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) Decoder Circuit

Theory/Description:

In this circuit, a CRC (Cyclic Redundancy Check) decoder has been simulated, with data word size (k) = 4 bits, and code word size (n) = 7 bits. This circuit can be used to decode parallelly generated 4-bit data after transmitting it over an error prone channel. At the sender side, the data word has to be encoded 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.

On the receiver side, the decoder generates a syndrome (remainder) of size (n-k) = 3 bits, from the received code word. 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. The extracted data word is essentially the first four bits of the code word, starting from MSB.

In this decoder, the syndrome 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 code words for the two cases are 1001_110 and 1000_110, and the divisor is 1011, same as that in the encoder. The 3-bit remainders in both the cases are called syndromes. In case 1, since the syndrome is 000, the extracted data word is accepted, while in case 2 it is discarded, since the syndrome is not 000.

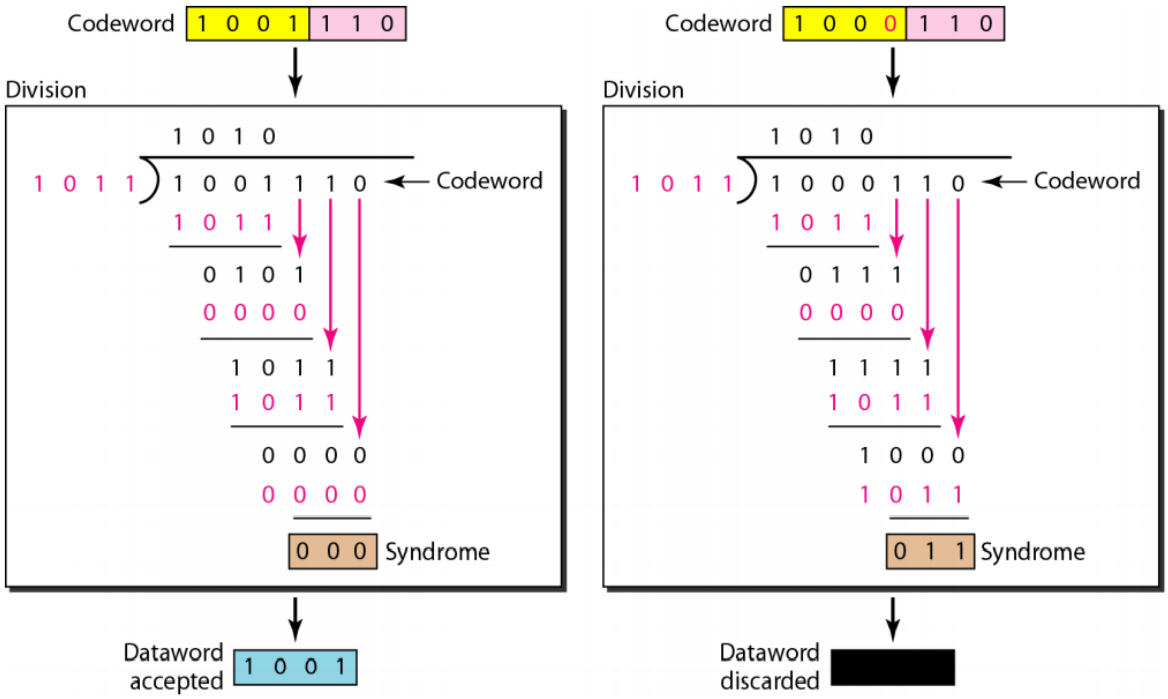

Figure 1: CRC (7, 4) Decoding

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 code 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. Since the minimum Hamming distance between any two code words is 3, this technique can detect errors of up to 2 bits.

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 undergo the AND operation with the MSB of that dividend. Figure 3a shows the schematic for the subcircuit. This subcircuit will be used 4 times, one for each stage, in the main circuit.

Figure 3b 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 syndrome. The 'accept' output should be 1 only when the syndrome is 000, hence a 3-input NOR gate subcircuit is used, for which the schematic and symbol are shown in Figures 4a and 4b respectively.

Circuit Diagram(s):

