IL2239 Course Project

Jordi Altayó
jordiag@kth.se

Björn Sunedahl
sunedahl@kth.se

January 2019

Introduction

The goal of this project is to design a SAR-ADC (Successive Approximation Register Analog to Digital Converter) that meets the following specifications:

- Comparator clock: 100 MHz
- SNDR > 28 dB, SFDR > 37 dB
- Technology: 150 nm CMOS
- Supply voltage: 1.8 V
- Input amplitude $V_{\text{in}} = 0.5\,\text{V}_{\text{pp}}$
- A common-mode input voltage in the range $0 \leq V_{\text{in,cm}} \leq 1.8$ V
- Voltage reference value: $V_{\text{ref}} < 1.8$ V
- Switching energy for a full conversion cycle: <30 pJ for $V_{\text{in}} = 300$ mV (DC)
- Resolution: 5 bits

The circuit topology for the SAR-ADC is given in Figure 1.

The project will be carried out in several steps (milestones).

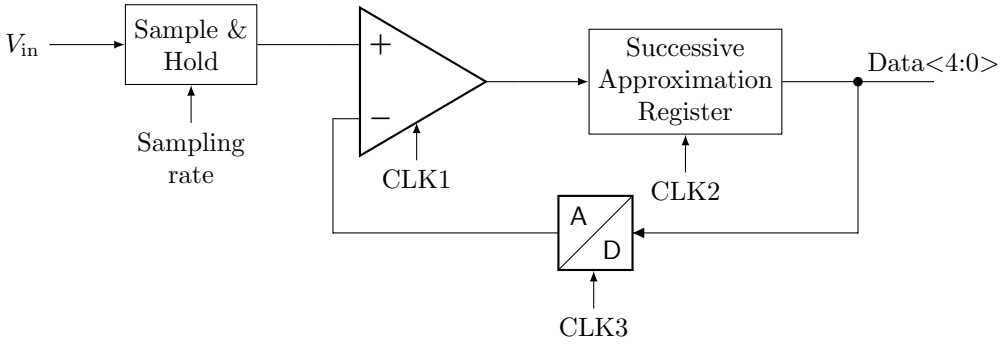


Figure 1: Block diagram of the SAR-ADC

Milestone 1

For this milestone the goals are:

- Design a comparator for the given clock frequency.
- Choose a common-mode input voltage for the comparator, taking into account the properties of the sample & hold circuit.

The comparator used will be of the StrongARM latch topology. To assist us with the design, template projects in Cadence Virtuoso were provided to us.

The schematic for the StrongARM latch comparator is given in Figure 2. In addition to the comparator schematic, two different testbenches (shown in Figures 3-4) were provided.

