Compiled analog and digital building blocks in 22nm FDSOI

Marjeris Romero

Master of Science in Electronics
Submission date: August 2018
Supervisor: Trond Ytterdal, IES

Norwegian University of Science and Technology
Department of Electronic Systems

Problem description

The continuous downscaling of CMOS technologies provides new challenges and opportunities for energy efficient integrated circuits.

The main objective of this master thesis is to implement compilation of some key analog and digital building blocks in 22nm FDSOI CMOS technologies for application in medical ultrasound imaging applications. The project consists of the following tasks:

- Choose a set of analog and digital building blocks
- Implement the blocks on transistor level
- Develop the required input files for the compiler
- Use the compiler to generate the layout
- Characterize and verify the blocks based on netlist extracted from layout

-

Abstract

This project shows the process of designing a cell library in a 22nm FDSOI process. Part of this project was also to inspect the viability of using a custom layout compiler presented in [8] for the 22nm node. For each cell in the library the input files for the compiler had to be generated and the compiled layout was created and compare to the manual layout of the cell.

Sammendrag

Dette prosjektet dokumenterer prosessen rundt å designe et teknologibibliotek for en 22nm FDSOI prosess. En del av prosjektarbeidet var også å undersøke muligheten for å bruke en layout-kompilator fra [8] på 22nm. For hver komponent i biblioteket ble inndata til kompilatoren generert og den kompilerte layouten ble sammenlignet med en tilsvarende layout som var tegnet manuelt.

Preface

This thesis was carried out during the spring of 2018, concluding a Master of Science degree in Electronics at the Norwegian University of Science and Technology (NTNU) in Trondheim.

The work in this thesis was aimed to help the transition into new technologies at the Centre for Innovative Ultrasound Solutions (CIUS), a research-based innovation centre focusing on ultrasound solutions for health care, maritime, oil & gas. The work involves design, modelling and verification of both individual circuits and larger building blocks in CMOS that can be used to implement larger systems, e.g, for ultrasound imaging.

I would like to thank my supervising professor Trond Ytterdal for his most needed help and support throughout this project. Thanks also to the board members at Omega Verksted, for the amount of coffee and laughters shared together these past years as a student. I also want to thank Torbjørn for his most kind words of encouragement when I was struggling the most.

Marjeris Romero

Contents

Problem description **i**

Abstract **iii**

Sammendrag **v**

Preface **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 Previous work 2

1.3 Main contributions 2

1.4 Thesis outline 2

2 Theory **5**

2.1 Transistor properties 5

2.1.1 Operating regions 5

2.1.2 Subthreshold operation 6

2.1.3 Series and parallel transistors 6

2.2 FDSOI transistors 7

2.2.1 Body biasing 8

2.2.2 Layout considerations 8

2.3 Layout generation tool 8

2.4 Digital components 9

2.4.1 The Inverter 9

2.4.2 The NAND Gate 9

2.4.3 Compound gates 10

2.5 Ring oscillator 10

2.6 Analog components 11

2.6.1 Basic Current Mirror 11

2.6.2 Common-Source Amplifier 11

3 Methodology **13**
3.1 Unit transistor 13
3.2 Inverter 14
3.3 NAND 15
3.4 7-stage Ring oscillator 16
3.5 Common-source amplifier 16
3.6 Layout generation 17

4 Results **19**

5 Discussion **23**
5.1 Using the compiler 23

6 Conclusion **25**
6.1 Further work 25

Appendix A Layout and schematic in Cadence **29**
A.1 Common source amplifier 30
A.2 Ring oscillator 32
A.3 NAND layout 34
A.4 Inverter layout 35

Appendix B Compiled cells **37**
B.1 Common source amplifier 38
B.2 Common source amplifier 39

Appendix C Compiler input code **41**
C.1 Core transistors definition file 41
C.2 Digital cells netlist 53
C.3 Digital cells object definition file 55

List of Figures

2.1 NMOS low frequency small signal model 5
2.2 Equivalent circuit for series connected transistors 7
2.3 Equivalent circuit for parallel connected transistors 7
2.4 Illustration of BULK CMOS and FDSOI CMOS 7
2.5 Symbol for an inverter logical gate 9
2.6 Symbol for a nand logical gate 10
2.7 Symbol for an and logical gate 10
2.8 A N-stage ring oscillator 11
2.9 A simple CMOS current mirror 11
2.10 A common-source amplifier with a current-mirror active load 12

3.1 Layout of NFET unit transistor 14
3.2 Schematic of the inverter 15
3.3 Schematic of the NAND gate 16
3.4 Schematic of the 7 stage ring oscillator 16
3.5 Schematics of the common source amplifier 17

4.1 Voltage transfer curve for the balanced inverter. 20
4.2 Transient analysis of the ring oscillator. 20
4.3 AC analysis of the common source amplifier. 21

A.1 Layout of the common source amplifier 30
A.2 Schematic of the common source amplifier 31
A.3 Testbench of the common source amplifier 31
A.4 Layout of the 7 stages ring oscillator 32
A.5 Testbench of the 7 stages ring oscillator 33
A.6 Layout of NAND gate 34
A.7 Layout of inverter gate 35

B.1 Layout of the compiled common source amplifier 38
B.2 Compiled layout of the 7 stage ring oscillator 39

List of Tables

2.1 Inverter truth table 9
2.2 NAND truth table 10
2.3 True table of a and gate 10

4.1 Measurements of the cells with the manual layout 19
4.2 Measurements of the cells with the compiled layout 19

Abbreviations

FD-SOI	Fully Depleted Silicon On Insulator
DRC	Design Rule Check
LVS	Layout Versus Schematic
MOSFET	Metal-Oxide Semiconductor Field-Effect Transistor
PMOS	P-channel MOSFET
NMOS	N-channel MOSFET
CMOS	Complementary Metal-Oxide Semiconductor
PC	Polycrystalline (silicon)
AUXPC	Auxillary Polycrystalline

1 Introduction

Technology around us is developing at an accelerated pace and this includes medical devices as well. There is a need for higher energy efficiency and lower power consumption and to explore technologies in 22nm and lower for ultra-low power implementations.

Silicon-on-insulator devices designed for optimum operation at 0.3V promise longer operational life than conventional application-specific integrated circuits [7]. Ultra-low power (ULP) transistors are enabling technology progress in areas such as implantable medical devices and energy harvesting circuits, but also increases the life span of any sensor system, since the most efficient way to reduce power is to reduce the operating voltage.

Each time the industry moves to a new technology node, there are certain challenges that need to be faced and a set of building blocks need to be made for a specific function and technology. FDSOI technology and bulk biasing can also contribute to even lower power consumption by reducing leakage and allowing lower supply voltage operation. Bulk biasing also allows equal sizing of NMOS and PMOS transistors, as opposed to the conventional 2*wider PMOS transistor size, resulting in reduced circuit area and capacitance[7].

1.1 Motivation

Technology libraries are usually sold as IPs from external vendors and are used by analog designers in order to speed up the design process. Since we need to scale down to a new technology node brings the need of developing new building blocks that target specific task and a speficic technology, it would be ideal to lower the designing time of some of the basic cells in a digital and analog library by using some of the layout generation tools available out there. The cicCreator is one of these tools and is open-source and available at [2].

This layout optimization tool helps speed up the process of generating multiple versions of analog IC layout for quicker layout parasitics extraction and post-layout simulation. Analog integrated circuits (ICs) have more considerations than the design of digital circuits, with long design cycles. Any tool that can speed up the design process and shorten time to market will help to reduce the overall cost of manufacturing ICs.

Since the compiler used in this project works in a hierarchical structure, there is the need to describe digital and analog circuit building-blocks so they can be used as custom library objects in future implementations of other CMOS circuits. All objects need to be fully technology-independent, and placement and routing should be defined in a way that allows easy implementation by the IC designer.

The optimization tool for the layout of the cells used in this project is presented in [8], and there was a need to examine the portability of the existing compiler to smaller process nodes. A task that may or not may be possible since smaller nodes imply more layout constraints.

Since low power has become the biggest concern for almost every practical use in the industry, transistors with an extremely low threshold voltage will be used.

1.2 Previous work

Previous work with compiled cells using the compiler tool[2] used in this project is presented in [8].

The author had difficulties finding cells previously made for 22nm FDSOI, apart for a single-stage power amplifier for WLAN in 22nm FDSOI[4], so it seems as the implementation of a cell library for this node had never been try before.

1.3 Main contributions

The work of this master thesis consists in the implementation of building blocks in the 22nm technology using the compiler presented in [8].

The main contributions of this thesis are

- A compiled inverter with minimum gate length in 22nm FDSOI.
- A ring oscillator with minimum gate length in 22nm FDSOI.
- A common-source amplifier using 22nm FDSOI technology.

1.4 Thesis outline

The rest of this thesis is organized as follows

Chapter 2 – Theory: This chapter contains the background theory used in the rest of this thesis.

Chapter 3 – Methodology: Shows the implementation of the cell library in Cadence.

Chapter 4 – Results: Shows results of the cells implemented after their parasitics had been extracted from layout.

Chapter 5 – Discussion: Discuss the results and some of the challenges in the use of the compiler for this technology.

Chapter 6 – Conclusion: Final thoughts and further work.

2 Theory

This chapter presents a brief summary of some of the concepts used throughtout the rest of this project. It is assumed that the reader possesses a basic knowledge of analog and digital circuits from before.

2.1 Transistor properties

There are two types of MOSFET transistors: nMOS (n-channel) and pMOS (p-channel). N-channel devices use electrons as the majority current carriers, and P-channel devices use holes to form a conductive channel.

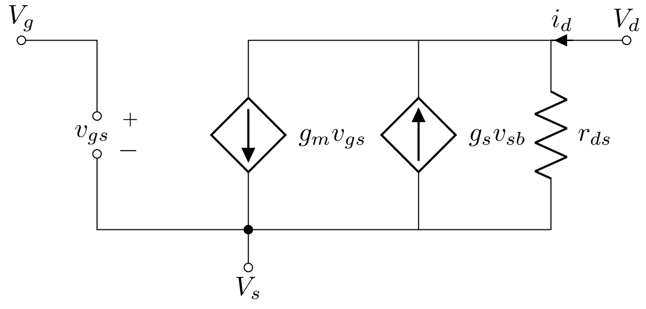

Figure 2.1: NMOS low frequency small signal model

2.1.1 Operating regions

The behavior of a transistor can be broken down into 3 main parts:

- Triode region
- Active region (saturation)
- Off (subtreshold)

Triode region - $V_{GS} > V_{th}, V_{DS} < (V_{GS} - V_{th})$. The drain current is proportional to V_{DS}, the same kind of relationship as in a resistor. Therefore the MOSFET can be use as a resistor in this region.

