Successive Approximation Register (SAR) ADC

Anthony Garraffo(CID03) and Hector Morrell(CID08)
MSEE San Jose State University
EE 288: Data Conversions/Analog Mixed Signal ICs

Abstract

Successive Approximation Register(SAR) Analog-to-Digital Converter (ADC) is an electrical circuit that converts an input analog signal to a discrete binary output through binary search. The SAR ADC is becoming one of the most common and most sought after ADC in industry due to their robustness, speed, and low Figure of Merit. In this report we have designed a SAR ADC for San Jose State University EE 288 Mixed Signal Analog Design as a final class project. This report will explain the SAR ADC system, Design procedures for the comparator we designed, and our results for the whole system.

Key words: SAR ADC, Figure of Merit, Low Power

Introduction

Successive Approximation Register ADC has 5 major circuit blocks for the entire system. Sample and Hold, Comparator, Clock/Timing, SAR Logic, and Digital-to-Analog Converter. Figure 1. Below shows a typical SAR block diagram for the whole system.

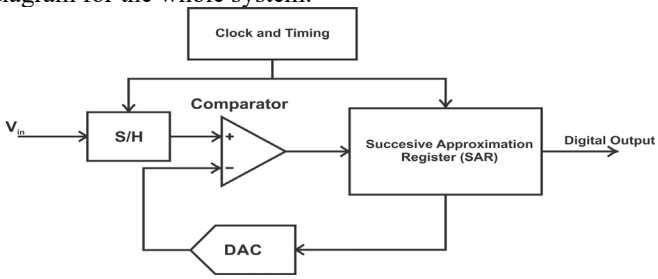

Figure 1. SAR ADC Block Diagram

Below are the metrics for us required to use or hit.

Target Specs

- Process Technology 45nm CMOS with 1V device only
- Supply Voltage VDD = 1V
- Resolution 8 bits
- Sampling Rate, fs 10 MS/s
- Input Range, VFS 1.2 Vppd
- Input Frequency (7/64)*fs MHz
- Input Clock, MCLK100 MHz with 0.1ns rise and fall time
- Walden FoM10 fJ/conversion-step

The first circuit is the Sample and Hold(S/H). Sample and Hold circuit takes a sample of the input signal during every clock cycle and holds the voltage value. Figure 2. and Figure 3. below show what the input sinusoidal signal will be then the subsequent output of the sample and hold circuit.

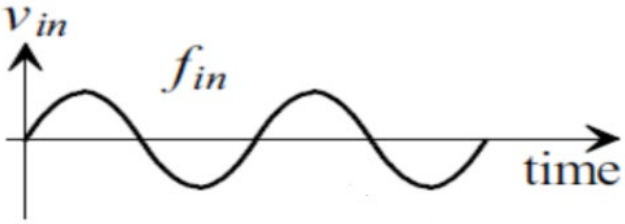

Figure 2. Input signal before S/H

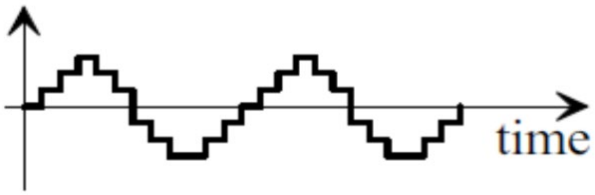

Figure 3. Output signal of S/H

Once the signal has passed through the S/H circuit it will then be imputed to the comparator. A comparator is an electrical circuit that takes in an input and then compares it to a bias voltage. There are 2 ways to use a comparator. You can use the non-inverting input (plus sign on comparator) and compare to see if your input is higher than your bias. The other way to connect the input signal to the inverting input(minus sign on comparator) to determine if the input is lower than the bias voltage. Depending on the architecture you are using, if whichever statement you are using is true then you output a 1 on the comparator, otherwise the output is a 0. Figure 4. below is the transfer characteristic of a comparator and what a comparator looks like.

