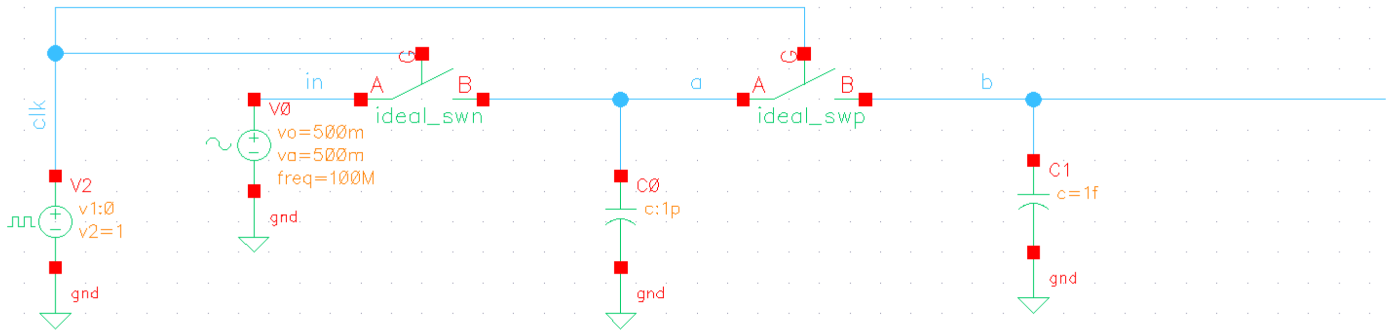
EE228 – HW1 Report
10-bit pipeline ADC & DAC using ideal components & VerilogA