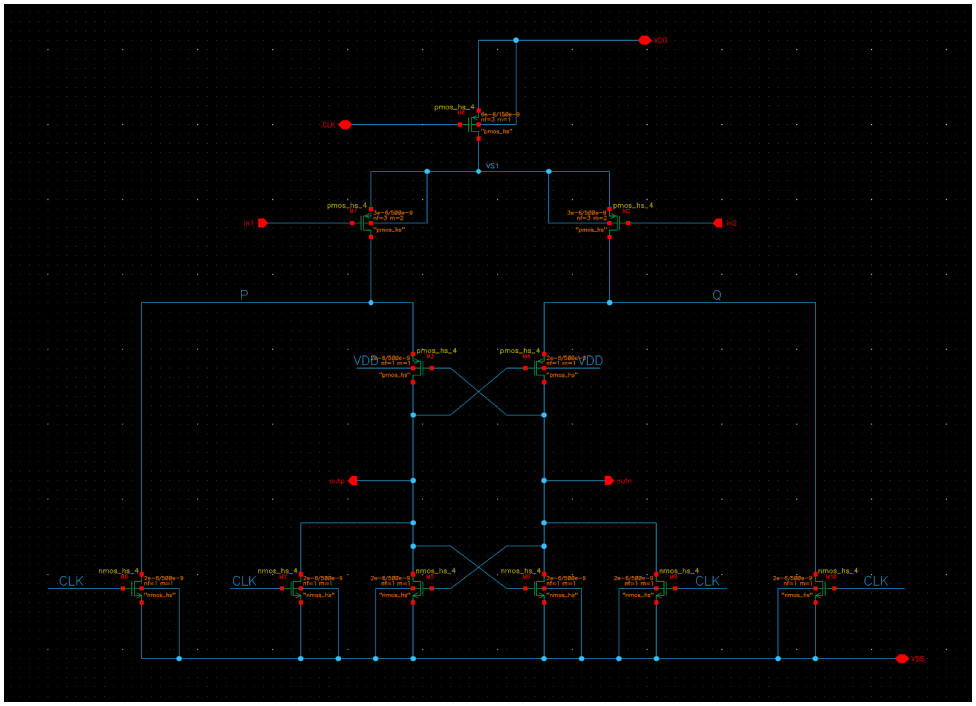


Figure 2: Schematic of the SAR comparator

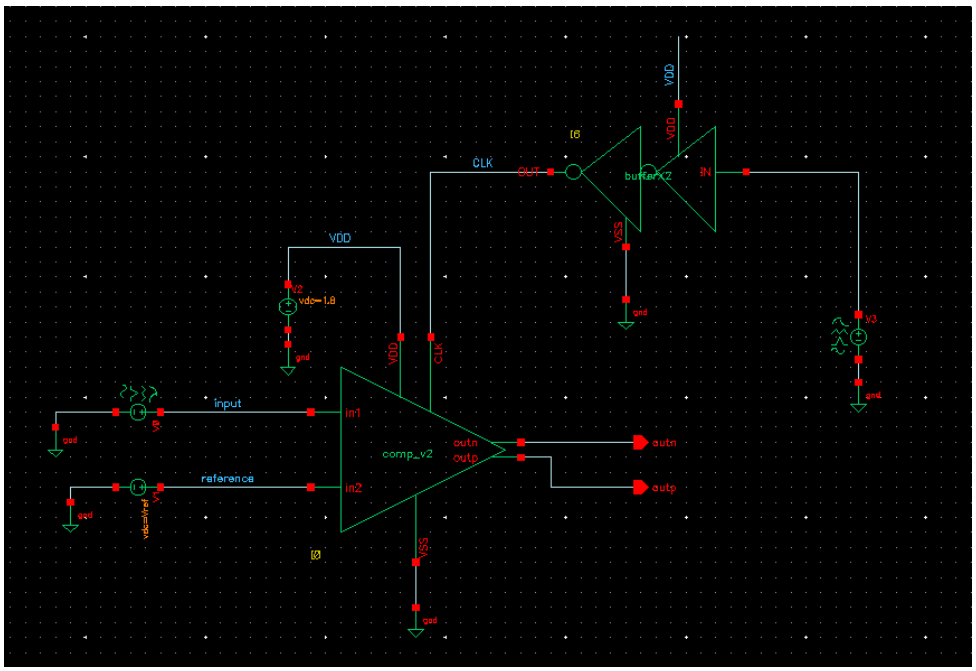


Figure 3: Testbench of the SAR comparator

The purpose of the first testbench in Figure 3 is to verify the basic functionality of the comparator. That is, that $V_{\text{out}}^{-} = V_{\text{DD}}$ and $V_{\text{out}}^{+} = 0$ when $V_{\text{in}}^{-} > V_{\text{in}}^{+}$, and viceversa. The outputs are changed on the negative edge of the input clock and will retain their values during the entire negative half of the clock

cycle. On the positive edge of the clock cycle, both outputs will be changed 0 and these values will be retained during the entire positive half of the clock cycle.

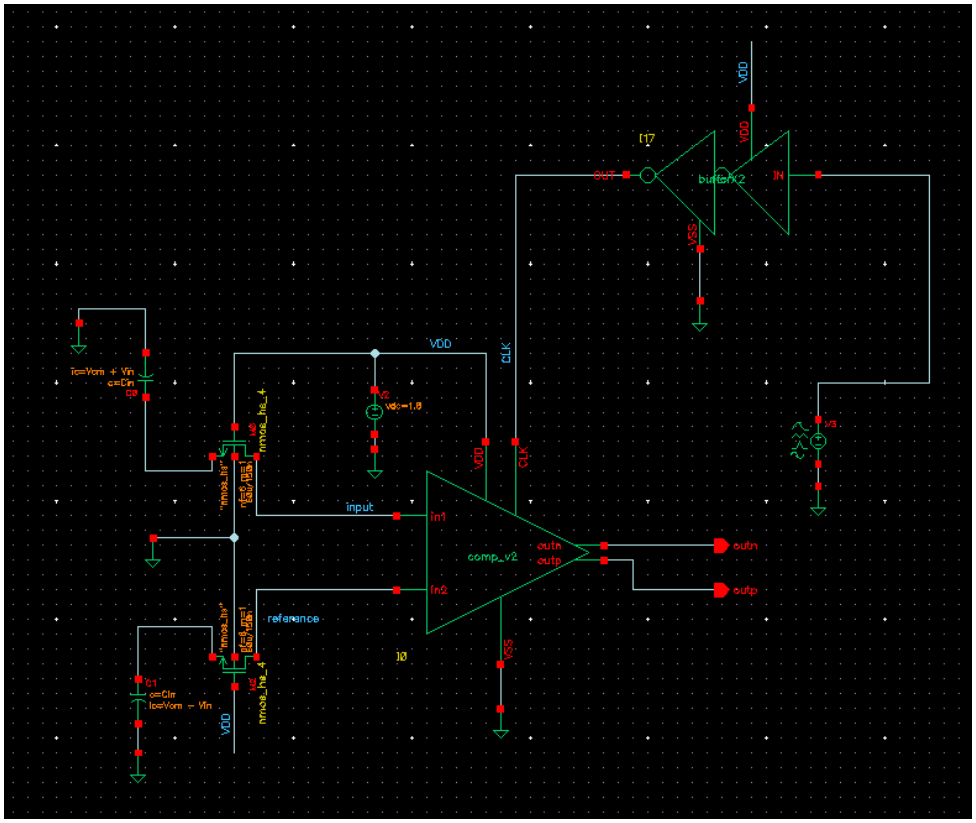


Figure 4: Testbench of the SAR comparator

The second testbench in Figure 4 is mainly useful for determining the amount of kickback noise. Kickback noise is a seen as a voltage drop or spike on the inputs as a result of capacitive coupling between the gate-drain and gate-source of transistors M1 and M2 in Fig. 1. This is an undesirable effect and can cause incorrect comparison results. For a comparator used in a SAR-ADC, the kickback noise must be less than $\frac{0.5/2^n}{2}$V, where n is the desired amounts of bits of resolution that the ADC must be able to handle, in our case 5.

In order to determine the common mode input voltage, we have to look at the sample-and-hold circuit which connects to in1 of the comparator.

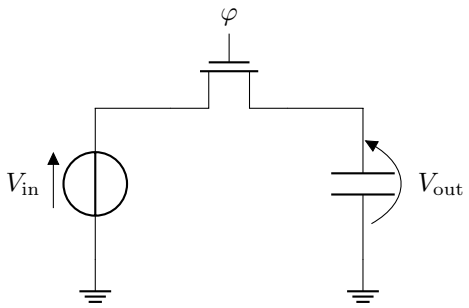


Figure 5: Simple sample & hold circuit

In Figure 5 a simple sample-and-hold circuit is shown. In our case V_{in} will be equal to $V_{\text{cm}} + V_{\text{sample}}$, that is, a common-mode voltage plus the voltage we are interested in sampling. in1 of the comparator is connected to the capacitor. The transistor used as a switch can be considered to be an n-channel MOSFET.

The gate of the transistor will be driven by VDD=1.8, and in order for it conduct, the Vgs voltage must be equal to or greater than the threshold voltage. We can make the assumption that Vth=0.7 V. Further, we can make the assumption that the voltage drop between the drain and source is negligible.

It's easy to see that the voltage across the capacitor can't be higher than 1.1 V, or else the transistor won't conduct. Since Vsample will have a swing of $0.5\,V_{pp}$, the maximum voltage present at the drain of the transistor will be $V_{cm} + 0.25$ V.

