Term Project
Gizmonic Smart-Watch Application

Lukas Krampitz 1192170
Eric Yates 1171802

2026–06–07

Date performed:	2025–11–27
Course identifier:	ENGG*4080
Course section:	0101
Course title:	Micro and Nano-Scale Electronics
Instructor:	Professor Stefano Gregori

1 Introduction

The integrated circuit design for an electrocardiogram (ECG) in a smart watch application has been divided into three modules: an analogue multiplexer, a programmable gain amplifier, and a 6-bit analogue to digital converter. The module designs are all to be implemented using the TSMC 65-nm CMOS technology. The watch platform has a RISC core which will provide control signals for the analogue modules. The designs were completed and simulated using Cadence Virtuoso. This report details the design process of each module including key design decision, schematics, and simulations.

2 Summary Design Decisions

2.1 Multiplexer

Transmission gates were used for the analogue Multiplexer to ensure a robust design that would maintain performance at any operating point, and while crossing 0 volts. **Binary coded inputs** were chosen over a single enable line per signal to have a more efficient use of area and prevent the possibility of enabling multiple input lines, which would risk overloading the system. A **Ground state** was chosen to be the fourth possibility given the use of two binary coded select lines.

2.2 Programmable Gain Amplifier

The **two stage op-amp design** was chosen as the core of the PGA for the noise handling benefits of the first stage differential amplifier, and simplicity and high gain of a common source second stage. Implementing this well understood configuration suited the design specification and timeline. The operating point of the op-amp was tuned to be the input signal common mode of 0.5 V. To realize the programmable gain levels of 1,2,3, and 4 a **switched capacitor feedback** circuit was used. The switched capacitors offer an integrated circuit solution to creating large resistances that is more compact and more accurate than the use of off chip resistors. However, to get the smooth continuous signal desired relatively larger **off chip capacitors** were used as allowed by the design specifications.

2.3 Analogue to Digital Converter

A 6-bit successive-approximation charge redistribution topology was designed as specified in the specification. The design decision was made to **maximize power efficiency** by not exceeding the specified sampling rate of 10^5 samples per second or 100 kHz. However, to ensure that the charge redistribution could take place and accommodate the as yet unknown circuitry of the digital team a clock rate of 1 MHz was used for digital simulation. Using a **clock rate of 1 MHz** with a period of $1\mu s$ the conversion logic takes place over 10 cycles. The purpose of each cycle is listed below. Although this may not be the exact implementation of the digital team it provides ample time and ensures that the specified sample rate is attainable. The final provided resolution specification of 15.625 mV per bit did not match the input signal specification well so the the design decision was made to tighten the envelope in which the discretization takes place to increase resolution. This was accomplished by changing the positive and negative reference voltages to be closer to the common mode of the signal. Rather than use 1 V and 0 V as references providing a resolution of 15.625 mV

over the range of 0-1 V the design team is proceeding with 0.625 V and 0.375 V creating a 250 mV wide envelope around the signal common mode with a resolution of 3.9 mV per bit. It is possible that the narrower envelope may not suit the other inputs of the multiplexor aside from the ECG, in which case it our team proposes programmable resolution by changing the reference voltages. The trade-off of programmable resolution of course increased area and power consumption which may not be suitable. For the provided information our team believes this is the optimal operating envelope.

6-Bit SAR ADC Conversion Cycle

1. Sample
2. Hold
3. Switch bit 5
4. Compare bit 5, switch if needed, then switch bit 4
5. Compare bit 4, switch if needed, then switch bit 3
6. Compare bit 3, switch if needed, then switch bit 2
7. Compare bit 2, switch if needed, then switch bit 1
8. Compare bit 1, switch if needed, then switch bit 0
9. Compare bit 0
10. Done: update binary outputs to match switch states

3 Hand Calculations

3.1 Programmable Gain Amplifier

The notes below seen in Fig. 1 and Fig. 2 summarize the theory and equations that were central the design of the PGA. Much of the thought process was built upon the recommended readings from the recommended text. Furthermore, from discussion with Dr. Gregori and Masoud additional thought processes were added and ironed out. There is a core trade off in the tuning of the op-amp at the core of the PGA. It has a compensation resistance and capacitance that must be tuned to the operating envelope of the design and the trade off of this began to be considered at this stage. Additionally some initial calculations were made to find the best trade off in current division from the current mirror. The thought process for this was to keep a steady but lower gain in the initial differential stage, to reduce the noise floor, and have a larger gain in the common source second stage to bring the signal as high as possible. Also a part of this though process were some hand calculations around the trade off for the centre of the envelope. It was decided to try and get the full swing of the op-amp centred around 0.5V inline with the ECG signal.

3.2 Analogue to Digital Converter

The hand calculations seen below in Fig. 3 and Fig. 4 demonstrate, using a 2-bit example, how the charge distribution design can have the resolution altered by changing the reference voltages. In these hand-calculation the thought process involved finding the best arrangement for the charge redistribution capacitors and make efficient use of the schematic area. Using a simplified 2-bit design also allowed for the thought process to permeate through the whole process of successive approximation and allow for any trade-off to be analysed. One such trade off is balancing the range of valid inputs to the resolution of the final ADC. Either way the design is limited to 6-bits. If the ADC is designed to convert analogue values from 0V to 1V then the value of the LSB is going to be large. If the range is reduced to be much tighter then the value of the LSB can be much smaller. The result of this early thought process can be seen in the trade off selected in Section 5.6 **Top Level** where the choices made to arrive at the final design are discussed with more detail.

PGA Hand calculations to guide design and simulation

Specifications: - op-amp open circuit gain of 40dB
- implement switched capacitors

Convert dB to linear scale

$$40dB = 20\log_{10}\left(\frac{V_{out}}{V_{in}}\right)$$

$$\frac{40}{20} = \log_{10}\left(\frac{V_{out}}{V_{in}}\right)$$

$$10^2 = \frac{V_{out}}{V_{in}} \Rightarrow \text{linear gain } A = 100$$

Two Stage Op-Amp schematic Fig. 6.3

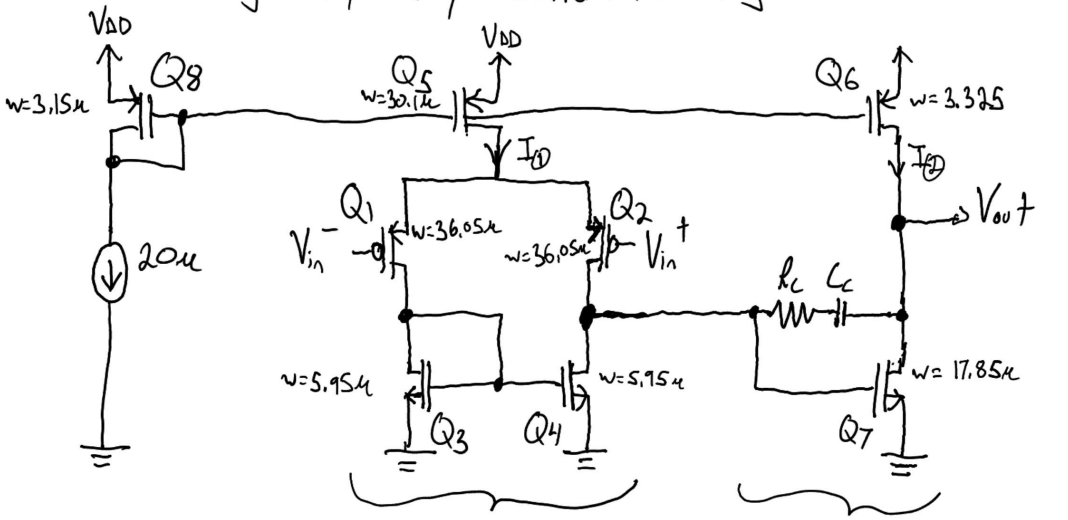


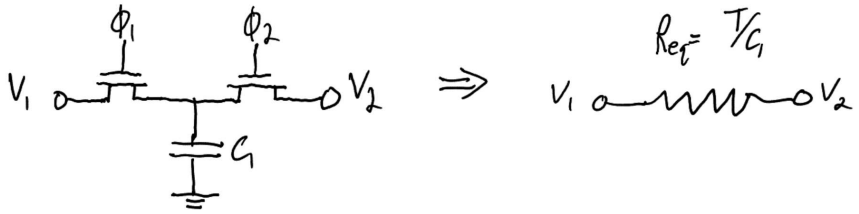
Figure 1: PGA Notes – Page 1

Gain equations by stage

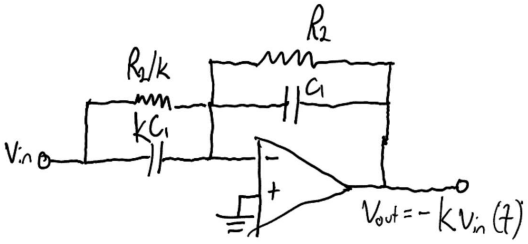
$A_{①} = -g_{m_1}(r_{ds2} \| r_{ds4})$ $A_{②} = -g_{m_7}(r_{ds6} \| r_{ds7})$

$g_m = \sqrt{2\mu_p C_{ox} \frac{W}{L} I_{bias}}$ $r_{ds} = (g_m)^{-1}$

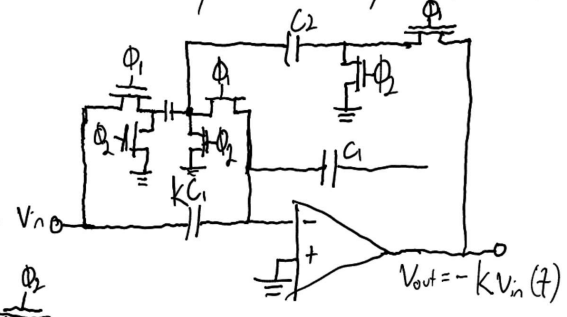
Switched Capacitor Resistance Equivalence Fig 14.4



