Design of a Fully Integrated CMOS Highly Temperature Stable Frequency Reference

Tsiolis Georgios

Master Program: Microelectronics
Delft University of Technology

Supervisors: Wouter A. Serdijn
Arie Van Staveren
Frank Kuijstermans

Location: National Semiconductor (Texas Instruments),
Delft Design Center

Period: summer 2010-autumn 2011

This essay contains all the results of the graduation project for the MSc program in Microelectronics that I followed at Delft University of Technology (August 2009-November 2011). The work presented here was conducted and completed in the Delft Design Center of the company National Semiconductor (already since October 2011 acquired by Texas Instruments) on a project of designing a frequency reference with low frequency drift over temperature deviations.

Tsiolis Georgios

November 2011

Contents

Abstract

A new fully integrated frequency reference has been designed, fabricated and measured. The goal of the design procedure has been high stability of the frequency over temperature variations. This should be achieved using a CMOS technology without MEMS and crystals, while being restricted by a very low power budget. The frequency is designed based on an RC time constant, where both R and C are integrated on chip. The novel topology is minimizing the sources of error that affect the frequency being produced (minimum offsets and delays that spread over temperature). According to the simulations, the fabricated oscillator is a competitive one among the RC-time-constant based ones, but the measurements showed that the tradeoff between performance and integration (accompanied with power minimization) is worse than its counterparts (MEMs based or LC based all-silicon choices), or its quartz crystal based ancestor. Ways to increase the quality even further are proposed.

Key words

Fully integrated, CMOS, all silicon, highly temperature stable, frequency reference, clock, RC time constant, no PLL, no MEMS, low power.

Acknowledgements

"Music is all about beautifying the ugly sounds, resolving the sound collisions", my music teacher and mentor **Foivos Papadopoulos** used to say. He was right, for a continuous stream of harmonic sounds is plain boring, it sounds as if it lacks passion. Thereafter, as I was growing up, I found out that the same holds for life. The only person you should completely agree with is the one that shows up in your mirror. Conversation promotes disputes and inevitable opinion-collisions and conversation is the only way to resolve them. Resolving them is the beauty of life and human interaction. Hence, life is art, isn't it?

What about studying/teaching then? Things get complicated in this field. "Constructive criticism" is very close to "destructive criticism" and there are very few professors that can manage to encourage their students to study through constructive criticism. The majority chooses either the "fun" approach - never scold, only tender push, only with a smile and by being nice, or the ignorant approach - "your duties are described in this piece of paper, perform them or not, it's up to you". Out of those who choose the criticism way, most mess it up and end up being considered impolite or simply annoying.

I can safely conclude that the Dutch way didn't suit me - at first. Dutch lifestyle is based upon harmonious situations. Divide and conquer. "This is your life in the living room, so I don't care if you have no curtains. Live your life as long as it doesn't bother me, and I won't care what you do". This is reflected on professors as well. A professor will ask you to do what you should, always with a smile, but will not pressure you- pressure would be intrusion in your life, and that's not Dutch at all. The tender approach, which means never making you feel bad by showing you your flaws (which is by the way the basis of criticism), makes me feel bored and less productive.

In other words, I wasn't used to setting my environment in a free space, but adjusting it according to the limitations proposed/imposed by others. May they have been colleagues with whom we shared the journey of a yearly thesis project working chair by chair (**Elena Papadomanolaki**), or friends editing my stories, my babies, with comments that may hurt sometimes, but they are so constructive and educational (**Marina Simini**), or friends teaching me good manners - that's life in the adult world (**Maria Tzedaki**). I usually didn't have to criticize myself. I received the correct amount of criticism from around me and was resetting myself when necessary.

So did I receive a "lesser" lesson, overall, here in the Netherlands? Not at all. I learned how to operate in such a new environment the hard way. New style (Dutch) new place (Work) new age approach (Colleagues were on average in the middle between mine and my dad's age). Considering this recent past, **Arie van Staveren** deserves to be the first to be acknowledged. He is the man in charge at the company where I performed my research, and it came as a surprise to me how someone as knowledgeable as he is can be so accessible, fun even, and all his positive traits make him very special. **Frank Kuijstermans** and everyone else in the Delft Design Center of National Semiconductor (now Texas Instruments) who helped me along the way one way or the other are people I am thankful to. Last but not least, my connection to the university, **Wouter Serdijn**, whose wise and continuous guidance made things roll with this project.

I was lucky enough to be given great opportunities and be surrounded by charismatic and approachable people. I was foolish enough to make use of these people's kindness less than I could-

and sometimes less than I was asked to (!), and foolish enough to exploit my chances far from the maximum. But I am happy I understood my mistakes after literally months of struggling, and I'm ready for an even more constructive next step - wherever in the world that may be, even in a place with "Dutch mentality", the Netherlands obviously included.

But all work and no play, makes Jack a dull boy. Friends with whom I spent most of my free time the last years - travelling to Stockholm, Rome, New York, Washington, Antwerp, Brussels, and having endless over-beer-and-food discussions all over the Netherlands, especially in Rotterdam, Delft and Den Haag, will be held dear in my heart forever.

In conclusion, I want to thank everyone who added diversity to my life, and made it interesting, and I hope I gave something back in return. I want to thank those who helped me become a better person, and a better scientist. Those who created the collisions that made my life interesting by trying to smooth them. Some still need fixing, but that's upon the years to come.
Ίδωμεν.

Γεώργιος Τσιώλης
Georgios Tsiolis
1st of November, 2011

PS Family. Family was a skype/voip discount part of my life the last 2 years, as well as my friends back in Greece - and the USA (**Panos Papadatos**). It seems like their contribution was minor, but that's simply because when someone has got your back, you often take them for granted and ignore (forget?) them. I won't. I will tell you how much you all meant to me by keeping me rooted somewhere. There. Thank you all.

Chapter 1
Introduction

The goal of this project has been to design a frequency reference with the following three important characteristics:

- Very good frequency stability over varying temperatures (100ppm stability for a temperature deviation of more than 100oC)
- Very low current consumption (less than 100μA).
- No use of quartz crystal

The design should be all silicon, in a CMOS technology in order to be fabricated easily on the same wafer with the rest of the chips needed for mobile phones, using a 1.8 V supply. The reason for the “no quartz crystal” specification is the general trend of trying to get rid of all non integrated components in electronic devices. Unfortunately nowadays, the most stable frequency references are based on crystals, so an external bulky off-chip component is needed. The reason for the low power consumption is the extension of battery life, and the high stability of frequency over changing temperatures is imposed by the demand for stable and robust driving of the digital circuitry in order to keep up correct operation of our device in all environments possible.

The essay is structured as follows. First all the strong points and weaknesses of the different existing approaches to such an all-silicon oscillator (MEMS based, RC time constant based or LC oscillators) will be discussed and one will be chosen to be implemented (Chapter 2).

After this choice, all steps of the design were met: high level design (system level in Chapter 3), implementation of the several parts (Chapter 4), final decisions and layout of the whole design (Chapter 5), measuring the chips after they came back from the factory, along with a comparison between measurements and simulation results (Chapter 6). All this procedure lasted 14 months, starting August 2010 and finishing October 2011. Conclusions for this design procedure and further improvements are analyzed in the end (Chapter 7).

Chapter 2
Specifications requested and what is already available

2.1 Journey through the history of clock references

The history of electronic oscillators (clock references) started 60 years ago. The basic steps in this journey were the development of crystal oscillators "XO", by making use of the piezoelectric effect that couples the mechanical properties of the crystal with an electrical circuit [2], the introduction of transistors instead of vacuum tubs for the surrounding circuitry, the achievement of developing temperature compensated crystal oscillators "TCXO" (where the compensation was achieved over the years with many different ways), and nowadays the attempts for all silicon clocks, without the use of crystals, but using MEMS, RC time constant, or just the LC oscillation principle. Size has been continuously reducing several orders of magnitude, while accuracy –surprisingly- not much, only a couple orders of magnitude, with the oldest manufactured and perfected choice (quartz crystal based oscillator) still beating its rising opponents.

In the 50's already, very accurate ovenized crystal frequency references were created, using vacuum tubes for the surrounding circuitry and one oven for keeping the temperature of the crystal stable. Also, when possible, the circuitry was also added within the oven to keep same conditions for everything contributing to the frequency.

Further improvement in the stability of the generated frequency was made when it was noticed how important the orientation in which the crystal was being cut was. It was found already really early (1937, Bell Labs, [1]) that a GT-cut crystal had a temperature coefficient that remained practically zero over a large temperature range. So using a GT quartz crystal, double oven and vacuum tubes, a crystal resonator was developed in 1951 of accuracy in the level of 10^{-3}[1].

After transistors became widely used (60's), the accuracy increased dramatically also because now the whole circuitry could be easily placed within the oven, thanks to the very small size of transistors. So already in the early 60's a 5MHz clock with accuracy of $5*10^{-10}$ burning 25Watts and size of approximately $10000cm^3$ was developed[1]. Another improvement (during the 70's) was the introduction of power transistors as heating elements instead of resistance elements. That reduced dramatically the power being consumed. At that time mainly SC-cut crystals were being used and over wide temperature range, stability in the order of 10^{-9} was achieved [1].

A different choice than using crystals in ovens, was to create temperature compensated crystal oscillators. In this case, varactor diodes are being used to transform the temperature behavior of the crystal into voltage, and then a thermistor network to match it and cancel it out, at least in temperature blocks, each block considered having linear behavior. Usually there were 3 different thermistor networks used, one for low, one for medium and one for high temperatures. Results of this method were to achieve accuracy of $5*10^{-7}$ over a huge temperature range [1].

This temperature compensation has changed forms within the last decades, of course going towards digital compensation, like a memory pre-programmed with the required compensation voltages, or a microcomputer implementing this digital compensation, or using dual excitation for more accurate control over the measurements, reaching 10^{-8} accuracy, or even using external (off chip) compensation.

Nowadays the temperature controlled crystal oscillators are being produced in the accuracy level of 10^{-8}[1]. Ultra stable ones go up to 10^{-10} accuracy [3].

Last novelty, is every attempt to get rid of the crystal needed for the clock, and this has been achieved in several different ways, resulting in many different Silicon Oscillators. These all silicon oscillators conquer the crystal ones in terms of robustness to vibration, humidity, resistance to EMI effects and shock, not to mention the easier (on-wafer) production, which gains in size, cost and time. A photo comparing the two choices (an all silicon oscillator with one the uses a crystal) can be seen in appendix, A.1. Using a MEMS resonator, clocks were designed that had a frequency stability over temperature of 10^{-5} [4], but the fact that MEMS are not CMOS compatible doesn't make them the first and most beloved choice. Using temperature compensated LC oscillators and then dividing that frequency to user-programmable range we see frequency stability accurate in the scale of 10^{-5} [5]. Finally using internal RC constants the accuracy achieved over temperature remains in the range of few percent (10^{-2}).

Applications where clock references (oscillators) are necessary are all digital circuitry that needs synchronizing, internet Ethernet connections, smart cards, peripheral communication like USB connection ports, flash drives, card readers, microcontroller references, microprocessor clock applications, display drivers, advanced battery chargers and many more.

2.2 Numerical achievements, by university publications.

Some of the oscillators published in papers originating from university research are presented in Matrix 2.1.

Matrix 2.1: Oscillators published by universities, one Crystal based and many all-Si alternatives.

Type	Freq (Hz)	Temp stability over 100oC(-20oC to 80oC approx)	Supply (V)	Current (A)	Comments	Reference
Crystal (ovenized)	-	$\pm 2*10^{-11}$	-	-		[3]
Fully integrated LC oscillator & divider	50M	$\pm 1*10^{-6}$	3.3	8m	The LC oscillator is accompanied with temp compensation	[7]
Bypassed RC-time based [*1]	100k-100M	$0.5*10^{-4}$	3.3	-		[9]
RC-time based	-	$\pm 1*10^{-3}$	3.3	-	Uses current reference, temp compensated up to second order.	[10]
Bypassed RC-time based [*1]	14M	$\pm 2*10^{-3}$	1.8	25u		[19]
Bypassed RC-time based [*1]	250k-1M	$\pm 5*10^{-3}$	3.3	400u	Requires an ideal external resistor	[8]
Ring oscillator	7M	$\pm 5*10^{-3}$	-	-		[18]
Ring oscillator	224M-974M	$8.6*10^{-3}$	-	8.3m		[12]
CMOS, RC based	16M	$\pm 2*10^{-2}$	1.8	-		[11]

[*1] Bypassed RC time based oscillator implies (as will be analyzed in Chapter 7) an oscillator whose frequency is produced based on an RC time constant but this core is bypassed by a "master" sub-

network that is responsible for the accuracy of the produced frequency by controlling something in the RC network.

2.3 Numerical achievements, by company products

Some of the products (oscillators) by various companies are presented in Matrix 2.2.

Matrix 2.2: Company commercial oscillators, one crystal and many all-silicon alternatives.

Product Name	Type	Freq (Hz)	Temp stability over 100oC(-20oC to 80oC approx)	Supply (V)	Current (A)	Comments
DS32B35 (Maxim)	Crystal (temp compensated)	32k	$\pm 6*10^{-6}$	3.3	260u	
Si500S (Silicon Labs)	All Si	900k-200M	$\pm 2*10^{-5}$	1.8-3.3	10m	
DSC1018 (Discera)	MEMS	1M-150M	$\pm 5*10^{-5}$	1.8	3m	
3C02 (IDT)	All-CMOS	6M-133M	$\pm 10^{-4}$	1.8-3.3	2m	
MAX7377 (MAXIM)	All Si	600k-10M	$\pm 5*10^{-3}$ (max $3*10^{-2}$)	2.7-5.5	3m	
STCLx32k (STMicroelectronics)	All Si	32k	$\pm 1.2*10^{-2}$	1.8	-	Obsolete product
LTC6909 (Linear Tech)	Bypassed RC-time constant	12k-6.6M	$\pm 1*10^{-2}$	2.7-5.5	400u	

All products below the thick line are specifically stating that they are Quartz free, PLL free, and all but the MEMS based one from Discera are MEMS free.

2.4 Placement of the requirements, expectations by our choices.

The project requested the design of a semiconductor-based oscillator, using a CMOS technology, with important characteristics ("critical parameters") to have temperature stability between -30oC and 85oC in the order of ±100ppm, which is $\pm 10^{-4}$, while burning less than 100uA. The rest of the non-critical characteristics are the load that has to be able to be driven by the oscillator (up to 12pF), the duty cycle (min-max 0.4-0.6xPeriod), the Jitter (120nS), the phase noise (-125dBc/Hz) and the power consumption on inactive mode (less than 2uA).

The first and very important choice that has to be made is how to approach our "all silicon, no quartz" oscillator, with which of the available choices:

- LC oscillators are producing results within our specifications, and they are basing their good behavior to the high quality (Q) LC-tank, requiring the on wafer design of an inductor.
- MEMS are facing more expensive fabrication process, but in return they manage to give very good results, within our specifications, based on the sealing that can be performed on their high quality factor mechanical elements on which the oscillation frequency is based.

- RC-time constant based in general are struggling to get below 1% (10^{-2}) accuracy over temperature.

A decision was made to implement the RC time constant based oscillator. For this type of oscillators it appears to be a difficult task to achieve accuracy sufficient for most applications [7], something that originates from a tradeoff – we are paying the price for less good performance in order to make use of the benefits this choice offers. The ease with which the frequency of this type of oscillators can be adjusted, the absence of inductors, the absence of PLL's and frequency dividers/multipliers, since the frequency is generated directly are some of them. Also competitive advantages are the facts that production of this oscillator type is cheap (standard CMOS technology is sufficient) compared not only to MEMS choice for example, but to the cost for quartz crystal oscillator stable over wide temperature range which remains relatively high [8] as well. As mentioned before, all the all-silicon oscillators are insensitive to vibration, shocks and EMI.

Our expectations are set in the following way: we will target to keep consumption within the limits and get the best possible result for temperature stability, minimizing or even cancelling any unwanted contribution(s).

Chapter 3
Oscillator design
The basic idea-topology and high level design

3.1 Introduction

The primary goal of every oscillator design is to create a stable (against all kinds of uncertainties) frequency. In this Chapter we will analyze how we built our oscillator, using a top down approach. That means that in this Chapter we are going to design with ideal blocks in system level, while in the next Chapter (Chapter 4) we will realize these ideal models using electronic components. This design methodology is well known as the one creates the best electronic circuits (this design procedure implies that Orthogonality, Simplicity and Hierarchy are being applied as described in [20]).

We have to state here that we are building an electronics oscillator. Even though the basic ideas that will be presented are not confined in electronics domain, but they could be implemented in other domains as well, and build –for example- a mechanical oscillator, we will introduce the basic components which are more suitable and easily realizable in electronics. In this Chapter we have to keep in mind the components electronic feasibility and when describing their function mathematically, will do so using electronic values. For example a threshold, is going to be implemented as a voltage threshold, so it will be named V_{th} when using mathematical formulas.

In section 3.2 we are going to address the general problem of designing an oscillator, approaching the design from the signals point of view. Then in 3.3 we are going to present the possible choices for the topology of the oscillator, starting from the simplest possible one and trying to keep it as simple as possible, in 3.4 we will perform a crash test of all these simple implementations presented in 3.3, discussing about their advantages and disadvantages, and finally in 3.5 we will justify our final choice and try to cancel out any non-idealities this choice has already in this design level. Section 3.6 contains results from cadence simulations.

3.2 Creating a periodical signal

3.2.1 Components Needed

Our goal is to create a signal that repeats itself indefinitely. In order to get there, we firstly need to think about the components that are needed order to create this repetitive signal.

The simplest oscillator type (a one pole oscillator) can only build a relaxation oscillator and not a harmonic one [13]. Having this as a starting point, we suppose the output of our oscillator to be a square-wave-like signal. Therefore we have to find a way to time correctly the rise up and the fall down edges of the signal. So we need a *timing reference*. Time cannot be measured directly, but it has to be measured indirectly through a quantity that varies in time. For us that should be a quantity that makes sense in electronics domain. In this domain we can measure and create/change voltages, currents and components values via dynamic elements (L's C's and delay lines).

One oscillator that uses a timing reference and no amplitude modulation could be a cascade connection of a number of inverters. The timing reference would be the internal delay of the inverters. The first one would change its output after τ_{delay}, and then the next one would read this as input signal, but also get delayed τ_{delay} before changing its output etc. This is called a ring oscillator, and the problem with this approach is how little control we have on that delay (which is temperature, supply, load and process dependent and relates to threshold, slew rate and several other aspects). The idea of a ring oscillator was quickly abandoned, so we decide that the timing reference needs to be measured in some kind of *amplitude modulation*.

A simple idea would be to have a rising slope, as seen in Figure 3.1. The system would compare the value of the rising slope to a threshold V_{th}, and every time it would rise above the value of the threshold (or in Figure 3.1 above the value of [(previous threshold)+V_{th}]), clock output would change (alternatively rising and falling edges of a square wave signal). So we count another 2 needs: the existence of *comparators*, and the creation of *threshold levels* to which the slope signal is compared.

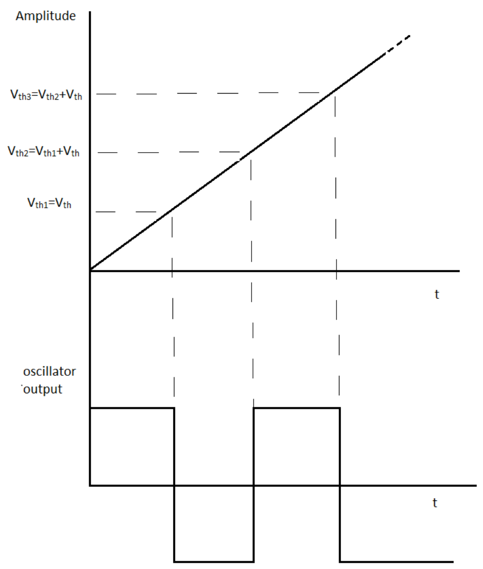


Figure 3.1: Timing information being derived from comparing a slope value to a threshold

We are talking about voltage thresholds, but the same could also be made for current thresholds when using current comparators. As for the third possible choice of component values, that would be very hard to be implemented – for example a capacitance value changing and being compared to a “unit” capacitance.

Of course this slope needs to be created somehow. Implicitly another need is presented, the need to create a varying signal. The simplest way to get a constant-slope signal is to integrate a constant value. So we need an *integrator*, and the creation of a *constant value to be integrated*.

One problem that we didn’t think of yet is the fact that our electronic circuit is going to have limited supplies, so voltages are going to be restricted between two values. The ideal Figure 3.1 is not implementable in electronics. A simple solution could be to integrate a positive value in order to reach a threshold, and then to change sign of integration, and reach another threshold, and so on (Figure 3.2). Each time the system reaches a threshold means a rising or falling slope in the output. This way we keep our signal restricted between two values, and this makes more sense for an electronic implementation of the periodic signal. Still, this thought does not add new "needs" to our design, apart from the need for two thresholds instead of one, and two signs for the signal to be integrated.

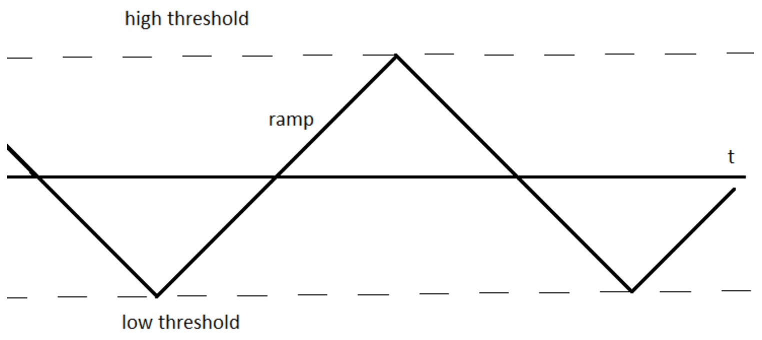


Figure 3.2: Integrator's output showing integration of different signed values

The final implicit need is the need to remember where the system is and what needs to be done next. For example in Figure 3.1 the previous threshold needs to be remembered in order to add the same amount once again and find the next threshold and so on. In Figure 3.2 what needs to be remembered is if the "coming soon" change of the comparator output should cause an integration of the positive or the negative value. So we have presented the need for the memory, that we will call *state memory*.

Of course we shouldn't forget that already the integrator is an analog memory, which memorizes the integrated value (a voltage value representing the integral for our implementation). So we have to keep in mind that our oscillator- the simplest kind we present here- needs two memory functions in order to operate.

To sum up, in order to build the system that will create the simplest (possible) periodic signal (a first order oscillator), we need one timing reference, created by:

- One varying signal, by means of an *integrator* and *value(s) to be integrated*
- *Comparator(s)* and *reference signal(s)* to which the varying signal will be compared to
- *A second memory(state memory)* that will remember the state the system is in

Finally, as far as the topology is concerned, we have to implement feedback in our system, since the output of the state memory (somehow "in the end") needs to choose the sign of integration (somehow "in the beginning"). But this will be implemented in the coming section 3.3.

3.2.2 Signal choices

So far we discussed only about signals with a controlled slope (like the output of the integrator) and constant signals (like the input of the integrator). We could also use signals like those in Figure 3.3b and c: we will call them *surges*. Surge will be a predefined increase or decrease of the signal of the integrator. By using a surge we can make an oscillator based on only one threshold, but still restricted between two values. The idea is that integration starts, the slope crosses the threshold, the surge value is being added and that brings the system to the previous situation and then integration starts again (like in Figure 3.3.b for positive and c for negative surge value). Here the second threshold is implicit in the predefined value of the surge.

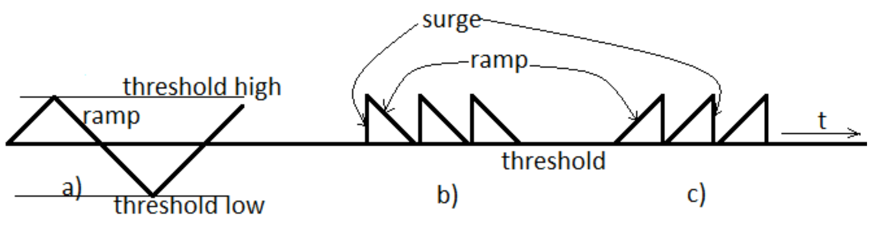


Figure 3.3: a) signals with slope b&c) using a surge signal

3.3 Different topologies

The discussion so far in section 3.2 on which components are needed to create a periodical signal, sums up in the following topology:

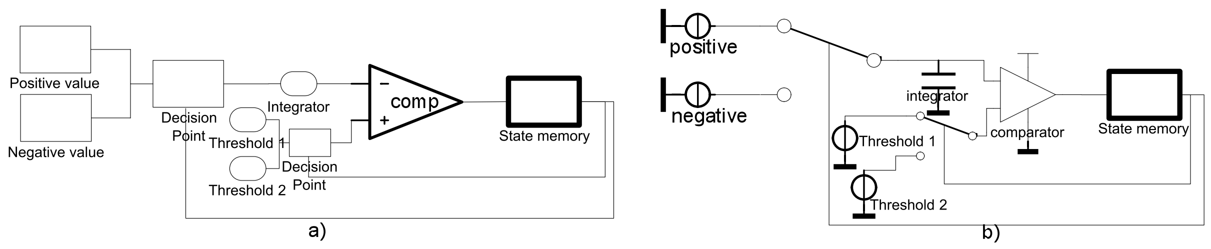


Figure 3.4: The basic oscillator a)in a block level diagram b)in an electronics domain block diagram

We notice the existence of decision points (on 3.4.b they are implemented by switches) that control which value (positive or negative value) has to be integrated and which threshold is being compared. After the slope (integrated value) is compared to the threshold by means of a comparator, the output of that comparator commands the state memory to change the position of the switches and therefore both the sign of value to be integrated and the threshold(s), according to the state saved in the memory.

From this simple idea, several different implementations can be derived. Of course we should keep in mind that innumerous different approaches to the topology of the oscillator could be created by adding more timing references. Here we only have one timing reference (time for the slope to get from one threshold to the other) and two memories (our integrator and a state memory). If we add more timing references or more memories the design becomes more flexible, but we will try to keep it as simple and efficient as possible.

3.3.1 Two thresholds oscillator

If we want to implement our system with the presence of two thresholds (the high and the low) then we can define two different topologies: one with one comparator and a switch in front of the threshold-to-be-compared, choosing which threshold is in use every moment (identical to the general one from Figure 3.4), and one with two comparators, each one continuously hooked up to one of the thresholds. These can be seen respectively in 3.5 a and b.

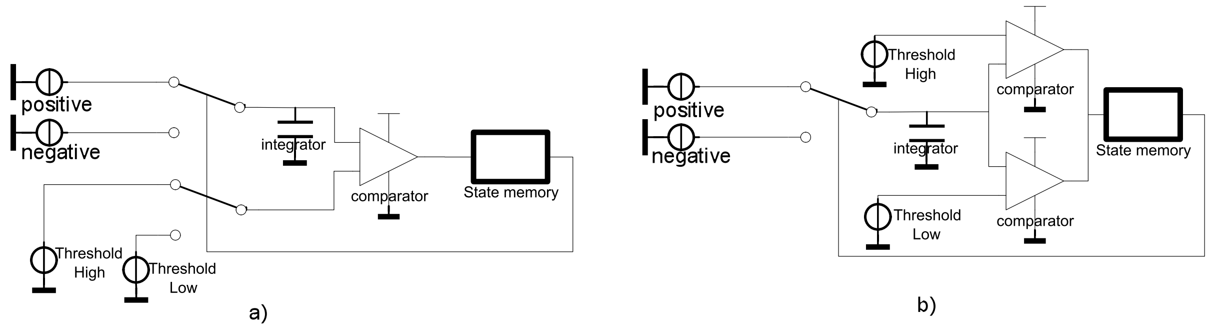


Figure 3.5: Two thresholds oscillator a) with one comparator and a switch to choose the threshold b)with two comparators.

As can be seen in Figure 3.5a, the switch to choose which threshold is going to be compared is controlled by the state memory that controls also which value is going to be integrated.

3.3.2 One threshold oscillator

If we want to implement a single-threshold oscillator, then as discussed before there should be somewhere hidden the control over the second "threshold". One way to implement this is by the use of a surge quantity. In that case, when the slope crosses the threshold level, the surge value is being added to the integrator and then integration starts again. In this implementation we can even use ground as the threshold, and therefore "create" no threshold of our own. The general topology can be seen in Figure 3.6.

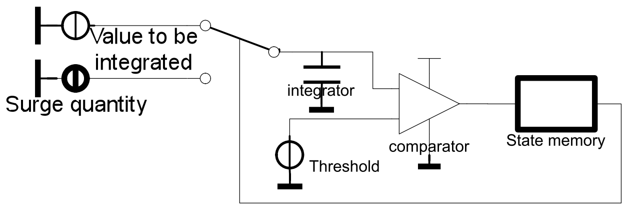


Figure 3.6: One threshold oscillator using a surge

We notice that this way only one value is being integrated (either positive or negative) and the change in the opposite direction is taken care of by the surge. Also the state memory once again is taking care of both switches: one to integrate the value or not, and another to add the surge quantity or stop it - actually they could be merged in one switch, and either the value to be integrated or the surge quantity would be connected to the integrator, like in Figure 3.6. In that case the memory must have some kind of connection to the surge quantity, so it knows when the surge quantity is added entirely to the integrator's value and then turn back the switch to the value to be integrated without change in the comparator's output (the state memory should have a way of knowing when the surge has been properly applied).

Another way to use only one threshold is by making a copy of our circuit, and creating a quadrature oscillator (which is no longer a first order oscillator, since we have two timing constants). Here a loop is formed. Crossing the threshold is not the end of integration. It just triggers the change in sign of integration of the following "copy" of the circuit (Figure 3.7). The upper and lower thresholds are not anymore present, and they are not needed. The system increases in complexity, in power being dissipated and in area needed on chip, but this topology presents some benefits that will be discussed in the next section 3.4.

