

DELFT UNIVERSITY OF TECHNOLOGY

Characterization of standard cell libraries in cryogenic applications

by

Herman Kroep

A thesis submitted in partial fulfillment for the
degree of Master of Science

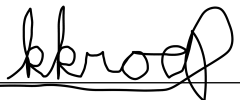
in the
EEMCS faculty

December 2018

Declaration of Authorship

I, Herman Kroep, declare that this thesis titled, ‘Characterization of standard cell libraries in cryogenic applications’ and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:  _____

Date: 12-7-2018 _____

“One cannot spell Cryogenic without Cry.”

Elizabeth Hatfield - 2018

DELFT UNIVERSITY OF TECHNOLOGY

Abstract

EEMCS faculty

Master of Science

by [Herman Kroep](#)

In order to improve the ability to design digital ICs for cryogenic temperatures, the objective of this work is to characterize the timing performance of standard cells in the TSMC 40 *nm* technology at a temperature range between 4 *K* and 300 *K*. A design was made to perform automated on-chip characterization of standard cell propagation delay, that makes use of random sampling. An ASIC was implemented and fabricated based on this design. A test setup was designed and build to perform the characterization. Besides that a design for a scalable gray-counter was developed and implemented that promises excellent performance for power critical applications.

Acknowledgements

I would like to thank my family and friends for supporting me through difficult times, and celebrating with me for my successes. From the group these include Ashish Sachdeva and Elizabeth Hatfield who were my roommates, and I could regularly share good laughs with. I would like to thank Job van Staveren, Jaco Salentijn, Edwin Shriek and Shiram Balamurali for sharing their knowledge developed during their master projects to help me with mine. I would like to thank Augusto Carimatto for sharing his knowledge and educating me on how to use chip design tools properly. I would also like to thank the overall group for providing feedback on my design ideas. A special mention for Alex Janssen, who helped me in his free time to brainstorm on the novel gray-counter design. Finally, I would like to thank my supervisor Edoardo Charbon for guiding me through the project, and providing valuable feedback on my designs. He provided me with the opportunity to experience high end chip design first hand, an experience I will carry forward for the rest of my life.

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Cryo-CMOS and the race for Quantum Computing	1
1.2 Characterization of standard cell libraries	2
1.3 Challenges of deploying CMOS at cryogenic temperatures	3
1.4 Contributions	5
2 Conceptual and schematic design	6
2.1 Single ring oscillator	6
2.2 component and TDC pairs	7
2.3 RO + Separate difference amplifier	8
2.4 Modified RO with inverter insertion	10
2.5 Random sampling method	12
2.5.1 Small, simple and robust sampling structure	13
2.5.2 Efficient scaling and read-out	16
2.5.3 Support structures	17
2.6 Sequential characterization	20
2.7 Gray-counters	23
2.7.1 4-bit gray-counter	24
2.7.2 Scalable asynchronous gray-counter	26
2.7.3 Scalable synchronous gray-counter	27
2.7.4 Scalable uniform gray-counter	28
2.7.5 Hard coded decoder	29
3 Implementation and ASIC design	31
3.1 Verilog, Genus and Modelsim	31

3.2	DUT selection and test benches	32
3.3	Implementation and simulation of designs blocks	34
3.4	Complete Schematic Design	36
3.5	Layout with Innovus	39
4	Measurements and results	43
5	Discussion and Conclusion	46
5.1	Discussion	46
5.2	Conclusion and Future work	47
	Bibliography	49
	Schematic and layout of PCB	51
	Photos of hardware	53

List of Figures

2.1	Schematic of a characterization method using a TDC with multiple instances at the input	8
2.2	optimized separate ring design	9
2.3	Schematic of a ring oscillator based design that has an oscillation frequency dependent on either rising or falling edge of DUT	11
2.4	Random sampling principle	12
2.5	Schematic overview of characterization building block. The dashed area indicates the time sensitive area	13
2.6	something	16
2.7	Behavior of 2-MUX with inputs I0 and I1, control signal S and output Q. Illustrating a glitch when used in a variable length ring oscillator	18
2.8	Structure for inside a ring oscillator that allows switching to different propagation delays without creating glitches	19
2.9	Schematic overview of the on-chip random sampler	19
2.10	Schematic of sense amplifier	20
2.11	Schematic structure of sequential characterization block	21
2.12	Schematic of course DPDE with a range between -11 and +12 buffer elements	22
2.13	Schematic of a 4-bit gray counter	25
2.14	Schematic of the first part of a uniform synchronous graycounter	29
2.15	Decoding for 7-bit gray counter made of two 4-bit blocks	30
3.1	Genus schematic of a building block with as DUT a buffer	34
3.2	Virtuoso simulation of building block structure	35
3.3	Virtuoso simulation of the random sampler	36
3.4	Virtuoso simulation of an asynchronous gray counter with 3 gray-4 blocks and 10 bit range	37
3.5	Schematic of the entire chip	38
3.6	Virtuoso simulation of building block structure	39
3.7	Design of the padding	40
3.8	Final chip layout, the structure within the red square is the contribution of this work	42
4.1	Photo of the wirebonded chip	43
4.2	Photo of the soldered PCB with the wirebonded chip	44
4.3	Overview of test setup	45
4.4	Photo of the room temperature test setup. Attached to the ARTIX-7 board are a UART to USB device and a JTAG programming cable	45

1	schematic of PCB design	51
2	layout of PCB design. Polygon pours are hidden for clarity	52
3	Photo of designed PCB (left) and motherboard ARTIX-7 PCB (right) . .	53

List of Tables

2.1	Sequence of states for the 4-bit gray-counter structure	25
3.1	Subset of standard cells and their occurrence in the ITC-99 test bench set, Bold cells indicate sequential cells	33
3.2	Subset of standard cells per set on the chip. Every set is 128 cells in size.	34

Chapter 1

Introduction

1.1 Cryo-CMOS and the race for Quantum Computing

Quantum computers are one of the most exciting and rapidly developing fields of research in the past years. At the time of writing the most important next step is to produce scalable logical qubits. A qubit is a two-level quantum mechanical system that makes use of fundamental quantum properties like entanglement and superposition. A logical qubit is a qubit that can be used directly for programming algorithms. Operating physical qubits involves decoherence times and noisy measurements. One promising method to achieve an effective logical qubit is the use of a surface code [1]. The aim is to use a combination of physical qubits, that do suffer from short decoherence times, and combine them into a two-dimensional structure. A subset of the physical qubits are used as data qubits that contain the value of the logical qubit. Other qubits are used as ancilla qubits. These ancillas are repeatedly entangled with the error on the data qubits, and when reading out the ancilla qubits, the error collapses into being no error or a full error, for which can be corrected. If this process can be done quick enough with enough fidelity, the combined state of the physical qubits can be used as a full logical qubit [2]. The measurements and corrections need to be executed by a classical chip or computer. The requirements are very steep. When measuring the qubits produce a signal that is of an extremely low amplitude. The measured errors need to be processed to generate the gate operations that undo the measured errors in a very short time-frame. The physical qubits are controlled with extremely precise electrical waveforms. All of these challenges need to be met while the qubits operate at deep-Cryogenic temperatures. Furthermore, the end goal is a system that is reasonably scalable with the number of logical qubits. This work attempts to aid in the development of the control structures for such devices.

When using control logic deployed at normal room temperatures, wires of at least 1 meter need to be deployed to and from the qubits. When developing a scalable solution this approach is impractical. To address the challenges, one can choose to build classical hardware as close to the qubits as possible, which means it needs to operate at Cryogenic temperatures [3]. A specific temperature that is targeted at the Delft University of Technology is 4 Kelvin. The 4 Kelvin operating point is chosen as a balancing act between distance to the quantum devices, and the thermal budget. A second reason for choosing this is that this specific temperature can be achieved by using liquid helium, which has a boiling point at 4.2 Kelvin.

1.2 Characterization of standard cell libraries

A standard cell is a structure build out of CMOS transistors that implements a logical function. Standard cells are the building block to perform Very Large Scale Integration (VLSI) of chip design. A standard cell library is a list of standard cells that can be used as building blocks by a synthesizer to implement software described functionality in hardware. The characterization information on these standard cells is the bridge between the complex functions to be implemented and the underlying physics on the chip.

An important distinction needs to be made between combinatory and sequential standard cells. The functional behavior of combinatory standard cells can be captured with a boolean truth table, implementing a simple logic function. Examples are AND's, OR's, inverters and buffers. The functional behavior of sequential standard cells needs to be captured with a state transition table. The most basic and common form of a sequential standard cell is a latch or a register.

Every cell in the library has a hardware description, that includes the schematic, layout, and symbol abstract. Every cell includes a timing abstract, that holds the functional definition, power, area, and timing information of each cell. The most common format to hold the timing abstract is the Liberty format. Out of all these timing abstract properties only the power consumption and timing information are affected by changing the temperature, so these are the only ones that would need to be characterized, when porting a room temperature library to a cryogenic library. Out of these two properties, the timing information is by far the most critical parameter that also requires the most precision. Characterizing this parameter for the TSMC 40 *nm* standard cell library is the main focus of this work.

The timing values of interest for combinatory logic is the propagation delay indicating the time difference between a change at the input and a change at the output. This delay is measured as the time in the 50%-to-50% definition [4].

All the combinations of a changing input and the propagation delay to every output might not be correlated so they all need to be measured separately. Furthermore, whether the input goes from active high to low or vice versa also needs to be calculated separately. Due to how CMOS is designed, the pull up and pull down networks are different structures. The difference between the pull up and pull down network is especially important for Cryogenic CMOS characterizations, as the pull up network is expected to scale differently with temperature than the pull down network.

A well known program called SPICE (Simulation Program with Integrated Circuit Emphasis) can be used to simulate power and timing of a circuit. Physics models are loaded into the simulator with environmental parameters like temperature and operating voltage. This setup can be used to characterize standard cells through software. Programs like Liberate are used to characterize a completely library. Liberate is essentially a wrapper around SPICE that loops through all the components in a library and simulates them with a constant set of models and variables. If this possibility is available, it is the most convenient way to characterize an existing standard cell library for low temperatures. However the prerequisite for this method is that the models used by SPICE are sufficiently accurate for the environmental variables used. In this case the low temperature models aren't fully matured yet, causing a major challenge to perform this re-characterization. However, for room temperature, this method is dominating the field and the methods of performing this type of characterization are the most developed and well-researched.

Besides using models to characterize a device it can of course also be measured directly. In present time this method is not used to perform the entire characterization, but rather to test and verify the simulations. There exist setups that allow the DC characteristics of a device to be measured [5], [6]. However, in this work the focus is to directly measure the propagation delay of standard cells.

1.3 Challenges of deploying CMOS at cryogenic temperatures

Devices operating at cryogenic temperatures are limited by the fridge it is operating in. The fridge only has limited cooling power, creating a limited thermal budget that depends on the temperature where the electronics are deployed. While the most advanced

quantum computers at the Delft University of Technology are based on dry cooling methods, the liquid helium is an affordable method of creating a relatively affordable test environment for the supporting electronics. This work also makes use of liquid helium to perform characterization.

Deploying classical hardware at cryogenic temperatures creates a unique set of challenges. With temperatures this low, conventional characterizations and models for CMOS technologies no longer hold true. Temperature dependencies like dopant freeze-out, band gap widening and Fermi-Dirac occupation of interface traps are phenomena that are not properly accounted for in the standard room temperature models [7]. The lack of these models is especially problematic for designing time-sensitive applications, which are most of the devices that operate close to the qubits.

Besides changes in expected average behavior, the mismatch between equally designed devices also increases at cryogenic temperatures. The effect of temperature can be seen as having a Gaussian smoothing effect on the performance. Defects in the device will have a larger impact when there is less of this smoothing effect.

Therefore the need for proper characterization of Cryogenic CMOS is apparent. At the time of writing, because of the lack of models and knowledge, technologies needs to be characterized separately through empirical measurements. This work is performed for the TSMC 40 *nm* technology.

Besides characterizing the TSMC 40 *nm* library for Cryogenic temperatures at conventional voltages, there is a large opportunity for research and characterization in low voltage operation points. The consumed power scales quadratically with the voltage. A decrease in voltage of 30% already increases the power efficiency by a factor of two. The main drawback of decreased voltage is the speed of the device. So there exists an opportunity of trading speed for power efficiency in devices with low speed requirements. This idea can be extended to design ASICs that operate in a hybrid way where parts of the chip are powered with different rail voltage to optimize for both performance and power consumption.

In recent years there has been a lot of development in constructing transistor models for cryogenic temperatures [7],[8],[9].

Thermal energy is a type of kinetic energy that describes the energy contained in the movement of particles. Essentially thermal energy describes the amount of vibration of particles. One of the effects of a lower temperature is a change in the electron and hole mobility of the conducting material. Most notably the decrease in temperature reduces the effects of scattering due to electrons colliding with the lattice. However, the decrease in temperature also gives rise to freeze-out effects in semi conductors. Semi

conductors work with a Fermi level between the valence and conductance band. To make a transistor, doping is used to skew this energy level between two sides of the transistor, causing the electrons to prefer a direction of movement. However, in order for electrons to escape from the valence to the conductance band, they need an extra source of energy, as the Fermi level is not high enough. Normally temperature provides the energy required for the electrons to make the jump. However, when the temperature is at absolute zero, there is no available energy, and the semi conductor behaves as an insulator. Traces of this affect are already affecting the mobility at cryogenic temperatures, including the targeted 4 Kelvin of this work [9].

A lower temperature also makes the behavior of a transistor more susceptible to defects and traps, as the energy required to break loose from a trap is hard to come by. Because of this there is an increased problem with $1/f$ noise, which is mainly dependent on the difference in behavior due to fluctuations in minority carriers. These minority carriers get trapped by interface traps, defects in the interface between the silicon and gate oxide. The increased sensitivity to impurities also affects the mismatch of the devices. The presence of traps can vary a lot between devices, and can effectively increase the mismatch by a factor of approximately two. To the authors knowledge, the specific effects of cryogenic temperatures on mismatch are still actively being researched and no conclusive answer has been found yet.

1.4 Contributions

Contributions by the author to this Thesis are:

- Designs were developed to perform on chip timing characterization, where the amount of performance critical components is minimized
- A novel design for a scalable gray-counter was proposed that promises the advantages of a gray-counter, while addressing challenges in efficiency and scalability
- A test setup was implemented to perform the characterization, including a fabricated custom ASIC that implements the developed designs.

Chapter 2

Conceptual and schematic design

This chapter details the design methodology that was applied in this work. A detailed description is provided of the different designs that were developed and considered, including the final designs.

2.1 Single ring oscillator

When measuring the delay of a component, there are two aspects that need to be carefully considered: accuracy and resolution. In the case of this work the room-temperature delay of the fastest component, the inverter, is slightly below 10 *ps*. Therefore at least a resolution of 1 *ps* and reasonable accuracy is desirable. A hardware setup needs at least one instance of the target component, and a form of Time interval to Digital Converter (TDC). The accuracy of the measurement is a sum of the accuracy of the TDC, the mismatch of the component, and the variation due to how the two are connected. It is often a good idea to use a method to amplify the delay of interest. One way to achieve this is to construct a ring oscillator of the DUT. The oscillation period of a ring oscillator is related to the rising time $t_{L \rightarrow H}$ and falling time $t_{H \rightarrow L}$ by eq. 2.1

$$T = N(t_{L \rightarrow H} + t_{H \rightarrow L}), \quad (2.1)$$

where N is the number of components in the ring, which must be an odd number, and T the resulting oscillation period. This period can be obtained by attaching a counter to it, and counting the oscillations for any desired duration. It is important to note that with this method one cannot measure the difference between the rising and falling

times. It is however, a very simply and robust method of obtaining the mean delay of these two values.

This approach has an important disadvantage in that it discards all information about the difference between the rising time and falling edge propagation time. For proper characterization to be used in design synthesis this simplification is unacceptable, so a solution needs to be found that can extract this information properly.

