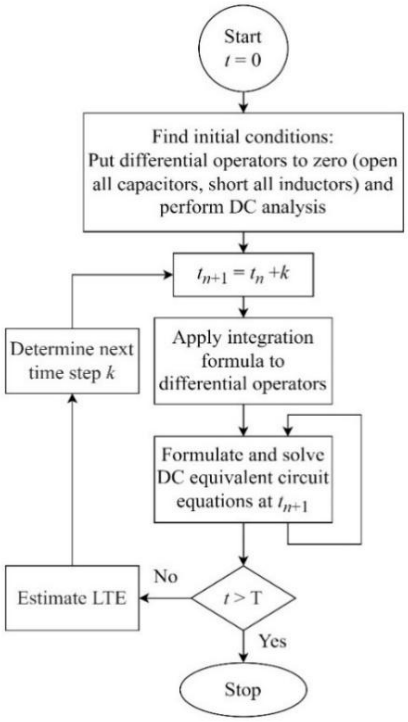


Fig. 5. Flow chart of transient analysis. T is simulation end time.

The BDF2 [2, pp. 173] can be described as

$$3U^{n+2} - 4U^{n+1} + U^n = 2kf(U^{n+2}), \tag{23}$$

where U^{n+2} is the value we are going to find, U^{n+1} the value solved from last time step, U^n is the value solved from the time before U^{n+1}, k the time step, and $f(U^{n+2})$ is the tangent line's slope at U^{n+2}.

A notable distinction between these two formulas is the local truncation error (LTE), which can be obtained from (22) and (23) with "U" substituted by "u", the true solution of the ODE. Equation (24) is an example of the BE's LTE [2, pp. 17]

$$\tau_{\mathrm{BE}} = \frac{(u^{n+1}-u^n)}{k} - f(u^{n+1}), \tag{24}$$

where $f(u^{n+1})$ is the derivative of $u(t)$ at $t = t_{n+1}$.

Using the method of Taylor expansion, we can demonstrate that the LTE of the BE method and BDF2 are (25) and (26):

$$\begin{aligned}\tau_{\mathrm{BE}} &= \frac{(u^{n+1}-u^n)}{k} - u_{n+1}^{(1)} \\ &= \frac{u^{n+1} - \sum_{i=0}^{\infty}\frac{(-k)^i}{i!}u_{n+1}^{(i)}}{k} - u_{n+1}^{(1)} \\ &= \frac{1}{2}ku_{n+1}^{(2)} - O(k^2)\end{aligned} \tag{25}$$

$$\begin{aligned}\tau_{\mathrm{BDF2}} &= \frac{(3u^{n+2}-4u^{n+1}+u^n)}{2k} - u_{n+2}^{(1)} \\ &= \frac{3u^{n+2} - 4\sum_{i=0}^{\infty}\frac{(-k)^i}{i!}u_{n+2}^{(i)} + \sum_{i=0}^{\infty}\frac{(-2k)^i}{i!}u_{n+2}^{(i)}}{k} - u_{n+2}^{(1)} \\ &= \frac{1}{3}k^2u_{n+2}^{(3)} - O(k^3),\end{aligned} \tag{26}$$

where $u_{n+2}^{(i)}$ represents the i-th derivative at t_{n+2}, which means $u_{n+2}^{(1)} = f(u^{n+2})$, and we have LTE of BE and BDF2 are $O(k)$ and $O(k^2)$, respectively. This indicates that they are first-order and second-order accurate. In our SPICE implementation, we check the LTE after the calculation of each time step.

However, since we cannot predict the exact solution $u(t)$ to the circuit, we can't utilize method (24) to obtain the exact LTE of BE. Instead, a practical method is to apply two numerical methods with different order of accuracy to estimate the LTE, using one as the numeric solution U^{n+1} and the other as the approximated true solution u^{n+1}. As mentioned earlier, the BDF2 is second-order accurate, so we take its solution as the approximated exact solution to calculate the LTE of the BE method. Specifically, we first find the one-step error of BE by [2, pp. 122]

$$\begin{aligned}\mathcal{L}_{\mathrm{BE}}^{\mathrm{n}} &= u^{n+1} - U_{\mathrm{BE}}^{n+1} \approx U_{\mathrm{BDF2}}^{n+1} - U_{\mathrm{BE}}^{n+1} \\ &= U_{\mathrm{BDF2}}^{n+1} - U_{\mathrm{BE}}^{n} - kf(U_{\mathrm{BE}}^{n+1}).\end{aligned} \tag{27}$$

If we divide the one-step error by k, we obtain the estimated LTE. If the LTE exceeds our restriction, the step of transient simulation will be reduced to 10 percent of the current step.

C. Processing of Nonlinear Components

For nonlinear elements, we need to consider their I-V characteristic curve first and decide which model should be used to analyze them. We have two kinds of nonlinear elements in our SPICE: diodes and MOSFETs.