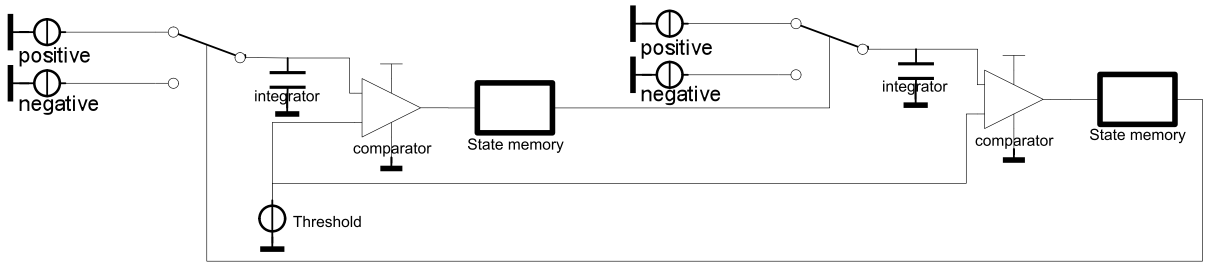


Figure 3.7: One threshold oscillator using a copy of the whole circuit

3.4 Pros and cons of the various implementations

In this section we are going to discuss the problems that arise from the various implementations of the oscillator. In 3.4.1 we are going to address all problems that occur in real life implementations while in 3.4.2 we are going to compare the mathematical expressions that govern the above mentioned topologies, and compare them.

3.4.1 Problems that occur in real life signals

As in any electronic circuit, the issues that decrease the quality of the performed operation in our oscillator and their effect should be minimized, are:

- inaccuracies, due to offsets and delays
- noise
- distortion
- process mismatch that implies frequency may have to be tuned to the correct value
- temperature stability (especially in our design)

The latter four effects will be dealt with in the next Chapters, when we'll start talking about the actual electronic implementation of the circuit, and things like noise and distortion will make sense. Here, since we are doing an abstract analysis of the problem, and we use ideal blocks (i.e. mathematical expressions) we will only introduce inaccuracies (delays in time and offsets in values) and compare the implementations in subsection 3.4.2 based on that. What hides behind delays is "time domain errors", and what hides behind offsets is "amplitude domain errors".

As far as the signals are concerned, an important statement must be made about the difference between ideal and real signals and transitions. In ideal signals, if we look at Figure 3.8.a (where for simplicity only the top of the integrator's triangular output is depicted) it's easy to perform many changes simultaneously. When the slope reaches the high threshold then we suppose that two things happen at the same time: a change in integration sign and a change in threshold level. The change in integration sign can be seen as a negative feedback action, and the change in threshold as a positive feedback action since it's the action that makes the whole system regenerative.

If now we move a step closer to real life signals, like in Figure 3.8.b, we see that since nothing happens instantly in real life, the slope signal will rise for a negligible (?) amount of time higher than the threshold in order to change the output of the comparator. Then the sign of integration changes along with the threshold level. If the threshold is not changed fast, but gets delayed more than the signal needs to cross again the high threshold, integrator's output can end up in a deadlock situation, like in 3.8.c, where the threshold doesn't change in time and system starts oscillating around the threshold. This means that the threshold needs to change before the sign of integration changes, in order to prevent this state. So there is actually a battle of loops, and the winner has to be the positive feedback loop if we want an oscillation [13].

In 3.8.d we see how things happen in real signals and real circuits: the system "senses" the proximity to the threshold of the output of the integrator at point 1, and so the sign of integration starts to change, causing a decreased slope in the output signal. The threshold crosses the integrator output with a reduced slope at point 2, which can cause many issues: increased noise on the timing of the transition, and most of all unpredictable outcomes. Finally at point 3, the value to be integrated has completely changed sign and we consider from this point that the negative value is being integrated.

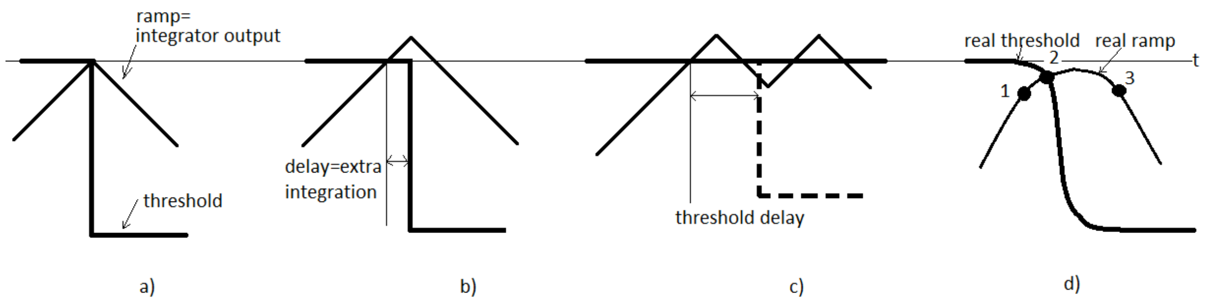


Figure 3.8: Zoom in the threshold crossing point: battle of the loops in a)ideal case b)ideal case with some comparator delay c) deadlock situation if threshold can't swap on time d)real case

In order to avoid this battle, we need to make sure that the two changes do not happen simultaneously. This can be taken care of by either inserting a surge (see Figure 3.9.a) that increases the value on the integrator, and gives time to the threshold to change safely, or just insert a time delay between the two actions (3.9.b)

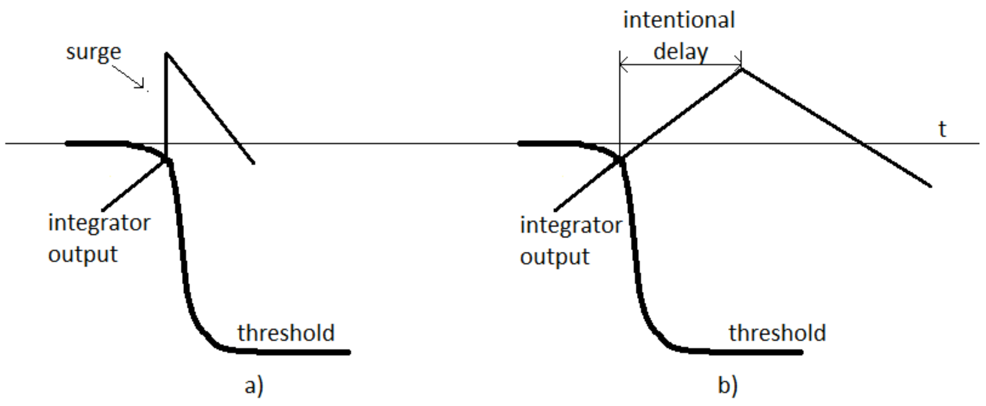


Figure 3.9: Ways to avoid the battle of loops: a) with a surge b) with time delay

3.4.2 Comparing the several implementations

Let's firstly name the implementations in the order they were presented, **first** the implementation of two thresholds and a switch to choose between them, **second** the one with two thresholds and two comparators, **third** the one using a surge and only one threshold and **fourth** the quadrature one (two integrators, where threshold crossings of one decide for the switch of the sign of integration for the other) By preliminary inspection we can guess that

-**first** and **second** implementation will suffer from the battle of loops, so even though ideally their behavior will be good, in real life they will suffer from the fact that they suppose a sharp and simultaneous change in both integration sign and threshold value

-**third** implementation will not suffer from this problem but it will have put the delay of the surge in the loop (delay for the surge to "force" its value on the integrator), so the period will be dependent on that, so this brings in a huge unknown (how well can we control the timing of the surge that in real life will not happen instantly)

-**fourth** implementation seems as the most promising. If the system doesn't suffer much from the possible mismatch between the two copies of the timing reference and if we have the freedom to spend power for both of them, then this idea will implement the "time delay" way to avoid the battle of loops: one signal that is deciding for the other one to change sing of integration by crossing a threshold, keeps on rising (or falling) and we have decoupled the two changes.

For all the upcoming mathematical explanation of the behavior of the four implementations, we are going to consider that we are working in the voltage area of the electronics domain, so we have a V_{con} available, i.e. a constant voltage (for example a bandgap reference), and for the slope, we integrate a current on a capacitor.

2.4.2.1 *Mathematical expressions for the first implementation*

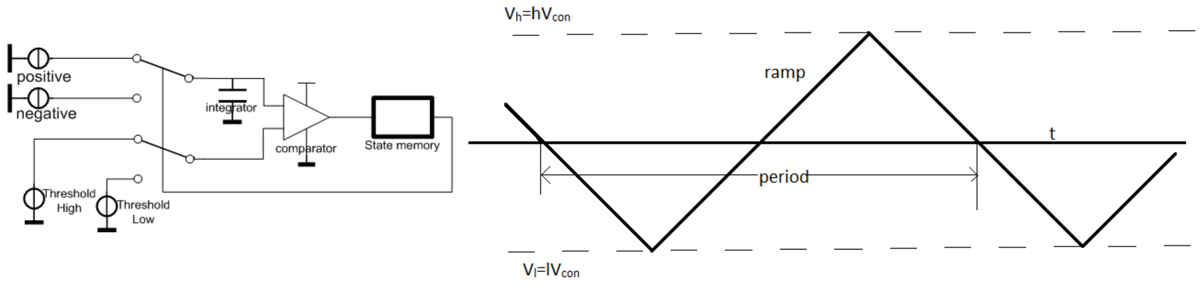


Figure 3.10: first implementation

We have $V_h=hV_{con}$, $V_l=lV_{con}$, because we suppose that the two thresholds are going to be created as portions of the constant voltage.

As for the slope, it will depend on the capacitor integrating the current, so:

$$slope = \frac{dV_i}{dt} = \frac{i}{C} = \frac{V_{con}}{R_{in}C},$$

$$T = \frac{2(h-l)V_{con}}{slope} = \frac{2(h-l)V_{con}}{\frac{i}{C}} = \frac{2(h-l)V_{con}}{\frac{V_{con}}{R_{in}C}} = 2(h-l)R_{in}C$$

The interesting result here is that we seem not to depend on the constant voltage value. Factor "2" depends on the matching of the values to be integrated (the plus sign and the minus), h, l, R and C are all based on passive components for which we can expect the most accurate results.

OFFSETS IN FIRST IMPLEMENTATION

Offsets can appear in every signal's amplitude, in:

-level of the values to be integrated (the currents), and these will be further analyzed

-thresholds (the two "levels" which are important to the oscillation), and these will be further analyzed as well

-comparator input, which will be ignored

-memory output level, which will be ignored

-switches, which will be ignored

We chose to ignore some of the offsets. The comparator input offset can be modeled with a constant offset in one of the two inputs of the comparator. If that's put in front of the input reading both thresholds, then that offset is going to shift both thresholds and the whole slope a V_{offset} higher or lower. So that has no effect on the frequency (which depends on the difference of the two offsets).

The memory output is supposed to be high or low supply (Vdd or Vss) If we suppose ideal sudden change of the memory output from low to high and vice verca, there's no change if low is Vss±offset or high is Vdd±offset. If the state switching is not sudden like in the ideal case where it lasts an indefinitely short amount of time, but there is a slope, then a small offset will shift that slope a bit to the right/to the left and introduce a timing error, concerning when the change of the memory state will be read by the following components (for example the switches). Timing errors will be discussed later, in the delays section.

Possible offset in the switches (supposing that the switches are going to add some non zero offset to the values they are delivering to the comparator) will introduce a delay, since the actual value will be "read" earlier or later.

<u>Offset in Value to be integrated:</u> If we get a change a% in the value of the currents, this means that we will get a change a% in the denominator of the period (the slope), so

$$T_{new} = \frac{2(2hlV_{con})}{\frac{i+ia\%}{C}} = \frac{2(2ChlV_{con})}{i+ia\%} = \frac{2(2ChlV_{con})}{i(1+a\%)} = \frac{T}{1+a\%}$$

So the period is changing to *T_{old}*1/(1+a%)* or the frequency is changing to *f_{old}*(1+a%).*

<u>Offset in Threshold(s):</u> Now we suppose the value of thresholds changes by a%: Mathematically we can solve any case of each one of them changing or both of them changing by a different amount. Theoretically the same change on both of them cancels out (shifts the slope higher or lower by that offset). Suppose a case where there is some difference in the two offsets of the two thresholds, resulting in a change of the difference of them by a%. Then as observed in the final formula we get

$$T_{new} = \frac{2(2hlV_{con})(1+a\%)}{slope} = \frac{2[2hlV_{con}(1+a\%)]}{\frac{i}{C}} = 2R_{in}C\,[2hl](1+a\%)$$

So the period is changing to *T_{old}*(1+a%)* or the frequency is changing to *f_{old}*1/(1+a%)*

Comment: Change in the value to be integrated, translates in change in slope, which translates in the accuracy of the values of C and $R_{in.}$ They both change the slope of integration in the same way: they appear in the denominator of the slope and so the changes transfer directly in the period. Moreover change in threshold translates to change in h-l which are based on passive components too (possibly resistive dividers), and as proven above changes to the h or l factors appear directly at the output.

DELAYS IN FIRST IMPLEMENTATION

Delays can appear in all transitions that are present in the circuit:
-in the change of values to be integrated (if there is a delay in the change of integration from positive to negative value and vice verca)
-in the thresholds to be compared (high/low: we have a switch controlling which is being compared)
-in the comparator output change (from high to low and vice verca)
-in the state memory output that controls both the switches.
All of them need to be analyzed.

<u>Delay in Value to be integrated:</u> If we get a delay in “reading” the change from positive to negative value to be integrated, as large as τ_d=a% of the period, then we see that until the change in integration sign is being noticed, the system will slightly keep integrating above the thresholds. So as seen in the following Figure 3.11, this τ_d is added 4 times in the period. We can show that mathematically from the fact that *$\tau_d = dV_{extra}/slope$*, so *$dV_{extra} = slope*\tau_d$* and this *$dV_{extra}$* needs to be added to the high (system is delayed from starting integrating downwards) and subtracted from the low value(system is delayed from starting integrating upwards), so

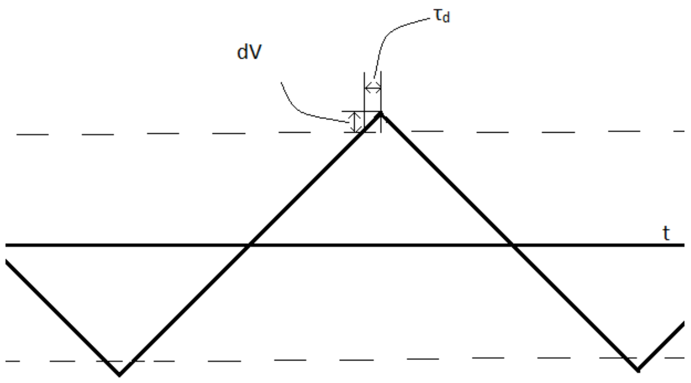


Figure 3.11: τ_d in the change of the sign of integration

$$T_{new} = 2\frac{hV_{con} + dV_{extra} - (lV_{con} - dV_{extra})}{\frac{iV}{C}} = 2\frac{(h-l)V_{con} + 2dV_{extra}}{\frac{V_{con}}{R_{in}C}}$$

$$T = T + \frac{4*slope\,\tau_d}{slope} = T + 4\tau_d = T(1+4a\%)$$

So the period is changing by adding 4 times the delay induced, and the frequency is changing to *f_{old}*1/(1+4a%)*

Delay in Threshold(s): When there is a delay in the change of threshold, all we have to avoid is running into the worst case scenario of a deadlock situation as described in 3.4.1 and can be seen in Figure 3.8.c. If the threshold is changing faster than the integration slope (Figure 3.12), then we face no problem. In this point it should be mentioned that all of the above were proven by simulations in Cadence platform using ideal blocks, and adding non-idealities like delays and offsets.

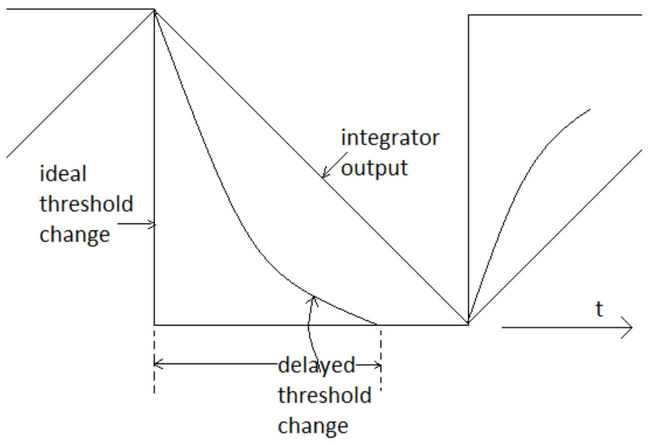


Figure 3.12: how threshold delay is of no issue as long as it is less than the time the slope needs to get to the new threshold

Delay in Comparator slope: The output of the comparator is used to trigger the change in the state memory which there after will change the sign of integration. Therefore purely by logic we can argue that there are 2 changes of the comparator output every period. If each change is delayed by τ_d that will cause an extension of the period by twice that delay, like in Figure 3.13. Unfortunately when the state memory change is delayed, the system keeps integrating towards the wrong direction and that will have to be compensated for, like shown in Figure 3.11. Therefore this delay has to be added 4 times per period, and not just two.

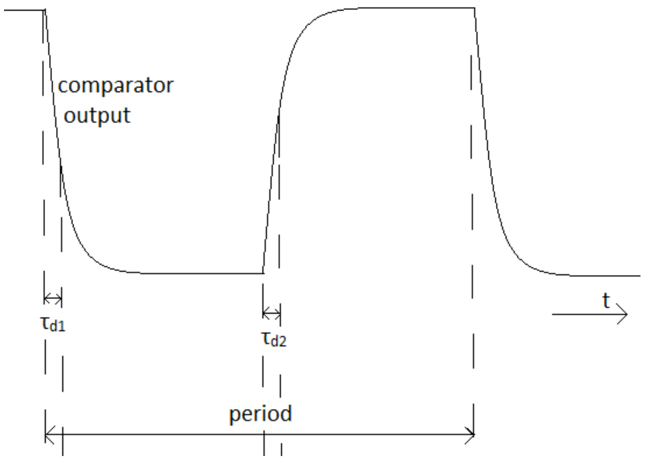


Figure 3.13: τ_d in the change of the comparator output

Delay in State memory output: Since the state memory output is actually our output, someone could think that a delayed signal gives only a phase error, and period would remain unchanged. But this signal is used also for controlling the switch of integration sign, so it will have some extra effect, obviously the same like a delay in the value to be integrated (a delay in the control signal of a perfect switch happening here or a perfectly controlled non-ideal switch like in the "delay in value to be integrated" is the same thing: it causes the signal to change slower, see Figure 3.14). If we get a delay $\tau_d=a\%$ of the period, then the period is changing 4 times the delay induced, and the frequency is changing $f_{old}*1/(1+4a\%)$. Of course if it delays more the change in threshold, integrator runs into the deadlock situation of Figure 3.8.c, and this is a highly undesirable case.

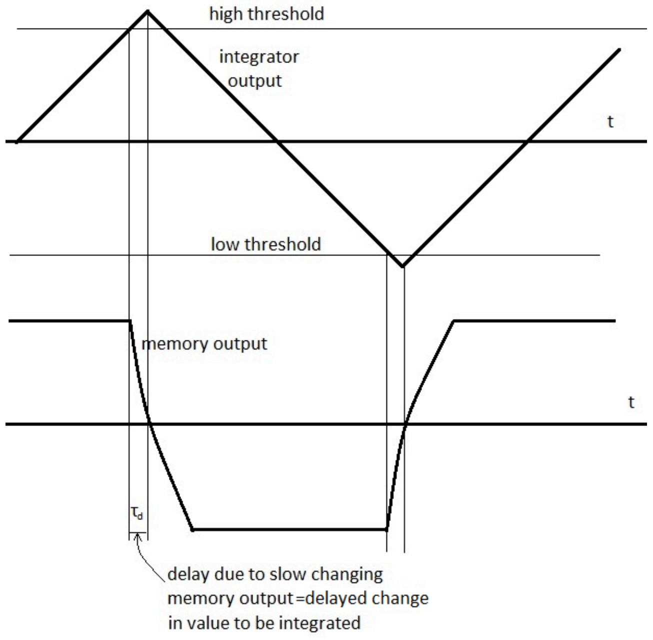

high threshold
integrator output
t
low threshold
memory output
t
τd
delay due to slow changing memory output=delayed change in value to be integrated

Figure 3.14: delay τ_d in the change of the state memory output is indeed a delay in this signal being "read and applied" so the same like the delay in changing the sign of integration

3.4.2.2 *Mathematical expressions for the second implementation*

The second implementation (two thresholds hooked in two comparators, and not controlled by a switch) is mathematically the same as the first one. The expression for the period is the same, and the dependencies are almost the same. The only things different are the following:

-offset in the comparator input is now not shifting both thresholds, so it cannot be neglected like in the first case. Here as can be seen in Figure 3.5.b a possible offset in one comparator will make integration last a bit more, and this will have to be corrected, so the period will be increased twice that dV cause by the offset. So if both comparators add a% and b% respectively, we will measure an increase in the period of *2a%+2b%*. Of course there is always a chance that they could cancel out.

-a delay in changing the threshold value is not valid here (since both comparators have the thresholds hooked up on their inputs all the time)

-the delays could vary differently in the two comparator paths: if a comparator output slope would be present in only one of them, this would change the period in $T_{old}(1+2a\%)$. If they would be in both comparators the result would be $T_{old}*(1+2a\%+2b\%)$ unless they are the same, in which case we end up in exactly the same equation like for the first implementation $T_{old}*(1+4a\%)$. Since it's a random error, there is always the chance they cancel each other with opposite signs.

-the requirement of a digital block reading the two outputs of the comparators, and ordering the state memory to change or stay put. This could be an XOR for example, which would be in our signal path, and could also introduce delays, but that delay can be included in the general “memory” delay.

3.4.2.3 *Mathematical expressions for the third implementation*

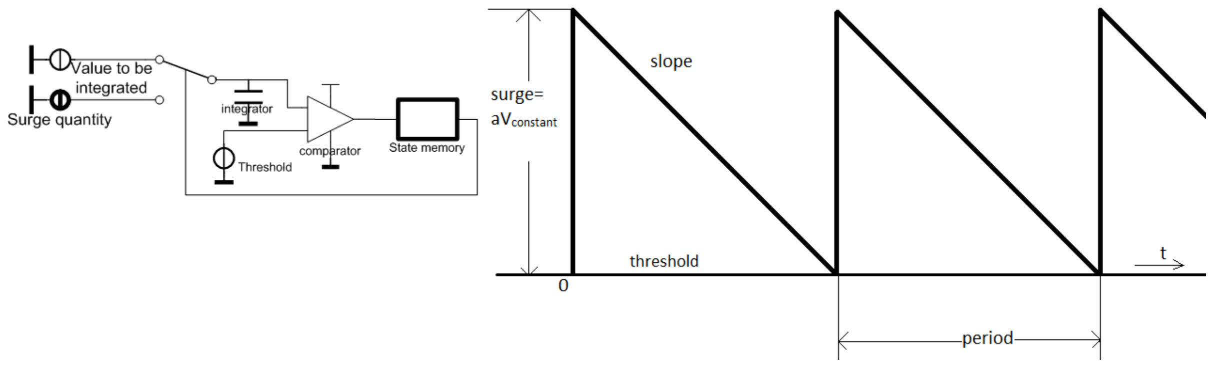


Figure 3.15: the third implementation

We implemented this idea, by taking the voltage across the capacitor as 0 (initial value), force it to rise with the predefined amount of surge, and then it integrates a negative value, so it reaches the threshold (ground in our case), and surge is added again etc.

$$slope = \frac{V_{con}}{RC_{in}}, \quad surge = dV = aV_{con}$$

$$T = \frac{dV}{slope} = \frac{aV_{con}}{\frac{V_{con}}{RC_{in}}} = aRC_{in}$$

We supposed surge is indefinitely fast. If it’s not, it is added in the period, as a constant t_{surge}, twice per period. If we would use another threshold instead of ground, then simply we would have on the numerator *surge-threshold*$=aV_{con}-bV_{con}=(a-b)V_{con}$.

The interesting result here is that period has the same expression like in the last two implementations, but the dependency here is a, instead of (h-l). Many of the results are expected to be similar.

OFFSETS IN THIRD IMPLEMENTATION

Like in the analysis for the first implementation, offsets that we care about can appear:
-in the level of the value to be integrated (the current-only with negative sign here)
-in the thresholds, one implicit in the surge value, or the other threshold (ground or any other) being used

Like in the first implementation, possible offsets in comparator output value and memory output are translated into delays and will be analyzed later. Also like in the first implementation, any offset in the input of the comparator appears like offset in the threshold, since the threshold is hooped up on the comparator input all the time. So it will be analyzed with the threshold’s offset.

<u>Offset in Value to be integrated:</u> Like before, the expression is the same:

$$T_{new} = \frac{aV_{con}}{\frac{i+b\%i}{C}} = \frac{aV_{con}C}{i+b\%i} = \frac{aV_{con}C}{i(1+b\%)} = \frac{T}{1+b\%}$$

So the period is changing to *T$_{old}$*1/(1+b%)* or the frequency is changing to *f$_{old}$*(1+b%).*

<u>Offset in Threshold(s):</u> Now we suppose we change the value of the surge by b%. We have:

$$T_{new} = \frac{surge}{slope} = \frac{aV_{con} + b\%aV_{con}}{\frac{i}{C}} = T(1+b\%)$$

So the period is changing to *T$_{old}$*(1+b%)* or the frequency is changing to *f$_{old}$*1/(1+b%)*

Unlike an offset in the surge, possible offset to the threshold level (or the comparator input) will not affect our period. Taking for granted a stable surge of a specific positive value, in the presence of threshold offset everything will shift and be superimposed to that offset. So the surge is added to the offset, integration takes place down to that offset, the surge is added again on top of the offset etc. The timing is hidden in the accurate surge. If on the other hand the surge is somehow implemented as reaching some voltage level and not a constant quantity, then possible offset in the threshold (lifting the zero level for our case) would mean a change in the period once per period (that means multiplied by (1+b%)).

DELAYS IN THIRD IMPLEMENTATION

Delays can again appear in all transitions:

-in the value to be integrated (if there is a delay in the transition from zero to surge value or from surge value to value to be integrated)

-in the comparator output

-in the state memory output that controls the switches operating the surge the ground and the integrating value.

We suppose that the surge has no delay, and even if it does, it can be included in the first delay that will be analyzed. Moreover the threshold is hooked on one input of the comparator, so there is nothing changing there that could introduce a delay.

<u>Delay in Value to be integrated:</u> If there is a delay τ_d=b% of the period, in which we include both some delay when getting from zero to the surge value or getting from surge value to start integrating the negative value, then we can argue that this will be added once in every period, as seen in Figure 3.16.

This obviously includes also the possible delays caused by the switches that will "provide" to our circuit the surge value, and the value to be integrated.

As seen in the Figure 3.16, the period is changing by the delay induced, and the frequency is changing to *f$_{old}$*1/(1+b%).*

The reason why this delay changes once our period and not multiple times (like in the first and second implementation), is because we suppose that the integrator cannot integrate lower than zero level, since zero is one of the supplies, and moreover when the surge is being "added", the integration process is stopped. Otherwise the system would keep integrating to negative values, while the surge would try to be “forced” on the integrator value.

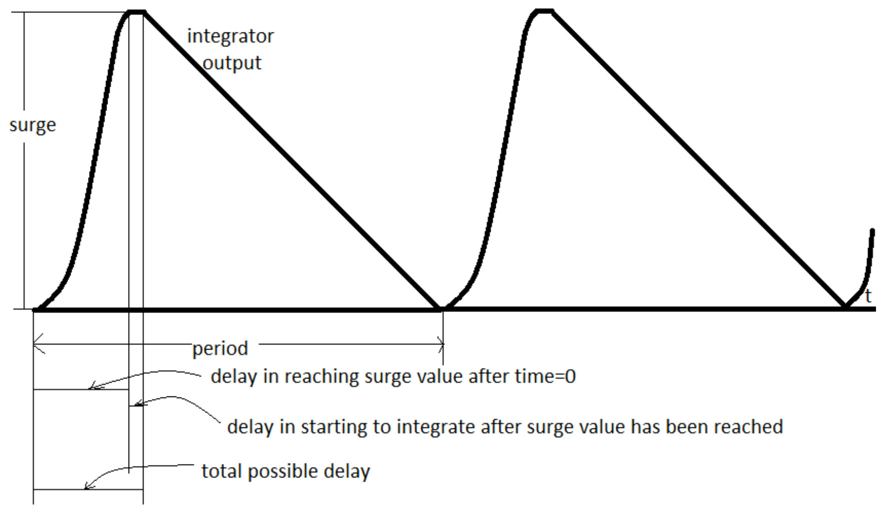

integrator output
surge
t
period
delay in reaching surge value after time=0
delay in starting to integrate after surge value has been reached
total possible delay

Figure 3.16: delay in the change of value of integration

Delay in Comparator slope: The output of the comparator is used to trigger the change in the state memory which there after will change the sign of integration. Therefore it’s exactly the same case like in analysis for the first implementation, but this time we do not integrate to the wrong direction and need the time to correct it, thanks to the surge, as analyzed in the previous paragraph. So we get that delay only once in our period (the output just saturates at zero level and waits for the surge). If we could integrate to negative values (like if the threshold would be an arbitrary one and not ground) then it will not affect our period at all, provided once again that the surge will be a constant quantity. Delay in the comparator will just be responsible for the part of under the threshold integration, and the highest point would be (surge) minus this part already integrated below the threshold.

Delay in State memory output: Here again, the state memory output is our output. It controls the switch that will apply the surge and the switch that will start applying the integrating value. Both these changes are performed with one change in state memory - we face the same questions as to how the surge is going to be implemented. Supposing surge and integrating process do not coexist but they are still controlled by one change in state memory output, that delay will be added once in every period.

3.4.2.4 *Mathematical expressions for the fourth implementation*

For the fourth implementation we get the output seen on Figure 3.17a, based on a circuit like in Figure 3.7. If any delay or offset is inserted, then the whole oscillation falls apart, since system lacks one control parameter, one threshold (Figure 3.17b). The integration value will be rising to infinity because the possible delay in the first integrator will be “memorized” in the value of the other integrator that will change integration sign later, cross the threshold later, and if the same delay will be added once every period, this will lead to instability (real signals would be saturated by the supplies eventually).