Active RC Gain



Switched Capacitor Implimentation



Resettable Gain

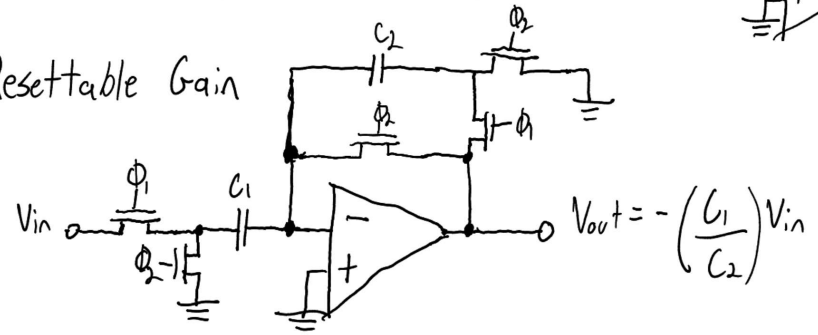


Figure 2: PGA Notes – Page 2

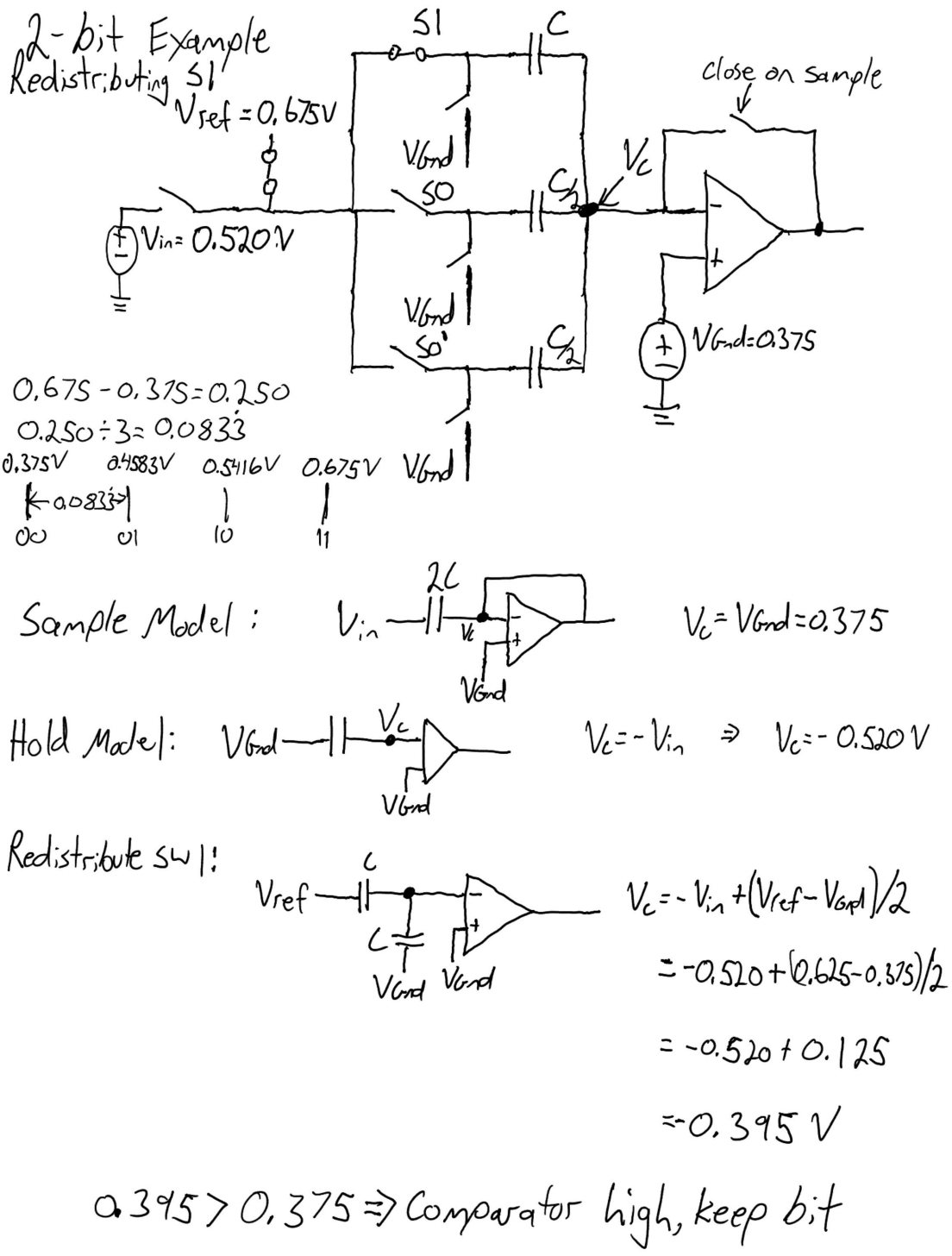
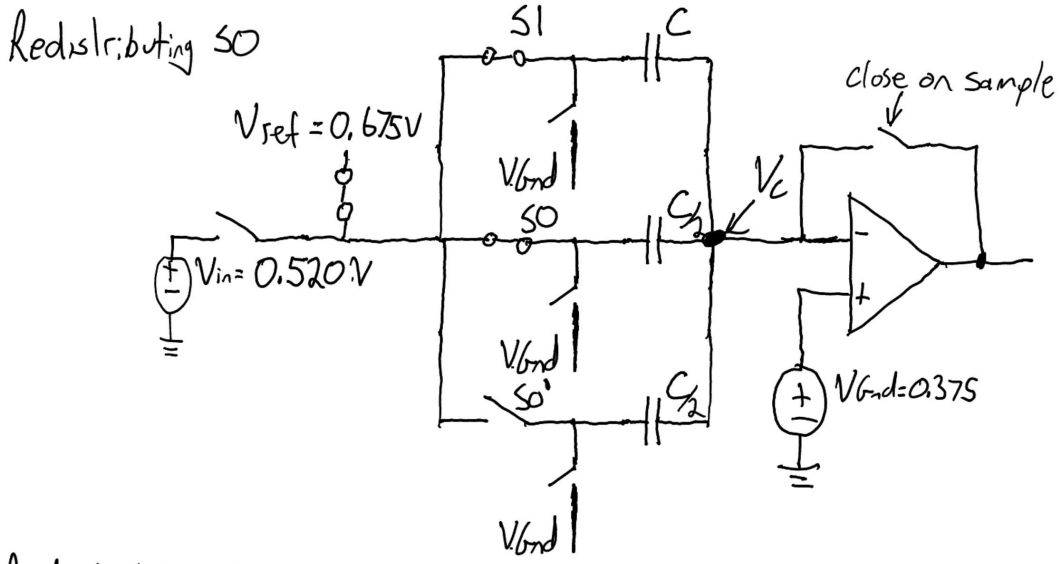
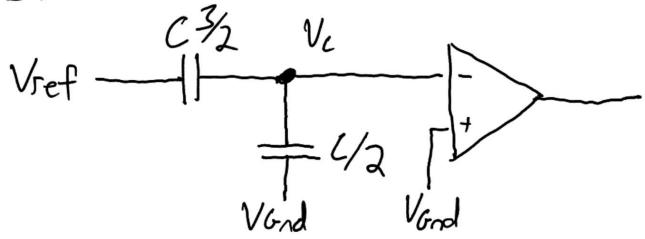


Figure 3: ADC Hand Calculation – Page 1



Redistribute Sw0:



$$V_c = -V_{in} + (V_{ref} - V_{Gnd})\,3/4$$

$$= -0.520 + (0.625 - 0.375)\,3/4$$

$$= -0.520 + (0.250)\,3/4$$

$$= -0.520 + 0.1875$$

$$= -0.3325$$

$0.3325 < 0.375 \Rightarrow$ Comparator low, switch back to VGnd

Figure 4: ADC Hand Calculation – Page 2

4 Final Design

This section features schematics of each module of the design, including the top level implementation. All transistor dimensions are indicated within the schematics, and can simply be read off the figure.

4.1 Multiplexer

This multiplexer uses transistors of the minimum allowed size. The sizes within the NOT inverters is double the minimum allowed dimensions. The schematic can be seen in Fig. 5 below.

4.2 Op-Amp

The Op-Amp schematic can be seen in Fig. 6 below. It makes use of a wider range of transistor lengths and width, all of which can be read off the schematic. Also available are the compensation resistance and capacitors choses for this design. Where $R_C = 5k\Omega$ and $C_C = 10fF$.

4.3 Clock-Splitter

Building up the the PGA it was very important to have a Clock-Splitter. This splitter ensures the PGA have two clock signals of the same frequency that are entirely free of any overlap. The schematic can be seen in Fig. 7 below. This Clock-Splitter is used to provided programmable gain for switched capacitor operation within the PGA.

4.4 Programmable Gain Amplifier

The PGA schematic can be seen in Fig. 8 below. At its core is the Op-Amp from Fig. 6, this is the amplifying stage and the array of switched capacitors allow for the selection of 1x, 2x, 3x, or 4x gain. Also prominent within the PGA is the Clock-Splitter from Fig. 7.

4.5 Analogue to Digital Converter

The successive approximation charge redistribution Analogue to Digital Converter schematic can be seen in Fig. 9 below. At its core is once again the Op-Amp from Fig. 6 this time taking on the role of a comparator. Also prominent within the ADC is the symbol of the control logic provided by the digital team. As they make their updates and improvements the new or updated symbol can be used right way to re-test the ADC.

4.6 Top Level

Bringing all the modules together one can see the top level schematic in Fig. 10 below.