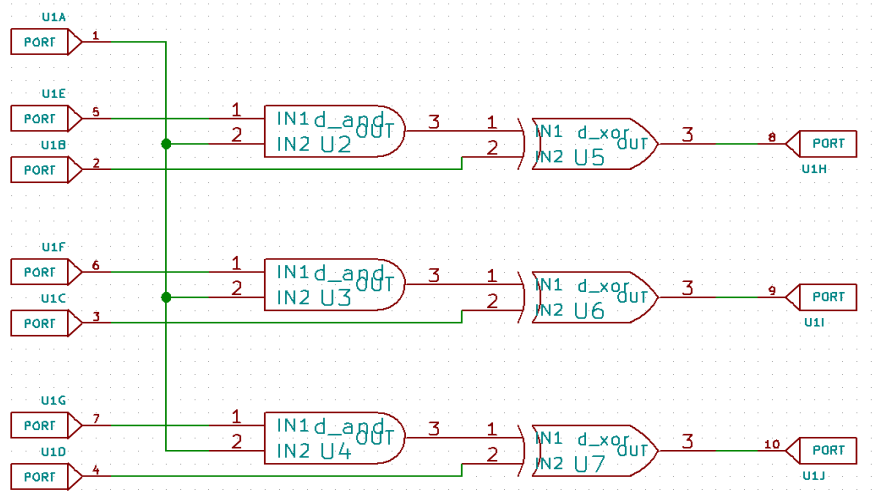

Figure 3a: Modulo 2 division subcircuit schematic

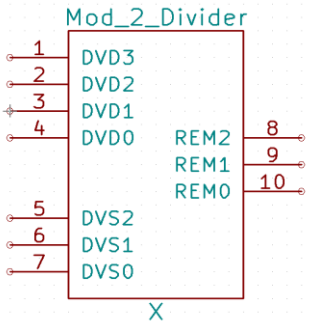

Figure 3b: Modulo 2 division subcircuit symbol

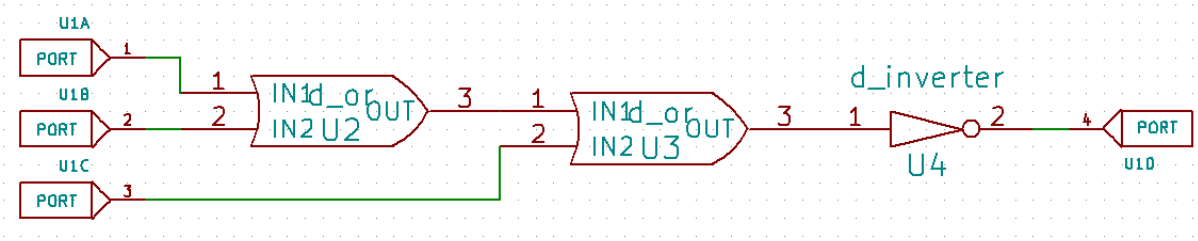

Figure 4a: 3-input NOR gate subcircuit schematic

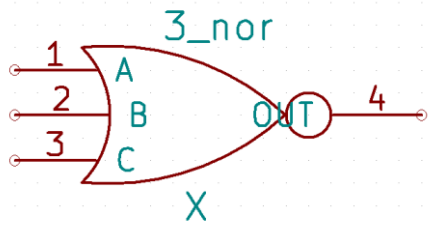

Figure 4b: 3-input NOR gate subcircuit symbol

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 1001_000 to 1001_111, such that the accept output can be cross-verified with the output in the case 1 of Figure 1.