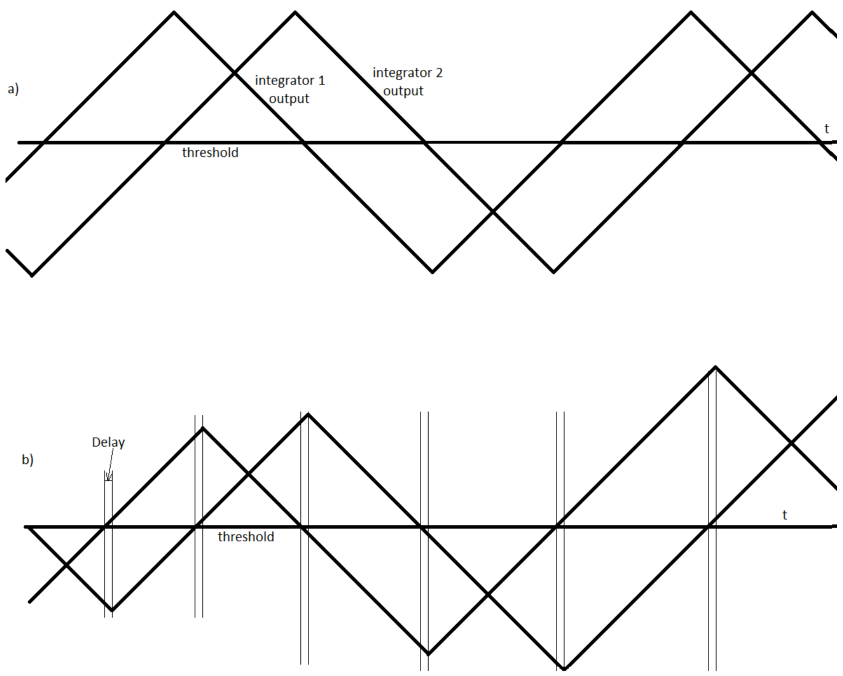


Figure 3.17: a: ideal quadrature oscillator b: output with delay, using only one threshold

One solution to the problem [13] could be to include two Schmitt triggers (Figure 3.18.a) that would provide 2 more levels of control (so now system has one extra) by including their two thresholds to the zero threshold already in use. The zero crossing of the first integrator output ignites the change in the thresholds of the second Schmitt trigger, which forces the second integrator to change sign of integration, like in Figure 3.18.b. Unfortunately, in case a delay existed, that would be memorized like in 3.17.b, reaching a point where the change in integration sign would happen with reaching the –non modified- thresholds of the Schmitt triggers. Therefore the system would keep oscillating but would not be controlled by the zero crossing or the other integrator but rather by reaching the thresholds (Figure 3.18c-where only the delay and the thresholds of one of the two Schmitt triggers is drawn, in an attempt not to make the Graph too messy. We assume the same things happening in the middle between the delays drawn, for the crossings of zero of the other integrator and for the other Schmitt trigger.)

In case Schmitt triggers are unwanted (and usually they are, since their strong point is –more or less- just their flexibility) some extra control should be brought in the system, like limiting low value to zero. In that case, with the existence of a delay, the signal would clip on zero waiting for the next threshold crossing. Therefore the duty cycle would not be 50%. It would be perfectly repetitive, but still not completely under our circuit's control (Figure 3.18d)

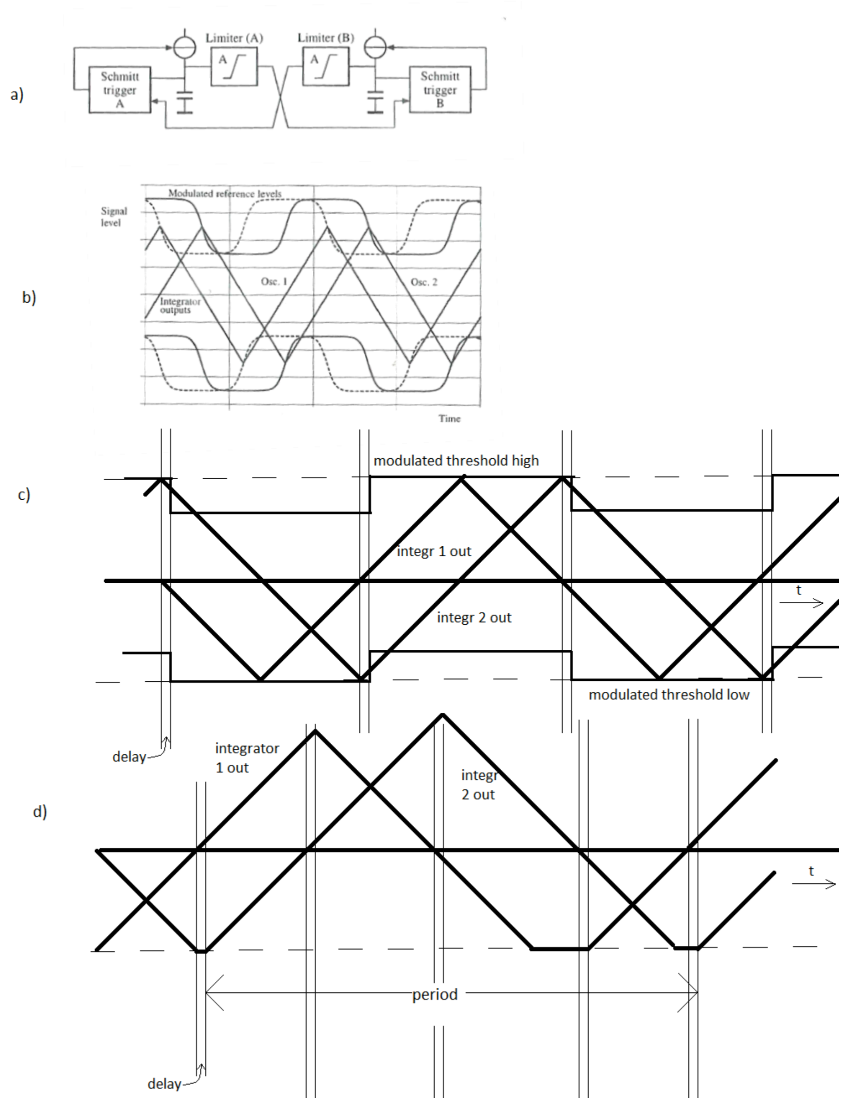


Figure 3.18: a: Westra's choice for a quadrature oscillator (picture taken from [13]) b:modulating the Schmitt trigger's thresholds(again from [13]) c: worst case scenario (for simplicity with ideal threshold modification) for delay d: if ground is added as another control

3.5 Final choice and canceling the inherent imperfections

Transforming all the previous equations/formulas/assumptions in sensitivities, for the first 3 implementations that were mathematically analyzed, we get the Matrix 3.1.

We are not particularly satisfied by any of the analyzed topologies, but we see that some (third in particular and fourth as a general idea) have some interesting advantages. So we are going to create a hybrid by a combination of the best parts of all the previous analyses.

3.5.1 New implementation design

The new design is the one shown in Figure 3.19, with signals like in Figure 3.20.

Matrix 3.1: Sensitivity Matrix; how much each component can affect <u>the period</u> of the signal for an <u>a%</u> change in the crucial parameter.

	First implementation	Second implementation	Third implementation
a%I offset in value to be integrated (current I)	≈-a% (if both currents change the same percentage)		≈-a%
a%ΔV offset in threshold (Volt difference)	≈a%		surge: ≈a% threshold and/or comparator: 0
a%ΔV offset in comparator input	0	≈2a%+2b% if a% for one and b% for the second comparator	0
Offset in binary outputs of comparator/state memory	NO EFFECT provided it's a small offset and doesn't affect the behavior of the switches being controlled. Worst case it introduces delays		
a%T delay in sign change of value to be integrated	≈4a%		≈a%
Delay in Threshold value switch	NO EFFECT if fast enough	NOT VALID	NOT VALID
a%T delay in comparator output switch	≈4a%	≈2a%+2b% if a and b are delays per comparator (if same, ≈4a%)	≈a%
a%T delay in state memory output change	≈4a%		≈a% if one memory change ignites both surge and integration

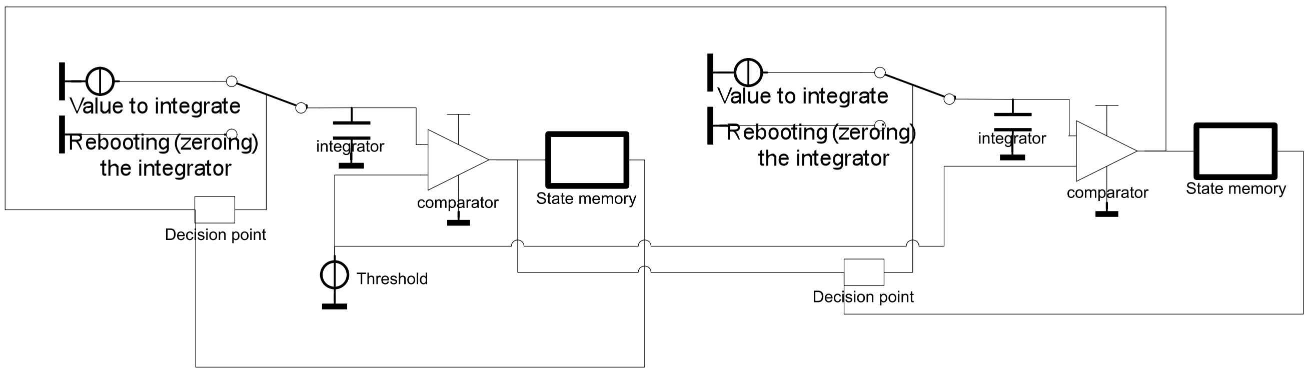


Figure 3.19: The new implementation of the oscillator

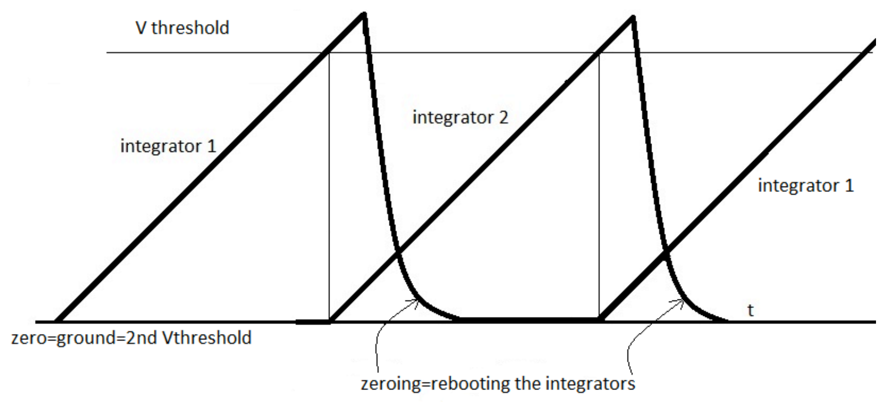


Figure 3.20: The signals in our final implementation

The idea is the following: In order to have enough controls, we keep zero (low supply, ground) as one threshold and create an extra one. So we will go for a two threshold oscillator, since this will be easier and better controlled than creating a surge (voltage) value, and making it (the surge) temperature independent and flexible enough to be added without imperfections to our integrator. Therefore we have solved the issue of fewer controls in the quadrature oscillator (4th implementation), but we will keep the basic idea of the quadrature oscillator that the crossing of a threshold of one integrator starts the integration in the other one, while the same one keeps integrating. This control is done by connecting the output of one comparator to the switch of the other timing reference via a decision point to “force” it to start integrating. In the meanwhile the integrator that crossed the threshold will keep integrating for some delay, which will be included in the state memory, but which will be shown that is arbitrary and not important for the timing of the circuit. After some time the output of the state memory of the first timing reference will finally change, and reboot the integrator (in electronics terms it is going to discharge the capacitor by connecting the ground to both sides of it), and wait for the crossing of the threshold of the other one, in order to start integrating again, giving time for the capacitor to safely discharge practically completely. The so called “decision point” in Figure 3.19 just takes care that when the memory commands, ground is being connected for discharge, while when the comparator of the other part commands, the integration starts – it will obviously be some logic function which we decoupled from the state memory since we want it to reply quickly.

Another improvement we can make is to include the characteristic of the third (and first) implementation, where thanks to using just one comparator, we eliminated the effect of offset of the comparator. In order to vanish the effect of the comparator offset, we will have to use only one comparator and interchange the inputs, so the effect of offset vanishes (see also 3.6). The price to pay for having switches choosing what is being compared at the inputs of that (one) comparator is less to be paid than having two comparators, two possibly different delays through there comparators in the signal path along with their temperature dependencies and twice the power needed for the comparator.

So the final implementation is the one shown in Figure 3.21.

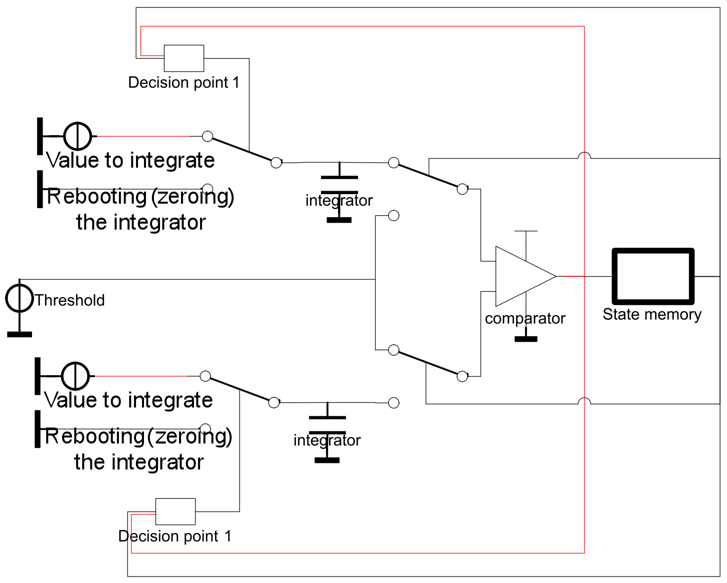


Figure 3.21: The final implementation of the oscillator (fast timing path marked red)

We have two separated control timing paths, a fast and a slow one (where by fast we characterize the one that is important for the timing of the oscillation and should be so fast as not to change our frequency, while by slow the one that is not important for the timing of the oscillation, even if the delays it introduces drift a bit over process or temperature). Integration takes place, until the threshold is crossed, then the change in the output of the comparator is taken into account in the decision point 1, which has to respond fast, in order to make the other integrator start integrating. Then, the slow path changes the output of the memory, and this will change two things: first change the position of the switches that choose the new pair of inputs that have to be compared (threshold and the other integrator's output in interchanged inputs of the comparator) and secondly discharge the first integrator.

3.5.2 Mathematical expressions for the new design

$$TaRC22\frac{dV}{slope} \qquad \frac{aV_{con}\,2\,0}{\frac{V_{con}}{RC_{in}}} \qquad {}_{in}$$

OFFSETS IN FIFTH IMPLEMENTATION

Offsets can appear in the levels of the following signals:
-in the level of the values to be integrated (the current)
-in the threshold (here only one "level" is important, the other one is ground).

There is also possibility for comparator offset at its input, but it disappears thanks to the topology that was designed. The way the comparator (input) offset disappears will be explained extensively in 3.6 showing cadence results. Moreover, if offsets appear in the second memory output and/or the level of the comparator output, we suppose they can only affect our frequency if they are translated in delay issues, and they will be analyzed later.

Offset in Value to be integrated: If we get a change b% in the value of the currents, means that we will get a change a% in the denominator of the period, so

$$T_{new} = 2\frac{aV_{con}}{\frac{i+ib\%}{C}} = 2\frac{aV_{con}C}{i+ib\%} = 2\frac{aV_{con}C}{i(1+b\%)} = \frac{T}{1+b\%}$$

So the period is changing to *T_{old}*1/(1+b%)* or the frequency is changing to *f_{old}*(1+b%)*.

Offset in Threshold: Now we suppose we change the value of threshold by b%:

$$T_{new} = 2\frac{dV}{slope} = 2\frac{aV_{con}(1+b\%)}{\frac{i}{C}} = T(1+b\%)$$

So the period is changing to *T_{old}*(1+b%)* or the frequency is changing to *f_{old}*1/(1+b%).*

DELAYS IN FIFTH IMPLEMENTATION

Delays can appear in all transitions that are present in the circuit:
-in the value to be integrated (in this case a delay from zero to start integrating or from integrating value to zero)
-in the comparator output
-in the state memory output that controls the switch of the other part.
The threshold is brought "on the spot" ready to be compared with the slope long before the crossing of the threshold, so we suppose there can be no delay in the swapping of the switches in front of the comparator that can result somehow to our frequency.

Delay in Value to be integrated: The situation is different in the two cases: if there is a delay from integrating value to zero, since this happens in a time slot that is of no importance for the timing of the clock, then it has no effect (provided that the discharge of the capacitor will have finished before the other one crosses the threshold and commands this one to start integrating again).

On the other hand, if there is a delay when going from zero to integrating value, any delay will be added once in every period. The situation can also be seen in Figure 3.22. If both integrators introduce a% and b% delays respectively, then the new period will be *T_{old}*(1+a%+b%).*

Delay in Comparator slope: The output of the comparator is used to trigger the change in the integrator of the other part via decision point 1. So any delay will cause this difference to be "read" later. This happens twice in every period, so this will be added twice in the period, like in previous cases.

Delay in State memory output: By taking the discharge phase out of the timing loop, any delay in the memory output is no longer important to the operation, provided once again that the discharge will have finished before the crossing of threshold of the other oscillator. This can be considered as the same effect like the delay in changing the integrated value from the reached value to ground (check Figure 3.22). It simply plays no role in the timing.

Now in Matrix 3.2 we include the results of the 5th implementation in the Matrix of sensitivities 3.1.

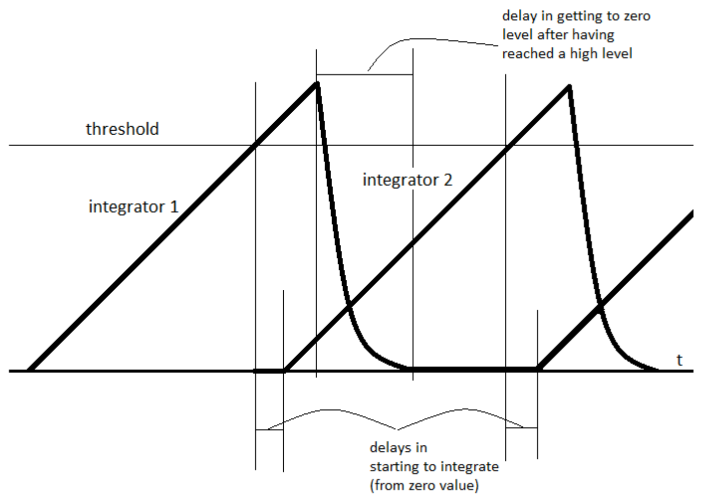

delay in getting to zero level after having reached a high level
threshold
integrator 1
integrator 2
t
delays in starting to integrate (from zero value)

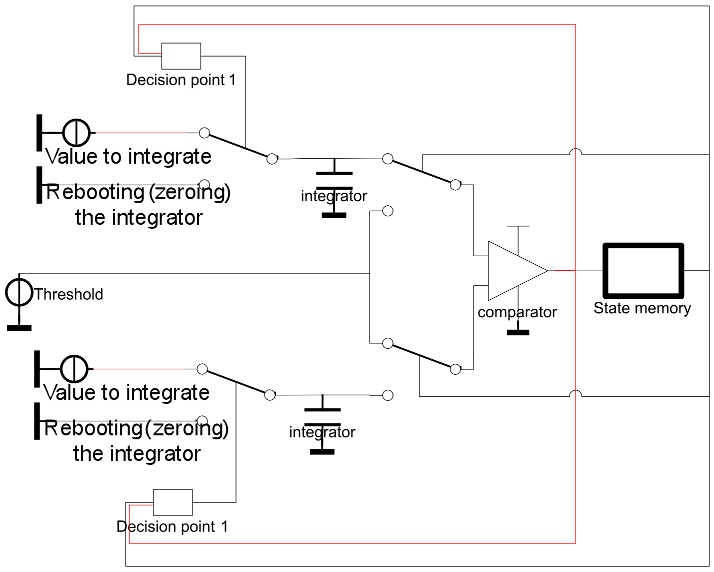
Figure 3.22: Possible delays in rising and falling slopes of the integrator(s)

Matrix 3.2 Sensitivities Matrix including the fifth implementation

	First implementation	Second implementation	Third implementation	Fifth (final) implementation
a%I offset in value to be integrated (current I)	≈-a% (if both currents change the same)		≈-a%	≈-a%
a%ΔV offset in threshold (Volt difference)	≈a%		Surge: ≈a%	Threshold: ≈a%
a%ΔV offset in comparator input	0	≈2a%+2b% if a% for one and b% for the second comparator	0	0
Offset in binary outputs of comparator/state memory	NO EFFECT provided it's a small offset and doesn't affect the behavior of the switches being controlled			
a%T delay in sign change of value to be integrated	≈4a%		≈a%	0 if in integr-> discharge ≈a%+b% if in discharge->integr
Delay in Threshold value switch	NO EFFECT if fast enough	NOT VALID	NOT VALID	NOT VALID
a%T delay in comparator output switch	≈4a%	≈4a% if both comparators change the same	≈a%	≈2a%
a%T delay in state memory output change	≈4a%		≈a% if one memory change ignites both surge and integration	0

3.5.3 Imperfections and ways to cancel them

Now we will analyze the imperfections per block, in Matrix 3.3 in correspondence to Figure 3.21 (repeated here for simplicity).



Matrix 3.3 All advantages and disadvantages of our implementation, block by block

Imperfections list (remaining in our design)			
block	Offset	Delay	Comments/Solutions
Comparator	No (see 3.6)	Yes	The comparator is in the timing path, but the effect of offset is cancelled. Delay will be minimized with careful design
Decision points 1	No	Yes	The decision points “1” are in the fast timing path, so any delay is important. Binary operation so no offset makes sense.
State Memory	No	No	Memory is now out of the timing loop.
Zeroing the integrator	Yes (see below)	No	Grounding and discharging the capacitor happens as explained out of the timing sequence. But the accuracy of the “0” is important.
Switches	Yes	No	These switches in front of the comparator are choosing what is being compared. They will open/close out of the timing important moments, so no delay is important. Offset will be dealt with during the design phase (here offset mainly means charge injection caused by these switches).
Threshold	Yes	No	Threshold is always being produced, and is always available, no timing issues. Offsets may occur, solved by trimming.
Value to be integrated-Integrator	Yes	Maybe	The current charging the capacitor will be important: if it will be switched off, we may have delay issues, on top of the possible offsets (again fixed by trimming and/or design).

In this implementation, like in implementation 3, only one value is being integrated, not two with different signs and that way mismatch problems in plus and minus values are avoided. Unlike in implementation 3, the time it takes for the surge(=fast discharge now) to happen is of no concern. Even the value of the surge is defined by the value the integrator has reached minus zero (ground), so not strictly designed (therefore in order to be in accordance with the definition, this transition can no longer be called “surge”). As already mentioned, like in implementation 4, we don’t have any battle of the loops (the positive feedback is delayed from the negative feedback action in this design). Unlike in implementation 4, we have the two control levels needed: zero and the threshold, so we don’t run into uncertainties, and we don’t introduce Schmitt triggers and/or extra components

Disadvantage is of course that we are using two time constants (actually two copies of the same time constant by copying the circuit) and the possible mismatch between the two will cause problems to the duty cycle, and the power consumption is doubled theoretically, but only for the integrator part - we are using only one comparator. We rely on the fact that matching is the most reliable characteristic of modern ic technology, so the two timing parts will be relatively equal.

Now we will analyze the effect of the zero produced by the discharge switches on the capacitor:

$$T = 2\frac{dV}{slope} = 2\frac{aV_{con} - gnd(0)}{\frac{V_{con}}{R_{in}C}} = 2aR_{in}C$$

Ground, which is our second “control” voltage, may not be perfect zero, as we have supposed so far. If the switch that “connects” ground for discharge to the capacitor has a specific on resistance, then all the current will go through this resistance causing an offset voltage instead of real zero in the input of the integrator. So when integration will start it will start from a non-zero and not-controlled offset voltage - possibly temperature dependent.

Two ways to remove it are proposed, both of which are adding this offset to the threshold, and therefore not changing the numerator of the period:

$$T = 2\frac{dV}{slope} = 2\frac{(aV_{con} + V_{offset}) - V_{offset}}{\frac{V_{con}}{R_{in}C}} = 2aR_{in}C$$

Both ways to do that cost us some extra components and consequently power:

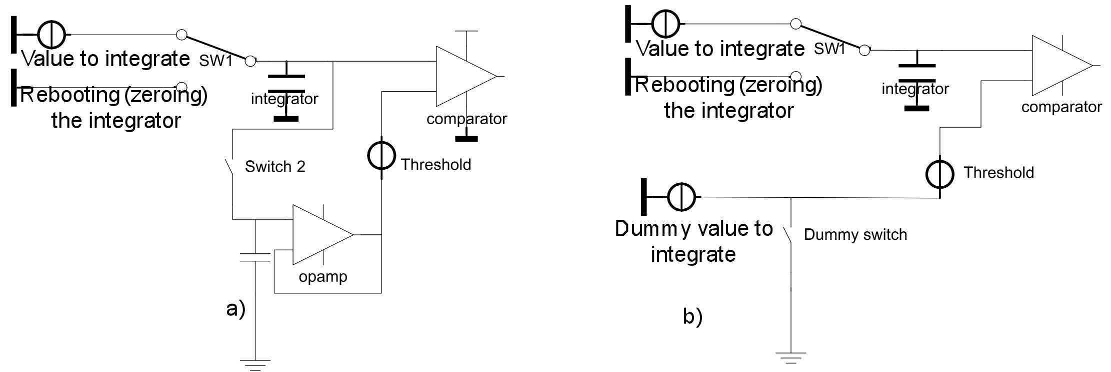


Figure 3.23: two ways to add the possibly varying offset created by the switch a: using an opamp b: using a dummy sub circuit to recreate the offset

In Figure 3.23.a we can see a direct way of using this offset. When SW1 is in discharging position, the offset is being created, and the solution is to put another switch (switch 2) closing at the same time like switch one, that helps save the offset value to a capacitor. Then, when SW1 will change position and start the integration, switch 2 will open too, offset will have been saved on the new capacitor, and threshold will be added to the offset: we removed the effect of the offset at the expense of a switch, an opamp, and a capacitor.

In Figure 3.23.b we can see an indirect way of adding the offset to the threshold: we create a dummy current source and a dummy switch. The current source will flow through the identical switch, creating – hopefully - exactly the same offset, which will be added to the threshold. The effect of the offset is removed with the cost of a dummy current source and dummy switch.

Of course we don't know what could be more power consuming, an opamp like in solution a, or a dummy current source just creating this offset like in solution b. Solution b seems more promising, for two reasons. Firstly the integrator is not affected by connecting extra components on his input, and secondly the dummy current can always be switched off, and by means of a capacitor save the value of the offset. This way we may add an extra switch and an extra capacitor but it will reduce dramatically the power consumption of this "offset removing" sub circuit and still not affecting in anyway our integrator.

The final choice will be made in the next Chapter where we will size the circuit and transform our needs into numbers.

3.6 Cadence simulations

Running transient simulations in cadence trying to recreate the signals of Figure 3.20 and the basic signals of topology in Figure 3.21, we get the following Figures 3.24, 3.25.

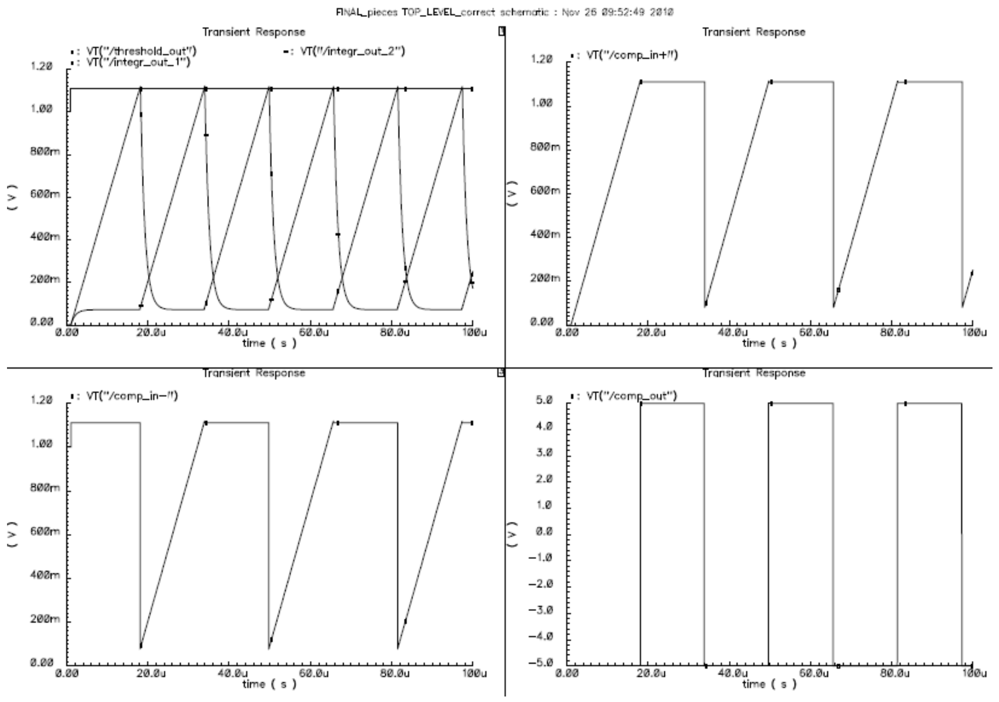


Figure 3.24: Cadence results for (a) the integrator and threshold outputs, (b,c) the inputs the comparator reads and (d) the total output of the comparator

In the cadence simulations presented in Figure 3.24 we didn't really keep long delays in the memory so the battle of loops doesn't seem to have been avoided when not zooming in. Since this delay can be arbitrarily chosen (see Figure 3.25 for timing details), that is not an issue, since the simulations were performed with ideal blocks. Also we notice that integration doesn't start from 0 volts. Indeed there is an issue to address there with the offset.

