Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática
Especialidad en Diseño de Sistemas en Chip

Registro de Aproximaciones Sucesivas y Convertidor Digital a Analógico para SAR ADC de 10 bits de Baja Potencia para Aplicaciones Biomédicas

---

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
ESPECIALISTA EN DISEÑO DE SISTEMAS EN CHIP

Presenta: **RAYMUNDO GARZA PÉREZ**

Director: Dr. CUAUHTÉMOC RAFAEL AGUILERA GALICIA

Tlaquepaque, Jalisco. 17 de julio de 2021.

ITESO – The Jesuit University of Guadalajara

Electronics, Systems, and Information Department

Specialization Program in Systems on Chip Design

Successive Approximation Register and Digital to Analog Converter for a 10-bit Low Power SAR ADC for Biomedical Applications

---

Thesis/Project to achieve the university **degree** of
SOC DESIGN SPECIALIST

Presents: **RAYMUNDO GARZA PEREZ**

Thesis Director: Dr. CUAUHTÉMOC RAFAEL AGUILERA GALICIA

Tlaquepaque, Jalisco. July 17, 2021

Acknowledgments

I would like to thanks to CONACYT for the financial aid and for their continuous support to develop better professionals. Without it I would not be able to take this opportunity.

I would like to thank ITESO for all the resources that were given for my development as an Engineer and as a person. This includes the professors that help me with this project, Cuauhtémoc Aguilera, Esteban Martínez, and José Luis Chávez.

Finally, I would like to thank my family and friend for all the support and your encouraging words in this journey.

Abstract

This document presents the design of a successive approximation register (SAR) and a digital to analog converter (DAC). The two circuits are designed, synthesized, and implemented in layout using TSMC 180nm technology. The DAC has a Charge-Scaling Capacitors architecture and uses a switching scheme different to the conventional one, which improves the power consumption of the circuit. These modules are used in a low power 10-bit ADC and are designed for Biomedical applications.

The SAR design is synthesized using a clock with a frequency of 250 MHz, the total power consumption of the SAR is 22045.030 uW. A testbench is designed to verify the functionality of the SAR by testing several cases of interest.

The base capacitance of the DAC is 1 pF. The functionality of the DAC is tested using a mixed signal simulator and a testbench with the required circuit. This also includes the Verilog model of the SAR block. The testbench is able to apply single conversions test case and a ramp-test case, in which all the possible inputs to the DAC are tested sequentially.

Table of Contents

Acknowledgments ..... **v**

Abstract ..... **vii**

Table of Contents ..... **ix**

1. ADC for Biosensor Applications **3**

1.1. Project Introduction 3
1.2. Analog to Digital Conversion 3
1.3. SAR ADC Architecture 5
1.4. Capacitive Split-Array DAC Architecture 7
1.5. SAR ADC Block Diagram 9
1.6. Functional Description of Low Power SAR ADC Block Diagram 10

2. Successive Approximation Register **11**

2.1. Flow Diagram of SAR Algorithm 12
2.2. SAR Implementation 14
2.3. Simulation of SAR module 16
2.4. Logic Synthesis 20

3. Digital to Analog Converter **23**

3.1. DAC Architecture 23
3.2. Low Power Switching Scheme 24
3.3. Transmission Gate Switch Design 26
3.4. Layout of Multiplexer and DAC Module 28

4. SAR and DAC Integration **31**

5. ADC Integration **36**

Logic synthesis 36

6. Conclusions and Future Work **45**

7. Bibliography **46**

Appendix A: Simulating using AMS Simulator **48**

Appendix B: SDC Constraint File **61**

1. ADC for Biosensor Applications

1.1. Project Introduction

This project, together with the designs presented in (Castañeda Villalpando, 2021) and (Gonzalez Ornelas, 2021), are developed with the intention of designing a functional 10 bit Low Power SAR ADC for biomedical purposes using TSMC 180nm technology. Each of these works contains a report about the design of different modules that together conform the ADC. The last chapter of this document focus on the process of integrating all the ADC modules, and the functional results that were obtained.

Biomedical applications have special requirements to work as smoothly as possible. One of them is to have a design as small as possible so that the chip is the least intrusive to user. That is the reason why a VLSI technology, like TSMC180 nm, is used. Another requirement for biomedical purposes is to have a low power design, since the chip is going to work with a battery, and it is expected to have a relative long life. Taking these in consideration, an architecture for a low power SAR module and DAC are proposed and develop in this document.

1.2. The Process of Analog to Digital Conversion

ADCs or analog to digital converters translate analog signals, which are characteristic of most phenomena in the world, into a digital signal, to be processed in digital systems (Floyd, 2014). Since most of the signals are originally analog and signal processing uses digital signals in current electronics design, data converters are used in electronic circuits at the interface between analog and digital world. ADC play a fundamental role in most of the applications, such as industrial, telecommunications, automotive, medical, etc.

The process of data conversion can be explained in four steps: sampling, analog to digital conversion, digital signal processing, and digital to analog conversion [6]. These steps are shown in Figure 1.

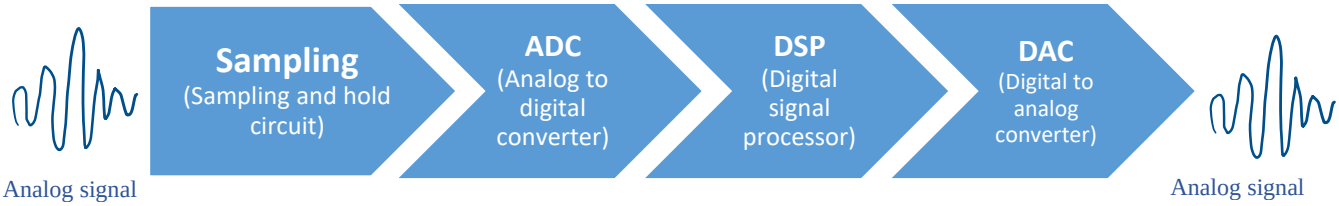


Figure 1: Analog to digital and digital to analog signal conversion process.

Sampling is the process that takes the value of the input signal every certain amount of time to have enough information of the input signal. This process converts an analog signal into a series of steps (like in a stairs), each one representing the amplitude of the signal at an instant in time. The more samples are taken, more accurate is the waveform representation.

The holding operation ensure the sampled value must be held constant for an instant defined, until the next sample is taken. This is necessary for the ADC to have time to process the sampled value. These sample and hold process results into a stairstep waveform that approximates the analog input signal, as is shown in Figure 2.

Since ADCs produces a digital output, they have a limited number of values that it can detect depending on the number of bits of resolution of the system. The resolution divides the range of voltage that the ADC can detect in levels of voltages. The sampled signal must be rounded up to one of these levels in order to continue the conversion. This process is called quantification.

Encoding is the process of converting the quantized signal into a series of binary codes that represent the amplitude of the analog input signal at each of the sample times. The ADC makes the codification in the time between sample pulses, or the time that sample and hold circuit is holding the sampled value.

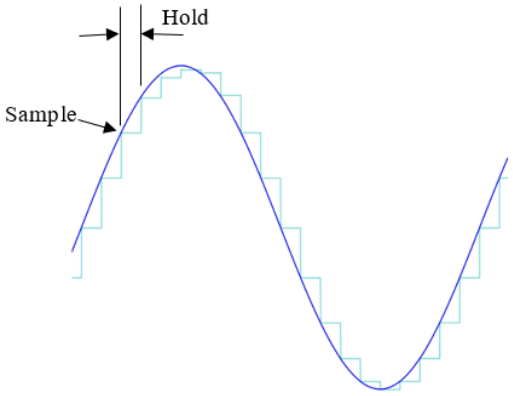


Figure 2: analog signal and its stairstep approximation

Digital Signal Processor (DSP)

Digital signal processors (DSP) are microprocessor chips that specializes in digital signal processing. Their architecture is optimized for those applications and their goal is to process data in real time. The DSP stage of Figure 1 takes its input from the ADC and produces a digital output, that goes to a DAC for conversion back into an analog form.

Digital to Analog Conversion

Digital to analog conversion is the process of transforming the result of the digital processing into an analog signal to send to back to "real world". Most of the DACs perform two basic functions: convert the digital input into an equivalent analog signal and then reconstructs the signal.

1.3. SAR ADC Architecture

There are several conversion techniques for analog signals to digital conversion. The proposal of this project uses a SAR ADC architecture. Figure 3 shows the main components of this architecture. The flow of an operation is that the circuit samples an input signal and compares it to several voltages that are generated by a DAC. The successive approximation register (SAR) controls the voltage of the DAC and saves the results of the comparison in a register. The output of register is the digital value of the sampled signal.

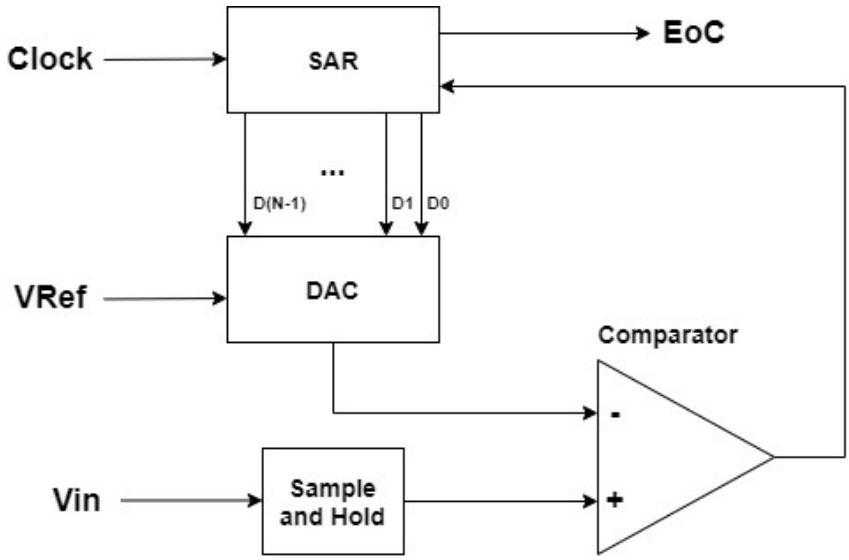


Figure 3: Block diagram of a standard SAR ADC architecture

Figure 3 shows the block diagram of a basic SAR ADC architecture. The main components of this architecture are a sample and hold circuit, a comparator, a DAC, and the SAR.

As stated before, the SAR controls the voltage that the DAC outputs. It starts by assigning a 1 to the most significant bit (MSB). This is done because that is the half-way value of the voltage that the DAC works with. The next step is to compare the DAC voltage with the sampled input voltage. If the analog input is higher than the DAC's voltage, the Comparator has an output of 1 (or the equivalent analog voltage) that the SAR saves, and then the SAR will assign an 1 to the next significant bit, repeating the process. But if the sampled voltage is smaller than the original DAC's voltage, then the SAR saves a 0, turn the MSB to 0 and assign a 1 to the next significant bit, starting the cycle again.

A SAR ADC converts a bit in each cycle. The resolution of the ADC is the number of bits used to represent the analog signal.

1.4. Capacitive Split-Array DAC Architecture

A SAR ADC needs a DAC to generate the voltages that will be compared against the input voltage. Since this project wants to be used in biosensor application, as stated in Section 1.1, our proposal must consume the least amount of power and must be as small as possible. For this reason, a capacitive split-array architecture for the DAC is selected.

Most of the DAC architecture uses either resistors or capacitors as their main component to generate the different voltages that are needed. The difference between them is that resistors are elements that are constantly consuming power since they only work in a state (static power), meanwhile capacitors consume power depending on the state the capacitor is in (dynamic power). If the capacitor is charging, then it consumes power, but if it's not charging then it does not. This behavior is a deciding factor to choose to use an architecture based on capacitors.

The most common capacitor architecture that is used is called Charge-Scaling Capacitors which is shown in Figure 4.

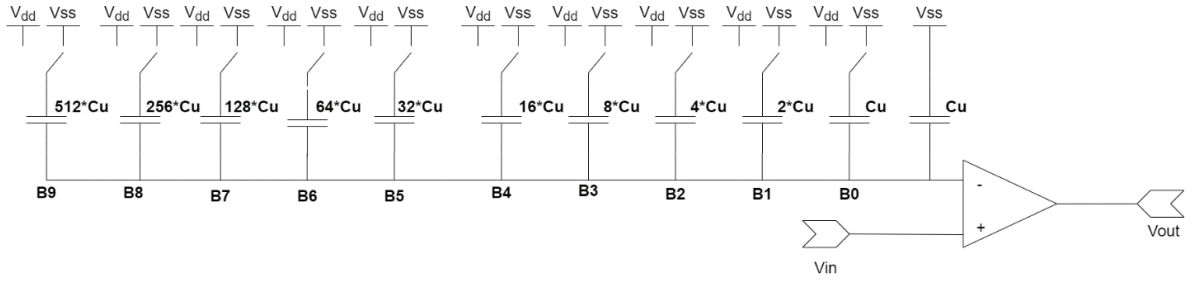


Figure 4: Schematic diagram of Charge-Scaling Capacitor topology for a 10-bit resolution DAC (Savitha & Venkat Siva Reddy, 2018)

The number of capacitors of Charge-Scaling Capacitor DAC defines the bits DAC works with. In this sense, Figure 4 shows the schematic of a 10-bits DAC. Each of the capacitor represents a bit, except for the last one that is used to provide the correct divisor factor in the inverting node. Each of the capacitor will switch between VSS and VDD, depending on what the SAR will assign to the control signals. This process will start from the MSB to the LSB.

The main problem with this architecture is that capacitor value of the MSB is 2^{n-1}; where *n* is the DAC resolution. The bigger the resolution of the DAC, the bigger the difference between the capacitance of the capacitors and it scales exponentially. In the example of Figure 4, the MSB capacitor is 512 times bigger than the smallest one, i.e., the unitary capacitance Cu. Having large capacitors is not desirable because they are area-consuming, which is an important trade-off when designing a chip. The more area they take the more expensive they are, and they consume more power.

To reduce the power consumption of DAC, the Capacitive Split-Array topology is utilized. It is shown in Figure 5.

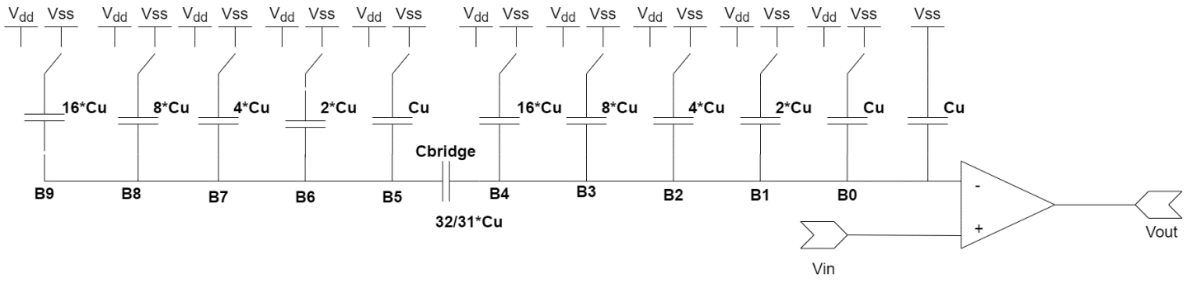


Figure 5: Schematic diagram of Capacitive Split-Array topology (Reyes et al., 2020)

The architecture called "Capacitive Split-Array" reduces power consumption by dividing the capacitors in two arrays, one for the least significant bits (LSB) b0 to b4 and the other one for the most significant bits (MSB) b5 to b9. Both arrays have capacitors from the same magnitude, and this works because of the bridge capacitor between both arrays (Cbridge). The only difference between the arrays is that the one for the LSBs also has the extra capacitor to provide the correct divisor factor in the inverting node. The value of the bridge capacitor can be calculated using equation 1:

$$C_{bridge} = \frac{Sum\ of\ LSB\ Capacitance}{Sum\ of\ MSB\ Capacitance} * Cu \qquad (1)$$

Eq 1: Equation for calculating the bridge capacitor in Capacitive Split-Array architecture

The DAC Capacitive Split-Array have all the benefits of the DAC Charge-Scaling Capacitors and uses smaller capacitors. The only disadvantage is that by adding the bridge

capacitor, the system stops having a linear behavior which could complicate mismatch fixing if the capacitor arrays are not properly balanced in silicon layout.

1.5. SAR ADC Block Diagram

Figure 6 shows the block diagram of the 10-bits low power SAR ADC proposal of this work.

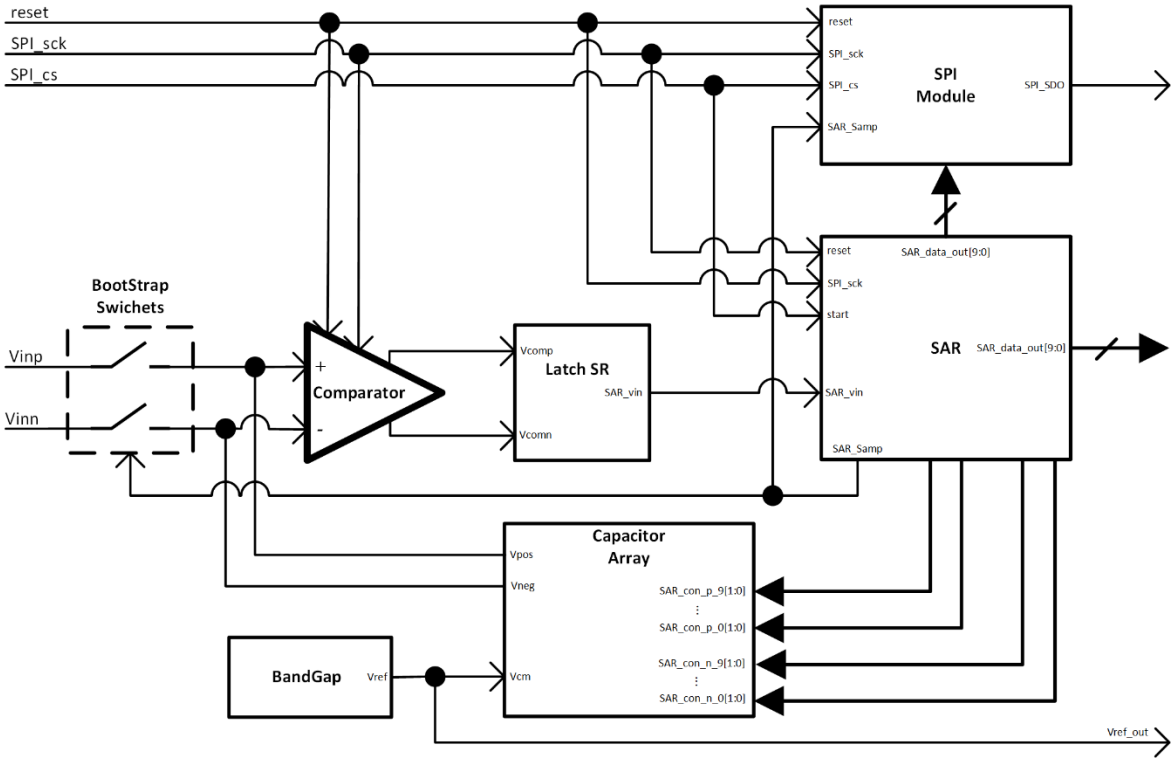


Figure 6: Block diagram of the proposed low power SAR ADC.