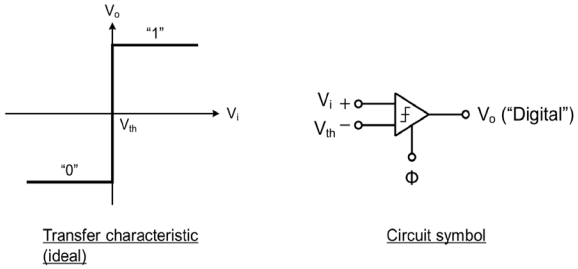

Figure 4. Comparator

The biasing input we are using for the comparator will be the output of the DAC. The DAC output is the signal that has gone through the entire cycle of the Comparator, SAR Logic, and DAC. We will be touching more on the SAR

Logic and DAC later in this paper. The benefits of this is that by using Binary Search Algorithm with this looping search we can design an ADC with a higher resolution and better Bit Error Rate(BER).

The next system in the SAR ADC is the SAR Logic. The purpose of the SAR Logic is to give an approximate digital code of Vin that is measured from the comparator and then outputs the code into the DAC.

The SAR Logics consists of 2N flops, these flops can be designed using an array of digital circuits. The comparator output is the bus for the flops used in the SAR logic. This logic is to reset the flops at end of conversion cycles, this is to create an output that is within the cycle. End of conversion cycles are what is then outputted to the DAC.

The Digital-to-Analog Converter is a circuit that input binary bits and then outputs an analog signal. The DAC that was used in this project was a Capacitive DAC. Figure 5. below shows the DAC used in the circuit.

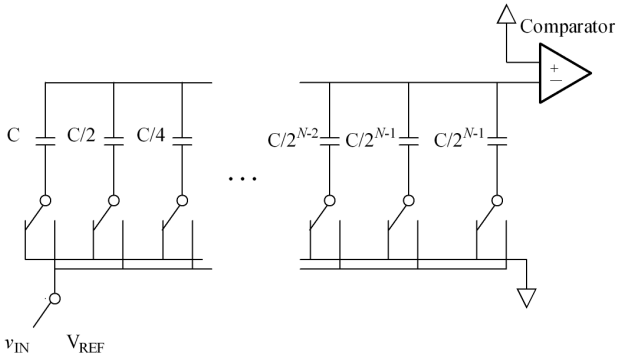

Figure 5. DAC Connected to the Comparator

The number of capacitors are dependent on bits, for the SAR ADC you will have the same number of capacitors as bits. The capacitor values are based on binary weighted, so the actual capacitance value is not as important as the ratio, since we are using the capacitors for gain. For instance for our SAR is a 8-bit ADC meaning our design has 8 capacitors with the value equalling 2^N, so the first DAC will equal 2^0 meaning a value of 1C, then the last Capacitor will be 2^{10}equalling a cap of 256C.

The Last block is the timing block. This block is very important as one of the foundations of the SAR ADC. The SAR ADC is sensitive to timing so designing a very robust and accurate timing block is very important. For SAR ADC circuit we need an asynchronous clock, which is a digital logic circuit that is a “self-timed” clock generator that is not governed by a local clock. It is dependent on the completion of a function or logic to move on to the next sequence. This is very important because during the bit cycling in the DAC we need to know when the Most Significant Bit is high to cycle to the next bit, or vice versa if MSB is low to switch back the original state and test the net binary set. This switching is referred to a Monotonic scheme. This helps conserved the switching energy between each stage.

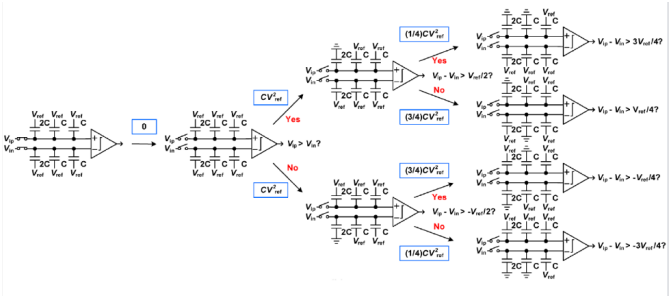

Figure 6: Monotonic Switching scheme [4]

These are the fundamental building blocks of the SAR ADC. Next we will be explaining the design of our circuit. The SAR Designed by our team is a fully differential ADC meaning the parameter shown in this introductory section will be essentially doubled to take into account the negative input we will be taking in.

Design

For the design we will begin with a top down view. Figure 7. below shows the test bench that we used to test the SAR ADC. The test bench below was provided by Dr. Lee.