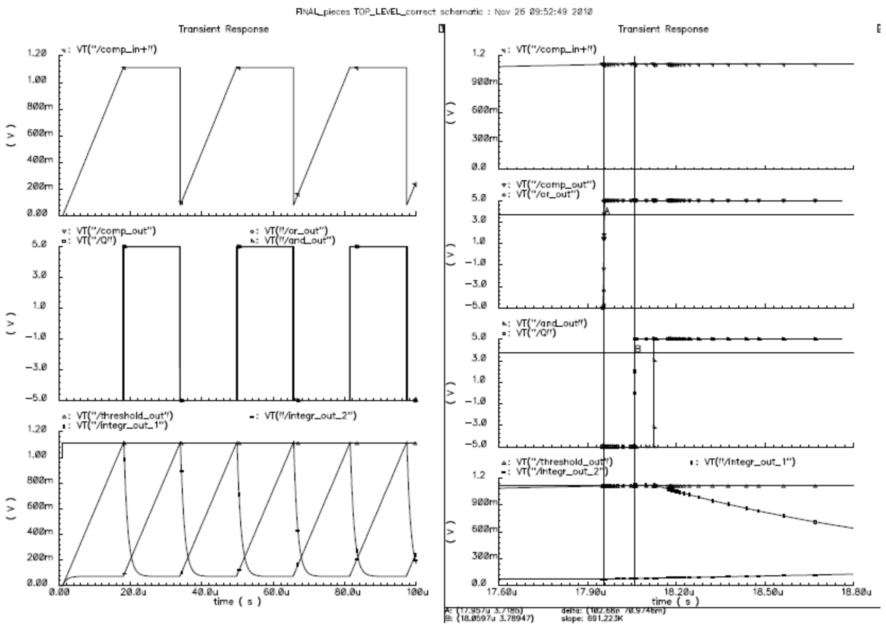


Figure 3.25: Cadence results showing the timing details

In Figure 3.25, we read on the right part on the first vertical line that comp_out is changing state, and therefore the other integrator starts integrating (not shown here: this change is what changes the state in decision point 1 in Figure 3.21, and completes the change in the fast control timing loop), Q changes a bit later (second vertical line), and commands the switches to change place, and and_out is changing even later (and_out as in AND gate), making our first integrator to start discharging. Every important event for timing is decoupled from each other. Moreover the two "decision point 1" belong both to the fast and the slow timing control loops. The one that has to integrate belongs to the "fast" one for each half period, while the other one that will have to start discharge the other one belongs to the "slow" one, but for the second half period they swap their roles.

Finally we work on an extreme case: the heavy presence of offset on the comparator. We see now (Figure 3.26) that this offset is being added once to the integrator's value and once to the threshold value, so one part of the period, since the threshold is higher will integrate more, but the next part will start integrating from a higher point ("ground"+offset) so it will end faster, compensating for the longer integration of the previous part. So this results in changes in the duty cycle only, not changes in the period. This is one of the huge main advantages of this topology, using only one comparator, since we don't care about the comparator offset, which usually is one of the major uncertainties in circuit design.

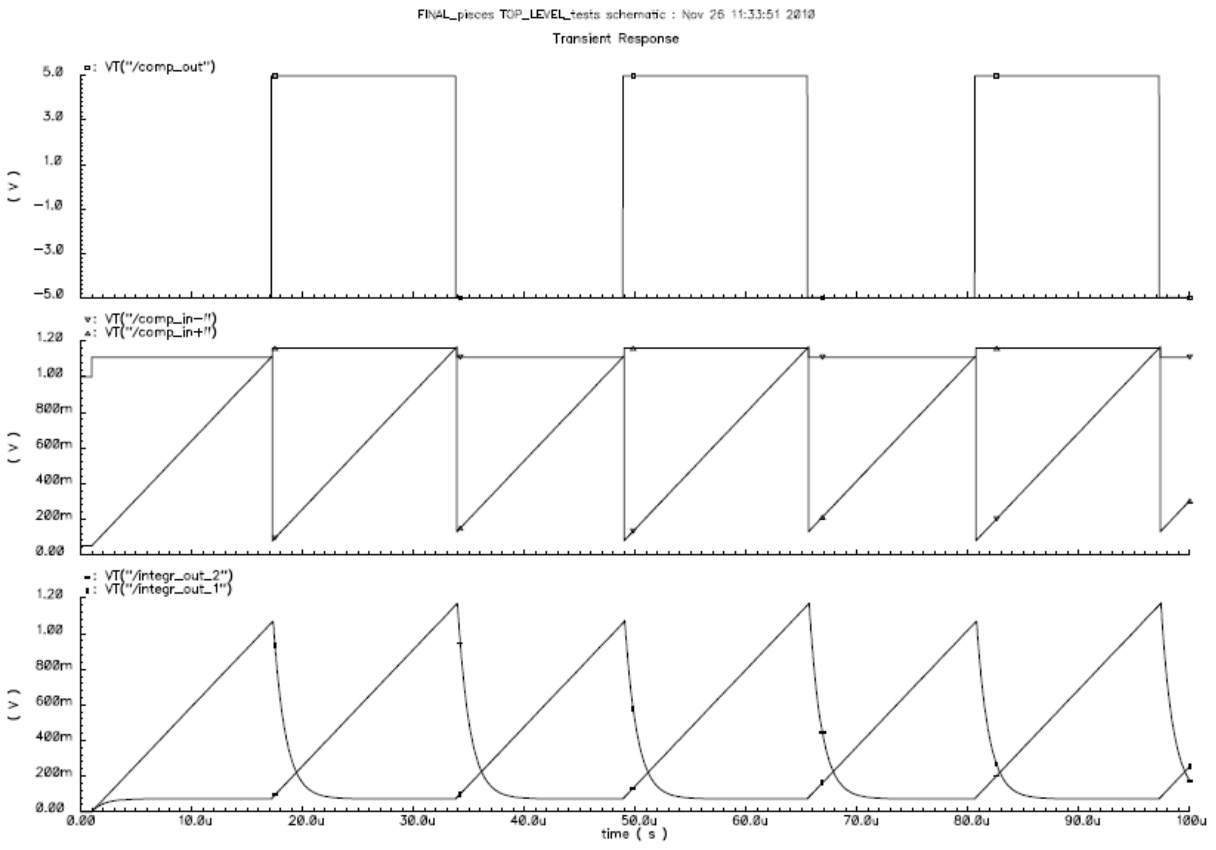


Figure 3.26: Cadence results showing the effect of (heavy) offset in the comparator

Chapter 4
Design of the stages of the oscillator

4.1 Threshold(s), integrator

4.1.1 Need for constant voltage

We have concluded that our final implementation is characterized by the following equation:

$$TaRC22\frac{dV}{slope} \quad \frac{aV_{con}\,2\,0}{\frac{V_{con}}{RC_{in}}} \quad in$$

The important thing to notice here is that V_{con} is deleted from the equation since it appears both in the numerator and the denominator, and this simply shows that there is no need for us to create a $V_{constant}$ for example with a bandgap reference. The general idea is that if some current is created, not necessarily stable over temperature, and it is forced on a resistor to create the threshold, and the same one on a capacitor so that it will be charged, then we see the following situation:

$$TRC2222\frac{dV}{slope} \quad \frac{ViR_{thresholdin}}{\frac{ii}{CC}} \quad \frac{*}{\frac{}{}} \quad in$$

where we notice that only the temperature dependency of the R and the C remain. Of course there will be some issues left to address later on (the comparator will not be “reading” the same voltage levels all the time, and matching of the currents that will create the threshold and charge the capacitor, should be really good, so these two currents can be deleted from the equation).

Luckily, in our technology (.5μm technology) there are one resistor type and one capacitor type that have really close and opposite temperature coefficients, which are also surprisingly low (compared to the rest of available resistors, for example N+poly resistor has 1 order and P+diffusion resistor 2 orders of magnitude higher temperature coefficients):

	PCA (poly-metal) capacitor	TFR (thin film resistor)
Temp Coefficient 1	+25ppm/oC	-27ppm/oC
Temp Coefficient 2	+0.034ppm	-0.014ppm
Spread (3 sigma)	-NA-	±11ppm

Matrix 4.1: Available resistor and capacitor type that will be used in this design

So the rest of the design is going to use for the resistor and the capacitor that are important for the timing of the oscillator only poly capacitor (C_{pca}) and thin film resistor (R_{tf}).

Comment: ideally if this is the only temperature dependency that remains, we will have approximately -2ppm/oC for a range of approximately 100oC, so 200ppm in total, which is already more than our target (which was 100ppm over the whole range of temperatures).

4.1.2 Integrator, correct “zero” voltage

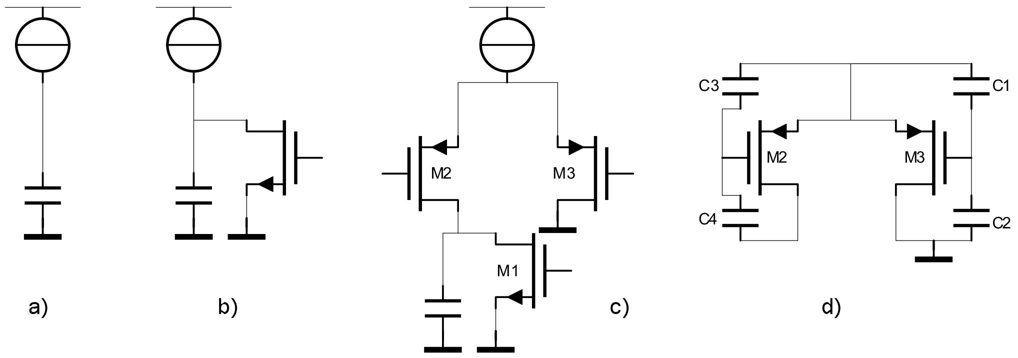


Figure 4.1 Integrator choices a-general idea, b-discharge switch, c-solving the "ground" issue, d-size of the transistors has to take into account the internal capacitors

A current will be integrated on a capacitor. The general approach is shown in Figure 4.1.a. Nevertheless we shouldn't forget that this capacitor will have to be discharged to ground, so we need a switch transistor to do this job, as shown in 4.1.b. When the transistor is on, all the current goes through this transistor to ground, and the capacitor is also discharged. Obviously this is chosen to be a nmos minimum size transistor, so our capacitor -whose size matters- isn't loaded with big parasitic capacitors, with unknown temperature behavior - besides minimum transistor means maximum speed, and we will need that speed when turning it off in order to re-start integrating. As it has already been pointed out, if the current isn't switched off (which could be an option), then all of it flows through the switch transistor and produces a substantial voltage across it, due to its on resistance, which voltage may also be temperature dependent.

Indeed, after simulations we see that this voltage changes over temperature a few tens of millivolts (starting from 27mV for 25oC, and moving from 10 to 45mV over our temperature range), which is huge deviation, considering our 1.8V supply. So something should be done in order not to force all this current through the switch, and that something should not be the source being switched on and off, because that would create a long starting time for the charging of the capacitor, and unknown transient effects. The solution is simple, and shown in 4.1.c. We add two switch transistors (M2, M3). When the capacitor is being discharged, the current is directed to ground through M3, while M1 will just take away the charge of the capacitor. Indeed this gave better results-the level of remaining offset and of the offset varying over temperature reduced by 5 orders of magnitude: now the remaining offset is on the level of 90nV, and deviating over temperature few tens of nV, which is really small and can be neglected.

Another unwanted effect was the remaining charge of all the switches. In 4.1.b it would have been only the charge of the small switch discharging the capacitor, but now the most important remaining charge is that trapped in the switch transistor that diverts the current to ground and must be turned off in order to allow the integration to begin.

To counteract all charge-related effects, two things could be done:

-one was to increase the current and the capacitor size, in order to reduce the effect of those residual charges. This is the reason why we decided to integrate on the capacitor 4uA (while we tended to use branches of 1uA) and use a capacitor of approximately 100pF. The capacitor of this size is large enough to be insensitive to charge injection and parasitic capacitances.

-the second thing has to do with the relative sizing of the transistors M2 and M3 of Figure 4.1.c. We started with the approach that the right one (M3) should be double size than the left one. See Figure 4.1.d: After the capacitor is discharged, we suppose internal capacitors c3,c4 fully discharged and c1,c2 fully charged. When integration starts again, we suspect capacitor c2 discharges to ground, while c1 injects its charge to the left side capacitors. But then we have one capacitor giving the initial charge to two capacitors (c3,c4). So we will start with double size. In simulations for room temperature this gave nice results, but since the relative size of the internal capacitors of each transistor is not equal (capacitors c3 and c1 were much larger than c4 and c2), then the over temperature effect was larger for keeping one transistor double size than keeping the same size. So we ended up with left and right transistors equal (for more simulations on that, see section 5.3.1)
analysis 1
The transistor is being used here as a switch, so the Vgs is huge, and it starts conducting with huge Vds across it. So we expect it to be in extreme saturation, and as a matter of fact we measure Cgs=32fF and Cgd<1fF, which can be made visible in Figure 4.2.

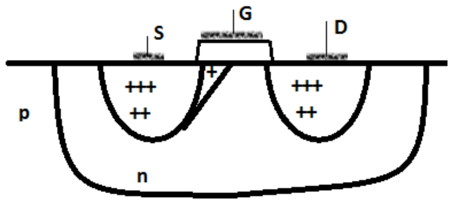


Figure 4.2: transistor incision showing the accumulation of charges in extreme saturation.

Indeed, after the voltage on the capacitor is rising, and Vds is suppressed, we measured that the capacitors start becoming equal (in the end Cgs=28fF, Cgd=18fF). For the moment when integration starts (which is the important moment), the right one has saved charge in the huge Cgs and needs to charge the huge Cgs of the left one. An equal transistor size is the solution.
end of analysis 1

These new switching transistors were chosen to be pmos since obviously the bottom part of them would be changing a lot over time since the voltage across the capacitor would rise. With pmos transistors this node is the drain that can change significantly while still keeping the transistor-switch on without addressing any issues (in case there would be nmos transistors in that position the voltage Vgs would be changing, something which is not wanted).

4.1.3 Threshold

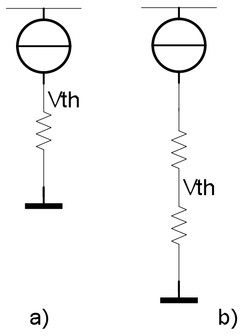


Figure 4.3: Creating the threshold a) with one resistor b) with two resistors

As shown in Figure 4.3.a the threshold will be created as the current being pushed through a resistor. Actually we decided to leave some extra resistor also on top of the point where the threshold is being measured (like in 4.3.b), so we can trim later for a higher/lower value.

4.1.4 Copying the current

Now we need to decide on transistor geometry and size, on the way to copy the current and on the way to create the current in the first place.

First matter is that the current mirror must be as good as possible, something that has already been discussed about its importance. In order to copy the current accurately, huge transistors will be used to mirror the current being produced (W=20um, L=6um), and we are going to equip them with cascode transistors, gain boosted with amplifiers. This way the resistance of the current source is as big as possible, and almost all the current is indeed delivered to the lower "needy" parts (threshold/integration). On top of that we run the DC MATCH simulation to check how similar the currents in the two right branches are, and after careful sizing, we got that the main contributors to mismatch are the current copying transistors (as expected) and the difference of the two currents is less than 0.035%. So we decided that the final look of that part is going to be like in Figure 4.4.

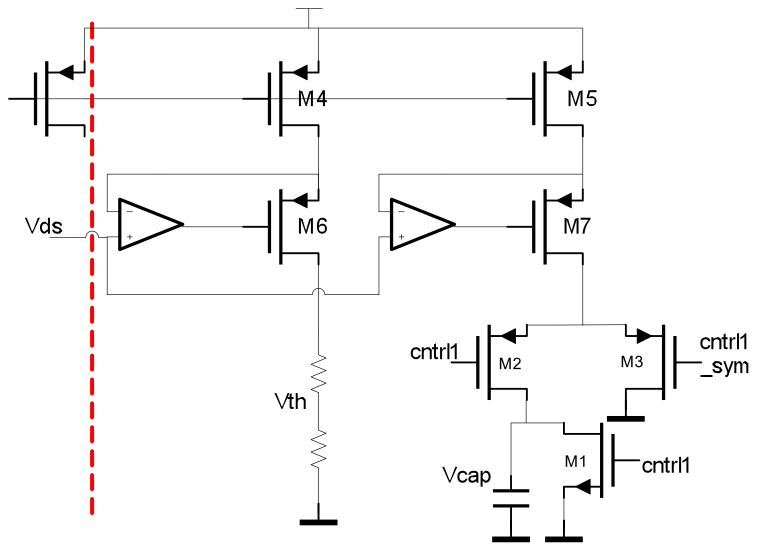


Figure 4.4: Final circuit for copying the current and creating the threshold and charging the capacitor

First comment we can make here is that for one branch there are three transistors from supply to ground (M5, M7, M2). In order for all of them to operate well, especially the two on top which need to be in saturation (room is needed for 2 overdrive voltages plus the "on" voltage of the pmos switch plus the rising voltage of the capacitor to fit in the supply), integration will take place until low voltages (approximately 0.9V maximum voltage) otherwise they are forced out of saturation$^{*\text{analysis } 2*}$. For the same reason (many transistors in the same branch and low supply) we chose to use the low threshold transistors of the technology (so instead of 0.5μm transistors that we had available we used 0.8μm ones). Unfortunately scaling down V_t to comply with the already scaled down Vdd, increases the leakage currents in OFF devices, [6] something that will be discussed further in Chapters 5 and 6 when we will want to turn off the oscillation function.

Moreover, we make an important remark: the current matching between the current that will be produced and the current actually used for charging the capacitor and creating the threshold is of no concern. Only the matching of these two currents is important, so they're the same and can be deleted from the equation of the formula of our oscillation period (or frequency).

Due to the fact that we were using huge transistors (for accurate matching and copying of the current) and low currents (for low current consumption) we ended up our main transistors that implement the current source to be operating in subthreshold region. That's not a problem in principle, apart from the fact that transistors in this region usually are not copying current that accurately. The best compensation to that matter is already considered, and is increasing the transistors size.

A final decision is the size of the cascode transistors. Theoretically they could be different size that the main current copying transistors, but since the amplifier could produce the correct level voltage for their biasing, we kept them the same size, even though that was huge.

The voltage that was created and was given as input to the gain boosting amplifiers (Vds in Figure 4.3) was 150mV lower than the supply: enough Vds to keep the transistors on top in saturation.

Analysis 2

We were noticing during our simulations the following undesirable effect: there was some deviation in the threshold level which was supposed to be completely flat while the slope was approaching the threshold, and at the same time some non constant current being integrated on the capacitor. That means we were getting the following result on the right of Figure 4.5, instead of the ideal one on the left (of course this Figure is excessive, in order to show the problem):

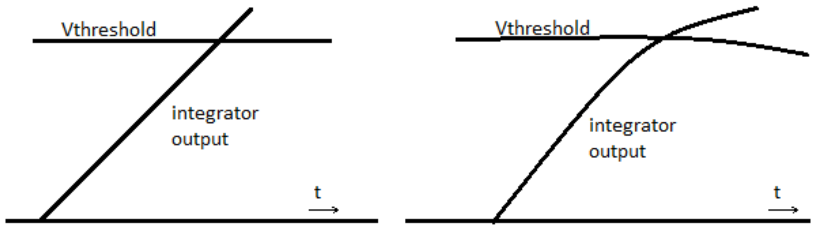


Figure 4.5: Results on threshold and ramp when pushing the current transistors towards an "out of saturation" condition.

After some analysis of this phenomenon, we found out that indeed it was happening because integration was taking place up to high (relatively) voltages on the capacitor, and the drain of the current source transistors on top of the capacitor (M5 in Figure 4.4) was pushed to higher voltages, reducing the current being delivered, and moreover through the internal capacitors of these transistors, the common Vg for the top current mirroring transistors was also being pushed a bit, forcing also the current on the "threshold" branch to be a bit less (over time) and therefore the threshold not to remain constant. Excessively (again) we can say that the current transistors were being pushed towards an "out of saturation" condition (not reaching it of course, but enough to be measurable with the simulator).

The transistor that "suffered" the most and therefore "transmitted" its problems to the rest is the cascode of the right branch (M7 in Figure 4.3)- which makes sense since its drain was the most heavily varying one. We measured the following deviations over time (as the integration was getting to higher voltages):

Matrix 5.1: Deviations when the integration goes to high voltages (transistor numbers correspond with Figure 4.4)

Vd of M2	1V (integration)		Changes:
Vd of M7	0.7V (almost the entire integration)	I on capacitor	2nA (≈0.1%)
Vg of M7	Few hundreds of μV	V threshold	Tens of uV (≈0.1%)
Vs of M7 (Vd of M5)	100 μV		
Vg of M5	Few μV		

These deviations unfortunately were also temperature dependent since they rely on non idealities of our transistors based on charge exchange and parasitic capacitors being charged. Therefore we see two new reasons for deviation from ideal behavior: the over time deviation, because the transistors are being pushed from the rising voltage on the capacitor, and the temperature dependency of those time deviations.

Simulating in order to check how much these affected our frequency (over temperature simulations), we can make the following comparison:

- With ideal current mirroring (ideal sources) all source of error was the remaining non matched temperature coefficients of the resistor and the capacitor, and in our case (cadence models) was giving a deviation of 2*3ns per period (2*100ppm)
- With realistic implementation (like in Figure 4.3) but using huge capacitors (like 1F) to keep voltages of nodes steady and not face the problem we're discussing here, we got deviations in the level of fV and pA, and over temperature 2*6ns per period (2*200ppm). The reason for that doubling of error will be explained below
- With all real implementation (exactly like Figure 4.3) we were getting again 2*6.2ns (2*200ppm approximately)

When we use the ideal huge capacitors to keep the node voltages stable, we still see that the over temperature deviation doubles. We found that the voltages of several nodes may be kept practically stable, but by using huge transistors for better matching, huge internal capacitors were inserted in our circuit that still need to be charged. So still some currents are lost on gate and body (bulk) while charging those capacitors (and/or leaking), and therefore these currents cause that effect to double. In fact the current source delivers a current stable in the order of fA, but we end up integrating on the capacitor a current that deviates a few nA, due to those small currents (and check that 2nA was the "normal" deviation of the current being integrated on the capacitor).

On the other hand, when we use all real implementation, we see that the deviation remains the same. The reason for that is that the two deviations as shown in the Figure above in the analysis, cancel each other, and period remains –almost- constant like before. Of course one non ideality canceling another one is not something we can depend on.

On silicon level all these effects can be explained as originating from currents charging the internal capacitors, gate leakage, weaker field between drain implant area and source implant area that allows more electrons and/or holes to "escape" towards the gate and bulk, accompanied with mobility deceleration due to lower gate voltage, that will make the electrons scatter more.

As a solution integration up to relatively low(er) voltages had to be implemented, because all these effects were becoming more aggressive the higher the voltages from current integration on the capacitor were becoming, but there is a practical limit to how low voltages could get. Another solution would have been smaller transistors, and therefore with higher internal "impedances" (and smaller internal capacitors) but then matching was not good enough.

Comment 1: To a small extent some deviation remains, due to the voltage coefficient of the capacitor. Almost negligible, but it's another source of non-ideality.
Comment 2: The capacitors in the output and input of the amplifier used for gain boosting, were attached to our crucial transistors and were contributing to the above analyzed error by their small charging currents. Another small but present source of errors.
end of Analysis 2

The amplifiers used for the gain boost were designed like in the Figure 4.6:

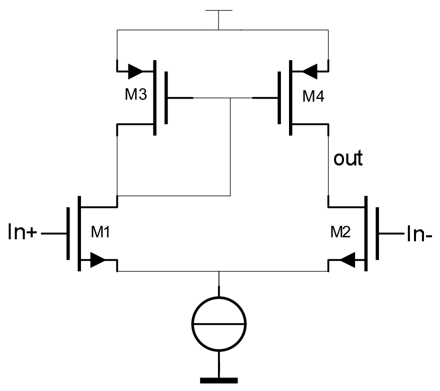


Figure 4.6: The amplifier implementing the gain boost

As we see, nothing more is needed than just a differential pair with pmos load mirror. The reason for which we chose nmos input, is because the voltages being compared are practically very high (approximately 1.65 if a headroom of 150mV is left for overdrive voltage of our current mirroring transistors), so pmos input transistors would be off.

Final choice is the current source that will create the current will be mirrored to create the threshold and charge the capacitor.

We used a simple PTAT (with mos devices) current source, as can be seen in the center of Figure 4.7. It was designed such that each branch gives approximately 1uA to be mirrored further. Apart from the source (consisting of the resistor R, one pair of pmos (M1, M2) and one of nmos transistors(M3, M4)) we see some extra network surrounding it.

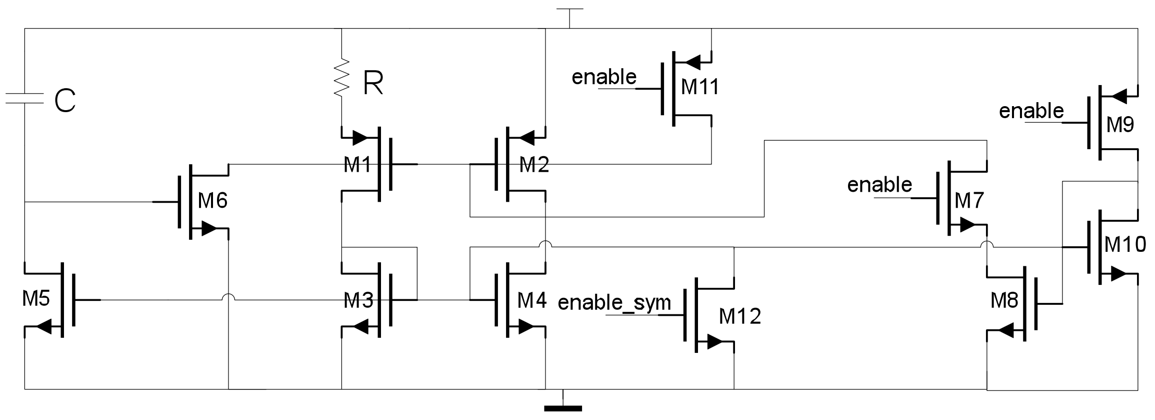


Figure 4.7: Our current source

One group (on the right) makes sure the source will shut down if we want to pause the operation of the clock, and also makes sure that it will restart later on. With the enable signal (and the symmetrical signal) we will be able to “turn” M11 and M12 on at the same time, pull the gate node of the two nmos’s to ground and the gate node of the two pmos’s of the current source to supply, and therefore shut down the current source, and therefore the whole circuit that is being mirrored from here.

Apart from those, we have to make sure that when we want the clock to restart, the current source will be able to restart as well. That is what all the rest of the four transistors (on the right, M7-M10) take care of. While everything is off, transistor M7 is off (gate low), M9 is on (gate “high”) so in the drain it has Vdd, and M8 is on as well (not conducting of course since the branch is broken). M10 is off as well. The moment we de-assert the enable signal, then for one instance, M9 is off and M7 is on. Since the current source doesn’t start operating instantly, M10 will remain off for sometime, keeping the gate of M8 high. So the branch of M7 and M8 is conducting, forcing the right branch of the current source to conduct some current and start operating. When this will happen the drain node in the right branch of the current source (drain of M2, M4) will eventually drop its voltage, and the left one start increasing its own (drain of M1, M3). Eventually after the current source is operating, gate of M10 is high, but M9 is not conducting, gate of M8 is low, so even though gate of M7 is high, neither of the two branches are conducting.

The rest of the surrounding circuitry (on the left of the current source) makes sure that the current source will start on startup (power up). When supply rises from 0 to Vdd, the other end of the capacitor instantly also goes from 0 to Vdd. Therefore transistor M6 is turned on. That way it makes it possible for a current to flow that will make the current source start conducting. When it will, gate of M5 will be high, bringing ground both to the gate of M6 shutting the transistor down, and to the bottom node of the capacitor. Again nothing is conducting in normal operation.

These were two really low (negligible) normal operation power consuming networks and pretty robust that help operation on power on, and on pausing and restarting the current source.

4.2 Decision point 1 = logic gates, Switches

The output of the comparator as seen in Figure 3.21, should go quickly to the decision point(s) 1, where it will be decided (fast) for one integrator to start integrating, but the other decision point 1 should also decide (slowly - delayed) for the integrator that ignited the change of comparator status, to discharge the capacitor. Therefore, we should use an as fast logic gate as possible, since those logic parts alternately should react in the fast timing path, and the delay will be taken care of by the state memory.

We chose to implement this logic function with an AND/NAND gate, as shown in Figure 4.8.

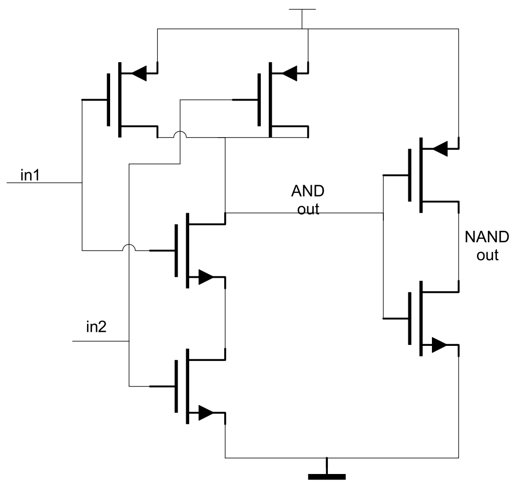


Figure 4.8: the "decision point 1" logic function implementation

As we can see, we need both symmetrical outputs, so we can start integration (see the transistors M1, M2, M3 in Figure 4.3). When one switch transistor shuts down from conducting the current, the other one should start conducting it so it'll be integrated on the capacitor, and the transistor that was discharging the capacitor also needs to shut down at the same time.

We have to design also the switches in front of the comparator. Those switches are choosing which one of the available inputs is going to be hooked up to the comparator and be compared. It is simply as shown in Figure 4.9.

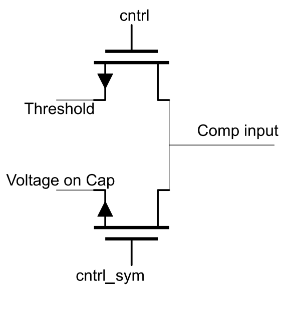


Figure 4.9: implementation of the switches

Obviously we will need for that again opposite control signals (high for one, low for the other).

These switches do indeed move some charges, that produce a small “jump” in the integrated value (in the order of tens of uV). That fades out since it happens early before crossing the threshold. Moreover this is almost stable over temperature, so it won’t really affect our oscillator’s behavior.

4.3 Memory

Memory is crucial for the general operation of the oscillator, but not for the quick timing path. So everything was designed with care for power and functionality, and not for timing accuracy. For the operation of our oscillator we have to create the following stream of signals: starting with two inputs (the outputs of the comparator) a delayed version of them will be created, which will help the system to keep integrating more (and avoiding the battle of loops as analyzed in Chapter 3), and also change the input switches of the comparator in non-timing-crucial moments.

Before seeing in more details the signals, it needs to be made clear that we designed the system so that the crucial timing moments are the FALLING EDGES of the comparator, since we used “AND” gates as control for decision points 1. That means that when the comparator says “0” then we want to read “0” fast (AND gate changing its output to 0), but when comparator says “1”, we want integration to keep up a bit longer for all the reasons analyzed so far (so the other input must be kept 0 a little while longer before both become 1 and AND gives output high).