Comparing the proposal in Figure 6 against the basic architecture in Figure 3, there are several differences. In the proposal of this work, a SPI module is added to communicate the ADC module with another system. A bandgap module is used to generate a stable reference voltage that the array of capacitors uses. Since the proposal uses a dynamic comparator, a latch SR module is added to generate an output that the SAR module can understand.

1.6. Functional Description of Low Power SAR ADC Block Diagram

The SAR ADC is made by different functional blocks.

Table 1: SAR blocks description.

Block Name	Description
Serial Interface	The SPI module sends the ADC result to an external port, in serial format.
Bandgap Reference	This module supplies a reference voltage to the capacitor array. This module is supplied by 1.8V and provides a stable output of 900mV ±500µV.
Successive Approximation Register	It controls the DAC and saves the conversion result of the analog signal into a 10-bits register. These 10-bits, named *SAR_data_out [9:0]*, is the output that are sent directly to the SPI module and to external ports.
Capacitor Array	This module consists of a digital to analog converter block that generates voltages to be compared to two external signals, V_{ip} and V_{inn}. It works in conjunction with the Latch SR, Comparator, and the SAR to convert the analog signal into digital.
Comparator	It compares the two external signals, V_{ip} and V_{inn}, after they have been sampled by the Bootstrap Switches. It works in conjunction with the latch SR, SAR and capacitor array to convert the analog signal to digital.
Latch SR	After the comparison between V_{ip} and V_{inn} has been finished, the Latch SR module converts the differential output signal, from the comparator to a single ended output.

2. Successive Approximation Register

A Successive Approximation Register (SAR) is a digital module that is used in the process of converting an analog signal to a digital signal with the help of a comparator, a sample and hold circuit, and a DAC (Baker, 2010). The block diagram of a general SAR ADC is shown in Figure 7.

The SAR controls the voltage value that the DAC is providing, and this voltage is compared with an input voltage that is sampled by the sample and hold circuit. Depending on the result of the comparison, the SAR saves either a 1 or a 0 and then change their output B[n-1:0] to change the output voltage of the DAC (*Vdac*) to a bigger or smaller voltage. This process is repeated depending on the number of bits that DAC works with.

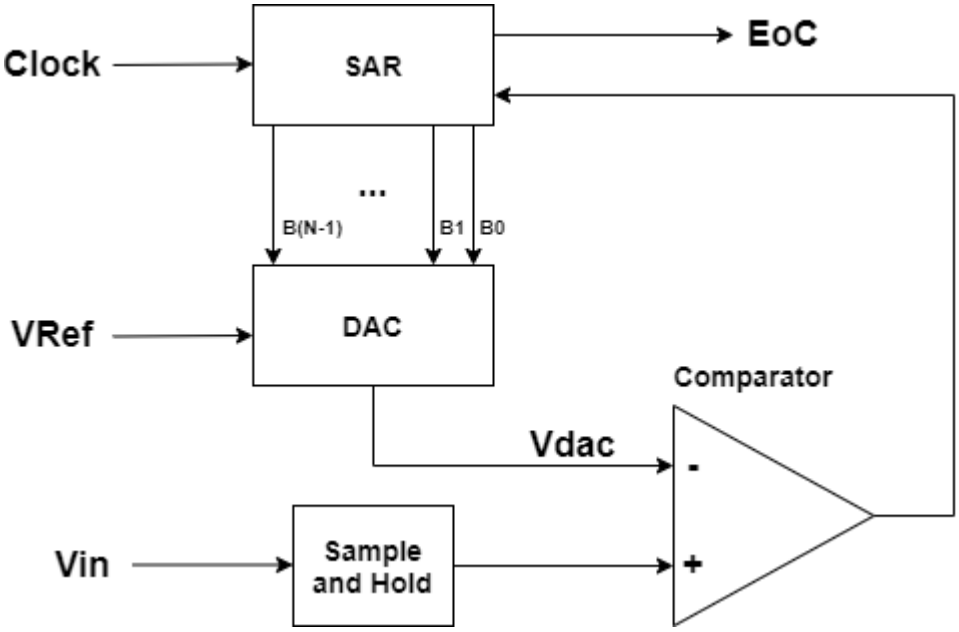


Figure 7: Circuit for Analog to Digital Conversion with SAR

2.1. Flow Diagram of SAR Algorithm

The functionality of the SAR is described in Figure 8. Variable *b* is the number of bits that the DAC works with, *b* is the resolution of the DAC. The variable *n* represents the iterations that the flow must be run.

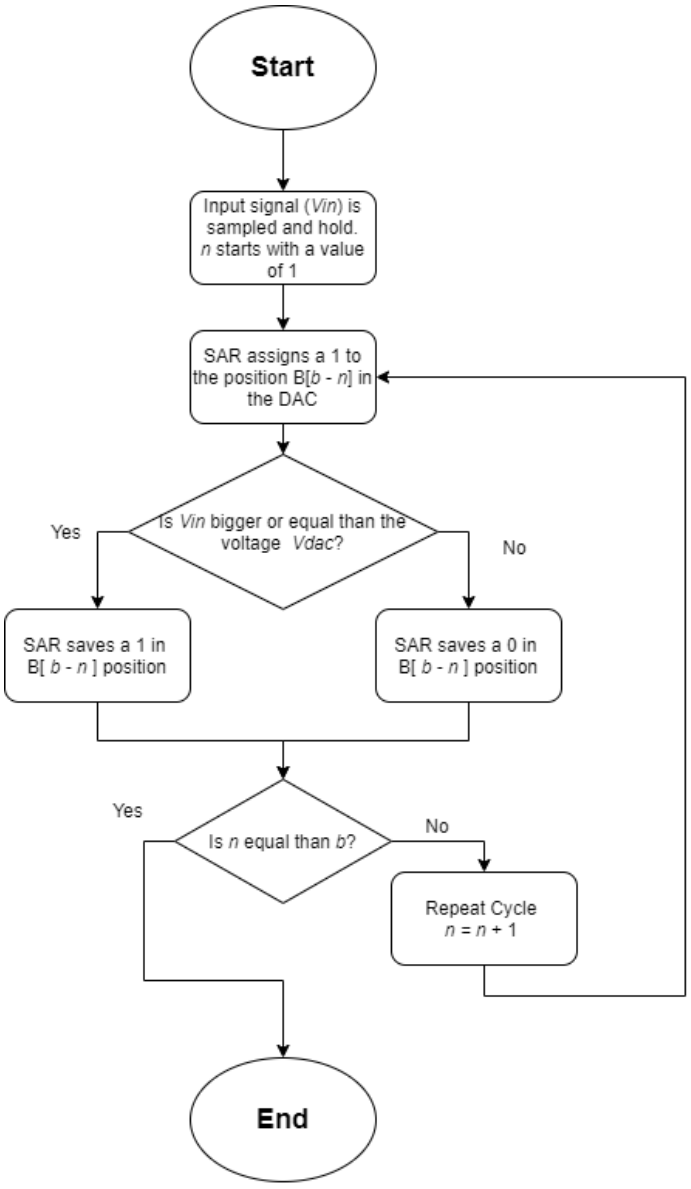


Figure 8: Flow Diagram of SAR algorithm

For example, if an ADC is working with 4 bits of resolution, b = 4, and n always starts with a value of 1. The ADC starts by sampling and holding an input signal (Vin) which is going to be used to compare against the value of the DAC (Vdac).

In the first cycle, the SAR assigns a 1 to the DAC in the position 4-1 = B3. In other words, the DAC receives a 1000_2. This value corresponds to half of the maximum voltage that the DAC can output, if the DAC works within the range of 0 to 1.8 V 1000_2 corresponds to 0.9 V. The comparator then compares Vin against the voltage of the Vdac (0.9 V). If Vin is bigger than 0.9 V, the SAR saves a 1 in the position B3. If Vin is smaller than 0.9 V, the SAR saves a 0 in the position B3. For this example, Vin has a value of 1.125 V which means that the SAR saves a 1 in position B3 ($1???_2$ where "?" stands for the bits that we have not converted yet).

Since n is smaller than b, the continues with the second cycle by increasing n by 1. This means that the SAR assigns a 1 to the DAC in the position 4-2 = B2 and keep the previous value that was saved in position B3. Since in the previous comparison the SAR saved a 1, the DAC receives a 1100_2 which represents a voltage of 1.575 V. Since Vin is smaller than 1.575 V, the SAR saves a 0 in position B2 ($10??_2$).

Since variable n is still smaller than b, the conversion has not finish. In this cycle n has a value of 3. The SAR assigns a 1 to the position 4-3= B1 to the DAC. It receives a value of 1010_2 as an input and outputs 1.125 V. The voltage of the DAC is equal to Vin which means that the SAR saves 1 in the position B1 ($101?_2$).

The conversion could end right now since the DAC has already reach a voltage that is equal to the input, but the system is not designed to reach to this conclusion. Therefore, a last cycle is needed. Now n = 4. The SAR assigns now a 1 to the DAC in the position 4-4 = B0. The DAC receives a 1011_2 which represents a Vdac of 1.2376 V. This voltage is bigger than 1.125 V (Vin), so the SAR saves a 0 in position B0 (1010_2). Finally, n is equal to b, which means that the conversion is finished.

2.2. SAR Implementation

The SAR algorithm is modeled in Verilog by using a state machine that implements the algorithm shown in Figure 8. The proposed SAR works with 10 bits since that is the resolution of the proposed ADC. The inputs and the outputs of this module are:

Table 2: Inputs and Outputs of the SAR module

Name	Direction	Description
SAR_clk	Input	Main Clock, works with 50 KHz
SAR_nrst	Input	Reset, active in 0
SAR_en	Input	Enable of the Module, 1 would enable 0 would turn off
SAR_in	Input	This input is connected to the output of the comparator.
SAR_ready	Output	This output indicates when the conversion is ready
SAR_sampling	Output	This output indicates that the SAR is in the sampling phase.
SAR_data_out[9:0]	Output	This bus represents the digital result of the conversion. It's the main output.
SAR_DAC_pos_#_out[1:0] # Represents the DAC switch number.	Output	These buses are the control signals which are connected to the non-inverting DAC input. After a reset, or at the start of a conversion, all of them start with a value of 01. Depending on the results of the comparator, this output could change to 00_2 or to 10_2.
SAR_DAC_neg_#_out[1:0] # Represents the DAC switch number.	Output	These buses are the control signals that are connected to the inverting DAC input. After a reset, or at the start of a conversion all of them start with a value of 01_2. Depending on the results of the comparator, this output could change to 00_2 or to 10_2.

Figure 9 shows the FPGA synthesis of the SAR module. In this schematic, which is generated by the Quartus synthesis tool, we can see all the modules that are included for the implementation.

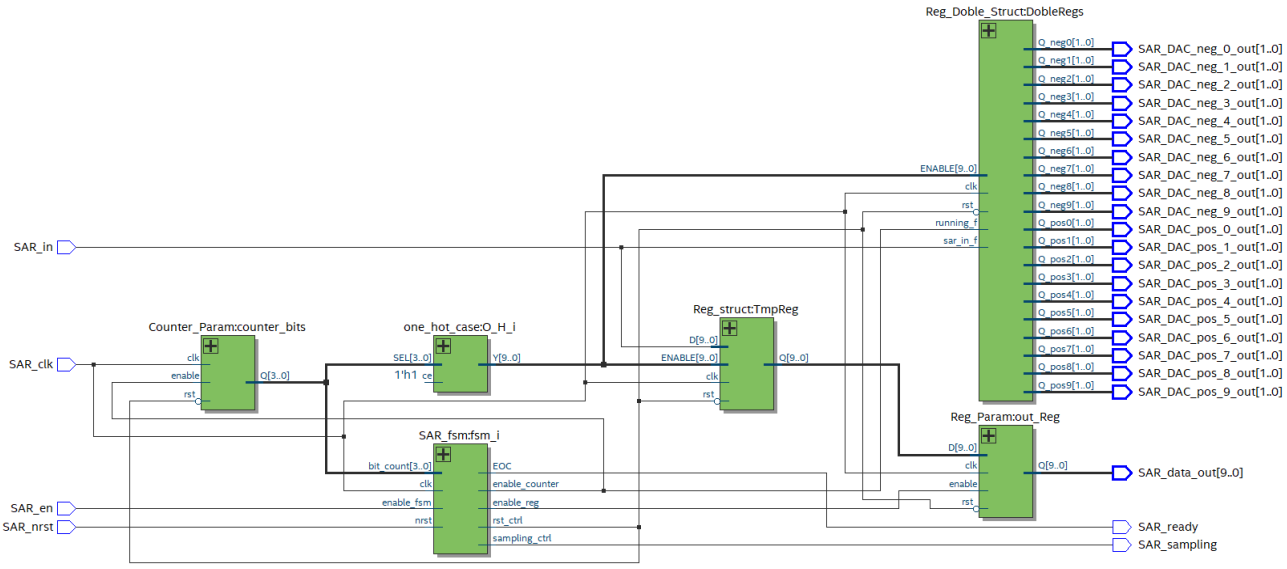


Figure 9: FPGA synthesis schematic of SAR implementation

The output signal SAR_sampling is added to control the bootstrap switches that samples the input signal (Vinp and Vinn) in the first cycle of a conversion. Since the ADC that we implemented is fully differential, this SAR provides the control signals for two DAC capacitive arrays. We use a low power switching scheme (this is explained in Chapter 3) that uses three references voltages for the capacitors instead of two (Zhu & Liang, 2015). This means that two bits are used to control each of the DAC bits, instead of only 1. That is the reason why all of control signals are 2-bit buses. These buses control the multiplexers that are connected to the capacitors and depending on the value of the control bus, it connects the capacitor to VDD (10_2), VCM (01_2) or VSS (00_2). This can be seen in Table 3.

Table 3: Voltage reference applied to the capacitive DAC accordingly to control signals values.

Control Signal Value	Voltage Reference	Voltage Label
00_2	0	*VSS*
01_2	0.9	*VCM*
10_2	1.8	*VDD*

All the control signals start with a value of 01_2. From Table 3 we can see that this binary value corresponds to a Vref = VCM. After each conversion, the control signal of the positive DAC either go to 00_2 (VSS) or to 10_2 (VDD) for the corresponding bit depending on the result of the comparator, a 0 or a 1 respectively. This happens because when a capacitor is connected to VSS the output voltage of that DAC increases, therefore if the result of the comparator is a 0, Vpos (the voltage that the positive DAC Outputs) must try to be bigger than the voltage Vneg (the voltage that the negative DAC Outputs). If a control signal bus for the positive DAC connects to VDD, the same control signal, but for the negative DAC, must connect to VSS, and vice versa.

For example, if in the first conversion the comparator result gives a 1 that means that the Vpos is bigger that the Vneg, therefore the SAR sends a 10_2 to connect the positive DAC's capacitor to VDD to make that Vpos lower and to the negative DAC a 00_2 to make that Vneg higher. What the SAR is trying to do is that Vpos and Vneg converge in almost the same value. This switching scheme is better explained in Chapter 3.

2.3. Simulation of SAR module

To simulate the functionality of the SAR module, a testbench is created. In this testbench, the inputs SAR_clk, SAR_en and SAR_nrst have the values to enable the SAR to work. The input SAR_in is the one that changes to verify that the outputs work correctly. This simulation verifies that SAR can save the values that are given as input, that it can use SAR_sampling and SAR_ready in the correct moments, and that the control signals are working correctly depending on the input that is given.

Test #1: In the first test, for half of the conversion cycle a 1 is given as an input to the SAR_in and after that it toggles to 0. The first outputs that we focus on are SAR_ready and SAR_sampling. SAR_sampling should have a value of 1 in the first cycle of each conversion cycle. SAR_ready should have a 1 when the conversion is ready and in a continuous process, SAR_ready is active before the next SAR_sampling becomes active. In Figure 10, the previous description can be seen in the blue signal for SAR_ready and in the pink one for SAR_sampling.

The next output that is focus on is SAR_data_out. We expect to see after each conversion that this output updates with the previous value of SAR_in, starting from the 3rd clock cycles. We can see how the output ends up with five 1's and five 0's, as intended.

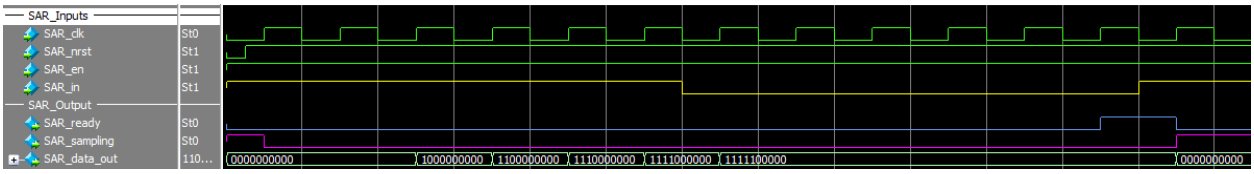


Figure 10: First part of simulation of Test 1

The final output that we focus on are the control signals, SAR_DAC_pos_#_out and SAR_DAC_neg_#_out. In Figure 11, we can see that buses starts with a value of 01 (VCM), and for the first 5 bits that are converted (9 to 5) the values corresponding to the positive DAC (SAR_DAC_pos_[9-5]out) change to 10 (VDD), meanwhile SAR_DAC_neg_[9-5]_out change to the contrary value, 00 (VSS). This is the behavior that we expect for an input of 1 in SAR_in. For the next 5 bits, this behavior is the contrary since the input SAR_in was toggled to 0. In the Figure 11 we can see how the control buses changes.