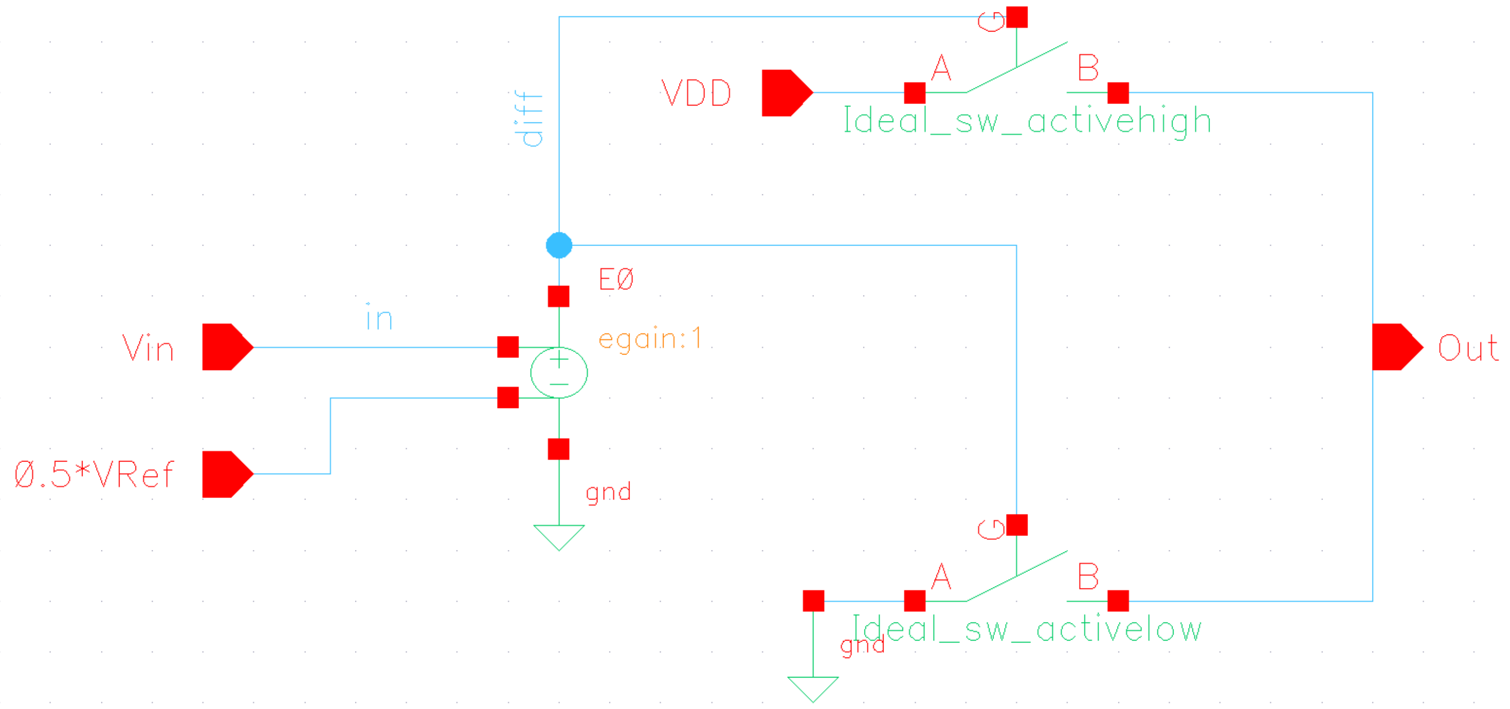
Muhammad Aldacher
Student ID: 011510317

(1)
Using Ideal Components

Sample & Hold circuit

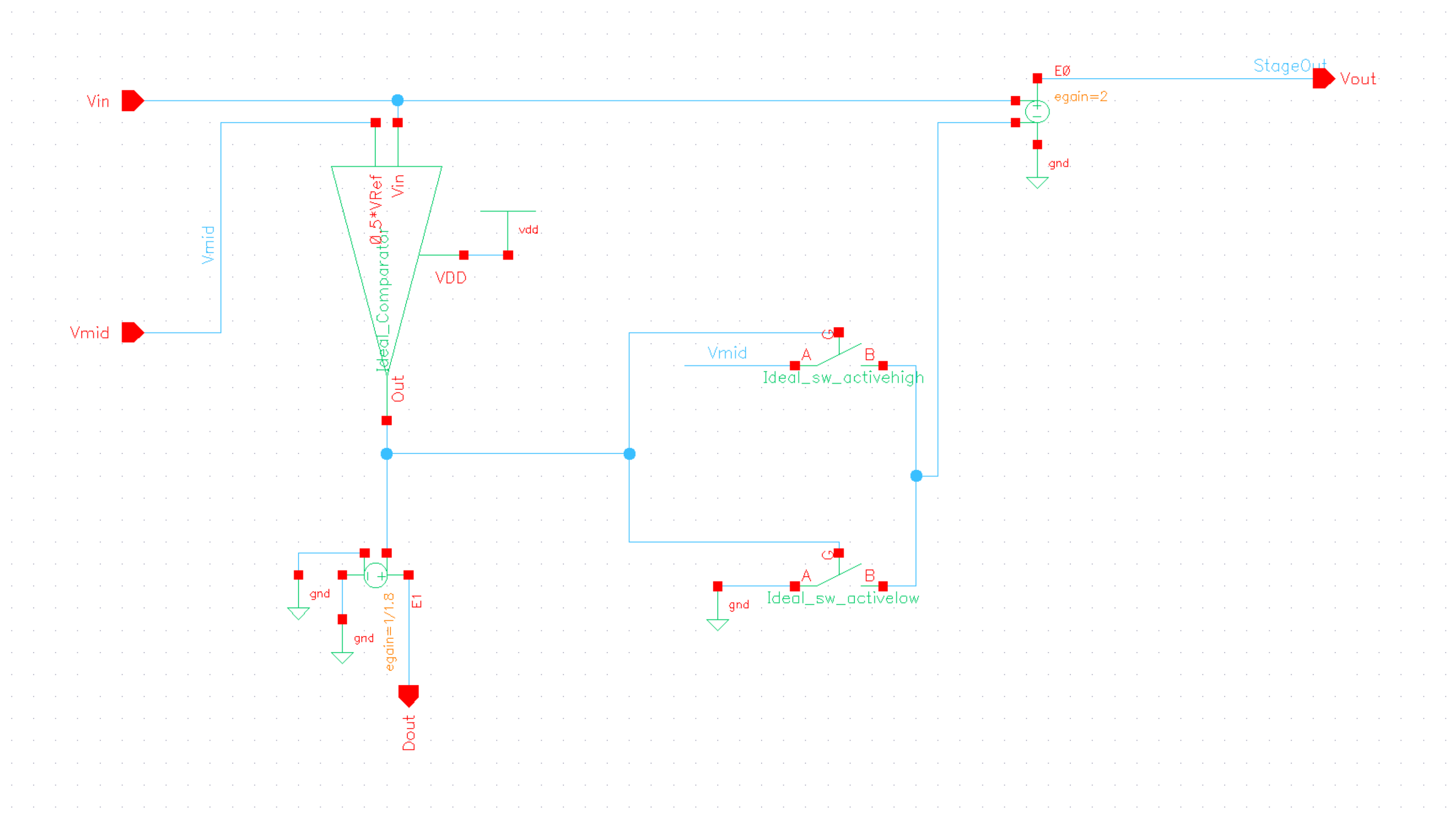


Comparator

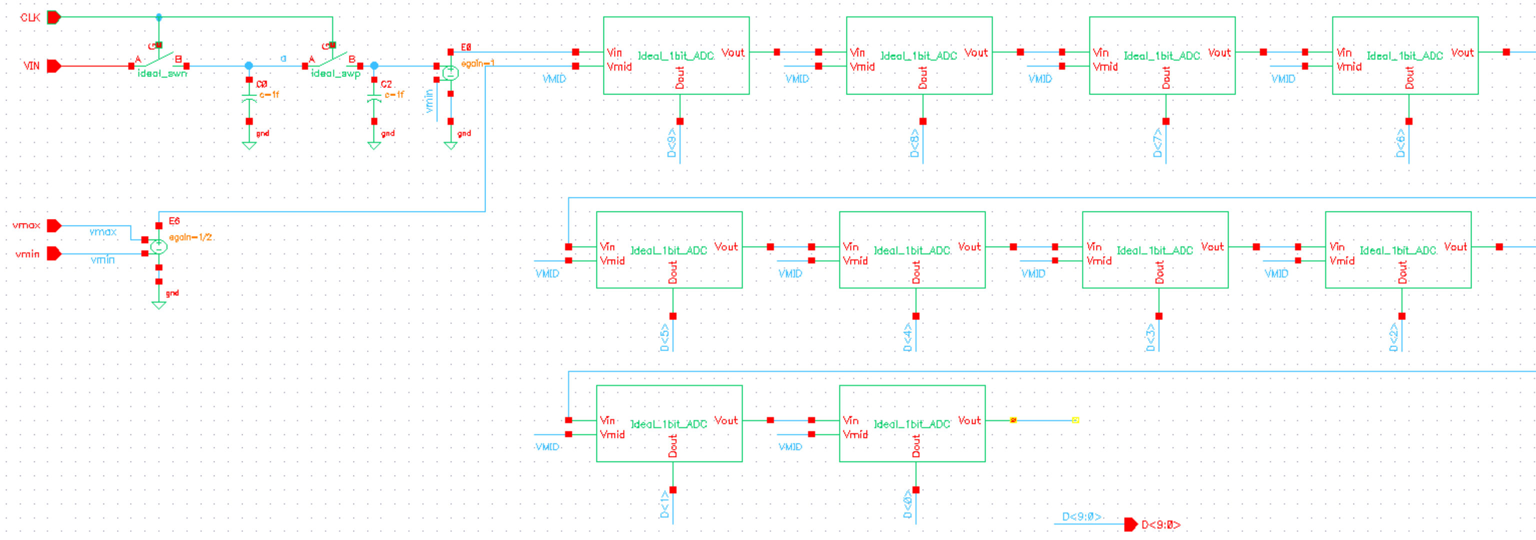


G
VDD
A
B
Ideal_sw_activehigh
diff
E0
in
Vin
egain:1
Out
0.5*VRef
gnd
G
A
B
gnd
Ideal_sw_activelow

1-bit ADC

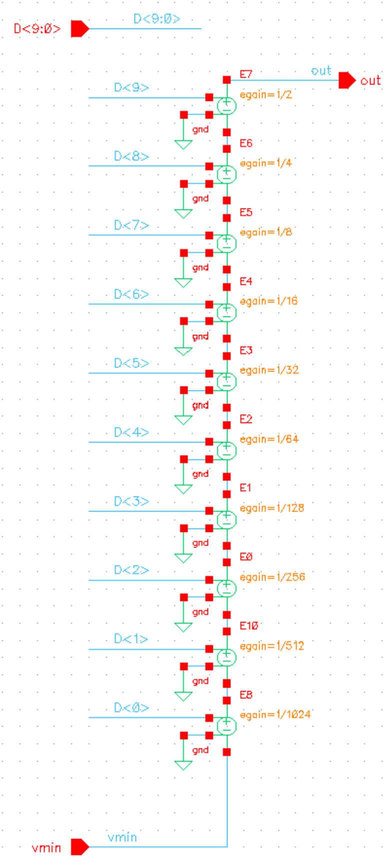

Vin
Vmid
Vmid
0.5*VRef
Vin
Ideal_Comparator
Out
VDD
vdd
E0
egain=2
StageOut
Vout
gnd
G
A
B
Vmid
Ideal_sw_activehigh
G
A
B
gnd
Ideal_sw_activelow
gnd
gnd
egain=1/1.8
E1
Dout

10-bit ADC

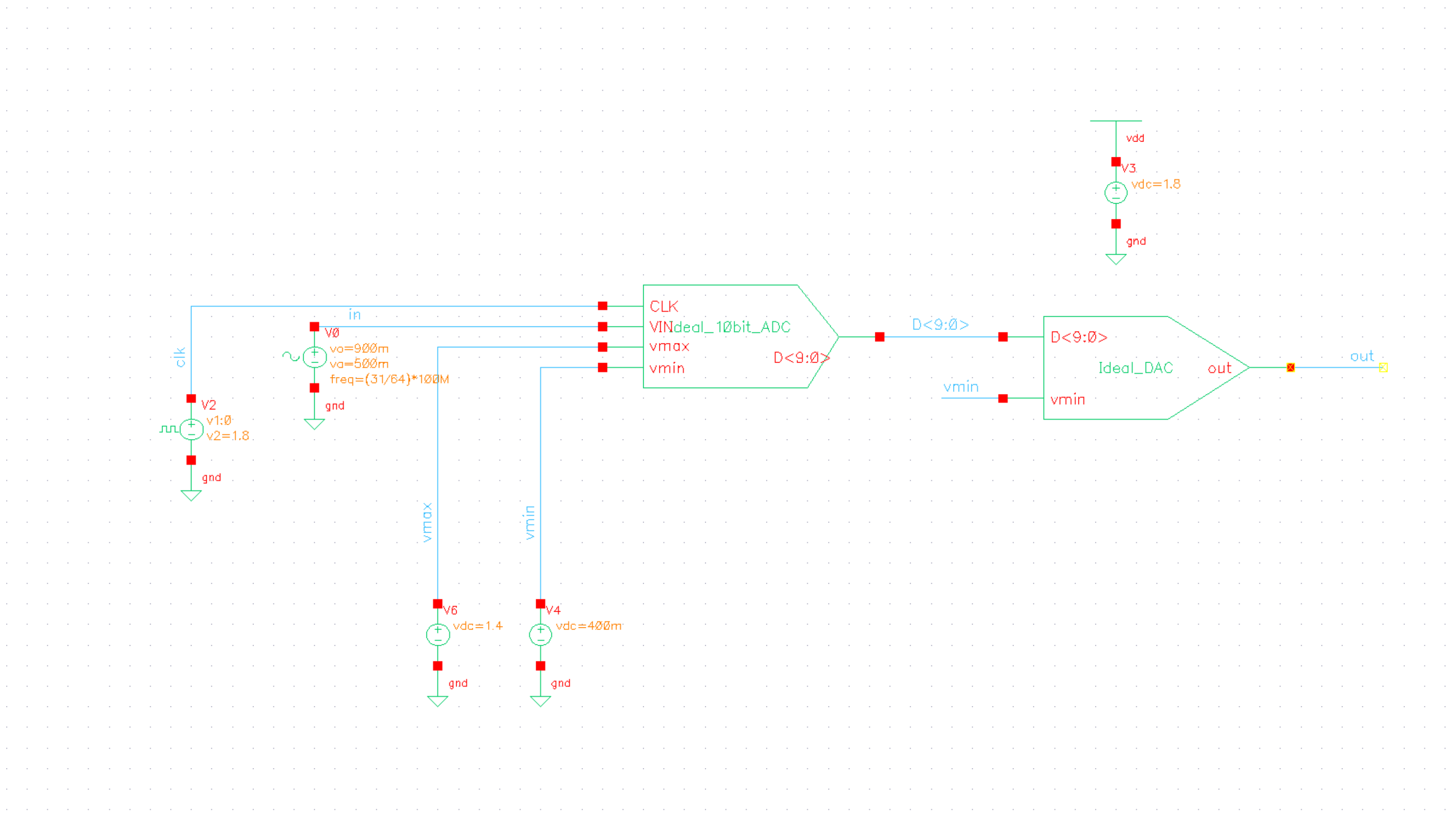


CLK
VIN
vmax
vmin
ideal_swn
ideal_swp
C0 c=1f
C2 c=1f
E0 egain=1
E6 egain=1/2
gnd
VMID
Vin
Vmid
Ideal_1bit_ADC
Vout
Dout
D<9>
D<8>
D<7>
D<6>
D<5>
D<4>
D<3>
D<2>
D<1>
D<0>
D<9:0>

