Circuit Simulation Project

https://esim.fossee.in/circuit-simulation-project

Name of the participant: Arjun Bathla

Project Guide: Dr R. Maheswari

Title of the circuit: Cyclic Redundancy Check (7, 4) Encoder Circuit

Theory/Description:

In this circuit, a CRC (Cyclic Redundancy Check) encoder has been simulated, with data word size (k) = 4 bits, and code word size (n) = 7 bits. This circuit can be used to encode parallelly generated 4-bit data before transmitting it over an error prone channel. At the receiver side, this encoded code word has to be decoded to detect any errors occurred during the transmission. On both sender and receiver sides, a common divisor of size (n-k+1) = 4 bits is used for encoding and decoding.

The encoder generates a remainder of size (n-k) = 3 bits and appends it to the data word before transmission. This transmitted data is the code word. On the receiver side, the encoder generates a syndrome of size (n-k) = 3 bits. If this syndrome is zero, no error is detected in the received code word and the extracted data word is accepted, otherwise error is detected and the extracted data word is discarded.

In this encoder, the remainder is generated using modulo 2 division, in which at every stage, the respective dividend and divisor undergo the XOR operation. This method is illustrated in Figure 1. The data word is 1001, and the divisor is 1011. Three 0's are appended to the data word. This same divisor is to be used in the decoder as well. The 3-bit remainder, 110 is appended to the data word to generate the 7-bit code word.

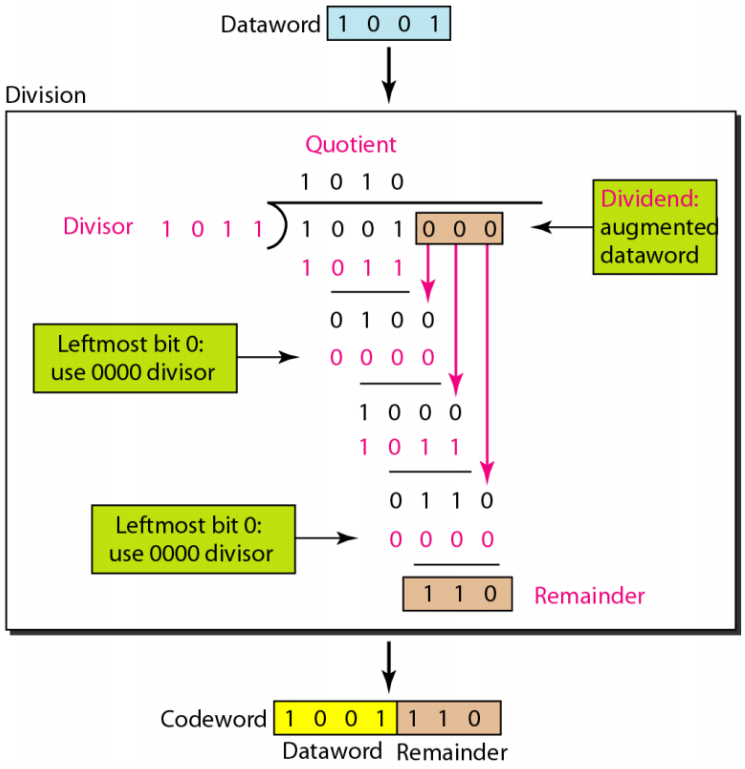

Figure 1: CRC (7, 4) Encoding

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

Figure 2: CRC (7, 4) Codebook

The circuit in this project assumes that the data word is being input in a parallel manner instead of serial, eliminating the need for a shift register.

At each step in the division, we observe that if the dividend at that stage has MSB = 0, its remaining 3 bits are XORed with 000, otherwise they are XORed with the last 3 bits of the divisor. This will be true for any divisor where MSB = 1, which is a necessary condition for CRC encoding as it uses modulo 2 division. Figure 2 shows the codebook for all possible 4-bit data words.

Essentially, the XOR operation with the last 3 bits of the dividend at each stage will be performed after the last 3 bits of the divisor are undergo the AND operation with the MSB of that dividend. Figure 3 shows the schematic for the subcircuit. This subcircuit will be used 4 times, one for each stage, in the main circuit.

Figure 4 shows the symbol of this subcircuit. Ports 1 through 4 are the four dividend bits from MSB to LSB respectively. Ports 5 through 7 are the last three bits of the divisor respectively, since the MSB will always be 1, eliminating the need for a separate port. Ports 8 through 10 are the three remainder bits from MSB to LSB respectively.

The remainder at each stage will act as the first three bits of the dividend for the next stage, and the LSB for that dividend would be 0. The last three bits of the divisor will be the same at each stage. The remainder obtained from the fourth stage will be the final remainder that is to be appended to the data word before transmission.

Circuit Diagram(s):

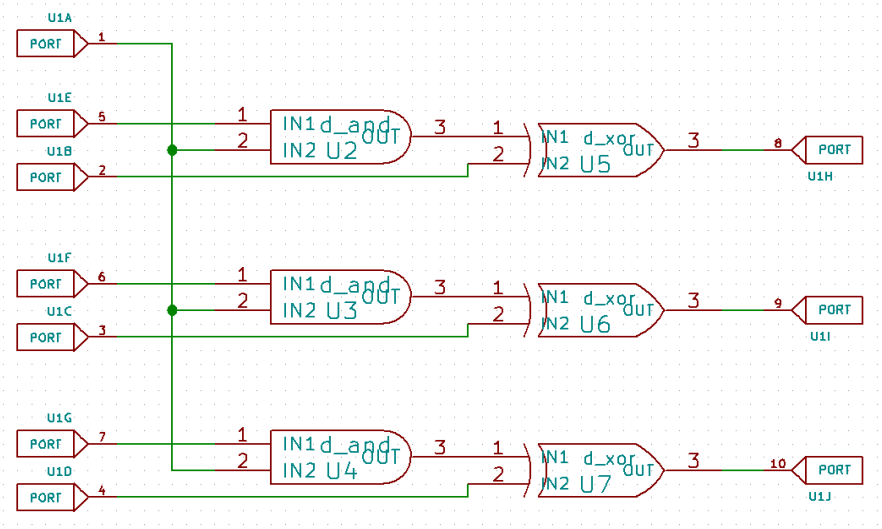

Figure 3: Modulo 2 division subcircuit schematic

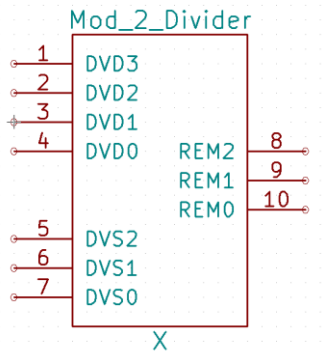

Figure 4: Modulo 2 division subcircuit symbol

Figure 5 shows the schematic of the main circuit. For an easy-to-understand simulation, the divisor has been fixed to 1011, while the code word varies from 0000 to 1111, such that the remainder for each dividend can be cross-verified with the codebook in Figure 2.