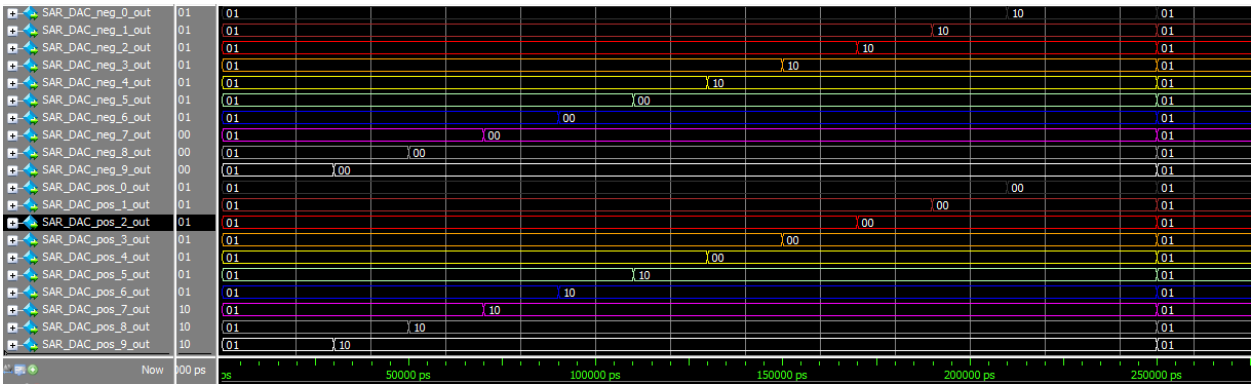


Figure 11: Second part of simulation of Test 1, focusing on how the control buses changes depending on the input SAR_in

Test #2: In the second test, the input signal SAR_in is toggled after each clock cycle. Since the first cycle is "sampling", the output SAR_data_out starts with a 0. In Figure 12, we can see that the bus SAR_data_out ends with a value of 0101010101_2 as expected since SAR_in toggles after every clock cycle.

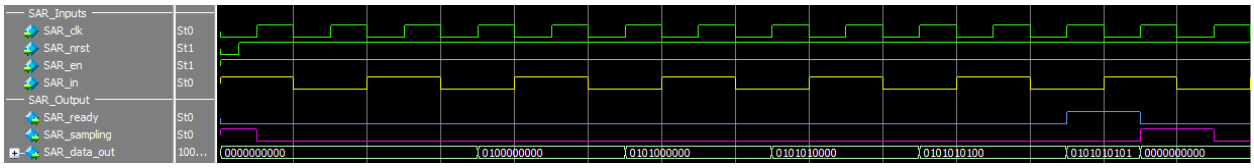


Figure 12: First part of simulation of Test 2, focusing on the input of the testbench

The output SAR_DAC_pos_#_out also changes after each cycle to either 00_2 or 10_2, depending on the value of the input SAR_in. If SAR_in has a 1 SAR_DAC_pos_#_out, changes to 10 (VDD), else it changes to (VSS). SAR_DAC_neg_#_out has the contrary value than SAR_DAC_pos_#_out. We can see this behavior in Figure 13.

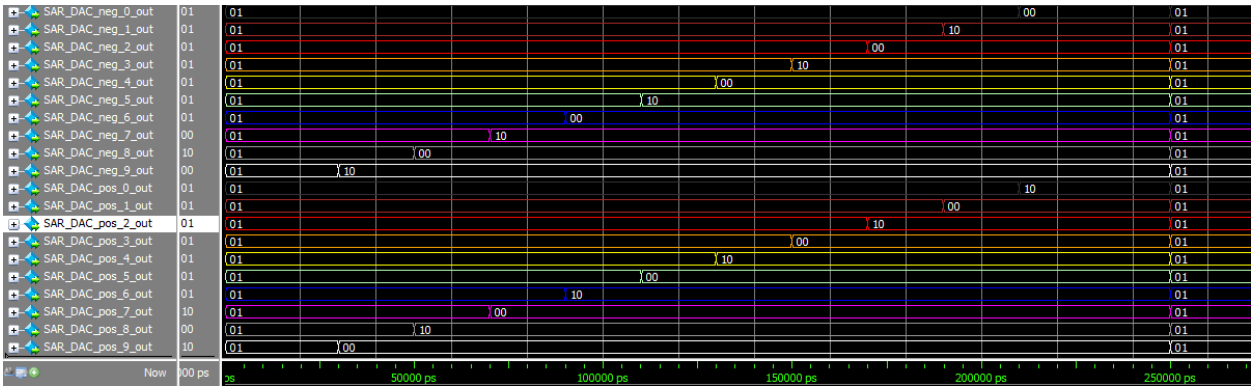


Figure 13: Second part of simulation of Test 2 focusing on how the control buses changes depending on the input SAR_in

Test #3: In this test we focus on doing two conversions, one after the other. The input signal SAR_in is toggling every two clock cycles, so we expect that the output is 1001100110_2 since the first cycle is for determining the sign of the analog input voltage. The result of the second conversion should be 0011001100_2 since it takes 3 cycles from the ending of the first conversion and the start of the next one. The inputs in those 3 cycles were 011_2. In Figures 14 and 15, this behavior can be verified.

Figure 14: First conversion of Test 3

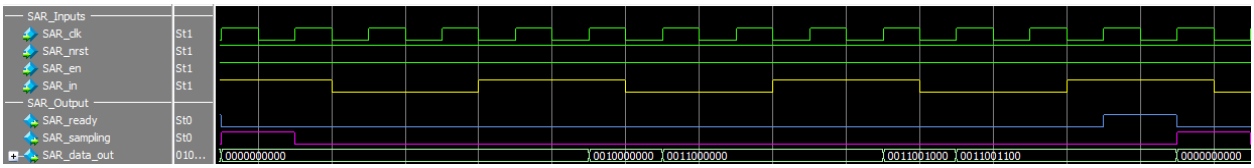


Figure 15: Second conversion of Test 3

We can see that the test was successful just by checking SAR_data_out in Figure 14 and Figure 15. In the Figures, 16 and 17, we can see how the control signals are acting. We can see that they are working correctly since we are already familiar of how they should act depending on SAR_in.

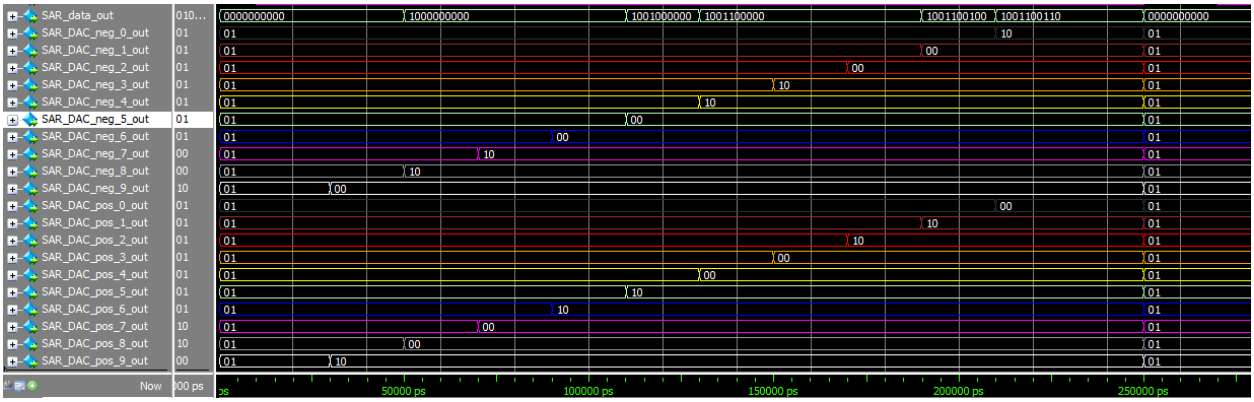


Figure 16: Control signals in first conversion of Test 3, focusing on the control buses

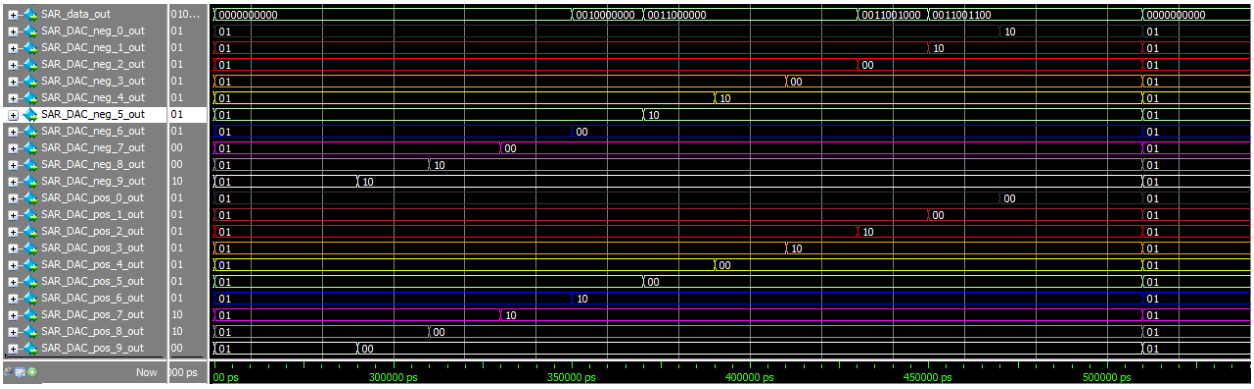


Figure 17: Control signals in second conversion of Test 3, focusing on the control buses

2.4. Logic Synthesis

The SAR is synthesized using RTL Compiler of Cadence (RC). To perform the logic synthesis of the SAR, the design constraints are defined. This information is presented in the sdc file (Appendix B of this document). Some of the most important constraints are the next ones:

- There's only 1 clock (Main_CLK) that works with a period of 4000 picoseconds, that represents a frequency of 250 MHz
- Both the input and output delay where 10% of the main clock's period
- The external driver is the buffer BUFFD12BWP7T.
- The maximum transition is 65% of the clock's period
- The maximum capacitance is 6000 femtofarads
- The maximum fanout is 50

In the Figure 18 one can see the schematic diagram of the SAR synthesized using the technology of 180 nm of TSMC.

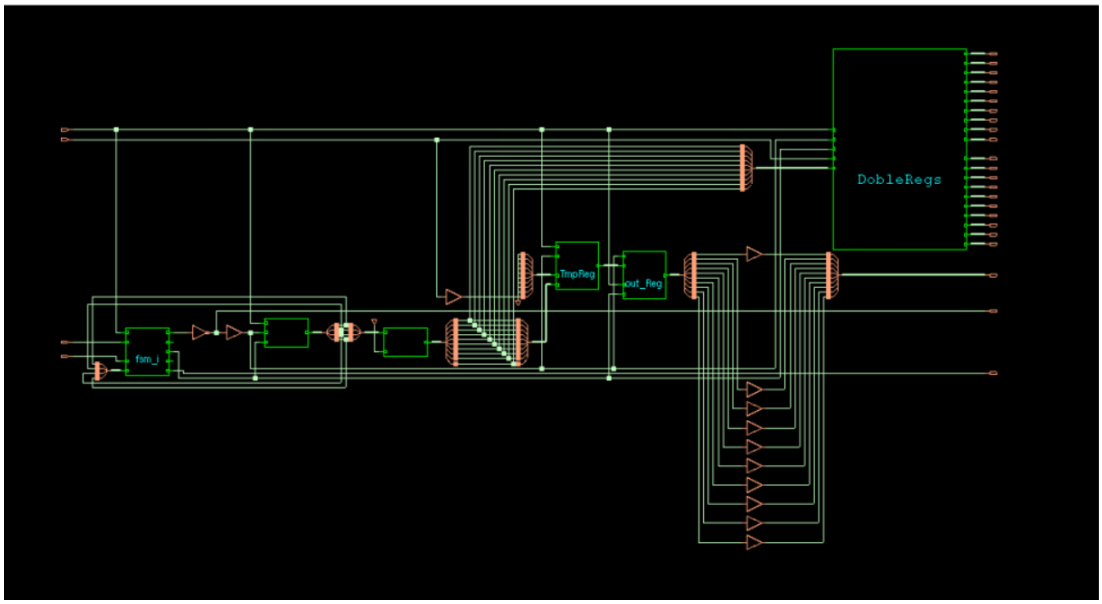


Figure 18: Schematic diagram of synthesized SAR module using RTL Compiler (RC).

The results of the main report are the following (the units for time are ps. and fF for capacitance).

```
Timing
--------

  Clock Period
----------------
Main_CLK 4000.0


  Cost     Critical       Violating
  Group   Path Slack TNS    Paths
-------------------------------------
C2C            937.0   0          0
C2O            989.7   0          0
I2C           2204.6   0          0
I2O         No paths   0
-------------------------------------
Total                  0          0

Instance Count
--------------
Leaf Instance Count             241
Sequential Instance Count        66
Combinational Instance Count    175
Hierarchical Instance Count      38

Area
----
Cell Area                          7555.878
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)   7555.878
Net Area                           2031.993
Total Area (Cell+Physical+Net)    9587.871

Power
-----
Leakage Power                      0.245 uW
Dynamic Power                      22044.785 uW
Total Power                        22045.030 uW

Max Fanout                         66 (SAR_clk)
Min Fanout                         0 (n_0)
Average Fanout                     2.3
Terms to net ratio                 3.0
Terms to instance ratio            3.2
```

The Critical Path Slack is the longest time that it takes to reach from one point of the design to another. This period of times is calculated for different groups: register to register (C2C), register to output (C2O), input to register (I2C) and input to output (I2O). For this design, the slack

cannot be calculated for the cost group I2O since there is no input which reaches and output without passing before to a register. This is not strange; this happens because the design is based on a State Machine type Moore. All the inputs reach this structure, and according to the state in which it is and the input it receives it assigns an output.

It is important that the design does not present any timing violation in any cost group, since lowest slack is the one that is going to be used to drive the synthesis.

Another thing to notice is that the max fanout is the main clock signal with 66. This is acceptable since we haven't added a clock tree to help improve the timing and diminish this factor.

3. Digital to Analog Converter

A digital to analog converter (DAC) is a device that converts a digital signal to an analog signal. SAR ADCs need a DAC for creating the voltages that is compared against the input of the ADC every cycle (Weste & Harris, 2011). In our proposal, two DACs are used since the proposed ADC is differential, it has a positive input (Vinp) and a negative input (Vinn) as seen in Figure 6 in Chapter 1. Both inputs work with 10 bits.

3.1. DAC Architecture

There are many architectures for DACs (Deng & Li, 2014; Singh et al., 2017), our proposal is based on the one called Capacitive Split-Array. DAC architectures based in capacitors are more power-consumption efficient than the ones that use resistances. Also, Capacitive Split-Array has the advantage of using smaller capacitors that other architectures like Charge-Scaling Capacitors (Savitha & Venkat Siva Reddy, 2018) .

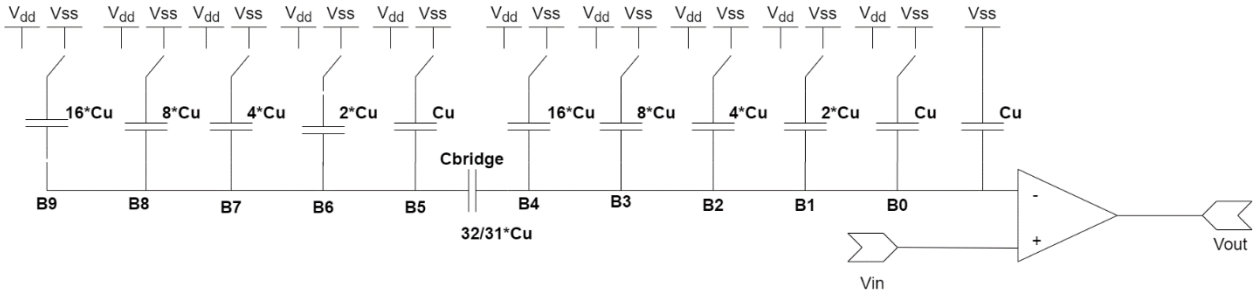


Figure 19: Ideal single ended 10-bit DAC using Split-Array architecture

Figure 19 shows an ideal 10-bit DAC using Split-Array architecture. One can see that it can be divided in 2 equal arrays of capacitors that have values equals to 2 elevated to the power of 0 to 4 (depending on the bit that they are representing). The array that represents the least valuable bits have an extra capacitor that goes directly to VSS with the value of the base capacitance (Cu). To determine the capacitance of the bridge capacitor (the capacitor that divides both arrays, between B5 and B4), Equation 1 is used:

$$C_{bridge} = \frac{Sum\ of\ LSB\ Capacitance}{Sum\ of\ MSB\ Capacitance} * C \qquad (1)$$

The capacitors are switched to VDD or VSS depending on the value that SAR assign to them, to generate a voltage that is used to compare against the inputs of the ADC.

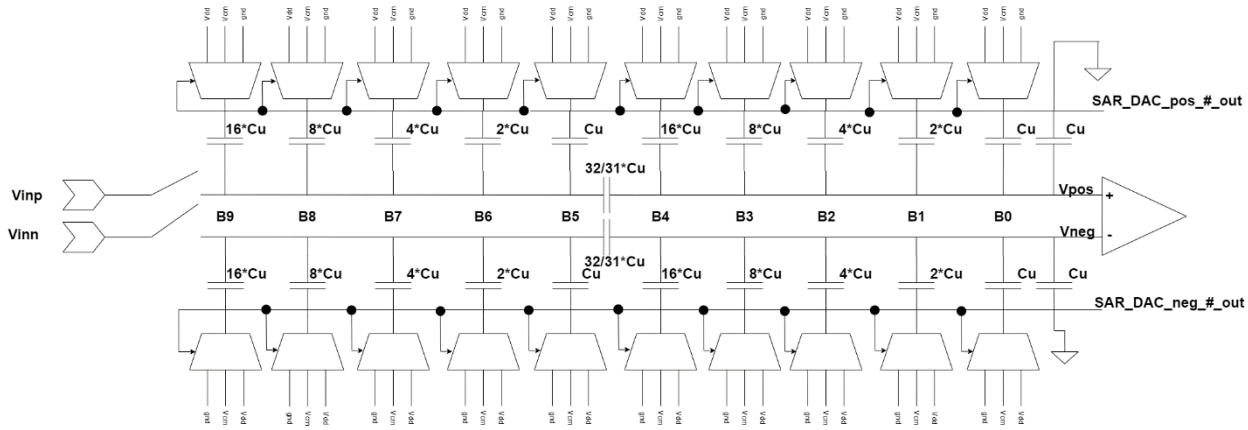


Figure 20: Schematic of the implementation of the DACS in this project

Figure 20 shows the proposal of the DAC module for the project. It consists of 2 Capacitive Split-Array, one for the positive input voltage, and one for the negative input voltage. As can be seen in Figure 20, capacitors can be switched to 3 signals, *VDD*, *VCM* or *VSS*, this is explained in section 3.2. Our base capacitance, *Cu,* is 1 pF. This value is chosen because it gives the best simulation results.