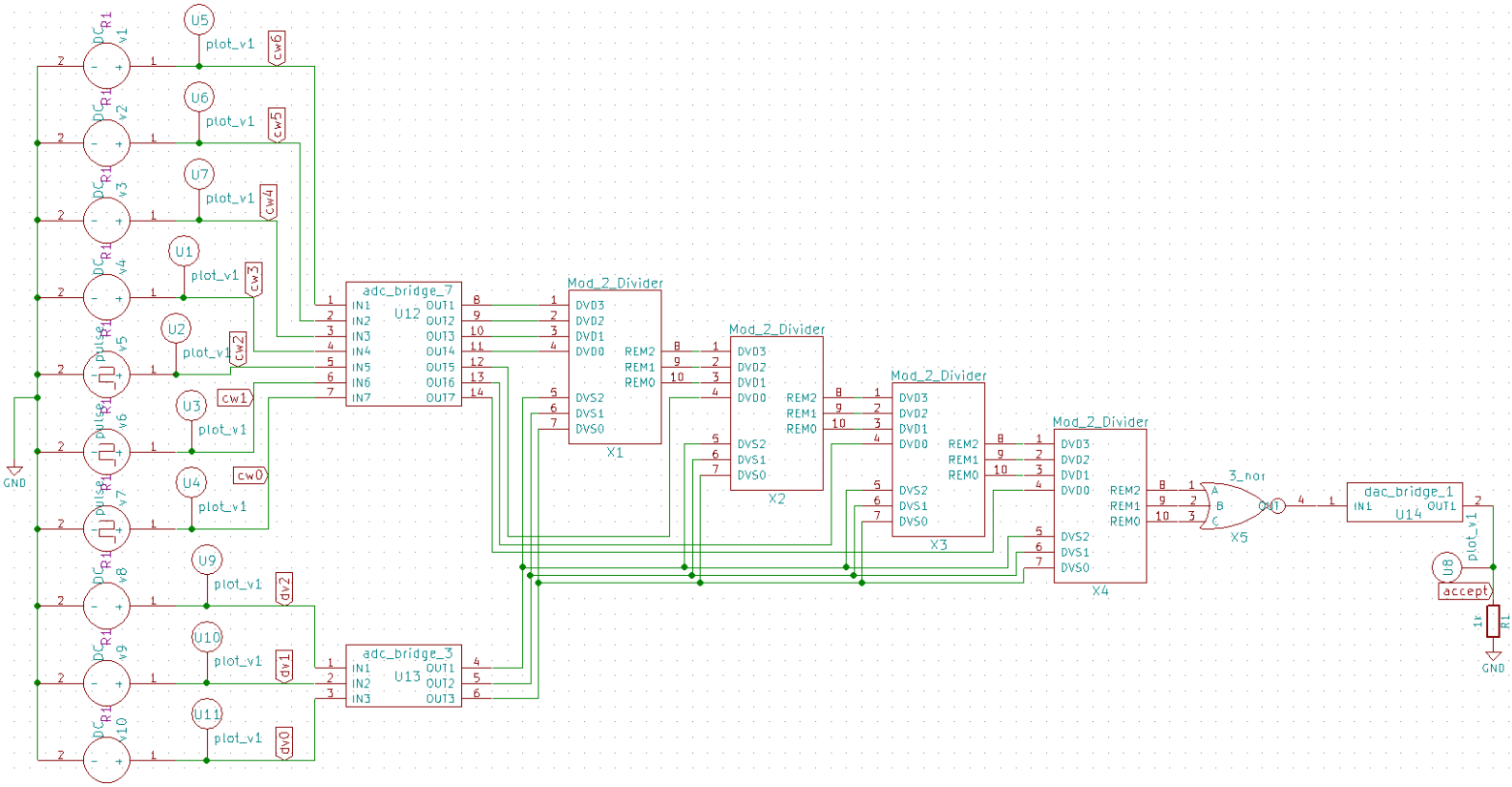

Figure 5: CRC (7, 4) Decoder Schematic

Labels cw6 to cw0 refer to the seven code word bits starting from MSB to LSB. Labels dv2 to dv0 refer to the last 3 bits of the common divisor. During simulation, dv2, dv1 and dv0 are fixed to 0V, 5V and 5V, i.e., the divisor is fixed as 1011. The signals cw6 through cw3 are fixed to 5V, 0V, 0V and 5V, i.e., the first four bits are fixed to 1001, while the signal cw2, cw1 and cw0 will vary between 0V and 5V, generating every possible combination for the code word from 1001_000 to 1001_111. Two ADC bridges are used to convert the analog signals to digital signals at the input side. At the output side, a DAC bridge converts the digital 'accept' output bit to analog signal, where 0 = 0V and 1 = 5V.

Results (Input, Output waveforms):

Figures 6a to 6c show the plots for dv2 to dv0. Figures 7a to 7g show the plots for cw6 to cw0. Figure 8 shows the plot for 'accept'. It can be verified that the 'accept' bit is 1 only when the code word is 1001_110.