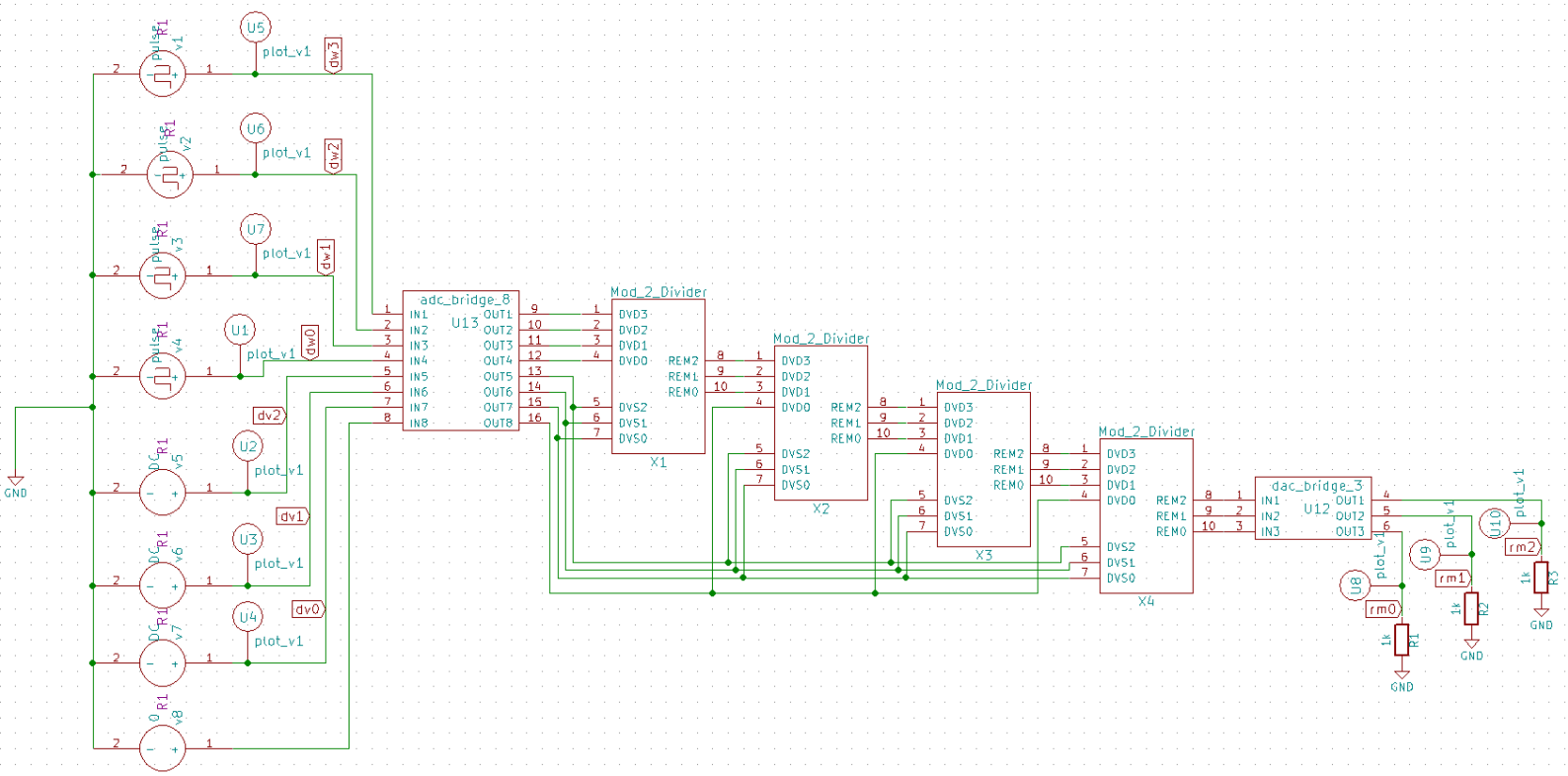

Figure 5: CRC (7, 4) Encoder Schematic

Labels dw3 to dw0 refer to the four data word bits starting from MSB to LSB. Labels dv2 to dv0 refer to the last 3 bits of the common divisor. The source v8 is fixed to 0V, which will act as the LSB of the dividend at each stage except the first. During simulation, dv2, dv1 and dv0 are fixed to 0V, 5V and 5V, i.e., the divisor is fixed as 1011. The four dividend signals will vary between 0V and 5V, generating every possible combination for the dividend from 0000 to 1111. An ADC bridge is used to convert the analog signals to digital signals at the input side. At the output side, a DAC bridge converts the digital remainder bits to analog signals, where 0 = 0V and 1 = 5V. Labels rm2 to rm0 refer to the remainder bits from MSB to LSB.

Results (Input, Output waveforms):

Figures 6a to 6c show the plots for dv2 to dv0. Figures 7a to 7d show the plots for dw3 to dw0. Figures 8a to 8c show the plots for rm2 to rm0. The remainders for each data word can be verified from the codebook.

Figures 9a to 9e show the same plots using Python plot for a better visualization.