(i). Diodes

The I-V characteristic curve is

$$i_\mathrm{d} = g(v_\mathrm{d}) = I_\mathrm{s}(e^{v_\mathrm{d}/\eta V_\mathrm{T}} - 1), \tag{28}$$

where i_d is the current flowing through the diode, I_S the saturation current, η is an ideality factor (normally between 1 and 2), v_d the voltage across diodes, and V_T the thermal voltage, which can be described by

$$V_\mathrm{T} = \frac{kT}{q}, \tag{29}$$

where T is the absolute temperature, k is the Boltzmann's constant, and q is the electron charge.

From equation (28), the relationship between v_d and i_d has been established, and there are no differentiation or integration terms in the characteristic equation. It can be seen that when v_d is in reverse bias, the current flowing through it will be very small; the diode appears to have high impedance. Conversely, when v_d is in forward bias, the current flowing through it will be very large, as if it were short circuit. As the behavior of the diode resembles that of a resistor whose resistance is controlled by the voltage, it can be regarded as a voltage-controlled resistor.

Though the relationship between current and voltage is established, solving the nonlinear function directly in the program is impractical. Therefore, the equation must be solved by iteration. The Newton method is applied, which has been explained in section II-(B)-(ii). Equation (28) can be linearized at $v_\mathrm{d}^{(k)}$, the voltage across the diode derived from the last solution $v^{(k)}$, by Taylor expansion as [1, pp. 206]

$$\begin{aligned} i &= g\left(v_\mathrm{d}^{(k)}\right) + \left[\frac{dg}{dv_\mathrm{d}}\right]_{v_\mathrm{d}=v_\mathrm{d}^{(k)}} \left(v_\mathrm{d} - v_\mathrm{d}^{(k)}\right) \\ &= a^{(k)} v_\mathrm{d} + g\left(v_\mathrm{d}^{(k)}\right) - a^{(k)} v_\mathrm{d}^{(k)} \\ &= a^{(k)} v_\mathrm{d} + b^{(k)} \end{aligned} \tag{30}$$

where $a^{(k)}$ and $b^{(k)}$ are given by [1, pp. 220]

$$a^{(k)} = \left.\frac{\mathrm{d}g(v_\mathrm{d})}{\mathrm{d}v_\mathrm{d}}\right|_{v_\mathrm{d}=v_\mathrm{d}^{(k)}} = \frac{I_s}{\eta V_\mathrm{T}} e^{v_\mathrm{d}^{(k)}/\eta V_\mathrm{T}} \tag{31}$$

$$b^{(k)} = g\left(v_\mathrm{d}^{(k)}\right) - a^{(k)} v_\mathrm{d}^{(k)}. \tag{32}$$

Based on (30), we can say $a^{(k)}$, the derivative of the I-V equation of diode, is the equivalent conductance of diode, and $b^{(k)}$ represents an independent current source in parallel with the conductance. The default values in our SPICE are $I_\mathrm{S} = 10^{-14}$ A, $V_\mathrm{T} = 0.026$ V, and $\eta = 1$.

Suppose the anode and cathode of a diode are connected to node 1 and node 2, respectively. We can fill the stamp of a diode as [1, pp. 207]

$$\begin{array}{c} \\ \text{node 1} \\ \text{node 2} \end{array} \begin{array}{c} \begin{array}{cccc} \text{node 1} & & \text{node 2} & \end{array} \\ \begin{bmatrix} \cdots & +a^{(k)} & \cdots & -a^{(k)} & \cdots \\ \cdots & -a^{(k)} & \cdots & +a^{(k)} & \cdots \end{bmatrix} \end{array} \begin{array}{c} \mathbf{X} \\ \begin{bmatrix} v_{\text{node 1}} \\ v_{\text{node 2}} \end{bmatrix} \end{array} = \begin{array}{c} \text{rhs} \\ \begin{bmatrix} -b^{(k)} \\ +b^{(k)} \end{bmatrix} \end{array}, \tag{33}$$

where elements of the **X** vector represent the solution of the current iteration but not the final solution, and we denote them as $v^{(k+1)}$.

-(ii). MOSFETs

MOSFETs have three different I-V characteristic curves for different operation regions

Cut-off region
$$I_\mathrm{DS} = I_\mathrm{D0}\left(1 - e^{\frac{-V_\mathrm{DS}}{V_\mathrm{T}}}\right) e^{\frac{V_\mathrm{ov}}{\eta V_\mathrm{T}}} \tag{34}$$

Linear region
$$I_\mathrm{DS} = k_n\left[V_\mathrm{ov}V_\mathrm{DS} - \frac{V_\mathrm{DS}^2}{2}\right] \tag{35}$$

Saturation region
$$I_\mathrm{DS} = \frac{1}{2}k_n V_\mathrm{ov}^2[1 + \lambda V_\mathrm{DS}] \tag{36}$$

and

$$V_\mathrm{ov} = V_\mathrm{GS} - V_\mathrm{T} \tag{37}$$

Since (34) ~ (36) are functions of both V_DS and V_GS, we could regard MOSFETs as three-terminal voltage-controlled resistors, which can be describe as Fig. 6:

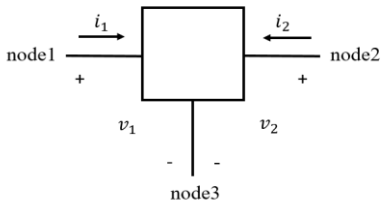


Fig. 6. Three-terminal voltage-controlled resistor [1, pp. 210].