2.2 component and TDC pairs

One approach is to pick a single component, and directly attaching it to a TDC. Because the targeted delay is not amplified one needs a TDC with a resolution of at least 1 ps . The input and the output of the DUT are then used as start and stop signals for the TDC. The delays of different components can vary quite a lot, and the design should be suitable to test at different voltages, so the maximum range of the TDC should at least extend to 1 ns . This means that the TDC should be at least 10 bits. Achieving a resolution of 1 ps with this technology is challenging but doable as sub-picosecond TDC's have been reported [10], [11]. However, if possible, it is desirable to avoid building a TDC with these performance requirements. Besides the problem of a complex TDC there is also the issue of scalability. If a component is characterized using only one instance of it, the measurements are extremely susceptible to variations. This is especially troublesome considering that variations have stronger effects at cryogenic temperatures. So if one wants to characterize multiple components, one should either make a separate TDC for every component to be characterized, or connect multiple components to the same TDC. The first option is questionable for multiple reasons. First of all, a high performance TDC should be very bulky and it would be wasteful to use a complete high resolution TDC to characterize a single instance of a component. Secondly, the TDC needs to be calibrated to account for device mismatch. Providing an accurate calibration structure for every TDC is prone to errors too, and makes the solution even more bulky. The second option where multiple instances are connected to the TDC, is more interesting. One can use multiplexing methods to achieve an effect like this. Calibrating would also be easier to achieve as one of the components connect to the TDC can be a simple short. A schematic of this approach is shown in figure 2.1. This approach gives rise to a new problem however, and that is how the logic connecting the component and affects the delay. This is a problem, as the required time accuracy is extremely small. The measured delay could very well be dominated by the surrounding logic. This problem, and the challenges of designing a robust high performance TDC make that this approach was dropped in favor of other options.

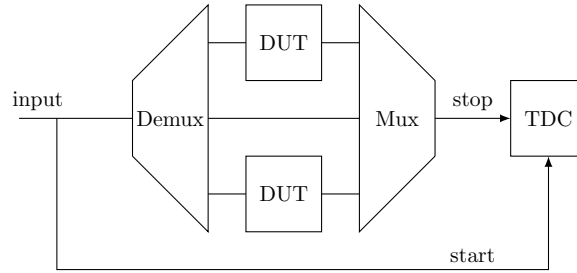


FIGURE 2.1: Schematic of a characterization method using a TDC with multiple instances at the input

2.3 RO + Separate difference amplifier

As many components need to be characterized, it is a good idea to design a simple and robust structure that can characterize multiple standard cells with acceptable resolution and accuracy. An easy way to extract the period from a ring oscillator is through the use of a simple asynchronous counter, as described in section 2.1. The main requirement for the ring oscillator is to have a period that doesn't violate the maximum frequency the counter can handle. This is not an issue as there should be many components in the ring anyway, to suppress variance. The ring oscillator is based on stacking components that behave like buffers or an even number of inverters, and a NAND or NOR to enable/disable the ring oscillator. So the DUTs need to be configured to behave like either buffers or inverters, to be able to construct a functioning ring oscillator. When the component is configured to behave like an inverter, only one input and one output are considered. If there are multiple inputs and outputs to a component, basically every component that is not an inverter or buffer, it can only characterize the delay of one of the propagation paths. In order to obtain the delay of all paths a separate ring oscillators needs to be constructed for every single configuration. Combining this with the fact that there are numerous items in a cell library, the need for a scalable design is apparent. A good way to scale this ring oscillator design efficiently is to attach multiple ring oscillators to the same counter. A selection method with a demultiplexer is then used to turn one of the rings on, and an OR tree is used to combine the ring oscillator outputs into one input for the counter. A schematic representation of this approach is shown in figure 2.2. This schematic already included how the system could be operated with a single shift register to select the correct ring oscillator and extract the results of the counter.

With respect to the ideas proposed in section 2.2, the ring oscillator approach is more robust and easy to design. The main issue behind this method is the lack of ability to measure the difference between the rising and falling time. One approach is to let the ring oscillators exist to measure the mean delay, and then separately measure the difference between the rising and falling time. In order to prevent the requirement of

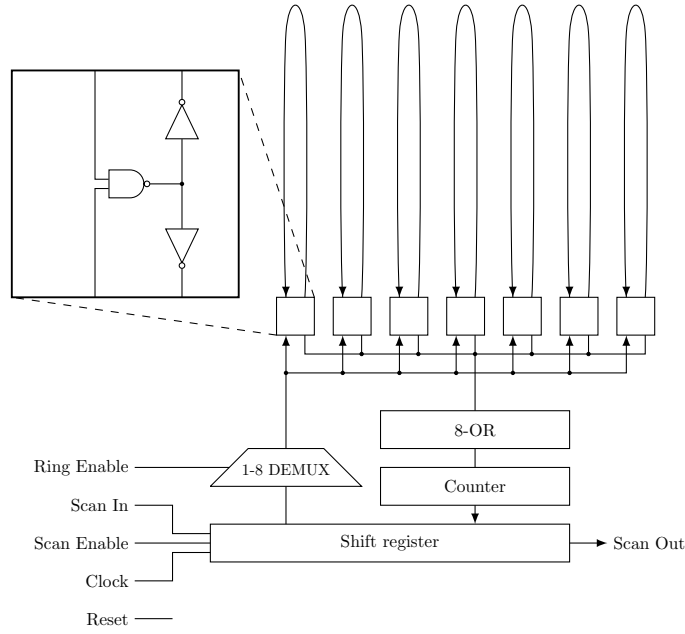


FIGURE 2.2: optimized seperate ring design

designing a highly precise TDC, this time difference needs to be amplified in some kind of way. What was tried is constructing a chain of the targeted components that connect a pulse generator to a TDC. The targeted component is configured in a method that makes it behave like a buffer. If the component is inverting by nature, an inverter is attached after each component. The TDC is configured to start measuring at the rising edge at the input, and stops at the falling edge at the input. Now a lengthy pulse is sent through the component chain. Every component in the chain observes first a rising edge, and then a falling edge at the input. The delays are applied accordingly, so for components where these delays differ, the shape of the pulse will change. When the front of the pulse, the rising edge, propagates faster through the chain, the pulse will grow in width. If the rising edge propagates slower, the pulse will shrink. The pulse width at the end of the chain relates to the chain as shown in eq 2.2

$$T_{pulse} = T_{init} + N(t_{H \rightarrow L} - t_{L \rightarrow H}), \quad (2.2)$$

where T_{init} is the initial width of the pulse, and N the number of components in the chain. With these two factors known, the difference between the rising and falling time can be extracted from the measured pulse width.

With this method the difference between rising and falling edge propagation delay can be amplified directly by the number of components in the chain, but a proper TDC still needs to be designed that fits the circumstances. It is desirable to use a TDC with low resolution, as it decreases the overall size of the structure, and the variability of the pulse width. The TDC should also have quite a bit of dynamic range, as different

components can have large and opposite differences between rising and falling time. The proposed design for the TDC is a hybrid between a ring oscillator and a counter. The average delay of an inverter in the used technology is slightly under 10 ps, which is sufficient as an LSB in the circumstance. A ring of shadow latches is connected to the ring oscillator, so that the phase of the ring can be accurately measured with a resolution equal to the delay of the inverter. An asynchronous counter can run at a frequency of 5 GHz at room temperature, so a ring with 31 components can drive the counter with a comfortable margin. An important consideration for designing TDC's with multiple parts, is to stitch them together without creating glitches. A glitch occurs when the two different time components make a change at exactly the same time. In practice this is not possible, so a glitch is made during the mismatch. In order to completely erase this issue, redundancy is introduced. A complimentary single bit counter is introduced, which counts at the falling edge instead of the rising edge. This complementary bit adds enough redundancy to completely rule out any possibility of glitches occurring, making the TDC more robust. As for the dynamic range of the TDC, it is relatively inexpensive to add more bits to the counter to increase this range, so for this design that is of no concern.

If one doesn't want to build one full TDC for every component chain to be characterized, there is a need to build logic surrounding the chains to connect multiple to the same TDC. However, this creates an uncertainty in the amount of delay or pulse shaping these surrounding structures add, decreasing the accuracy of the system.

2.4 Modified RO with inverter insertion

The design in section 2.3 uses idea of making a chain of components that has a different propagation delay for the rising and the falling edge. This gives rise to the idea of constructing a ring oscillator that runs on either the rising, or falling edge propagation speed. If this can be achieved, the need for a TDC would be removed and substituted for a simple counter. Any pulse shaping or delay concerns outside the ring oscillator itself would be removed, increasing the accuracy of the system.

The main challenge is to design the ring oscillator. As it is desirable to be able to turn the ring on or off, a NAND or NOR can be used to act as an inverting component and on/off switch. The remaining components can then remain buffers like in section 2.3. This iteration doesn't work however, as the total period of this ring oscillator would still be the average between the falling and rising edge. For half a clock cycle the duty cycle would shrink, and for half a clock cycle it would increase back to it's original size. A mechanism needs to be constructed, where either only the rising time or falling time

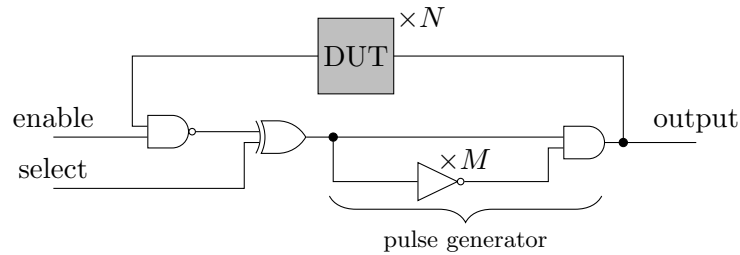


FIGURE 2.3: Schematic of a ring oscillator based design that has an oscillation frequency dependent on either rising or falling edge of DUT

of the component determines the oscillation frequency. In order to achieve this effect a pulse generator is made that reacts to either the rising, or the falling edge. Hooking the input to an XOR gate allows this to be selectable. In principle this solution creates the desired effect. A schematic representation is shown in figure 2.3.

The idea behind this approach is very promising, as it keeps some of the nice advantages of a ring oscillator approach. There is however one important problem that is not addressed with this design: The component chain changes the width of the pulses, and while the idea is to reset the pulse with the pulse generator there are some problems. The first one is that the pulse generator can only generate a pulse that is larger than the incoming pulse. So if the pulses are shrunk in the component chain, the pulse generator will have no positive effect. If the pulses are widened, this is less of a problem, but there still exists the concern that the duty cycle will grow to 100%. If this happens the oscillation comes to a complete halt. The first problem, when the pulses are shrinking, can be addressed by placing the component chain in a sandwich of two XOR gates. When these XOR gates are manually controlled the tester can ensure that the pulses in the component chain are always widening. The second problem however, means that the frequency is low enough so that a 100% duty cycle never occurs. Unfortunately there is no easy and robust way to achieve this. The frequency is directly dependent on the exact ring oscillator in question. The frequency can be decreased by adding more components, but if the same components are added that are already present, the pulse widening will increase proportionally and never solve the issue. Therefore in order to get the frequency up without affecting the pulse widening effect, components need to be used that do not contribute a lot to pulse widening. A logical candidate is the inverter, and indeed, if enough inverters are added the issue is resolved. The question remains however, as to how many inverters should be added to stabilize the system. One does not want too many inverters as it adds more components that can introduce more errors and skew the measurements. The amount of inverters needed also varies wildly across different standard cells. Added to that is the expectancy that the performance of the pull up and pull down network will not be affected equally when the temperature is reduced.

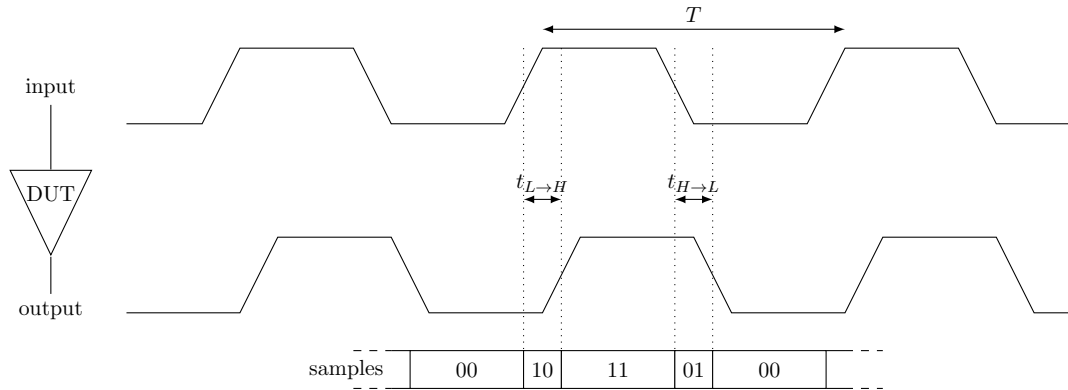


FIGURE 2.4: Random sampling principle

Ideally the system would also be able to characterize the component under different operating voltages and temperatures, making the system even more inconsistent across its use cases. Unfortunately, a satisfying simple and robust solution to this issue was not found.

2.5 Random sampling method

The work of [12] suggests an idea that makes use of random sampling to get the desired propagation delay data. The DUT is put in a propagating mode, acting as either an inverter or a buffer, and the input is attached to a clock with a known frequency. A set of latches sample the input and output of the component at the same time. The timing of when the latches do this needs to be completely uncorrelated to the clock connected to the input of the component. This method uses the principle that directly after an edge at the input, the following edge of the output is slightly delayed. Therefore there is a chance that the latches sample exactly during a transition period. If the sampling signal of the latches is truly random, the chance of sampling during a transition related directly to the clock period and propagation delay as shown in eq 2.3. An illustration of this effect is shown in figure 2.4.

$$P_{L \rightarrow H} = \frac{t_{L \rightarrow H}}{T}. \quad (2.3)$$

Supplying a known clock frequency to the input of a component is not an issue, so the only thing that needs to be considered, is how to extract an accurate and high resolution probability of observing a rising or falling edge transition, as this can then directly be translated into the desired propagation delay. The answer to achieving this is two fold. The first part is to create a simple, small and robust design that connects two latches to

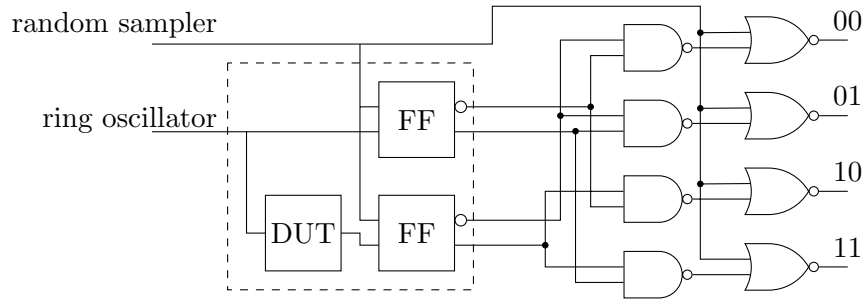


FIGURE 2.5: Schametic overview of characterization building block. The dashed area indicates the time sensitive area

an inverting or buffer component and an oscillating input, so that random samples of the latches achieve the required accuracy. The second part is to design a structure around this, that can accumulate a large number of samples. By increasing the sample size the accuracy of the probability density measurements go up, resulting in more accurate measurements overall.

2.5.1 Small, simple and robust sampling structure

The samples of every register need to be acquired. To get this info one can convert the outputs to pulses and count them with simple asynchronous counters. The conversion from registers to pulses is done in two steps. The first step is a demultiplexer that converts the two outputs to four signals representing 00, 01, 10 and 11. If the rising edges of these outputs are directly counted, multiple consecutive identical observations will be counted only once. A second structure is added to guarantee that every observation results in a separated countable pulse. In order to achieve this all signals are put into OR structures, together with the sampling signal connected to the flip-flops. This way the OR only conducts information for half of the time, creating the desired pulse effect. The resulting structure is illustrated in 2.5. Note that the area in the dashed square is the only part where the device is critically sensitive to delay.

