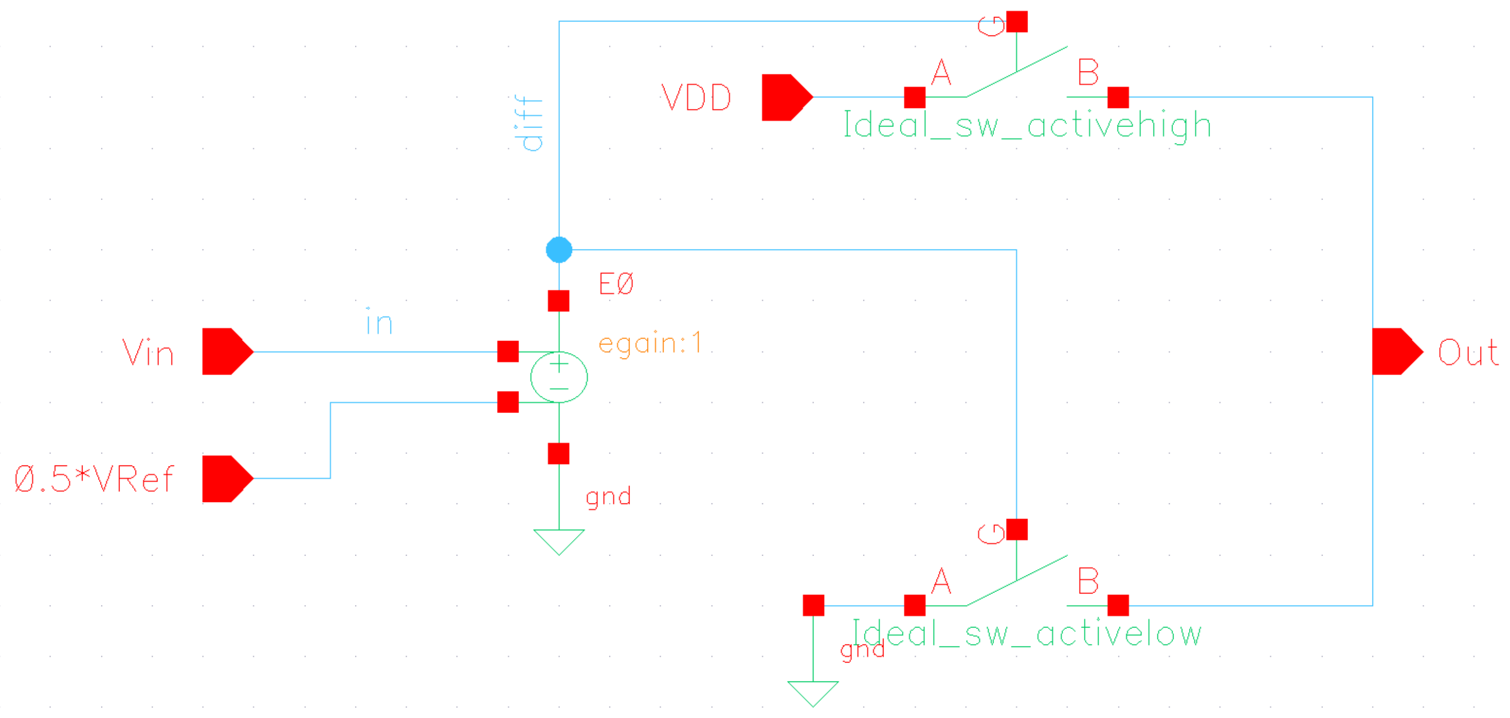
EE228 – HW2 Report
4-bit Flash ADC & DAC using ideal components & VerilogA

Muhammad Aldacher
Student ID: 011510317

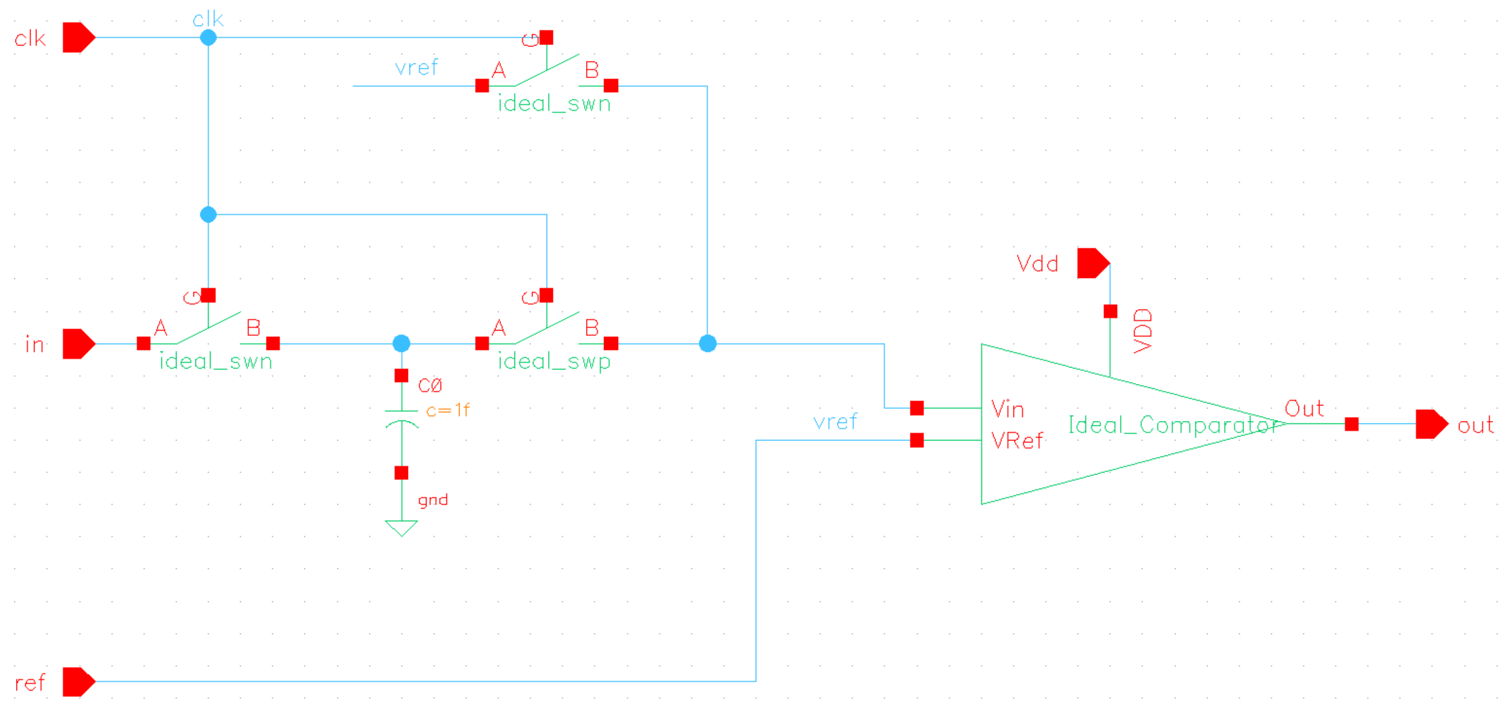
(1)
Using Ideal Components

Comparator

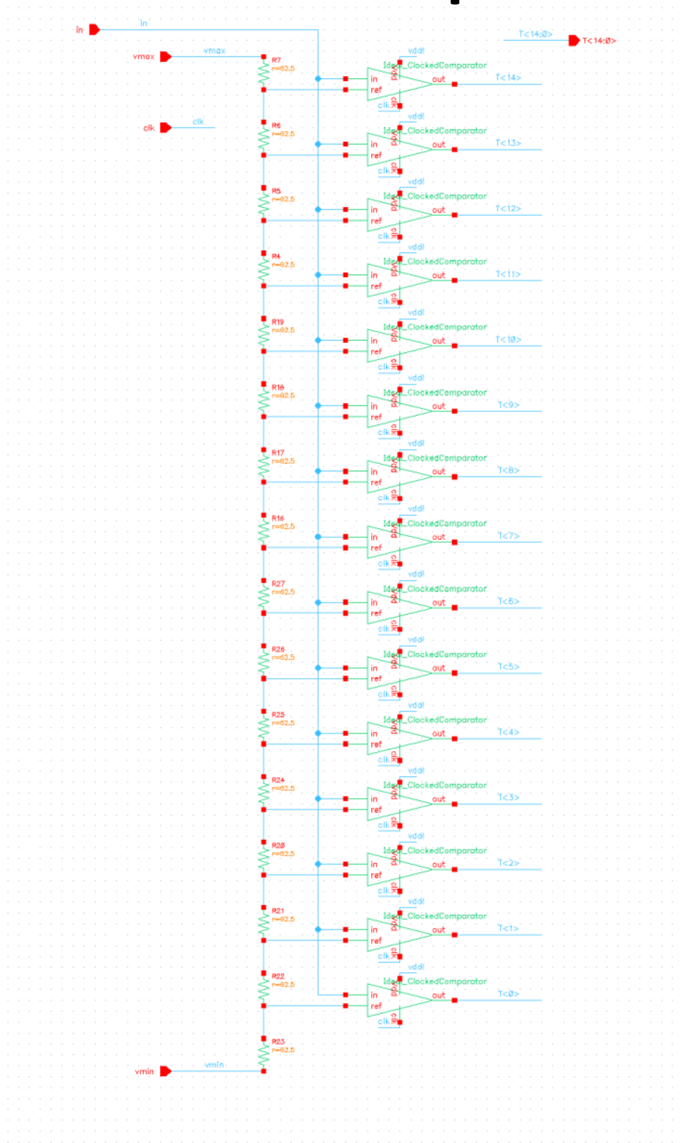


G
VDD
A
B
Ideal_sw_activehigh
diff
EØ
in
Vin
egain:1
Out
Ø.5*VRef
gnd
G
A
B
gnd
Ideal_sw_activelow