The relationship between v_1, v_2 , i_1, i_2 in Fig. 6 are

$$i_1 = f_1(v_1, v_2) \tag{38}$$

$$i_2 = f_2(v_1, v_2) \tag{39}$$

Since both f_1 and f_2 are nonlinear functions, we need to use iteration to solve their solutions. Taylor expansion of the two functions to first order gives

$$i_1 = a_{11}^{(k)} v_1 + a_{12}^{(k)} v_2 + b_1^{(k)} \tag{40}$$

$$i_2 = a_{21}^{(k)} v_1 + a_{22}^{(k)} v_2 + b_2^{(k)} \tag{41}$$

where

$$b_1^{(k)} = f_1\left(v_1^{(k)}, v_2^{(k)}\right) - \left(a_{11}^{(k)} v_1^{(k)} + a_{12}^{(k)} v_2^{(k)}\right) \tag{42}$$

$$b_2^{(k)} = f_2\left(v_1^{(k)}, v_2^{(k)}\right) - \left(a_{21}^{(k)} v_1^{(k)} + a_{22}^{(k)} v_2^{(k)}\right) \tag{43-a}$$

and

$$\begin{aligned} a_{11}^{(k)} &= \frac{\partial f_1}{\partial v_1}, \quad a_{12}^{(k)} = \frac{\partial f_1}{\partial v_2} \\ a_{21}^{(k)} &= \frac{\partial f_2}{\partial v_1}, \quad a_{22}^{(k)} = \frac{\partial f_2}{\partial v_2}. \end{aligned} \tag{43-b}$$

Our object is to solve the v_1 and v_2 by iteration, and the stamp of it will become [1, pp. 211]

$$\begin{matrix}\text{node 1} & \text{node 2} & \text{node 3}\end{matrix}$$
$$\begin{bmatrix} +a_{11}^{(k)} & +a_{12}^{(k)} & -(a_{11}^{(k)}+a_{12}^{(k)}) \\ +a_{21}^{(k)} & +a_{22}^{(k)} & -(a_{21}^{(k)}+a_{22}^{(k)}) \\ -(a_{11}^{(k)}+a_{21}^{(k)}) & -(a_{12}^{(k)}+a_{22}^{(k)}) & a_{33}^{(k)} \end{bmatrix} \begin{bmatrix} v_{\text{node 1}} \\ v_{\text{node 2}} \\ v_{\text{node 3}} \end{bmatrix}$$
$$\text{rhs}$$
$$\begin{bmatrix} -b_1^{(k)} \\ -b_2^{(k)} \\ +b_1^{(k)} + b_2^{(k)} \end{bmatrix} \tag{44}$$

In (44), $a_{33}^{(k)} = (a_{11}^{(k)}+a_{21}^{(k)}+a_{12}^{(k)}+a_{22}^{(k)})$, where $a_{11}^{(k)}$ and $a_{12}^{(k)}$ represent the variations of i_1 relative to v_1 and v_2, respectively, and $a_{21}^{(k)}$ and $a_{22}^{(k)}$ represent the variations of i_2 relative to v_1 and v_2, respectively. We can find that all currents are flowing into node 3 in Fig. 6; therefore all of the element in row 1 and 2 are added into the row 3 with negative sign.

The case above is general three-terminal voltage-controlled resistor, but for MOSFETs, it can be simplified. We set node 1 to be the gate of the MOSFET, node 2 as drain, and node 3 the source. Since we know there is no current flowing into the gate, i_1 in (38) is always zero, Therefore, we only have to consider i_2 and f_2, which is the current function of MOSFETs (34~36). Hence, the stamp of MOSFETs can be simplified as [1, pp. 225]

$$\begin{matrix}\text{G} & \text{D} & \text{S} & \text{rhs}\end{matrix}$$
$$\begin{bmatrix} 0 & 0 & 0 \\ +a_{11}^{(k)} & +a_{12}^{(k)} & -(a_{11}^{(k)}+a_{12}^{(k)}) \\ -a_{11}^{(k)} & -a_{12}^{(k)} & (a_{11}^{(k)}+a_{12}^{(k)}) \end{bmatrix} \begin{bmatrix} V_{\text{G}} \\ V_{\text{D}} \\ V_{\text{S}} \end{bmatrix}, \begin{bmatrix} 0 \\ -b_1^{(k)} \\ +b_1^{(k)} \end{bmatrix} \tag{45}$$