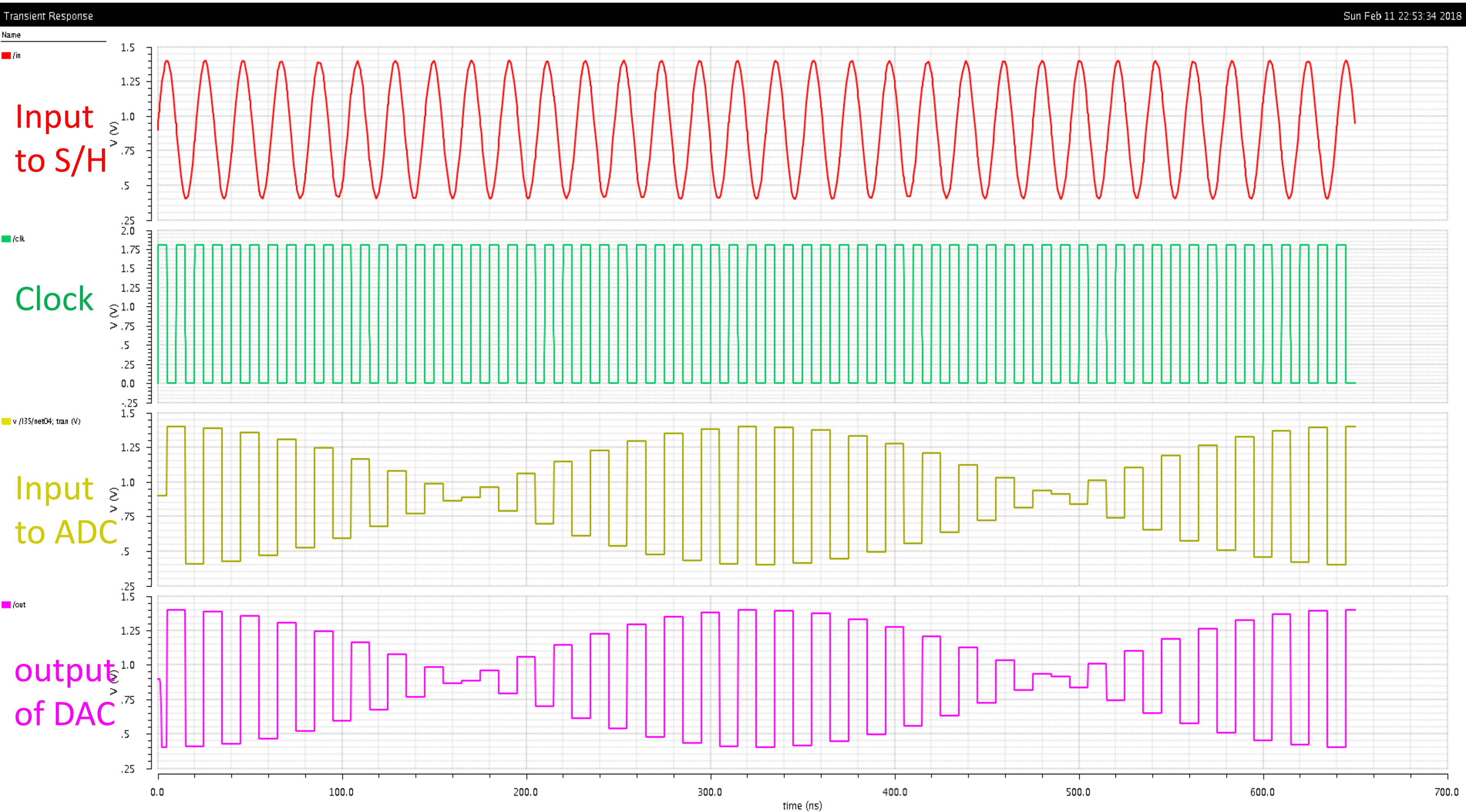
10-bit DAC


D<9:0>
D<9:0>
out
out
E7
egain=1/2
E6
egain=1/4
E5
egain=1/8
E4
egain=1/16
E3
egain=1/32
E2
egain=1/64
E1
egain=1/128
E0
egain=1/256
E10
egain=1/512
E8
egain=1/1024
D<9>
D<8>
D<7>
D<6>
D<5>
D<4>
D<3>
D<2>
D<1>
D<0>
gnd
vmin
vmin