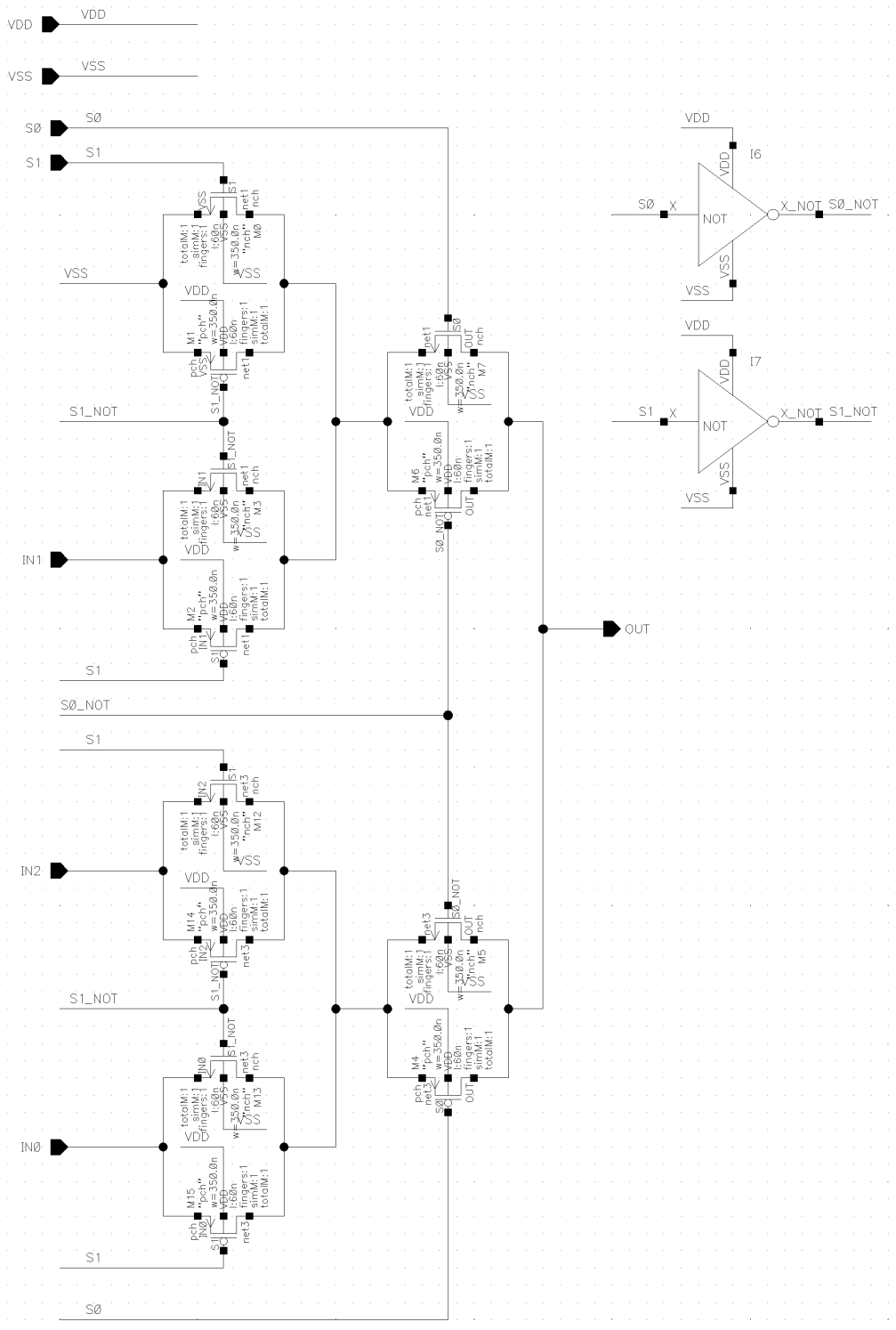


Figure 5: Schematic of the Multiplexer

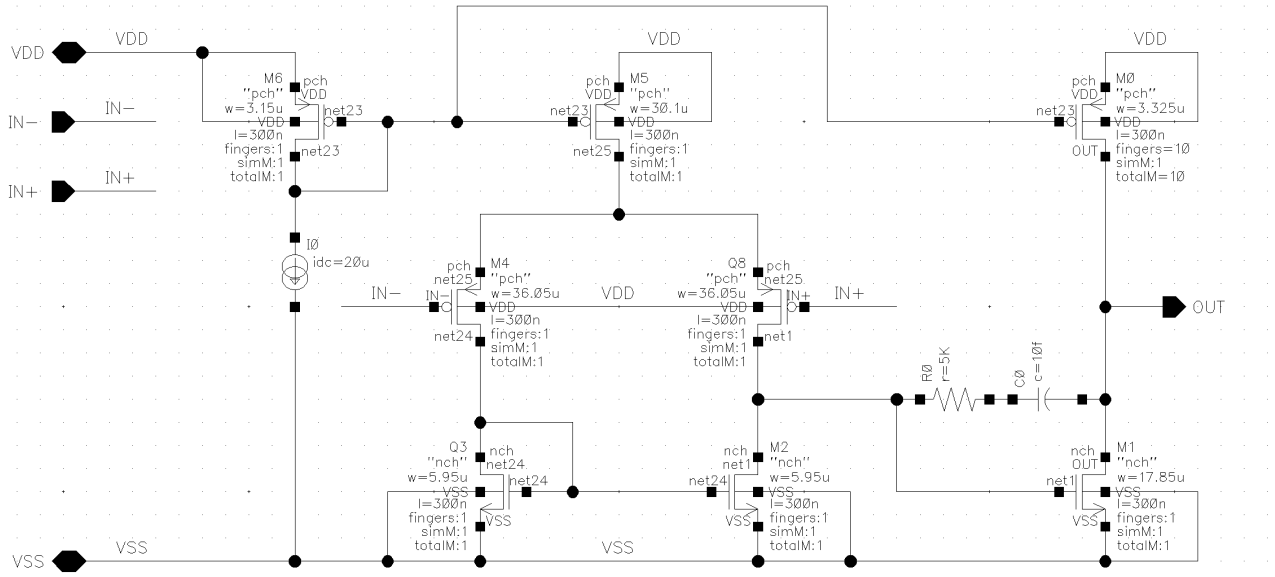


Figure 6: Schematic of the Op-Amp

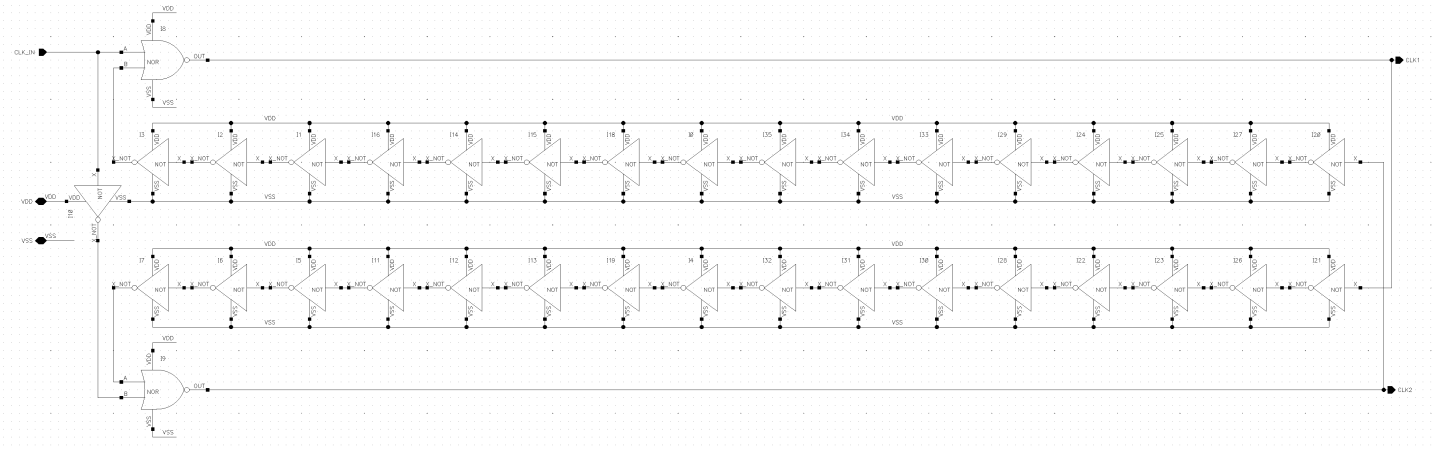


Figure 7: Schematic of the CLK

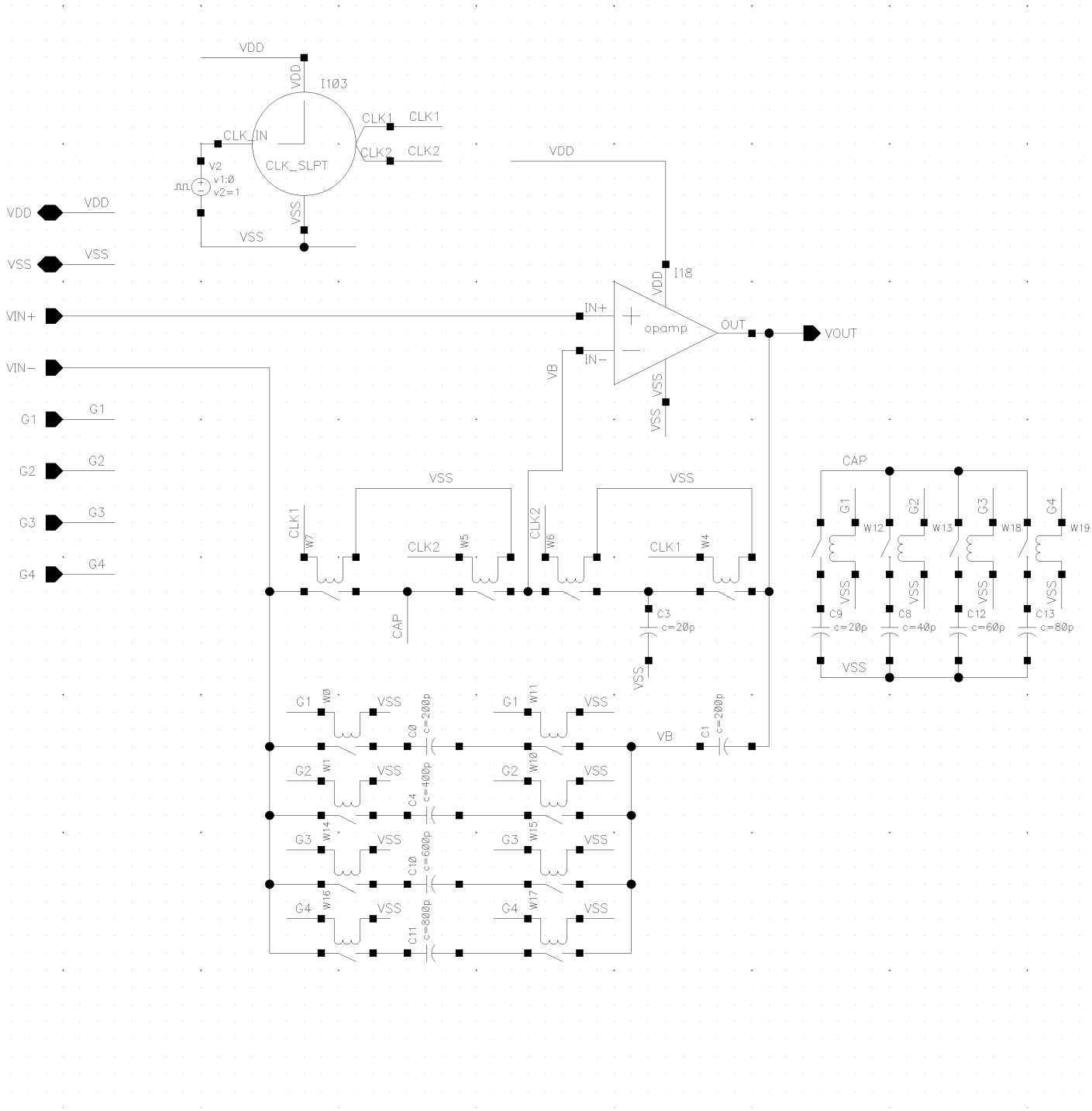


Figure 8: Schematic of the Programmable Gain Amplifier

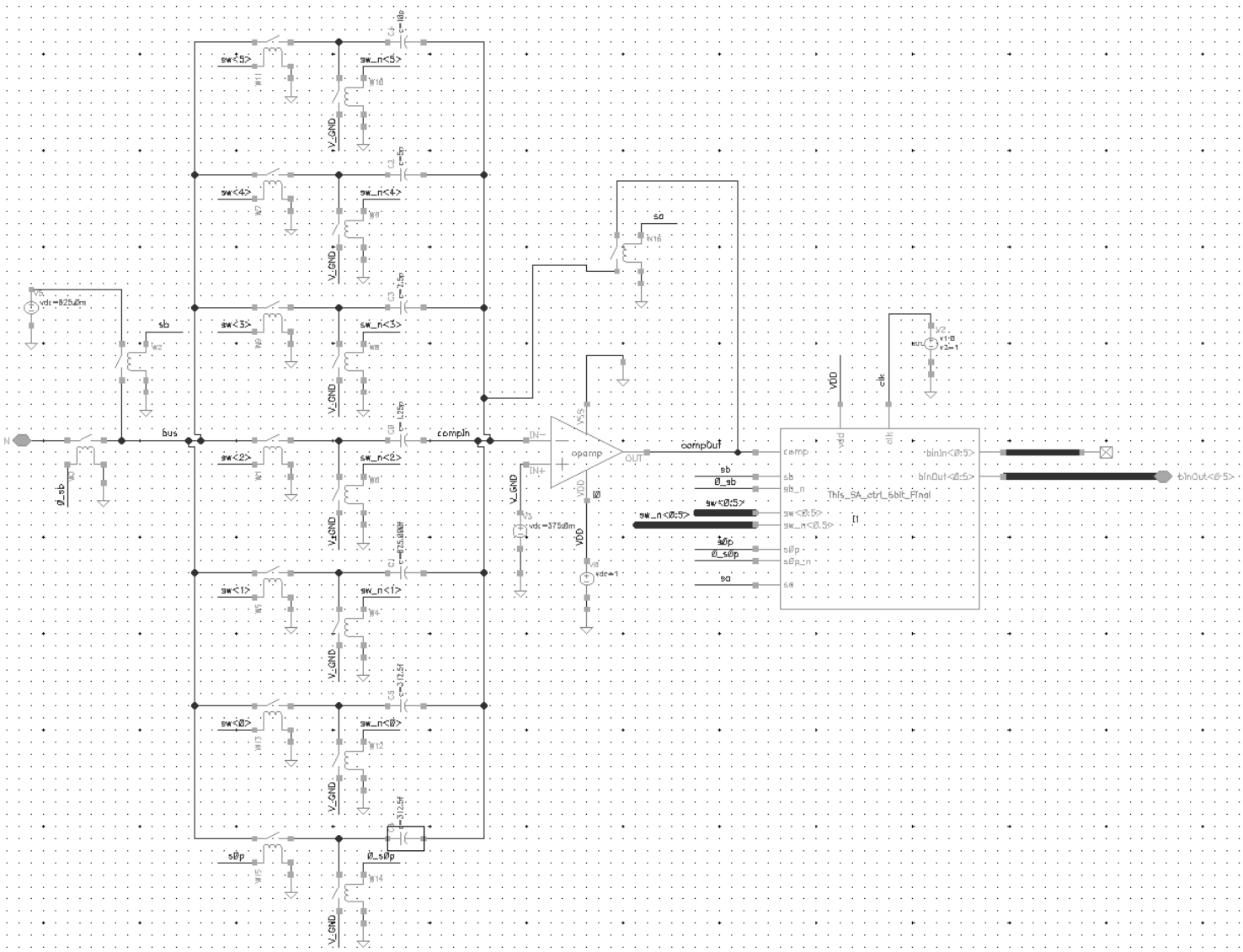


Figure 9: Schematic of the Analogue to Digital Converter

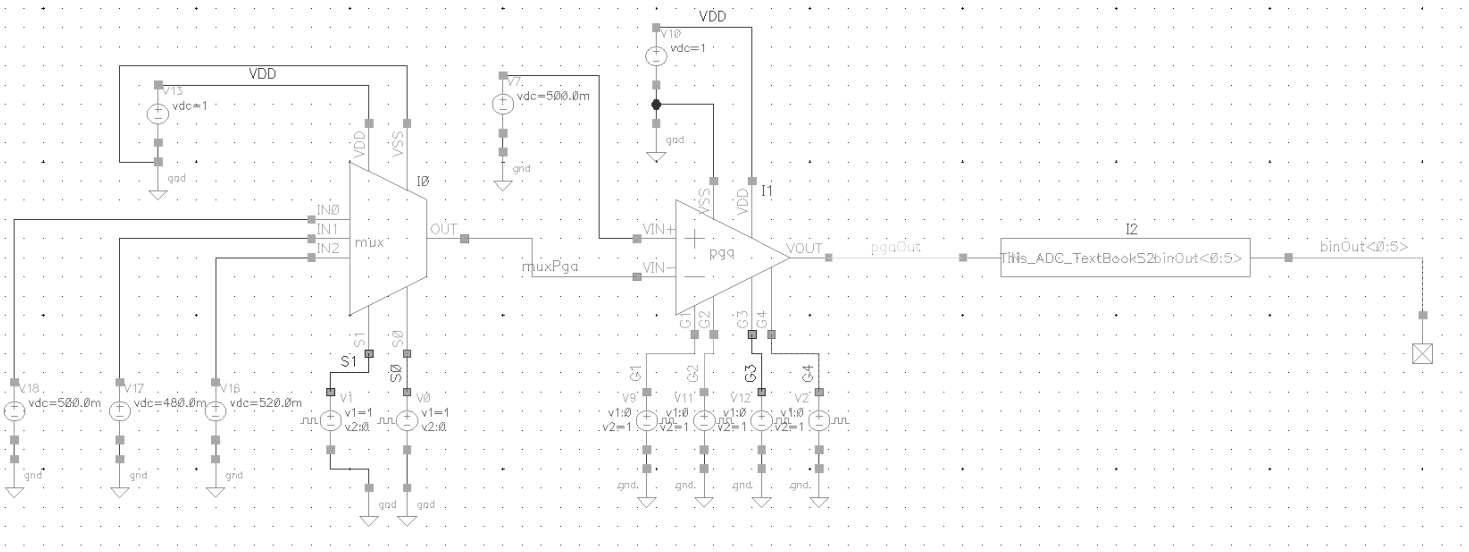


Figure 10: Schematic of the Top Level

5 Test-Benches

5.1 Multiplexer

The multiplexer was characterized with a range of frequencies from 1Hz to 10Hz, then again around the 200kHz rage. The test bench setup can be seen in Fig. 11 below. Looking at the resulting wave form in Fig. 12 one can see the output in blue follow each of the three given inputs only when its number is given as an input to the selections line. When a binary input of 11 is given then the simulation shows the output of the MUX at VSS. This was a triumphant success and perfect for the multiplexing operations within the Gizmonic smart watch.