Active region - $V_{GS} > V_{th}, V_{DS} \geq (V_{GS} - V_{th})$. A change in I_D can be achieved by changing V_{GS}. This is also called the saturation region.

$$I_D = \frac{1}{2}\mu_n C_{ox}(\frac{W}{L})(V_{GS} - V_{tn})^2 \tag{2.1}$$

The transconductance g_m is then given by

$$\frac{dI_D}{dV_{GS}} = \mu_n C_{ox}(\frac{W}{L})(V_{GS} - V_{tn}) \tag{2.2}$$

2.1.2 Subthreshold operation

In subthreshold operation, also called weak inversion, transistors obey an exponential voltage current relationship instead of a square-law. A small but finite current flows even when $V_{GS} = 0$. In the subthreshold region, the drain current is approximately given by an exponential relationship:

$$I_{D(sub-th)} \cong I_{D0}(\frac{W}{L})e^{(qV_{eff}/nkT)} \tag{2.3}$$

Plotting drain current on a logarithmic axis versus V_{GS} in the subthreshold region gives a straight line. The inverse of this slope, called the subthreshold slope and equal to $ln(10) * \frac{nkT}{q}$ is a measure of the voltage change in V_{GS} required to effect an order-of-magnitude change in subthreshold drain current.

The current does not drop to zero even when $V_{GS} = 0V$. This residual drain current is called the subthreshold leakage and is given by

$$I_{off} = I_{D0}(\frac{W}{L})e^{(-qV_t/nkT)} = (n-1)\mu_n C_{ox}(\frac{W}{L})(\frac{kT}{q})^2 e^{(-qV_t/nkT)} \tag{2.4}$$

As we see from the equation above, the subthreshold offset drain current has a high dependency on the absolute temperature (T), carrier mobility (μ_n) and threshold voltage (V_t). In general, subthreshold leakage increases significantly with temperature and is often a dominant source of power consumption in modern technologies.

2.1.3 Series and parallel transistors

It is common practice to connect several unit transistors in series or parallel to scale the effective width or length up or down. If two unit transistors are connected in series, the effective length will increase as shown in figure 2.2. The width of the equivalent transistor will as well increased by connecting the unit transistors in parallel.

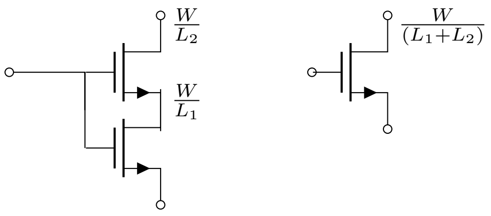

Figure 2.2: Equivalent circuit for series connected transistors

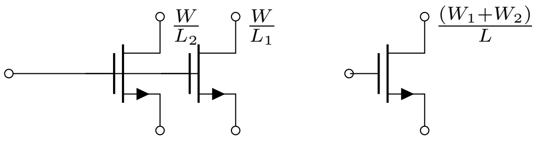

Figure 2.3: Equivalent circuit for parallel connected transistors

2.2 FDSOI transistors

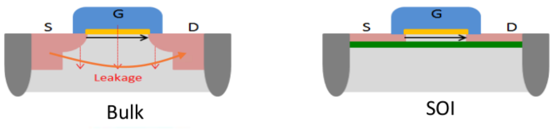

Figure 2.4: Illustration of BULK CMOS and FDSOI CMOS

The market for semiconductors now focus on energy savings and the fully-depleted silicon-on-insulator (FDSOI) is a planar process which is thought to help extend the relevance period of Moore's law[3].

FDSOI reduces the leakage and thus has the possibility to minimize power consumption. Advantages of an FDSOI technology includes the reduction of parasitic capacitance between the source and drain of the transistor. The buried oxide layer also constrains electrons flowing between the source and drain to reduce performance- and power-degrading leakage currents significantly. FDSOI also allows to further control transistor behaviour by applying a voltage to the substrate underneath the device, called also body biasing. Body biasing introduces a new concept in processor design, different voltages applied to the top and buried gate affect the characteristics of the transistor, which can be then be optimized for either high performance or low power.

2.2.1 Body biasing

Modern FDSOI processes introduces the possibility to apply a voltage into the back gate and use it as a fourth terminal. The back gate allows controlling the threshold voltage by about $85mV/V$ when changing the back gate voltage[5], this is what is called as body biasing.

Biasing is more efficient in FDSOI, thanks to the dielectric isolation by the buried oxide layer. For Forward Body Biasing (FBB), the transistor required less voltage in the gate to switch, resulting in faster transistor switching and lower active power consumption. Similarly, Reverse Body-Biasing (RBB) can be applied to the transistors to higher the threshold voltage of the transistor, which lowers the off-stage leakage and minimised the static power consumption when the transistors are off.

Body biasing capabilities in FDSOI opens a variety of opportunities such as achieving lower threshold voltages for the devices and lower power, and it can also be used for compensating process variations in a cell.

2.2.2 Layout considerations

In current advanced processes such as 20nm or 14nm there are several layout considerations that must be taken into account[1]. The process used in this project allows for activating what is called multi-patterning in the photolithography process. This technique helps to enhance the feature density, allowing layout engineers to place the devices with closer spacing between them. The spacing between the metal shapes is now so small that current light sources cannot print them reliably, so the solution with multiple patterning consists of splitting the dense shapes into two masks and relying on interference patterns between the light from both masks to make the final projection. This way the metal layer M1 will actually consist of 2 layer masks, marked with different colors in Cadence.

2.3 Layout generation tool

The layout compiler used in this project is presented in [8], where an ADC was compiled from a SPICE netlist, a technology file and an object definition file into a DRC/LVS clean layout and schematic in 28-nm FDSOI.

The compiler borrows the concept of inheritance from object-oriented programming and outputs a GDSII file that can be loaded in Cadence Virtuoso. Parasitic extractions, simulation and verification can then be performed.

Both the SPICE netlist and the object definition file are technology independent. Instead of specifying transistors widths and lengths, the SPICE netlist only contains permutations of unit transistors, either by series-connecting or parallel connecting these. The routing of blocks is done in either by connectivity routing or in the object definition file, which is written in JavaScript Object Notation, a commonly used data exchange format.

The technology rule file specifies the dimension constraints for a specific technology, the GDSII layer numbers and data-type, layer material definitions, and other design rules.

2.4 Digital components

Digital logic gates describes the functionality of a circuit in terms of Boolean values. In CMOS, logic gates are composed of a pull-up network made by PMOS devices, and a pull-down network made by NMOS devices.

2.4.1 The Inverter

The inverter is one of the most basic blocks in all digital systems. The static CMOS inverter is composed of a NFET and a PFET. Its operation is easily understood with a simple switch model of the MOS transistor. The transistor is modelled with an infinite off-resistance (for $|V_{GS}| < |V_T|$), and a finite on-resistance (fr $|V_{GS}| > |V_T|$). When the input voltage is high and equal to the supply voltage V_{DD} the NMOS transistor is on, while the PMOS is off, resulting in V_{out} and the ground node being connected and the resulting voltage of zero. On the other hand a input voltage of 0V causes the NMOS transistor to be off and PMOS to be on, a direct path between V_{DD} and V_{out}, yielding in a high output voltage.

For a balanced inverter the voltage swing is equal to the supply voltage, and the swiching threshold V_M is located around the middle of the available voltage swing (or at VDD/2). This usually requires making the PMOS devices a bit larger than the NMOS devices, which means making the PMOS wider, increasing the strength of the PMOS. Increasing the strength of the NMOS, on the other hand moves the switching threshold closer to GND.

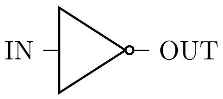

Figure 2.5: Symbol for an inverter logical gate

Table 2.1: Inverter truth table

A	Y
0	1
1	0

2.4.2 The NAND Gate

The NAND gate is other of the basic blocks in digital design. Figure 2.7 shows a 2-input CMOS NAND gate. It consists of two series NMOS transistors between the output and V_{SS}, and two parallel PMOS transistors between the output and V_{DD}. The truth table is given in Table 2.2 and the symbol is shown in Figure 2.7.

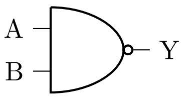

Figure 2.6: Symbol for a nand logical gate

Table 2.2: NAND truth table

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

2.4.3 Compound gates

The AND gate can be formed by combining NOT and NAND gates.

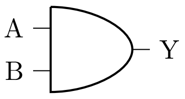

Figure 2.7: Symbol for an and logical gate

Table 2.3: True table of a and gate

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

2.5 Ring oscillator

Single-ended ring oscillators are realized by placing an odd number of open-loop inverting amplifiers or delay cells in a feedback loop configuration. Assuming each inverter has a delay of T_d and that there are N number of inverters, the half period of oscillation would be given by

$$\frac{T_0}{2} = nT_d \tag{2.5}$$

and thus

$$f_0 = \frac{1}{T_0} = \frac{1}{2nT_d} \tag{2.6}$$

where f_0 is the operational frequency of the oscillator and T_d is the delay through one delay stage.

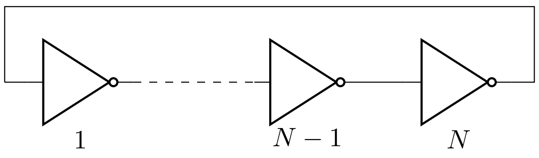

Figure 2.8: A N-stage ring oscillator

2.6 Analog components

2.6.1 Basic Current Mirror

Figure 2.9 shows an ideal current mirror that accepts an input current I_{in} and produces an output current $I_{out} = I_{in}$. An ideal current mirror will have zero input resistance and high output resistance, and reproduces the input current regardless of the source and load impedances that are connected to it[6]. It is assumed that Q_1 and Q_2 are both in the active region. When Q_1 and Q_2 are the same size, the drain current through the transistors will be identical, since they have the same gate-source voltage, V_{gs}. A common use of a simple current mirror is in a single-stage amplifier with active load.