In different operation region, we have different $a_{11}^{(k)}$, $a_{21}^{(k)}$, and $b_1^{(k)}$. For cut-off region:

$$\begin{aligned} a_{11}^{(k)} &= \frac{I_{D0}}{\eta V_T} e^{V_{\text{ov}}^{(k)}/\eta V_{\text{T}}},\ a_{12}^{(k)} = 0, \\ b_1^{(k)} &= I_{D0} e^{V_{\text{ov}}^{(k)}/\eta V_{\text{T}}} - a_{11}^{(k)} V_{\text{GS}}^{(k)}, \end{aligned} \tag{46}$$

where I_{D0} is the drift current, which has default value 10^{-14} A in our SPICE. In linear region:

$$a_{11}^{(k)} = k_n V_{\text{DS}}^{(k)},\ a_{12}^{(k)} = k_n (V_{\text{ov}}^{(k)} - V_{\text{DS}}^{(k)}) \tag{47}$$

$$b_1^{(k)} = k_n \left[V_{\text{ov}}^{(k)} V_{\text{DS}}^{(k)} - \frac{V_{\text{DS}}^{(k)2}}{2} \right] - a_{11}^{(k)} V_{\text{GS}}^{(k)} - a_{12}^{(k)} V_{\text{DS}}^{(k)}. \tag{48}$$

At saturation region:

$$a_{11}^{(k)} = \frac{1}{2} k_n \{ 2V_{\text{ov}}^{(k)} \left[1 + \lambda \left(V_{\text{DS}}^{(k)} - V_{\text{ov}}^{(k)} \right) \right] - \lambda \left(V_{\text{ov}}^{(k)} \right)^2 \} \tag{49}$$

$$a_{12}^{(k)} = \frac{1}{2} k_n \lambda \left(V_{\text{ov}}^{(k)} \right)^2 \tag{50}$$

$$b_1^{(k)} = \frac{1}{2} k_n \left(V_{\text{ov}}^{(k)} \right)^2 \left[1 + \lambda \left(V_{\text{DS}}^{(k)} - V_{\text{ov}}^{(k)} \right) \right] - a_{11}^{(k)} V_{\text{GS}}^{(k)} - a_{12}^{(k)} V_{\text{DS}}^{(k)}, \tag{51}$$

where λ is a process technology parameter that is inversely proportional to the length selected for the channel. We choose $k_n = 2.072\text{e}{-}05$ A/V^2, and $\lambda = 0.02$ V^{-1}. Since MOSFETs are nonlinear elements, we have to check the convergence of V_{GS} and V_{DS} after each iteration until they meet our convergence conditions.

D . Elements Involving Time-derivative

(i). Capacitors

The characteristic equation of a capacitor is

$$q_{\text{C}} = C \times v_{\text{C}} = f(v_{\text{c}}), \qquad i_{\text{C}} = \frac{dq_{\text{C}}}{dt} \tag{52}$$

where C is the value of the capacitor, v_{C} the voltage across capacitor, and q_{C} the charge stored in capacitor.

Then we can apply the BE formula to it

$$\frac{dq_{\text{C}}}{dt} = i_{\text{C}}^{n+1} \approx \frac{{q_{\text{C}}}^{n+1} - {q_{\text{C}}}^{n}}{k} = C \times \frac{{v_{\text{C}}}^{n+1} - {v_{\text{C}}}^{n}}{k}, \tag{53}$$

where v_{C}^{n} and v_{C}^{n+1} represent the voltage across capacitors at time $t = t_n$ and t_{n+1}, respectively. Therefore, we must store the voltage across each capacitor after the calculation of each time step. Then, we apply the stamp of the BE formula on capacitors as follows: [1, pp. 291]

$$\begin{matrix} & i & & j & \\ & \vdots & & \vdots & \\ \cdots & +\frac{C}{k} & \cdots & -\frac{C}{k} & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\frac{C}{k} & \cdots & +\frac{C}{k} & \cdots \\ & \vdots & & \vdots & \end{matrix} \begin{bmatrix} \\ v_i \\ \\ v_j \\ \\ \end{bmatrix}, \begin{matrix}\text{rhs} \\ \\ \begin{bmatrix} +\frac{Cv_{\text{C}}^{n}}{k} \\ \\ -\frac{Cv_{\text{C}}^{n}}{k} \end{bmatrix} \end{matrix}, \tag{54}$$

where the capacitor is connected between nodes i and j. As mentioned in section II-B-iii, we must apply two integration formulas to calculate the LTE in each time step. Therefore, we apply BDF2 on capacitors. Its equation can be described as:

$$\begin{aligned} \frac{dq_{\text{C}}}{dt} = i_{\text{C}}^{n+1} &\approx \frac{3{q_{\text{C}}}^{n+1} - 4{q_{\text{C}}}^{n} + {q_{\text{C}}}^{n-1}}{2k} \\ &= C \times \left(\frac{3{v_{\text{C}}}^{n+1}}{2k} - \frac{2{v_{\text{C}}}^{n}}{k} + \frac{{v_{\text{C}}}^{n-1}}{2k} \right) \end{aligned} \tag{55}$$

where ${v_{\text{C}}}^{n}$, v_{C}^{n+1} and v_{C}^{n-1} are the voltage across capacitors at time $t = t_n$, t_{n+1} and t_{n-1}, respectively. We need the two former calculated voltages to calculate the result in BDF2. Therefore, the stamp of the BDF2 formula applying on capacitors is: [1, pp. 292]

$$\begin{matrix} & i & & j & \\ & \vdots & & \vdots & \\ \cdots & +\frac{3C}{2k} & \cdots & -\frac{3C}{2k} & \cdots \\ & \vdots & & \vdots & \\ \cdots & -\frac{3C}{2k} & \cdots & +\frac{3C}{2k} & \cdots \\ & \vdots & & \vdots & \end{matrix} \begin{bmatrix} \\ v_i \\ \\ v_j \\ \\ \end{bmatrix}$$
$$\text{rhs}$$
$$\begin{bmatrix} +\frac{2Cv_{\text{C}}^{n}}{k} - \frac{Cv_{\text{C}}^{n-1}}{2k} \\ \\ -\frac{2Cv_{\text{C}}^{n}}{k} + \frac{Cv_{\text{C}}^{n-1}}{2k} \end{bmatrix} \tag{56}$$

The above are how the stamp of capacitors is filled in transient analysis.