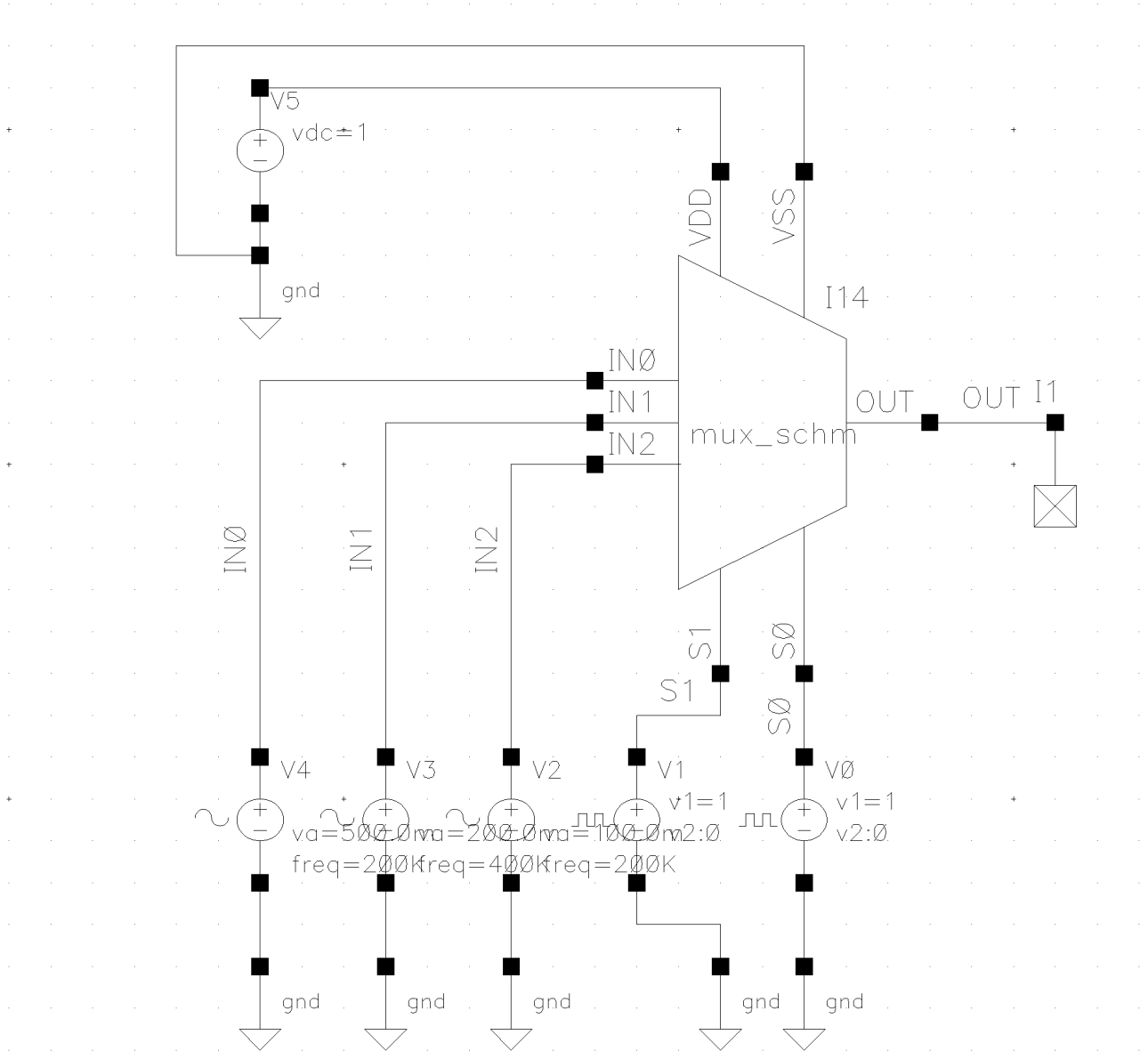


Figure 11: Test bench of the Multiplexer

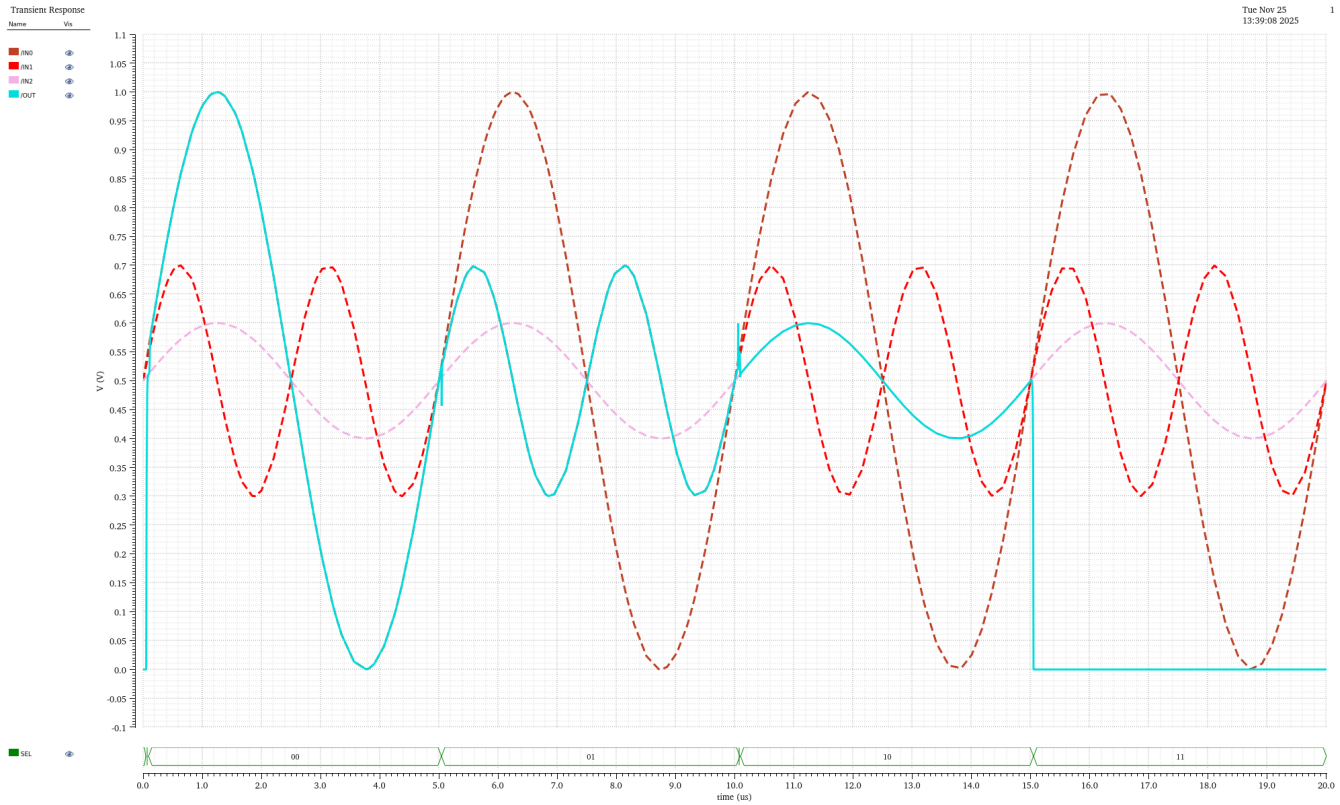


Figure 12: Simulation result of the Multiplexer

5.2 Clock-Splitter

The simulation of the splitter can be seen in Fig. 13 below. Much like the MUX a similar schematic was entered as a test harness for the splitter. Looking at Fig. 13 one can see there there in no overlap the the two clocks. The duration of time with neither clock high is also limited to 0.8ns.

5.3 Op-Amp

The simulation of the Op-Amp can be seen in Fig. 14 below. Much like the MUX a similar schematic was entered as a test harness for the Op-Amp. Here it was configured as a comparator to test this functionality for the ADC, but it also allowed the gain to the analysed and tuned for the PGA. As seen by the AC analysis of Fig. 14 the gain is 570x or 55.11 dB. Also seen in the DC analysis is that the full swing amplification is perfectly centred around 0.5V for the best performance when used with the ECG. This simulation guided many redesigns of the current mirror in the Op-Amp to ensure the best possible gain. An iterative approach was used with this simulation to tune the width of the FET providing bias current to the common source 2nd stage of the Op-Amp.

5.4 PGA

The simulation of the PGA can be seen in Fig. 15 below. Much like the MUX a similar schematic was entered as a test harness for the PGA. This allowed the PGA to be characterized and look at the error it produces. For a desired gain 1x the PGA has an actual gain of 0.9280x, giving an error

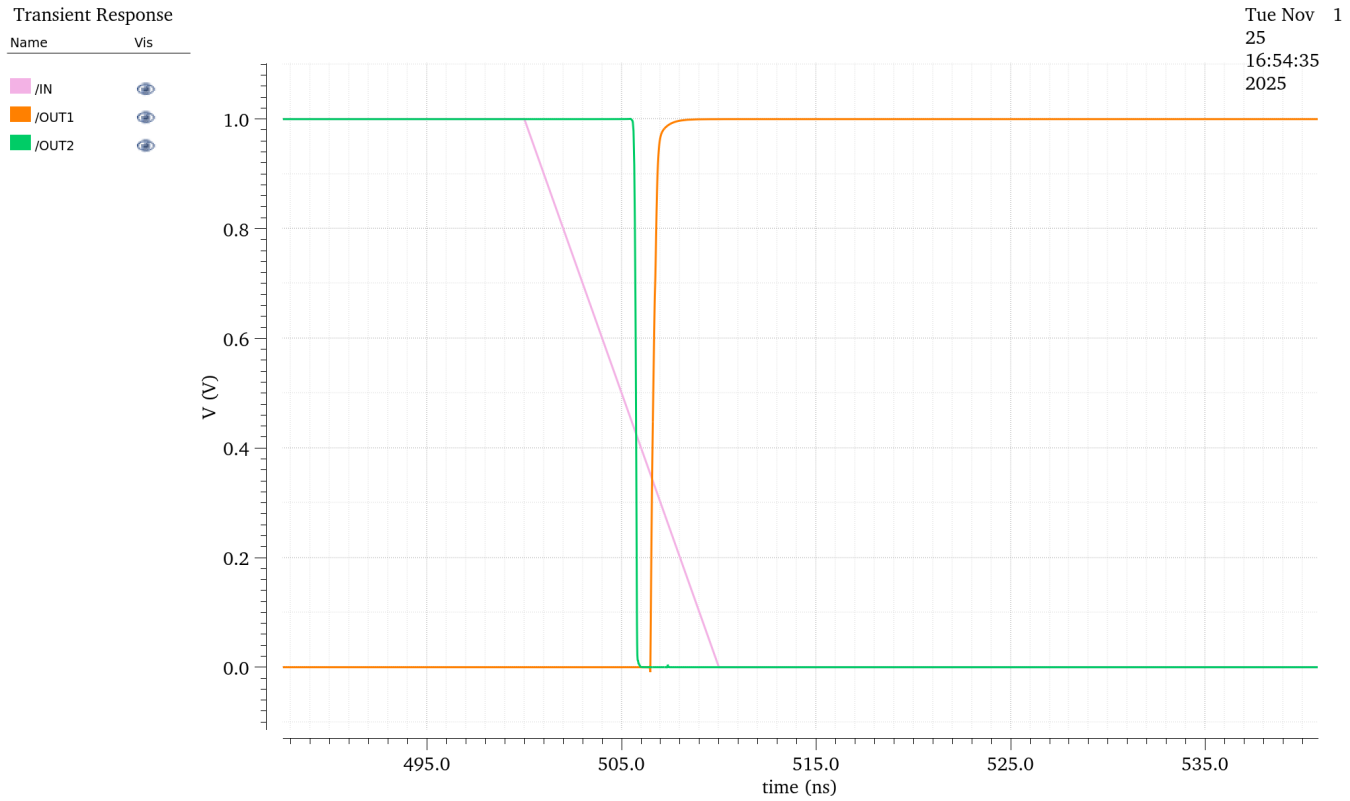


Figure 13: Simulation result of the Clock-Splitter

of 7.2%. For a gain of 2x the actual was 2.0007x with 0.03% error. The 3x gain setting is actually 3.0131x with 0.44% error. Finally for a desired gain 4x the actual gain was 4.0128x yielding an error of 0.32%.

5.5 ADC

The simulation of the ADC verified the sample rate of 100 kHz demonstrated the effectiveness verilogA code implementation, which can be found in Section 7 **Appendix - Verilog code**. Fig. 16 demonstrates the 10 cycle conversion moving through the sample, hold, and conversion states before reaching the done state and writing the switch positions to the binary output. The simulation in Fig. 16 was from an earlier design before the conversion envelope was tightened to increase resolution. In the depicted simulation the conversion range is from 0 - 1 V with a resolution of 7.8125 mV per bit. Over the larger envelope the accumulated error is greater. It can be seen that the sample of approximately 0.7 V is converting to the mid point of the discretized envelope, representing an error of 40%. This was unacceptable and led to the tightening of the envelope seen in the top level implementation.

5.6 Top Level

The fully assembled system containing all modules was simulated in Fig. 17 through Fig. 20 displaying gain levels 1 through 4 respectively. In these simulations, each displaying one gain level, the multiplexer is used to switch between DC sources at the signal common mode and maximum

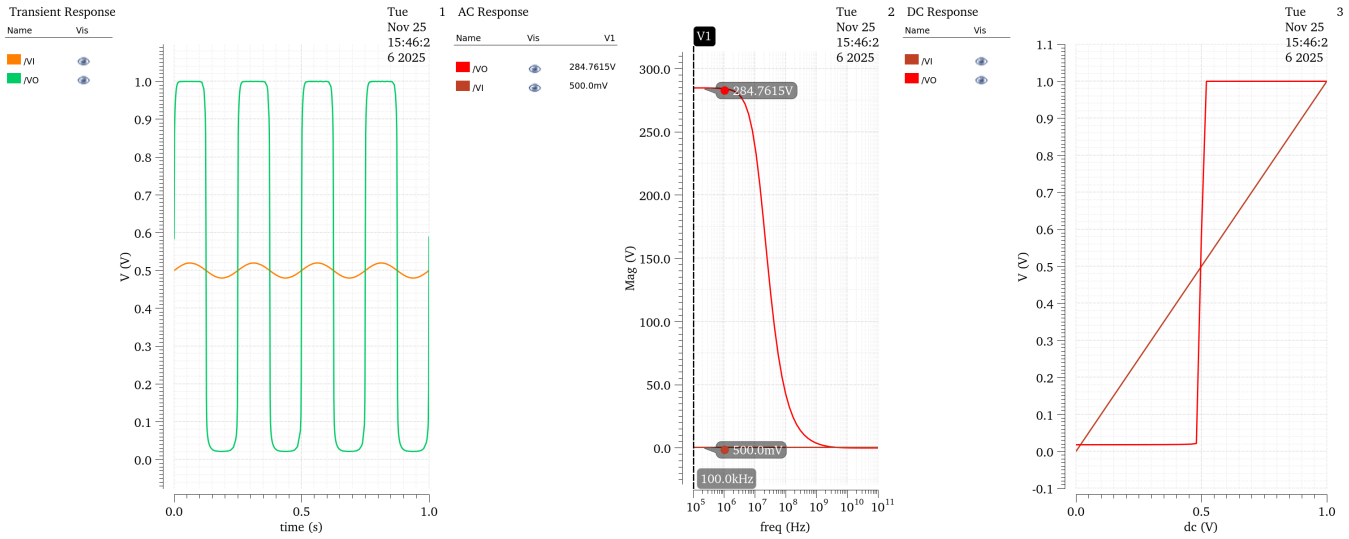


Figure 14: Simulation result of the Op-Amp

and minimum peak voltages. Each voltage level is given enough time for an ADC conversion to take place and the resulting 6-bit digital values are shown in the binOut strip of the graph.

Looking more closely at the binary outputs of Fig. 20 it can be seen that the ADC converts the full range from every bit set to 0 to every bit set to 1. Through tightening the conversion envelope as described in the Design Decisions section a resolution of 3.9 mV per bit is realized. The usage of the full discretized range is best observed in this figure since a gain of 4 achieves the peak values expected. In this simulation the signal maximum is converted to decimal value 48, and the minimum is converted to 22, utilizing 40% of the range which can be discretized by the converter. This example show that the system is robust enough to handle fluctuations and spikes. In contrast the binary conversions of Fig. 17 demonstrate that the system has enough resolution to provide meaningful numbers at the lowest gain level. With a gain of 1 the maximum and minimum conversions are 37 and 32 respectively providing 5 units of resolution thereby displaying robustness and tolerance at the bottom of the gain range as well.