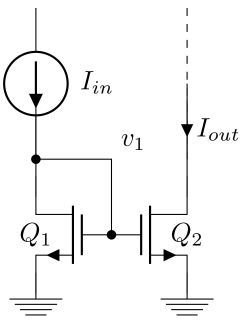

Figure 2.9: A simple CMOS current mirror

2.6.2 Common-Source Amplifier

An amplifier increases the amplitude of a input signal. The gain of a common source amplifier is derived by using the small signal model of the the amplifier as shown in 2.10.

The common source amplifier is a popular gain stage, specially when high input impedance is desired. The use of a current mirror as an active load provides large small signal resistances without large dc voltage drops.

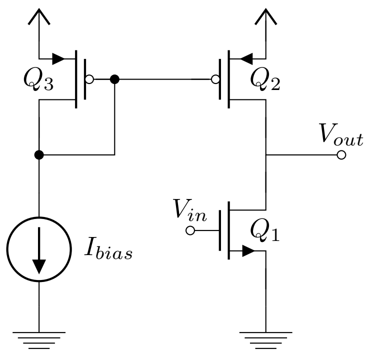

Figure 2.10: A common-source amplifier with a current-mirror active load

3 | Methodology

For this project two versions of each cell were made. First a manually drawn version using the schematic and layout GUI in Cadence Virtuoso. Then a second version was made by writing a spice netlist and an object definition file for the compiler that generated an automatic layout. The goal was then to simulate the two versions with parasitic capacitances extracted from the layout and discuss differences. The digital cells had a targeted V_{DD} of $300mV$.

The 22nm FDSOI process used in this project provides both low threshold voltage PMOS and NMOS devices. The back-gate bias can be used to calibrate the threshold voltage of the transistors as explained in Section 2.2.1

To extract the different parameters and properties of the devices designed, a simple test bench was created in order to simulate the devices under more realistic conditions.

3.1 Unit transistor

In order to make the modifications of the cells easier, a unit transistor was made in Virtuoso. For the digital cells the layout of the transistors were made in a way that allows stacking multiple transistors together into other larger cells. Since the 22nm FD-SOI technology has much potential for using body biasing in new ways, the bulk contact of both NFET and PFET devices are always made available in the layout of every digital cell. The unit transistor for NFET is depicted in Figure 3.1 with dimensions 20nm in length and 100nm width.

For placing devices adjacent to each other in the most area efficient way, a unit transistor with two dummy polys was made instead of the four dummy poly each transistor need when the gate length is $20nm$. The auxiliary poly (AUXPC) polys are shown in red with diagonal lines. This layer is used over each end of the active region of the transistor in order to minimize the mechanical stress and is one of the design rules for this technology.

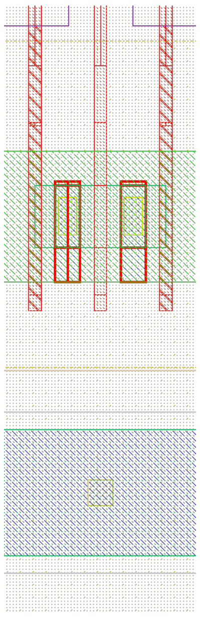
Figure 3.1: Layout of NFET unit transistor

3.2 Inverter

For the inverter cell both the PMOS ("pch") and NMOS ("nch") unit transistors were used, as seen in Figure 3.2.

One of the many constraints that exist for minimal gate sizing in the used technology is that the a each gate needs to have 2 dummy polys on each side for the design to be DRC clean. Therefore a termination cell with 2 dummy polys was placed on each side of every cell before running DRC and LVS checks. The layout of the inverter cell can be seen in Figure A.7.

The bulk voltages are set to $V_{BN} = 0V$ and $V_{BP} = 0,075V$ in order to balance the inverter, as explained in Section 2.4.1.

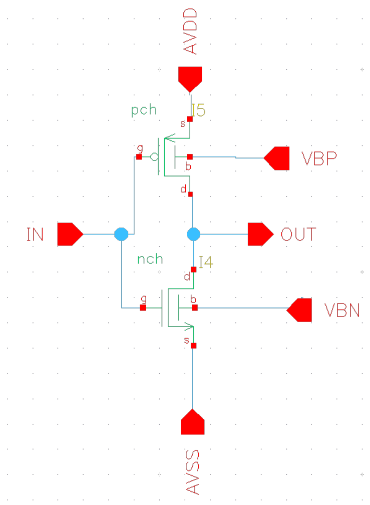

Figure 3.2: Schematic of the inverter

The values for the body biasing voltages were found by performing a sweeping simulation of the bulk voltages of both nch and pch transistors until $V_{DD}/2$ at the input leads to $V_{DD}/2$ on the output. This allows using the same width for both NMOS and PMOS devices.

3.3 NAND

The schematic in Figure 3.3 shows a basic NAND gate as described in Section 2.4.2. The bulk voltages of the NFET and PFET were set as the same as in the inverter in Section 3.2.

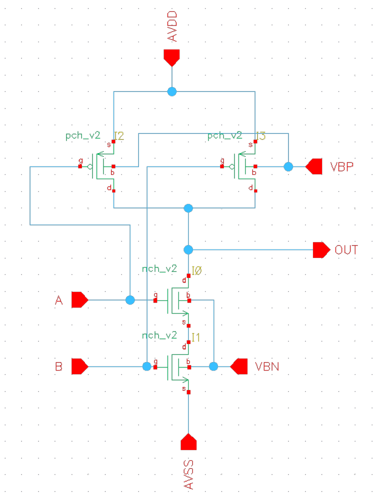

Figure 3.3: Schematic of the NAND gate

3.4 7-stage Ring oscillator

Figure 3.4 depicts a 7-stage ring oscillator with target frequency of 500 MHz and supply voltage of $V = 0,7V$. The NAND gate is used to enable the oscillation. The testbench is shown in A.5 and the layout in A.4.

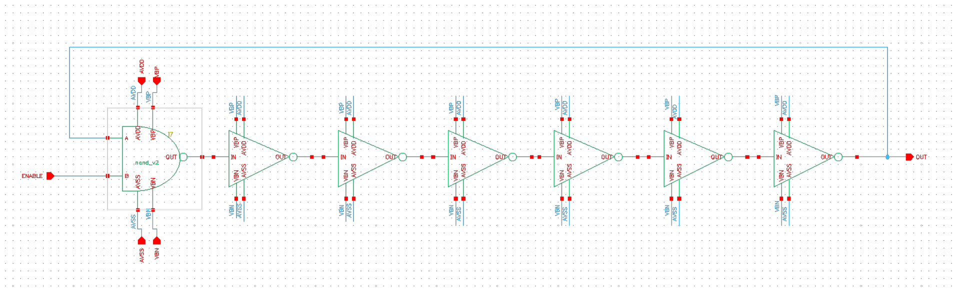

Figure 3.4: Schematic of the 7 stage ring oscillator

3.5 Common-source amplifier

Figure A.2 shows the schematics of a common source amplifier in 32nm. The width of the unit transistors used here are set to $320nm$. The supply voltage was $700mV$ and $I_{bias} = 10\mu A$. A current mirror is used as the load like described in Section 2.6.2. The current through P0 and the two current mirrors is controlled by I_{bias}.

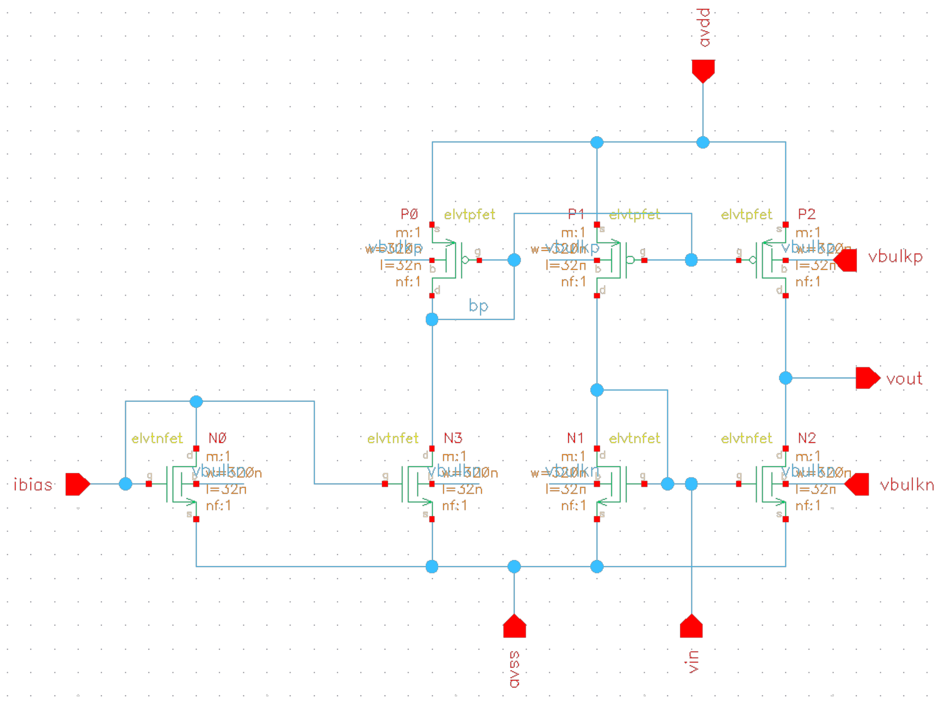

Figure 3.5: Schematics of the common source amplifier

3.6 Layout generation

Part of the work in this project involved defining different aspects 22nm manufacturing that had to be considered in other to improve the compiler.

For each cell a spice netlist was created and the object input files for the compiler were written in order to generate the layout and can be seen in the Appendix C.1.

Some adjustments in the technology file had to be made in order to introduce other layers that were needed. Part of this project was to check the usability of the compiler for lower nodes at minimum gate length. The problems encounter in this stage of the project are discussed in Section 5.1.

4 Results

The voltage transfer curve of the inverter gate after being balanced with body biasing is shown in Figure 4.1.

The transient analysis of the 7 stage ring oscillator is shown in Figure 4.2. The operational frequency of the 7-stage ring oscillator was measured to $569MHz$.

The AC analysis of the common source amplifier is shown in figure 4.3.

The generated layout of each cell is shown in Appendix B.

The measured sizes of the cells are presented in Table 4.1 and Table 4.2.