Clocked Comparator

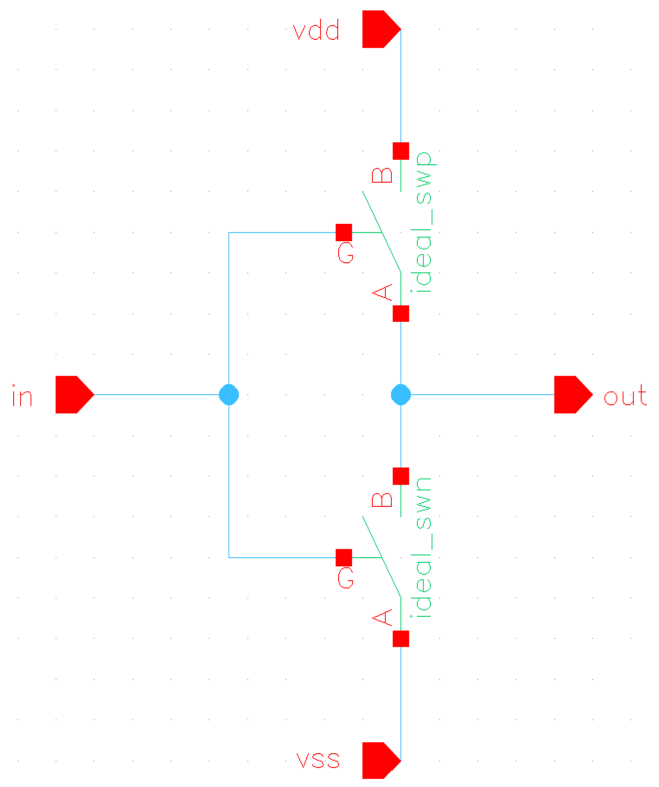


Resistive Ladder & Clocked Comparators

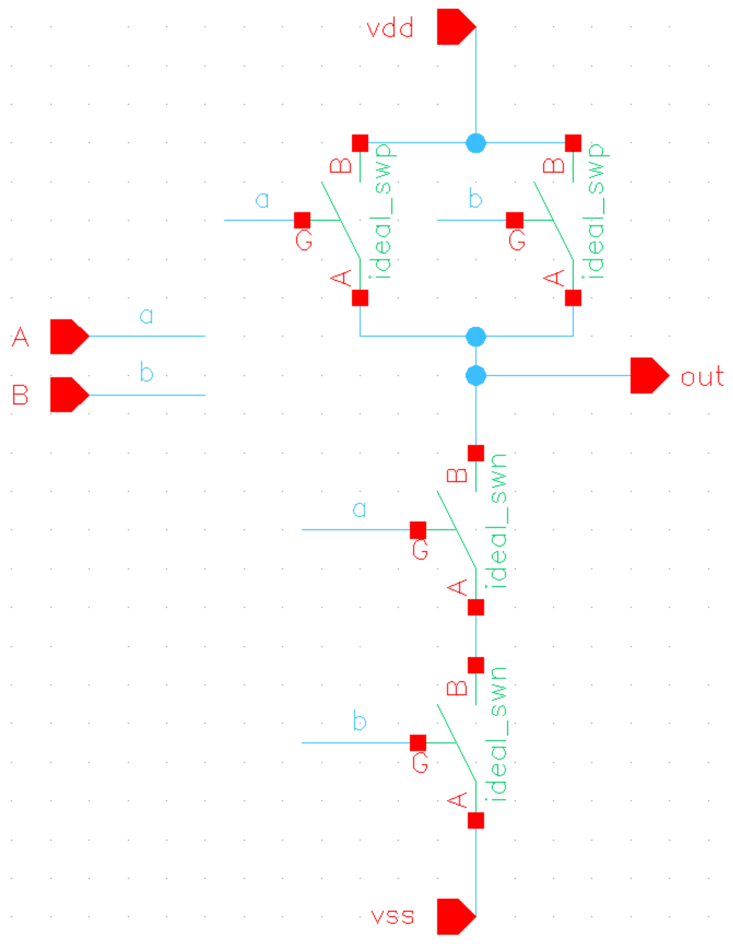


Logic Gates

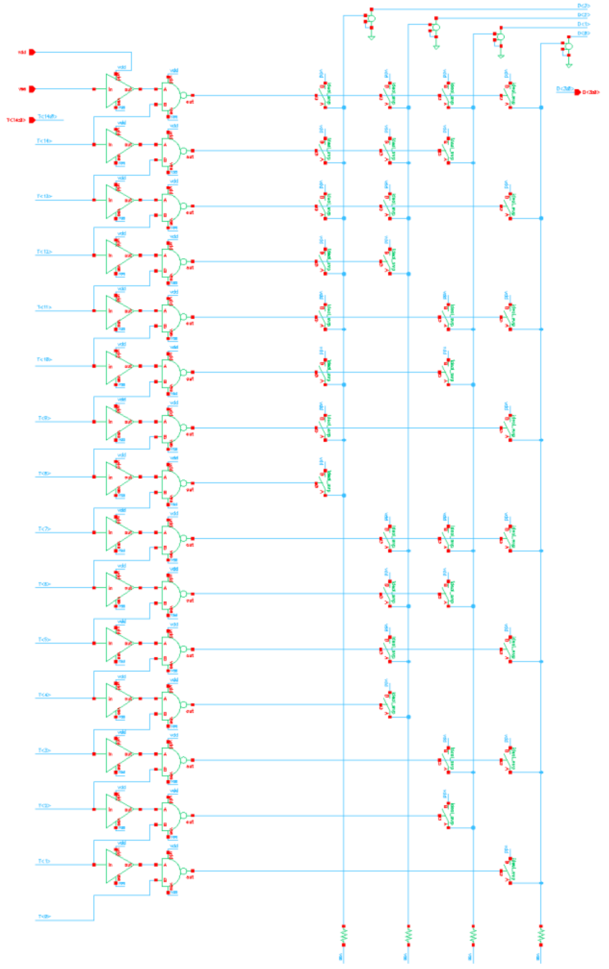
Inverter



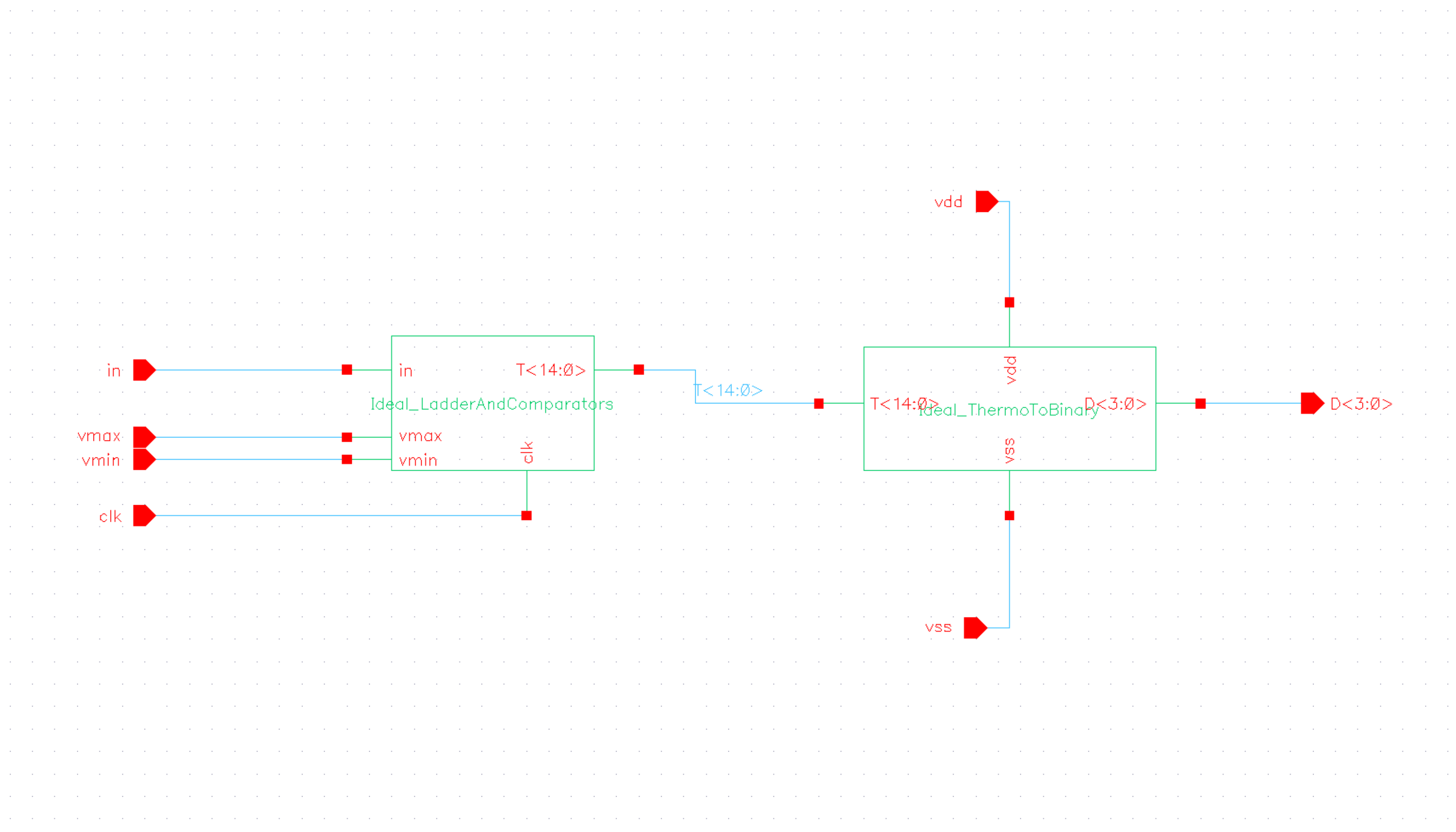
NAND



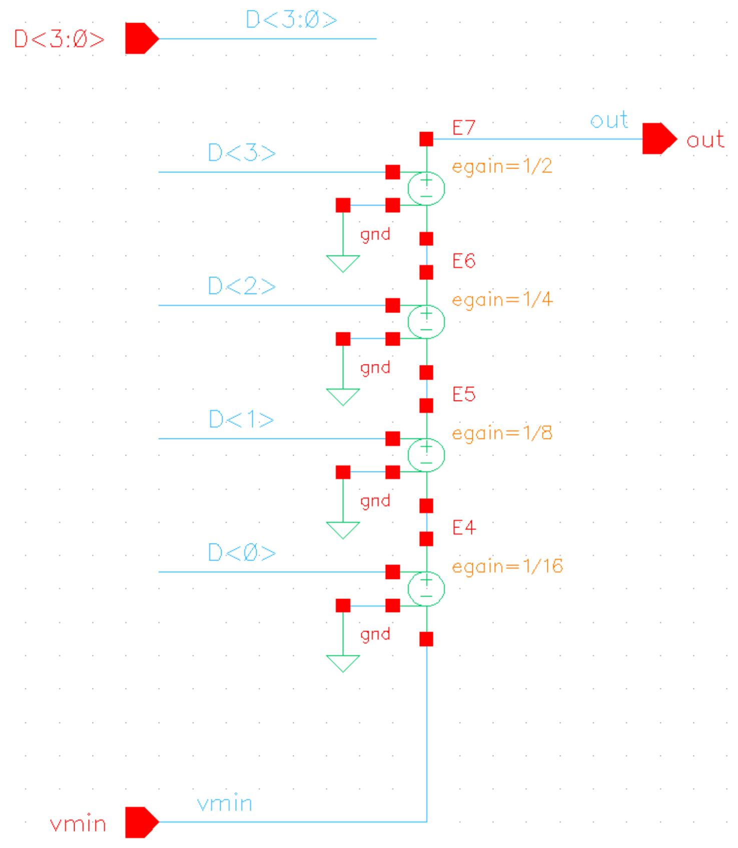
Thermometer-to-Binary Encoder (ROM Encoder)



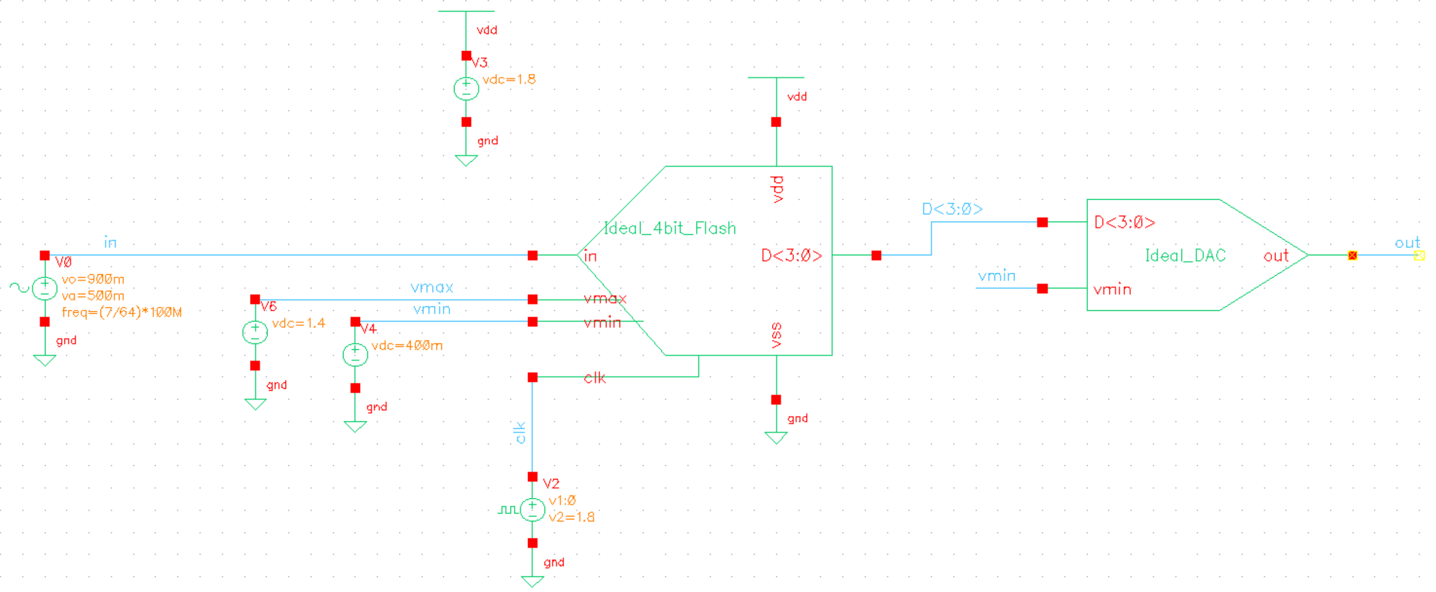
4-bit Flash ADC



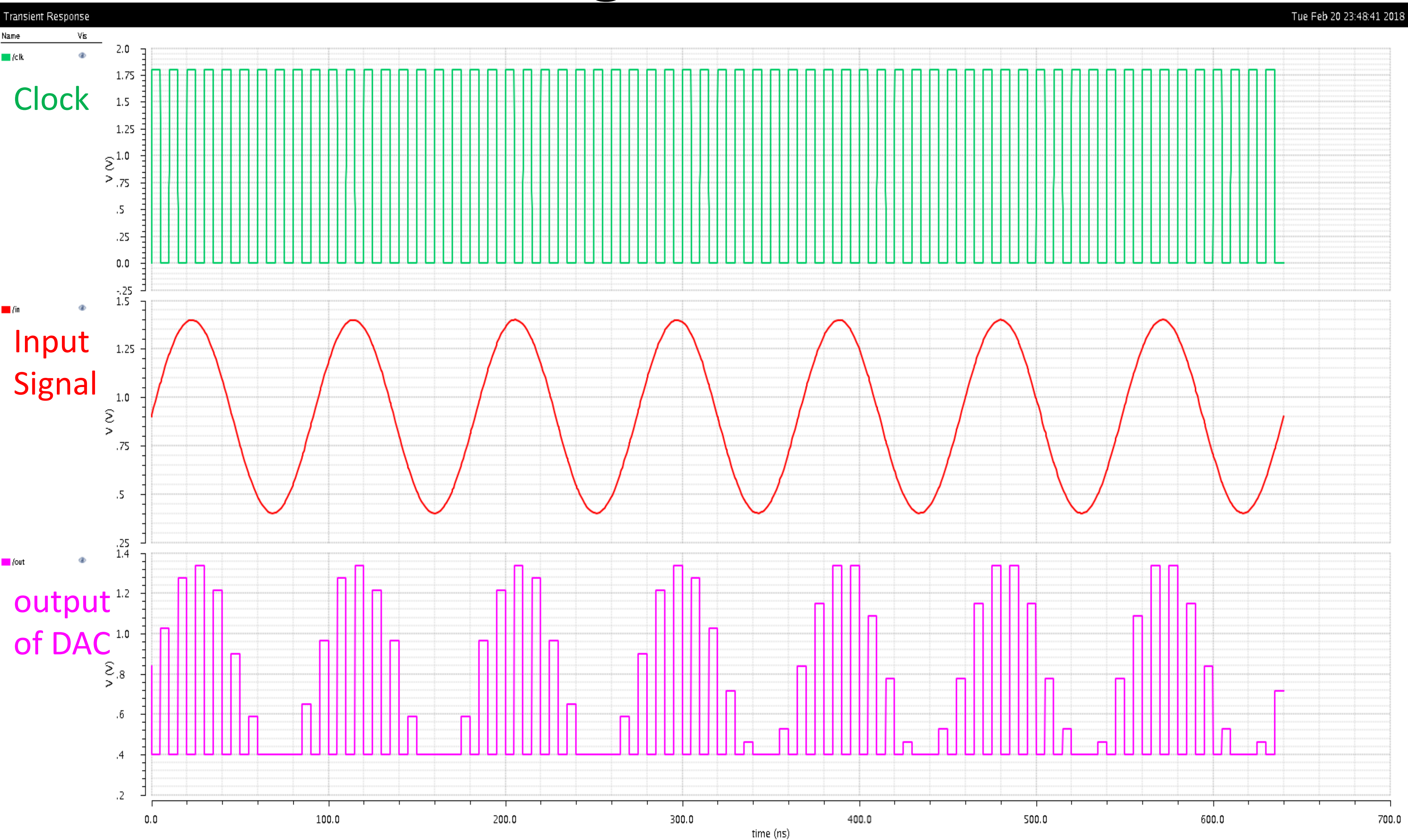
4-bit DAC


D<3:Ø>
D<3:Ø>
E7
out
out
egain=1/2
D<3>
gnd
E6
egain=1/4
D<2>
gnd
E5
egain=1/8
D<1>
gnd
E4
egain=1/16
D<Ø>
gnd
vmin
vmin