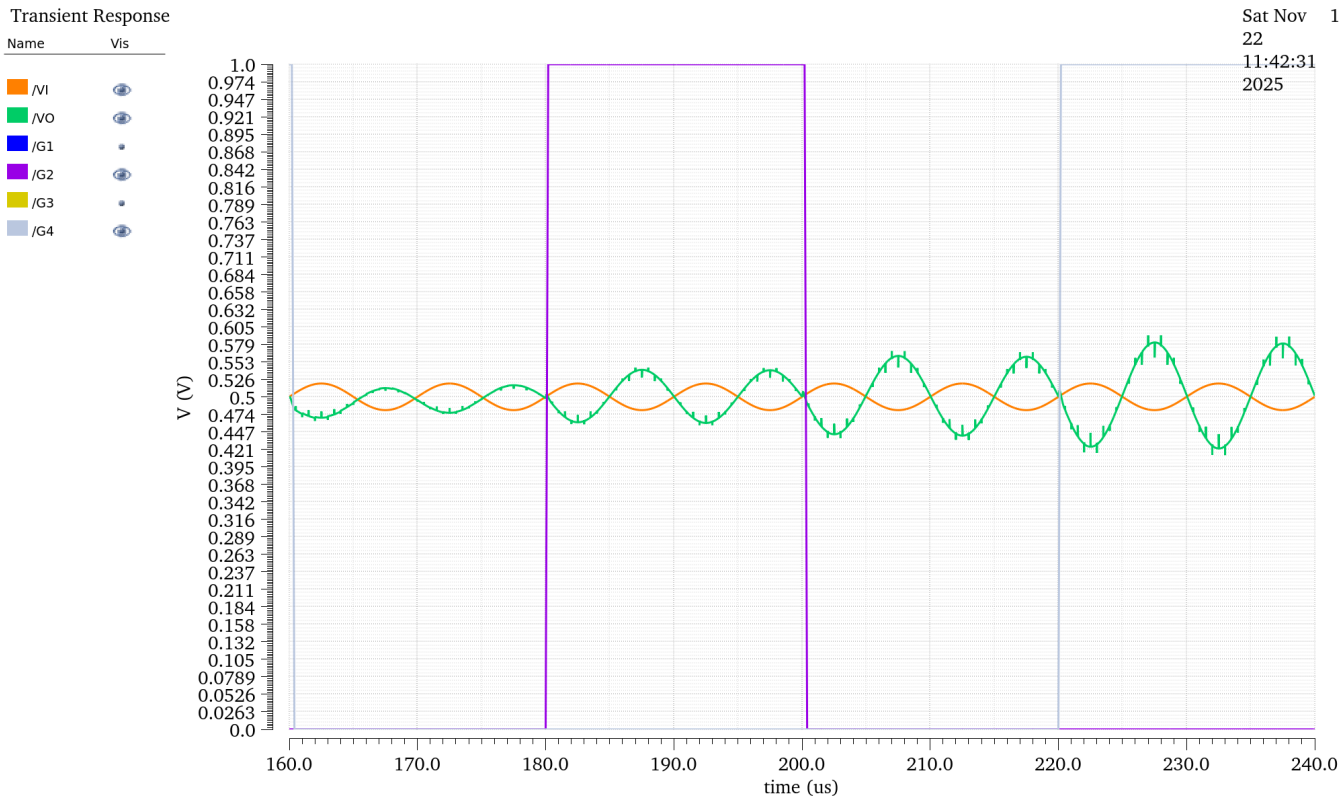


Figure 15: Simulation result of the PGA

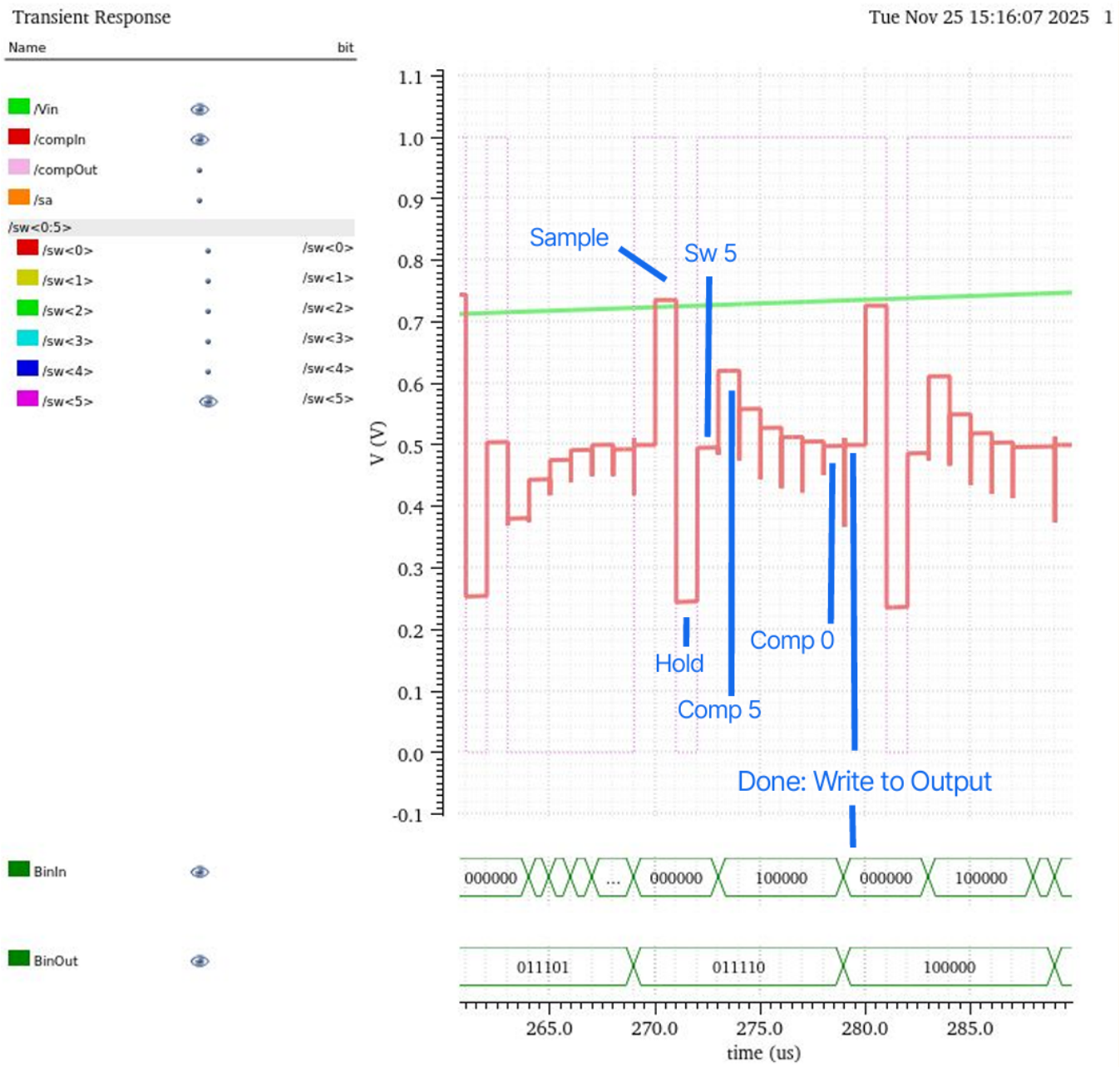


Figure 16: 10 cycle conversion

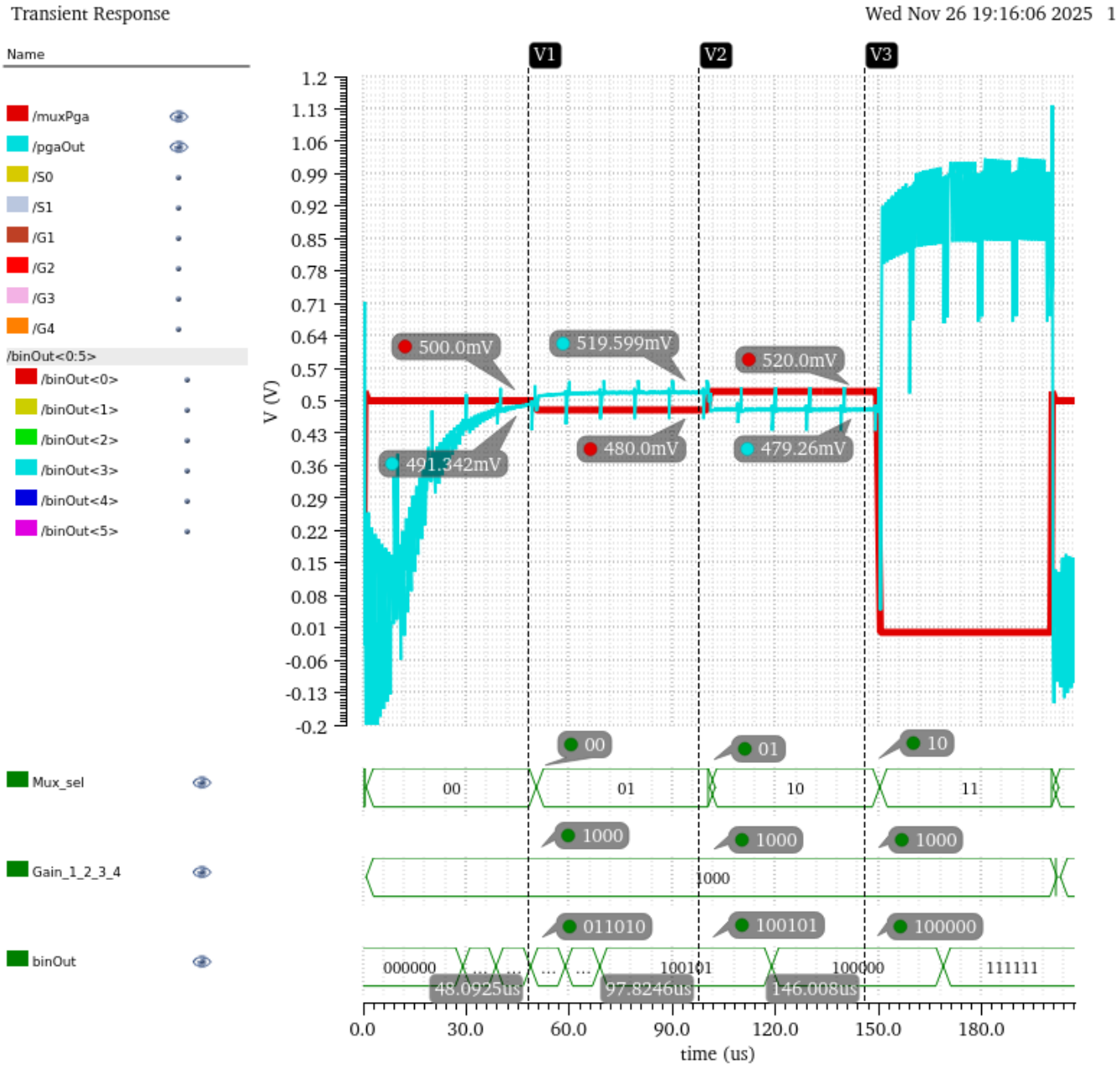


Figure 17: Top simulation of gain set to 1

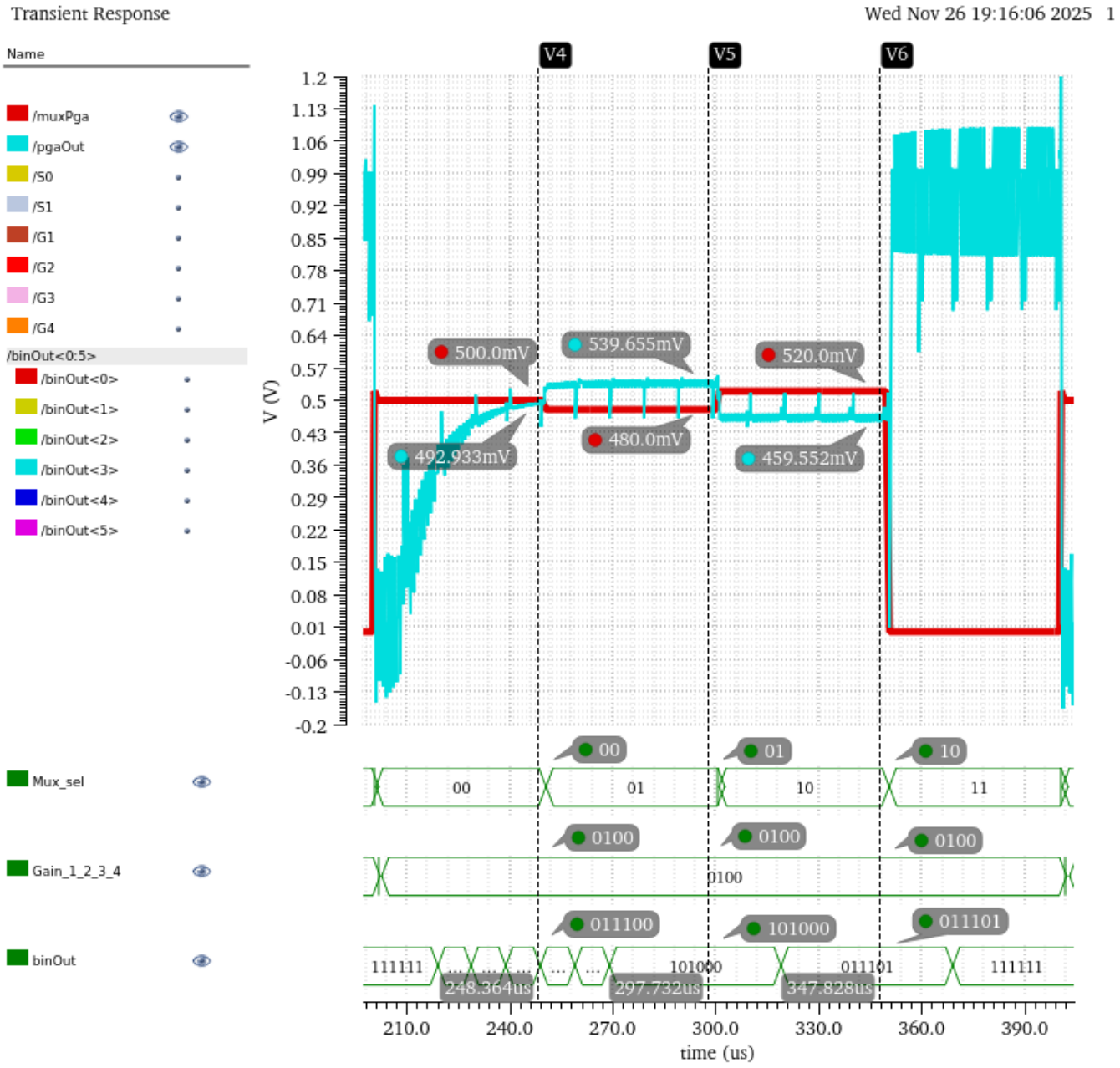


Figure 18: Top simulation of gain set to 2

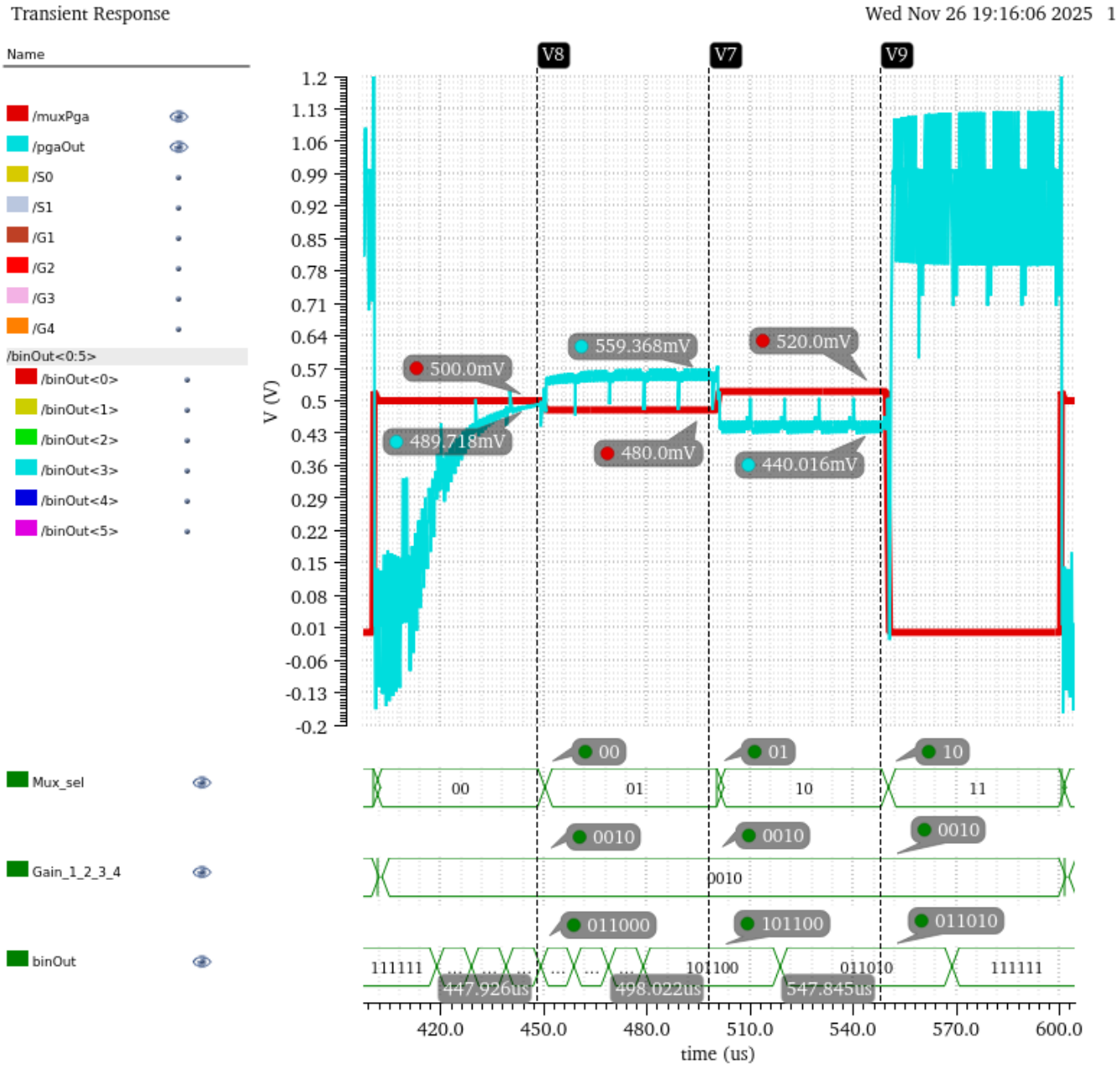


Figure 19: Top simulation of gain set to 3

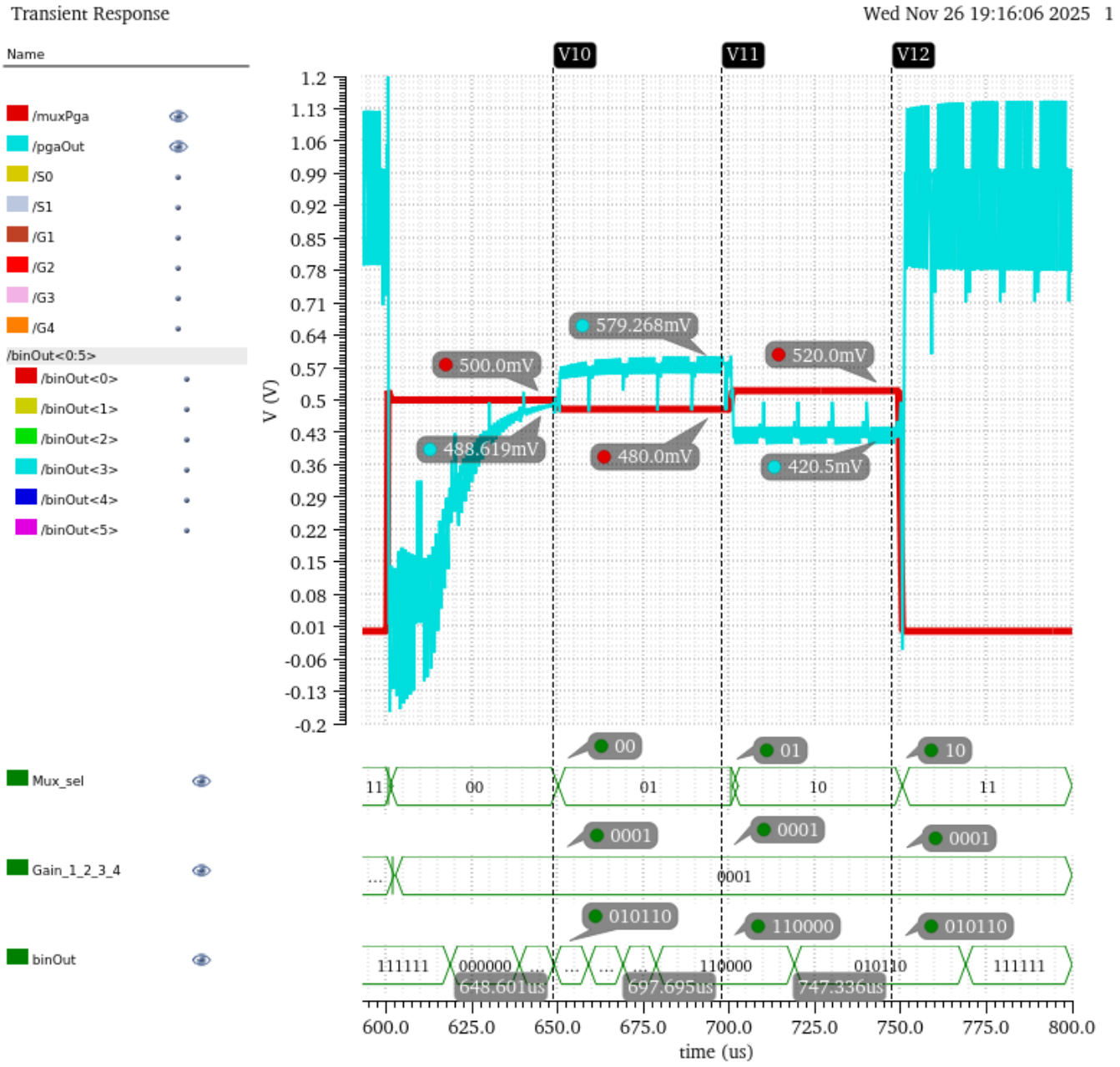


Figure 20: Top simulation of gain set to 4

6 Conclusions

The scope of this project was quite large and tied together many topics from the start to the end of lecture and used all of the skills learned within the three laboratory activities. Despite that there were some key take-aways learned along the way of the design of the Gizmonic smart watch analogue electronics. The reduction of area of the final chip is key, along with the management of parasitic capacitances and resistances. It will be imperative to remember to reduce parasitics and even more important to remember the techniques to do so. To manage the size of the project it was essential to use a hierarchical design approach within a shared library. This allows for higher levels of abstraction and faster completion of more complex elements while still maintaining control over the low level implementation. For example this project saw the creation of an Inverter symbol with a basic implementation at the transistor level using *nch* and *pch* instances from the TSMC 65nm node. This Inverter was used in multiple higher level schematics like the MUX and many times in the Clock Splitter. It was determined that the performance of both would benefit if the Inverter would be modified with larger widths and lengths. This made the MUX slightly slower but less susceptible to noise in the inverting process. More importantly, it made the Clock Splitter a lot slower which was required to guarantee no overlap in the clocks. Instead of needing to update over 20 transistors directly time was save by simply updating the two in the Inverter Schematic and thus updating all instances of said Inverter in a matter of seconds. These sort of benefits can be realized all throughout a hierarchical design.

A lesson learnt was to start all modules simultaneously. Developing all the top level modules in parallel would have been a more enlightening in process, as it is more agile than sequentially producing modules. Luckily a hierarchical approach made retroactive design changes and modifications more manageable, but it would have been better to reduce the amount of retroactive changes. On multiple occasions throughout the design process, as new discoveries were made and challenges were overcome, in the newest block the design team would need to make modifications to earlier blocks or other lower level symbols. If a more parallel design approach were taken this could have been avoided. An example is understanding the operating point of each stage from the beginning. This means more trial and error can happen in the hand calculation phase to minimize head scratching and confusion in design and simulation states.

As a whole the results of this project were very satisfactory. The MUX has a fast response with low noise transitions from one input to another. It operates at a very wide frequency range and is a robust hierarchical design. The Op-Amp design conforms to expectations and greatly exceeds the specified gain which was a triumphant success. This allowed the PGA to provide the desired results. The design team is very proud of the continuous and smooth operation of the PGA. There are no discontinuities and with the exceptions of the periodic spikes from the switched capacitors, the main operations of the PGA is fantastic. The design team would have greatly appropriated more time to refine the PGA further. With greater resources and a more accommodating time line the team aspires to reduce or eliminate the $50\mu s$ warm up time that the PGA needs on power up. Additionally, the team would also have liked to spend more time reducing the magnitude and frequency of the noise spikes from the switch capacitors. In conclusion, given the aggressive timeline to get the Gizmonic smart watch to market the design team is pleased with the final results as they achieved the desired performance in complete top level implementation.