Table 4.1: Measurements of the cells with the manual layout

Name of the cell	Length$[nm]$	Width$[nm]$	Area$[nm^2]$
Inverter	$0,312 \cdot 10^{-2}$	$1,868 \cdot 10^{-2}$	$0,583 \cdot 10^{-4}$
7-stages RO	$2,864 \cdot 10^{-2}$	$1,868 \cdot 10^{-2}$	$5,350 \cdot 10^{-4}$
CS amp	$2,107 \cdot 10^{-2}$	$2,799 \cdot 10^{-2}$	$5,897 \cdot 10^{-4}$

Table 4.2: Measurements of the cells with the compiled layout

Name of the cell	Length$[nm]$	Width$[nm]$	Area$[nm^2]$
Inverter	$0,732 \cdot 10^{-2}$	$4,144 \cdot 10^{-2}$	$3,033 \cdot 10^{-4}$
7-stage RO	$3,168 \cdot 10^{-2}$	$4,191 \cdot 10^{-2}$	$13,277 \cdot 10^{-4}$
CS amp	$1,845 \cdot 10^{-2}$	$4,160 \cdot 10^{-2}$	$7,675 \cdot 10^{-4}$

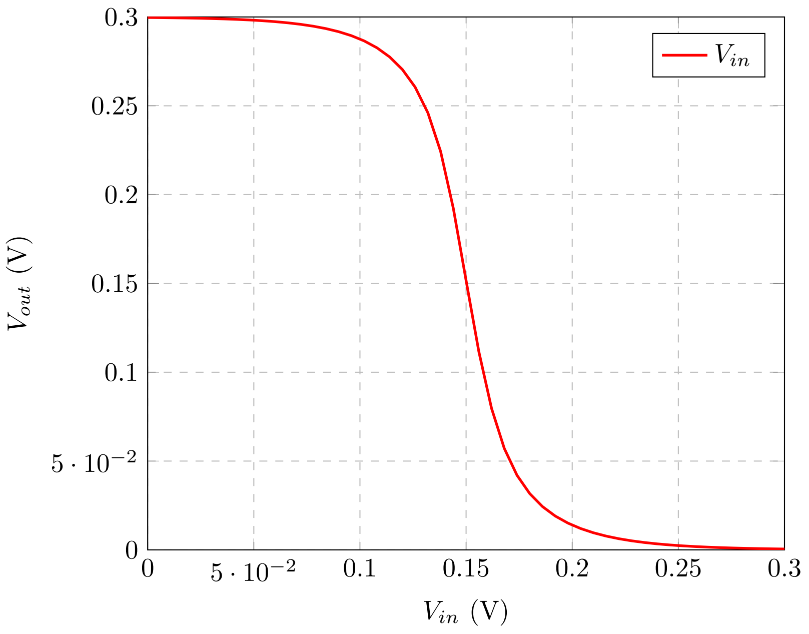

Figure 4.1: Voltage transfer curve for the balanced inverter.

Trans analysis of the ring oscillator, with initial condition $V_{out} = 0V$

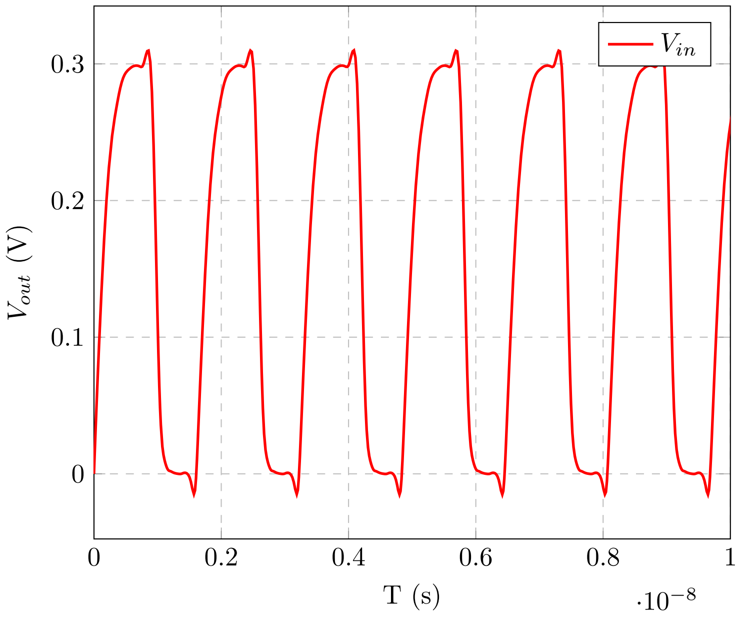

Figure 4.2: Transient analysis of the ring oscillator.

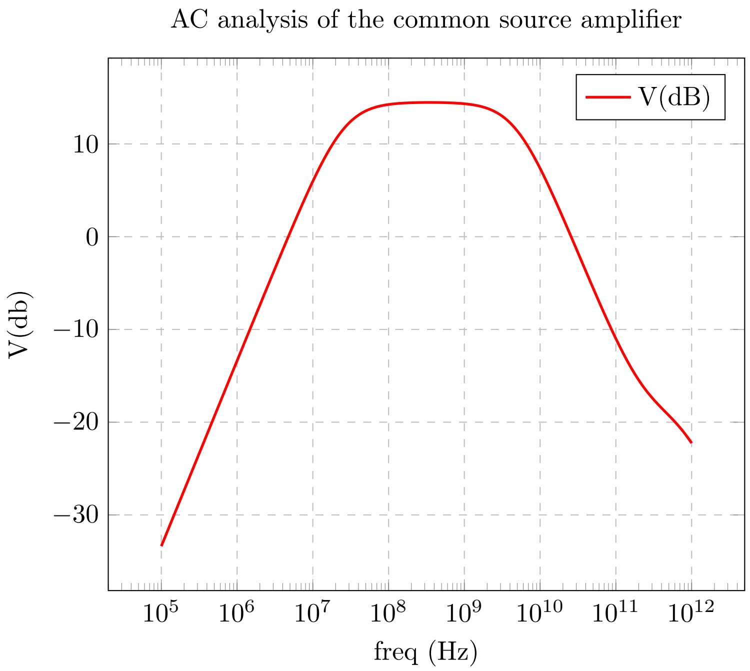

Figure 4.3: AC analysis of the common source amplifier.

5 Discussion

The desired output of this project was to have two versions of each cell in the library. One version with the manually drawn layout, and the other with the compiled layout, and then to characterize each cell version and compare with the other. In order to do this each cell needed to be able to pass both DRC and LVS checks to extract parasitic capacitances. However this turned out to be more difficult to achieve than expected, because the 22nm FDSOI technology had many new layout constraints compared to what was supported in the current version of the compiler. The problems encountered when using the compiler are discussed in Section 5.1.

The results presented in Chapter 4 are simulated from the manual layout of the cells and are in compliance with the design specifications.

The area of each cell was compared and we see that smaller area is achieved with the manual layout, in some cases as much as 50%. The designer should then consider if this is a critical factor or if some extra area is an acceptable tradeoff for lower design time of a cell by using the compiler.

5.1 Using the compiler

A number of challenges were encountered when using the available compiler with the minimum gate length for this technology of $20nm$.

When using minimum length the gate of each transistor needs to have 4 supporting gates with defined size and spacing. This was solved by using two termination cells with dummy polys at each cell with minimum length before running DRC.

When routing with the poly layer (PC) one must also have an auxiliary poly (AUXPC) as a dummy poly all the way from one transistor edge to another. The problem is that an AUXPC poly used for the edges of the active region of the transistor cannot be connected with ports of different nets. A poly cut layer (CT) is used to cut the dummy poly afterwards. The CT layer was added in the technology definition file and incorporated in the definition of the core transistor cell, so when transistors are stacked, the cut layer will respect the correct size and spacing rules automatically.

In the technology file of the compiler, one can change the number of cuts and vias used for each layer. The standard is two cuts, but using two cuts in both drain and source causes spacing errors for vias from M1 to M2. Therefore the compiler in its current state is not suitable for more complex cells that require routing in M2 or M3 in adjacent transistors when the gate length lower than $32nm$.

Though the compiler was not an optimal tool for minimum gate devices, it was easy to use when the gate length was set to $32nm$, since this eliminates the problems with the sizing and spacing of the vias for higher metals.

6 Conclusion

FD-SOI technology provides many advantages in order to make circuits more energy efficient and lower the area. Back-biasing mechanisms gives more effective optimization of circuits.

In this project the design of an inverter gate, a nand gate, a 7-stage ring oscillator and a common-source amplifier was presented. The layout generation tool from [8] is used and compare with manual layout in Virtuoso.

As can be seen from the simulation results the area of the compiled cells is in some cases significantly increased, but when considering the drastically reduced design time and possibility of rapid experimentation and prototyping this seems like a promising trade.

6.1 Further work

Compound cells like for instance the AND gate can now be made from the cells presented in this project. Could have made more powerful cells with double the transistors, CSX2. Other things for further investigation includes activating sharing between the drain and source of two different transistors to make a continuous RX with the compiler, so we could have achieve smaller size of the cells, though this would have presented new challenges with via spacing.

Bibliography

[1] *Cadence Unveils New Virtuoso Advanced Node for 20nm Design.* `https://www.cadence.com/content/cadence-www/global/en_US/home/company/newsroom/press-releases/pr/2013/cadenceunveilsnewvirtuosoadvancednodefor20nmde` `html`. (Accessed on 01/02/2018).

[2] C.Wulff. *Custom IC Creator.* URL: `https://github.com/wulffern/ciccreator`.

[3] *FD-SOI Technology Innovations Extend Moore's Law.* `https://www.globalfoundries.com/sites/default/files/technicalpaper/fd-soi-technology-extend-moores-law.pdf`. (Accessed on 19/01/2018).

[4] S. T. Lee, A. Bellaouar, and S. Embabi. "A high-efficiency single-stage power amplifier for WLAN 802.11ac in 22nm FDSOI". In: *2017 IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S).* Oct. 2017, pp. 1–3. DOI: `10.1109/S3S.2017.8309265`.

[5] S. S. Rao et al. "Body biasing for analog design: Practical experiences in 22 nm FD-SOI". In: *2017 IEEE 20th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS).* Apr. 2017, pp. 73–78. DOI: `10.1109/DDECS.2017.7934580`.

