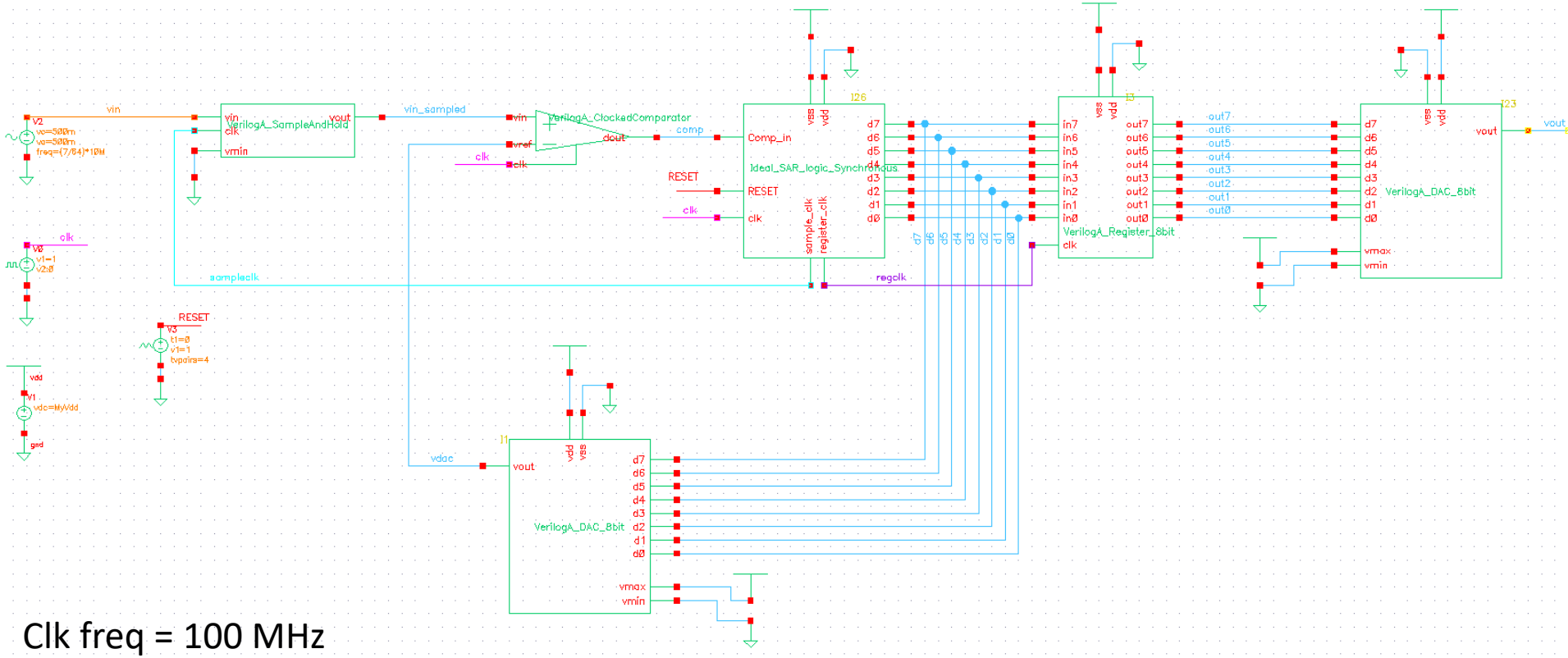
VerilogA Models for 8-bit Synchronous SAR ADC