7 Appendix - Verilog code

```
// 6-bit SAR ADC Control & Logic (Verilog-A behavioral model)

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

module SA_ctrl_6bit (clk, comp, vdd, sb, sb_n, sa, sw, sw_n, binIn,
    binOut, s0p,s0p_n);
  parameter integer sample_cycles = 1;   // number of rising edges spent
     sampling (edge to edge)
  parameter real td = 0;               // output delay
  parameter real tt = 0;               // output transition time

  input clk, comp, vdd;
  output sb, sb_n, sa;                 // sampling & bottom plate control
  output [0:5] sw;                     // capacitor switch controls (MSB=5)
  output [0:5] sw_n;                   // complement of switch controls
  output [0:5] binIn;          // Port for monitoring internal binary,
     same as switches
  output [0:5] binOut;                 // final conversion result bits
     updated every 10 cycles after complete conversion
  output s0p,s0p_n;                    // S0` control (always GND during
     conversion)

  // Analog disciplines
  electrical clk, comp, vdd;
  electrical sb, sb_n, sa;
  electrical [0:5] sw;
  electrical [0:5] sw_n;
  electrical [0:5] binIn;
  electrical [0:5] binOut;
  electrical s0p,s0p_n;

  // Internal state variables
  integer state;               // 0=SAMPLE, 1=HOLD_INIT, 2=CONVERT, 3=DONE
  integer bit_index;           // current bit being resolved (5..0)
  integer i;

  // Switch & binOut storage (0/1 logic values)
  integer sw_bits[0:5];
  integer sw_n_bits[0:5];
  integer binIn_bits[0:5];
  integer binOut_bits[0:5];

  real supply;

  analog begin
      supply = V(vdd);
```

```
43
44        // Initialize at the start of simulation
45        @(initial_step) begin
46            state = 0; // SAMPLE
47            bit_index = 5;
48            for (i=0; i<6; i=i+1) begin
49                sw_bits[i] = 1;
50                sw_n_bits[i] = 0;
51          binIn_bits[i] = 0;
52                binOut_bits[i] = 0;
53            end
54        end
55
56        // Rising clock edge event driver
57        @(cross(V(clk) - supply/2, +1)) begin
58            case (state)
59                // SAMPLE: keep array connected to Vin => sa & sb closed
60                0: begin
61                state = 1;// Move to HOLD_INIT
62                end
63
64                // HOLD_INIT: open bottom plate, prepare for conversion (no
                      bit set here)
65                1: begin
66                    bit_index = 5;
67                for (i=0; i<6; i=i+1) begin
68                        sw_bits[i] = 0;
69                        sw_n_bits[i] = 1;
70                    end
71                state = 2; // proceed to CONVERT
72                end
73
74                // CONVERT: single edge per bit pipeline. compare last and
                     set output before switching to hold for rest of cycle
75                2: begin
76                    // Can't compare right after switch because charge
                         needs cycle to distribute
77            if ((bit_index < 5) && (bit_index >= -1)) begin
78                      if (V(comp) <= supply/2) begin          // Comparison
                            from last cycle
79                          // Clear if comparator indicates lower
80                          sw_bits[bit_index + 1] = 0;
81                          sw_n_bits[bit_index + 1] = 1;
82                      end
83
84              // Latch result
85                      binIn_bits[bit_index + 1] = sw_bits[bit_index + 1];//
                            set internal binary bit for monitoring