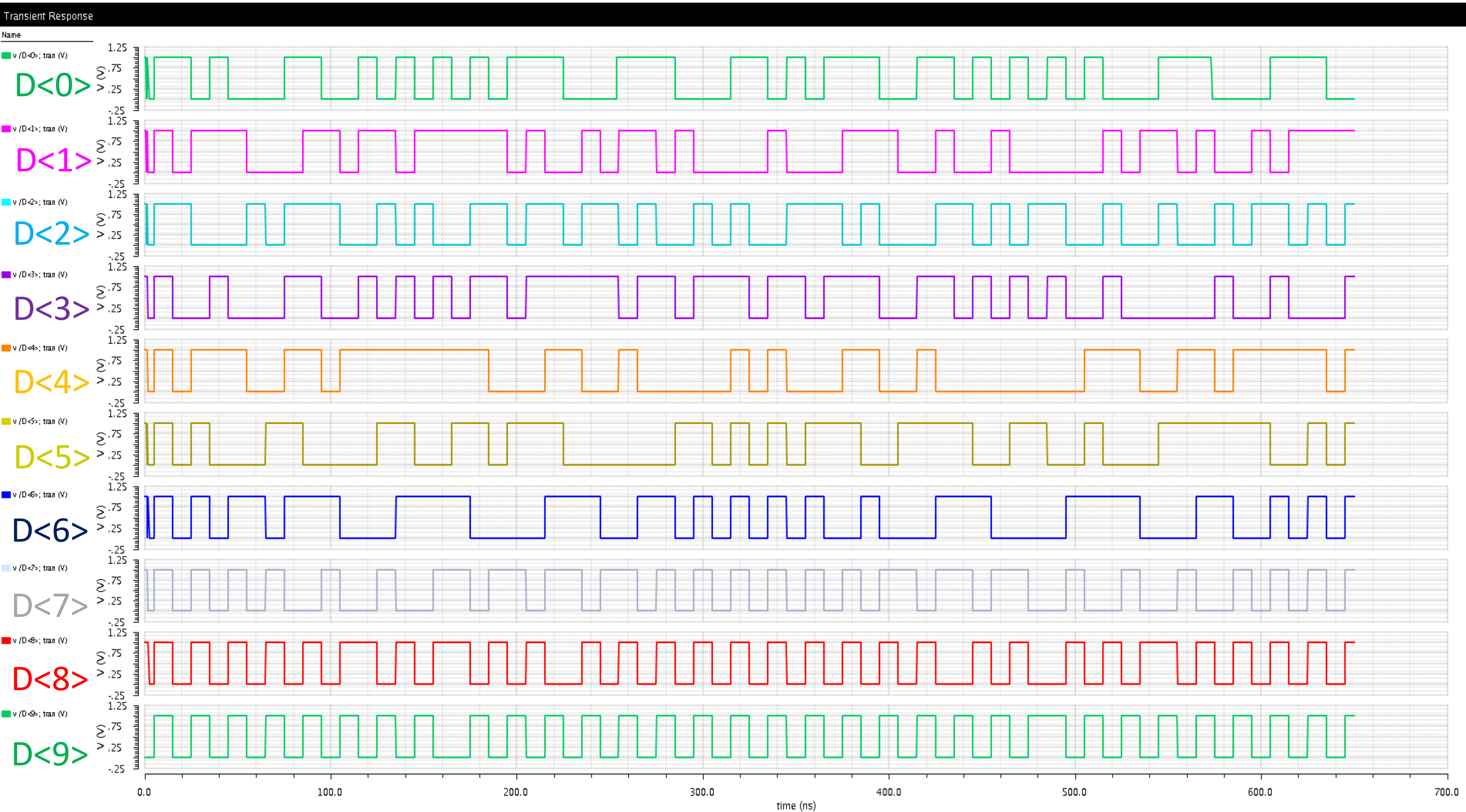
Testbench for the ADC & the DAC



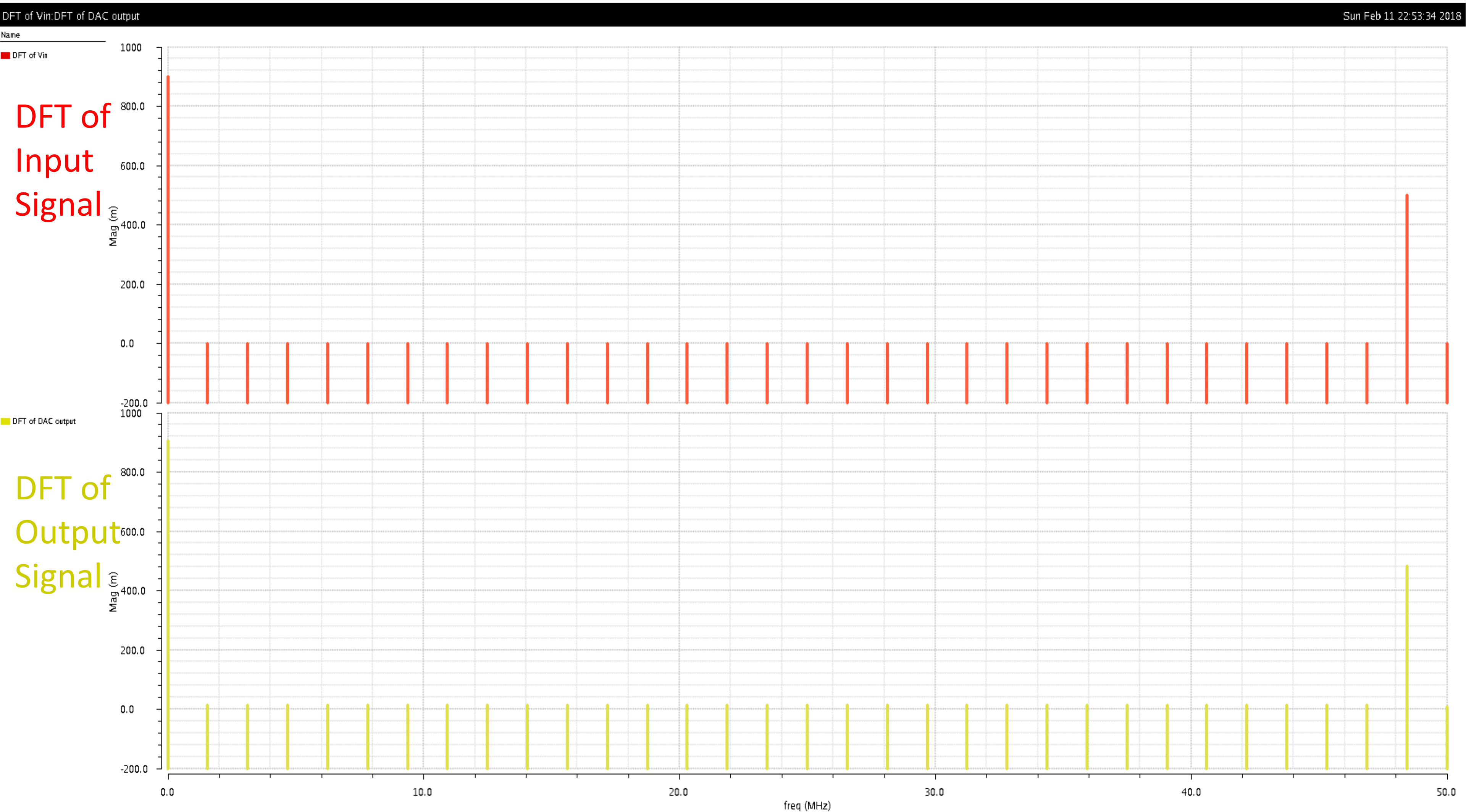
Test Results: Analog Waveforms



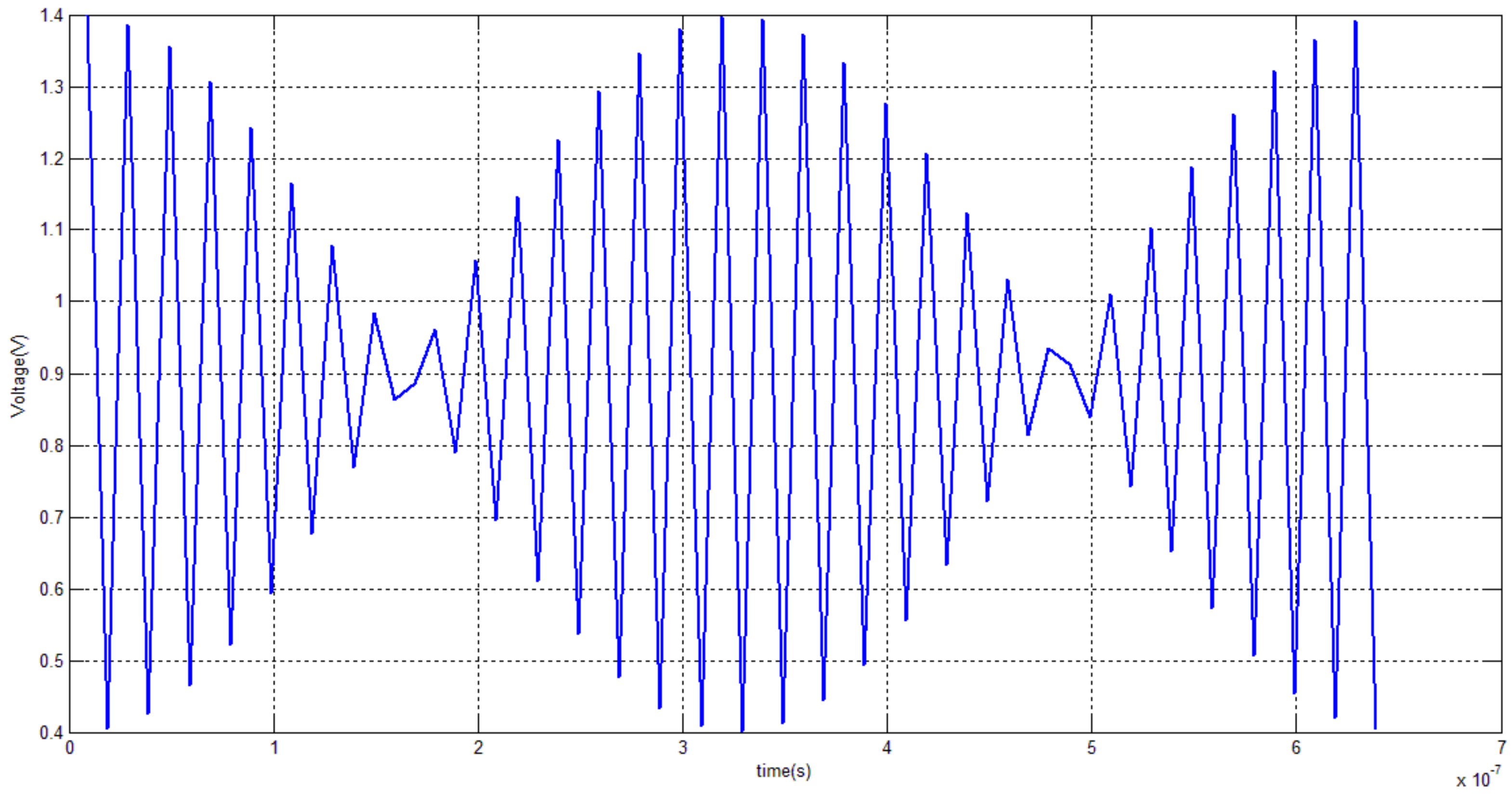
Test Results: Digital Outputs from ADC



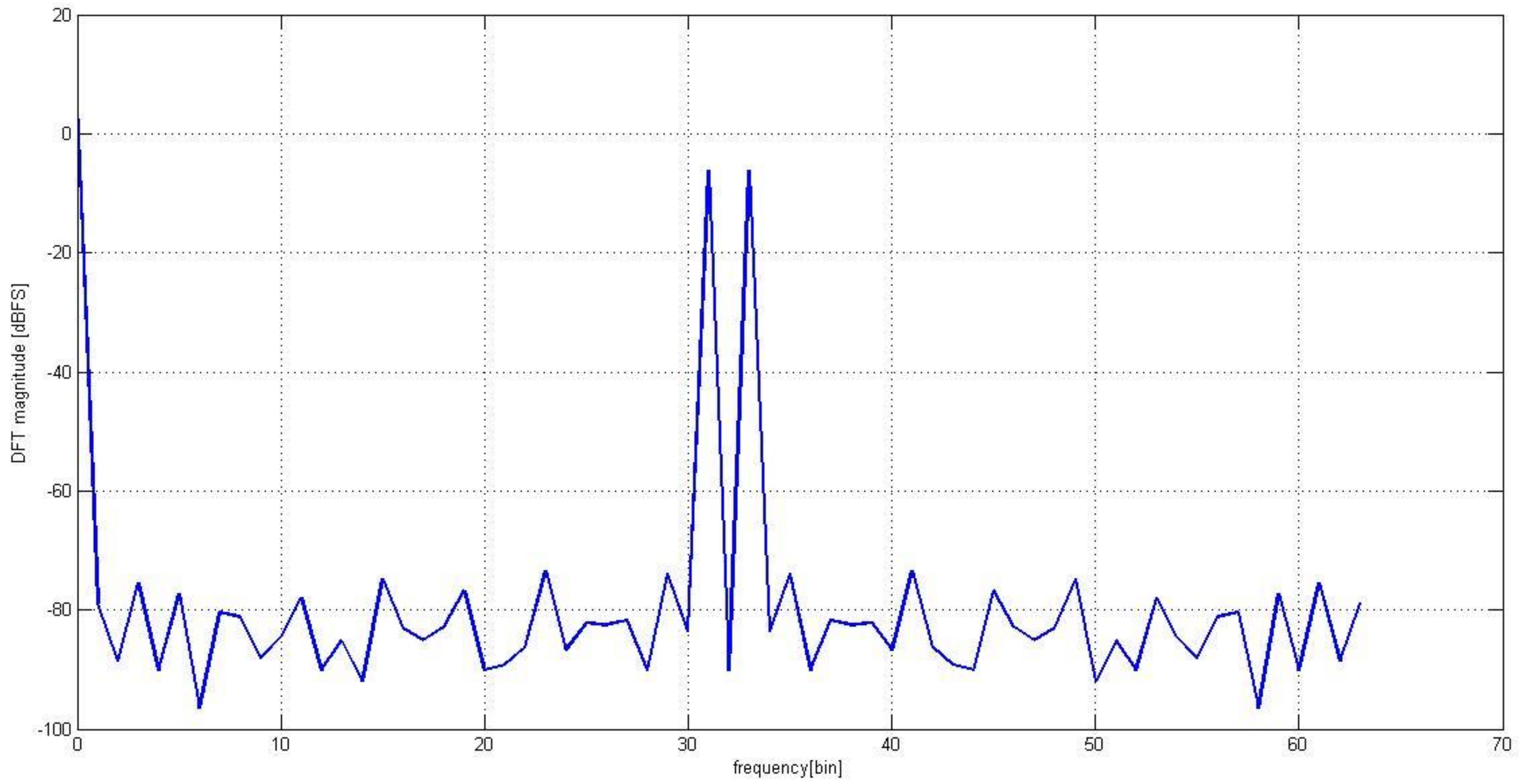