Thus, to determine a suitable value for the common mode voltage, we use the following relation:

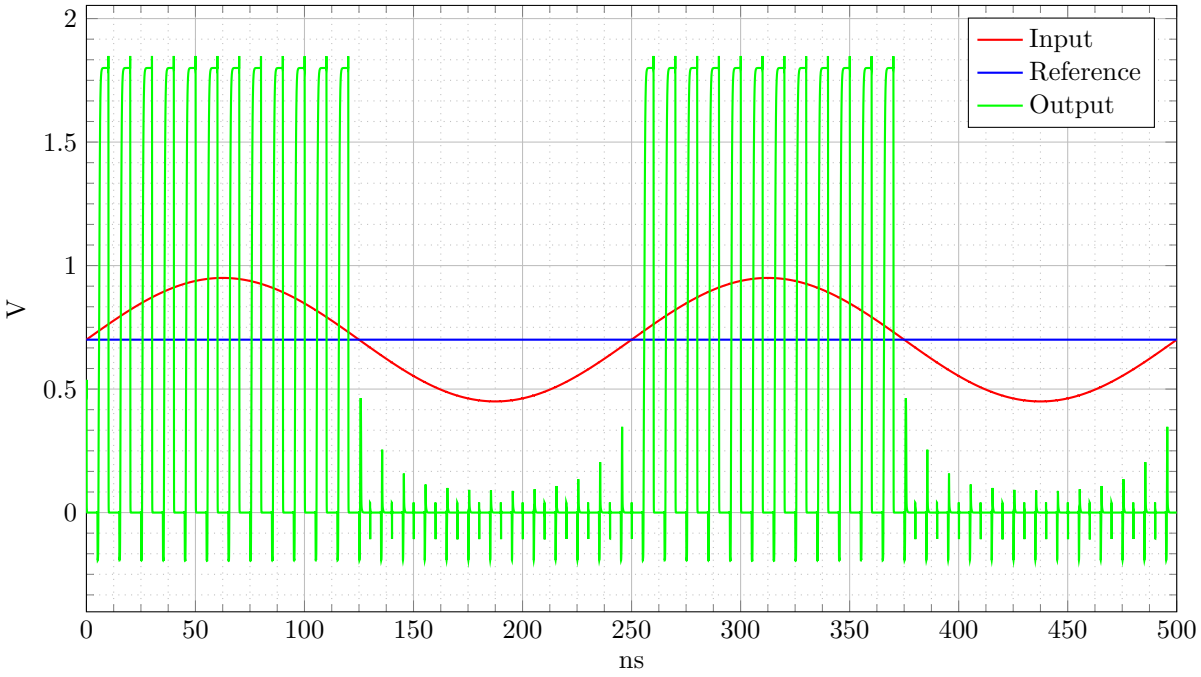
$$V_{cm} + 0.25 = 1.1 \tag{1}$$

Which gives a maximum common mode voltage of $V_{cm} = 0.85$ V. In order to have a little more margin, we choose $V_{cm} = 0.7$ V.

1.1 Transient analysis

To verify the functionality of our circuit we perform a transient simulation where we expect to see the ouput of the comparator changing to high according to the relation $V_{in}^{+} \lessgtr V_{in}^{-}$.

Figure 6: Transient analysis of the SAR comparator



In Figure 6 we can see the results of the transient simulation for two periods of a 4 MHz input sine wave. It is easy to check that the functionality of the comparator is as expected.

1.2 Kickback simulation

Untill this point, the effects of the capacitive coupling between the input and intermediate nodes of the comaprator has been ignored. In reality this effect can lead into significant errors due to the effect of the kickback.

Figure 7 shows the effect of the so called kickback. As we can see the inputs, which are not supposed to change, present a significant variation due to the capacitive coupling through the parasitic capacitances of the transistors. To quantize this effec we use the following expression.

```
(ymax((vtime('tran "/input") - vtime('tran "/reference"))) - (2 * VAR("Vin")))
```

This effect can be visualized in Figure 8. With the current sizes and biasing we obtain a kickback of only 7.1 mV.