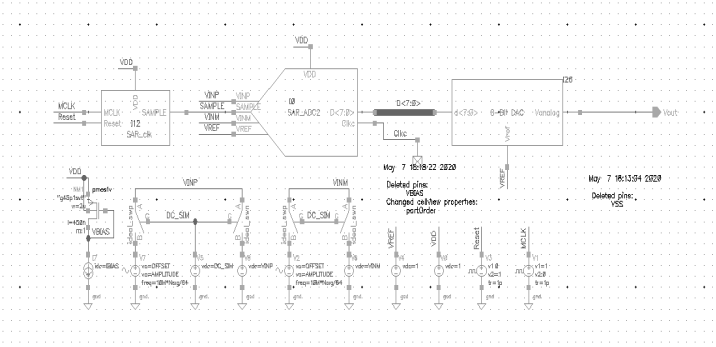

Figure 7. SAR ADC test bench

The SAR ADC architecture is shown below, figure 8. The design below has the Sample and Hold, DAC, Comparator, SAR Logic, and Asynchronous clock.

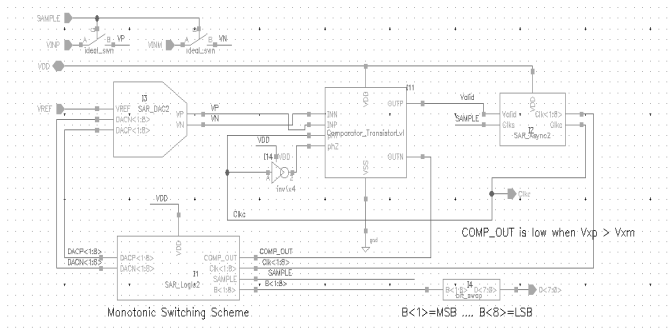

Figure 8. SAR ADC

The SAR logic is below. Using an array of Data Flip Flops with NAND gates and inverters we can get binary outputs to be sent to the bit swap. We also get the outputs for the DAC to do the bit cycling to continuously check the sampled voltage from the S/H.

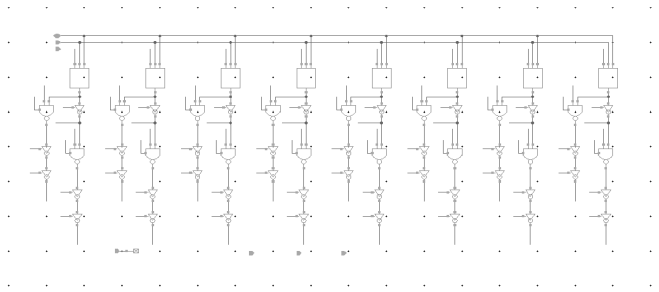

Figure 9. SAR Logic

The DAC used in this project is a capacitive DAC. The capacitive DAC uses an N-bit number of caps in parallel that are binary weighted. Binary weighted meaning that the capacitive value of each one is the same but each will have a different ratios size. The size is $2^N * C$.

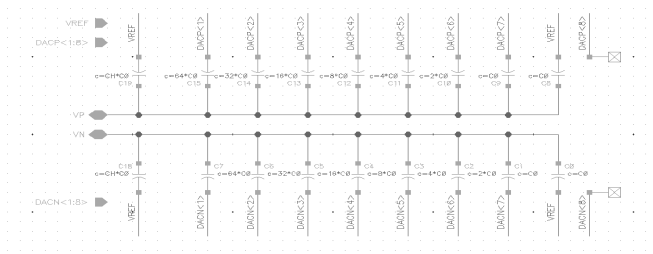

Figure 10. SAR Capacitor DAC

The Asynchronous clock shown below uses an array of Digital Flip Flop Registers to get a clock that is independent of the local clock that is based on the switching cycle of the SAR Logic and the bit cycling of the DAC. This was provided by Dr. Lee to implement in our circuit.