3.2. Low Power Switching Scheme

In the most basic Capacitive Split-Array architecture, the capacitors are switched between VDD or VSS. Figure 20 shows that the implementation of this project switches between 3 levels of voltage VDD, VSS, or VCM because the proposal is doing a low power switching scheme (Reyes et al., 2020).

The purpose of having 3 voltage levels is to make sure that all the changes of voltage that the capacitors do are no bigger than VDD/2. For example, if a capacitor is commuted from 0 V to 1.8 V for a conversion, it must switch again to 0 to be ready for the next conversion. Each of these changes takes certain amount of power that depends on the initial and final voltage of the capacitor. If the difference between the initial and final voltage is smaller, then the power needed for each change is also lower.

By using 3 voltage levels instead of only VDD and VSS, a virtual ground is created in VCM and each change only have a difference of VCM volts (0.9 V), lowering the difference between the initial and final voltage and taking less power (Ginsburg & Chandrakasan, 2005).

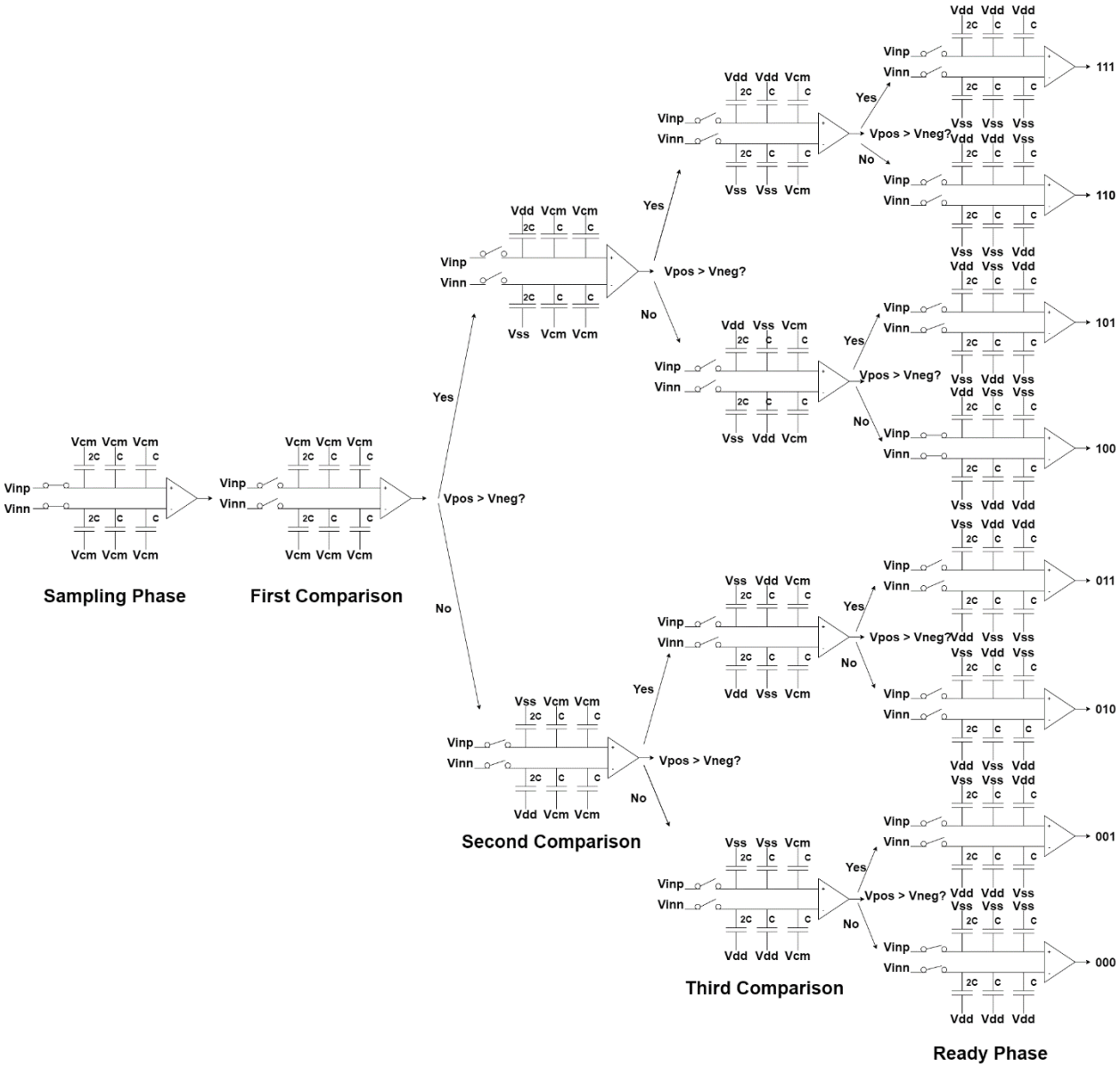


Figure 21: Diagram explaining the low power switching scheme for a 3-bit DAC

Figure 21 shows how this switching scheme works for a 3-bit DAC. At the beginning all the capacitors are connected to VCM since that is the virtual ground, or the value in the middle of VDD voltage. In this phase, the DAC is connected to the inputs Vinp and Vinn since it needs to sample those voltage. To refer to the node that is connected to the comparators non-inverting input the name Vpos is used and for the node connected to the inverting input the Vneg is used.

After the sampling phase, Vpos and Vneg are disconnected from the inputs and the first comparison starts. If Vpos is bigger than Vneg, then the biggest capacitor of the positive DAC connects to VDD and the one of the negative DAC connect to VSS. This is because connecting a capacitor to VDD lowers the voltage of the node and connecting it to VSS makes it bigger. The SAR is trying to make that the voltage in both nodes converge in almost the same value.

This process repeats depending on the number of bits that the DAC works with. After each cycle, the voltage of the nodes either increase or decrease by VDD/[2*2^(n)], where n is the cycle that is running. For example if in the first cycle (n = 1) Vpos is bigger than Vneg, then Vpos has to decrease its voltage 1.8/ [2*2^(1)] = 0.45 V, and Vneg add to itself the same 0.45 V by connecting the biggest capacitor of the negative branch to VSS. The SAR saves a 1 to its memory.

If in the second cycle (n=2) Vpos is smaller than Vneg, then Vpos increases 1.8/ [2*(2^2)] = 0.225 V by connecting the next capacitor of the positive branch to VSS, meanwhile Vneg decreases the same amount. The SAR saves a 0 to its memory

3.3. Transmission Gate Switch Design

To manage the 3 different reference voltages and one output, an analog multiplexer circuit is needed. A simple transmission gate switch is used as a base to design this multiplexor.

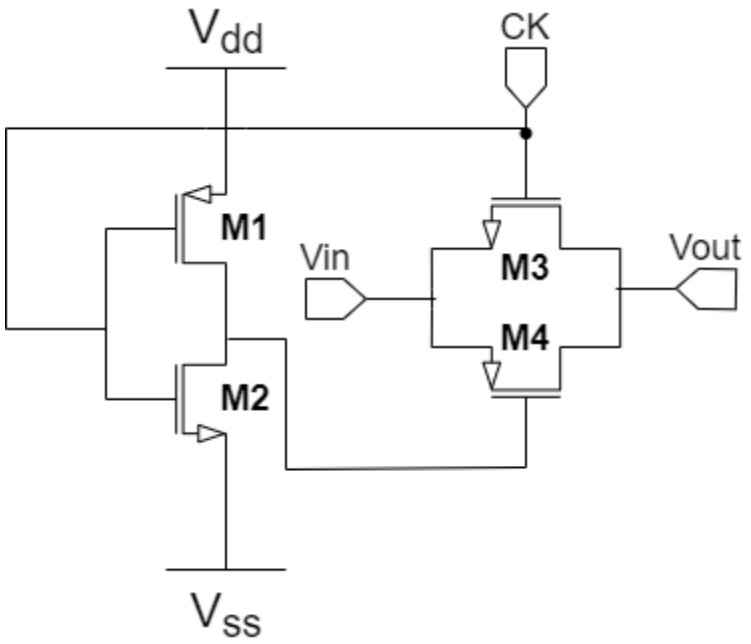


Figure 22: Schematic of a simple transmission gate switch

Figure 22 shows the schematic of a simple transmission gate switch. It consists of 2 pairs of nMOS and pMOS transistors. The inputs of the switch are Vin, and CK and the output is Vout. The switch receives a signal Vin and depending on the value of CK the output is either Vin or Z (high impedance).

Table 4: Dimensions of the transistors used in the transmission gate switch.

Transistor	w/l
M1	1.9u / 200n
M2	760n / 200n
M3	1.9u / 200n
M4	1.9 / 200n

The first pair of transistors are used to create an inverter for the signal CK. The second pair of transistors have their sources connected with the input Vin, and their drains are connected with Vout. CK is connected directly to the gate of the nMOS transistor, and the inverted CK signal is connected to the pMOS signal. When CK is driving a 1 (ideally since it's an analog component) the nMOS and the pMOS is activated and Vin is able to pass to Vout. When CK is driving a 0, then neither of the transistors are working and Vout is connected to nothing.

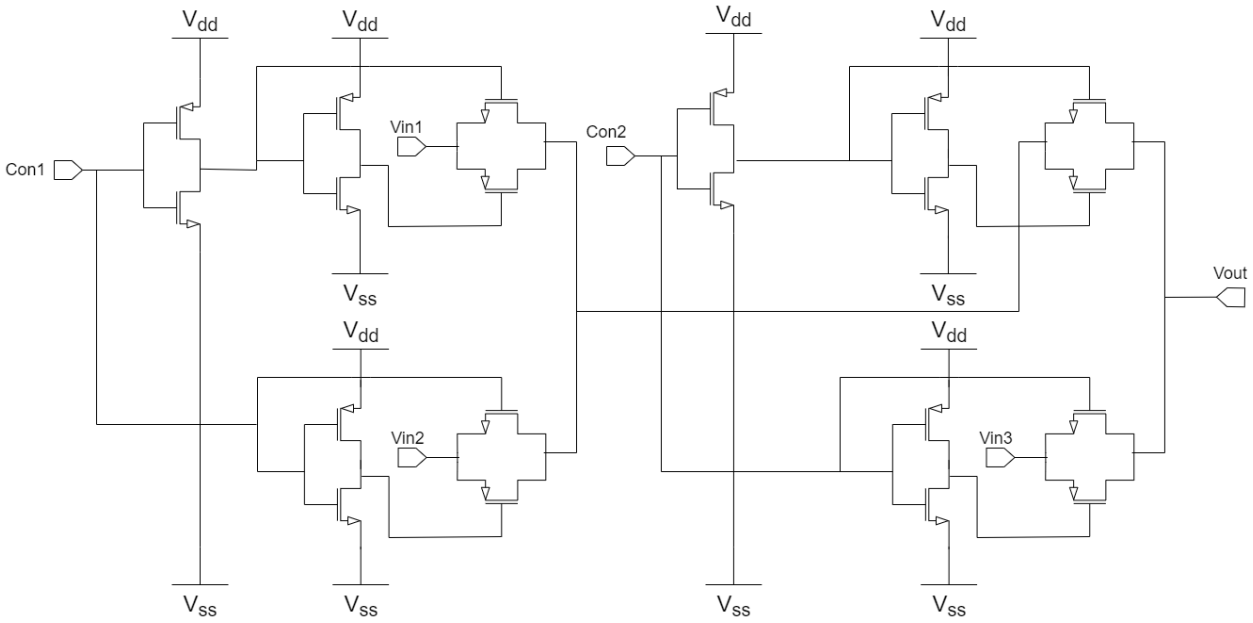


Figure 23: Schematic of a 3 to 1 multiplexor

Using 4 simple transmission gate switches a 3 input to 1 output multiplexer is created with 2 Control signals (Figure 23). The multiplexor functionality can be seen in Table 5

Table 5. Truth table of the multiplexor

Con2	Con1	Vout
0	0	Vin1
0	1	Vin2
1	0	Vin3
1	1	Vin3

3.4. Layout of Multiplexer and DAC Module

For creating the layout of the DAC module, the layout of the multiplexer is needed first to use as a base. After having this design, a second layout is designed using 20 multiplexers and the capacitors.

All the transistors have their width modified by dividing the magnitude. The purpose of this is to have all of them of the same dimensions (w = 360n and l = 200n), and to only modify the

number of fingers that they use. For example, now the transistor M2 (see Table 4) has a w =360n, l = 200 and it consists of 2 fingers to have a final w of 760n (360n x 2 fingers).

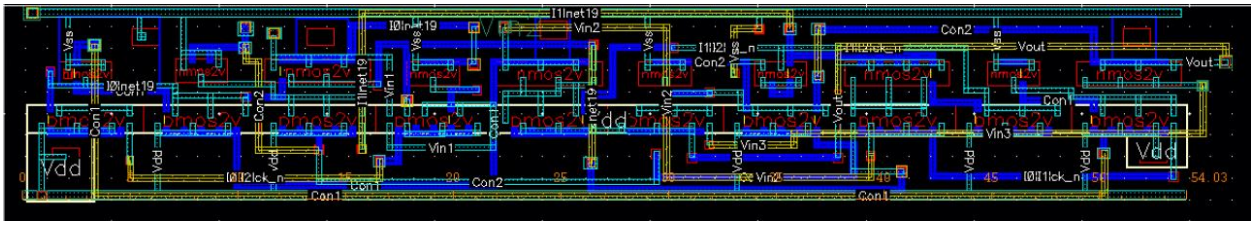


Figure 24: Layout of the multiplexer 4:1 using simple transmission gate switch.

Figure 24 shows the layout of the multiplexer 4:1 using simple transmission gate switch. Most of the routing is in Metal 1, Metal 2, Metal 3 (for ports) and Poly All of the nMOS are laid in a single row, and below were all the pMOS also in a single row.

The layout passes through DRC test and LVS test using Calibre from Siemens. The layout is clean of almost all DRC errors. The LVS results pass cleanly, giving us assurance that we pass to the complete layout.

```
                    OVERALL COMPARISON RESULTS

          #       ####################
        #  #      #                  #
  #   #          #     CORRECT      #
   # #           #                  #
    #            ####################

********************************************************************************************************************
                                        CELL  SUMMARY
********************************************************************************************************************

 Result         Layout                      Source
 -----------    -----------                 --------------
 CORRECT        bootstrap_sw_3pole_v2       bootstrap_sw_3pole_v2


********************************************************************************************************************
```

Figure 25: LVS result of multiplexer's layout

For the complete layout of the DAC, 2 equal arrays of capacitors are used. Since all the capacitors are multiples of Cu is possible to organize the capacitors in straight arrays of 16 x 4. The multiplexors are arranged in different positions surrounding these arrays to facilitate the connections between them.

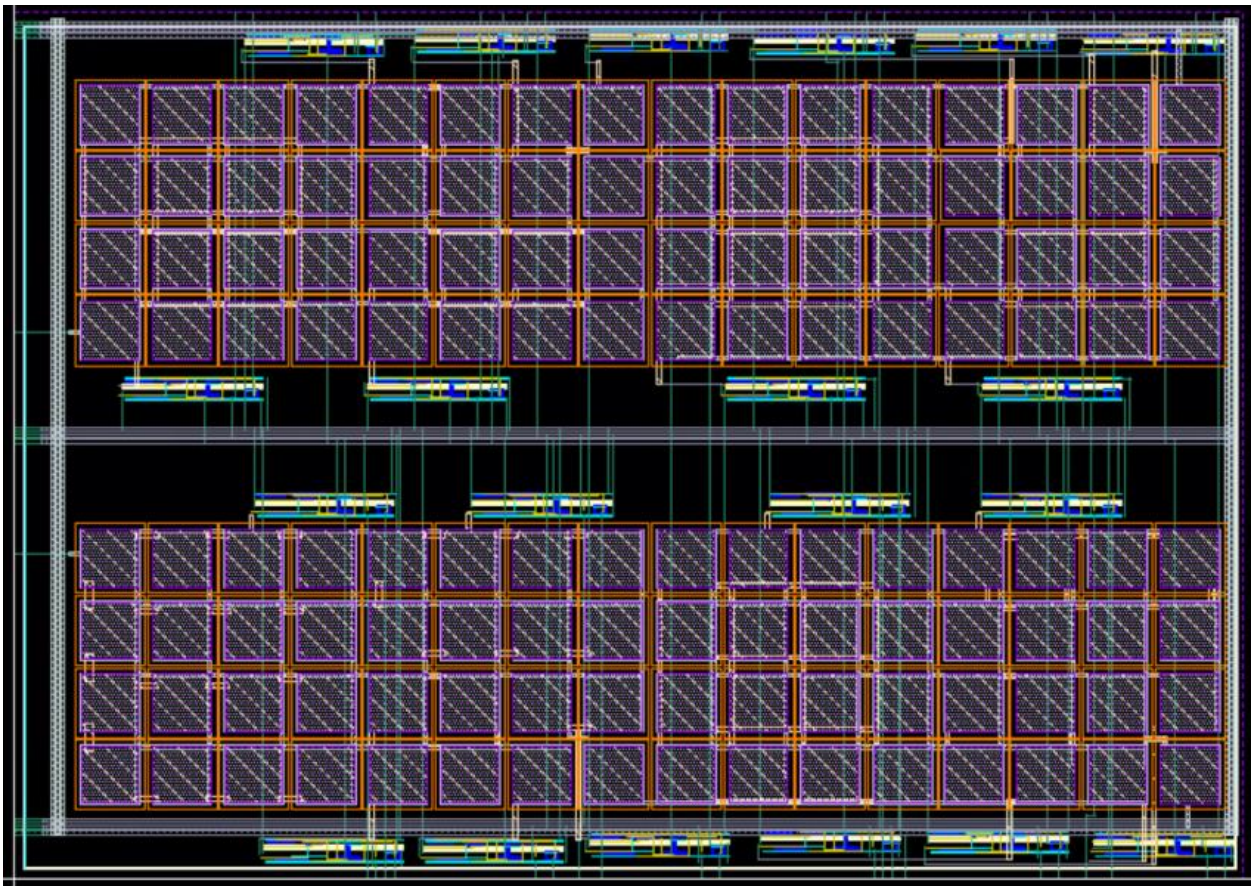


Figure 26: Implementation of the layout of the DAC module

Cell DAC_SplitArray_v1 Summary (Clean)

CELL COMPARISON RESULTS (TOP LEVEL)

```
                    #       ##################
                  #         #                #       ∓   ∓
                #   #       #     CORRECT    #         |
                 # #        #                #       \___/
                  #         ##################
```

Warning: Ambiguity points were found and resolved arbitrarily.