MUHAMMAD ALDACHER

3/1/2020

Synchronous SAR ADC

Synchronous SAR ADC

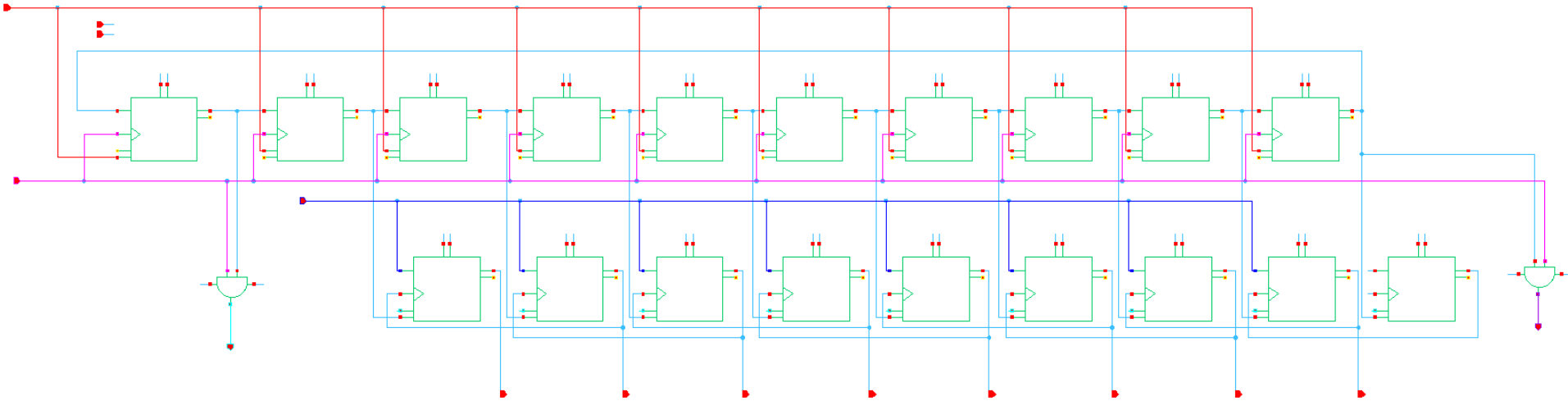


Clk freq = 100 MHz

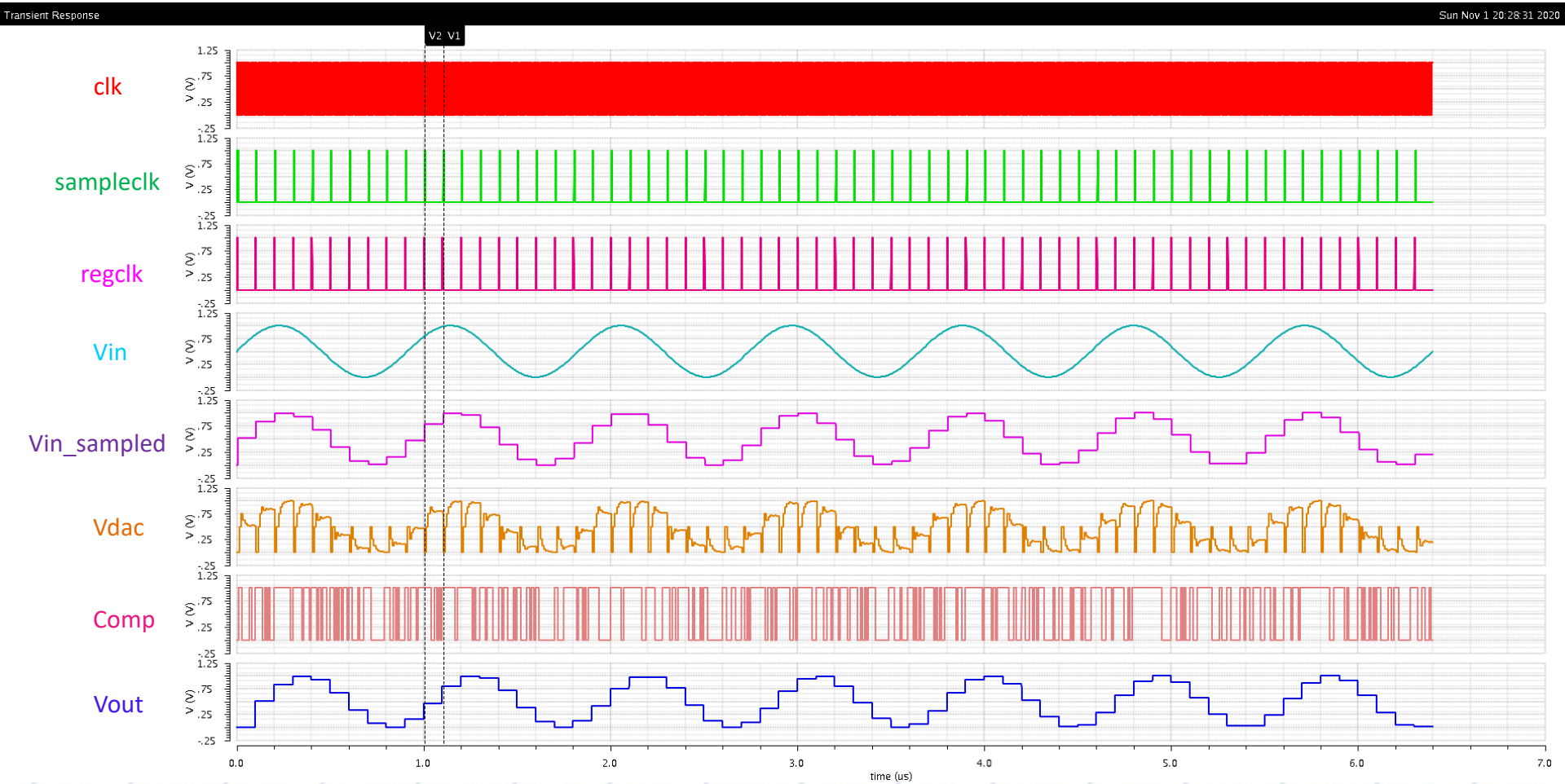
Vin freq = (7/64)*10 MHz

Output = 10 Msamples/sec

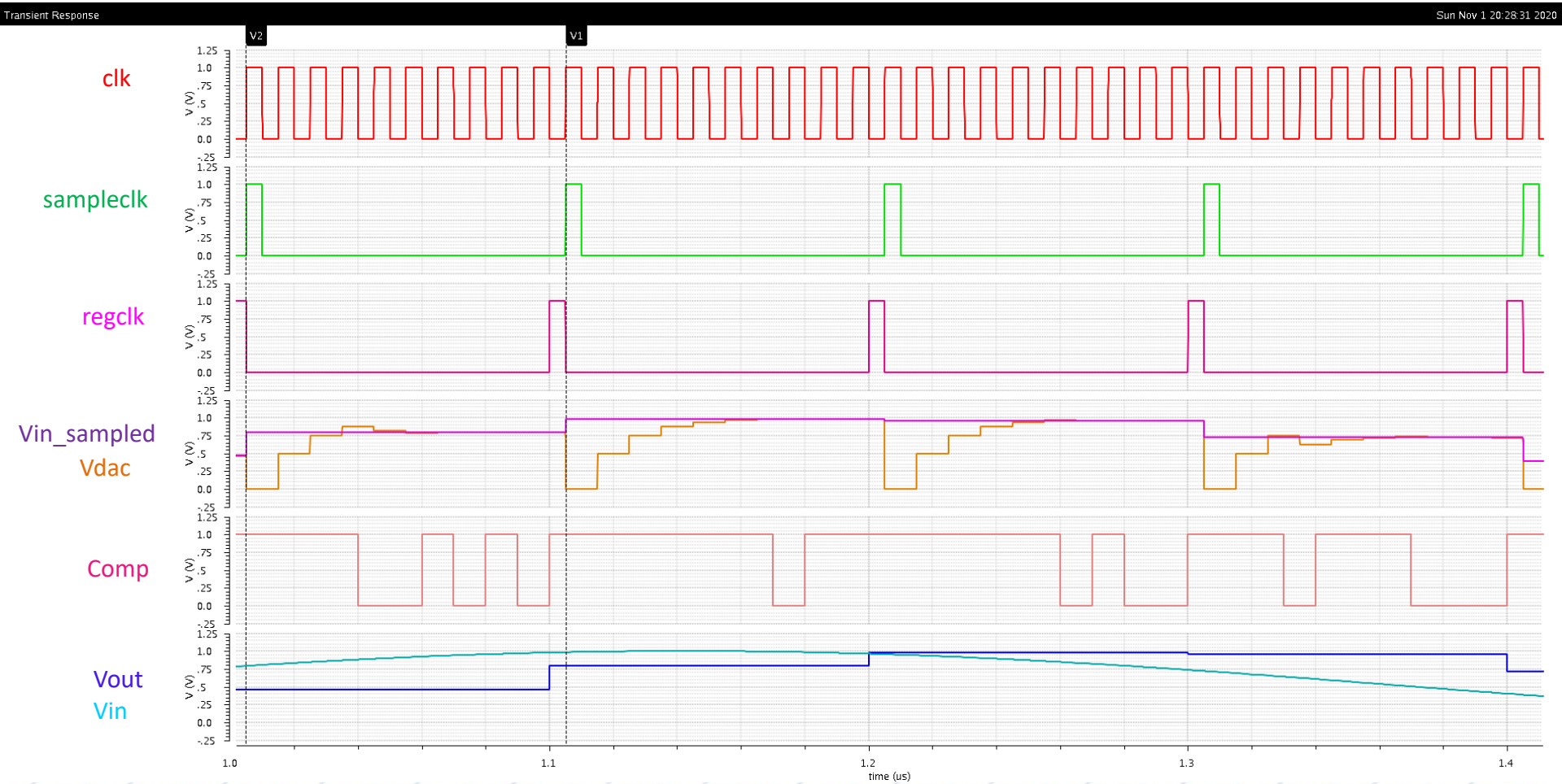
Synchronous SAR Logic Block



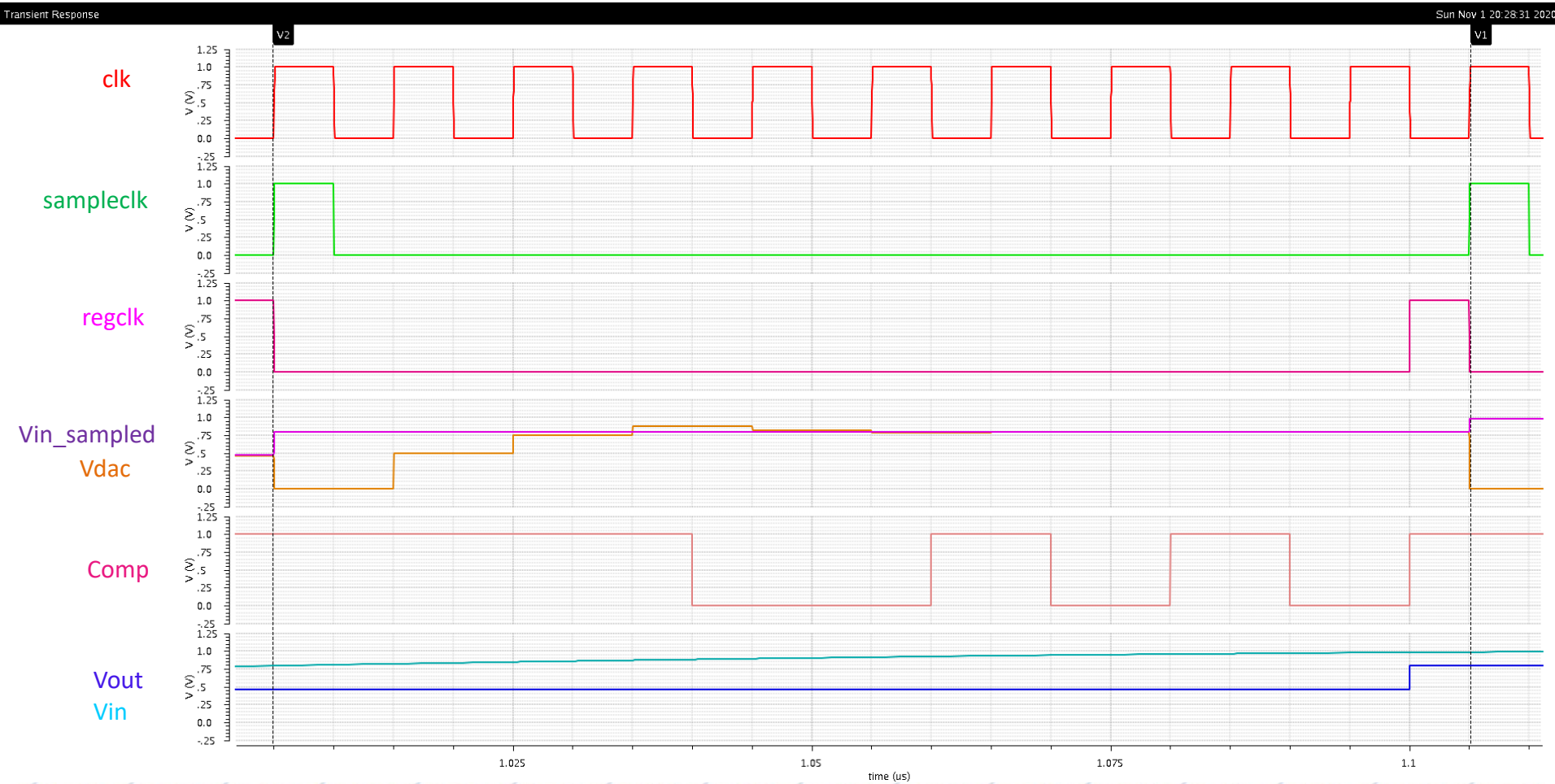
Waveforms


Transient Response
Sun Nov 1 20:28:31 2020
V2 V1
clk
sampleclk
regclk
Vin
Vin_sampled
Vdac
Comp
Vout
V (V)
1.25
.75
.25
-.25
0.0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
time (us)

Waveforms


Transient Response
Sun Nov 1 20:28:31 2020
V2
V1
clk
sampleclk
regclk
Vin_sampled
Vdac
Comp
Vout
Vin
V (V)
1.25
1.0
.75
.5
.25
0.0
-.25
1.0
1.1
1.2
1.3
1.4
time (us)

Waveforms (1 sample)



VerilogA Codes

1) Sample-And-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 = 0.5;                    //vdd is 1v

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