[6] D. Johns T.C Carusone and K. W. Martin. *Analog Integrated Circuit Design.* Wiley, 2012. ISBN: 9781118092330.

[7] S. A. Vitale et al. "FDSOI Process Technology for Subthreshold-Operation Ultralow-Power Electronics". In: *Proceedings of the IEEE* 98.2 (Feb. 2010), pp. 333–342. ISSN: 0018-9219. DOI: `10.1109/JPROC.2009.2034476`.

[8] C. Wulff and T. Ytterdal. "A Compiled 9-bit 20-MS/s 3.5-fJ/conv.step SAR ADC in 28-nm FDSOI for Bluetooth Low Energy Receivers". In: *IEEE Journal of Solid-State Circuits* 52.7 (July 2017), pp. 1915–1926. ISSN: 0018-9200. DOI: `10.1109/JSSC.2017.2685463`.

A Layout and schematic in Cadence

A.1 Common source amplifier

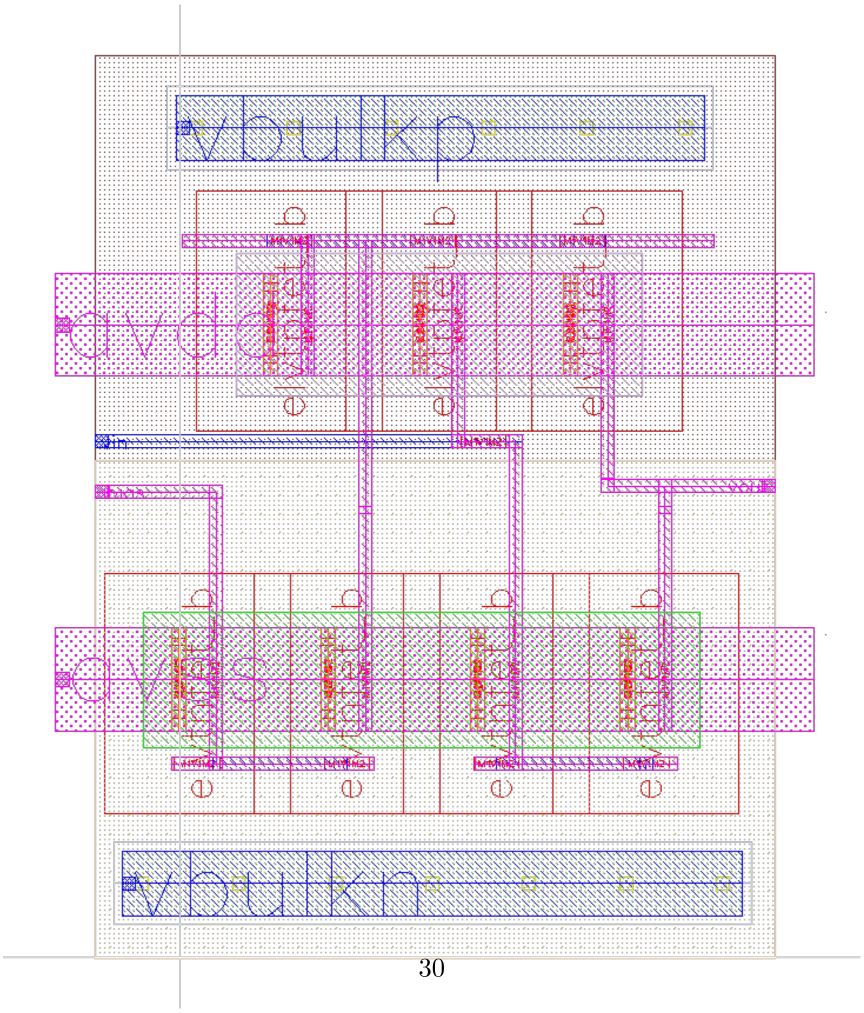

Figure A.1: Layout of the common source amplifier

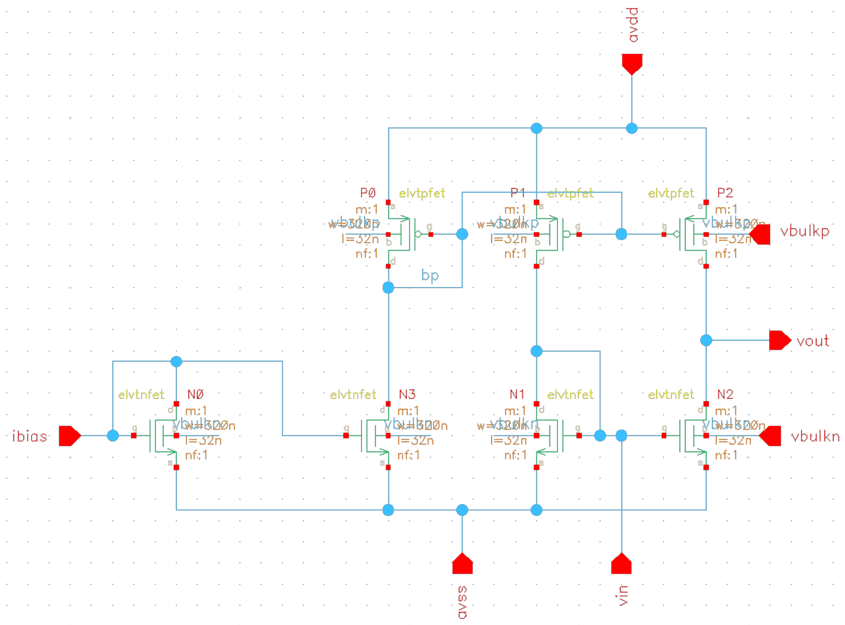

Figure A.2: Schematic of the common source amplifier

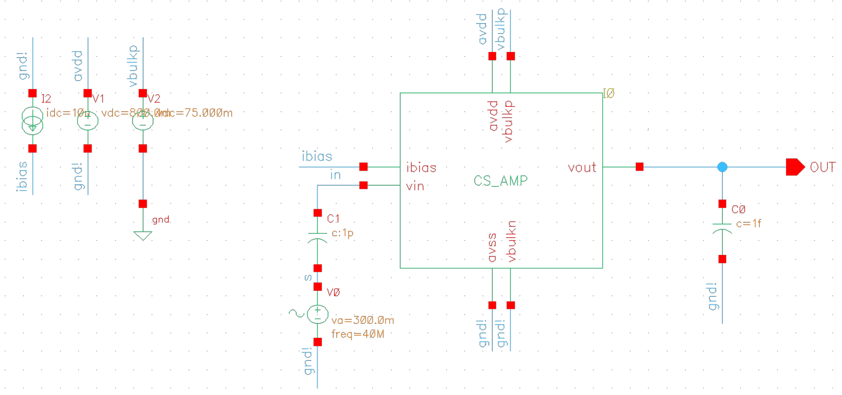

Figure A.3: Testbench of the common source amplifier

A.2 Ring oscillator

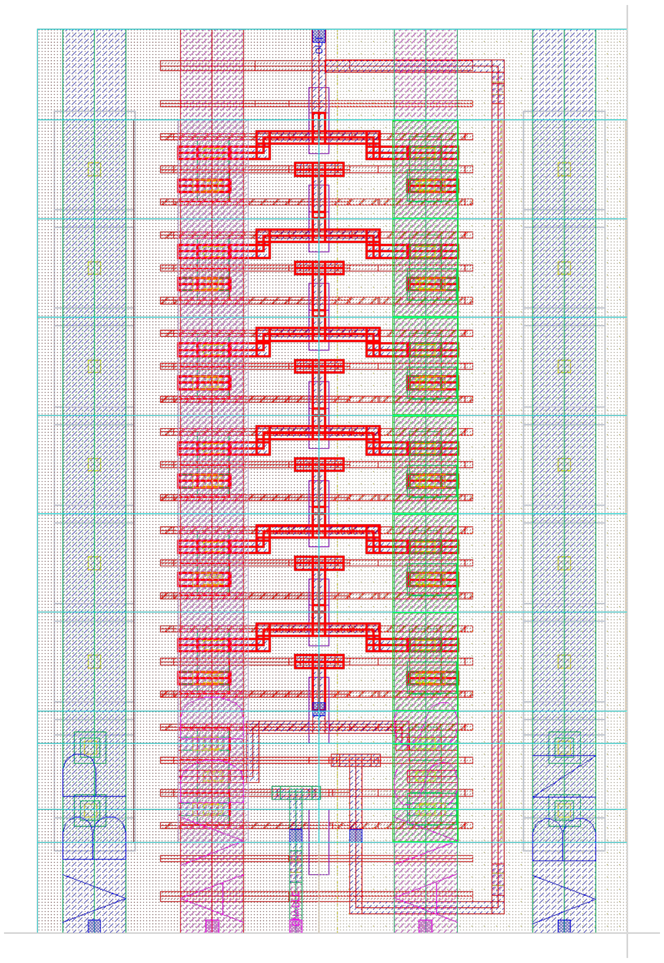

Figure A.4: Layout of the 7 stages ring oscillator

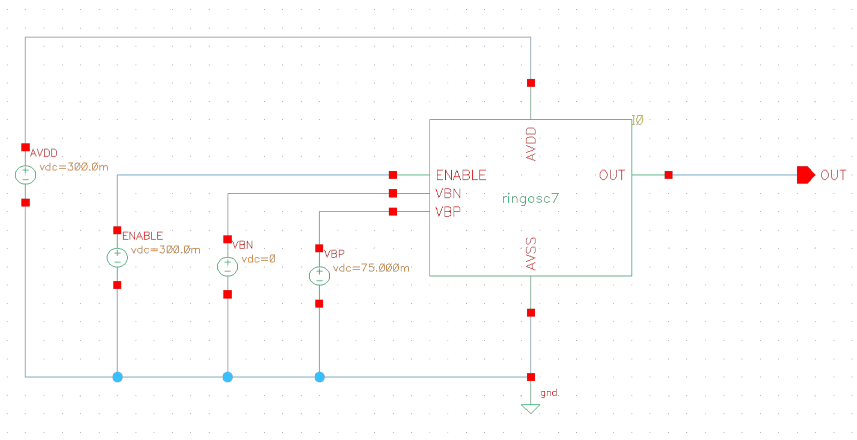

Figure A.5: Testbench of the 7 stages ring oscillator

A.3 NAND layout

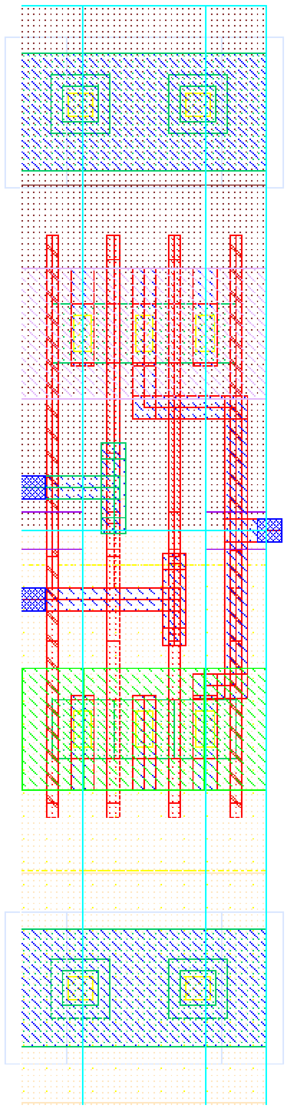

Figure A.6: Layout of NAND gate

A.4 Inverter layout

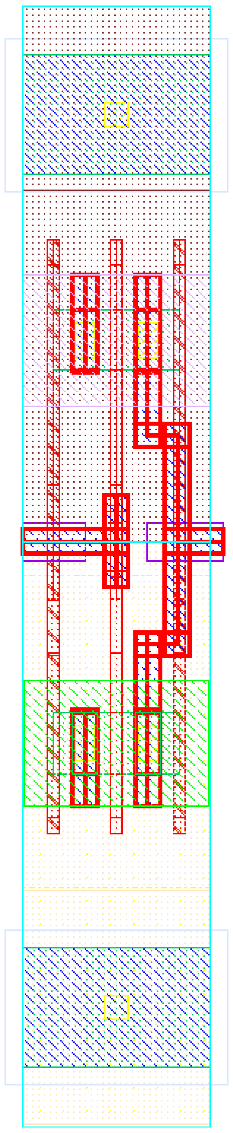

Figure A.7: Layout of inverter gate

B | Compiled cells

B.1 Common source amplifier

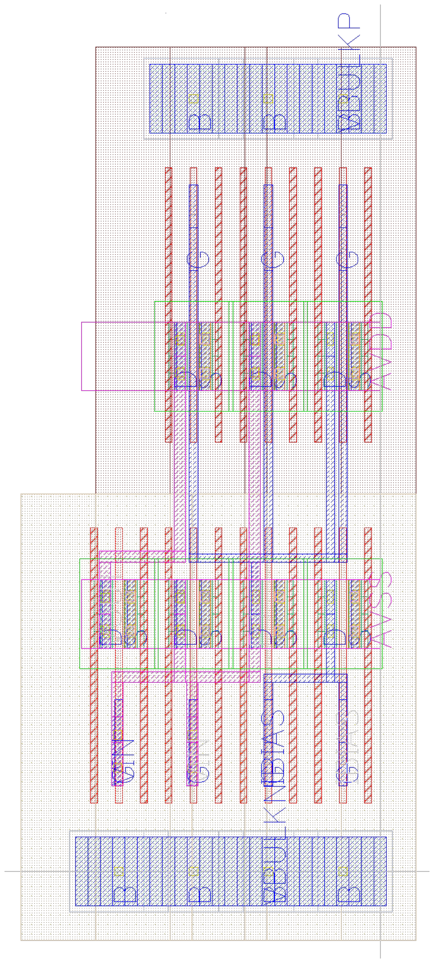

Figure B.1: Layout of the compiled common source amplifier

B.2 Common source amplifier

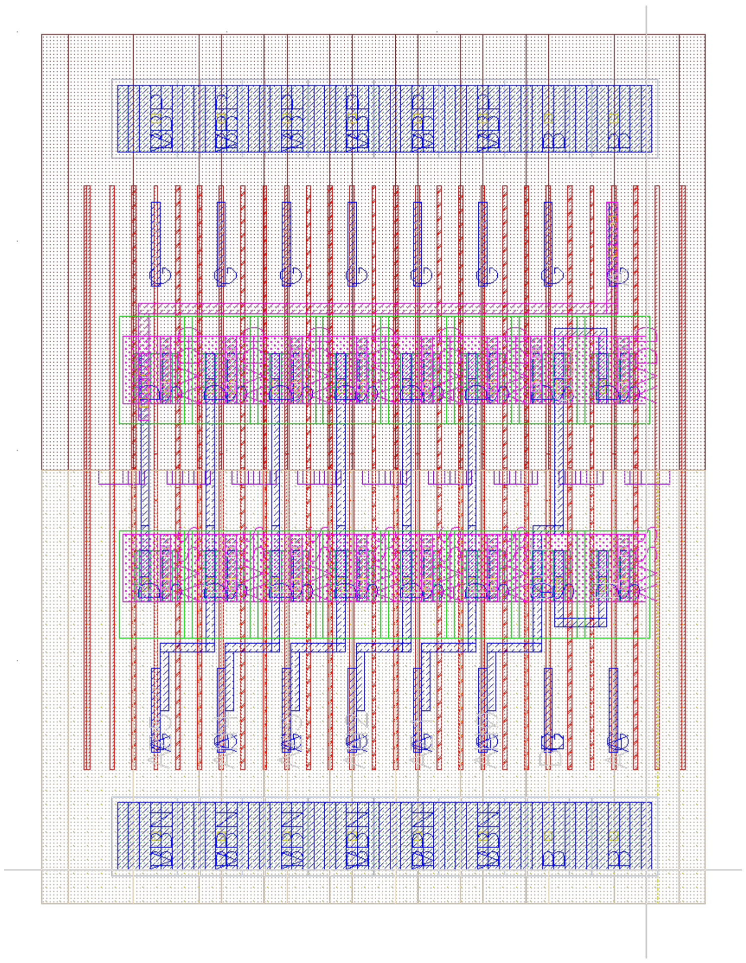

Figure B.2: Compiled layout of the 7 stage ring oscillator

C | Compiler input code

C.1 Core transistors definition file

```
//==================================================================
//         Copyright (c) 2018 Carsten Wulff Software, Norway
//
    ↪ ==================================================================
// Created        : wulff at 2018-2-17
//
    ↪ ==================================================================
// The MIT License (MIT)
//
// Permission is hereby granted, free of charge, to any person
    ↪ obtaining a copy
// of this software and associated documentation files (the
    ↪ "Software"), to deal
// in the Software without restriction, including without
    ↪ limitation the rights
// to use, copy, modify, merge, publish, distribute, sublicense,
    ↪ and/or sell
// copies of the Software, and to permit persons to whom the
    ↪ Software is