LAYOUT CELL NAME: DAC_SplitArray_v1
SOURCE CELL NAME: DAC_SplitArray_v1

INITIAL NUMBERS OF OBJECTS

	Layout	Source		Component Type
Ports:	45	45		
Nets:	207	207		
Instances:	640	200	*	MN (4 pins)
	1000	200	*	MP (4 pins)
	128	128		mimcap_2p0_sin (2 pins)
Total Inst:	1768	528		

Figure 27: LVS Test for the DAC

Figure 26 shows us that the LVS test results are clean. After passing both LVS and DRC tests, the design is exported to a LEF file that is used in the logical synthesis.

4. SAR and DAC Integration

SAR and DAC modules can now be simulated using a mixed signal simulator (see Appendix A) to verify that our designs work correctly. A testbench is created using the SAR module, the DAC module, an ideal comparator, ideal bootstrap switches, and several voltage sources so simulate the clock, the reset signal, the enable signal, and the input of the system.

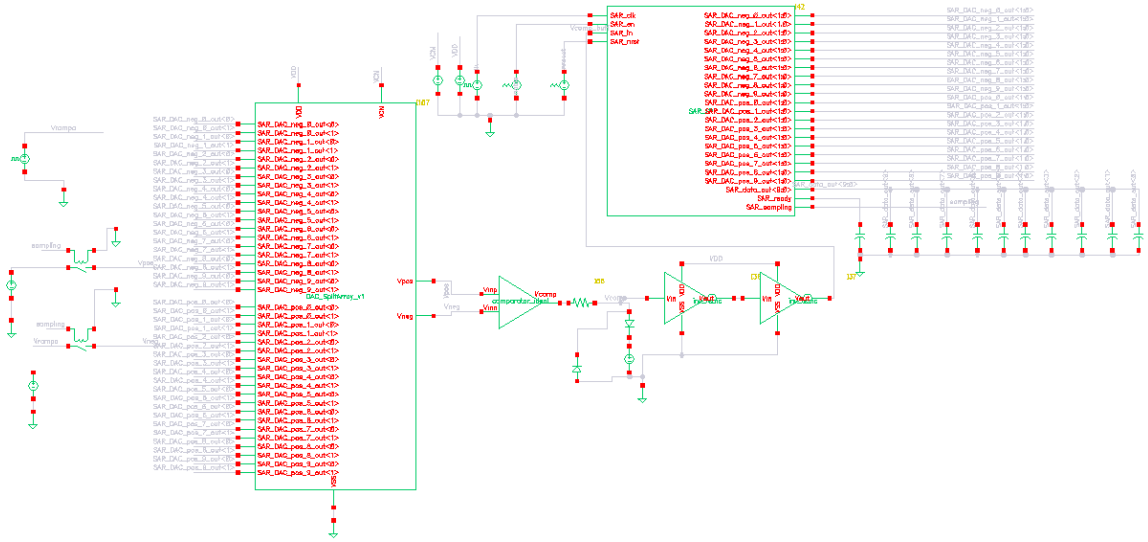


Figure 28: Schematic of the testbench of SAR and DAC module integration

Figure 28 shows the schematic of the complete testbench that is used for simulating the functionality of the SAR and DAC modules. For the ideal comparator, a voltage dependent voltage source is used with a big gain, and in between the inputs a really big resistance is used. A small circuit to limit the output voltage is used, and 2 inverters are used to create a signal that could only change from 0 to 1.8 imitating a digital signal. For this testbench a clock signal with a period of 20 µs is used, therefore 50 KHz of frequency.

To verify that the testbench works correctly, 2 test are designed. In the first one, several input voltages are used to see what output we get. This process is repeated for several pairs of input voltages. The output bus of the SAR register SAR_data_out, is then converted to an analog signal again and is compared against the difference between the original inputs.

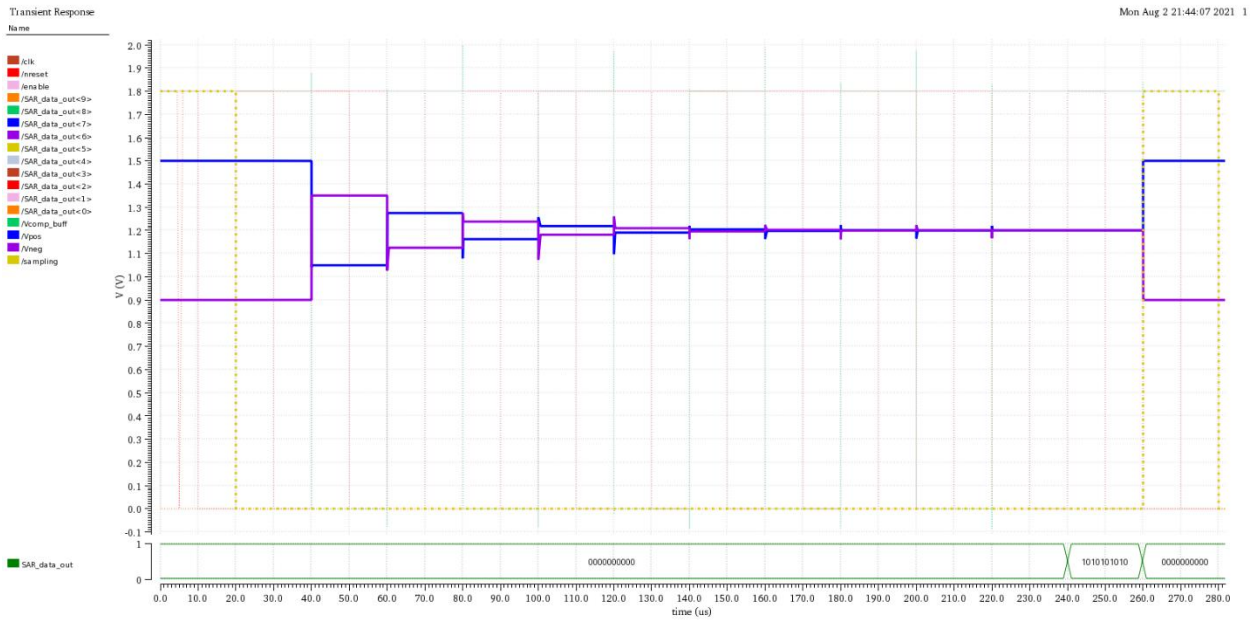


Figure 29: Simulation of conversion using the testbench

Figure 29 shows the result of one of these simulations. Since SAR_data_out only changes when the conversion is ready, these signals are converted in a digital bus to see the value. In this example, an input signal of 1.5 V and 0.9 V is used for positive input and negative input respective, the result of the conversion should be 0.6 V. We can see that the blue and the purple signals are the voltage of the DAC outputs that are connected to the positive inputs (Vpos) and negative input (Vneg) of the comparator respectively. These signals are that ones that are going to be changing depending on the result of the previous comparison.

To know if this conversion is successful, we take the value of the bus SAR_data_out 10101010102 and convert it to an analog signal. Although this is a 10-bit vector, the MSB only mean if the number is positive or negative, depending on if it MSB is a 1 or a 0 respectively. This also means that our resolution is 1.8/ (2^9) = 0.003515625 V per bit. If the number is positive, then the next bits are taken just as they are without modifying them. This is our case. For 0101010102 this is in decimal 170 and multiplying it by the resolution 170*0.003515625 V = 0.5976 V. Comparing it to the original difference voltage 0.6 – 0.5976 = 0.0024 V. This is less than the resolution, so it is an acceptable result for this case.

For a negative result (MSB is 0), the process to convert SAR_data_out to an analog value is different. All the bits are negated (except the MSB) and after that the vector is converted to

decimals. That result is then multiplied times the resolution and then times -1 since it is a negative number.

For example, if the result of SAR_data_out is 01010101012, the MSB indicates the conversation is negative. The rest of the vector is negated 1010101012 - > 0101010102, the is converted to decimal 170, and finally it is multiplied times the resolution to have a result of 0.5976 V. Since the MSB was a 0, the previous result must be multiplied times -1, ending in a result of -0.5976 V.

This test is repeated for several values and the results can be seen in Table 6.

Table 6: Results of applying test 1 to several values

Vinp	0.9	0.6	1.2	0.6	1.8	0.9
Vinn	0.6	0.9	0.6	1.2	0.9	1.8
Vdiff	0.3	-0.3	0.6	-0.6	0.9	-0.9
Vout	1001010101	0110101010	1010101010	0101010101	1011111111	0011111111
Binary conv	1010101	001010101	10101010	010101010	11111111	100000000
Decimal conv	85	85	170	170	255	256
Voltage out	-0.298828125	0.298828125	-0.59765625	0.59765625	-0.896484375	-0.9
Diff	-0.001171875	0.001171875	-0.00234375	0.00234375	-0.003515625	0
Error %	33%	-33%	67%	-67%	100%	0%

Vinp	1.4	0.4	1.5	0.3	1.7
Vinn	0.4	1.4	0.3	1.5	0.2
Vdiff	1	-1	1.2	-1.2	1.5
Vout	1100011100	0011100011	1101010100	0010101010	1110101010
Binary conv	100011100	100011100	101010100	101010101	110101010
Decimal conv	284	284	340	341	426
Voltage out	0.9984375	-0.9984375	1.1953125	-1.198828125	1.49765625
Diff	0.0015625	-0.0015625	0.0046875	-0.001171875	0.00234375
Error %	44%	-44%	133%	-33%	67%

Vinp	0.2	1.8	0	1
Vinn	1.7	0	1.8	1
Vdiff	-1.5	1.8	-1.8	0
Vout	0001010101	1111111111	0000000000	0111111111
Binary conv	110101010	111111111	111111111	000000000
Decimal conv	426	511	511	0
Voltage out	-1.49765625	1.79648438	-1.79648438	0
Diff	-0.00234375	0.00351562	-0.00351562	0
Error %	-67%	100%	-100%	0%

Explaining Table 6, the first 2 rows are the positive (Vinp) and negative (Vinn) input voltages respectively. Vdiff is the difference voltage between the inputs. Vout is the result of the conversion in the testbench in binary. In the row of binary conv, depending on the sign of Vout it ignores the MSB for a positive Vout, or it negates the vector if Vout is negative. Decimal conv is the result of converting the binary number of the previous row to decimal. The row of Voltage Out represents the value of multiplying the decimal number times the resolution. Diff is the difference between Vdiff minus Voltage Out, and finally Error % is the value of Diff divided by the resolution. If "Error %" is in between -100% and 100% it means that difference is less than one bit which is acceptable. For this project, a value between -200% and 200% is also accepted since it is only a difference of 1 bit.

Table 6 shows that the results are acceptable. The biggest "Error %" is 133% which means that the difference is at most a bit. With those results, it is concluded that our design passes this test.

The second test that is applied to the testbench is one in which we test all the possible cases of inputs that the design can have. A signal called Vramp is used which every 12 clock cycles (240 us) it increases its voltage by 0.003515625 V (the resolution of a bit) until reaching 1.8. This signal is used first as an input voltage for Vinp, meanwhile Vinn is constantly 0, and then it is running again but with the inputs inverted.

The objective of this test is to have an output that increases from 0 to 511. If successful, the DAC and SAR modules are verified that they work correctly for all the possible inputs.

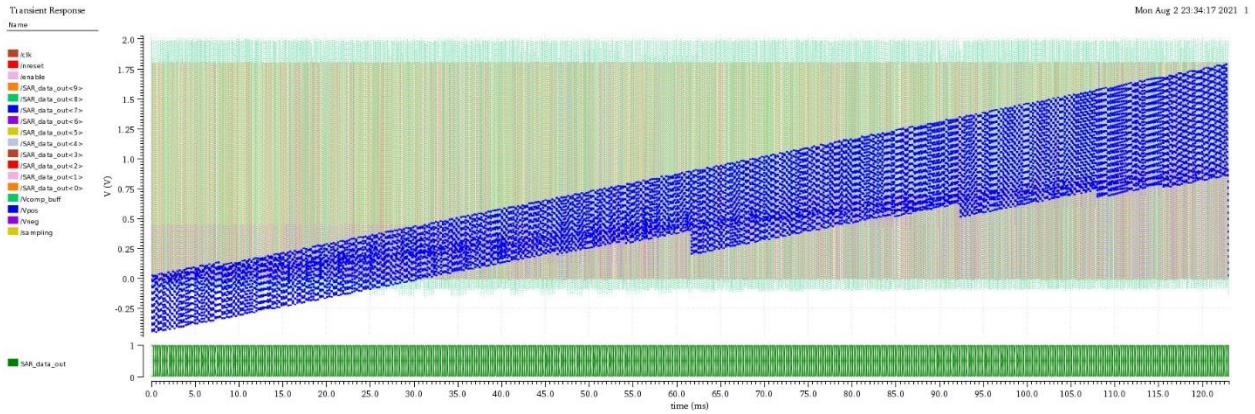


Figure 30: Simulation of ramp test connecting *Vramp* to the positive input *Vinp*

Figure 30 shows the result of doing this test for the case in which Vramp is connected to the positive input Vinp. This graph has a lot of information. To have an easier time analyzing this data, the results of SAR_data_out was exported as table. In this table, it is easier to see how the output is increasing by one after each iteration. To be sure that all these values were corrected we used a formula called Mean Square Error (MSE), Equation 2.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i) \qquad (2)$$

Where:

n is the number of data points used
Y_i is the expected result for iteration i
$\hat{Y}_i$ is the result that the testbench calculated for iteration i

To get the expected result, a test where all the components are ideal is run to have the best results possible and use as reference. The testbench has a MSE of 0.4186. There is no standard to say that this value is acceptable or not. Analyzing the table for expected results and testbenches results we concluded that they were similar and that they don't really have that many differences. For the standards of this project, this test has also pass successfully.

5. ADC Integration

Logic synthesis

The process of Logic synthesis transforms a digital circuit hardware description language (HDL) model into a *netlist* describing the hardware as a model represented by logic blocks and the connections between them using a standard-cell library as a reference. In this project, logic synthesis was done by Encounter RTL Compiler (RC). This process needs several files as inputs such as the HDL files that describe the circuit, a standard cell library to use as a reference, LIB and LEF libraries, and a constraint file. The output of RC is a netlist and some synthesis reports like total area, fanout, total power consumption, number of used cells, etc.

To integrate a full custom design module into the synthesis work flow, , such as the band gap, dynamic comparator, bootstrap switch, S-R latch, and the DAC, it is necessary to generate the Library Exchange Format (LEF) files of each of this cell. LEF Files contents information about the area that the cell uses, which materials are used for the metal connection and vias. With this information, the tool can have an idea about how to include this customs cells in the design.

The LEF Files are generated from the abstract view of the layout once it is DRC and LVS verified. Figure 31 to 34 show the abstracts of the full-custom modules of the proposed ADC.

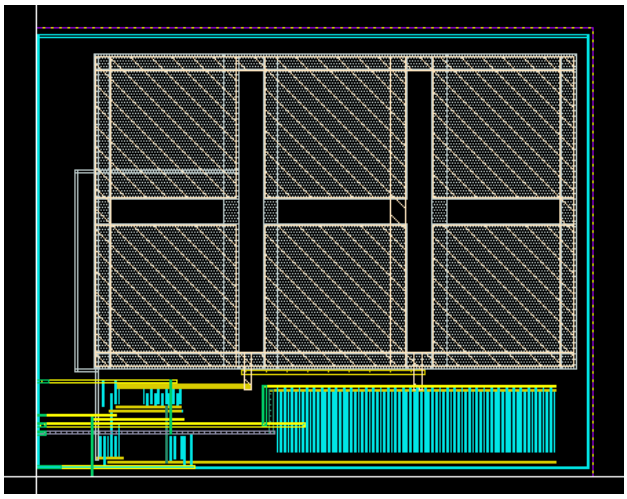


Figure 31: Boostrap switch abstract view layout.

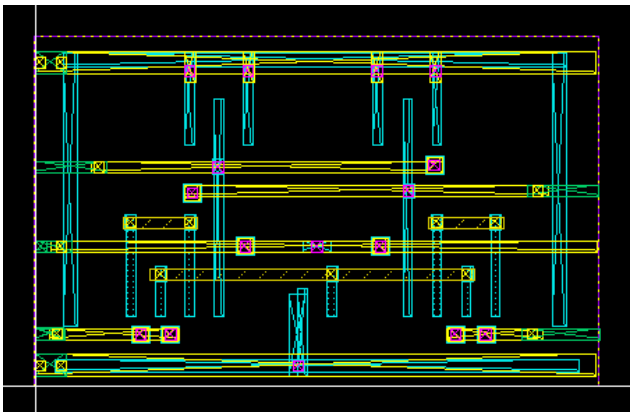


Figure 32: Dynamic comparator abstract view layout.

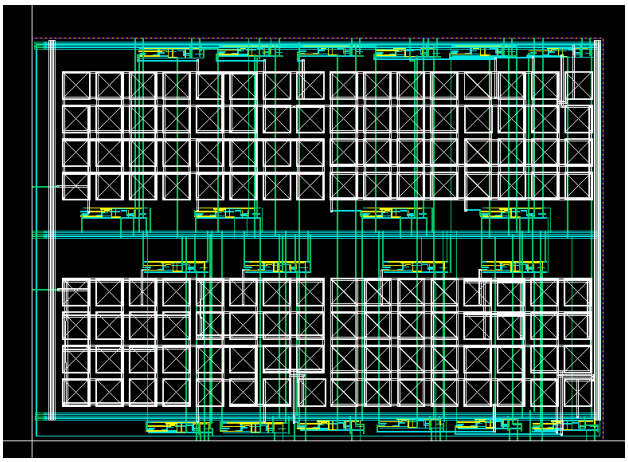


Figure 33: DAC capacitive abstract view layout.

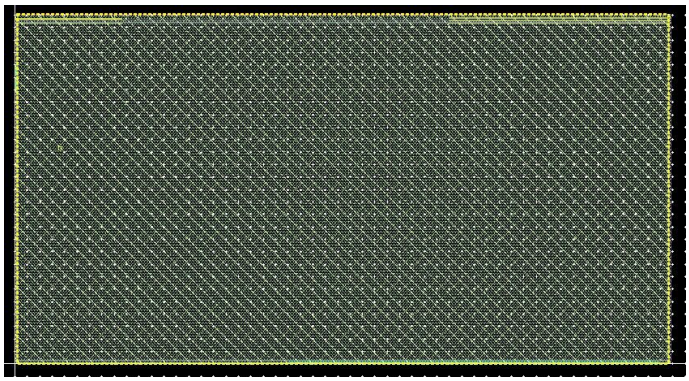


Figure 34: Band gap abstract view layout.