end
endmodule
```

2) Clocked Comparator (single output)

```
// VerilogA for ClockedComparator

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

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

parameter real clk_th=0.5;
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

// We want the comparator to keep its output value when it is in Reset
                //@(cross(V(clk) - clk_th, +1)) begin
                //      d_result = 0;
                //end

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

end

endmodule
```

3) SAR Logic

```
parameter real ttime = 1p;
parameter real clk_threshold = 0.5;
// 8-bits (8-cycles) + 1-cycle for sampling + 1-cycle for Register output = 10-cycles

inout vdd,vss;
input in_comp, clk;
output d0,d1,d2,d3,d4,d5,d6,d7;
output sampleclk,regclk;

electrical in_comp,clk,d0,d1,d2,d3,d4,d5,d6,d7,regclk,sampleclk,vdd,vss;

real d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7,sample_en,sar_counter,reg_out;

analog begin

  // Initial State
  @(initial_step) begin
      sample_en = 1;
      d_7 = 0;
      d_6 = 0;
      d_5 = 0;
      d_4 = 0;
      d_3 = 0;
      d_2 = 0;
      d_1 = 0;
      d_0 = 0;
      sar_counter = 9; //10-bit counter
  end
```

```
// Operation

  // Comparison
  // At Rising Edge of clk, the bits are updated for comparison.
  // At Falling Edge of clk, the Comparator produces the new bit (Regeneration)
                  @(cross(V(clk) - clk_threshold, +1))
                  begin
    if(sar_counter == 9) begin
      sample_en = 1; //1st count is for sampling
      reg_out = 0;
                    d_7 = 0; d_6 = 0; d_5 = 0; d_4 = 0; d_3 = 0; d_2 = 0; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 8) begin
      d_7 = 1; d_6 = 0; d_5 = 0; d_4 = 0; d_3 = 0; d_2 = 0; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 7) begin
      d_7 = V(in_comp); d_6 = 1; d_5 = 0; d_4 = 0; d_3 = 0; d_2 = 0; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 6) begin
      d_6 = V(in_comp); d_5 = 1; d_4 = 0; d_3 = 0; d_2 = 0; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 5) begin
      d_5 = V(in_comp); d_4 = 1; d_3 = 0; d_2 = 0; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end
```

3) SAR Logic "Cont'd"

```
    else if(sar_counter == 4) begin
      d_4 = V(in_comp); d_3 = 1; d_2 = 0; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 3) begin
      d_3 = V(in_comp); d_2 = 1; d_1 = 0; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 2) begin
      d_2 = V(in_comp); d_1 = 1; d_0 = 0;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 1) begin
      d_1 = V(in_comp); d_0 = 1;
      sar_counter = sar_counter - 1; end

    else if(sar_counter == 0) begin
      d_0 = V(in_comp);
      sar_counter = 9; end
  end