To estimate the expected performance of this structure there are two things to consider. First there are physical limitations, and secondly there are imperfections in the chip design that can affect the performance. All delay sensitive components are found in the building block. The base accuracy of the system is determined by the accuracy and resolution of the period of the oscillating input, and the resolution and accuracy of the constructed probability density function. Assuming an ideal system with an ideal input period of exactly 1 ns , in order to get a 1 ps resolution, one needs an accuracy of at least 0.1% on the probability density function. One can calculate the expected accuracy of the probability distribution function based on the amount of samples. The measurement result is divided into four parts representing the mean probability of measuring that

state. This can be modeled as a Bernoulli distribution. The variance of the probability measurement σ_p^2 of this system can be obtained with

$$\sigma_p^2 = \frac{1}{N} \sum \frac{(X - \mu)^2}{N} \quad (2.4)$$

$$= \frac{1}{N} \sum \frac{(1 - \mu)|0 - \mu|^2 + \mu|1 - \mu|^2}{N} \quad (2.5)$$

$$= \frac{1}{N} \sum \frac{\mu(1 - \mu)}{N} \quad (2.6)$$

$$= \frac{\mu(1 - \mu)}{N}, \quad (2.7)$$

where μ is the probability of measuring the state, and X a discrete random variable for which $\Pr(X = 1) = \mu = 1 - \Pr(X = 0)$. This shows that smaller probabilities can be obtained with higher accuracy, which is expected. However, the final measurement is obtained by multiplying the probability with the clock period. If we take the propagation delay as Δt and the clock period as T we can derive the variance on the time measurement σ_t^2 with

$$\sigma_t^2 = T \cdot \sigma_p^2 \quad (2.8)$$

$$= \frac{T \frac{\Delta t}{T} (1 - \frac{\Delta t}{T})}{N} \quad (2.9)$$

$$= \frac{\Delta t}{N} (1 - \frac{\Delta t}{T}). \quad (2.10)$$

The result in eq 2.10 holds some important design guidelines. First of all the variance is inversely proportional to the amount of samples, which is expected. Secondly the equation shows that the period T should be as small as possible. There are some restrictions on this period however. One doesn't want the value of the input to change before the output is settled. Otherwise the measurements are interpreted the wrong way. Therefore in practice μ should be at least less than half, and for safety less than a quarter. A second observation is that the accuracy is proportional to the amount of samples acquired. The cost of acquiring more samples is mainly the measuring time, so there is a relatively low cost to increase the accuracy this way. This only works when the accuracy is dominated by the amount of samples however, and the expectation is that in most cases this will not be the limiting factor.

There are several aspects of this design that can cause skewed and/or noisy measurements. The first one is that the two registers don't have the same structure driving the input, and therefore there might be a skew because of this difference. In order to address this issue, at the beginning of the building block, the oscillating input is buffered, but the skew cannot be completely removed this way. A second source of error is found in the

registers. The characterization makes heavy use of flipflop that experience setup/hold violations. The assumption is that the setup and hold violations are neutralized by the fact that both the input and output measurements experience the same issue. In simulations this assumption holds true, but put on a physical chip this is a potential extra source of variance. There is a second problem with setup and hold violations that is more troublesome however. The setup and hold violations are generally not equal for rising and falling inputs. This means that when inverting components are tested, this difference will skew the measurements. Assuming the registers are correctly characterized, in theory, this offset can be compensated for. However, the accuracy of the setup/hold measurement then also affects the accuracy of the propagation delay characterization. A different solution to this issue is to only characterize buffering structures. If inverting components need to be measured, an inverter is added either before or after, to make sure that the two registers receive the same type of edge. However, this time the accuracy of the characterization of the inverter affects the resulting accuracy. Not only that, but the inverter itself is naturally an inverting component. So there still needs to be a way to extract the rising and falling delay of an inverter. For the chip in this work the decision was made to not design around this issue and instead measure it's significance when the chip is developed. As the library is already characterized for room temperature by the foundry, this can be used to verify the performance of the system.

Besides errors caused by the components, there are also errors due to the fabrication process. There will be variance for components on the same wafer, and a more significant variance is found for devices on different wafers. The only ways to suppress the fabrication variance is through calibration and by characterizing more chips. The amount of chips characterized causes the variance to drop proportional to the amount of chips characterized.

Besides the errors listed above, that are limited by the design itself, there are some errors that can be caused due to how the design is implemented on the chip. One of those factors is the wires that connect the DUT to the two registers. A variation in the propagation delay of these wires can skew the device. This is not only affected by the length of the wires, but also by the parasitic capacitance. A precise analog layout is required to minimize errors due to this. If the components are placed by an automated physical design tool, this cannot be guaranteed. This is an important detail, because an automated physical design tool is exactly what is used for the layout of the chip. This will be further discussed in Chapter 3.

In order to estimate the performance a lot of information is required on how it is implemented and as for this design, a lot of information is missing because the required control over the layout couldn't be asserted when designing the chip. The best way to estimate

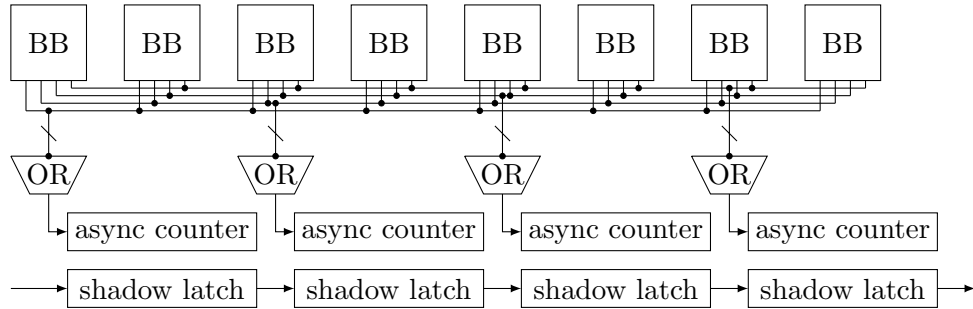


FIGURE 2.6: something

the performance of the chip is to perform a lot of characterizations, and comparing them to the characterizations supplied by the foundry. These supplied characterizations are assumed to be perfect.

Finally there is a potential source of error in that the register has a lot less problem not changing value, than changing value. Therefore the registers have a preference towards the previous measured value. Because of this one wants a roughly equal probability of measuring a '1' as a '0' in the registers. This filters the effect from the measurement, as all measurements are equally affected.

2.5.2 Efficient scaling and read-out

With the building blocks in place one would ideally be able to place a lot of these on a chip, while restraining the need for a lot of I/O and additional logic. For the sake of area efficiency and in order to simplify the read-out structure, every block is connected to the same set of asynchronous counters. As pulses are counted, and paths can be disabled manually, a simple or-tree is sufficient to drive the asynchronous counters. Now the counters need to be read out. Initially the approach was to design the asynchronous counter and the shadow latch as a hybrid structure that would save on area by using less registers. However, a good design principle is to keep the chip design simple and stupid. The idea was discarded for a simpler and better understood approach, where every counter was supplied with a separate shadow latch. This is less area efficient than what was tried, but as this part of the design is not critical, it is not worth the risk to optimize this part that way. The resulting readout structure is illustrated in 2.6. The readout design proposed here scales well, as only the size of the or trees is affected by the amount of building blocks used in the design.

2.5.3 Support structures

In order to enable characterizing the building blocks, the correct support signals must be supplied. Specifically there are two signals that need to be constructed: an oscillating source with constant frequency, and a random sampling signal. As discussed in Section 2.5.1, the optimal frequency of the ring oscillator varies on the component that is characterized. Because of this it is desirable to be able to configure the oscillation frequency post fabrication. The oscillator is implemented as a simple ring oscillator with some added functionality. The added functionalities are the options to divide the clock to lower frequencies, disable the internal ring oscillator, and use an external oscillation signal.

The random sampler needs to generate a signal that samples at a decent speed so the measuring time isn't too long. The most important feature is that the the time of sampling needs to be uncorrelated to the oscillator period. This is essential, as the characterization only works if this holds true. In order to achieve this property, the design of the random sampler needs to be carefully considered. One solution would be to use an alternative entropy source. For example, one could make a SPAD, and use the event of detecting a photon to flip the output of the random sampler. As the time for which a new photon arrives is a random source, there will be no correlation between the random sampler and the ring oscillator. For this project it was chosen that such an approach would be too complex to justify the application. A different method would be to have the random sampler be generated outside of the chip. This would decouple the design of the random sampler with the ring oscillator, an FPGA for example. The FPGA can be run at a different clock speed, and generate a digital output that is used as a random sampler signal. There is still a danger in using such a method, in that there is a danger that the frequency of the FPGA is using a system clock with a periodically stable frequency. Therefore the frequency of the oscillator and FPGA output can potentially be correlated. The approach that was tried in this work is to design a random sampler that takes a very long time to repeat. The idea is that if it takes long enough for the random sampler to repeat its behavior, it is less likely that the correlation will cause trouble for the measurements. In order to achieve this the random sampler was designed as a ring oscillator, with a part of the ring designed in such a way that it can change propagation delay. The decision of when to change length is made by a Linear Phase Shift Register (LFSR). The LFSR generates a pseudo-random stream of bits. The clock of the LFSR is driven by the random sampler output, so depending on the amount of bits in the LFSR, it takes a relatively long time before the LFSR cycled through all its states and the cycle starts over again. The resulting signal has a period that is effectively

unstable. The idea is that a signal with a stable period cannot easily be correlated to a signal with an unstable period.

Designing a Random sampler with a variable length is non-trivial however. One cannot simply use a MUX with two inputs with varying number of buffers. For a static solution where the control bit of the MUX doesn't change during operation this solution works. However, when the control signal changes dynamically, it creates the possibility of glitches. This problem is illustrated in Figure 2.7

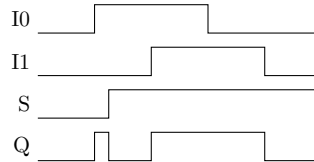


FIGURE 2.7: Behavior of 2-MUX with inputs I0 and I1, control signal S and output Q. Illustrating a glitch when used in a variable length ring oscillator

To achieve a glitch-less ring oscillator with variable lengths, a custom structure was designed. The structure combines two paths with different propagation delays without generating glitches. The idea is that the output of the path can only switch to zero when the input clock switches to zero, by using a feedback loop. This structure ensures that a change of the control bit cannot result in a change from 1 to 0, but only from 0 to 1. Basically whenever the control signal switches during a high, it guarantees that the structure will finish the high until the clock drops down again. The duty cycle of the resulting signal will therefore be slightly higher than 50%, because the system prefers a 1 over a 0. Assuming that the timing of the local circuit is handled properly and the input clock is slow enough, the glitches should no longer be occurring. The chosen design is illustrated in figure 2.8. The buffer in the structure represents an arbitrary delay devised of any number of buffer elements to get the desired delay.

This variable propagation delay structure can now be used to have the oscillation frequency of the ring oscillator change with a digital control signal. If a random digital input is provided, the period of the random sampler output signal is no longer periodic. A true random source could be used to drive this, but for the scope of this application that requires an unnecessary investment. The source of random digital bits is therefore chosen to be a linear phase shift register (LFSR). This is a pseudo random structure that, if configured correctly, cycles through all possible binary states before it repeats again.

With that all the parts of the on-chip random sampler are in place. The complete design is shown in figure 2.9. Note that the ring oscillator is very similar to the random sampler, besides missing some structures, and having a ring oscillator with less elements.

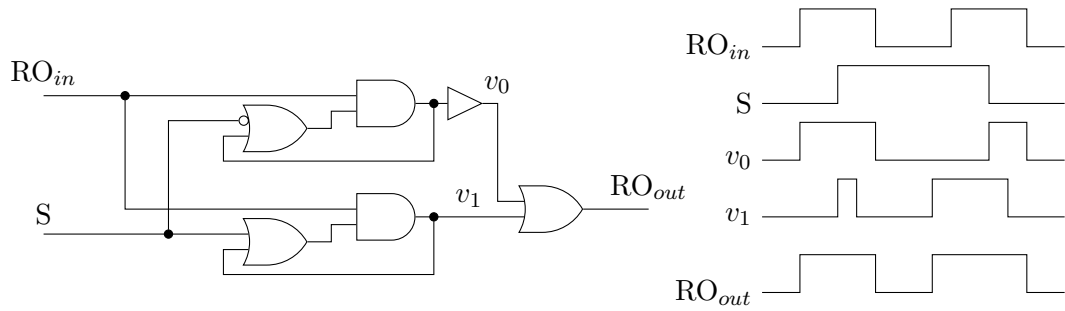


FIGURE 2.8: Structure for inside a ring oscillator that allows switching to different propagation delays without creating glitches

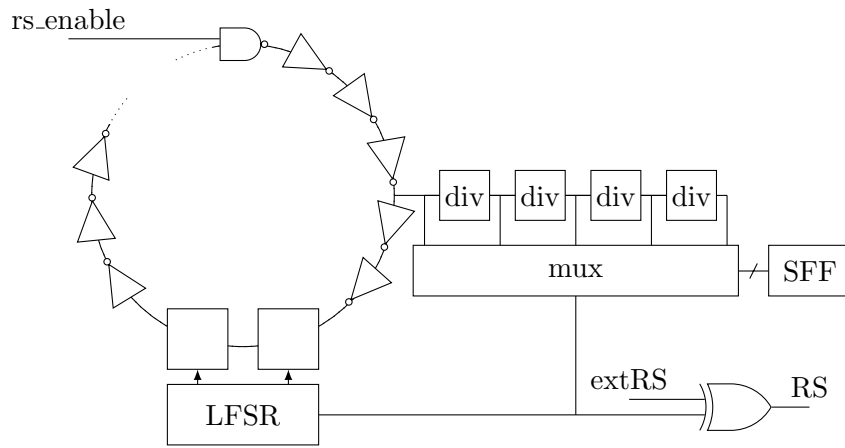


FIGURE 2.9: Schematic overview of the on-chip random sampler

In order to test for low voltages, there need to be level shifters to convert the low voltages to the nominal voltage where characterizing structures operate. A conventional method for achieving this is using a sense amplifier, which is commonly seen in SRAM structures. For SRAM structures, the sense amplifiers are part of the critical path directly determining the maximum speed of the device or component. In the case of this work however, the level shifters are never put into a timing critical path. The chosen approach is a phase alignment structure with a differential amplifier. A schematic of the chosen design is shown in figure 2.10. Note that the inverters are powered in the low voltage domain, and the transistors in the differential amplifier powered by the high voltage domain. An important design decision is how the differential amplifier is build. In conventional differential amplifiers the transistors are usually three NMOS and two PMOS transistors. However, this doesn't work here, because as the difference amplifier and phase alignment structure share a common ground, the input will be close to zero, where NMOS transistors will be less fast than PMOS transistors, or even completely off. Another important observation is that more power is dissipated when the input is not properly phase aligned, creating a temporary short between power and ground. If this is a likely occurrence additional phase alignment stages can be added. For this design,

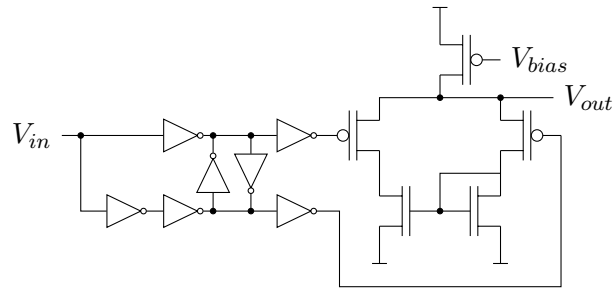


FIGURE 2.10: Schematic of sense amplifier

two phase alignment stages where chosen.