Figure 7: Close-up on the kickback noise effects

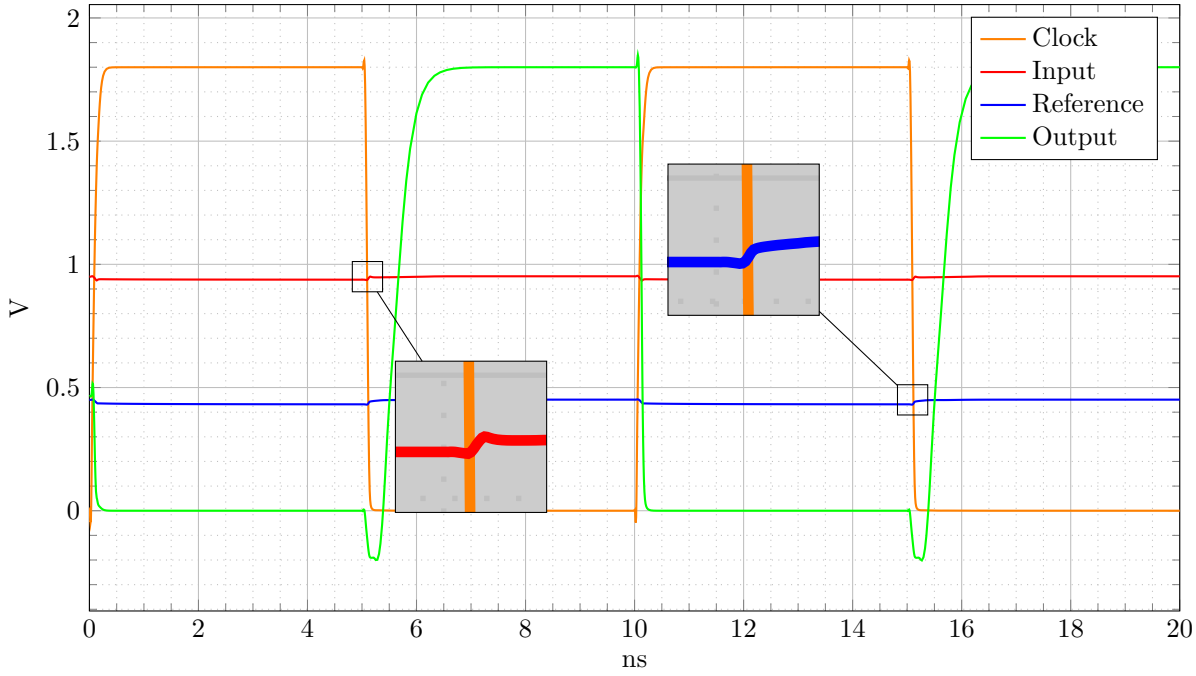
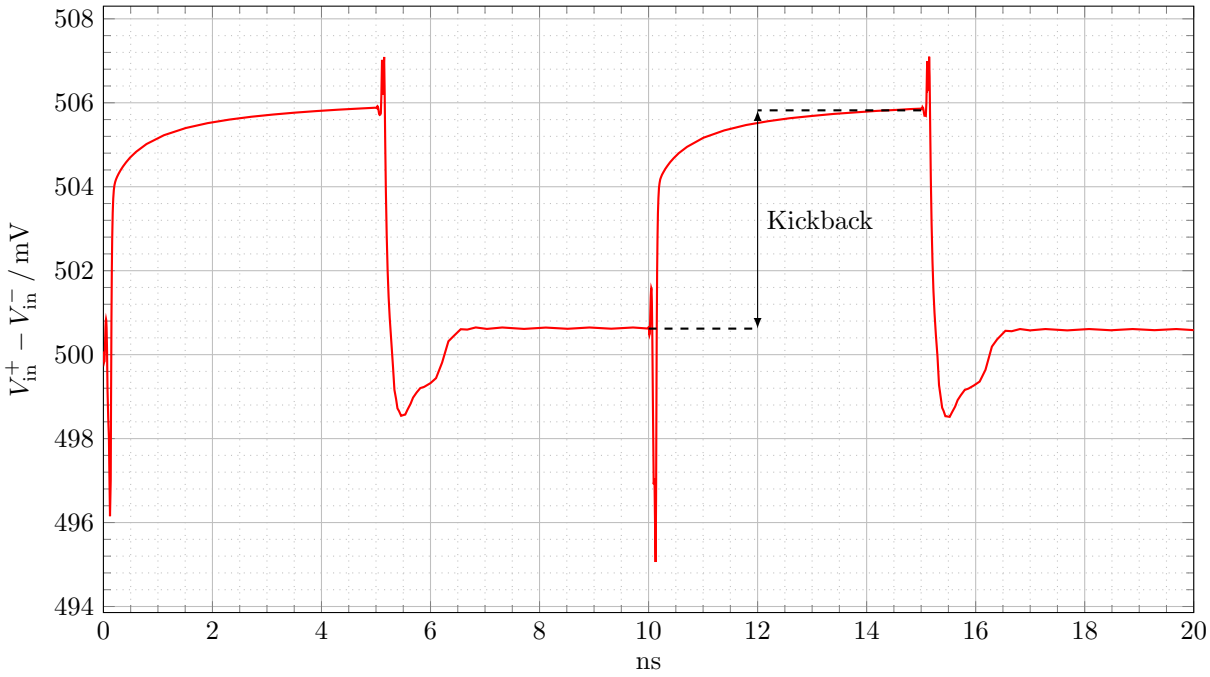


Figure 8: Differential error generated by the kickback



1.3 Noise simulation

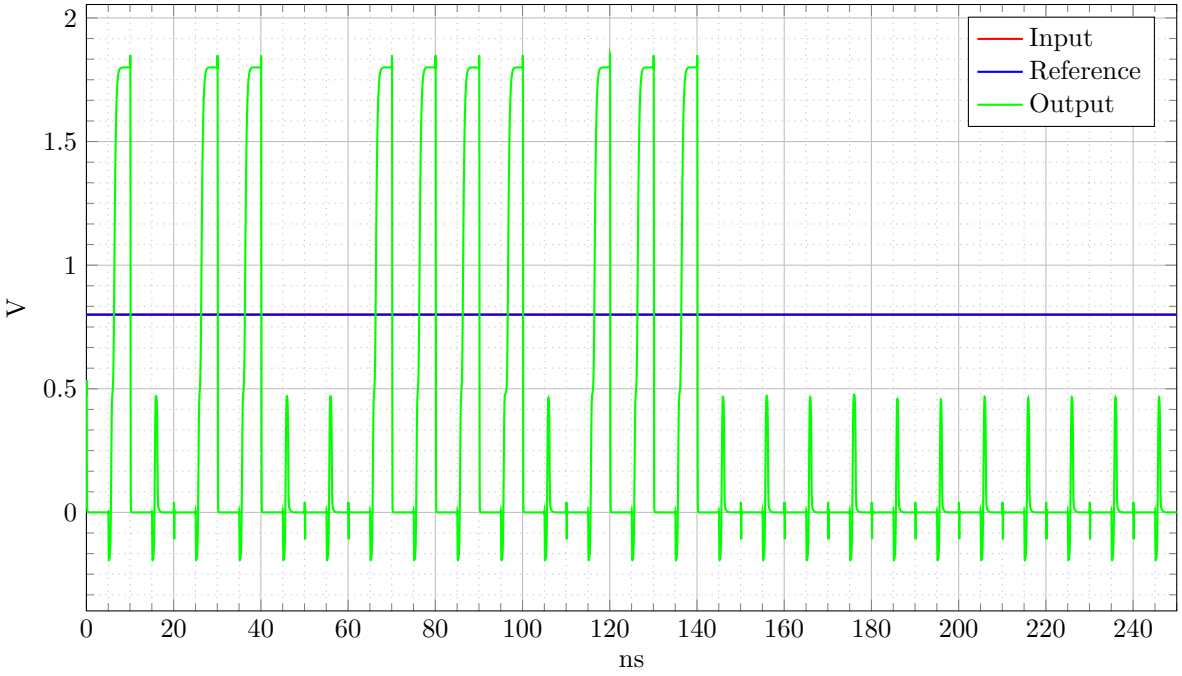
Untill this point, all the simulations were performed with noise-free components. The noise generated by the transistors and passive devices in the comparator can lead to significant error.

To simulate this effects we enable the transient noise and we perform a transient analysis.

When using a input signal of 0.5% of the V_{LSB} the results were identical to Figure 6, this means that the circuit works as expected.

In the other hand, when using a signal of 0.1% of the V_{LSB} the comparator was not working as expected, as can be seen in Figure 9. Note that the input signal cannot be seen in the plot since its amplitude was under 1 mV. This is a extermely low value so we dont have to worry about noise effects

Figure 9: Transient analysis with noise



with this configuration.