// furnished to do so, subject to the following conditions:
//
// The above copyright notice and this permission notice shall be
    ↪ included in all
// copies or substantial portions of the Software.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY
    ↪ KIND, EXPRESS OR
// IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
    ↪ MERCHANTABILITY,
//   FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO
↪  EVENT SHALL THE
//   AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES
↪  OR OTHER
//   LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR
↪  OTHERWISE, ARISING FROM,
//   OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
↪  DEALINGS IN THE
//   SOFTWARE.
//
//====================================================================

{
  "patterns" : {
    "Z" : [ "-",
            "x"],
    "Y" : [ "x",
            "-"],
    "z" : [ "-----",
            "-xxxx"],
    "y" : [ "-xxxx",
            "-----"],
    "u" : [ "-xxxx"],
    "p" : [ "-----",
            "xxxx-"],
    "o" : [ "xxxx-",
            "-----"],
    "v" : [ "xxxx-"]
  },
  "cells":
    [
      {
        "name": "DMOS",
        "class": "Gds::GdsPatternTransistor",
        "yoffset": -0.5,
        "type": "pch",
        "widthoffset": 0,
        "fillCoordinatesFromStrings": [
          [
            "OD",
            "------------------------",
            "-----YYYY---------------",
            "-----xCxC---------------",
            "-----xxxx---------------",
            "-----xCxC---------------",
            "-----ZZZZ---------------",
            "------------------------"
          ],
          [
            "ODB",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxCx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx"
          ],
          [
            "PO",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "--mmmmmmGmmmmcmcmm------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------"
          ],
          [
            "POD",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "------------------------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------"
          ],
          [
            "M1",
            "--------------------xxxx",
            "--------------------xxxx",
            "-----wDww-----------xxxx",
            "------------wGwww---xBxx",
            "-----wSww-----------xxxx",
            "--------------------xxxx",
            "--------------------xxxx"
          ]