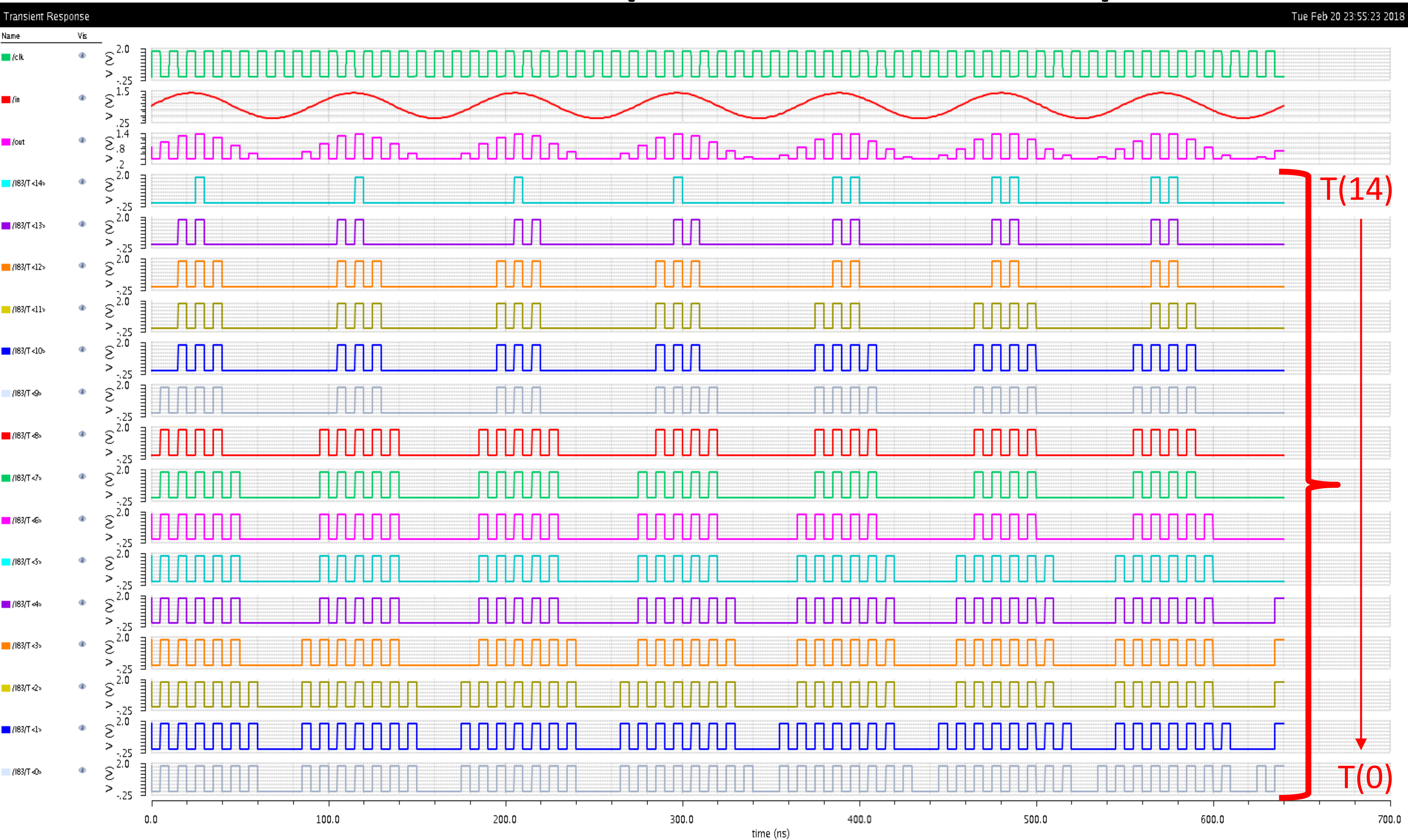
Testbench for the ADC & the DAC



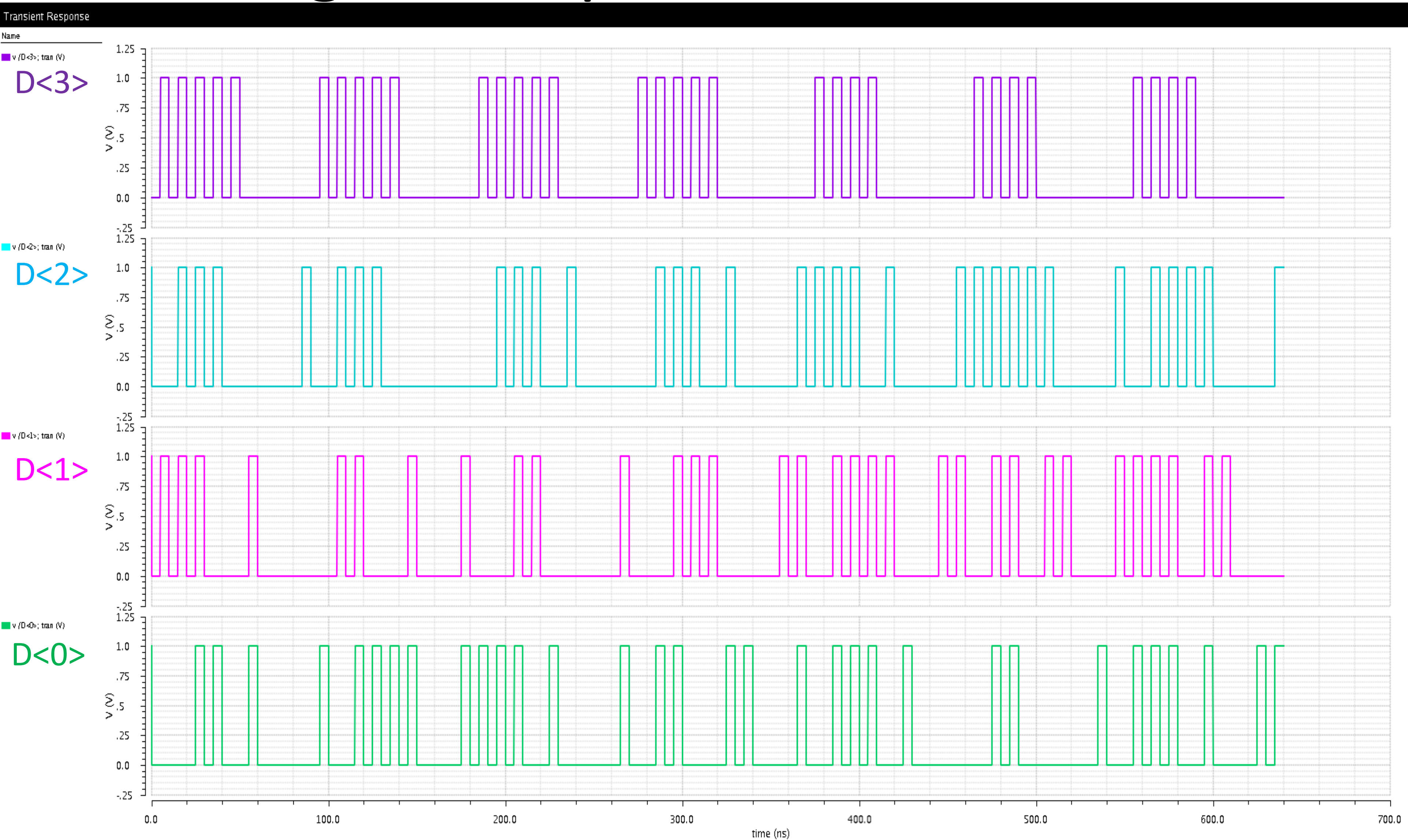
Test Results: Analog Waveforms



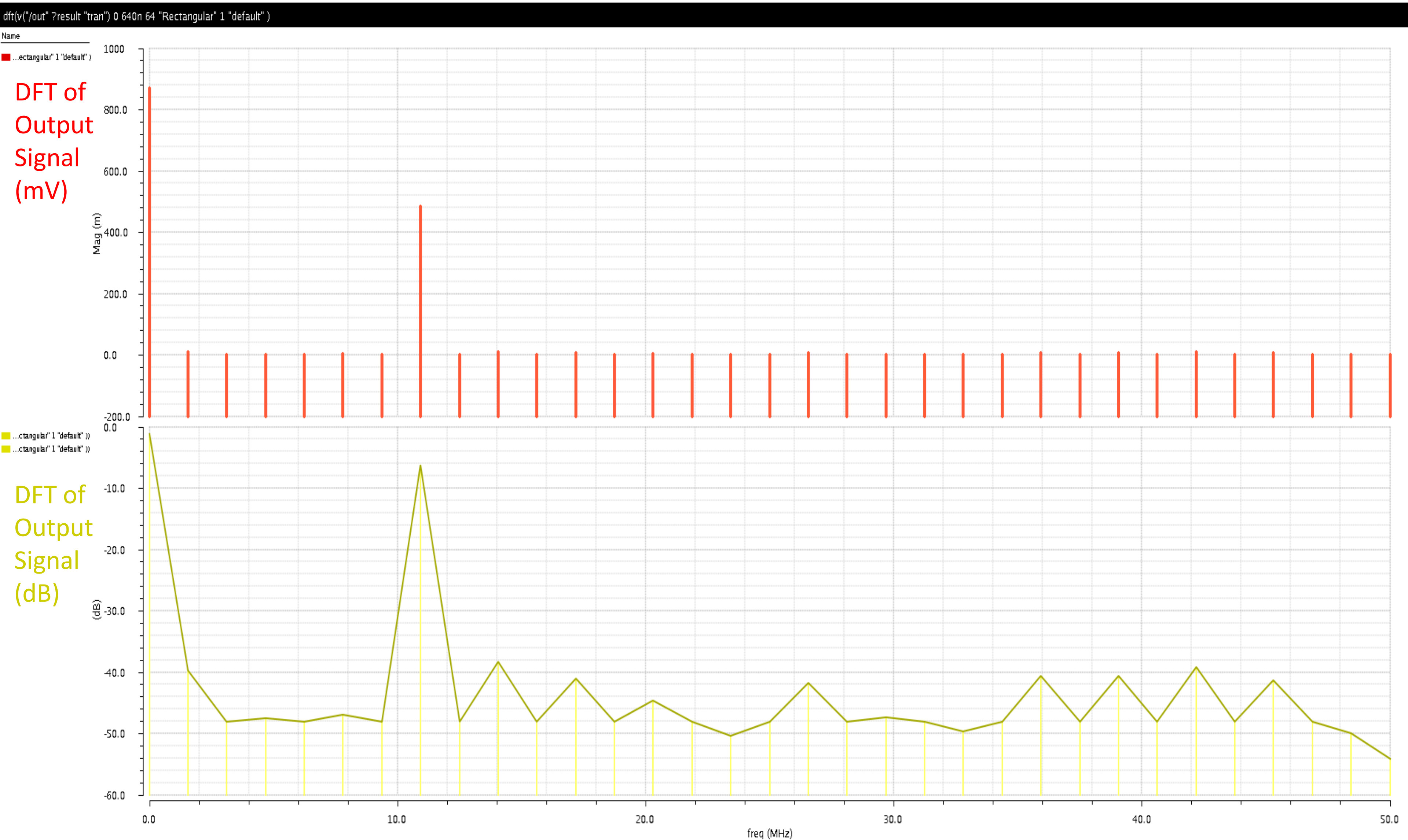
Test Results: Thermometer Outputs from Comparators