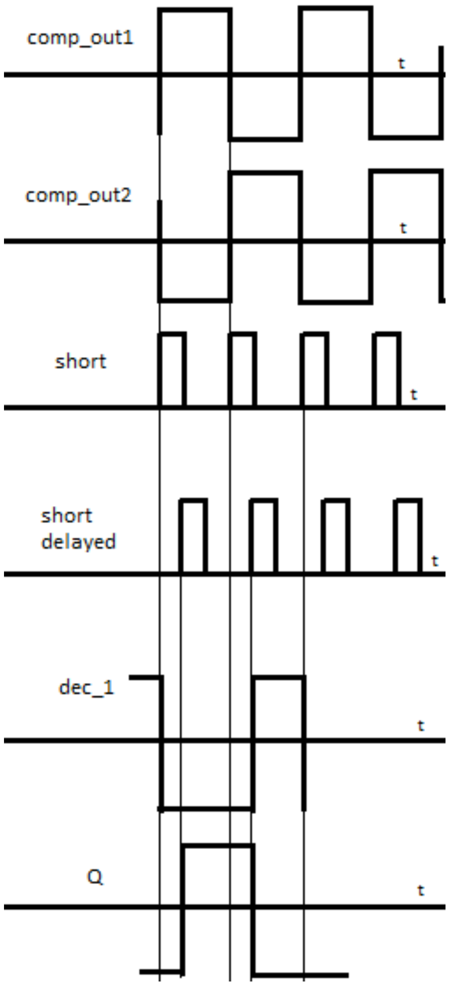


Figure 4.10: Signals in and out of the memory

Inputs of the memory block are the two comparator outputs, which we consider to be correct in timing and symmetrical (see for this analysis Figure 4.10). Starting from these, we need to create the rest of the signals. A short version of our pulses will be created, so we will control an RS flip flop without running the risk that both R and S may be "1" at the same time (signals "short"). Then these signals will be delayed (signals "delayed short"). Then, suppose we care about the comp_out_2: it gives a fast 0, so decision point 1 (the AND gate) needs to deliver a quick zero. Thanks to the change of comp_out_2, the fast zero will be created. With the proper connections, one signal "short" will control the RS FF so the other input of the AND will also become 0, so AND will have both inputs zero-if the second one is a bit delayed doesn't matter, because the fast 0 will have been taken care of by the comp_out_2 itsself. And here is the trick: when comp_out_2 gives a 1, the other input of the RS FF is controlled with a short delayed, so the second input of the AND gate gets 1 a bit delayed, so integration will have taken place for a bit longer that just reaching the crossing point, since AND will have given 0 as output a bit longer. Therefore two inputs of decision point 1 (AND gate) are comp_out_2 and dec_1 (the output of memory).

Finally with a bit different delay we want a symmetrical signal to control the interchange of the switches. That is shown as Q in Figure 4.10.

Since the delay we want to insert is significant-with a period of 32usec approximately, half period of 16usec, we don't mind if the capacitor will be integrating more for approximately 1usec, which cannot be considered as the internal delay of any digital block. So the signals will have to be delayed intentionally. For that reason a delay block was created like shown in Figure 4.11.

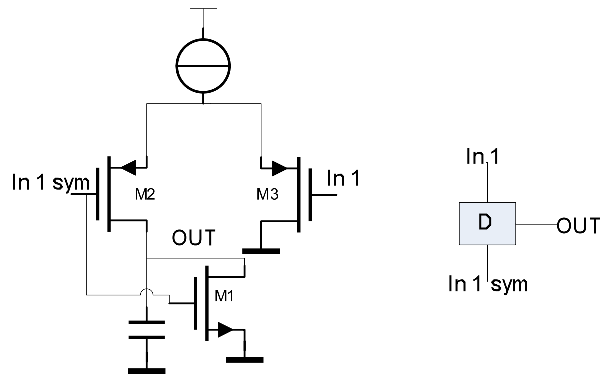


Figure 4.11: Delay block, implementation and block representation

This is based on the fact that an inverter following the "OUT" will not change state unless the voltage has risen above some specific value (threshold), the exact height of which is of no importance to the operation (even if it changes over temperature), since the timing of this block is not in the fast timing loop. So if In_1 turns high and this has to propagate, then OUT will read a delayed high (on low, it reads almost immediately, as fast as the transistor M1 is discharging the capacitor). In this topology a really low current can be integrated on the capacitor. Therefore the power consumption is really small, and the capacitor size is also small (important for the layout).

Using that delay unit (boxes marked with a D), we design the memory block as in Figure 4.12.

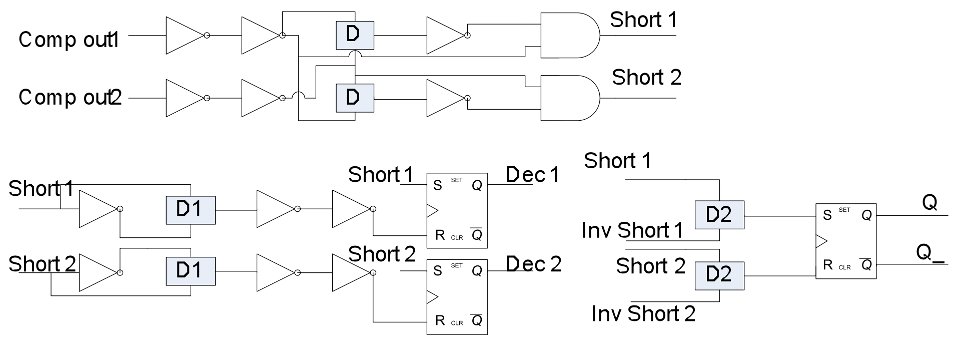


Figure 4.12: memory diagram

Here we see a perfect correspondence with the Figure 4.10: First thing we did was to use two inverters in dealing with the input signal (the two outputs of the comparator) so the following circuitry doesn't load these comparator outputs too much. Then the comparator outputs are delayed and with an AND gate the short signals are created. Then these are delayed a bit, and these signals control one input of the RS FF while the other is controlled by the non delayed short signals. Therefore we create dec_1 and dec_2. With a different delay block signals Q and Q_ are created, which will change the position of the switches in front of the comparator.

4.4 Comparator design

The comparator is not going to be a one stage design, but it will comprise a number of different stages. So the analysis will be made following the needs and purpose of each of these stages.

-General idea about a comparator

A comparator as its name implies is a block that is designed to compare two signals and produce a binary level at the output, a 1 (high supply) or 0 (low supply) depending on this exact polarity, i.e. which of the inputs is "larger" than the other. The output has to be a binary state so the comparator helps real world signals to interface digital circuitry, while the design of it can merely be considered as a cascade of amplifiers.
Issues that concern a comparator design and our choices:

- *Positive Feedback*: In most cases comparators that use positive feedback are being used, because that way the circuit can be non conducting in non decisive moments. But common practice is that there is a fast clock deciding when the inputs should be compared. We had to avoid this idea, since we are building a clock, and cannot rely on another clock, even though this type of comparators is much faster and more power efficient than the multi-stage linear amplification that finally we will use [17]. Moreover if we wanted that type of comparator (with positive feedback) to operate without a clock we would have to add significant hysteresis but then again we don't want that either (will be discussed further below).
- *Inputs*: Comparators can be designed for input signals changing rail to rail. Our inputs will change, but not rail to rail, just a few hundreds of mVs.

- *Opamp as comparator*: Another decision we made for our implementation, was to design a specific-for-the-application comparator and not use an opamp as a comparator, because opamps usually have a long recovery time from saturation (due to the presence of the compensation capacitor) and the power consumption would be questionable when used in such specific operation.
- *Hysteresis:* A comparator usually is designed to have some hysteresis. This implies that the actual change (at the output) doesn't happen exactly when the inputs are equal, but a bit later, so we're sure we avoid wrong misinterpretations of the inputs due to noise or other undesired small effects. On the other hand this implies that the comparator cannot resolve signals within the hysteresis band. This is not desired for our design, and no hysteresis was included in our comparator, in order to be really sensitive on the moment of crossing the threshold, and not be based on some extra parameter (the implementation of the hysteresis) and therefore ideas like a Schmitt trigger had to be abandoned. Our inputs signals were nevertheless slow varying and we wanted our comparator to be sensitive to them and ready to switch state all the time.
- *Speed and power tradeoff* applies also in comparator design. We have limited power to spend (less than 70uA) and therefore we expect to trade for a not-so fast comparator, but that is yet to be determined.
- *Offset* is usually a huge problem with comparator design, and several ways to compensate for that exist (auto-zeroing and chopping being the most prestigious ones) but as discussed in the previous Chapters this is of no concern to our topology.
- Last thing that remains to be addressed in our comparator design, is the *propagation delay* within the comparator, since it will be added to our fast timing loop.

<u>-Input stage</u>

The main purpose of the input stage is to take the input signals which are not symmetrical and to deliver a symmetrical output signal, so that it can be amplified correctly and have the transition as fast as possible in the output. Moreover this input stage has to operate with high current, and should be conducting that current all the time. By doing that the inputs will not be affected by any charge injection from the output stages where we want a fast and sudden change rail to rail. That change in the output usually moves around a lot of charge, due to the large currents that are present, and through the parasitic gate-source and gate-drain capacitors could be coupled to the inputs and increase phase noise. In some cases there is a special preamplifier implemented mainly for this reason (reducing the kick back noise, [15],[17]). We will use for that reason as our input stage/preamplifier a stage similar to the input stage of the static latched comparators [15]. It also has to be dc level insensitive (to some extent) and with constant gm [14]. Finally the transistors in the input stage must be in saturation all the time, so they can resolve accurately and fast the slowly changing inputs, and for another implicit reason: If transistors enter the linear region, the parasitic Cgs's of the input pair change significantly, especially over temperature, and that is a situation that should be avoided, because it would affect the delay of the comparator if significantly different parasitic capacitors would have to be charged.

As we have already discussed our topology is offset immune, and therefore the mismatch of the input differential pair is of no concern to us.

So what we designed as input stage the one depicted in Figure 4.13.

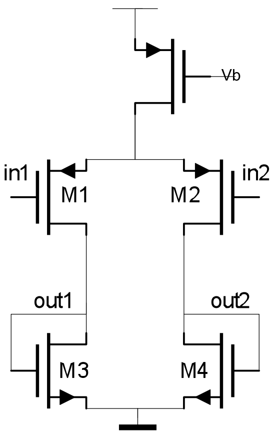


Figure 4.13: input stage of the comparator

Here indeed we compromised: the gain of the first stage was just 3dB. But it was conducting all the time, none of the transistors was leaving saturation, and the delay added by this stage was in the order of 3nsecs.

-Main stage

Here we want a lot of gain, so that the input small symmetrical signal will be amplified and start delivering a rail to rail change in the output. In order to use a lot of current, but control it, we designed the stage shown in Figure 4.14

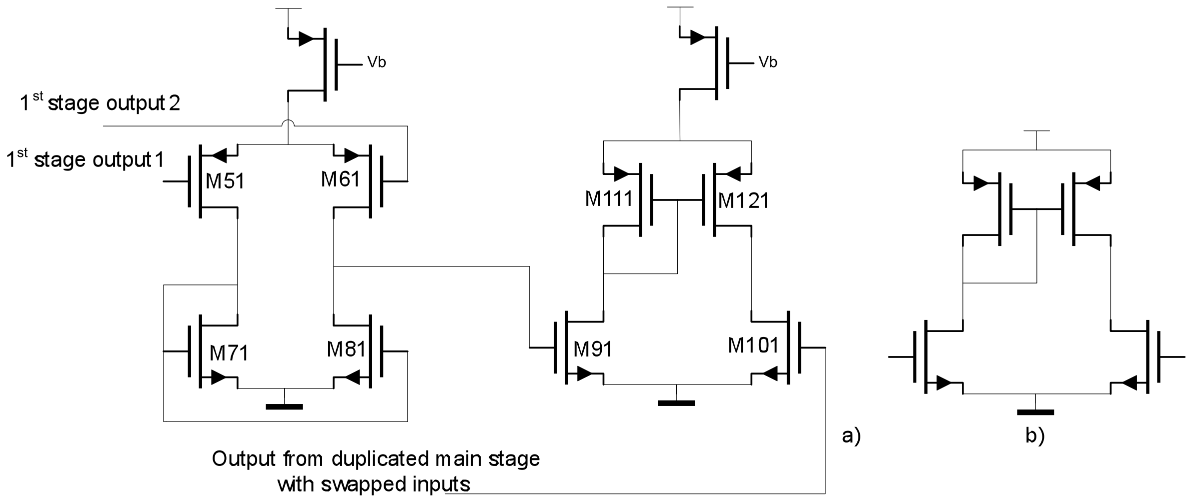


Figure 4.14: a-main stage of the comparator, b-design of stage that we didn't use

Stage on 4.14.a, left: it has the advantage that the left branch is always conducting (transistors M51, M71), even if the right branch (transistors M6, M81) is delivering a high or a low voltage to the next stage, and is almost off. That way it can recover faster, because the current source is never turned off, and can quickly deliver the new "2nd stage output".

Stage on 4.14a, right: Designed as shown in 4.14.a (right) and not in 4.14.b, because current consumption in 4.14.b had extreme spikes. For that reason the current source on top have been added, in order to control the current. The gain was still very high, and therefore a pretty sharp output was delivered, but not rail to rail, because it was limited by the current source. The signal delivered was 1.3V to 0V. That’s why an extra output stage was added, which will be discussed further in the following section, in order to get a clear two-level-output. Nevertheless we noticed that it responded faster when the “rail to rail” output demand was moved to another stage (along with the charging of the load), instead of keeping it in this high gain stage.

Measuring the gain, the main stage delivers 70dB of gain (both parts together). The low dependency on temperature is an outcome of the high gain, not of any other reason. High gain means sharp changing, which means low dependency on temperature. The delay added by the main stage is in the order of 30nsec, most of which by the stage on the right, in which both “branches” saturate, and needed time to recover.

Unfortunately the non symmetrical topology (here in the main stages) can produce an offset in the two paths, that cannot be compensated by the topology. Careful layout will be our way to reduce that effect.

-Output stage

Purpose of the output stage is just to provide a fast rail to rail signal (push pull type), so we designed it as just an inverter. This idea, used in many cases ([15], [16], [18]) enhances the slew rate of the comparator [16].

Thanks to this output stage we take advantage of the fast change of the main stage’s output (from 1.3V to gnd) and transform it into rail to rail change. That way the load is out of the “fast core” of our comparator, so it responds faster than without having this inverter. This output stage offered no gain, and added a few nsecs of delay, strongly dependant on the load being charged.

Overall comparator

Our comparator appears to have a delay in the order of 49nsec, while burning approximately 70uA, which power speed compromise is according to common practice (for a propagation delay of a few ns the power being consumed usually is in the order of a few to hundreds of mWs [14], while for a propagation delay of hundreds of ns, the current could be dropped to a few μA).

About the *over temperature deviation of the delay* in the comparator, we simulated that for lower temperature the comparator had slightly more gain, so the transition is sharper, and for higher voltages we have higher –and more clear- input voltages, and the change happens a bit faster. These two non-idealities come close and cancel each other to some extent, because we will have low voltages for low temperatures and high voltages for high temperatures. We cannot rely on that of course, this is just an observation, but on the other hand there was nothing else that could be done that wouldn’t move the accuracy to a new part of the design (a new loop could be included, measuring (how well) some quantity of the circuit (how stable quantity) and try to keep the delay of the comparator more stable over temperature, but the issue that the accuracy would just move somewhere else, remains. More details in recommendations, section 7.3).

As we simulated over temperature, in all corners, the delay added by the comparator was deviating only 2 nsecs in every case from maximum (85oC) to minimum temperature (-30oC) (for half period), over the delay of 49nsecs (for half a period, once through the comparator). Moreover from

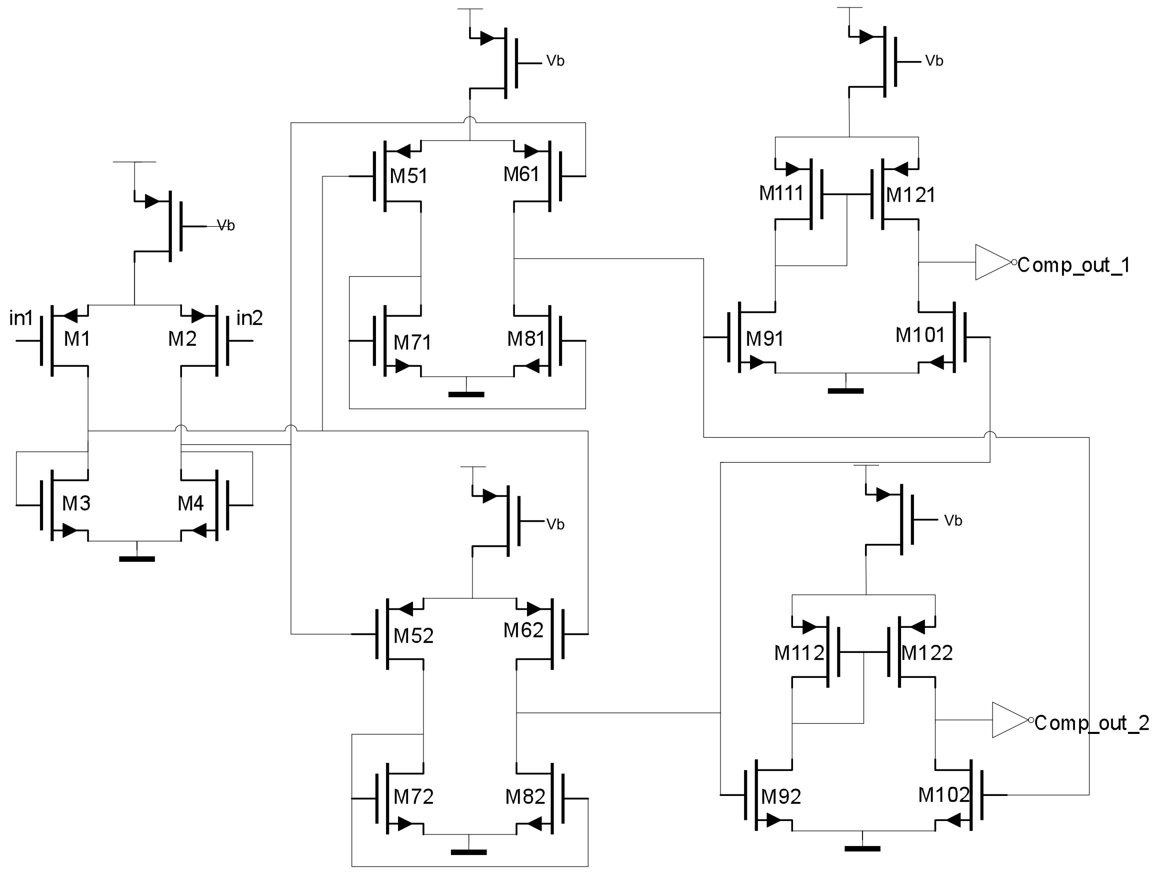


Figure 4.15: the whole comparator topology

simulations, the offset of the input stage was indeed of no concern thanks to the topology, but unfortunately the offset of the rest of the differential pairs could produce some issues since it was creating some unbalance in the two "paths" of the signal.

Chapter 5
Final decisions and Layout generation

*In the whole Chapter, when referring to deviations of frequency over temperature, it's always how much the low temperatures result is different than the high temperature result. If there is deviation -5nsec means that at -30oC the period is 5 nsecs shorter than the period at 85oC. This holds because the temperature behavior is linear, so the total deviation over temperature corresponds to the deviation between the maximum temperature differences.
**In the whole Chapter, and for the rest of the essay, for all the Graphs that will appear, when the horizontal axis shows % deviation of the frequency over temperature, the number on the axis (for example -2.3%) represents the whole space between this value and 0.1% closer to zero (in this example between -2.3 and -2.2%). That is why there is a vertical value for -0.1% and one for 0.1% and none for 0%. "-0.1%" represents space [-0.1,0) and "0.1%" the space [0, 0.1). Another small detail is that I used windows excel for the Graphs in the Greek language, so "." is ",". So 3.5% is represented as 3,5%. But this will be minor when reading the Graphs.

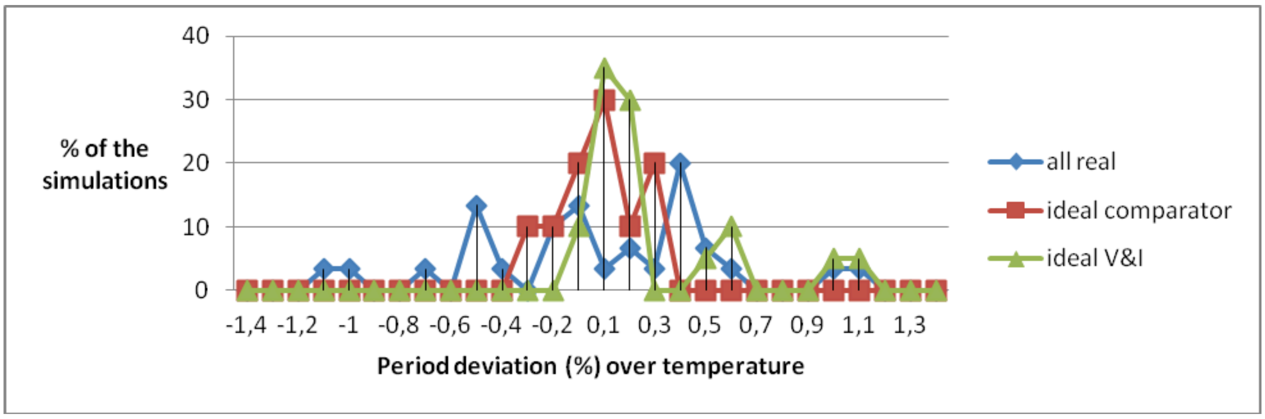
5.1 Remaining issues and choices

First thing we did was to run some simulations with ideal comparator, ideal production of threshold and charging of the capacitor (with ideal current sources, keeping only the temperature coefficient mismatch of the resistor and the capacitor) and with both of them ideal, in order to have an estimate of what to expect. The results were the following:

- Both ideal comparator and ideal threshold/charging system: We had after 30 monte carlo repetitions a steady deviation +19nsec, where if we take out the remaining mismatch of the temperature coefficients of the resistor and the capacitor (-2ppm/oC, causing +6nsec per half period=+12nsec in total), we have +7/28500=+0.025%. This was linear, stable and the same for all simulations, which means that non idealities of the AND gate and the switches give a frequency deviation, but pretty predictable and small.
- Ideal comparator or Ideal threshold/charging system: As seen in Graph 5.1 and Matrix 5.1, none of the two can be blamed more than the other, since both of them deviate almost the same, and deviate significantly.

Matrix 5.1: Interpretation of the monte carlo results shown in Graph 5.1

	Deviation
Ideal comparator=Blaming the threshold and the currents	100% of the simulations were in the window of ±0.3% deviation
Ideal threshold &currents=blaming the comparator	75% of the simulations had less than ±0.3% deviation, while 90% less than ±0.6%



Graph 5.1: monte carlo results for frequency deviation when the topology is realized with all real components, with ideal only the comparator, and with ideal only the threshold/current block

Copying the Matrix 3.3 from Chapter 3 in order to see if this analysis complies with our theoretical expectations, we have the new Matrix 5.2.

Matrix 5.2: How the implementation affects delays and offsets of the several blocks of our design

Imperfections list (remaining in our design)			
block (also see Figure 3.21)	Offset	Delay	Comments, remaining amount, deviation over temperature
Comparator	No	Yes	Comparator delay, approx 50nsec, varying 4nsec *[1]
Decision points 1	No	Yes	Delay through the decision point 1, the AND/NAND gate, is small and deviates very low over temperature (part of +19nsec)
State Memory	No	No	Memory is now out of the timing loop.
Zeroing the integrator	Yes (not anymore)	No	The ground on the capacitor is almost perfect (few nV) so we consider that as not important.
Switches	Yes	No	These switches (choosing what is being compared) have a charge injection, and this is analyzed later *[2]
Threshold	Yes	No	Two issues remain while producing the threshold and charging the capacitor. One is the non ideal copying resulting in issues, and the other is the charge from the switches on top of the capacitor. *[3] The delay in start to integrate is part of the +19nsec (more details in *[2])
Value to be integrated-Integrator	Yes	Maybe	

The remaining sources of non ideality are the following:

*[1]: **Comparator delay.** In ideal case the comparator delay is approximately 50nsec, and changing 4nsec over temperature. Keeping in mind that it's not built completely symmetrical, there can be different delays in each of the paths of the signal, ending up in different behavior over temperature for the two outputs. It's important to notice that as temperature increases, mobility decreases, and

the delay of any amplifier stage is getting bigger, so (overall) the delay through the comparator increases for higher temperatures [9].

Comparator will be responsible for (at least) a 0.02% of the change of the frequency. Which can easily multiply to larger values, as seen in the monte carlo simulation, where it can be responsible for up to ±0.3% deviations over temperature, and if we want to include more than 90% of the results, we should hold the comparator responsible for deviations up to ±0.6%.

*[2]: **Switches deciding what is being compared.** In simulations an interesting effect was noticed.

In ideal case (ideal threshold/capacitor and ideal comparator) 15nsec period deviation was being read and in the all real case 12 nsec, but there was significant difference in those two results, because in all ideal case deviation was +15nsec, while for all real -12nsec. This was brought down to the parts shown in Matrix 5.3 after careful simulations.

Matrix 5.3: Analyzing which blocks are responsible and how much for over temperature deviations of the period in three specific cases

	Inherited [*a]	AND[*b]	Read the signal[*b]	Comparator	Total deviation
All ideal	+7nsec	4nsec	3nsec	0	+14 nsec
Real comparator	+7nsec	3nsec	3nsec	3 nsec	+16 nsec
All real	-22nsec (?)	3nsec	3nsec	6 nsec	-12 nsec

*a: inherited is what's left from the different temperature coefficients of the resistor and the capacitor, which is already causing this deviation over temperature
*b: parts of the +7nsec that the surrounding circuitry causes even with ideal comparator and threshold. Indeed it's 6 or 7nsec here in our simulations. It's mainly the AND gate and how fast we read the signals switching and indeed start integrating.

In Matrix 5.3 we notice that in order for the -12nsec to be correct, we need an extra -22nsec, caused by some unknown (so far) reason.

We managed to measure this -22nsec. First thing we did was to break the loop of Figure 3.21. Keeping the switches in front of the comparator in one positions without swapping them, we measured +12nsec, which means that the part that creates the threshold and charges the capacitor does not add any more over temperature deviation on its own (inherited +7nsec and the constant +6nsec, approximately, since there is no delay to be read). Then we swapped them ideally with a source, and not with the output of the comparator, and then we measured -22nsec total, which hides the +12 without swapping the switches (or the ideal case if you prefer, +7nsec inherited+6nsec comparator) -36nsec from this unknown reason. First thing we can notice is that when everything is implemented with real components we probably underestimated the "delay to read the change" because it must be +10nsec in order to give an overall result of -12nsec.

The important question to be answered is where this amount of period/frequency deviation is coming from. The only change in the circuit was the swapping of the switches. So we cannot blame the kickback noise (charge) from the comparator, or the block that is copying the current (for example the switches deciding if the capacitor is being charged or discharged). We can only assume that this is charge being moved from the swapping of the switches in front of the comparator.

Verification: manipulating the temperature coefficient of R from -27ppm to -16ppm, we measured -72nsec which is -42nsec without changing the position (so that is the constant delays plus the new mismatch of the temperature coefficients)-36nsec step (almost) which indeed remains the same.

*[3]: **Threshold production and capacitor charging.** One important non ideality in this block is the one for which we already compromised (§4.1.4, analysis 2) to integrate up to relatively low voltages, because the rising voltage on the capacitor was "mirrored" back to our current sources and we were having a non stable current both being integrated on the capacitor and creating the threshold.

The second –very- important aspect is the charges being exchanged, while the switches on top of the capacitor divert the current to ground or on the capacitor (see Figure 5.1 below). They play a role in both the results we get from non ideal charging the capacitor/creating the threshold, and the small delay noticed before (in *[1]) called "delay to start integrating". Here the size of the two transistors was measured with monte carlo simulations, since they played a very important role in the over temperature behavior, and we had the results shown in Graph 5.2.

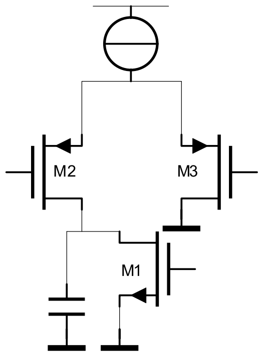
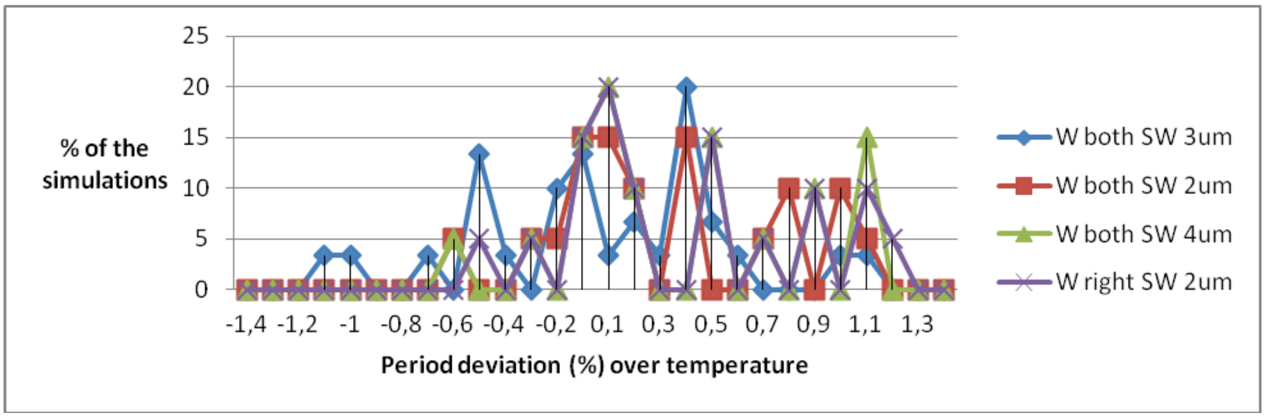


Figure 5.1: Reminder of the system charging/discharging the capacitor

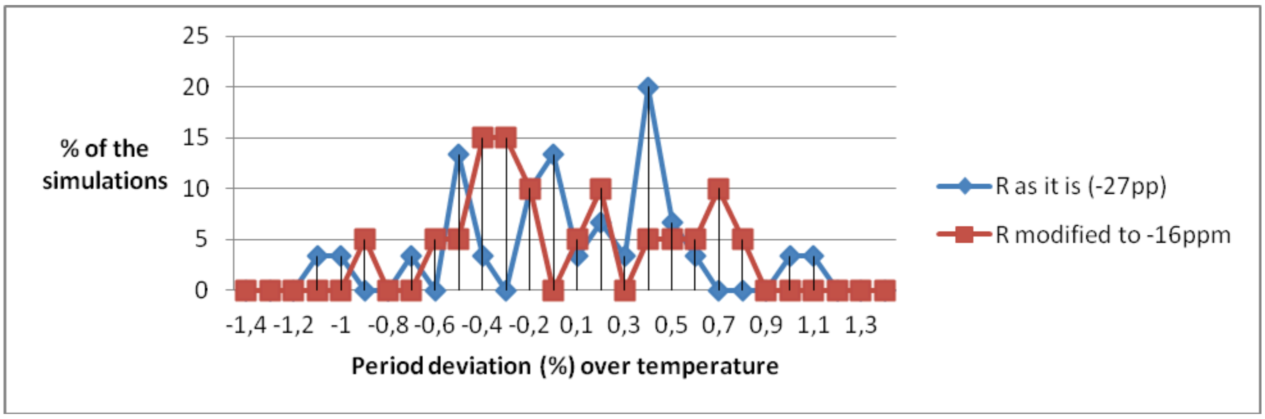


Graph 5.2: monte carlo results for different sizes of the two transistors M2, M3 of Figure 5.1

During the simulations we were changing the width size of the switches, with a starting point of 3um both, and keeping all lenghts in 3um.