Test Results:
DFT from Cadence



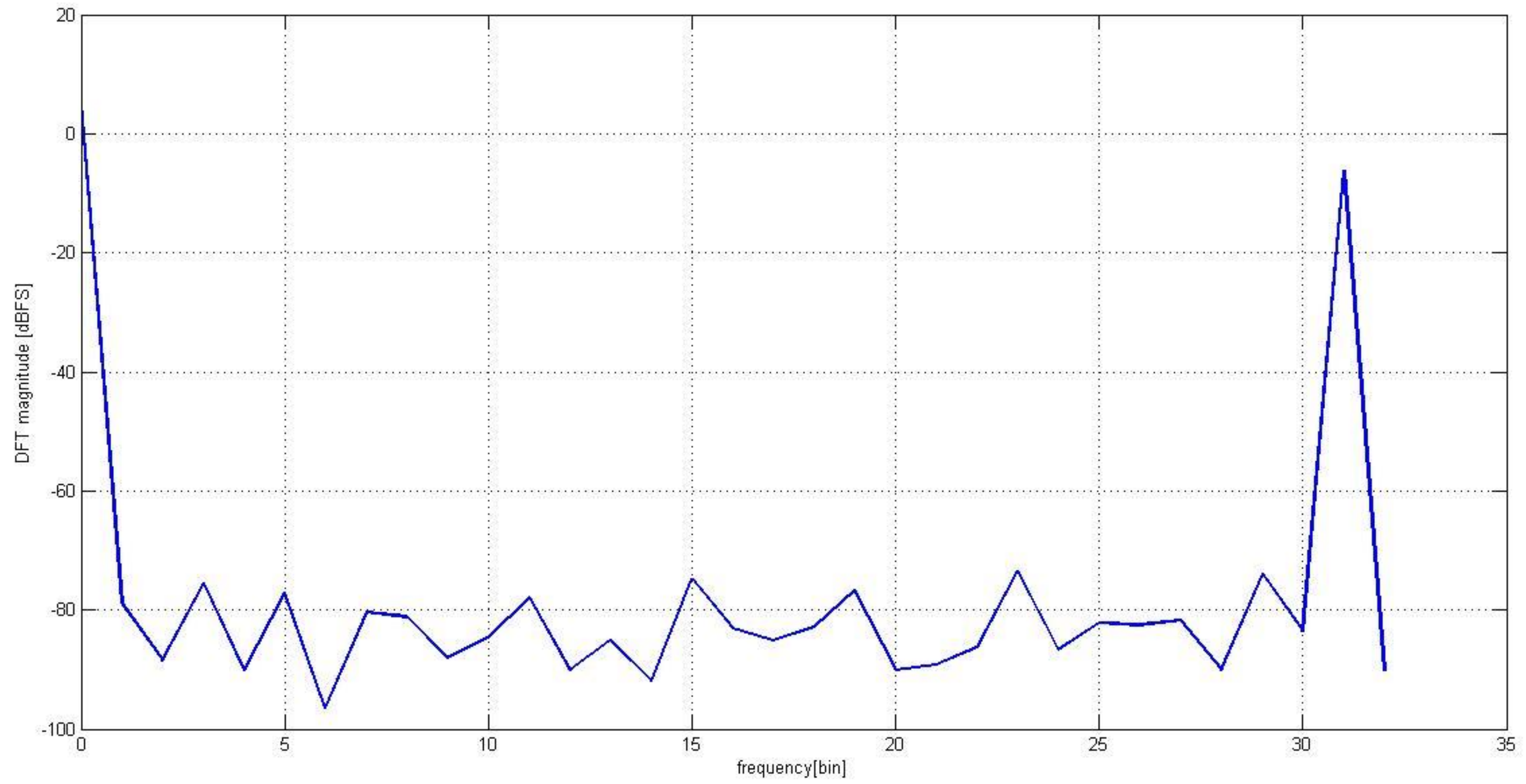
Matlab: Output Waveform reconstructed from sampled data



Matlab:
Output spectrum using DFT
(from N = 1→64)

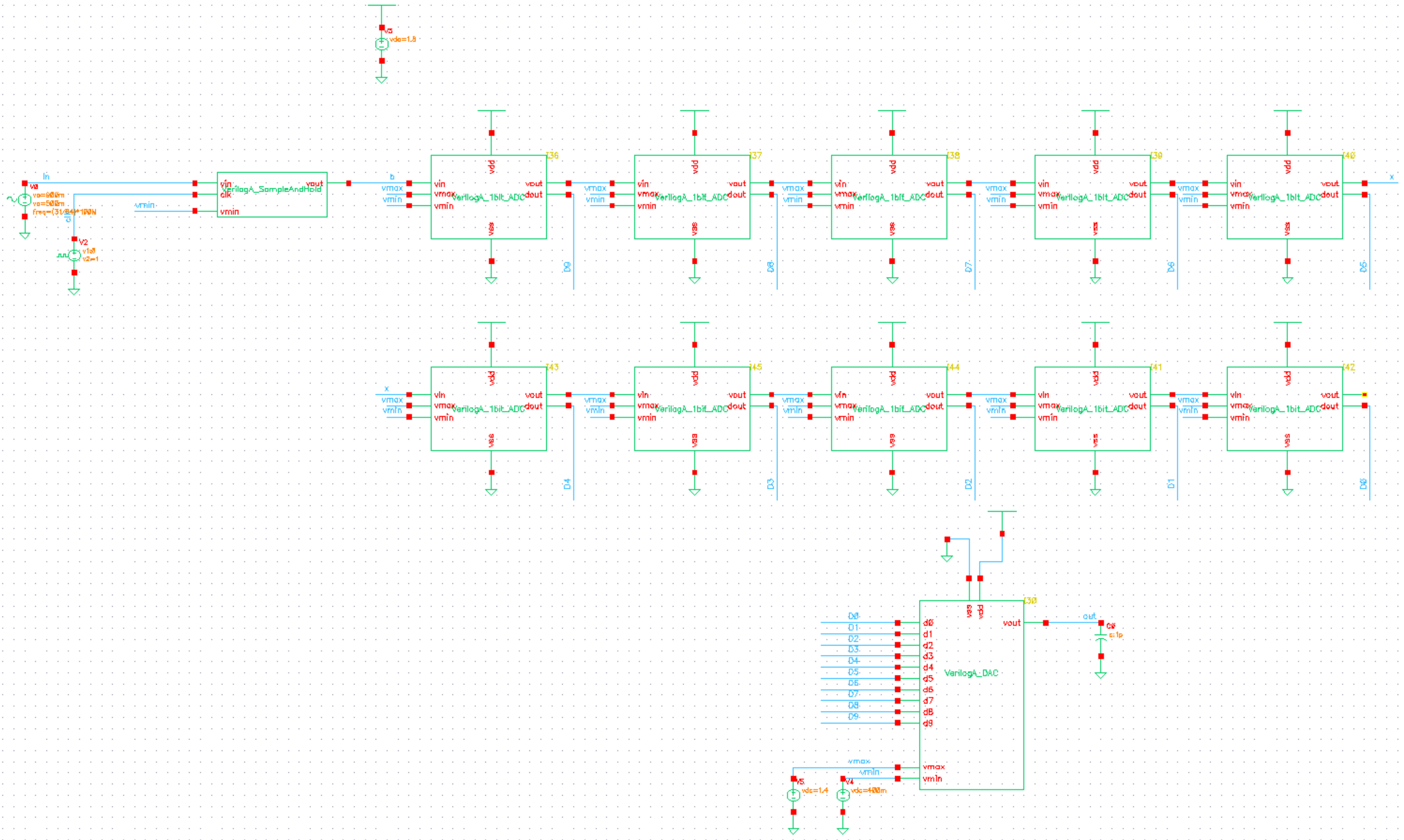


Matlab: Output spectrum using DFT (from N = 1→33)