Test Results:
Digital Outputs from Encoder

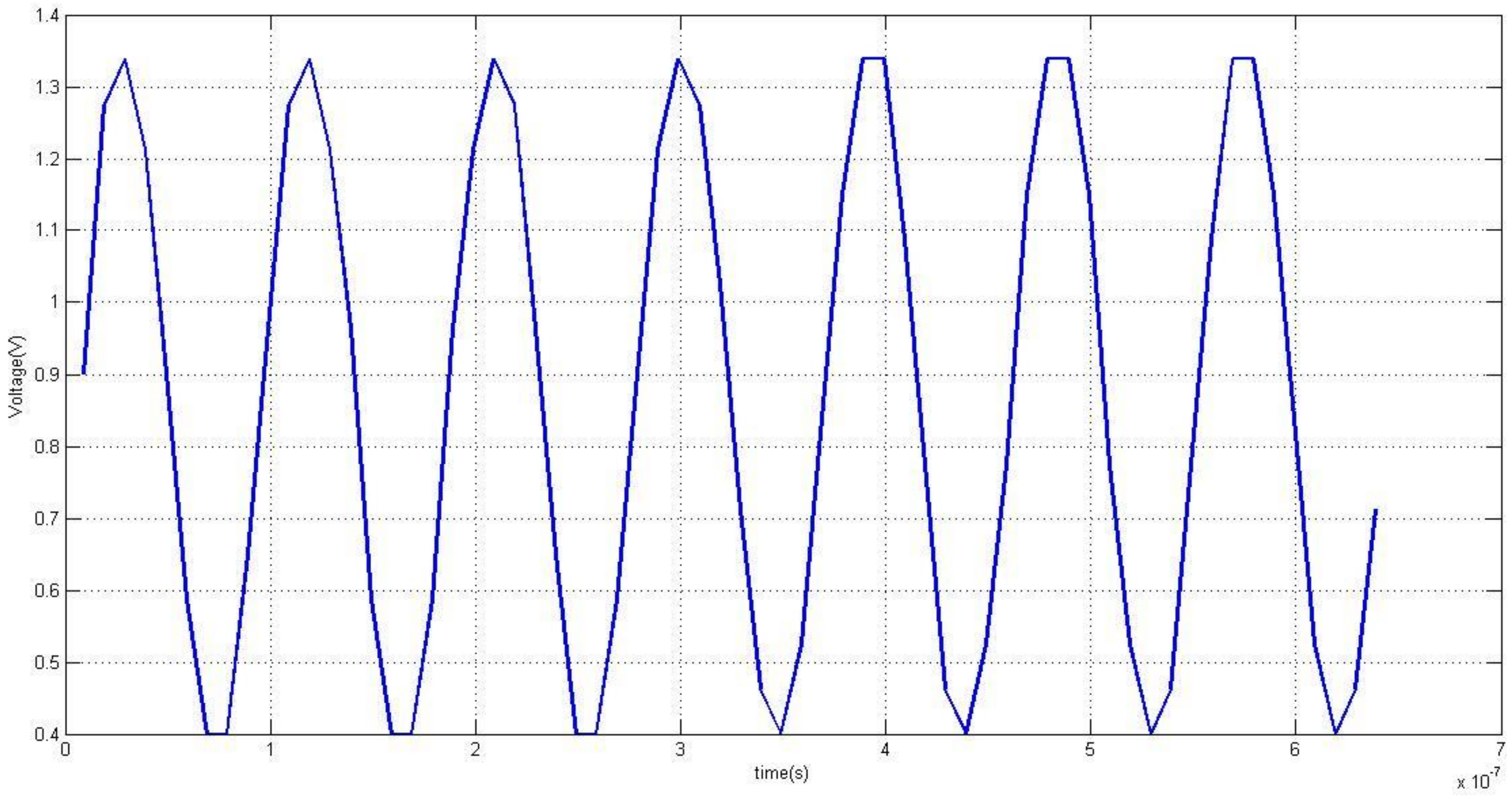


Test Results: DFT from Cadence

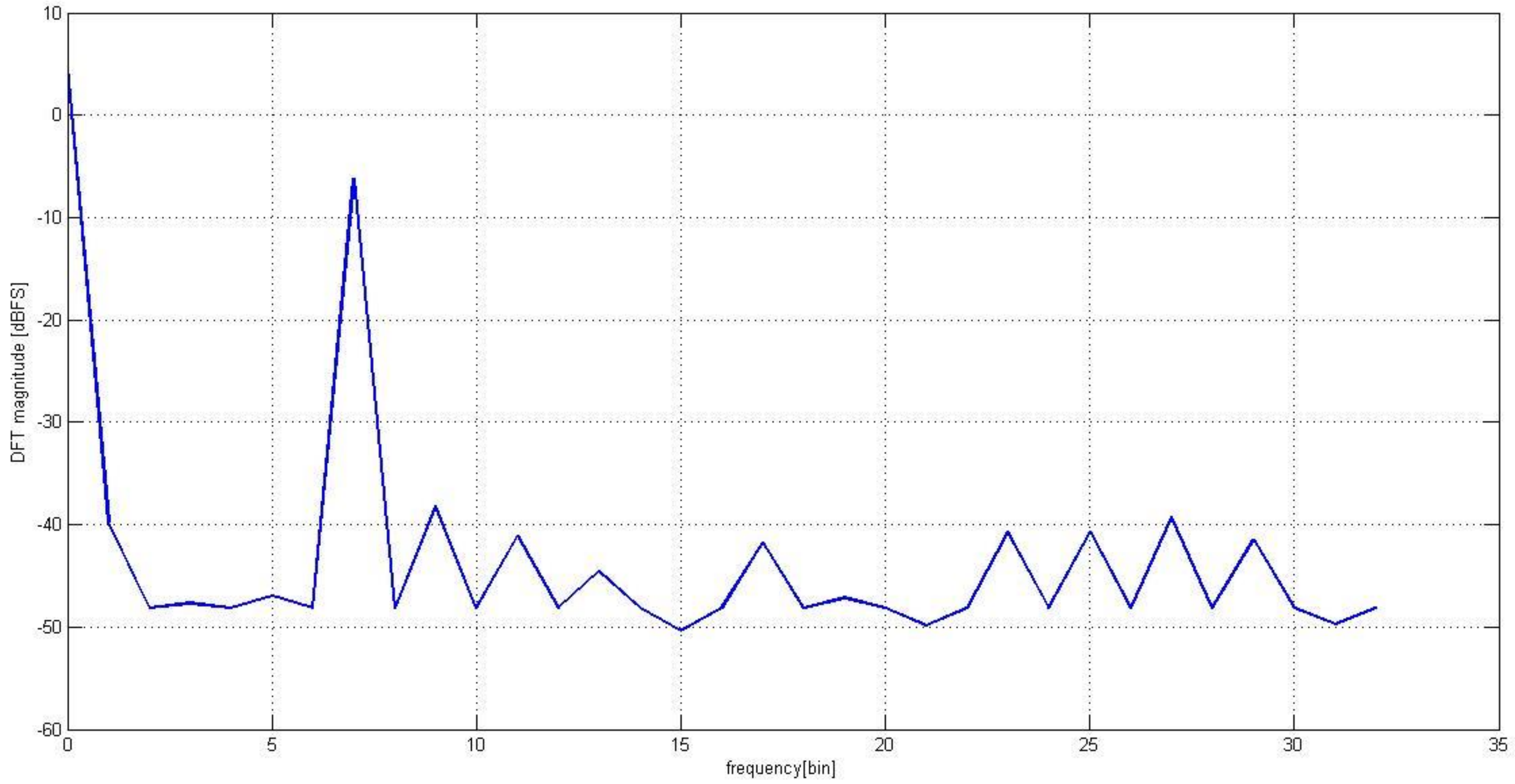


dft(v("/out" ?result "tran") 0 640n 64 "Rectangular" 1 "default")

Matlab: Output Waveform reconstructed from sampled data



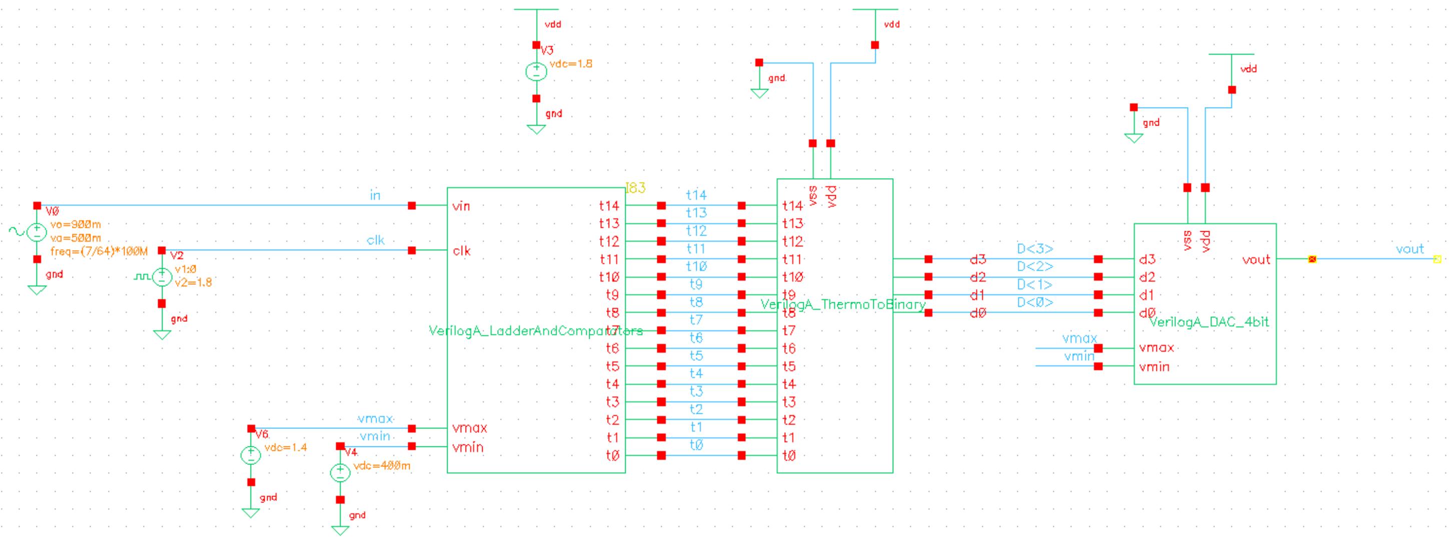
Matlab:
Output spectrum using DFT
(from N = 1→33)



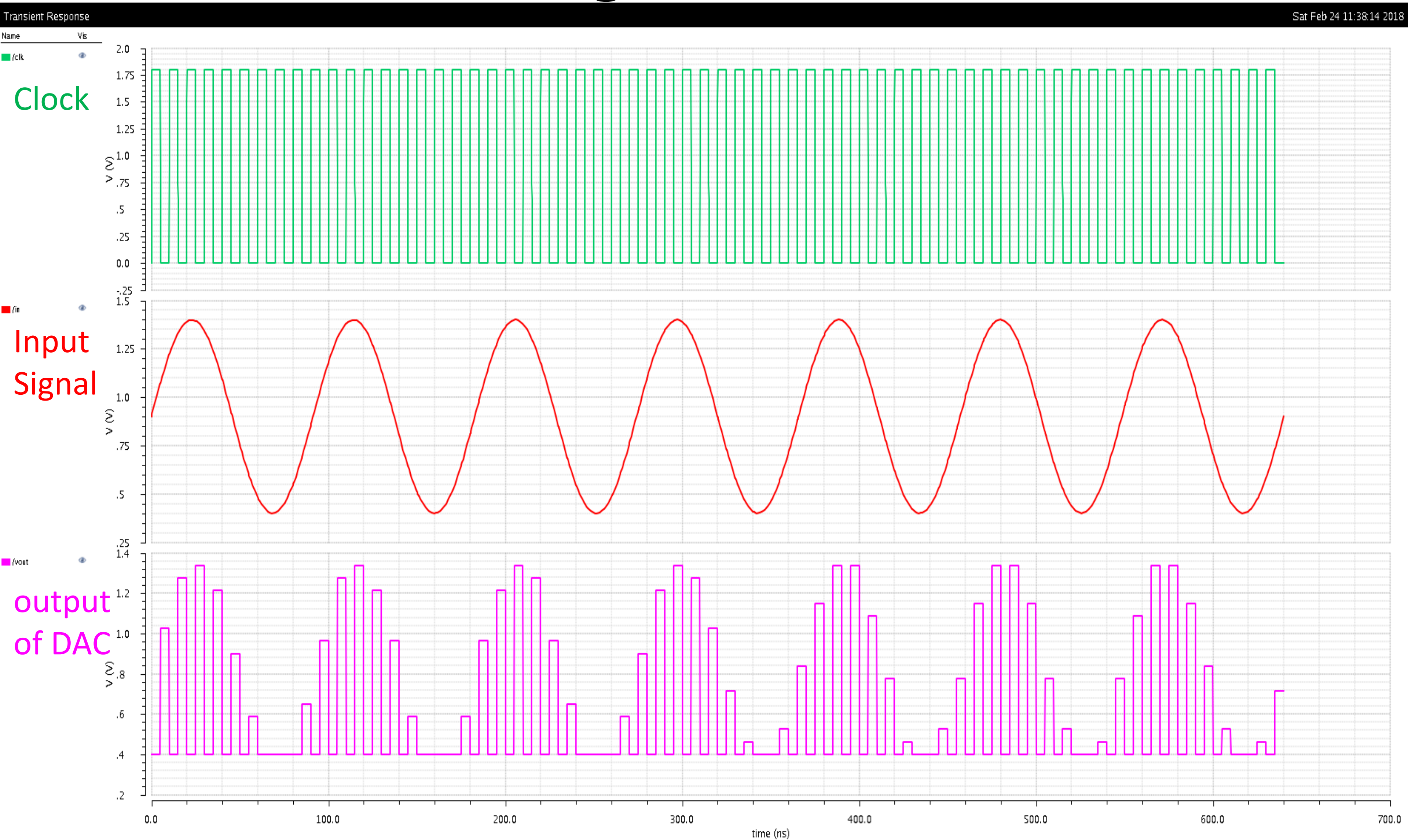
(2)
Using VerilogA Blocks

(Using a “Ladder & Comparators” VerilogA Block)