To perform a logic synthesis of a circuit, the constraints of the system must be defined. This information should be completely specified in a sdc file. The following lines shown some of the most important constraints of the proposed design:

- The existence of one clock (Main_CLK) that works with a period of 4000 ps
- The system has an input and an output delay of 10% the main clock's period
- The external driver for the ADC is the buffer BUFFD12BWP7T using the pin Z
- The max transition is 65% of the clock's period
- The max capacitance is 6000 fF
- The maximum fanout is 50

Figure 35 shows the schematic diagram of the synthesized DAC. All the digital modules are green, and the full-custom modules are shown in orange color. The schematic diagram shows all connections are correct. To have a tidier design, most of the digital modules are inside a top module called digital_inst, only the latch of the comparator is outside.

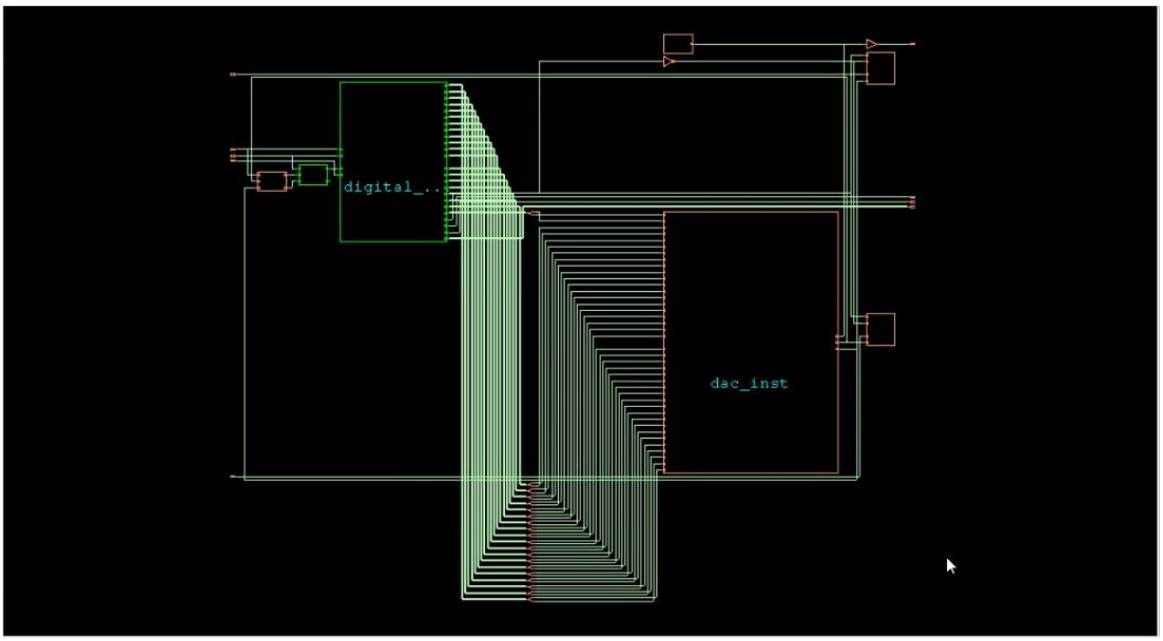


Figure 35: Schematic of synthesized ADC.

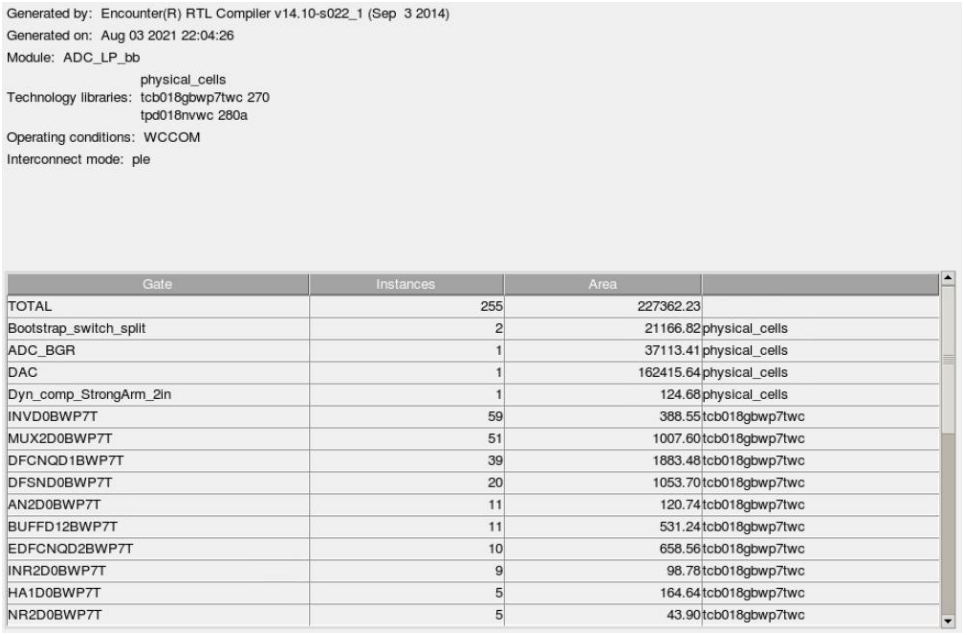
The synthesis process takes in consideration constraint for a typical case and a worst case. After finishing the process, the tool creates the following reports.

```
Resultados Typ:
  Module:                 ADC_LP_bb
  Technology libraries:   tcb018gbwp7twc 270
                          tpd018nvwc 280a
                          physical_cells
  Operating conditions:   WCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================


Timing
--------


  Clock Period
---------------
Main_Clk 4000.0



  Cost     Critical       Violating
  Group   Path Slack TNS    Paths
-------------------------------------
C2C            947.2   0          0
C2O           1490.4   0          0
default       3683.9   0          0
I2C           3329.5   0          0
I2O         No paths   0
-------------------------------------
Total                  0          0


Instance Count
--------------
Leaf Instance Count             249
Sequential Instance Count        72
Combinational Instance Count    177
Hierarchical Instance Count      44


Area
----
Cell Area                          227032.622
Physical Cell Area                 0.000
Total Cell Area (Cell+Physical)   227032.622
Net Area                           2232.648
Total Area (Cell+Physical+Net)    229265.270


Power
-----
Leakage Power                      0.139 uW
Dynamic Power                      2148.675 uW
Total Power                        2148.814 uW


Max Fanout                         68 (ADC_SPI_sck)
Min Fanout                         0 (digital_inst/sar_lp_inst/DobleRegs/sw9n/SW_ctrl[0])
Average Fanout                     2.5
Terms to net ratio                 3.3
Terms to instance ratio            3.6
Runtime                            14.996 seconds
Elapsed Runtime                    31 seconds
RC peak memory usage:              175.00
EDI peak memory usage:             no_value
Hostname                           FV00
Final Runtime & Memory.
=========================================
The RUNTIME after FINAL is 15 secs
and the MEMORY_USAGE after FINAL is 173.00 MB
=========================================
```

```
Results WC:
   Module:                 ADC_LP_bb
  Technology libraries:   tcb018gbwp7twc 270
                          tpd018nvwc 280a
                          physical_cells
  Operating conditions:   WCCOM
  Interconnect mode:      global
  Area mode:              physical library
============================================================


Timing
--------


  Clock Period
----------------
Main_CLK 4000.0



  Cost     Critical       Violating
  Group   Path Slack TNS    Paths
-------------------------------------
C2C            934.7   0          0
C2O           1348.1   0          0
default       3574.9   0          0
I2C           3250.1   0          0
I2O         No paths   0
-------------------------------------
Total                  0          0

Instance Count
--------------
Leaf Instance Count             249
Sequential Instance Count        72
Combinational Instance Count    177
Hierarchical Instance Count      44

Area
----
Cell Area                         227032.622
Physical Cell Area                0.000
Total Cell Area (Cell+Physical)   227032.622
Net Area                          2232.648
Total Area (Cell+Physical+Net)    229265.270

Power
-----
Leakage Power                     0.139 uW
Dynamic Power                     2150.416 uW
Total Power                       2150.556 uW

Max Fanout                        68 (ADC_SPI_sck)
Min Fanout                        0 (digital_inst/sar_lp_inst/DobleRegs/sw9n/SW_ctrl[0])
Average Fanout                    2.5
Terms to net ratio                3.3
Terms to instance ratio           3.6
Runtime                           13.993 seconds
Elapsed Runtime                   26 seconds
RC peak memory usage:             175.00
EDI peak memory usage:            no_value
Hostname                          FV00
Final Runtime & Memory.
```

The report describes that a main clock (Main_CLK) is used for both cases with a 4000 ps period, therefore working with a frequency of 250 MHz. There are no time violations in the design cost group, which is something that the design needs to have to be correct.

The synthesis report also describes the number of instances that are used in the design and the total area. The tool reports that the physical cell area is 0, which is not true since the synthesis take in consideration the LEF files of the custom cells. In a different report, the information of the physical cells is described. In figure 36, in the first 5 rows after total, the information of the area of the custom cells can be seen.



Generated by: Encounter(R) RTL Compiler v14.10-s022_1 (Sep 3 2014)
Generated on: Aug 03 2021 22:04:26
Module: ADC_LP_bb
Technology libraries: physical_cells
tcb018gbwp7twc 270
tpd018nvwc 280a
Operating conditions: WCCOM
Interconnect mode: ple

Gate	Instances	Area	
TOTAL	255	227362.23	
Bootstrap_switch_split	2	21166.82	physical_cells
ADC_BGR	1	37113.41	physical_cells
DAC	1	162415.64	physical_cells
Dyn_comp_StrongArm_2in	1	124.68	physical_cells
INVD0BWP7T	59	388.55	tcb018gbwp7twc
MUX2D0BWP7T	51	1007.60	tcb018gbwp7twc
DFCNQD1BWP7T	39	1883.48	tcb018gbwp7twc
DFSND0BWP7T	20	1053.70	tcb018gbwp7twc
AN2D0BWP7T	11	120.74	tcb018gbwp7twc
BUFFD12BWP7T	11	531.24	tcb018gbwp7twc
EDFCNQD2BWP7T	10	658.56	tcb018gbwp7twc
INR2D0BWP7T	9	98.78	tcb018gbwp7twc
HA1D0BWP7T	5	164.64	tcb018gbwp7twc
NR2D0BWP7T	5	43.90	tcb018gbwp7twc

Figure 36: Report of area utilized in the ADC project

The total power that the ADC consumes is 6587.202 uW. This is high for the intended and should be improve. And finally, the maximum fanout is the signal ADC_SPI_sck with 68 connections. This is expected since the clock tree has not been added in this stage.

Physical synthesis

The netlist generated by the logic-synthesis process, is given to a place and route tool, in this case Encounter Digital Implementation (EDI), to perform the physical-synthesis process. The process objective is to optimize the design in terms of area, routing, and timing.

The inputs that EDI needs to run the physical synthesis are netlist created in the logic synthesis, the LEF libraries of the custom cells, timing libraries and timing constraints. Every cell must have a LEF library that contains detail information of area and routing. In this project, timing libraries were not generated of the full-custom modules.

In this project, the physical synthesis consisted in six steps: floorplan definition, power ring creation, placement of the cells, clock tree synthesis, and routing. Figure 37 shows the placement of the modules (DAC, dynamic comparator, bootstrap switches, and band gap), standard cells and connections.

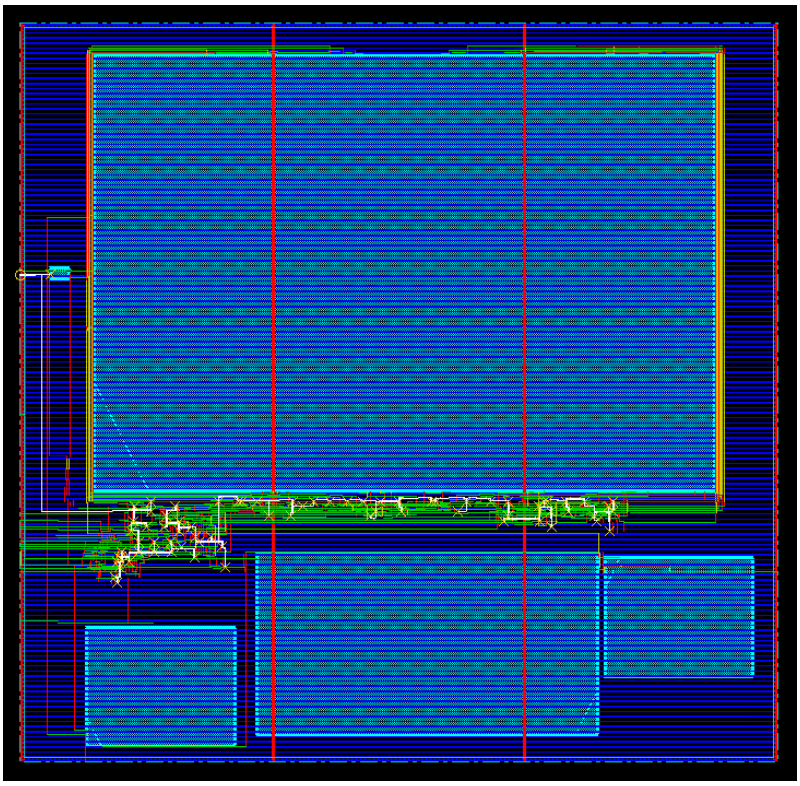


Figure 37: Physical view of the ADC after physical synthesis.

Figure 38 shows the result of displaying only the Clock Tree, ignoring all other routing.

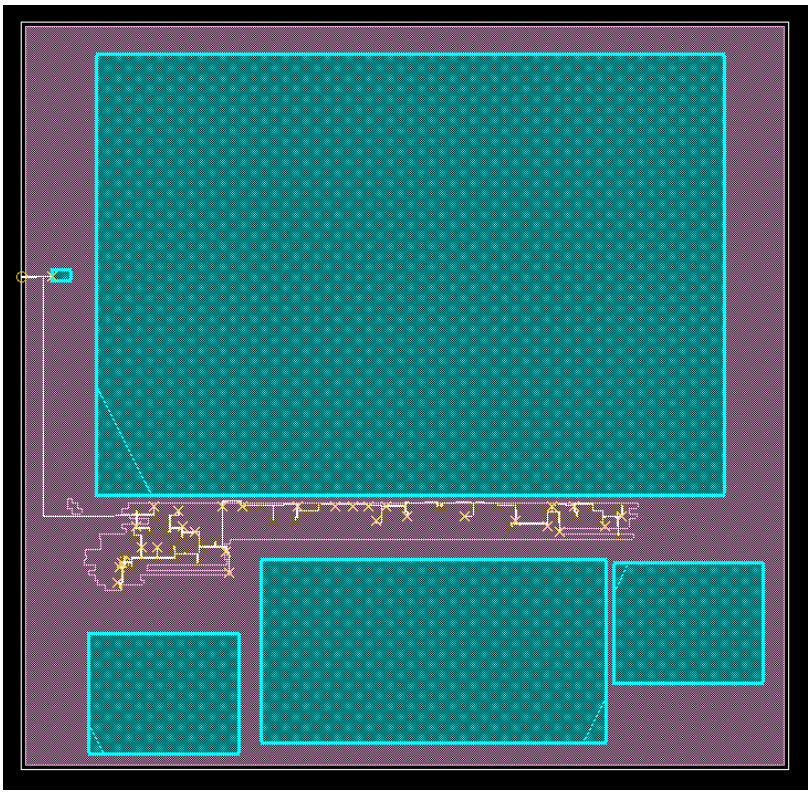


Figure 38: Clock Tree view of the ADC design

Next report shows the geometry verification. This test check for violations of wiring, shorts between nets, overlapping between cells, etc.

```
*** Starting Verify Geometry (MEM: 884.1) ***

  VERIFY GEOMETRY ..... Starting Verification
  VERIFY GEOMETRY ..... Initializing
  VERIFY GEOMETRY ..... Deleting Existing Violations
  VERIFY GEOMETRY ..... Creating Sub-Areas
                  ..... bin size: 8320
  VERIFY GEOMETRY ..... SubArea: 1 of 1
  VERIFY GEOMETRY ..... Cells         :  0 Viols.
  VERIFY GEOMETRY ..... SameNet       :  0 Viols.
  VERIFY GEOMETRY ..... Wiring        :  182 Viols.
  VERIFY GEOMETRY ..... Antenna       :  0 Viols.
  VERIFY GEOMETRY ..... Sub-Area: 1 complete 182 Viols. 0 Wrngs.
VG: elapsed time: 1.00
Begin Summary ...
  Cells      : 0
  SameNet    : 0
  Wiring     : 2
  Antenna    : 0
  Short      : 180
  Overlap    : 0
End Summary

  Verification Complete: 182 Viols.  0 Wrngs.

**********End: VERIFY GEOMETRY**********
```

Connectivity verification

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Sun Aug 8 22:12:17 2021

Design Name: ADC_LP_bb
Database Units: 2000
Design Boundary: (0.0000, 0.0000) (587.7050, 572.3200)
Error Limit = 1000; Warning Limit = 50
Check all nets
Net vref_gnd_w: no routing.
Net vref_vdd_w: no routing.

Begin Summary
    2 Problem(s) (ENCVFC-98): Net has no global routing and no special routing.
    2 total info(s) created.
End Summary

End Time: Sun Aug 8 22:12:18 2021
Time Elapsed: 0:00:01.0

******** End: VERIFY CONNECTIVITY ********
```

DRC verification

```
*** Starting Verify DRC (MEM: 959.9) ***

  VERIFY DRC ..... Starting Verification
  VERIFY DRC ..... Initializing
  VERIFY DRC ..... Deleting Existing Violations
  VERIFY DRC ..... Creating Sub-Areas
  VERIFY DRC ..... Using new threading
  VERIFY DRC ..... Sub-Area: 1 of 1
  VERIFY DRC ..... Sub-Area: 1 complete 182 Viols.

  Verification Complete: 182 Viols.