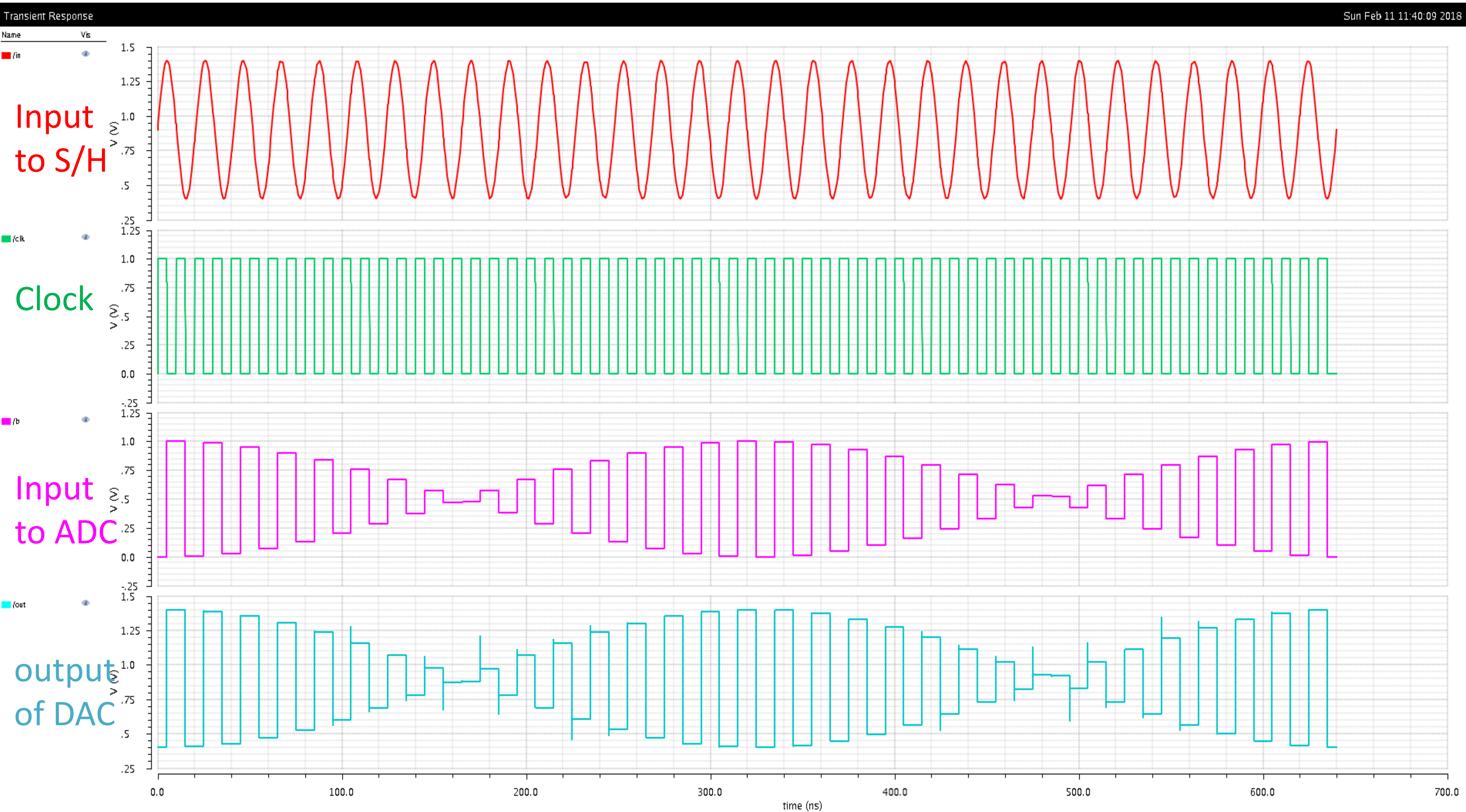


(2)
Using VerilogA Blocks

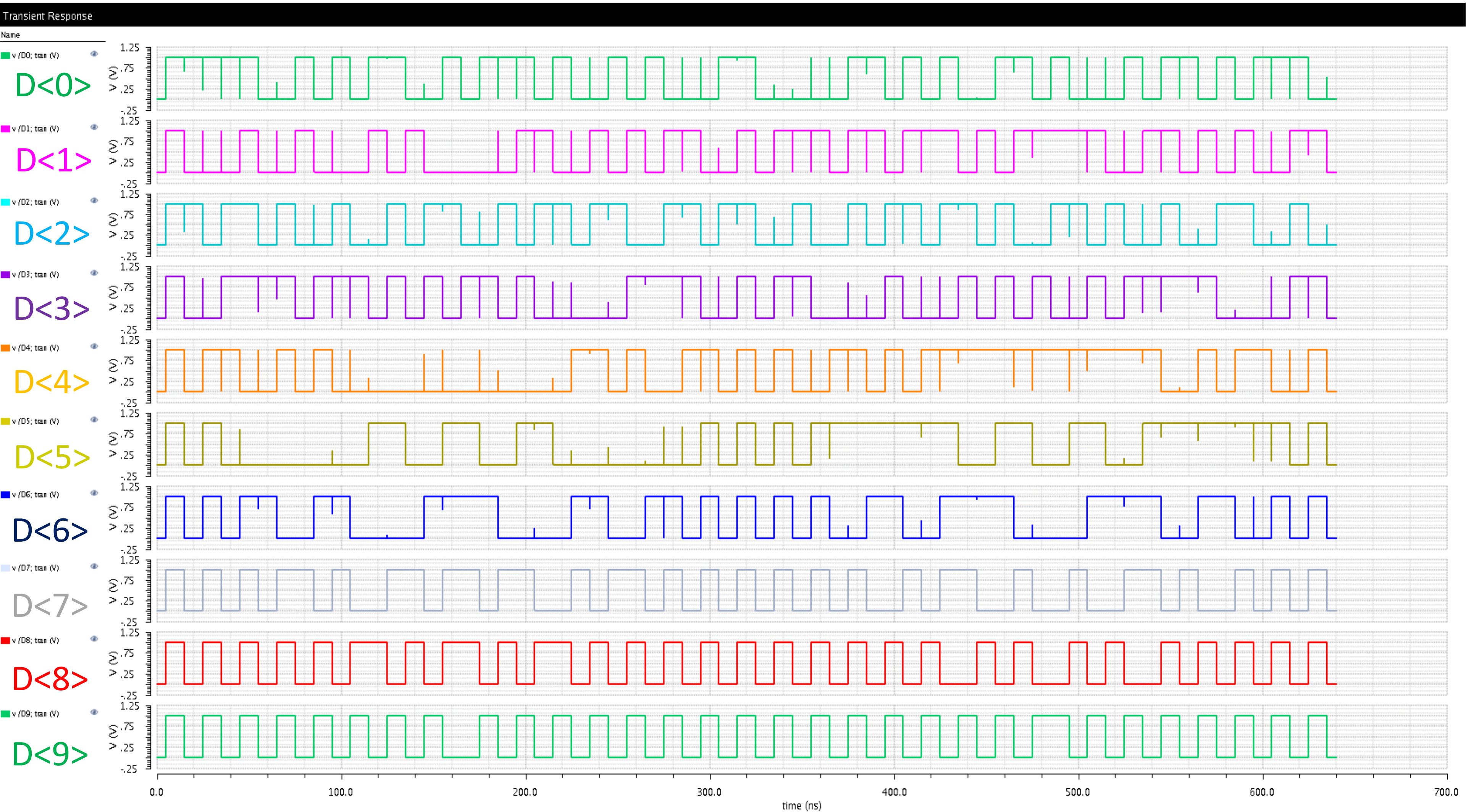
10-bit ADC & DAC



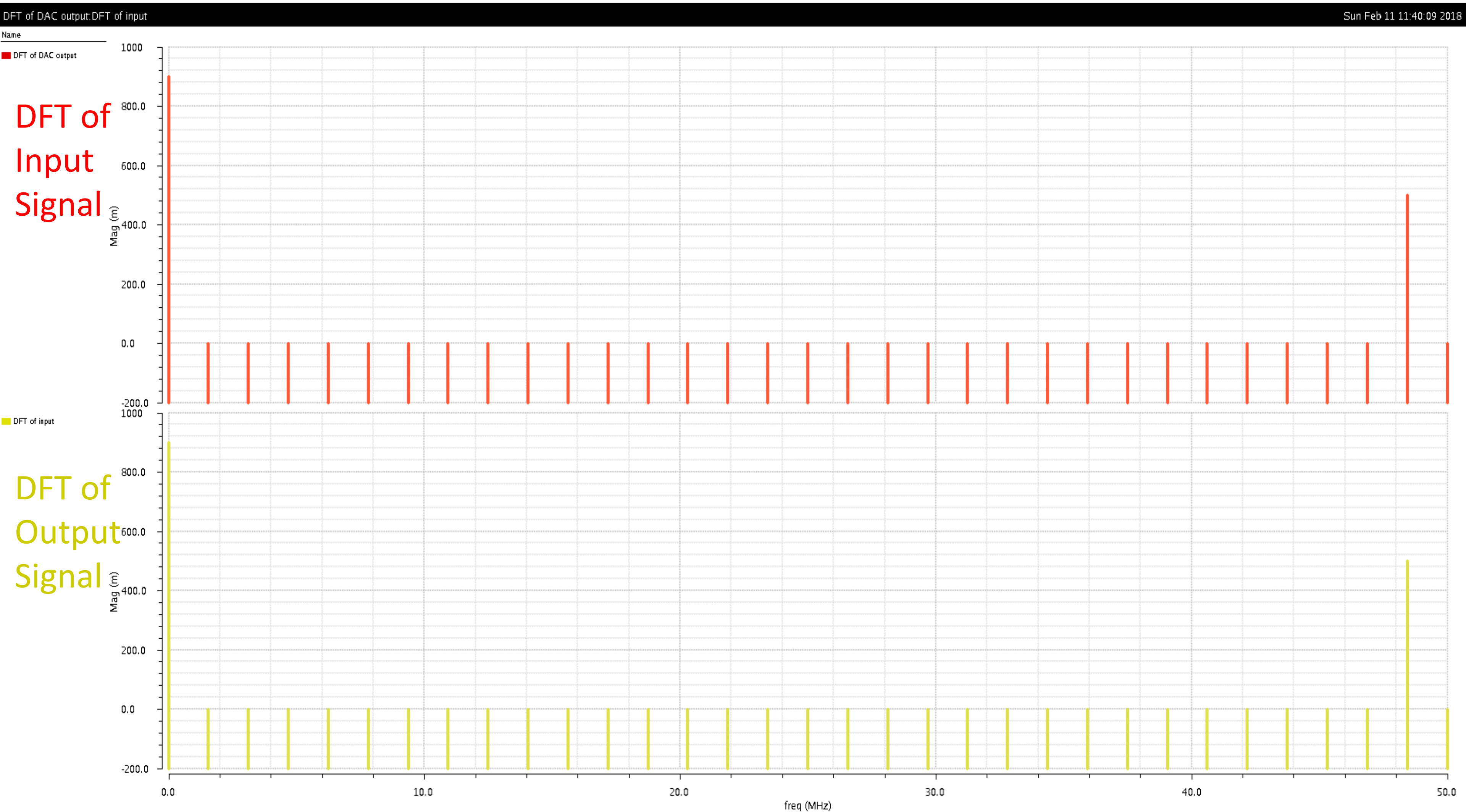
Test Results:
Analog Waveforms



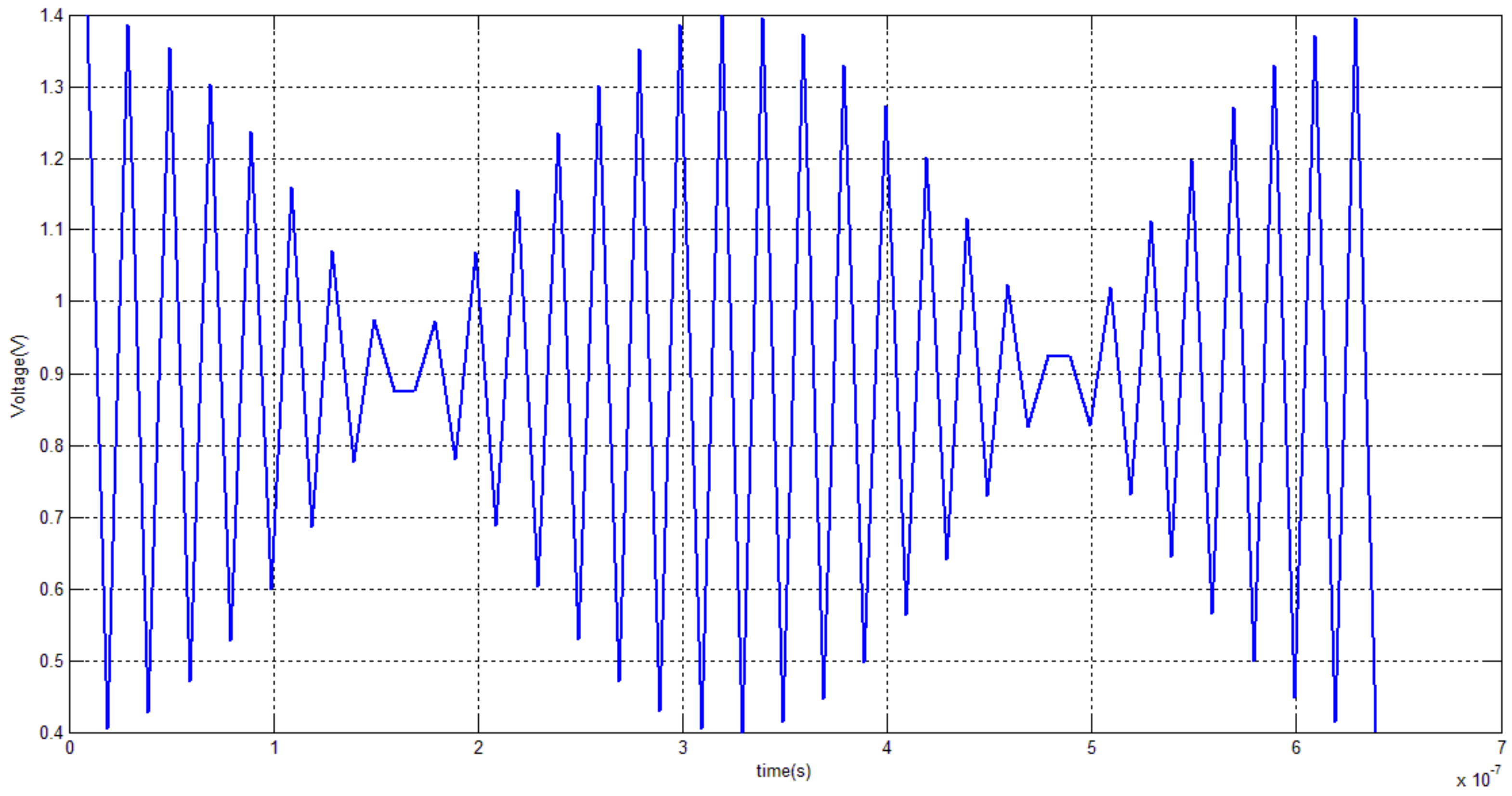
Test Results:
Digital Outputs from ADC



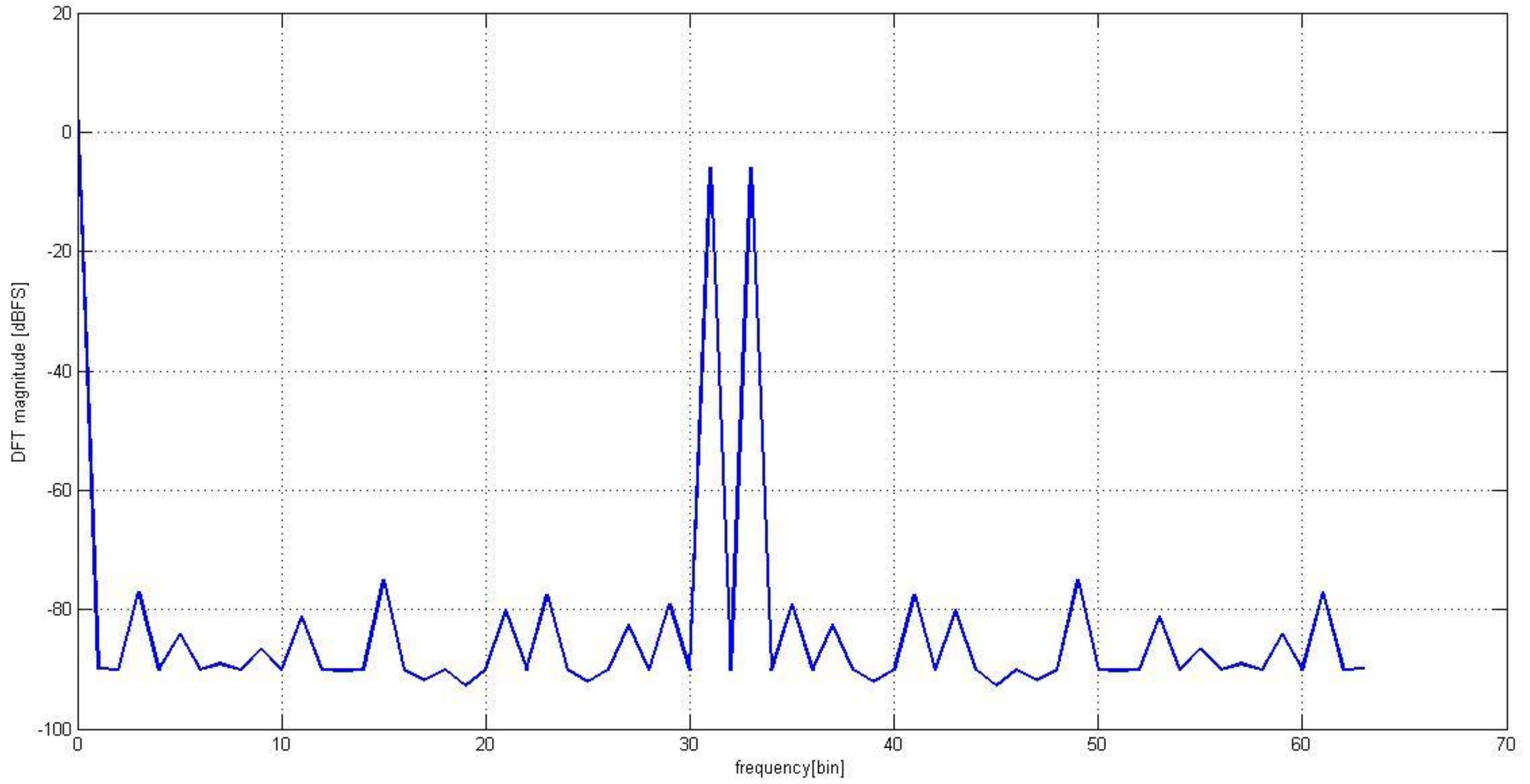
Test Results: DFT from Cadence



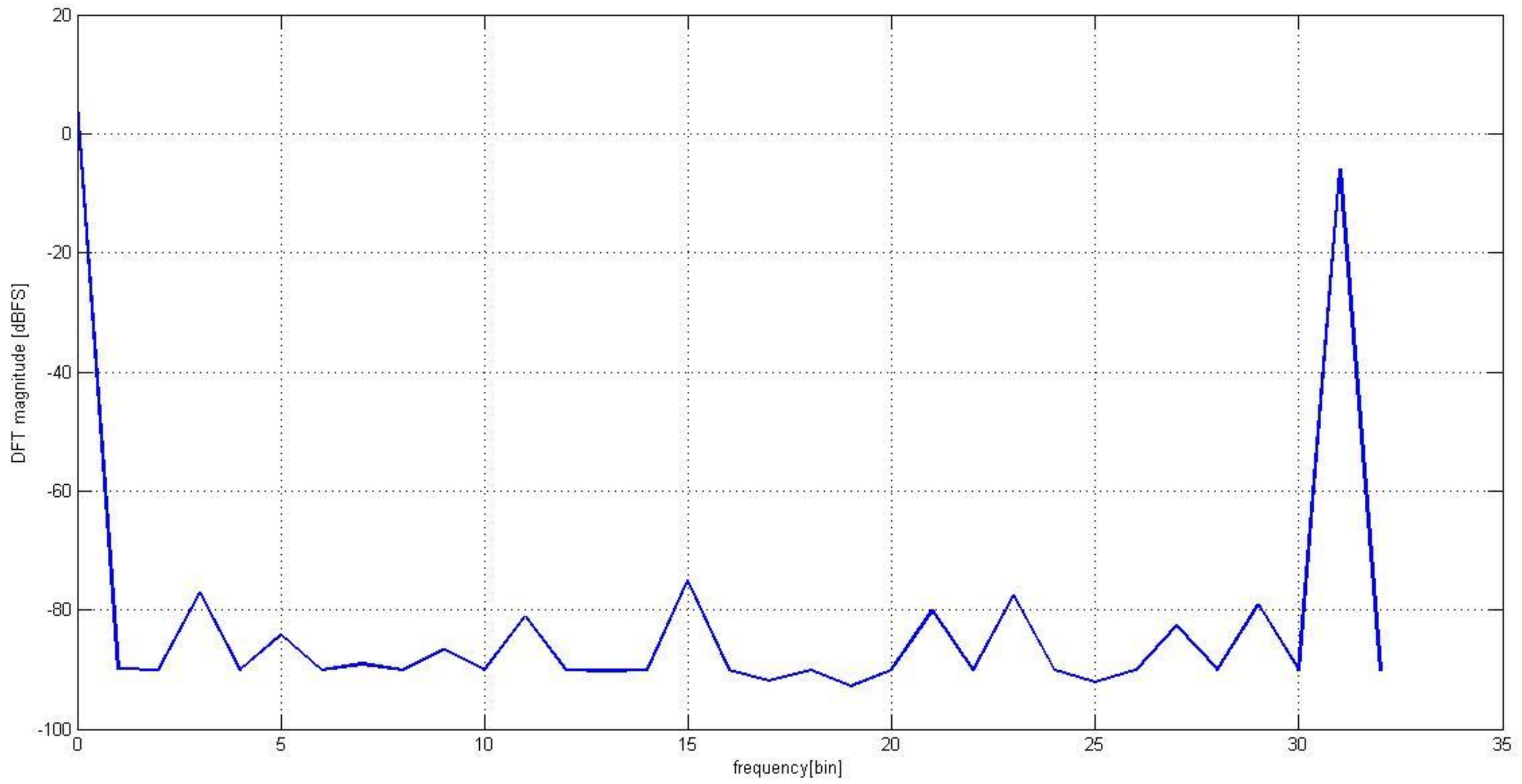
Matlab:
Output Waveform reconstructed from sampled data



Matlab: Output spectrum using DFT (from N = 1→64)



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



➔

Matlab & VerilogA Codes

VerilogA:
Sample & Hold

```
// VerilogA for SampleAndHold

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

module VerilogA_SampleAndHold(clk,vin,vmin,vout);

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;
output vout;

electrical vout,vin,vmin,clk;

real v;

analog begin

            // Sampling Phase (+1 is for rising edge, -1 is for falling edge)
            @(cross(V(clk) - clk_threshold, -1))
                        v = V(vin) - V(vmin);

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