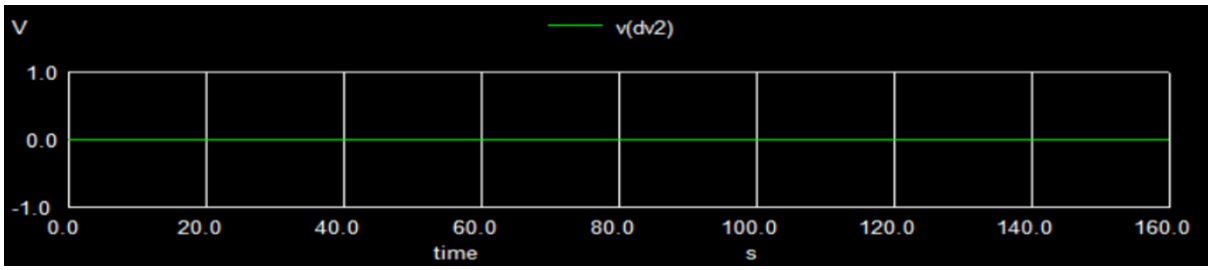

Figure 6a: Analog signal for dv2

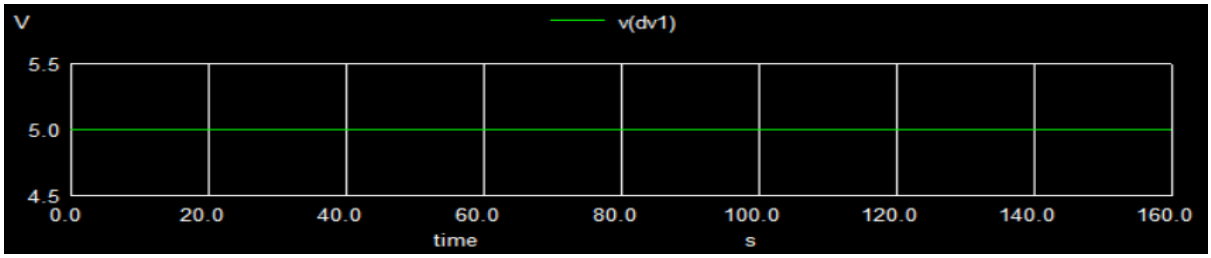

Figure 6b: Analog signal for dv1

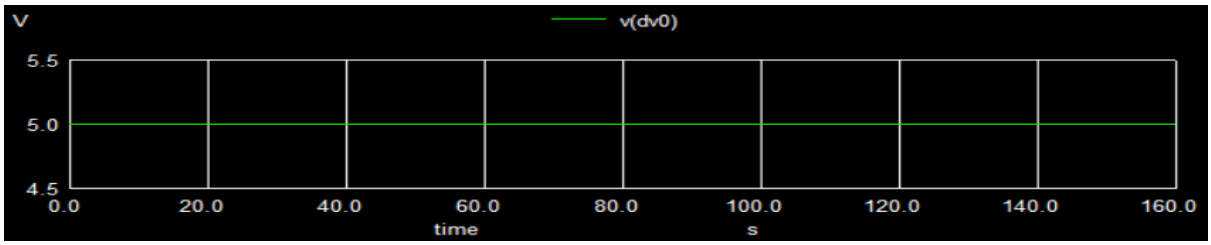

Figure 6c: Analog signal for dv0