2.6 Sequential characterization

Sequential components are more complex to characterize than combinatory components. They are not just depended on the current input, but also on the history of inputs before that. For measuring the propagation delay, there are different measurements to be made for the different states of the sequential cell. For sequential cells operating on the edge of an input clock, there are additional measurements required for characterizing. The setup and hold time of the cell indicate how long the data input must be stable before and after the clock edge to ensure that a state transition will be properly executed. Measuring this time information is very different with respect to measuring the propagation delay.

When measuring the propagation delay of a register, one can make use of the exact same setup as used to characterize the propagation delay of standard cells. However, where combinatory cells are connected in such a way that they behave as inverters or a buffers. For sequential structures the propagation delay is from the edge of the clock to a change in the output. One can get this behavior by connecting the oscillating input to the clock input of the sequential cell, and then connecting the inverse of the output back to the input. That way, for every rising clock edge, the cell will invert. In order to prevent inaccuracies due to setup and/or hold time violations, the oscillating clock should be kept at a low speed. This solution is convenient because it can directly be integrated into the existing propagation delay characterization structures.

Sample and hold times are difficult to properly characterize, because there is no simple way of amplifying the setup and hold time. The setup and hold times work as thresholds. Going over the threshold results in an undefined state of the cell, while below the threshold results in a successful state transition. The only way that the author knows to test through hardware whether an input sequence violates the sample/hold times is to supply the signal and check whether the state transition occurred or not. This is

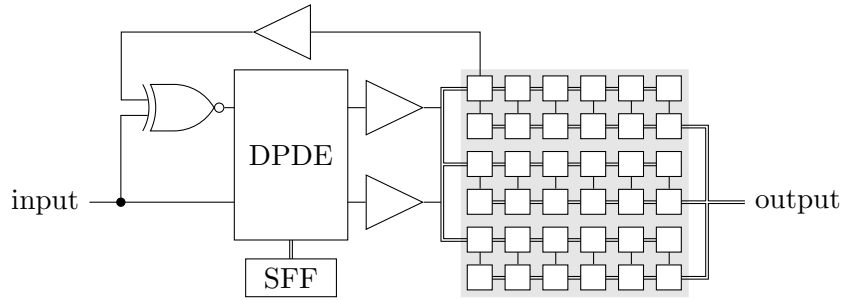


FIGURE 2.11: Schematic structure of sequential characterization block

problematic as there is no way to amplify the setup/hold time to have an easier time on performing the measurements. The target resolution and accuracy are 1 ps . The target range is at least $\pm 100\text{ ps}$ for high voltages, and scale up for lower voltages to cover unexpected changes due to cryogenic temperatures.

The proposed solution by this work is to construct a structure that is able to accurately supply a clock signal and a data signal with a programmable delay between the edge of the data signal and the edge of the clock signal. These signals are then fed into an set of cells that need to be characterized. After every pulse is applied, all the states are stored in shadow latches, and shifted out through a shift register with a slow clock. This process is repeated to accumulate a lot of data in a computer, that can then extract the setup and hold times for each of the cells. A schematic of the sequential characterizer design is shown in figure 2.11. On the schematic one can see the feedback loop with a buffer and the XNOR gate. The only interesting input value is the opposite of what is currently stored. Therefore the structure tries to always change the input rather than stay the same. The buffer is in place and should provide a reasonable delay, so that there is no sample/hold violation due to this feedback loop. The device is controlled by providing it with a rising edge at the input. This will produce a rising edge in the clock path, and a data change edge in the data path. Both paths go through a Digitally Programmable Delay Element (DPDE) that is configured with bits from a shift register. The outputs of the DPDE are put through a large fan-out buffer, so that multiple sequential cells can be driven and characterized with the same structure. After both the clock and the data signal are supplied, the shadow latches store the values in the DUT's and shift them out to an external device for further data storage and processing.

The critical part of the design is the Digitally Programmable Delay Element (DPDE). The accuracy, resolution and range are mainly determined by this component. In order to achieve a long range, the chosen approach was to have a separate course delay element and a fine delay element. Delay elements are convenient to stack on top of each other as long as the course element is sufficiently accurate. This is not the case with Time to Digital Converters for example, where stacking a course and fine TDC on top of each

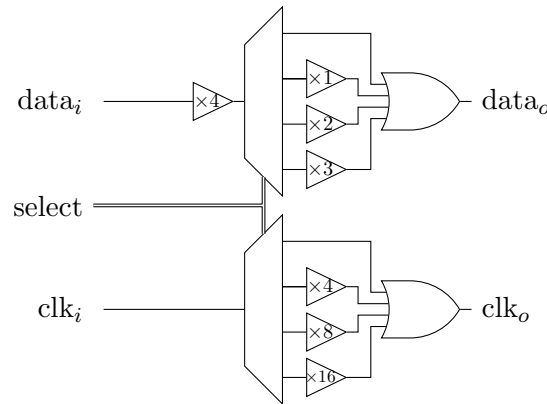


FIGURE 2.12: Schematic of course DPDE with a range between -11 and +12 buffer elements

other is not a trivial problem. The design of the course delay element consists of a set of MUX's, a bunch of buffers, and two OR trees. The chosen design is shown in figure 2.12. The only time we are interested in is the time difference between the edge of the clock line and the data line, so absolute delays need not be considered. The provided structure can provide a course delay between -11 and +12 buffer propagation delays. As a buffer produces a delay of approximately 18 ps , this results in a range of approximately $\pm 200\text{ ps}$.

The fine DPDE needs to cover the range of at least a single buffer propagation delay to combine with the course counter. The LSB of the fine DPDE then determines the resolution and accuracy. The fine DPDE also determines the accuracy, because the course DPDE design already provides enough accuracy to meet the requirements. A single step of the course DPDE is approximately 18 ps , so a fine DPDE with 1 ps resolution should have 5 bits in range. This would give the fine DPDE a range of 32 ps , which provides a substantial margin.

There are several ways to delay the edge of a digital signal with small and precise time intervals. Two commonly used approaches are shunt capacitor elements and current starved delay elements [13]. The shunt capacitor approach uses transistors to connect or disconnect capacitors to the data line. The signal edge is delayed due to the larger capacitance. The current starved element is a buffer with two standard inverters. The first inverter is sandwiched between a structure that starves it from current using two current mirrors and a bias voltage. A third method is to alter the voltage of the inverter. Lower voltage decreases the speed of the device. If this can be done locally one can delay the signal that way.

At first effort was put into the shunt capacitor approach, but the problem of this approach is that it cannot be calibrated after manufacturing. The other two approaches

allow the delay to be tunable with an external voltage, which allows for a superior flexibility. This is especially important when measuring in extreme environments, as this not only affects the component, but also the characterizing structure. The design chosen was altering the voltage, because it is the most simple solution. To build a DPDE with this building block, the idea behind a vernier line was used. As we are only interested in the relative delay between the clock and data path, a vernier line approach allows a fine resolution without the need for the added delay to be as small as the LSB. Both the data signal and the clock are put through a chain of buffers, where the buffer chain of the clock has a variable voltage. After every buffer a branch goes into a large MUX, that allows one to select how many buffer elements are passed before the signal is routed to the DUT's.

2.7 Gray-counters

The characterization makes use of 4 large asynchronous counters. In searching for power efficient solutions for cryogenic operation an idea for a gray-counter was developed. While the gray-counter could theoretically be used directly in the characterization design, it was omitted for a the safety and robustness of using the simple and well understood asynchronous counter. The structure itself remains of high potential however, especially for cryogenic applications due to unrivaled power efficiency. For this reason it was put on the chip next to the characterization structure to test it's potential. The proposed gray-counter is detailed in this section.

In digital chip design there are two counters that are used in almost every case a counter is needed. There is a synchronous counter, and an asynchronous counter. The former being used almost always. Both counters count in a binary way, where the synchronous counter changes all its outputs on the clock flank, hence the name synchronous. The asynchronous counter starts at the least significant bit (LSB) and trickles the counting through the more latches representing the more significant bits. Both have their advantages. The main advantage of the asynchronous counter is that it is a simpler and smaller structure. It scales better, and due to a smaller critical path, especially for counters with many bits. Therefore it can handle input signals with much higher frequency. The synchronous counter can guarantee that the output of the counter is stable, if there is no rising clock flank. This point is the key reason why synchronous counters are used so much more, because it can guarantee that the output is ready, while the asynchronous counter might still be busy propagating.

Gray counters are different to both asynchronous and synchronous counters in that they don't count in the binary order. Gray counters shuffle the order of the binary codes

around in such a way that only one bit changes per count. The potential upsides to this are very appalling. First of all, only one register changes value. This is quite a large upside from conventional binary counters, where the average bit-flips per clock tick are 2. For CMOS structures power is dissipated only when values are being changed, so there is a potential 50% power saving in this property. A second advantage of gray-counters is that the values can never be misinterpreted. As only one bit changes at a time, the maximum error is always at most one LSB. The problem of synchronous counters where different bits are changing at the same clock cycle is completely eliminated. A third advantage of gray-counters is that, if implemented correctly, there is a uniform effort to executing a count as only one register changes. So when an unsynchronized source is tracked, there is a uniform probability of sampling every number. This is not the case with conventional counters where counting from an even to an odd number only changes one bit, and is therefore less effort than all the even counts.

The opportunities of these property are hard to grasp because there aren't very good gray-counter solutions yet to fully utilize this property. This is because gray-counters have a very large disadvantage, that determining whether a bit should flip is based on the value of every single other bit in the system. This makes the scalability of gray-counter much worse when compared to conventional counters. This work proposes a solution that gets rid of this disadvantage by first designing an efficient 4-bit gray-counter, and then proposing a method to scale blocks of 4-bit gray-counters into larger gray counters., This is done while still maintaining the promising advantages of a gray-counter.

2.7.1 4-bit gray-counter

The building block is chosen to have a size of 4 because an elegant solution was found to implement a 4-bit gray-counter. The solution of the 4-bit gray-counter starts with a 3-bit gray-counter. There exists a solution where a 3-bit gray-counter can be made with 3 registers, an XNOR gate and two MUX's, using the state machine described in eq. 2.11, 2.12 and 2.13.

$$B_{k+1} = C_k \text{ XNOR } D_k \quad (2.11)$$

$$C_{k+1} = B_k ? C_k : \overline{D_k} \quad (2.12)$$

$$D_{k+1} = B_k ? C_k : D_k \quad (2.13)$$

If all these state transitions are executed at the same time, only one of the three will change at every count. Which means that the gray-counter properties are achieved. Not only that, but this configuration covers all possible combinations with three bits. Now this structure only ticks on the rising edge of the clock. There is also a falling edge

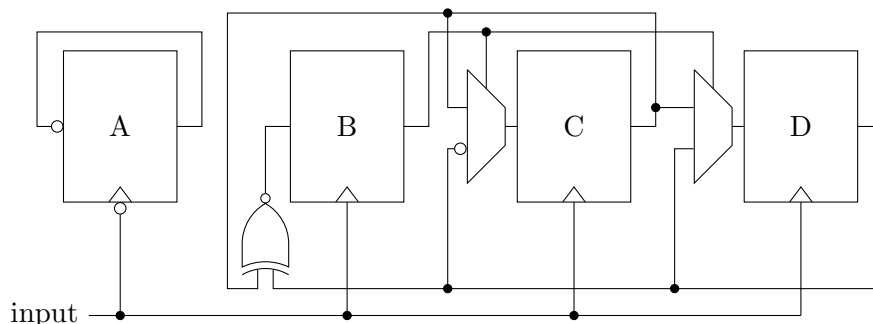


FIGURE 2.13: Schematic of a 4-bit gray counter

that can be used. There is an opportunity in using the falling edge, as all registers connected to the rising edge wont change during the falling edge and vice-versa. So we can add a clock divider on the falling edge, to transform the 3-bit gray-counter into a 4-bit gray-counter. A schematic overview of this counter is shown in figure 2.14. The sequence of states is shown in table 2.1.

binary	gray	binary	gray
0000	0000	1000	1100
0001	0001	1001	1101
0010	0011	1010	1111
0011	0010	1011	1110
0100	0110	1100	1010
0101	0111	1101	1011
0110	0101	1110	1001
0111	0100	1111	1000

TABLE 2.1: Sequence of states for the 4-bit gray-counter structure

If one is limited to using standard cells in the design, the resulting structure is effectively the most optimal 4-bit gray-counter structure that is theoretically possible. The critical path is the input of the B register, which is determined by a single register and a single XNOR.

When comparing this structure to a conventional counter, it is important to observe the difference in counting on the rising edge, or on both edges of the input signal. In applications where only the rising edge is of interest, and there is no convenient work around, we have to consider only 3 bits to compare. One could use a clock divider to get the desired effect, but the divider would use a relatively large amount of power. The amount of average bit-flips for conventional counters for a system with N bits can be

calculated with eq. 2.14.

$$\text{flip}_{av} = \frac{1}{2^N} \sum_{k=1}^N 2^k \quad (2.14)$$

$$= 1.75 \quad \text{for } N = 3 \quad (2.15)$$

$$= 1.875 \quad \text{for } N = 4 \quad (2.16)$$

$$= 2 \quad \text{for } N = \infty \quad (2.17)$$

Of course for the gray-counter by definition always flips 1 bit on average. So considering the 3 bit system one can potentially use 43% less power, for the 4 bit system with the falling edge bit 46% less power, and if a good scaling method was found, up to half the power could be saved. However, 3-4 bit counters aren't often used, so in order to harvest the potential of this gray-counter structure more work needs to be put into scalability.

2.7.2 Scalable asynchronous gray-counter

The idea of a conventional theoretical gray-counter is that every tick only one bit flips. The scalable problem of scalability that rises from this property makes the use of gray-counters questionable. The main point of the scalability method proposed here is to loosen this requirement to produce a counter that has all the nice properties of a gray-counter, without the scalability problems. The idea is to use the 4-bit gray-counters devised earlier as building blocks to make a larger gray-counter. When looking closely at table 2.1, and more specifically into the column of bit D, one can see that the output of D behaves like a signal that is divided 4 times with respect to the input signal. There is a second property of this output that is very important, which is that every time D changes, all the other bits in the system stay the same. So if we can manage to use the output of D as a clock signal to another 4-bit gray counter, besides the D serving as an input signal to the next block, all remaining bits behave as a 7-bit gray-counter.

This methodology shows how the 4-bit gray-counter can be used as a building block. One can substitute registers used in conventional counter with this 4-bit gray-counter structure to harvest the properties of a gray-counter. The simplest counter to do this with is an asynchronous counter. If we want to have a 16-bit asynchronous gray-counter we need a total of 5 4-bit gray-counter blocks, the first 4 blocks use D to drive the next block, and the last block uses all 4 bits as data bits. Now to compare this structure to a 16 asynchronous counter, first the average amount of bit-flips is calculated using eq. 2.18. Therefore this gray-counter uses 43% less bit-flips for the same range.

$$\text{flip}_{av} = 1 + 8^{-1} + 8^{-2} + 8^{-3} = 1.14 \quad (2.18)$$

Not all properties of gray-counters are there, because there is one problem introduced by borrowing the asynchronous counter structure, and that is that the counter has a propagation delay. The property where at every point in time there is at maximum one LSB of error can no longer be guaranteed unless the time between two edges at the input can be guaranteed to have at least a certain time difference between them. If this property is desired for the asynchronous counter a critical path of 16 registers needs to be considered. For the 16-bit asynchronous gray-counter this delay is 5 registers, 3 MUX's and one XOR gate. This is a lot faster than the asynchronous counter as the contribution to the critical path is an order of magnitude larger than the combinatory cells.

2.7.3 Scalable synchronous gray-counter

To address this problem the same idea can be used as with synchronous counters. Asynchronous counters are made synchronous by letting registers anticipate upcoming bit-flips at the input of it's less significant brothers. However, there is a problem with this setup. For a normal synchronous counter, every register is connected to the same clock, and the other bits only control the input of the registers. In the current setup, both the rising edge and the falling edge of the D flipflop is used, so more work needs to be done to create the desired behavior of a synchronous gray-counter.