  // Producing the neg edge of the pulse to sample & to get the output of the
register
                @(cross(V(clk) - clk_threshold, -1))
                begin
   if(sample_en == 1) begin
      sample_en = 0; end

   if(sar_counter == 0) begin
      reg_out = 1; end
  end
```

```
    V(d7) <+ transition(d_7,delay,ttime);
    V(d6) <+ transition(d_6,delay,ttime);
    V(d5) <+ transition(d_5,delay,ttime);
    V(d4) <+ transition(d_4,delay,ttime);
    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);
    V(regclk) <+ transition(reg_out,delay,ttime);
    V(sampleclk) <+ transition(sample_en,delay,ttime);

end

endmodule
```

4) Output Register (8-bit)

```
// VerilogA for SAR_VerilogA, VerilogA_Register_8bit, veriloga

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

module
VerilogA_Register_8bit(clk,in0,in1,in2,in3,in4,in5,in6,in7,out0,out1,out
2,out3,out4,out5,out6,out7,vdd,vss);

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

inout vdd,vss;
input clk,in0,in1,in2,in3,in4,in5,in6,in7;
output out0,out1,out2,out3,out4,out5,out6,out7;

electrical
clk,in0,in1,in2,in3,in4,in5,in6,in7,out0,out1,out2,out3,out4,out5,out6,o
ut7,vdd,vss;
real d_0,d_1,d_2,d_3,d_4,d_5,d_6,d_7;

analog begin
        @(cross(V(clk) - clk_threshold, +1))
        begin
                d_7 = V(in7);
                d_6 = V(in6);
                d_5 = V(in5);
                d_4 = V(in4);
                d_3 = V(in3);
                d_2 = V(in2);
                d_1 = V(in1);
                d_0 = V(in0);
        end

        V(out7) <+ transition(d_7,delay,ttime);
        V(out6) <+ transition(d_6,delay,ttime);
        V(out5) <+ transition(d_5,delay,ttime);
        V(out4) <+ transition(d_4,delay,ttime);
        V(out3) <+ transition(d_3,delay,ttime);
        V(out2) <+ transition(d_2,delay,ttime);
        V(out1) <+ transition(d_1,delay,ttime);
        V(out0) <+ transition(d_0,delay,ttime);

end
endmodule
```

5) DAC (8-bit)

```
// VerilogA for VerilogA_DAC_8bit

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

module
VerilogA_DAC_8bit(d0,d1,d2,d3,d4,d5,d6,d7,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,d4,d5,d6,d7;
input vmin, vmax;
output vout;

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

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

analog begin
    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_7+d_6+d_5+d_4+d_3+d_2+d_1+d_0) * ((V(vmax)-
V(vmin))/(256))) + V(vmin) ;

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


endmodule
```

* D Flipflop (with asynchronous Set & Reset)

```
// VerilogA for SAR_VerilogA, VerilogA_D_FlipFlop, veriloga
`include "constants.vams"
`include "disciplines.vams"

module VerilogA_DFlipFlop(D,clk,Q,Qb,set,reset,vdd,vss);

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

inout vdd,vss;
input D,clk,set,reset;
output Q,Qb;

electrical D,clk,Q,Qb,set,reset,vdd,vss;

real d_0,d_0b;

analog begin

  @(cross(V(set) - vtrans, +1)) begin
    d_0 = 1;
    d_0b = 0; end

  @(cross(V(reset) - vtrans, +1)) begin
    d_0 = 0;
    d_0b = 1; end

  @(cross(V(clk) - clk_threshold, +1))
  begin
    if((V(set) < vtrans) && (V(reset) < vtrans) ) begin
      if(V(D) > vtrans) begin
      d_0 = 1; d_0b = 0; end

      else begin
      d_0 = 0; d_0b = 1; end
    end
  end


  V(Q) <+ transition(d_0,delay,ttime);
  V(Qb) <+ transition(d_0b,delay,ttime);
end

endmodule
```