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

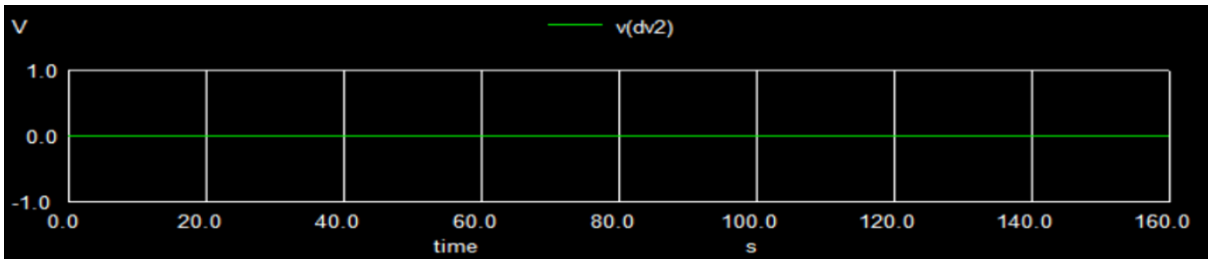

Figure 6a: Analog signal for dv2

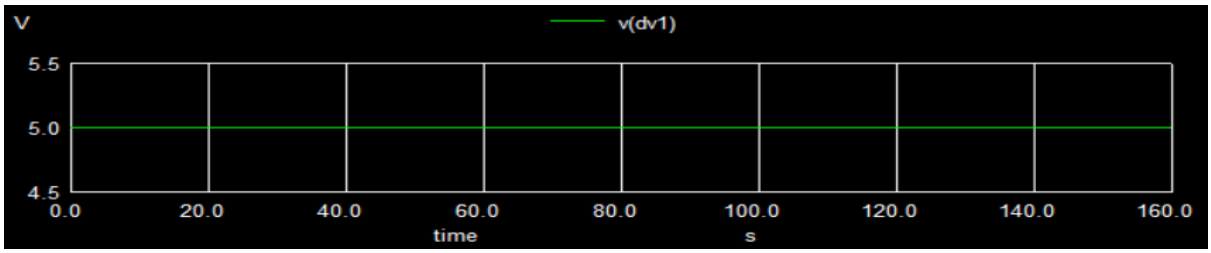

Figure 6b: Analog signal for dv1

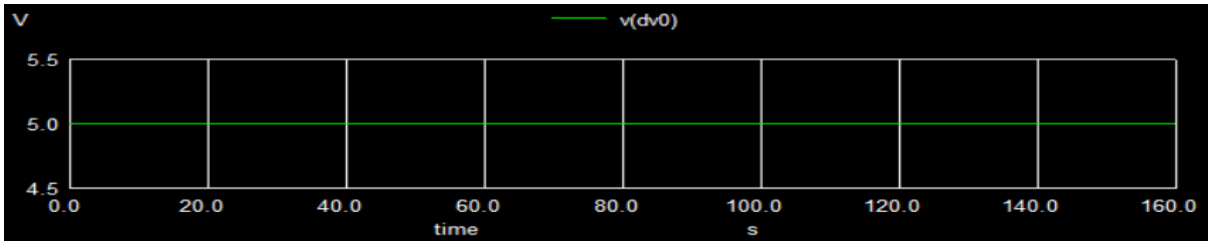

Figure 6c: Analog signal for dv0