It is important to realize that the D bit connecting the next block, doesn't hold any valuable information, as the value of D can be reconstructed by the next block. One can tell by looking at the value of the next block, whether the last edge was a rising or a falling edge, and from that one can backwards reason the value of D. We can therefore play around with how the D register is connected to make the system synchronous. The goal is to have every D register that is connected to a different 4-bit gray-counter block, be driven by the original input signal. A similar idea is applied in conventional synchronous counters, where a series of and gates is used to circumvent the problem of trickling information. The input of every D register, is the value it should have at the next clock cycle. By combining the input and the output of each D register one can predict a rising edge of this register. Now when one looks at the input of the D register, one can tamper with the control signal to create the desired effect. The resulting formula for a three stage system is shown in eq. 2.19

$$D_{1,k+1} = (B_{1,k} \mid (D_{0,k+1} \& \overline{D_{0,k}})) \ ? \ C_{1,k} : D_{1,k} \quad (2.19)$$

This method of scaling, which is borrowed from conventional synchronous counters, now creates a new scaling method where a synchronous counter can be constructed. If you look at the power consumption of the resulting system, the power analysis is even more favorable with respect to synchronous counters than asynchronous counters. That is because every block representing 3 bits worth of data, only receives the original signal on of the registers. The fastest changing signals dominate the power consumption, which in this case is the input signal, and with respect to the synchronous counter the input needs to be attached to three times less registers. All other calculations stay the same as for the asynchronous gray-counter. However, as with the asynchronous counter, another disadvantage rises due to the use of the synchronous counter structure. The critical path goes up. The critical path for a conventional 16-bit synchronous counter is 15 AND gates and one register. For the 16-bit synchronous gray-counter this would be 1 MUX, 3 AND gates, 1 OR gate and 1 register. One can see that the critical path problem still persists, but it scales a lot better than the synchronous counter because the signals go in blocks of three instead of one. The resulting synchronous gray-counter can therefore be superior to its conventional counterpart in performance critical applications.

2.7.4 Scalable uniform gray-counter

The solution provided in section 2.7.3 works as a scalable synchronous counter as the setup time doesn't scale with the number of bits. However, the solution provided is not uniform. When a perfect 50% duty cycle clock is provided to that gray-counter and one samples the output at random intervals, there won't be a uniform probability distribution for every number. If a closer look is given at how long it takes for a register to flip when the input is changed, for most bits in the system first the D register flips, which provides the required clock signal to then flip. Therefore the delay is more than 2 registers. However, for the first block, every bit is directly connected to the clock, so the first 3 bits don't have this problem. Therefore there will be a higher probability of observing that one of the first 3 bits changed last. Hence the problem of non-uniformity. A solution to this problem was found, where the input signal is first connected to a clock divider. The output of the clock divider is then used to drive the first 3 bits. A same structure for anticipating a rising edge can be used to make the whole system behave as a synchronous counter. The problem with this solution is that the power efficiency is lost, as the one added clock divider completely dominates the power consumption. However, there are applications where this uniformity is essential to the application, and in those cases, the performance of this uniform gray-counter is unrivaled. This field of applications is the place where gray-counters are already used, and the increase in efficiency of the proposed gray-counter structures is quite substantial.

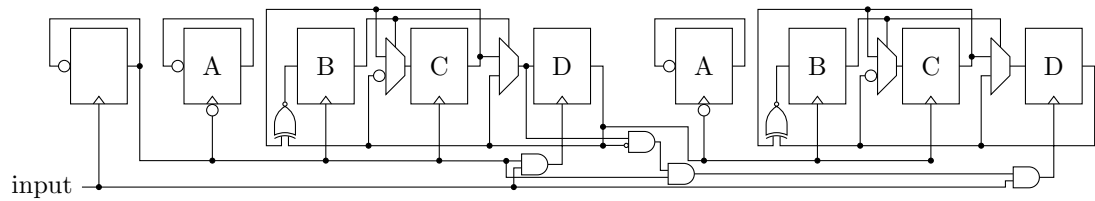


FIGURE 2.14: Schematic of the first part of a uniform synchronous gray counter

2.7.5 Hard coded decoder

One of the biggest disadvantages of the proposed gray-counter structures is that the numbers are not binary representations, so the codes need to be decoded before they can be processed further. There are applications where the decoding part isn't critical so it can be done with a large look-up table or post-processed. One can however hard code a structure that makes a binary representation of the data.

There are some interesting mathematical properties of the adopted gray-counter encoding scheme that can be taken advantage of. This was used to make the 4-bit gray-counter with such a little amount of gates. For the decoding part there is another property that can be used here. When examining table 2.1 one more time there is an interesting relationship between the gray encoding and binary encoding. First of all, the most significant bits are identical. Every time the most significant bit of the gray-counter changes, the most significant binary bit also changes. For the second row, one can see that the second binary MSB only changes when either the MSB or the second MSB of the gray-counter changes. The third binary bit changes when any of the first 3 MSBs of the gray-counter change, and the LSB of the binary counter changes whenever any bit in the gray-counter changes. This property is not a coincidence, as the gray-counter encoding was deliberately chosen to have this property. To decode the a gray-counter of N bits one needs $N - 1$ XOR gates. An example for such a decoder is shown in figure 2.15. The decoding structure uses trickle down to be area and power efficient. The assumption is that the decoder will not be a part of a critical part of the counter. If one directly connects the decoder to the gray-counter, one simply has an inefficient binary counter, without the nice properties of the gray-counter. One could for example use a shadow latch between the gray-counter and the decoder to use the gray-counter more effectively.

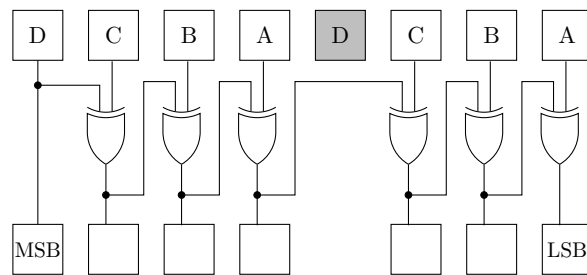


FIGURE 2.15: Decoding for 7-bit gray counter made of two 4-bit blocks

Chapter 3

Implementation and ASIC design

This chapter describes all the steps that were taken to make the ASIC out of the designs produced in Chapter 2. The steps include the test benches, Verilog programming and simulating, design synthesis, and layout. Not all of the designs produced in Chapter 2 made it on the chip, so only the implementations that were fabricated are discussed here.

3.1 Verilog, Genus and Modelsim

When starting the project, at first the design was entirely performed in Cadence Virtuoso, an analog design environment. Later in the project it was decided that a digital design flow is a superior solution for these type of designs. The entire work was ported and implemented in Verilog. Verilog was chosen over VHDL, because the native language for the Cadence tool-chain is Verilog. Having only one language to work in simplified the work process. Using behavioral Verilog, the designs were first verified in Modelsim, a Hardware Description Language simulation Environment. The behavioral simulation allows a verification of the implemented functionality. The Verilog code can be debugged relatively easily in this step.

The second step is to invoke Genus. Genus is a synthesis solution developed by Cadence. Its most important feature is that it is extremely quick, minimizing the iteration time. Genus provides a structural netlist file in Verilog, containing cells from the standard library. It also provides a Synopsis Design Constraint file (.SDC), that specifies the timing constraints that have to be met during the layout phase. Genus also provides a Standard Delay Format file (.SDF), that includes the timing information for all the standard cells in the netlist. Note that the information in the SDF file is sourced from the

library. They are exactly the propagation delays this work attempts to characterize. A room temperature library was used, so uncertainties remain for performance at cryogenic temperatures. Through tickle scripts (.tcl) one can automatize the steps performed in Genus completely. This includes initializing the program, reading in the technology library, reading in the behavioral Verilog files, defining the timing constraints, performing the design synthesis, and exporting the relevant output files.

The third step is to take the structural Verilog files, and the SDF files, and importing them back into Modelsim. When the standard cell library and SDF files are added to Modelsim, it can simulate the structure with the characterized propagation delays of the standard cells. The simulation performed is not very precise, as it simply adds the delay directly to a transition. This is a limitation to Modelsim, but the trade-off is that the simulation is very fast, and allows for long simulations and large structures. Especially for designs that are meant to operate on large time intervals, this is required to verify certain functionality. A convenient bonus to returning to Modelsim for verification is that in most cases the same benchmark used for behavioral verification can be used again for structural verification, speeding up the design flow.

An optional fourth step is to directly import the structural Verilog into Virtuoso. Virtuoso can simulate the netlist with a much higher accuracy than Modelsim can, where it uses a SPICE model with varying voltage levels instead of a timing model with ones and zeros. However, this simulation method is very slow on larger designs.

3.2 DUT selection and test benches

The entire TSMC 40 nm standard cell library contains approximately a thousand cells. A conservative estimate gives the cells an average of three inputs. This means that at least 3000 configurations would need to be characterized to cover the full library. This is not a feasible number in the context. It is important to realize why this library is so large in the first place. In principle every functional design can be made with a select few basic cells. The way a standard library works, is that every cell is implemented as a full layout with the technology. This gives room for optimization by optimizing more complex functions into extra cells. A complex logic gate can be implemented with a single pull up and pull down network, increasing the efficiency substantially. An example is an XOR gate. It can be built from 4 NAND gates, using a total of 16 transistors, but a custom solution exists where one pull up and one pull down network are used with only 6 transistors.

cell	count	cell	count
CKND2	36474	MUX2	1581
ND2	20648	NR4	1009
XNR2	11814	HA1	990
IND2	7644	DFCN	974
ND4	6264	NR3	599
DFCNQ	5905	OR3	459
SDFCNQ	3730	IND3	389
IND4	3089	OR2	314
ND3	2873	NR2	268
FA1	1620	INV	245

TABLE 3.1: Subset of standard cells and their occurrence in the ITC-99 test bench set, Bold cells indicate sequential cells

When selecting a subset of cells to characterize there are a few important things to consider. First of all, the characterized subset needs to be sufficient for the design synthesis. Secondly the characterizer scales with the amount of inputs, so cells with smaller amounts of inputs should be preferred. A third consideration is the drive strength of the cells. A higher drive strength equals a larger amount of power dissipation. As this library is characterized for power constraint application a smaller drive strength should be preferred. The decision was made to use the lowest available drive strength for most of the cells, while using a range of drive strengths for inverters and buffers. This way a fan-out inverter or buffer can be used after a minimum drive strength cell if required. The fourth thing to consider is which cells are most commonly used. Combining the most commonly used cells with a minimal amount of additions to form a complete set results in the most efficient subset to characterize.

To find the most commonly used standard cells, a set of test benches was used. For this work the ITC99 test bench set was selected. This test bench set is an open-source set of test benches available in different hardware programming languages including VHDL and Verilog. Added to the test benches was the implementation of the characterization design itself, as all the components used in the characterization structure should be characterized. In order to determine which standard cells are most commonly used, the test benches were synthesized using the existing TSMC 40 nm library through the program Genus from Cadence. The process of selecting the best subset was performed through elimination. Through several iterations, the least used cells were deleted with priority for cells with more inputs. The synthesis was repeated to see the effects of removing those cells, and the process of elimination was repeated. The resulting subset of cells is shown in table 3.1. Note that this set doesn't consider driving strength. That is because Genus doesn't decide what the optimal drive strength should be for each component. This is done by Innovus in a later stage, described in section 3.5. Instead of adding components with higher drive strengths, inverters with high drive strengths were added to

cell	amount per set	cell	amount per set
INVD0	4	ND2D1	4
INVD1	4	AN2D0	8
INVD2	4	CKND2D0	4
INVD4	4	XOR2D0	8
INVD8	4	XNR2D0	8
INVD16	4	NR3D0	12
BUFFD0	4	ND3D0	12
NR2D0	8	MUX2D0	8
NR2D1	4	IND2D0	4
OR2D0	8	HA1D0	8
ND2D0	8	FA1D0	16

TABLE 3.2: Subset of standard cells per set on the chip. Every set is 128 cells in size.

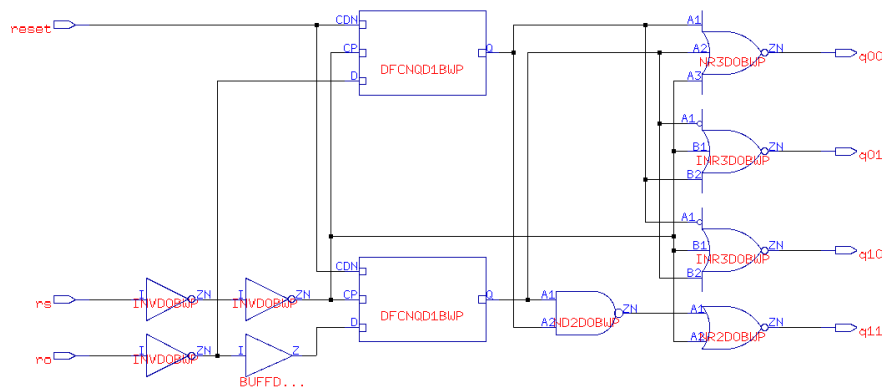


FIGURE 3.1: Genus schematic of a building block with as DUT a buffer

the subset to allow for fan-out structures. One of the advantages of characterizing inverters is that they only have one input, so it is very cheap to characterize inverters. The inverter drive strengths used are 0, 1, 2, 4, 8 and 16. Due to difficulties during the layout phase, not all the cells planned for characterization made it onto the final design of the chip. The subset that was managed to be put on the chip is shown in table 3.2. Note that the amount of cells is managed in sets. In the design there is one set that is instantiated multiple times to decrease the variance of the measurement.

3.3 Implementation and simulation of designs blocks

The building block described in Section 2.5.1 was implemented in Genus resulting in blocks as shown in figure 3.1. Note that both the `ro` and `rs` inputs are buffered, and that genus found a more efficient way of getting `q00-q11` with respect to the design shown in figure 2.5. The behavior ends up the same because as soon as the values are sampled in the register, the signals are no longer sensitive to delay. A Virtuoso simulation is shown in figure 3.2. In this simulation the signals `a0` and `a1` represent

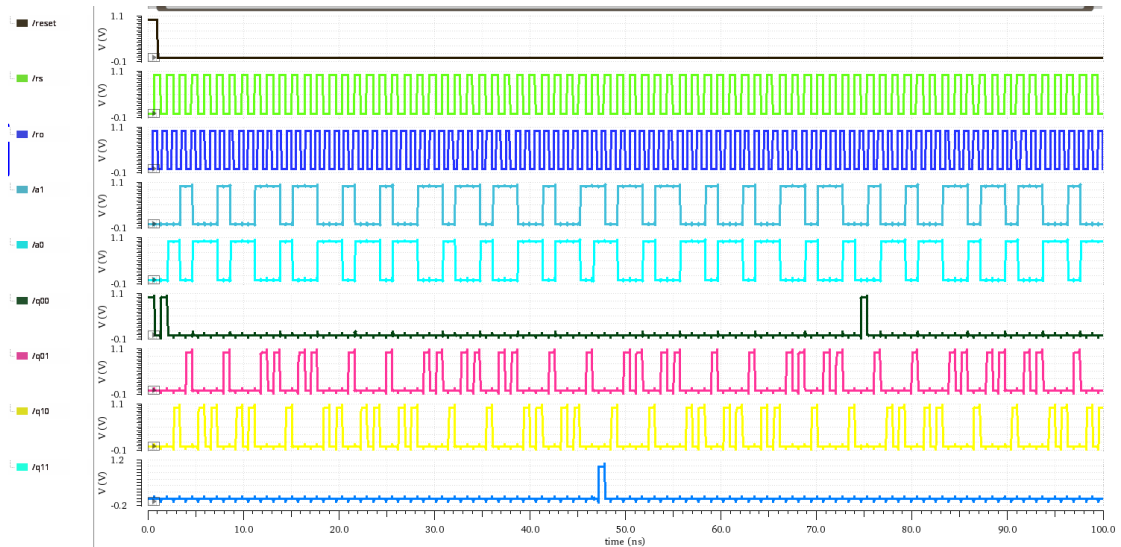


FIGURE 3.2: Virtuoso simulation of building block structure

the output of the two registers, and q00-q11 the output after it is chopped up by the random sampler signal. One can see that in situations where the same value in the registers is sampled twice, there are multiple consecutive pulses that can be counted. These signals can directly be fed into counters, and if enough samples are extracted, the propagation delay can be extracted. The ring oscillator and random sampler are also implemented using Verilog and the same design loop with Genus. Normally one wants to design ring oscillators by hand if specific frequencies are required. However in this case the requirements for the frequency are not specific, as long as the frequency is known. So in order to achieve this an additional counter is added to the system, that directly counts the rising edges of the ring oscillator. Based on how long the ring oscillator was enabled by the FPGA, the frequency can be extracted. This is a more robust approach than trying to design for a specific frequency, not only because the implementation is much simpler and less error prone, but also because the system is designed for different operating voltages and extreme temperatures. A calibration during the characterization is a very sensible approach in the context. The ring oscillator was designed with 45 inverting elements for an output frequency of approximately 2.5 GHz. There are 7 clock dividers and 3 control signals to select between 8 different frequencies.