Here we notice that with both smaller (2um) or larger (4um) size, or even with unequal sizing (only the width of the right transistor switch 2um) we get no improvement, because the slightly more results gathered around ±.1% are “compensated” with larger spikes in larger deviation (especially between +0.7 and +1.1 %).
None of them seemed significantly better than the one we started with, the one with same width 3um for both of the transistors/switches. That size is the one we kept in the end.

Considering all the above mentioned non idealities, we run the following monte carlo simulations, after changing manually the temperature coefficient of the resistor, with the results shown in Graph 5.3.



Graph 5.3: Results of implementations with all components real with resistor’s temperature coefficient both -16ppm and (the original value) -27ppm

We see as expected the total spread shifted a bit to higher frequency deviation over temperature, since now the initial matching of the temperature coefficients of the resistor and the capacitor do not match that well.
In fact we know now that we have a mismatch of the temperature coefficients in the order of 10ppm/oC so 10 times bigger RC spread. RC spread of the models in cadence was responsible for 6nsec deviation, or 0.02%.Now it changed to 0.2%. Indeed after calculating the average deviation, from 0.45% it reached 0.58% (calculating the absolute deviation over temperature). The rest of the reasons for inaccuracy (comparator, gate AND etc) remained the same, that’s why we see only this small shift to higher deviations over temperature.

Summing up, we create the Matrix 5.4.
If we add up all our expectations, we expect from our chips when they come back from the factory, most of them (more than 80% according to monte carlo simulations) to have a deviation of less than 1%.

5.2 Output stage and input choices

The comparator output is going to be considered as the output signal. Reasonably the comparator operation shouldn’t be loaded with our external load, so a series of inverters was

Matrix 5.4: How much each block is responsible for frequency drifts over temperature deviations

Target deviation over temp	0.01%
Origin of distortion	% deviation
RC spread	Up to 0.15%
Errors in charging the capacitor/ creating the V_{th}	Up to 0.3%
Comparator delay spread	Up to 0.3/0.6%
Switches and AND gate	≈0.1%
TOTAL	Up to 0.8%

designed, in order to be able to handle our maximum output load (12p). So the output shown in Figure 5.2 was created:

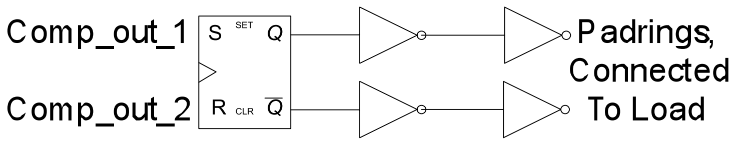


Figure 5.2: The output stage, where the output load will be connected.

The two inverters were sized (*1 and *7) in order to be able to charge the 12pF in less than 100ns. For more separation between the comparator output and the load, an RS flip flop was inserted, being controlled by the outputs of the comparator, so the comparator does not interact at all with the load.

As far as inputs are concerned, we had to include two inputs to our design. One was designed for the POR pulse whose job is to set the digital parts (memory, flip flops etc) before the actual operation starts. It takes care that on power up the digital part of our circuit sets in a condition so the oscillation can safely start.

The second input has to do with the enable operation (reminder: enable is an external signal that will shut down the whole circuit so when not needed it wouldn't consume power). We already saw (§4.1, Figure 4.7) that in order to implement this operation the current source is shut down, so theoretically, since everything is mirrored from this current source, everything should also stop conducting. Nevertheless, the current sources of the comparator that conduct a lot of current were enhanced with an extra transistor that shorts their gate to supply so we're sure these huge transistors do not conduct, and also just before the inverters in the output stage of the comparator, so all the digital part that follows (memory etc) is fixed to a value and there is no "parasitic" oscillation.

5.3 Issues not addressed

The most important thing that was not addressed, was trimming. Trimming could be easily implemented, by breaking the resistor that creates the threshold (as in Figure 4.3) into a ladder of small resistors, and therefore with just one measurement at room temperature a different threshold could be chosen in order to correct the frequency.

The second thing that we didn't address was to implement the POR pulse internally on chip, so we get rid of this one extra input. There are circuits predesigned for this reason, so it's mainly a procedure of combining standard cell elements, and fixing them to fit in our case and our 1.8V supply.

5.4 Layout

Every part of the design mentioned so far was laid out, and then the ESD protection ring was designed and laid out as well, and the chip was ready to be sent to the factory. The full layout can be seen in Figure 5.4.

The ESD protection ring is mainly composed of huge transistors short circuiting the supplies when there is an ESD event on the main chip, so the extra current is guided through those transistors and not through the main chip. This way the main chip is protected from destruction. One protection cell was designed for every bond pad. A so called "RC timer" had to be included, in order to decide when actually something happens, in order to turn the huge transistors (called "Merrill Clamps") on.

Things worth to mention about the layout are the following:

- Where matching of transistors was important, dummy transistors were added to reduce the effects of how the surroundings affected the operation of these crucial transistors.
- Where copying of currents was important, and a transistor would be diode connected in order to "produce" the gate voltage for copying that current to other branches, we preferred to bring with a long line the current of this diode connected transistor - and the transistor itself - closer to the crucial transistors. On one hand this made the gate voltage being generated more resistant to interference, thanks to the proximity to the crucial transistors, on the other hand the large parasitic capacitor was not avoided because the long line was still on the layout. But that was of less concern since the long line carried now a current and not the voltage. Electrically of course both ideas are identical, but in real life, taking into account the resistance of metal lines on chip and proximities and protection rings surrounding the wells, this was a better choice.

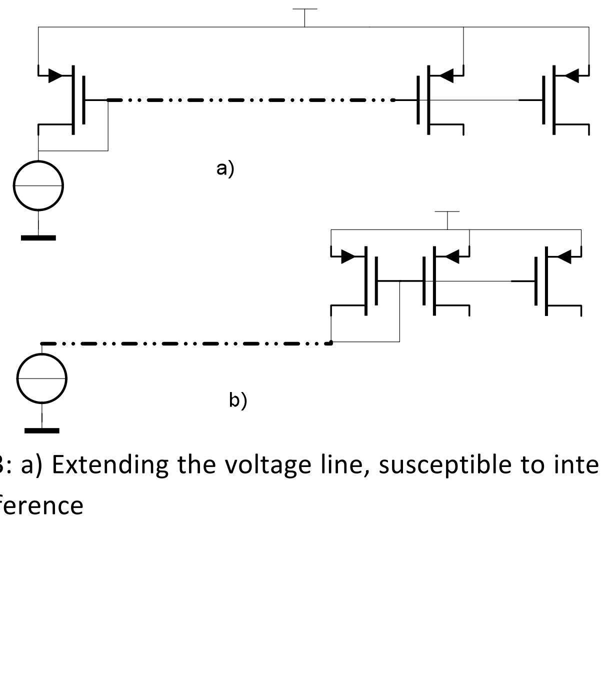


Figure 5.3: a) Extending the voltage line, susceptible to interference b) Extending the current line for less interference

- The large size of the capacitors (almost half of the total area in layout) is due to the low frequency: A rise of 1 V is supposed to happen in about 15usec and a low current in the order of 4uA (couldn't go much lower). Therefore we will have to have capacitors of dozens of picofarads, the type of which was chosen due to the temperature coefficient, and could not be chosen with "area saving" criteria:

$$C\,pF = approximately = \frac{dQ}{dV} = \frac{\frac{dQ}{dt}dt}{dV} = \frac{i\,dt}{dV} = \frac{4u\,dt}{V} = \frac{*4*15\sec}{1} = 60,$$

- Layout was spread more than it could: the digital parts (copied from the library) were not brought close to each other to create a united nmos well for the pmos's and the distances were not strictly kept to the absolute minimum allowed.

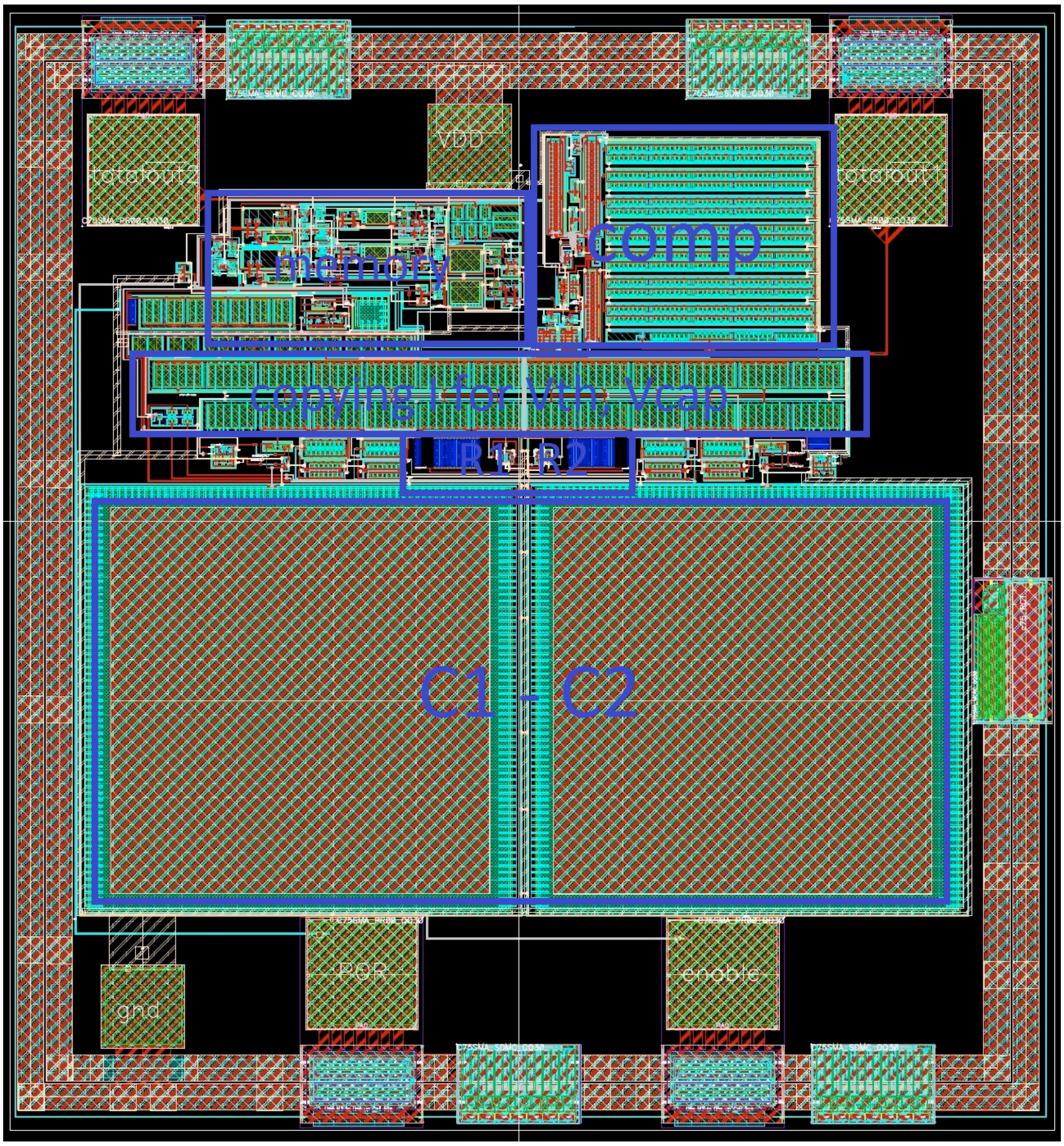


Figure 5.4: Layout of the whole chip, with the ESD protection ring. Total size: 934µmx1000µm, "core" 905µmx714µm.

5.5 Measurements procedure

After the chips came back from the factory, we had 50 of them. In order to measure these 50 samples, we created the following test board (see Figure 5.5):
-In the center we would connect our *chips*.
-As inputs we have the *POR pulse*, which first was going to be created with an XOR gate, with inputs the supply and a delayed version of the supply, but finally since this was not stable enough we decided to use an ideal pulse input from a generator. *Supply Vdd* was accompanied with decoupling capacitors, in the size of a few uF and a few nF, placed as close to the chip as possible - to collect any unwanted parasitics on the board. *Ground* was the whole plane on which we were going to work (it was actually a conductive copper plate). *Enable signal* was also there to help us shut the oscillation function down.
-We had the two (complementary) *outputs*.
-Apart from those, our outputs were not designed to drive a huge load and we needed something that could interact with the 50Ω probe for the phase noise measurements. For that reason, we included a *buffer*, with supplies ±5V (with their decoupling capacitors) and the output of the buffer was going to be our final output. Besides, this buffer is not expected to affect our result in any way, on one hand, since it was chosen to be very high bandwidth, and our oscillator runs at just 32kHz, and on the other a voltage buffer is noise optimal: feedback network has R=0, which implies no amplification of the amplifier noise and no production on its own.

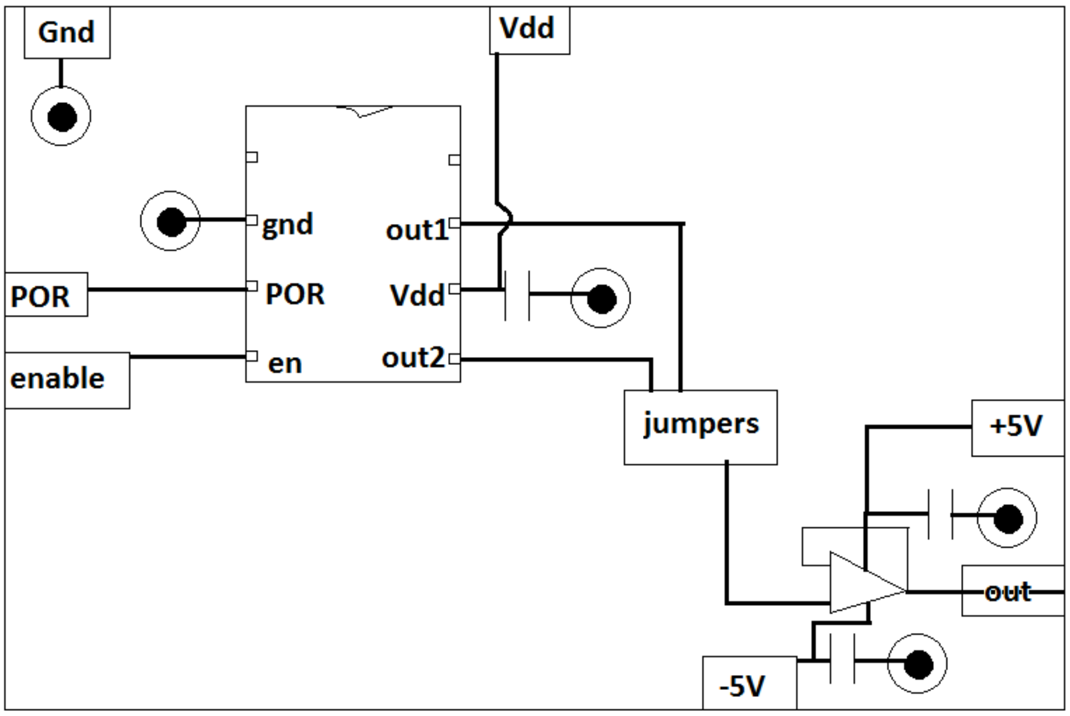


Figure 5.5: Measurements board

A photo of the measurement board can be seen in appendix A.4. A photo of the chip follows:

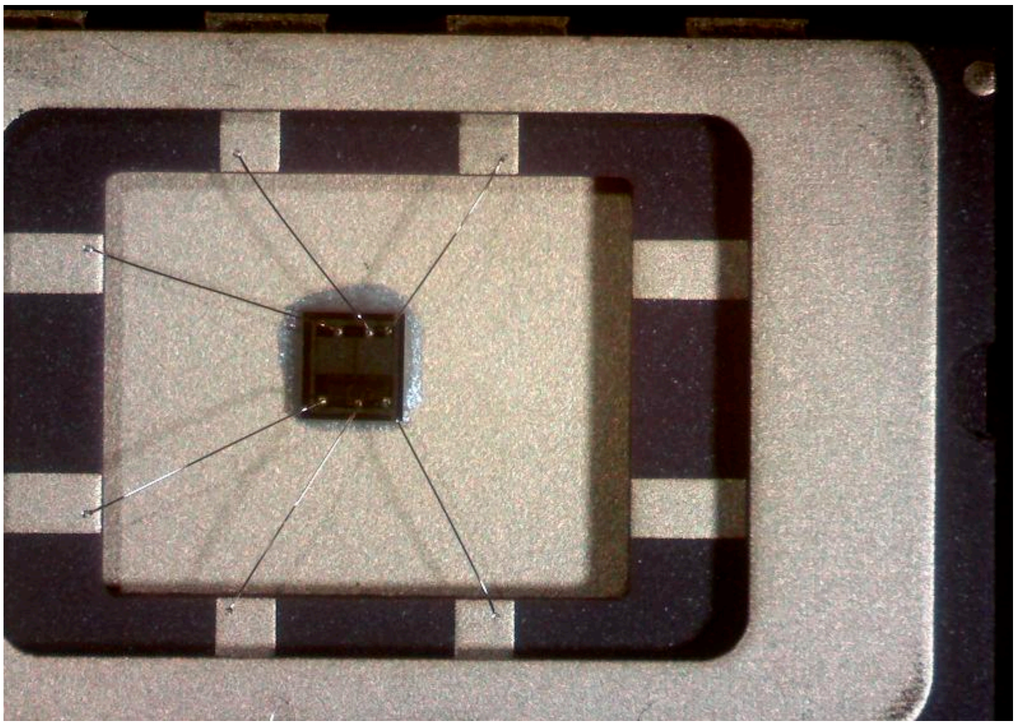


Figure 5.6: A caption of the chip with the bond wires

Chapter 6
Simulations VS Measurements

6.1 Time

Matrix 6.1: Rise Time-Fall time with 2 different loads

	With load (F)	Simulation	Measurement
Rise time	1p	67nsec	88nsec
	12p	135nsec	103.2nsec
Fall time	1p	28nsec	54.4nsec
	12p	62nsec	57.7nsec

The measurement result is the average of 3 different chips being measured.
The measurement results show longer time because the cables in the output were adding extra load.

Matrix 6.2: Duty cycle

	Simulation	Measurement
Out high/low	50±0.13% of period	50±2.3% of period

The measurement result is the average of 3 different chips.
Duty cycle reveals the existence of offset in the comparator and/or problems with the output stage, plus any possible mismatch in the R's or C's of the two main blocks integrating the current and creating the threshold.

Matrix 6.3: Start up time/Rise time

	Simulation	Measurement
After power up (without POR pulse)	Doesn't start	Doesn't always start (especially in extreme temperatures). When it does, in less than 250usec for a power on that takes 400usec
After power up (with POR pulse)	Immediately after the sharp POR pulse is removed	Immediately after the sharp POR pulse is removed
After enable stops	When enable signal is being read	Immediately (for sharp enable signal)

6.2 Voltages

Matrix 6.4: Output voltage(s)

		Simulation	Measurement
Vout	High	1.8V	1.830V
	Max (spike height)	1.8+170μV	2.105V
	Low	1.8nV	0mV
	Min (spike low)	0-920μV	-272mV

The measurement result is the average of 3 different chips.

6.3 Currents

Matrix 6.5: Currents in normal operation and various temperatures

	Current (A) in Simulations			In Measurements		
Temperature	85oC	27oC	-30oC	85oC	27oC	-30oC
Fast	119.5u	106.4u	94.75u	82.7u	73.1u	66.9u
Typical	99.72u	90.14u	80.53u			
Slow	88.01u	78.7u	68.02u			

The above simulations and measurements were done with a load of 1p.

The monte carlo results showed that in -30oC, the spread of current consumption in slow corner was 65-71uA, and in typical 69-81uA, so the measurements were mostly changing in the lowest possible consumptions, while for 85oC in slow corner 88-98uA and in typical 96-105uA, and we see that measurements were a few uA lower than the lowest expected consumption.

With 12pF load, we measured in the actual chips an increase of the average current of almost 2uA. Approximately the same result came from simulations as well. This corresponds with a draft calculation: $C = \frac{dQ}{dV} = \frac{i\,dt}{dV} = \frac{\frac{dQ}{dt} * dt}{dV}$

We measure 2uA more current, so gathered in the increase of the rise time (15nsec approximately) means it's a spike of 2mA in the crucial short time. So the capacitor load is indeed in the order of tens of pF: $C = \frac{i\,dt}{dV} = \frac{2m * 15n}{2V} = 15pF$

Matrix 6.6: Currents when the operation of the clock is paused (changing the enable input)

	Current (A) in Simulations			in Measurements		
	85oC	27oC	-30oC	85oC	27oC	-30oC
Fast	15u	2.6u	250n	2.35u	0.28u	0.11u
Typical	3u	30n	27n			
Slow	397n	87n	58n			

The measurement results are the average of 3 different chips.

The reason for the huge simulated current (15uA for fast corner for high temperatures) is the intrinsic design of the low threshold transistors causing them to leak too much current when they are supposed to be off (non conducting), something that gets worse when at high frequency and fast corner.

6.4 Phase Noise

Matrix 6.8: Phase noise at 1kHz from the carrier frequency.

	-1kHz (measurement)	+1kHz (measurement)	Simulation
Phase noise	-69.225dBc/Hz	-73.05dBc/Hz	-75dBc/Hz

Phase noise was measured also over temperature, but remained practically the same.

The analysis of the phase noise result (in simulations) showed that approximately 40% of it was originating from the transistors in the PTAT current source (mainly the diode connected ones M2 and M3 in Figure 4.7, but also M1 and M4 in the same Figure), while 15% was originating from the transistor acting as a current source and creating the threshold (M4 in Figure 4.4) and 15% from the transistors in the comparators (especially M71, M72, M81, M82 in Figure 4.15). The transistor creating the threshold plays an important role in one of the crucial nodes for the creation of the frequency, so its contribution doesn't surprise us. That contribution is mainly via its id, meaning that phase noise from these transistors originated from their channel thermal noise and it's an inherent issue of the topology (cannot be reduced by changing biasing point for example). The comparator transistors should not have been important, but we've seen that the comparator delay is part of our period, so a small change in the response time of one of those transistors will affect the accuracy and increase phase noise in our frequency (or make jitter issues worse). Their contribution was via their fn, which implies that the phase noise contribution of these transistors is originating from their 1/f noise and it could be improved if the current density through them would be reduced (so for same size, less current). Finally the current source should not have contributed to the noise at all. So filtering of that noise could be applied to minimize this effect.

6.5 Supply

Matrix 6.7: Frequency deviations when supply changes

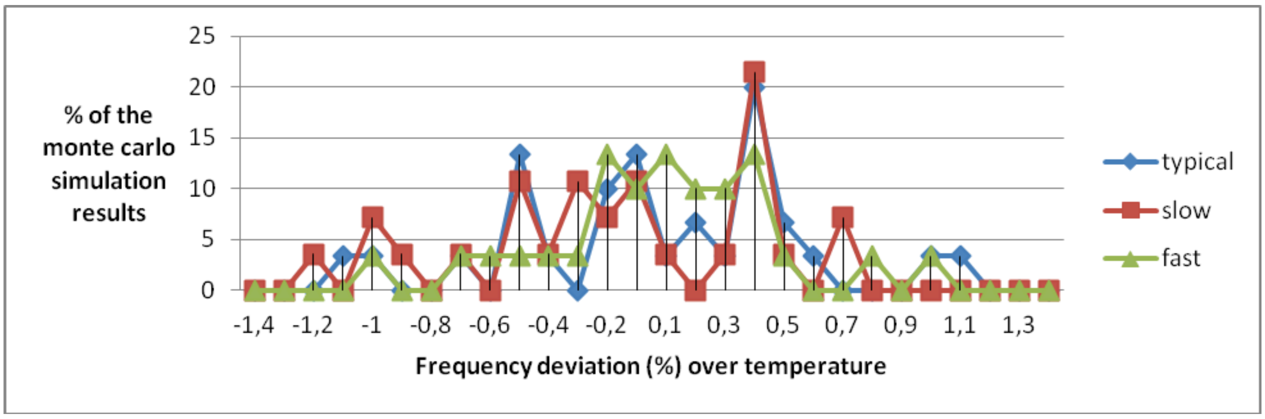
Supply (V)	Simulated frequency(kHz)	Measured frequency (kHz)		
2 (+11.1%)	29.77	28.16	28.26	29
1.8	29.79	28.18	28.23	29.025
1.65	29.76	28.2	28.21	29.05
1.5(-16.66%)	27.61	28.218	28.2	29.09
1.35	21.76	28.206	28.168	29.1
1.25		27.08	26.69	27.26

Over supply variations (+ 11.1% and – 16.6%) the deviation is just below 0.13%, better than the simulated one (0.6%)

6.6 Over temperature

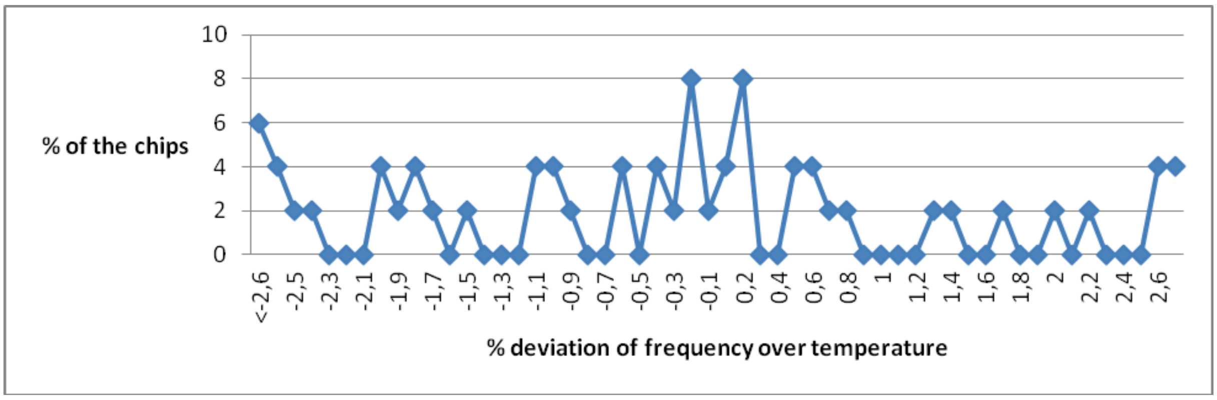
In Graph 6.1 our expectations are depicted, after simulations for all three corners (these simulations were performed in a file that contained all components of the design, even the ESD protection ring).

88.6% of the chips were expected to have a deviation of their frequency within ±0.8% over the whole range of temperatures compared to their room temperature value, while 96.6% was under ±1% deviation.
On average 0.468% frequency deviation was expected over temperature.
If we add the spread of the RC temperature coefficient, we can expect something like more than 80% of the chips to deviate less than ±1% over temperature.



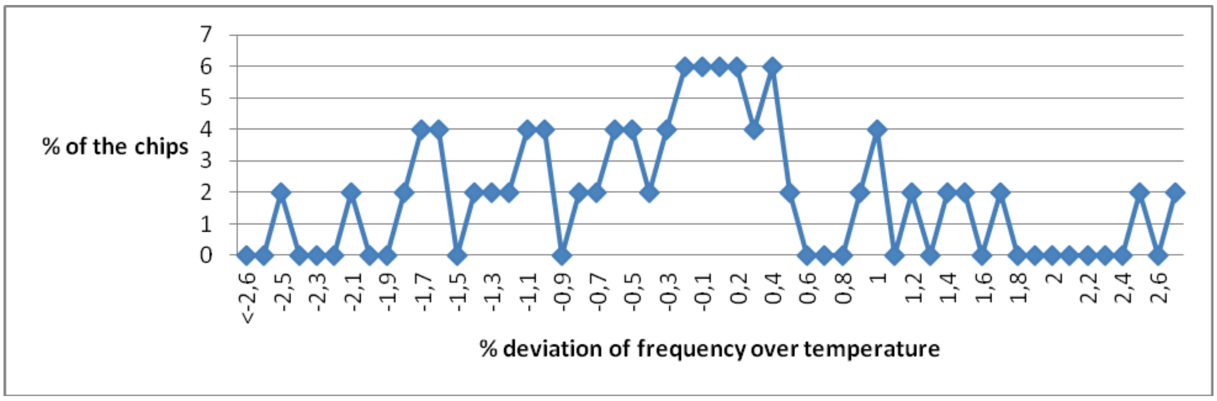
Graph 6.1: Frequency shift over temperature after simulations in all corners (data in appendix A.5).

The measurement results are shown in Graph 6.2, for the whole range of temperatures (-30oC up to 85oC), and in 6.3 for a shorter one (-10oC up to 60oC)



Graph 6.2: Measurement results for the 50 chips (data in appendix, A.6)

Measurements were performed with all the chips, (functioning chips 100%)- and for all of them, -30oC and +85oC, the extreme temperatures, were the two with the most extreme frequencies as well, and the behaviour was almost linear in between.