endmodule
```

VerilogA:
1-bit ADC

```
// VerilogA for ADC_Ideal_10bit_Pipeline, VerilogA_1bit_ADC, veriloga

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

module VerilogA_1bit_ADC(dout,vout,vdd,vss,vmin,vmax,vin);

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

inout vdd,vss;
input vin;
input vmin, vmax;
output vout,dout;

electrical dout,vout,vdd,vss,vmin,vmax,vin;
real vref,v_result,d_result;

analog begin

          vref = (V(vmax) - V(vmin))/2;

          if(V(vin) > vref) begin
                    d_result = V(vdd)/V(vdd);
                    v_result = (V(vin) - vref)*2;
                    end
          else begin
                    d_result = 0;
                    v_result = (V(vin))*2;
                    end

          V(vout) <+ transition(v_result,delay,ttime);
          V(dout) <+ transition(d_result,delay,ttime);
end

endmodule
```

VerilogA: 10-bit DAC

```
// VerilogA for ADC_Class, IdealDAC_10bit, veriloga

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

module
IdealDAC_10bit(d0,d1,d2,d3,d4,d5,d6,d7,d8,d9,vout,vdd,vss,vmin,vma
x);

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

inout vdd,vss;
input d0,d1,d2,d3,d4,d5,d6,d7,d8,d9;
input vmin, vmax;
output vout;

electrical vout,vdd,vss,d0,d1,d2,d3,d4,d5,d6,d7,d8,d9,vmin,vmax;

real result,d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7,d_8,d_9;
```

```
analog begin
                d_9 = V(d9)*512;
                d_8 = V(d8)*256;
                d_7 = V(d7)*128;
                d_6 = V(d6)*64;
                d_5 = V(d5)*32;
                d_4 = V(d4)*16;
                d_3 = V(d3)*8;
                d_2 = V(d2)*4;
                d_1 = V(d1)*2;
                d_0 = V(d0)*1;

                result =
((d_9+d_8+d_7+d_6+d_5+d_4+d_3+d_2+d_1+d_0) * ((V(vmax)-
V(vmin))/(1023))) + V(vmin) ;

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

endmodule
```

Matlab Code: for DFT

```matlab
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 = length(x);
cycles = 31;
fx = (cycles/N)*fs;
Afs = 1;

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

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

```matlab
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]')
```