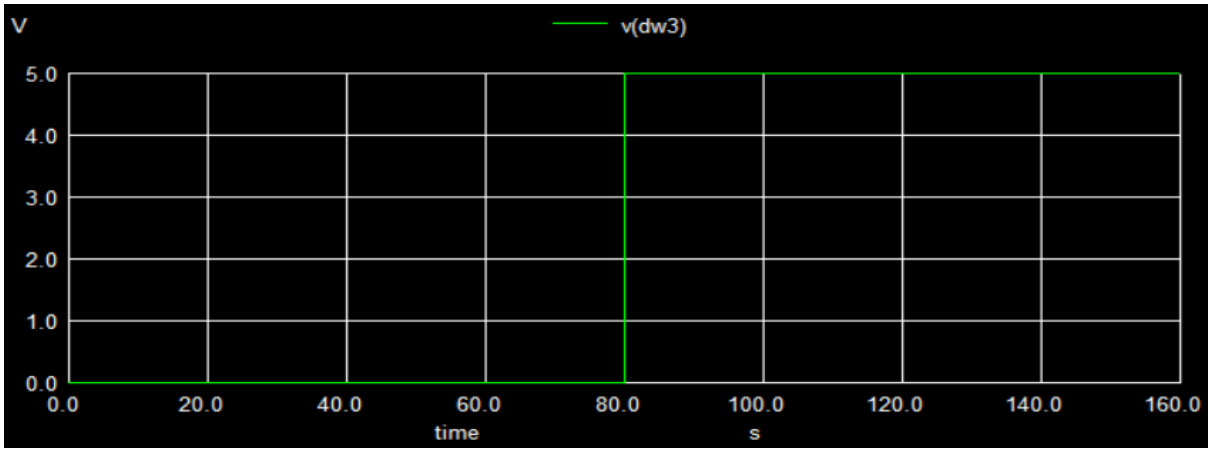

Figure 7a: Analog signal for dw3

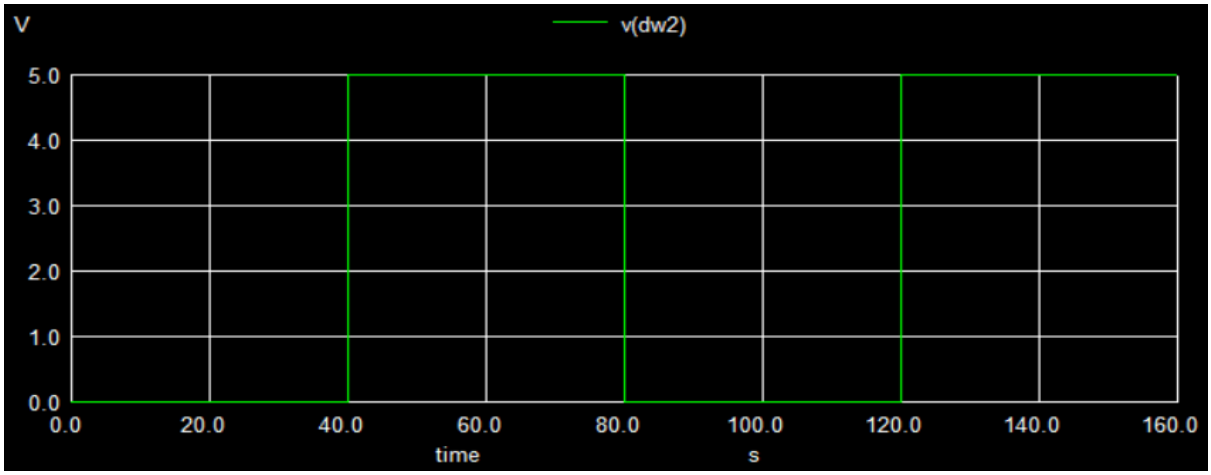

Figure 7b: Analog signal for dw2

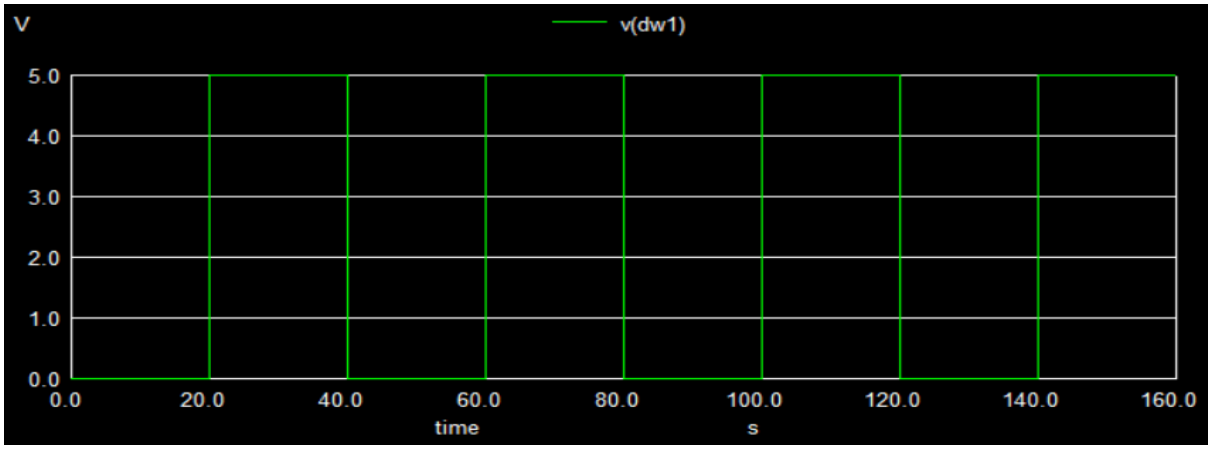

Figure 7c: Analog signal for dw1

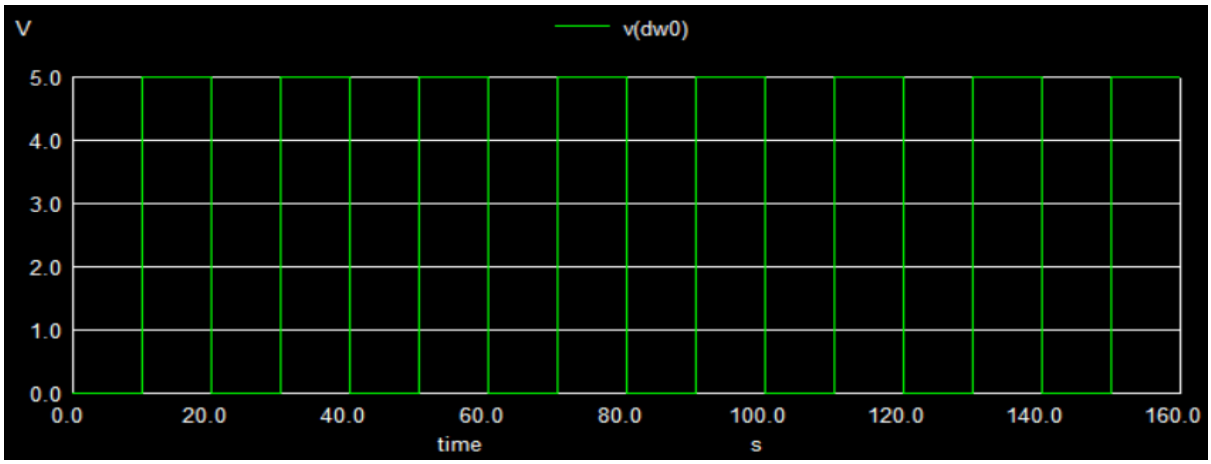