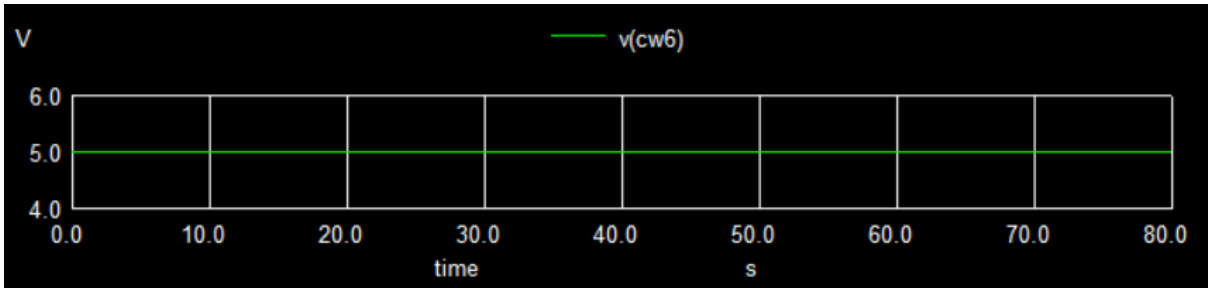

Figure 7a: Analog signal for cw6

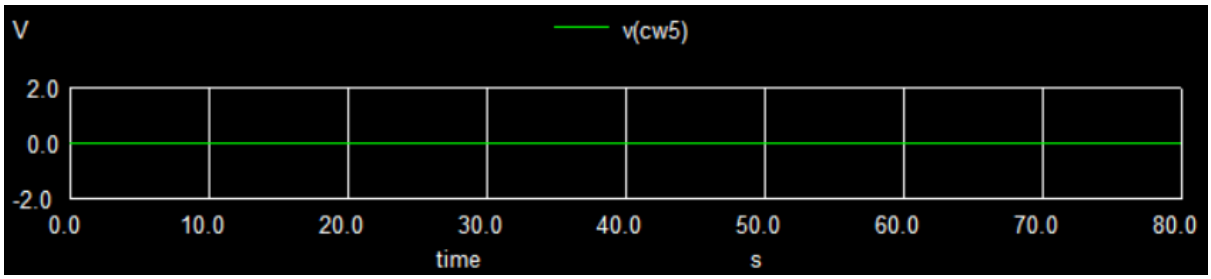

Figure 7b: Analog signal for cw5

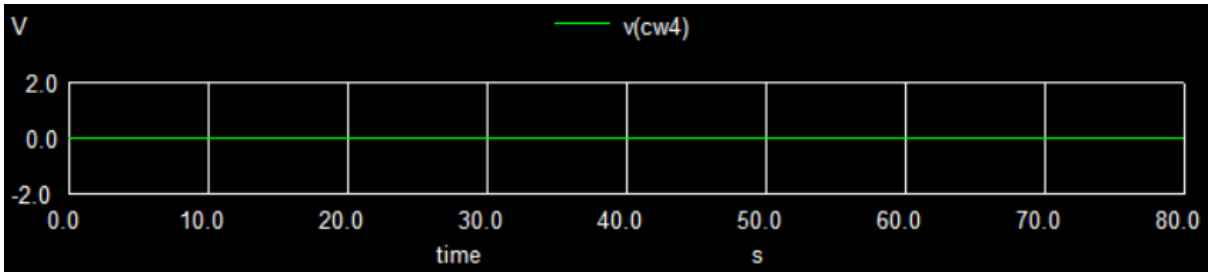

Figure 7c: Analog signal for cw4

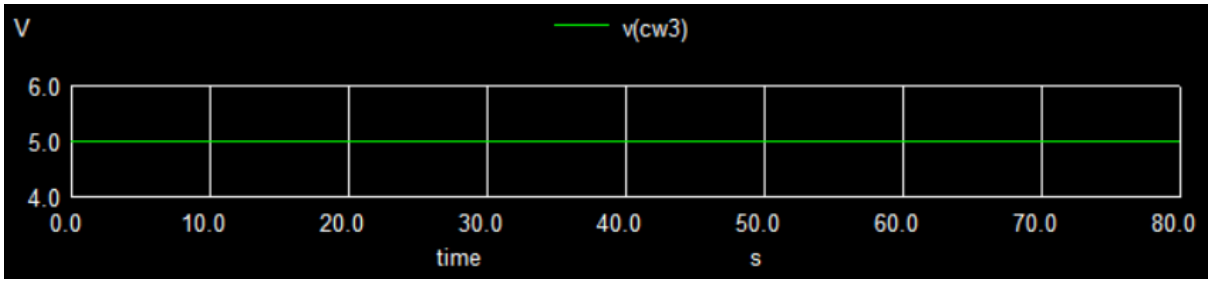

Figure 7d: Analog signal for cw3