(ii). Inductors

The characteristic equation of an inductor is

$$\varphi_{\mathrm{L}} = L \times i_{\mathrm{L}} = f(i_{\mathrm{L}}), v_{\mathrm{L}} = \frac{d\varphi_{\mathrm{L}}}{dt}, \tag{57}$$

where L is the value of the inductance, φ_{L} the magnetic flux of inductors, and i_{L} is the current flowing through inductor.

Then we can apply the BE formula to it

$$\frac{d\varphi_{\mathrm{L}}}{dt} = v_{\mathrm{L}}^{n+1} \approx \frac{{\varphi_{\mathrm{L}}}^{n+1} - {\varphi_{\mathrm{L}}}^{n}}{k} = L \times \frac{{i_{\mathrm{L}}}^{n+1} - {i_{\mathrm{L}}}^{n}}{k}, \tag{58}$$

where i_L^n and i_L^{n+1} represent currents flowing through inductors at time $t = t_n$ and t_{n+1}, respectively. i_L^{n+1} is our objective to find. The stamp of the BE formula applied on inductors is shown as follows: [1, pp. 296]

$$\begin{array}{c} \begin{matrix} i & \quad & j & \quad & i_{\mathrm{L}} \end{matrix} \\ \begin{bmatrix} \vdots & & \vdots & \\ \cdots & & \cdots & +1 \\ \cdots & & \cdots & -1 \\ +1 & & -1 & -\frac{L}{k} \end{bmatrix} \begin{bmatrix} v_i \\ v_j \\ i_{\mathrm{L}} \end{bmatrix} = \begin{bmatrix} \\ \\ -\frac{L}{k} i_{\mathrm{L}}^{n} \end{bmatrix} \end{array} \quad \text{rhs} \tag{59}$$

where the inductor is connected between node i and j. The BDF2 equation of inductors can be described as:

$$\begin{aligned} \frac{d\varphi_{\mathrm{L}}}{dt} = v_{\mathrm{L}}^{n+1} &\approx \frac{3{\varphi_{\mathrm{L}}}^{n+1} - 4{\varphi_{\mathrm{L}}}^{n} + {\varphi_{\mathrm{L}}}^{n-1}}{2k} \\ &= L \times \left(\frac{3{i_{\mathrm{L}}}^{n+1}}{2k} - \frac{2{i_{\mathrm{L}}}^{n}}{k} + \frac{{i_{\mathrm{L}}}^{n-1}}{2k}\right), \end{aligned} \tag{60}$$

where ${i_{\mathrm{L}}}^{n}$, ${i_{\mathrm{L}}}^{n+1}$ and ${i_{\mathrm{L}}}^{n-1}$ are currents flowing through inductors at time $t = t_n$, t_{n+1} and t_{n-1}, respectively. The stamp of the BDF2 formula applying on inductors can be described as below: [1, pp. 297]

$$\begin{array}{c} \begin{matrix} i & \quad & j & \quad & i_{\mathrm{L}} \end{matrix} \\ \begin{bmatrix} \vdots & & \vdots & \\ \cdots & & \cdots & +1 \\ \cdots & & \cdots & -1 \\ +1 & & -1 & -\frac{3L}{2k} \end{bmatrix} \begin{bmatrix} v_i \\ v_j \\ i_{\mathrm{L}} \end{bmatrix} = \begin{bmatrix} \\ \\ -\frac{2L}{k} i_{\mathrm{L}}^{n} - \frac{L}{2k} i_{\mathrm{L}}^{n-1} \end{bmatrix} \end{array} \quad \text{rhs} \tag{61}$$

The above are how we fill in the stamp of inductors in transient analysis.

III. SPICE OPERATION

A. SPICE Syntax

(i). SPICE File Types

SPICE simulation files mostly have extensions ".cir" or ".sp", and our software also accepts files with the ".txt" extension for the same purpose. These files contain circuit descriptions, model definitions, simulation commands, and other related information.

(ii). Syntax for Components

Syntax for each component is as follows:

```
* Resistor
R<name> <node 1> <node 2> <value>
* Capacitor
C<name> <node 1> <node 2> <value>
* Inductor
L<name> <node 1> <node 2> <value>
* Diode
D<name> <node 1> <node 2> <modelname>
* MOSFET
M<name> <drain> <gate> <source> <body> <modelname>
w=<width> l=<length>
```

- `<name>`: The name or identifier of the component.
- `<node1>` and `<node2>`: The nodes to which the component is connected.
- `<value>`: The resistance, capacitance, or inductance value, respectively.
- `<modelname>`: The model of diode or MOSFET.
- `<drain>`, `<gate>`, `<source>` and `<body>`: The drain, gate, source, and body terminals of the MOSFET, respectively.
- `<width>` and `<length>`: The width and the length of the MOSFET channel.