86              //binOut_bits[bit_index + 1] = sw_bits[bit_index + 1];//
```

```
                future replace binout with holding variable
87              end
88
89              if (bit_index < 0) begin
90                              state = 3; // all bits done move to done state
91                      end else begin
92                // Tentatively set current bit HIGH first, cycle round
93                sw_bits[bit_index] = 1;
94                sw_n_bits[bit_index] = 0;
95
96                // Advance to next bit
97                bit_index = bit_index - 1;
98              end
99                  end
100
101               // DONE: final binOut available; restart sequence
                     automatically next edge
102               3: begin
103                   // Prepare for next conversion (one dedicated edge
                         keeps timing predictable)
104                   bit_index = 5;
105                   for (i=0; i<6; i=i+1) begin
106                       sw_bits[i] = 1;
107                       sw_n_bits[i] = 0;
108             binOut_bits[i] = binIn_bits[i]; // write to output at end of
                   conversion
109             binIn_bits[i] = 0; // Reset
110                     // binOut_bits[i]=0; // Reset for testing. Need to
                           update bin out to hold variable
111                   end
112                   state = 0; // back to SAMPLE; sa/sb go HIGH immediately
                         after this edge
113               end
114             endcase
115       end
116
117       // Drive outputs (ideal logic levels 0 or Vdd)
118       // Sampling switch: HIGH during SAMPLE state
119       V(sb_n) <+ transition((state==0 ? supply : 0.0), td, tt);
120       V(sb) <+ transition((state==0 ? 0.0 : supply), td, tt);
121
122     // Bottom plate switch: closed (HIGH) only during SAMPLE
123       V(sa) <+ transition((state==0 ? supply : 0.0), td, tt);
124
125       // S0‘: conncect vin during sample. connect GND during conversion
126       V(s0p) <+ transition((state==0 ? supply : 0.0), td, tt);
127       V(s0p_n) <+ transition((state==0 ? 0.0 : supply), td, tt);
128
129       // Capacitor switch controls (1 => connect to Vref)
```

```
130         V(sw[0]) <+ transition(sw_bits[0]*supply, td, tt);
131         V(sw_n[0]) <+ transition(sw_n_bits[0]*supply, td, tt);
132         V(sw[1]) <+ transition(sw_bits[1]*supply, td, tt);
133         V(sw_n[1]) <+ transition(sw_n_bits[1]*supply, td, tt);
134         V(sw[2]) <+ transition(sw_bits[2]*supply, td, tt);
135         V(sw_n[2]) <+ transition(sw_n_bits[2]*supply, td, tt);
136         V(sw[3]) <+ transition(sw_bits[3]*supply, td, tt);
137         V(sw_n[3]) <+ transition(sw_n_bits[3]*supply, td, tt);
138         V(sw[4]) <+ transition(sw_bits[4]*supply, td, tt);
139         V(sw_n[4]) <+ transition(sw_n_bits[4]*supply, td, tt);
140         V(sw[5]) <+ transition(sw_bits[5]*supply, td, tt);
141         V(sw_n[5]) <+ transition(sw_n_bits[5]*supply, td, tt);
142
143         // Internal Binary port driver
144         V(binIn[0]) <+ transition(binIn_bits[0]*supply, td, tt);
145         V(binIn[1]) <+ transition(binIn_bits[1]*supply, td, tt);
146         V(binIn[2]) <+ transition(binIn_bits[2]*supply, td, tt);
147         V(binIn[3]) <+ transition(binIn_bits[3]*supply, td, tt);
148         V(binIn[4]) <+ transition(binIn_bits[4]*supply, td, tt);
149         V(binIn[5]) <+ transition(binIn_bits[5]*supply, td, tt);
150
151       // Binary output driver
152         V(binOut[0]) <+ transition(binOut_bits[0]*supply, td, tt);
153         V(binOut[1]) <+ transition(binOut_bits[1]*supply, td, tt);
154         V(binOut[2]) <+ transition(binOut_bits[2]*supply, td, tt);
155         V(binOut[3]) <+ transition(binOut_bits[3]*supply, td, tt);
156         V(binOut[4]) <+ transition(binOut_bits[4]*supply, td, tt);
157         V(binOut[5]) <+ transition(binOut_bits[5]*supply, td, tt);
158
159     end
160 endmodule
```