1.4 Power Consumption

To obtain the power consumption of our circuit, we can measure the current that is being drawn from the V_{DD} power supply. Since the behaviour of the comaprator is highly dynamic we cant use the DC values of those signals, integration over a period has to be done. Using this method we obtain a power consumption of 76.82 µW.

1.5 HDL Description

The Verilog-AMS description of the comparator can be seen in the Sample Code 1. To account for the propagation delay we used the functions `riseTime` and `fallTime` that are provided by Cadence ADE. We measured a rise time of 574.4 ps and a fall time of 71.03 ps. To simulate this propagation delay we added the delays expressions to the Verilog-AMS code (lines 21 and 29).

Sample Code 1: Verilog-AMS description of the comparator

```
1  `timescale 10ps/1ps
2  `include "constants.vams"
3  `include "disciplines.vams"
4
5  module comp_v2 ( outn, outp, CLK, VDD, VSS, in1, in2 );
6      output outn;
7      inout  VDD;
8      inout  CLK;
9      input  in2;
10     input  in1;
11     inout  VSS;
12     output outp;
13
14     logic CLK;
15     logic outn, outp;
16     electrical VDD, VSS, in1, in2;
17     reg outp, outn;
```

```
18
19      always @(posedge CLK)
20      begin
21          #5
22          outn = 0;
23          outp = 0;
24      end
25      always @(negedge CLK)
26      begin
27          if (V(in1) > V(in2))
28          begin
29              #57
30              outn = 1;
31              outp = 0;
32          end
33          else
34          begin
35              #57
36              outn = 0;
37              outp = 1;
38          end
39      end
40  endmodule
```

1.6 Layout

The layout was based on the layout that was provided. We just changed the sizes of the input transistors and adjusted the connections. Te final layout can be seen in Figure 10.

Milestone 2

In this milestone we will choose the reference voltage that the SAR ADC is going to use as well as calculating the resolution of it.

2.1 Resolution of the SAR ADC

From the simulation results we got a kickback noise of 7.1 mV, and we know that the kickback noise must be at least half of the LSB. We can write this condition as

$$V_{\text{glitch}} < \frac{V_{\text{fullswing}} \cdot 0.5}{2^N} \tag{2}$$

If we solve this for N, we get

$$N = \log_2\left(\frac{V_{\text{fullswing}} \cdot 0.5}{0.0071}\right) = 5.138 \tag{3}$$

Which rounded down is 5, which is the maximum resolution that this ADC can support with this comparator.

2.2 Voltage reference selection

For the voltage reference, we choose $V_{\text{ref}} = 500$ mV. This is the maximum anticipated value on the input. According to the behavioral modeling tutorial, the common mode needs to be half of this. This means we might have to revise our initial choice of $V_{\text{cm}} = 700$ mV as we go along in this process

Milestone 3

For this milestone we need to analyze the behaviour of the sucecssive approximation register and determine the sample rate of the ADC.

3.1 Resolution of the SAR ADC

In order to obtain a resolution of 5 bits, we need to modify the provided Verilog code to adapt it to 5 bits resolution. This modification is achieved by adding a 5th state to the state machine performing the binary search algorithm. This modification can be seen in Sample Code 2.

Sample Code 2: Verilog description of the SAR

```
module SAR4 ( reset, clock, comp, start, SW_inp, SW_ref, DAC_out4, DAC_out3, DAC_out2,
↪  DAC_out1, DAC_out0, out4, out3, out2, out1, out0, eoc, dclk );

  output eoc;
  output SW_inp;
  output SW_ref;
  output out0;
  output out1;
  output out2;
  output out3;
  output out4;
  output DAC_out0;
  output DAC_out1;
  output DAC_out2;
  output DAC_out3;
  output DAC_out4;

  output dclk;

  input start;
  input reset;
  input comp;
  input clock;

  reg eoc;
  reg SW_inp = 1'b0;
  reg SW_inp_reg = 1'b0;
  reg SW_ref = 1'b0;
  reg out0 = 1'b0;
  reg out1 = 1'b0;
  reg out2 = 1'b0;
  reg out3 = 1'b0;
  reg out4 = 1'b0;
  reg DAC_out0 = 1'b0;
  reg DAC_out1 = 1'b0;
  reg DAC_out2 = 1'b0;
  reg DAC_out3 = 1'b0;
  reg DAC_out4 = 1'b0;

parameter [2:0] idle = 3'b000;
parameter [2:0] sample = 3'b001;
parameter [2:0] convInit= 3'b010;
parameter [2:0] conv4 = 3'b011;
parameter [2:0] conv3 = 3'b100;
parameter [2:0] conv2 = 3'b101;
parameter [2:0] conv1 = 3'b110;
parameter [2:0] endConv = 3'b111;

reg [2:0] state;        //        Current State
reg [2:0] next;         //        Next State

assign dclk = clock;
```

```
52
53  always @(negedge clock)
54  begin
55          if (reset) state <= idle;
56          else state <= next;
57
58  end
59
60  always @(state or start)
61  begin
62          next = idle;
63          eoc = 1'b0;
64          case (state)
65          idle : begin
66                  if (~start)
67                  begin
68                  next = idle;
69                  DAC_out0 = 1'b0;
70                  DAC_out1 = 1'b0;
71                  DAC_out2 = 1'b0;
72                  DAC_out3 = 1'b0;
73                  DAC_out4 = 1'b0;
74                  out0 = 1'b0;
75                  out1 = 1'b0;
76                  out2 = 1'b0;
77                  out3 = 1'b0;
78          out4 = 1'b0;
79                  eoc = 1'b0;
80                  SW_inp_reg = 1'b0;
81                  SW_ref = 1'b0;
82                  end
83                  else if (start == 1)
84                  next = sample;
85                  end
86          sample : begin
87                  next = convInit;
88                  SW_inp_reg = 1'b1;        //Connects the bottom plate of the caps to
                    ↪  Vin
89                  SW_ref = 1'b0;        //Connects the bottom plate of the caps to Vref.
                    ↪  Should not overlap with SW_in as DAC_out drops (charges escape to
                    ↪  CM)
90                  eoc = 1'b0;
91                  DAC_out0 = 1'b1;
92                  DAC_out1 = 1'b1;
93                  DAC_out2 = 1'b1;
94                  DAC_out3 = 1'b1;
95                  DAC_out4 = 1'b1;
96                  end
97          convInit : begin
98                  DAC_out0 = 1'b0;
99                  DAC_out1 = 1'b0;
100                 DAC_out2 = 1'b0;
101                 DAC_out3 = 1'b0;
102                 DAC_out4 = 1'b1;
103                 SW_inp_reg = 1'b0;
104                 SW_ref = 1'b1;
105                 eoc = 1'b0;
106                 next = conv4;
```

```
107                 end
108     conv4 : begin
109                 DAC_out0 = 1'b0;
110                 DAC_out1 = 1'b0;
111                 DAC_out2 = 1'b0;
112                 DAC_out3 = 1'b1;
113         DAC_out4 = comp;
114                 SW_inp_reg = 1'b0;
115                 SW_ref = 1'b1;
116                 next = conv3;
117                 end
118         conv3 : begin
119                 DAC_out0 = 1'b0;
120                 DAC_out1 = 1'b0;
121                 DAC_out2 = 1'b1;
122                 DAC_out3 = comp;
123                 SW_inp_reg = 1'b0;
124                 SW_ref = 1'b1;
125                 next = conv2;
126                 end
127         conv2 : begin
128                 DAC_out0 = 1'b0;
129                 DAC_out1 = 1'b1;
130                 DAC_out2 = comp;
131                 SW_inp_reg = 1'b0;
132                 SW_ref = 1'b1;
133                 next = conv1;
134                 end
135         conv1 : begin
136                 DAC_out0 = 1'b1;
137                 DAC_out1 = comp;
138                 SW_inp_reg = 1'b0;
139                 SW_ref = 1'b1;
140                 next = endConv;
141                 end
142         endConv : begin
143                 DAC_out0 = comp;
144                 SW_inp_reg = 1'b0;
145                 SW_ref = 1'b1;
146                 eoc = 1'b1;
147                 next = sample;
148                 out0 = DAC_out0;
149                 out1 = DAC_out1;
150                 out2 = DAC_out2;
151                 out3 = DAC_out3;
152                 out4 = DAC_out4;
153                 end
154         endcase
155
156 end
157
158 always @(clock or SW_inp_reg) SW_inp = SW_inp_reg*(~clock);
159
160 endmodule
```