There are three main types of voltage sources: DC, Pulse, and Sinusoidal.

```
* DC
V<name> <node 1> <node 2> dc <value>
* Pulse
V<name> <node 1> <node 2> pulse <VL> <VH> <delay>
<rise> <fall> <width> <period>
* Sinusoidal
V<name> <node 1> <node 2> sin <offset> <amp> <freq>
<delay>
```

- `<VL>` and `<VH>`: The low voltage level and high level of the pulse.
- `<delay>`: The delay time before the pulse starts.
- `<rise>` and `<fall>`: The rise time and fall time of the pulse.
- `<width>`: The width of the pulse.
- `<period>`: The period of the pulse.
- `<offset>`: The DC offset of the sinusoidal signal.
- `<amp>`: The amplitude of the sinusoidal waveform.
- `<freq>`: The frequency of the sinusoidal signal.

B. SPICE Parser

(i). Expand Parser with Subcircuit (Subckt)

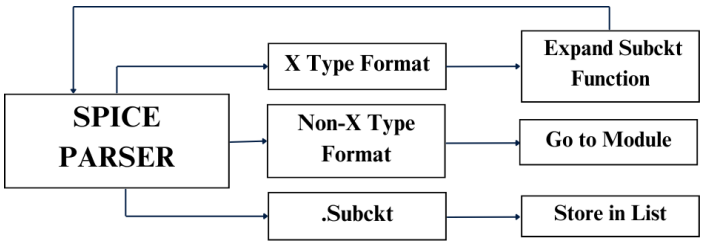


Fig. 7. Flow chart of SPICE parser.

The execution of our Python-based parser can be divided into three parts. Firstly, components starting with "X" represent subcircuit calls. When encountering components starting with "X", the parser identifies the subcircuit by finding the last element in that line, then

expands this subcircuit using the created library, and continues parsing the subcircuit using the basic parser.

The second part deals with non-subcircuit components (components not starting with "X"), such as RLC. When encountering such components, the parser analyzes the properties of the circuit by referring to the corresponding module.

The third part involves subcircuit definitions, indicated by lines starting with ".subckt". Upon encountering them, the parser gathers all the components seen until ".ends" into a list, facilitating their use as a library later on.

(ii). Subckt Node Name Substitution Rule

Syntax for each subckt and use is as follows:

```
* Subckt
.subckt <subckt name> <node 1> <node 2> <node 3>…
R<name> <node 1> <node 2> <value>
L<name> <node 2> <int_node 1> <value>
C<name> <int_node 1> <node 3> <value>
.ends
* Use Subckt
X<name> <node 1> <node 2> <node 3> <subckt name>
```

- <node 1> and <node 2>…: The external nodes to which the component is connected.

For the expansion of subcircuit nodes, we handle internal nodes and external nodes separately.

External nodes are the nodes that appear at the .subckt line ('node 1', 'node 2', 'node 3'…) , and also the nodes of the "X" component. The node names at the .subckt line and at the "X" component line can be different, because we just substitute the name by its location. Ultimately, during expansion, the nodes are named according to the nodes of the "X" component.

For internal nodes like "int_node 1", we append an underscore followed by the name after "X" to differentiate internal nodes (int_node 1_name). By doing this operation, we can avoid duplicate naming the nodes when having different calls. This prevents calculation errors that might occur when calling the same component with identical node names.

After distinguishing whether the node is an external or internal node, the parser starts expanding the subcircuit by calling the library using the subckt's name. This enables us to start reading the subckt netlist without node naming error.

(iii). Read the Expansion Subckt

After parsing the subcircuit function expansion, the expanded netlist is used for our SPICE, matching corresponding values or nodes by classifying different components according to their properties.

Following the SPICE syntax, a consistent format is employed to determine the attributes of each component, which has been introduced in section III-(A)-(ii). For instance, RLC components share the same attributes: name, two nodes, and finally the value. Subsequently, we store each attribute in the lists we created, such as "valuelist" or "nodelist", and utilize them when employing MNA for calculations.

C. Command Setting

In SPICE simulations, the AC, DC, and TRAN sweep commands are essential for analyzing circuit behavior under different conditions.

(i). AC Sweep (AC)

The AC sweep command allows for the analysis of circuit response to varying frequency inputs.

Syntax:

```
* AC sweep
ac <variation type> <number of points> <start
frequency> <stop frequency>
```

- <variation type>: The type of frequency variation (decade, linear, or octave).
- <number of points>: The number of points or steps in the frequency range.
- <start frequency>: The starting frequency of the sweeping.
- <stop frequency>: The ending frequency of the sweeping.

(ii). DC Sweep (DC)

The DC sweep command is used to sweep a voltage or current source over a range of values.

Syntax:

```
* DC sweep
dc <source> <start value> <stop value> <increment>
```

- <source>: The name of the object source.
- <start value>: The starting value of object source.
- <stop value>: The ending value of object source.
- <increment>: The increment of object source.