Figure 7d: Analog signal for dw0

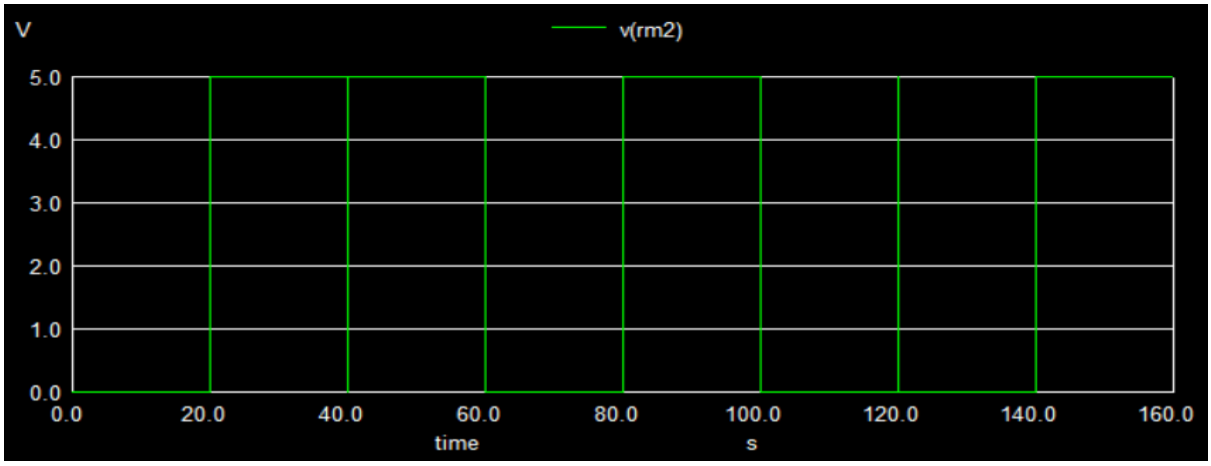

Figure 8a: Analog signal for rm2

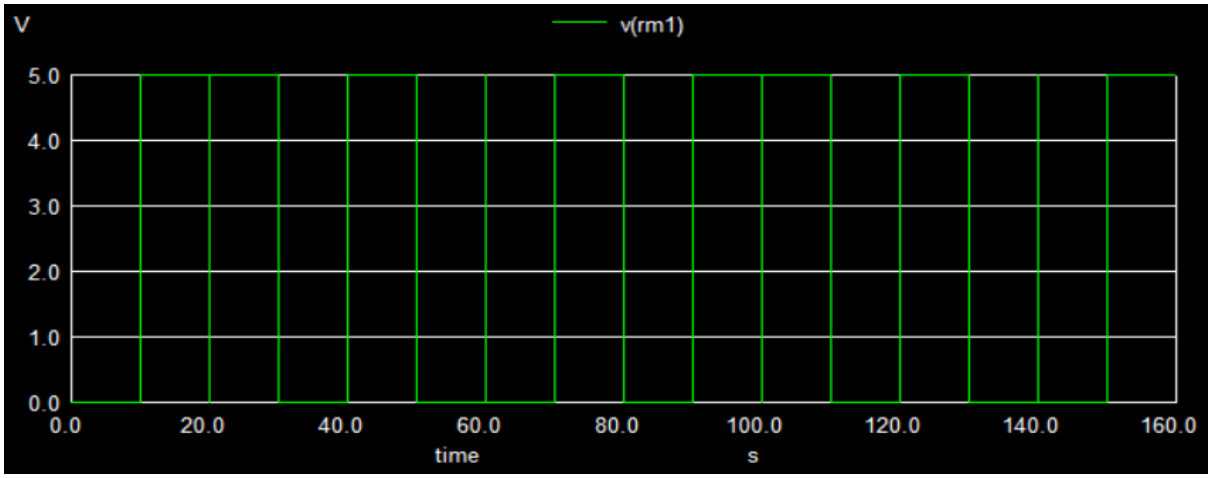

Figure 8b: Analog signal for rm1

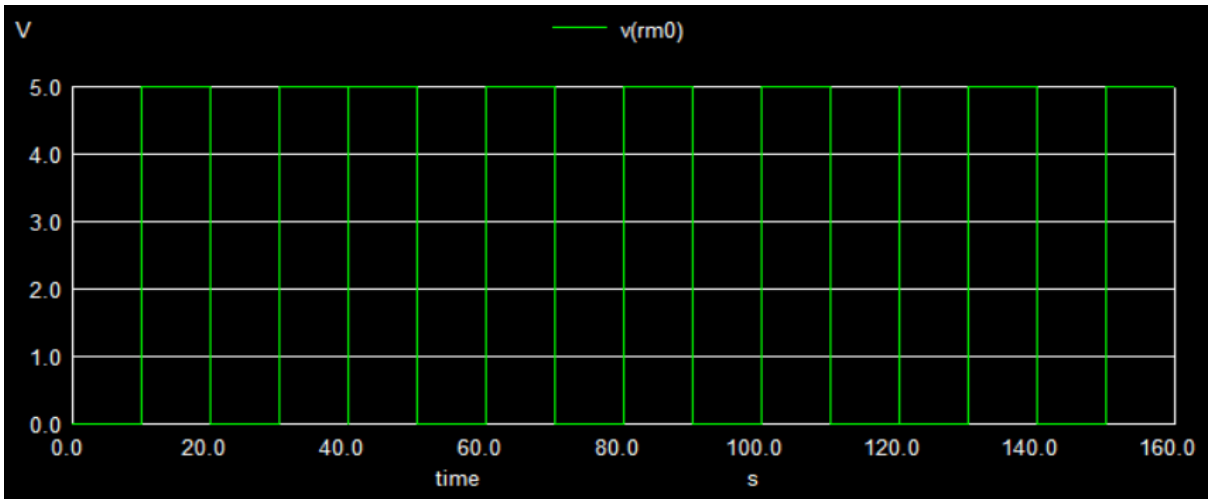