The random sampler is implemented in a similar way to the ring oscillator, except that it has the LFSR and variable delay elements. The random sampler has a ring with 75 inverting elements. The LFSR has 22 bits and implementing the function $x_0(k+1) = x_{21}(k) + x_{20}(k) + 1$. The size of this LFSR was chosen so that it would take comfortably long enough for the LFSR output to repeat. The function chosen is one of multiple options that represents the least amount of standard cells required to implement the LFSR. The difference between different functions is the order of the sequence, but

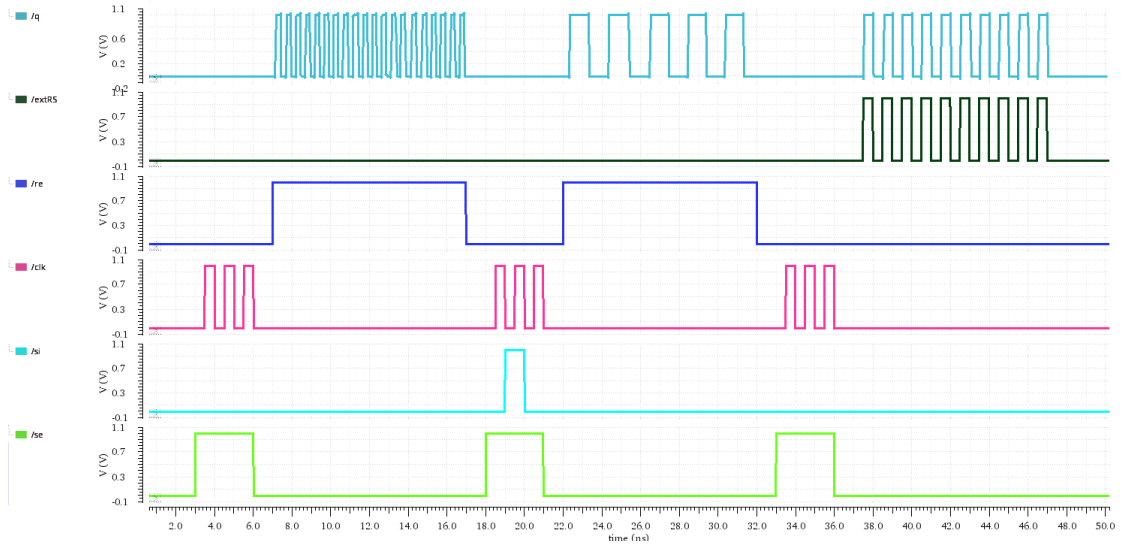


FIGURE 3.3: Virtuoso simulation of the random sampler

as every binary number is covered, this doesn't affect the overall performance for this use case. The two least significant bits of the LFSR are used to control two separate variable delay elements that can produce a variance of 2 or 4 inverters respectively, so the period of random samplers varies with an extra delay of 0, 2, 4 or 6 inverter delays. In the same way as the implementation of the ring oscillator there are 7 clock dividers and 3 control signals to choose between 8 base frequencies. A virtuoso simulation of the random sampler is shown in figure 3.3. The testbench tests three cases, two where the shift register is controlled to select how many times the clock is divided, and one where the random sampler is turned off, and an external signal is supplied instead.

The gray-counter is also implemented with Verilog and Genus. An asynchronous gray-counter was implemented with 7 blocks, resulting in a 22 bit counter. A Virtuoso simulation of a gray-counter with 3 blocks is shown in figure 3.4. It is interesting to observe that every data signal is in itself a periodic signal, that all data signals have a different frequency, except for the $D<i>$ signals, and that the frequencies differ by exact multiples of two. However, the base frequency of the least significant bit is twice as high for normal counters, which is the reason for the reduction in power consumption. Secondly all the data signals are phase shifted with respect to the behavior of classical counters. The data signals are all phase shifted in such a way, that no signals have an edge at the same time, which is the fundamental gray-counter property.

3.4 Complete Schematic Design

This section explains the final schematic design that was put on the chip. In various cases, concessions were made on the final design due to difficulties in producing a

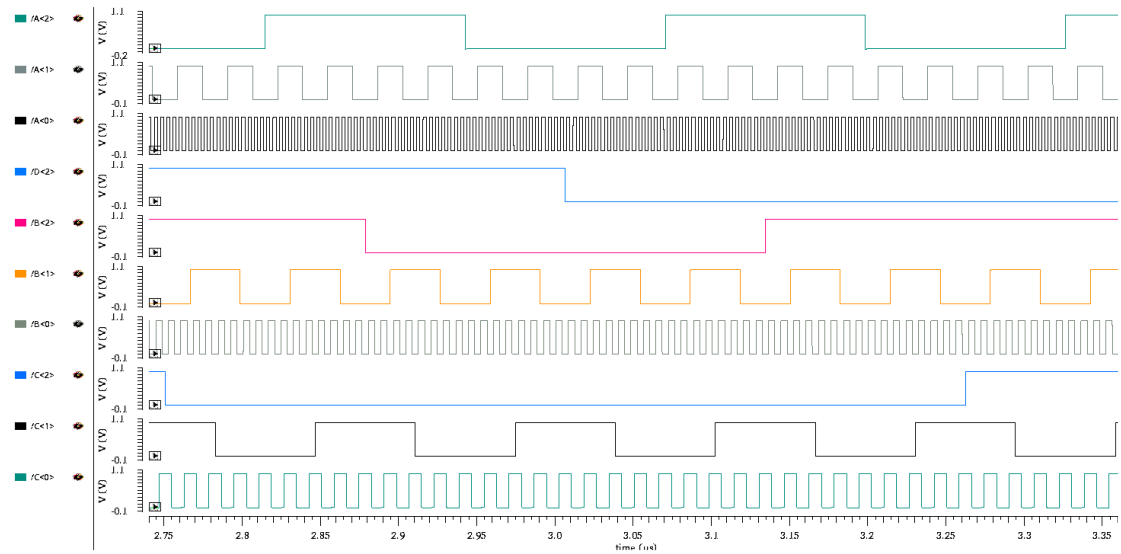


FIGURE 3.4: Virtuoso simulation of an asynchronous gray counter with 3 gray-4 blocks and 10 bit range

properly functioning layout for the chip. Because of this, both the level shifters and the entire sequential characterization structures didn't end up on the fabricated chip. An overview of what did end up on the chip is shown in figure 3.5. Operating the chip is done with a large shift register, that connects all the parts of the chip together. By using a single large shift register, the amount of required I/O is minimized. The shift register, and how it is connected to the rest of the chip is illustrated in figure 3.5. The first 3 bits control the MUX in the ring oscillator block that determines which clock divider is connected to the output. A same situation is found for the random sampler block with the next 3 bits. Bits 6 and 7 control two MUX's to send the RO and RS signals to one of four different structures. The next 7 bits control a larger MUX that is connected to an array of building blocks as described in section 2.5.1. The reason why the MUX's are implemented specifically in a separate way is because originally the first MUX was meant to be much larger. The set contains all the cells that require testing, and the first MUX can then be used to duplicate these structures to fill up the chip. Unfortunately the scaling caused problems in the layout phase that couldn't be fixed before the tape-out deadline. The resulting design features 512 separately addressable building blocks with different DUT configurations. The selection procedure of which cells are chosen is explained in section 3.2. Every building block has 4 outputs. In bundles of 512 these outputs are merged in OR tree structures. The output of which is then fed into 4 asynchronous counters. The asynchronous counters have 40 bits each. This number is so large so that a long duration of testing can be performed without an overflow. Because the counter is asynchronous, the cost of making these counters larger is very small.

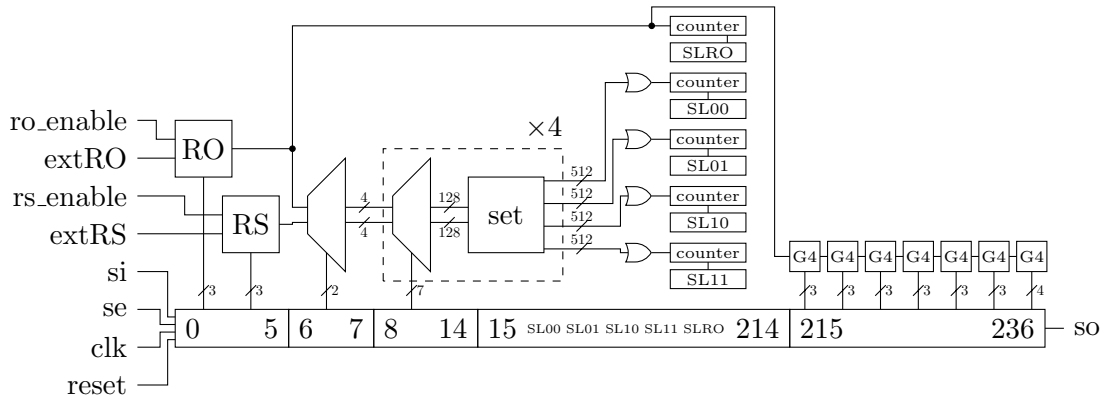


FIGURE 3.5: Schematic of the entire chip

Every counter has a shadow latch with an equal 40 bits. These shadow latches are part of the main shift register and make up the largest chunk of it. Besides the four counters with shadow latches that are used for characterization, a fifth counter and shadow latch are added. This counter has the ring oscillator as input. The value in the counter and the duration of the `ro_enable` signal provided by the FPGA can be used to calculate the frequency of the ring oscillator. This value needs to be extracted to calculate the propagation delay of the DUT's. It is important to be able to track this on-chip during the test, as the varying operating conditions will not only affect the DUT's but also the ring oscillator.

After the combinatory characterizing structure the gray-counter test structure is added. The asynchronous gray-counter design was chosen, with 7 gray-4 blocks, resulting in a 22 bit asynchronous gray counter. This structure doesn't include the hard coded decoder, so the direct values of the counter can be processed off-chip. The gray counter shares the ring oscillator as an input. This is convenient because with a combination of `extRO` and `ro_enable`, the FPGA has full control over this signal. This way the gray-counter can be tested without adding additional I/O to the chip. Finally the output of the shift register is connected to the SO I/O output, which is the only output on the chip.

The sequence for operating the chip is as follows. First the chip is reset and then a bit sequence is shifted into the shift register that specifies the configuration. More specifically it selects the active DUT, and the frequency of the ring oscillator and the random sampler. Then the `ro_enable` is put to high, and afterwards the `rs_enable` is put to high. Then for a duration of choosing, the accumulator collects samples for every rising edge of the random sampler, until the `rs_enable` and `ro_enable` are put back to zero. Finally the accumulated counts are stored in the shift register and shifted out to be further processed externally.

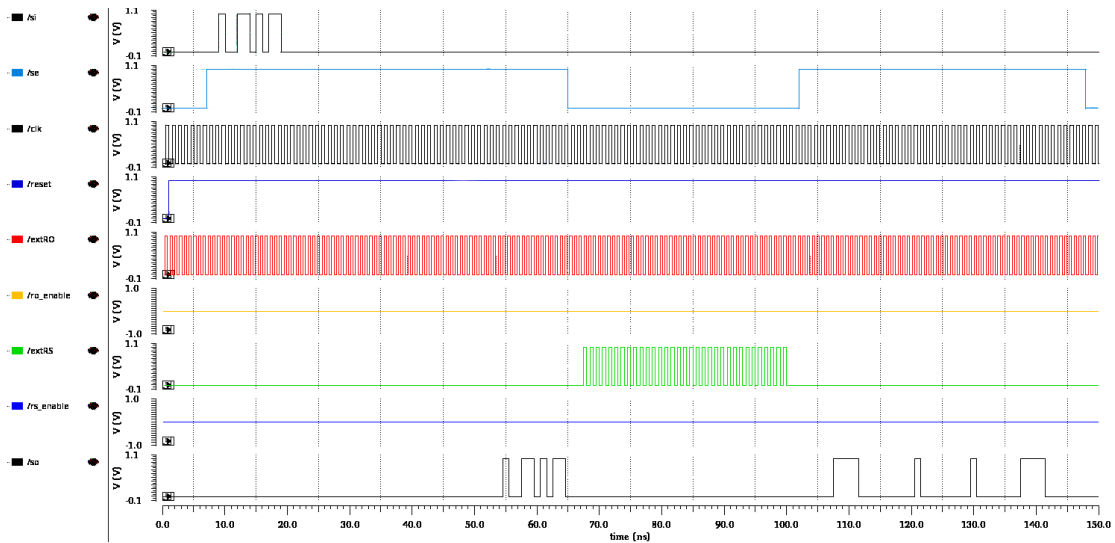


FIGURE 3.6: Virtuoso simulation of building block structure

A Virtuoso simulation of a sequence to characterize a single DUT is shown in 3.6. Note that this simulation was done with counters with 10 elements instead of 40, and the gray-counter is not attached, to speed up the virtuoso simulation. One can see that first a sequence is shift into the device to test the shift register. The setup is then left at all zeros. An external signal is used for both the Ring oscillator and the random sampler. After a period of sampling the accumulated samples are stored in the shift register and shifted out.

3.5 Layout with Innovus

When the netlist is completely verified the next step is the layout phase in Innovus. Innovus is a tool made by Cadence and is the successor to the more commonly known SOC Encounter. Innovus is an extremely powerful tool for efficient layout with a large variety of options available to an experienced user. A lot of the steps taken in Innovus are identical or very similar when designing different chips. The same way as in Genus, Innovus can be automated using tickle (.tcl) scripts. Once the scripts are set-up properly, the combination of Innovus and Genus allows for a very fast iteration time.

There are several steps that need to be performed in Innovus that are specific to layout design. These include but are not limited to floor-planning, I/O design, Power planning, well tap placement, cell placement, clock tree synthesis, routing and verification.