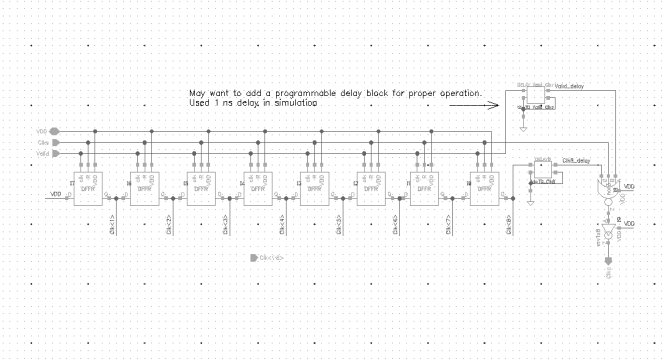

Figure 11. Asynchronous Clock

A dynamic comparator was used for this design in order to help lower the overall power consumption. This typology, found in [4], consists of a PMOS differential pair connected to an NMOS latch structure. The tail currents that biased the system used a bias voltage signal to apply consistent power to the pre amplifier and a clock signal that triggered the latch regeneration period. This system provided a working system that accurately evaluated the magnitude of the input signals when VinP was greater than VinM. There were glitches when these signals crossed that could not be remedied in the time allotted. The outputs of the comparator were fed into an ideal NAND gate that would trigger and turn on the Asynchronous clock.

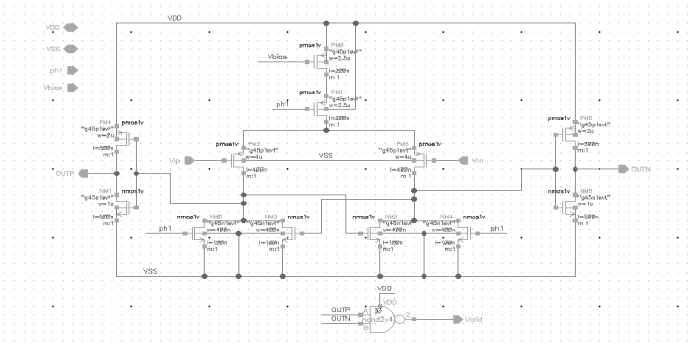

Figure 12. SAR Comparator

When the clock signal is high both of the output signals are pulled high. Then when the clock signal is low the preamplifier compares the two input voltages and the latch regeneration forces one signal high and the other low based on the results. Because of the binary value of the outputs the *Valid* signal is set to high and turns the asynchronous clock system on.

Results

Overall the SAR ADC was unsuccessful in converting the differential input signal to a digital output code. Multiple variations of the typologies outlined in this paper were used and tweaked in order to create a correct output voltage

The Walden Figure of Merit was evaluated by finding the power consumed by each of the individual blocks. The following equation was used to evaluate this rating:

$$F_{A1} = \frac{Power}{(2^{ENOB} * f_s)}$$

The table below shows the power rating for each of the blocks in the system.

Block	Power Consumption
Overall SAR ADC	20 uW
Comparator	240 pW
SAR DAC	2.13 pW
Asynchronous Clock	577 nW
SAR Logic	0W
Bit Stream	0 W

Table. 1 Power consumption by Block

Due to the fact that most of the components used in the system were ideal blocks the overall power consumption is small. The SAR Logic consisted of ideal components so there is no power consumed in this evaluation. The dynamic characteristics could not be ascertained because the output voltage flatlined and did not respond to input stimulus. With the output voltage that was evaluated the ENOB was approximately -0.9 bits. This value is not a real result but it was used to reach an FoM value. The final Figure of Merit came out to be 373fJ/Conversion Step. This value is not accurate though because the ENOB could not be evaluated

While the system did not create the output expected by an SAR ADC there were indicators that the system was operational. The comparator was evaluated in a seperate test bench to check its functionality. Figure 13 shows the results of this test and its ability to compare the input signals and accurately compare them.

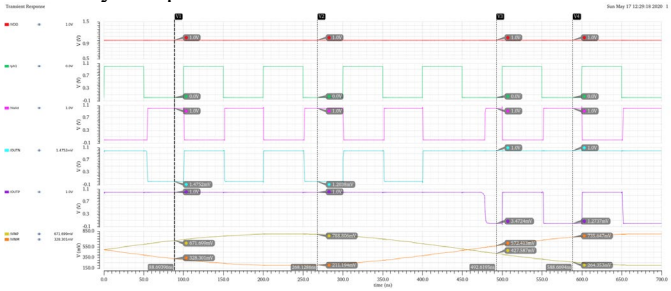

Figure 13: Comparator evaluation

With the verification that this system was working properly, the attention was turned to the SAR Logic and Asynchronous clock. As seen in the results on slides 9 and 10 of the accompanying slideshow and figure 14 below, the clocking scheme labeled Sample and CLKC triggered correctly to the corresponding stimulus from the comparator. This validated the Asynchronous clock system. Comparing results to the experimental results provided, the digital output values were arguably correct. The figure below the bit values for the positive and negative signals. This shows that the SAR Logic was able to accurately trigger the correct bit states to digitize the analog comparator signal. The SAR Logic was also implemented in Verilog A code and is shown in the Appendix section. These bits may not have been accurately translated back to an analog signal via the SAR DAC.