Figure 8c: Analog signal for rm0

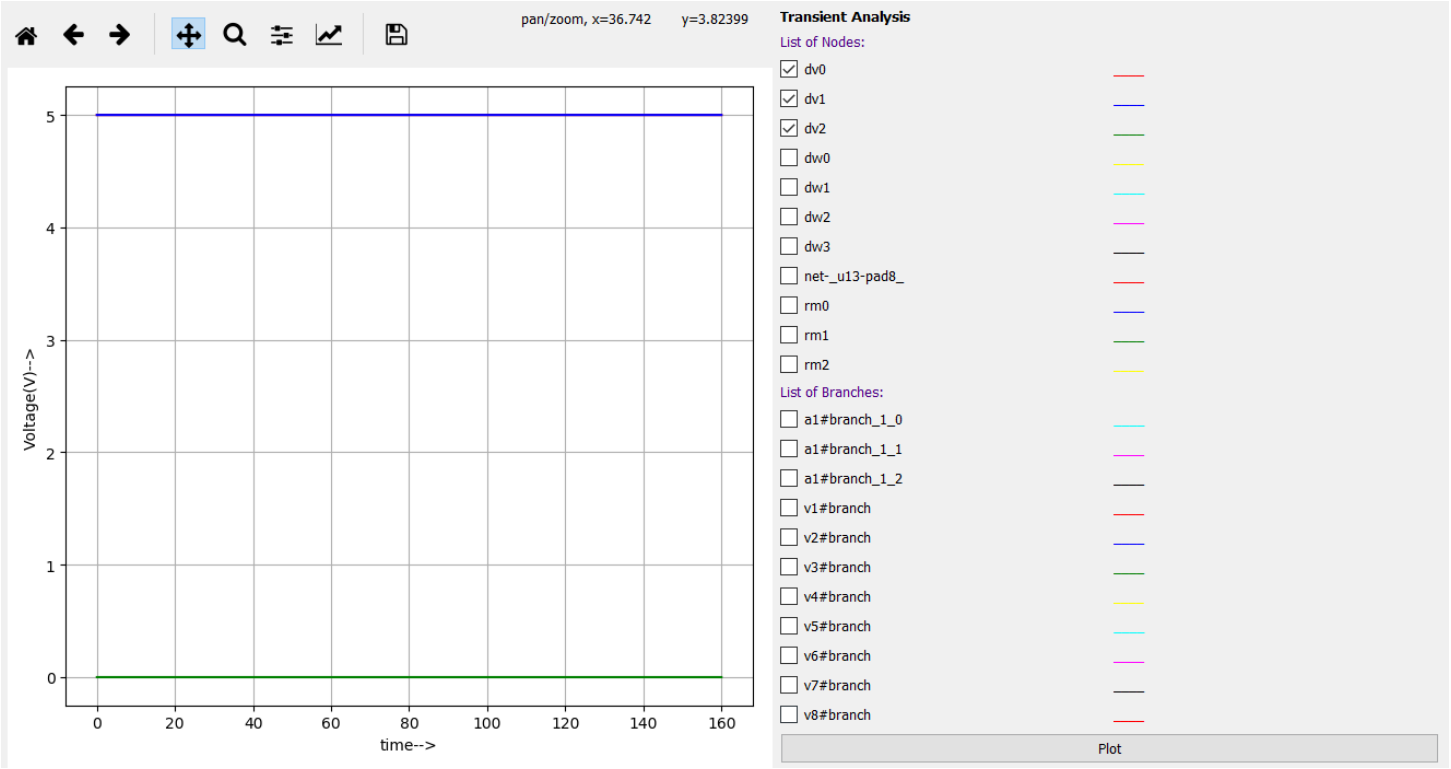

Figure 9a: Analog signals for dv2 to dv0

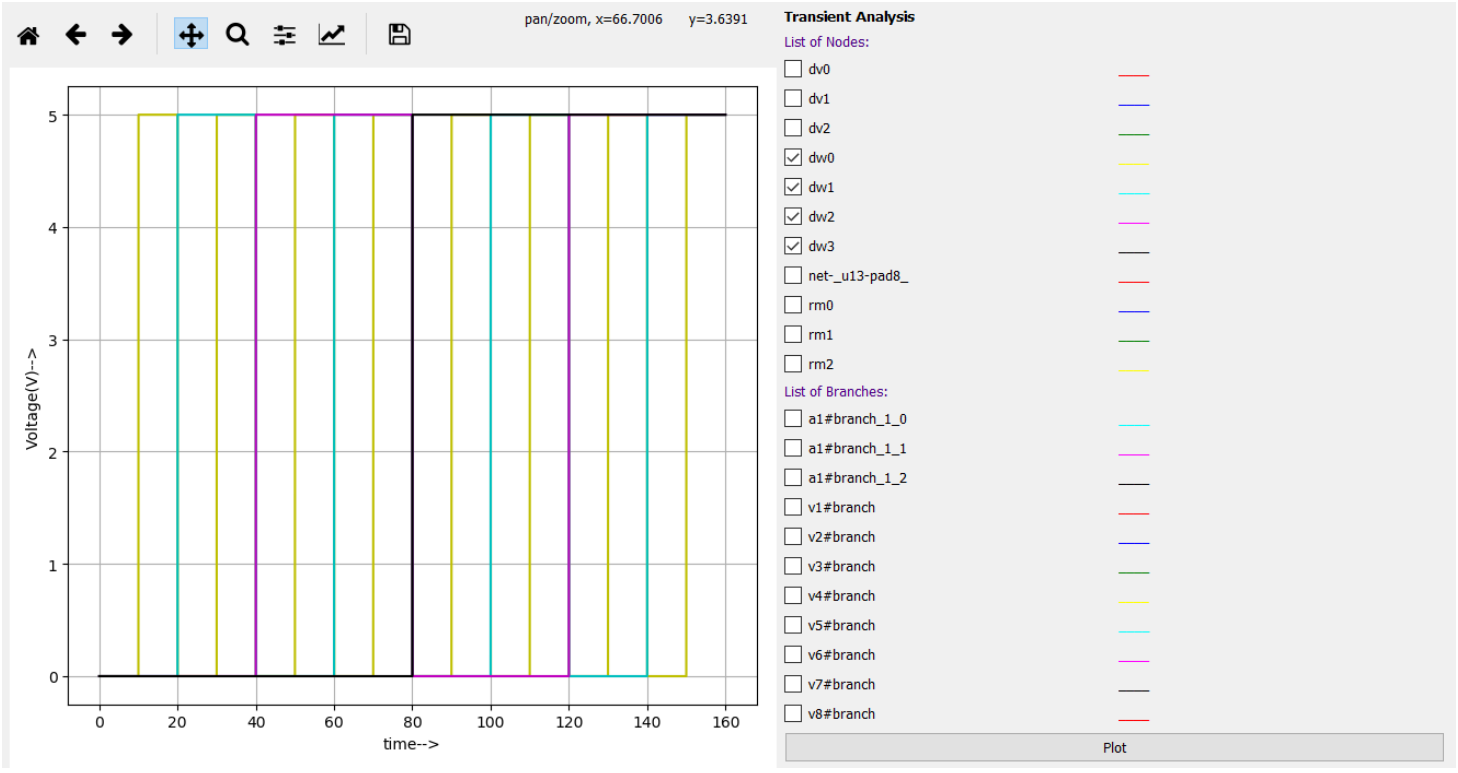

Figure 9b: Analog signals for dw3 to dw0

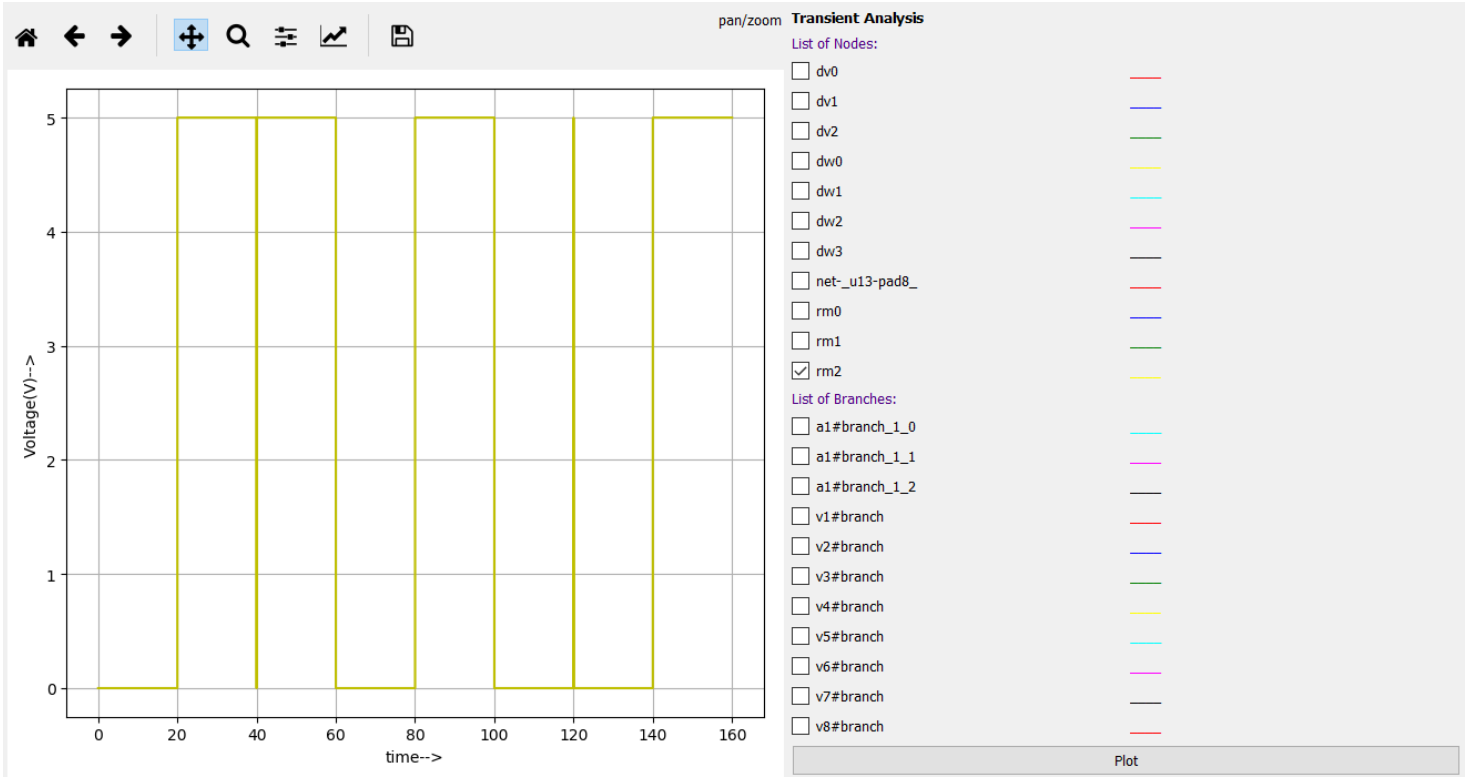