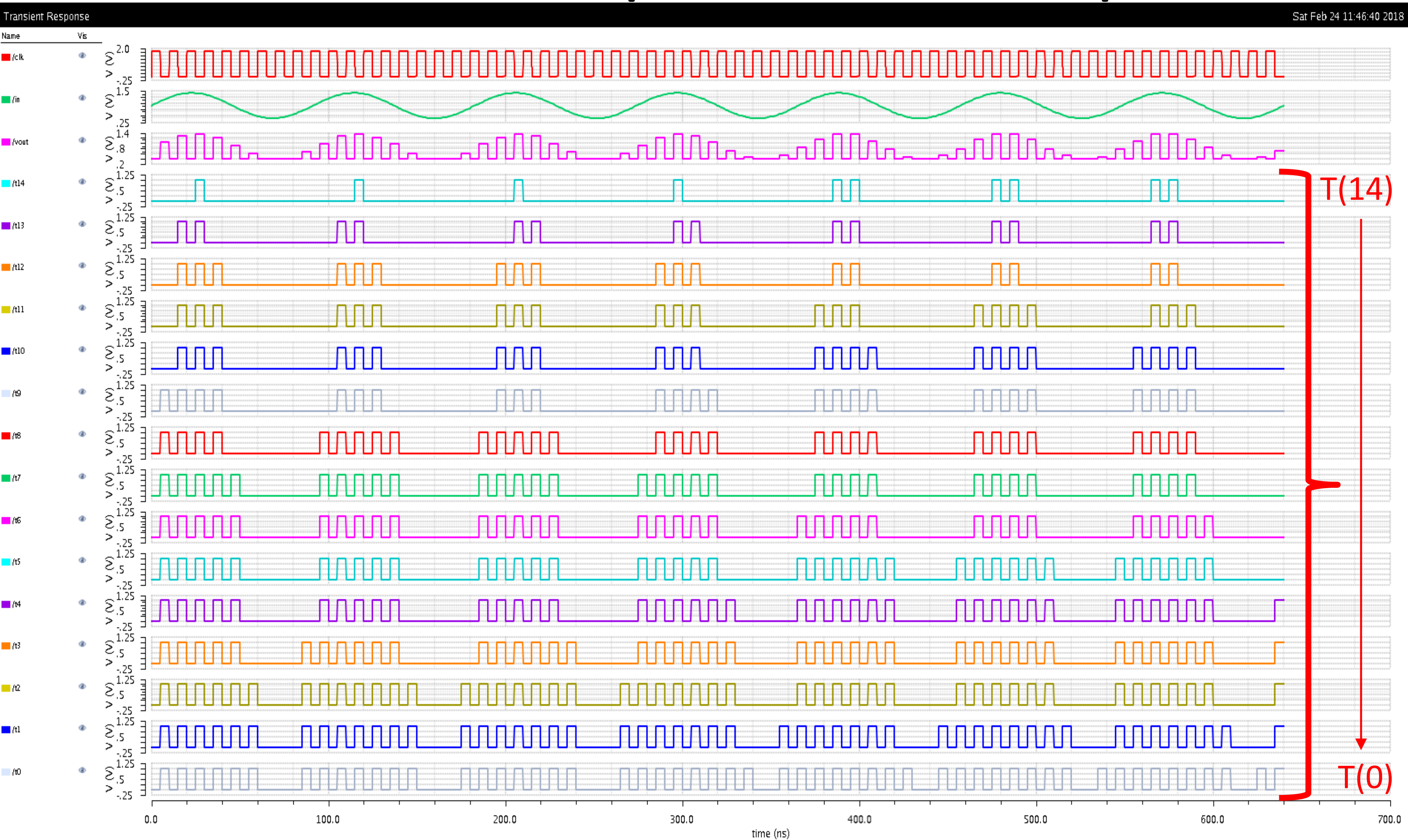
4-bit Flash ADC & DAC



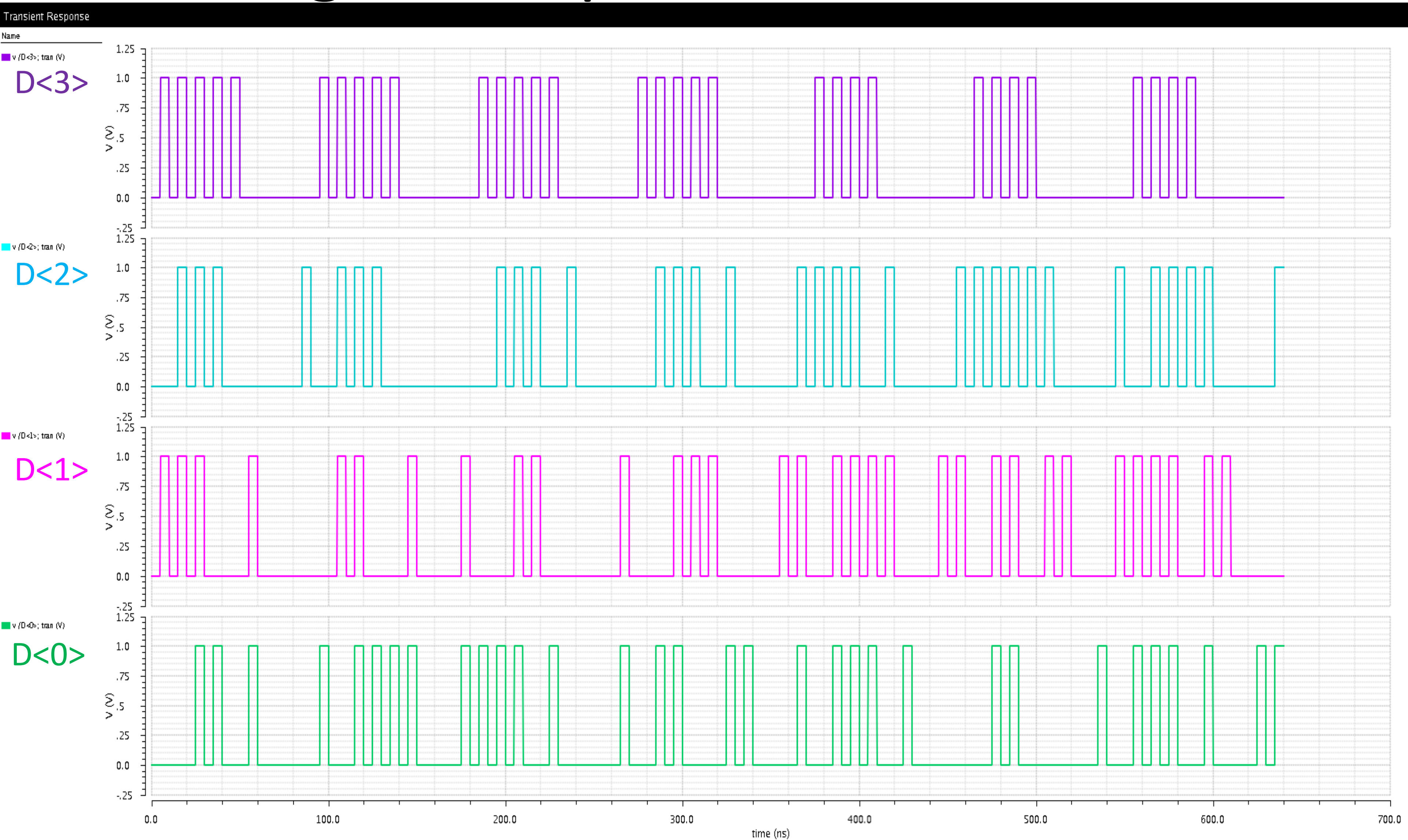
Test Results: Analog Waveforms



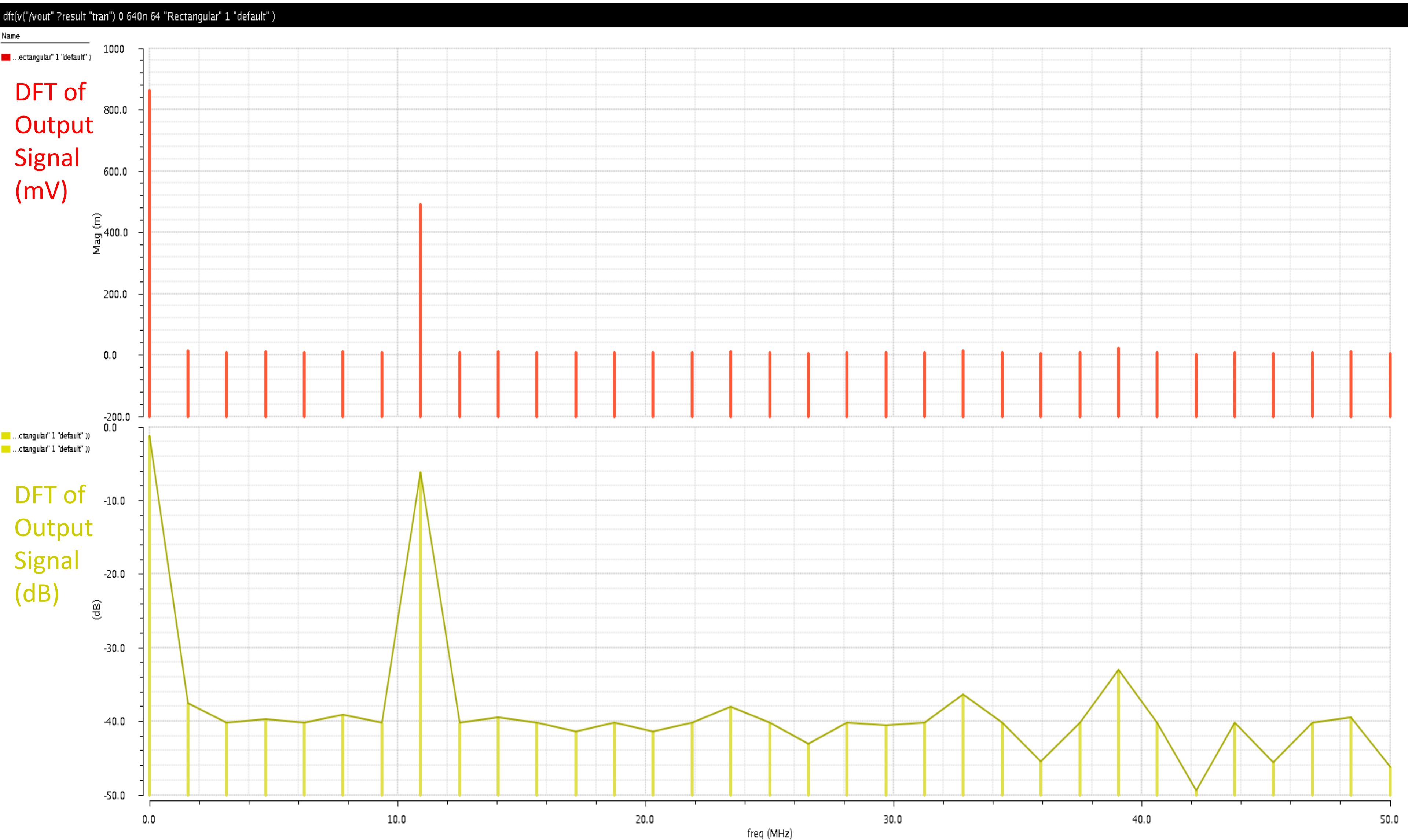
Test Results: Thermometer Outputs from Comparators