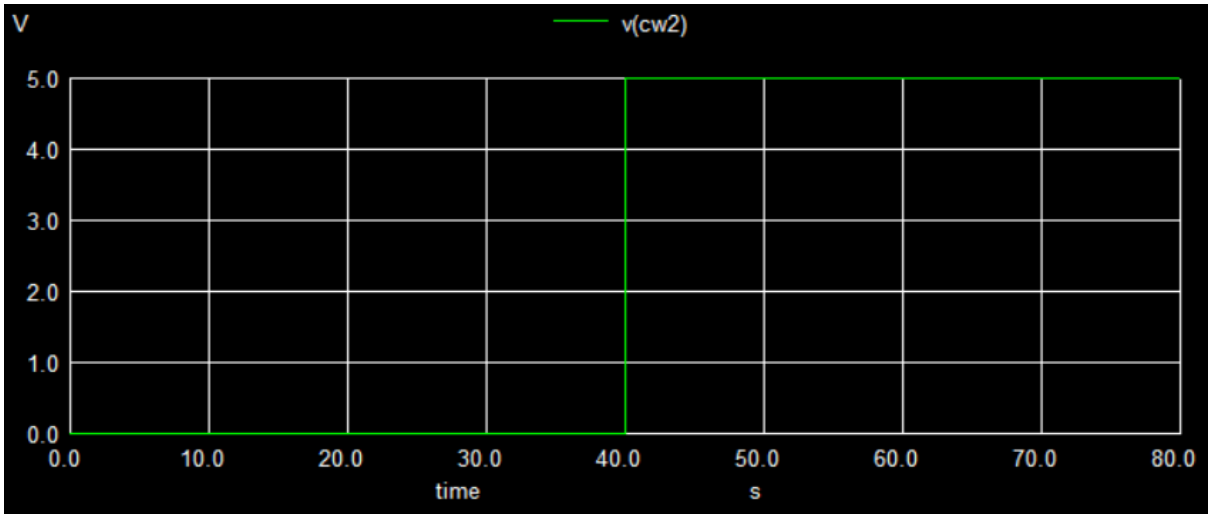

Figure 7e: Analog signal for cw2

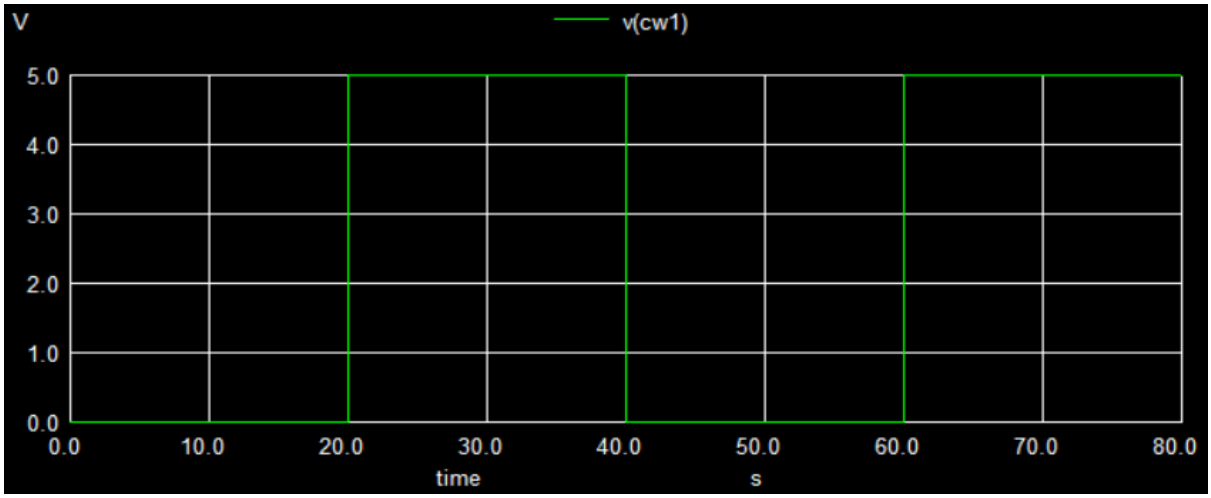

Figure 7f: Analog signal for cw1

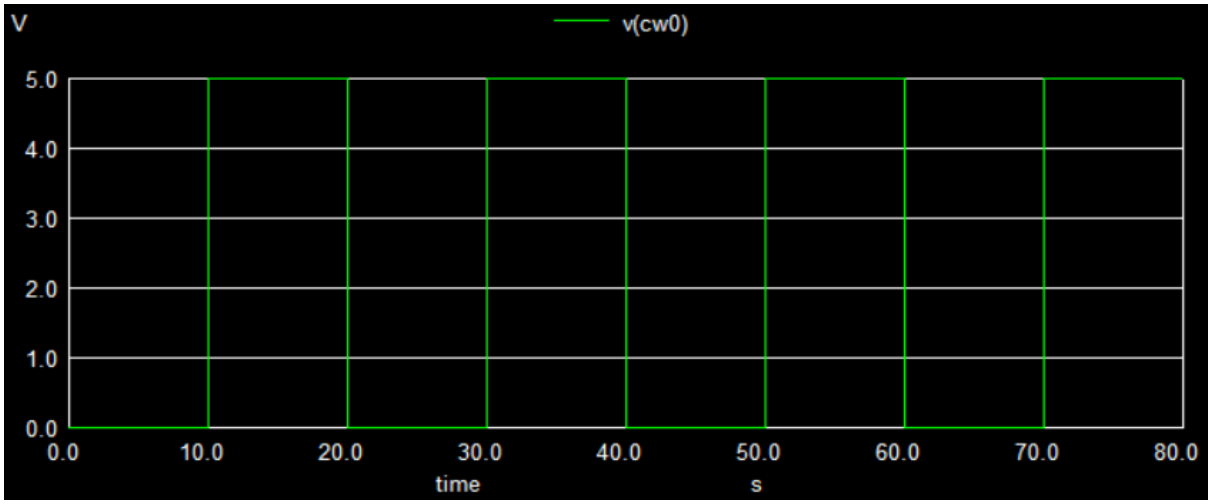

Figure 7g: Analog signal for cw0

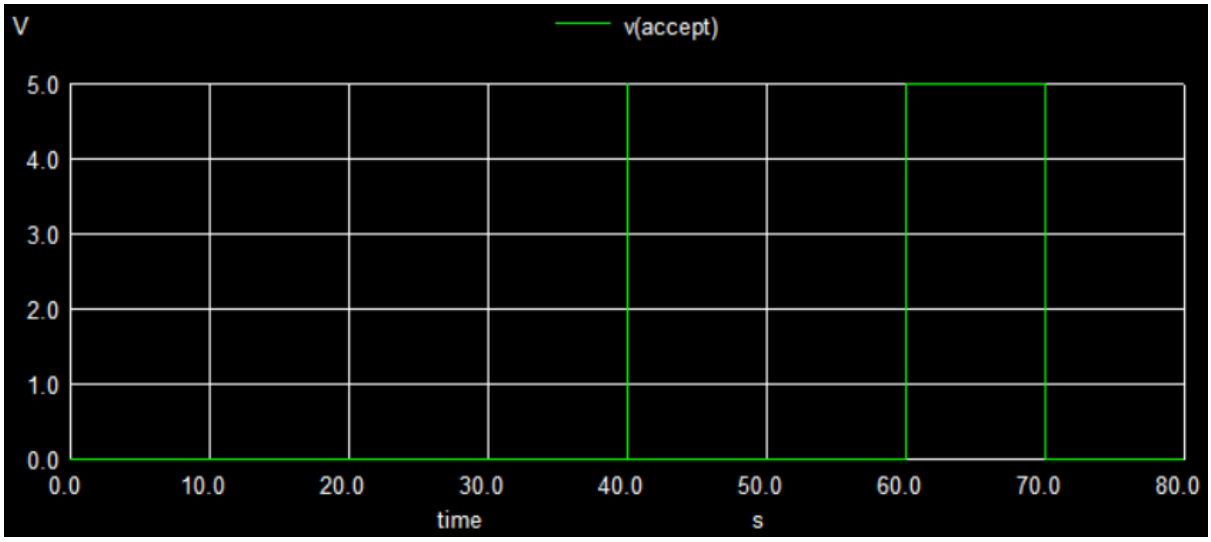

Figure 8: Analog signal for 'accept'