```

```
          ]
        },

         {
          "name": "DMOS_PO",
          "class": "Gds::GdsPatternTransistor",
          "yoffset": -0.5,
          "type": "pch_lvt",
          "widthoffset": -2,
          "fillCoordinatesFromStrings": [
            [
              "OD",
              "------------------------",
              "-----YYY----------------",
              "-----xCx----------------",
              "-----xxx----------------",
              "-----xCx----------------",
              "-----ZZZ----------------",
              "------------------------"
            ],
            [
              "ODB",
              "--------------------xxxx",
              "--------------------xxxx",
              "--------------------xxxx",
              "--------------------xxCx",
              "--------------------xxxx",
              "--------------------xxxx",
              "--------------------xxxx"
            ],
            [
              "PO",
              "------------------------",
              "mmmmmmmmmmmmmmmmmm------",
              "------------------------",
              "--mmmmmmGmmmmcmcmm------",
              "------------------------",
              "mmmmmmmmmmmmmmmmmm------",
              "------------------------"
            ],
            [
              "POD",
              "------------------------",
              "-mmmmmmmmmmmmmmmmm------",
              "------------------------",
              "------------------------",
              "------------------------",
              "-mmmmmmmmmmmmmmmmm------",
              "------------------------"
            ],
            [
              "M1",
              "--------------------xxxx",
              "--------------------xxxx",
              "-----wDw------------xxxx",
              "------------wGwww---xBxx",
              "-----wSw------------xxxx",
              "--------------------xxxx",
              "--------------------xxxx"
            ],
            [
              "CT",
              "y-----------------------",
              "u-----------------------",
              "z-----------------------",
              "------------------------",
              "y-----------------------",
              "u-----------------------",
              "z-----------------------"
          ]
          ]
        },


        },
         {
          "name": "DMOS_PO_mirror",
          "class": "Gds::GdsPatternTransistor",
          "yoffset": -0.5,
          "type": "pch_lvt",
          "widthoffset": -2.2,
          "fillCoordinatesFromStrings": [
            [
              "OD",
              "------------------------",
              "----------------YYY-----",
              "----------------xCx-----",
              "----------------xxx-----",
              "----------------xCx-----",
              "----------------ZZZ-----",
              "------------------------"
            ],
            [
              "ODB",
              "xxxx--------------------",
              "xxxx--------------------",
              "xxxx--------------------",
              "xxCx--------------------",
              "xxxx--------------------",
              "xxxx--------------------",
              "xxxx--------------------"
            ],
            [
              "PO",
              "------------------------",
              "------mmmmmmmmmmmmmmmmmm",
              "------------------------",
              "------mmcmcmmmmGmmmmmm--",
              "------------------------",
              "------mmmmmmmmmmmmmmmmmm",
              "------------------------"
            ],
            [
              "POD",
              "------------------------",
              "------mmmmmmmmmmmmmmmmm-",
              "------------------------",
              "------------------------",
              "------------------------",
              "------mmmmmmmmmmmmmmmmm-",
              "------------------------"
            ],
            [
              "M1",
              "xxxx--------------------",
              "xxxx--------------------",
              "xxxx------------wDw-----",
              "xBxx---wwwGw------------",
              "xxxx------------wSw-----",
              "xxxx--------------------",
              "xxxx--------------------"
            ],
            [
              "CT",
              "-----------------------o",
            "-----------------------v",
            "-----------------------p",
            "------------------------",
            "-----------------------o",
            "-----------------------v",
            "-----------------------p"
        ]
        ]
      },

      {
        "name": "DMOSDMY",
        "class": "cIcCore::PatternTile",
        "yoffset": -0.5,
        "type": "pch",
        "widthoffset": 0,
        "fillCoordinatesFromStrings": [
          [
            "OD",
            "------------------------",
            "------------------------",
            "------------------------",
            "-----YYYY---------------",
            "-----xxxx---------------",
            "-----xxxx---------------",
            "-----xxxx---------------",
            "-----ZZZZ---------------",
            "------------------------",
            "------------------------",
            "------------------------"
          ],
          [
            "ODB",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx",
            "--------------------xxxx"
          ],
          [
            "PO",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------"
          ],
          [
            "POD",
            "------------------------",
            "------------------------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "------------------------",
            "------------------------",
            "--mmmmmmmmmmmmmmmm------",
            "------------------------",
            "------------------------",
            "------------------------"
          ]
        ]
      },


        {
        "name": "PCHDL",
        "inherit": "DMOS",
        "type": "pch_lvt",
        "widthoffset": -1,
        "beforePlace": {
          "addEnclosures": [
            [
              "ODB",
              0,
              [ "HYBRID" ]
            ]
          ],
          "addEnclosuresByRectangle": [
            [
              "ODB",
              [ 0, -1, 24, 11 ],
              [ "PP" ]
            ],
            [
              "OD",
              [ 5, 2, 4, 3 ],
              [ "LVTP" ]
            ]

          ]
        }
      },
      {
        "name": "NCHDL",
        "inherit": "DMOS",
        "xoffset": -2,
        "afterNew": {
          "mirrorPatternString": 1
        },
        "type": "nch_lvt",
        "beforePlace": {
          "addEnclosures": [
          ],
          "addEnclosuresByRectangle": [
            [
              "ODB",
              [ -1, -1, 24, 11 ],
              [ "NW", "NP" ]
            ],
            [
              "OD",
              [ 15, 2, 4, 3 ],
              [ "LVTN" ]
            ],
            [
              "ODB",
              [ 0, 0, 4, 7 ],
              [ "HYBRID" ]
            ]
          ]
        }
      },

        {
        "name": "PCHDL_PO",
        "inherit": "DMOS_PO",
        "type": "pch_lvt",
        "widthoffset": -1,
        "beforePlace": {
          "addEnclosures": [
            [
              "ODB",
              0,
              [ "HYBRID" ]
            ]
          ],
          "addEnclosuresByRectangle": [
            [
              "ODB",
              [2, -1, 24, 11 ],
              [ "PP" ]
            ],
            [
              "OD",
              [ 5, 2, 4, 3 ],
              [ "LVTP" ]
            ]

          ]
        }
      },
      {
        "name": "NCHDL_PO",
        "inherit": "DMOS_PO_mirror",
        "xoffset": 0,
        "afterNew": {
          "mirrorPatternString": 0
        },
        "type": "nch_lvt",
        "beforePlace": {
          "addEnclosures": [
          ],
          "addEnclosuresByRectangle": [
            [
              "ODB",
              [ -1, -1, 23.8, 11 ],
              [ "NW", "NP" ]
            ],
            [
              "OD",
              [ 15, 2, 4, 3 ],
              [ "LVTN" ]
            ],
            [
              "ODB",
              [ 0, 0, 4, 7 ],
              [ "HYBRID" ]
            ]
          ]
        }
      },
      {
        "name": "PCHDLDMY",
        "inherit": "DMOSDMY",
        "widthoffset": -1,
        "beforePlace": {
          "addEnclosures": [
            [
              "ODB",
              0,
              [ "HYBRID" ]
            ]
          ],
          "addEnclosuresByRectangle": [
            [
              "ODB",
              [ 0, -1, 24, 13 ],
              [ "PP" ]
            ],
            [
              "OD",
              [ 5, 2, 4, 7 ],
              [ "LVTP" ]
            ]

          ]
        }
      },
      {
        "name": "NCHDLDMY",
        "inherit": "DMOSDMY",
        "xoffset": -2,
        "afterNew": {
          "mirrorPatternString": 1
        },
        "beforePlace": {
          "addEnclosures": [
          ],
          "addEnclosuresByRectangle": [
            [
              "ODB",
              [ -1, -1, 24, 13 ],
              [ "NW", "NP" ]
            ],
            [
              "OD",
              [ 15, 2, 4, 7 ],
              [ "LVTN" ]
            ],
            [
              "ODB",
              [ 0, 0, 4, 11 ],
              [ "HYBRID" ]
            ]
          ]
        }
      },
      {
        "name": "NCHDLRDMY",
        "type": "nch_lvt",
        "xoffset": -2,
        "widthoffset": 2,
        "inherit": "NCHDLDMY",
        "afterPaint": { "mirrorCenterX": -1 }
      },
      {
        "name": "NCHDLR",
        "type": "nch_lvt",
        "xoffset": -2,
        "widthoffset": 2,
        "inherit": "NCHDL",
        "afterPaint": { "mirrorCenterX": -1 }
      }
    ]
}
```

C.2 Digital cells netlist

```
**********************************************************************
**        Copyright (c) 2016 Carsten Wulff Software, Norway
** *******************************************************************
** Created       : wulff at 2016-11-16
** *******************************************************************


.subckt IVX1 A Y AVDD AVSS VBP VBN
MN0 Y A AVSS VBN  NCHDL_P0
MP0 Y A AVDD VBP PCHDL_P0 xoffset=2
.ends

.subckt IVX2 A Y AVDD AVSS
MN0 Y A AVSS AVSS NCHDL
MN1 AVSS A Y AVSS NCHDL
MP0 Y A AVDD AVSS PCHDL
MP1 AVDD A Y AVSS PCHDL
.ends

.subckt IVX4 A Y AVDD AVSS
MN0 Y A AVSS AVSS NCHDL
MN1 AVSS A Y AVSS NCHDL
MN2 Y A AVSS AVSS NCHDL
MN3 AVSS A Y AVSS NCHDL
MP0 Y A AVDD AVSS PCHDL
MP1 AVDD A Y AVSS PCHDL
MP2 Y A AVDD AVSS PCHDL
MP3 AVDD A Y AVSS PCHDL
.ends

.subckt NRX1 A B Y AVDD AVSS VBP VBN
MN0 Y A AVSS VBN NCHDL
MN1 AVSS B Y VBN NCHDL
MP0 N1 A AVDD VBP PCHDL
MP1 Y B N1 VBP PCHDL
.ends

.subckt NDX1 A B OUT AVDD AVSS VBP VBN
MN0 N1 A AVSS VBN NCHDL_P0
MN1 OUT B N1 VBN  NCHDL_P0
MP0 OUT A AVDD VBP PCHDL_P0 xoffset=2
```

```
MP1 AVDD B OUT VBP PCHDL_PO
.ends

.subckt NDX2 A B Y AVDD AVSS
MN0 N1 A AVSS AVSS NCHDL
MN1 Y B N1 AVSS NCHDL
MN2 N2 A Y AVSS NCHDL
MN3 AVSS B N2 AVSS NCHDL
MP0 Y A AVDD AVSS PCHDL
MP1 AVDD B Y AVSS PCHDL
MP2 Y A AVDD AVSS PCHDL
MP3 AVDD B Y AVSS PCHDL
.ends

.subckt ANX1 A B Y AVDD AVSS VBP VBN
XA1 A B YN AVDD AVSS VBP VBN NDX1
XA2 YN Y AVDD AVSS VBP VBN IVX1
.ends

.subckt EONX1 A B AN BN Y AVDD AVSS
MN1 N1 A AVSS AVSS NCHDL
MN2 Y B N1 AVSS NCHDL
MN3 N3 BN Y AVSS NCHDL
MN4 AVSS AN N3 AVSS NCHDL
MP1 NP1 A Y AVSS PCHDL
MP2 AVDD BN NP1 AVSS PCHDL
MP3 NP2 B AVDD AVSS PCHDL
MP4 Y AN NP2 AVSS PCHDL
.ends

.subckt IVTRIX1 A C CN Y AVDD AVSS
MN0 N1 A AVSS AVSS NCHDL
MN1 Y C N1 AVSS NCHDL
MP0 N2 A AVDD AVSS PCHDL
MP1 Y CN N2 AVSS PCHDL
.ends

.subckt NDTRIX1 A C CN RN Y AVDD AVSS
MN2 N1 RN AVSS AVSS NCHDL
MN0 N2 A N1 AVSS NCHDL
MN1 Y C N2 AVSS NCHDL
MP2 AVDD RN N2 AVSS PCHDL
MP0 N2 A AVDD AVSS PCHDL
MP1 Y CN N2 AVSS PCHDL
.ends
```

```

.subckt CS_AMP VIN IBIAS VOUT VBULKP VBULKN AVDD AVSS
MN3 IBIAS IBIAS AVSS VBULKN NCHDL
MN0 P1 IBIAS AVSS VBULKN NCHDL
MN1 VIN VIN AVSS VBULKN NCHDL
MN2 VOUT VIN AVSS VBULKN  NCHDL
MP0 P1  P1  AVDD VBULKP PCHDL
MP1 VIN P1 AVDD VBULKP  PCHDL
MP2 VOUT P1 AVDD VBULKP  PCHDL
.ends

.subckt RINGOSC7 A E AVDD AVSS VBP VBN
XA0 A E N0 AVDD AVSS VBP VBN NDX1
XA1 N0 Z1 AVDD AVSS VBP VBN IVX1
XA2 Z1 N2 AVDD AVSS VBP VBN IVX1
XA3 N2 Z3 AVDD AVSS VBP VBN IVX1
XA4 Z3 N4 AVDD AVSS VBP VBN IVX1
XA5 N4 Z5 AVDD AVSS VBP VBN IVX1
XA6 Z5 A AVDD AVSS VBP VBN IVX1
.ends
```