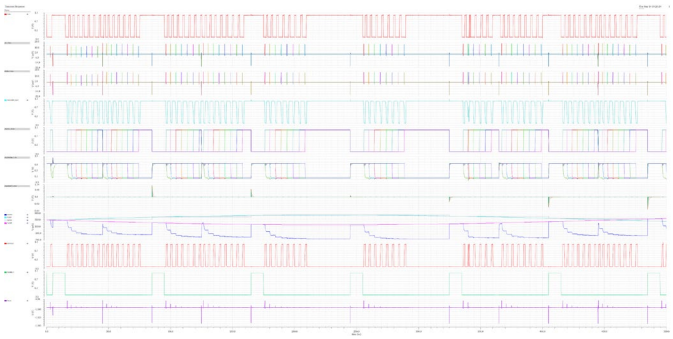

Figure 14: SAR Logic outputs

The SAR DAC consisted of a differential DAC array that used a string of binary weighted capacitors. This seemed to be the last area of possible issues and various trial and error attempts were completed to see how the system would react. When the unit capacitor value was varied the signals had an additional ripple. When the topside capacitance, CH, was lowered drastically there was larger spiking on each step of the bit cycling signal.

One of the main indicators that the system is not properly working is that the output voltage flat lines at -1V regardless of the stimulus applied. The next indicator is found that when VinM crosses over VinP, the comparator begins to fail. In the first few clock cycles the latch system forces Comp_out to low and the negative output high but by the third trigger, the comparison is no longer valid. The clocking system continues to work because the Valid signal is still set to high but the signal passed to the SAR Logic is incorrect.

Summary

This project was a success in some aspects and fell short in others. There were several goals for this project and our team was able to meet some of them. First our goal was to learn as much and practice our design skills with a SAR ADC. This was very important as SAR ADCs are becoming more prominent in design and coming up as one of the better overall systems to use. The second goal was to design a fully functioning SAR ADC and achieve the metrics that were laid out in the project requirements. This was a success in some aspects and fell short in others because we were able to obtain a function clocking scheme and comparator however our system was no bit cycling as needed resulting in an output that flatlined at -1V. Unfortunately due to time we were unable to decipher this issue but our system seems to be at 80% functionality and with a little more time and work we would have a fully functioning SAR ADC.

Roadblocks that occurred during this project was foremost the Novel Covid-19 Pandemic that is sweeping the world. Unfortunately with the pandemic we were not allowed to be at school to work in the lab or in person with our partner. Our group also experienced some difficulty when it came to remote log in with slow server times that made simulations at times difficult. Remote working was a difficult task but what was achieved at the end of the day.