 *** End Verify DRC (CPU: 0:00:00.1 ELAPSED TIME: 0.00  MEM: 1.5M) ***
```

6. Conclusions and Future Work

A 10-bit successive approximation register (SAR) and a digital to analog converter (DAC) were successfully designed for an ADC using the technology of TSMC for 180 nm. The DAC has a capacitive split array architecture with a unitary capacitance of 1 pF and works with a three-reference voltage switching scheme to achieve low power consumption. The SAR controls the DAC and is the control unit of the conversion process.

The SAR design is synthesized using a clock with a frequency of 250 MHz. The design does not present any issue of timing.

The functionality of both modules is verified with a testbench and a mixed signal simulator. The design responds correctly to independent inputs. It also responds correctly to a test in which a ramp signal of voltage is used as an input to test all the possible cases.

The layout for the DAC is designed and it passed both DRCs and LVS test. The SAR is logically synthesized using RTL Compiler of Cadence.

These 2 modules are integrated to a SAR ADC and this was logically and physical synthesized. The other modules of the SAR ADC are implemented by Karina Castañeda (Castañeda Villalpando, 2021) and Luis Angel Gonzalez (Gonzalez Ornelas, 2021) using the same technology.

An improvement that could be done to the proposed ADC is to lower the total capacitance of DAC array , to reduce the ADC power consumption and silicon area.

7. Bibliography

Baker, R. J. (2010). *Circuit Design, Layout and Simulation* (Third). Wiley.

Castañeda Villalpando, K. (2021). *Dynamic comparator, SR latch and bootstrap switch for 10 bits SAR ADC for biomedical applications* [Tesina para Especialidad de Diseño de Sistemas en Chip]. ITESO.

Deng, H., & Li, P. (2014, December 12). *A 8-bit 10MS/s asynchronous SAR ADC with resistor-capacitor array DAC*. 2014 International Conference on Anti-Counterfeiting, Security and Identification (ASID), Macao, China.

Ginsburg, B. P., & Chandrakasan, A. P. (2005, May). *An Energy-Efficient Charge Recycling Approach for a SAR Converter With Capacitive DAC*. 2005 IEEE International Symposium on Circuits and Systems, Kobe, Japan. https://doi.org/10.1109/ISCAS.2005.1464555

Gonzalez Ornelas, L. A. (2021). *Voltage Reference BandGap and Serial Communication Module for a 10-Bit Low Power SAR ADC for Biomedical Purposes* [Tesina para Especialidad de Diseño de Sistemas en Chip]. ITESO.

Reyes, D., Martins, T., Hernandez, H., & Van Noije, W. (2020, August). *A 10.75-ENOB 20 MS/s SAR ADC for an UWB Transceiver Applied in Breast Cancer Detection in 180 nm CMOS*. 2020 IEEE 63rd International Midwest Symposium on Circuits and Systems (MWSCAS. https://doi.org/10.1109/MWSCAS48704.2020.9184488

Savitha, M., & Venkat Siva Reddy, R. (2018, February 9). *14 -bit Low Power Successive Approximation ADC using Two Step Split Capacitive array DAC with multiplexer switching*. 2018 Second International Conference on Advances in Electronics, Computers and Communications (ICAECC), Bangalore, India. https://doi.org/10.1109/ICAECC.2018.8479477

Singh, V. P., Sharma, G. K., & Shukla, A. (2017). *Power Efficient SAR ADC Designed in 90 nm CMOS Technology*. 5. https://doi.org/10.1109/TEL-NET.2017.8343542

Weste, N. H. E., & Harris, D. M. (2011). *CMOS VLSI Design A Circuits and Systems Perspective* (Fourth Edition). Addison-Wesley.

Zhu, Z., & Liang, Y. (2015). A 0.6-V 38-nW 9.4-ENOB 20-kS/s SAR ADC in 0.18- CMOS for Medical Implant Devices. *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS*, *63*(9).

Floyd, T. L. (2014). Digital fundamentals: A systems approach. Pearson Education Limited.

Appendix A: Simulating using AMS Simulator

In this project, we needed to simulate using both analog and digital components. We use AMS Simulator for this since it is already included in Virtuoso. In this section we run the steps for creating testbench that uses both digital and analog components, and how to simulate them.

We use the following 3 Verilog files:

```
module my_or (
        input a,
        input b,
        output x) ;

        assign x = a | b;

endmodule

module my_and (
        input a,
        input b,
        output x) ;

        assign x = a & b;

endmodule


module logic_calc (
        input a,
        input b,
        output and_out,
        output or_out) ;

        my_or or_inst ( .a(a), .b(b), .x(or_out) );
        my_and and_inst ( .a(a), .b(b), .x(and_out) );