Figure 9c: Analog signal for rm2

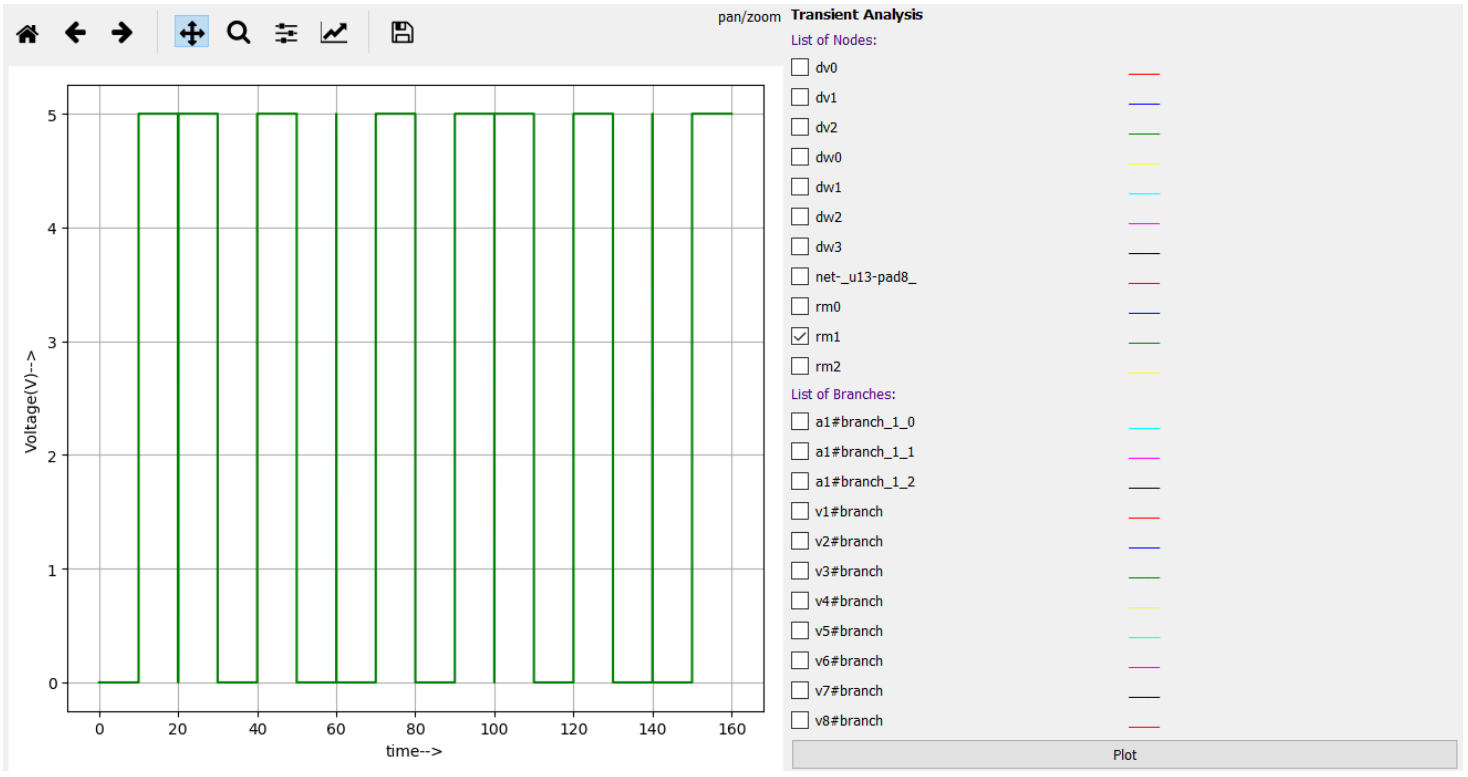

Figure 9d: Analog signal for rm1

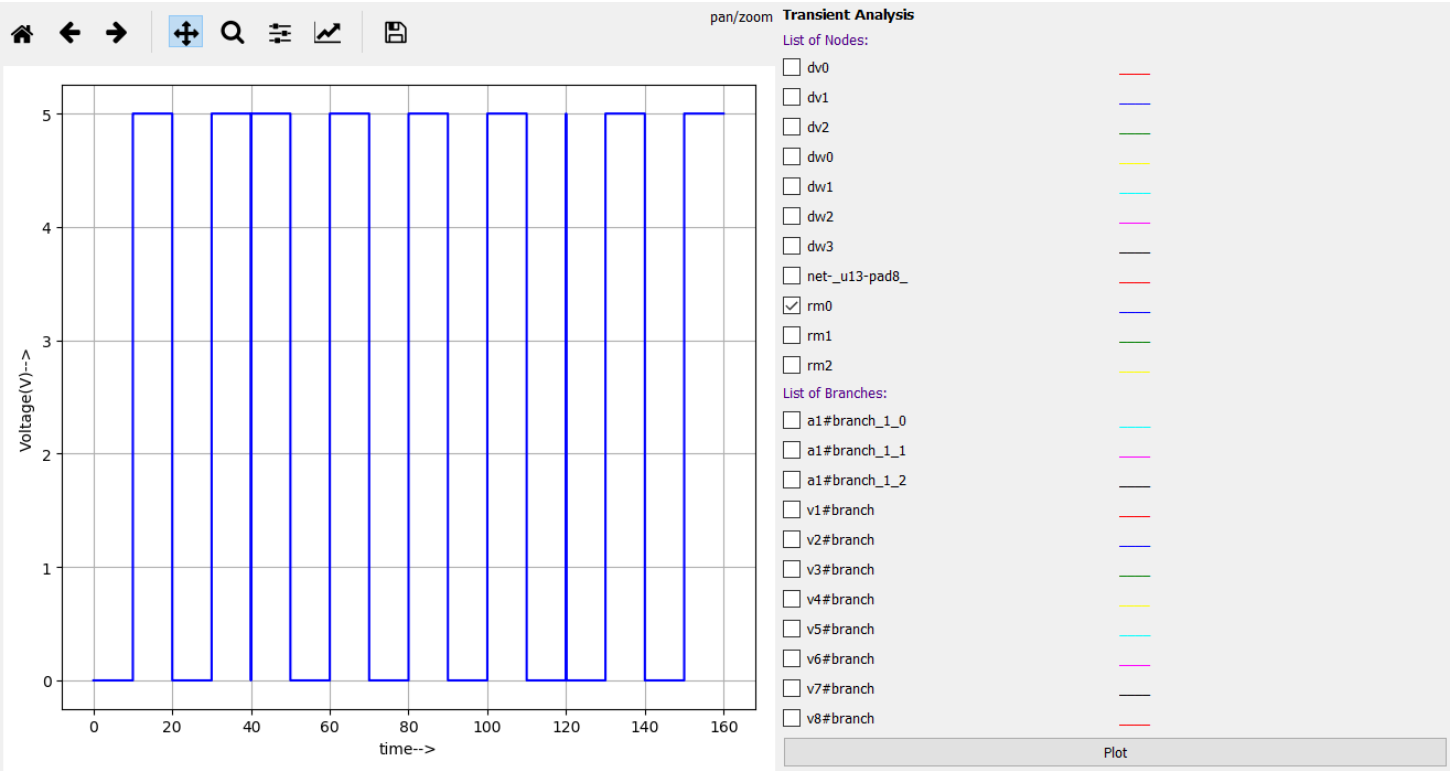

Figure 9e: Analog signal for rm0

Simulation Parameters for reference:

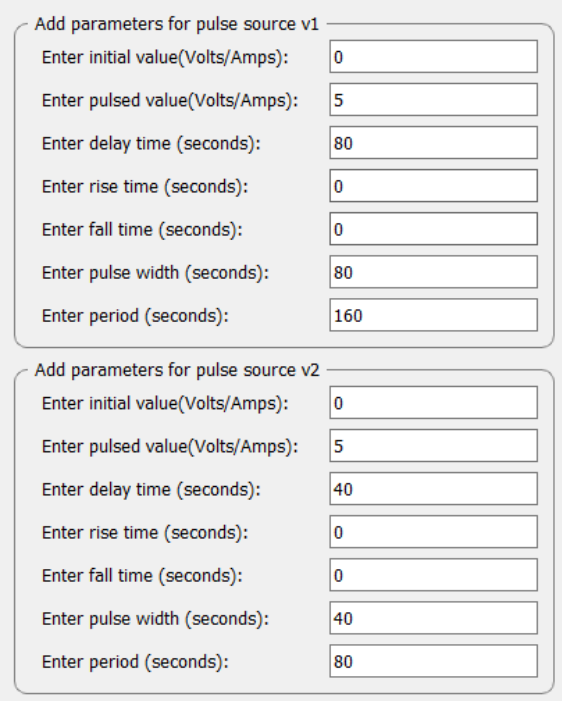

Figure 10a

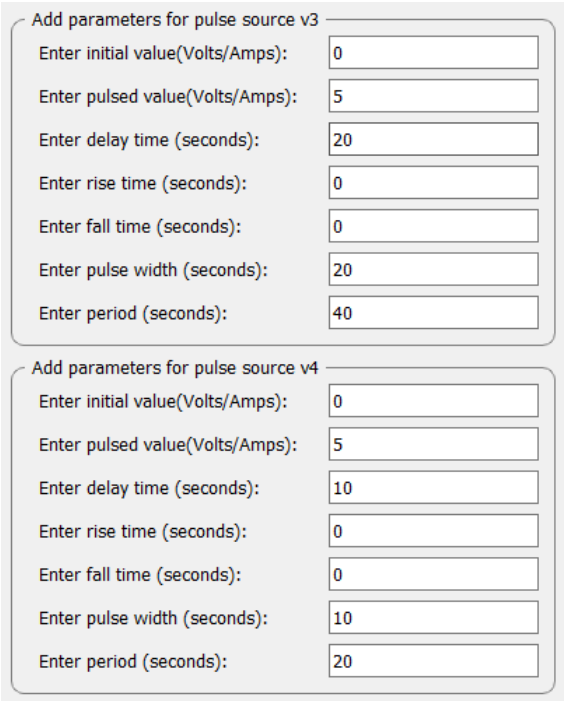

Figure 10b

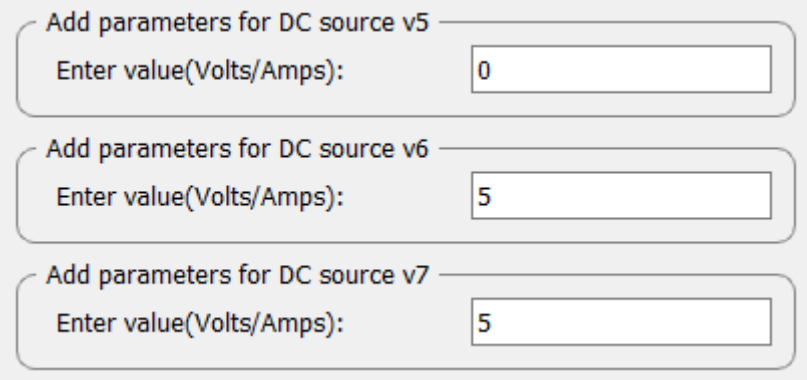

Figure 10c

Source/Reference(s):

https://cse.iitkgp.ac.in/~ksrao/pdf/iti-18/slide-3.pdf

Pages 58-60