Appendix
Verilog-a Code

```
// VerilogA for 288_SAR_ADC, SAR_CODE, veriloga

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

module SAR_CODE(VDD, compout, sample, Clk, DACN, DACP, Bits);

        parameter bits = 8;

        ////////////////////////////////////////////////////////////////// Input of System
//////////////////////////////////////////////////////////////////

        input VDD;  // Input voltage of the system
        input compout; // Output of comparator
        input sample;  // Sample Frequency
        input Clk[7:0];  // Clk cycle from the Asynchronous clock


        ////////////////////////////////////////////////////////////////// Output of System
//////////////////////////////////////////////////////////////////
        reg output Bits[7:0];  // Bits for Bit swap
        reg output DACP[7:0];  // DACP output
        reg output DACN[7:0];  // DACN output


        ////////////////////////////////////////// Digital Flip-Flop Logic // This section take in VDD, Compout, and Clk[8:0]
//////////////////////////////////////
        module DFF(Q[7:0], Qbar[7:0], compout, Clk[7:0], VDD);

                output  reg  Q[7:0];

                output   Qbar;

                input   compout, Clk[7:0], VDD

                assign Qbar = ~Q;

                always @(posedge Clk)

                begin

                 if (VDD == 1'b1) //If not at reset

                                Q = 1'b0;

                else

                 Q[7:0] = compout

        endmodule


        ////////////////////////////////////////////////////////// Binary Bits after the inverter in the SAR
Logic//////////////////////////////////////////////////
        module inv1(input Q[7:0], output Bits[7:0]);

         //-- Both the input and the output are "wires"
         wire Q[7:0];

                                                                                                1


          wire Bits[7:0];

          //-- Assign the inverse of the input, to the output
          assign Bits[7:0] = ~Q[7:0];

        endmodule


        ////////////////////////////////////////////////////////////////// DACN NAND
    Operation//////////////////////////////////////////////////////////////////////////////
        module NAND_N(Q[7:0], Clk[7:0], DACN_NAND) ;

                input Q[7:0], Clk[7:0];

                output DACN_NAND;

                wire DACN_NAND;

                assign DACN_NAND =v (Q[7:0]&Clk[7:0]);

        endmodule


        ////////////////////////////////////////////////////////////////// DACP NAND
    Operation//////////////////////////////////////////////////////////////////////////////
        module NAND_P(B[7:0], Clk[7:0], DACP_NAND) ;

                input Bits[7:0], Clk[7:0];

                output DACP_NAND;

                wire DACP_NAND;

                assign DACP_NAND =v (B[7:0]&Clk[7:0]);

        endmodule


        ////////////////////////////////////////////////////////// Double inversion after
    DACN_NAND//////////////////////////////////////////////////////////////////////////////
        module inv2(input DACN_NAND[7:0], output DACN_inv1[7:0]);

         //-- Both the input and the output are "wires"
         wire DACN_NAND[7:0];
         wire DACN_inv1[7:0];

         //-- Assign the inverse of the input, to the output
         assign DACN_inv1[7:0] = DACN_NAND[7:0];

        endmodule


        // Double inversion after DACN_NAND
        module inv3(input DACN_inv1[7:0], output DACN[7:0]);

         //-- Both the input and the output are "wires"
         wire DACN_inv1[7:0];
         wire DACN[7:0];

                                                                                                2


         //-- Assign the inverse of the input, to the output
         assign DACN[7:0] = ~DACN_inv1[7:0];

        endmodule


        ////////////////////////////////////////////////////////// Double inversion after
    DACP_NAND//////////////////////////////////////////////////////////////////////////////
        module inv2(input DACP_NAND[7:0], output DACP_inv2[7:0]);

         //-- Both the input and the output are "wires"
         wire DACP_NAND[7:0];
         wire DACP_inv1[7:0];

         //-- Assign the inverse of the input, to the output
         assign DACP_inv1[7:0] = DACP_NAND[7:0];

        endmodule


        // Double inversion after DACN_NAND
        module inv3(input DACP_inv2[7:0], output DACP[7:0]);

         //-- Both the input and the output are "wires"
         wire DACP_inv1[7:0];
         wire DACP[7:0];

         //-- Assign the inverse of the input, to the output
         assign DACP[7:0] = ~DACP_inv1[7:0];

        endmodule


endmodule
```

References

[1] M. J. M. Pelgrom, *Analog-to-digital conversion*. Cham: Springer, 2017.

[2] Olga Kardonik, MS Thesis, University of Texas, Austin, A study of SAR ADC and implementation of 10-bit asynchronous design, 2013

[3] Albert Hsu Ting Chang, PhD Thesis, MIT, Low-power high-performance SAR ADC with redundancy and digital background calibration, 2013

[4] C. Liu, S. Chang, G. Huang, Y. Lin, “A 10-bit 50-MS/s SAR ADC with a monotonic capacitor switching procedure,” IEEE J. Solid-State Circuits, pp. 731–740, April 2010

[5]Pieter Harpe, Cui Zhou, et. al. “A 26 W 8 bit 10 MS/s asynchronous SAR ADC for low energy radios,” IEEE J. Solid-State Circuits, pp. 1585–1595, July 2011

[6] T. Cao, S. Aunet, T. Ytterdal, “A 9-bit 50MS/s asynchronous SAR ADC in 28nm CMOS,” NORCHIP 2012

[7] Brian P. Ginsburg and Anantha P. Chandrakasan, “An energy-efficient charge recycling approach for a SAR converter with capacitive DAC,” IEEE ISCAS 2005

[8] Victor Gylling, MS Thesis, Lund University, Sweden, Implementation of a 200 MSps 12-bit SAR ADC, 2015