Test Results:
Digital Outputs from Encoder



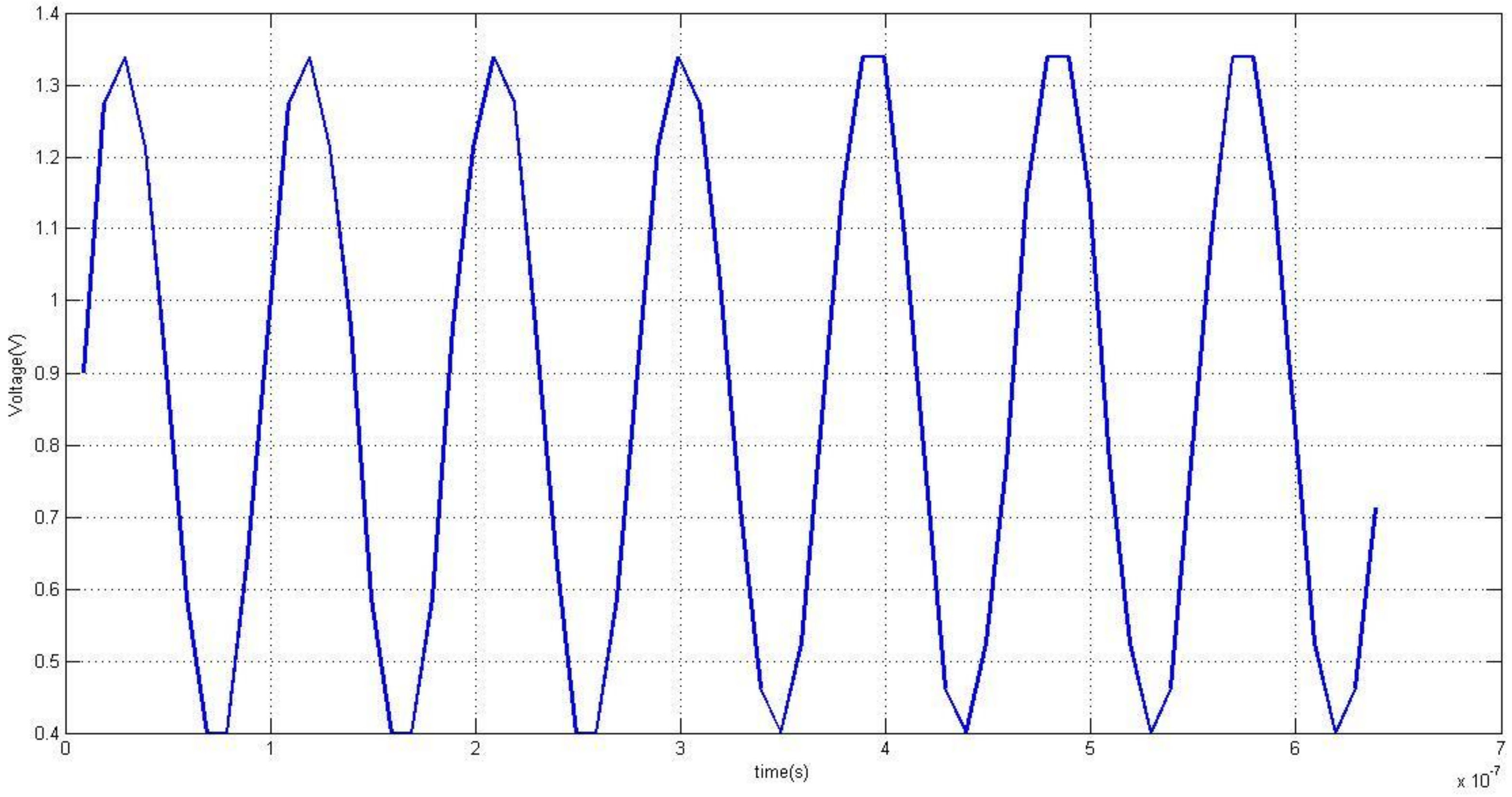
Test Results: DFT from Cadence



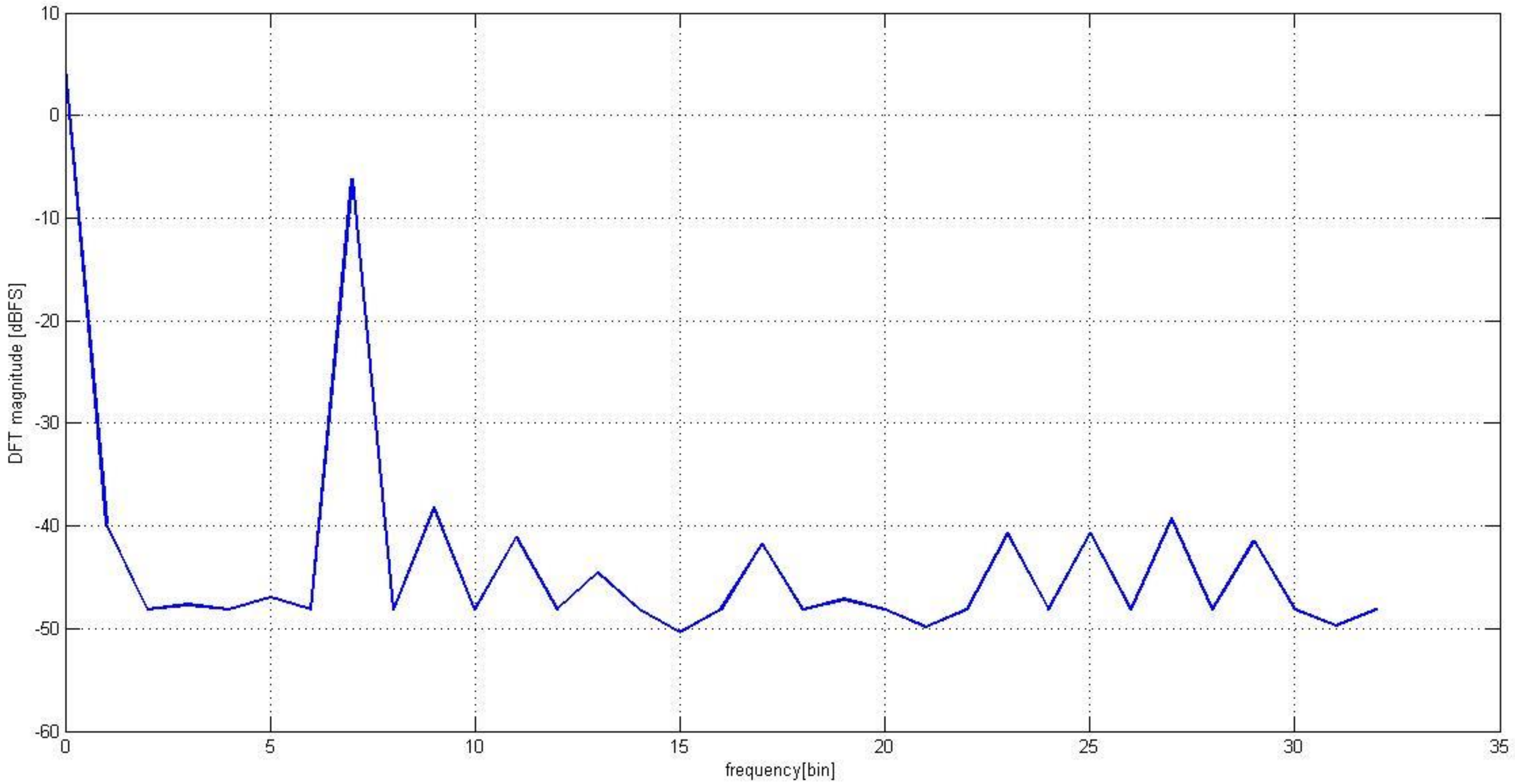
dft(v("/vout" ?result "tran") 0 640n 64 "Rectangular" 1 "default")

Name
🟥 ...ectangular" 1 "default")
DFT of Output Signal (mV)
Mag (m)
1000
800.0
600.0
400.0
200.0
0.0
-200.0
🟨 ...ctangular" 1 "default"))
🟨 ...ctangular" 1 "default"))
DFT of Output Signal (dB)
(dB)
0.0
-10.0
-20.0
-30.0
-40.0
-50.0
0.0
10.0
20.0
30.0
40.0
50.0
freq (MHz)

Matlab: Output Waveform reconstructed from sampled data



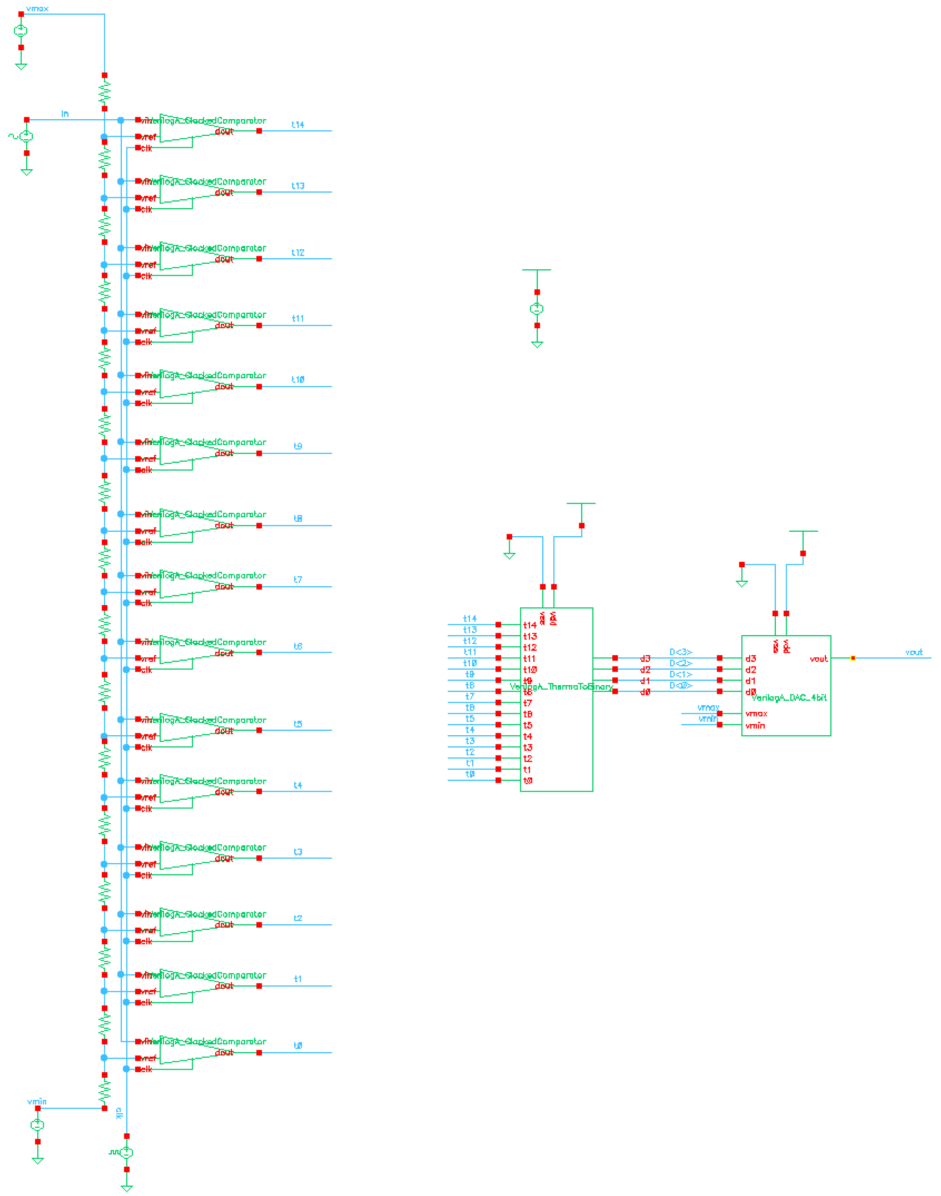
Matlab:
Output spectrum using DFT
(from N = 1→33)



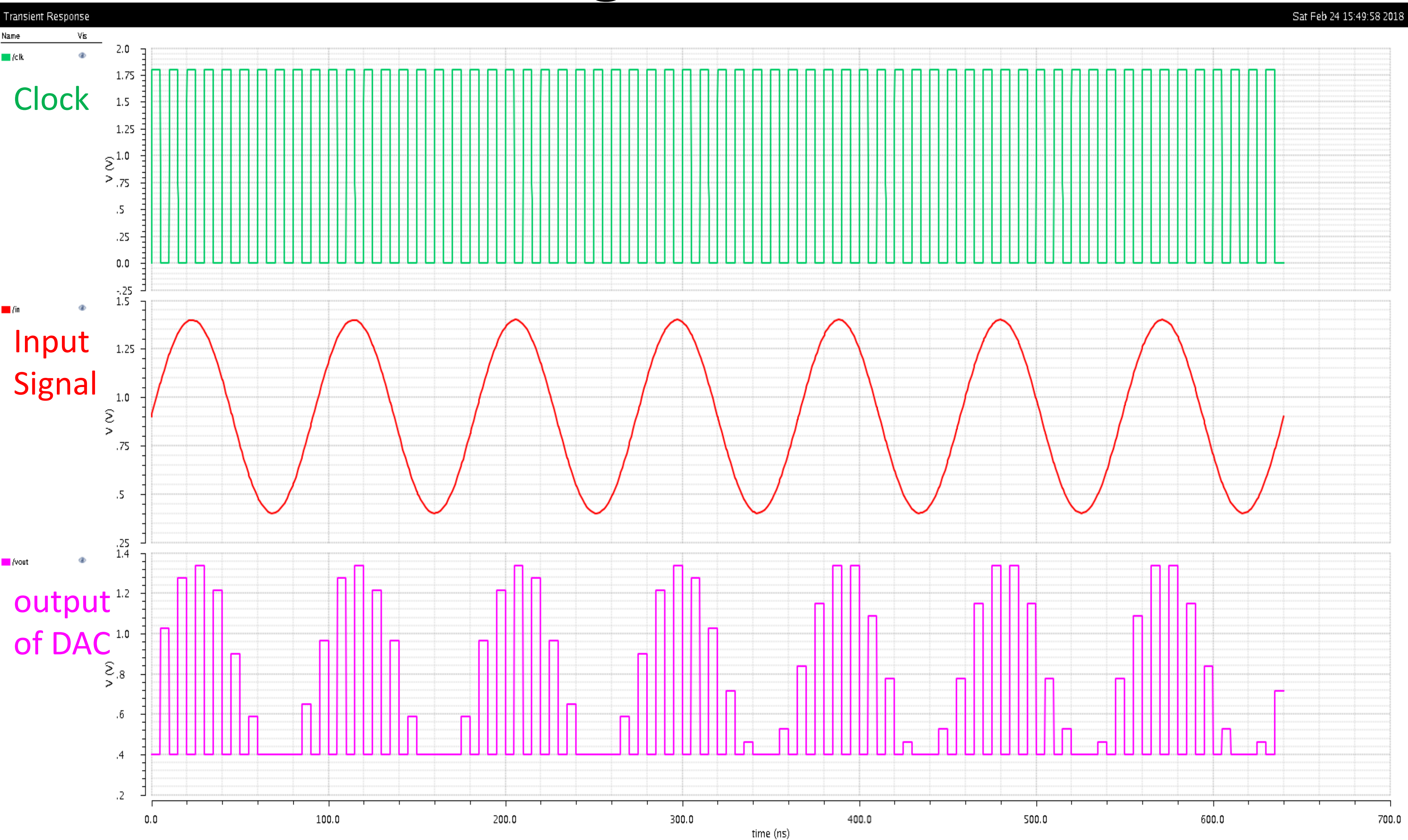
(3)
Using VerilogA Blocks

(Using “Clocked Comparators” VerilogA Blocks)