(iii). Transient Analysis (TRAN)

The TRAN sweep command performs transient analysis, simulating the behavior of a circuit over a specified time interval.

Syntax:

```
* Transient Analysis
tran <step size> <stop time>
```

- <step size>: The time step for the simulation.
- <stop time>: The total simulation time.

IV. SIMULATED CIRCUIT

Below are the circuit test results under different simulation commands for our SPICE.

A. AC Simulation

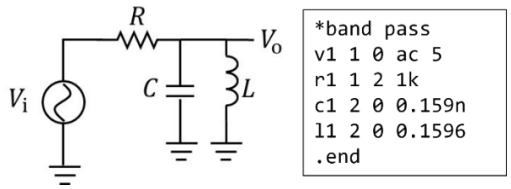


Fig. 8. Bandpass filter circuit and its netlist.

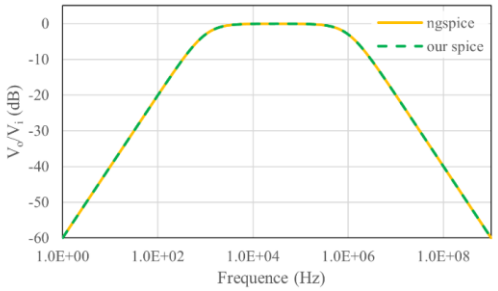


Fig. 9. Bandpass filter V_o/V_i in dB (our SPICE overlap ngspice).

In AC simulation, we use the bandpass filter circuit in Fig. 8, where R = 1 kΩ, C = 159 nF, L = 1.596 kH. In Fig. 9, we can see that this circuit has passband 10^3~10^6 Hz. The poles are at 10^3 and 10^6 s^{-1}. Our results agree well with ngspice.

B. DC Simulation

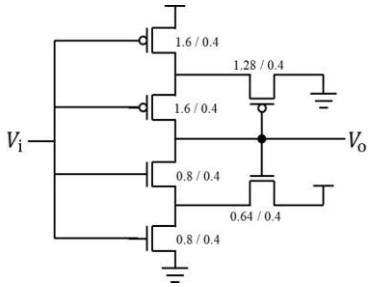
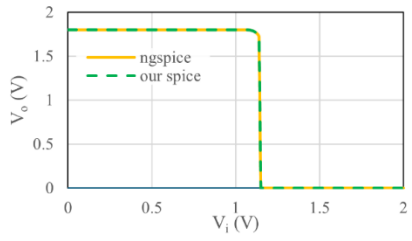
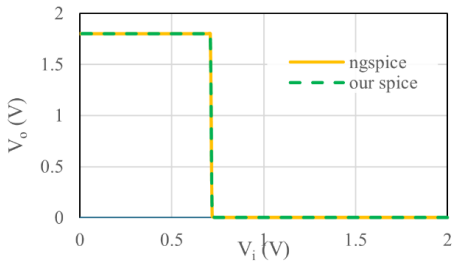


Fig. 10. Schmitt trigger circuit (W/L in unit μm/μm).



(a)

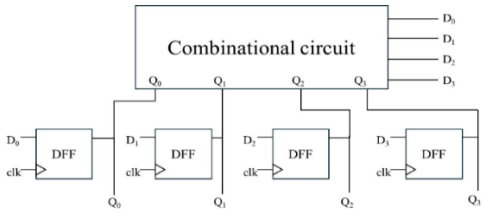


(b)

Fig. 11. (a) V_o with V_i sweep from 0 V to 2 V. (V_{T+} = 1.2 V) (b) V_o with V_i sweep from 2 V to 0 V. (V_{T-} = 0.6 V)

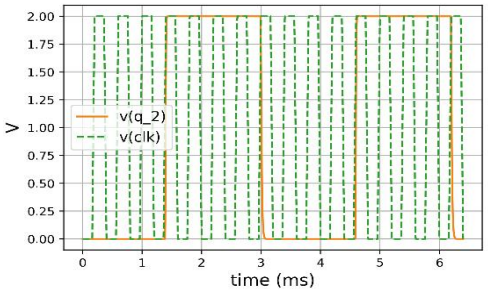
We use the Schmitt trigger circuit for example of DC simulation, which is composed of PMOS and NMOS as in Fig. 10. Fig. 11 shows the output voltage (V_o) when the input voltage (V_i) goes from 0 V to 2 V. It can be seen that when $V_i = V_{T+}$, the output voltage decreases from 1.8 V to 0 V. Similarly, when V_i decreases from 2 V to 0 V, the output rises from 0 V to 1.8 V when $V_i = V_{T-}$.

C. Transient Simulation

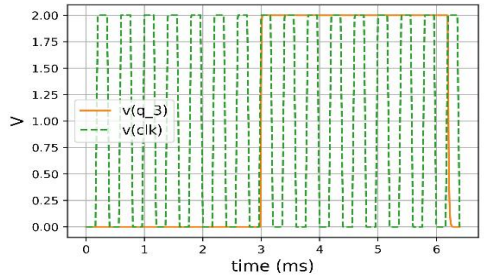


$D_0 = Q_0'$
$D_1 = Q_1 \oplus Q_0$
$D_2 = Q_2Q_0' \vee (Q_2 \oplus Q_1)Q_0$
$D_3 = Q_3'Q_2Q_1Q_0 \vee Q_3Q_2' \vee Q_1' \vee Q_3Q_0'$

Fig. 12. 4-bit counter circuit and Boolean functions in combinational circuit.