endmodule
```

1. The first step is to open Virtuoso by using the Querio shortcut in the desktop. The open Library Manager.

2. Select the Library where the modules are saved.

3. Go to File -> New -> Cell View and the next window opens:

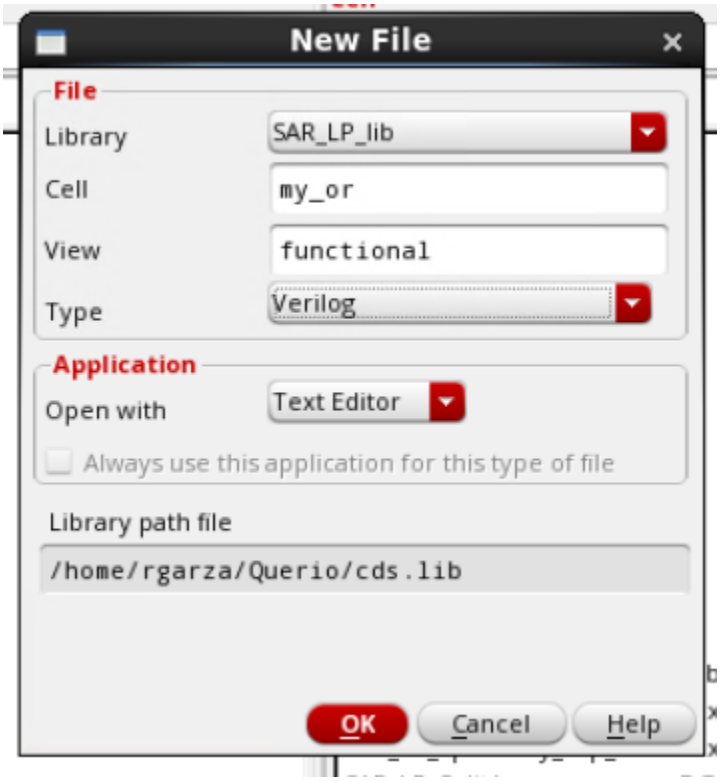


Figure 39: New File Window

In here we write the name of our cell, and make sure that the library is the correct one. In Type choose Verilog, and the View changes automatically. Click in OK. (If Virtuoso ask you to start a session just press yes to everything)

4. A text editor window opens. Write your Verilog code in there.

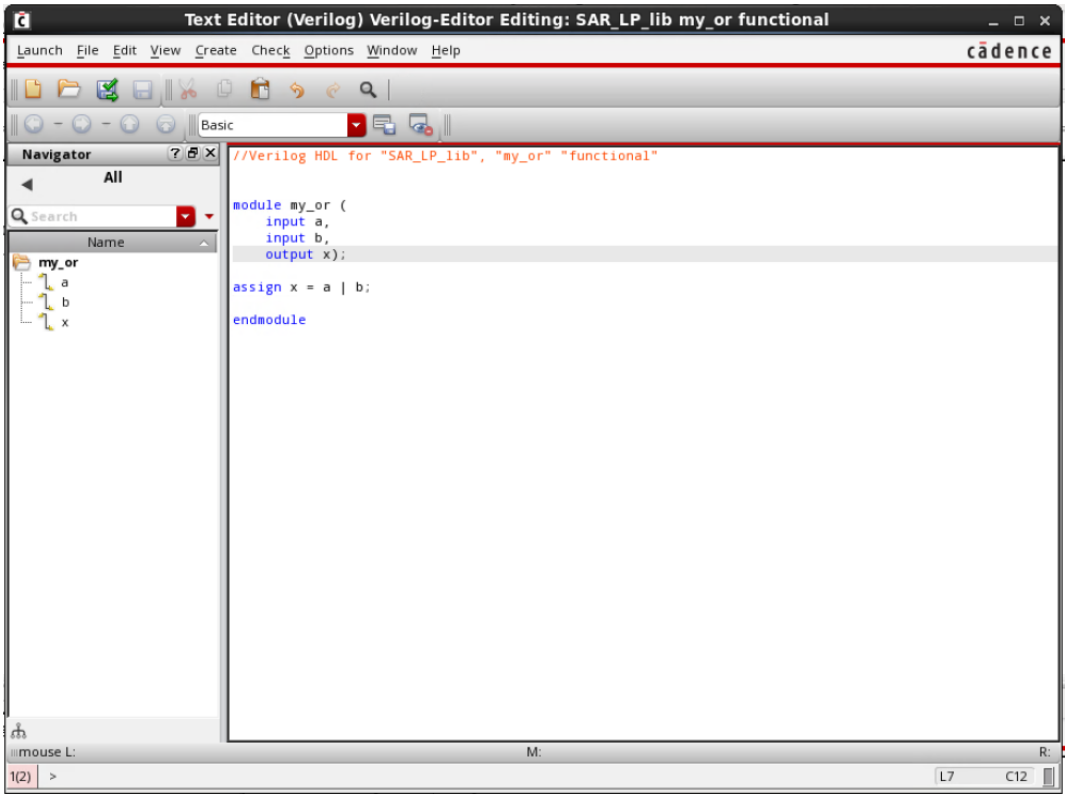


Figure 40: Text Editor for creating Verilog modules

After finishing writing your code, save using the Extract button (the floppy disk with a check mark and an arrow). If everything is fine, the Virtuoso asks you if you want to create a symbol for this file, say yes.

If there's an error, you can check it in View - > Parser Log File. You must correct it before advancing.

Repeat step 3 and 4 for the other files.

5. Next step is to do the testbench. Create a file as in step 3 but with the Type Schematic.

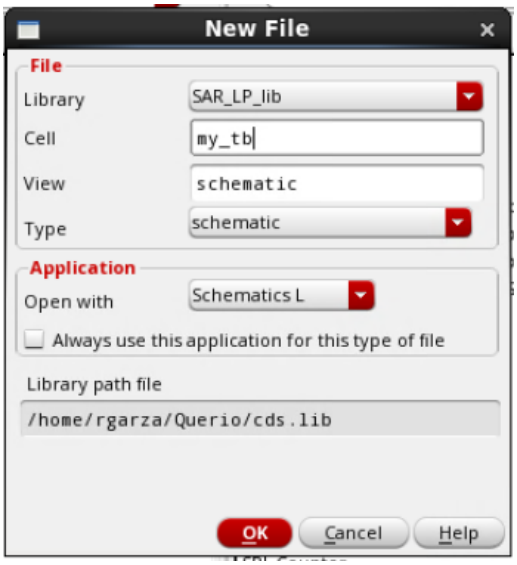


Figure 41. Creating schematic for testbench

6. A black canvas is opened. If you press "i" the window to add instances appears. In here you are able to find the modules that we just created. Since we have a top module (logic_calc), we only add this file and some voltages sources to act as inputs.

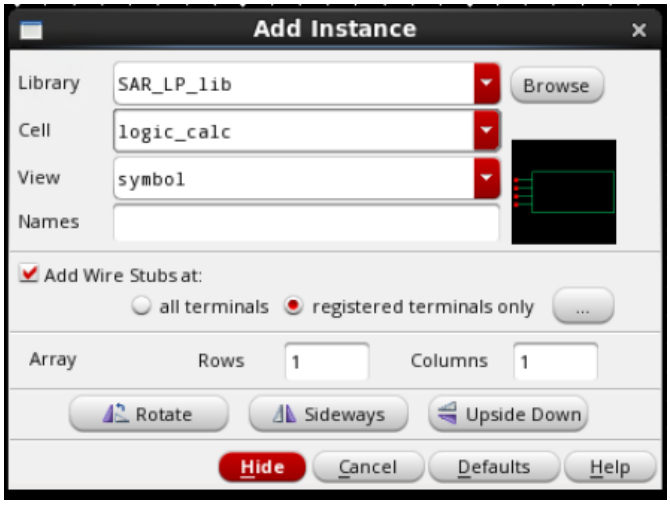


Figure 42: Window for Adding Instance

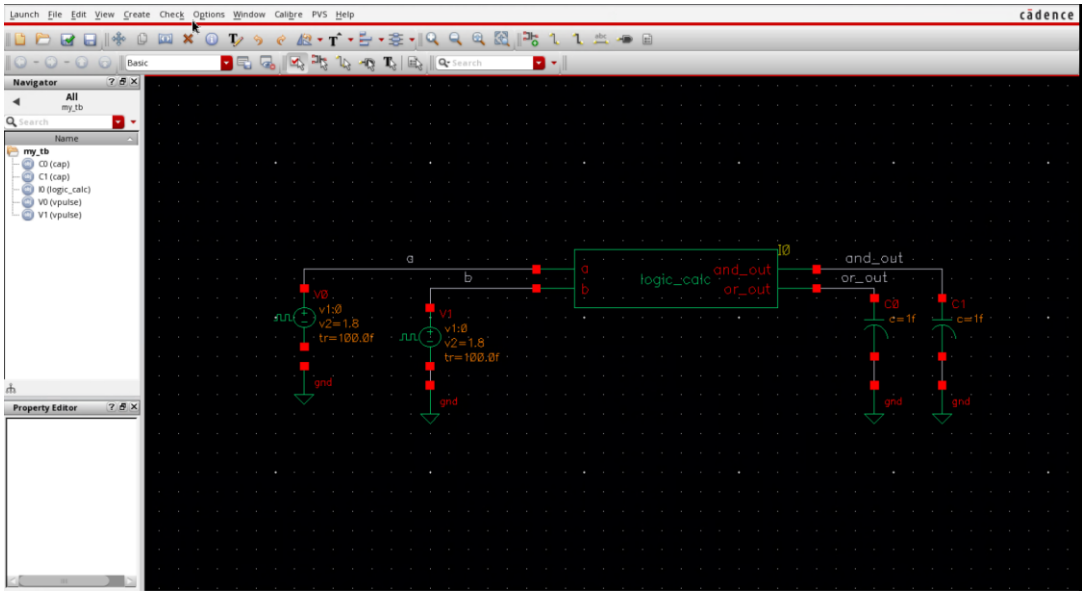


Figure 43: Test bench of logic_calc

For this example, we used 2 Vpulse voltage sources one working with a period of 20 uS and the other one with a period of 40 uS. We added some capacitors for the outputs. Remember to use labels. Save your design and close it.

It's important to know which technology you are using. In this case we are using 0.180 um of TSMC and this one works from a voltage range of 0 to 1.8V. This means that our digital modules understand as a logic 1 voltage values that are near 1.8 V. My voltage sources is mainly changing from 0 to 1.8V.

7. Next step is to create the config view. To do this, in the Library Manager Window, go to File -> New -> Cell View. Make sure to have selected the same cell that was used for the schematic. In type choose config, this also changes the view. Press okay

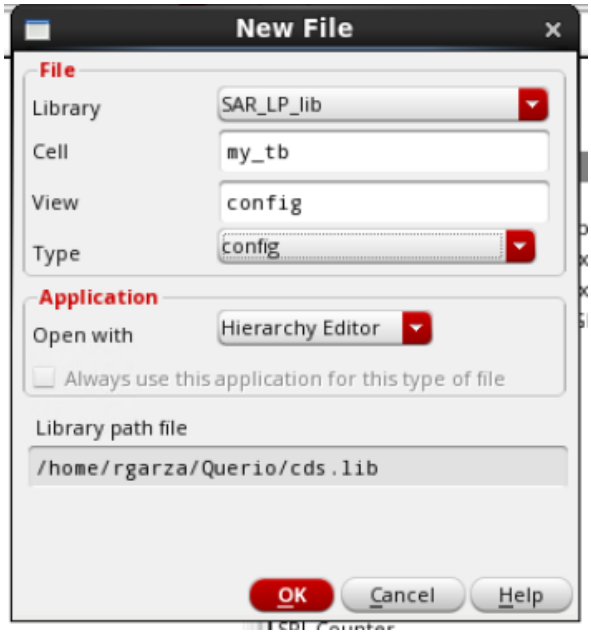


Figure 44: Creating config view for my_tb

8. This opens the New Configuration Window. We are going to use the template for AMS so click the button of Use Template that is below.

Figure 45: New Configuration Window

Choose AMS and press OK.

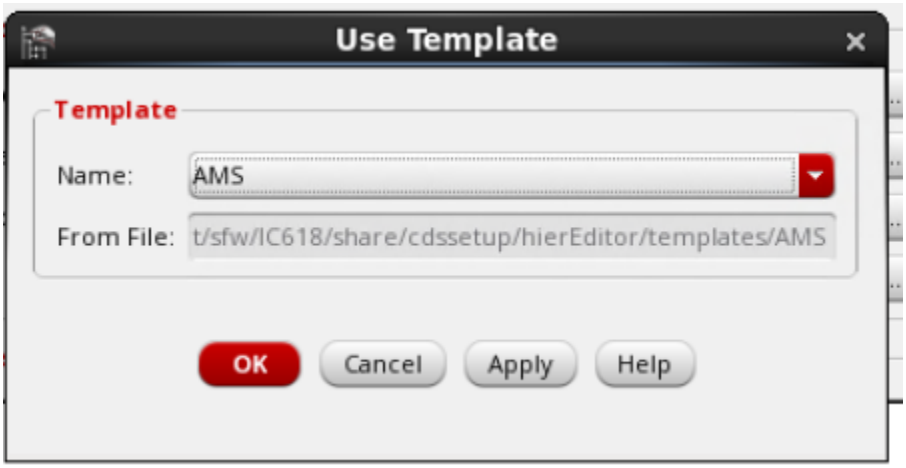


Figure 46: Template Window

In the New Configuration Window, you must change the view from myView to schematic. Also, in the part of Global Bindings, be sure to add the library where your files are located. Press Okay.

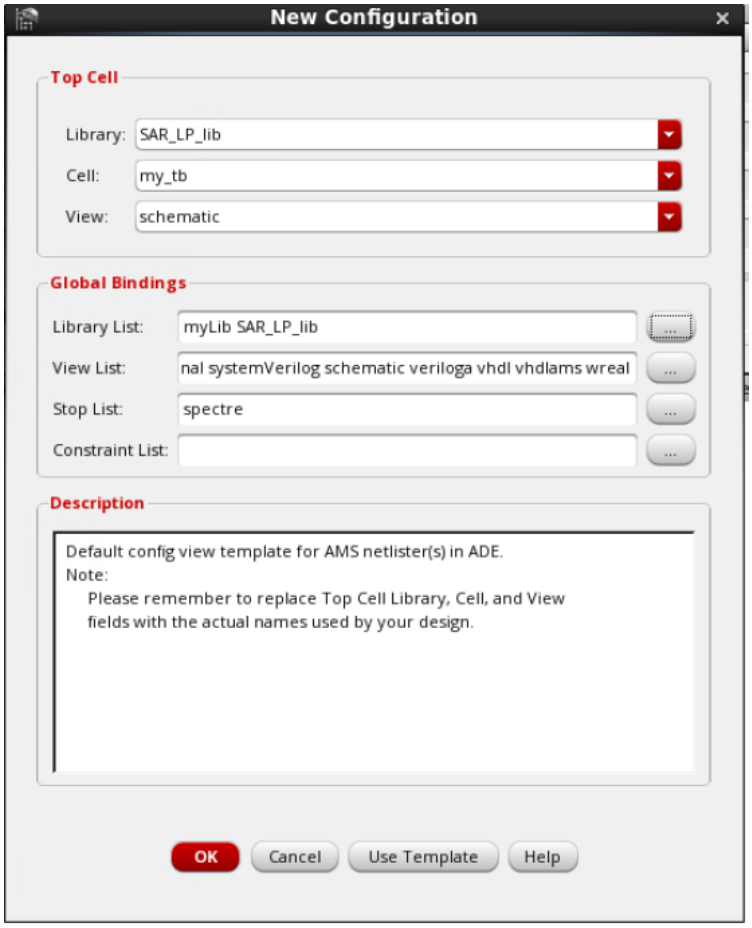


Figure 47: New Configuration Window filled up with template

9. A window like the next one opens, just be sure that all the cell bindings have a View Found. If it doesn't have it, it means that you are missing adding libraries.

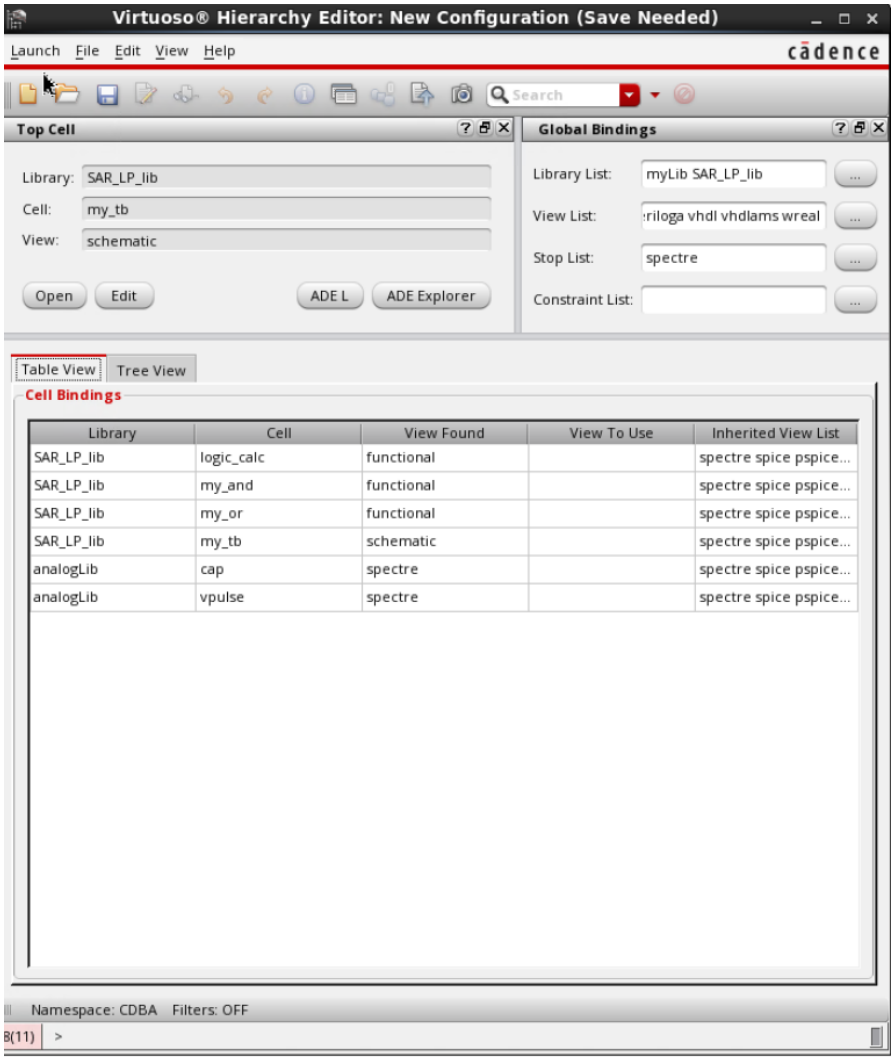


Figure 48: Configuration View

If everything is okay. Just save and close this file.

10. Open the config file again. Virtuoso asks you if you want to open the config file and the schematic. Select yes for both.

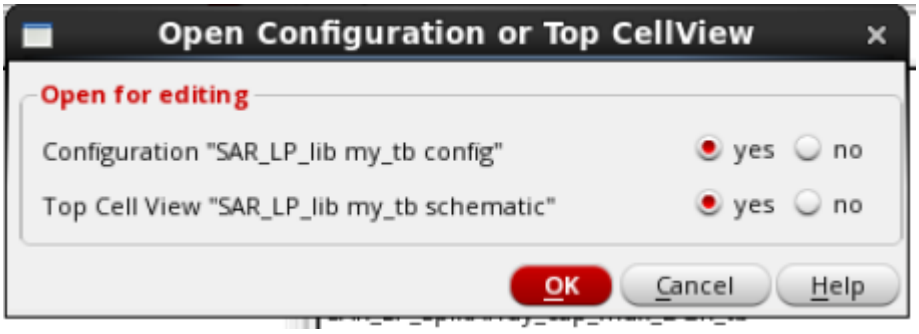


Figure 49: Open Configuration Windows

This opens the window of step 9 and the schematic. Every change that you applied is also applied to the schematic since it's the same file.

11. In the schematic go to Launch -> ADEL. If some pop ups appeared saying to start a new session press yes to everything. The ADE L Window appears next

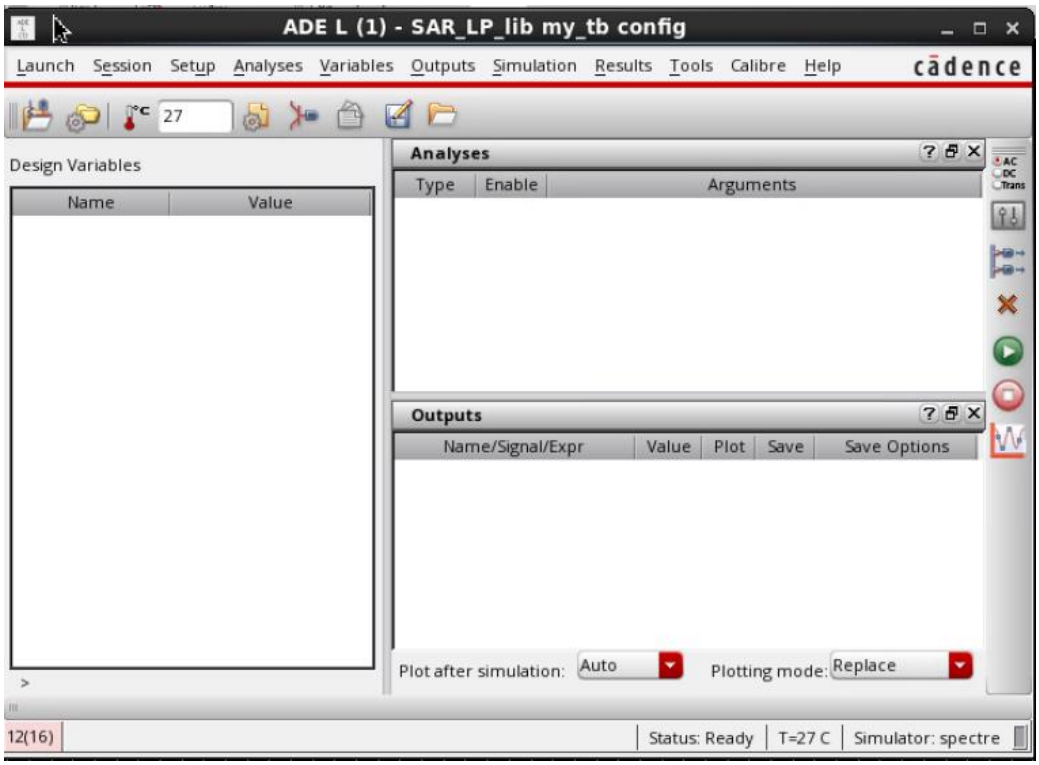


Figure 50: ADE L main window

The first thing to do in here is to change the simulator. Go to Setup -> Simulator/Directory/Host … In the window select AMS in the simulator option.

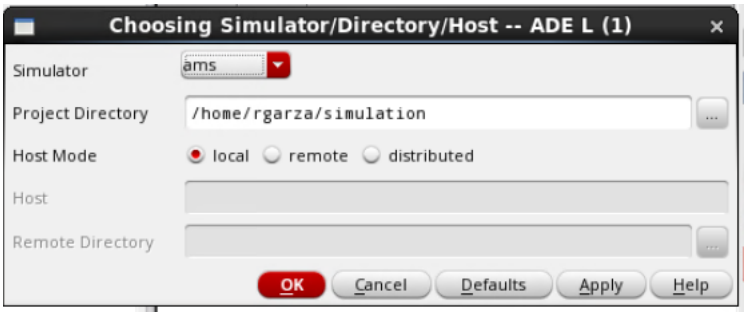


Figure 51: Selecting Simulator

12. From here on forward, the simulation is like every other. We have to choose an analysis, which output is going to be traced, add variables if necessary.

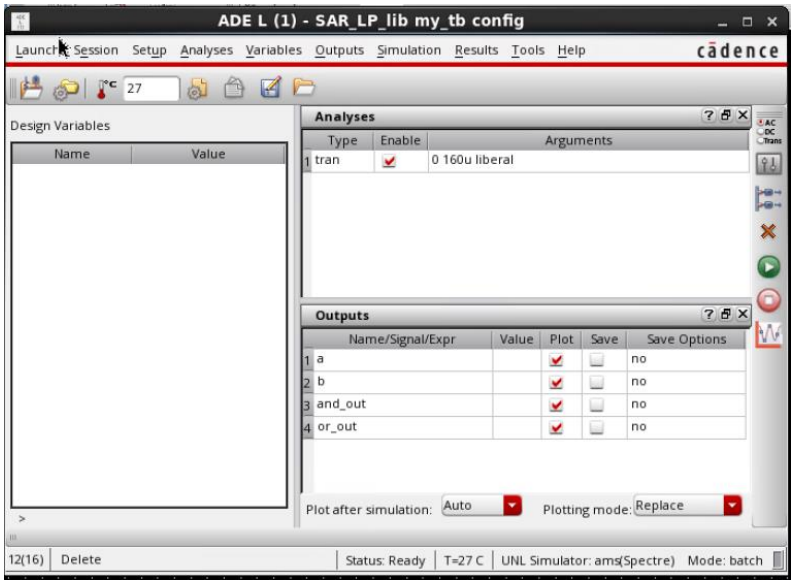


Figure 52: Simulation setup for this example

13. Analyze your results:

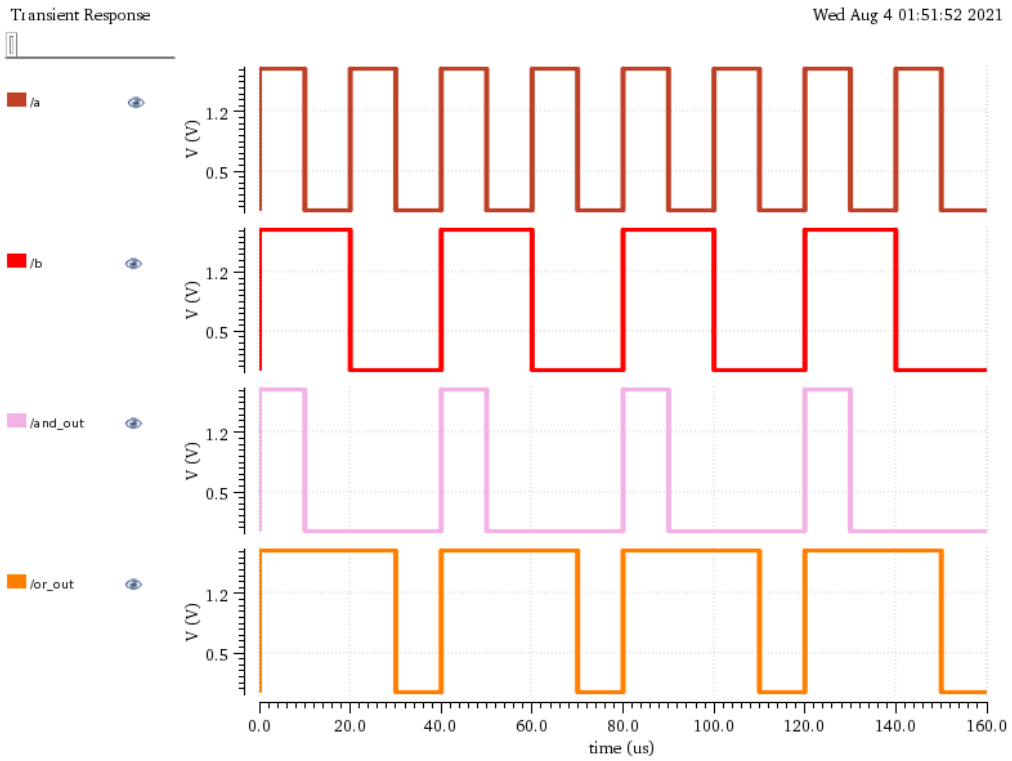


Figure 53: Resulting signals of simulation

Extra tip: For creating a nrst signal

Creating a reset signal is really important for all sequential modules. It's the first stage of most of them. To create a nrst, we used a "vpwl" voltage source from the library analogLib. We used 4 pairs of points with the following characteristics.

Figure 54: Instance used for creating a nrst signal

In the first stage, we describe the beginning stat of the signal. It starts a 1.8V. After 4.5 uS we want that the signal starts to change, we have to state that up until this second there is no changes to signal, it is still 1.8 V until this moment. After 0.5 uS (5 uS globally), the reset signal must reach 0. This is described in pair 3. And finally, the voltage must return to 1.8V and stay there forever, as described in the fourth pair.

Appendix B: SDC Constraint File

The SDC file that is used in this project was the following:

```
# ITESO University
# User Constraint File: ADC_LP_Typ.sdc
# This sdc file is considering the I/O blocks

set_time_unit -picoseconds
set_load_unit -femtofarads

# Clock definition
define_clock -name Main_Clk -period 4000 -rise 50 -fall 50 [clock_ports]

# slew attribute: Specifies the minimum rise, minimum fall, maximum rise, and
# maximum fall slew values, respectively, in picoseconds.
# The following sentence define the (min rise, min fall, max rise, max fall).
# In this example we are considerin 2% of the clock period.
set_attribute slew { 150 120 165 132 } Main_Clk

# network clock latency 1%
set_attribute clock_network_late_latency 60 Main_Clk
set_attribute clock_network_early_latency 50 Main_Clk

# source clock latency
set_attribute clock_source_late_latency 60 Main_Clk
set_attribute clock_source_early_latency 50 Main_Clk

# clock skew
set_attribute clock_setup_uncertainty {17 10} Main_Clk
set_attribute clock_hold_uncertainty  {14 8} Main_Clk

# Input delay definition: This is the delay coming from outside the design
# for this design it's defined at 5% the period of the clock.
external_delay -clock [find / -clock Main_Clk] -input 200 -name IDelay [find /des* -port
ports_in/*]

# Output delay definition: This is the delay going outside the design
# for this design it's defined at 5% the period of the clock.
external_delay -clock [find / -clock Main_Clk] -output 200 -name ODelay [find /des* -port
ports_out/*]

# Driving cell definition
## Available buffers: BUFFER_{PL}, where PL can be from C-O.
## Considering PL = J:

set_attribute external_driver [find [find / -libcell BUFFD12BWP7T] -libpin Z] { ports_in/* }
# BUFFD2P5BWP7T FilterChip

# Specifies the maximum acceptable transition time on the library pin.
# This attribute applies to input and output pins.
# We are considering around 10 times the clock slew rate.
# The following example specifies a maximum transition design rule limit
# for all nets in a design:
# The following parameter is set to 65% of clock cycle
set_attribute max_transition 2600 /designs/ADC_LP_bb
```

```
# max_capacitance specifies the maximum capacitance in femtofarads that an output pin can
drive.
# Note: This attribute has no value for input pins.
# The input capacitance of a I/O block PDUW0204CDG I pin  is 52.9 fF
# The PDUW0204CDG PAD-pin capacitance is 2,546 fF
# The Artix 7 input cap. is 8pF do no include package cap.
# # The typical capacitance of a digital osciloscope is 11 pF
set_attribute max_capacitance 6000 /designs/ADC_LP_bb

# external_pin_cap
# port attribute: Indicates the external capacitive load (in femtofarads) due to pins
# that are connected to this port.
# The typical capacitance of a digital osciloscope is 11 pF
# Considering a I/O block PDUW0204CDG PAD pin load Capacitance
set_attribute external_pin_cap 3000 /designs/ADC_LP_bb/ports_out/*

# Setting maximum value of fanout
set_attribute max_fanout 50 /designs/*

# lp_power_unit {nW | mW | pW | uW}}
# Default: nW
# Read-write root attribute. Specifies the power unit to be used when analyzing net power,
# cell internal power, or cell leakage power. The power units are case sensitive.
set_attribute lp_power_unit {uW}

# lp_power_optimization_weight
# Controls the weight factors to be used when optimizing
# leakage power and dynamic power simultaneously during global mapping, mapping, and
# incremental optimization. Specify a value between zero and one. Assuming the attribute is
# set to w, the RC-LP engine optimizes for total power. Total power is computed as follows:
# Total power = w x leakage_power + (1-w) x dynamic_power
set_attribute lp_power_optimization_weight 0.2 [current_design]

## To enable the recommended leakage power optimization flow,  use the root
## attribute  leakage_power_effort  set to low, medium or high-
## with an optional specification of max_leakage_power attribute for a specific power budget.
## Setting leakage_power_effort to 'none' will enable the backward compatible mode.
set_attribute leakage_power_effort medium
set_attribute max_leakage_power 200 [current_design]
       set_attribute max_dynamic_power 800 [current_design]
```