Graph 6.3: Same like Figure 6.2, but leaving out the 2 extreme temperatures (this time -10 to 60 oC are the extreme temperatures) (data in appendix, A.7)

Summing up, in the envelope ±0.8% deviation were 44% of the chips, while 60% was under ±1.6% deviation, and finally 80% under ±2.4% deviation. Considering the second less extreme temperature both for low and high temperatures, in the envelope ±0.8% deviation were 54% of the chips, 84% was under ±1.6% deviation, and only the rest 16% above that 1.6%.
On average 1.32%, or for lower temperature range 0.83%.

Matrix 6.9: Center frequency (at 25oC):

(Hz)	Simulations (Hz)	Measurements
>30000	(fast corner) 32479.9	4% of the chips
29000<<30000	(typical corner) 29773.9	16%
28000<<29000		52%
<28000	(slow corner) 27446.4	28%

6.7 Unexpected findings

1) A fast frequency while using the POR pulse.

Observation: When the POR pulse was applied before the actual function of the oscillation was allowed to begin, the existence of the pulse was supposed to make the digital block set in one condition so the operation could start correctly when the pulse would be de-asserted. The output was supposed to be quiet. On the contrary, we noticed a high frequency change of state, recovering really fast, which can be seen in appendix, picture A.8.

Origin: In order to include the effect of POR pulse on our RS flip flops of the state memory, we used some OR and XOR gates in front of their inputs. For this see Figure 6.1. The output of OR and XOR gates was the input in the RS ff, while the inputs of these extra gates were two: the inputs that were supposed to be R or S signal in the RS ff (the output of the previous stage), together with the POR signal as a second input. That way we managed to have constant complementary inputs to the FF, and therefore a constant output of the logic blocks.

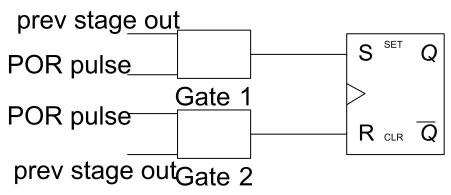


Figure 6.1: How the POR pulse is controlling our RS ff inputs.

But while on enable operation (another pulse to which we behaved in a similar way), the current source is turned off, and there is no current to be integrated on the capacitor or to create a threshold, so the inputs of the comparator are always reading zero, and everything stays idle, since we also control the (digital) output of the comparator not to switch for any reason, when POR pulse is asserted unfortunately the current source is working fine. The capacitor is being charged, and the comparator finally switches since no control on its output was applied (for the POR pulse). As the comparator output swaps, the charge/discharge commands swap as well, but not the position of the switches in front of the comparator. That way we switch state really quickly.

The problem arises from the fact that when the comparator output swaps, suddenly both inputs of the RS ff become high (1), which is not supposed to happen, and RS ff changes state, and system commands the integrator to stop integrating but start discharging (a command controlled by the memory and not by the comparator output).

As in Figure 6.2, we get some spikes really fast, corresponding to a period, analyzed as follows: integrator's output crosses the threshold (time 0) + delay through the comparator(a) + discharge time (b) until integrator's output recrosses the threshold + delay through the comparator until the new situation is being read(c=a)+start charging again till the threshold is crossed (d).

Solution: Controlling the output of the comparator so it doesn't flip. We did do that for the enable/disable operation, because in simulations the pulse used was long enough. For POR pulse, we always simulated with a shorter pulse than the one we measured with, and that's why we didn't notice the issue, although we managed to recreate it in our simulations. We simulated a period of 0.31usec, and measured a longer one (1.1usec approximately).

A different solution would be the POR pulse to have duration of less than 15usec (like we did in the simulations). Then the capacitor that is being charged does not have the time to reach the threshold and the comparator does not change state. That can be kept into account when designing the sub circuit that can implement the POR pulse internally, on chip.

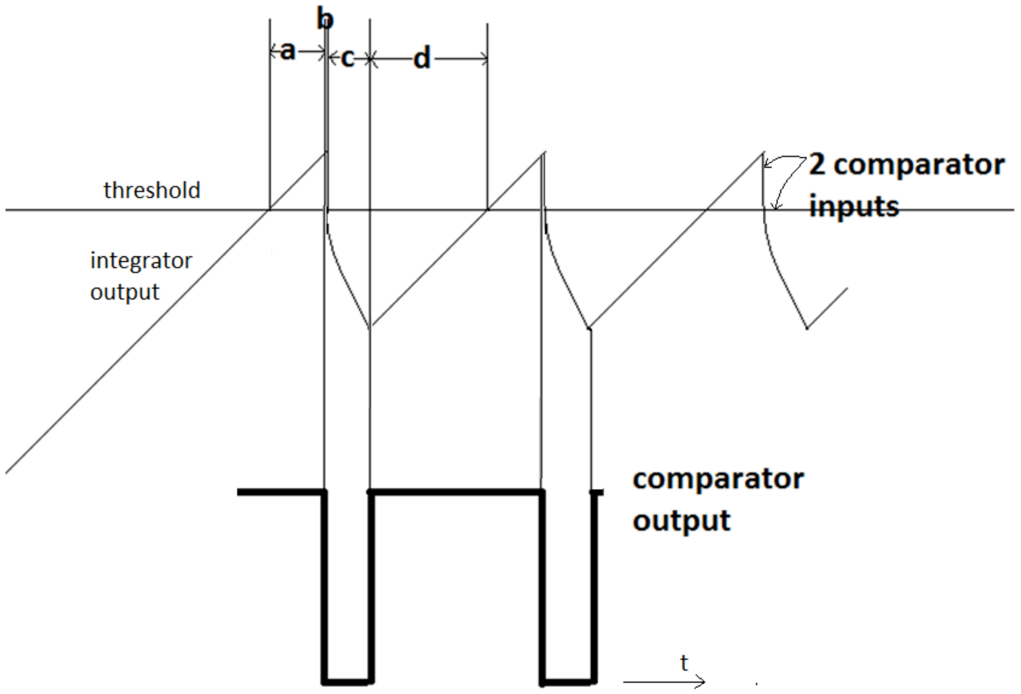
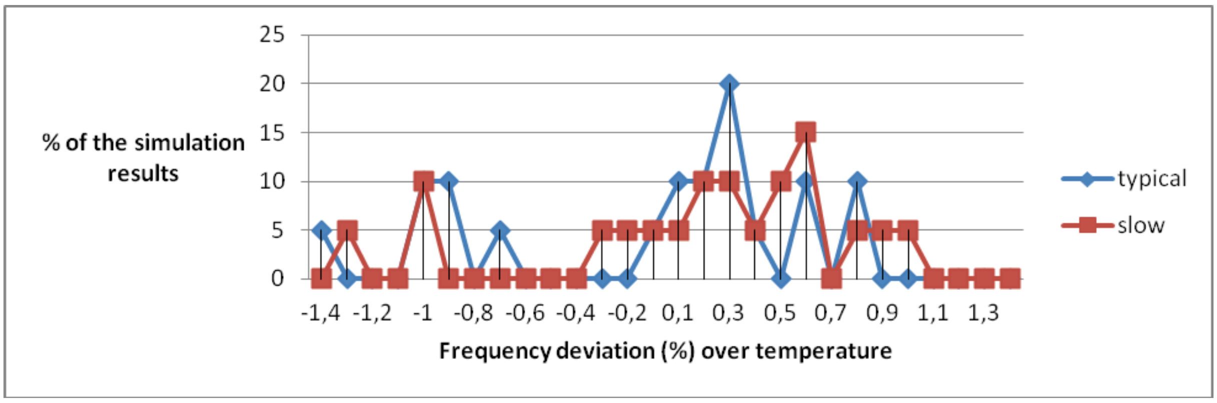


Figure 6.2: What happens in case of a long POR pulse, time slots a/b/c/d explained in text above.

2) Why measurements deviate from simulations that much

Observation: As seen in section 6.6, the over temperature stability of our oscillator is not as good as the simulations predicted. In order to check if the parasitic are responsible for that, we extracted the layout, in order to be able to perform simulations with all the parasitic capacitors present. The results are shown in Graph 6.4.



Graph 6.4: Simulation results from the extracted layout file. Including all capacitive parasitic. (Matrix with the data can be found in appendix, A.3)

We notice here that the results taking into account all parasitic capacitors (approximately 1200 of them) are not much different than without them. The majority of the simulation results remain in the envelope ±0.8% deviation over temperature (75% of them), and a few are higher, but 95% of the simulation results remain below 1% deviation. So there is no way we could have predicted the measured result from our simulator.

Origin: The reasons for the above mismatch in the results, is either the simulator itself (some effects are probably not modeled correctly or not modelled at all), or the way the measurements were made. We were waiting some time on every temperature for it to stabilize, but maybe the time wasn't enough, or maybe the contribution of the measuring equipment (from bondpads until the machines measuring) was harmful to the final outcome. Or the method (we used the spectrum analyzer and checking the frequency with the most energy in the output, i.e. the fundamental frequency) was faulty. Finally in Matrix 4.1, we see that we had no data on the spread of the temperature coefficient of the PCA. We considered a similar spread like the one of the TFR to get the up to 0.15% spread of the RC product. If the spread was hundreds of ppm, then it would be the spread of RC product that produces this difference in results.

3) Glitch

Observation: In time measurements of the chips, a glitch was present, on both outputs and the picture of that can be found in appendix A.2. This glitch was present also on the extracted layout simulations (even though it was not present in the previous simulations) so we managed to find the reason for that.

Origin: One parasitic capacitor of just 10.4fF was found, the absence of which made the glitch disappear (clean rise and fall for both outputs). The presence of this parasitic capacitor in the electrical circuit for simulations (not the extracted one) created actual several glitches, not just one. The reason for the several glitches is that when all 1200 capacitors were present, the (parasitic) charging of the crucial nodes made the effect less dominant and only one glitch appeared.

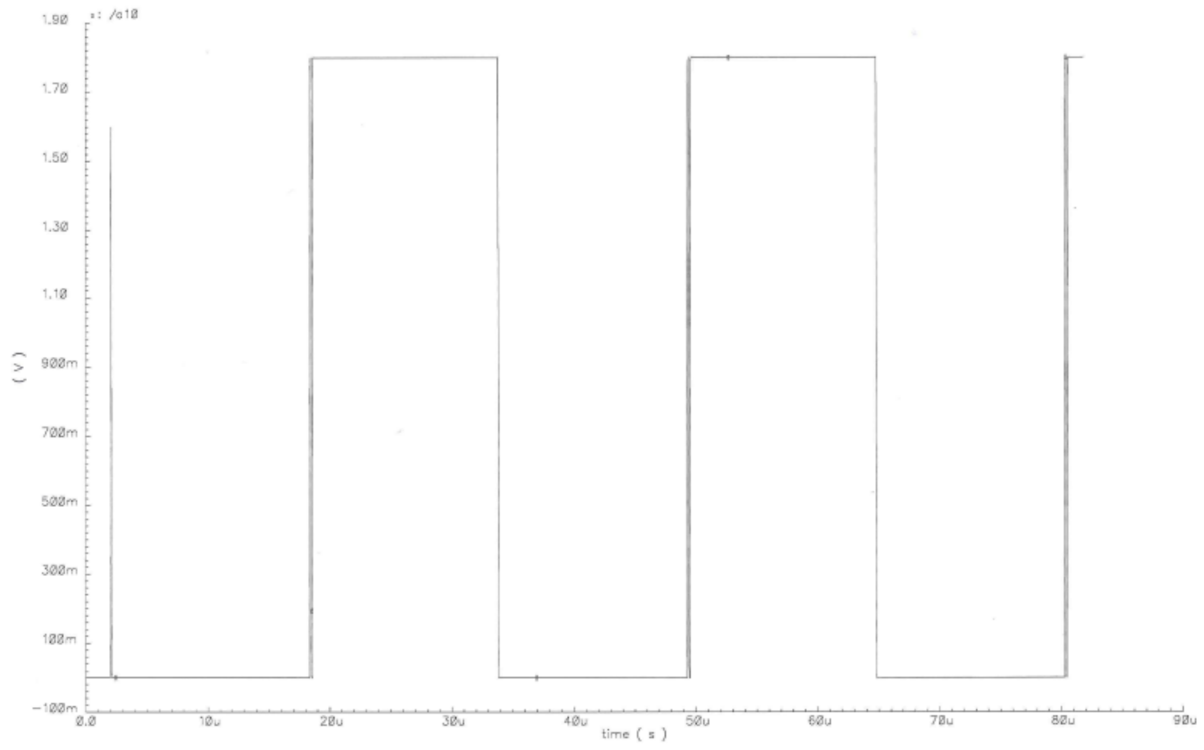


Figure 6.3: The comparator output with the presence of a glitch (simulation result in the extracted file)

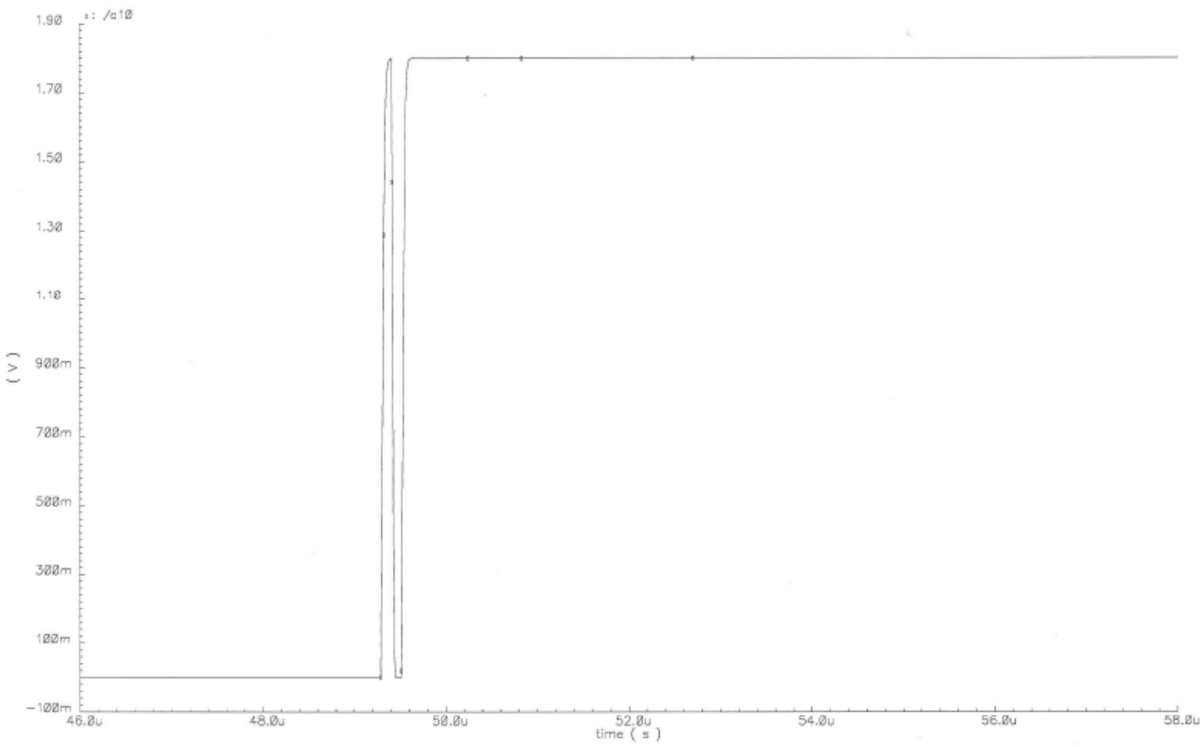


Figure 6.4: The comparator output with the presence of a glitch, zoomed in

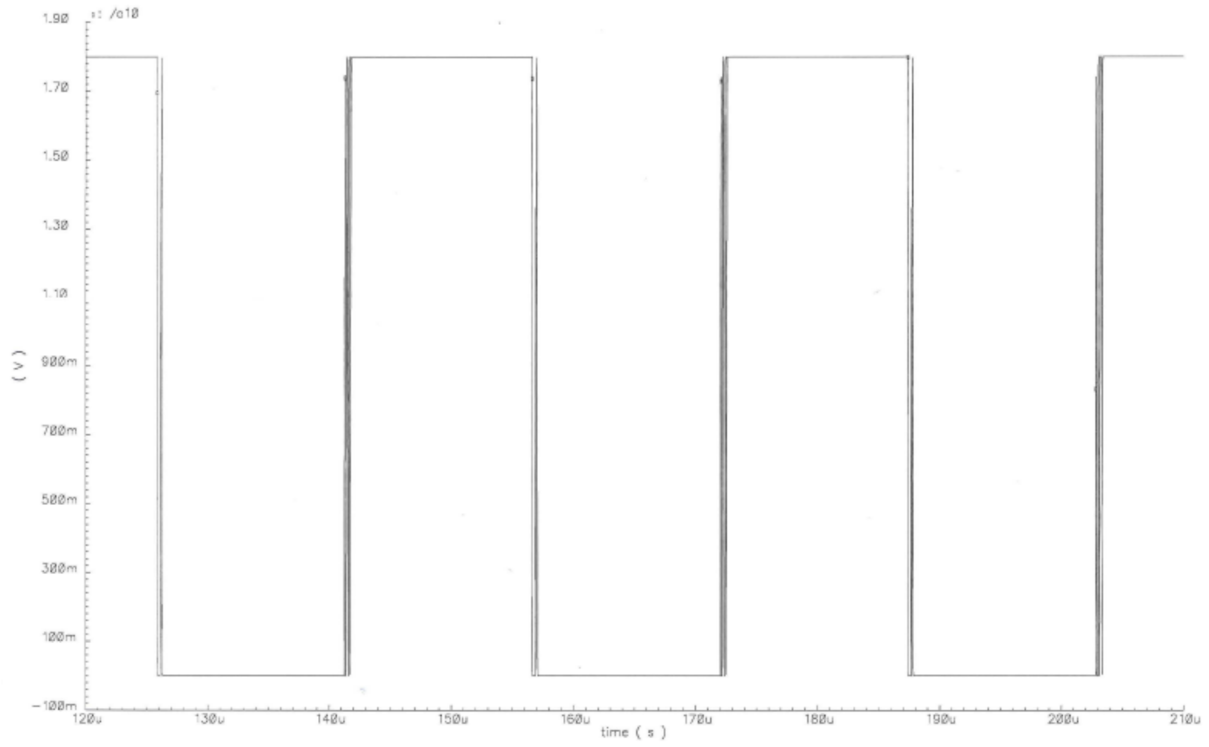


Figure 6.5: The comparator output when a parasitic capacitor of 10fF is added between total output and Vds node

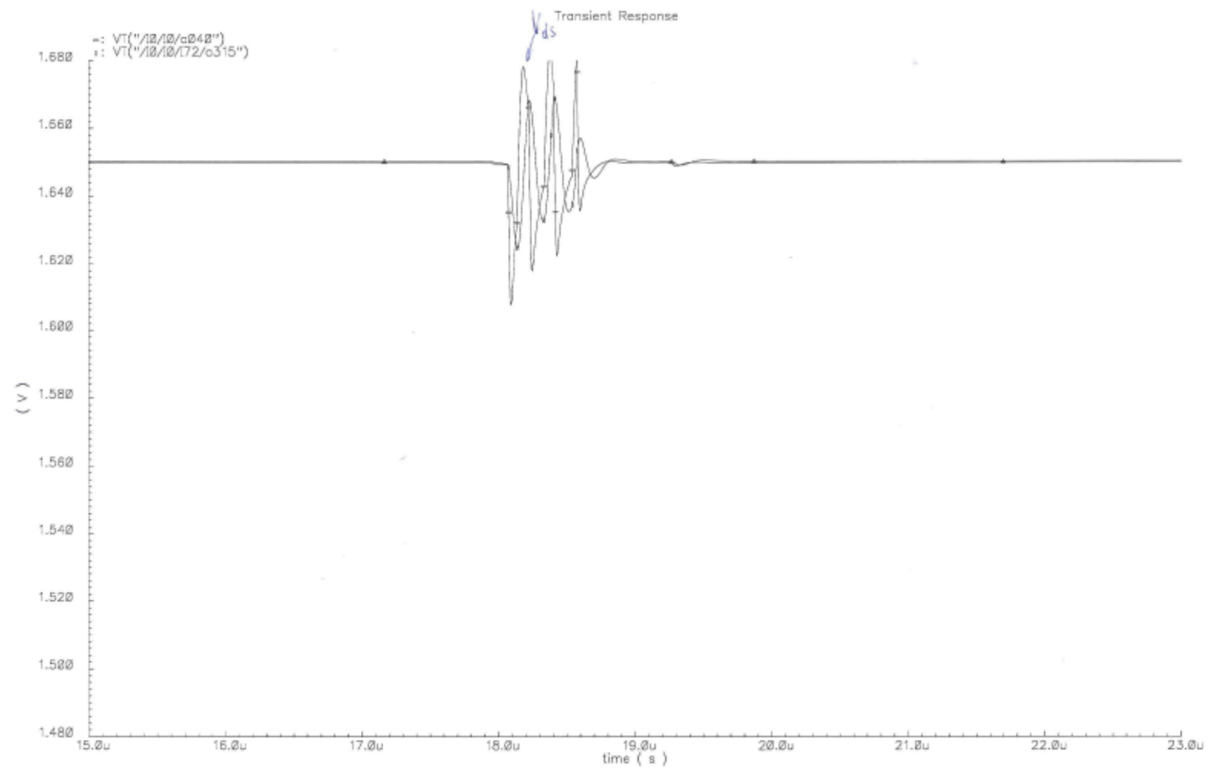


Figure 6.6: Vds node voltage changing a bit faster than the Vd node which is virtually short circuited to Vds via the gain boost amplifiers, of Figure 4.4

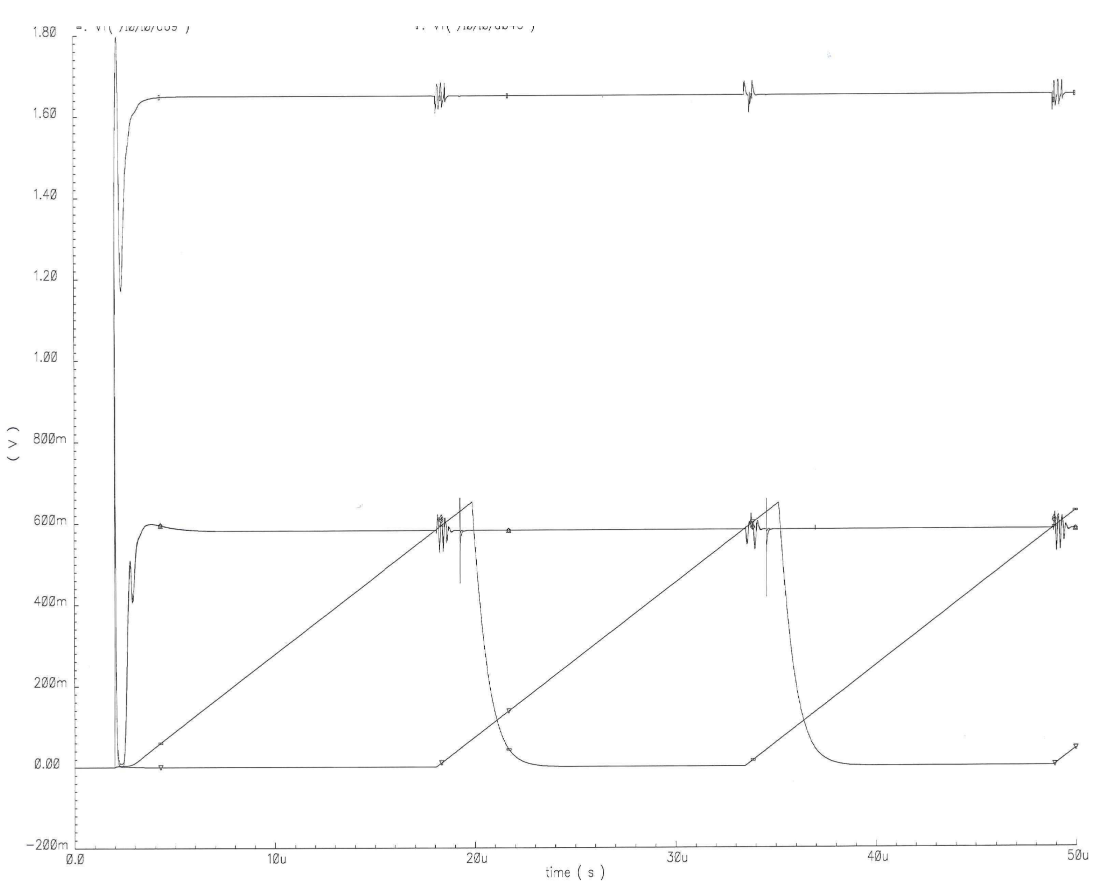


Figure 6.7: Around 1.65 V is the Vds node voltage, then three lines below are the two integrators' outputs and the threshold (suffering from many glitches)

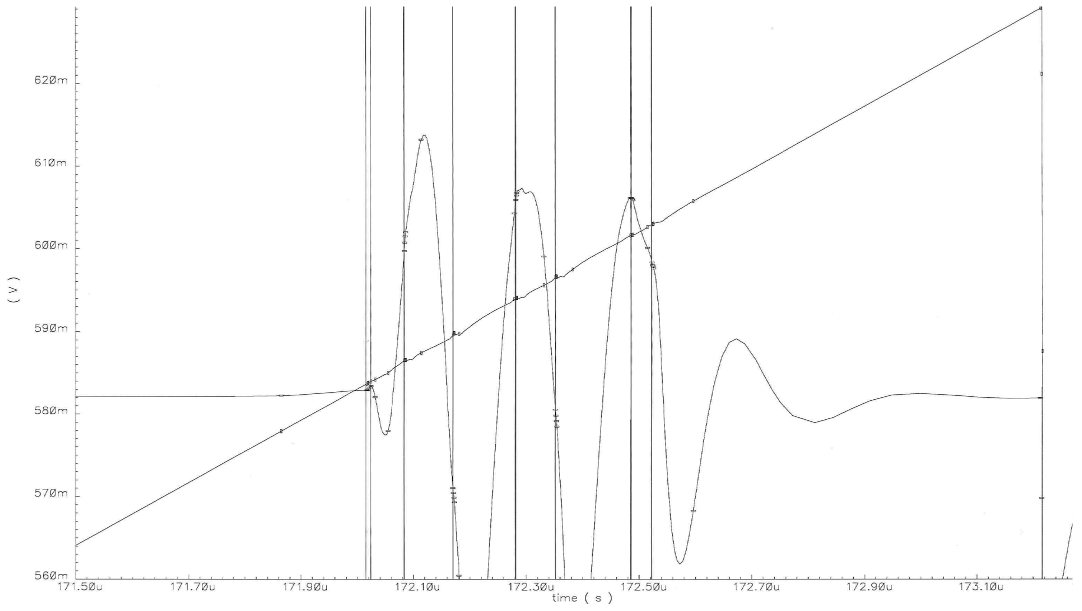


Figure 6.8: Zoomed in the threshold crossing many times the rising slope, and the comparator changing approximately 50nsecs after each crossing, creating several glitches.

The capacitor was placed between the output (the total output) and the Vds line, shown in Figure 4.4. As seen in Figure 6.6 when the output changes from high to low, the other end of this capacitor (Vds hand marked on Figure 6.6) is pushed a bit lower. With some delay (depending on the speed of the amplifier who gain boosts the cascode transistors in Figure 4.4) this appears also on the drains of the current source transistors. This sudden change (dip) in Vds of 30mV causes 60nA increase in the current source that creates the threshold. Unfortunately our huge resistor of 500k, translates this in a sudden voltage increase of 30mV approximately, enough to re-cross the slope (since the capacitor keeps integrating) and enough for the comparator which has no hysteresis to recreate that on the output (as shown in Figure 6.8). The parasitic capacitor was caused by 2 lines (Vds and total output 2) running in parallel for 200um in the layout.

Solution: One of the possible solutions is to change the layout so either the two metal lines are not running in parallel (in order to reduce the effect of this parasitic) or a grounded line is introduced between these two lines, so the parasitic capacitors are formed to ground and they are not coupling nodes of the circuit. If we want a non-layout-invasive solution, we could filter the Vds node with a huge capacitor to supply Vdd and keep the voltage of the node more stable, so the small changes introduced by the parasitic capacitor will be filtered out. Finally we could simply take as our output the output of the memory (which was not fooled by the fast changes in the comparator output) and keep the glitches with no problem. Although keeping a non-ideality that can be corrected wouldn’t be a wise choice, and the memory output belongs to the slow timing path so it’s not the best possible choice.

Chapter 7
Conclusion and Recommendations

7.1 Contributions

In the course of this design a novel oscillator topology was developed that could be implemented in a cmos standard technology. This topology materializes a second order quadrature-like oscillator and manages to decouple in time the positive feedback loop (regenerative action of the oscillator) from the negative feedback loop, and takes out of the timing loop all issues related to the state memory.

This topology was implemented, simulated, fabricated and measured. In Matrix 7.1 the results of this design are compared to a few other all silicon oscillation choices based on the same timing principle or similar principles. The outcome from this procedure is that RC-time constant based oscillators should not be trusted as the core of keeping the frequency stable over temperature, but should be surrounded by a master circuitry that implements a tuning of the frequency as explained extensively in the recommendations in Section 7.3. Nevertheless this is one of the best non-tuned RC-timing-constant-based integrated oscillators.

Matrix 7.1: This work and a few others recently published, ranked in descending performance of frequency stability over temperature deviations

Type	Freq (Hz)	Temp stability over 100oC(-20oC to 80oC approx)	Supply (V)	Current (A)	Comments	Reference
Fully integrated LC oscillator & divider	50M	$\pm1*10^{-6}$	3.3	8m	The LC oscillator is equipped with temp compensation	[7]
Bypassed RC-time based [*1]	100k-100M	$0.5*10^{-4}$	3.3	-		[9]
RC-time based	-	$\pm1*10^{-3}$	3.3	-	Uses current reference, temp compensated up to second order.	[10]
Bypassed RC-time based[*1]	14M	$\pm2*10^{-3}$	1.8	25u		[19]
This work: RC-time based (simulations)	**32k**	**$\pm4.68*10^{-3}$(avg)**	**1.8**	**90.1u**		
Bypassed RC-time based[*1]	250k-1M	$\pm5*10^{-3}$	3.3	400u	Requires an ideal external resistor	[8]
This work: RC-time based (measurements)	**32k**	**$\pm1.32*10^{-2}$(avg)**	**1.8**	**73.1u**		
CMOS, RC based	16M	$\pm2*10^{-2}$	1.8	-		[11]

[*1] Bypassed RC time based oscillator implies (as will be analyzed in section 7.3) an oscillator whose frequency is produced based on an RC time constant but this core is bypassed by a "master" sub-

network that is responsible for the accuracy of the produced frequency by controlling the RC network.