The size of the chip is limited by either the contents of the chip, or the amount of space required by the pad-ring. In the case of this work however, due to the logistics of how multiple chips of the group are put on the same wafer, the available space is exactly

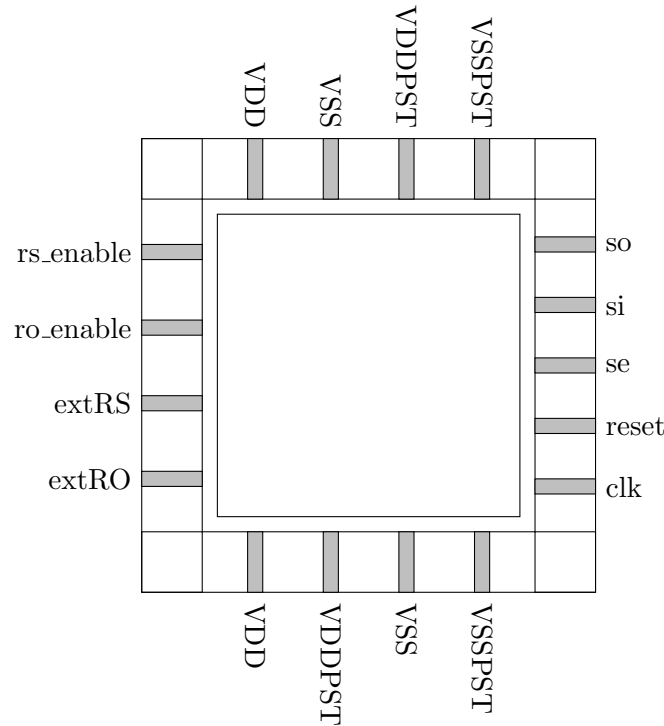


FIGURE 3.7: Design of the padding

$1 \times 1 \text{ mm}^2$. As the design of the chip is quite scalable, the original goal was to use that provided space, and put as many characterization structures on there as possible. Unfortunately there were problems with the scaling that couldn't be solved before the tape-out deadline. Because of that, only a small part of the chip is actually utilized.

For the pad-ring it was decided to duplicate all the power and ground ports, conform the guidance of TSMC. These I/O Cells are placed on opposite sides of the chip to minimize the distance between the power I/O Cells and every part of the chip. The other I/O cells were grouped on functionality and divided equally over the two remaining sides. The resulting pad design is shown in figure 3.7.

For the procedure of adding the power lines, the goal of the design was to use power domains and level shifters to allow the DUT's to run at different operating voltages than the characterizing structures. Unfortunately the implementation in Innovus didn't work before the tape-out deadline, so the entire feature was omitted. The resulting power design has one power domain for the entire core. Because at Cryogenic temperatures the substrate has more resistance, the distance between the substrate and the nearest well tap should be a lot smaller than for room temperature. In order to be area efficient, it is desirable to use the least amount of well taps, but area is not an important constraint for this chip design. Because of that a well-tap was placed every $8 \mu\text{m}$, reducing the maximum distance to a well tap to be $4 \mu\text{m}$.

The flow of place-and-routing the netlist is very similar for most chips. There are several steps taken to efficiently reach a well optimized layout. The first step is to place every cell roughly in the core to minimize the distance between all of the connections. As a second step the tie-high and tie-low cells are added to hard code logic ones and logic zeros. The third step is place the cells on specific spots on the chip, where they fit inside the power and ground rails correctly and no cells overlap, including the previously placed well taps. In this stage very simple routing is applied. The fourth step is to add the clock tree. The clock tree is optimized to minimize the difference in delay for all parts of the chip. Innovus has functionality to figure out how the clock tree should be constructed, and which buffer cells should be used to fit the requirements most efficiently. The clock tree synthesis step includes repositioning all the cells to their final locations and rerunning a cheap version of the routing. For the fifth step the routing is optimized, so that the lengths are minimized, and the use of higher metal layers is minimized. This is the first stage where the design could be LVS and DRC clean, as the routing used before this optimization stage doesn't meet the design rules. As a sixth step the design is checked for setup and hold violations. Alterations to the design are made where required. For the seventh and final step the entire design is checked for design rule violations, and if the design passes it is exported with the GDSII format. This format is a binary file format containing the planer geometric shapes, labels and other details required by the manufacturer. Besides this file a new netlist file is exported that contains all the cells and alterations that were added to the original netlist provided by Genus.

The final step before submitting the design for fabrication is to check it in Cadence Virtuoso. The GDS file and Verilog netlist that are exported by Innovus can be read into Virtuoso directly. This allows Virtuoso to perform checks of DRC and LVS conform the rules provided by the foundry. While DRC and LVS checks can be performed by Innovus, it does not catch all errors. Passing DRC and LVS in Virtuoso however, is the best conformation one can get that the layout works. For the case of this project there were quite a few errors that popped up during verification in Virtuoso that needed attention.

The layout of the final chip design is shown in figure 3.8. Note that the chip is fabricated together with other chips from the group. The structure produced by this work is marked by a red square.

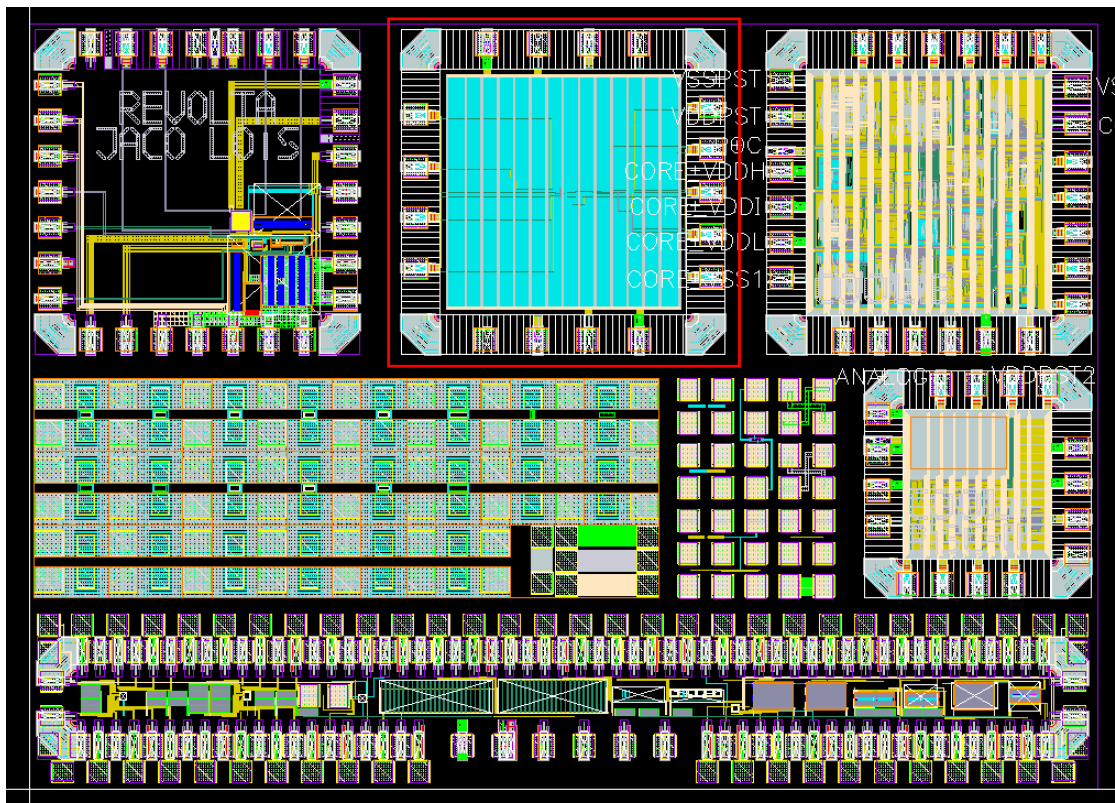


FIGURE 3.8: Final chip layout, the structure within the red square is the contribution of this work

Chapter 4

Measurements and results

To test at Cryogenic temperatures, there is a dipstick setups that uses liquid helium to cool to 4 Kelvin. In order to automate testing chips at cryogenic temperatures, a board was developed by Harald Homulle. The board features an ARTIX-7 FPGA, two connectors for a daughter board, and a connector as output, that can be connected to the dip stick. In order to test the chip using this FPGA board, a daughter board was designed to be mounted on the motherboard. The schematic and layout are shown in Appendix 6. The design features 4 sets of two decoupling capacitors with 100 nF and $100\text{ }\mu\text{F}$ connected to all of the power pads on the chip. The chip socket is a Leaded Chip Carrier (LCC) socket as this is a proven technology for cryogenic temperatures. A picture of both the designed PCB and the motherboard with the ARTIX-7 FPGA can be found in Appendix 6. The chip was wirebonded according to the schematic shown in Appendix 6. A picture of the wirebonded chip is shown in figure 4.1. A picture of the designed PCB with the wirebonded chip in the socket is shown in figure 4.2

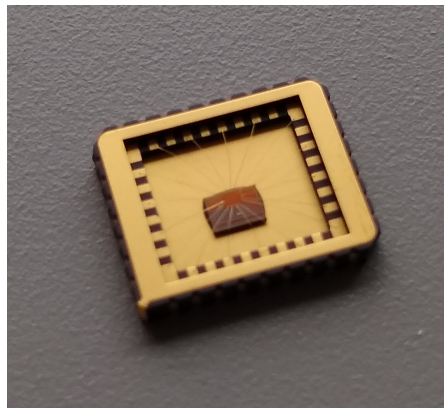


FIGURE 4.1: Photo of the wirebonded chip

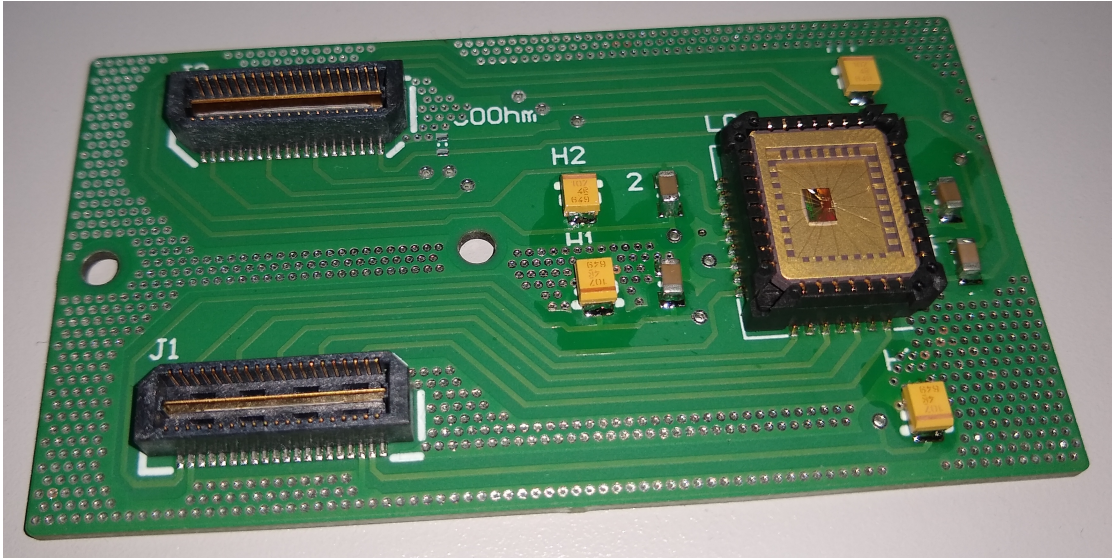


FIGURE 4.2: Photo of the soldered PCB with the wirebonded chip

The FPGA features a UART interface that connects to a UART to USB device that hooks into a computer. The UART connection is used for the computer to communicate to the FPGA what settings should be used for the next measurement, and if the FPGA is ready to perform the measurement. The FPGA receives the info, handles the measurement, and sends back the result to the computer. This way the entire characterization sequence can be fully automated using a computer running Matlab.

In order to operate the chip, the FPGA shifts a bit-string into the shift register on the chip, that determines all the settings. The FPGA then enables the ring oscillator by sending an active high to `ro_enable`, or supplies an external oscillating signal through `extRO`. Afterwards the random sampler is enabled by sending an active high to `rs_enable` or by supplying an external sampling signal to `extRS`. After a duration chosen by the FPGA, first the sampling signal and then the oscillator are turned off. After a brief settling period for the asynchronous counters, the results are shifted out of the chip into the FPGA. The FPGA then sends the entire bit-string from the chip to the computer for further processing. An abstract illustration of the test-setup is shown in figure 4.3. A picture of the setup for room temperature is shown in figure 4.4. On this picture the motherboard is connected through a custom PCB designed by Edwin Shriek to both a UART to USB interface and a JTAG to USB programming cable to program the ARTIX-7.

Unfortunately by the time of finishing the thesis, the test setup never worked properly. The design of the FPGA was properly implemented on a NEXYS 4 DDR development board and laptop. But porting the implementation to the development board caused

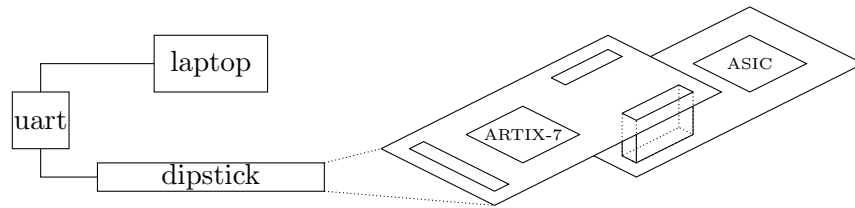


FIGURE 4.3: Overview of test setup

problems that couldn't be solved within the available time frame. Most of the problems were caused due to initial misunderstandings of how the designed motherboard operates, and too little time to fix them when the errors were discovered. This is very unfortunate because there is a large difference between barely being able to measure the chip, and actually testing the chip. The author estimates that completing the measurements would take anywhere between 2 and 5 weeks of extra work. The main problems are found in correctly configuring the resources on the FPGA, the powering of both the chip and the FPGA, and the problems that would turn up past that stage.

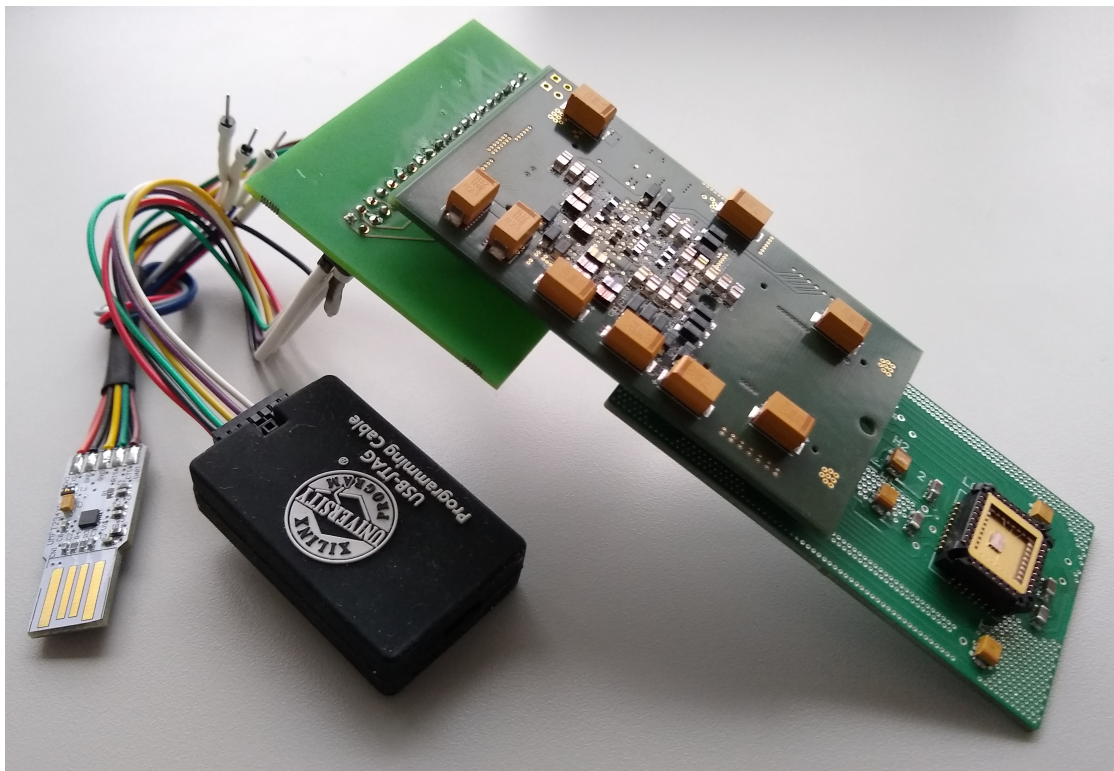


FIGURE 4.4: Photo of the room temperature test setup. Attached to the ARTIX-7 board are a UART to USB device and a JTAG programming cable

Chapter 5

Discussion and Conclusion

This chapter reflects back on the work that was performed and the lessons that were learned. A suggestion is made to how to build on and improve the project. Finally a conclusion to this Thesis is given.