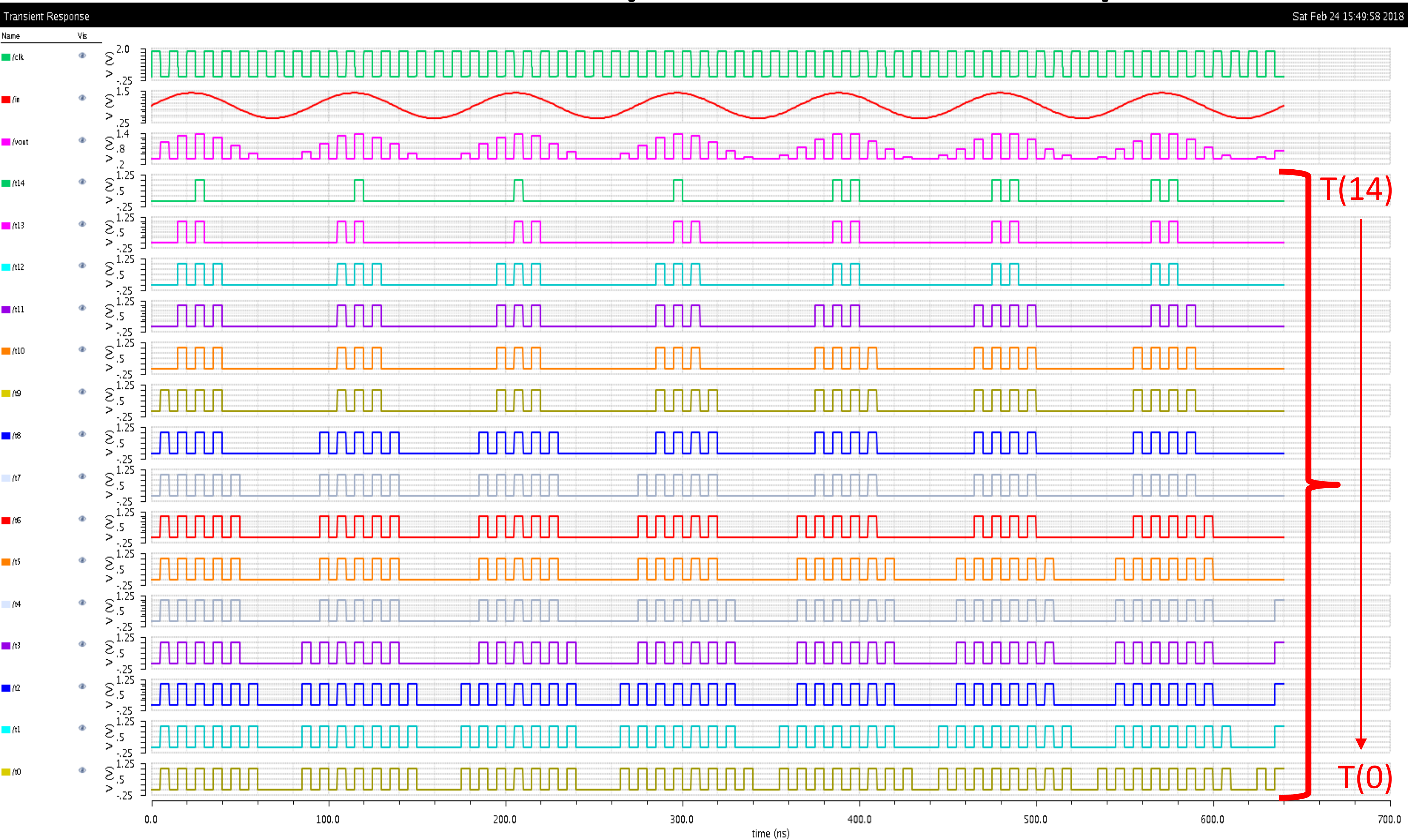
4-bit Flash ADC & DAC



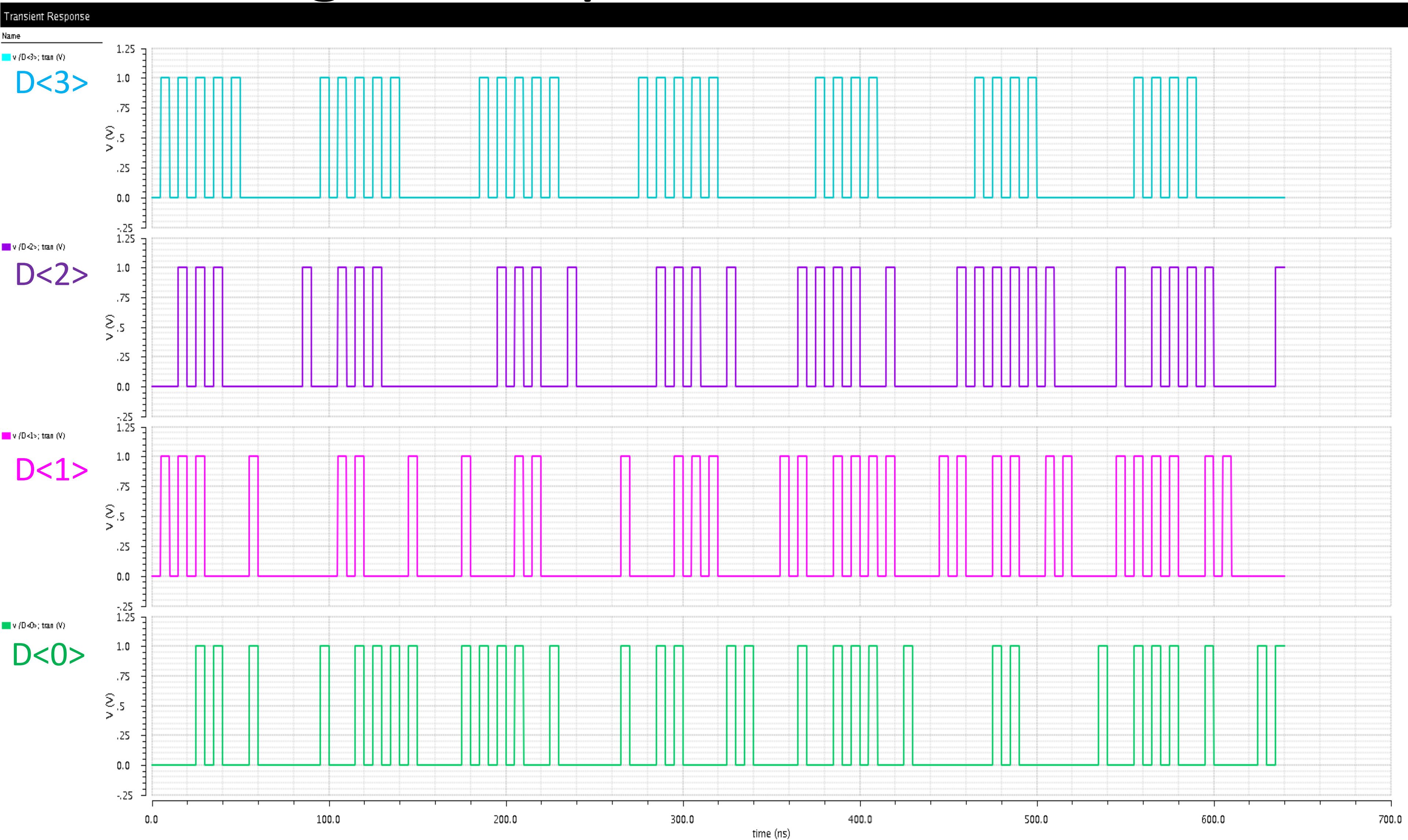
Test Results: Analog Waveforms



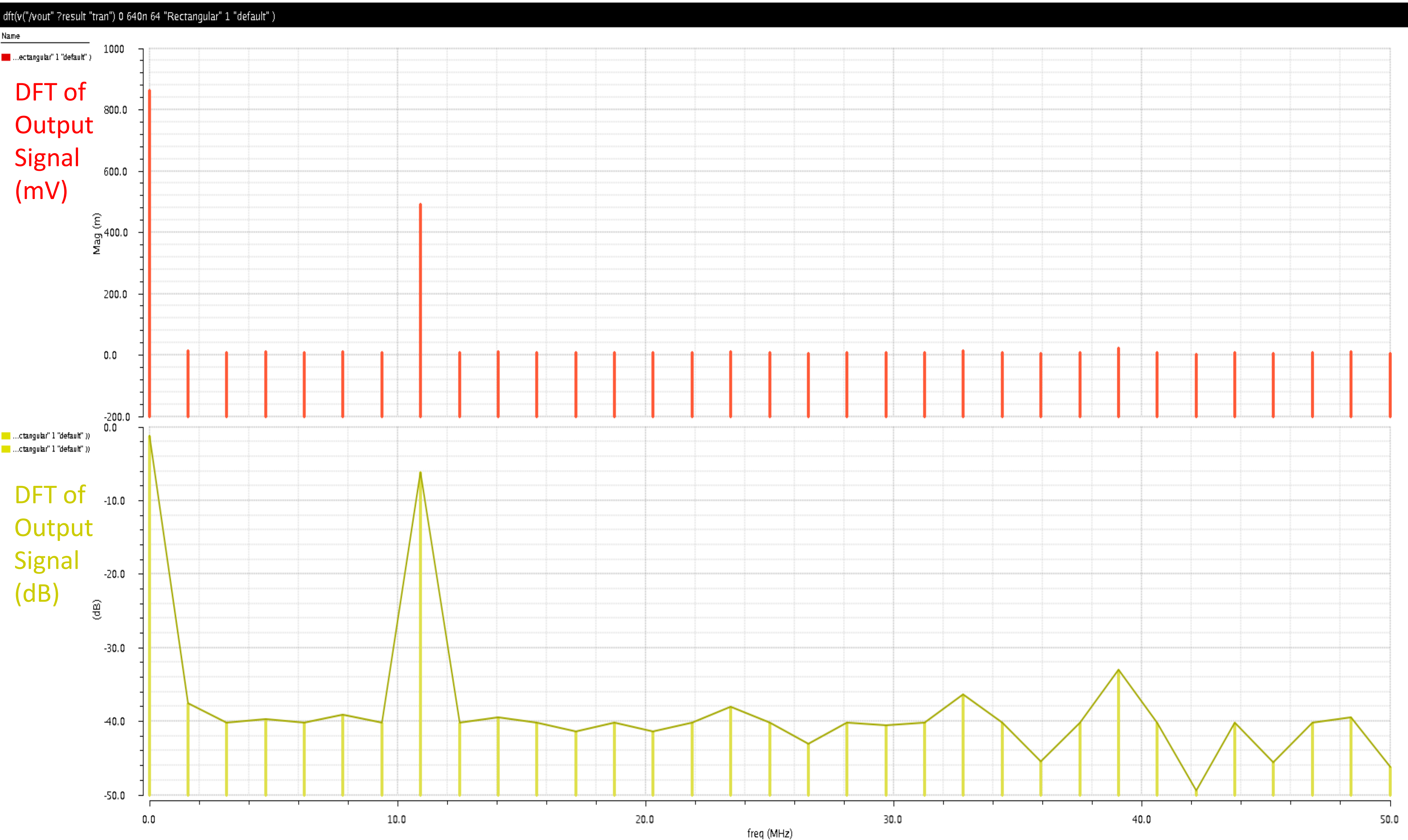
Test Results: Thermometer Outputs from Comparators