7.2 The Outcome compared to the Specifications

The target of this project was to design a stable frequency reference over temperature deviations using a low power budget. This frequency reference had to be fully integrated, so the tradeoff between integratability (on chip) accompanied with low power against performance characteristics was examined.

The Outcome of this project, i.e. the Measurement Results (as far as frequency stability over temperature is concerned) show worse performance than the Simulated Results, due to non-modeled effects of temperature deviations on our circuit. We concluded that (after analysis in Chapter 6) since the extracted layout simulations were still giving the same results like the electrical model simulations. This means that it wasn't the parasitics that deteriorated the frequency stability, but some non modeled –by the simulator- effects.

The Simulated Results (again as far as behavior over temperature is concerned) were worse than the Specifications, due to issues like the spread of the temperature coefficients of the resistor and the capacitor implementing the RC product that determines the frequency, the deviation over temperature of the delay of the comparator, and a few more reasons analyzed extensively in Section 5.1. Room for improvement on those remaining issues is very limited, so a different approach for the whole design (still keeping it an RC time constant based oscillator) can be implemented and try to improve the performance. Ideas on that can be found in the next section (7.3).

As far as the rest of the specifications are concerned, the chips were behaving according to expectations (and specifications) since the oscillation produced is very stable over supply variations, power-wise it is really appealing (we achieved a power consumption much less than the available budget), and the current when shutting the operation down is very low. The deviation of the RC product is low (approximately ±6%) due to the huge size of the capacitors and the large resistor (as passive components we would have been expecting a ±10% (at least) process deviation from the nominal values in the sizes of the resistor and capacitor, and therefore a ±20% deviation from nominal value for the frequency). The phase noise results are according to the expectations. The load can be driven without creating any issues to the oscillation frequency or the operation of the circuit. Duty cycle is always within specifications as well.

7.3 Recommendations

For the issue of frequency stability over temperature deviations, there are two approaches:

- Correcting issues that can be improved
- Bypassing some errors, by using a different approach that shifts the point of accuracy in the creation of the frequency somewhere else (in those "bypassing" new blocks).

The problem with the first choice is that there are not many choices in our circuit, not many things that are in the designer's discretion to be improved. For example, the copying of currents that create the threshold and charge the capacitor can be blamed - after extensive simulations – for only up to 0.3% deviation of frequency over temperature as analyzed in paragraph 5.1 and there is not much room for improvement. On the other hand things like the deviation of the temperature coefficients of the passive components cannot be avoided at all.

The second choice (bypassing the errors) can improve the results, but it transfers the point of accuracy from the blocks that create that frequency to the blocks that will be correcting it (creation of frequency becomes the slave, and the correcting sub-circuit becomes the master). So a very lousy frequency/comparator/memory can be built and can be put up with, if there is a robust way for it to be corrected. Nevertheless some examples of such improvements are presented later.

A completely different initial decision of the way to go for the all silicon oscillator –like an LC oscillator or a MEMS design like analyzed in Chapter 2- perhaps could perform better, but these choices cannot be considered improvements to this design, since their starting point is completely different. One improvement in a second step for our design –without changing the general idea and without bypassing any source of error- would be trimming over temperature, which would unavoidably increase the cost per chip. Thankfully the over temperature behavior of our chips is linear, but unfortunately varies a lot, so even that would be tricky - a flexible temperature coefficient element should be present in order to compensate for the random linear resulting temperature coefficient of the frequency and two measurements in different temperatures should be taken.

But now let’s present some improvements that would bypass some of the errors that exist in our design:

One idea would be the auto tuning of the threshold level by an imitation of the auto-zero technique [8] that cancels the effect of the comparator delay (and the effect of the constant errors that are presented in Section 5.1, coming from switches, constant steps, and the AND gates).

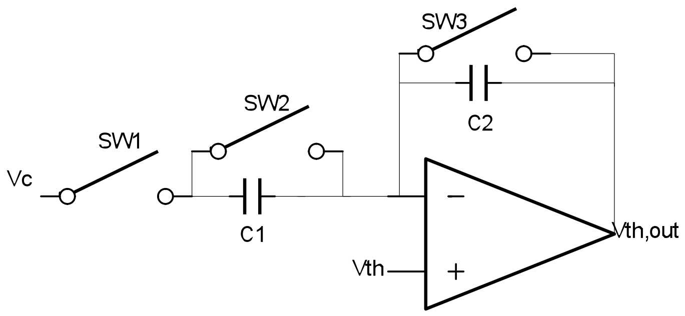


Figure 7.1: A way to bypass the errors of our oscillator, by manipulating the threshold level with this configuration. In this Figure Vc is the voltage on the capacitor, and Vth is the threshold.

Let’s examine the configuration shown in Figure 7.1. When the comparator switches state, SW1 is closed and SW2 is opened. Then, the actual V on the capacitor will be Vcap+Verror, because integration will have lasted a bit more, due to the internal delay of the comparator and possible charge issues related to switches. Then this Verror will be saved on C1, and Vth,out will be Vth-(C1/C2)Verror. This will be used as threshold level in the second period, and after some periods (depending on C1/C2) the Verror will be completely subtracted from Vthreshold and therefore it will have compensated for these errors, by using a lower threshold. In “normal operation”, SW1 is open, SW2 is closed and the amplifier is just delivering Vth+VC2 (which is a negative part of Verror as analyzed before). SW3 is always open, unless charge on C2 has to be reset.

The advantage of this is that it is automatic and can be tuned again when temperature changes and comparator (and switch related) delay changes too. The disadvantage of that is that it takes the threshold level and the ramp on the capacitor as error-free (in [8] it is proposed that an ideal off chip R should be used for that reason) so it is not correcting for errors there (spread of R,C's temperature coefficients remain), and it takes out the comparator delay issue, but brings back an offset issue: the offset of this new comparator (that can be varying over temperature) and the charge issues introduced by the new switches. Even with ideal off chip resistor, in [8], and only by simulation results a ±0.5% deviation over temperature is achieved.

Another idea to bypass sources of error (actually here the whole oscillator is bypassed) is the following. A feedback network [19] around the relaxation oscillator, is averaging the voltage being produced, and tries to implement

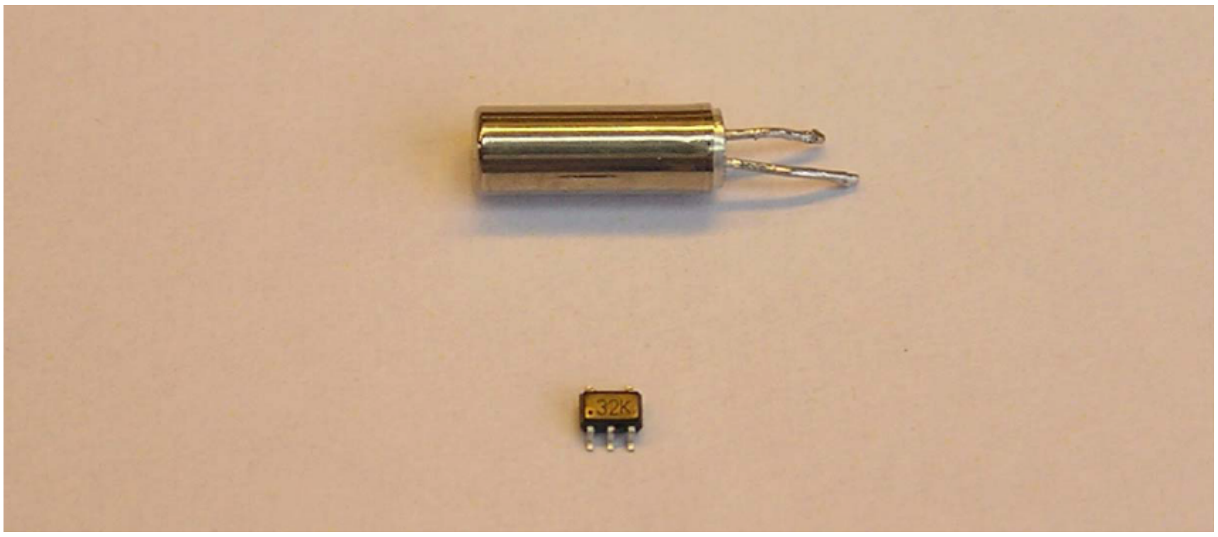
$$V_{ref} = \frac{1}{T}\int_0^T V_{osc}\,dt$$

by changing the threshold to a bit lower or higher voltages in order to compensate for all the timing errors. Therefore, since all delays –expected, unexpected, temperature varying or not- are going to be "hidden" in the process of averaging, the accuracy of oscillation is moved into creating a stable reference voltage (BG reference) and into the new comparator who is going to decide how "close" the averaged voltage is to that reference (with offset issues etc). It appears nevertheless to have achieved nice results (approximately ±0.2% [19])

APPENDIX

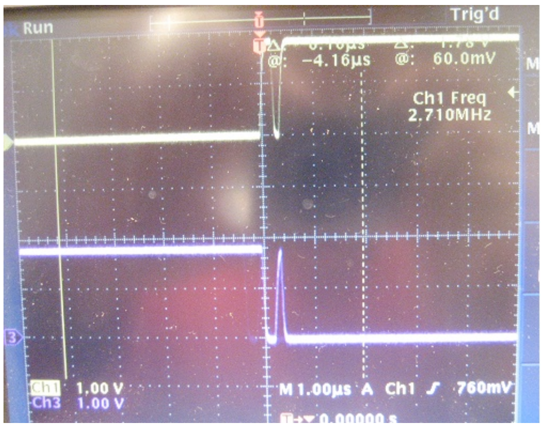
A.1

A photoGraph comparing the size of one all silicon oscillator (encapsulated in its package, and therefore with increased overall size) with one that uses a quartz crystal, from products' AN2745 (ST microelectronics) datasheet



A.2

PhotoGraph of the glitch found on our frequency output



A.3

Simulation results on the extracted layout:

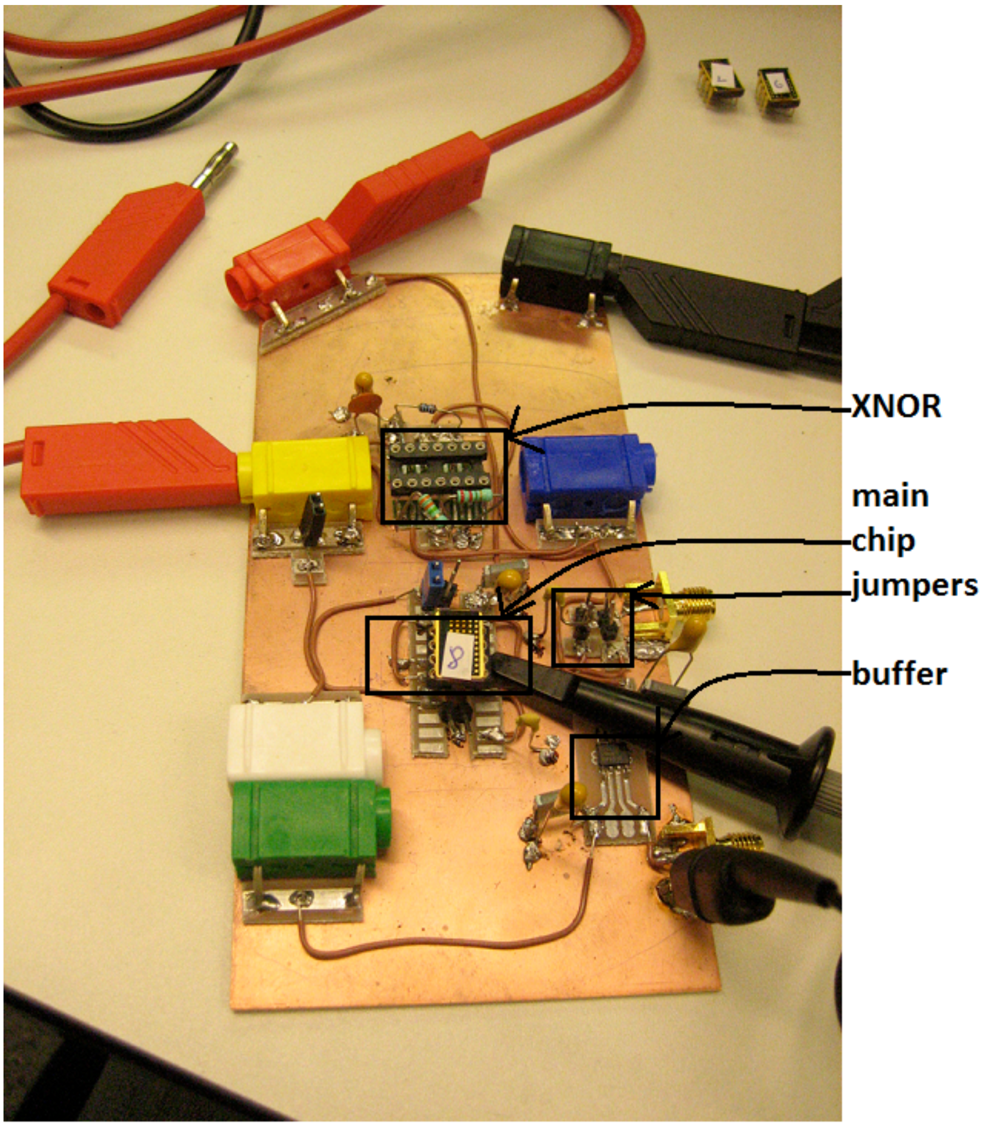
Hz difference of -30oC result compared to 85oC result for the frequency, for 20 repetitions:

Typical (#1-#10)	-396	162	69	23	-285	49	76	227	-19	108
% deviation	-1,32	0,54	0,23	0,07	-0,95	0,163	0,25	0,75	-0,063	0,36
Typical (#11-#20)	162	70	24	-284	-252	50	77	222	-195	-252
% deviation	0,54	0,23	0,08	-0,94	-0,84	0,16	0,256	0,74	-0,65	-0,84

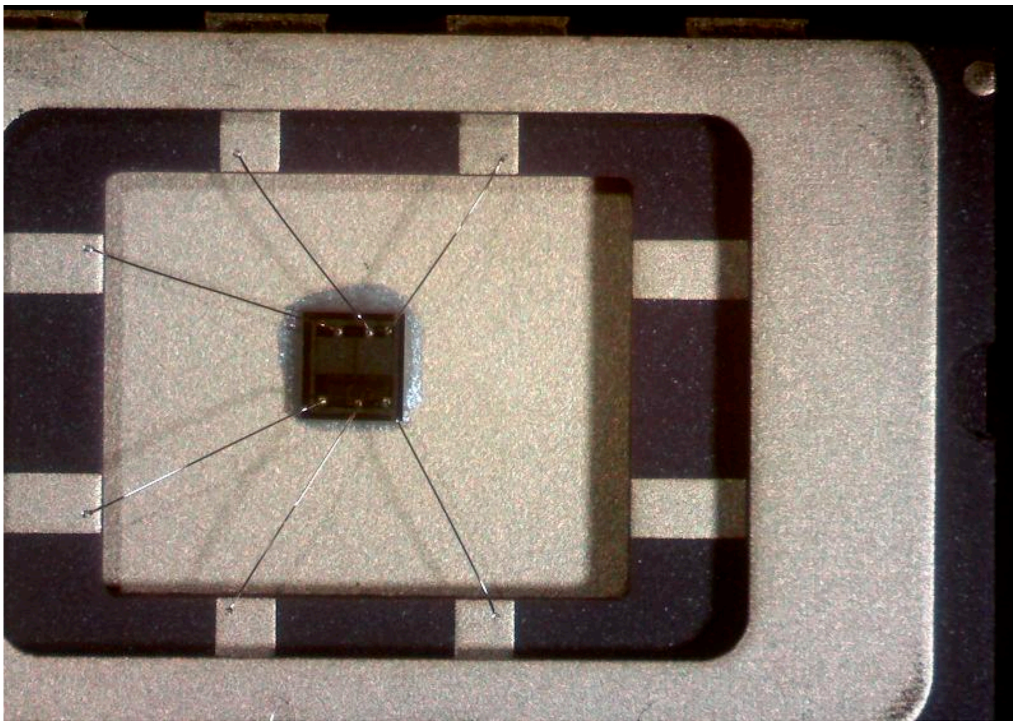
Slow (#1-#10)	-359	154	45	22	-260	-261	51	61	234	-24
% deviation	-1,28	0,55	0,16	0,078	-0,92	-0,932	0,182	0,21	0,83	-0,085
Slow (#11-#20)	122	131	220	56	-35	156	-72	152	258	109
% deviation	0,435	0,46	0,78	0,2	-0,12	0,557	-0,25	0,542	0,92	0,38

A.4

Measurement board. On top of the chip, you can see the XNOR that initially was used for creating the POR pulse, without an external generator, but the poor results forced us to use an external generator for a clear –but longer- pulse. For more details about what is what, you can find the general plan in section 5.5



PhotoGraph from inside the cast of our chip, showing the connection between the bondpads and the package with the bondwires.



A.5

Repetitions in monte carlo simulations.

Hz difference of -30oC result compared to 85oC result for the frequency, for all repetitions and corners:

Typical (#1-#10)	100	100	90	-320	320	-150	110	-40	120	-140
% deviation	0,333	0,333	0,3	-1,06	1,06	-0,5	0,36	-0,13	0,4	-0,46
Typical (#11-#20)	-290	100	-110	0	40	290	-20	-210	-60	110
% deviation	-0,96	0,333	-0,366	0	0,133	0,96 6	-0,06	-0,7	-0,2	0,36 6
Typical (#21-#30)	-10	-150	80	30	-30	-20	-60	-130	170	120
% deviation	-0,03	-0,5	0,266	0,1	-0,1	-0,06	-0,2	-0,433	0,566	0,4
Slow (#1-#10)	100	100	80	-280	-320	-130	90	-50	100	-170
% deviation	0,357	0,357	0,285	-1	-1,14	-0,46	0,32	-0,17	0,357	-0,60
Slow (#11-#20)	-280	110	-90	-20	-40	-240	-70	110	-10	-120
% deviation	-1	0,39	-0,32	-0,071	-0,14	-0,85	-0,25	0,392	-0,035	-0,42

Slow (#21-#28)	180	20	-60	-10	-60	-120	180	120		
% deviation	0,642	0,0714	-0,21	-0,035	-0,214	-0,42	0,64	0,428		
Fast (#1-#10)	60	110	60	-210	300	-160	120	-50	110	-40
% deviation	0,187	0,343	0,18	-0,65	0,93	-0,5	0,37	-0,15	0,34	-0,12
Fast (#11-#20)	-290	90	-110	0	30	240	-20	-190	30	80
% deviation	-0,90	0,281	-0,343	0	0,093	0,75	-0,06	-0,59	0,093	0,25
Fast (#21-#30)	-10	-80	70	10	-30	-40	-60	40	150	110
% deviation	-0,03	-0,25	0,218	0,031	-0,093	-0,12	-0,18	0,125	0,468	0,34

A.6+A.7

Measurement results: frequencies of the chips measured:

Temp (oC)	1	2	3	4	5	6	7	8	9	10
-30	27836,25	28175	28473,75	28455	27408,75	29228,75	28946,25	27195	27862,5	29197,5
-10	27956,25	28166,25	28451,25	28428,75	27562,5	29175	28863,75	27247,5	28065	29182,5
10	28091,25	28207,5	28447,5	28395	27735	29085	28766,25	27341,25	28173,75	29141,25
25	28158,75	28233,75	28481,25	28397,25	27780	29025	28717,5	27416,25	28275	29103,75
40	28260	28260	28488,75	28376,25	27802,5	28965	28721,25	27517,5	28327,5	29073,75
60	28342,5	28286,25	28526,5	28387,5	27840	28908,75	28612,5	27663,75	28357,5	29040
85	28376,25	28316,25	28575	28398,75	27855	28841,25	28578,75	27851,25	28361,25	28987,5
Max Δfreq	540	150	127,5	78,75	446,25	387,5	367,5	656,25	498,75	210
%	1,917699	0,53127905	0,447663	0,277316	1,606371	1,335056	1,279707	2,393653	1,763926	0,721557
For -10 to 60	386,25	120	79	52,5	277,5	266,25	251,25	416,25	292,5	142,5
%	1,371687	0,42502324	0,277375	0,184877	0,99892	0,917313	0,874902	1,51826	1,034483	0,489628

Temp (oC)	11	12	13	14	15	16	17	18	19	20
-30	30965	28455	28290	27588,75	27645	28713,75	28458,75	30550	28852,5	28635
-10	30675	28430	28290	27641,25	27750	28683,75	28492,5	30275	28852,5	28837,5

10	30440	28401	28301,25	27682,5	27885	28646,25	28537,5	30055	28848,75	28935
25	30230	28397,75	28312,5	27705,75	27993,75	28627,5	28567,5	29940	28848,75	28953,75
40	30060	28405	28323,75	27765	28072,5	28616,25	28597,5	29870	28845	28953,75
60	29865	28405	28335	27821,25	28218,75	28575	28616,25	29555	28845	28938,75
85	29690	28411	28368,75	27888,75	28372,5	28548,75	28623,75	29445	28841,25	28920
Max Δfreq	1275	57,25	78,75	300	727,5	165	165	1105	11,25	318,75
%	4,217665	0,20160048	0,278146	1,082808	2,598794	0,576369	0,577579	3,690715	0,038996	1,100894
For -10 to 60	810	32,25	45	180	468,75	108,75	123,75	720	7,5	116,25
%	2,679457	0,11356534	0,15894	0,649685	1,674481	0,379879	0,433185	2,40481	0,025998	0,401502

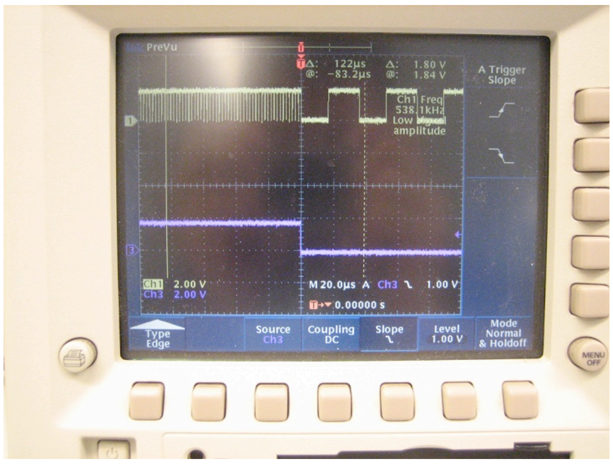
Temp (oC)	21	22	23	24	25	26	27	28	29	30
-30	29385	27720	28080	29478,75	27798,75	27180	29058,75	29685	28695	27963,75
-10	29186,25	27791,25	28098,75	29351,25	27911,25	27255	28931,25	29700	28650	27926,25
10	29047,5	27896,25	28136,25	29220	28016,25	27363,75	28830	29683,75	28616,25	27903,75
25	28965	27963,75	28166,25	29163,75	28072,5	27438,75	28773,75	29670	28593,75	27903,75
40	28867,5	28012,5	28120,38	29092,5	28136,25	27525	28717,5	29647,5	28578,75	27903,75
60	28762,5	28065	28267,5	29013,75	28222,5	27667,5	28665	29610	28567,5	27911,25
85	28635	28113,75	28350	28916,25	28327,5	27858,7	28597,5	29546,25	28545	27926,25
Max Δfreq	750	393,75	270	562,5	528,75	678,7	461,25	153,75	150	60
%	2,589332	1,40807295	0,958594	1,928764	1,883516	2,473509	1,603024	0,5182	0,52459	0,215025
For -10 to 60	423,75	273,75	168,75	337,5	311,25	412,5	266,25	90	82,5	22,5
%	1,462973	0,97894596	0,599121	1,157259	1,108736	1,503348	0,925323	0,303337	0,288525	0,080634

Temp (oC)	31	32	33	34	35	36	37	38	39	40
-30	26963,75	28818,75	29655	28698,75	30515	28301,25	29220	27270	28383,75	28005
-10	27030,63	28833,75	29490	28698,75	30305	28323,75	29220	27322,5	28395	28065
10	27191,25	28848,75	29343,75	28702,5	30125	28323,75	29216,25	27468,75	28402,5	28136,25
25	27277,5	28856,75	29276,25	28713,75	30015	28320	29212,5	27547,5	28406,25	28203,75
40	27331,25	28862,5	29253,75	28710	29980	28305	29205	27630	28410	28290

60	27506,25	28866	29096,25	28721,25	29820	28293,75	29190	27772,5	28421,25	28406,25
85	27707,5	28862,5	29032,5	28728,75	29740	28278,75	29163	27963,75	28428,75	28503,75
Max Δfreq	743,75	47,25	622,5	30	775	45	57	693,75	45	498,75
%	2,726606	0,16373985	2,126297	0,10448	2,582042	0,158898	0,195122	2,518377	0,158416	1,768382
For -10 to 60	475,625	32,25	393,75	22,5	485	30	30	450	26,25	341,25
%	1,743653	0,11175895	1,344947	0,07836	1,615859	0,105932	0,102696	1,633542	0,092409	1,209945

Temp (oC)	41	42	43	44	45	46	47	48	49	50
-30	27978,75	28500	28473,75	27926,25	28567,5	29137,5	26475	27611,25	26945	28005
-10	27918,75	28503,75	28447,5	27933,75	28650	29130	26715	27731,25	27150	28065
10	27881,25	28507,5	28421,25	27941,25	28721,25	29107,5	26895	27765	27337,5	28143,75
25	27866,25	28507,5	28413,75	27956,25	28770	29088,75	27015	27832,5	27442,5	28200
40	27858,75	28511,25	28443,75	27971,25	28785	29055	27142,5	27915	27427,5	28250
60	27855	28518,75	28485	27993,75	28811,25	29017,5	27367,5	28016	27720	28278
85	27858,75	28518,75	28526,25	28023,75	28818,75	28957,5	27611,25	28151,25	27810	28305
Max Δfreq	123,75	18,75	112,5	97,5	251,25	180	1136,25	540	865	300
%	0,444086	0,06577217	0,395935	0,348759	0,873306	0,618796	4,205997	1,940178	3,152045	1,06383
For -10 to 60	63,75	15	71,25	60	161,25	112,5	652,5	284,75	570	213
%	0,228771	0,05261773	0,250759	0,214621	0,56048	0,386747	2,415325	1,023085	2,07707	0,755319

A.8
The high frequency from POR pulse



REFERENCES

[1] Marvin E.Frerking, “UFFC Fifty Years of Progress in Quartz Crystal Frequency Standards”, Proceedings of the 1996 IEEE international Frequency Control Symposium, p.p.33-46, 1996

[2] John R.Vig, “introduction to Quartz Frequency Standards-Quartz and the Quartz crystl unit”, in IEEE ultrasonics, ferroelectrics and frequency control society, by the army research laboratory, Fort Monmouth, USA, October 1992

[3] Serge Galliou and Marc Mourey, “Temperature Processing of an Ultra Stable Quartz Oscillator”, in IEEE transaction on ultrasonics, ferroelectrics and frequency control, vol 48,pp.1539-1546, November 2001

[4] Brian Warren, article titled “Silicon oscillator-Benefits of Silicon Oscillator” on EzineMark.com

[5] R. Colin Johnson, article titled “CMOS oscillator said to beat quartz”, on EE times, 22/10/2010

[6] Patrik Larsson, “A 2-1600-MHz CMOS Clock Recovery PLL with Low-Vdd Capability”, in Solid-State Circuits, vol 34, p.p.1951-1960, December 1999

[7] Erdogan Ozgur Ates, Devrim Yilmaz Aksin, “Fully Integrated 50 MHz Frequency Reference with 1ppm Absolute accuracy within 0-80^{o}C”, Ph.D. Research in Microelectronics and Electronics (PRIME), 2010 Conference, p.p1-4, July 2010

[8] Federico Vincis, Luca Fanucci, “A trimmable RC_Oscillator for automotive applications, with low process, supply and temperature sensitivity”, Electronics Circuits and Systems, p.p.1-4, Dec 2005

[9] T.O’Shaughnessy, “A CMOS Self Calibrating, 100MHz RC-Oscillator for ASIC Applications”, in Proceedings of the eighth annual IEEE International ASIC conference & Exhibit 1995, Austin TX, Sept. 1995, pp279-282.

[10] Qinyu Zhou, Liming Li and Guican Chen, “An RC Oscillator with Temperature Compensation for Accurate Delay Control in Electronic Detonators”, Electron Devices and Solid-State Circuits, p.p.274-277, Dec 2009

[11] Bo Wang, Myeong-Lyong Ko, Qi Yan, “A high accuracy CMOS on chip RC oscillator”, Solid-State and Integrated Circuit Technology, p.p.400-402, Nov 2010

[12] Zhi-Ming Lin, Kuei-chen Huagn, Jun-da Chen and Mei-Yuan Liao, “A CMOS Voltage Controlled Oscillator with Temperature Compensated”, AP-ASIC 2000, p.p. 85-86, August 2000

[13] Jan R. Westra, “High performance Oscillators and Oscillator Systems”, essay for his PHD project, TUDelft- Phillips Laboratory, May 1998

[14] Kirill Kozmin, Jonny Hohansson, Jerker Delsing, “A low power, propagation delay stable, continuous time comparator”, Norchip Conference, p.p.261-264, Nov 2004

[15] Jia Chen, Satoshi Kurachi, Shimin Shen, Haiwen Liu, Toshihiko Yoshimasu, Yong Ju Suh, “A low kickback-noise latched comparator for high speed flash analog to digital converters”, Communications and Information Technology, p.p.250-253, Oct 2005

[16] Paul M. Furth, Yen-Chun Tsen, Vishnu B. Kulkarni, Thilak K. Poriyani House Raju, “On the design of low power cmos comparators with programmable hysteresis”, Circuits and Systems, p.p.1077-1080, Aug 2010

[17] Pedro M. Figueiredo, Joao C. Vital, “Low kickback noise techniques for cmos latched comparators”, Circuits and systems, p.p.537-540, May 2004

[18] Krishnakumar Sundaresan, Phillip E. Allen, Farrokh Ayazi, “Process and Temperature Compensation in a 7-MHz CMOS clock oscillator”, Solid State Circuits, p.p. 433-442, Feb 2006

[19] Yusuke Tokunaga, Shiro Sakiyama, Akinori Matsumoto and Shiro Dosho, “An on-chip cmos relaxation oscillator with voltage averaging feedback”, Solid State Circuits, p.p. 1150-1158, June 2010
[20] C.J.M. Verhoeven, A.V. Staveren, G.L.E. Monna, M.H.L. Kouwenhoven, E.Yildiz, “Structured Electronic Design”, Kluwer academic publishers, 2003