C.3 Digital cells object definition file

```
//----------------------------------------------------------------------
//        Copyright (c) 2016 Carsten Wulff Software, Norway
//----------------------------------------------------------------------
// Created        : wulff at 2016-11-16
//----------------------------------------------------------------------

{
  "noPortTranslation" : 1,
  "cells":
  [
    {
      "name": "CS_AMP" ,
      "symbol" : "cs_amp",
      "class" : "Layout::LayoutDigitalCell",
      "beforeRoute" : {
                         "addConnectivityRoutes" : [

                          ["M1","^P","-|--"],
                          ["M2","VOUT","-|--"],
                          ["M2","VIN","-|--"],
                          ["M1","IBIAS","-|--"]
```

```
                        ]
         },
      "afterRoute"  : {
         "addPortOnRects" : [ ["VIN","M1", "MN2:G"] , ["VOUT", "M1",
         ↪  "MN2:D"],
         ↪  ["IBIAS","M1","MN0:G"],["VBULKN","M1","MN0:B"],["VBULKP","M1","MP0:B"
                           ]
      }
    },

    {
      "name": "IVX1" ,
      "symbol" : "inv",
      "class" : "Layout::LayoutDigitalCell",
      "beforeRoute" : {
        "addDirectedRoutes" : [ ["M1","Y","MN0:D-|--MP0:D"],
                                ["P0","A","MN0:G-MP0:G"] ]
      },
      "afterRoute"  : {
        "addPortOnRects" : [ ["Y", "M1", "MN0:D"],
        ↪  ["VBN","M1","MN0:B"],["VBP","M1","MP0:B"]]
      }
    },


    {
      "name": "IVX2" ,
      "class" : "Layout::LayoutDigitalCell",
      "symbol" : "inv",
      "setYoffsetHalf" :  "" ,
      "rows" : 2,
      "beforeRoute" : {
        "addDirectedRoutes" : [ ["M1","Y","MN0:D-|--MP0:D"],
                                ["P0","A","MN:G-MP:G"] ,
                                ["M1","A","MN0:G||MN1:G"] ,
                                ["M1","A","MP0:G||MP1:G"]
                              ]
      },
      "afterRoute"  : {
        "addPortOnRects" : [  ["A","M1", "MN0:G"] , ["Y", "M1",
        ↪  "MN0:D"]]
      }
    },
    {
      "name": "IVX4" ,
      "class" : "Layout::LayoutDigitalCell",
      "symbol" : "inv",
      "setYoffsetHalf" :  "" ,
      "rows" : 4,
      "beforeRoute" : {
        "addDirectedRoutes" : [ ["M1","Y","MN0:D,MN2:D-|--MP0:D,MP2:D"],
                                ["PO","A","MN:G-MP:G"] ,
                                ["M1","A","MN0:G||MN3:G"] ,
                                ["M1","A","MP0:G||MP3:G"]
                              ]
      },
      "afterRoute"  : {
        "addPortOnRects" : [  ["A","M1", "MN0:G"] , ["Y", "M1", "MN0:D"]]
      }
    },
    {
      "name": "NRX1",
      "class" : "Layout::LayoutDigitalCell",
      "rows" : 2,
      "symbol" : "nor",
      "setYoffsetHalf" :  "" ,
      "beforeRoute" : {
        "addDirectedRoutes" : [ ["M1","Y","MN0:D-|--MP1:D"],
                                ["PO","A","MN0:G-MP0:G"],
                                ["PO","B","MN1:G-MP1:G"]
                              ]
      },
      "afterRoute" : {
        "addPortOnRects" : [ ["A", "M1" ,"MN0:G"], ["B", "M1", "MN1:G"], ["Y", "M1", "MN1:S" ]]
      }
    },
    {
       "name": "NDX1" ,
       "class" : "Layout::LayoutDigitalCell",
       "rows" : 2,
       "symbol" : "nand",
       "setYoffsetHalf" :  "" ,
       "beforeRoute" : {
         "addConnectivityRoutes": [
                                  //  ["M1","^N","-|--"]
                                  ],
         "addDirectedRoutes" : [ ["M1","OUT","MN1:D-|--MP1:S"],
                                 ["M1","OUT","MP1:S-|MP0:D"],
                                 ["P0","A","MN0:G-MP0:G"],
                                 ["P0","B","MN1:G-MP1:G"],
                                 ["M1","N1","MN0:D|-MN1:S"]
                               ]
       },
       "afterRoute" : {
         "addPortOnRects" : [ ["A", "M1" ,"MN0:G"], ["B", "M1", "MN1:G"], ["Y", "M1", "MN1:D" ]]
       }
    },
    {
      "name": "NDX2",
      "class": "Layout::LayoutDigitalCell",
      "rows": 4,
      "symbol": "nand",
      "setYoffsetHalf": "",
      "beforeRoute": {
        "addConnectivityRoutes": [
          [ "M1", "Y", "-|--", "onTopL", "", "" ],
          [ "M2", "A$", "-|--", "", "", "NCH" ],
          [ "M1", "A$", "--|-", "", "", "PCH" ],
          [ "M2", "B$", "--|-", "", "", "NCH" ],
          [ "M1", "B$", "-|--", "", "", "PCH" ]

        ],
        "addDirectedRoutes": [
          [ "P0", "A", "MN0:G-MP0:G" ],
          [ "P0", "B", "MN1$:G-MP1$:G" ],
          [ "P0", "A", "MN2:G-MP2:G" ],
          [ "P0", "B", "MN3:G-MP3:G" ]
        ]
      },
      "afterRoute": {
        "addPortOnRects": [
          [ "A", "M1", "MN0:G" ],
          [ "B", "M1", "MN1:G" ],
          [ "Y", "M1", "MN2:S" ]
        ]
      }
    },
    {
      "name": "ANX1",
      "class": "Layout::LayoutDigitalCell",
      "composite": 1,
      "symbol": "and",
      "beforeRoute": {
        "addDirectedRoutes": [
          [ "M1", "YN", "XA2:MN0:G-|--XA1:MN1:D" ]
        ]
      },
      "afterRoute": {
        "addPortOnRects": [
          [ "A", "M1", "XA1:MN0:G" ],
          [ "B", "M1", "XA1:MN1:G" ],
          [ "Y", "M1", "XA2:MN0:D" ]
        ]
      }
    },
    {
      "name": "EONX1",
      "class": "Layout::LayoutDigitalCell",
      "setYoffsetHalf": "",
      "rows": 4,
      "beforeRoute": {
        "addDirectedRoutes": [
          [ "PO", "A", "MN1:G-MP1:G" ],
          [ "PO", "A", "MN4:G-MP4:G" ]
        ],
        "addConnectivityRoutes": [
        ]
      },
      "afterRoute": {
        "addPortOnRects": [
          [ "A", "M1", "MN1:G" ],
          [ "B", "M1", "MN2:G" ],
          [ "AN", "M1", "MN4:G" ],
          [ "BN", "M1", "MP2:G" ]
        ]
      }
    },
    {
      "name": "IVTRIX1",
      "class": "Layout::LayoutDigitalCell",
      "rows": 2,
      "setYoffsetHalf": "",
      "description": "Tristate inverter, Y = A if C, Y =HiZ if CN",
      "beforeRoute": {
        "addDirectedRoutes": [
          [ "M1", "Y", "MN1:D-|--MP1:D" ],
          [ "P0", "A", "MN0:G-MP0:G" ]
        ]
      },
      "afterRoute": {
        "addPortOnRects": [
          [ "A", "M1", "MN0:G" ],
          [ "CN", "M1" ],
          [ "C", "M1" ],
          [ "Y", "M1", "MN1:D" ]
        ]
      }
    },
    {
      "name": "NDTRIX1",
      "class": "Layout::LayoutDigitalCell",
      "rows": 3,
      "setYoffsetHalf": "",
      "description": "Tristate nand, Y = !A if C and !RN, Y =HiZ if
      ↪  CN",
      "beforeRoute": {
        "addDirectedRoutes": [
          [ "M1", "Y", "MN1:D-|--MP1:D" ],
          [ "M1", "N2", "MP2:S|-MP0:D" ],
          [ "M1", "N2", "MN0:D-MP0:D" ],
          [ "P0", "A", "MN0:G-MP0:G" ],
          [ "P0", "RN", "MN2:G-MP2:G" ]
        ]
      },
      "afterRoute": {
        "addPortOnRects": [
          [ "A", "M1", "MN0:G" ],
          [ "CN", "M1" ],
          [ "C", "M1" ],
          [ "RN", "M1", "MN2:G" ],
          [ "Y", "M1", "MN1:D" ]
        ]
      }
    },

   {
      "name": "RINGOSC7" ,
      "symbol" : "ringosc",
      "class" : "Layout::LayoutDigitalCell",
      "beforeRoute" : {
        "addDirectedRoutes" : [ ["M2","A","XA0:MP0:G--|-XA6:MP0:D"]
```

```
                ],
          "addConnectivityRoutes":[
                  ["M1","^N","-|--","offsetlow"],
                  ["M1","^Z","-|--","offsetlow"]
                  ]
        },
        "afterRoute"  : {
          "addPortOnRects" : [  ["E","M1","XA0:MN1:G"]]
        }
      }
    ]
}
```