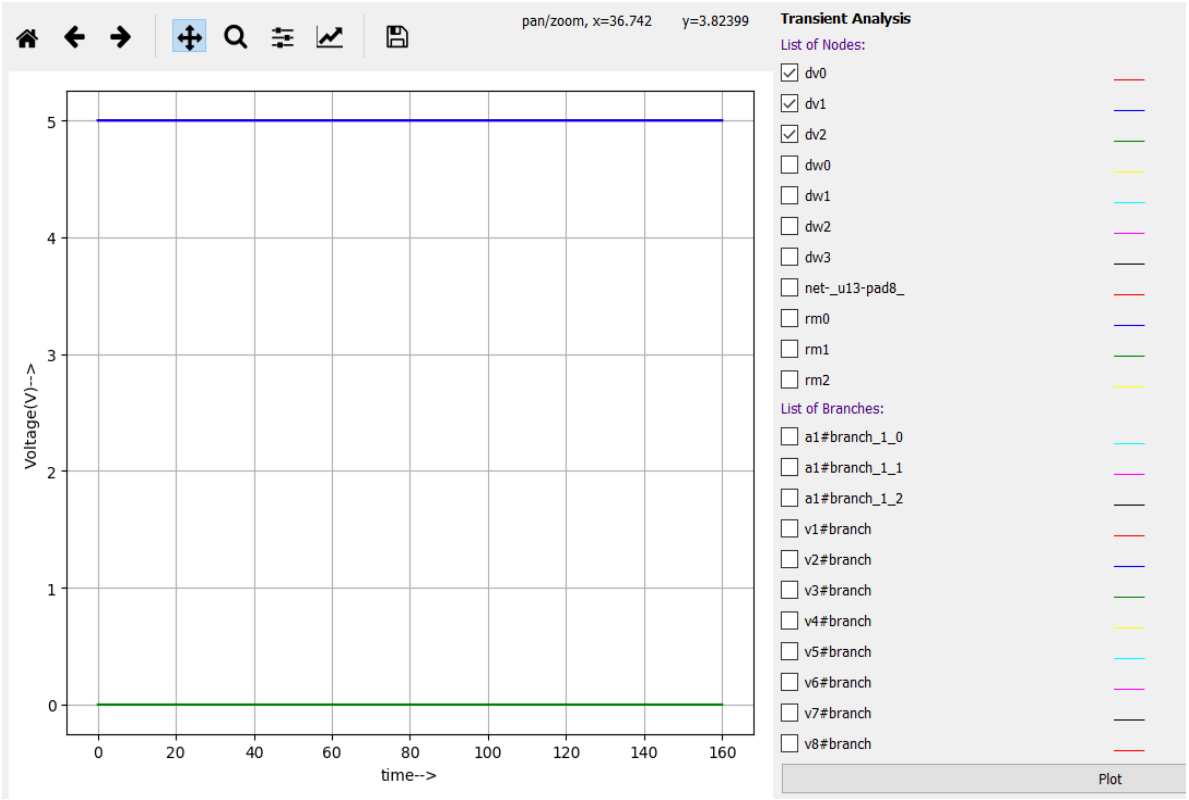

Figure 9a: Analog signals for dv2 to dv0

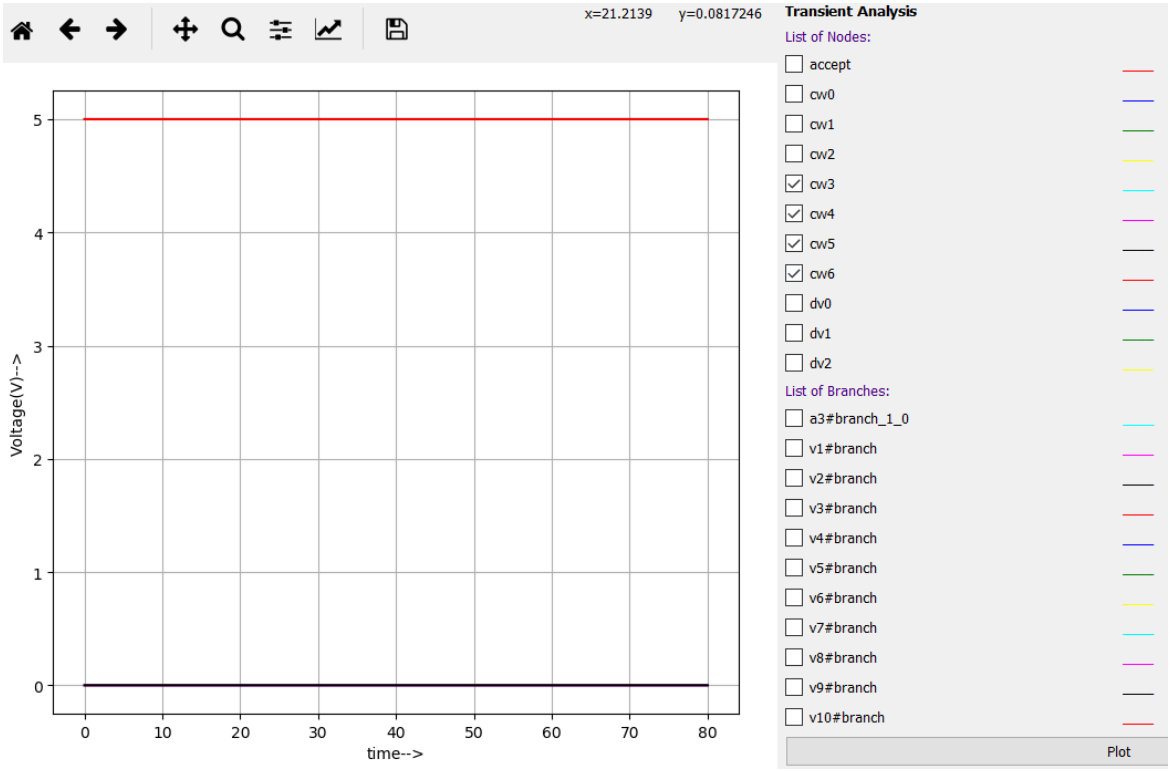

Figure 9b: Analog signals for cw6 to cw3

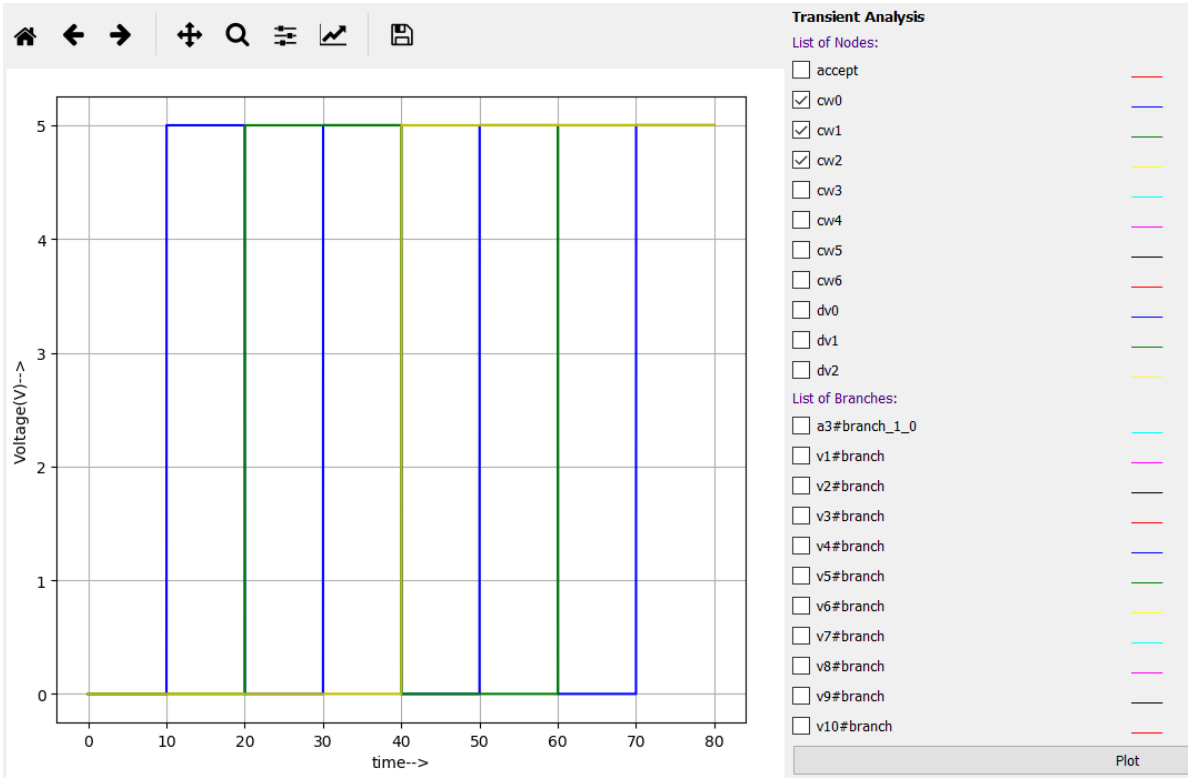

Figure 9c: Analog signals for cw2 to cw0

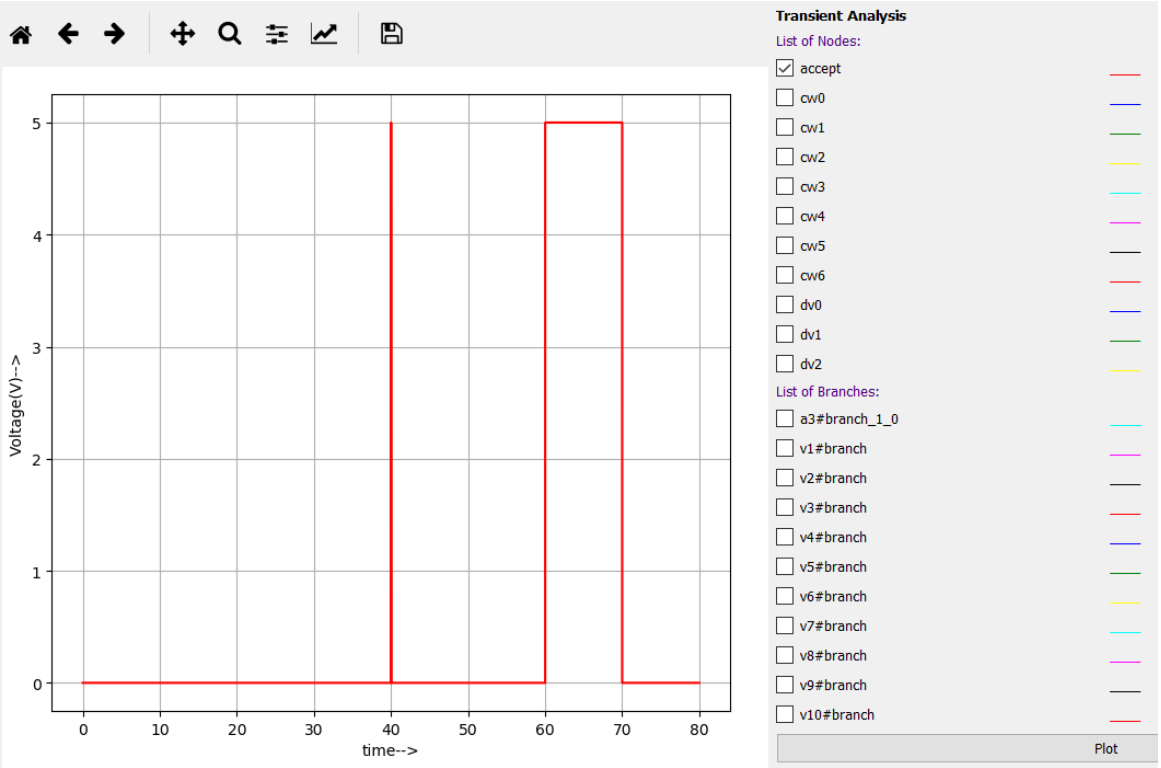