5.1 Discussion

Designing a chip for a Master Thesis is one of the more compelling options out there. There are so many facets to designing a properly functioning chip, especially if that includes operating it and measuring its performance. I had never designed a chip layout, both analog nor digital, nor experienced a tape-out, never designed for environments with unknown factors like cryogenic temperatures, programmed an FPGA, designed a PCB, the list goes on. This concept of gaining experience was the core reason for choosing a thesis topic like this one. I wanted to experience these things, and learn how they worked. To some extent this idea worked out as intended. I got the opportunity to witness all of the things mentioned above, and learned lessons in engineering that aren't easy to learn if not experienced first hand. It was however also one of the main obstacles for my thesis. Due to the lack of my own experience in all of these parts, I had large struggles to figure out the basics of the facets I was working on. In particular, I had issues with the digital IC design flow. As both me and the vast majority of the group had little experience with the digital design tools I needed to use, it posed a major obstacle for me to overcome. Unfortunately I have to conclude that this obstacle wasn't fully overcome. There are many things I would have liked to be included on the chip. These include more of the structures that I designed, different power domains, level shifters, isolated oscillator sources, custom analog designs for the characterization

building blocks, and more. Also unfortunate is the malfunctioning of the test setup, so that the potential of the current chip cannot be extracted properly.

During my thesis I was well aware of my lack of experience and knowledge in the topic, especially with respect to designing, which I am much better at. In order to compensate for this I tried to design elegant structures that were not too complicated to put together. I am very happy with how this turned out and some of the presented designs, especially the gray-counter design, belong to some of the most elegant structures I ever came up with. The simple designs didn't help me as much as I hoped in the design flow however. There is a very large checklist of details to go through before any chip works, regardless of functionality or design, and this took up the bulk of my time. In retrospect it would have been better to start the Thesis by going through the entire design process, from design to chip layout, with a very simple structure, like a shift register or a counter, before doing any design specific work. The knowledge I gained from going through this process once could have helped me refine my designs much faster and focus on the most important points earlier. It would also help a lot in estimating the workload of a task, reducing preventadline crunches. This would also be my recommendation for all future master students that want to design a chip for their master thesis.

5.2 Conclusion and Future work

The objective of this work was to characterize the timing performance of cells in the TSMC 40 *nm* standard cell library. In this work a solution was developed to characterize the timing information of standard cells in cryogenic temperatures. Besides that a design for a novel gray-counter was proposed. With the exception of the structures for sequential cells the designs were implemented in the TSMC 40 *nm* technology, resulting in an ASIC that was fabricated. A test setup was designed to measure the chip.

This work has a lot of good ideas, that couldn't be implemented yet. Those ideas and designs would be desirable to pursue in the future. However the most important future work would be to fix the problems with the test setup and perform the characterizations on the chip. By comparing the measurements with the supplied timing measurements of TSMC, the quality of the chip design can be assessed, and where required, improvements can be made. There are many optimizations that can be performed in the design flow without changing the designs. Eventually the goal is to aid in the development of models that will allow SPICE simulations to directly calculate the timing information without the need to perform hardware measurements at cryogenic temperatures. At this point, there is less need to characterize the components through a structure like the one designed in this work. The proposed gray-counter has the potential to enable

a large range of applications that weren't effective previously due to them missing this solution. For example if a uniform output is required of the counter. As gray-counters can be read out asynchronously without glitchess, it allows one system to communicate values to the other without the need for synchronization. Otherwise further research into harvesting the power savings can be interesting for cryogenic ASICs in particular.

Bibliography

- [1] Austin G Fowler, Matteo Mariantoni, John M Martinis, and Andrew N Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.
- [2] Rami Barends, Julian Kelly, Anthony Megrant, Andrzej Veitia, Daniel Sank, Evan Jeffrey, Ted C White, Josh Mutus, Austin G Fowler, Brooks Campbell, et al. Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature*, 508(7497):500, 2014.
- [3] E Charbon, F Sebastiano, A Vladimirescu, H Homulle, S Visser, L Song, and R M Incandela. Cryo-cmos for quantum computing. In *Electron Devices Meeting (IEDM), 2016 IEEE International*, pages 13–5. IEEE, 2016.
- [4] Jos B Sulistyo and Dong S Ha. A new characterization method for delay and power dissipation of standard library cells. *VLSI design*, 15(3):667–678, 2002.
- [5] Rosario M Incandela, L Song, Harald AR Homulle, Fabio Sebastiano, Edoardo Charbon, and Andrei Vladimirescu. Nanometer cmos characterization and compact modeling at deep-cryogenic temperatures. In *2017 47th European Solid-State Device Research Conference (ESSDERC)*, pages 58–61. IEEE, 2017.
- [6] Lin Song, Harald Homulle, Edoardo Charbon, and Fabio Sebastiano. Characterization of bipolar transistors for cryogenic temperature sensors in standard cmos. In *SENSORS, 2016 IEEE*, pages 1–3. IEEE, 2016.
- [7] Arnout Beckers, Farzan Jazaeri, Heorhii Bohuslavskyi, Louis Hutin, Silvano De Franceschi, and Christian Enz. Characterization and modeling of 28-nm fdsoi cmos technology down to cryogenic temperatures. *arXiv preprint arXiv:1809.09013*, 2018.
- [8] Fabio Sebastiano, Harald AR Homulle, Jeroen PG van Dijk, Rosario M Incandela, Bishnu Patra, Mohammadreza Mehrpoo, Masoud Babaie, Andrei Vladimirescu, and Edoardo Charbon. Cryogenic cmos interfaces for quantum devices. In *Advances in*

- Sensors and Interfaces (IWASI), 2017 7th IEEE International Workshop on*, pages 59–62. IEEE, 2017.
- [9] Arnout Beckers, Farzan Jazaeri, and Christian Enz. Cryogenic mos transistor model. *arXiv preprint arXiv:1806.02142*, 2018.
- [10] Mehmet Batuhan Dayanik and Michael P Flynn. Digital fractional-n plls based on a continuous-time third-order noise-shaping time-to-digital converter for a 240-ghz fmcw radar system. *IEEE Journal of Solid-State Circuits*, 2018.
- [11] Amr Elshazly, Sachin Rao, Brian Young, and Pavan Kumar Hanumolu. A noise-shaping time-to-digital converter using switched-ring oscillators-analysis, design, and measurement techniques. *J. Solid-State Circuits*, 49(5):1184–1197, 2014.
- [12] S Maggioni, A Veggetti, Alessandro Bogliolo, and L Croce. Random sampling for on-chip characterization of standard-cell propagation delay. In *Quality Electronic Design, 2003. Proceedings. Fourth International Symposium on*, pages 41–45. IEEE, 2003.
- [13] Mohammad Maymandi-Nejad and Manoj Sachdev. A digitally programmable delay element: design and analysis. *IEEE transactions on very large scale integration (VLSI) systems*, 11(5):871–878, 2003.

Schematic and layout of PCB

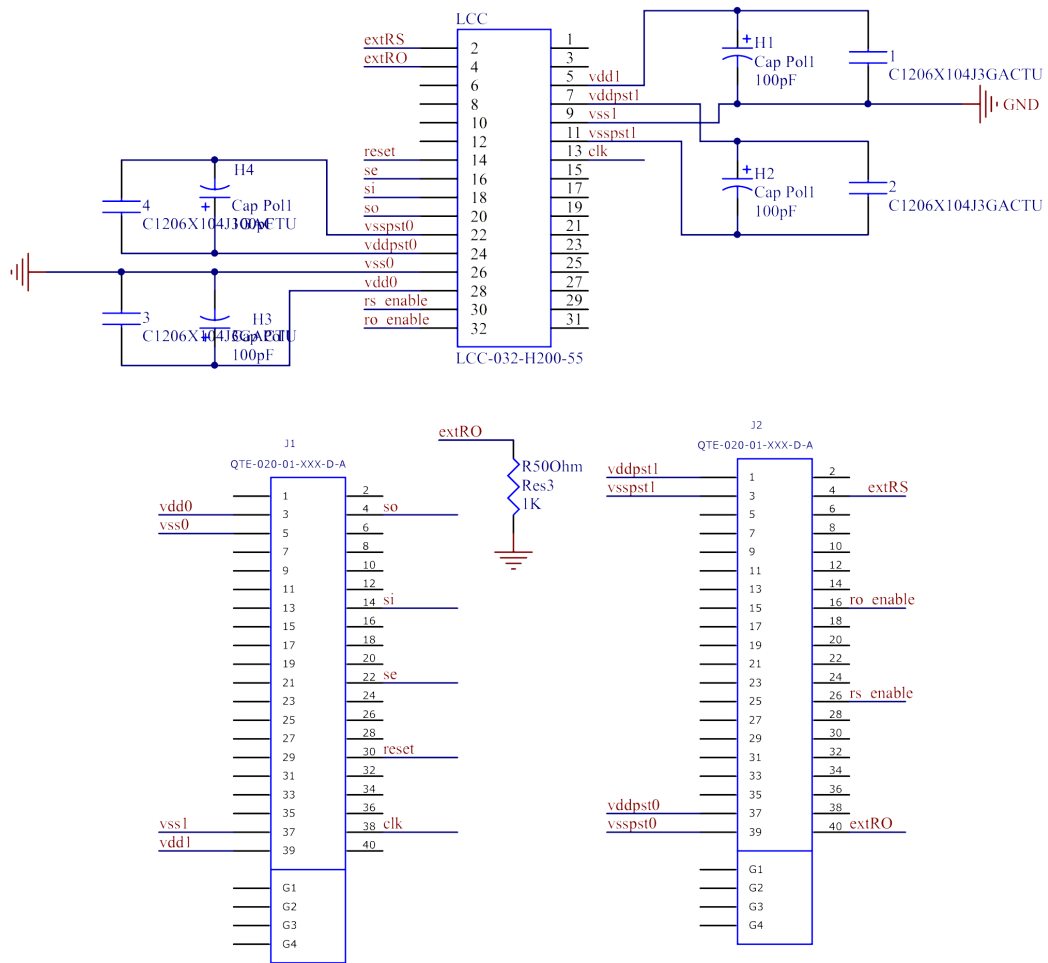


FIGURE 1: schematic of PCB design

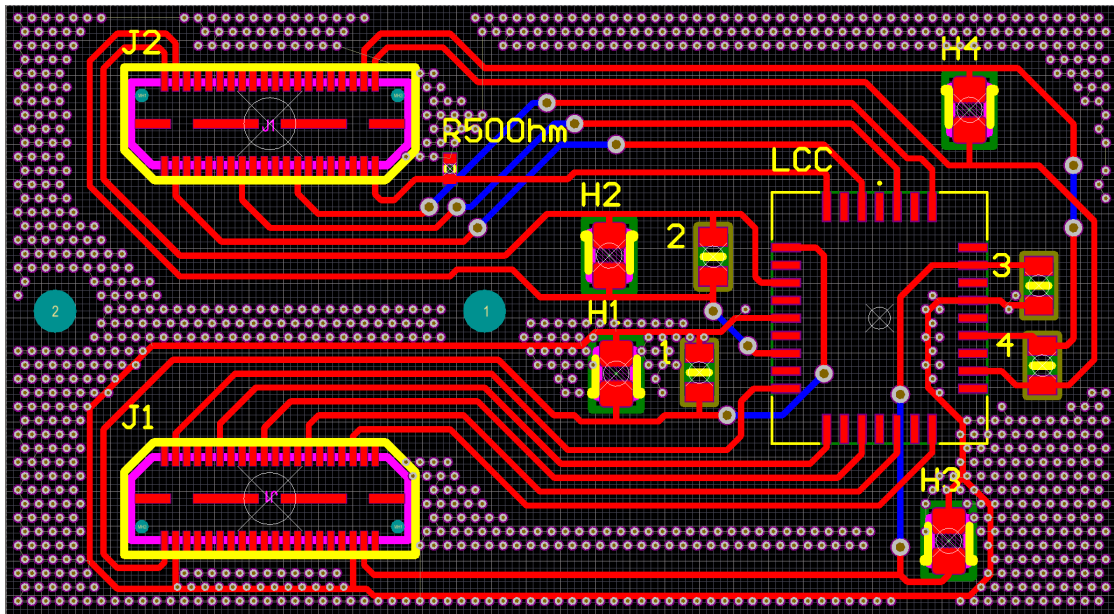


FIGURE 2: layout of PCB design. Polygon pours are hidden for clarity

Photos of hardware

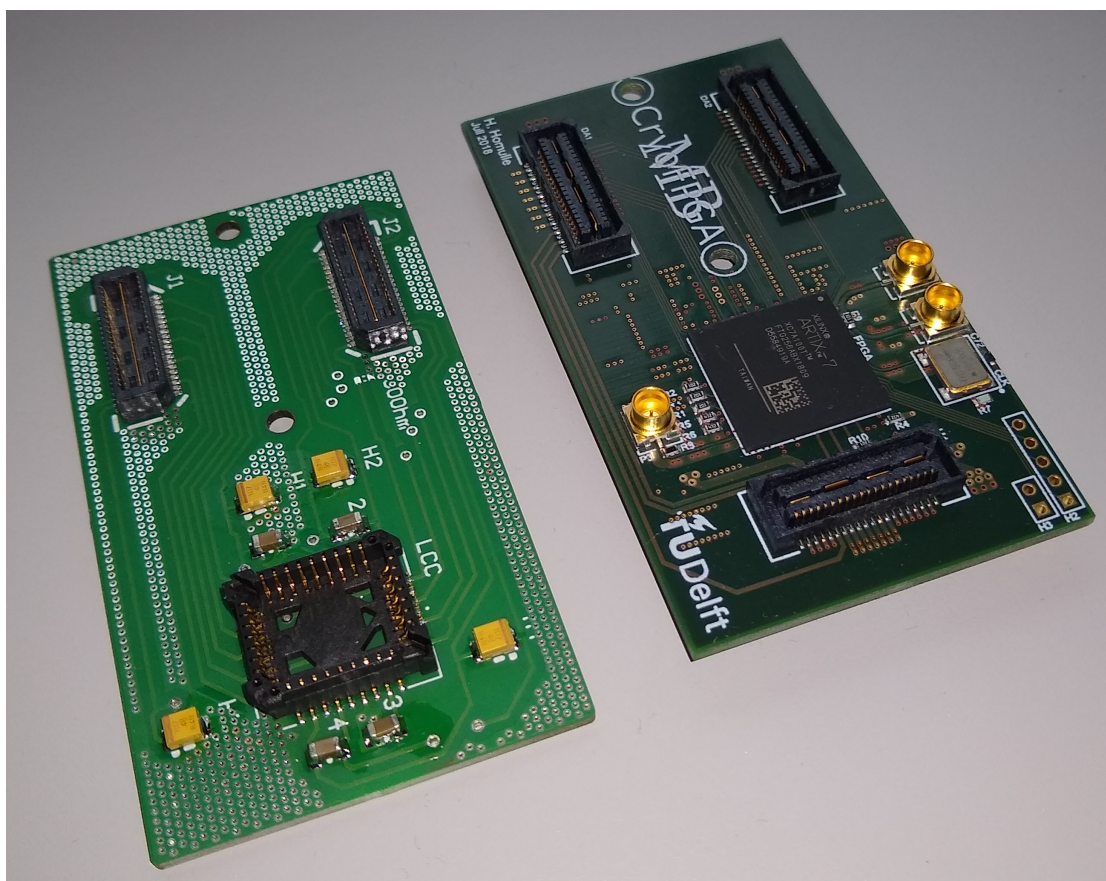


FIGURE 3: Photo of designed PCB (left) and motherboard ARTIX-7 PCB (right)