Test Results: Digital Outputs from Encoder

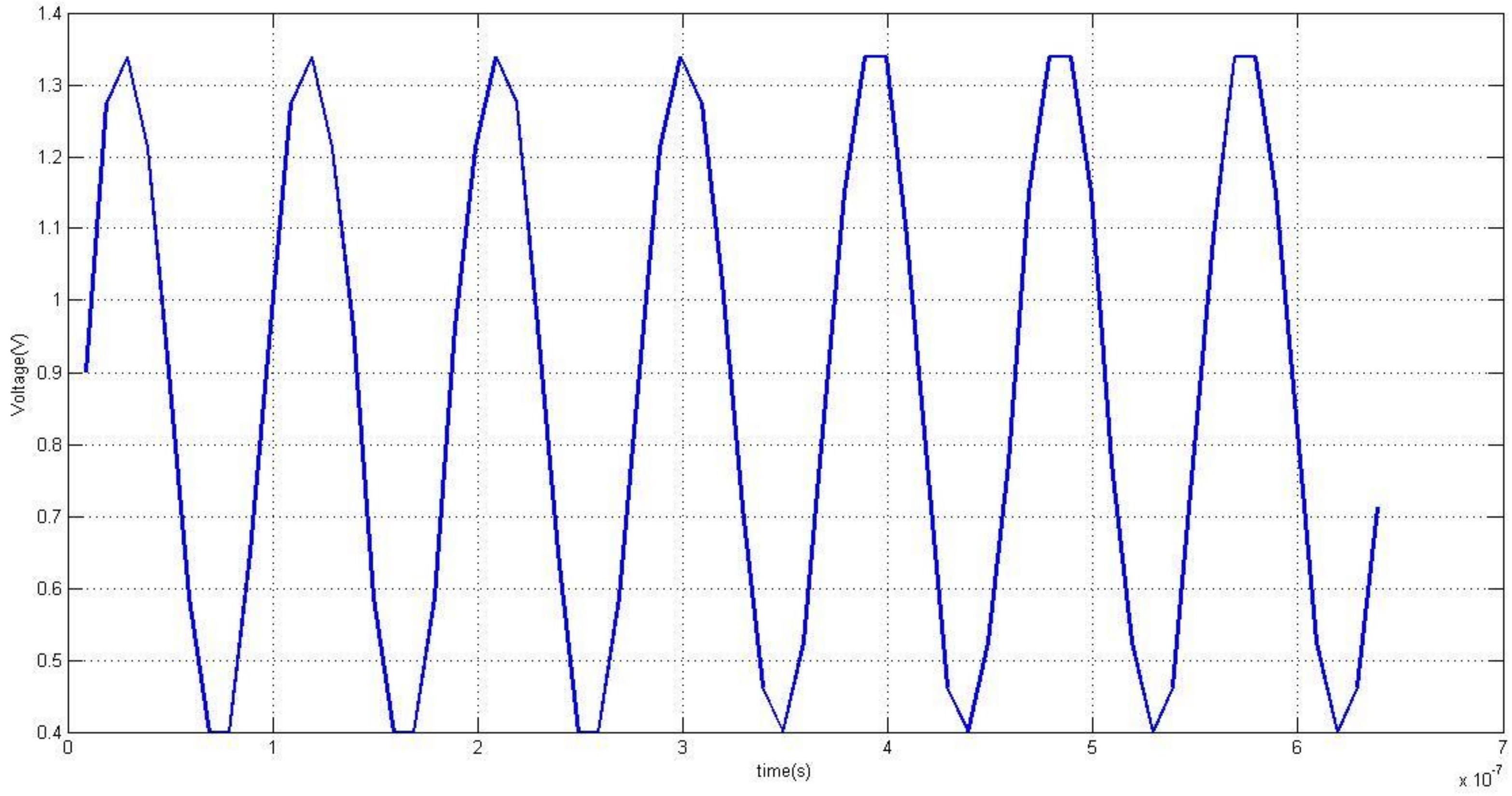


Test Results: DFT from Cadence

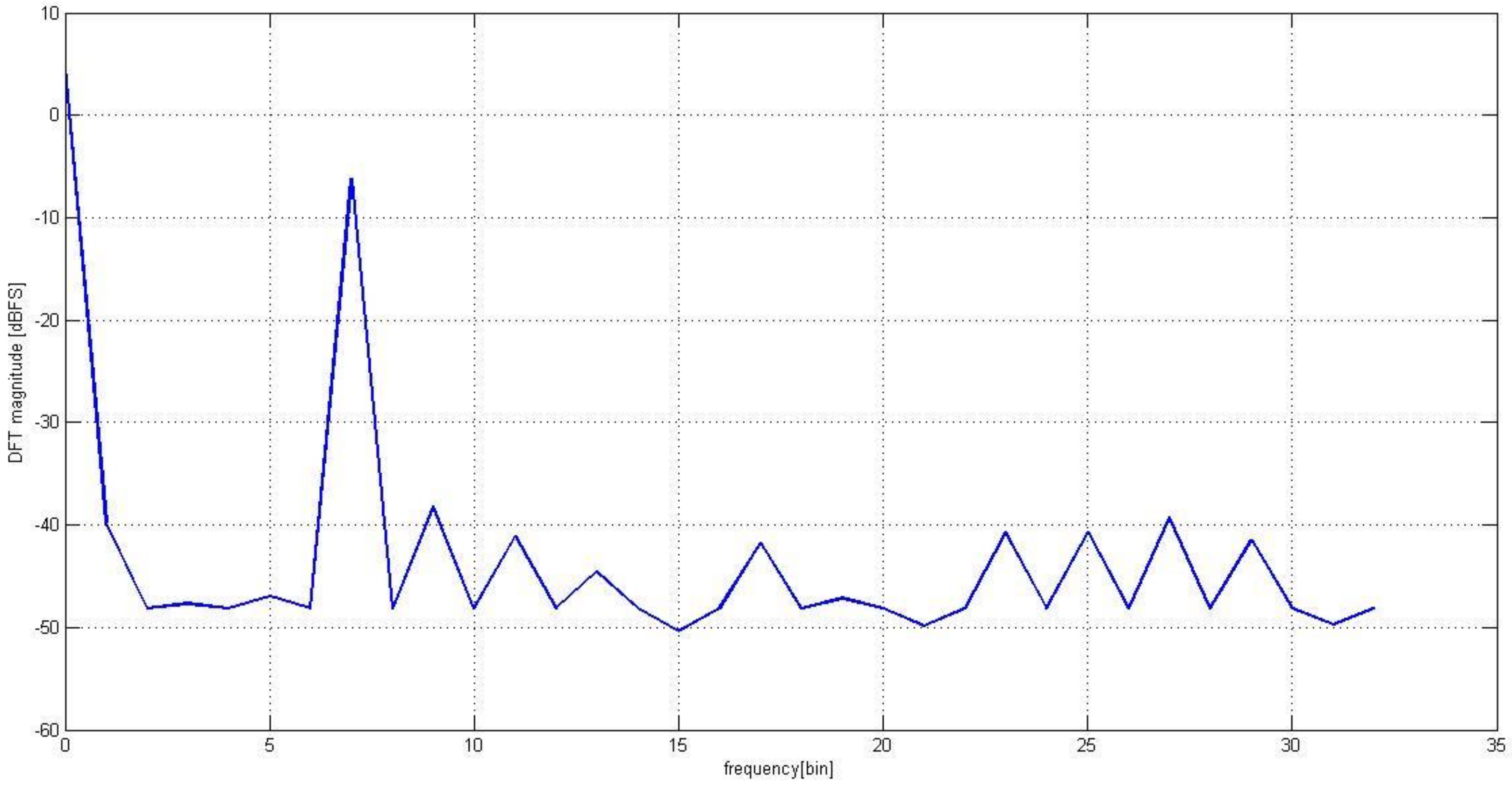


```
dft(v("/vout" ?result "tran") 0 640n 64 "Rectangular" 1 "default" )
```

Matlab: Output Waveform reconstructed from sampled data



Matlab:
Output spectrum using DFT
(from N = 1→33)



➔

Matlab & VerilogA Codes

VerilogA: Clocked Comparator

```
// VerilogA for ADC_Ideal_4bit_FlashADC, VerilogA_ClockedComparator, veriloga

`include "constants.vams"
`include "disciplines.vams"

module VerilogA_ClockedComparator(dout,vref,vin,clk);

parameter real clk_th=0.9;
parameter real delay = 0;
parameter real ttime = 1p;

input vin,vref,clk;
output dout;

electrical dout,vref,vin,clk;
real d_result;

analog begin

		@(cross(V(clk) - clk_th, -1)) begin
			if(V(vin) > V(vref)) begin
				d_result = 1;
			end
			else begin
				d_result = 0;
			end
		end

		@(cross(V(clk) - clk_th, +1)) begin
			d_result = 0;
		end


		V(dout) <+ transition(d_result,delay,ttime);
end
endmodule
```

VerilogA: Resistive Ladder & Comparators

{1}

```
// VerilogA for ADC_Ideal_4bit_FlashADC, VerilogA_LadderAndComparators, veriloga

`include "constants.vams"
`include "disciplines.vams"

module
VerilogA_LadderAndComparators(vin,clk,vmax,vmin,t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t1
2,t13,t14);

parameter real vtrans=0.5;
parameter real delay = 0;
parameter real ttime = 1p;
parameter real clk_threshold = 0.9;                    //vdd is 1.8v

input clk,vin,vmin,vmax;
output t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14;

electrical vin,vmin,vmax,clk;
electrical t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14;
real step,t_0,t_1,t_2,t_3,t_4,t_5,t_6,t_7,t_8,t_9,t_10,t_11,t_12,t_13,t_14;

analog begin

        step = (V(vmax)-V(vmin))/16;

        // Sampling Phase (-1 is for falling edge)
        @(cross(V(clk) - clk_threshold, -1))
        begin
                if (V(vin) > step+V(vmin))
                        t_0 = 1;
                else
                        t_0 = 0;

                if (V(vin) > ((2*step)+V(vmin)))
                        t_1 = 1;
                else
                        t_1 = 0;

                if (V(vin) > ((3*step)+V(vmin)))
                        t_2 = 1;
                else
                        t_2 = 0;

                if (V(vin) > ((4*step)+V(vmin)))
                        t_3 = 1;
                else
                        t_3 = 0;

                if (V(vin) > ((5*step)+V(vmin)))
                        t_4 = 1;
                else
                        t_4 = 0;

                if (V(vin) > ((6*step)+V(vmin)))
                        t_5 = 1;
                else
                        t_5 = 0;

                if (V(vin) > ((7*step)+V(vmin)))
                        t_6 = 1;
                else
                        t_6 = 0;

                if (V(vin) > ((8*step)+V(vmin)))
                        t_7 = 1;
                else
                        t_7 = 0;

                if (V(vin) > ((9*step)+V(vmin)))
                        t_8 = 1;
                else
                        t_8 = 0;
```

VerilogA: Resistive Ladder & Comparators

{2} Cont'd

```
		if (V(vin) > ((10*step)+V(vmin)))
				t_9 = 1;
		else
				t_9 = 0;

		if (V(vin) > ((11*step)+V(vmin)))
				t_10 = 1;
		else
				t_10 = 0;

		if (V(vin) > ((12*step)+V(vmin)))
				t_11 = 1;
		else
				t_11 = 0;

		if (V(vin) > ((13*step)+V(vmin)))
				t_12 = 1;
		else
				t_12 = 0;

		if (V(vin) > ((14*step)+V(vmin)))
				t_13 = 1;
		else
				t_13 = 0;

		if (V(vin) > ((15*step)+V(vmin)))
				t_14 = 1;
		else
				t_14 = 0;
	end
```

```
	@(cross(V(clk) - clk_threshold, +1))
	begin
	t_0 = 0;		t_1 = 0;		t_2 = 0;
	t_3 = 0;		t_4 = 0;		t_5 = 0;
	t_6 = 0;		t_7 = 0;		t_8 = 0;
	t_9 = 0;		t_10 = 0;		t_11 = 0;
	t_12 = 0;		t_13 = 0;		t_14 = 0;
	end

	V(t0) <+ transition(t_0,delay,ttime);
	V(t1) <+ transition(t_1,delay,ttime);
	V(t2) <+ transition(t_2,delay,ttime);
	V(t3) <+ transition(t_3,delay,ttime);
	V(t4) <+ transition(t_4,delay,ttime);
	V(t5) <+ transition(t_5,delay,ttime);
	V(t6) <+ transition(t_6,delay,ttime);
	V(t7) <+ transition(t_7,delay,ttime);
	V(t8) <+ transition(t_8,delay,ttime);
	V(t9) <+ transition(t_9,delay,ttime);
	V(t10) <+ transition(t_10,delay,ttime);
	V(t11) <+ transition(t_11,delay,ttime);
	V(t12) <+ transition(t_12,delay,ttime);
	V(t13) <+ transition(t_13,delay,ttime);
	V(t14) <+ transition(t_14,delay,ttime);
end

endmodule
```

VerilogA: Thermometer-To-Binary Encoder

```
// VerilogA for ADC_Class, ThermometerToBinary, veriloga

`include "constants.vams"
`include "disciplines.vams"

module
ThermometerToBinary(t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14,d0,d1,d2,d3,vdd,vss
);

parameter real vtrans=0.5;
parameter real delay = 0;
parameter real ttime = 1p;

inout vdd,vss;
input t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14;
output d0,d1,d2,d3;

electrical vdd,vss;
electrical t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,t10,t11,t12,t13,t14;
electrical d0,d1,d2,d3;

real d_0,d_1,d_2,d_3;
real vmid;

//My inputs are from 0 to Vdd, My outputs are from 0 to 1 (binary codes)

analog begin
                    vmid = (V(vdd)+V(vss))/2;

                    if (V(t14)>0.9) begin
d_3 = 1;            d_2 = 1;            d_1 = 1;            d_0 = 1; end
                    else if (V(t13)>0.9) begin
d_3 = 1;            d_2 = 1;            d_1 = 1;            d_0 = 0; end
                    else if (V(t12)>0.9)  begin
d_3 = 1;            d_2 = 1;            d_1 = 0;            d_0 = 1; end
                    else if (V(t11)>0.9) begin
d_3 = 1;            d_2 = 1;            d_1 = 0;            d_0 = 0; end
                    else if (V(t10)>0.9) begin
d_3 = 1;            d_2 = 0;            d_1 = 1;            d_0 = 1; end
                    else if (V(t9)>0.9) begin
d_3 = 1;            d_2 = 0;            d_1 = 1;            d_0 = 0; end
                    else if (V(t8)>0.9) begin
d_3 = 1;            d_2 = 0;            d_1 = 0;            d_0 = 1; end
                    else if (V(t7)>0.9) begin
d_3 = 1;            d_2 = 0;            d_1 = 0;            d_0 = 0; end
                    else if (V(t6)>0.9) begin
d_3 = 0;            d_2 = 1;            d_1 = 1;            d_0 = 1; end
                    else if (V(t5)>0.9) begin
d_3 = 0;            d_2 = 1;            d_1 = 1;            d_0 = 0; end
                    else if (V(t4)>0.9) begin
d_3 = 0;            d_2 = 1;            d_1 = 0;            d_0 = 1; end
                    else if (V(t3)>0.9) begin
d_3 = 0;            d_2 = 1;            d_1 = 0;            d_0 = 0; end
                    else if (V(t2)>0.9) begin
d_3 = 0;            d_2 = 0;            d_1 = 1;            d_0 = 1; end
                    else if (V(t1)>0.9) begin
d_3 = 0;            d_2 = 0;            d_1 = 1;            d_0 = 0; end
                    else if (V(t0)>0.9) begin
d_3 = 0;            d_2 = 0;            d_1 = 0;            d_0 = 1; end
                    else begin
d_3 = 0;            d_2 = 0;            d_1 = 0;            d_0 = 0; end

                    V(d3) <+ transition(d_3,delay,ttime);
                    V(d2) <+ transition(d_2,delay,ttime);
                    V(d1) <+ transition(d_1,delay,ttime);
                    V(d0) <+ transition(d_0,delay,ttime);


end

endmodule
```

VerilogA:
4-bit DAC

```
// VerilogA for ADC_Ideal_4bit_FlashADC, VerilogA_DAC_4bit, veriloga

`include "constants.vams"
`include "disciplines.vams"

module VerilogA_DAC_4bit(d0,d1,d2,d3,vout,vdd,vss,vmin,vmax);

parameter real vtrans=0.5;
parameter real delay = 0;
parameter real ttime = 1p;

inout vdd,vss;
input d0,d1,d2,d3;
input vmin, vmax;
output vout;

electrical vout,vdd,vss,d0,d1,d2,d3,vmin,vmax;

real result,d_0,d_1,d_2,d_3;

analog begin
			d_3 = V(d3)*8;
			d_2 = V(d2)*4;
			d_1 = V(d1)*2;
			d_0 = V(d0)*1;

		result = ((d_3+d_2+d_1+d_0) * ((V(vmax)-V(vmin))/(16))) + V(vmin) ;

			V(vout) <+ transition(result,delay,ttime);
end


endmodule
```

Matlab Code: for DFT

```
clc; close all;
%data = importdata('verilogA_DAC_output.csv');
data = importdata('ideal_DAC_output.csv');
t = data(:,1);
x = data(:,2);
plot(t,x,'linewidth',2); grid on;
xlabel('time(s)'); ylabel('Voltage(V)')

FS = 1;
fs = 100e6;
fnyquist = fs/2;
N = 64;
cycles = 7;
fx = (cycles/N)*fs;
Afs = 1;

% spectrum
% PrettyFFT Gives ENOB, SNDR, SFDR, SNR
figure
prettyFFT(x); grid on;

figure
s = abs(fft(x,N))
s = s + 1e-7
p = s(1:(N/2)+1);
p = 2*p/N/FS;
f = [0:(N/2)]
stem(f,p,'linewidth',2); grid on;
```

```
figure
f = [0:N-1]
s = 20*log10(2*s/N/FS)
plot(f,s,'linewidth',2); grid on;
xlabel('frequency[bin]'); ylabel('DFT magnitude
[dBFS]')

figure
m = s(1:(N/2)+1);
f = [0:(N/2)]
plot(f,m,'linewidth',2); grid on;
xlabel('frequency[bin]'); ylabel('DFT magnitude
[dBFS]')
```