Figure 9d: Analog signal for 'accept'

Simulation Parameters for reference:

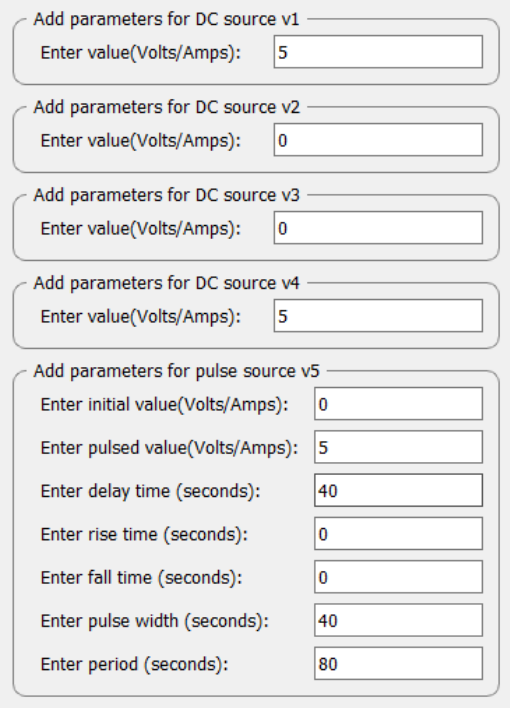

Figure 10a

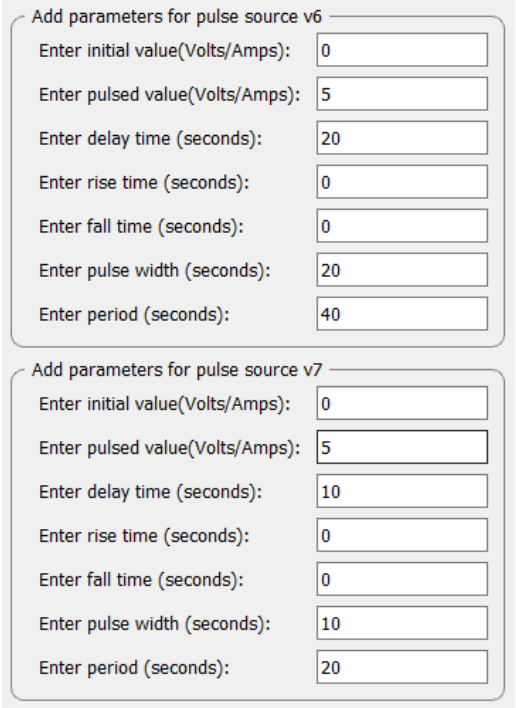

Figure 10b

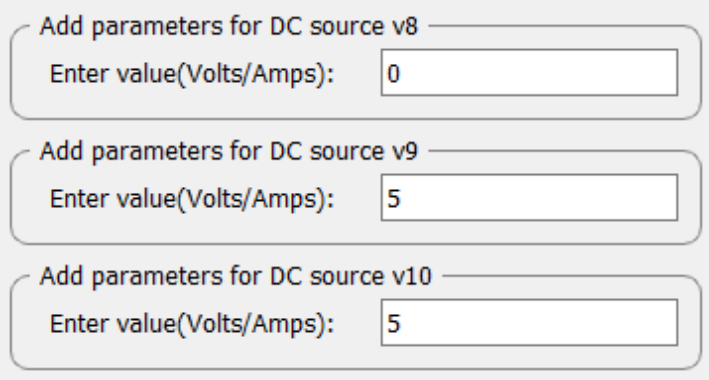

Figure 10c

Source/Reference(s):

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

Pages 58-61