3.2 Top level simulations

After doing the modifications to the SAR description, we can proceed to perform a top level simulation of all the blocks in our system. To do that we use the testbench seen in Figure 11. The DAC used in the feedback loop of the SAR ADC (see Figure 1) provided consisted only of 4 bits so an aditional branch of capacitors. The final schematic of the DAC can be seen in Figure 13.

Performing a transient simulation with an input signal and observing the output of the ADC we can see the conversion results and observe the offset between the reconstructed signal and the input signal. This results can be seen in Figure 12. To reconstruct the digital signal an ideal DAC was used.

3.3 Sampling Rate

To determine the sampling rate of the ADC we need to analyze the how the convertion is done. The SAR takes N clock samples to converge to a final value and an aditional two clocks are needed for the setteling of the signal and additional logic and reset phases. Thus we can expres the sampling rate of our ADC as.

$$f_{\text{sample}} = \frac{f_{\text{clk}}}{N+2} \tag{4}$$

In our project $f_{\text{clk}} = 100\,\text{MHz}$ and $N = 5$ so $f_{\text{sample}} = 14.29\,\text{MHz}$.

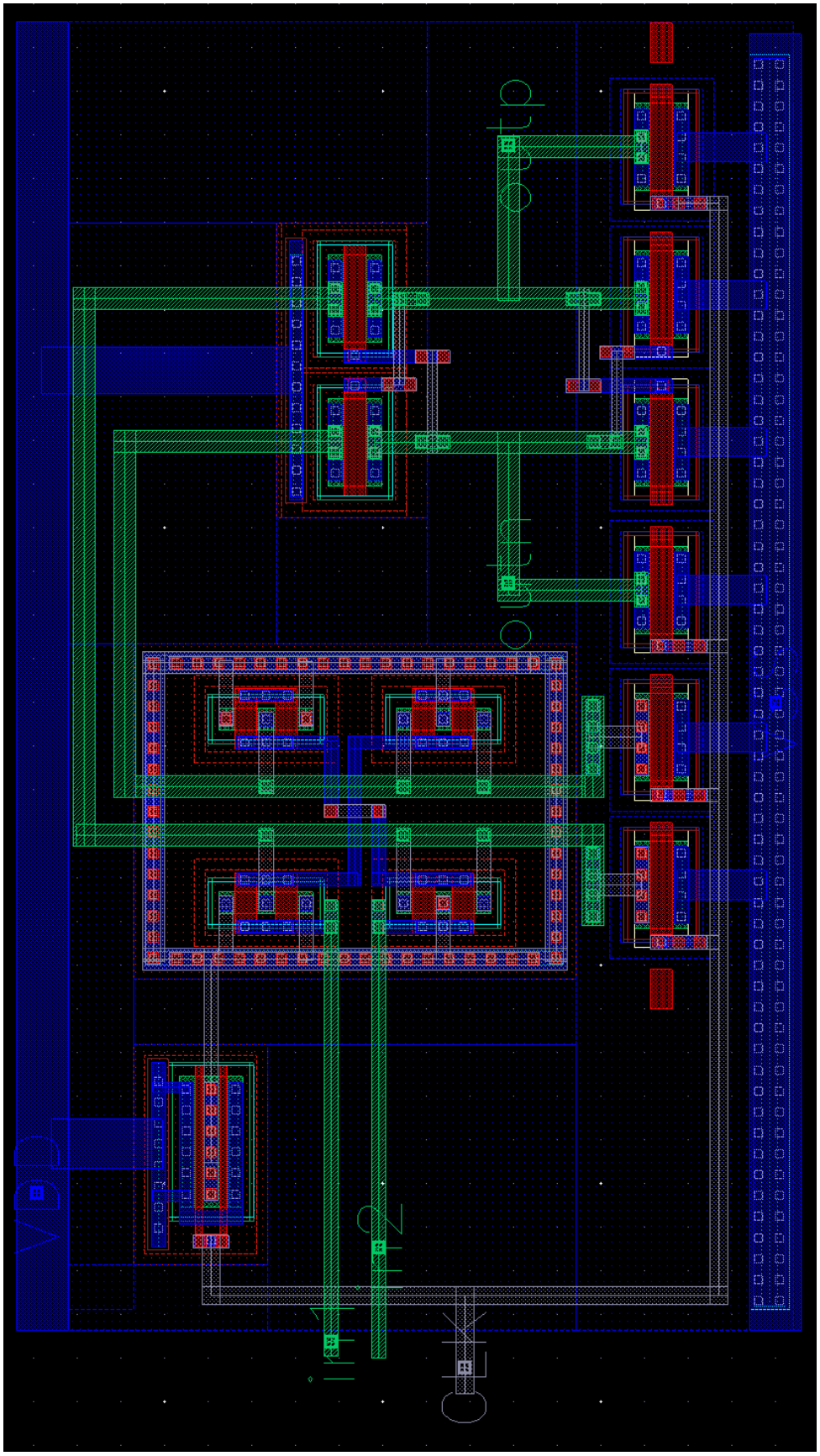


Figure 10: Final layout of the ADC

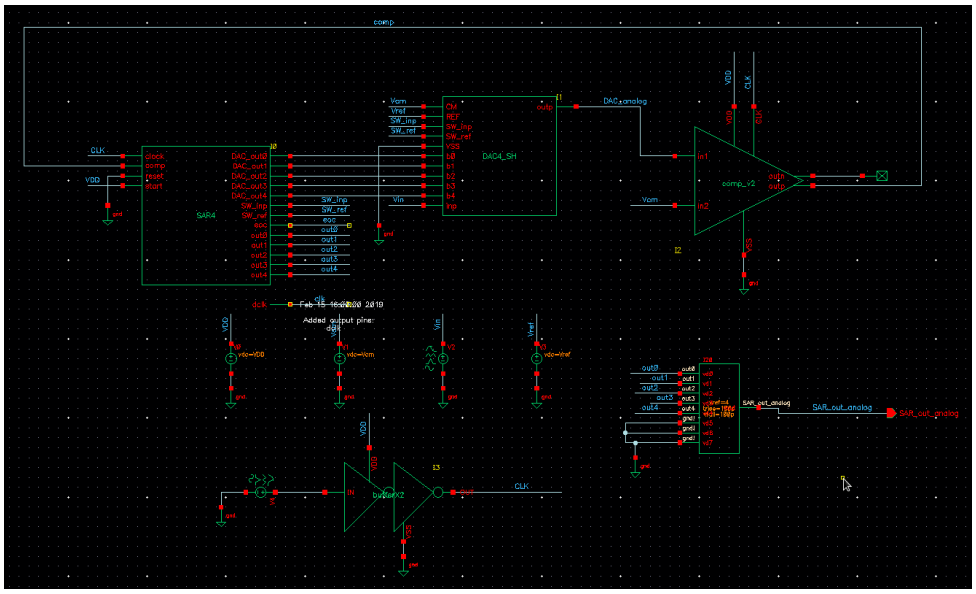


Figure 11: SAR testbench

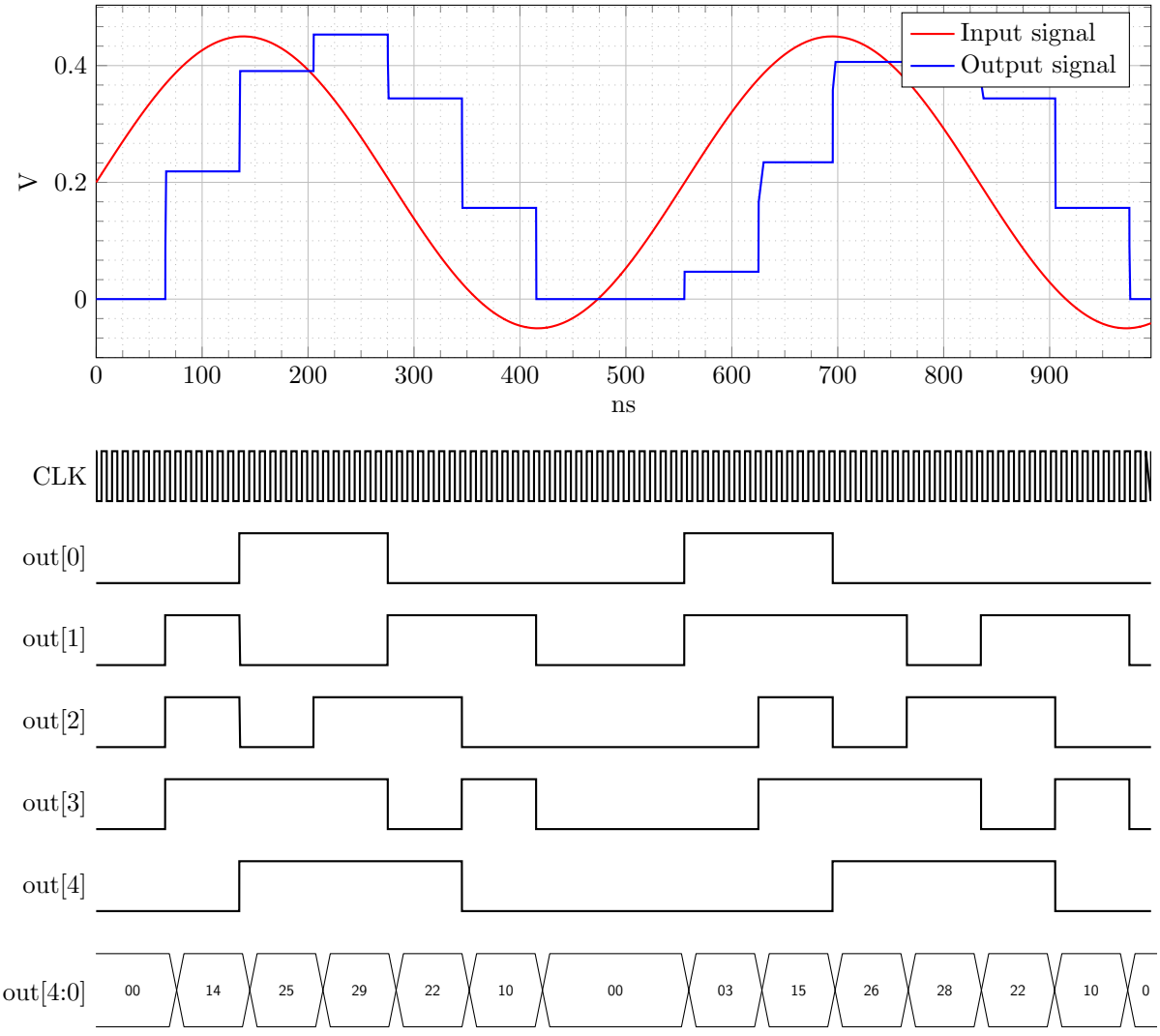


Figure 12: Waveforms from a sinusoidal signal conversion

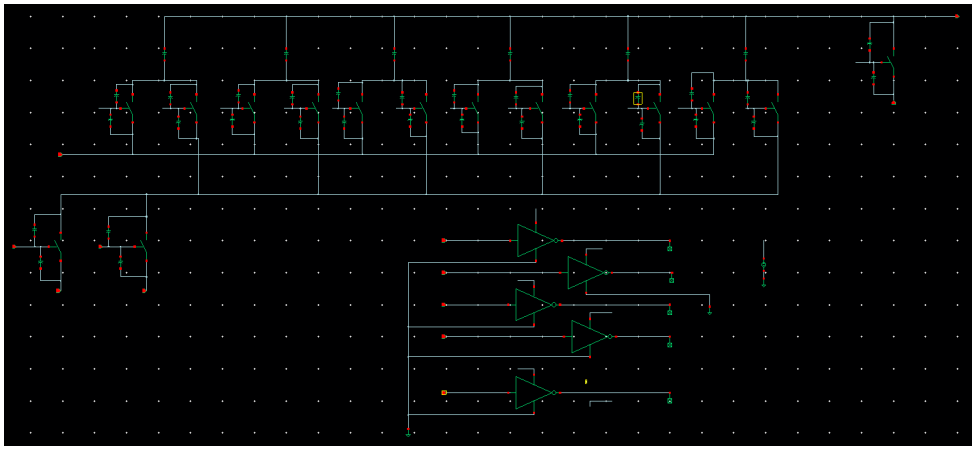


Figure 13: DAC Schematic

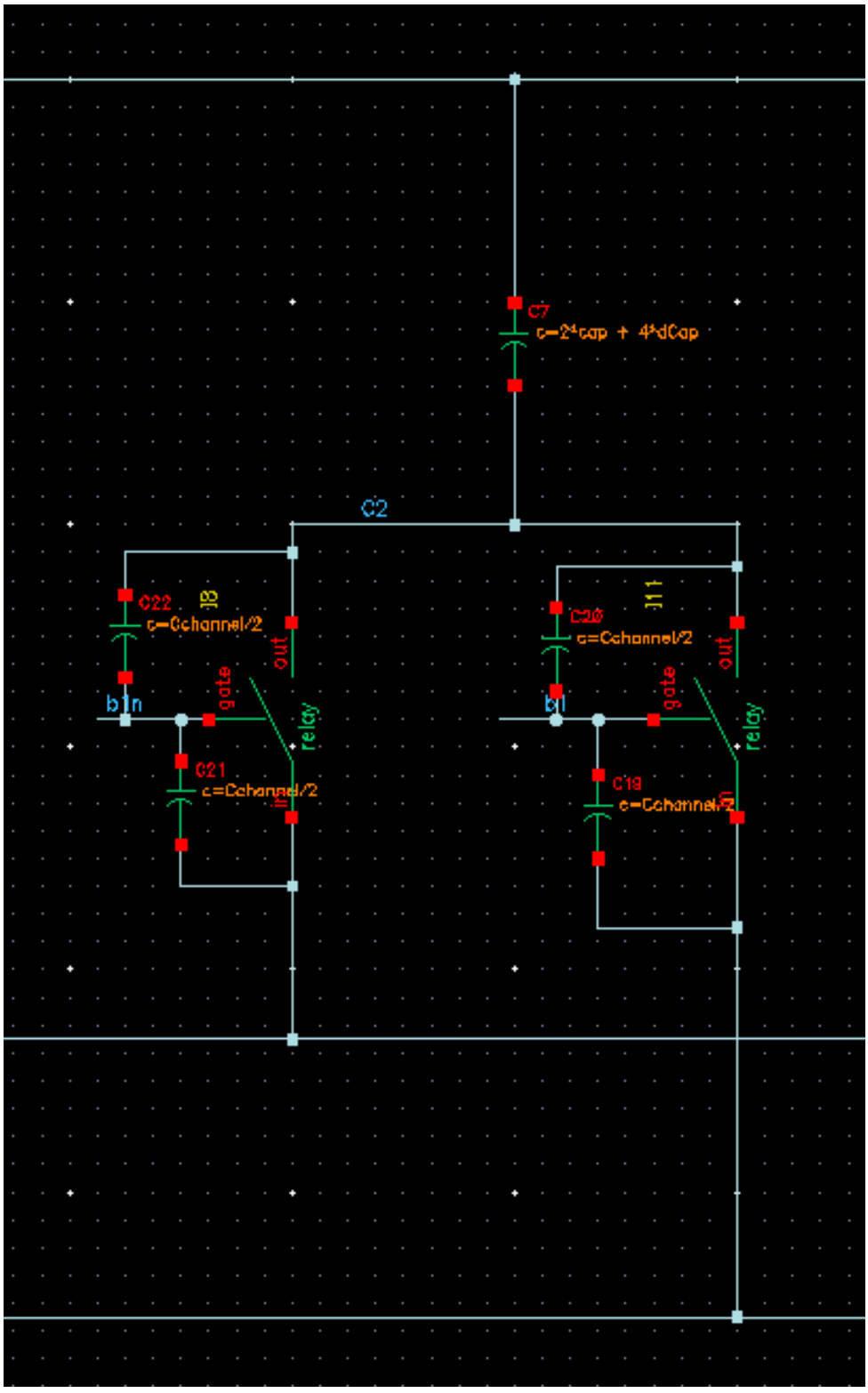


Figure 14: SAR testbench