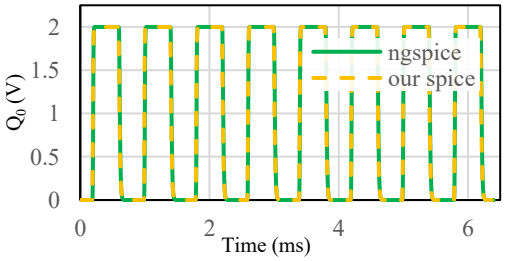


(a)

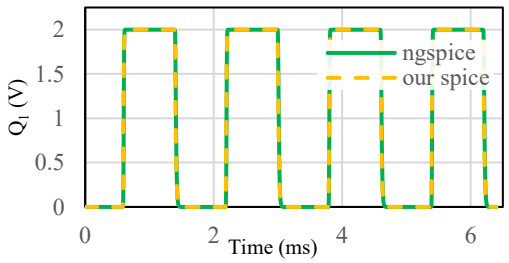


(b)

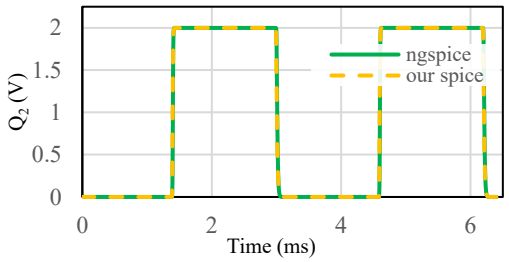
Fig. 13. Simulation result of the 4 bit counter by our SPICE. (a) and (b) are Q2 and Q3 (orange line), respectively. Green dotted line is clk signal.



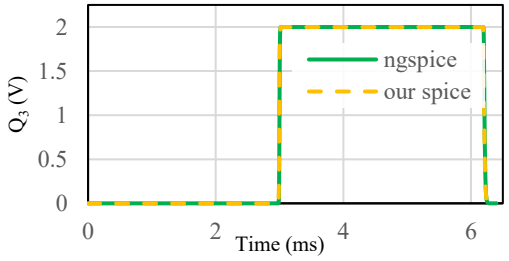
(a)



(b)



(c)



(d)

Fig .14. (a)~(d) are Q_0, Q_1, Q_2, Q_3, respectively.

For transient simulation, we use the 4-bit counter in Fig. 12 for testing, which is implemented using D flip-flops (DFF) and combinational circuit. D_0, D_1, D_2, D_3 are outputs of the combinational circuit, and 4-bit counter's outputs are Q_0, Q_1, Q_2, Q_3.

In Fig. 12, there are six 3-input NAND gates, two resistors and two capacitors in each D flip-flop, and in the combination circuit, we use two 2-input XOR gates, one 4-input AND gate, one 4-input OR gate, one 2-input OR gate, five 2-input AND gates, and five inverters. Hence, in total 4 DFFs and 15 logic gates are used in the circuit, which amounts to 117 PMOS, 117 NMOS, 57 resistors, and 57 capacitors.

Fig. 13 is our spice's result. Simulation end time is 6.4 ms, time step is 5 μs, input signal (clk) period is 0.4 ms, outputs are Q_2 and Q_3, and Q_2 transition occurs at the 4th clk rising edge while Q3 transition occurs at the 8th clk rising edge.

Fig. 14 is our spice's results compared with ngspice. The output periods for Q_0, Q_1, Q_2, and Q_3 are respectively 0.8, 1.6, 3.2, and 6.4 ms. This is in line with the characteristics of a counter.

In all simulation examples, comparing the yellow and green dotted lines, our spice and ngspice nearly perfectly agree, indicating that our spice has computed the results of such a large circuit successfully.

V. Conclusion

This project involves parsing netlists to extract component attributes, utilizing numerical methods like BDF2 and Backward Euler for analysis, error handling for accuracy, and completing various AC/DC/TRAN calculations. By comparing results with those obtained from ngspice, which is an open-source spice tool, we have validated the accuracy of our implementation.

VI. References

[1] I. N. Hajj, *Computational Methods in Circuit Simulation*, 2016.

[2] R. J. Leveque, *Finite Difference Methods for Ordinary and Partial Differential Equations*, SIAM, 2007.

[3] L. W. Johnson and R. D. Riess, *Numerical Analysis*, Adison-Wesley, 1977.

[4] H. Vogt, G. Atkinson, P. Nenzi, and D. Waring, "Ngspice User's Manual Version 40", 2023. [Online]. Available: https://ngspice.sourceforge.io/docs/ngspice-40-manual.pdf

[5] Newton's method introduction [Online]. Available: https://amsi.org.au/ESA_Senior_Years/SeniorTopic3/3j/3j_2content_2.html